Statistical Methods for Data Science

Elizabeth Purdom

2023-05-08

Contents

Chapter 1

Introduction

This book consists of materials to accompany the course "Statistical Methods for Data Science" (STAT 131A) taught at UC Berkeley. STAT 131A is an upper-division course that is a follow-up course to an introductory statistics, such as DATA 8 or STAT 20 taught at UC Berkeley.

The textbook will teach a broad range of statistical methods that are used to solve data problems. Topics include group comparisons and ANOVA, standard parametric statistical models, multivariate data visualization, multiple linear regression and logistic regression, classification and regression trees and random forests.

These topics are covered at a very intuitive level, with only a semester of calculus expected to be able to follow the material. The goal of the book is to explain these more advanced topics at a level that is widely accessible.

In addition to an introductory statistics course, students in this course are expected to have had some introduction to programming, and the textbook does not explain programming concepts nor does it generally explain the R Code shown in the book. The focus of the book is understanding the concepts and the output. To have more understanding of the R Code, please see the accompanying `.Rmd` that steps through the code in each chapter (and the accompanying `.html` that gives a compiled version). These can be found at epurdom.github.io/Stat131A/Rsupport/index.html.

The datasets used in this manuscript should be made available to students in the class on bcourses by their instructor.

1.1 Acknowledgements

This manuscript is based on lecture notes originally developed by Aditya Guntuboyina (Chapters 6-8) and Elizabeth Purdom (Chapters 2-5) in the Spring of 2017, the first time the course was taught at UC Berkeley. Shobhana Stoyanov provided materials that aided in the writing of Chapter 2, section 2.2 and useful feedback.

```
## Linking to ImageMagick 6.9.12.3
## Enabled features: cairo, fontconfig, freetype, heic, lcms, pango, raw, rsvg, webp
## Disabled features: fftw, ghostscript, x11
```

Chapter 2

Data Distributions

We're going to review some basic ideas about distributions you should have learned in Data 8 or STAT 20. In addition to review, we introduce some new ideas and emphases to pay attention to:

- Continuous distributions and density curves
- Tools for visualizing and estimating distributions: boxplots and kernel density estimators
- Types of samples and how they effect estimation

2.1 Basic Exporatory analysis

Let's look at a dataset that contains the salaries of San Francisco employees.[1] We've streamlined this to the year 2014 (and removed some strange entries with negative pay). Let's explore this data.

```
dataDir <- "../finalDataSets"
nameOfFile <- file.path(dataDir, "SFSalaries2014.csv")
salaries2014 <- read.csv(nameOfFile, na.strings = "Not Provided")
dim(salaries2014)
```

```
## [1] 38117    10
```

```
names(salaries2014)
```

```
##  [1] "X"                "Id"               "JobTitle"         "BasePay"
##  [5] "OvertimePay"      "OtherPay"         "Benefits"         "TotalPay"
##  [9] "TotalPayBenefits" "Status"
```

[1] https://www.kaggle.com/kaggle/sf-salaries/

```
salaries2014[1:10, c("JobTitle", "Benefits", "TotalPay",
    "Status")]
```

```
##                          JobTitle Benefits TotalPay Status
## 1                 Deputy Chief 3 38780.04 471952.6     PT
## 2               Asst Med Examiner 89540.23 390112.0     FT
## 3        Chief Investment Officer 96570.66 339653.7     PT
## 4                  Chief of Police 91302.46 326716.8     FT
## 5          Chief, Fire Department 91201.66 326233.4     FT
## 6               Asst Med Examiner 71580.48 344187.5     FT
## 7                     Dept Head V 89772.32 311298.5     FT
## 8     Executive Contract Employee 88823.51 310161.0     FT
## 9  Battalion Chief, Fire Suppress 59876.90 335485.0     FT
## 10   Asst Chf of Dept (Fire Dept) 64599.59 329390.5     FT
```

Let's look at the column 'TotalPay' which gives the total pay, not including benefits.

Question: How might we want to explore this data? What single number summaries would make sense? What visualizations could we do?

```
summary(salaries2014$TotalPay)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0   33482   72368   75476  107980  471953
```

Notice we have entries with zero pay! Let's investigate why we have zero pay by subsetting to just those entries.

```
zeroPay <- subset(salaries2014, TotalPay == 0)
nrow(zeroPay)
```

```
## [1] 48
```

```
head(zeroPay)
```

```
##            X     Id                        JobTitle BasePay OvertimePay OtherPay
## 34997 145529 145529           Special Assistant 15       0           0        0
## 35403 145935 145935 Community Police Services Aide       0           0        0
## 35404 145936 145936     BdComm Mbr, Grp3,M=$50/Mtg       0           0        0
## 35405 145937 145937     BdComm Mbr, Grp3,M=$50/Mtg       0           0        0
## 35406 145938 145938                       Gardener       0           0        0
## 35407 145939 145939                       Engineer       0           0        0
##       Benefits TotalPay TotalPayBenefits Status
## 34997  5650.86        0          5650.86     PT
## 35403  4659.36        0          4659.36     PT
## 35404  4659.36        0          4659.36     PT
```

```
## 35405  4659.36        0          4659.36     PT
## 35406  4659.36        0          4659.36     PT
## 35407  4659.36        0          4659.36     PT
```

```
summary(zeroPay)
```

```
##        X                Id           JobTitle            BasePay   OvertimePay
##  Min.   :145529   Min.   :145529   Length:48          Min.   :0   Min.   :0
##  1st Qu.:145948   1st Qu.:145948   Class :character   1st Qu.:0   1st Qu.:0
##  Median :145960   Median :145960   Mode  :character   Median :0   Median :0
##  Mean   :147228   Mean   :147228                      Mean   :0   Mean   :0
##  3rd Qu.:148637   3rd Qu.:148637                      3rd Qu.:0   3rd Qu.:0
##  Max.   :148650   Max.   :148650                      Max.   :0   Max.   :0
##     OtherPay    Benefits       TotalPay TotalPayBenefits    Status
##  Min.   :0   Min.   :   0   Min.   :0   Min.   :   0     Length:48
##  1st Qu.:0   1st Qu.:   0   1st Qu.:0   1st Qu.:   0     Class :character
##  Median :0   Median :4646   Median :0   Median :4646     Mode  :character
##  Mean   :0   Mean   :2444   Mean   :0   Mean   :2444
##  3rd Qu.:0   3rd Qu.:4649   3rd Qu.:0   3rd Qu.:4649
##  Max.   :0   Max.   :5651   Max.   :0   Max.   :5651
```

It's not clear why these people received zero pay. We might want to remove them, thinking that zero pay are some kind of weird problem with the data we aren't interested in. But let's do a quick summary of what the data would look like if we did remove them:

```
summary(subset(salaries2014, TotalPay > 0))
```

```
##        X                Id           JobTitle            BasePay
##  Min.   :110532   Min.   :110532   Length:38069       Min.   :     0
##  1st Qu.:120049   1st Qu.:120049   Class :character   1st Qu.: 30439
##  Median :129566   Median :129566   Mode  :character   Median : 65055
##  Mean   :129568   Mean   :129568                      Mean   : 66652
##  3rd Qu.:139083   3rd Qu.:139083                      3rd Qu.: 94865
##  Max.   :148626   Max.   :148626                      Max.   :318836
##   OvertimePay        OtherPay         Benefits        TotalPay
##  Min.   :     0   Min.   :     0   Min.   :    0   Min.   :     1.8
##  1st Qu.:     0   1st Qu.:     0   1st Qu.:10417   1st Qu.: 33688.3
##  Median :     0   Median :   700   Median :28443   Median : 72414.3
##  Mean   :  5409   Mean   :  3510   Mean   :24819   Mean   : 75570.7
##  3rd Qu.:  5132   3rd Qu.:  4105   3rd Qu.:35445   3rd Qu.:108066.1
##  Max.   :173548   Max.   :342803   Max.   :96571   Max.   :471952.6
##  TotalPayBenefits      Status
##  Min.   :     7.2   Length:38069
##  1st Qu.: 44561.8   Class :character
##  Median :101234.9   Mode  :character
##  Mean   :100389.8
```

```
##  3rd Qu.:142814.2
##  Max.   :510732.7
```

We can see that in fact we still have some weird pay entires (e.g. total payment of $1.8). This points to the slippery slope you can get into in "cleaning" your data – where do you stop?

A better observation is to notice that all the zero-entries have "Status" value of PT, meaning they are part-time workers.

```
summary(subset(salaries2014, Status == "FT"))
```

```
##        X                Id           JobTitle            BasePay
##  Min.   :110533   Min.   :110533   Length:22334       Min.   : 26364
##  1st Qu.:116598   1st Qu.:116598   Class :character   1st Qu.: 65055
##  Median :122928   Median :122928   Mode  :character   Median : 84084
##  Mean   :123068   Mean   :123068                      Mean   : 91174
##  3rd Qu.:129309   3rd Qu.:129309                      3rd Qu.:112171
##  Max.   :140326   Max.   :140326                      Max.   :318836
##   OvertimePay        OtherPay         Benefits        TotalPay
##  Min.   :     0   Min.   :     0   Min.   :    0   Min.   : 26364
##  1st Qu.:     0   1st Qu.:     0   1st Qu.:29122   1st Qu.: 72356
##  Median :  1621   Median :  1398   Median :33862   Median : 94272
##  Mean   :  8241   Mean   :  4091   Mean   :35023   Mean   :103506
##  3rd Qu.: 10459   3rd Qu.:  5506   3rd Qu.:38639   3rd Qu.:127856
##  Max.   :173548   Max.   :112776   Max.   :91302   Max.   :390112
##  TotalPayBenefits    Status
##  Min.   : 31973   Length:22334
##  1st Qu.:102031   Class :character
##  Median :127850   Mode  :character
##  Mean   :138528
##  3rd Qu.:167464
##  Max.   :479652
```

```
summary(subset(salaries2014, Status == "PT"))
```

```
##        X                Id           JobTitle            BasePay
##  Min.   :110532   Min.   :110532   Length:15783       Min.   :     0
##  1st Qu.:136520   1st Qu.:136520   Class :character   1st Qu.:  6600
##  Median :140757   Median :140757   Mode  :character   Median : 20557
##  Mean   :138820   Mean   :138820                      Mean   : 31749
##  3rd Qu.:144704   3rd Qu.:144704                      3rd Qu.: 47896
##  Max.   :148650   Max.   :148650                      Max.   :257340
##   OvertimePay         OtherPay           Benefits          TotalPay
##  Min.   :    0.0   Min.   :     0.0   Min.   :    0.0   Min.   :     0
##  1st Qu.:    0.0   1st Qu.:     0.0   1st Qu.:  115.7   1st Qu.:  7359
##  Median :    0.0   Median :   191.7   Median : 4659.4   Median : 22410
##  Mean   : 1385.6   Mean   :  2676.7   Mean   :10312.3   Mean   : 35811
```

```
##  3rd Qu.:  681.2   3rd Qu.:  1624.7   3rd Qu.:19246.2   3rd Qu.: 52998
##  Max.   :74936.0   Max.   :342802.6   Max.   :96570.7   Max.   :471953
##  TotalPayBenefits    Status
##  Min.   :     0   Length:15783
##  1st Qu.:  8256   Class :character
##  Median : 27834   Mode  :character
##  Mean   : 46123
##  3rd Qu.: 72569
##  Max.   :510733
```

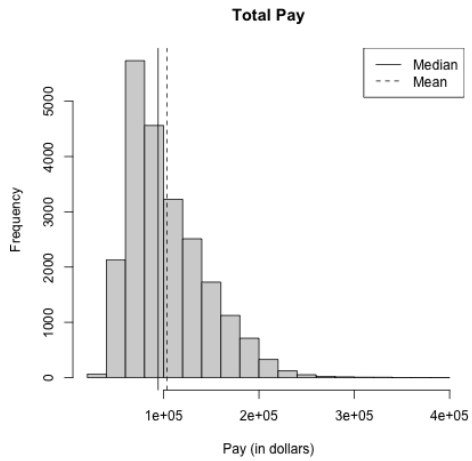
So it is clear that analyzing data from part-time workers will be tricky (and we have no information here as to whether they worked a week or eleven months). To simplify things, we will make a new data set with only full-time workers:

```
salaries2014_FT <- subset(salaries2014, Status == "FT")
```

2.1.1 Histograms

Let's draw a histogram of the total salary for full-time workers only.

```
hist(salaries2014_FT$TotalPay, main = "Total Pay",
    xlab = "Pay (in dollars)")
abline(v = mean(salaries2014_FT$TotalPay), lty = "dashed")
abline(v = median(salaries2014_FT$TotalPay))
legend("topright", legend = c("Median", "Mean"), lty = c("solid",
    "dashed"))
```



Question: What do you notice about the histogram? What does it tell you about the data?

Question: How good of a summary is the mean or median here?

2.1.1.1 Constructing Frequency Histograms

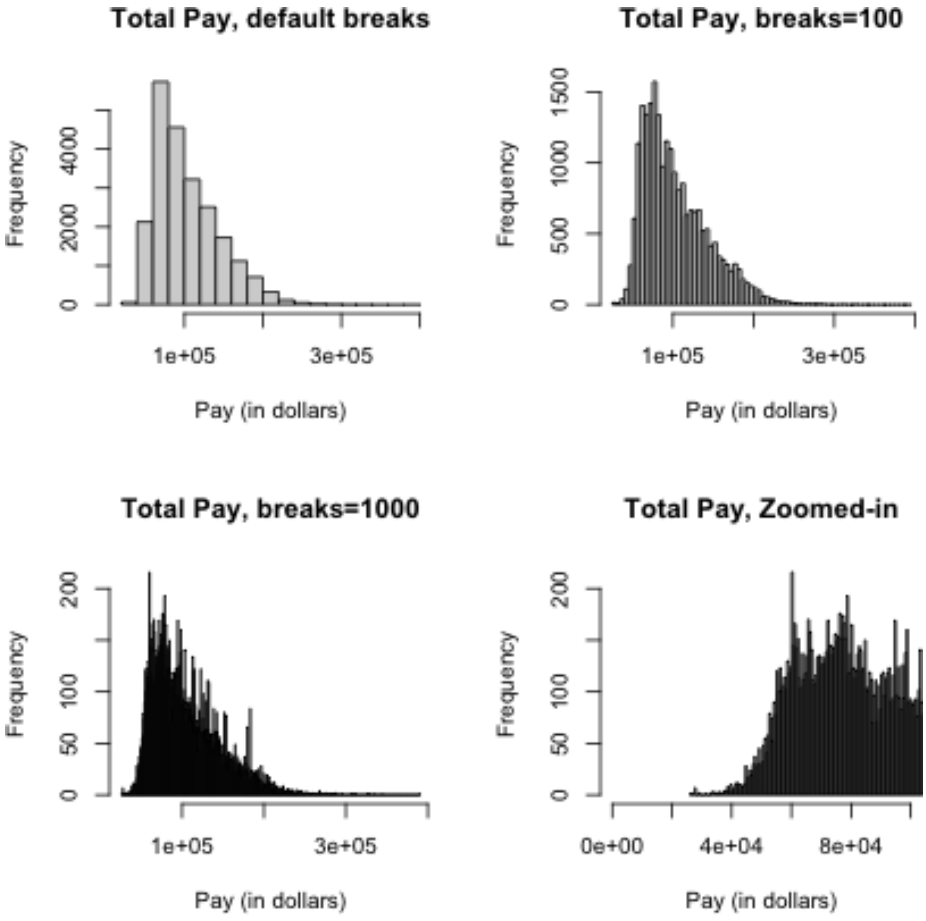
How do you construct a histogram? Practically, most histograms are created by taking an evenly spaced set of K breaks that span the range of the data, call them $b_1 \leq b_2 \leq ... \leq b_K$, and counting the number of observations in each bin.[2] Then the histogram consists of a series of bars, where the x-coordinates of the rectangles correspond to the range of the bin, and the height corresponds to the number of observations in that bin.

2.1.1.1.1 Breaks of Histograms

Here's two more histogram of the same data that differ only by the number of breakpoints in making the histograms.

```
par(mfrow = c(2, 2))
hist(salaries2014_FT$TotalPay, main = "Total Pay, default breaks",
    xlab = "Pay (in dollars)")
hist(salaries2014_FT$TotalPay, main = "Total Pay, breaks=100",
    xlab = "Pay (in dollars)", breaks = 100)
hist(salaries2014_FT$TotalPay, main = "Total Pay, breaks=1000",
    xlab = "Pay (in dollars)", breaks = 1000)
hist(salaries2014_FT$TotalPay, main = "Total Pay, Zoomed-in",
    xlab = "Pay (in dollars)", xlim = c(0, 1e+05),
    breaks = 1000)
```

[2] You might have been taught that you *can* make a histogram with uneven break points, which is true, but in practice is rather exotic thing to do. If you do, then you have to calculate the height of the bar differently based on the width of the bin because it is the *area* of the bin that should be proportional to the number of entries in a bin, not the height of the bin.



Question: What seems better here? Is there a right number of breaks?

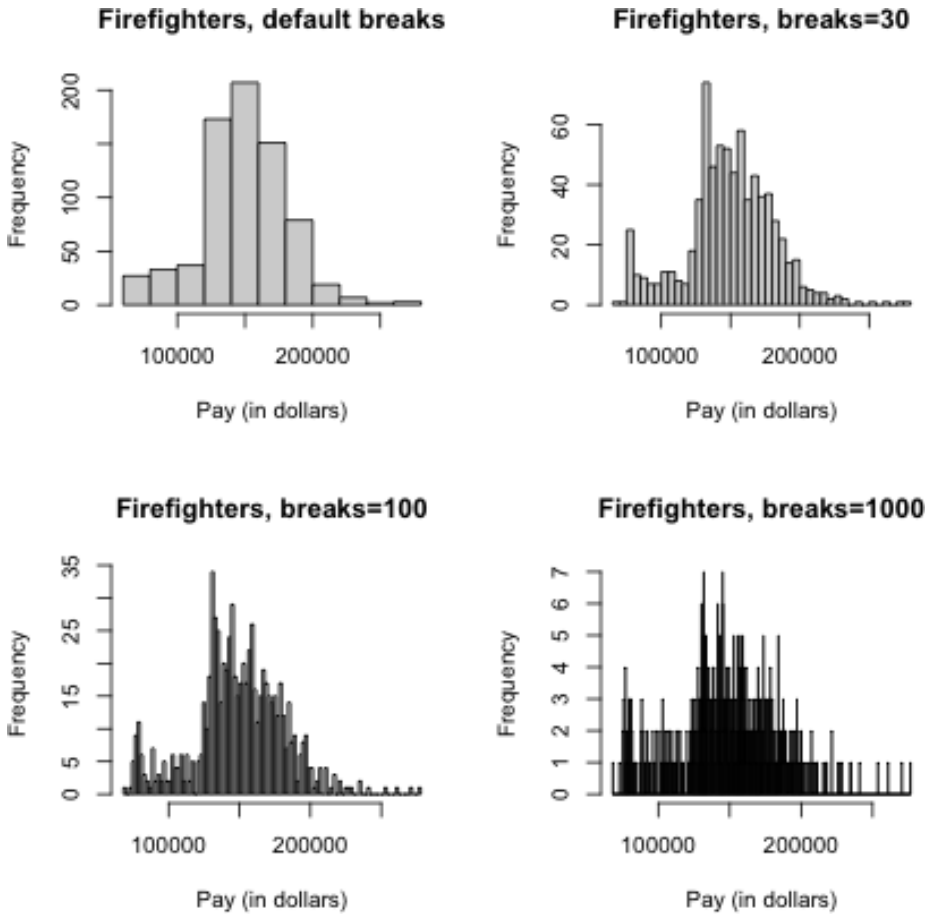
What if we used a subset, say only full-time firefighters? Now there are only 738 data points.

```
salaries2014_FT_FF <- subset(salaries2014_FT, JobTitle ==
    "Firefighter" & Status == "FT")
dim(salaries2014_FT_FF)
```

```
## [1] 738  10
```

```
par(mfrow = c(2, 2))
hist(salaries2014_FT_FF$TotalPay, main = "Firefighters, default breaks",
    xlab = "Pay (in dollars)")
hist(salaries2014_FT_FF$TotalPay, main = "Firefighters, breaks=30",
    xlab = "Pay (in dollars)", breaks = 30)
```

```
hist(salaries2014_FT_FF$TotalPay, main = "Firefighters, breaks=100",
    xlab = "Pay (in dollars)", breaks = 100)
hist(salaries2014_FT_FF$TotalPay, main = "Firefighters, breaks=1000",
    xlab = "Pay (in dollars)", breaks = 1000)
```



2.1.1.2 Density Histograms

The above are called **frequency histograms**, because we plot on the y-axis (the height of the rectangles) the count of the number of observations in each bin. **Density histograms** plot the height of rectangles so that the *area* of each rectangle is equal to the proportion of observations in the bin. If each rectangle has equal width, say w, and there are n total observations, this means for a bin

k, it's height is given by

$$w * h_k = \frac{\#\text{observations in bin } k}{n}$$

So that the height of a rectangle for bin k is given by

$$h_k = \frac{\#\text{observations in bin } k}{w \times n}$$

In other words, the *density* histogram with equal-width bins will look like the frequency histogram, only the heights of all the rectangles will be divided by wn.

We will return to the importance of density histograms more when we discuss continuous distributions.

2.1.2 Boxplots

Another very useful visualization can be a boxplot. A boxplot is like a histogram, in that it gives you a visualization of how the data are distributed. However, it is a much greater simplification of the distribution.

Box: It plots only a box for the bulk of the data, where the limits of the box are the 0.25 and 0.75 quantiles of the data (or 25th and 75th percentiles). A dark line across the middle is the median of the data.

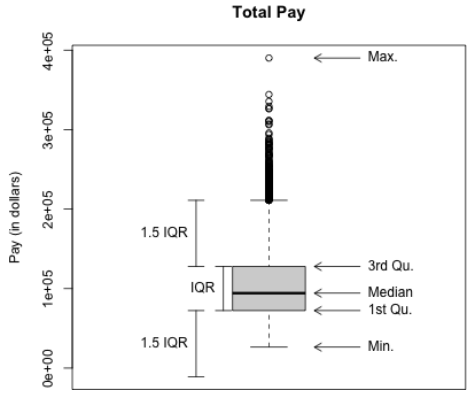
Whiskers: In addition, a boxplot gives additional information to evaluate the extremities of the distribution. It draws "whiskers" out from the box to indicate how far out is the data beyond the 25th and 75th percentiles. Specifically it calculates the interquartitle range (IQR), which is just the difference between the 25th and 75th percentiles:

$$IQR = \text{3rd Qu.} - \text{1st Qu.}$$

It then draws the whiskers out an additional 1.5 IQR distance from the boxes OR to the smallest/largest data point (whichever is closest to the box).

$$\text{lower whisker} : max(\text{1st Qu.} - 1.5IQR, \text{Min})$$

$$\text{upper whisker} : min(\text{3rd Qu.} + 1.5IQR, \text{Max})$$



Any data points outside of this range of the whiskers are ploted individually. These points are often called "outliers" based the 1.5 IQR rule of thumb. The term **outlier** is usually used for unusual or extreme points. However, we can see a lot of data points fall outside this definition of "outlier" for our data; this is common for data that is skewed, and doesn't really mean that these points are "wrong", or "unusual" or anything else that we might think about for an outlier.[3]

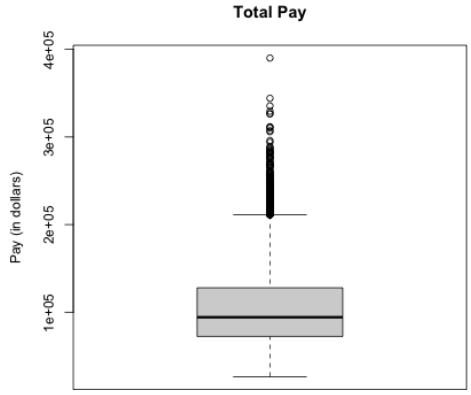
Whiskers Why are the whiskers set like they are? Why not draw them out to the min and max?[4] The motivation is that the whiskers give you the range of "ordinary" data, while the points outside the whiskers are "outliers" that might be wrong or unrepresentative of the data. As mentioned above, this is often not the case in practice. But that motivation is still reasonable. We don't want our notion of the general range of the data to be manipulated by a few extreme points; 1.5 IQR is a more stable, reliable (often called "robust") description of the data.

Taking off the explanations from the plot and going back to our data, our boxplot is given by:

```
par(mfrow = c(1, 1))
boxplot(salaries2014_FT$TotalPay, main = "Total Pay",
    ylab = "Pay (in dollars)")
```

[3] If our data had a nice symmetric distribution around the median, like the normal distribution, the rule of thumb would be more appropriate, and this wouldn't happen to the same degree. Specifically, for a normal distribution with standard deviation, $IQR = 1.35\sigma$, so the whiskers would be a distance of 2.17σ from the mean/median, so the chance of a single observation from a normal being outside of the range of the whiskers would be 0.03.

[4] Some boxplots do define the whiskers to be the min and max, showing the range of the data. This isn't the accepted definition anymore in most areas, but it is always good to check.



You might think, why would I want such a limited display of the distribution, compared to the wealth of information in the histogram? I can't tell at all that the data is bimodal from a boxplot, for example.

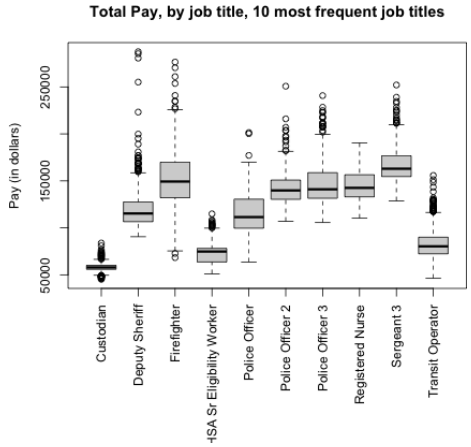
First of all, the boxplot emphasizes different things about the distribution. It shows the main parts of the bulk of the data very quickly and simply, and emphasizes more fine grained information about the extremes ("tails") of the distribution.

Furthermore, because of their simplicity, it is far easier to plot many boxplots and compare them than histograms. For example, I have information of the job title of the employees, and I might be interested in comparing the distribution of salaries with different job titles (firefighters, teachers, nurses, etc). Here I will isolate only those samples that correspond to the top 10 most numerous full-time job titles and do side-by-side boxplots of the distribution within each job title for all 10 jobs.

```
tabJobType <- table(subset(salaries2014_FT, Status ==
    "FT")$JobTitle)
tabJobType <- sort(tabJobType, decreasing = TRUE)
topJobs <- head(names(tabJobType), 10)
salaries2014_top <- subset(salaries2014_FT, JobTitle %in%
    topJobs & Status == "FT")
salaries2014_top <- droplevels(salaries2014_top)
dim(salaries2014_top)
```

```
## [1] 5816   10
```

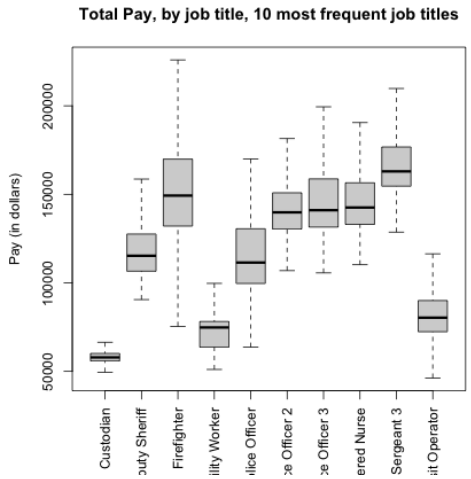
```
par(mar = c(10, 4.1, 4.1, 0.1))
boxplot(salaries2014_top$TotalPay ~ salaries2014_top$JobTitle,
    main = "Total Pay, by job title, 10 most frequent job titles",
    xlab = "", ylab = "Pay (in dollars)", las = 3)
```



This would be hard to do with histograms – we'd either have 10 separate plots, or the histograms would all lie on top of each other. Later on, we will discuss "violin plots" which combine some of the strengths of both boxplots and histograms.

Notice that the outliers draw a lot of attention, since there are so many of them; this is common in large data sets especially when the data are skewed. I might want to mask all of the "outlier" points as distracting for this comparison,

```
boxplot(TotalPay ~ JobTitle, data = salaries2014_top,
    main = "Total Pay, by job title, 10 most frequent job titles",
    xlab = "", ylab = "Pay (in dollars)", las = 3,
    outline = FALSE)
```



2.1.3 Descriptive Vocabulary

Here are some useful terms to consider in describing distributions of data or comparing two different distributions.

Symmetric refers to equal amounts of data on either side of the 'middle' of the data, i.e. the distribution of the data on one side is the mirror image of the distribution on the other side. This means that the median of the data is roughly equal to the mean.

Skewed refers to when one 'side' of the data spreads out to take on larger values than the other side. More precisely, it refers to where the mean is relative to the median. If the mean is much bigger than the median, then there must be large values on the right-hand side of the distribution, compared to the left hand side (**right skewed**), and if the mean is much smaller than the median then it is the reverse.

Spread refers to how spread out the data is from the middle (e.g. mean or median).

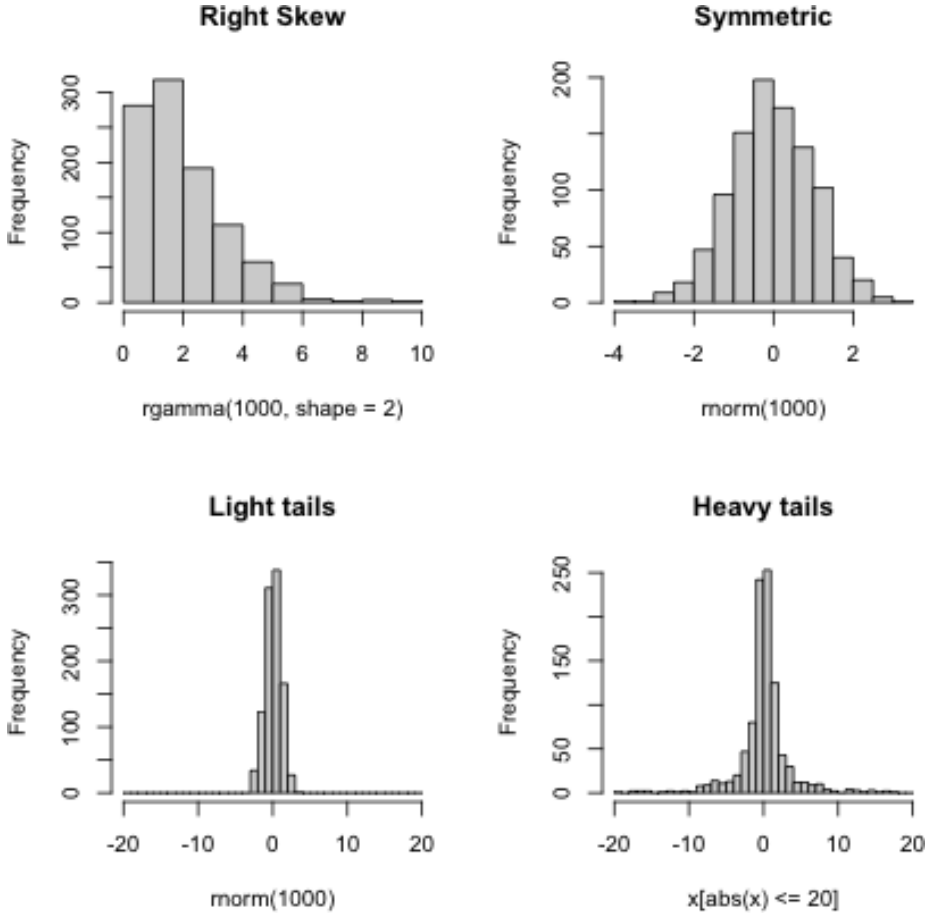
Heavy/light tails refers to how much of the data is concentrated in values far away from the middle, versus close to the middle.

As you can see, several of these terms are mainly relevant for comparing two distributions.[5]

Here are the histograms of some simulated data that demonstrate these features

```
set.seed(1)
par(mfrow = c(2, 2))
hist(rgamma(1000, shape = 2), main = "Right Skew")
hist(rnorm(1000), main = "Symmetric")
breaks = seq(-20, 20, 1)
hist(rnorm(1000), main = "Light tails", xlim = c(-20,
    20), breaks = breaks, freq = TRUE)
x <- rcauchy(1000)
hist(x[abs(x) <= 20], main = "Heavy tails", xlim = c(-20,
    20), breaks = breaks, freq = TRUE)
```

[5] But they are often used without providing an explicit comparison distribution; in this case, the comparison distribution is always the normal distribution, which is a standard benchmark in statistics



2.1.4 Transformations

When we have skewed data, it can be difficult to compare the distributions because so much of the data is bunched up on one end, but our axes stretch to cover the large values that make up a relatively small proportion of the data. This is also means that our eye focuses on those values too.

This is a mild problem with this data, particularly if we focus on the full-time workers, but let's look quickly at another dataset that really shows this problem.

2.1.4.1 Flight Data from SFO

This data consists of all flights out of San Francisco Airport in 2016 in January (we will look at this data more in the next module).

```
flightSF <- read.table(file.path(dataDir, "SFO.txt"),
    sep = "\t", header = TRUE)
dim(flightSF)
```

```
## [1] 13207    64
```

```
names(flightSF)
```

```
##  [1] "Year"                "Quarter"             "Month"
##  [4] "DayofMonth"          "DayOfWeek"           "FlightDate"
##  [7] "UniqueCarrier"       "AirlineID"           "Carrier"
## [10] "TailNum"             "FlightNum"           "OriginAirportID"
## [13] "OriginAirportSeqID"  "OriginCityMarketID"  "Origin"
## [16] "OriginCityName"      "OriginState"         "OriginStateFips"
## [19] "OriginStateName"     "OriginWac"           "DestAirportID"
## [22] "DestAirportSeqID"    "DestCityMarketID"    "Dest"
## [25] "DestCityName"        "DestState"           "DestStateFips"
## [28] "DestStateName"       "DestWac"             "CRSDepTime"
## [31] "DepTime"             "DepDelay"            "DepDelayMinutes"
## [34] "DepDel15"            "DepartureDelayGroups" "DepTimeBlk"
## [37] "TaxiOut"             "WheelsOff"           "WheelsOn"
## [40] "TaxiIn"              "CRSArrTime"          "ArrTime"
## [43] "ArrDelay"            "ArrDelayMinutes"     "ArrDel15"
## [46] "ArrivalDelayGroups"  "ArrTimeBlk"          "Cancelled"
## [49] "CancellationCode"    "Diverted"            "CRSElapsedTime"
## [52] "ActualElapsedTime"   "AirTime"             "Flights"
## [55] "Distance"            "DistanceGroup"       "CarrierDelay"
## [58] "WeatherDelay"        "NASDelay"            "SecurityDelay"
## [61] "LateAircraftDelay"   "FirstDepTime"        "TotalAddGTime"
## [64] "LongestAddGTime"
```

This dataset contains a lot of information about the flights departing from SFO. For starters, let's just try to understand how often flights are delayed (or canceled), and by how long. Let's look at the column 'DepDelay' which represents departure delays.

```
summary(flightSF$DepDelay)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   -25.0    -5.0    -1.0    13.8    12.0   861.0     413
```

Notice the NA's. Let's look at just the subset of some variables for those observations with NA values for departure time (I chose a few variables so it's easier to look at)

```
naDepDf <- subset(flightSF, is.na(DepDelay))
head(naDepDf[, c("FlightDate", "Carrier", "FlightNum",
    "DepDelay", "Cancelled")])
```

```
##     FlightDate Carrier FlightNum DepDelay Cancelled
## 44  2016-01-14      AA       209       NA         1
## 75  2016-01-14      AA       218       NA         1
## 112 2016-01-24      AA        12       NA         1
## 138 2016-01-22      AA        16       NA         1
## 139 2016-01-23      AA        16       NA         1
## 140 2016-01-24      AA        16       NA         1
```

```
summary(naDepDf[, c("FlightDate", "Carrier", "FlightNum",
    "DepDelay", "Cancelled")])
```

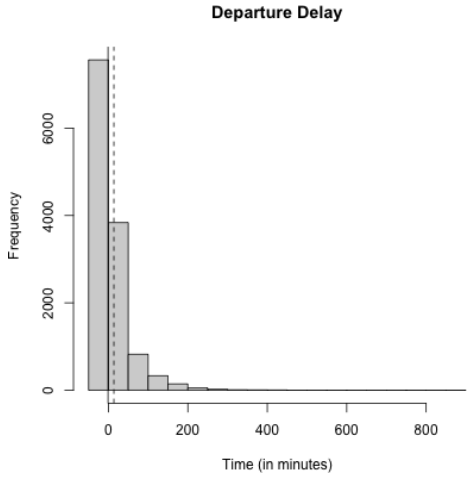
```
##   FlightDate          Carrier            FlightNum       DepDelay     Cancelled
##  Length:413         Length:413         Min.   :   1   Min.   : NA   Min.   :1
##  Class :character   Class :character   1st Qu.: 616   1st Qu.: NA   1st Qu.:1
##  Mode  :character   Mode  :character   Median :2080   Median : NA   Median :1
##                                        Mean   :3059   Mean   :NaN   Mean   :1
##                                        3rd Qu.:5555   3rd Qu.: NA   3rd Qu.:1
##                                        Max.   :6503   Max.   : NA   Max.   :1
##                                                       NA's   :413
```

So, the NAs correspond to flights that were cancelled (Cancelled=1).

2.1.4.1.1 Histogram of flight delays

Let's draw a histogram of the departure delay.

```
par(mfrow = c(1, 1))
hist(flightSF$DepDelay, main = "Departure Delay", xlab = "Time (in minutes)")
abline(v = c(mean(flightSF$DepDelay, na.rm = TRUE),
    median(flightSF$DepDelay, na.rm = TRUE)), lty = c("dashed",
    "solid"))
```



Question: What do you notice about the histogram? What does it tell you about the data?

Question: How good of a summary is the mean or median here? Why are they so different?

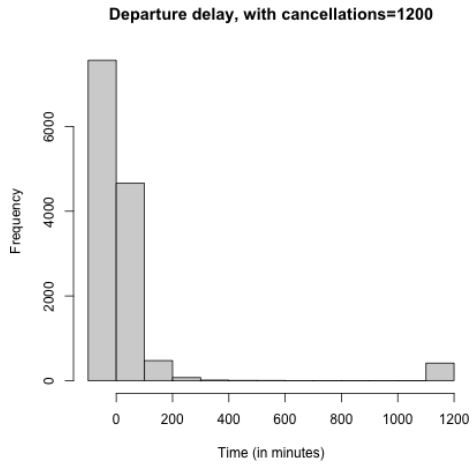
Effect of removing data

What happened to the NA's that we saw before? They are just silently not plotted.

Question: What does that mean for interpreting the histogram?

We could give the cancelled data a 'fake' value so that it plots.

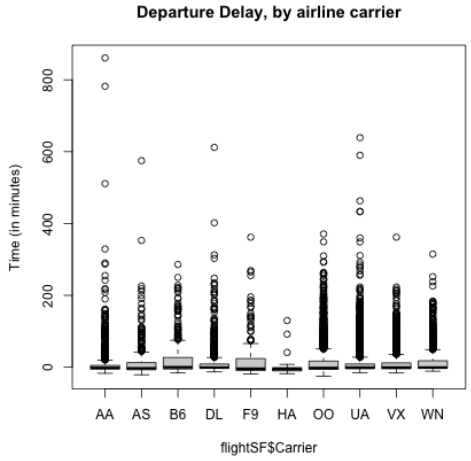
```
flightSF$DepDelayWithCancel <- flightSF$DepDelay
flightSF$DepDelayWithCancel[is.na(flightSF$DepDelay)] <- 1200
hist(flightSF$DepDelayWithCancel, xlab = "Time (in minutes)",
    main = "Departure delay, with cancellations=1200")
```



Boxplots

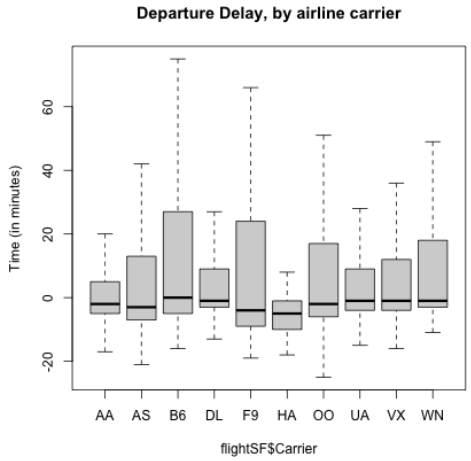
If we do boxplots separated by carrier, we can see the problem with plotting the "outlier" points

```
boxplot(flightSF$DepDelay ~ flightSF$Carrier, main = "Departure Delay, by airline carrier",
    ylab = "Time (in minutes)")
```



Here is the same plot suppressing the outlying points:

```
boxplot(flightSF$DepDelay ~ flightSF$Carrier, main = "Departure Delay, by airline carri
    ylab = "Time (in minutes)", outline = FALSE)
```



2.1.4.2 Log and Sqrt Transformations

In data like the flight data, we can remove these outliers for the boxplots to better see the median, etc, but it's a lot of data we are removing – what if the different carriers are actually quite different in the distribution of these outer points? This is a problem with visualizations of skewed data: either the outlier points dominate the visualization or they get removed from the visualization.

A common way to get around this is to transform our data, which simply means we pick a function f and turn every data point x into $f(x)$. For example, a log-transformation of data point x means that we define new data point y so

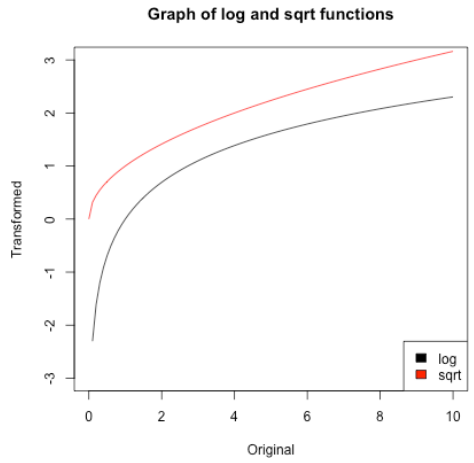
that

$$y = \log(x).$$

A common example of when we want a transformation is for data that are all positive, yet take on values close to zero. In this case, there are often many data points bunched up by zero (because they can't go lower) with a definite right skew.

Such data is often nicely spread out for visualization purposes by either the log or square-root transformations.

```
ylim <- c(-3, 3)
curve(log, from = 0, to = 10, ylim = ylim, ylab = "Transformed",
    xlab = "Original")
curve(sqrt, from = 0, to = 10, add = TRUE, col = "red")
legend("bottomright", legend = c("log", "sqrt"), fill = c("black",
    "red"))
title(main = "Graph of log and sqrt functions")
```



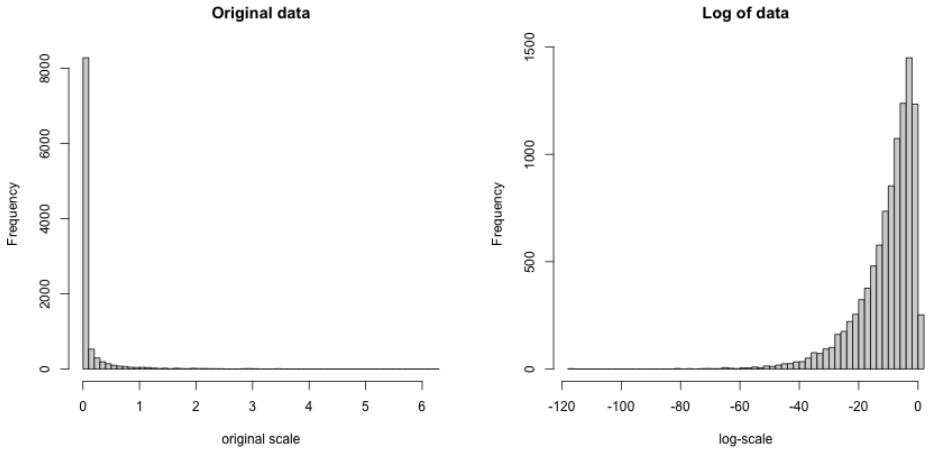
These functions are similar in two important ways. First, they are both *monotone increasing*, meaning that the slope is always positive. As a result, the rankings of the data points are always preserved: if $x_1 > x_2$ then $f(x_1) > f(x_2)$, so the largest data point in the original data set is still the largest in the transformed data set.

The second important property is that both functions are *concave*, meaning that the slope of $f(x)$ gets smaller as f increases. As a result, the largest data points are pushed together while the smallest data points get spread apart. For example, in the case of the log transform, the distance between two data points depends only on their ratio: $\log(x_1) - \log(x_2) = \log(x_1/x_2)$. Before transforming, 100 and 200 were far apart but 1 and 2 were close together, but after transforming, these two pairs of points are equally far from each other. The log scale can make a lot of sense in situations where the ratio is a better

match for our "perceptual distance," for example when comparing incomes, the difference between making $500,000 and $550,000 salary feels a lot less important than the difference between $20,000 and $70,000.

Let's look at how this works with simulated data from a fairly skewed distribution (the Gamma distribution with shape parameter 1/10):

```
y <- rgamma(10000, scale = 1, shape = 0.1)
par(mfrow = c(1, 2))
hist(y, main = "Original data", xlab = "original scale",
    breaks = 50)
hist(log(y), main = "Log of data", xlab = "log-scale",
    breaks = 50)
```



Note that in this case, after transforming the data they are even a bit *left*-skewed because the tiny data points are getting pulled very far apart: $\log(x) = -80$ corresponds to $x = e^{-80} = 1.8 \times 10^{-35}$, and $\log(x) = -40$ to $x = 4.2 \times 10^{-18}$. Still, it is much less skewed than before.

Does it make sense to use transformations? Doesn't this mess-up our data?

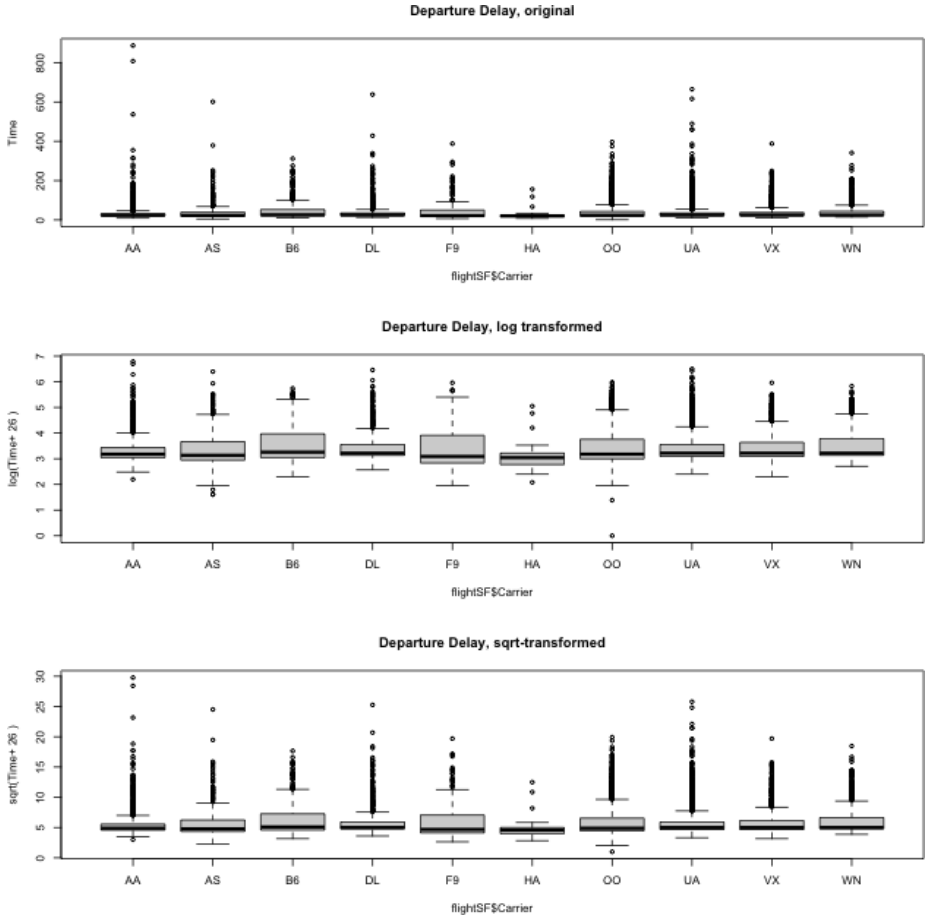
Notice an important property is that these are **monotone** functions, meaning we are preserving the rank of our data – we are not suddenly inverting the relative order of the data. But it does certainly change the meaning when you move to the log-scale. A distance on the log-scale of '2' can imply different distances on the original scale, depending on where the original data was located.[6]

[6] Of course the distance of '2' on the log-scale *does* have a very specific meaning: a distance of '2' on the (base 10) log scale is equivalent to being 100 times greater

2.1.4.3 Transforming our data sets

Our flight delay data is not so obliging as the simulated data, since it also has negative numbers. But we could, for visualization purposes, shift the data before taking the log or square-root. Here I compare the boxplots of the original data, as well as that of the data after the log and the square-root.

```
addValue <- abs(min(flightSF$DepDelay, na.rm = TRUE)) +
    1
par(mfrow = c(3, 1))
boxplot(flightSF$DepDelay + addValue ~ flightSF$Carrier,
    main = "Departure Delay, original", ylab = "Time")
boxplot(log(flightSF$DepDelay + addValue) ~ flightSF$Carrier,
    main = "Departure Delay, log transformed", ylab = paste("log(Time+",
        addValue, ")"))
boxplot(sqrt(flightSF$DepDelay + addValue) ~ flightSF$Carrier,
    main = "Departure Delay, sqrt-transformed", ylab = paste("sqrt(Time+",
        addValue, ")"))
```



Notice that there are fewer 'outliers' and I can see the differences in the bulk of the data better.

Question: Did the data become symmetrically distributed or is it still skewed?

2.2 Probability Distributions

Let's review some basic ideas of sampling and probability distributions that you should have learned in Data 8/STAT 20, though we may describe them somewhat more formally than you have seen before. If any of these concepts in this section are completely new to you or you are having difficulty with some of the mathematical formalism, I recommend that you refer to the online book for STAT 88 by Ani Adhikari that goes into these ideas in great detail.

In the salary data we have *all* salaries of the employees of SF in 2014. This a *census*, i.e. a complete enumeration of the entire population of SF employees.

We have data from the US Census that tells us the median household income in 2014 in all of San Fransisco was around \$72K.[7] We could want to use this data to ask, what was the probability an employee in SF makes less than the regional median household number?

We really need to be more careful, however, because this question doesn't really make sense because we haven't defined any notion of randomness. If I pick employee John Doe and ask what is the probability he makes less than \$72K, this is not a reasonable question, because either he did or didn't make less than that.

So we don't actually want to ask about a particular person if we are interested in probabilities – we need to have some notion of asking about a randomly selected employee. Commonly, the randomness we will assume is that a employee is randomly selected from the full population of full-time employees, with all employees having an equal probability of being selected. This is called a **simple random sample**.

Now we can ask, what is the probability of such a randomly selected employee making less than \$72K? Notice that we have exactly defined the randomness mechanism, and so now can calculate probabilities.

This kind of sampling is called a simple random sample and is what most people mean when they say "at random" if they stop to think about it. However, there are many other kinds of samples where data are chosen randomly, but not every data point is equally likely to be picked. There are, of course, also many samples that are not random at all.

Notation and Terminology

We call the salary value of a randomly selected employee a **random variable**. We can simplify our notation for probabilities by letting the variable X be short hand for the value of that random variable, and make statements like $P(X > 20K)$. We call the complete set of probabilities of a random variable X the **probability distribution** of X. Once you know all the probabilities of X you can calculate more general statements. For example, assuming X only takes on values of increments of \$1K, we have

$$P(10K \leq X \leq 20K) = P(X = 10K) + P(X = 11K) + ... + P(X = 20K)$$

So the probability distribution of X provides the entire set of possible probabilities we can calculate for X. We will frequently speak of the distribution of a random variable X, and it's important to remember that if we know the distribution of X we know everything there is to know about X.

[7] http://www.hcd.ca.gov/grants-funding/income-limits/state-and-federal-income-limits/docs/inc2k14.pdf

2.2.1 Definition of a Probability Distribution

Let's take a moment and pull together some formal ideas about probability distributions. Formal probability is not the main focus of this class, and hopefully much of this is review, but it's important to have some background and be comfortable with the notation and vocabulary.

What is a probability distribution? We've said that it is the complete set of probabilities of a random variable. For example, if we roll a six-sided dice and assume each side is equally likely, we would have the following distribution of probabilities of the possible outcomes:

k	Probability
0	1/6
1	1/6
2	1/6
...	
6	1/6

This is similar to our simple random sample of SF employee salaries – each empolyee salary is a possible outcome and each is equally likely (though obviously too long to write down as a table!).

But we don't have to have equal probabilities for each outcome to be a probability distribution. Here's a random variable that takes on the values 0, 1 or 2 with different probabilities:

k	Probability
0	0.5
1	0.25
2	0.25

These tables give us the distribution of the random variable.

Formal Definitions

Let's discuss probability distributions more formally and with mathematical notation.

A **random variable** X designates the outcome of the random experiment. The **sample space** Ω are all the possible values that our random variable X can take. For the dice example $\Omega = \{1, ..., 6\}$. A random draw from the SF salaries has an Ω equal to all of the salaries in our dataset. Ω could also be non-numeric values. For example, we could have a bag of M&Ms and randomly draw an

M&M and record the color of the M&M. X would be the color and our sample space $\Omega = \{\text{red}, \text{blue}, ...\}$.

An **event** is something we take the probability of. You often hear "event" used to describe a possible outcome of a random experiment. However, an event is actually more than just the specific outcomes that are possible (Ω). This is because we can be interested in the probability of an outcome that is a combination of values of Ω, like the probability of rolling an odd number. In that case the event actually corresponds to a set of three values in Ω, $\{1, 3, 5\}$. So an event is defined as any subset of the values of Ω. Usually we will write an event as "$X = 2$" or "$X \in \{1, 3, 5\}$" or more informally as "dice is odd". We can sometimes emphasize that this concerns the outcome of a random experiment by saying a "random event"

A **probability distribution** is a function P that gives a value between 0 and 1, inclusive, to every possible event. The value that P assigns to an event is called the probability of the event and we write it like $P(X = 2)$ or $P(X \in \{1, 3, 5\})$[8] or $P(\text{dice is odd})$. The requirements on this function P to be a probability is that

1. P **is gives a value for all subsets of Ω.** This ensures that all possible events have a probability (the probability could be zero!)
2. P **gives values only in** $[0, 1]$ This ensures we don't have negative probabilities or probabilities greater than 1. This is pretty intuitive to our idea of probability.
3. $P(X \in \Omega) = 1$ This one might be harder to parse the notation, but it means the probability you will see an outcome from Ω is 1. This is like saying the probability that my dice rolls some number between 1 and 6 is 1. This rule ensures that every possible observed outcome has been included in Ω – there's no "missing" probability.

Translating words into probabilities and events

Remember that an "event" corresponds to a set of possible outcomes. We often use words to describe events, but it's helpful to go back and forth between words and the set of outcomes that correspond to the event. Furthermore in manipulating probabilities, we will often be interested in multiple events, like "dice rolls odd" or "dice rolls number greater than 4", and want to put them together, like "what is the probability that the dice rolls an odd number OR the dice rolls a number greater than 4".

To think how to work with such a question, we want to convert between our words to a mathematical notation. We will assign events a variable like

$$A = \text{"dice rolls odd"} = X \in \{1, 3, 5\}$$

[8]The notation $\in$ means "in", as part of the set. So $X \in \{1, 3, 5\}$ means that X is one of the values 1,3, or 5

and

$$B = \text{“dice rolls number greater than 4”} = X \in \{5, 6\}.$$

Then “OR” refers to either the outcome observed is in A or the outcome observed is in B (or both!):

$$A \cup B = X \in \{1, 3, 5, 6\}$$

and our probability is defined as

$$P(A \cup B).$$

Alternatively, we might ask “what is the probability that the dice rolls an odd number AND the dice rolls a number greater than 4”. The AND refers to an outcome that is in both A and B:

$$A \cap B = \{X = 6\}$$

and the probability is written

$$P(A \cap B).$$

We call two events A and B are **mutually exclusive** if they don't share any outcomes,

$$A \cap B = \emptyset$$

For example

$$A = \text{Dice rolls an odd number} = X \in \{1, 3, 5\}$$

and

$$B = \text{Dice rolls an even number} = X \in \{2, 4, 6\}$$

are mutually exclusive events because the union of their two sets is empty.

Finally we might ask questions about an event described in the negative, like “the dice is NOT even”. Again, we have

$$A = \text{”the dice is even”} = X \in \{2, 4, 6\}$$

The NOT is the complement of A,

$$A^C = \Omega \backslash A = \text{”the dice is NOT even”} = X \in \{1, 3, 5\}.$$

Properties of Probabilities

We can now talk about a couple of important properties of P you should be familiar with for calculating the probability of multiple events.

1. If A and B are mutually exclusive events, then

$$P(A \cup B) = P(A) + P(B).$$

This can be extended to many mutually exclusive events $A, B, C, ...$

$$P(A \cup B \cup C ...) = P(A) + P(B) + P(C)$$

2. Otherwise, for general events A and B

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

3. $P(A^C) = 1 - P(A)$

Notice these rules allow me to do things like

$$P(\text{Dice rolls an odd number}) = P(X \in \{1,3,5\}) = P(X=1)+P(X=2)+P(X=3) = 3/6$$

Similarly I can make some complicated questions simple by using the negative of an event

$$P(\text{Dice is NOT one}) = P(X \in \{2,3,4,5,6\}) = 1 - P(X=1) = 1 - 1/6 = 5/6$$

In this case it is not overly complicated to figure out $P(X \in \{2,3,4,5,6\})$, since its the sum of the individual outcomes and each outcome has the same probability. But if there are a lot of outcomes, each with a different probability, $1 - P(X=1)$ is *much* easier to work with.

How would you calculate the following probabilities of a single random sample from the SF salaries?

1. $P(\text{income} = \$72K)$
2. $P(\text{income} \leq \$72\text{K})$
3. $P(\text{income} > \$200\text{K})$

2.2.1.1 Probabilities and Histograms

In the section on histograms we plotted **frequency histograms** of the SF data, where the height of each bin is the number of observations falling in the interval of the bin. Histograms of a data also have a relationship to the probability distribution of a single random sample from the data. Specifically, if the interval of a histogram bin is $(b_1, b_2]$, the probability a single sample from the data lies in this range is

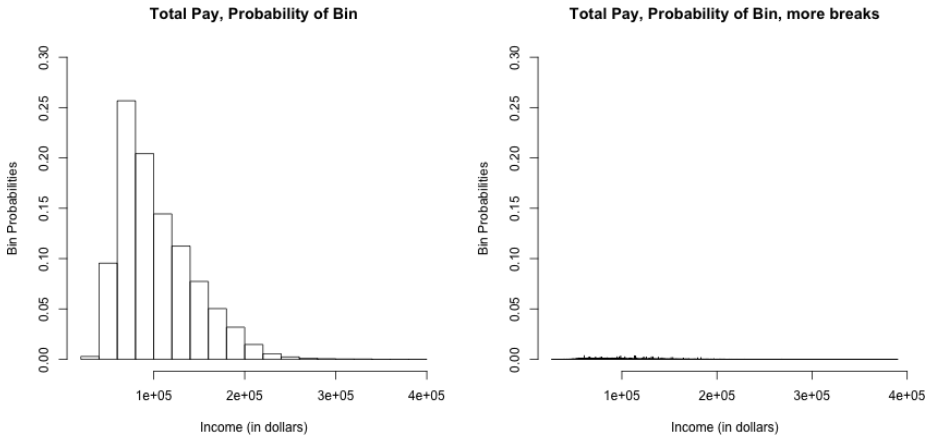
$$P(b_1 < X \leq b_2).$$

This probability is

$$P(b_1 < X \leq b_2) = \frac{\#\text{data in } (b_1, b_2]}{n}$$

The numerator of this fraction is the height of the corresponding of a frequency histogram. So the histogram gives us a visualization of what values are most probable.

I'm going to plot these probabilities for each bin of our histogram, for both large and small size bins. [9]



Question: What happens as I decrease the size of the bins?

However, be careful because this plot is for instructive purposes, and is *not* what we usually use to visualize a distribution (we will show more visualizations later). In particular, this plot is *not* the same thing as a **density histogram** that you have probably learned about. A density histogram requires that the *area* of a bin is equal to the probability of being in our bin. We will learn more about why density histograms are defined in this way when we discuss continuous distributions below, but density histograms are what should be considered as the primary tool to visualize a probability distribution. To motivate why density histograms are more useful, however, you should note a density histograms will not drop to zero values as you make the bins smaller, so you can get the sense of the spread of probabilities in the distribution more independently from the choice of the size of the bin.

2.2.1.2 Probability Mass Function (pmf)

Notice that we've subtly switched in how we describe a probability distribution. Previously I implied that a probability distribution was the complete set of probabilities of the values in Ω – that is what my tables I initially showed above were. But we see that the actual definition of a probability distribution is a function that gives a probability to every *event*. We've learned that events

[9] Plotting these probabilities is not done automatically by R, so we have to mainpulate the histogram command in R to do this (and I don't normally recommend that you make this plot – I'm just making it for teaching purposes here).

involve combinations of values of Ω, so there are a lot more events than there are values in Ω. We'll explain now why we can go back and forth between these concepts.

The function that gives the probabilities of the all the values in Ω is a separate quantity called the **probability mass function** often abbreviated as "pmf." An example from our simple table above has a probability mass function p given by

$$p(k) = P(X = k) = \begin{cases} 1/2, & k = 0 \\ 1/4, & k = 1 \\ 1/4 & k = 2 \end{cases}$$

The probability mass function is a function that goes from the values in Ω to a value in $[0, 1]$

As we will see later, not all probability distributions *have* probability mass functions. But if they do, I can actually go back and forth between the probability mass function p and the probability distribution P. By which I mean if I know one, then I can figure out the other. Clearly, if I know my probability distribution P, I can define the probability mass function p. But what is more interesting is that if I know p, I can get P, i.e. I can get the probability of *any* event. How?

Any event $X \in \{\nu_1, \nu_2, ...\}$ is a set of outcomes where the ν_i are some values in Ω. If we let $A = X \in \{\nu_1, \nu_2, ...\}$, we can write $A = X \in \nu_1 \cup X \in \nu_2 \cup ....$ Moreover, $X \in \nu_1$ and $X \in \nu_2$ are clearly mutually exclusive events because X can only take on one of those two possibilities. So for any event A we can write

$$\begin{aligned} P(A) &= P(X \in \{\nu_1, \nu_2, ...\}) \\ &= P(X \in \nu_1 \cup X \in \nu_2 \cup ...) \\ &= P(X \in \nu_1) + P(X \in \nu_2) + ... \end{aligned}$$

So we can get the entire probability distribution P from our probability mass function p. Which is fortunate, since it would be quite difficult to write down the probability of *all* events – just enumerating all events is not feasible in complicated settings.

Properties of a Probability Mass Function (pmf)

We need some restrictions about how the probabilities of the events combine together. Otherwise we could have the following probability distribution

$$P(X = k) = \begin{cases} 1, & k = 0 \\ 1, & k = 1 \\ 1 & k = 2 \end{cases}$$

Every possible outcome has probability 1 of occurring! The following example is less obvious, but still a problem

$$P(X = k) = \begin{cases} 3/4, & k = 0 \\ 3/4, & k = 1 \\ 3/4 & k = 2 \end{cases}$$

This would imply that the probability of the event $X \in \{1, 2\}$ (we get either a 1 OR a 2) would be,

$$P(X \in \{1, 2\}) = P(X = 1) + P(X = 2) = 1.5 > 1$$

These examples will violate the basic properties of P.

This means that the properties of P (the probability distribution) imply a valid probability mass function p has certain properties:

1. p is defined for all values of Ω
2. $p(k)$ is in $[0, 1]$
3. $\sum_{k \in \Omega} p(k) = 1$

2.2.2 More Examples of Probability Distributions

A random variable X is the outcome of a "random experiments" (or we could say "random process"). The only random experiments we've discussed so far is rolling a dice and randomly drawing a sample from a fixed population on which we have data (SF Full Time Employees). There are other kinds of descriptions of random processes. Here are two simple examples,

- You flip a coin 50 times and count the number of heads you get. The number of heads is a random variable (X).
- You flip a coin *until* you get a head. The number of times it takes to get a head is a random variable (Y).

These descriptions require multiple random actions, but still result in a single outcome. This outcome is a random variable because if we repeated the process we would get a different number. We could ask questions like

- What is the probability you get out of 50 flips you get 20 or more heads, $P(X \geq 20)$
- What is the probability it takes at least 20 flips to get a head, $P(Y \geq 20)$

In both examples, the individual actions that make up the random process are the same (flipping a coin), but the outcome of interest describes random variables with different distributions. For example, if we assume that the probability of heads is 0.5 for every flip, we have:

- $P(X \geq 20) = 0.94$

- $P(Y \geq 20) = 1.91 \times 10^{-6}$

Where did these numbers come from? When we were dealing with a simple random sample from a population, we had a very concrete random process for which we could calculate the probabilities. Similarly, when we flip coins, if we make assumptions about the coin flipping process (e.g. that we have a 0.5 probability of a head on each flip), we can similarly make precise statements about the probabilities of these random variables. These are standard combinatoric exercises you may have seen. For example, the probability that you get your first head in the 5th flip ($Y = 5$) is the same as saying you have exactly four tails and then a head. If the result of each flip is independent of each other, then you have (0.5 x 4) as the probability of four tails in a row, and then (0.5) as the probability of the final head, resulting in the total probability being $P(Y = 5) = (0.5)^4(0.5) = (0.5)^5$.

We can write down the entire the probability mass function $p(k)$ of both of these random variables (i.e. the probabilities of all the possible outcomes) for both of these two examples as a mathematical equation. These distributions are so common the distributions have a name:

- **Binomial Distribution**

$$p(k) = P(X = k) = \frac{n!}{k!(n-k)!} p^k (1-p)^{n-k}$$

  where $n \in \{1, 2, ...\}$ is equal to the number of flips (50), $0 \leq p \leq 1$ is the probability of heads (0.5), and $k \in \{0, 1, ..., 50\}$
- **Geometric Distribution**

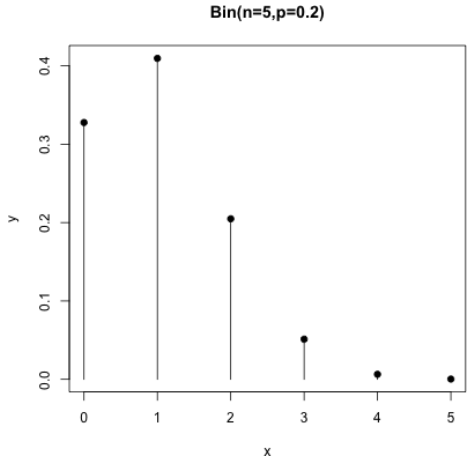
$$p(k) = P(Y = k) = (1-p)^{k-1} p$$

  where $0 \leq p \leq 1$ is the probability of heads (0.5) and $k \in \{1, ...\}$.

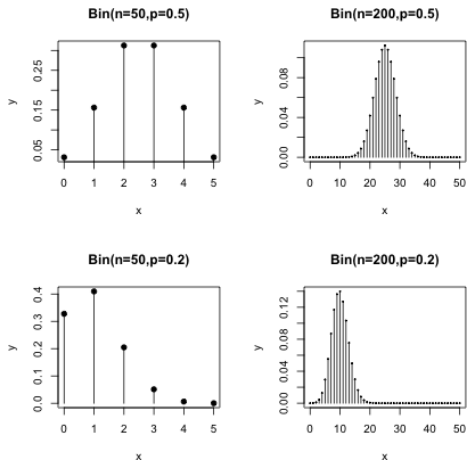
Recall that we showed that knowledge of the pmf gives us knowledge of the entire probability distribution. Thus the above equations *define* the binomial and geometric distributions.

There are many standard probability distributions and they are usually described by their probability mass functions. These standard distributions are very important in both probability and statistics because they come up frequently.

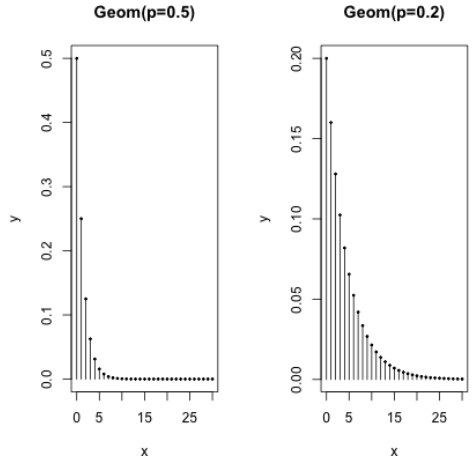
We can visualize pmfs and see how their probabilities change for different choices of parameters. Here is a plot of the binomial pmf for $n = 5$ and $p = 0.1$



Notice that the lines are just there to visualize, but the actual values are the points



Here is the geometric distribution:



Question: Think about the shape of the pmfs of these distributions. How do they change as you change p? What would you expect to happen if $p = 0.9$?

Relationship to Histograms? Notice with the lines drawn, the pdf start to look a bit like histograms. Histograms show the probability of being within an interval, where the *area* of the rectangle is the probability. Of course, there's no probability of being between 1 and 2 for the binomial distribution (you can't get 1.5 heads!), so in fact if we drew a "histogram" for this distribution, it would look similar, only the height would have to account for the size of the bins of the histogram, so would not be the actual probability of being equal to any point. We can think of these visualizations being like histograms with "infinitely small" sized bins. And we can interpret them similarly, in the sense of understanding the shape and spread of the distribution, whether it is symmetric, etc.

Common features of probability mass functions (pmfs)

Notice some common features of these pmfs. k corresponds to the possible values the random variables can take on. So to get a specific probability, like $P(X = 5)$, you would substitute 5 in for k in the equation. Also k can only take on specific values. The set of all of these values is our sample space Ω. For the binomial the sample space is $\Omega = \{0, 1, \ldots, 50\}$ and for the geometric distribution the sample space is $\Omega = \{1, ...\}$ (an infinite sample space).

There are also other variables in the equation, like p (and n for the binomial distribution). These are called **parameters** of the distribution. These are values that you set depending on your problem. For example, in our coin problem, the probability of a head was $p = 0.5$ and the total number of flips was $n = 50$. However, this equation could be also be used if I changed my setup and decided that $n = 2000$. It is common for a standard distribution to have parameters that

you need to set. This allows for a single expression that can be used for multiple settings. However, it's important to recognize which values in the equation are parameters defined by your setting (p and n) and which is specific to probability you decided to calculate (k). In the end you need all of them defined, of course, to calculate a specific probability.

Similar to k, sometimes parameters can only take on a limited range of values. p for example has to be between 0 and 1 (it's a probability of heads – makes sense!), and n needs to be a positive integer. The set of values allowed for parameters is called the **parameter space**.

Notation conventions

Please be aware that the choice of variables for all of these equations and for the random variable is arbitrary! Here are some common variations to be ready for

- We can use many different variables for the random variable, usually capitalized. Commonly they are at the end of the alphabet, X, Y, Z, and even U, V, and W.
- I used "k" in the probability mass function to indicate the particular outcome of which we want to calculate the probability. This is common (especially for the distributions we are considering right now). But it's also common to write $p(x) = P(X = x)$. The capital letter, e.g. X, is for keeping track of the random variable and the lower case letter (of the same letter, e.g. x) is for indicating the particular outcome of which we want to calculate the probability. That outcome we are calculating the probability of is also called a **realization** of X. This notation can be confusing, but as we'll see it is also a notation that is easier to expand to multiple random variables e.g.

  $$P(X = x, Y = y, Z = z),$$

  without needing to introduce a large number of additional variables. Otherwise we start to run out of letters and symbols once we have multiple random variables – we don't want statements like $P(W = v, X = y, Z = u)$ because it's hard to remember which value goes with which random variable.
- The choice of variables for the parameters change a good bit from person to person. It is common that they are Greek letters (α, β, θ, ψ, ϕ, λ, μ,σ,τ, π are all common). This helps the parameters stand out from the other variables floating around in the equation. This is obviously not a universal rule, as the two distributions above clearly demonstrate (p for a probability is a quite common choice of parameter…)
- The choice of $p(k)$ for the pmf is common, but it can also be a different letter. It's not uncommon to see $f(k)$ or $g(k)$.
- A probability distribution is a function generally called P, and this is why we write $P(X = 2)$. P *is* pretty universal – so you don't even have to

explain P denotes a probability distribution when you write $P(X = 2)$. But even this we could change to another letter, like Q; in this case we'd write $Q(X = 2)$ and it would still be a probability. Doing so is mainly when we might want to consider multiple distributions, and we need different letters to keep them apart.

Probability Calculations In R

Calculations with these standard probability distributions are built into R. Specifically functions `dbinom` and `dgeom` calculate $P(X = k)$ for the binomial and geometric distributions.

```
dbinom(2, size = 50, prob = 0.5)
```

```
## [1] 1.088019e-12
```

```
dgeom(3 - 1, prob = 0.5)
```

```
## [1] 0.125
```

and the functions `pbinom` and `pgeom` calculate $P(X \leq k)$ for these distributions

```
pbinom(19, size = 50, prob = 0.5)
```

```
## [1] 0.05946023
```

```
pgeom(19 - 1, prob = 0.5)
```

```
## [1] 0.9999981
```

Question: How can you put these results together to get $P(X \geq 20)$?

2.2.2.1 Modeling real-life settings

We can also have situations in life that are usefully thought of as a random variables but are not well described as the result of sampling from a population:

- Suppose 5% of adults experience negative side-effects from a drug which result in a negative side effect. A research study enrolls 200 adults using this drug. The number of people in the study experiencing these negative side-effects can be considered a random variable.
- A polling company wants to survey people who have do not have college diplomas about their job experiences; they will call random numbers until they reach someone without a college diploma, and once they identify someone will ask their questions. The number of people the polling company will have to call before reaching someone without a college diploma can be considered as a random variable.

- A call center generally receives an average of 180 calls per day in 2023. The actual number of calls in a particular day can be considered as a random variable.

Already, we can see that analyzing these real life situations through the lens of probability is tricky, since our descriptions are clearly making simplifications (shouldn't the volume of calls be more on some days of the weeks than others?). We could make more complicated descriptions, but there will be a limit. This is always a non-trivial consideration. At the same time, it can be important to be able to quantify these values to be able to ask questions like: do we need more staff members at the call center? Is our polling strategy for identifying non-college graduates feasible, or will it take too long? We just have to be careful that we consider the limitations of our probability **model** and recognize that it is not equivalent to the real-life situation.

The other tricky question is how to make these real-life situations quantifiable – i.e. how can we actually calculate probabilities? However, if you look at our examples, the first two actually look rather similar to the two coin-flipping settings we described above:

- Suppose we let "coin is heads" be equated to "experience negative side-effects from the drug", and "the total number of coin flips" be "200 adults using this drug". Then our description is similar to the binomial coin example. In this case the probability p of heads is given as $p = 0.05$ and $n = 200$. And X is the number of people in the study experiencing negative side-effects.
- Suppose we let a "coin flip" be equated to "call a random number", and "coin is heads" be equated to "reach someone without a college diploma". Then this is similar to our description of the geometric distribution, and our random variable Y is the number of random numbers that will need to be called in order to reach someone without a college diploma. Notice, however, that the geometric distribution has a parameter p, which in this case would translate into "the probability that a random phone number is that of a person without a college diploma". This parameter was not given in the problem, so we can't calculate any probabilities for this problem. A reasonable choice of p would be the proportion of the entire probability that does not have a college diploma, an estimate of which we could probably grab from government survey data.

The process of relating a probability distribution to a real-life setting is often referred to as *modeling*, and it will never be a perfect correspondence – but can still be useful! We will often spend a lot of time considering how the model deviates from reality, because being aware of a model's shortcomings allows us to think critically about whatever we calculate. But it doesn't mean it is a bad practice to model real life settings. Notice how in this last example of the polling survey, by translating it to a precise probability distribution, it becomes clear what additional information (p) we need to be able to make the probability calculations for this problem.

Returning to our examples, the example of the call center is not easily captured by the two distributions above, but is often modeled with what is called a Poisson distribution

- **Poisson Distribution** $P(Z = k) = \frac{\lambda^k e^{-\lambda k}}{k!}$ where $\lambda > 0$.

When modeling a problem like the call center, the needed parameter λ is the *rate* of the calls per the given time frame, in this case $\lambda = 180$.

2.2.3 Conditional Probability and Independence

Previously we asked about the population of all FT employees, so that X is the random variable corresponding to income of a randomly selected employee from *that population*. We might want to consider asking questions about the population of employees making less than \$72K. For example, low-income in 2014 for an individual in San Francisco was defined by the same source as \$64K – what is the probability of a random employee making less than \$72K to be considered low income?

We can write this as $P(X \leq 64 \mid X < 72)$, which we say as the probability a employee is low-income *given that* or *conditional on* the employee makes less than the median income. A probability like $P(X \leq 64 \mid X < 72)$ is often called a **conditional probability.**

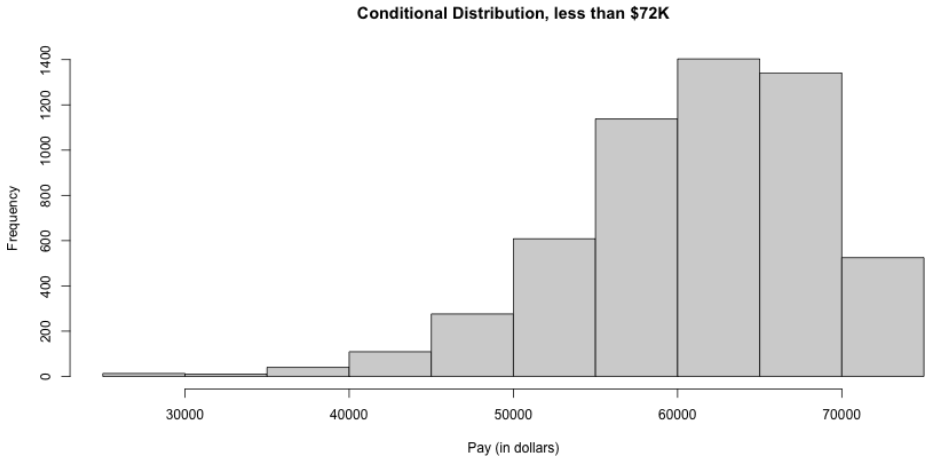
Question: How would we compute a probability like this?

Note that this is a different probability than $P(X \leq 64)$.

Question: How is this different? What changes in your calculation?

Once we condition on a portion of the population, we've actually defined a new random variable. We could call this new random variable a new variable like Y, but we usually notated it as $X \mid X > 72K$. Since it is a random variable, it has a new probability distribution, which is called the **conditional distribution**. We can plot the histogram of this conditional distribution:

```
condPop <- subset(salaries2014_FT, TotalPay < 72000)
par(mfrow = c(1, 1))
hist(condPop$TotalPay, main = "Conditional Distribution, less than $72K",
    xlab = "Pay (in dollars)")
```



We can think of the probabilities of a conditional distribution as the probabilities we would get if we repeatedly drew X from its marginal distribution but only "keeping" it when we get one with $X < 72K$.

Consider the flight data we looked at briefly above. Let X for this data be the flight delay, in minutes, where if you recall NA values were given if the flight was cancelled.

Question: How would you state the following probability statements in words?

$$P(X > 60 | X \neq \text{NA})$$

$$P(X > 60 | X \neq \text{NA} \& X > 0)$$

Conditional probability versus population

Previously we limited ourselves to only full-time employees, which is also a subpopulation of the entire set of employees. Why weren't those conditional probabilities too? It's really a question of what is our reference point, i.e. how have we defined our full population. When we write a conditional probability, we keeping our reference population the same, but writing questions that consider a subpopulation of it. This allows us, for example, to write statements regarding different subpopulations in the same sentence and compare them, for example $P(X \leq 64 \mid X < 72)$ versus $P(X \leq 64 \mid X > 72)$, because the full population space is the same between the two statements (full-time employees), while the conditional statement makes clear that we are considering a subpopulation.

2.2.3.1 Formal Notation for Conditional Probabilities

Our more formal notation of probability distributions can be extended to condition probabilities. We've already seen that a probability is a value given to a *random event*. Our conditional probabilities deal with *two* random events. The conditional probability

$$P(X \leq 64 \mid X < 72)$$

involves the event $A = X \leq 64$ and $B = X < 72$. Our conditional probability is thus $P(A|B)$.

Note that $P(A|B)$ (outcome is in A given outcome is in B) is different from $P(A \cap B)$ (outcome is in both A and B=) or $P(A \cup B)$ (outcome is in either A or B). Remember the conditional distribution is really just a completely distribution than P – we cannot create an event from Ω (or combination of events) whose probability would (always) be equal to $P(A|B)$. We noted above that we could call this new distribution Q and just write $Q(A)$ with possibly a different sample space Ω and definitely completely different values assigned to A than P would provide. However, there is an important property that links $P(A|B)$ to $P(A \cap B)$:

$$P(A \cap B) = P(A|B)P(B)$$

You can also see this written in the equivalent form

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

This formula relates our conditional probability distribution to our probability distribution P. So there's not an event whose probability using P equals $P(A|B)$, we *can* make calculations of the distribution of $A|B$ using the distribution P. This is an important reason why we don't just create a new distribution Q – it helps us remember that we can go back and forth between these distributions.

For example, for the dice example, we can say something like

$$\begin{aligned} P(X \in \{1,2,3\}|X \in \{2,4,5\}) &= \frac{P(X \in \{1,2,3\} \cap X \in \{2,4,5\})}{P(X \in \{2,4,5\})} \\ &= \frac{1/6}{3/6} = 1/3 \end{aligned}$$

We can compare this to

$$P(X \in \{1,2,3\}) = P(X=1) + P(X=2) + P(X=3) = 3/6 = 1/2$$

So of all rolls where the outcome has X in $\{2,4,5\}$ the probability of X being in $\{1,2,3\}$ is smaller than if we consider all rolls of the dice.

2.2.3.2 Independence

If we are considering two events, we frequently talk about events being independent. Informally, we say two events are independent if whether one event happens doesn't affect whether the other event happens. For example, if we roll a dice twice, we don't expect the fact the first die rolled is an odd number to affect whether the second rolled is even (unless someone is cheating!). So if we model these two dice rolls as a random process, we'd usually assume these two events are independent.

However we might have two events describing the outcome of a *single* role of the dice. We've seen many examples above, where we might have an event $X \in \{1, 2, 3\}$ and $X \in \{2, 4, 5\}$ and consider the joint probability of these

$$P(X \in \{1, 2, 3\} \cap X \in \{2, 4, 5\})$$

or the conditional probability

$$P(X \in \{1, 2, 3\} | X \in \{2, 4, 5\})$$

We can similarly consider whether these two events are independent. It's clearly a trickier question to answer on the same role of the dice, but it doesn't seem like it should be independent. Clearly if you know whether "dice is odd" should have an effect on whether "dice is even" when it's the same dice roll!

The formal definition of independence allows us to answer this question. Two events A and B are defined as **independent** if

$$P(A \cap B) = P(A)P(B)$$

Notice that this means that if two events are independent

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A)P(B)}{P(B)} = P(A)$$

So if A is independent from B, the probability of A is the same regardless of the outcome of B.

2.2.4 Expectation and Variance

The last formal probability idea I want to review is the expectation and variance of a distribution. These are things we can calculate from a probability distribution that describe the probability distribution, similar to how we can calculate summary statistics from data to summarize the dataset.

Expectation

The **expectation** or **mean** of a distribution is defined as

$$E(X) = \sum_{k \in \Omega} kp(k)$$

where $E(X)$ stands for the expectation of X.[10]

For example, for our dice example $\Omega = \{1, 2, 3, 4, 5, 6\}$ and $p(k) = 1/6$ for all k. So we have

$$\begin{aligned} E(X) = \sum_{k \in \Omega} kp(k) &= 1P(X = 1) + 2P(X = 2) + ... + 6P(X = 6) \\ &= 1/6(1 + 2 + ... + 6) \\ &= 21/6 = 3.5 \end{aligned}$$

Notice that because each outcome is equally likely in this example, the expectation is just the mean of all the values in Ω; this is why the expectation of a distribution is also called the mean of the distribution.

Consider our earlier simple example where we *don't* have equal probabilities,

$$p(k) = P(X = k) = \begin{cases} 1/2, & k = 0 \\ 1/4, & k = 1 \\ 1/4 & k = 2 \end{cases}$$

In this case

$$\begin{aligned} E(X) = \sum_{k \in \Omega} kp(k) &= 0P(X = 0) + 1P(X = 1) + 2P(X = 2) \\ &= 0 + 1/4 + 1/2 \\ &= 3/4 \end{aligned}$$

This is smaller than the average of the values in Ω (which would be 1). This is because we have more probability on zero, which pulls down our expectation. In the case of unequal probabilities, the expectation can be considered a *weighted* mean, meaning it gives different weights to different possible outcomes depending on how likely they are.

Variance

Just as there is a mean defined for a probability distribution, there is also a variance defined for a probability distribution. It has the same role as the sample variance of data – to describe the spread of the data. It has a similar definition

$$var(X) = E(X - E(X))^2 = \sum_{k \in \Omega} (k - E(X))^2 p(k)$$

The variance of a probability distribution measures the average distance a random draw from the distribution will be from the mean of the distribution – a measure of spread of the distribution.

[10] This is the definition of a distribution with a pdf. We will expand this definition for other distributions when we discuss continuous distributions.

Notice the similarity to the equation for the variance for data:

$$\frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2 = \sum_{i=1}^{n}(X_i - \bar{X})^2\frac{1}{n-1}$$

The equations are pretty much equivalent, except that for the variance of a probability distribution we weight different values differently based on how likely they are, while the data version weighs each observation equally.[11]

Properties of Expectation and Variance

The following properties are important to know for calculations involving expectation and variance

1. $E(a + bX) = a + bE(X)$
2. $var(a+bX) = b^2 var(X)$ – adding a constant to a random variable doesn't change the variance
3. Generally, $E(g(X)) \neq g(E(X))$ and $var(g(X)) \neq g(var(X))$
4. $var(X) = E(X - E(X))^2 = E(X^2) - [E(X)]^2$

2.3 Continuous Distributions

Our discussions so far primarily relied on probability from **discrete distributions**. This often has meant that the complete set of possible values (the sample space Ω) that can be observed is a finite set of values. For example, if we draw a random sample from our salary data we know that only the 35711 unique values of the salaries in that year can be observed – not all numeric values are possible. We saw this when we asked what was the probability that we drew a random employee with salary exactly equal to \$72K.

We also saw the geometric distribution where the sample space $\Omega = \{1, 2, 3, ...\}$. The geometric distribution is also a discrete distribution, even though Ω is infinite. This is because we cannot observe 1.5 or more generally the full range of values in any interval – the distribution does not allow continuous values.[12]

However, it can be useful to think about probability distributions that allow for all numeric values in a range (i.e. continuous values). These are **continuous distributions**.

Continuous distributions are useful even for settings *when we know the actual population is finite*. For example, suppose we wanted to use this set of data about SF salaries to make decisions about policy to improve salaries for a certain class

[11] Of course $1/(n-1)$ can't be a probability for n samples because they wouldn't sum to 1! It would need to be $1/n$ to be a probability. This is a small difference from the definition for a probability distribution; we won't go into the reasons for $1/(n-1)$ right now.

[12] For more mathematically minded, a discrete distribution is one where Ω is a countable set, which can be infinite.

of employees. It's more reasonable to think that there is an (unknown) probability distribution that defines what we expect to see for that data that is defined on a continuous range of values, not the specific ones we see in 2014. We might reasonably assume the sample space of such a distribution is $\Omega = [0, \infty)$. Notice that this definition still kept restrictions on Ω – only non-negative numbers; this makes sense because these are salaries. But the probability distribution on $\Omega = [0, \infty)$ would still be a continuous distribution because its a restriction to an interval and thus takes on all the numeric values in that interval.

Of course some data are "naturally" discrete, like the set of job titles or the number of heads in a series of coin tosses, and there no rational way to think of them being continuous.

2.3.1 Probability with Continuous distributions

Many of the probability ideas we've discussed carry forward to continuous distributions. Specifically, our earlier definition of a probability distribution is universal and includes continuous distributions. But some probability ideas become more complicated/nuanced for continuous distributions. In particular, for a discrete distribution, it makes sense to say $P(X = 72K)$ (the probability of a salary exactly equal to $72K$). For continuous distributions, such an innocent statement is actually fraught with problems.

To see why, remember what you know about a probability distributions. In particular, a probability must be between 0 and 1, so

$$0 \leq P(X = 72,000) \leq 1$$

Moreover, this is a property of any probability statement, not just ones involving '=': e.g. $P(X \leq 10)$ or $P(X \geq 0)$. This is a fundamental rule of a probability distribution that we defined earlier, and thus also holds true for continuous distributions as well as discrete distributions.

Okay so far. Now another thing you learned is if I give all possible values that my random variable X can take (the sample space Ω) and call them $v_1, \ldots, v_K$, then if I sum up all these probabilities they must sum exactly to 1,

$$P(\Omega) = \sum_{i=1}^{K} P(X = v_i) = 1$$

Well this becomes more complicated for continuous values – this leads us to an infinite sum since we have an infinite number of possible values. If we give *any* positive probability (i.e. $\neq 0$) to each point in the sample space, then we won't 'sum' to one.[13] These kinds of concepts from discrete probability just don't translate over exactly to continuous random variables.

[13]For those with more math: convergent infinite series can of course sum to 1. This is the case for distributions like the geometric distribution, which is a distribution and has an infinite

To deal with this, *continuous distributions do not allow any positive probability for a single value*: if X has a continuous distribution, then $P(X = x) = 0$ for any value of x.

Instead, continuous distributions only allow for positive probability of an *interval*: $P(x_1 \leq X \leq x_2)$ *can* be greater than 0.

Question: Note that this also means that for continuous distributions $P(X \leq x) = P(X < x)$. Why?

Key properties of continuous distributions

1. $0 \leq P(A) \leq 1$, inclusive.
2. Probabilities are only calculated for events that are intervals, not individual points/outcomes.
3. $P(\Omega) = 1$.

Giving zero probability for a single value isn't so strange if you think about it. Think about our flight data. What is your intuitive sense of the probability of a flight delay of exactly 10 minutes – and not 10 minutes 10 sec or 9 minutes 45 sec? You see that once you allow for infinite precision, it is actually reasonable to say that *exactly* 10 minutes has no real probability that you need worry about.

For our salary data, of course we don't have infinite precision, but we still see that it's useful to think of ranges of salary – there is no one that makes exactly \$72K, but there is 1 employee within \$1 dollar of that amount, and 6 employees within \$10 dollars of that amount. These are all equivalent salaries in any practical discussion of salaries.

What if you want the chance of getting a 10 minute flight delay? Well, you really mean a small interval around 10 minutes, since there's a limit to our measurement ability anyway. This is what we also do with continuous distributions: we discuss the probability in terms of increasingly small intervals around 10 minutes.

The mathematics of calculus give us the tools to do this via integration. In practice, the functions we want to integrate are not tractable anyway, so we will use the computer. We are going to focus on understanding how to think about continuous distributions so we can understand the statistical question of how to *estimate* distributions and probabilities (rather than the more in-depth probability treatment you would get in a probability class).

---

sample space Ω. But we are working with the continuous real line (or an interval of the real line), and there is not a bijection between the integers and the continuous line. The interval of the real line isn't a countable set.

2.3.2 Cummulative Distribution Function (cdfs)

For discrete distributions, we can *completely* describe the distribution of a random variable by describing the probability mass function. In other words, knowing $P(X = v_i)$ for all possible values of v_i in the sample space Ω completely defines the probability distribution.

However, we just said that $P(X = x) = 0$ for any value of x, so clearly the probability mass function doesn't make sense for continuous distributions, and certainly doesn't define the distribution. Indeed, continuous distributions are not considered to have a probability mass function.

Are we stuck going back to basics and defining the probability of every possible event? All events with non-zero probability can be described as a combination of intervals, so it suffices to define the probably of every single possible *interval.* This is still a daunting task since there are an infinite number of intervals, but we can use the simple fact that

$$P(x_1 < X \leq x_2) = P(X \leq x_2) - P(X \leq x_1)$$

Question: Why is this true? (Use the case of discrete distribution to reason it out)

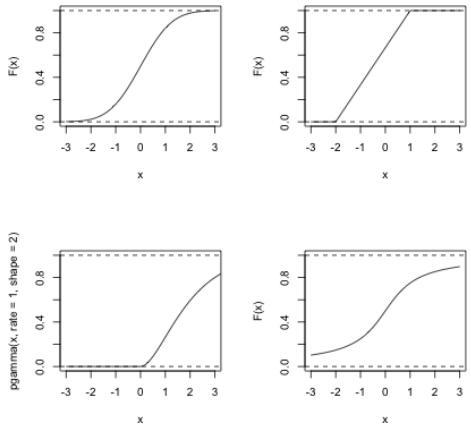
Thus rather than define the probably of every single possible interval, we can tackle the simpler task to define $P(X \leq x)$ for every single x on the real line. That's just a function of x

$$F(x) = P(X \leq x)$$

F is called a **cumulative distribution function (cdf)** and it has the property that if you know F you know P, i.e. you can go back and forth between them.

While we will focus on continuous distributions, discrete distributions can also be defined in the same way by their cumulative distribution function instead of their probability mass functions. In fact cumulative distribution functions are the most general way to numerically describe a probability distribution^[More specifically, for all real-valued probability distributions, where Ω is a subset of R^p.

Here are some illustrations of different F functions for x between -3 and 3:



Question: Consider the following questions about a random variable X defined by each of these distributions:

- Which of these distributions is likely to have values of X less than -3?
- For which is it equally likely for X to be positive or negative?
- What is $P(X > 3)$ – how would you calculate that from the cdfs pictured above? Which of the above distributions are likely to have $P(X > 3)$ be large?
- What is $\lim_{x\to\infty} F(x)$ for all cdfs? What is $\lim_{x\to-\infty} F(x)$ for all cdfs? Why?

2.3.3 Probability Density Functions (pdfs)

You see from these questions at the end of the last section, that you can make all of the assessments we have discussed (like symmetry, or compare if a distribution has heavier tails than another) from the cdf. But it is not the most common way to think about the distribution. More frequently the **probability density function (pdf)** is more intuitive. It is similar to a histogram in the information it gives about the distribution and is the continuous analog of the probability mass functions for discrete distributions.

Formally, the pdf $p(x)$ is the derivative of $F(x)$: if $F(x)$ is differentiable

$$p(x) = \frac{d}{dx}F(x).$$

If F isn't differentiable, the distribution doesn't have a density, which in practice you will rarely run into for continuous variables.[14]

[14]Discrete distributions have cdfs where $F(x)$ is not differentiable, so they do not have densities. But even some continuous distributions can have cdfs that are non-differentiable

Conversely, $p(x)$ is the function such that if you take the area under its curve for an interval (a,b), i.e. take the integral of $p(x)$, that area gives you probability of that interval:

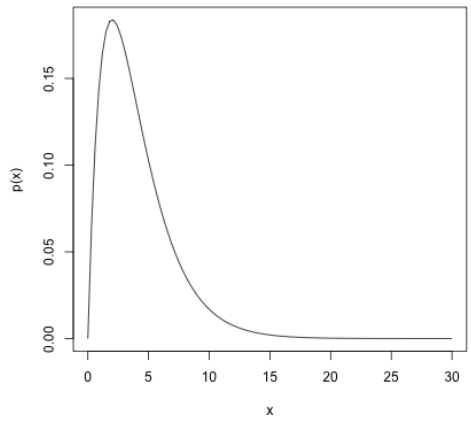
$$\int_a^b p(x) = P(a \leq X \leq b) = F(b) - F(a)$$

More formally, you can derive $P(X \leq v) = F(v)$ from $p(x)$ as

$$F(v) = \int_{-\infty}^{v} p(x)dx.$$

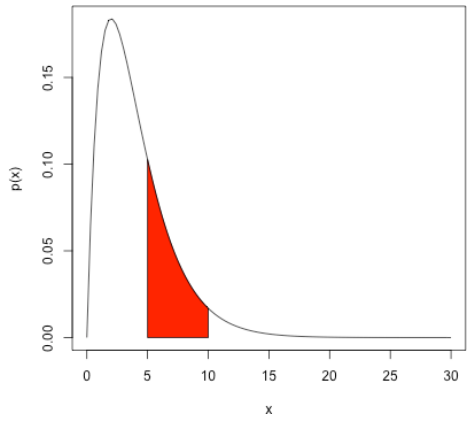
Let's look at an example with the following pdf, which is perhaps vaguely similar to our flight or salary data, though on a different scale of values for X,

$$p(x) = \frac{1}{4}xe^{-x/2}$$



Suppose that X is a random variable from a distribution with this pdf. Then to find $P(5 \leq X \leq 10)$, I find the area under the curve of $p(x)$ between 5 and 10, by taking the integral of $p(x)$ over the range of $(5, 10)$:

$$\int_5^{10} \frac{1}{4}xe^{-x/2}$$



In this case, we can actually solve the integral through integration by parts (which you may or may not have covered),

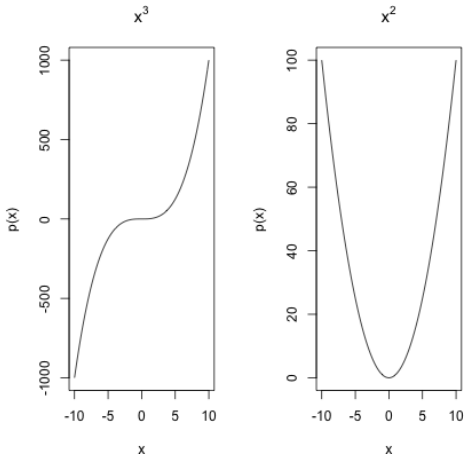
$$\int_5^{10} \frac{1}{4} x e^{-x/2} = (-\frac{1}{2} x e^{-x/2} - e^{-x/2}) \Big|_5^{10} = (-\frac{1}{2}(10)e^{-10/2} - e^{-10/2}) - (-\frac{1}{2}(5)e^{-5/2} - e^{-5/2})$$

Evaluating this gives us $P(5 \leq X \leq 10) = 0.247$. Most of the time, however, the integrals of common distributions that are used as models for data have pdfs that cannot be integrated by hand, and we rely on the computer to evaluate the integral for us.

Question: Recall above that same rule from discrete distribution applies for the total probability, namely that the probability of X being in the entire sample space must be 1. For continuous distributions the sample space is generally the whole real line (or a specific interval). What does this mean in terms of the total area under the curve of $p(x)$?

Question: The following plots show functions that cannot be pdfs, at least not over the entire range of $\Omega = (-\infty, \infty)$, why?

What if I restrict $\Omega = [-10, 10]$, could these functions be pdfs?



Interpreting density curves

"Not much good to me" you might think – you can't evaluate $p(x)$ and get any probabilities out. It just requires the new task of finding an area. However, finding areas under curves is a routine integration task, and even if there is not an analytical solution, the computer can calculate the area. So pdfs are actually quite useful.

Moreover, $p(x)$ *is* interpretable, just not as a direct tool for probability calculations. For smaller and smaller intervals you are getting close to the idea of the "probability" of $X = 72K$. For this reason, where discrete distributions use $P(X = 72K)$, the closest corresponding idea for continuous distributions is $p(72,000)$: though $p(72,000)$ is not a probability like $P(X = 72,000)$ the value of $p(x)$ gives you an idea of more likely regions of data.

More intuitively, the curve $p(x)$ corresponds to the idea of of a histogram of data. It's shape tells you about where the data are likely to be found, just like the bins of the histogram. We see for our example of $\bar{X}$ that the histogram of $\bar{X}$ (when properly plotted on a density scale) approaches the smooth curve of a normal distribution. So the same intuition we have from the discrete histograms carry over to pdfs.

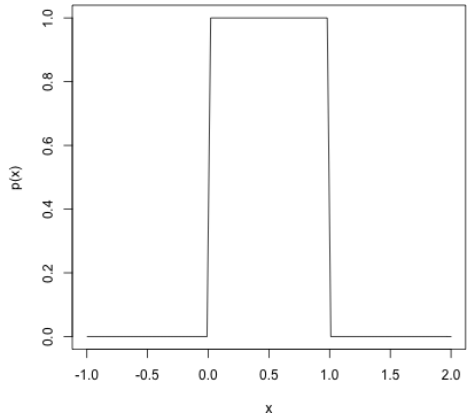
Properties of pdfs

1. A probability density function gives the probability of any interval by taking the area under the curve
2. The total area under the curve $p(x)$ must be exactly equal to 1.
3. Unlike probabilities, the value of $p(x)$ can be ≥ 1 (!).

This last one is surprising to people, but $p(x)$ is not a probability – only the area under it's curve is a probability.

To understand this, consider this very simple density function:

$$p(x) = \begin{cases} 1 & x \in [0,1] \\ 0 & x > 1, x < 0 \end{cases}$$



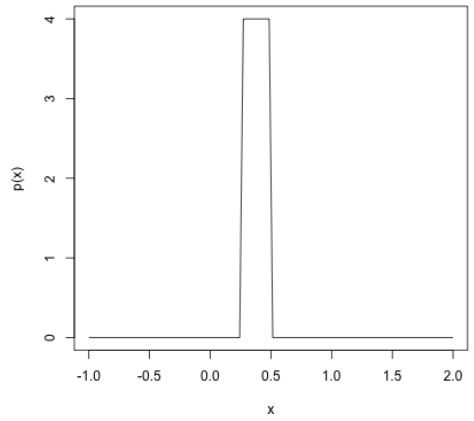
This is a density function that corresponds to a random variable X that is equally likely for any value between 0 and 1.

Question: Why does this density correspond to being equally likely for any value between 0 and 1?

Question: What is the area under this curve? (Hint, it's just a rectangle, so...)

This distribution is called a *uniform distribution* on [0,1], some times abbreviated $U(0,1)$.

Suppose instead, I want density function that corresponds to being equally likely for any value between $1/4$ and $1/2$ (i.e. $U(1/4, 1/2)$).



Again, we can easily calculate this area and confirm it is equal to 1. This is why $p(x)$ must be able to take on values greater than 1 – if $p(x)$ was required to be less than one, you couldn't get the total area of this rectangle to be 1.

You can see that the scale of values that X takes on matters to the value of $p(x)$. If X is concentrated on a small interval, then the density function will be quite large, while if it is diffuse over a large area the value of the density function will be small.

Example: Changing the scale of measurements:

Suppose my random variable X are measurements in centimeters, with a normal distribution, $N(\mu = 100\text{cm}, \sigma^2 = 100\text{cm}^2)$.

Question: What is the standard deviation?

Then I decide to convert all the measurements to meters (FYI: 100 centimeters=1 meter).

Question: What is now the mean? And standard deviation?

Expectation and Variance

The density has a similar role in calculation expectations and variances as the pmf for discrete distributions

$$E(X) = \int_{\Omega} xp(x)dx$$

$$var(X) = \int_{\Omega} (x - E(X))^2 p(x)dx$$

We basically replace the sum with an integral. We won't be doing these calculations in this class (they are generally intractable), but it is important to know that the same definitions and properties of variance and expectation carry over.

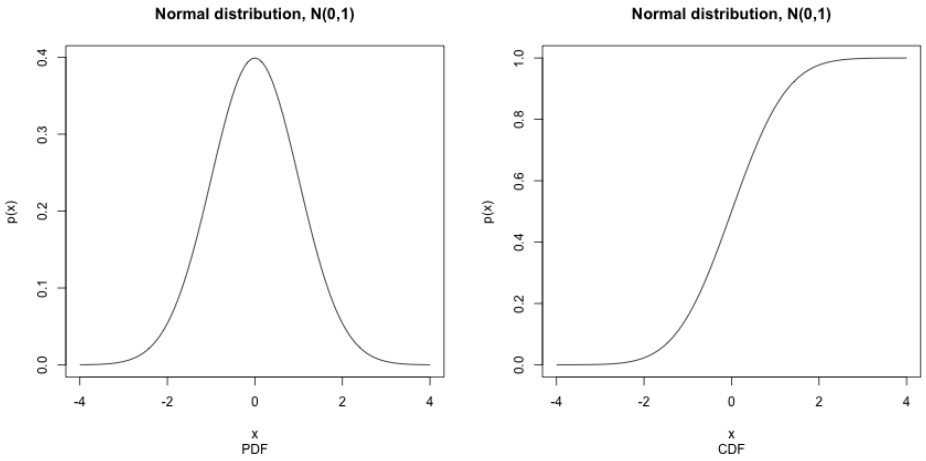
2.3.4 Examples of Continuous Distributions

Let's look at some examples of common probability distributions to make the pdf more concrete.

The most well-known probability distribution is the **normal distribution** with pdf

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

It's a mouthful, but easy for a computer to evaluate. It has two **parameters** that define the distribution: its mean μ and variance σ^2 (recall the variance is the standard deviation squared). We often write a normal distribution as $N(\mu, \sigma^2)$ to indicate its parameters. A standard normal distribution is $\mu = 0$,$\sigma^2 = 1$, written $N(0, 1)$



The cdf of the normal – the integral of this equation – is intractable to write down, but again easy for a computer to approximate to arbitrarily good precision.

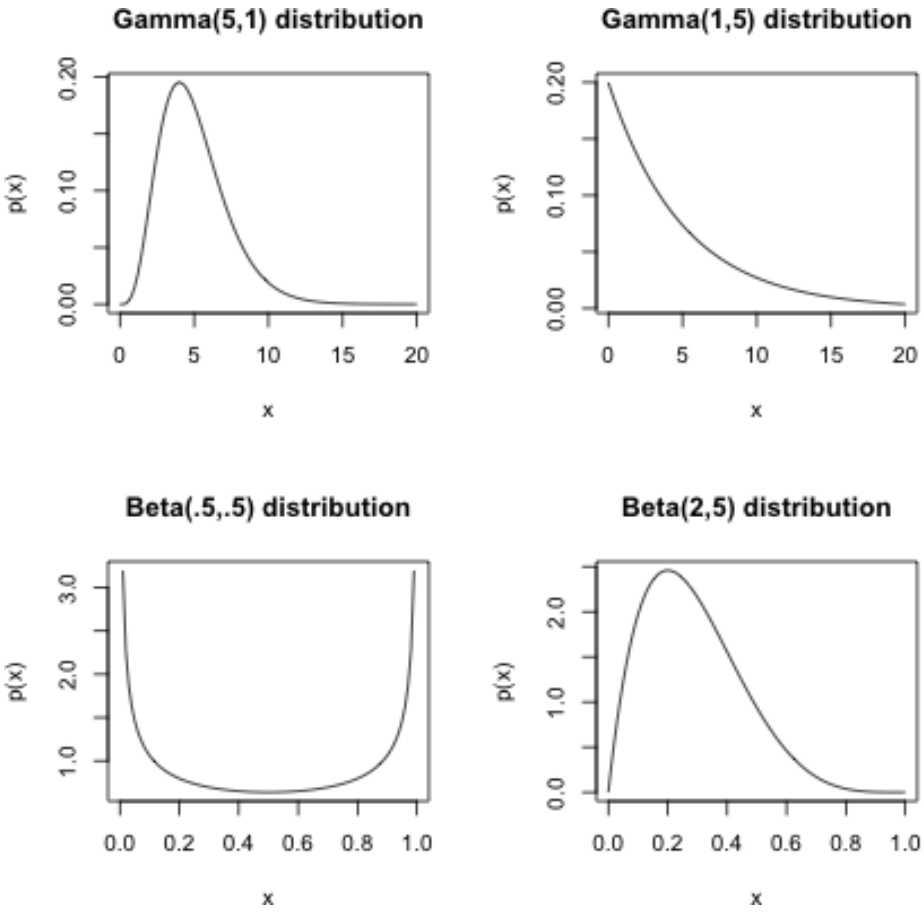
For a normal distribution, the probability of being within 1 standard deviation of μ is roughly 0.68 and the probability of being within 2 standard deviations of μ is roughly 0.95.

```
## [1] 0.6826895
```

```
## [1] 0.9544997
```

Question: What is the probability that a observed random variable from a $N(\mu, \sigma^2)$ distribution is *less* than μ by more than 2σ?

Other distributions Here are some examples of some pdfs from some two common continuous distributions other than the normal:



These are all called **parametric distributions**. The discrete distributions we introduced earlier (the Binomial, Geometric, and Poisson) are also parametric distributions, only instead of a probability density function, they have probability mass functions. You should look back on the introduction of those distributions to see the discussion of important aspects of the notation for parametric distributions which will also be true for continuous distributions. We make a few more comments illustrated by these examples:

- that 'a' parametric distribution is actually a family of distributions that differ by changing the **parameters**. For example, a normal distribution

has two parameters, the mean and the standard deviation. All possible normal distributions can be created by combinations a mean and a standard deviation parameter. We say "the" normal distribution to encompass all of these distributions.
- Unlike the normal, many distributions have very different shapes for different parameters
- Continuous distributions can be limited to an interval or region (i.e. Ω does not take on all values of the real line). They are still considered continuous distributions because the range of points that have positive probability is still a continuous range.

2.4 Distributions of Sample Data

Usually the data we work with is a sample from a population, not the complete population. Moreover our data is not usually a single observation drawn from a population but many observations. We designate the number of observations with n or N. This means we have n random variables, $X_1, \ldots, X_n$.

2.4.1 The Sampling Distribution

We are often are often interested in features of the total population, like the mean salary of employees, and we want to use our sample to *estimate* it. The logical way to do this is to take the mean of our sample. But we know it won't be exactly the same as the true population. In fact, for different samples we'd get different values of the sample mean. How can I be confident about what is the true mean? What if I was very unlucky and my $\bar{X}$ was very far from the truth?

As a thought experiment, I could think, what if I had done it another time and gotten another sample $X_1^*, \ldots, X_n^*$ (I use $*$ to indicate it's different)? I would get a different $\bar{X}^*$ and it wouldn't equal my previous $\bar{X}$. Nor would it equal the true mean of the population. I could ask myself, is it similar to $\bar{X}$? How much do they vary?

I could carry this further and do this many times: get a lot of samples of size n, each time take the mean. I would get a lot of $\bar{X}^*$ values. Let's call them

$$\bar{X}^{*(1)}, \ldots, \bar{X}^{*(B)}.$$

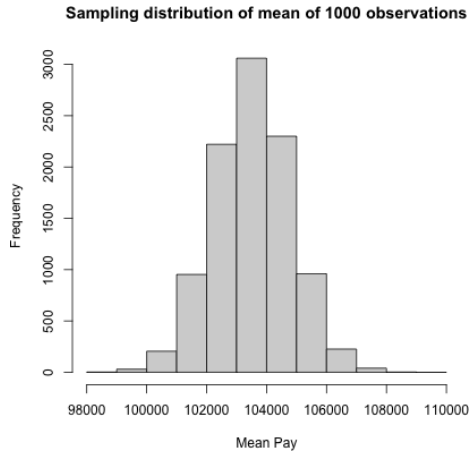
None of them would be exactly equal to the true population mean, but this would start to give me a good idea of how likely I am to get a "weird" value from a sample.

This describes the **sampling distribution** of $\bar{X}$, and is very important in statistics. It's how we think probabilistically about an estimate. We can make

probability statements from this distribution to answer questions about how likely it is that our estimate is far from the truth.

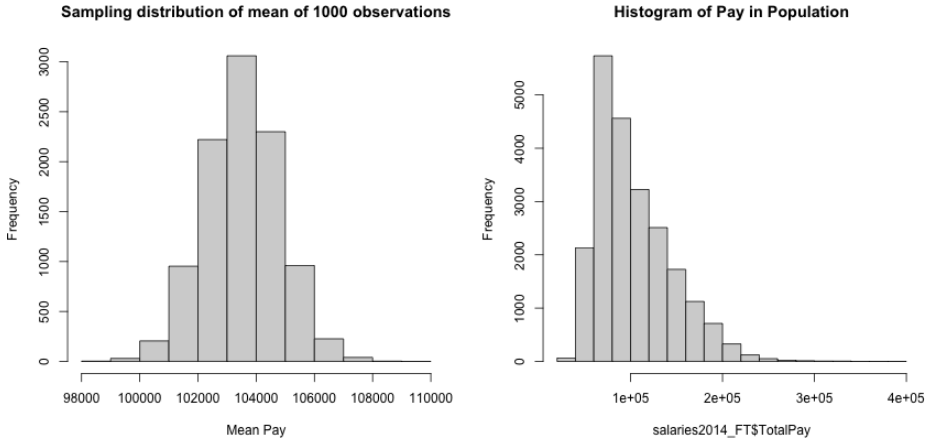
I can do this with the salary data, creating many samples of size 1,000 and calculating the mean of each sample. Here's a histogram of the $\bar{X}^{*(1)}, \ldots, \bar{X}^{*(B)}$ this creates

```
sampleSize <- 1000
sampleMean <- replicate(n = 10000, expr = mean(sample(salaries2014_FT$TotalPay,
    size = sampleSize, replace = TRUE)))
hist(sampleMean, xlab = "Mean Pay", main = paste("Sampling distribution of mean of",
    sampleSize, "observations"))
```



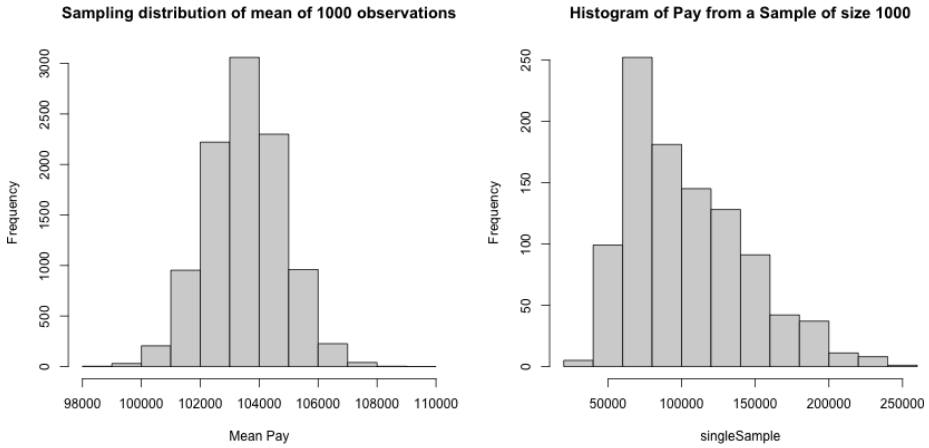
Note that the sampling distribution of the mean is very different from the histogram of the actual population of salary data:

```
par(mfrow = c(1, 2))
hist(sampleMean, xlab = "Mean Pay", main = paste("Sampling distribution of mean of",
    sampleSize, "observations"))
hist(salaries2014_FT$TotalPay, main = "Histogram of Pay in Population")
```



It is also different from the histogram of a single sample (which *does* look like the histogram of the population), so this is not due to a difference between the sample and the true population.

```
singleSample <- sample(salaries2014_FT$TotalPay, size = sampleSize,
    replace = TRUE)
par(mfrow = c(1, 2))
hist(sampleMean, xlab = "Mean Pay", main = paste("Sampling distribution of mean of",
    sampleSize, "observations"))
hist(singleSample, main = paste("Histogram of Pay from a Sample of size",
    sampleSize))
```



It is due to the fact that the sampling distribution of the mean is quite different from the distribution of the data that created these means. We will discuss this further, when we discuss the CLT below, but it is very important to keep distinct the distribution of the individual data points, and the distribution of summary statistics that are created from the data.

What Probability? Why is $\bar{X}$ random? It is random because the draws from the population are random, and this induces $\bar{X}$ to be random. So $\bar{X}$ is a random variable that has a distribution, as we have seen above in our simulation.

I emphasized above that we need a precise definition of random mechanisms to create probability statements. What does that look like for a sample of data? To make precise probability statements on $\bar{X}$, I need a description of how each of the $X_1, \ldots, X_n$ samples was chosen from the population. We'll discuss this more below, but you have already seen a **simple random sample** in previous classes, where each observation has equal probability of being selected. This is what we modelled with the salary data.

Moving Beyond the Thought Experiment The sampling distribution of an estimate is an important "thought experiment". But to be clear, we only get *one* sample $X_1, \ldots, X_n$ and one mean $\bar{X}$. *Our* sample data gave us $\bar{X}$, and we will never get to compare our $\bar{X}$ with these other potential $\bar{X}^*$ to see if the value we got was strange or unusual.

If we knew the true population distribution, we *could* say something about the sampling distribution of $\bar{X}$. (How to do that is for a higher probability class, but these are common questions answered in probability). But that kind of defeats the point – if we knew the true distribution, we wouldn't need to draw a sample from it.

However, sometimes probability theory can say something about the sampling distribution without knowing the population distribution. We need to make some assumptions, but they are usually much less than knowing the *entire* distribution.

What kind of assumptions? Well, it depends. Here's some examples:

- knowing the parametric distribution, but not the exact parameters. e.g. I assume each $X_i \sim Bin(n, p)$, but I don't know p.
- knowing how the data was collected, e.g. $X_1, \ldots, X_n$ are from a **Simple Random Sample**
- knowing that $X_1, \ldots, X_n$ are independently drawn from the same distribution, but not knowing the distribution they come from.

Obviously some of these assumptions require more knowledge than others, and how many assumptions are needed are an important consideration in learning data analysis tools.

2.4.2 Types of Samples

We said above that we need to know the random mechanism for how the samples were selected to be able to make statements about the sampling distribution of estimates created from the data.

A common description of how samples might be chosen from a population is a **simple random sample**. We previously described this for a *single* sample ($n = 1$) drawn from the population, and in that case it means each member of the population has equal probability of being selected. We won't describe the details of how to extend to n samples (you should have seen that in previous classes), but the basic idea is the sample is created by repeated draws from the same population.

Alternatively, we might not specify how the samples were drawn, but specify certain characteristics of the resulting random variables, $X_1, \dots, X_n$. The most common assumption is that they are **independent and identically distributed** (i.i.d). This means every X_i was drawn from the same distribution P, and that they were drawn independently from every other X_j. Note this means from our definitions above that we can say

$$P(X_1 = x_1, \dots, X_n = x_n) = P(X_1 = x_1)P(X_2 = x_2) \dots P(X_n = x_n)$$

SRS and i.i.d samples are the most common examples, and have very similar properties so that we can almost think of SRS as a special case of i.i.d samples. However, there are subtle issues that make a SRS not exactly a i.i.d sample. A SRS is the result of successive draws, meaning that you remove a member from the population once you draw it. This means the resulting data has a small amount of correlation between the data, but for large n the correlation becomes negligible.

Some datasets might be a sample of the population with no easy way to describe the process of how the sample was chosen from the population, for example data from volunteers or other **convenience samples** that use readily available data rather than randomly sampling from the population. Having convenience samples can make it quite fraught to try to make any conclusions about the population from the sample; generally we have to make assumptions about the data was collected, but because we did not control how the data is collected, we have no idea if the assumptions are true.

Examples of other types of samples Consider the following concrete example of different ways to collect data. Suppose that I want to compare the salaries of fire-fighters and teachers in all of California. To say this more precisely for data analysis, I want to see how similar is the distribution of salaries for fire-fighters to that of teachers in 2014 in California. Consider the following *samples* of data I might take

- All salaries in San Francisco (the data we have)
- A simple random sample drawn from a list of all employees in all localities in California.
- A separate simple random sample drawn from every county, combined together into a single dataset

Question: Why do I now consider all salaries in San Franscisco as a sample, when before I said it was a census?

All three of these are samples from the population of interest and for simplicity let's assume that we make them so that they all result in data with the same total sample size.

One is *not* a *random sample* (which one?). We can't reasonably make probability statements about data that is not a random sample from a population.

Only one is a *simple random sample* . The last sampling scheme, created by doing a SRS of each locality and combining the results, is a random sampling scheme, its just not a SRS. We know it's random because if we did it again, we wouldn't get exactly the same set of data (unlike our SF data). But it is not a SRS – it is called a **Stratified random sample**.

So are only SRS good random samples?

NO! The stratified random sample described above can actually be a much better way to get a random sample and give you *better* estimates – but you must correctly create your estimates to account for .

For the case of the mean, you have to estimate the population mean in such a way that it correctly estimates the distribution of population. How? The key thing is that because it is a random sample, drawn according to a *known probability mechanism*, it is possible to make a correct estimate of the population – but it won't be the simple mean of the sample.

How to make these kind of estimates for random samples that are not SRS is beyond the scope of this class, but there are standard ways to do so for stratified samples and many other sampling designs (this field of statistics is called *survey sampling*). Indeed most national surveys, particularly any that require face-to-face interviewing, are not SRS but much more complicated sampling schemes that can give equally accurate estimates, but often with less cost.

2.4.3 Normal Distribution and Central Limit Theorem

Let's go back to thinking about the sampling distribution. You've seen in previous classes an example of the kind of probability result we want, which tells you about the sampling distribution of an estimate – known as the **central limit theorem**. Let's review this important theorem.

The idea is that if you have i.i.d data, and the size of the sample (n) is large enough, the central limit theorem tells us that the distribution of $\bar{X}$ will be well approximated by a normal distribution. What's so important about this theorem is that it tells us that for large sample sizes this always happens – *regardless of the original distribution of the data.*

Specifically, the central limit theorem says that if you have i.i.d. sample of size n from an (unknown) distribution with mean μ_{true} and variance τ^2_{true}, then the distribution of $\bar{X}$ will be approximately

$$N(\mu_{true}, \frac{\tau^2_{true}}{n})$$

Many natural estimates we will encounter are actually means in one form or another. There are also many extensions of the CLT that give this same result in other settings too, for example a SRS (which is not i.i.d. data). This is the reason that the normal is a key distribution for statistics.

For $\bar{X}$, which is approximately normal, if the original population had mean μ and standard deviation τ, the standard deviation of that normal is $\tau/\sqrt{n}$.

Question: What does this mean for the chance of a single mean calculated from your data being far from the true mean (relate your answer to the above information about probabilities in a normal)?

Back to the Salary data

This means that our sample mean $\bar{X}$ from our salary data should start to follow the normal distribution. For most actual datasets, of course, we don't know the true mean of the population, but in this case, since we sampled from a known population, we do,

Question: What would be the parameters of this normal distribution?

```
mean(salaries2014_FT$TotalPay)
```

```
## [1] 103505.8
```

```
sqrt(var(salaries2014_FT$TotalPay)/sampleSize)
```

```
## [1] 1287.772
```

Recall that for a normal distribution, the probability of being within 1 standard deviation of μ is roughly 0.68 and the probability of being within 2 standard deviations of μ is roughly 0.95. So the CLT gives us an idea of what mean values are likely.

Question: What are the range of values this corresponds to in the salary data?

```
c(mean(salaries2014_FT$TotalPay) - 2 * sqrt(var(salaries2014_FT$TotalPay)/sampleSize),
    mean(salaries2014_FT$TotalPay) + 2 * sqrt(var(salaries2014_FT$TotalPay)/sampleSize))
```

```
## [1] 100930.2 106081.3
```

```
summary(salaries2014_FT$TotalPay)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   26364   72356   94272  103506  127856  390112
```

2.4.4 Density Histograms Revisited

A natural way to visualize the CLT on our salary data is to overlay the normal distribution on our histogram of our many samples of $\bar{X}$. Let's discuss briefly why this makes sense.

We've been showing histograms with the frequency of counts in each bin on the y-axis. But histograms are actually meant to represent the distribution of continuous measurements, i.e. to approximate density functions. In which case you want histogram to be drawn on the scale we expect for a density, called **density histograms.** This is done by requiring that the total area, when combined across all of the rectangles of the histogram, to have area 1. This means that the height of each rectangle for an interval $(b_1, b_2]$ is given by

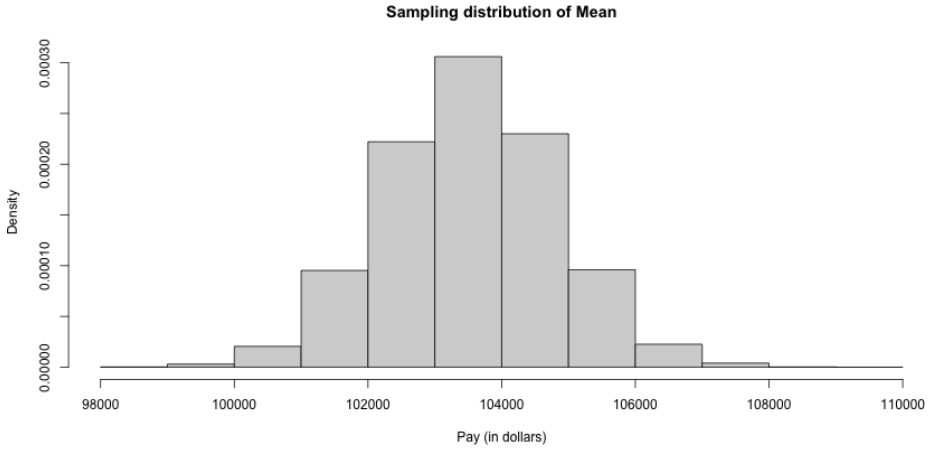
$$\frac{\#\text{obs. in } (b_1, b_2]}{(b_2 - b_1)n}$$

The area of each rectangle will then be

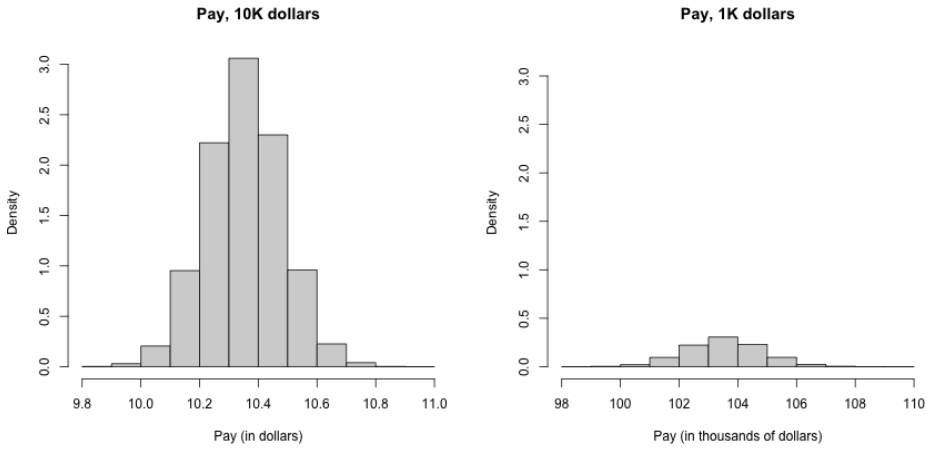
$$\frac{\#\text{obs. in } (b_1, b_2]}{n}$$

and so therefore they will sum up to 1. This matches our requirements for a density function as well, and results in the histogram being on the same scale as the density.

Notice that this area is also the proportion of observations in the interval $(b_1, b_2]$ and is our natural estimate of $P(b_1 \leq X \leq b_2)$ from our data. So they also match densities in that the area in each rectangle is an estimate of the probability.

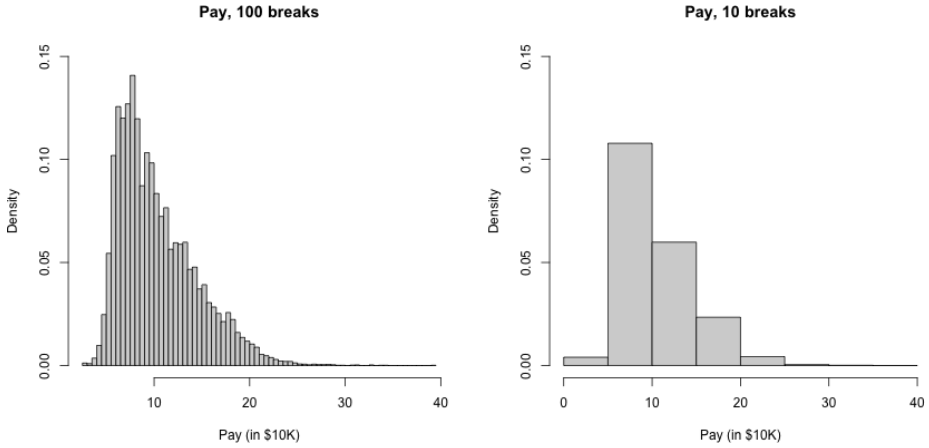


We can plot the density of pay in \$10K values or \$1K units instead.

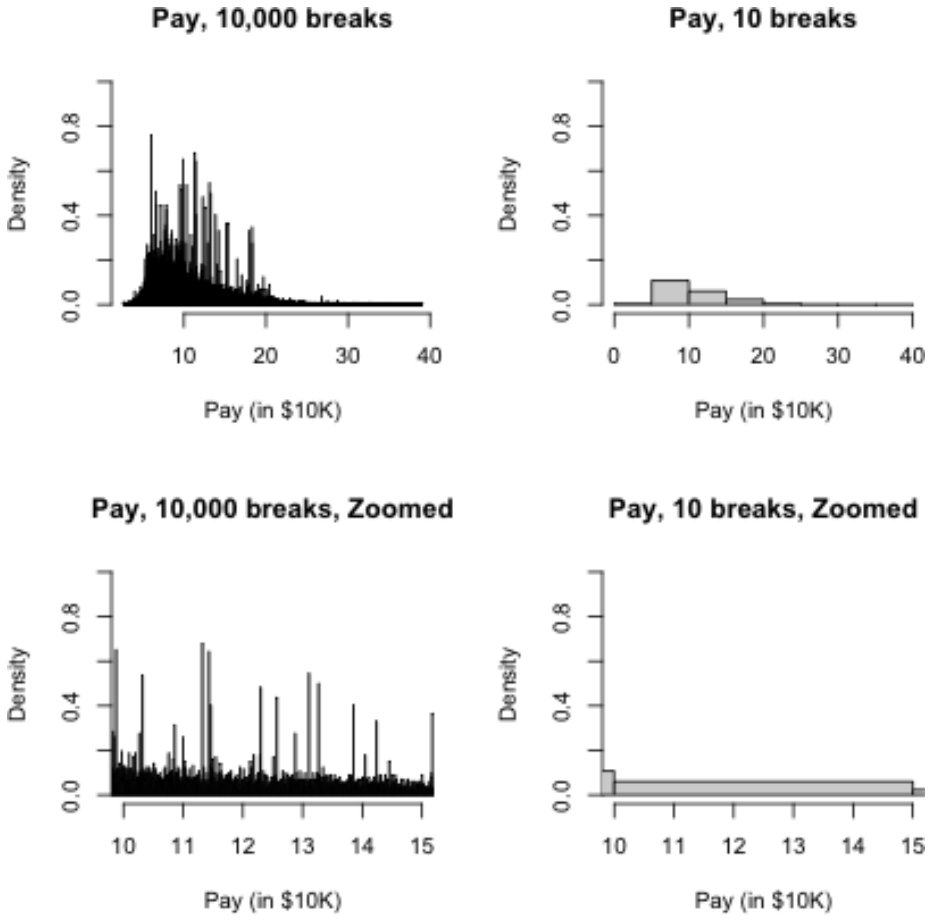


This demonstrates the effect of the scale of the data on this density histogram. Just like in our earlier discussion of density values, the width of our bins after dividing by 10,000 is a smaller number than if we divide by 1,000, so to get rectangles to have total area 1, we have to have larger values. And, if you plot histograms on the density scale, you can get values greater than 1, like densities.

Notice how density values stay on similar scales as you change the breaks.

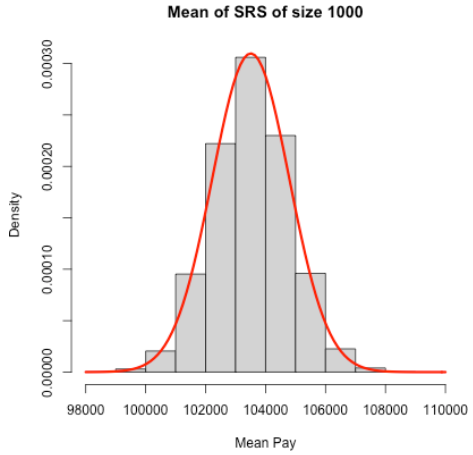


If I pick very small bins, I have the *appearance* of larger numbers, but when I zoom in, I can this is more due to a few bins being very large (and some have dropped to zero) but most of them are on the same scale.



Back to the CLT

Having thought about this, we now can return to the question of comparing our sampling distribution of $\bar{X}$ with the prediction given by the CLT. In other words, we can overlay the normal distribution, as predicted by the CLT, with the histogram of the actual sampling distribution and compare them. Notice to do this, we also have to pick the right mean and standard deviation for our normal distribution for these to align.

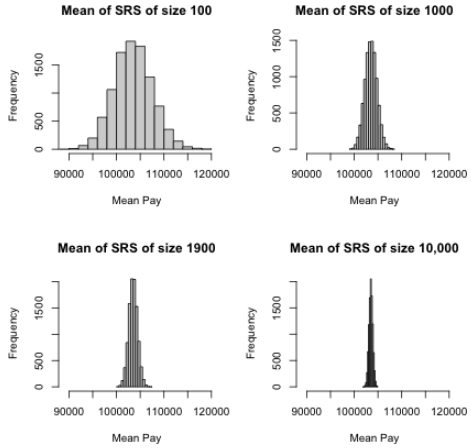


Notice how when I overlay the normal curve for discussing the central limit theorem, I had to set my `hist` function to `freq=FALSE` to get proper density histograms. Otherwise the histogram is on the wrong scale.

2.4.5 Improvement with larger n

We generally want to increase the sample size to be more accurate. What does this mean and why does this work? The mean $\bar{X}$ we observe in our data will be a random, single observation. If we could collect our data over and over again, we know that $\bar{X}$ will fluctuates around the truth for different samples. If we're lucky, τ is small, so that variability will be small, so any particular sample (like the one we get!) will be close to the mean. But we can't control τ. We can (perhaps) control the sample size, however – we can gather more data. The CLT tells us that if we have more observations, n, the fluctations of the mean $\bar{X}$ from the truth will be smaller and smaller for larger n – meaning the particular mean we observe in our data will be closer and closer to the true mean. So means with large sample size should be more accurate.

However, there's a catch, in the sense that the amount of improvement you get with larger n gets less and less for larger n. If you go from n observations to $2n$ observations, the standard deviation goes from $\frac{\tau_{true}}{\sqrt{n}}$ to $\frac{\tau_{true}}{\sqrt{2n}}$ – a decrease of $1\sqrt{2}$. In other words, the standard deviation decreases as n decreases like $1/\sqrt{n}$.



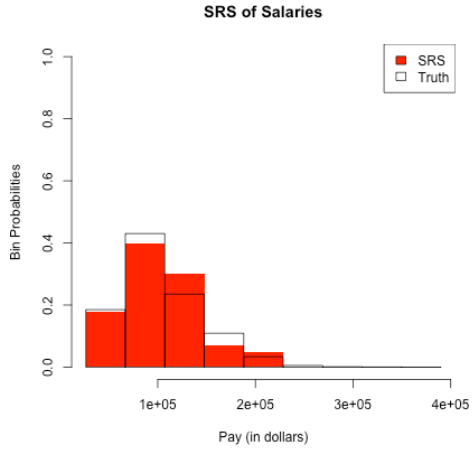
2.4.6 Visualizations as Estimates

To do exploratory analysis of our sample, we often use the same techniques we described above for the population, like histograms and boxplots. But working with a sample changes our interpretation of these plots. Consider what happens if you take a simple random sample of 100 employees from our complete set of full-time employees.

```
salariesSRS <- sample(x = salaries2014_FT$TotalPay,
    size = 100, replace = FALSE)
sample(1:5)
```

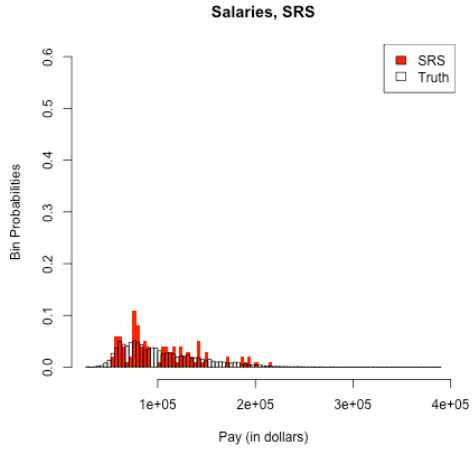
```
## [1] 5 3 2 1 4
```

Let's draw a plot giving the proportions of the total sample in each bin like I did in the previous section (remember – not a histogram!). I'm going to also draw the true population probabilities of being in each bin as well, and put it on the same histogram as the sample proportions. To make sure they are using the same bins, I'm going to define the break points manually (otherwise the specific breakpoints will depend on the range of each dataset and so be different)



Question: Suppose I had smaller width breakpoints (next figure), what conclusions would you make?

We can consider the above plots, but with more breaks:



When we are working with a sample of data, we should always think of probabilities obtained from a sample as an *estimate* of the probabilities of the full population distribution. This means histograms, boxplots, quantiles, and *any* estimate of a probability calculated from a sample of the full population have variability, like any other estimate.

This means we need to be careful about the dual use of histograms as both visualization tools and estimates. As visualization tools, they are always appropriate for understanding *the data you have*: whether it is skewed, whether there are outlying or strange points, what are the range of values you observe, etc.

To draw broader conclusions from histograms or boxplots performed on a sample, however, is by definition to view them as estimates of the entire population. In this case you need to think carefully about how the data was collected.

2.4.6.1 Thinking about Histograms as Estimates

If we draw histograms from samples they will all describe the observed distribution of *the sample we have*, but they will not all be good estimates of the underlying population distribution depending on what was the probability mechanism of how the sample was created. Recall we proposed three types of samples from SF salary data.

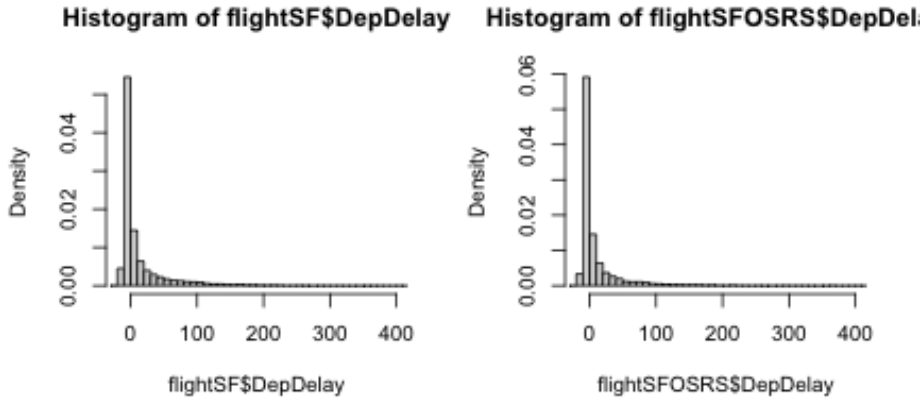
- All salaries in San Franscisco (the data we have)
- A simple random sample drawn from a list of all employees in all localities in California.
- A separate simple random samples drawn from every locality, combined together into a single dataset

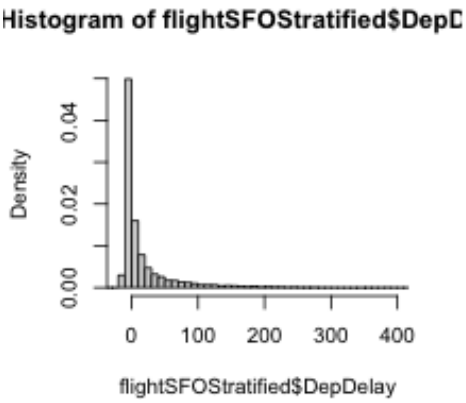
We don't have each of these three types of samples from SF salary data, but we do have the full year of flight data in 2015/2016 academic year (previously we analyzed only the month of January). Consider the following ways of sampling from the full set of flight data and consider how they correspond to the above:

- 12 separate simple random samples drawn from every month in the 2015/2016 academic year, combined together into a single dataset
- All flights in January
- A simple random sample drawn from all flights in the 2015/2016 academic year.

We can actually make all of these samples and compare them to the truth (I've made these samples previously and I'm going to just read them, because the entire year is a big dataset to work with in class).

```
flightSFOSRS <- read.table(file.path(dataDir, "SFO_SRS.txt"),
    sep = "\t", header = TRUE, stringsAsFactors = FALSE)
flightSFOStratified <- read.table(file.path(dataDir,
    "SFO_Stratified.txt"), sep = "\t", header = TRUE,
    stringsAsFactors = FALSE)
par(mfrow = c(2, 2))
xlim <- c(-20, 400)
hist(flightSF$DepDelay, breaks = 100, xlim = xlim,
    freq = FALSE)
hist(flightSFOSRS$DepDelay, breaks = 100, xlim = xlim,
    freq = FALSE)
hist(flightSFOStratified$DepDelay, breaks = 100, xlim = xlim,
    freq = FALSE)
```





Question: How do these histograms compare?

In particular, drawing histograms or estimating probabilities from data as we have done here only give good estimates of the population distribution *if the data is a SRS*. Otherwise they can vary quite dramatically from the actual population.

2.5 Density Curve Estimation

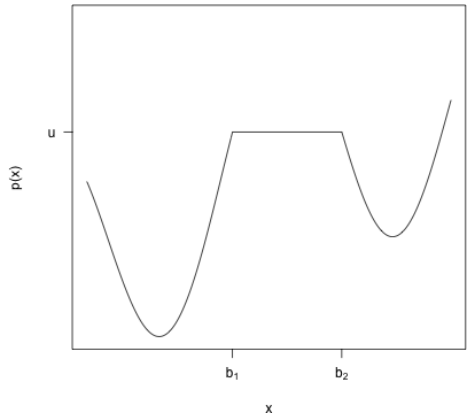
We've seen that histograms can approximate density curves (assuming we make the area in the histogram sum to 1). More formally, we can consider that if we observe continuous valued data from an unknown distribution, we would like an estimate of the unknown pdf $p(x)$ for the distribution that created the data. This is a problem known as **density curve estimatation**. This is moving

far beyond estimating a single value, like the mean, and trying to estimate the entire pdf.

The setup we consider is that we assume we have a sample of $i.i.d$ data $X_1, \ldots, X_n$ from an unknown distribution with density $p(x)$. We want to create a function $\widehat{p}(x)$ (based on the data) that is an estimate of $p(x)$. As we'll see, a density histogram is one such simple estimate of $p(x)$, but we will also discuss other estimates that are better than a histogram.

2.5.1 Histogram as estimate of a density

To understand why a histogram is an estimate of a density, let's think of an easy situation. Suppose that we want to estimate $p(x)$ between the values b_1, b_2, and that in that region, we happen to know that $p(x)$ is constant, i.e. a flat line.



If we actually knew $p(x)$ we could find $P(b_1 \leq X \leq b_2)$ as the area under $p(x)$. Since $p(x)$ is a flat line in this region, this is just

$$P(X \in [b_1, b_2]) = u * (b_2 - b_1)$$

where $u = p(b_1) - p(b_2)$. To estimate $p(x)$ in this region is to estimate u. So in this very simple case, we have a obvious way to estimate $p(x)$ if $x \in [b_1, b_2]$: first estimate $P(b_1 \leq X \leq b_2)$ and then let

$$\widehat{u} = \widehat{p}(x) = \frac{\widehat{P}(b_1 \leq X \leq b_2)}{b_2 - b_1}$$

We have already discussed above one way to estimate $P(b_1 \leq X \leq b_2)$ if we have i.i.d data: count up the data in that interval, and divide by total number of data points n,

$$\widehat{P}(b_1 \leq X \leq b_2) = \frac{\#\text{ Points in } [b_1, b_2]}{n}$$

Using this, a good estimate of $p(x)$ (if it is a flat function in that area) is going to be:

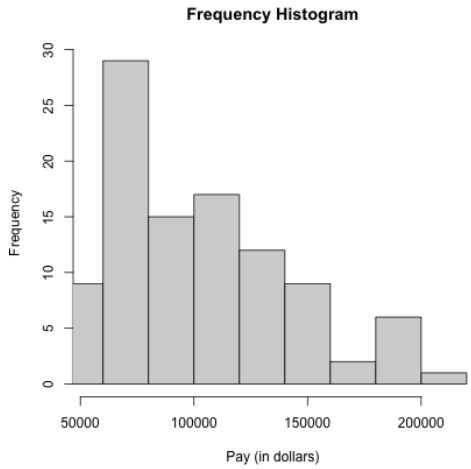
$$\hat{p}(x) = \hat{P}(b_1 \leq X \leq b_2)/(b_2 - b_1) = \frac{\#\text{ Points in } [b_1, b_2]}{(b_2 - b_1) \times n}$$

Relationship to Density Histograms

In fact, this is a pretty familiar calculation, because it's also exactly what we calculate for a density histogram.

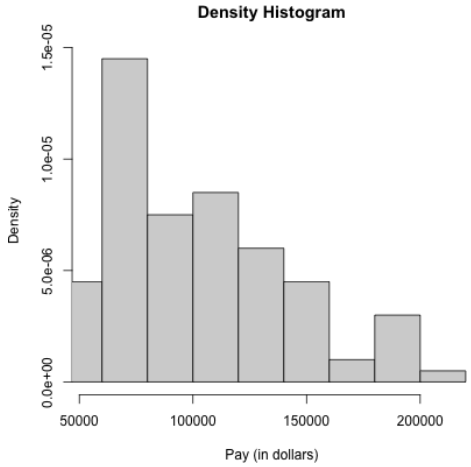
However, we don't expect the true $p(x)$ to be a flat line, so why do we use this density histograms when we know this isn't true? If the pdf $p(x)$ is a pretty smooth function of x, then in a *small enough* window around a point x, $p(x)$ is going to be not changing too in the scheme of things. In other words, it will be roughly the same value in a small interval–i.e. flat. So if x is in an interval $[b_1, b_2]$ with width w, and the width of the interval is small, we can more generally say a reasonable estimate of $p(x)$ would be the same as above.

With this idea, we can view our (density) histogram as a estimate of the pdf. For example, suppose we consider a frequency histogram of our SRS of salaries,



We showed this histogram as is commonly done using frequencies. But a density histogram will divide by the width of the interval of the bins (this is what is meant by the density values in a histogram), i.e. each bin defines an interval (b_1, b_2), and the density histogram value is

$$\frac{\#\text{ Points in } [b_1, b_2]}{(b_2 - b_1) \times n}$$



Question: Suppose we want to calculate $\hat{p}_{hist}(60K)$, and we've set up our breaks of our histogram so that $x = 60K$ is in the bin with interval $[50K, 70K)$. How do you calculate $\hat{p}_{hist}(60K)$ from a sample of size 100?

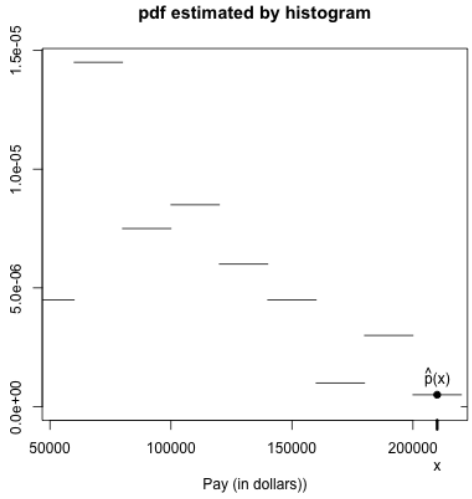
This is exactly the same as our estimate of $p(x)$ above in the special case when $p(x)$ is a flat line in (b_1, b_2). Thus, if our true $p(x)$ is not changing much in the interval $(b_1, b_2]$, then the density histogram value is an estimate of $p(x)$ in (b_1, b_2).

Thus, the density histogram is a function that estimates $p(x)$. We can call it $\hat{p}_{hist}(x)$ to denote that it is a histogram estimate of the density.

We of course need to do this for a lot of intervals to cover the range of x. This gives us, for every x, an estimated value of $\hat{p}_{hist}(x)$, based on what interval x is in:

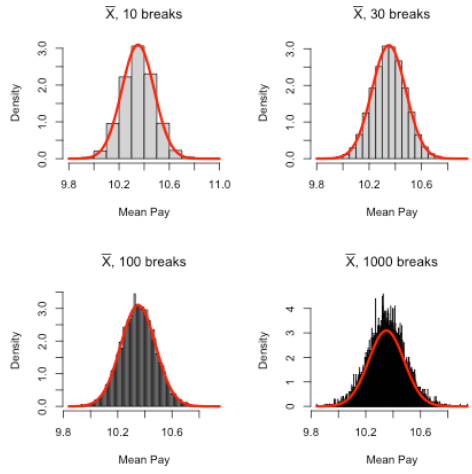
$$\hat{p}_{hist}(x) = \frac{\hat{P}(\text{data in bin of } x\ )}{w}$$

$\hat{p}_{hist}(x)$ is a function that is what is called a *step function* and as a function we can visualize it as:



Sensitivity to breaks

How we choose the breaks in a histogram can affect their ability to be a *good* estimate of the density. Consider our sample of $\bar{X}$ values from the previous section when we were discussing the central limit theorem. We know roughly many samples of $\bar{X}$ should look like it comes from a normal distribution. Below we draw histograms based our repeated values of $\bar{X}$ for different breaks. We also overlay the normal distribution which represents the distribution the mean should follow.



We can see that the accuracy of $\hat{p}_{hist}(x)$ as an approximation of the values of the pdf of a normal varies a great deal with the number of breaks (or equivalently the width of the intervals).

2.5.2 Kernel density estimation

A step function as an estimate of $p(x)$ does not seem to make sense if we think the true $p(x)$ a continuous function. We will now consider the most commonly used method of estimating the density, kernel density smoothing, for estimate $p(x)$. To understand how this works, we will go through several "improvements" to the histogram, each one of which improves on our estimate. The kernel density smoothing will put all of these ideas together.

2.5.2.1 Moving Windows

Let's consider a simple improvement first: using a moving window or bin to calculate a histogram, rather than a fixed set of non-overlaping intervals.

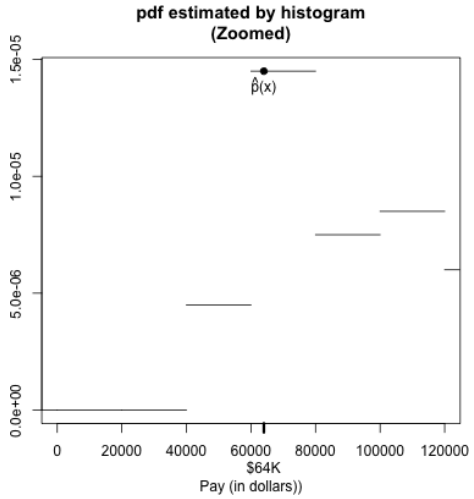
Motivation

Previously, we said if the pdf $p(x)$ is a pretty smooth function of x, then in a *small enough* window around a point x, $p(x)$ is going to be not changing too much, so it makes sense to assume it's approximately a flat line. So if you want to estimate $p(x)$ at a specific x, say $x = 64,000$, we would make an interval (b_1, b_2) with $64,000$ at the center and calculate our estimate, say $(b_1, b_2) = (54,000, 74,000)$

$$\widehat{p}(64,000) = \frac{\#\text{ Points in } (b_1, b_2]}{(b_2 - b_1) \times n} = \frac{\#\text{ Points in } (54K, 74K]}{20K \times n}.$$

However, when we make a histogram, we set a fix intervals of the bins, irrelevant of where $64,000$ lies. In our histogram above, our bins were every $20K$ starting with zero. So our estimate of $p(64,000)$ is

$$\widehat{p}_{hist}(64,000) = \frac{\#\text{ Points in } (60K, 80K]}{20K \times n}.$$

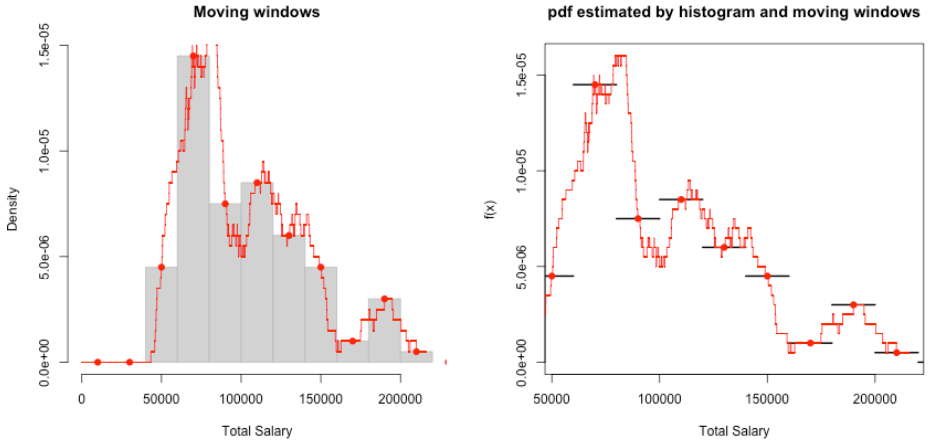


While this makes sense for our plot, this is strange if our goal is to estimate $p(64,000)$. We would do better to use the first interval we considered above of $(54,000, 74,000]$

This is the example just a single specific value of $x = 64,000$. But in estimating the function $p(x)$, we are really wanting to estimate $p(x)$ *for every* x. So by the same analogy, I should estimate a $\widehat{p}(x)$ by making a bin centered at x, *for every* x. I.e. for every value of x, we make an interval of $\pm 20,000$ and use the same formula.

Question: For example, for $x = 80,000$, how would you estimate $p(80,000)$?

Doing this for *every single* x would give us a curve like this (as compared to the density histogram):



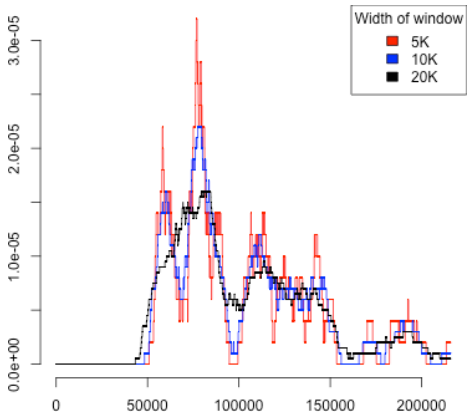
More formally, for chosen bin width w our estimate of $p(x)$, is

$$\hat{p}(x) = \frac{\#X_i \in [x - \frac{w}{2}, x + \frac{w}{2})}{w \times n}$$

Of course, in our plots, we don't actually calculate for *every* x, but take a large number of x values that span the range.

Window size

We had a fixed size of $20K$ on either size of x, but we can consider using different size windows around our point x:



Question: What is the effect of larger windows or bins?

2.5.2.2 Weighted Kernel Function

Now we consider further improving our estimate of $p(x)$.

Re-writing our estimate

We said our estimate of $p(x)$, is

$$\hat{p}(x) = \frac{\#X_i \in [x - \frac{w}{2}, x + \frac{w}{2})}{w \times n},$$

where w is the width of our are interval.

To estimate the density around x, this estimate counts individual data observations if and only if they within $w/2$ to x. We could write this as a sum over all of our data in our SRS, where some of the data are not counted depending on whether it is close enough to x or not.

To do that mathematically, we're going to create a function that tells os for each observation whether it is within $w/2$ of x. Let

$$I(X_i \in [x - \frac{w}{2}, x + \frac{w}{2})) = \begin{cases} 1 & X_i \in [x - \frac{w}{2}, x + \frac{w}{2})) \\ 0 & otherwise \end{cases}$$

This is called an **indicator** function. Generally and indicator function $I(\cdot)$ is a function which is 1 if the value inside it is true and zero otherwise. If we want to count how many times something happens, we can write it as a sum of indicator functions.

Then we can write our estimate as

$$\hat{p}(x) = \frac{1}{w \times n} \sum_{i=1}^{n} I(X_i \in [x - \frac{w}{2}, x + \frac{w}{2}))$$

Since we only get 1 for the observations that are in the interval, this sum is the same as

$$\#X_i \in [x - \frac{w}{2}, x + \frac{w}{2})$$

If we rearrange this a bit, we have

$$\hat{p}(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{w} I(X_i \in [x - \frac{w}{2}, x + \frac{w}{2})),$$

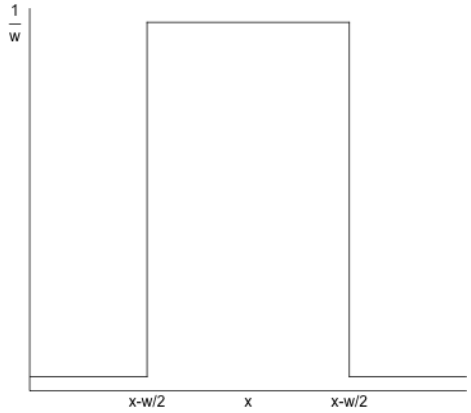
and we see that in this way that we are starting to get an estimate that looks more like quantities like the sample mean or sample variance, i.e.

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} X_i, \quad var = \frac{1}{n} \sum_{i=1}^{n} (X_i - \bar{X})^2$$

where we are taking a function of all our observations and then taking an average over these values.

Interpreting our estimate

Using this expression, we can see that every observation is contributing, in principle, to our estimate. We can thus interpret how much a point X_i counts toward estimating $p(x)$: it either contributes $1/w$ or 0 depending on how far it is from x. We can visualize this:



We can think of this as a function f with input variables x *and* X_i: for every x for which we want to estimate $p(x)$, we have a function that tells us how much each of our data points X_i should contribute.

$$f(x, X_i) = \begin{cases} \frac{1}{w} & X_i \in [x - \frac{w}{2}, x + \frac{w}{2}) \\ 0 & otherwise \end{cases}$$

It's a function that is different for every x, but just like our moving windows, it's the same function and we just apply it across all of the x. So we can simply write our estimate at each x as an average of the values $f(x, X_i)$

$$\hat{p}(x) = \frac{1}{n} \sum_{i=1}^{n} f(x, X_i)$$

Is this a proper density?

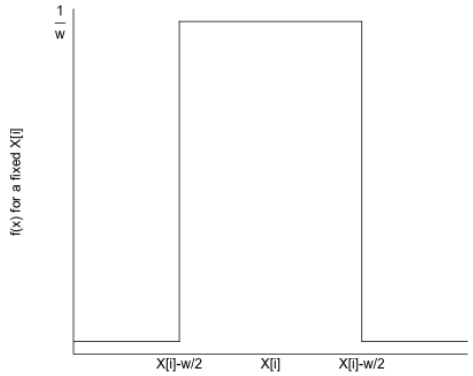
Does $\hat{p}(x)$ form a proper density, i.e. is the area under its curve equal 1? We can answer this question by integrating $\hat{p}(x)$,

$$\begin{aligned} \int_{-\infty}^{\infty} \hat{p}(x)dx &= \int_{-\infty}^{\infty} \frac{1}{n} \sum_{i=1}^{n} f(x, X_i)dx \\ &= \frac{1}{n} \sum_{i=1}^{n} \int_{-\infty}^{\infty} f(x, X_i)dx \end{aligned}$$

So if $\int_{-\infty}^{\infty} f(x, X_i)dx = 1$ for any X_i, we will have,

$$\int_{-\infty}^{\infty} \hat{p}(x)dx = \frac{1}{n} \sum_{i=1}^{n} 1 = 1.$$

Is this the case? Well, considering $f(x, X_i)$ as a function of x with a fixed X_i value, it is equal to $1/w$ when x is within $w/2$ of X_i, and zero otherwise (i.e. the same function as before, but now centered at X_i) which we can visualize below:



This means $\int_{-\infty}^{\infty} f(x, X_i)dx = 1$ for any fixed X_i, and so it is a valid density function.

Writing in terms of a kernel function K

For various reasons, we will often speak in terms of the distance between x and the X_i relative to our the width on one side of x, given by h:

$$\frac{|x - X_i|}{h}$$

The parameter h is called the **bandwidth** parameter.

You can think of this as the amount of h units X_i is from x. So if we are trying to estimate $p(64,000)$ and our bin width is $w = 5,000$, then $h = 2,500$ and $\frac{|x-X_i|}{h}$ is the number of $2.5K$ units a data point X_i is from $64,000$.}

Doing this we can write

$$f_x(X_i) = \frac{1}{h}K(\frac{|x - X_i|}{h})$$

where

$$K(d) = \begin{cases} \frac{1}{2} & d \leq 1 \\ 0 & otherwise \end{cases}$$

We call a function $K(d)$ that defines a weight for each data point at h-units distance d from x a **kernel function**.

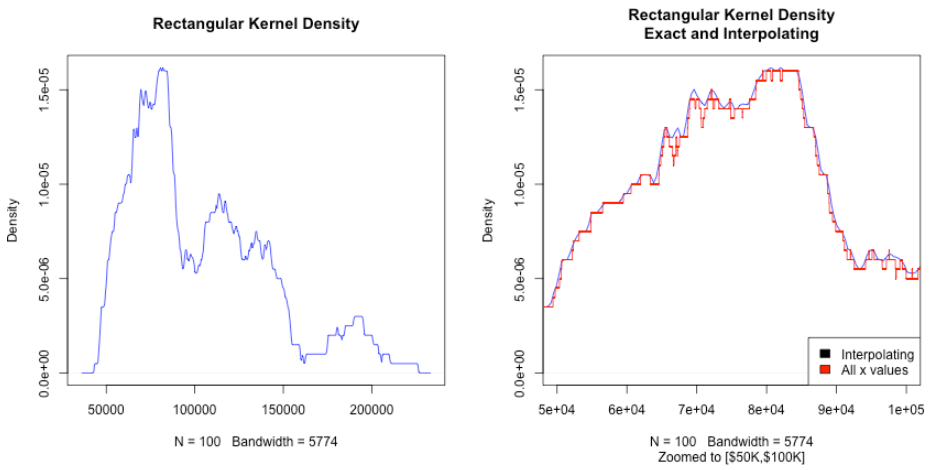
$$\widehat{p}(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h}K(\frac{|x - X_i|}{h})$$

All of this mucking about with the function K versus $f(x, X_i)$ is not really important – it gives us the same estimate! K is just slightly easier to discuss

mathematically because we took away it's dependence on x, X_i and (somewhat) h.

Example of Salary data

In R, the standard function to calculate the density is `density`. Our moving window is called the "rectangular" kernel, and so we can replicate what we did using the option `kernel="rectangular"` in the `density` function[15]

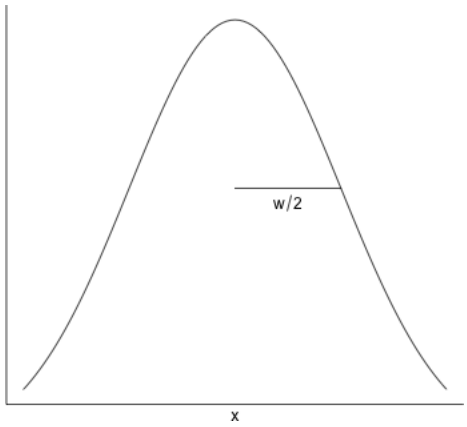


2.5.2.3 Other choices of kernel functions

Once we think about our estimate of $p(x)$ as picking a weight for neighboring points, we can think about not having such a sharp distinction for the interval around x. After all, what if you have a data point that is $5,100$ away from x rather than $5,000$? Similarly, if you have 50 data points within 100 of x shouldn't they be more informative about the density around x than 50 data points more than $4,500$ away from x?

This generates the idea of letting data points contribute to the estimate of $p(x)$ based on their distance from x, but in a smoother way. For example, consider this more 'gentle' visualization of the contribution or weight of a data point X_i to the estimate of the density at x:

[15] It's actually hard to exactly replicate what I did above with the `density` function, because R is smarter. First of all, it picks a bandwidth from the data. Second, it doesn't evaluate at every possible x like I did. It picks a number, and interpolates between them. For the rectangular density, this makes much more sense, as you can see in the above plot.



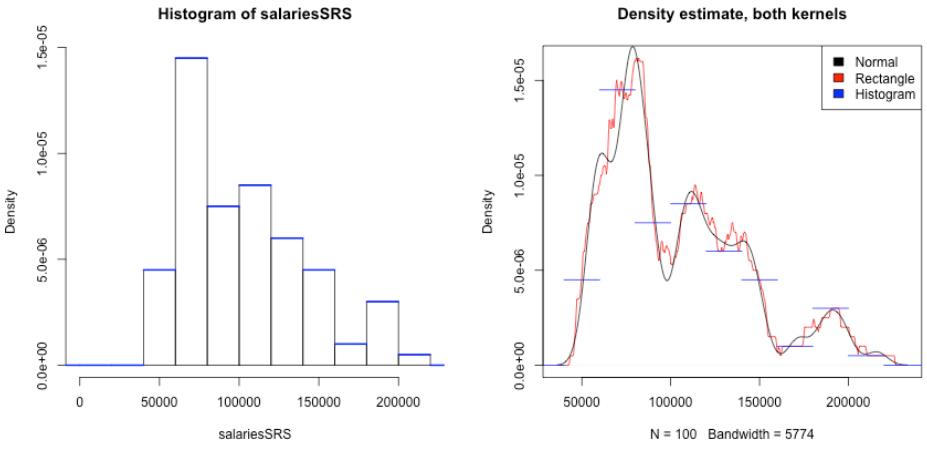
This is also the form of a kernel function, called a normal (or gaussian) kernel and is very common for density estimation. It is a normal curve centered at x[16]; as you move away from x you start to decrease in your contribution to the estimate of $p(x)$ but more gradually than the rectangle kernel we started with.

If we want to formally write this in terms of a function K, like above then we would say that our $K(\cdot)$ function is the standard normal curve centered at zero with standard deviation 0. This would imply that

$$\frac{1}{h}K(\frac{|x - X_i|}{h})$$

will give you the normal curve with mean x and standard deviation h.

We can compare these two kernel estimates. The next plot is the estimate of the density based on the rectangular kernel and the normal kernel (now using the defaults in `density`), along with our estimate from the histogram:



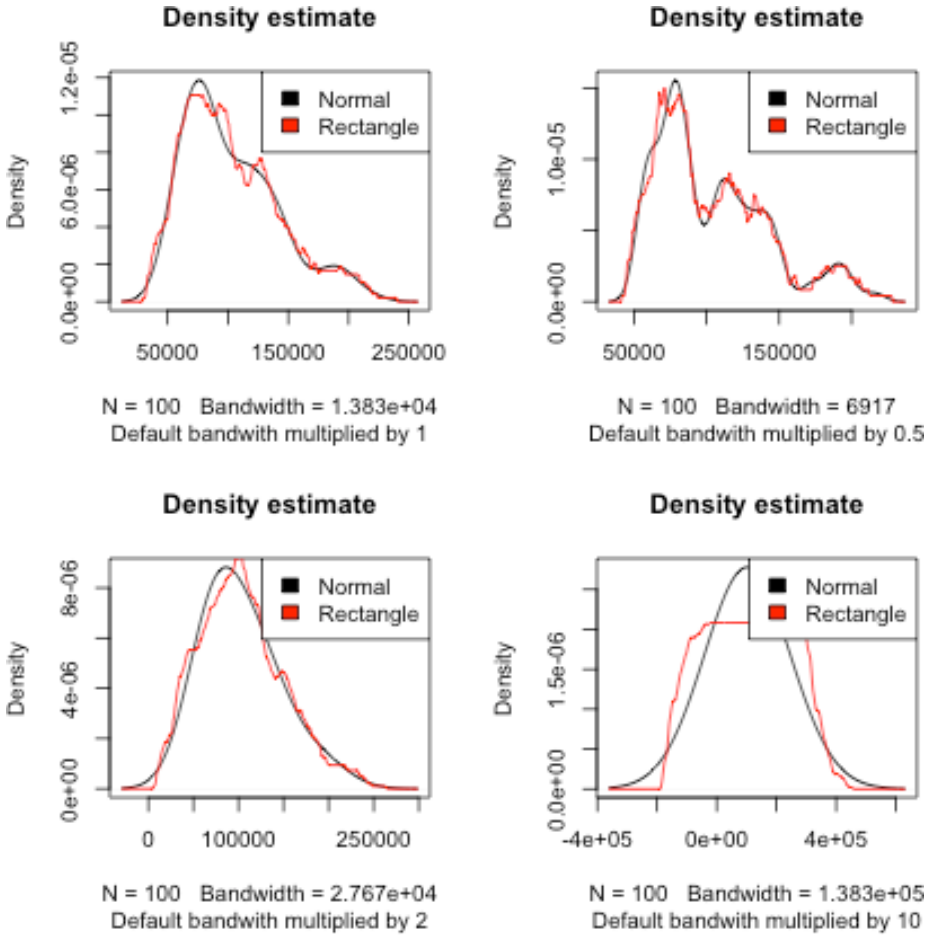
[16]You have to properly scale the height of the kernel function curve so that you get area under the final estimate $\hat{p}(x)$ curve equal to 1

Question: What do you notice when comparing the estimates of the density from these two kernels?

Bandwidth

Notice that I still have a problem of picking a width for the rectangular kernel, or the spread/standard deviation for the gaussian kernel. This w value is called generically a **bandwidth** parameter. In the above plot I forced the functions to have the same bandwidth corresponding to the moving window of \$20K.

Here are plots of the estimates using different choices of the bandwidth:



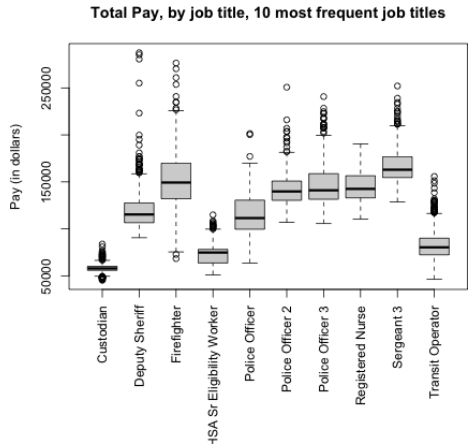
The default parameter of the `density` function is usually pretty reasonable, particularly if used with the gaussian kernel (also the default). Indeed, while we discussed the rectangular kernel to motivate going from the histogram to

the kernel density estimator, it's rarely used in practice. It is almost always the gaussian kernel.

2.5.3 Comparing multiple groups with density curves

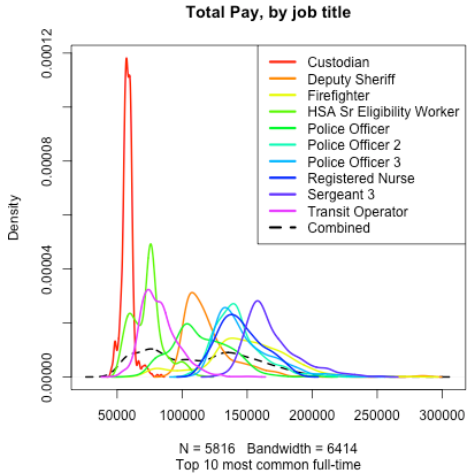
In addition to being a more satisfying estimation of a pdf, density curves are much easier to compare between groups than histograms because you can easily overlay them.

Previously we considered dividing the SF salary data into different groups based on their job title and comparing them. Because of the large number of job titles, we earlier created a smaller dataset `salaries2014_top` with just the top 10 job titles (by frequency) which we will use again here. Here is the boxplot we created previously.



The boxplots allow us to compare some basic summary statistics of the distributions visually. We could ask if there were more subtle differences by estimating the density of each group and comparing them. I've defined a small function `perGroupDensity` to do this (not shown, see the accompanying code for the book for the function) and I will plot each density with a different color:

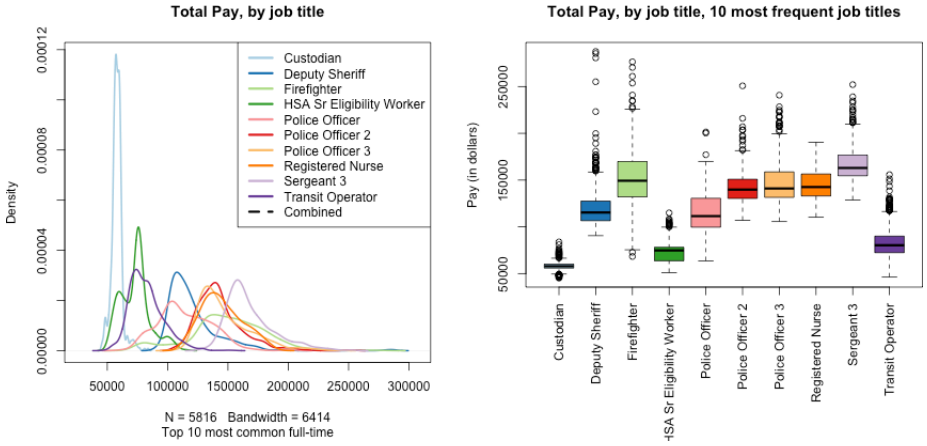
```
par(mfrow = c(1, 1))
output <- perGroupDensity(x = salaries2014_top$TotalPay,
    salaries2014_top$JobTitle, main = "Total Pay, by job title",
    sub = "Top 10 most common full-time")
```



A note on colors Before we talk about what we can see, first of all notice that the default colors were not perhaps the best choice of colors. We can't actually tell apart the various greens, for example. These kinds of details are really important in thinking about visualizations. Sometimes you want similar colors. For example, we might want all of the police categories with a similar shade. Or we might be dividing based on a continuous variable, like age, and it would make sense to have the colors for the ages follow a spectrum (though it's still a problem if you *can't* tell them apart). But generally for distinct categories we want some distinct colors that we can easily tell apart.

So now I'm going to define some colors and replot the densities for the multiple groups (I'm not going to plot the combined density to simplify the plot). I will plot the boxplot next to it so we can compare.

```
nGroups <- nlevels(factor(salaries2014_top$JobTitle))
library(RColorBrewer)
cols <- brewer.pal(n = nGroups, "Paired")
par(mfrow = c(1, 2))
output <- perGroupDensity(x = salaries2014_top$TotalPay,
    salaries2014_top$JobTitle, cols = cols, main = "Total Pay, by job title",
    sub = "Top 10 most common full-time", includeCombined = FALSE)
par(mar = c(10, 4.1, 4.1, 0.1))
boxplot(salaries2014_top$TotalPay ~ salaries2014_top$JobTitle,
    main = "Total Pay, by job title, 10 most frequent job titles",
    xlab = "", col = cols, ylab = "Pay (in dollars)",
    las = 3)
```



Compared to the boxplot, we can see that "HSA Sr Eligibility Worker" seems to be *bimodal* (two peaks in the density). This suggests there are two groups in this category with different salary rates. One of those modes/groups overlaps with the lower wages of the "Custodian" category while the other mode is higher.

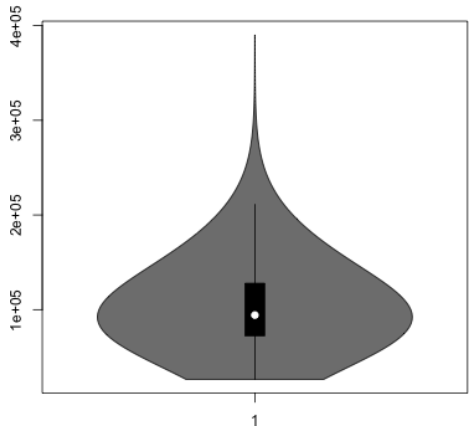
We can see that with density plots we can see more subtle differences between the groups, but it is a much noisier plot. It's easier to see the big shifts between the job titles with a boxplot. Whether boxplots or multiple density plots is better depends a lot on the data and what your question is. It also depends on how many groups you are comparing.

What are we estimating? We discussed density estimation as estimating the density $p(x)$ of an unknown distribution. In the case of the SF salaries, as we've discussed, the data is an entire census of the population, so there's nothing to estimate. This is a rare situation, since normally your data will *not* be a census. Furthermore, we've already discussed that histograms can be used as either a visualization of the existing data or an estimate of the unknown generating distribution. The same is true for kernel density estimates. So in this case it's just a visualization tool for comparing the groups, and not an estimate of an unknown distribution.

2.5.3.1 Violin Plots

We can combine the idea of density plots and boxplots to get something called a "violin plot".

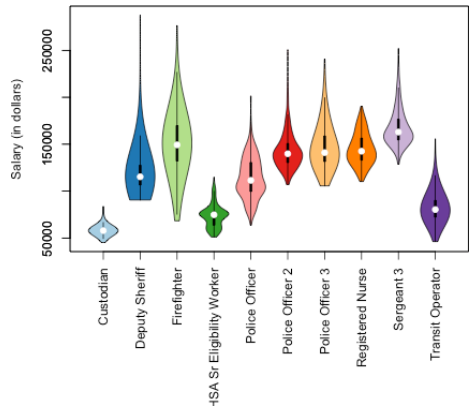
```
library(vioplot)
vioplot(salaries2014_FT$TotalPay)
```



This is basically just turning the density estimate on its side and putting it next to the boxplot so that you can get finer-grain information about the distribution.

Like boxplots, this allows you to compare many groups.

```
source("http://www.stat.berkeley.edu/~epurdom/RcodeForClasses/myvioplot.R")
par(mar = c(10, 4.1, 4.1, 0.1))
vioplot2(salaries2014_top$TotalPay, salaries2014_top$JobTitle,
    col = cols, las = 3, ylab = "Salary (in dollars)")
```




```
## Linking to ImageMagick 6.9.12.3
## Enabled features: cairo, fontconfig, freetype, heic, lcms, pango, raw, rsvg, webp
## Disabled features: fftw, ghostscript, x11
```

Chapter 3

Comparing Groups and Hypothesis Testing

We've mainly discussed informally comparing the distribution of data in different groups. Now we want to explore tools about how to use statistics to make this more formal. Specifically, how can we quantify whether the differences we see are due to natural variability or something deeper? We will do this through hypothesis testing.

In addition to reviewing specific hypothesis tests, we have the following goals:

- Abstract the ideas of hypothesis testing: in particular what it means to be "valid" and what makes a good procedure
- Dig a little deeper as to what assumptions we are making in using a particular test
- Learn about two paradigms for hypothesis testing:
  - parametric methods
  - resampling methods

Depending on whether you took STAT 20 or Data 8, you may be more familiar with one of these paradigms than the other.

We will first consider the setting of comparing two groups, and then expand out to comparing multiple groups.

3.1 Choosing a Statistic

Example of Comparing Groups

Recall the airline data, with different airline carriers. We could ask the question about whether the distribution of flight delays is different between carriers.

Question: If we wanted to ask whether United was more likely to have delayed flights than American Airlines, how might we quantify this?

The following code subsets to just United (`UA`) and American Airlines (`AA`) and takes the mean of `DepDelay` (the delay in departures per flight)

```
flightSubset <- flightSFOSRS[flightSFOSRS$Carrier %in%
    c("UA", "AA"), ]
mean(flightSubset$DepDelay)
```

```
## [1] NA
```

Question: What do you notice happens in the above code when I take the mean of all our observations?

Instead we need to be careful to use `na.rm=TRUE` if we want to ignore NA values (which may not be wise if you recall from Chapter 2, NA refers to cancelled flights!)

```
mean(flightSubset$DepDelay, na.rm = TRUE)
```

```
## [1] 11.13185
```

We can use a useful function `tapply` that will do calculations by groups. We demonstrate this function below where the variable `Carrier` (the airline) is a factor variable that defines the groups we want to divide the data into before taking the mean (or some other function of the data):

```
tapply(X = flightSubset$DepDelay, flightSubset$Carrier,
    mean)
```

```
## AA UA
## NA NA
```

Again, we have a problem of NA values, but we can pass argument `na.rm=TRUE` to `mean`:

```
tapply(flightSubset$DepDelay, flightSubset$Carrier,
    mean, na.rm = TRUE)
```

```
##        AA        UA
##  7.728294 12.255649
```

We can also write our own functions. Here I calculate the percentage of flights delayed or cancelled:

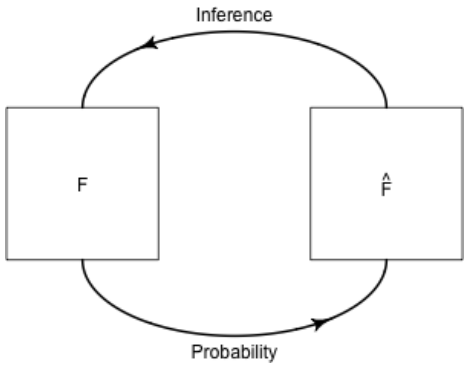
```
tapply(flightSubset$DepDelay, flightSubset$Carrier,
    function(x) {
        sum(x > 0 | is.na(x))/length(x)
    })
```

```
##         AA         UA
## 0.3201220 0.4383791
```

These are **statistics** that we can calculate from the data. A statistic is *any* function of the input data sample.

3.2 Hypothesis Testing

Once we've decided on a statistic, we want to ask whether this is a meaningful difference between our groups. Specifically, with different data samples, the statistic would change. **Inference** is the process of using statistical tools to evaluate whether the statistic observed indicates some kind of actual difference, or whether we could see such a value due to random chance even if there was no difference.



Therefore, to use the tools of statistics – to say something about the generating process – we must be able to define a random process that we imagine created the data.

Hypothesis testing encapsulate these inferential ideas. Recall the main components of hypothesis testing:

1. Hypothesis testing sets up a **null hypothesis** which describes a feature of the population data that we want to test – for example, are the medians of the two populations the same?

2. In order to assess this question, we need to know what would be the distribution of our sample statistic if that null hypothesis is true. To do that, we have to go further than our null hypothesis and further describe the random process that could have created our data if the null hypothesis is true.

   If we know this process, it will define the specific probability distribution of our statistic if the null hypothesis was true. This is called the **null distribution**.

   There are a lot of ways "chance" could have created non-meaningful differences between our populations. The null distribution makes specific and quantitative what was previously the qualitative question "this difference might be just due to chance."

3. How do we determine whether the null hypothesis is a plausible explanation for the data? We take the value of the statistic we actually observed in our data, and we determine whether this observed value is too unlikely under the null distribution to be plausible.

   Specifically, we calculate the probability (under the null distribution) of randomly getting a statistic X under the null hypothesis *as extreme as or more extreme* than the statistic we observed in our data (x_{obs}). This probability is called a **p-value**.

   "Extreme" means values of the test-statistic that are unlikely under the null hypothesis we are testing. In almost all tests it means large numeric values of the test-statistic, but whether we mean large positive values, large negative values, or both depends on how we define the test-statistic and which values constitute divergence from the null hypothesis. For example, if our test statistic is the *absolute* difference in the medians of two groups, then large positive values are stronger evidence of not following the null distribution:

   $$\text{p-value}(x_{obs}) = P_{H_0}(X \geq x_{obs})$$

   If we were looking at just the difference, large positive *or* negative values are evidence against the null that they are the same,[1]

   $$\text{p-value}(x_{obs}) = P_{H_0}(X \leq -x_{obs}, X \geq x_{obs}) = 1 - P_{H_0}(-x_{obs} \leq X \leq x_{obs}).$$

Question: Does the p-value give you the probability that the null is true?

4. If the observed statistic is too unlikely under the null hypothesis we can say we **reject the null hypothesis** or that we have a **statistically significant** difference.

[1] In fact the distribution of X and $|X|$ are related, and thus we can simplify our life by considering just $|X|$.

How unlikely is *too* unlikely? Often a proscribed cutoff value of 0.05 is used so that p-values *less* than that amount are considered too extreme. But there is nothing magical about 0.05, it's just a common standard if you have to make a "Reject"/"Don't reject" decision. Such a standard cutoff value for a decision is called a **level**. Even if you need to make a Yes/No type of decision, you should report the p-value as well because it gives information about *how* discordant with the null hypothesis the data is.

3.2.1 Where did the data come from? Valid tests & Assumptions

Just because a p-value is reported, doesn't mean that it is correct. You must have a **valid** test. A valid test simply means that the p-value (or level) that you report is accurate. This is only true if the null distribution of the test statistic is correctly identified. To use the tools of statistics, we must assume some kind of random process created the data. When your data violates the assumptions of the data generating process, your p-value can be quite wrong.

What does this mean to violate the assumptions? After all, the whole point of hypothesis testing is that we're trying to detect when the statistic doesn't follow the null hypothesis distribution, so obviously we will frequently run across examples where the assumption of the null hypothesis is violated. Does this mean p-values are not valid unless the null-hypothesis is true? Obviously not, other. Usually, our null hypothesis is about one specific feature of the random process – that is our actual null hypothesis we want to test. The random process that we further assume in order to get a precise null statistic, however, will have *further assumptions*. These are the assumptions we refer to in trying to evaluate whether it is legitimate to rely on hypothesis testing/p-values.

Sometimes we can know these assumptions are true, but often not; knowing where your data came from and how it is collected is critical for assessing these questions. So we need to always think deeply about where the data come from, how they were collected, etc.

Example: Data that is a Complete Census For example, for the airline data, we have one dataset that gives *complete* information about the month of January. We can ask questions about flights in January, and get the answer by calculating the relevant statistics. For example, if we want to know whether the average flight is more delayed on United than American, we calculate the means of both groups and simply compare them. End of story. There's no randomness or uncertainty, and we don't need the inference tools from above. It doesn't make sense to have a p-value here.

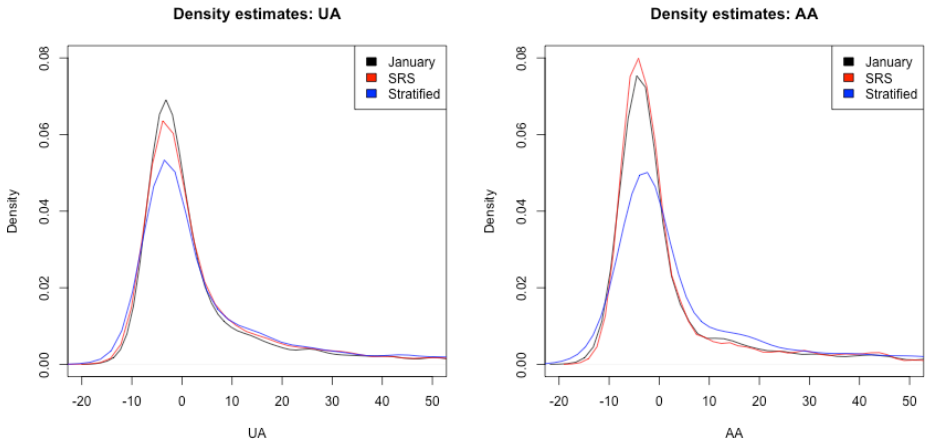
Types of Samples

For most of statistical applications, it is not the case that we have a complete census. We have a *sample* of the entire population, and want to make statements about the entire population, which we don't see. Notice that having a sample does not necessarily mean a random sample. For example, we have all of January which is a complete census of January, but is also a sample from the entire year, and there is no randomness involved in how we selected the data from the larger population.

Some datasets might be a sample of the population with no easy way to describe the process of how the sample was chosen from the population, for example data from volunteers or other *convenience samples* that use readily available data rather than randomly sampling from the population. Having convenience samples can make it quite fraught to try to make any conclusions about the population from the sample; generally we have to make assumptions about the data was collected, but because we did not control how the data is collected, we have no idea if the assumptions are true.

Question: What problems do you have in trying to use the flight data on January to estimate something about the entire year? What would be a better way to get flight data?

We discussed this issue of how the data was collected for estimating histograms. There, our histogram is a good estimate of the population when our data is a i.i.d sample or SRS, and otherwise may be off base. For example, here is the difference in our density estimates from Chapter 2 applied to three different kinds of sampling, the whole month of January, a i.i.d sample from the year, and a Stratified Sample, that picked a SRS of the same size from each month of the year:



Recall in Chapter 2, that we said while the method we learned is appropriate for

a SRS, there are also good estimates for other kind of *random* samples, like the Stratified Sample, though learning about beyond the reach of this course. The key ingredient that is needed to have trustworthy estimates is to precisely know the probability mechanism that drew the samples. This is the key difference between a random sample (of any kind), where we control the random process, and a sample of convenience – which may be random, but we don't know *how* the random sample was generated.

Assumptions versus reality

A prominent statistician, George Box, gave the following famous quote,

All models are wrong but some are useful

All tests have assumptions, and most are often not met in practice. This is a continual problem in interpreting the results of statistical methods. Therefore there is a great deal of interest in understanding how badly the tests perform if the assumptions are violated; this is often called being **robust** to violations. We will try to emphasize both what the assumptions are, and how bad it is to have violations to the assumptions.

For example, in practice, much of data that is available is not a carefully controlled random sample of the population, and therefore a sample of convenience in some sense (there's a reason we call them convenient!). Our goal is not to make say that analysis of such data is impossible, but make clear about why this might make you want to be cautious about over-interpreting the results.

3.3 Permutation Tests

Suppose we want to compare the the proportion of flights with greater than 15 minutes delay time of United and American airlines. Then our test statistic will be the difference between that proportion

The permutation test is a very simple, straightforward mechanism for comparing two groups that makes very few assumptions about the distribution of the underlying data. The permutation test basicallly assumes that the data we saw we could have seen anyway even if we changed the group assignments (i.e. United or American). Therefore, any difference we might see between the groups is due to the luck of the assignment of those labels.

The null distribution for the test statistic (difference of the proportion) under the null hypothesis for a permutation tests is determined by making the following assumptions:

1. There is no difference between proportion of delays greater than 15 minutes between the two airlines,

$$H_0 : p_{UA} = p_{AA}$$

This is the main feature of the null distribution to be tested

2. The statistic observed is the result of randomly assigning the labels amongst the observed data. This is the additional assumption about the random process that allows for calculating a precise null distribution of the statistic. It basically expands our null hypothesis to say that the distribution of the data between the two groups is the same, and the labels are just random assignments to data that comes from the same distribution.

3.3.1 How do we implement it?

This is just words. We need to actually be able to compute probabilities under a specific distribution. In other words, if we were to have actually just randomly assigned labels to the data, we need to know what is the probablility we saw the difference we actually saw?

The key assumption is that the data we measured (the flight delay) was fixed for each observation and completely independent from the airline the observation was assigned to. We imagine that the airline assignment was completely random and separate from the flight delays – a bunch of blank airplanes on the runway that we at the last minute assign to an airline, with crew and passengers (not realistic, but a thought experiment!)

If our data actually was from such a scenario, we could actually rerun the random assignment process. How? By randomly reassigning the labels. Since (under the null) we assume that the data we measured had nothing to do with those labels, we could have instead observed another assignment of those airline labels and we would have seen the same data with just different labels on the planes. These are called **permutations** of the labels of the data.

Here is some examples of doing that. Below, are the results of three different possible permutations, created by assigning planes (rows/observations) to an airline randomly – only the first five observations are shown, but all of the original observations get an assignment. Notice the number of planes assigned to UA vs AA stays the same, just which plane gets assigned to which airline changes. The column `Observed` shows the assignment we actually saw (as opposed to the assignments I made up by permuting the assignments)

```
##   FlightDelay Observed Permutation1 Permutation2 Permutation3
## 1           5       UA           AA           UA           AA
## 2          -6       UA           UA           UA           UA
## 3         -10       AA           UA           UA           UA
## 4          -3       UA           UA           AA           UA
## 5          -3       UA           UA           AA           UA
## 6           0       UA           UA           UA           UA
```

For each of these three permutations, I can calculate proportion of flights de-

layed, among those assigned to UA vs those assigned to AA, and calculate the difference between them

```
## Proportions per Carrier, each permutation:

##     Observed Permutation1 Permutation2 Permutation3
## AA 0.1554878    0.2063008    0.1951220    0.1910569
## UA 0.2046216    0.1878768    0.1915606    0.1929002

## Differences in Proportions per Carrier, each permutation:

##     Observed Permutation1 Permutation2 Permutation3
##  0.049133762 -0.018424055 -0.003561335  0.001843290
```

I've done this for three permutations, but we could enumerate (i.e. list) all possible such assignments of planes to airlines. If we did this, we would have the complete set potential flight delay datasets possible under the null hypothesis, and for each one we could calculate the difference in the proportion of delayed flights between the airlines.

So in principle, it's straightforward – I just do this for every possible permutation, and get the difference of proportions. The result set of differences gives the distribution of possible values under the null. These values would define our null distribution. With all of these values in hand, I could calculate probabilities – like the probability of seeing a value so large as the one observed in the data (p-value!).

Too many! In practice: Random selection

This is the principle of the permutation test, but I'm not about to do that in practice, because it's not computationally feasible!

Consider if we had only, say, 14 observations with two groups of 7 each, how many permutations do we have? This is 14 "choose" 7, which gives 3,432 permutations.

So for even such a small dataset of 14 observations, we'd have to enumerate almost 3500 permutations. In the airline data, we have 984-2986 observations per airline. We can't even determine how many permutations that is, much less actually enumerate them all.

So for a reasonably sized dataset, what can we do? Instead, we consider that there exists such a null distribution and while we can't calculate it perfectly, we are going to just approximate that null distribution.

How? Well, this is a problem we've actually seen. If we want to estimate a true distribution of values, we don't usually have a census at our disposal – i.e. all values. Instead we draw a i.i.d. sample from the population, and with that sample we can estimate that distribution, either by a histogram or by calculating probabilities (see Chapter 2).

How does this look like here? We know how to create a single random permutation – it's what I did above using the function `sample`. If we repeat this over and over and create a lot of random permutations, we are creating a i.i.d. sample from our population. Specifically, each possible permutation is an element of our sample space, and we are randomly drawing a permutation. We'll do this many times (i.e. many calls to the function `sample`), and this will create a i.i.d. sample of permutations. Once we have a i.i.d. sample of permutations, we can calculate the test statistic for each permutation, and get an estimate of the true null distribution. Unlike i.i.d. samples of an actual population data, we can make the size of our sample as large as our computer can handle to improve our estimate (though we don't in practice need it to be obscenely large)

Practically, this means we will repeating what we did above many times. The function `replicate` in R allows you to repeat something many times, so we will use this to repeat the sampling and the calculation of the difference in medians.

I wrote a little function `permutation.test` to do this for any statistic, not just difference of the medians; this way I can reuse this function repeatedly in this chapter. You will go through this function in lab and also in the accompanying code.

```
permutation.test <- function(group1, group2, FUN, n.repetitions) {
    stat.obs <- FUN(group1, group2)
    makePermutedStats <- function() {
        sampled <- sample(1:length(c(group1, group2)),
            size = length(group1), replace = FALSE)
        return(FUN(c(group1, group2)[sampled], c(group1,
            group2)[-sampled]))
    }
    stat.permute <- replicate(n.repetitions, makePermutedStats())
    p.value <- sum(stat.permute >= stat.obs)/n.repetitions
    return(list(p.value = p.value, observedStat = stat.obs,
        permutedStats = stat.permute))
}
```

Example: Proportion Later than 15 minutes

We will demonstrate this procedure on our the i.i.d. sample from our flight data, using the difference in the proportions later than 15 minutes as our statistic.

Recall, the summary statistics on our actual data:

```
tapply(flightSFOSRS$DepDelay, flightSFOSRS$Carrier,
    propFun)[c("AA", "UA")]
```

```
##        AA        UA
## 0.1554878 0.2046216
```

I am going to choose as my statistic the *absolute* difference between the pro-

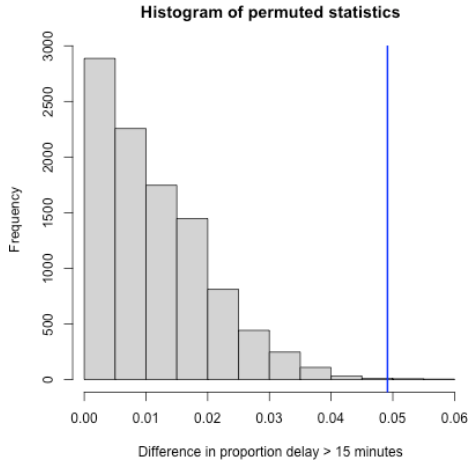
portion later than 15 minutes. This will mean that large values are always considered extreme for my p-value computations. This is implemented in my `diffProportion` function:

```
diffProportion <- function(x1, x2) {
    prop1 <- propFun(x1)
    prop2 <- propFun(x2)
    return(abs(prop1 - prop2))
}
diffProportion(subset(flightSFOSRS, Carrier == "AA")$DepDelay,
    subset(flightSFOSRS, Carrier == "UA")$DepDelay)
```

```
## [1] 0.04913376
```

Now I'm going to run my permutation function using this function.

Here is the histogram of the values of the statistics under all of my permutations.



If my data came from the null, then this is the (estimate) of the actual distribution of what the test-statistic would be.

How would I get a p-value from this? Recall the definition of a p-value – the probability under the null distribution of getting a value of my statistic as large or larger than what I observed in my data,

$$\text{p-value}(x_{obs}) = P_{H_0}(X \geq x_{obs})$$

So I need to calculate that value from my estimate of the null distribution (demonstrated in the histogram above), i.e. the proportion of the that are greater than observed.

My function calculated the p-value as well in this way, so we can output the value:

```
## pvalue= 0.0011
```

Question:

1. So what conclusions would you draw from this permutation test?
2. What impact does this test have? What conclusions would you be likely to make going forward?
3. Why do I take the absolute difference? What difference does it make if you change the code to be only the difference?

Median difference

What about if I look at the difference in median flight delay between the two airlines? Let's first look at what is the median flight delay for each airline:

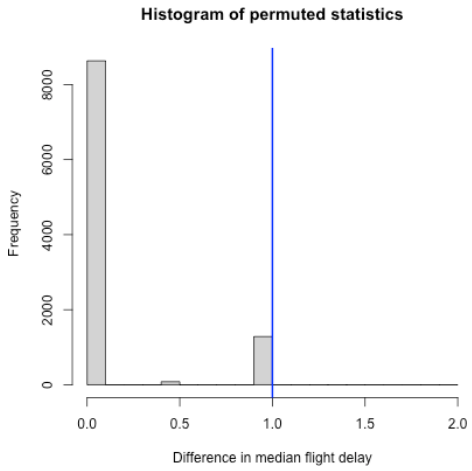
```
tapply(flightSFOSRS$DepDelay, flightSFOSRS$Carrier,
    function(x) {
        median(x, na.rm = TRUE)
    })[c("AA", "UA")]
```

```
## AA UA
## -2 -1
```

The first thing we might note is that there is a very small difference between the two airlines (1 minute). So even if we find something significant, who really cares? That is not going to change any opinions about which airline I fly. Statistical significance is not everything.

However, I can still run a permutation test (you can always run tests, even if it's not sensible!). I can reuse my previous function, but just quickly change the statistic I consider – now use the absolute difference in the median instead of proportion more than 15min late.

Here is the histogram I get after doing this:



This gives us a p-value:

```
## pvalue (median difference)= 0.1287
```

Question:

1. What is going on with our histogram? Why does it look so different from our usual histograms?
2. What would have happened if we had defined our p-value as the probability of being *greater* rather than *greater than or equal to*? Where in the code of `permutation.test` was this done, and what happens if you change the code for this example?

3.3.2 Assumptions: permutation tests

Let's discuss what might be limitations of the permutation test.

Assumption: the data generating process

What assumption(s) are we making about the random process that generated this data in determining the null distribution? Does it make sense for our data?

We set up a model that the assignment of a flight to one airline or another was done at random. This is clearly not a plausible description of our of data.

Some datasets do have this flavor. For example, if we wanted to decide which of two email solicitations for a political campaign are most likely to lead to someone to donate money, we could randomly assign a sample of people on our mailing list to get one of the two. This would perfectly match the data generation assumed in the null hypothesis.

What if our assumption about random labels is wrong?

Clearly random assignment of labels is not a good description for how the datasets regarding flight delay data were created. Does this mean the permutation test will be invalid? No, not necessarily. In fact, there are other descriptions of null random process that do not explicitly follow this description, but in the end result in the same null distribution as that of the randomly assigned labels model.

Explicitly describing the full set of random processes that satisfy this requirement is beyond the level of this class[2], but an important example is if each of your data observations can be considered under the null a random, independent draw from the same distribution. This is often abbreviated **i.i.d: independent and identically distributed**. This makes sense as an requirement – the very

[2]Namely, if the data can be assumed to be *exchangeable* under the null hypothesis, then the permutation test is also a valid test.

act of permuting your data implies such an assumption about your data: that you have similar observations and the only thing different about them is which group they were assigned to (which under the null doesn't matter).

Assuming your data is i.i.d is a common assumption that is thrown around, but is actually rather strong. For example, non-random samples do not have this property, because there is no randomness; it is unlikely you can show that convenience samples do either. However, permutation tests are a pretty good tool even in this setting, however, compared to the alternatives. Actual random assignments of the labels is the strongest such design of how to collect data.

Inferring beyond the sample population

Note that the randomness queried by our null hypothesis is all about the specific observations we have. For example, in our political email example we described above, the randomness is if we imagine that we assigned *these same people* different email solicitations – our null hypothesis asks what variation in our statistic would we expect? However, if we want to extend this to the general population, we have to make the assumption that these people's reaction are representative of the greater population.

As a counter-example, suppose our sample of participants was only women, and we randomly assigned these women to two groups for the two email solicitations. Then this data matches the assumptions of the permutation test, and the permutation test is valid for answering the question about whether any affect seen amongst *these women* was due to the chance assignment to these women. But that wouldn't answer our question very well about the general population of interest, which for example might include men. Men might have very different reactions to the same email. This is a rather obvious example, in the sense that most people in designing a study wouldn't make this kind of mistake. But it's exemplifies the kind of problems that can come from a haphazard selection of participants, even if you do random assignment of the two options. Permutation tests do not get around the problem of a poor data sample. Random samples from the population are needed to be able to make the connection back to the general population.

Conclusion

So while permuting your data seems to intuitive and is often thought to make no assumptions, it does have assumptions about where your data come from.

Generally, the assumptions for a permutation test are much less than some alternative tests (like the parametric tests we'll describe next), so they are generally the safest to use. But it's useful to realize the limitations even for something as non-restrictive as permutation tests.

3.4 Parametric test: the T-test

In parametric testing, we assume the data comes from a specific family of distributions that share a functional form for their density, and define the features of interest for the null hypothesis based on this distribution.

Rather than resampling from the data, we will use the fact that we can write down the density of the data-generating distribution to analytically determine the null distribution of the test statistic. For that reason, parametric tests tend to be limited to a narrower class of statistics, since the statistics have to be tractable for mathematical analysis.

3.4.1 Parameters

We have spoken about parameters in the context of parameters that define a family of distributions all with the same mathematical form for the density. An example is the normal distribution which has two parameters, the mean (μ) and the variance (σ^2). Knowing those two values defines the entire distribution of a normal. The parameters of a distribution are often used to define a null hypothesis; a null hypothesis will often be a direct statement about the parameters that define the distribution of the data. For example, if we believe our data is normally distributed in both of our groups, our null hypothesis could be that the mean parameter in one group is equal to that of another group.

General Parameters However, we can also talk more generally about a parameter of any distribution beyond the defining parameters of the distribution. A parameter is any numerical summary that we can calculate from a distribution. For example, we could define the .75 quantile as a parameter of the data distribution. Just as a statistic is any function of our observed data, a **parameter** is a function of the true generating distribution F. Which means that our null hypothesis could also be in terms of other parameters than just the ones that define the distribution. For example, we could assume that the data comes from a normal distribution and our null hypothesis could be about the .75 quantile of the distribution. Indeed, we don't have to have assume that the data comes from any parametric distribution – every distribution has a .75 quantile.

If we do assume our data is generated from a family of distributions defined by specific parameters (e.g. a normal distribution with unknown mean and variance) then those parameters completely define the distribution. Therefore any arbitrary parameter of the distribution we might define can be written as a function of those parameters. So the 0.75 quantile of a normal distribution is a parameter, but it is also a function of the mean parameter and variance parameter of the normal distribution.

Notation Parameters are often indicated with greek letters, like θ, α, β, σ.

Statistics of our data sample are often chosen because they are estimates of our parameter. In that case they are often called the same greek letters as the parameter, only with a "hat" on top of them, e.g. $\hat{\theta}$, $\hat{\alpha}$, $\hat{\beta}$, $\hat{\sigma}$. Sometimes, however, a statistic will just be given a upper-case letter, like T or X, particularly when they are not estimating a parameter of the distribution.

3.4.2 More about the normal distribution and two group comparisons

Means and the normal distribution play a central role in many parametric tests, so lets review a few more facts.

Standardized Values

If $X \sim N(\mu, \sigma^2)$, then

$$\frac{X - \mu}{\sigma} \sim N(0, 1)$$

This transformation of a random variable is called standardizing X, i.e. putting it on the standard $N(0, 1)$ scale.

Sums of normals

If $X \sim N(\mu_1, \sigma_1^2)$ and $Y \sim N(\mu_2, \sigma_2^2)$ and X and Y are independent, then

$$X + Y \sim N(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$$

If X and Y are both normal, but not independent, then their sum is still a normal distribution with mean equal to $\mu_1 + \mu_2$ but the variance is different.[3]

CLT for differences of means

We've reviewed that a sample mean of a i.i.d. sample or SRS sample will have a sampling distribution that is roughly a normal distribution if we have a large enough sample size – the Central Limit Theorem. Namely, that if $X_1, \ldots, X_n$ are i.i.d from a distribution[4] with mean μ and variance σ^2, then $\hat{\mu} = \bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$ will have a roughly normal distribution

$$N(\mu, \frac{\sigma^2}{n}).$$

If we have two groups,

- $X_1, \ldots, X_{n_1}$ i.i.d from a distribution with mean μ_1 and variance σ_1^2, and

[3] $X + Y \sim N(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2 + 2cov(X, Y))$, where cov is the covariance between X and Y

[4] And in fact, there are many variations of the CLT, which go beyond i.i.d samples

- $Y_1, \dots, Y_{n_2}$ i.i.d from a distribution with mean μ_2 and variance σ_2^2

Then if the X_i and Y_i are independent, then the central limit theorem applies and $\bar{X} - \bar{Y}$ will have a roughly normal distribution equal to

$$N(\mu_1 - \mu_2, \frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2})$$

3.4.3 Testing of means

Let μ_{UA} and μ_{AA} be the true means of the distribution of flight times of the two airlines in the population. Then if we want to test if the distributions have the same mean, we can write our null hypothesis as

$$H_0 : \mu_{AA} = \mu_{UA}$$

This could also be written as

$$H_0 : \mu_{AA} - \mu_{UA} = \delta = 0,$$

so in fact, we are testing whether a specific parameter δ is equal to 0.

Let's assume $X_1, \dots, X_{n_1}$ is the data from United and $Y_1, \dots, Y_{n_2}$ is the data from American. A natural sample statistic to estimate δ from our data would be

$$\hat{\delta} = \bar{X} - \bar{Y},$$

i.e. the difference in the means of the two groups.

Null distribution

To do inference, we need to know the distribution of our statistic of interest. Our central limit theorem will tell us that under the null, for large sample sizes, the difference in means is distributed normally,

$$\bar{X} - \bar{Y} \sim N(0, \frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2})$$

This is therefore the null distribution, under the assumption that our random process that created the data is that the data from the two groups is i.i.d from normal distributions with the same mean. Assuming we know σ_1 and σ_2, we can use this distribution to determine whether the observed $\bar{X} - \bar{Y}$ is unexpected under the null.

We can also equivalently standardize $\bar{X} - \bar{Y}$ and say,

$$Z = \frac{\bar{X} - \bar{Y}}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}} \sim N(0, 1)$$

and instead use Z as our statistic.

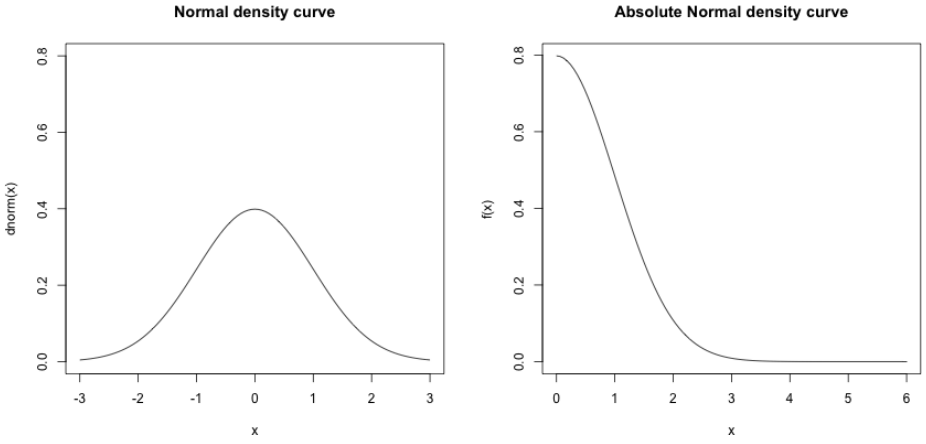
Calculating a P-value

Suppose that we observe a statistic $Z = 2$. To calculate the p-value we need to calculate the probability of getting a value as extreme as 2 or more under the null. What does extreme mean here? We need to consider what values of Z (or the difference in our means) would be considered evidence that the null hypothesis didn't explain the data. Going back to our example, $\bar{X} - \bar{Y}$ might correspond to $\bar{X}_{AA} - \bar{Y}_{UA}$, and clearly large positive values would be evidence that they were different. But large negative values also would be evidence that the means were different. Either is equally relevant as evidence that the null hypothesis doesn't explain the data.

So a reasonable definition of extreme is large values in either direction. This is more succinctly written as $|\bar{X} - \bar{Y}|$ being large.

So a better statistic is,

$$|Z| = \frac{|\bar{X} - \bar{Y}|}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$$



Question: With this better $|Z|$ statistic, what is the p-value if you observe $Z = 2$? How would you calculate this using the standard normal density curve? With R?

$|Z|$ is often called a 'two-sided' t-statistic, and is the only one that we will consider.[5]

[5]There are rare cases in comparing means where you might consider only evidence against the null that is positive (or negative). In this case you would then calculate the p-value correspondingly. These are called "one-sided" tests, for the same value of the observed statistic Z they give you smaller p-values, and they are usually only a good idea in very specific

3.4.4 T-Test

The above test is actually just a thought experiment because $|Z|$ is not in fact a statistic because we don't know σ_1 and σ_2. So we can't calculate $|Z|$ from our data!

Instead you must estimate these unknown parameters with the **sample variance**

$$\hat{\sigma}_1^2 = \frac{1}{n-1}\sum (X_i - \bar{X})^2,$$

and the same for $\hat{\sigma}_2^2$. (Notice how we put a "hat" over a parameter to indicate that we've estimated it from the data.)

But once you must estimate the variance, you are adding additional variability to inference. Namely, before, assuming you knew the variances, you had

$$|Z| = \frac{|\bar{X} - \bar{Y}|}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}},$$

where only the numerator is random. Now we have

$$|T| = \frac{|\bar{X} - \bar{Y}|}{\sqrt{\frac{\hat{\sigma}_1^2}{n_1} + \frac{\hat{\sigma}_2^2}{n_2}}}.$$

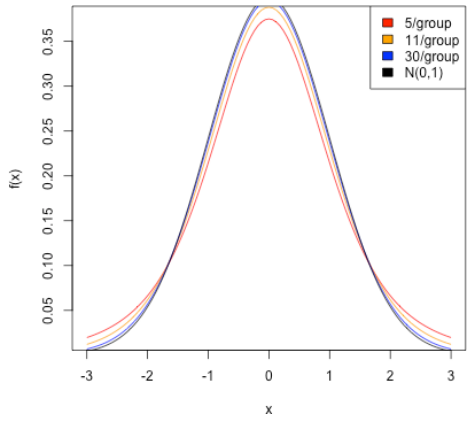
and the denominator is also random. T is called the **t-statistic**.

This additional uncertainty means seeing a large value of $|T|$ is more likely than of $|Z|$. Therefore, $|T|$ has a different distribution, and it's not $N(0,1)$.

Unlike the central limit theorem, which deals only with the distributions of means, when you additionally estimate the variance terms, determining even approximately what is the distribution of T (and therefore $|T|$) is more complicated, and in fact depends on the distribution of the input data X_i and Y_i (unlike the central limit theorem). But if the distributions creating your data are reasonably close to normal distribution, then T follows what is called a t-distribution.

---

examples.



You can see that the t distribution is like the normal, only it has larger "tails" than the normal, meaning seeing large values is more likely than in a normal distribution.

Question: What happens as you change the sample size?

Notice that if you have largish datasets (e.g. $> 30 - 50$ samples in *each* group) then you can see that the t-distribution is numerically almost equivalent to using the normal distribution, so that's why it's usually fine to just use the normal distribution to get p-values. Only in small samples sizes are there large differences.

Degrees of Freedom

The t-distribution has one additional parameter called the **degrees of freedom**, often abreviated as *df*. This parameter has nothing to do with the mean or standard deviation of the data (since our t-statistic is already standardized), and depends totally on the sample size of our populations. The actual equation for the degrees of freedom is quite complicated:

$$df = \frac{\left(\frac{\hat{\sigma}_1^2}{n_1} + \frac{\hat{\sigma}_2^2}{n_2}\right)^2}{\frac{(\frac{\hat{\sigma}_1^2}{n_1})^2}{n_1 - 1} + \frac{(\frac{\hat{\sigma}_2^2}{n_2})^2}{n_2 - 1}}.$$

This is not an equation you need to learn or memorize, as it is implemented in R for you. A easy approximation for this formula is to use

$$df \approx min(n_1 - 1, n_2 - 1)$$

This approximation is mainly useful to try to understand how the degrees of freedom are changing with your sample size. Basically, the size of the smaller group is the important one. Having one huge group that you compare to a

small group doesn't help much – you will do better to put your resources into increasing the size of the smaller group (in the actual formula it helps a little bit more, but the principle is the same).

3.4.5 Assumptions of the T-test

Parametric tests usually state their assumptions pretty clearly: they assume a parametric model generated the data in order to arrive at the mathematical description of the null distribution. For the t-test, we assume that the data $X_1, \ldots, X_{n_1}$ and $Y_1, \ldots, Y_{n_2}$ are normal to get the t-distribution.

What happens if this assumption is wrong? When will it still make sense to use the t-test?

If we didn't have to estimate the variance, the central limit theorem tells us the normality assumption will work for any distribution, *if* we have a large enough sample size.

What about the t-distribution? That's a little tricker. You still need a large sample size; you also need that the distribution of the X_i and the Y_i, while not required to be exactly normal, not be too far from normal. In particular, you want them to be symmetric (unlike our flight data).[6]

Generally, the t-statistic is reasonably robust to violations of these assumptions, particularly compared to other parametric tests, if your data is not too skewed and you have a largish sample size (e.g. 30 samples in a group is good). But the permutation test makes far fewer assumptions, and in particular is very robust to assumptions about the distribution of the data.

For small sample sizes (e.g. < 10 in each group), you certainly don't really have any good justification to use the t-distribution unless you have a reason to trust that the data is normally distributed (and with small sample sizes it is also very hard to justify this assumption by looking at the data).

3.4.6 Flight Data and Transformations

Let's consider the flight data. Recall, the t-statistic focuses on the difference in means. Here are the means of the flight delays in the two airlines we have been considering:

```
##        AA        UA
##  7.728294 12.255649
```

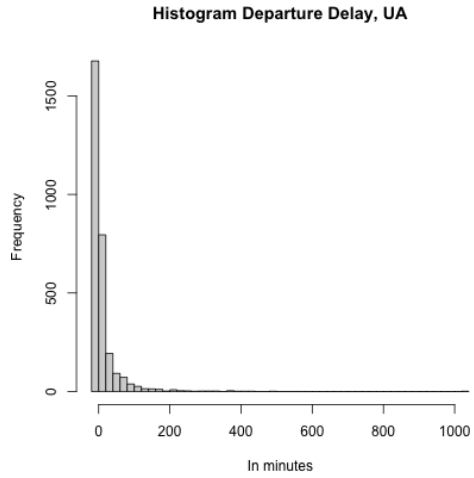
[6]Indeed, the central limit theorem requires large data sizes, and how large a sample you need for the central limit theorem to give you a good approximation also depends on things about the distribution of the data, like how symmetric the distribution is.

Question: Why might the difference in the means not be a compelling comparison for the flight delay?

The validity of the t-test depends assumptions about the distribution of the data, and a common way to assess this is to look at the distribution of each of the two groups.

With larger sample sizes there is less worry about the underlying distribution, but very non-normal input data will not do well with the t-test, particularly if the data is **skewed**, meaning not symmetrically distributed around its mean.

Here is a histogram of the flight delay data:



Question: Looking at the histogram of the flight data, would you conclude that the t-test would be a valid choice?

Note that nothing stops us from running the test, whether it is a good idea or not, and it's a simple one-line code:

```
t.test(flightSFOSRS$DepDelay[flightSFOSRS$Carrier ==
    "UA"], flightSFOSRS$DepDelay[flightSFOSRS$Carrier ==
    "AA"])
```

```
##
##  Welch Two Sample t-test
##
## data:  flightSFOSRS$DepDelay[flightSFOSRS$Carrier == "UA"] and flightSFOSRS$DepDelay
## t = 2.8325, df = 1703.1, p-value = 0.004673
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
```

```
##  1.392379 7.662332
## sample estimates:
## mean of x mean of y
## 12.255649  7.728294
```

This is a common danger of parametric tests. They are implemented everywhere (there are on-line calculators that will compute this for you; excel will do this calculation), so people are drawn to doing this, while permutation tests are more difficult to find pre-packaged.

Direct comparison to the permutation test

The permutation test can use any statistic we like, and the t-statistic is a perfectly reasonable way to compare two distributions if we are interested in comparing the means (though as we mentioned, we might not be!). So we can compare the t-test to a permutation test of the mean *using the t-statistic*. We implement the permutation test with the t-statistic here:

```
set.seed(489712)
tstatFun <- function(x1, x2) {
    abs(t.test(x1, x2)$statistic)
}
dataset <- flightSFOSRS
output <- permutation.test(group1 = dataset$DepDelay[dataset$Carrier ==
    "UA"], group2 = dataset$DepDelay[dataset$Carrier ==
    "AA"], FUN = tstatFun, n.repetitions = 10000)
cat("permutation pvalue=", output$p.value)
```

```
## permutation pvalue= 0.0076
```

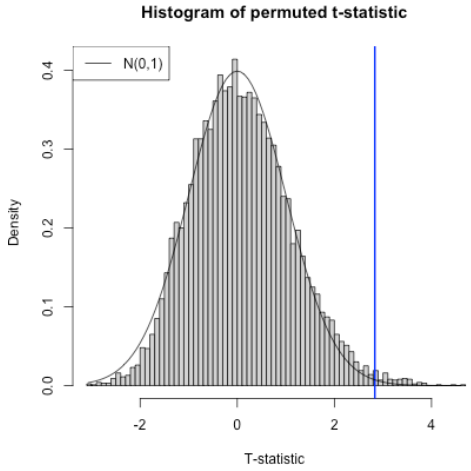
We can also run the t-test again and compare it.

```
tout <- t.test(flightSFOSRS$DepDelay[flightSFOSRS$Carrier ==
    "UA"], flightSFOSRS$DepDelay[flightSFOSRS$Carrier ==
    "AA"])
cat("t-test pvalue=", tout$p.value)
```

```
## t-test pvalue= 0.004673176
```

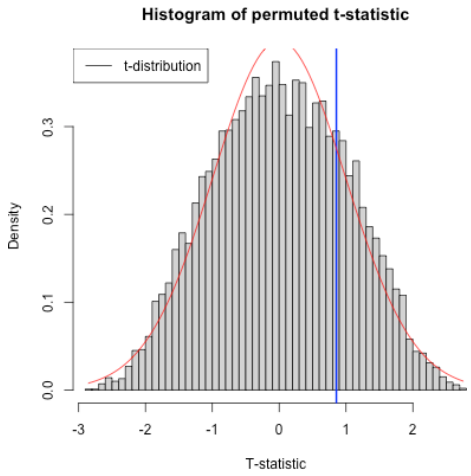
They don't result in very different conclusions.

We can compare the distribution of the permutation distribution of the t-statistic, and the density of the $N(0,1)$ that the parametric model assumes. We can see that they are quite close, *even though our data is very skewed and clearly non-normal*. Indeed for large sample sizes like we have here, they will often give similar results, even though our data is clearly not meeting the assumptions of the t-test. The t-test really is quite robust to violations.



Smaller Sample Sizes

If we had a smaller dataset we would not get such nice behavior. We can take a sample of our dataset to get a smaller sample of the data of size 20 and 30 in each group. Running both a t-test and a permutation test on this sample of the data, we can see that we do not get a permutation distribution that matches the (roughly) N(0,1) we use for the t-test.




```
## pvalue permutation= 0.4446
```

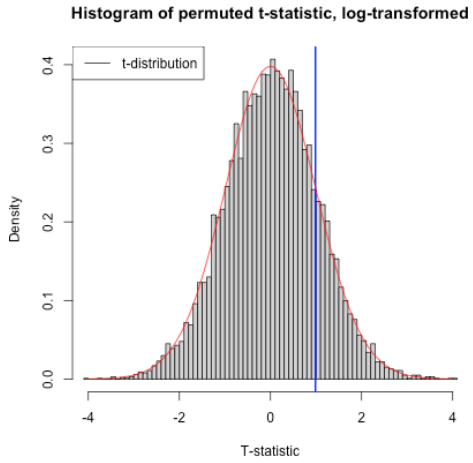
```
## pvalue t.test= 0.394545
```

Question: What different conclusions do you get from the two tests with these smaller datasizes?

Transformations

We saw that skewed data could be problematic in visualization of the data, e.g. in boxplots, and transformations are helpful in this setting. Transformations can also be helpful for applying parametric tests. They can often allow the parametric t-test to work better for smaller datasets.

If we compare both the permutation test and the t-test on log-transformed data, then even with the smaller sample sizes the permutation distribution looks much closer to the t-distribution.




```
## pvalue permutation= 0.4446
```

```
## pvalue t.test= 0.3261271
```

Question: Why didn't the p-value for the permutation test change?

Question: What does it mean for my null hypothesis to transform to the log-scale? Does this make sense?

3.4.7 Why parametric models?

We do the comparison of the permutation test to the parametric t-test not to encourage the use of the the t-test in this setting – the data, even after transformation, is pretty skewed and there's no reason to not use the permutation test instead. The permutation test will give pretty similar answers regardless of the transformation[7] and is clearly indicated here.

[7] In fact, if we were working with the difference in the means, rather than the t-statistics, which estimates the variance, the permutation test would give exactly the same answer since

This exercise was to show the use and limits of using the parametric tests, and particularly transformations of the data, in an easy setting. Historically, parametric t-tests were necessary in statistics because there were not computers to run permutation tests. That's clearly not compelling now! However, it remains that parametric tests are often easier to implement (one-line commands in R, versus writing a function), and you will see parametric tests frequently (even when resampling methods like permutation tests and bootstrap would be more justifiable).

The take-home lesson here regarding parametric tests is that when there are large sample sizes, parametric tests can overcome violations of their assumptions[8] so don't automatically assume parametric tests are completely wrong to use. But a permutation test is the better all-round tool for this question: it is has more minimal assumptions, and can look at how many different statistics we can use.

There are also some important reasons to learn about t-tests, however, beyond a history lesson. They are the easiest example of a parameteric test, where you make assumptions about the distribution your data (i.e. $X_1, \ldots, X_{n_1}$ and $Y_1, \ldots, Y_{n_2}$ are normally distributed). Parametric tests generally are very important, even with computers. Parametric models are particularly helpful for researchers in data science for the development of new methods, particularly in defining good test statistics, like T.

Parametric models are also useful in trying to understand the limitations of a method, mathematically. We can simulate data under different models to understand how a statistical method behaves.

There are also applications where the ideas of bootstrap and permutation tests are difficult to apply. Permutation tests, in particular, are quite specific. Bootstrap methods, which we'll review in a moment, are more general, but still are not always easy to apply in more complicated settings. A goal of this class is to make you comfortable with parametric models (and their accompanying tests), in addition to the resampling methods you've learned.

3.5 Digging into Hypothesis tests

Let's break down some important concepts as to what makes a test. Note that all of these concepts will apply for *any* hypothesis test.

1. A null hypothesis regarding a particular feature of the data
2. A test statistic for which extreme values indicates less correspondence with the null hypothesis
3. An assumption of how the data was generated under the null hypothesis

the log is a monotone transformation.

[8] At least those tests based on the central limit theorem!

4. The distribution of the test statistic under the null hypothesis.

As we've seen, different tests can be used to answer the same basic "null" hypothesis – are the two groups "different"? – but the specifics of how that null is defined can be quite different. For any test, you should be clear as to what the answer is to each of these points.

3.5.1 Significance & Type I Error

The term significance refers to measuring how incompatible the data is with the null hypothesis. There are two important terminologies that go along with assessing significance.

p-values You often report a p-value to quantify how unlikely the data is under the null.

Decision to Reject/Not reject Make a final decision as to whether the null hypothesis was too unlikely to have reasonably created the data we've seen – either reject the null hypothesis or not.

We can just report the p-value, but it is common to also make an assessment of the p-value and give a final decision as well. In this case we pick a cutoff, e.g. p-value of 0.05, and report that we reject the null.

You might see sentences like "We reject the null at level 0.05." The **level** chosen for a test is an important concept in hypothesis testing and is the cutoff value for a test to be significant. In principle, the idea of setting a level is that it is a standard you can require before declaring significance; in this way it can keep researchers from creeping toward declaring significance once they see the data and see they have a p-value of 0.07, rather than 0.05. However, in practice a fixed cutoff value can have the negative result of encouraging researchers to fish in their data until they find *something* that has a p-value less than 0.05.

Commonly accepted cutoffs for unlikely events are 0.05 or 0.01, but these values are too often considered as magical and set in stone. Reporting the actual p-value is more informative than just saying yes/no whether you reject (rejecting with a p-value of 0.04 versus 0.0001 tells you something about your data).

More about the level

Setting the level of a test defines a repeatable procedure: "reject if p-value is $< \alpha$". The level of the test actually reports the uncertainity in this procedure. Specifically, with any test, you can make two kinds of mistakes:

- Reject the null when the null is true (**Type I error**)
- Not reject the null when the null is in fact not true (**Type II error**)

Then the **level** of a decision is the probability of this procedure making a type I error: if you always reject at 0.05, then 5% of such tests will wrongly reject the null hypothesis when in fact the null is true is true.

Note that this is no different in concept that our previous statement saying that a p-value is the likelihood under the null of an event as extreme as what we observed. However, it does a quantify how willing you are to making Type I Error in setting your cutoff value for decision making.

3.5.2 Type I Error & All Pairwise Tests

Let's make the importance of accounting and measuring Type I error more concrete. We have been considering only comparing the carriers United and American. But in fact there are 10 airlines.

We might want to compare all of them, in other words run a hypothesis test on each pair of carriers. That's a hypothesis test for each pair of airlines:

```
## number of pairs: 45
```

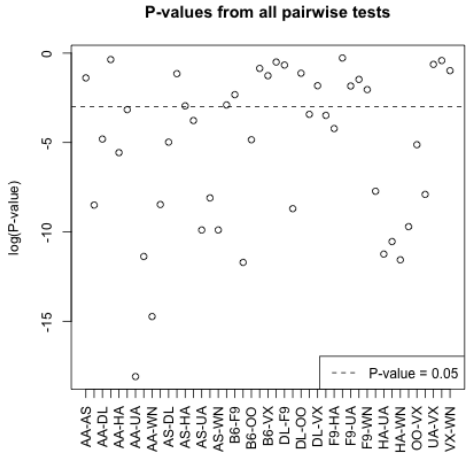
That starts to be a lot of tests. So for speed purposes in class, I'll use the t-test to illustrate this idea and calculate the t-statistic and its p-value for every pair of airline carriers (with our transformed data):

```
## [1]  2 45
```

```
##       statistic.t      p.value
## AA-AS   1.1514752 0.2501337691
## AA-B6  -3.7413418 0.0002038769
## AA-DL  -2.6480549 0.0081705864
## AA-F9  -0.3894014 0.6974223534
## AA-HA   3.1016459 0.0038249362
## AA-OO  -2.0305868 0.0424142975
```

```
##       statistic.t      p.value
## AA-AS   1.1514752 0.2501337691
## AA-B6  -3.7413418 0.0002038769
## AA-DL  -2.6480549 0.0081705864
## AA-F9  -0.3894014 0.6974223534
## AA-HA   3.1016459 0.0038249362
## AA-OO  -2.0305868 0.0424142975
```

```
## Number found with p-value < 0.05:  26 ( 0.58  proportion of tests)
```



What does this actually mean? Is this a lot to find significant?

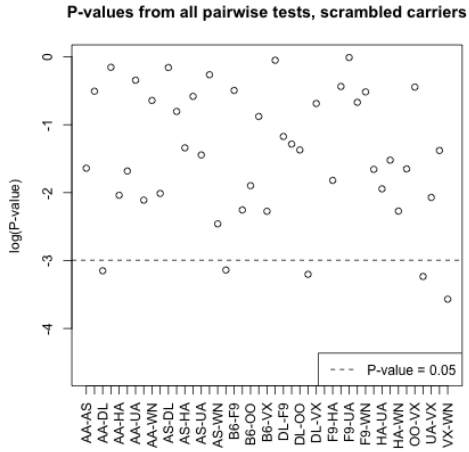
Roughly, if each of these tests has level 0.05, then even if *none* of the pairs are truly different from each other, I might expect on average around 2 to be rejected at level 0.05 just because of variation in sampling.[9] This is the danger in asking many questions from your data – something is likely to come up just by chance.[10]

We can consider this by imagining what if I scramble up the carrier labels – randomly assign a carrier to a flight. Then I know there shouldn't be any true difference amongst the carriers. I can do all the pairwise tests and see how many are significant.

```
## Number found with p-value < 0.05:  6 ( 0.13  proportion)
```

[9] In fact, this is not an accurate statement because these tests are reusing the same data, so the data in each test are not independent, and the probabilities don't work out like that. But it is reasonable for understanding the concepts here.

[10] Indeed this true of all of science, which relies on hypothesis testing, so one always has to remember the importance of the iterative process of science to re-examine past experiments.



Question: What does this suggest to you about the actual data?

Multiple Testing

Intuitively, we consider that if we are going to do all of these tests, we should have a stricter criteria for rejecting the null so that we do not routinely find pairwise differences when there are none.

Question: Does this mean the level should be higher or lower to get a 'stricter' test? What about the p-value?

Making such a change to account for the number of tests considered falls under the category of **multiple testing adjustments**, and there are many different flavors beyond the scope of the class. Let's consider the most widely known correction: the **Bonferroni correction**.

Specifically, say we will quantify our notion of `stricter` to require "of all the tests I ran, there's only a 5% chance of a type I error". Let's make this a precise statement. Suppose that of the K tests we are considering, there are $V \leq K$ tests that are type I errors, i.e. the null is true but we rejected. We will define our cummulate error rate across the set of K tests as

$$P(V \geq 1)$$

So we if we can guarantee that our testing procedure for the set of K tests has $P(V \geq 1) \leq \gamma$ we have controlled the **family-wise error rate** to level γ.

How to control the family-wise error rate?

We can do a simple correction to our K individual tests to ensure $P(V \geq 1) \leq \gamma$. If we lower the level α we require in order to reject H_0, we will lower our chance of a single type I error, and thus also lowered our family-wise error rate. Specifically, if we run the K tests and set the individual level of *each individualtest* to be $\alpha = \gamma/K$, then we will guarantee that the family-wise error rate is no more than γ.

In the example of comparing the different airline carriers, the number of tests is 45. So if we want to control our family-wise error rate to be no more than 0.05, we need each individual tests to reject only with $\alpha = 0.0011$.

```
## Number found significant after Bonferonni:  16
```

```
## Number of shuffled differences found significant after Bonferonni:  0
```

If we reject each tests only if

$$p - value \leq \alpha = \gamma/K$$

, then we can equivalently say we only reject if

$$K\frac{p - value}{\leq}\gamma$$

We can therefore instead think only about γ (e.g. 0.05), and create **adjusted p-values**, so that we can just compare our adjusted p-values directly to γ. In this case if our standard (single test) p-value is p, we have

$$\text{Bonferroni adjusted p-values} = p \times K$$

```
##       statistic.t      p.value  p.value.adj
## AA-AS   1.1514752 0.2501337691 11.256019611
## AA-B6  -3.7413418 0.0002038769  0.009174458
## AA-DL  -2.6480549 0.0081705864  0.367676386
## AA-F9  -0.3894014 0.6974223534 31.384005904
## AA-HA   3.1016459 0.0038249362  0.172122129
## AA-OO  -2.0305868 0.0424142975  1.908643388
```

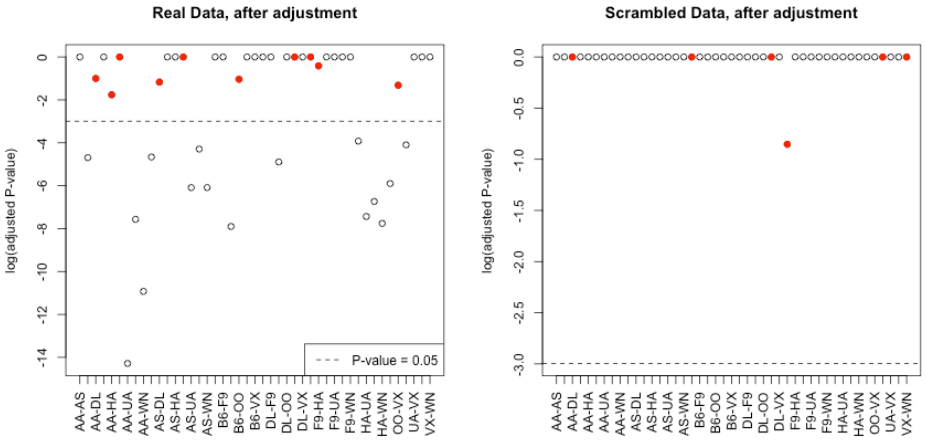
```
##       statistic.t    p.value p.value.adj
## AA-AS  -1.3008280 0.19388985    8.725043
## AA-B6   0.5208849 0.60264423   27.118990
## AA-DL  -2.0270773 0.04281676    1.926754
## AA-F9  -0.1804245 0.85698355   38.564260
## AA-HA  -1.5553127 0.13030058    5.863526
## AA-OO  -1.3227495 0.18607903    8.373556
```

Notice some of these p-values are greater than 1! So in fact, we want to multiply by K, unless the value is greater than 1, in which case we set the p-value to be 1.

$$\text{Bonferroni adjusted p-values} = min(p \times K, 1)$$

```
##       statistic.t      p.value  p.value.adj p.value.adj.final
## AA-AS   1.1514752 0.2501337691 11.256019611       1.000000000
## AA-B6  -3.7413418 0.0002038769  0.009174458       0.009174458
## AA-DL  -2.6480549 0.0081705864  0.367676386       0.367676386
## AA-F9  -0.3894014 0.6974223534 31.384005904       1.000000000
## AA-HA   3.1016459 0.0038249362  0.172122129       0.172122129
## AA-OO  -2.0305868 0.0424142975  1.908643388       1.000000000
```

Now we plot these adjusted values, for both the real data and the data I created by randomly scrambling the labels. I've colored in red those tests that become non-significant after the multiple testing correction.



%%%%%%%%%%%%%%%%%%%%%%%%% %%%%%%%%%%%%%%%%%%%%%%%%% %%%%%%%%%%%%%%%%%%%%%%%%% %%%%%%%%%%%%%%%%%%%%%%%%% %%%%%%%%%%%%%%%%%%%%%%%%%

3.6 Confidence Intervals

Another approach to inference is with confidence intervals. Confidence intervals give a range of values (based on the data) that are most likely to overlap the true parameter. This means confidence intervals are only appropriate when we are focused on estimation of a specific numeric feature of a distribution (a parameter of the distribution), though they do *not* have to require parametric models to do so.[11]

Form of a confidence interval

[11] We can test a null hypothesis without having a specific parameter of interest that we are estimating. For example, the Chi-squared test that you may have seen in an introductory statistic class tests whether two discrete distributions are independent, but there is no single parameter that we are estimating.

Confidence intervals also do not rely on a specific null hypothesis; instead they give a range of values (based on the data) that are most likely to overlap the true parameter. Confidence intervals take the form of an interval, and are paired with a confidence, like 95% confidence intervals, or 99% confidence intervals.

Question: Which should should result in wider intervals, a 95% or 99% interval?

General definition of a Confidence interval

A 95% confidence interval for a parameter θ is a interval (V_1, V_2) so that

$$P(V_1 \leq \theta \leq V_2) = 0.95.$$

Notice that this equation *looks* like θ should be the random quantity, but θ is a fixed (and unknown) value. The random values in this equation are actually the V_1 and V_2 – those are the numbers we estimate from the data. It can be useful to consider this equation as actually,

$$P(V_1 \leq \theta \text{ and } V_2 \geq \theta) = 0.95,$$

to emphasize that V_1 and V_2 are the random variables in this equation.

3.6.1 Quantiles

Without even going further, it's clear we're going to be inverting our probability calculations, i.e. finding values that give us specific probabilities. For example, you should know that for X distributed as a normal distribution, the probability of X being within about 2 standard deviations of the mean is 0.95 – more precisely 1.96 standard deviations.

Figuring out what number will give you a certain probability of being less than (or greater than) that value is a question of finding a **quantile** of the distribution. Specifically, quantiles tell you at what point you will have a particular probability of being less than that value. Precisely, if z is the α quantile of a distribution, then

$$P(X \leq z) = \alpha.$$

We will often write z_α for the α quantile of a distribution.

So if X is distributed as a normal distribution and z is a 0.25 quantile of a normal distribution,

$$P(X \leq z) = 0.25.$$

z is a 0.90 quantile of a normal if $P(X \leq z) = 0.90$, and so forth

These numbers can be looked up easily in R for standard distributions.

```
qnorm(0.2, mean = 0, sd = 1)
```

```
## [1] -0.8416212
```

```
qnorm(0.9, mean = 0, sd = 1)
```

```
## [1] 1.281552
```

```
qnorm(0.0275, mean = 0, sd = 1)
```

```
## [1] -1.918876
```

Question: What is the probability of being between -0.84 and 1.2815516 in a $N(0,1)$?

3.7 Parametric Confidence Intervals

This time we will start with using parametric models to create confidence intervals. We will start with how to construct a parametric CI for the mean of single group.

3.7.1 Confidence Interval for Mean of One group

As we've discussed many times, a SRS will have a sampling distribution that is roughly a normal distribution (the Central Limit Theorem). Namely, that if $X_1, \ldots, X_n$ are a i.i.d from a distribution with mean μ and variance σ^2, then $\hat{\mu} = \bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$ will have a roughly normal distribution

$$N(\mu, \frac{\sigma^2}{n}).$$

Let's assume we know σ^2 for now. Then a 95% confidence interval can be constructed by

$$\bar{X} \pm 1.96\frac{\sigma}{\sqrt{n}}$$

More generally, we can write this as

$$\bar{X} \pm zSD(\bar{X})$$

Where did $z = 1.96$ come from?

Note for a r.v. $Y \sim N(\mu, \sigma^2)$ distribution, the value $\mu - 1.96\sqrt{\sigma^2}$ is the 0.025 quantile of the distribution, and $\mu + 1.96\sqrt{\sigma^2}$ is the 0.975 quantile of the distribution, so the probability of Y being between these two values is 0.95. By the

CLT we'll assume $\bar{X} \sim N(\mu, \frac{\sigma^2}{n})$, so the the probability that $\bar{X}$ is within

$$\mu \pm 1.96\sqrt{\sigma^2}$$

is 95%. So it looks like we are just estimating μ with $\bar{X}$.

That isn't quite accurate. What we are saying is that

$$P(\mu - 1.96\sqrt{\frac{\sigma^2}{n}} \leq \bar{X} \leq \mu + 1.96\sqrt{\frac{\sigma^2}{n}}) = 0.95$$

and we really need is to show that

$$P(\bar{X} - 1.96\sqrt{\frac{\sigma^2}{n}} \leq \mu \leq \bar{X} + 1.96\sqrt{\frac{\sigma^2}{n}}) = 0.95$$

to have a true 0.95 confidence interval. But we're almost there.

We can invert our equation above, to get

$$\begin{aligned}
0.95 &= P(\mu - 1.96\sqrt{\frac{\sigma^2}{n}} \leq \bar{X} \leq \mu + 1.96\sqrt{\frac{\sigma^2}{n}}) \\
&= P(-1.96\sqrt{\frac{\sigma^2}{n}} \leq \bar{X} - \mu \leq 1.96\sqrt{\frac{\sigma^2}{n}}) \\
&= P(-1.96\sqrt{\frac{\sigma^2}{n}} - \bar{X} \leq -\mu \leq 1.96\sqrt{\frac{\sigma^2}{n}} - \bar{X}) \\
&= P(1.96\sqrt{\frac{\sigma^2}{n}} + \bar{X} \geq \mu \geq -1.96\sqrt{\frac{\sigma^2}{n}} + \bar{X}) \\
&= P(\bar{X} - 1.96\sqrt{\frac{\sigma^2}{n}} \leq \mu \leq \bar{X} + 1.96\sqrt{\frac{\sigma^2}{n}})
\end{aligned}$$

General equation for CI

Of course, we can do the same thing for any confidence level we want. If we want a $(1-\alpha)$ level confidence interval, then we take

$$\bar{X} \pm z_{\alpha/2} SD(\bar{X})$$

Where $z_{\alpha/2}$ is the $\alpha/2$ quantile of the $N(0,1)$.

In practice, we do not know σ so we don't know $SD(\bar{X})$ and have to use $\hat{\sigma}$, which mean that we need to use the quantiles of a t-distribution with $n-1$ degrees of freedom for smaller sample sizes.

Example in R

For the flight data, we can get a confidence interval for the mean of the United flights using the function `t.test` again. We will work on the log-scale, since we've already seen that makes more sense for parametric tests because our data is skewed:

```
t.test(log(flightSFOSRS$DepDelay[flightSFOSRS$Carrier ==
    "UA"] + addValue))
```

```
##
##  One Sample t-test
##
## data:  log(flightSFOSRS$DepDelay[flightSFOSRS$Carrier == "UA"] + addValue)
## t = 289.15, df = 2964, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  3.236722 3.280920
## sample estimates:
## mean of x
##  3.258821
```

Notice the result is on the (shifted) log scale! Because this is a monotonic function, we can invert this to see what this implies on the original scale:

```
logT <- t.test(log(flightSFOSRS$DepDelay[flightSFOSRS$Carrier ==
    "UA"] + addValue))
exp(logT$conf.int) - addValue
```

```
## [1] 3.450158 4.600224
## attr(,"conf.level")
## [1] 0.95
```

3.7.2 Confidence Interval for Difference in the Means of Two Groups

Now lets consider the average delay time between the two airlines. Then the parameter of interest is the difference in the means:

$$\delta = \mu_{United} - \mu_{American}.$$

Using the central limit theorem again,

$$\bar{X} - \bar{Y} \sim N(\mu_1 - \mu_2, \frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2})$$

You can do the same thing for two groups in terms of finding the confidence interval:

$$P((\bar{X} - \bar{Y}) - 1.96\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}} \leq \mu_1 - \mu_2 \leq (\bar{X} - \bar{Y}) + 1.96\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}) = 0.95$$

Then a 95% confidence interval for $\mu_1 - \mu_2$ if we knew σ_1^2 and σ_2^2 is

$$\bar{X} - \bar{Y} \pm 1.96\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}$$

Estimating the variance

Of course, we don't know σ_1^2 and σ_2^2, so we will estimate them, as with the t-statistic. We know from our t-test that if $X_1, \ldots, X_{n_1}$ and $Y_1, \ldots, Y_{n_2}$ are normally distributed, then our t-statistic,

$$T = \frac{|\bar{X} - \bar{Y}|}{\sqrt{\frac{\hat{\sigma_1}^2}{n_1} + \frac{\hat{\sigma_2}^2}{n_2}}}.$$

has actually a t-distribution.

How does this get a confidence interval (T is not an estimate of δ)? We can use the same logic of inverting the equations, only with the quantiles of the t-distribution to get a confidence interval for the difference.

Let $t_{0.025}$ and $t_{0.975}$ be the quantiles of the t distribution. Then,

$$P((\bar{X} - \bar{Y}) - t_{0.975}\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}} \leq \mu_1 - \mu_2 \leq (\bar{X} - \bar{Y}) - t_{0.025}\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}) = 0.95$$

Of course, since the t distribution is symmetric, $-t_{0.025} = t_{0.975}$.

Question: Why does symmetry imply that $-t_{0.025} = t_{0.975}$?

We've already seen that for reasonably moderate sample sizes, the difference between the normal and the t-distribution is not that great, so that in most cases it is reasonable to use the normal-based confidence intervals only with $\hat{\sigma}_1^2$ and $\hat{\sigma}_2^2$. This is why ± 2 standard errors is such a common mantra for reporting estimates.

2-group test in R

We can get the confidence interval for the difference in our groups using `t.test` as well.

```
logUA <- log(flightSFOSRS$DepDelay[flightSFOSRS$Carrier ==
    "UA"] + addValue)
logAA <- log(flightSFOSRS$DepDelay[flightSFOSRS$Carrier ==
    "AA"] + addValue)
t.test(logUA, logAA)
```

```
##
##  Welch Two Sample t-test
##
## data:  logUA and logAA
## t = 5.7011, df = 1800.7, p-value = 1.389e-08
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.07952358 0.16293414
## sample estimates:
## mean of x mean of y
##  3.258821  3.137592
```

Question: What is the problem from this confidence interval on the log-scale that we didn't have before when we were looking at a single group?

3.8 Bootstrap Confidence Intervals

The Setup

Suppose we are interested instead in whether the median of the two groups is the same.

Question: Why might that be a better idea than the mean?

Or, alternatively, as we saw, perhaps a more relevant statistic than either the mean or the median would be the difference in the proportion greater than 15 minutes late. Let θ_{United}, and $\theta_{American}$ be the true proportions of the two groups, and now

$$\delta = \theta_{United} - \theta_{American}.$$

Question: The sample statistic estimating δ would be what?

To be able to do hypothesis testing on other statistics, we need the distribution of our test statistic to either construct confidence intervals or the p-value. In the t-test, we used the central limit theorem that tells us the difference in the means is approximately normal. We can't use the CLT theory for the median, however, because the CLT was for the difference in the means of the groups. We would need to have new mathematical theory for the difference in the medians or proportions. In fact such theory exists (and the proportion is actually a type of mean, so we can in fact basically use the t-test, which

some modifications). Therefore, many other statistics can also be handled with parametric tests as well, but each one requires a new mathematical theory to get the null distribution. More importantly, when you go with statistics that are beyond the mean, the mathematics often require more assumptions about the data-generating distribution – the central limit theorem for the mean works for most any distribution you can imagine (with large enough sample size), but that's a special property of the mean. Furthermore, if you have an uncommon statistic, the mathematical theory for the statistic may not exist.

Another approach is to try to estimate the distribution of our test-statistic from our data. This is what the bootstrap does. We've talked about estimating the distribution of our data in Chapter 2, but notice that estimating the distribution of our *data* is a different question than estimating the distribution of a *summary statistic*. If our data is i.i.d. from the same distribution, then we have n observations from the distribution from which to estimate the data distribution. For our test statistic (e.g. the median or mean) we have only 1 value. We can't estimate a distribution from a single value!

The bootstrap is a clever way of estimating the distribution of most any statistic from the data.

3.8.1 The Main Idea: Create many datasets

Let's step back to some first principles. Recall for a confidence interval based on my statistic δ, I would like to find numbers w_1 and w_2 so that

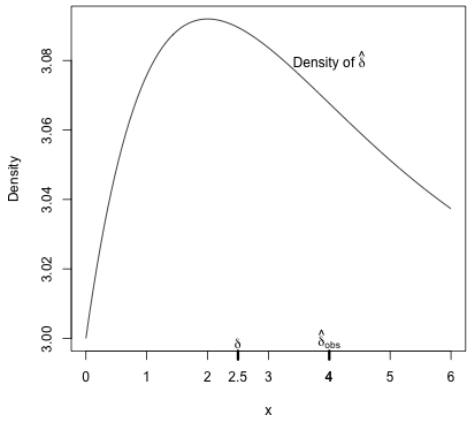
$$0.95 = P(\hat{\delta} - w_1 \leq \delta \leq \hat{\delta} + w_2)$$

so that I would have a CI $(V_1 = \hat{\delta} - w_1, V_2 = \hat{\delta} + w_2)$.

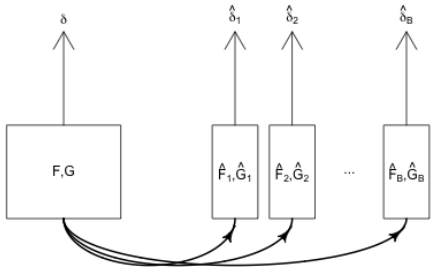
Note that this is the same as

$$P(\delta - w_2 \leq \hat{\delta} \leq \delta + w_1)$$

In other words, if I knew $\hat{\delta}$ had following distribution/density, I could find quantiles that gave me my w_1,w_2.



In order to get the distribution of $\hat{\delta}$, we would like to be able to do is collect multiple data sets, and for each data set calculate our $\hat{\delta}$. This would give us a collection of $\hat{\delta}$ from which we could estimate the distribution of $\hat{\delta}$. More formally, if we knew F, G – the distributions of the data in each group – we could simulate datasets from each distribution, calculate $\hat{\delta}$, and repeat this over and over. From each of these multiple datasets, we would calculate $\hat{\delta}$, which would give us a distribution of $\hat{\delta}$. This process is demonstrated in this figure:



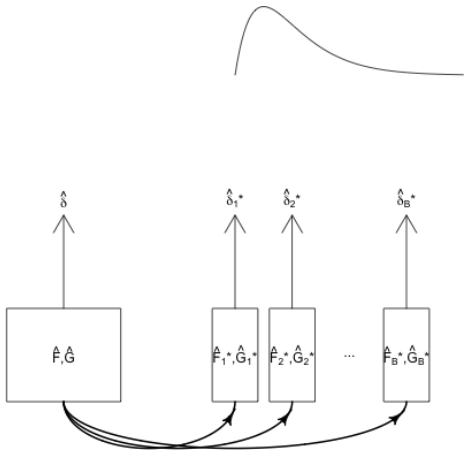
But since we have only one data set, we only see one $\hat{\delta}$, so none of this is an option.

What are our options? We've seen one option is to use parametric methods, where the distribution of $\hat{\delta}$ is determined mathematically (but is dependent on our statistic δ and often with assumptions about the distributions F and G). The other option we will discuss, the bootstrap, tries instead to create lots of datasets with the computer.

The idea of the bootstrap is if we can estimate the distributions F and G with $\hat{F}$ and $\hat{G}$, we can create new data sets by simulating data from $\hat{F}$ and $\hat{G}$. So we

can do our ideal process described above, only without the true F and G, but with an estimate of them. In other words, while what we need is the distribution of $\hat{\delta}$ from many datasets from F, G, instead we will create many datasets from $\hat{F}, \hat{G}$ as an approximation.

Here is a visual of how we are trying to replicate the process with our bootstrap samples:



How can we estimate $\hat{F}, \hat{G}$? Well, that's what we've discussed in the Chapter 2. Specifically, when we have a i.i.d sample, Chapter 2 went over methods of estimating the unknown true distribution F, and estimating probabilities from F. What we need here is a simpler question – how to draw a sample from an estimate of $\hat{F}$, which we will discuss next.

Assume we get a i.i.d. sample from F and G. We've previously estimated for example, the density of our distribution (which then defines the entire distribution). But right now, we really need to be able to draw a random sample $X_1, \ldots, X_{n_1}$ and $Y_1, \ldots, Y_{n_2}$ from our estimates of the distribution so we can calculate a δ. Our density estimate doesn't give us a way to do that.

So we are going to think how can we get an estimate of F and G that we can get a sample from? We've seen how a full census of our data implies a distribution, if we assume random independent draws from that full data. In other words, a data set combined with a mechanism for drawing samples defines a distribution, with probabilities, etc. So our observed sample gives us an estimated distribution (also called the **empirical distribution**) $\hat{F}$ and $\hat{G}$ from which we can draw samples.

Question: How would you make a i.i.d. sample from your data?

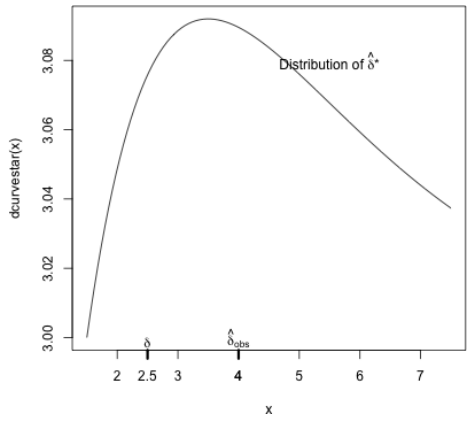
Let $X_1^*, \ldots, X_{n_1}^*$ be i.i.d. draws from my observed data. This is called a **boot-**

strap sample. I can calculate δ from this data, call it $\hat{\delta}^*$. Unlike the $\hat{\delta}$ from my actual data, I can repeat the process and calculate many $\hat{\delta}^*$ values. If I do this enough I have a distribution of $\hat{\delta}^*$ values, which we will call the **bootstrap distribution** of δ. So the bootstrap distribution of $\hat{\delta}$ is the distribution of $\hat{\delta}$ if the data was drawn from $\hat{F}$ and $\hat{G}$ distribution.

Another funny thing is that $\hat{\delta}^*$ is an estimate of $\hat{\delta}$, i.e. if I didn't know $\hat{F}, \hat{G}$ and only saw the bootstrap sample $X_1^*, \ldots, X_{n_1}^*$ and $Y_1^*, \ldots, Y_{n_2}^*$, $\hat{\delta}^*$ is an estimate of $\hat{\delta}$. Of course I don't need to estimate $\hat{\delta}$ – I know them from my data! But my bootstrap sample can give me an idea of how good of an estimate I can expect $\hat{\delta}$ to be. If the distribution of $\hat{\delta}^*$ shows that we are likely to get estimates of $\hat{\delta}$ is far from $\hat{\delta}$, then it is likely that $\hat{\delta}$ is similarly far from the unknown δ. It's when we have simulated data to see what to expect the behavior of our statistic compared to the truth (only now our observed data and $\hat{\delta}$ are our "truth").

Back to Confidence Intervals

If draw many bootstrap samples, I can get the following distribution of $\hat{\delta}^*$ (centered now at $\hat{\delta}$!):



So $\hat{\delta}^*$ is not a direct estimate of the distribution of $\hat{\delta}$! The true distribution of $\hat{\delta}$ should be centered at δ (and we know $\hat{\delta} \neq \delta$ because of randomness). So the bootstrap distribution is NOT the distribution of $\hat{\delta}$

But if the distribution of $\hat{\delta}^*$ around $\hat{\delta}$ is like that of $\hat{\delta}$ around δ, then that gives me useful information about how likely it is that my $\hat{\delta}$ is far away from the true δ, e.g.

$$P(|\hat{\delta} - \delta| > 1) \approx P(|\hat{\delta}^* - \hat{\delta}| > 1)$$

Or more relevant for a confidence interval, I could find W_1^* and W_2^* so that

$$0.95 = P(\hat{\delta} - W_2^* \leq \hat{\delta}^* \leq \hat{\delta} + W_1^*)$$

Once I found those values, I could use the same W_1^*, W_2^* to approximate that

$$0.95 \approx P(\delta - W_2^* \leq \hat{\delta} \leq \delta + W_1^*) = P(\hat{\delta} - W_1^* \leq \delta \leq \hat{\delta} + W_2^*)$$

This gives us a confidence interval $(\hat{\delta}-W_1^*, \hat{\delta}+W_2^*)$ which is called the **bootstrap confidence interval** for δ.

In short, we don't need that $\hat{\delta}^*$ approximates the distribution of $\hat{\delta}$. We just want that the distance of $\hat{\delta}^*$ from it's true generating value $\hat{\delta}$ replicate the distance of $\hat{\delta}$ from the (unknown) true generating value δ.

3.8.2 Implementing the bootstrap confidence intervals

What does it actually mean to resample from $\hat{F}$? It means to take a sample from $\hat{F}$ just like the kind of sample we took from the actual data generating process, F.

Specifically in our two group setting, say we assume we have a i.i.d sample $X_1, \ldots, X_{n_1}, Y_1, \ldots, Y_{n_2}$ from an unknown distributions F and G.

Question: What does this actually mean? Consider our airline data; if we took the full population of airline data, what are we doing to create a i.i.d sample?

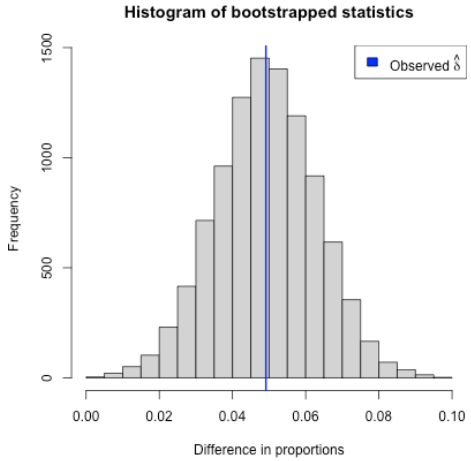
Then to recreate this we need to do *the exact same thing*, only from our sample. Specifically, we resample *with replacement* to get a single bootstrap sample *of the same size* consisting of new set of samples, $X_1^*, \ldots, X_{n_1}^*$ and $Y_1^*, \ldots, Y_{n_2}^*$. Every value of X_i^* and Y_i^* that I see in the bootstrap sample will be a value in my original data.

Question: Moreover, some values of my data I will see more than once, why?

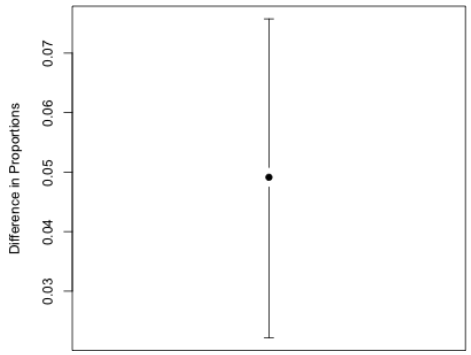
From this single bootstrap sample, we can recalculate the difference of the medians on this sample to get $\hat{\delta}^*$.

We do this repeatedly, and get a distribution of $\hat{\delta}^*$; specifically if we repeat this B times, we will get $\hat{\delta}_1^*, \ldots, \hat{\delta}_B^*$. So we will now have a distribution of values for $\hat{\delta}^*$.

We can apply this function to the flight data, and examine our distribution of $\hat{\delta}^*$.



To construct a confidence interval, we use the 0.025 and 0.975 quantiles as the limits of the 95% confidence interval.[12] We apply it to our flight data set to get a confidence interval for the difference in proportion of late or cancelled flights.



Question: How do you interpret this confidence interval?

3.8.3 Assumptions: Bootstrap

Assumption: Good estimates of $\hat{F}$, $\hat{G}$

A big assumption of the bootstrap is that our sample distribution $\hat{F}, \hat{G}$ is a good estimate of F and G. We've already seen that will not necessarily the case.

[12] There are many different strategies for calculating a bootstrap CI from the distribution of $\hat{\delta}^*$; this method called the **percentile method** and is the most common and widespread. It doesn't exactly correspond to the v_1, v_2 strategy from above – known as using a pivotal statistic. If it looks like the v_1, v_2 method is backward compared to the percentile method, it pretty much is! But both methods are legitimate methods for creating bootstrap intervals and we focus on the percentile method because of it's simplicity and wider applicability.

Here are some examples of why that might fail:

- Sample size n_1/n_2 is too small
- The data is not i.i.d sample (or SRS)

Assumption: Data generation process

Another assumption is that our method of generating our data X_i^*, and Y_i^* matches the way X_i and Y_i were generated from F, G. In particular, in the bootstrap procedure above, we are assuming that X_i and Y_i are i.i.d from F and G (i.e. a SRS with replacement).

Assumption: Well-behaved test statistic

We also need that the parameter θ and the estimate $\hat{\theta}$ to be well behaved in certain ways

- $\hat{\theta}$ needs to be an **unbiased** estimate of θ, meaning across many samples, the average of the $\hat{\theta}$ is equal to the true parameter θ [13]
- θ is a function of F and G, and we need that the value of θ changes smoothly as we change F and G. In other words if we changed from F to F', then θ would change to θ'; we want if our new F' is very "close" to F, then our new θ' would be very close to θ. This is a pretty mathematical requirement, and requires a precise definition of "close" for two distributions that is not too important for this class to understand.

But here's an example to make it somewhat concrete: if the parameter θ you are interested in is the maximum possible value of a distribution F, then θ does NOT change smoothly with F. Why? because you can choose distributions F' that are very close to F in every reasonable way to compare two distributions, but their maximum values θ and θ' are very far apart.[14]

Another bootstrap confidence interval (Optional)

We can actually use the bootstrap to calculate a confidence interval similarly to that of the normal distribution based on estimating the distribution of $\hat{\delta} - \delta$.

Notice with the previous calculation for $\bar{X}$, if I know

$$0.95 = P(1.96\sqrt{\frac{\sigma^2}{n}} \leq \bar{X} - \mu \leq 1.96\sqrt{\frac{\sigma^2}{n}})$$

Then I can invert to get

$$0.95 = P(\bar{X} - 1.96\sqrt{\frac{\sigma^2}{n}} \leq \mu \leq \bar{X} - 1.96\sqrt{\frac{\sigma^2}{n}})$$

[13] There are methods for accounting for a small amount of bias with the bootstrap, but if the statistic is wildly biased away from the truth, then the bootstrap will not work.

[14] This clearly assumes what is a "reasonable" definition of "close" between distributions that we won't go into right now.

So more generally, suppose we have points $z_{0.025}$ and $z_{0.975}$ so that

$$0.95 = P(z_{0.025} \leq \hat{\delta} - \delta \leq z_{0.975})$$

e.g. the 0.025 and 0.975 quantiles of $\hat{\delta} - \delta$. Then I can invert to get

$$0.95 = P(\hat{\delta} - z_{0.975} \leq \delta \leq \hat{\delta} - z_{0.025})$$

So if I can get the quantiles of $\hat{\delta} - \delta$, I can make a confidence interval.

So we could use the bootstrap to get estimates of the distribution of $\hat{\delta} - \delta$ instead of the distribution of $\hat{\delta}$ and use the quantiles of $\hat{\delta} - \delta$ to get confidence intervals that are $(\hat{\delta} - z_{0.975}, \hat{\delta} - z_{0.025})$. This actually gives a different confidence interval, particularly if the distribution of $\hat{\delta}$ is not symmetric. The earlier method we talked about is called the percentile method, and is most commonly used, partly because it's easier to generalize than this method.[15]

3.9 Thinking about confidence intervals

Suppose you have a 95% confidence interval for δ given by $(.02, .07)$.

Question: What is wrong with the following statements regarding this confidence interval? - δ has a 0.95 probability of being between $(.02, .07)$ - If you repeatedly resampled the data, the difference δ would be within $(.02, .07)$ 95% of the time.

Confidence Intervals or Hypothesis Testing?

Bootstrap inference via confidence intervals is more widely applicable than permutation tests we described above. The permutation test relied on being able to simulate from the null hypothesis, by using the fact that if you detach the data from their labels you can use resampling techniques to generate a null distribution. In settings that are more complicated than comparing groups, it can be difficult to find this kind of trick.

Frequently, confidence intervals and hypothesis testing are actually closely intertwined, particularly for parametric methods. For example, for the parametric test and the parametric confidence interval, they both relied on the distribution of the same statistic, the t-statistic. If you create a 95% confidence interval, and then decide to reject a specific null hypothesis (e.g. $H_0 : \delta = 0$) only when it does not fall within the confidence interval, then this will exactly correspond to

[15] If it looks like this method is backward compared to the percentile method, it pretty much is! But both methods are legitimate methods for creating bootstrap intervals.

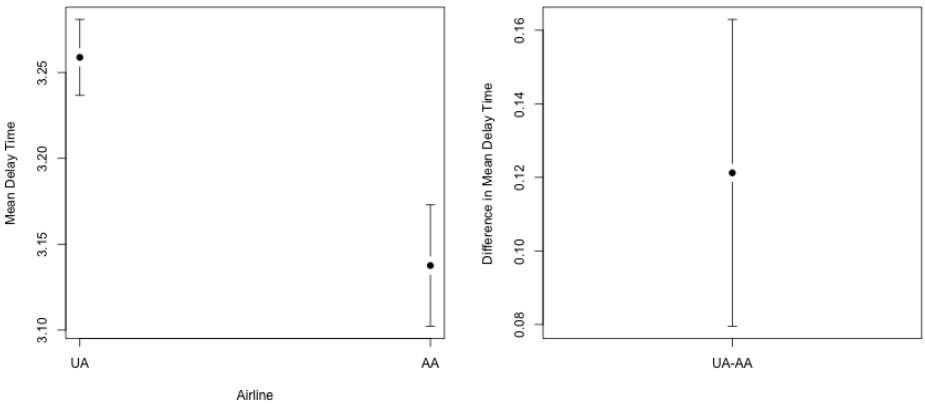
a test with level 0.05. So the same notions of level, and type I error, also apply to confidence intervals

Confidence intervals, on the other hand, give much greater interpretation and understanding about the parameter.

3.9.1 Comparing Means: CI of means vs CI of difference

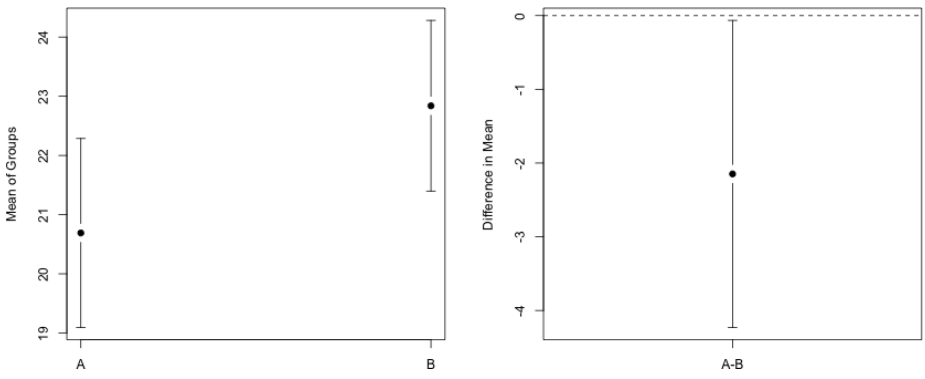
We have focused on creating a confidence interval of the difference (δ). Another common strategy is to do a confidence interval of each mean, and compare them.

We can compare these two options using the t-statistic:



We see that their confidence intervals don't overlap, and that the CI for the difference in the means doesn't overlap zero, so we draw the same conclusion in our comparison, namely that the means are different.

However, this doesn't have to be the case. Here's some made-up data[16]:



[16]From https://statisticsbyjim.com/hypothesis-testing/confidence-intervals-compare-means/

What to think here? What is the right conclusion? The confidence interval for the difference for the means corresponds to the test for the difference of the means, which means that if the CI for δ doesn't cover zero, then the corresponding p-value from the t-test will be < 0.05. So this is the "right" confidence interval for determining statistical significance.

Why does this happen?

Basically, with the t-test-based CI, we can examine this analytically (a big advantage of parametric models).

In the first case, for a CI of the difference δ to be significantly larger than zero, it means that the lower end of the CI for delta is greater than zero:

$$\bar{X} - \bar{Y} > 1.96\sqrt{\frac{\hat{\sigma}_1^2}{n_1} + \frac{\hat{\sigma}_2^2}{n_2}}$$

Alternatively, if we create the two confidence intervals for $\bar{X}$ and $\bar{Y}$, separately, to have them not overlap, we need that the lower end of the CI for X be greater than the upper end of the CI of Y:

$$\begin{aligned}
\bar{X} - 1.96\sqrt{\frac{\hat{\sigma}_1^2}{n_1}} &> \bar{Y} + 1.96\sqrt{\frac{\hat{\sigma}_2^2}{n_2}} \\
\bar{X} - \bar{Y} &> 1.96\left(\sqrt{\frac{\hat{\sigma}_2^2}{n_2}} + \sqrt{\frac{\hat{\sigma}_1^2}{n_1}}\right)
\end{aligned}$$

Note that these are not the same requirements. In particular,

$$\sqrt{\frac{\hat{\sigma}_1^2}{n_1} + \frac{\hat{\sigma}_2^2}{n_2}} < \left(\sqrt{\frac{\hat{\sigma}_2^2}{n_2}} + \sqrt{\frac{\hat{\sigma}_1^2}{n_1}}\right)$$

(take the square of both sides…).

So that means that the difference of the means doesn't have to be as big for CI based for δ to see the difference as for comparing the individual mean's CI. We know that the CI for δ is equivalent to a hypothesis test, so that means that IF there is a difference between the individual CI means there is a significant difference between the groups, but the converse is NOT true: there could be significant differences between the means of the groups but the CI of the individual means are overlapping.

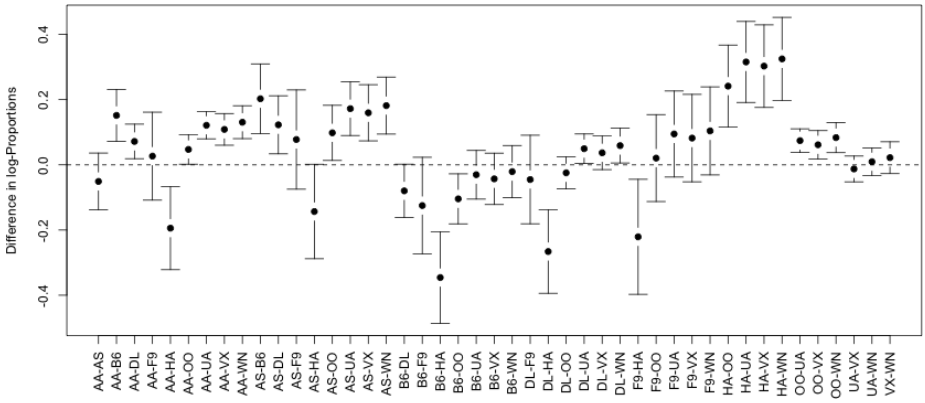
Reality Check

However, note that the actual difference between the two groups in our toy example is pretty small and our significance is pretty marginal. So it's not such a big difference in our conclusions after all.

3.10 Revisiting pairwise comparisons

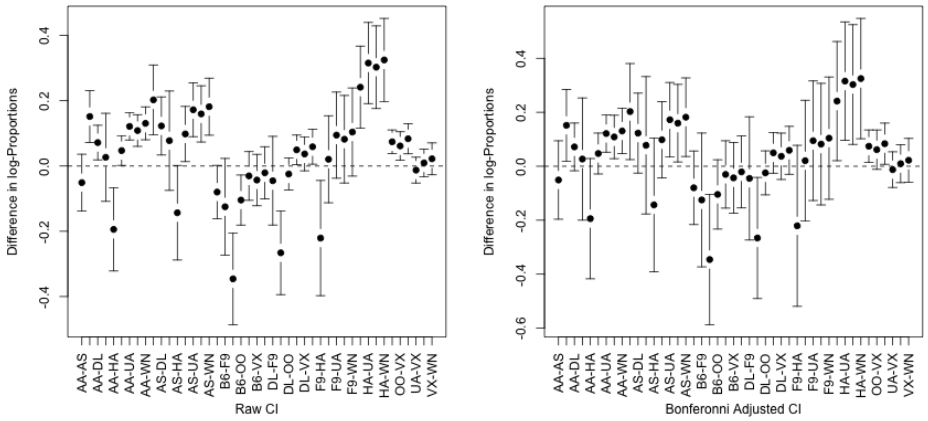
Just as with hypothesis testing, you can have multiple comparison problems with confidence intervals. Consider our pairwise comparisons of the different carriers. We can also create confidence intervals for them all. Again, we will use the t-test on the log-differences to make this go quickly.

```
##       mean.of.x mean.of.y        lower       upper
## AA-AS  3.086589  3.137592 -0.138045593  0.03603950
## AA-B6  3.289174  3.137592  0.071983930  0.23118020
## AA-DL  3.209319  3.137592  0.018600177  0.12485342
## AA-F9  3.164201  3.137592 -0.108192832  0.16141032
## AA-HA  2.943335  3.137592 -0.321473062 -0.06704092
## AA-OO  3.184732  3.137592  0.001615038  0.09266604
```



These confidence intervals suffer from the same problem as the p-values: even if the null value (0) is true in every test, roughly 5% of them will happen to not cover 0 just by chance.

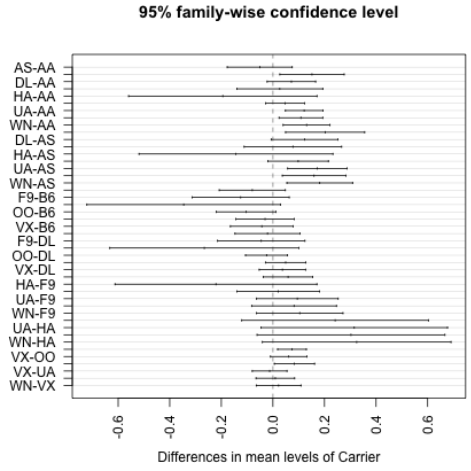
So we can do bonferonni corrections to the confidence intervals. Since a 95% confidence interval corresponds to a level 0.05 test, if we go to a $0.05/K$ level, which is the bonferonni correction, that corresponds to a $100 * (1 - 0.05/K)\%$ confidence interval.



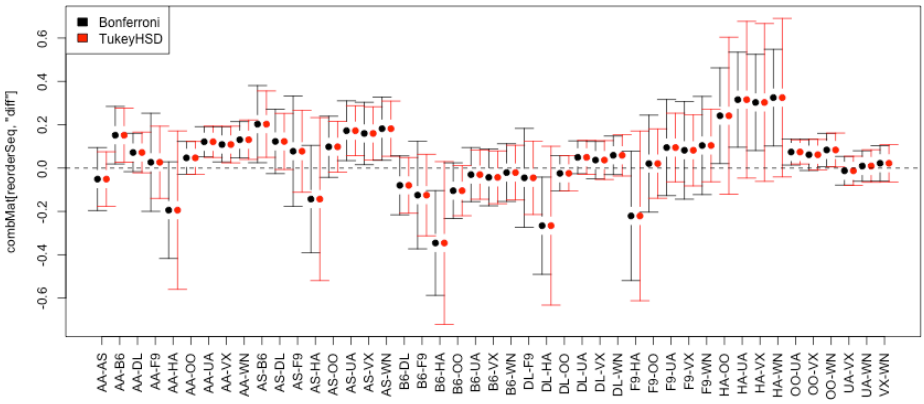
TukeyHSD

In fact, as mentioned, there are many ways to do multiple testing corrections, and Bonferonni is the simplest, yet often most crude correction. There is a multiple testing correction just for pairwise comparisons that use the t-test, called the Tukey HSD test.

```
tukeyCI <- TukeyHSD(aov(logDepDelay ~ Carrier, data = flightSFOSRS))
plot(tukeyCI, las = 2)
```



Let's compare them side-by-side.



Question: What differences do you see?

Which to use?

The TukeyHSD is a very specific correction – it is only valid for doing pairwise comparisons with the t-test. Bonferonni, on the other hand, can be used with any set of p-values from any test, e.g. permutation, and even if not all of the tests are pairwise comparisons.

Chapter 4

Curve Fitting

```
## Linking to ImageMagick 6.9.12.3
## Enabled features: cairo, fontconfig, freetype, heic, lcms, pango, raw, rsvg, webp
## Disabled features: fftw, ghostscript, x11
```

Comparing groups evaluates how a **continuous variable** (often called the response or independent variable) is related to a **categorical variable**. In our flight example, the continuous variable is the flight delay and the categorical variable is which airline carrier was responsible for the flight. We asked how did the flight delay differ from one group to the next?

However, we obviously don't want to be constrained to just categorical variables. We want to be able to ask how a continuous variable affects another continuous variable – for example how does flight delay differ based on the time of day?

In this chapter, we will turn to relating two continuous variables. We will review the method that you've learned already – simple linear regression – and briefly discuss inference in this scenario. Then we will turn to expanding these ideas for more flexible curves than just a line.

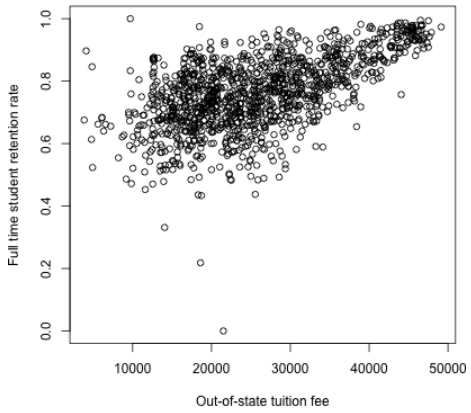
4.1 Linear regression with one predictor

An example dataset

Let's consider the following data collected by the Department of Education regarding undergraduate institutions in the 2013-14 academic year ((https://catalog.data.gov/dataset/college-scorecard)). The department of education collects a great deal of data regarding the individual colleges/universities (including for-profit schools). Let's consider two variables, the tuition costs and the retention rate of students (percent that return after first year). We will exclude the

for-profit institutes (there aren't many in this particular data set), and focus on out-of-state tuition to make the values more comparable between private and public institutions.

```
dataDir <- "../finalDataSets"
scorecard <- read.csv(file.path(dataDir, "college.csv"),
    stringsAsFactors = FALSE)
scorecard <- scorecard[-which(scorecard$CONTROL ==
    3), ]
xlab = "Out-of-state tuition fee"
ylab = "Full time student retention rate"
plot(scorecard[, c("TUITIONFEE_OUT", "RET_FT4")], xlab = xlab,
    ylab = ylab)
```



Question: What do you observe in these relationships?

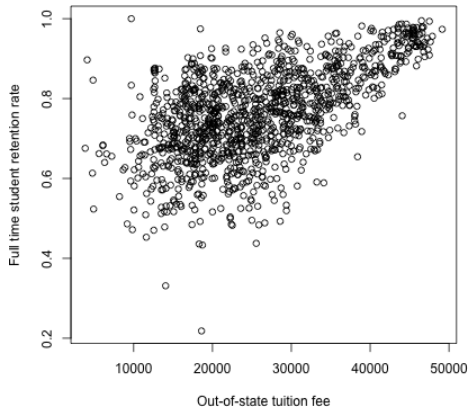
It's not clear what's going on with this observation with 0% of students returning after the first year, but a 0% return rate is an unlikely value for an accreditated institution and is highly likely to be an error. So for now we'll drop that value. This is not something we want to do lightly, and points to the importance of having some understanding of the data – knowing that *a priori* 0% is a suspect number, for example. But just looking at the plot, it's not particularly clear that 0% is any more "outlying" than other points; we're basing this on our knowledge that 0% of students returning after the first year seems quite surprising. If we look at the college (Pennsylvania College of Health Sciences), a google search shows that it changed it's name in 2013 which is a likely cause.

```
scorecard[scorecard[, "RET_FT4"] == 0, ]
```

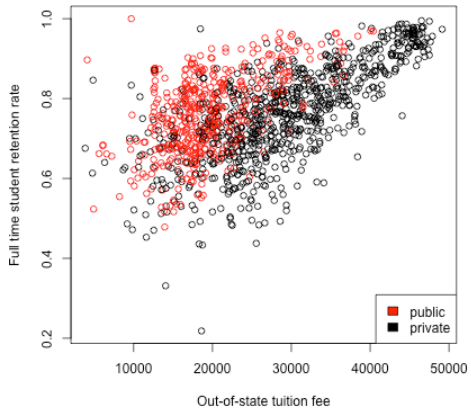
```
##         X                                  INSTNM STABBR ADM_RATE_ALL SATMTMID
## 1238 5930 Pennsylvania College of Health Sciences     PA          398      488
##      SATVRMID SAT_AVG_ALL AVGFACSAL TUITFTE TUITIONFEE_IN TUITIONFEE_OUT
```

```
## 1238      468         955      5728   13823         21502          21502
##      CONTROL UGDS UGDS_WHITE UGDS_BLACK UGDS_HISP UGDS_ASIAN UGDS_AIAN
## 1238       2 1394     0.8364     0.0445    0.0509     0.0294     7e-04
##      UGDS_NHPI UGDS_2MOR UGDS_NRA UGDS_UNKN  INC_PCT_LO  INC_PCT_M1  INC_PCT_M2
## 1238    0.0029    0.0014        0    0.0337 0.367788462 0.146634615 0.227163462
##       INC_PCT_H1  INC_PCT_H2 RET_FT4 PCTFLOAN C150_4 mn_earn_wne_p10
## 1238 0.175480769 0.082932692       0   0.6735 0.6338           53500
##      md_earn_wne_p10 PFTFAC
## 1238           53100 0.7564
```

```
scorecard <- scorecard[-which(scorecard[, "RET_FT4"] ==
    0), ]
plot(scorecard[, c("TUITIONFEE_OUT", "RET_FT4")], xlab = xlab,
    ylab = ylab)
```



Question: In the next plot, I do the same plot, but color the universities by whether they are private or not (red are public schools). How does that change your interpretation?



This highlights why it is very important to use more than one variable in trying to understand patterns or predict, which we will spend much more time on later in the course. But for now we are going to focus on one variable analysis, so lets make this a more valid exercise by just considering one or the other (public or private). We'll make two different datasets for this purpose, and we'll mainly just focus on private schools.

```
private <- subset(scorecard, CONTROL == 2)
public <- subset(scorecard, CONTROL == 1)
```

4.1.1 Estimating a Linear Model

These are convenient variables to consider the simplest relationship you can imagine for the two variables – a linear one:

$$y = \beta_0 + \beta_1 x$$

Of course, this assumes there is no noise, so instead, we often write

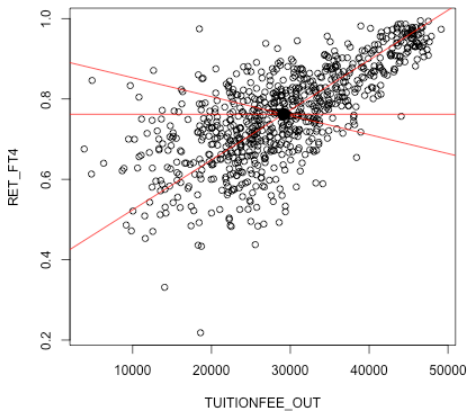
$$y = \beta_0 + \beta_1 x + e$$

where e represents some noise that gets added to the $\beta_0 + \beta_1 x$; e explains why the data do not exactly fall on a line.[1]

We do not know β_0 and β_1. They are parameters of the model. We want to estimate them from the data.

How to estimate the line

There are many possible lines, of course, even if we force them to go through the middle of the data (e.g. the mean of x,y). In the following plot, we superimpose a few "possible" lines for illustration, but any line is a potential line:



[1] It is useful to remember that *adding* noise is not the only option – this is a *choice* of a model.

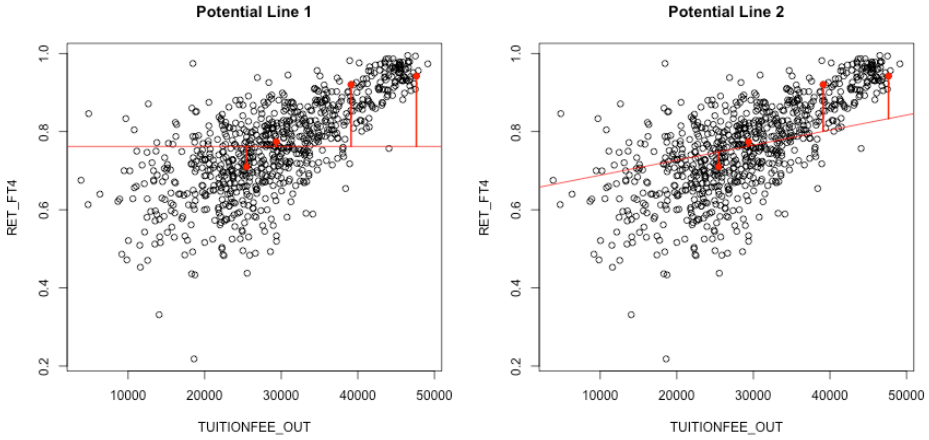
How do we decide which line is best? A reasonable choice is one that makes the smallest errors in predicting the response y. For each possible β_0, β_1 pair (i.e. each line), we can calculate the prediction from the line,

$$\hat{y}(\beta_0, \beta_1, x) = \beta_0 + \beta_1 x$$

and compare it to the actual observed y. Then we can say that the error in prediction for the point (x_i, y_i) is given by

$$y_i - \hat{y}(\beta_0, \beta_1, x_i)$$

We can imagine these errors visually on a couple of "potential" lines:



Of course, for any particular point (x_i, y_i), we can choose a β_0 and β_1 so that $\beta_0 + \beta_1 x_i$ is *exactly* y_i. But that would only be true for one point; we want to find a *single* line that seems "good" for all the points.

We need a measure of the **fit** of the line to all the data. We usually do this by taking the average error across all the points. This gives us a measure of the total amount of error for a possible line.

4.1.2 Choise of error (loss function)

Using our error from above (the difference of y_i and $\hat{y}_i$), would give us the average error of

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)$$

But notice that there's a problem with this. Our errors are allowed to cancel out, meaning a very large positive error coupled with a very large negative error cancel each other and result in no measured error! That's not a promising way to pick a line – we want every error to count. So we want to have a strictly positive measure of error so that errors will accumulate.

The choice of how to quantify the error (or loss) is called the **loss function**, $\ell(y, \hat{y}(\beta_0, \beta_1))$. There are two common choices for this problem

- **Absolute loss**
$$\ell(y_i, \hat{y}_i) = |y_i - \hat{y}_i(\beta_0, \beta_1)|$$
- **Squared-error loss**
$$\ell(y_i, \hat{y}_i) = (y_i - \hat{y}_i(\beta_0, \beta_1))^2$$

Then our overall fit is given by

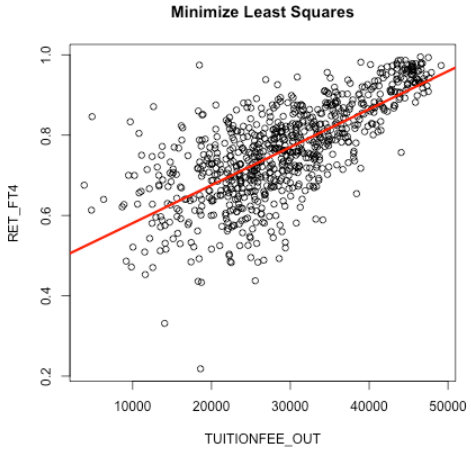
$$\frac{1}{n}\sum_{i=1}^{n} \ell(y_i, \hat{y}_i(\beta_0, \beta_1))$$

4.1.3 Squared-error loss

The most commonly used loss is squared-error loss, also known as **least squares regression**, where our measure of overall error for any particular β_0, β_1 is the average squared error,

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i(\beta_0, \beta_1))^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_i)^2$$

We can find the β_0 and β_1 that minimize the least-squared error, using the function `lm` in R. We call the values we find $\hat{\beta}_0$ and $\hat{\beta}_1$. These are *estimates* of the unknown β_0 and β_1. Below we draw the predicted line, i.e. the line we would get using the estimates $\hat{\beta}_0$ and $\hat{\beta}_1$:



Question: What do you notice about this line?

Calculating the least squares estimates in R

`lm` is the function that will find the least squares fit.

```
lm(RET_FT4 ~ TUITIONFEE_OUT, data = private)
```

```
##
## Call:
## lm(formula = RET_FT4 ~ TUITIONFEE_OUT, data = private)
##
## Coefficients:
##    (Intercept)  TUITIONFEE_OUT
##      4.863e-01       9.458e-06
```

Question:

1. How do you interpret these coefficients that are printed? What do they correspond to?
2. How much predicted increase in do you get for an increase of $10,000 in tuition?

Notice, as the below graphic from the Berkeley Statistics Department jokes, the goal is not to exactly fit any particular point, and our line might not actually go through any particular point.[2]

[2]The above graphic comes from the 1999 winner of the annual UC Berkeley Statistics department contest for tshirt designs

The estimates of β_0 and β_1

If we want, we can explicitly write down the equation for $\hat{\beta}_1$ and $\hat{\beta}_0$ (you don't need to memorize these equations)

$$\hat{\beta}_1 = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$
$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

Question: What do you notice about the denominator of $\hat{\beta}_1$?

The numerator is also an average, only now it's an average over values that involve the relationship of x and y. Basically, the numerator is large if for the same observation i, both x_i and y_i are far away from their means, with large positive values if they are consistently in the same direction and large negative values if they are consistently in the opposite direction from each other.

4.1.4 Absolute Errors

Least squares is quite common, particularly because it quite easily mathematically to find the solution. However, it is equally compelling to use the absolute error loss, rather than squared error, which gives us a measure of overall error

as:

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}(\beta_0, \beta_1)|$$

We can't write down the equation for the $\hat{\beta}_0$ and $\hat{\beta}_1$ that makes this error the smallest possible, but we can find them using the computer, which is done by the `rq` function in R. Here is the plot of the resulting solutions from using least-squares and absolute error loss.

While least squares is more common for historical reasons (we can write down the solution!), using absolute error is in many ways more compelling, just like the median can be better than the mean for summarizing the distribution of a population. With squared-error, large differences become even larger, increasing the influence of outlying points, because reducing the squared error for these outlying points will significantly reduce the overall average error.

We will continue with the traditional least squares, since we are not (right now) going to spend very long on regression before moving on to other techniques for dealing with two continuous variables.

4.2 Inference for linear regression

One question of particular interest is determining whether $\beta_1 = 0$.

Question: Why is β_1 particularly interesting? (Consider this data on college tuition – what does $\beta_1 = 0$ imply)?

We can use the same strategy of inference for asking this question – hypothesis testing, p-values and confidence intervals.

As a hypothesis test, we have a null hypothesis of:

$$H_0 : \beta_1 = 0$$

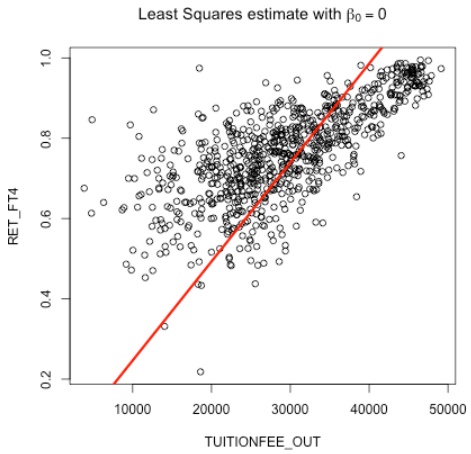
We can also set up the hypothesis

$$H_0 : \beta_0 = 0$$

However, this is (almost) never interesting.

Question: Consider our data: what would it mean to have $\beta_0 = 0$?

Does this mean we can just set β_0 to be anything, and not worry about it? No, if we do not get the right intercept, our line won't fit. Forcing the intercept to $\beta_0 = 0$ will make even the "best" line a terrible fit to our data:



Rather, the point is that we just don't usually care about *interpreting* that intercept. Therefore we also don't care about doing hypothesis testing on whether $\beta_0 = 0$ for most problems.

4.2.1 Bootstrap Confidence intervals

Once we get estimates $\hat{\beta}_0$ and $\hat{\beta}_1$, we can use the same basic idea we introduced in comparing groups to get bootstrap confidence intervals for the parameters. Previously we had two groups $X_1, \ldots, X_{n_1}$ and the other group $Y_1, \ldots, Y_{n_2}$ and we resampled the data *within* each group to get new data $X_1^*, \ldots, X_{n_1}^*$ and $Y_1^*, \ldots, Y_{n_2}^*$, each of the same size as the original samples. From this resampled data, we estimated our statistic $\hat{\delta}^*$ (e.g. the difference between the averages of

the two groups). We repeated this B times to get the distribution of $\hat{\delta}^*$, which we used to create a bootstrap distribution to create confidence intervals.

We are going to do the same thing here, only now we only have one population consisting of N pairs of data (x_i, y_i). We will resample N times from the data to get

$$(x_1^*, y_1^*), \dots, (x_N^*, y_N^*)$$

Some pairs will show up multiple times, but notice that each pair of x_i and y_i will always be together because *we sample the pairs*.

To get confidence intervals, we will use this bootstrap sample to recalculate $\hat{\beta}_0, \hat{\beta}_1$, and do this repeatedly to get the bootstrap distribution of these values.

Specifically,

1. We create a bootstrap sample by sampling *with replacement* N times from our data $(x_1, y_1), \dots, (x_N, y_N)$
2. This gives us a sample $(x_1^*, y_1^*), \dots, (x_N^*, y_N^*)$ (where, remember some data points will be there multiple times)
3. Run regression on $(x_1^*, y_1^*), \dots, (x_N^*, y_N^*)$ to get $\hat{\beta}_1^*$ and $\hat{\beta}_0^*$
4. Repeat this B times, to get

$$(\hat{\beta}_0^{(1)*}, \hat{\beta}_1^{(1)*}), \dots, (\hat{\beta}_0^{(B)*}, \hat{\beta}_1^{(B)*})$$

5. Calculate confidence intervals from the percentiles of these values.

I will write a small function in R that accomplishes this (you will look at this more closely in a lab):

```
bootstrapLM <- function(y, x, repetitions, confidence.level = 0.95) {
    stat.obs <- coef(lm(y ~ x))
    bootFun <- function() {
        sampled <- sample(1:length(y), size = length(y),
            replace = TRUE)
        coef(lm(y[sampled] ~ x[sampled]))
    }
    stat.boot <- replicate(repetitions, bootFun())
    nm <- deparse(substitute(x))
    row.names(stat.boot)[2] <- nm
    level <- 1 - confidence.level
    confidence.interval <- apply(stat.boot, 1, quantile,
        probs = c(level/2, 1 - level/2))
    return(list(confidence.interval = cbind(lower = confidence.interval[1,
        ], estimate = stat.obs, upper = confidence.interval[2,
        ]), bootStats = stat.boot))
}
```

We'll now run this on the `private` data

```
privateBoot <- with(private, bootstrapLM(y = RET_FT4,
    x = TUITIONFEE_OUT, repetitions = 10000))
privateBoot$conf
```

```
##                         lower     estimate        upper
## (Intercept)    4.628622e-01 4.863443e-01 5.094172e-01
## TUITIONFEE_OUT 8.766951e-06 9.458235e-06 1.014341e-05
```

Question: How do we interpret these confidence intervals? What do they tell us about the problem?

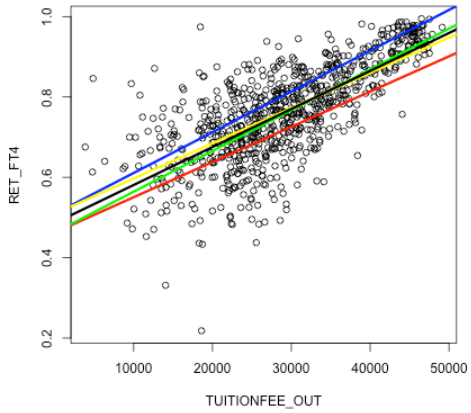
Again, these slopes are very small, because we are giving the change for each $1 change in tuition. If we multiply by 10,000, this number will be more interpretable:

```
privateBoot$conf[2, ] * 10000
```

```
##      lower   estimate      upper
## 0.08766951 0.09458235 0.10143414
```

Note that these confidence intervals means that there are a variety of different lines that are possible under these confidence intervals. For example, we can draw some lines that correspond to different combinations of these confidence interval limits.

```
plot(private[, c("TUITIONFEE_OUT", "RET_FT4")], col = "black")
abline(a = privateBoot$conf[1, 1], b = privateBoot$conf[2,
    1], col = "red", lwd = 3)
abline(a = privateBoot$conf[1, 3], b = privateBoot$conf[2,
    3], col = "blue", lwd = 3)
abline(a = privateBoot$conf[1, 1], b = privateBoot$conf[2,
    3], col = "green", lwd = 3)
abline(a = privateBoot$conf[1, 3], b = privateBoot$conf[2,
    1], col = "yellow", lwd = 3)
abline(lmPrivate, lwd = 3)
```



However, this is not really quite the right way to think about these two confidence intervals. If we look at these two separate confidence intervals and try to put them together, then we would think that anything covered jointly by the confidence intervals is likely. But that is not quite true. Our confidence in where the true line is located actually is narrower than what is shown, because some of the combinations of values of the two confidence intervals don't actually ever get seen together. This is because these two statistics $\widehat{\beta}_0$ and $\widehat{\beta}_1$ aren't independent from each other. Separate confidence intervals for the two values don't give you that additional information.[3]

How does this relate to our bootstrap for two groups?

Let's review our previous bootstrap method we used to compare groups, but restating the setup using a notation that matches our regression. In the previous chapter, we had a measurement of a continuous value (like flight delay) which we divided into two groups based on another characteristic (like airline). Previously we kept track of this by letting one group be $X_1, \ldots, X_{n_1}$ and the other group $Y_1, \ldots, Y_{n_2}$.

Let's introduce a different notation. Let y be our continuous measurement, flight delay, *for all our data*. To keep track of which of these observations were in which group, we can instead create another variable x that gives an observation's airline. This can equivalently store all our information (and indeed matches more closely with how you might record the data in a spreadsheet, with one column for the measurement and another column for the airline).

This means x is not continuous – it can only take 10 values corresponding to the 10 different airlines. This is not the same as the linear regression case we consider in this chapter, where x is continuous, but it gives us a similar notation to write the two problems, because now each observation in the flight data consisted of the pairs

$$(x_i, y_i)$$

[3]You can actually have joint confidence regions that demonstrate the dependency between these values, but that is beyond this class.

This is similar to our regression case, only with our regression example x_i is now continuous.

In our previous notation, when we did the bootstrap, we described this as re-sampling values from *each* of our group $X_1, \ldots, X_{n_1}$ and $Y_1, \ldots, Y_{n_2}$, so that we created a new dataset $X_1^*, \ldots, X_{n_1}^*$ and $Y_1^*, \ldots, Y_{n_2}^*$ each of the same size as the original distribution.

We can see that the bootstrap we introduced here would resample $N = n_1 + n_2$ samples from the pairs of (x_i, y_i), i.e. the union of the two groups. So if we applied the bootstrap strategy from this chapter to the two groups, this is a slight variation from the method in chapter 3. In particular, notice that the pair-resampling will not result in the two groups having n_1 and n_2 observations – it will be a random number in each group usually close to n_1 and n_2.

Both strategies are valid for comparing two groups, and there are arguments for both. Generally unless you have small sample sizes it will not create very large differences. The strategy in this chapter is more general – it can generalize to arbitrary numbers of variables as we will see in future chapters.

4.2.2 Parametric Models

Just as in the two-group setting, we can also consider a parametric model for creating confidence intervals. For linear regression, this is a widely-used strategy and its important to be familiar with it. Indeed, if we look at the summary of the `lm` function that does linear regression in R, we see a lot of information beyond just the estimates of the coefficients:

```
summary(lmPrivate)
```

```
##
## Call:
## lm(formula = RET_FT4 ~ TUITIONFEE_OUT, data = private)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.44411 -0.04531  0.00525  0.05413  0.31388
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    4.863e-01  1.020e-02   47.66   <2e-16 ***
## TUITIONFEE_OUT 9.458e-06  3.339e-07   28.32   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08538 on 783 degrees of freedom
## Multiple R-squared:  0.5061, Adjusted R-squared:  0.5055
```

```
## F-statistic: 802.3 on 1 and 783 DF,  p-value: < 2.2e-16
```

We see that it automatically spits out a table of estimated values and p-values along with a lot of other stuff.

Question: Why are there 2 p-values? What would be the logical null hypotheses that these p-values correspond to?

This output is exceedingly common – all statistical software programs do this – and these are based on a standard parametric model. This is a really ubiquitous model in data science, so its important to understand it well (and we will come back to it in multiple regression later in the semester).

Parametric Model for the data:

`lm` uses a standard parametric model to get the distributions of our statistics $\hat{\beta}_0$ and $\hat{\beta}_1$.

Recall that in fitting our line, we have already assumed a linear model:

$$y = \beta_0 + \beta_1 x + e.$$

This is a parametric model, in the sense that we assume there are unknown parameters β_0 and β_1 that describe how our data y was created.

In order to do inference (i.e. p-values and confidence intervals) we need to further assume a probability distribution for the errors e. Otherwise, there's nothing random about y. Specifically, we assume

- $e \sim N(0, \sigma^2)$, i..e normal with the *same* (unknown) variance σ^2.
- The unknown errors $e_1, \ldots, e_n$ are all independent from each other

Notice, this probability model for e implies a probability model for y. For a given x_i, each y_i is just a normal (e_i) with a (unknown) constant added to it ($\beta_0 + \beta_1 x_i$). So y_i is normally distributed, with

$$y_i | x_i \sim N(\beta_0 + \beta_1 x_i, \sigma^2)$$

Question: However, even though the errors e_i are assumed *i.i.d* the y_i are not i.i.d, why?

This assumption regarding the probability distribution of the errors allows us to know the distribution of the $\hat{\beta}_1$ (recall β_1 is a fixed constant, $\hat{\beta}_1$ is an estimate based on random data, so it is a random variable and has a distribution).

We won't show the derivation of its distribution, but since each y_i is normally distributed, $\hat{\beta}_1$ is as well.[4]

$$\hat{\beta}_1 \sim N(\beta_1, \nu_1^2)$$

where

$$\nu_1^2 = var(\hat{\beta}_1) = \frac{\sigma^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

In what follows, just try to follow the logic, you don't need to memorize these equations or understand how to derive them.

Notice the similarities in the broad outline to the parametric t-test for two-groups. We have an statistic, $\hat{\beta}_1$, and the assumptions of the parametric model gives us that the distribution of $\hat{\beta}_1$ is normal with a variance that depends on the (unknown) σ^2, i.e. the true variance in our individual data observations.

Estimating σ^2

Of course, we have the same problem as the t-test – we don't know σ^2! But like the t-test, we can estimate σ^2 and get an estimate of the variance of $\hat{\beta}_1$ (we'll talk more about how we estimate $\hat{\sigma}$ in a moment)

$$\hat{\nu}_1^2 = \hat{var}(\hat{\beta}_1) = \frac{\hat{\sigma}^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Hypothesis Testing

Using this knowledge, we can use the same idea as the t-test for two-groups, and create a similar test statistic for $\hat{\beta}_1$ that standardizes $\hat{\beta}_1$[5]

$$T_1 = \frac{\hat{\beta}_1}{\sqrt{\hat{var}(\hat{\beta}_1)}}$$

Just like the t-test, T_1 should be normally distributed[6] This is exactly what `lm` gives us:

```
summary(lm(RET_FT4 ~ TUITIONFEE_OUT, data = private))
```

```
##
## Call:
## lm(formula = RET_FT4 ~ TUITIONFEE_OUT, data = private)
##
```

[4] If you look at the equation of $\hat{\beta}_1$, then we can see that it is a linear combination of the y_i, and linear combinations of normal R.V. are normal, even if the R.V. are not independent.

[5] In fact, we can also do this for $\hat{\beta}_0$, with exactly the same logic, but β_0 is not interesting so we don't do it in practice.

[6] with the same caveat, that when you estimate the variance, you affect the distribution of T_1, which matters in small sample sizes.

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.44411 -0.04531  0.00525  0.05413  0.31388
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    4.863e-01  1.020e-02   47.66   <2e-16 ***
## TUITIONFEE_OUT 9.458e-06  3.339e-07   28.32   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08538 on 783 degrees of freedom
## Multiple R-squared:  0.5061, Adjusted R-squared:  0.5055
## F-statistic: 802.3 on 1 and 783 DF,  p-value: < 2.2e-16
```

Confidence intervals

We can also create parametric confidence intervals for $\hat{\beta}_1$ in the same way we did for the difference in two groups:

$$\hat{\beta}_1 \pm 1.96\hat{\nu}_1$$

```
confint(lmPrivate)
```

```
##                       2.5 %       97.5 %
## (Intercept)    4.663136e-01 5.063750e-01
## TUITIONFEE_OUT 8.802757e-06 1.011371e-05
```

4.2.2.1 Estimating σ^2

How do we estimate σ^2? Recall that σ^2 is the variance of the error distribution. We don't know the true errors e_i, but if we did, we know they are i.i.d and so a good estimate of σ^2 *would be* the sample variance of the true errors:

$$\frac{1}{n-1}\sum(e_i - \bar{e})^2$$

However, these true errors are unknown.

Question: If we knew the true β_0 and β_1 we could calculate the true e_i, how?

Thus with the true coefficients, we could calculate e_i and therefore use the above equation to estimate σ^2 from the data. But the coefficents β_0, β_1 are also unknown, so this isn't possible. Yet, this above thought-experiment does give

us an idea for how we could estimate σ^2. Specifically, though we don't *know* β_0 or β_1, we have estimates of β_0 and β_1. Namely, we can calculate the error of our data from the *estimated* line,

$$r_i = y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i)$$

The r_i are called the **residuals**. They are often called the errors, but they are not the actual (true) error, however ($r_i \neq e_i$). They are the error from the *estimated* line, and as a group think of them as estimates of the true error.

Using the residuals, we can take the sample variance of the residuals as a good first estimate of σ^2,

$$\frac{1}{n-1}\sum(r_i - \bar{r})^2$$

Mean of residuals, $\bar{r}$ In fact, it is an algebraic fact that $\bar{r} = 0$. So we can rewrite the above equation as

$$\frac{1}{n-1}\sum r_i^2$$

Be careful! Just because you "discover" in your data that $\bar{r} = 0$, this is NOT a sign your line is a good fit to the data. It is just a mathematical fact due to the way we estimated our coefficients and is always true for residuals from a least squared regression – even when the line is a lousy fit to the data.

Better estimate of σ

For regression, a better estimate is to divide by $n-2$ rather than $n-1$. Doing so makes our estimate **unbiased**, meaning that the average value of $\hat{\sigma}^2$ over many repeated samples will be σ. This is the same reason we divide by $n-1$ in estimating the sample variance rather than $1/n$ for the estimate of the variance of a single population.

These two facts gives us our final estimate:

$$\hat{\sigma}^2 = \frac{1}{n-2}\sum_i r_i^2.$$

The residuals r_i are not always great estimates of e_i (for example, they aren't independent, they don't have the same variance, etc). But, despite that, it turns out that $\hat{\sigma}^2$ is a *very* good, reliable estimate of σ^2, including if our assumptions about the errors being normally is wrong.

4.2.3 Assumptions

Like the t-test, the bootstrap gives a more robust method than the parametric linear model for creating confidence intervals.

The parametric linear model makes the following assumptions:

- Errors e_i are independent
- Errors e_i are i.i.d, meaning they have the same variance
- Errors are normally distributed

The bootstrap makes the same kind of assumptions as in the two group comparisons:

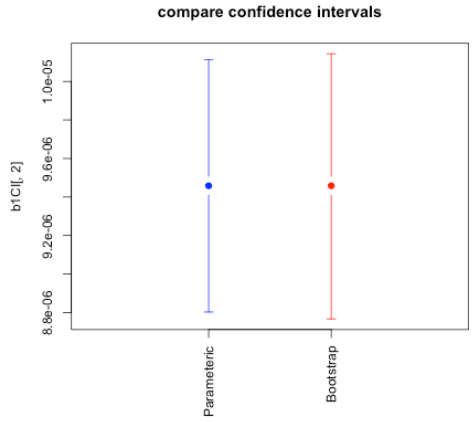
- The i.i.d resampling of the bootstrapped data mirrors how the actual data was generated (i.e. the actual data is i.i.d)
- The sample size is large enough that the sample distribution is close to the real distribution.

- The test statistic is well behaved (e.g. unbiased) – this *is* true for regression

Notice, that both methods assume the data points are *independent*. This is the most critical assumption for both methods. Both implicitly assume that all of the observations have the same variance (that's part of being i.i.d). The parametric method makes the further assumption of normality of the errors (like the t-test).

In practice, we do not see much difference in these two methods for our data:

```
##              lower     estimate        upper
## [1,] 8.802757e-06 9.458235e-06 1.011371e-05
## [2,] 8.766951e-06 9.458235e-06 1.014341e-05
```



4.2.4 Prediction Intervals

In addition to evaluating the coefficients, we can also look at the prediction we would make. This is a better way than the plots we did before to get an idea of what our predictions at a particular value would actually be.

Prediction

Question: How does our model predict a value, say for tuition of $20,000?

```
coef(lmPrivate)[1] + coef(lmPrivate)[2] * 20000
```

```
## (Intercept)
##    0.675509
```

```
predict(lmPrivate, newdata = data.frame(TUITIONFEE_OUT = 20000))
```

```
##        1
## 0.675509
```

These predictions are themselves statistics based on the data, and the uncertainty/variability in the coefficients carries over to the predictions. So we can also give confidence intervals for our prediction. There are two types of confidence intervals.

- Confidence intervals about the predicted average response – i.e. prediction of what is the average completion rate for all schools with tuition $20,000.
- Confidence intervals about a particular future observation, i.e. prediction of any particular school that has tuition $20,000. These are actually not called confidence intervals, but **prediction intervals**.

Clearly, we will use the same method to predict a value for both of these settings (the point on the line), but our estimate of the *precision* of these estimates varies.

Question: Which of these settings do you think would have wider CI?

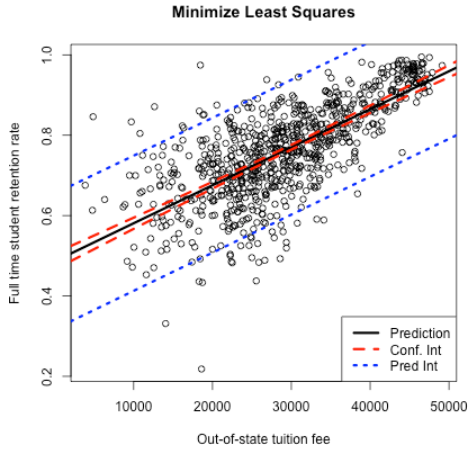
```
predict(lmPrivate, newdata = data.frame(TUITIONFEE_OUT = 20000),
    interval = "confidence")
```

```
##        fit       lwr       upr
## 1 0.675509 0.6670314 0.6839866
```

```
predict(lmPrivate, newdata = data.frame(TUITIONFEE_OUT = 20000),
    interval = "prediction")
```

```
##        fit       lwr      upr
## 1 0.675509 0.5076899 0.843328
```

We can compare these two intervals by calculating them for a large range of x_i values and plotting them:



Question: What do you notice about the difference in the confidence lines? How does it compare to the observed data?

Parametric versus Bootstrap

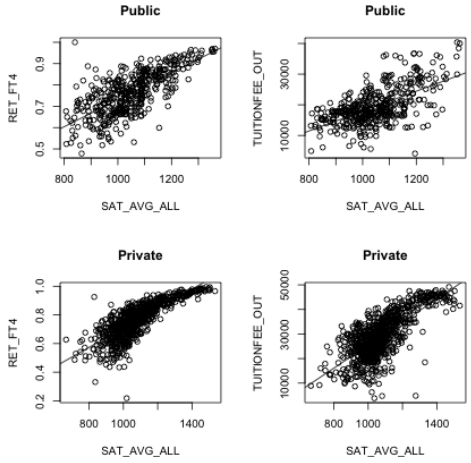
Notice that all of these predict commands use the parametric assumptions about the errors, rather than the bootstrap. We could bootstrap the *confidence intervals* for the prediction average.

Question: How would we do that?

The *prediction intervals*, on the other hand, rely more on the parametric model for estimating how much variability and individual point will have.

4.3 Least Squares for Polynomial Models & Beyond

Least squares will spit out estimates of the coefficients and p-values to any data – the question is whether this is a good idea. For example, consider the variable `SAT_AVG_ALL` that gives the average SAT score for the school.



Question: Looking at the public institutions, what do you see as it's relationship to the other two variables?

We might imagine that other functions would be a better fit to the data for the private schools.

Question: What might be some reasonable choices of functions?

We can fit other functions in the same way. Take a quadratic function, for example. What does that look like for a model?

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + e$$

We can, again, find the best choices of those co-efficients by getting the predicted value for a set of coefficients:

$$\hat{y}_i(\beta_0, \beta_1, \beta_2) = \beta_0 + \beta_1 x_i + \beta_2 x_i^2,$$

and find the error

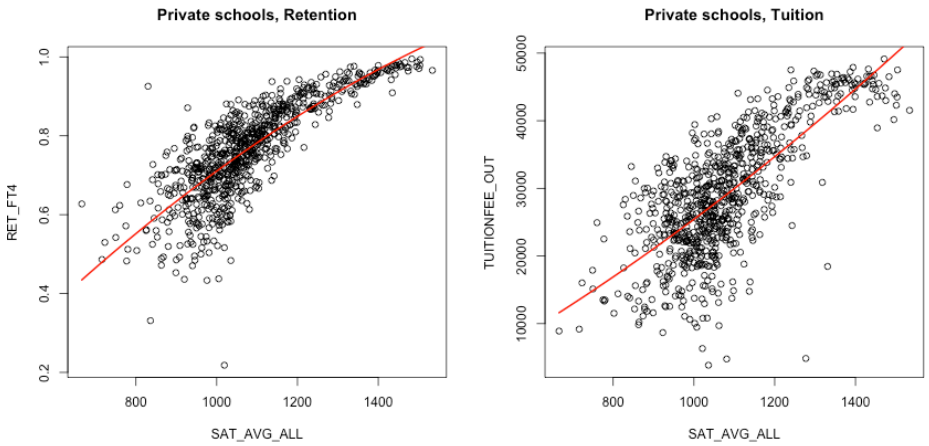
$$\ell(y_i, \hat{y}_i(\beta_0, \beta_1, \beta_2))$$

and trying to find the choices that minimizes the average loss over all the observations.

If we do least squares for this quadratic model, we are trying to find the coefficients $\beta_0, \beta_1, \beta_2$ that minimize,

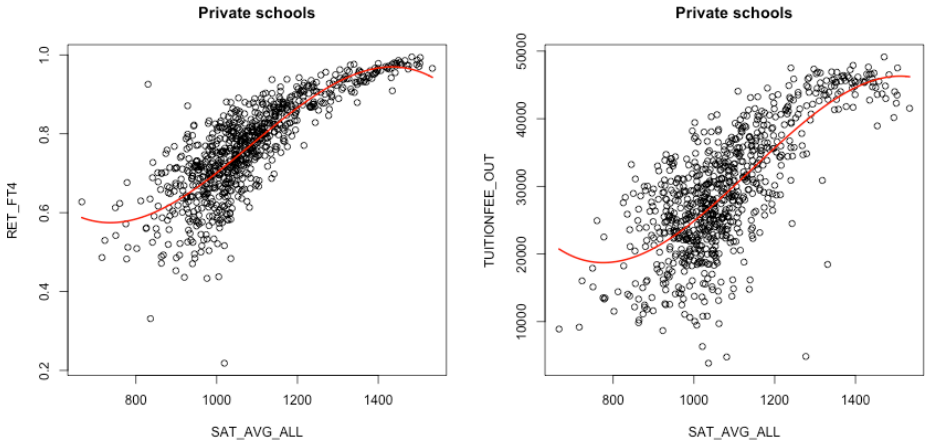
$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_i - \beta_2 x_i^2)^2$$

Here are the results:



It's a little better, but not much. We could try other functions. A cubic function, for example, is exactly the same idea.

$$\hat{y}_i(\beta_0, \beta_1, \beta_2) = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3.$$



Question: What do you think about the cubic fit?

We can, of course use other functions as well. For example, we could use log,

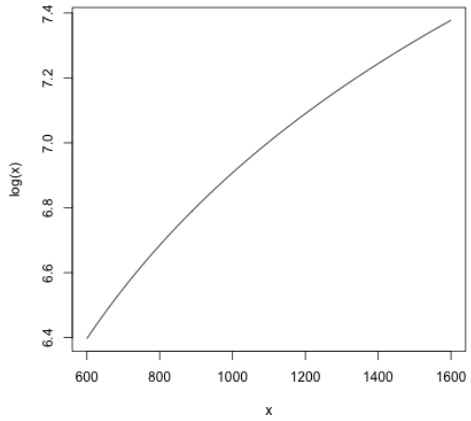
$$y = log(x) + e$$

We don't show this model fitted to the data here, but this seems unlikely to be the right scale of the data. If we are going to search for functions to fit, we want

a function that describes the *shape* of the curve, but we will undoubtably need more flexibility than that to match our data. In particular, this model has no parameters that we can adjust to make it match our data.

If we look at our x values, we see that they are in the range of 800-1400 (i.e. SAT scores!).Consider what the *log* looks like in this range:

```
par(mfrow = c(1, 1))
curve(log, 600, 1600)
```



Question: Does this seem like an effective transformation?

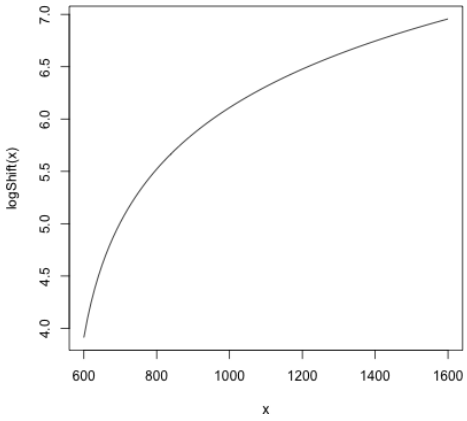
We could add an intercept term

$$y = \beta_0 + log(x) + e$$

Question: What does adding β_0 mean intuitively?

If we add a constant *inside* the log, we get flexibility in the shape.

```
par(mfrow = c(1, 1))
logShift <- function(x) {
    log(x - 550)
}
curve(logShift, 600, 1600)
```



We could play around with adjusting this model to give us appropriate parameters to work with and then fit it with least squares as well, e.g.

$$y = \beta_0 + log(\beta_1 x) + e$$

However, we are not going to spend time on this, because it will be an arbitrary and ad hoc way to model the data.[7] In general, finding some combination of standard functions to match the entire scope of the data is unlikely to work for most data sets. Instead we want something that is more flexible. We'd really like to say

$$y = f(x) + e$$

and just estimate f, without any particular restriction on f.

So we are going to think much more broadly about how to create a method that will be adaptive to our data and not require us to define functions in advance that match our data.

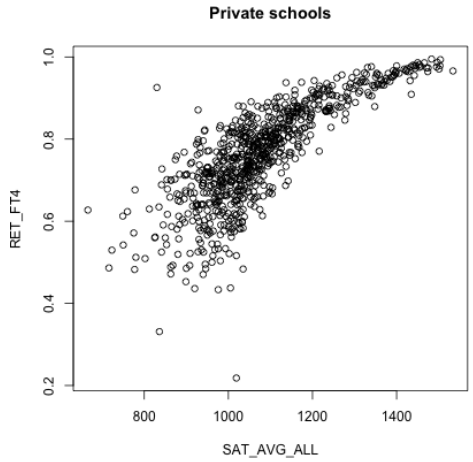
4.4 Local fitting

There are many strategies to estimate f without assuming what kind of function f is. Many such strategies have the flavor of estimating a series of "easy" functions on small regions of the data and then putting them together. The combination of these functions is much more complex than the "easy" functions are.

Question: What ideas can you imagine for how you might get a descriptive curve/line/etc to describe this data?

[7] This can be useful if you have a pre-existing physical model that describes the process that created the data, e.g. from physics or chemistry

```
plot(RET_FT4 ~ SAT_AVG_ALL, data = private, main = "Private schools")
```



We are going to look at once such strategy that builds up a function f as a series estimates of easy functions like linear regression, but only for a very small region.

Like with density estimation, we are going to slowly build up to understanding the most commonly used method (LOESS) by starting with simpler ideas first.

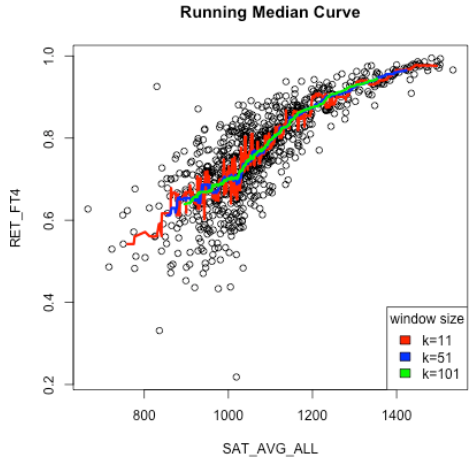
4.4.1 Running Mean or Median

Like with our development of density estimation, we will start with a similar idea to estimate $f(x)$ by taking a "window" – a fixed width region of the x-axis – with its center at x. We will identify the points captured in this window (meaning their x values are in the window), and our estimate of $f(x)$ will be the mean (or median) of the corresponding y values of those points.

We can write this as an equation.

$$\hat{f}(x) = \frac{1}{\#\text{in window}} \sum_{i:x_i \in [x-\frac{w}{2}, x+\frac{w}{2})} y_i$$

Just like with density estimation, we can do this for all x, so that we slide the window over the x-axis. For this reason, this estimate is often called a "running mean" or "running median".



There are a lot of varieties on this same idea. For example, you could make the window not fixed width w, but a window centered at x that has variable width, but a fixed number of points for all x. This is what is plotted above (while it's conceptually easy to code from scratch, there are a lot of nitpicky details, so in the above plot we used a built-in implementation which uses this strategy instead).

Question: What do you notice when I change the number of fixed points in each window? Which seems more reasonable here?

Comparison to density estimation

If this feels familiar, it should! This is very similar to what we did in density estimation. However, in density estimation, when estimating the density $p(x)$, we captured the data x_i that were in windows around x, and calculated basically the *number* of points in the window to get $\hat{p}(x)$,

$$\hat{p}(x) = \frac{\#\ x_i \text{ in window}}{nw} = \sum_{i:x_i \in [x-\frac{w}{2}, x+\frac{w}{2})} \frac{1}{nw}$$

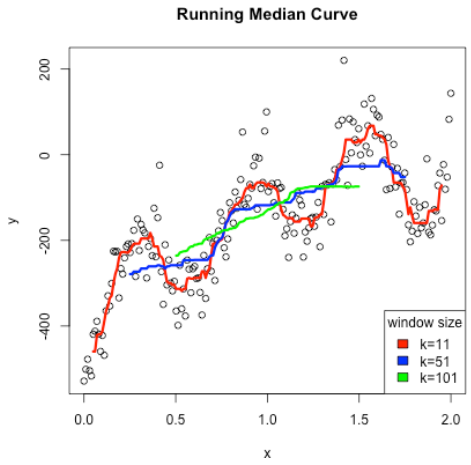
With function estimation, we are instead finding the x_i that are near x and then taking the mean of their corresponding y_i to calculate $\hat{f}(x)$. So for function estimation, the x_i are used to determining which points (x_i, y_i) to use, but the y_i are used to calculate the value.

$$\begin{aligned}\hat{f}(x) &= \frac{\text{sum of } y_i \text{ in window}}{\#\ x_i \text{ in window}} \\ &= \frac{\sum_{i:x_i \in [x-\frac{w}{2}, x+\frac{w}{2})} y_i}{\sum_{i:x_i \in [x-\frac{w}{2}, x+\frac{w}{2})} 1}\end{aligned}$$

4.4.2 Kernel weighting

One disadvantage to a running median is that it can create a curve that is rather jerky – as you move the window you are added or taking away a point so the mean of the y_i abruptly changes. This is particularly pronounced with windows with a small number of points – changing a single y_i really affects the mean. Alternatively, if you have a wide window with many points, then adding or subtracting a single point doesn't have a big effect, so the jerky steps are smaller, so less noticable (though still there!). But the downside of wide windows is that then your estimate of $f(x)$ at any point x will be the result of the average of the y values from points that are quite far away from the x you are interested in, so you won't capture trends in the data.

You can see this in this simulated data I created which has a great deal of local changes:



We've already seen a similar concept when we talked about kernel density estimation, instead of histograms. There we saw that we could describe our windows as *weighting* of our points x_i based on their distance from x. We can do the same idea for our running mean:

$$\begin{aligned}\hat{f}(x) &= \frac{\sum_{i:x_i \in [x-\frac{w}{2}, x+\frac{w}{2})} y_i}{\sum_{i:x_i \in [x-\frac{w}{2}, x+\frac{w}{2})} 1} \\ &= \frac{\sum_{i=1}^{n} y_i f(x, x_i)}{\sum_{i=1}^{n} f(x, x_i)}\end{aligned}$$

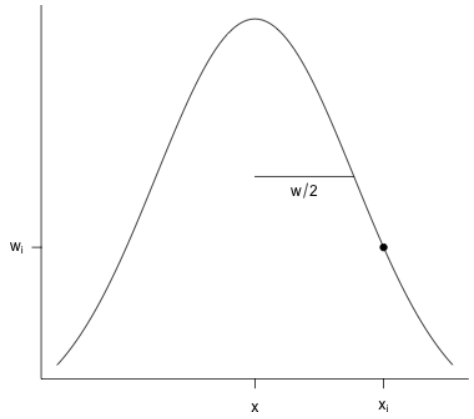
where again, $f(x, x_i)$ weights each point by $1/w$

$$f(x, x_i) = \begin{cases} \frac{1}{w} & x_i \in [x - \frac{w}{2}, x + \frac{w}{2}) \\ 0 & otherwise \end{cases}$$

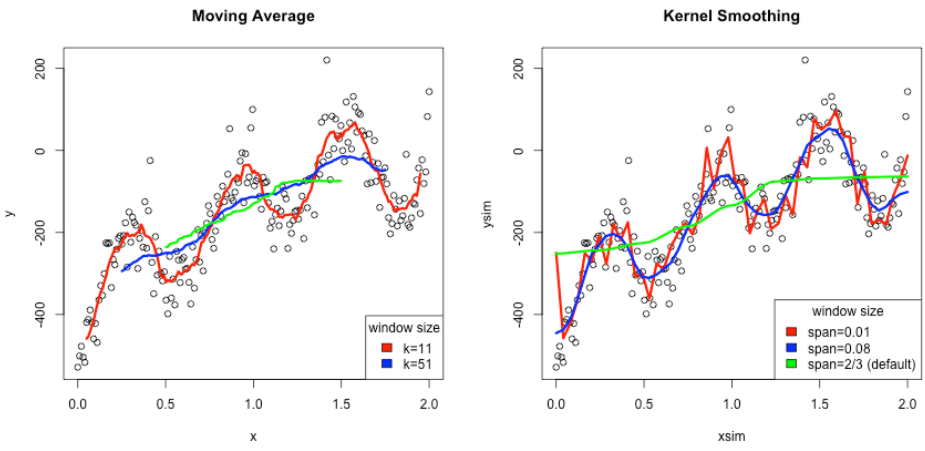
(notice the constant $1/w$ cancels out, but we leave it there to look like the kernel density estimation).

This is called the Nadaraya-Watson kernel-weighted average estimate or kernel smoothing regression.

Again, once we write it this way, it's clear we could again choose different weighting functions, like the gaussian kernel, similar to that of kernel density estimation. Just as in density estimation, you tend to get smoother results if our weights aren't abruptly changing from 0 once a point moves in or out of the window. So we will use the same idea, where we weight our point i based on how close x_i is to the x for which we are trying to estimate $f(x)$. And just like in density estimation, a gaussian kernel is the common choice for how to decide the weight:

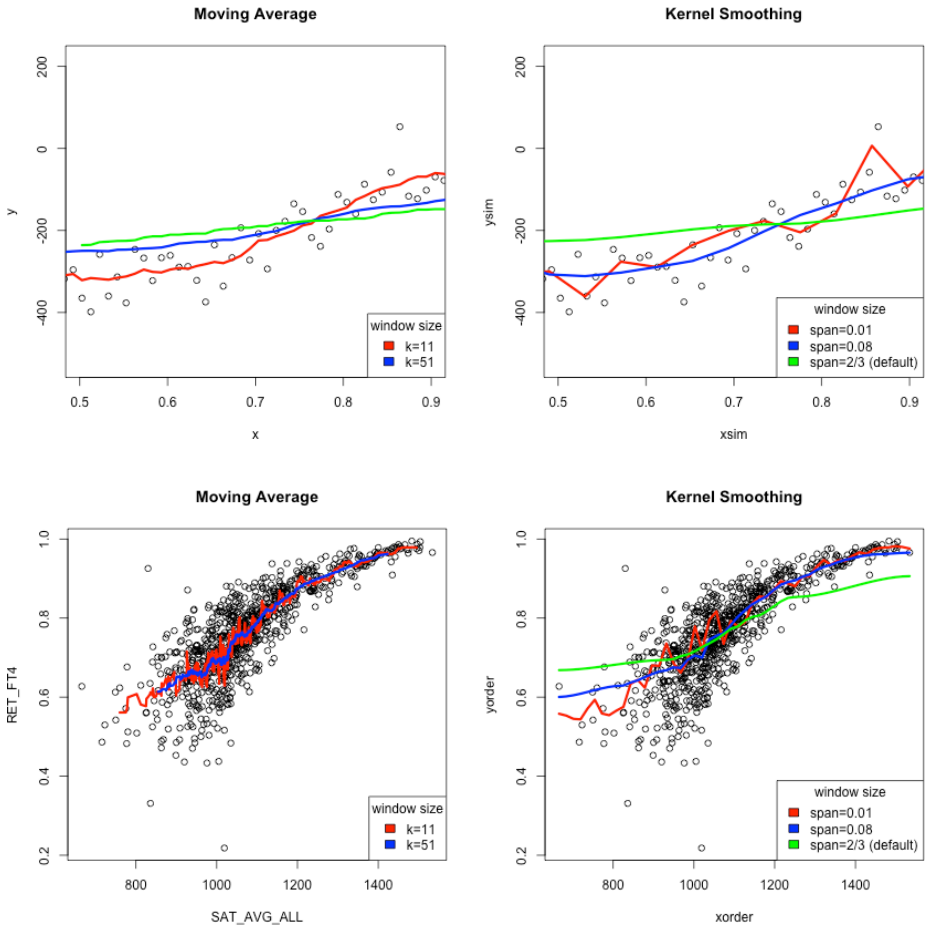


Here's how the gaussian kernel smoothing weights compare to a rolling mean (i.e. based on fixed windows) on the very wiggly simulated data I made



It's important to see that both methods can overflatten the curve or create overly

wiggly curves depending on how you change the choices of the algorithms. But the difference is that the moving average will always have these tiny bumps if you zoom in, while the kernel smoothing won't



Window width

The `span` argument tells you what percentage of points are used in predicting x (like bandwidth in density estimation)[8]. So there's still an idea of a window size; it's just that within the window, you are giving more emphasis to points near your x value.

Notice that one advantage is that you can define an estimate for any x in the range of your data – the estimated curve doesn't have to jump as you add new points. Instead it transitions smoothly.

[8]There's a lot of details about span and what points are used, but we are not going to worry about them. What I've described here gets at the idea

Question: What other comparisons might you make here?

Weighted Mean

If we look at our estimate of $f(x)$, we can actually write it more simply as a **weighted mean** of our y_i

$$\begin{aligned}\hat{f}(x) &= \frac{\sum_{i=1}^{n} y_i f(x, x_i)}{\sum_{i=1}^{n} f(x, x_i)} \\ &= \sum_{i=1}^{n} w_i(x) y_i\end{aligned}$$

where

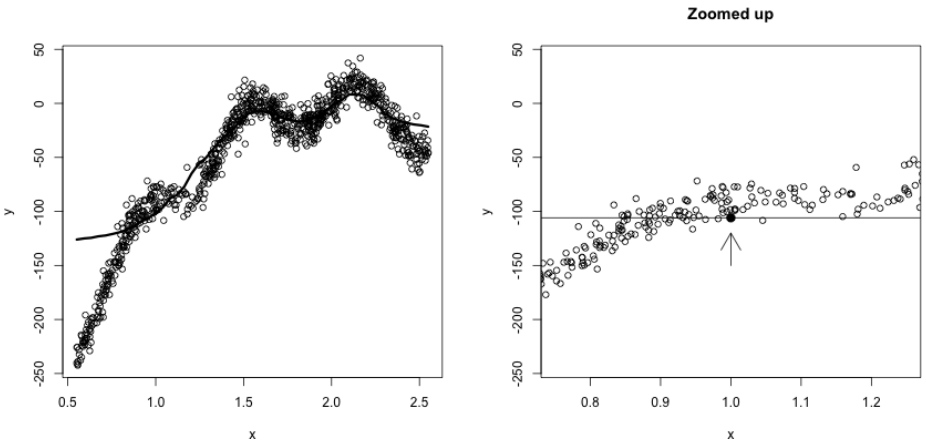
$$w_i(x) = \frac{f(x, x_i)}{\sum_{i=1}^{n} f(x, x_i)}$$

are weights that indicate how much each y_i should contribute to the mean (and notice that these weights sum to one). The standard mean of all the points is equivalent to choosing $w_i(x) = 1/n$, i.e. each point counts equally.

4.4.3 Loess: Local Regression Fitting

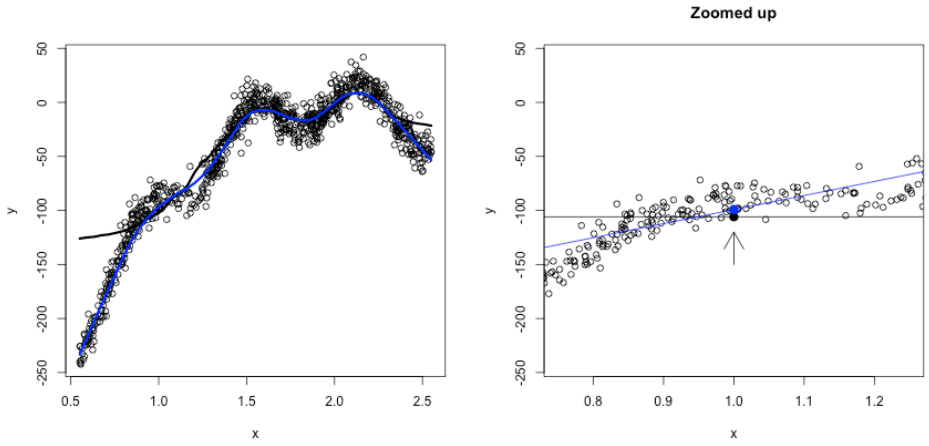
In the previous section, we use kernels to have a nice smooth way to decide how much impact the different y_i have in our estimate of $f(x)$. But we haven't changed the fact that we are essentially taking just a mean of the nearby y_i to estimate $f(x)$.

Let's go back to our simple windows (i.e. rectangular kernel). When we estimate $f(x)$, we are doing the following:

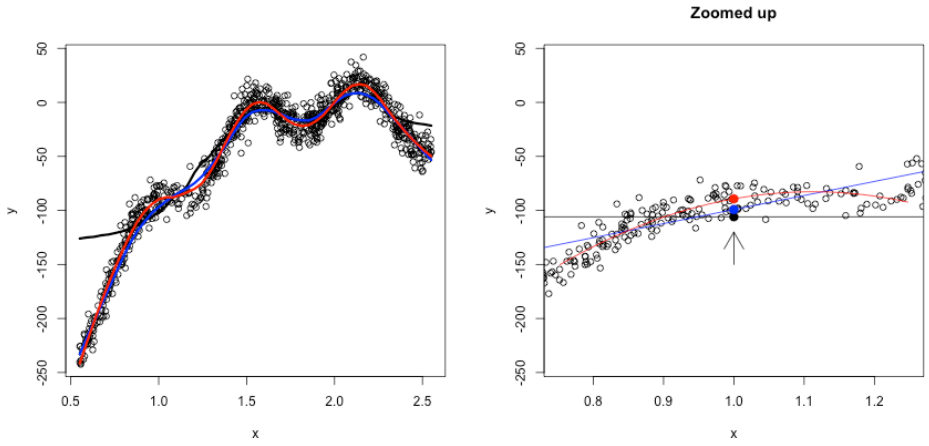


We see that for our prediction $\hat{f}(x)$ at $x = 1$, we are not actually getting into where the data is because of the in balance of how the x_i values are distributed. That's because the function is changing around $x = 1$; weighting far-away points would help some, we're basically trying to "fit" a constant line to what clearly is changing in this window.

We could do this for every x, as our window keeps moving, so we would never actually be fitting a polynomial across the entire function. So while we wouldn't think a line fit the overall data very well, locally around $x = 1$ it would be more reasonable to say it is roughly like a line:



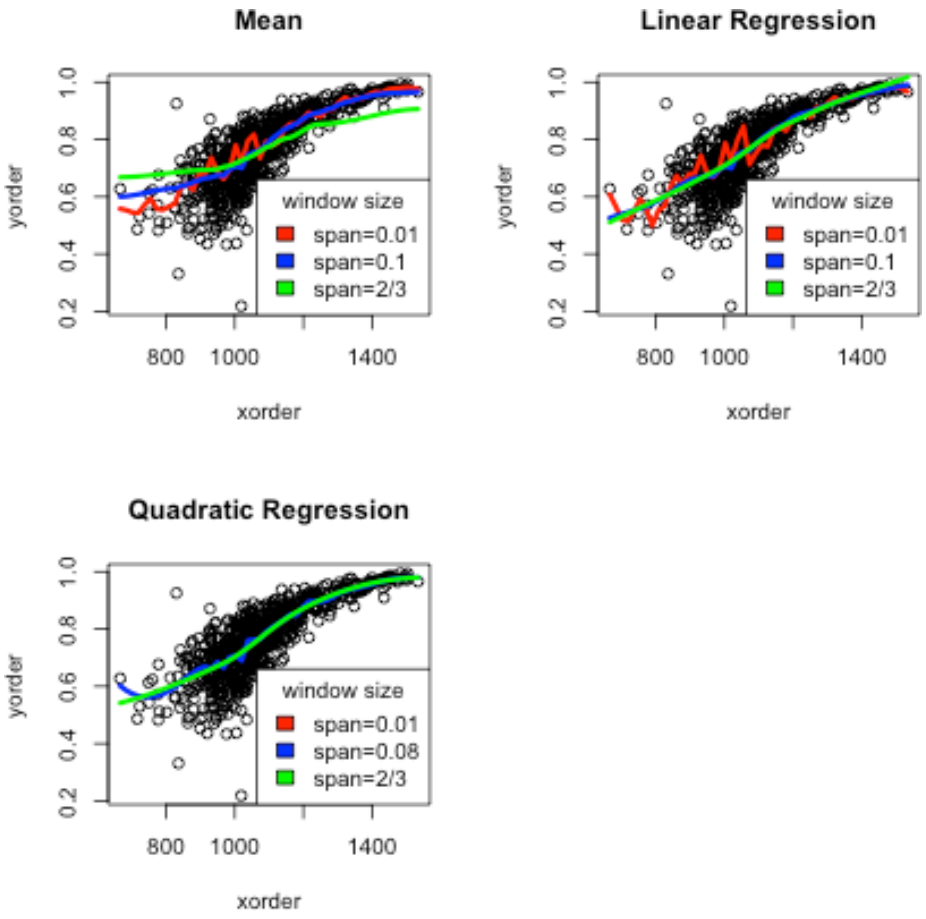
We could go even further and say a quadratic would be better:



In short, we are saying, to estimate $f(x)$ *locally* some simple polynomials will work well, even though they don't work well globally.

So we now have the choice of the degree of the polynomial *and* the span/window

size.



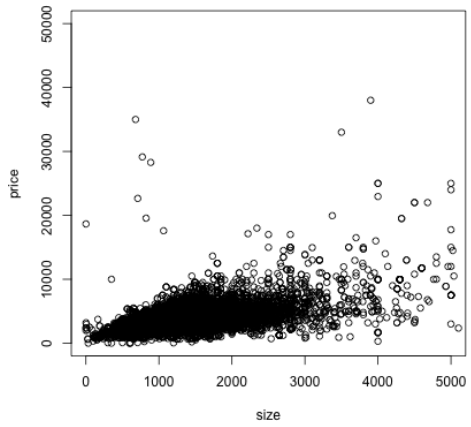
Question: What conclusions would you draw about the difference between choosing the degree of the fit (mean/linear/quadratic)?

Generally degree is chosen to be 2, as it usually gives better fitting estimates, while the span parameter might be tweaked by the user.

4.5 Big Data clouds

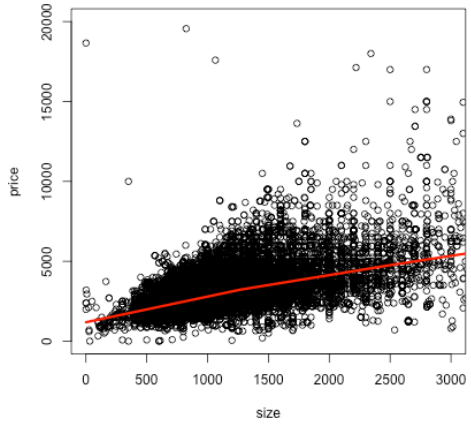
It can be particularly helpful to have a smooth scatter for visualization when you have a lot of data points. Consider the following data on craigs list rentals

that you saw in lab. We would suspect that size would be highly predictive of price, and indeed if we plot price against size that's pretty clear.



But, because of the number of points, we can't really see much of what's going on. In fact our eye is drawn to outlying (and less representative) points, while the rest is just a black smear where the plots are on top of each other.

We can add a loess smooth curve to get an idea of where the bulk of the data lie. We'll zoom in a bit closer as well by changing the x and y limits of the axes.



Question: What does this tell you about the data?

4.5.1 2D density smoothing plots

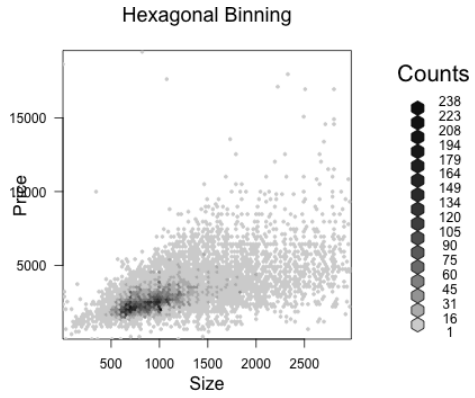
If we really want to get a better idea of what's going on under that smear of black, we can use 2D density smoothing plots. This is the same idea as density smoothing plots for probability densities, only for 2D. Imagine that instead of a histogram along the line, a 2D histogram. This would involve griding the 2D

plane into rectangles (instead of intervals) and counting the number of points within each rectangle. The high of the bars (now in the 3rd dimension) would give a visualization of how many points there are in different places in the plot.

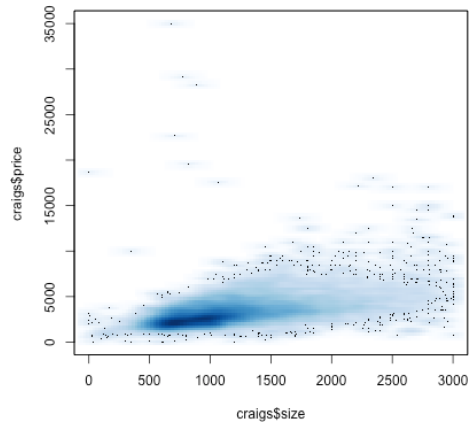
Then just like with histograms, we can smooth this, so that we get a smooth curve over the 2 dimensions.

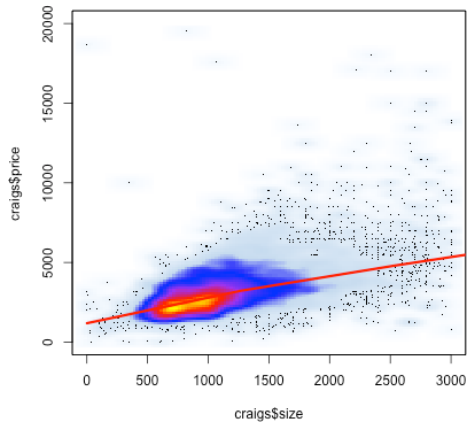
A 3D picture of this would be cool, but difficult to actually see information, axes, etc. So its common to instead smash this information into 2D, by representing the 3rd dimension (the density of the points) by a color scale instead.

Here is an example of such a visualization of a 2D histogram (the `hexbin` package)



We can use a smoother version of this and get more gradual changes (and a less finicky function) using the `smoothScatter` function

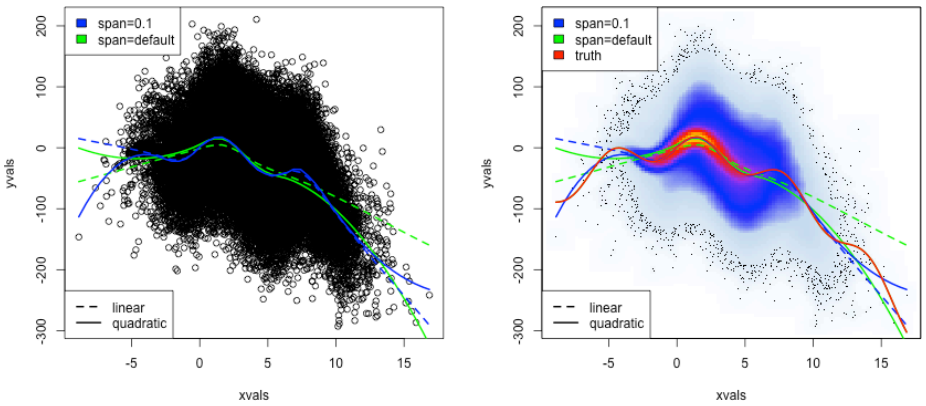


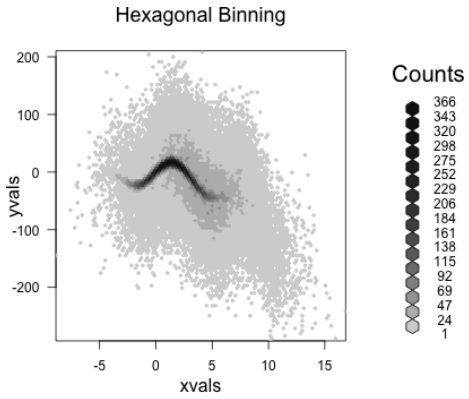


Question: What do these colors tell you? How does this compare to the smooth line? What do you see about those points that grabbed our eye before (and which the loess line ignored)?

Simulated Example

For this data, it turned out that the truth was pretty linear. But many times, the cloud of data can significantly impair our ability to see the data. We can simulate a more complicated function with many points.





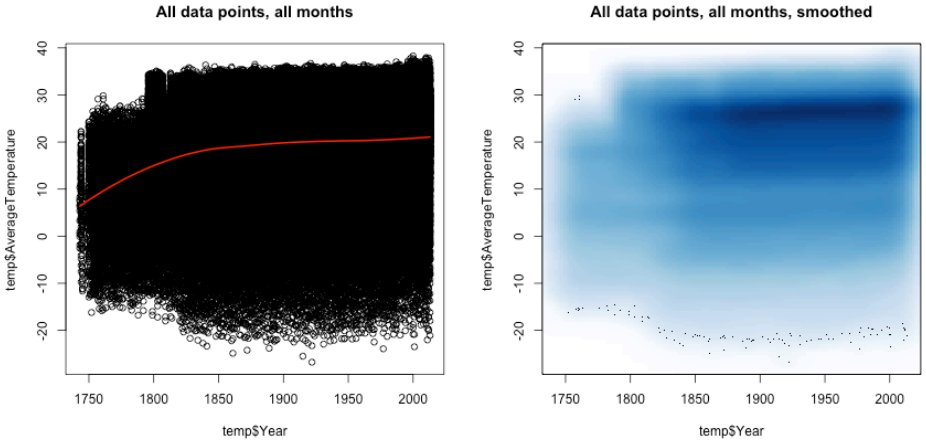
4.6 Time trends

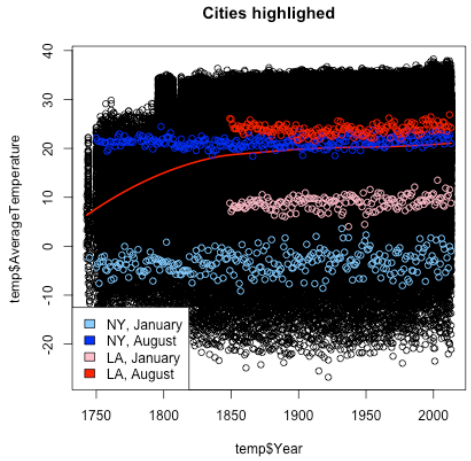
Let's look at another common example of fitting a trend – time data. In the following dataset, we have the average temperatures (in celecius) by city per month since 1743.

```
##           dt AverageTemperature AverageTemperatureUncertainty    City
## 1 1849-01-01             26.704                         1.435 Abidjan
## 2 1849-02-01             27.434                         1.362 Abidjan
## 3 1849-03-01             28.101                         1.612 Abidjan
## 4 1849-04-01             26.140                         1.387 Abidjan
## 5 1849-05-01             25.427                         1.200 Abidjan
## 6 1849-06-01             24.844                         1.402 Abidjan
##         Country Latitude Longitude
## 1 Côte D'Ivoire    5.63N     3.23W
## 2 Côte D'Ivoire    5.63N     3.23W
## 3 Côte D'Ivoire    5.63N     3.23W
## 4 Côte D'Ivoire    5.63N     3.23W
## 5 Côte D'Ivoire    5.63N     3.23W
## 6 Côte D'Ivoire    5.63N     3.23W
```

Given the scientific consensus that the planet is warming, it is interesting to look at this data, limited though it is, to see how different cities are affected.

Here, we plot the data with `smoothScatter`, as well as plotting just some specific cities

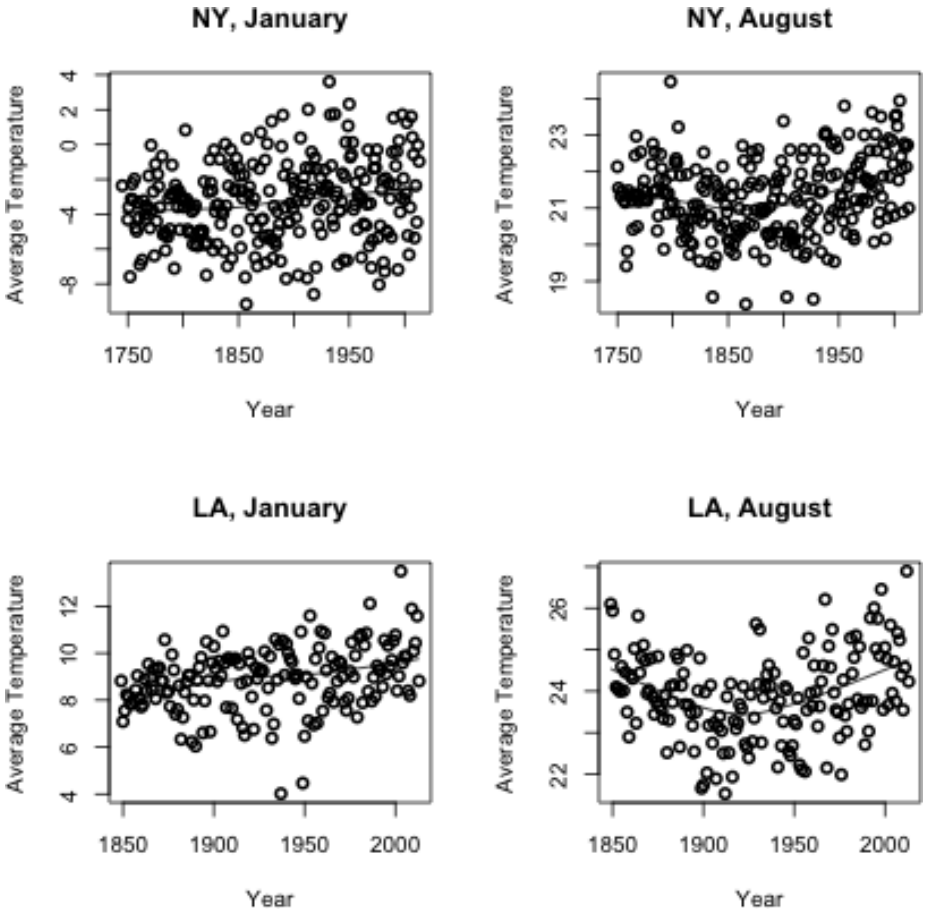




This is a very uninformative plot, despite our best efforts.

Question: Why was it uninformative?

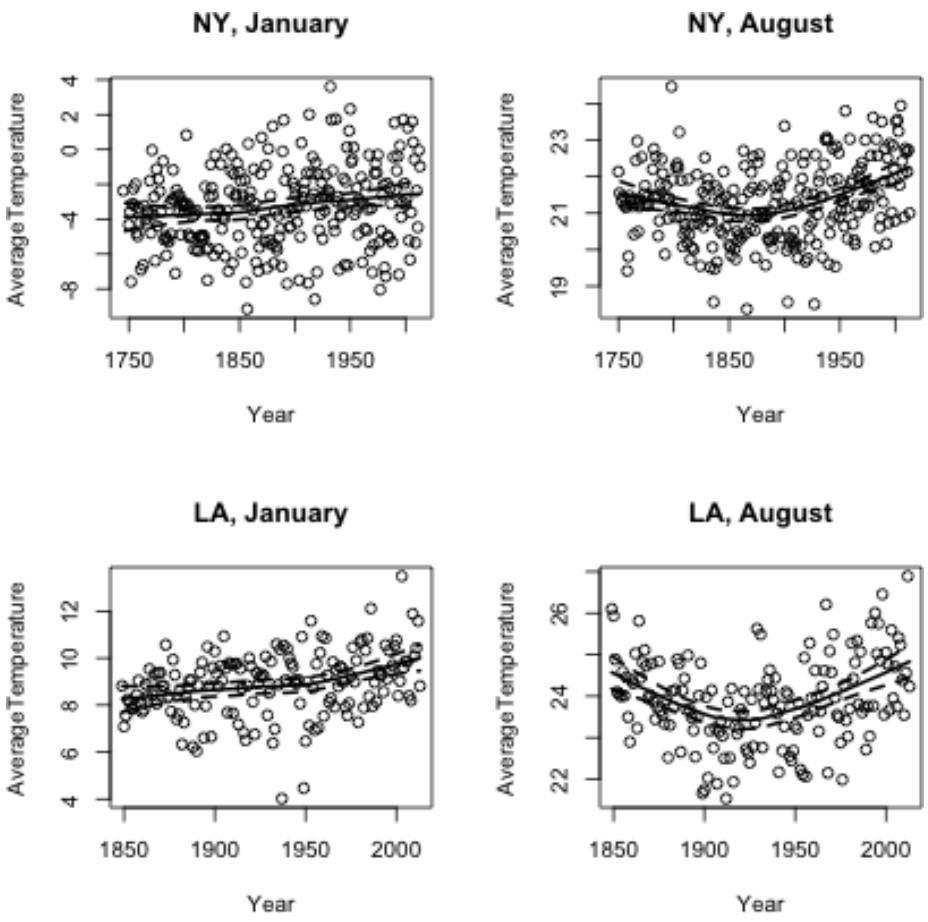
We can consider for different cities or different months how average temperatures have changed. We use the function `scatter.smooth` that both plots the points and places a loess curve on top.



Loess Prediction Intervals

We can even calculate (parametric) confidence intervals around these curves (based on a type of t-statistic for kernel smoothers), with a bit more lines of code. They are called prediction intervals, because they are confidence intervals for the prediction at each point.

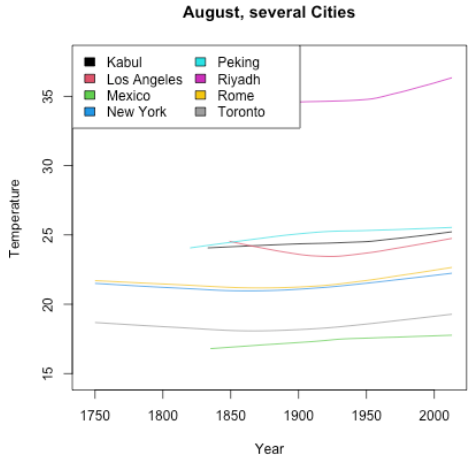
In fact, since it's a bit annoying, I'm going to write a little function to do it.



Question: Look at the code above. In what way does it look like t-statistic intervals?

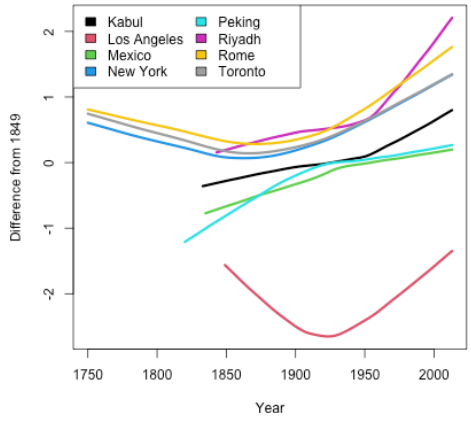
Comparing Many Cities

Smooth scatter plots can be useful to compare the time trends of many groups. It's difficult to plot each city, but we can plot their loess curve. I will write a function to automate this. For ease of comparison, I will pick just a few cities in the northern hemisphere.

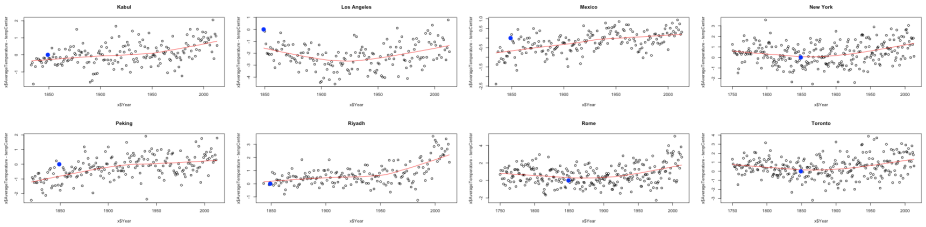


Question: What makes these curves so difficult to compare?

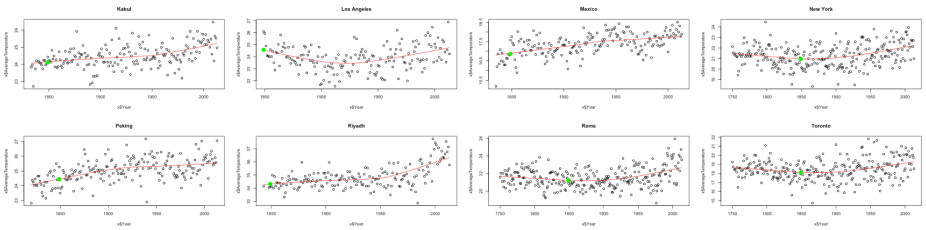
Notice that because these cities have a different baseline temperature, that is a big part of what the plot shows – how the different lines are shifted from each other. We are interested in instead how they compare when changing over time. So instead, I'm going to subtract off their temperature in 1849 before we plot, so that we plot not the temperature, but the change in temperature since 1849, i.e. change relative to that temperature.



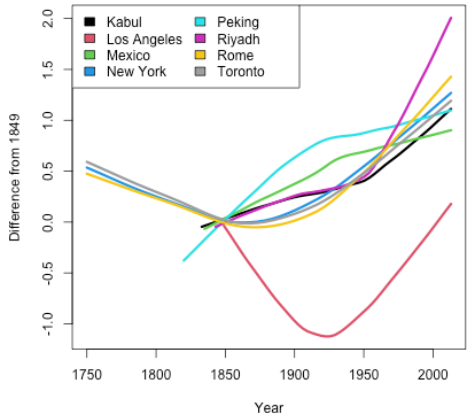
That still didn't accomplish my goal of having a similar baseline. Why not? Consider the following plots of the data from each of the 8 cities, where I highlight the 1849 temperature in blue.



We see that in fact, the temperature in any particular year is variable around the overall "trend" we see in the data. So by subtracting off 1849, we are also subtracting off that noise. We would do better to find, using loess, the value of the function that predicts that trend in 1849 (in green below):



Notice how much better that green point is as a reference point. Now we can subtract off that value instead, and use that as our baseline:



Notice how difficult it can be to compare across different cities; what we've shown here is just a start. The smoothed curves make it easier to compare, but also mask the variability of the original data. Some curves could be better representations of their cities than others. I could further try to take into account the scale of the change – maybe some cities temperature historically vary quite a lot from year to year, so that a difference in a few degrees is less meaningful. I could also plot confidence intervals around each curve to capture some of this variability.

Chapter 5

Visualizing Multivariate Data

```
## Linking to ImageMagick 6.9.12.3
## Enabled features: cairo, fontconfig, freetype, heic, lcms, pango, raw, rsvg, webp
## Disabled features: fftw, ghostscript, x11
```

We've spent a lot of time so far looking at analysis of the relationship of two variables. When we compared groups, we had 1 continuous variable and 1 categorical variable. In our curve fitting section, we looked at the relationship between two continuous variables.

The rest of the class is going to be focused on looking at many variables. This chapter will focus on visualization of the relationship between many variables and using these tools to explore your data. This is often called **exploratory data analysis** (EDA)

5.1 Relationships between Continous Variables

In the previous chapter we looked at college data, and just pulled out two variables. What about expanding to the rest of the variables?

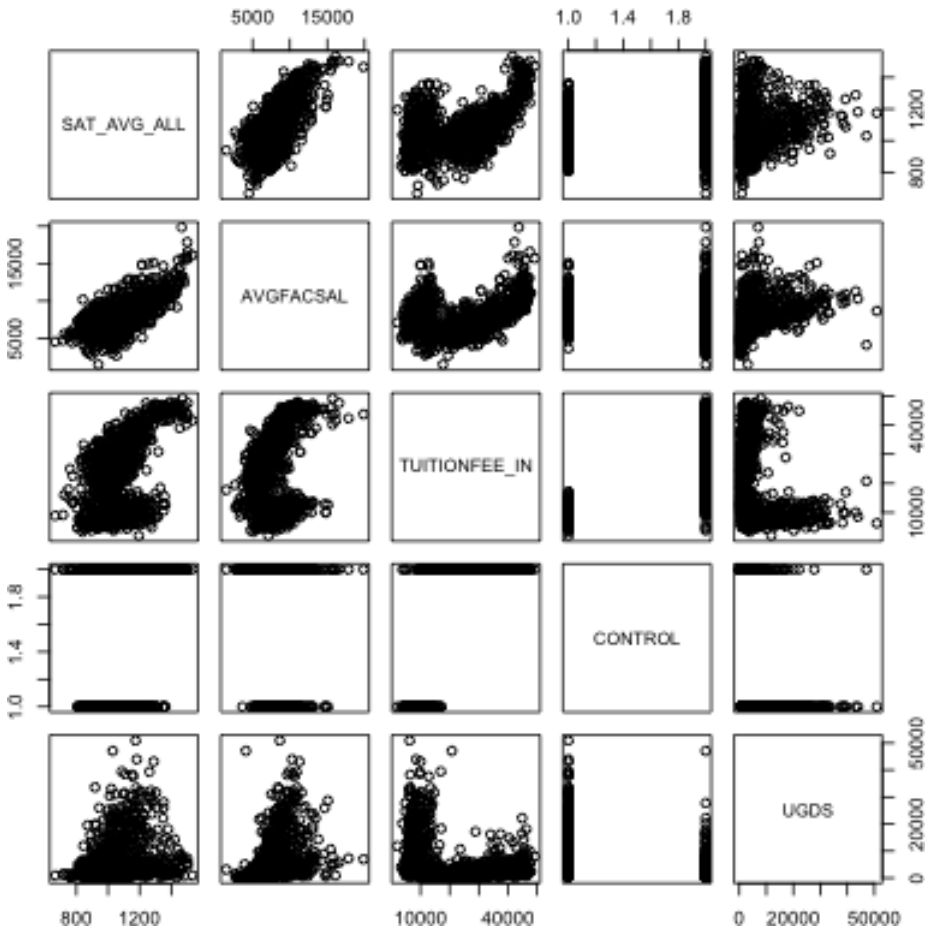
A useful plot is called a **pairs plot**. This is a plot that shows the scatter plot of all pairs of variables in a matrix of plots.

```
dataDir <- "../finalDataSets"
scorecard <- read.csv(file.path(dataDir, "college.csv"),
    stringsAsFactors = FALSE, na.strings = c("NA",
        "PrivacySuppressed"))
scorecard <- scorecard[-which(scorecard$CONTROL ==
```

```
    3), ]
smallScores <- scorecard[, -c(1:3, 4, 5, 6, 9, 11,
    14:17, 18:22, 24:27, 31)]
```

Let's first start with a small number of variables, just the first four variables

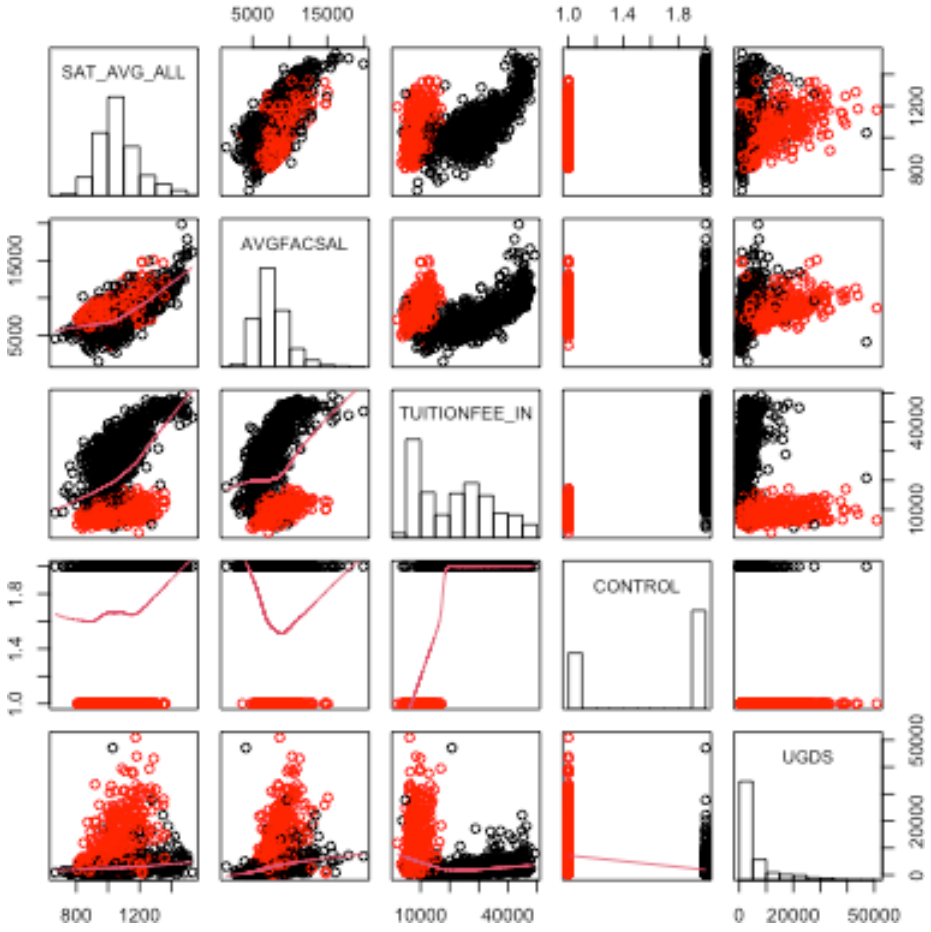
```
pairs(smallScores[, 1:5])
```



Question: What kind of patterns can you see? What is difficult about this plot? How could we improve this plot?

We'll skip the issue of the categorical `Control` variable, for now. But we can add in some of these features.

```
panel.hist <- function(x, ...) {
    usr <- par("usr")
    on.exit(par(usr))
    par(usr = c(usr[1:2], 0, 1.5))
    h <- hist(x, plot = FALSE)
    breaks <- h$breaks
    nB <- length(breaks)
    y <- h$counts
    y <- y/max(y)
    rect(breaks[-nB], 0, breaks[-1], y)
}
pairs(smallScores[, 1:5], lower.panel = panel.smooth,
    col = c("red", "black")[smallScores$CONTROL], diag.panel = panel.hist)
```

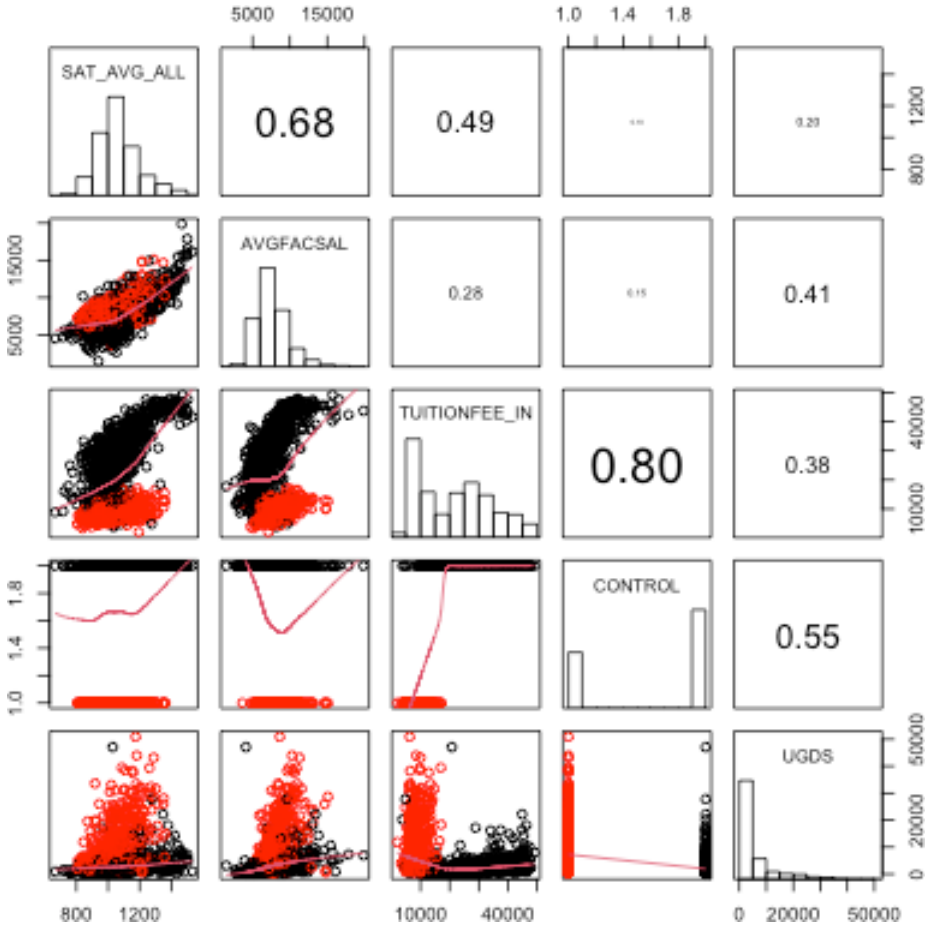


In fact double plotting on the upper and lower diagonal is often a waste of space.

Here is code to plot the sample correlation value instead,

$$\frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

```
panel.cor <- function(x, y, digits = 2, prefix = "",
    cex.cor, ...) {
    usr <- par("usr")
    on.exit(par(usr))
    par(usr = c(0, 1, 0, 1))
    r <- abs(cor(x, y, use = "pairwise.complete.obs"))
    txt <- format(c(r, 0.123456789), digits = digits)[1]
    txt <- paste0(prefix, txt)
    if (missing(cex.cor))
        cex.cor <- 0.8/strwidth(txt)
    text(0.5, 0.5, txt, cex = cex.cor * r)
}
pairs(smallScores[, 1:5], lower.panel = panel.smooth,
    upper.panel = panel.cor, col = c("red", "black")[smallScores$CONTROL],
    diag.panel = panel.hist)
```

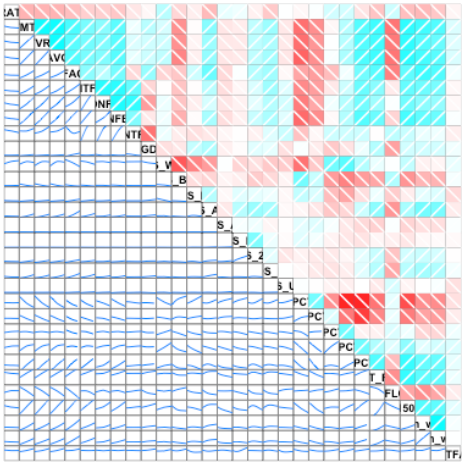


This is a pretty reasonable plot, but what if I want to look at more variables?

```
pairs(smallScores, lower.panel = panel.smooth, upper.panel = panel.cor,
    col = c("red", "black")[smallScores$CONTROL], diag.panel = panel.hist)
```

Even with 11 variables, this is fairly overwhelming, though still potentially useful. If I want to look at all of the 30 variables this will be daunting. For many variables, we can make a simpler representation, that simply plots the correlations using colors on the upper diagonal and a summary of the data via loess smoothing curves on the lower diagonal. This is implemented in the `gpairs` function (it also has default options that handle the categorical data better, which we will get to below).

```
library(gpairs)
suppressWarnings(corrgram(scorecard[, -c(1:3)]))
```



The lower panels gives only the loess smoothing curve and the upper panels indicate the correlation of the variables, with dark colors representing higher correlation.

Question: What do you see in this plot?

5.2 Categorical Variable

Let's consider now how we would visualize categorical variables, starting with the simplest, a single categorical variable.

5.2.1 Single Categorical Variable

Question: For a single categorical variable, how have you learn how you might visualize the data?

Barplots

Let's demonstrate barplots with the following data that is pulled from the General Social Survey (GSS) ((http://gss.norc.org/)). The GSS gathers data on contemporary American society via personal in-person interviews in order to monitor and explain trends and constants in attitudes, behaviors, and attributes over time. Hundreds of trends have been tracked since 1972. Each survey from 1972 to 2004 was an independently drawn sample of English-speaking persons 18 years of age or over, within the United States. Starting in 2006 Spanish-speakers were added to the target population. The GSS is the single best source for sociological and attitudinal trend data covering the United States.

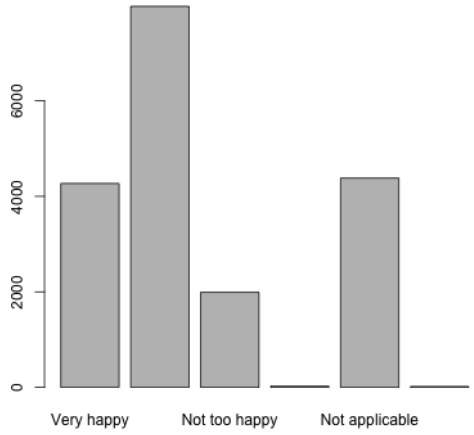
Here we look at a dataset where we have pulled out variables related to reported measures of well-being (based on a report about trends in psychological well-being (https://gssdataexplorer.norc.org/documents/903/display)). Like many surveys, the variables of interest are categorical.

Then we can compute a table and visualize it with a barplot.

```
table(wellbeingRecent$General.happiness)
```

```
##
##     Very happy   Pretty happy  Not too happy     Don't know Not applicable
##           4270           7979           1991             25           4383
##      No answer
##             18
```

```
barplot(table(wellbeingRecent$General.happiness))
```



Relationship between a categorical and continuous variable?

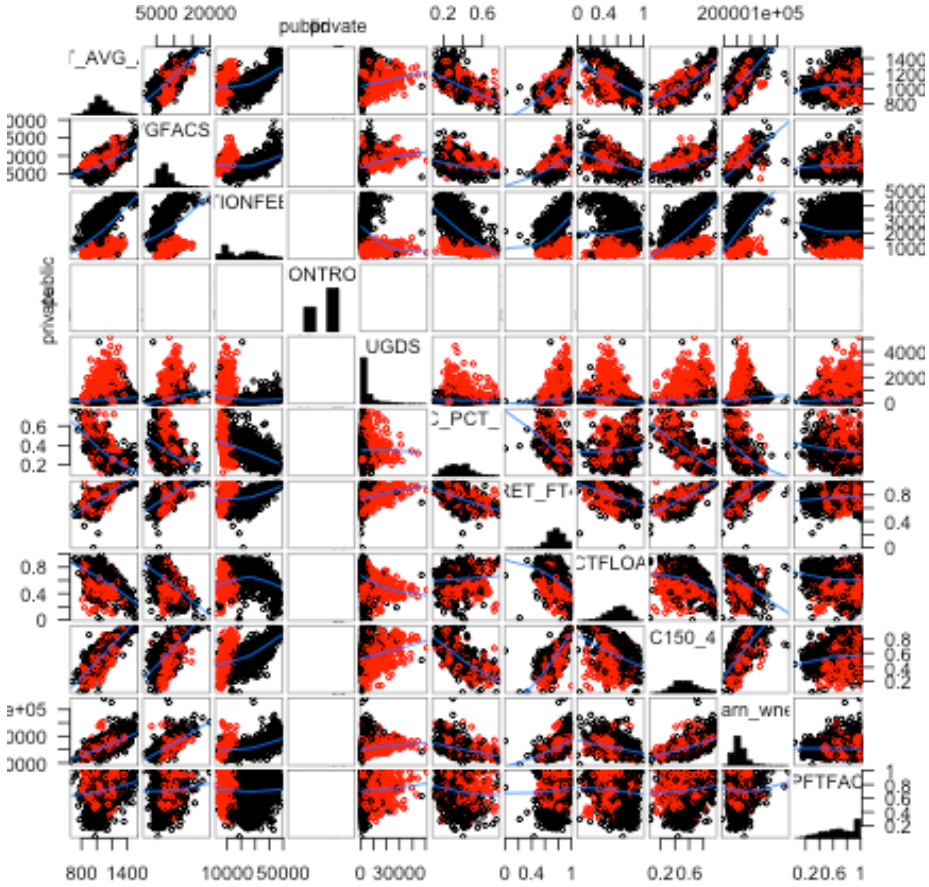
Recall from previous chapters, we discussed using how to visualize continuous data from different groups:

- Density plots
- Boxplots
- Violin plots

Numerical data that can be split into groups is just data with two variables, one continuous and one categorical.

Going back to our pairs plot of college, we can incorporate pairwise plotting of one continuous and one categorical variable using the function `gpairs` (in the package `gpairs`). This allows for more appropriate plots for our variable that separated public and private colleges.

```
library(gpairs)
smallScores$CONTROL <- factor(smallScores$CONTROL,
    levels = c(1, 2), labels = c("public", "private"))
gpairs(smallScores, lower.pars = list(scatter = "loess"),
    upper.pars = list(scatter = "loess", conditional = "boxplot"),
    scatter.pars = list(col = c("red", "black")[smallScores$CONTROL]))
```



5.2.2 Relationships between two (or more) categorical variables

When we get to two categorical variables, then the natural way to summarize their relationship is to cross-tabulate the values of the levels.

5.2.2.1 Cross-tabulations

You have seen that **contingency tables** are a table that give the cross-tabulation of two categorical variables.

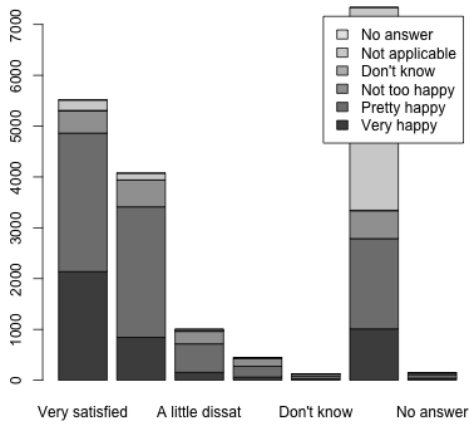
```
tabGeneralJob <- with(wellbeingRecent, table(General.happiness,
    Job.or.housework))
tabGeneralJob
```

```
##                   Job.or.housework
```

```
## General.happiness Very satisfied Mod. satisfied A little dissat
##    Very happy               2137            843             154
##    Pretty happy             2725           2569             562
##    Not too happy             436            527             247
##    Don't know                 11              1               4
##    Not applicable            204            134              36
##    No answer                   8              2               1
##                  Job.or.housework
## General.happiness Very dissatisfied Don't know Not applicable No answer
##    Very happy                   61         25            1011        39
##    Pretty happy                213         61            1776        73
##    Not too happy               161         39             549        32
##    Don't know                    0          1               8         0
##    Not applicable               12          1            3990         6
##    No answer                     3          0               4         0
```

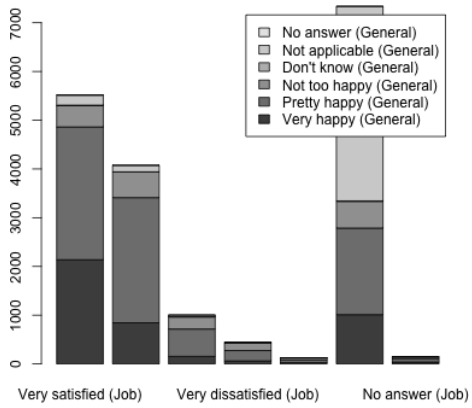
We can similarly make barplots to demonstrate these relationships.

```
barplot(tabGeneralJob, legend = TRUE)
```

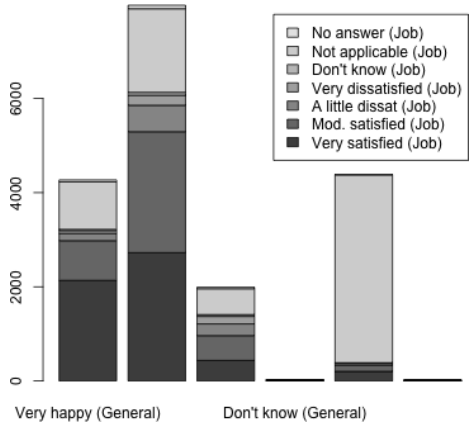


This barplot is not very satisfying. In particular, since the two variables have the same names for their levels, we don't know which is which!

```
colnames(tabGeneralJob) <- paste(colnames(tabGeneralJob),
    "(Job)")
rownames(tabGeneralJob) <- paste(rownames(tabGeneralJob),
    "(General)")
barplot(tabGeneralJob, legend = TRUE)
```

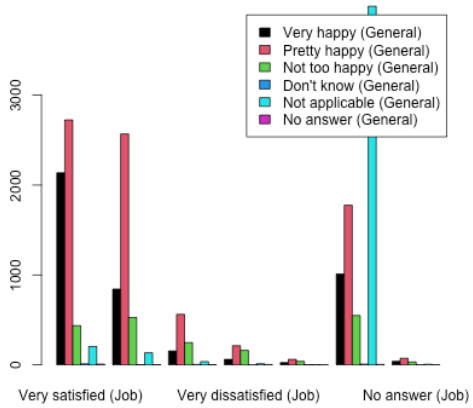



```
barplot(t(tabGeneralJob), legend = TRUE)
```



It can also be helpful to separate out the other variables, rather than stacking them, and to change the colors.

```
barplot(tabGeneralJob, beside = TRUE, legend = TRUE,
    col = palette()[1:6])
```



5.2.2.2 Conditional Distributions from Contingency Tables

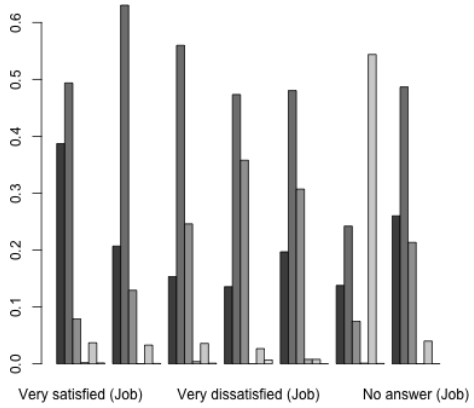
When we look at the contingency table, a natural question we ask is whether the distribution of the data changes across the different categories. For example, for people answering `Very Satisfied' for their job, there is a distribution of answers for the`General Happiness' question. And similarly for 'Moderately Satisfied'. We can get these by making the counts into proportions within each category.

```
prop.table(tabGeneralJob, margin = 2)
```

```
##                           Job.or.housework
## General.happiness          Very satisfied (Job) Mod. satisfied (Job)
##   Very happy (General)              0.3870675602         0.2068204122
##   Pretty happy (General)            0.4935700054         0.6302747792
##   Not too happy (General)           0.0789712009         0.1292934249
##   Don't know (General)              0.0019923927         0.0002453386
##   Not applicable (General)          0.0369498279         0.0328753680
##   No answer (General)               0.0014490129         0.0004906771
##                           Job.or.housework
## General.happiness          A little dissat (Job) Very dissatisfied (Job)
##   Very happy (General)              0.1533864542             0.1355555556
##   Pretty happy (General)            0.5597609562             0.4733333333
##   Not too happy (General)           0.2460159363             0.3577777778
##   Don't know (General)              0.0039840637             0.0000000000
##   Not applicable (General)          0.0358565737             0.0266666667
##   No answer (General)               0.0009960159             0.0066666667
##                           Job.or.housework
## General.happiness          Don't know (Job) Not applicable (Job)
##   Very happy (General)         0.1968503937          0.1377759608
##   Pretty happy (General)       0.4803149606          0.2420278005
##   Not too happy (General)      0.3070866142          0.0748160262
```

```
##   Don't know (General)          0.0078740157          0.0010902153
##   Not applicable (General)      0.0078740157          0.5437448896
##   No answer (General)           0.0000000000          0.0005451077
##                             Job.or.housework
## General.happiness            No answer (Job)
##   Very happy (General)          0.2600000000
##   Pretty happy (General)        0.4866666667
##   Not too happy (General)       0.2133333333
##   Don't know (General)          0.0000000000
##   Not applicable (General)      0.0400000000
##   No answer (General)           0.0000000000
```

```
barplot(prop.table(tabGeneralJob, margin = 2), beside = TRUE)
```



We could ask if these proportions are the same in each column (i.e. each level of `Job Satisfaction'). If so, then the value for`Job Satisfaction' is not affecting the answer for 'General Happiness', and so we would say the variables are unrelated.

Question: Looking at the barplot, what would you say? Are the variables related?

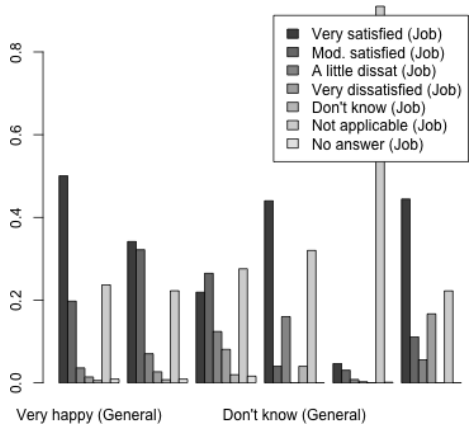
We can, of course, flip the variables around.

```
prop.table(tabGeneralJob, margin = 1)
```

```
##                             Job.or.housework
## General.happiness            Very satisfied (Job) Mod. satisfied (Job)
##   Very happy (General)                0.5004683841          0.1974238876
##   Pretty happy (General)              0.3415214939          0.3219701717
##   Not too happy (General)             0.2189854345          0.2646911100
##   Don't know (General)                0.4400000000          0.0400000000
##   Not applicable (General)            0.0465434634          0.0305726671
```

```
##   No answer (General)              0.4444444444         0.1111111111
##                            Job.or.housework
## General.happiness           A little dissat (Job) Very dissatisfied (Job)
##   Very happy (General)              0.0360655738            0.0142857143
##   Pretty happy (General)            0.0704348916            0.0266950746
##   Not too happy (General)           0.1240582622            0.0808638875
##   Don't know (General)              0.1600000000            0.0000000000
##   Not applicable (General)          0.0082135524            0.0027378508
##   No answer (General)               0.0555555556            0.1666666667
##                            Job.or.housework
## General.happiness           Don't know (Job) Not applicable (Job)
##   Very happy (General)         0.0058548009         0.2367681499
##   Pretty happy (General)       0.0076450683         0.2225842837
##   Not too happy (General)      0.0195881467         0.2757408338
##   Don't know (General)         0.0400000000         0.3200000000
##   Not applicable (General)     0.0002281542         0.9103353867
##   No answer (General)          0.0000000000         0.2222222222
##                            Job.or.housework
## General.happiness           No answer (Job)
##   Very happy (General)         0.0091334895
##   Pretty happy (General)       0.0091490162
##   Not too happy (General)      0.0160723255
##   Don't know (General)         0.0000000000
##   Not applicable (General)     0.0013689254
##   No answer (General)          0.0000000000
```

```
barplot(t(prop.table(tabGeneralJob, margin = 1)), beside = TRUE,
    legend = TRUE)
```



Notice that flipping this question gives me different proportions. This is because we are asking different question of the data. These are what we would call **Conditional Distributions**, and they depend on the order in which you condition your variables. The first plots show: conditional on being in a group

in Job Satisfaction, what is your probability of being in a particular group in General Happiness? That is different than what is shown in the second plot: conditional on being in a group in General Happiness, what is your probability of being in a particular group in Job Satisfaction?

5.2.3 Alluvial Plots

It can be complicated to look beyond two categorical variables. But we can create cross-tabulations for an arbitrary number of variables.

```
with(wellbeingRecent, table(General.happiness, Job.or.housework,
    Happiness.of.marriage))
```

This is not the nicest output once you start getting several variables. We can also use the `aggregate` command to calculate these same numbers, but not making them a table, but instead a data.frame where each row is a different cross-tabulation. This isn't helpful for looking at, but is an easier way to store and access the numbers.

```
wellbeingRecent$Freq <- 1
wellbeingAggregates <- aggregate(Freq ~ General.happiness +
    Job.or.housework, data = wellbeingRecent[, -2],
    FUN = sum)
head(wellbeingAggregates, 10)
```

```
##    General.happiness Job.or.housework Freq
## 1         Very happy   Very satisfied 2137
## 2       Pretty happy   Very satisfied 2725
## 3      Not too happy   Very satisfied  436
## 4         Don't know   Very satisfied   11
## 5     Not applicable   Very satisfied  204
## 6          No answer   Very satisfied    8
## 7         Very happy   Mod. satisfied  843
## 8       Pretty happy   Mod. satisfied 2569
## 9      Not too happy   Mod. satisfied  527
## 10        Don't know   Mod. satisfied    1
```

This format extends more easily to more variables:

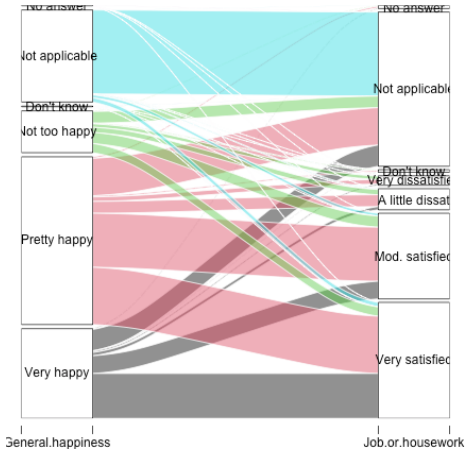
```
wellbeingAggregatesBig <- aggregate(Freq ~ General.happiness +
    Job.or.housework + Satisfaction.with.financial.situation +
    Happiness.of.marriage + Is.life.exciting.or.dull,
    data = wellbeingRecent[, -2], FUN = sum)
head(wellbeingAggregatesBig, 5)
```

```
##   General.happiness Job.or.housework Satisfaction.with.financial.situation
## 1        Very happy   Very satisfied                                Satisfied
```

```
## 2      Pretty happy   Very satisfied                                Satisfied
## 3     Not too happy   Very satisfied                                Satisfied
## 4        Very happy   Mod. satisfied                                Satisfied
## 5      Pretty happy   Mod. satisfied                                Satisfied
##   Happiness.of.marriage Is.life.exciting.or.dull Freq
## 1            Very happy                Exciting  333
## 2            Very happy                Exciting   54
## 3            Very happy                Exciting    3
## 4            Very happy                Exciting   83
## 5            Very happy                Exciting   38
```

An alluvial plot uses this input to try to track how different observations "flow" through the different variables. Consider this alluvial plot for the two variables 'General Happiness' and 'Satisfaction with Job or Housework'.

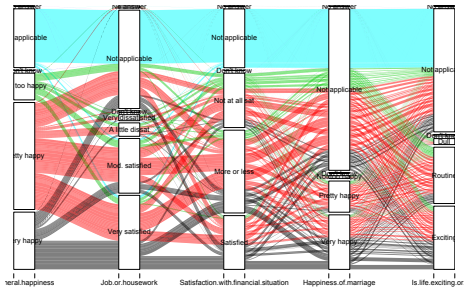
```
library(alluvial)
alluvial(wellbeingAggregates[, c("General.happiness",
    "Job.or.housework")], freq = wellbeingAggregates$Freq,
    col = palette()[wellbeingAggregates$General.happiness])
```



Notice how you can see the relative numbers that go through each category.

We can actually expand this to be many variables, though it gets to be a bit of a mess when you have many levels in each variable as we do. Moreover, this is a *very* slow command when you start adding additional variables, so I've run the following code off line and just saved the result:

```
alluvial(wellbeingAggregatesBig[, -ncol(wellbeingAggregatesBig)],
    freq = wellbeingAggregatesBig$Freq, col = palette()[wellbeingAggregatesBig$General
```



Putting aside the messiness, we can at least see some big things about the data. For example, we can see that there are a huge number of 'Not Applicable' for all of the questions. For some questions this makes sense, but for others is unclear why it's not applicable (few answer `Don't know'` `or`No answer)

Question: What other things can you see about the data from this plots?

These are obviously **self-reported** measures of happiness, meaning only what the respondent says is their state; these are not external, objective measures like measuring the level of a chemical in someone's blood (and indeed, with happiness, an objective, quantifiable measurement is hard!).

Question: What are some possible problems in interpreting these results?

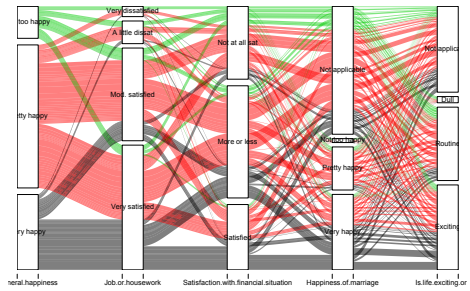
While you are generally stuck with some problems about self-reporting, there are other questions you could ask that might be more concrete and might suffer somewhat less from people instinct to say 'fine' to every question. For example, for marital happiness, you could ask questions like whether fighting more with your partner lately, feeling about partner's supportiveness, how often you tell your partner your feelings etc., that would perhaps get more specific responses. Of course, you would then be in a position of interpreting whether that adds up to a happy marriage when in fact a happy marriage is quite different for different couples!

Based on this plot, however, it does seem reasonable to exclude some of the categories as being unhelpful and adding additional complexity without being useful for interpretation. We will exclude observations that say `Not applicable' on all of these questions. We will also exclude those that do not answer or say`don't know' on any of these questions (considering non-response is quite important, as anyone who followed the problems with 2016 polls should know, but these are a small number of observations here).

I've also asked the alluvial plot to hide the very small categories, which makes it faster to plot. Again, this is slow, so I've created the plot off-line.

```
wh <- with(wellbeingRecent, which(General.happiness ==
    "Not applicable" | Job.or.housework == "Not applicable" |
    Satisfaction.with.financial.situation == "Not applicable"))
wellbeingCondenseGroups <- wellbeingRecent[-wh, ]
wellbeingCondenseGroups <- subset(wellbeingCondenseGroups,
    !General.happiness %in% c("No answer", "Don't know") &
        !Job.or.housework %in% c("No answer", "Don't know") &
        !Satisfaction.with.financial.situation %in%
            c("No answer", "Don't know") & !Happiness.of.marriage %in%
        c("No answer", "Don't know") & !Is.life.exciting.or.dull %in%
        c("No answer", "Don't know"))
wellbeingCondenseGroups <- droplevels(wellbeingCondenseGroups)
wellbeingCondenseAggregates <- aggregate(Freq ~ General.happiness +
    Job.or.housework + Satisfaction.with.financial.situation +
    Happiness.of.marriage + Is.life.exciting.or.dull,
    data = wellbeingCondenseGroups, FUN = sum)
```

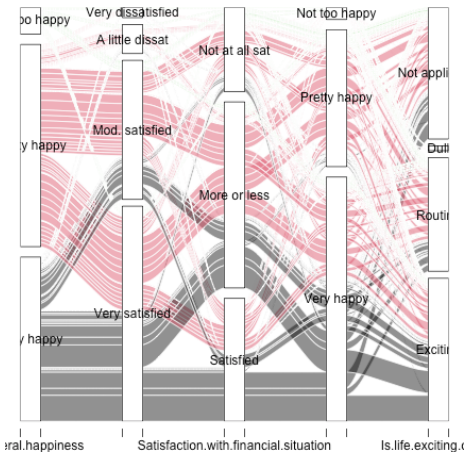
```
alluvial(wellbeingCondenseAggregates[, -ncol(wellbeingCondenseAggregates)],
    freq = wellbeingCondenseAggregates$Freq, hide = wellbeingCondenseAggregates$Freq <
        quantile(wellbeingCondenseAggregates$Freq,
            0.5), col = palette()[wellbeingCondenseAggregates$General.happiness])
```

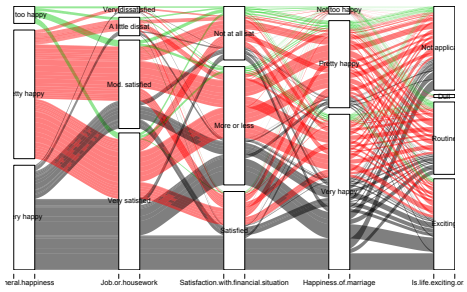


It's still rather messy, partly because we have large groups of people for whom some of the questions aren't applicable ('Happiness in marriage' only applies if you are married!) We can limit ourselves to just married, working individuals (including housework).

```
wh <- with(wellbeingCondenseGroups, which(Marital.status ==
    "Married" & Labor.force.status %in% c("Working fulltime",
    "Working parttime", "Keeping house")))
wellbeingMarried <- wellbeingCondenseGroups[wh, ]
wellbeingMarried <- droplevels(wellbeingMarried)
wellbeingMarriedAggregates <- aggregate(Freq ~ General.happiness +
    Job.or.housework + Satisfaction.with.financial.situation +
    Happiness.of.marriage + Is.life.exciting.or.dull,
    data = wellbeingMarried, FUN = sum)
```

```
alluvial(wellbeingMarriedAggregates[, -ncol(wellbeingMarriedAggregates)],
    freq = wellbeingMarriedAggregates$Freq, hide = wellbeingMarriedAggregates$Freq <
        quantile(wellbeingMarriedAggregates$Freq, 0.5),
    col = palette()[wellbeingMarriedAggregates$General.happiness])
```

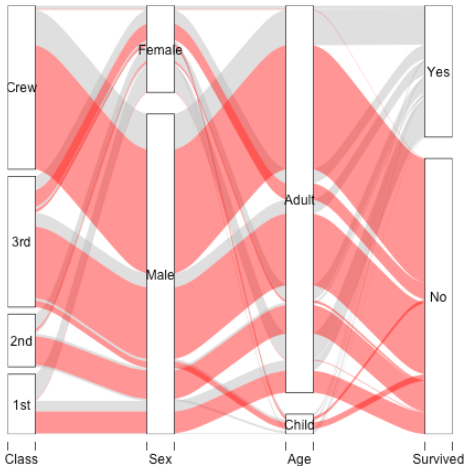




Cleaner example

The `alluvial` package comes with an example that provides a cleaner depiction of alluvial plots on several categories. They use data from the list of passangers on the Titantic disaster to demonstrate the demographic composition of those who survived.

```
data(Titanic)
tit <- as.data.frame(Titanic)
alluvial(tit[, 1:4], freq = tit$Freq, border = NA,
    col = ifelse(tit$Survived == "No", "red", "gray"))
```



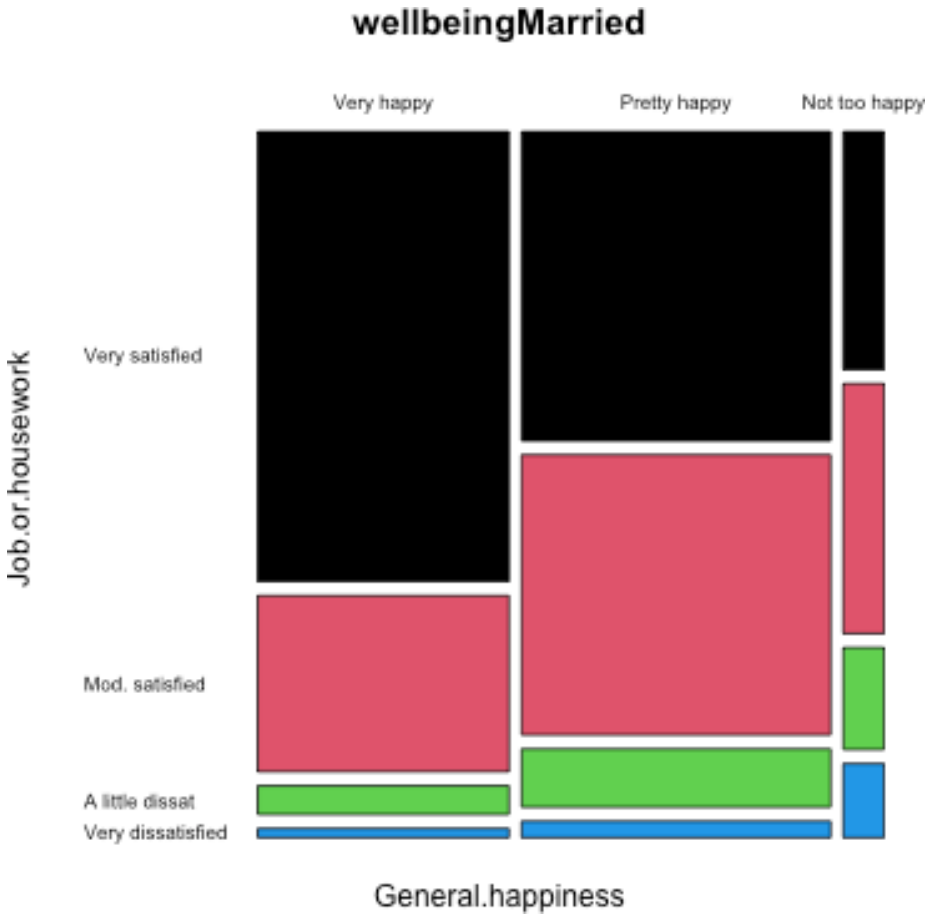
Like so many visualization tools, the effectiveness of a particular plot depends on the dataset.

5.2.4 Mosaic Plots

In looking at alluvial plots, we often turn to the question of asking whether the percentage, say happy in their jobs, is very different depending on whether they report that they are generally happy. Visualizing these percentages is often done better by a **mosaic** plot.

Let's first look at just 2 variables again.

```
mosaicplot(~General.happiness + Job.or.housework, data = wellbeingMarried,
    las = 1, col = palette())
```



How do we interpret this plot? Well first, like the plots above, these are showing *conditional dependencies*, so there is an order to these variables, based on how we put them in. First was General Happiness (x-axis). So the amount of space on the x-axis for `Very Happy' is proportional to the number of people who respondedVery` Happy' on the general happiness question. Next is `Job Satisfaction' (y-axis). *Within* each group of general happiness, the length on the y-axis is the proportion within that group answering each of the categories forJob` Satisfaction'. That is the conditional dependencies that we saw above.

Let's add a third variable, 'Satisfaction with financial situation'.

```
mosaicplot(~General.happiness + Job.or.housework +
    Satisfaction.with.financial.situation, data = wellbeingMarried,
    las = 1, col = palette())
```

This makes another subdivision on the x-axis. This is now subsetting down to the people, for example, that are very satisfied with both Job and their General life, and looking at the distribution of 'Satisfaction with financial situation' for just those set of people.

Question: Using this information, how do you interpret this plot? What does this tell you about people who are 'Very Happy' in general happiness?

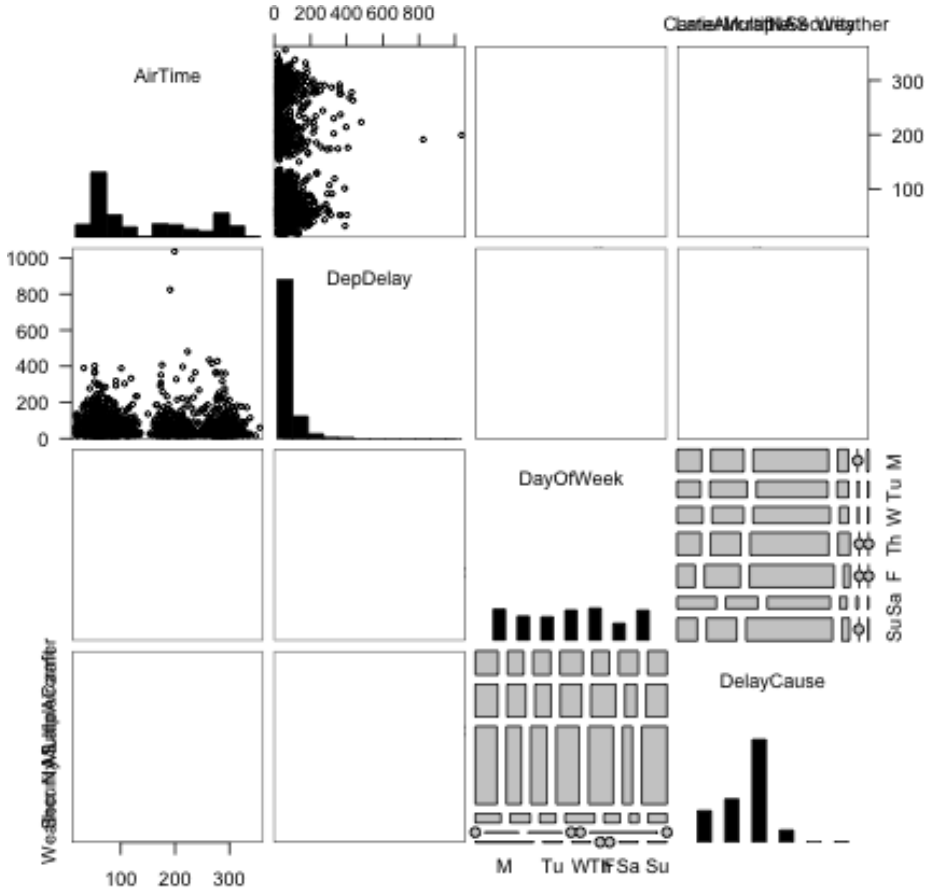
5.2.5 Pairs plots including categorical data

We can use some of these visualizations of categorical data in our pairs plots in the `gpairs` function. Our college data has only 1 categorical variable, and our well-being data has only categorical variables. So to have a mix of the two, we are going to return to our flight data, and bring in some variables that we didn't consider. We will also create a variable that indicates the cause of the delay

(there is no such variable, but only the amount of delay time due to different delay causes so we will use this information to create such a variable).

We will consider only delayed flights, and use `gpairs` to visualize the data.

```
gpairs(droplevels(flightSFOSRS[whDelayed, c("AirTime",
    "DepDelay", "DayOfWeek", "DelayCause")]), upper.pars = list(conditional = "boxplot"))
```



Question: How do you interpret the different elements of this pairs plot?

5.3 Heatmaps

Let's consider another dataset. This will consist of "gene expression" measurements on breast cancer tumors from the Cancer Genome Project. This data measures for all human genes the amount of each gene that is being used in

the tumor being measured. There are measurements for 19,000 genes but we limited ourselves to around 275 genes.

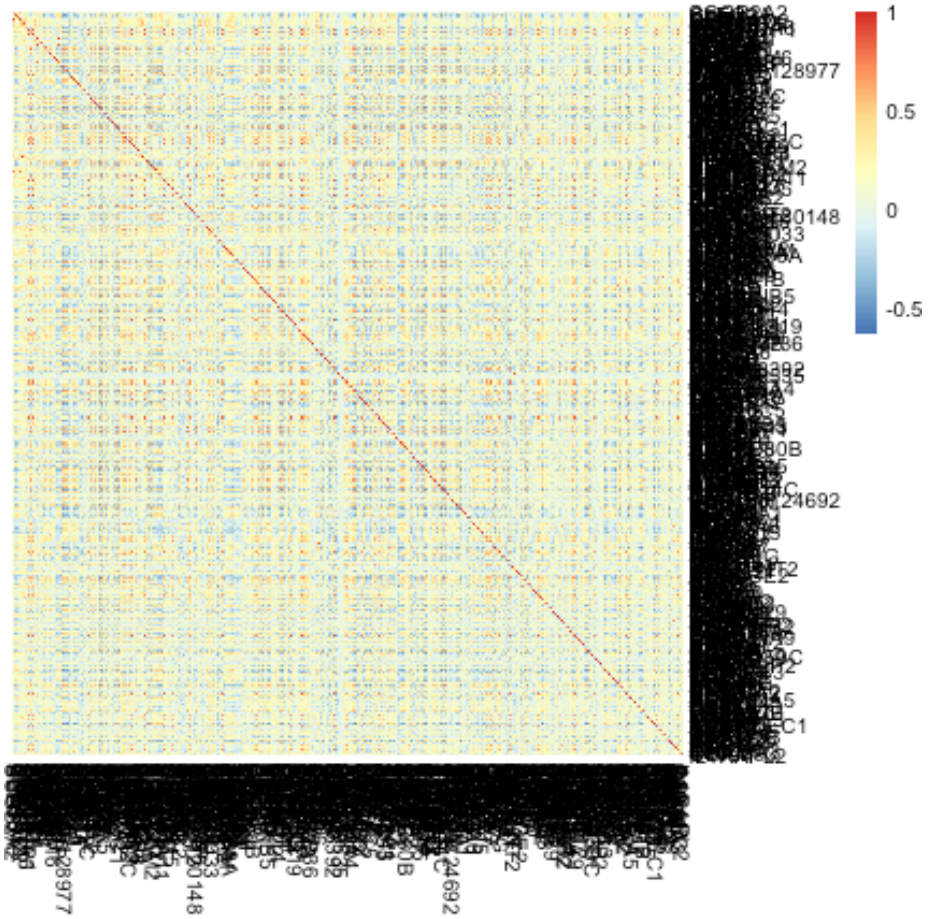
```
breast <- read.csv(file.path(dataDir, "highVarBreast.csv"),
    stringsAsFactors = TRUE)
```

One common goal of this kind of data is to be able to identify different types of breast cancers. The idea is that by looking at the genes in the tumor, we can discover similarities between the tumors, which might lead to discovering that some patients would respond better to certain kinds of treatment, for example.

We have so many variables, that we might consider simplifying our analysis and just considering the pairwise correlations of each variable (gene) – like the upper half of the pairs plot we drew before. Rather than put in numbers, which we couldn't easily read, we will put in colors to indicate the strength of the correlation. Representing a large matrix of data using a color scale is called a **heatmap**. Basically for any matrix, we visualize the entire matrix by putting a color for the value of the matrix.

In this case, our matrix is the matrix of correlations.

```
library(pheatmap)
corMat <- cor(breast[, -c(1:7)])
pheatmap(corMat, cluster_rows = FALSE, cluster_cols = FALSE)
```



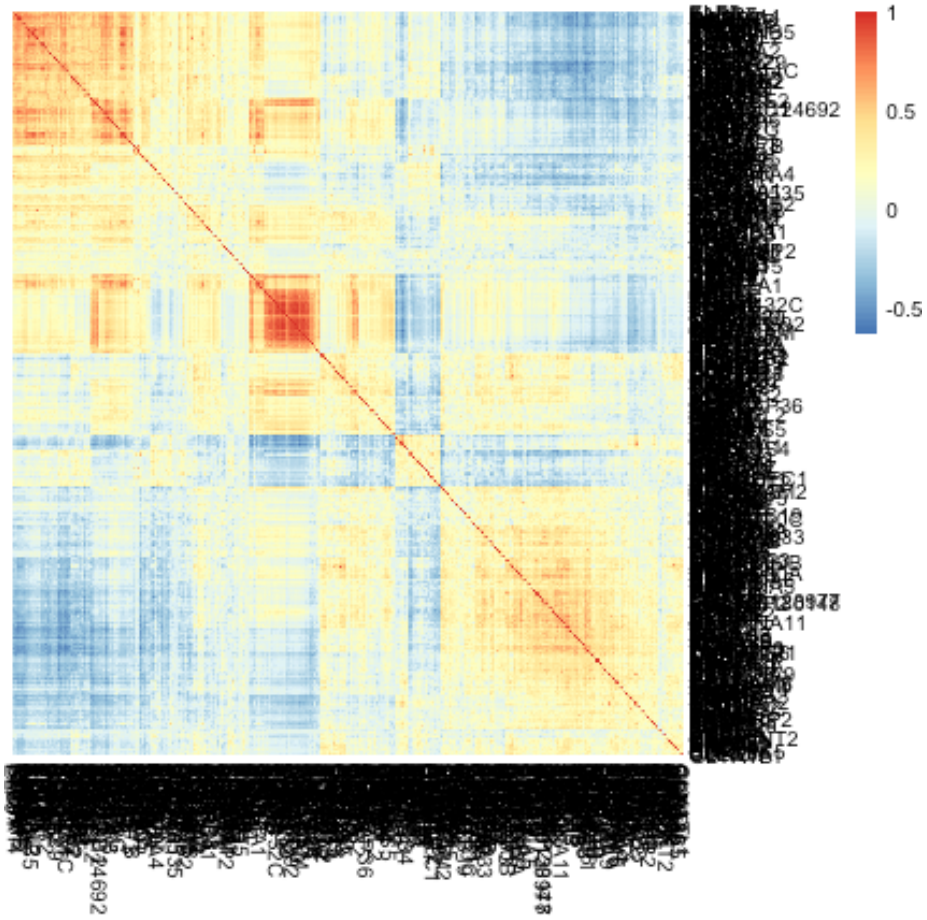
Question: Why is the diagonal all dark red?

This is not an informative picture, however – there are so many variables (genes) that we can't discover anything here.

However, if we could reorder the genes so that those that are highly correlated are near each other, we might see blocks of similar genes like we did before. In fact this is exactly what heatmaps usually do by default. They reorder the variables so that similar patterns are close to each other.

Here is the same plot of the correlation matrix, only now the rows and columns have been reordered.

```
pheatmap(corMat, cluster_rows = TRUE, cluster_cols = TRUE,
    treeheight_row = 0, treeheight_col = 0)
```

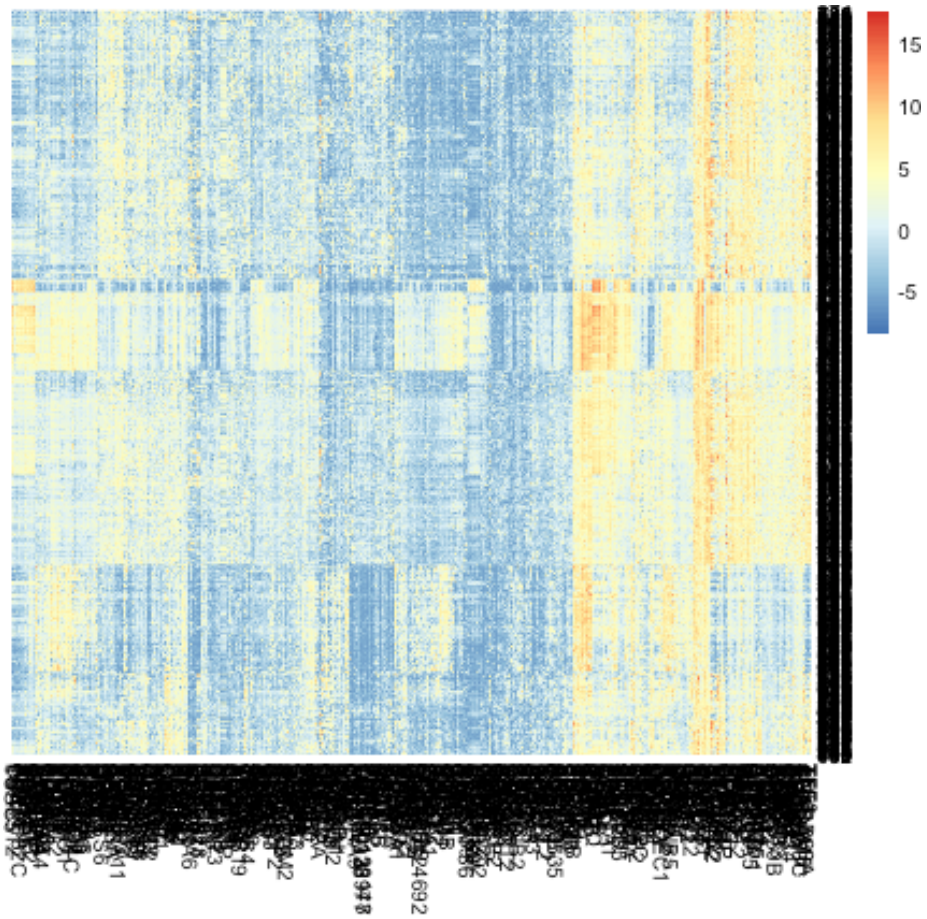


Question: What do we see in this heatmap?

5.3.1 Heatmaps for Data Matrices

Before we get into how that ordering was determined, lets consider heatmaps more. Heatmaps are general, and in fact can be used for the actual data matrix, not just the correlation matrix.

```
pheatmap(breast[, -c(1:7)], cluster_rows = TRUE, cluster_cols = TRUE,
    treeheight_row = 0, treeheight_col = 0)
```



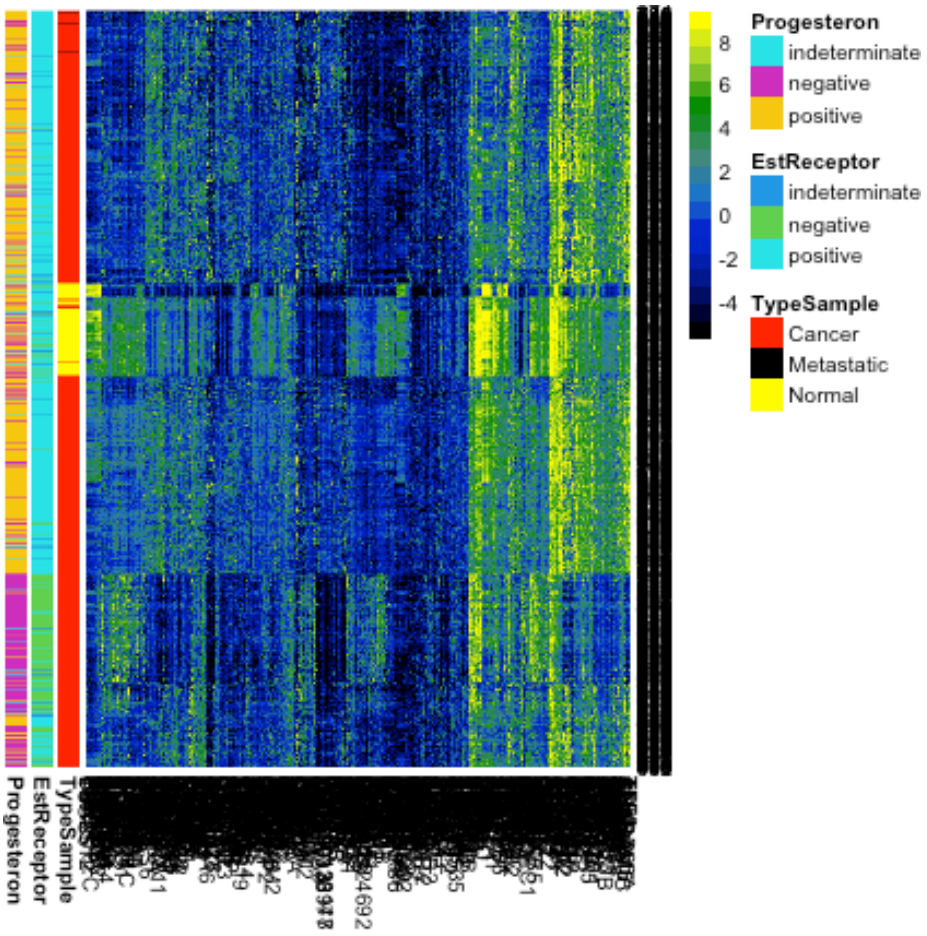
Question: What do we see in this heatmap?

We can improve upon this heatmap. I prefer different colors for this type of data, and we can add some information we have about these samples. I am also going to change how the heatmap assigns colors to the data. Specifically, heatmap gives a color for data by binning it and all data within a particular range of values gets a particular color. By default it is based on equally spaced bins across all of the data in the matrix – sort of like a histogram. However, this can frequently backfire if you have a few outlying points. One big value will force the range to cover it. The effect of this can be that most of the data is only in a small range of colors, so you get a heatmap where everything is mostly one color, so you don't see much. I am going to change it so that most of the bins go from the 1% to the 99% quantile of data, and then there is one end bin on each end that covers all of the remaining large values.

```
typeCol <- c("red", "black", "yellow")
names(typeCol) <- levels(breast$TypeSample)
estCol <- palette()[c(4, 3, 5)]
names(estCol) <- levels(breast$EstReceptor)
proCol <- palette()[5:7]
names(proCol) <- levels(breast$Progesteron)
qnt <- quantile(as.numeric(data.matrix((breast[, -c(1:7)]))),
    c(0.01, 0.99))
brks <- seq(qnt[1], qnt[2], length = 20)
head(brks)
```

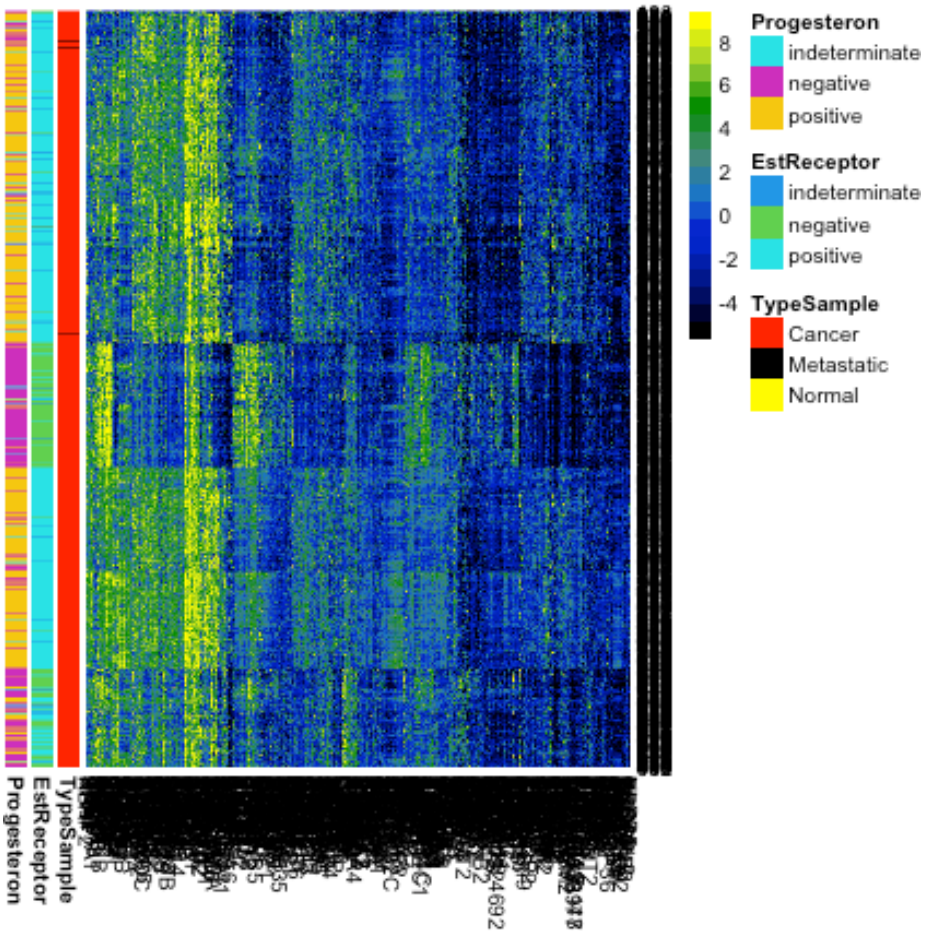
```
## [1] -5.744770 -4.949516 -4.154261 -3.359006 -2.563751 -1.768496
```

```
seqPal5 <- colorRampPalette(c("black", "navyblue",
    "mediumblue", "dodgerblue3", "aquamarine4", "green4",
    "yellowgreen", "yellow"))(length(brks) - 1)
row.names(breast) <- c(1:nrow(breast))
fullHeat <- pheatmap(breast[, -c(1:7)], cluster_rows = TRUE,
    cluster_cols = TRUE, treeheight_row = 0, treeheight_col = 0,
    color = seqPal5, breaks = brks, annotation_row = breast[,
        5:7], annotation_colors = list(TypeSample = typeCol,
        EstReceptor = estCol, Progesteron = proCol))
```



Question: What does adding this information allow us to see now?

```
whCancer <- which(breast$Type != "Normal")
pheatmap(breast[whCancer, -c(1:7)], cluster_rows = TRUE,
    cluster_cols = TRUE, treeheight_row = 0, treeheight_col = 0,
    color = seqPal5, breaks = brks, annotation_row = breast[whCancer,
        5:7], annotation_colors = list(TypeSample = typeCol,
        EstReceptor = estCol, Progesteron = proCol))
```



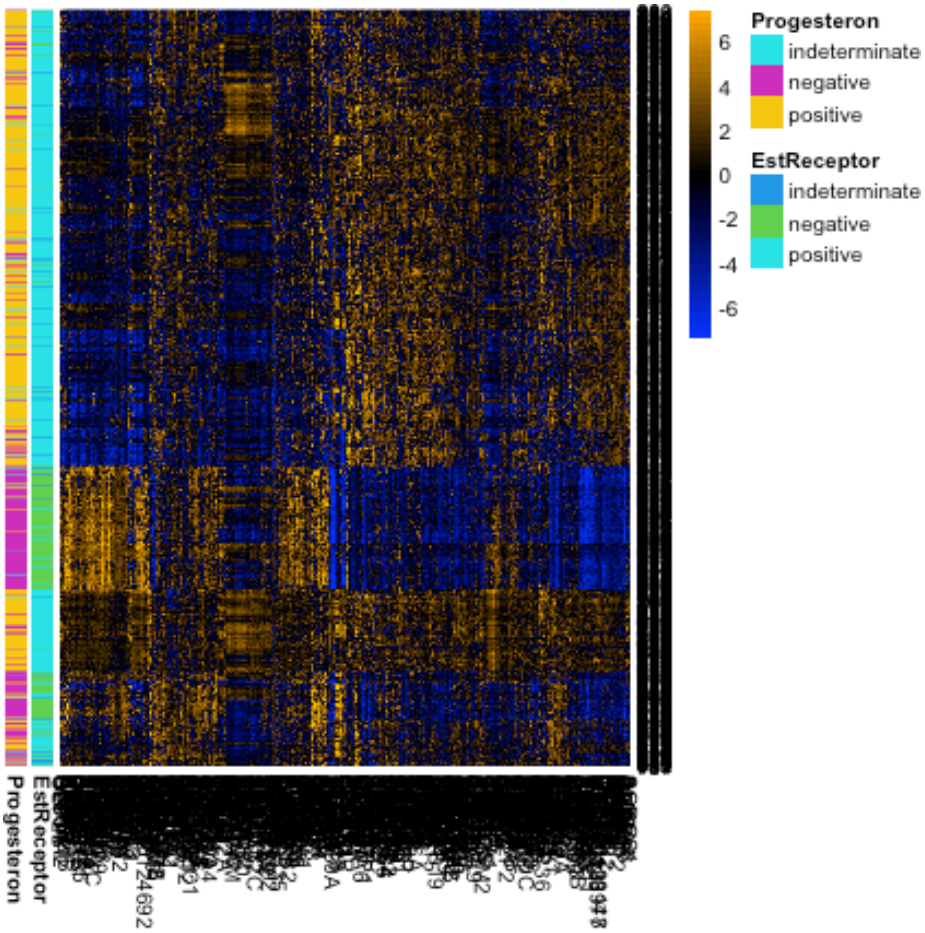
Centering/Scaling Variables

Some genes have drastic differences in their measurements for different samples. But we might also can notice that many of the genes are all high, or all low. They might show similar patterns of differences, but at a lesser scale. It would be nice to put them on the same basis. A simple way to do this is to subtract the mean or median of each variable.

Notice our previous breaks don't make sense for this centered data. Moreover, now that we've centered the data, it makes sense to make the color scale symmetric around 0, and also to have a color scale that emphasizes zero.

Question: Why this focus on being centered around zero?

```
breastCenteredMean <- scale(breast[, -c(1:7)], center = TRUE,
    scale = FALSE)
colMedian <- apply(breast[, -c(1:7)], 2, median)
breastCenteredMed <- sweep(breast[, -c(1:7)], MARGIN = 2,
    colMedian, "-")
qnt <- max(abs(quantile(as.numeric(data.matrix((breastCenteredMed[,
    -c(1:7)]))), c(0.01, 0.99))))
brksCentered <- seq(-qnt, qnt, length = 50)
seqPal2 <- colorRampPalette(c("orange", "black", "blue"))(length(brksCentered) -
    1)
seqPal2 <- (c("yellow", "gold2", seqPal2))
seqPal2 <- rev(seqPal2)
pheatmap(breastCenteredMed[whCancer, -c(1:7)], cluster_rows = TRUE,
    cluster_cols = TRUE, treeheight_row = 0, treeheight_col = 0,
    color = seqPal2, breaks = brksCentered, annotation_row = breast[whCancer,
        6:7], annotation_colors = list(TypeSample = typeCol,
        EstReceptor = estCol, Progesteron = proCol))
```



We could also make their range similar by scaling them to have a similar variance. This is helpful when your variables are really on different scales, for example weights in kg and heights in meters. This helps put them on a comparable scale for visualizing the patterns with the heatmap. For this gene expression data, the scale is more roughly similar, though it is common in practice that people will scale them as well for heatmaps.

5.3.2 Hierarchical Clustering

How do heatmaps find the ordering of the samples and genes? It performs a form of clustering on the samples. Let's get an idea of how clustering works generally, and then we'll return to heatmaps.

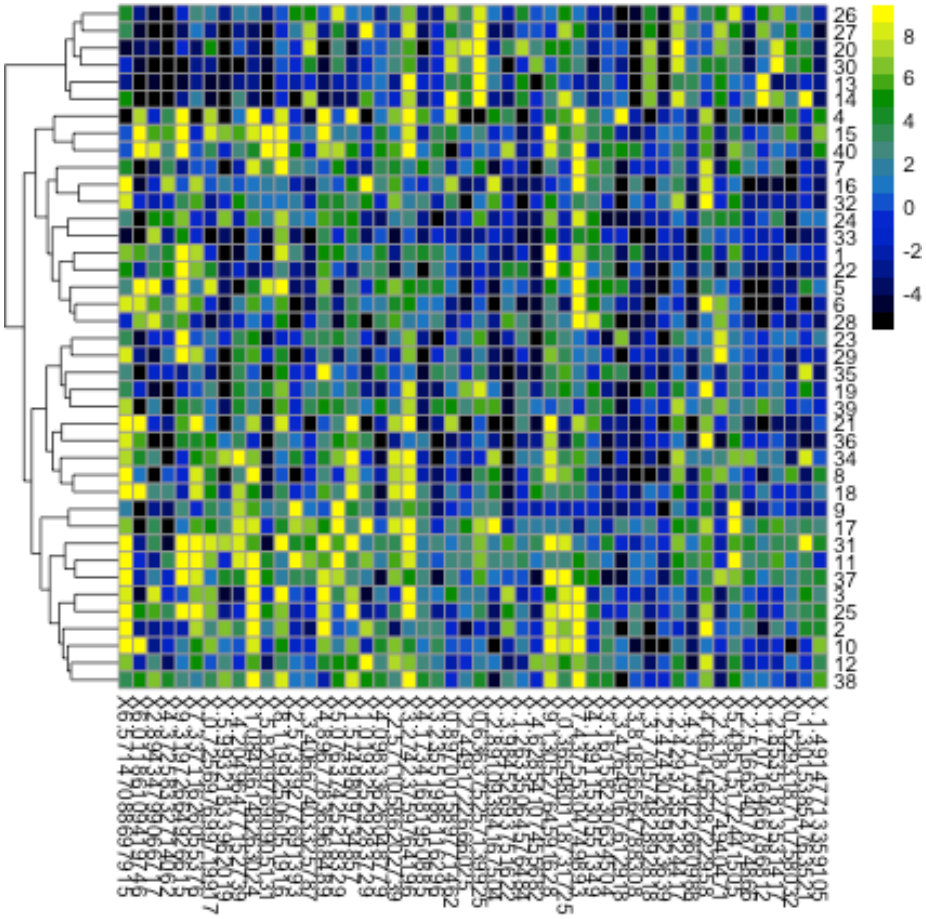
The idea behind clustering is that there is an unknown variable that would tell you the 'true' groups of the samples, and you want to find it. This may not

actually be true in practice, but it's a useful abstraction. The basic idea of clustering relies on examining the distances between samples and putting into the same cluster samples that are close together. There are countless number of clustering algorithms, but heatmaps rely on what is called **hierarchical clustering**. It is called hiearchical clustering because it not only puts observations into groups/clusters, but does so by first creating a hierarchical tree or **dendrogram** that relates the samples.

Here we show this on a small subset of the samples and genes. We see on the left the dendrogram that relates the samples (rows).[1]

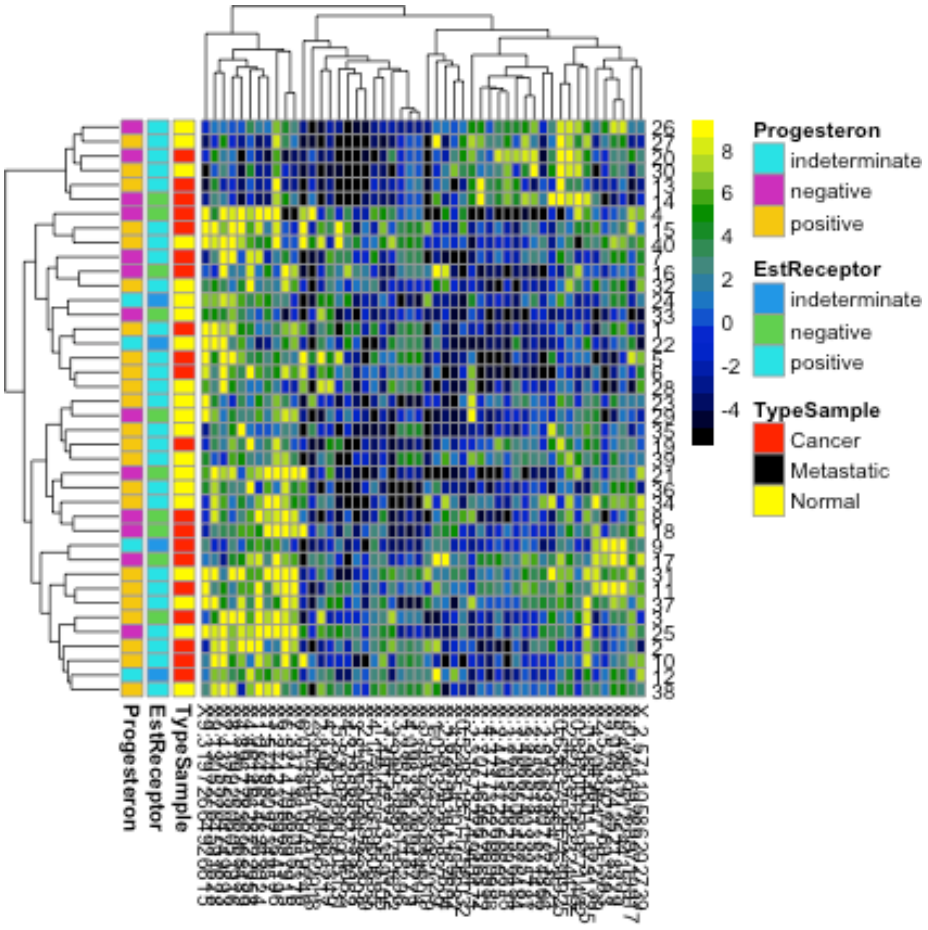
```
smallBreast <- read.csv(file.path(dataDir, "smallVarBreast.csv"),
    header = TRUE, stringsAsFactors = TRUE)
row.names(smallBreast) <- 1:nrow(smallBreast)
pheatmap(smallBreast[, -c(1:7)], cluster_rows = TRUE,
    cluster_cols = FALSE, treeheight_col = 0, breaks = brks,
    col = seqPal5)
```

[1] I have also clustered the variables (columns) in this figure because otherwise it is hard to see anything, but have suppressed the drawing of the dendrogram to focus on the samples – see the next figure where we draw both.



We can use the same principle for clustering the variables:

```
pheatmap(smallBreast[, -c(1:7)], , cluster_rows = TRUE,
    cluster_cols = TRUE, breaks = brks, col = seqPal5,
    annotation_row = smallBreast[, 5:7], annotation_colors = list(TypeSample = typeCol
        EstReceptor = estCol, Progesteron = proCol))
```



Notice that with this small subset of genes and samples, we don't see the same discrimination between normal and cancer samples.

Where are the clusters?

If hierarchical clustering is a clustering routine, where are the clusters? The idea is that the dendrogram is just a first step toward clustering. To get a cluster, you draw a line across the dendrogram to "cut" the dendrogram into pieces, which correspond to the clusters. For the purposes of a heatmap, however, what is interesting is not the clusters, but ordering of the samples that it provides.

5.3.2.1 How Hierarchical Clustering Works

Hierarchical clustering is an iterative process, that builds the dendrogram by *iteratively* creating new groups of samples by either

1. joining pairs of individual samples into a group

2. add an individual samples to an existing group
3. combine two groups into a larger group[2]

Step 1: Pairwise distance matrix between groups We consider each sample to be a separate group (i.e. n groups), and we calculate the pairwise distances between all of the n groups.

For simplicity, let's assume we have only one variable, so our data is $y_1, \ldots, y_n$. Then the standard distance between samples i and j could be

$$d_{ij} = |y_i - y_j|$$

or alternatively squared distance,

$$d_{ij} = (y_i - y_j)^2.$$

So we can get all of the pairwise distances between all of the samples (a distance matrix of all the $n \times n$ pairs)

Step 2: Make group by joining together two closest "groups" Your available choices from the list above are to join together two samples to make a group. So we choose to join together the two samples that are closest together, and forming our first real group of samples.

Step 3: Update distance matrix between groups Specifically, say you have already joined together samples i and j to make the first true group. To join update our groups, our options from the list above are:

1. Combine two samples k and ℓ to make next group (i.e. do nothing with the group previously formed by i and j.
2. Combine some sample k with your new group

Clearly, if we join together two samples k and ℓ it's the same as above (pick two closest). But how do you decide to do that versus add sample k to my group of samples i and j? We need to decide whether a sample k is closer to the group consisting of i and j than it is to any other sample ℓ.

We do this by recalculating the pairwise distances we had before, replacing these two samples i and j by the pairwise distance of the new *group* to the other samples.

Of course this is easier said than done, because how do we define how close a group is to other samples or groups? There's no single way to do that, and in fact there are a lot of competing methods. The default method in R is to say that if we have a group $\mathcal{G}$ consisting of i and j, then the distance of that group to a sample k is the maximum distance of i and j to k[3] ,

$$d(\mathcal{G}, k) = \max(d_{ik}, d_{jk}).$$

[2]This is called an agglomerative method, where you start at the bottom of the tree and build up. There are also divisive method for creating a hiearchical tree that starts at the "top" by continually dividing the samples into two group.

[3]This is called complete linkage.

Now we have a updated $n-1 \times n-1$ matrix of distances between all our current list of "groups" (remember the single samples form their own group).

Step 4: Join closest groups Now we find the closest two groups and join the samples in the group together to form a new group.

Step 5+: Continue to update distance matrix and join groups Then you repeat this process of joining together to build up the tree. Once you get more than two groups, you will consider all of the three different kinds of joins described above – i.e. you will also consider joining together two existing groups $\mathcal{G}_1$ and $\mathcal{G}_2$ that both consist of multiple samples. Again, you generalize the definition above to define the distance between the two groups of samples to be the maximum distance of all the points in $\mathcal{G}_1$ to all the points in $\mathcal{G}_2$,

$$d(\mathcal{G}_1, \mathcal{G}_2) = \max_{i \in \mathcal{G}_1, j \in \mathcal{G}_2} d_{ij}.$$

Distances in Higher Dimensions

The same process works if instead of having a single number, your y_i are now vectors – i.e. multiple variables. You just need a definition for the distance between the y_i, and then follow the same algorithm.

What is the equivalent distance when you have more variables? For each variable ℓ, we observe $y_1^{(\ell)}, \ldots, y_n^{(\ell)}$. And an observation is now the vector that is the collection of all the variables for the sample:

$$y_i = (y_i^{(1)}, \ldots, y_i^{(p)})$$

We want to find the distance between observations i and j which have vectors of data

$$(y_i^{(1)}, \ldots, y_i^{(p)})$$

and

$$(y_j^{(1)}, \ldots, y_j^{(p)})$$

The standard distance (called Euclidean distance) is

$$d_{ij} = d(y_i, y_j) = \sqrt{\sum_{\ell=1}^{p} (y_i^{(\ell)} - y_j^{(\ell)})^2}$$

So its the cummulative (i.e. sum) amount of the individual (squared) distance of each variable. You don't have to use this distance – there are other choices that can be better depending on the data – but it is the default.

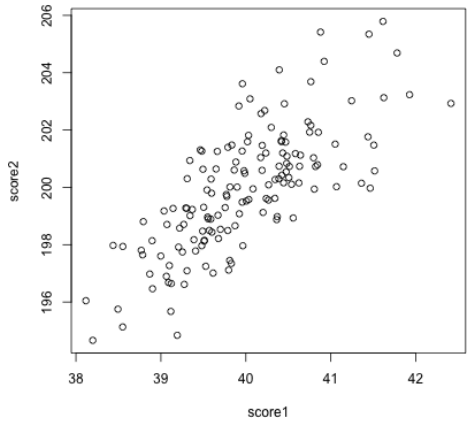
We generally work with squared distances, which would be

$$d_{ij}^2 = \sum_{\ell=1}^{p} (y_i^{(\ell)} - y_j^{(\ell)})^2$$

5.4 Principal Components Analysis

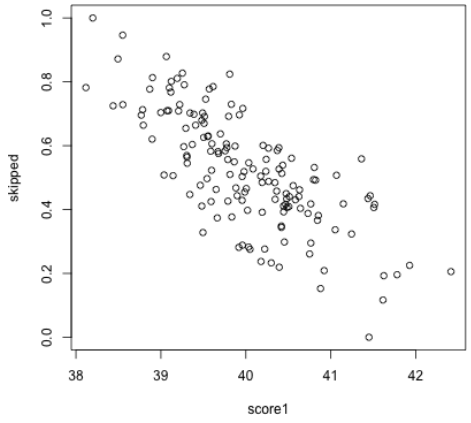
In looking at both the college data and the gene expression data, it is clear that there is a lot of redundancy in our variables, meaning that several variables are often giving us the same information about the patterns in our observations. We could see this by looking at their correlations, or by seeing their values in a heatmap.

For the purposes of illustration, let's consider a hypothetical situation. Say that you are teaching a course, and there are two exams:



These are clearly pretty redundant information, in the sense that if I know a student has a high score in exam 1, I know they are a top student, and exam 2 gives me that same information.

Consider another simulated example. Say the first value is the midterm score of a student, and the next value is the percentage of class and labs the student skipped. These are negatively correlated, but still quite redundant.



The goal of principal components analysis is to reduce your set of variables into

the most informative. One way is of course to just manually pick a subset. But which ones? And don't we do better with more information – we've seen that averaging together multiple noisy sources of information gives us a better estimate of the truth than a single one. The same principle should hold for our variables; if the variables are measuring the same underlying principle, then we should do better to use all of the variables.

Therefore, rather than picking a subset of the variables, principal components analysis *creates new variables* from the existing variables.

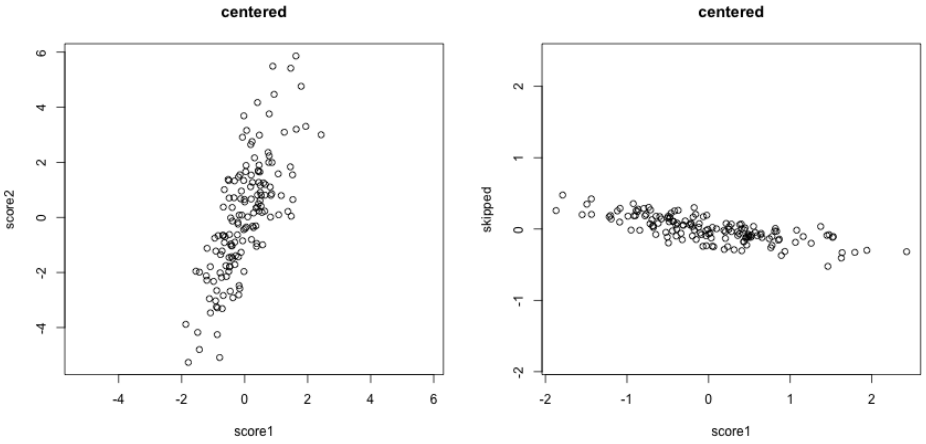
There are two *equivalent* ways to think about how principal components analysis does this.

5.4.1 Linear combinations of existing variables

You want to find a single score for each observation that is a summary of your variables. We will first consider as a running example the simple setting of finding a summary for a student with two grades, but the power is really when you want to find a summary for a lot of variables, like with the college data or the breast cancer data.

Question: What is the problem with taking the mean of our two exam scores?

Let's assume we make them have the same mean:



Question: What problem remains?

If we are taking the mean, we are treating our two variables $x^{(1)}$ and $x^{(2)}$ equally, so that we have a new variable z that is given by

$$z_i = \frac{1}{2}x_i^{(1)} + \frac{1}{2}x_i^{(2)}$$

The idea with principal components, then, is that we want to weight them differently to take into account the scale and whether they are negatively or positively correlated.

$$z_i = a_1 x_i^{(1)} + a_2 x_i^{(2)}$$

So the idea of principal components is to find the "best" constants (or coefficients), a_1 and a_2. This is a little bit like regression, only in regression I had a response y_i, and so my best coefficients were the best predictors of y_i. Here I don't have a response. I only have the variables, and I want to get the best summary of them, so we will need a new definition of "best".

So how do we pick the best set of coefficients? Similar to regression, we need a criteria for what is the best set of coefficients. Once we choose the criteria, the computer can run an optimization technique to find the coefficients. So what is a reasonable criteria?

If I consider the question of exam scores, what is my goal? Well, I would like a final score that separates out the students so that the students that do much better than the other students are further apart, etc. %Score 2 scrunches most of the students up, so the vertical line doesn't meet that criteria.

The criteria in principal components is to find the line so that the new variable values have the most variance – so we can spread out the observations the most. So the criteria we choose is to maximize the sample variance *of the resulting* z.

In other words, for every set of coefficients a_1, a_2, we will get a set of n new values for my observations, $z_1, \ldots, z_n$. We can think of this new z as a new variable.

Then for any set of cofficients, I can calculate the sample variance of my resulting z as

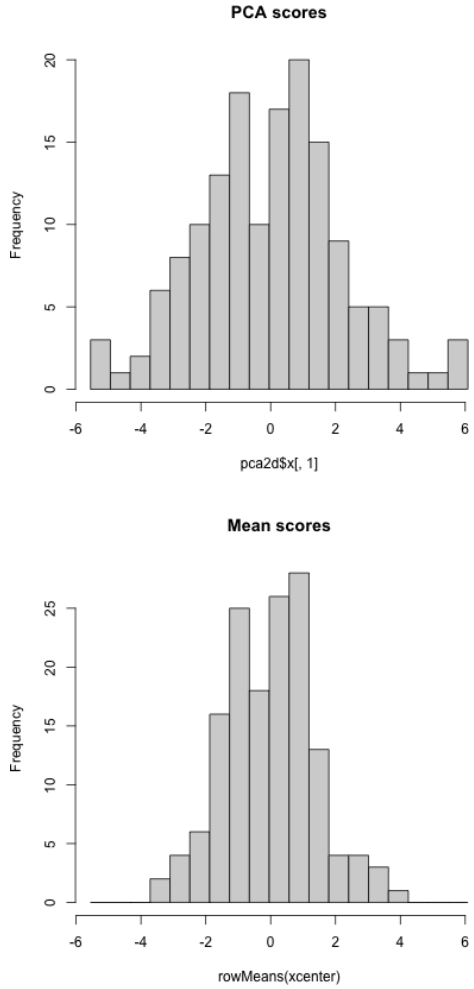
$$\hat{var}(z) = \frac{1}{n-1}\sum_{i=1}^{n}(z_i - \bar{z})^2$$

Of course, $z_i = a_1 x_i^{(1)} + a_2 x_i^{(2)}$, this is actually

$$\hat{var}(z) = \frac{1}{n-1}\sum_{i=1}^{n}(a_1 x_i^{(1)} + a_2 x_i^{(2)} - \bar{z})^2$$

(I haven't written out $\bar{z}$ in terms of the coefficients, but you get the idea.) Now that I have this criteria, I can use optimization routines implemented in the computer to find the coefficients that maximize this quantity.

Here is a histogram of the PCA variable z and that of the mean.



Why *maximize* variance – isn't that wrong?

We often talk about PCA as "preserving" the variance in our data. But in many settings we say we want low variability, so it frequently seems wrong to students to maximize the variance. But this is because frequently we think of variability as the same thing as noise. But variability among samples should only be considered noise among *homogeneous* samples, i.e. samples there are no interesting reasons for why they should be different. Otherwise we can have variability in our variables due to important differences between our observations, like what job title our employees have in the SF data in Chapter 2. We can see this in our data examples above, where we see different meaningful groups are separated from each other, such as cancer and normal patients. Genes that have a lot of differences between cancer and normal will have a large amount of spread. The difference in the groups is creating a large spread in our observations. Capturing

the variance in PCA is capturing these meaningful differences, as we can see in our above examples.

5.4.1.1 More than 2 variables

This procedure expands easily to more than 2 variables. Specifically assume that our observation i is a vector of values, $(x_i^{(1)}, ..., x_i^{(p)})$ where p is the number of variables. With PCA, I am looking for a **linear combination** of these p variables. As before, this means we want to multiply each variable by a coefficient and add them together (or subtract if the coefficient is negative) to get a new variable z for each observation,

$$z_i = a_1 x_i^{(1)} + ... + a_p x_i^{(p)}$$

So finding a linear combination is equivalent to finding a set of the p constants that I will multiply my variables by.

Question: If I take the mean of my p variables, what are my choices of a_k for each of my variables?

I can similarly find the coefficients a_k so that my resulting z_i have maximum variance.

PCA is really most powerful when considering many variables.

Unique only up to a sign change

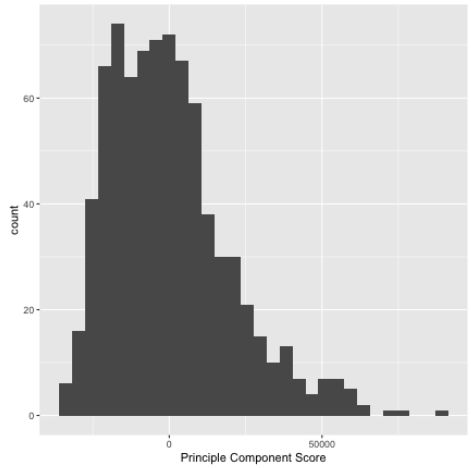
Notice that if I multiplied *all* of the coefficients a_k by -1, then z_i will become $-z_i$. However, the variance of $-z_i$ will be the same as the variance of z_i, so either answer is equivalent. In general, PCA will only give a unique score z_i *up to a sign change.*

Question: You do NOT get the same answer if you multiply only *some* a_k by -1, why?

Example: Scorecard Data

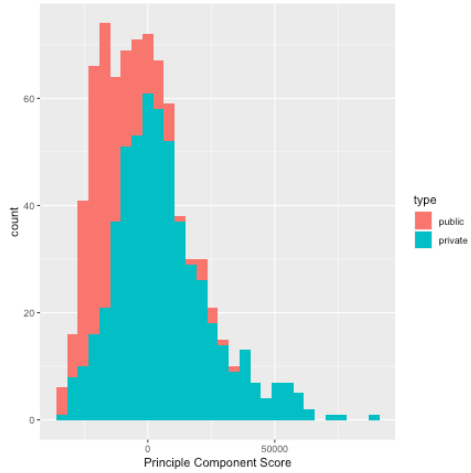
Consider, for example, our scorecard of colleges data, where we previously only considered the pairwise scatterplots. There are 30 variables collected on each institution – too many to easily visualize. We will consider a PCA summary of all of this data that will incorporate all of these variables. Notice that PCA only makes sense for continuous variables, so we will remove variables (like the private/public split) that are not continuous. PCA also doesn't handle NA values, so I have removed samples that have NA values in any of the observations.

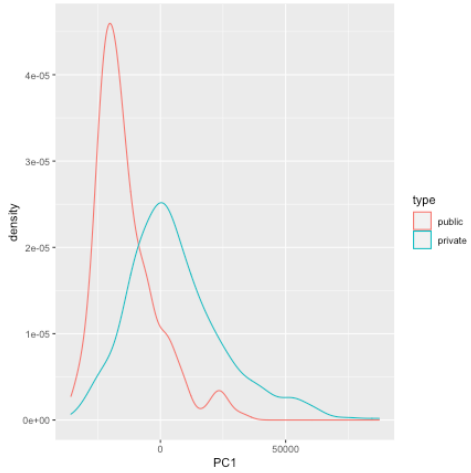
I can plot a histogram of the scores that each observation received:



We can see that some observations have quite outlying scores. When I manually look at their original data, I see that these scores have very large (or very small) values for most of the variables, so it makes sense that they have outlying scores.

I can also compare whether public and private colleges seem to be given different scores:

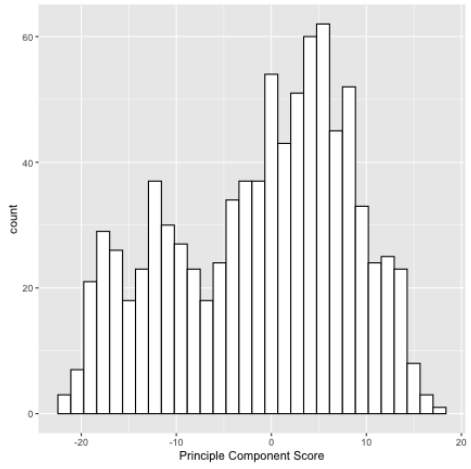




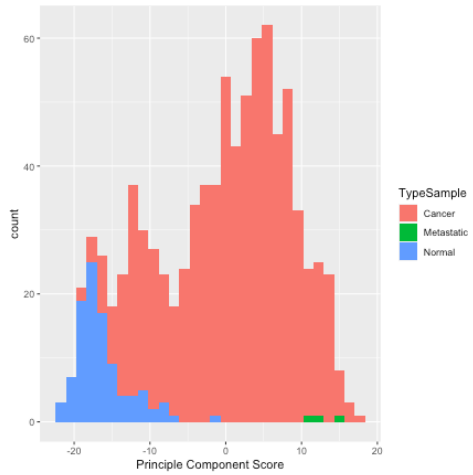
We can see some division between the two, with public seeming to have lower scores than the private. Notice that I only care about relative values here – if I multiplied all of my coefficients a_k by -1, then I would flip which is lower or higher, but it would be equivalent in terms of the variance of my new scores z_i. So it does not mean that public schools are more likely to have lower values on any particular variables; it does mean that public schools tend to have values that are in a different *direction* than private schools on some variables.

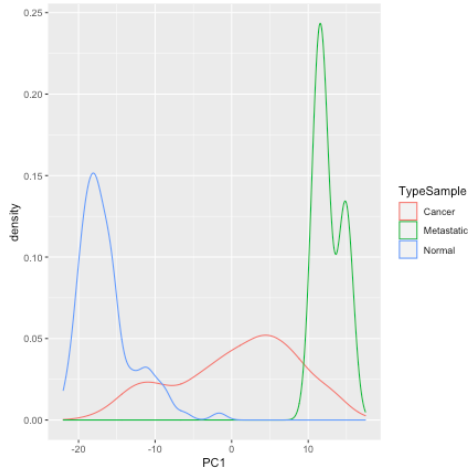
Example: Breast Cancer Data

Similarly we can see a big difference between cancer and normal observations in the first two principal components.

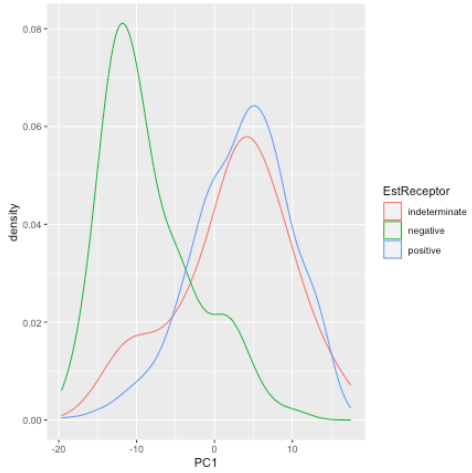


We can see that, at least based on the PC score, there might be multiple groups in this data, because there are multiple modes. We could explore the scores of normal versus cancer samples, for example:





We can also see that cancer samples are really spread out; we have other variables that are particularly relevant for separating cancer samples, so we could see how they differ. For example, by separating estrogen receptor status, we see quite different distributions:



In summary, based on our PCA score, I can visually explore important patterns in my data, even with very large numbers of variables. Because I know that this is the linear combination that most spreads out my observations, hopefully large shared differences between our samples (like normal vs cancer, or outlying observations) will be detected, particularly if they are reiterated in many variables.

5.4.1.2 Multiple principal components

So far we have found a single score to summarize our data. But we might consider that a single score is not going to capture the complexity in the data. For example, for the breast cancer data, we know that the normal and cancer samples are quite distinct. But we also know that within the cancer patients, those negative on the Estrogen Receptor or Progesteron are themselves a subgroup within cancer patients, in terms of their gene measurements. Capturing these distinctions with a single score might be to difficult.

Specifically, for each observation i, we have previously calculated a single score,

$$z = a_1 x^{(1)} + ... + a_p x^{(p)}$$

What if instead we want two scores for each observation i, i.e. $(z_i^{(1)}, z_i^{(2)})$. Again, we want each score to be linear combinations of our original p variables. This gives us

$$z^{(1)} = a_1^{(1)} x^{(1)} + ... + a_p^{(1)} x^{(p)}$$
$$z^{(2)} = b_1^{(2)} x^{(1)} + ... + b_p^{(2)} x^{(p)}$$

Notice that the coefficients $a_1^{(1)}, \ldots, a_p^{(1)}$ belong to our first PC score, $z^{(1)}$, and the second set of coefficients $b_1^{(2)}, \ldots, b_p^{(2)}$ are *entirely different* numbers and belong to our second PC score, $z^{(2)}$.

We can think of this as going from each observation having data $(x^{(1)}, \ldots, x^{(p)})$ to now having $(z^{(1)}, z^{(2)})$ as their summarized data. This is often called a **reduced dimensionality representation** of our data, because we are going from p variables to a reduced number of summary variables (in this case 2 variables). More generally, if we have many variables, we can use the principal components to go from many variables to a smaller number.

How are we going to choose $(z^{(1)}, z^{(2)})$? Previously we chose the coefficients a_k so that the result is that $z^{(1)}$ has maximal variance. Now that we have two variables, what properties do we want them to have? They clearly cannot *both* maximimize the variance, since there's only one way to do that – we'd get $z^{(1)} = z^{(2)}$ which doesn't give us any more information about our data!

So we need to say something about how the new variables $z^{(1)}, z^{(2)}$ relate to each other so that *jointly* they maximally preserve information about our original data.

How can we quantify this idea? There are ways of measuring the total variance between multiple variables $z^{(1)}$ and $z^{(2)}$ variables, which we won't go into in detail. But we've seen that when variables are highly correlated with each other, they don't give a lot more information about our observations since we can predict one from the other with high confidence (and if perfectly correlated we get back to $z^{(1)} = z^{(2)}$). So this indicates that we would want to choose our coefficients $a_1^{(1)}, \ldots, a_p^{(1)}$ and $b_1^{(2)}, \ldots, b_p^{(2)}$ so that the resulting $z^{(1)}$ and $z^{(2)}$ are completely uncorrelated.

How PCA does this is the following:

1) Choose $a_1^{(1)} \ldots a_p^{(1)}$ so that the resulting $z_i^{(1)}$ has maximal variance. This means it will be exactly the same as our PC that we found previously.

2) Choose $b_1^{(2)}, \ldots, b_p^{(2)}$ so that the resulting $z_i^{(2)}$ is uncorrelated with the $z_i^{(1)}$ we have already found. That does not create a unique score $z_i^{(2)}$ (there will be many that satisfy this property). So we have the further requirement

3) Among all $b_1^{(2)}, \ldots, b_p^{(2)}$ that result in $z_i^{(2)}$ uncorrelated with $z_i^{(1)}$, we choose $b_1^{(2)}, \ldots, b_p^{(2)}$ so that the resulting $z_i^{(2)}$ have maximal variance.

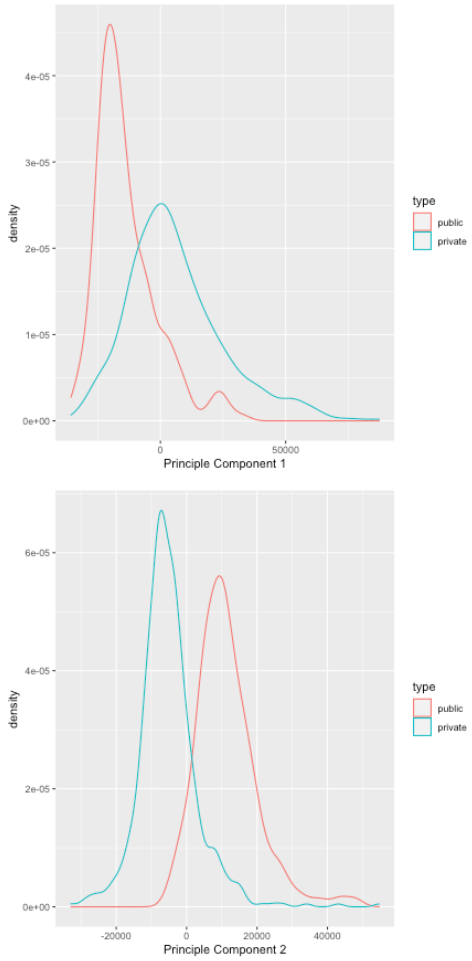
This sounds like a hard problem to find $b_1^{(2)}, \ldots, b_p^{(2)}$ that satisfies both of these properties, but it is actually equivalent to a straightforward problem in linear algebra (related to SVD or eigen decompositions of matrices).

The end result is that we wind up with two new variables for each observation and these new variables have correlation equal to zero and jointly "preserve" the maximal amount of variance in our data.

Example: College Data

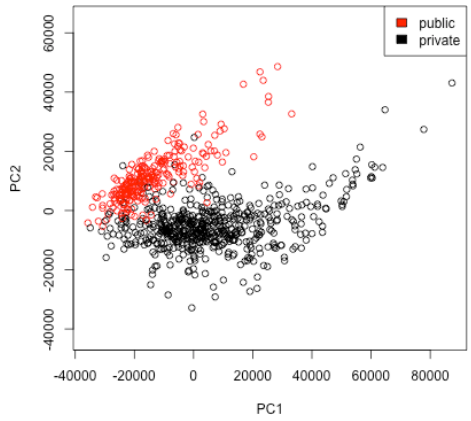
Let's consider what going to two PC components does for our previous data examples. Previously in the college data, with one principal component we saw that there was a difference in the distribution of scores, but that was also a great deal of overlap.

Individually, we can consider each PC, and see that there is a bit more separation in PC2 than PC1



But even more importantly, we can consider these variables *jointly* in a scatter plot:

```
plot(pcaCollegeDf[, c("PC1", "PC2")], col = c("red",
    "black")[pcaCollegeDf$type], asp = 1)
legend("topright", c("public", "private"), fill = c("red",
    "black"))
```

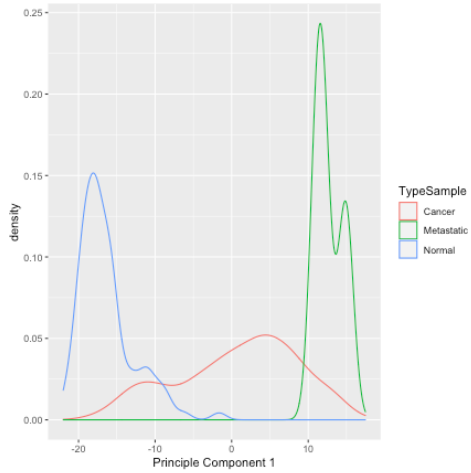


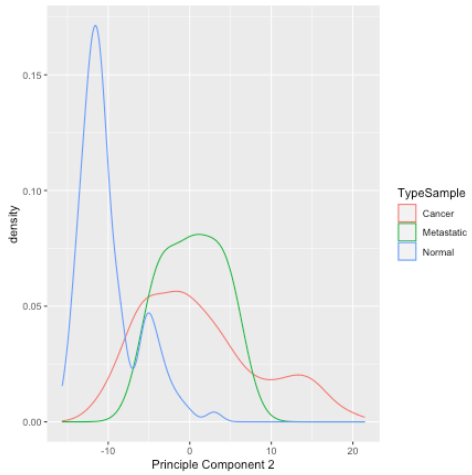
We see that now the public and private are only minimally overlapping; we've gained a lot of information about this particular distinction (public/private) by *adding* in PC2 *in addition* to PC1.

Remember, we didn't use the public or private variable in our PCA; and there is no guarantee that the first PCs will capture the differences you are interested in. But when these differences create large distinctions in the variables (like the public/private difference does), then PCA is likely to capture this difference, enabling you to use it effectively as a visualization tool.

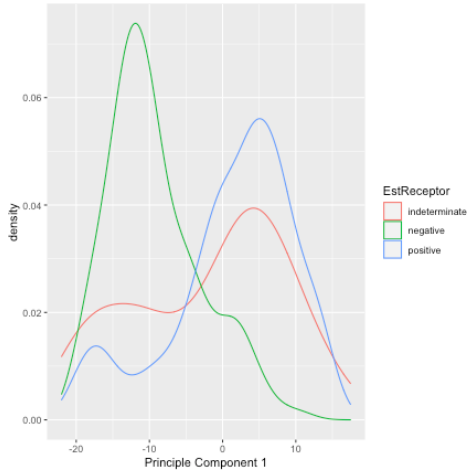
Example: Breast Cancer Data

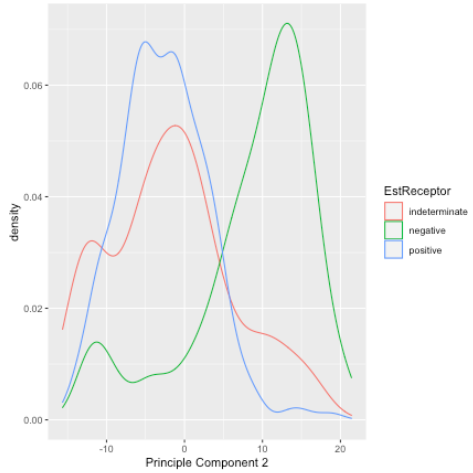
We now turn to the breast cancer data. We can see that PC2 is probably slightly worse at separating normal from cancer compared to PC1 (and particularly doesn't give similar scores to metastases):



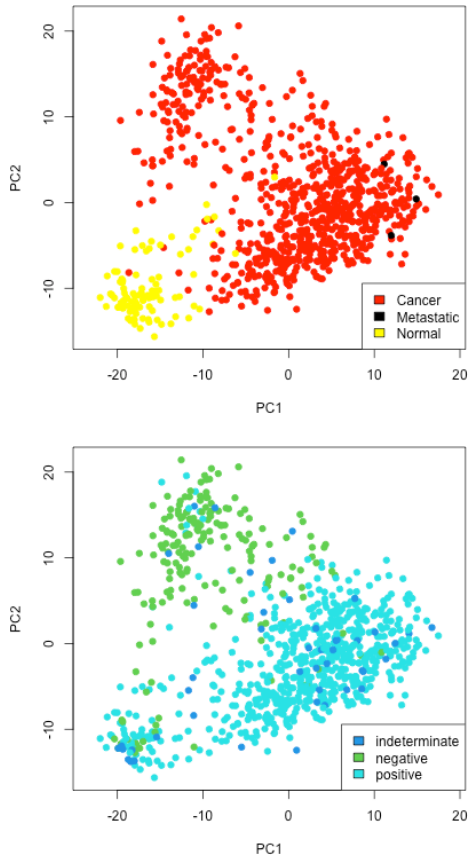


It does a arguably a better job of separating our negative estrogen receptor patients:





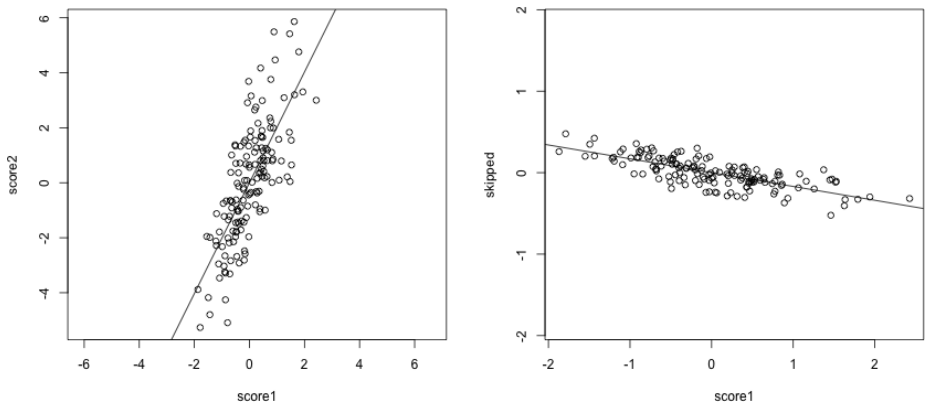
When we consider these variables *jointly* in a scatter plot we see much greater separation:



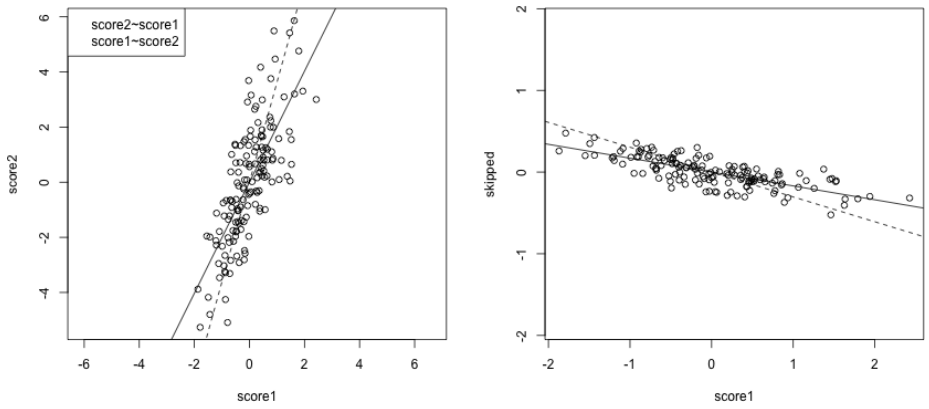
Question: What differences do you see when you use both principal components rather than either one singly?

5.4.2 Geometric Interpretation

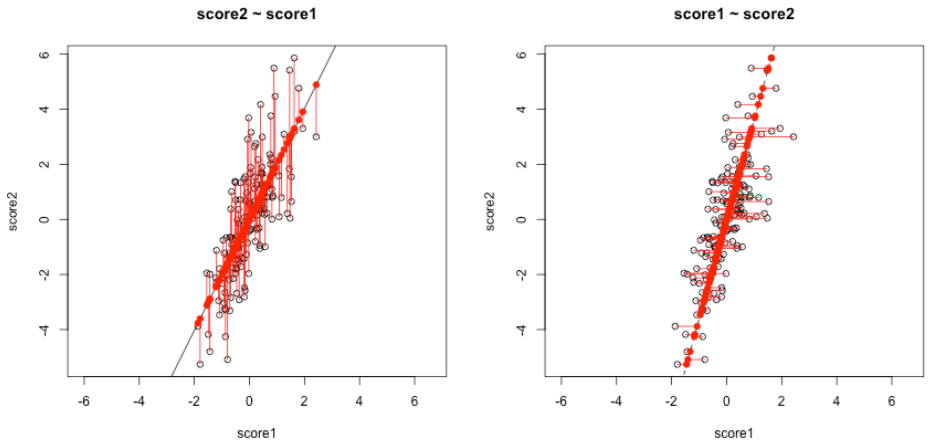
Another way to consider our redundancy is geometrically. If this was a regression problem we would "summarize" the relationship betweeen our variables by the regression line:



This is a summary of how the x-axis variable predicts the y-axis variable. But note that if we had flipped which was the response and which was the predictor, we would give a *different* line.



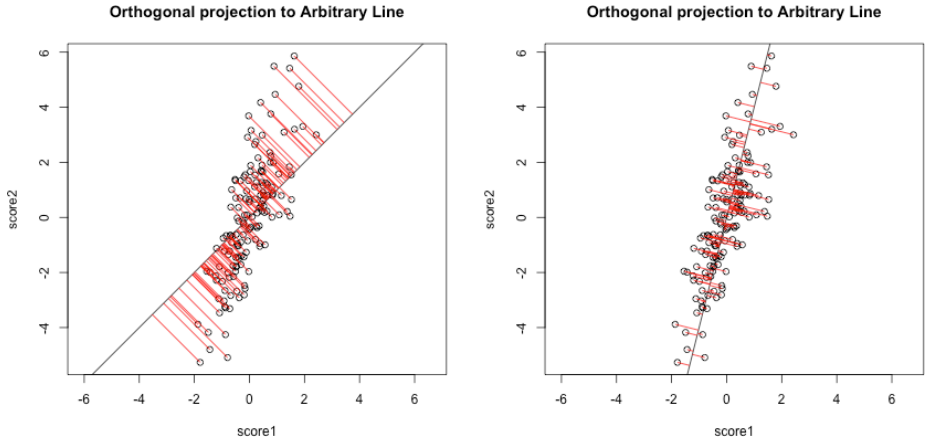
The problem here is that our definition of what is the best line summarizing this relationship is not symmetric in regression. Our best line minimizes error in the y direction. Specifically, for every observation i, we project our data onto the line so that the error in the y direction is minimized.



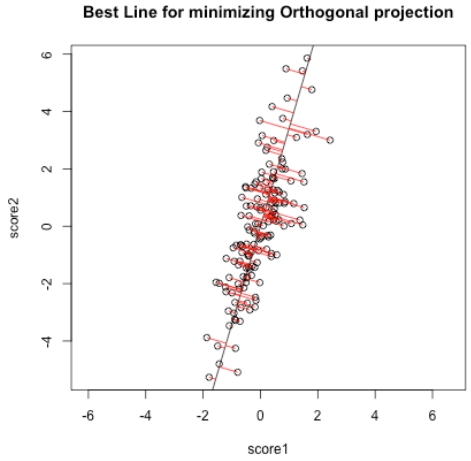
However, if we want to summarize both variables symmetrically, we could instead consider picking a line to minimize the distance from each point to the line.

By distance of a point to a line, we mean the minimimum distance of any point to the line. This is found by drawing another line that goes through the point and is orthogonal to the line. Then the length of that line segment from the point to the line is the distance of a point to the line.

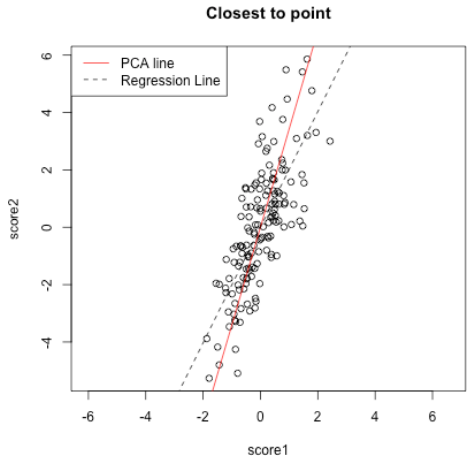
Just like for regression, we can consider all lines, and for each line, calculate the average distance of the points to the line.



So to pick a line, we now find the line that minimizes the average distance to the line across all of the points. This is the PCA line:



Compare this to our regression line:



Creating a new variable from the PCA line

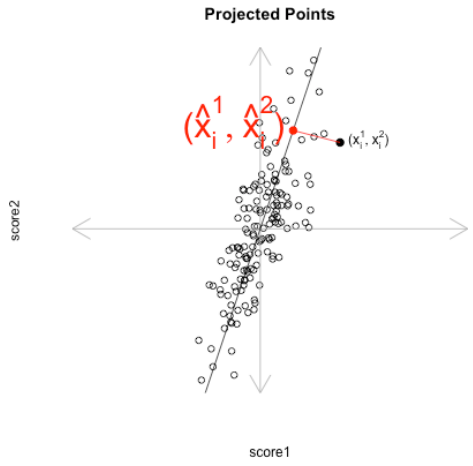
Drawing lines through our data is all very well, but what happened to creating a new variable, that is the best summary of our two variables? In regression, we could view that our regression line gave us the "best" prediction of the average y for an x (we called it our predicted value, or $\hat{y}$). This best value was where our error line drawn from y_i to the regression line (vertically) intersected.

Similarly, we used lines drawn from our data point to our PCA line to define the best line summary, only we've seen that for PCA we are interested in the line orthogonal to our point so as to be symmetric between our two variables – i.e. not just in the y direction. In a similar way, we can say that the point on the line where our perpendicular line hits the PCA line is our best summary of the value of our point. This is called the **orthogonal projection** of our point onto the line. We could call this new point $(\hat{x}^{(1)}, \hat{x}^{(2)}$.

This doesn't actually give us a single variable in place of our original two variables, since this point is defined by 2 coordinates as well. Specifically, for any line $x^{(2)} = a + bx^{(1)}$, we have that the coordinates of the projection onto the line are given by[4]

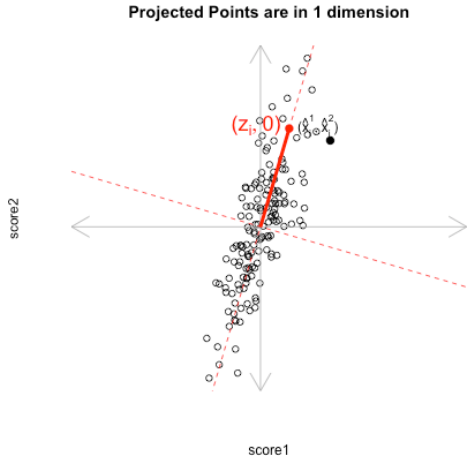
$$\hat{x}^{(1)} = \frac{b}{b^2+1}(\frac{x^{(1)}}{b} + x^{(2)} - a)$$
$$\hat{x}^{(2)} = \frac{1}{b^2+1}(bx^{(1)} + b^2x^{(2)} + a)$$

(and since we've centered our data, we want our line to go through $(0,0)$, so $a = 0$)



But geometrically, if we consider the points $(\hat{x}_i^{(1)}, \hat{x}_i^{(2)})$ as a summary of our data, then we don't actually need two dimensions to describe these summaries. From a geometric point of view, our coordinate system is arbitrary for describing the relationship of our points. We could instead make a coordinate system where one of the coordiantes was the line we found, and the other coordinate the orthogonal projection of that. We'd see that we would only need 1 coordinate (z_i) to describe $(\hat{x}_i^{(1)}, \hat{x}_i^{(2)})$ – the other coordinate would be 0.

[4]See, for example, (https://en.wikipedia.org/wiki/Distance_from_a_point_to_a_line), on Wikipedia, where they give a proof of these statements



That coordiante, z_i, would equivalently, from a geometric perspective, describe our projected points. And the value z_i is found as the distance of the projected point along the line (from $(0,0)$).[5] So we can consider z_i as our new variable.

Relationship to linear combinations

Is z_i a linear combination of our original $x^{(1)}$ and $x^{(2)}$? Yes. In fact, as a general rule, if a line going through $(0,0)$ is given by $x^{(2)} = bx^{(1)}$, then the distance along the line of the projection is given by[6]

$$z_i = \frac{1}{\sqrt{1+b2}}(x^{(1)} + bx^{(2)})$$

Relationship to variance interpretation

Finding z_i from the geometric procedure described above (finding line with minimimum orthogonal distance to points, then getting z_i from the projection of the points on to the line) is actually mathematically *equivalent* to finding the linear combination $z_i = a_1x^{(1)} + a_2x^{(2)}$ that results in the greatest variance of our points. In other words, finding a_1, a_2 to minimize $\hat{var}(z_i)$ is the same as finding the slope b that minimizes the average distance of $(x_i^{(1)}, x_i^{(2)})$ to its projected point $(\hat{x}_i^{(1)}, \hat{x}_i^{(2)})$.

To think why this is true, notice that if I assume I've centered my data, as I've done above, then the total variance in my two variables (i.e. sum of the

[5] From $(0,0)$, because I centered the data, so the center of the points is at $(0,0)$.

[6] You can see this by using the coordinates of $\hat{x} = (\hat{x}^{(1)}, \hat{x}^{(2)})$ given above, and using the pythagorean theorem, since the points $(0,0)$, $\hat{x} = (\hat{x}^{(1)}, \hat{x}^{(2)})$, and $(x^{(1)}, x^{(2)})$ form a right angled triangle. Note that it is important that our line has $a = 0$ for this calculation.

variances of each variable) is given by

$$\frac{1}{n-1}\sum_i (x_i^{(1)})^2 + \frac{1}{n-1}\sum_i (x_i^{(2)})^2$$
$$\frac{1}{n-1}\sum_i \left[(x_i^{(1)})^2 + (x_i^{(2)})^2\right]$$

So that variance is a geometrical idea once you've centered the variables – the sum of the squared length of the vector $((x_i^{(1)}, x_i^{(2)})$. Under the geometric interpretation your new point $(\hat{x}_i^{(1)}, \hat{x}_i^{(2)})$, or equivalently z_i, has mean zero too, so the total variance of the new points is given by

$$\frac{1}{n-1}\sum_i z_i^2$$

Since we know that we have an orthogonal projection then we know that the distance d_i from the point $(x_i^{(1)}, x_i^{(2)})$ to $(\hat{x}_i^{(1)}, \hat{x}_i^{(2)})$ satisfies the Pythagorean theorem,

$$z_i(b)^2 + d_i(b)^2 = [x_i^{(1)}]^2 + [x_i^{(2)}]^2.$$

That means that finding b that minimizes $\sum_i d_i(b)^2$ will also maximize $\sum_i z_i(b)^2$ because

$$\sum_i d_i(b)^2 = \text{constant} - \sum_i z_i(b)^2$$

so minimizing the left hand size will maximize the right hand side.

Therefore since every $z_i(b)$ found by projecting the data to a line through the origin is a linear combination of $x_i^{(1)}, x_i^{(2)}$ AND minimizing the squared distance results in the $z_i(b)$ having maximum variance across all such $z_i^2(b)$, then it MUST be the same z_i we get under the variance-maximizing procedure.

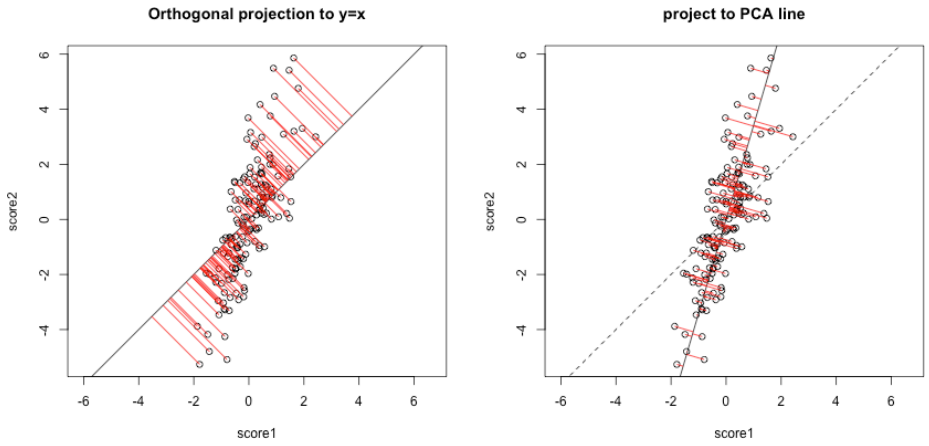
The above explanation is to help give understanding of the mathematical underpinnings of why they are equivalent. But the important take-home fact is that both of these procedures are the same: if we minimize the distance to the line, we *also* find the linear combination so that the projected points have the most variance (i.e. we can spread out the points the most).

Compare to Mean

We can use the geometric interpretation to consider what is the line corresponding to the linear combination defined by the mean,

$$\frac{1}{2}x^{(1)} + \frac{1}{2}x^{(2)}$$

It is the line $y = x$,



We could see geometrically how the mean is not a good summary of our cloud of data points.

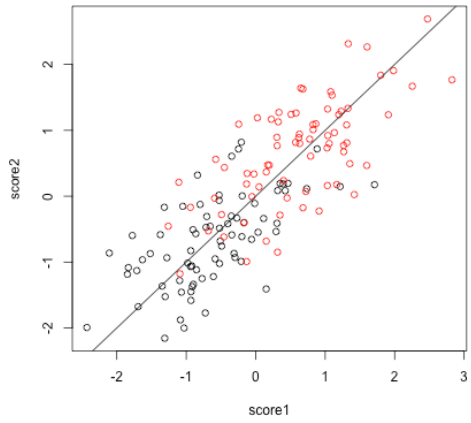
Note on Standardizing the Variables

You might say, "Why not standardize your scores by the standard deviation so they are on the same scale?" For the case of combining 2 scores, if I normalized my variables, I would get essentially the same z from the PCA linear combination and the mean.[7] However, as we will see, we can extend PCA summarization to an arbitrary number of variables, and then the scaling of the variables does not have this equivalency with the mean. This is just a freak thing about combining 2 variables.

Why *maximize* variance – isn't that wrong?

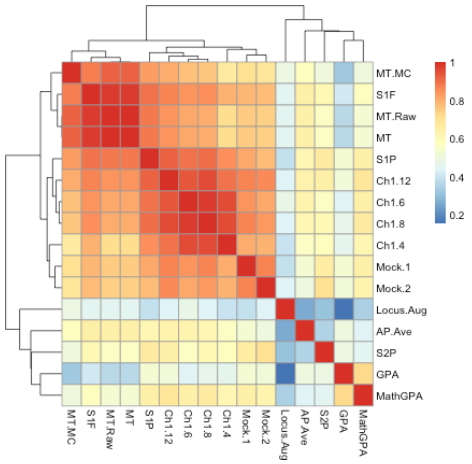
This geometric interpretation allows us to go back to this question we addressed before – why maximize variance? Consider this simple simulated example where there are two groups that distinguish our observations. Then the difference in the groups is creating a large spread in our observations. Capturing the variance is capturing these differences.

[7] If the data is scaled so the two variances have the same st.deviation, then they are exactly the same up to a constant; PCA uses $\frac{1}{\sqrt{2}}$ rather than $\frac{1}{2}$ for the constant. But they both give equal weight to the two variables.



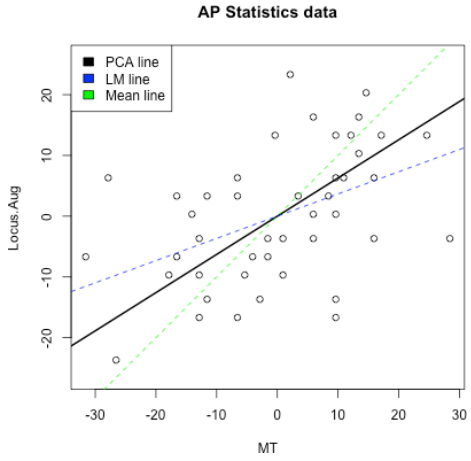
Example on real data

We will look at data on scores of students taking AP statistics. First we will draw a heatmap of the pair-wise correlation of the variables.



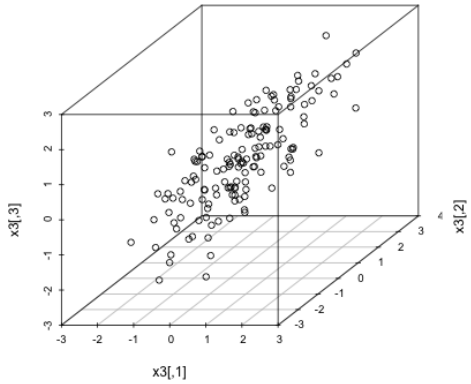
Not surprisingly, many of these measures are highly correlated.

Let's look at 2 scores, the midterm score (MT) and the pre-class evaluation (Locus.Aug) and consider how to summarize them using PCA.



5.4.2.1 More than 2 variables

We could similarly combine three measurements. Here is some simulated test scores in 3 dimensions.



Now a good summary of our data would be a line that goes through the cloud of points. Just as in 2 dimensions, this line corresponds to a linear combination of the three variables. A line in 3 dimensions is written in it's standard form as:

$$c = b_1 x_i^{(1)} + b_2 x_i^{(2)} + b_3 x_i^{(3)}$$

Since again, we will center our data first, the line will be with $c = 0$.[8]

The exact same principles hold. Namely, that we look for the line with the smallest average distance to the line from the points. Once we find that line

[8]This is the standard way to write the equation for a line in higher dimensions and is symmetric in the treatment of the variables. Note the standard way you were probably taught to write a line in 2-dimensions, $y = a + bx$ can also be written in this form with $c = b$, $b_1 = b$, and $b_2 = -1$.

(drawn in the picture above), our z_i is again the distance from 0 of our point projected onto the line. The only difference is that now distance is in 3 dimensions, rather than 2. This is given by the Euclidean distance, that we discussed earlier.

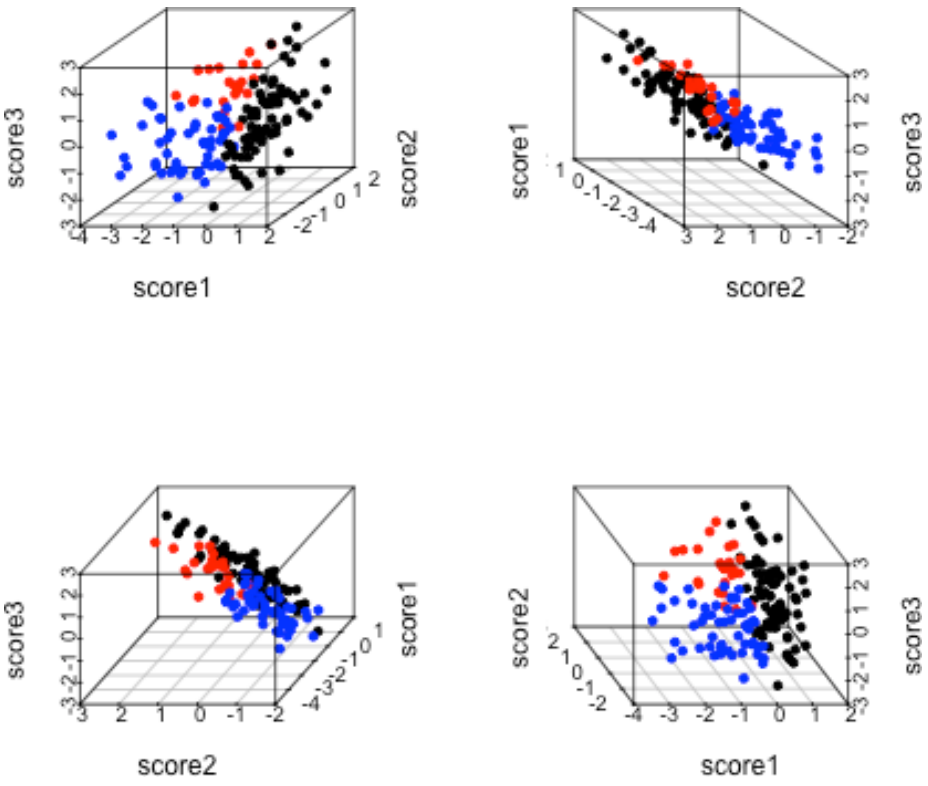
Just like before, this is exactly equivalent to setting $z_i = a_1 x_i^{(1)} + a_2 x_i^{(2)} + a_3 x_i^{(3)}$ and searching for the a_i that maximize $\hat{var}(z_i)$.

Many variables

We can of course expand this to as many variables as we want, but it gets hard to visualize the geometric version of it. The variance-maximizing version is easier to write out.

5.4.2.2 Adding another principal component

What if instead my three scores look like this (i.e. line closer to a plane than a line)?



I can get one line through the cloud of points, corresponding to my best linear combination of the three variables. But I might worry whether this really rep-

resented my data, since as we rotate the plot around we can see that my points appear to be closer to a lying near a plane than a single line.

Question: For example, can you find a single line so that if you projected your data onto that line, you could separate the three groups shown?

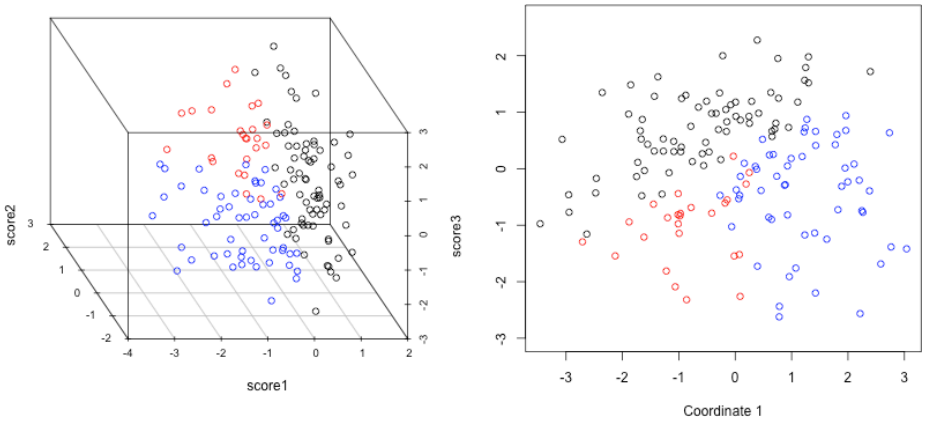
So there's some redundancy, in the sense that I don't need three dimensions to geometrically represent this data, but it's not clear that with only 1 new variable (i.e. line) we can summarize this cloud of data geometrically.

5.4.2.3 The geometric idea

I might ask whether I could better summarize these three variables by two variables, i.e. as a plane. I can use the same geometric argument – find the best plane, so that the orthogonal projection of the points to the plane is the smallest. This is equivalent to finding two lines, rather than one, since a plane can be defined by any two lines that lie on it.

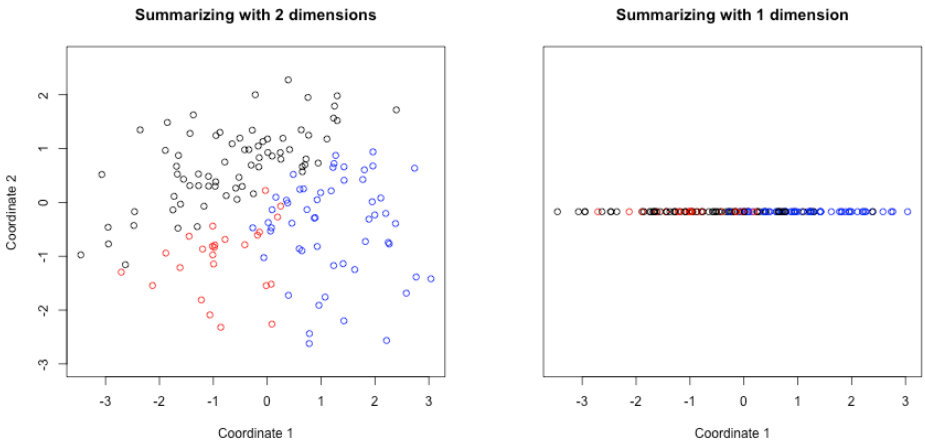
I could just search for the plane that is closest to the points, just like previously I searched for a line that is closest to the points – i.e. any two lines on the plane will do, so long as I get the right plane. But that just gives me the plane. It doesn't give me new data points. To do that, I need coordiantes of each point projected onto the plane, like previously we projected onto the line.

I need to set up an orthogonal coordinate axis so I can define $(z_i^{(1)}, z_i^{(2)})$ for each point.



Thus the new points $(z_i^{(1)}, z_i^{(2)})$ represent the points after being projected on to that plane in 3d. So we can summarize the 3 dimensional cloud of points by this two dimensional cloud. This is now a *summary* of the 3D data. Which is

nice, since it's hard to plot in 3D. Notice, I can still see the differences between my groups, so I have preserved that important variability (unlike using just a single line):



5.4.2.4 Finding the Best Plane

I want to be smarter than just finding any coordinate system for my "best" plane – there is an infinite number of equivalent choices. So I would like the new coordinates $(z_i^{(1)}, z_i^{(2)})$ to be useful in the following way: I want my first coordinate $z_i^{(1)}$ to correspond to the coordinates I would get if I just did just 1 principal component, and then pick the next coordinates to be the orthogonal direction from the 1st principal component that also lies on the plane.[9]

This reduces the problem of finding the plane to 1) finding the 1st principal component, as described above, then 2) finding the "next best" direction.

So we need to consider how we find the next best direction.

Consider 2-dimensions

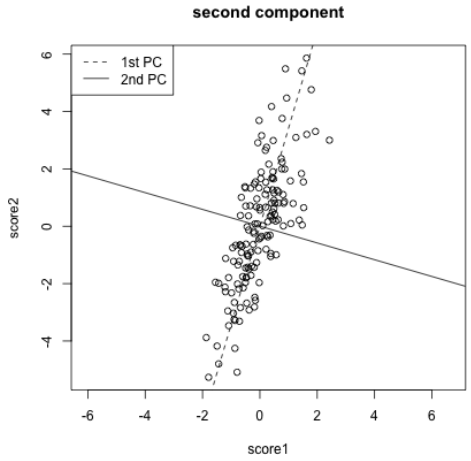
Let's return to our 2-dim example to consider how we can "add" another dimension to our summary. If I have my best line, and then draw another line very similar to it, but slightly different slope, then it will have very low average distance of the points to the line. And indeed, we wouldn't be able to find "next best" in this way, because the closest to the best line would be choosen – closer and closer until in fact it is the same as the best line.

Moreover, such a line that is close to the best doesn't give me very different information from my best line. So I need to force "next best" to be separated and distinct from my best line. How do we do that? We make the requirement

[9]The first principal component direction will by definition fall on the "best" plane.

that the next best line be orthogonal from the best line – this matches our idea above that we want an orthogonal set of lines so that we set up a new coordinate axes.

In two dimensions that's a pretty strict constraint – there's only 1 such line! (at least that goes through the center of the points).



Return to 3 dimensions

In three dimensions, however, there are a whole space of lines to pick from that are orthogonal to the 1st PC and go through the center of the points.

Not all of these lines will be as close to the data as others lines. So there is actually a choice to be made here. We can use the same criterion as before. Of all of these lines, which minimize the distance of the points to the line? Or (equivalently) which result in a linear combination with maximum variance?

To recap: we find the first principal component based on minimizing the points' distance to line. To find the second principal component, we similarly find the line that minimize the points' distance to the line *but* only consider lines orthogonal to the the first component.

If we follow this procedure, we will get two orthogonal lines that define a plane, and this plane is the closest to the points as well (in terms of the orthogonal distance of the points to the *plane*). In otherwords, we found the two lines without thinking about finding the "best" plane, but in the end the plane they create will be the closest.

5.4.2.5 Projecting onto Two Principal Components

Just like before, we want to be able to not just describe the best plane, but to summarize the data. Namely, we want to project our data onto the plane. We

do this again, by projecting each point to the point on the plane that has the shortest distance, namely it's orthogonal projection.

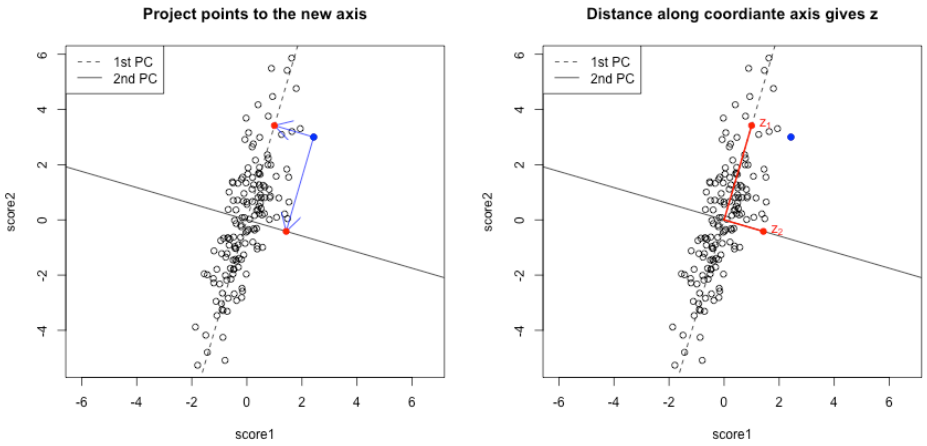
We could describe this project point in our original coordinate space (i.e. with respect to the 3 original variables), but in fact these projected points lie on a plane and so we only need two dimensions to describe these projected points. So we want to create a new coordinate system for this plane based on the two (orthogonal) principal component directions we found.

Finding the coordiantes in 2Dim

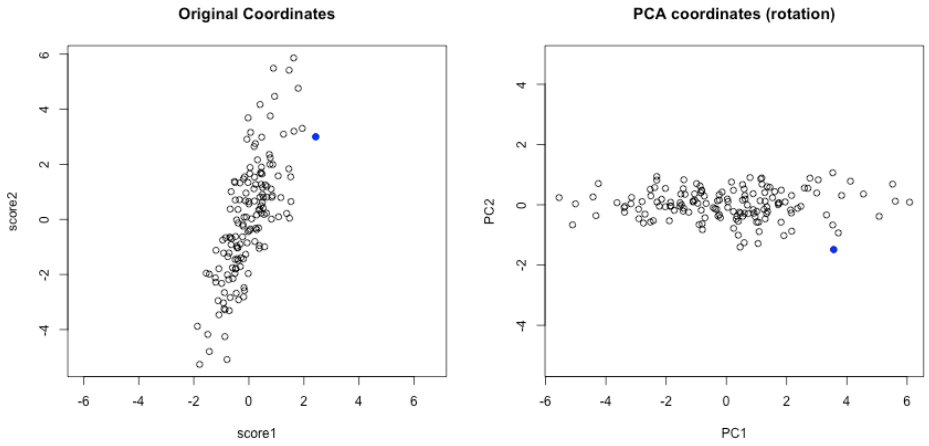
Let's consider the simple 2-d case again. Since we are in only 2D, our two principal component directions are equivalent to defining a new orthogonal coordinate system.

Then the new coordinates of our points we will call $(z_i^{(1)}, z_i^{(2)})$. To figure out their values coordiantes of the points on this new coordinate system, we do what we did before:

1. Project the points onto the first direction. The distance of the point along the first direction is $z_i^{(1)}$
2. Project the points onto the second direction. The distance of the point along the second direction is $z_i^{(2)}$



You can now consider them as new coordinates of the points. It is common to plot them as a scatter plot themselves, where now the PC1 and PC2 are the variables.



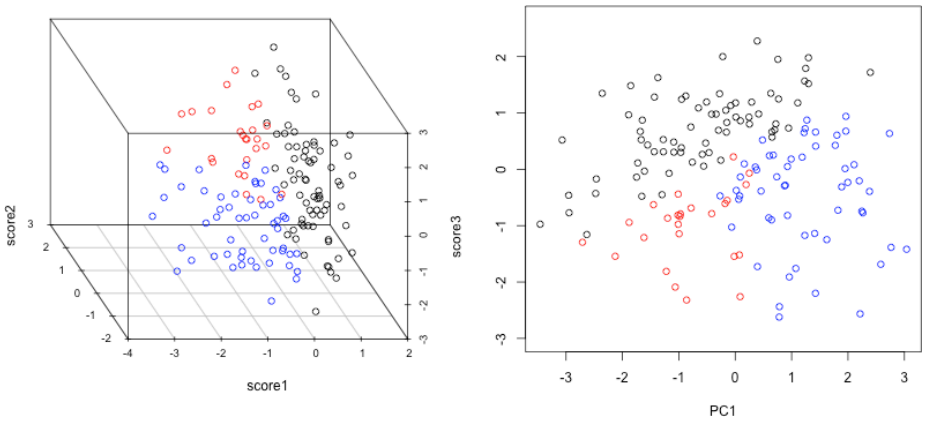
Preserving distances in 2D

In two dimensions, we completely recapture the pattern of the data with 2 principal components – we've just rotated the picture, but the relationship of the points to each other (i.e. their distances to each other), are exactly the same. So plotting the 2 PC variables instead of the 2 original variables doesn't tell us anything new about our data, but we can see that the relationship of our variables to each other is quite different.

Of course this distance preserving wasn't true if I projected only onto one principal component; the distances in the 1st PC variable are not the same as the distances in the whole dimension space.

3-dimensions and beyond

For our points in 3 dimensions, we will do the same thing: project the data points to each of our two PC directions separately, and make $z_i^{(1)}$ and $z_i^{(2)}$ the distance of the projection along each PC line. These values will define a set of coordinates for our points *after being projected to the best plane.*



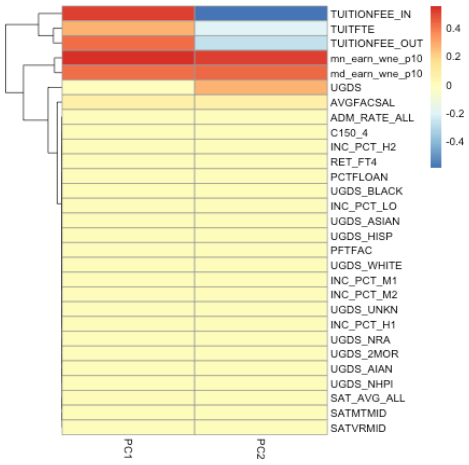
But unlike our 2D example, the projection of these points to the plane don't preserve the entire dataset, so the plot of the data based on these two coordinates is not equivalent to their position in the 3-dimensional space. We are not representing the noise around the plane (just like in 2D, where the projection of points to the line misses any noise of the points around the line). In general, if we have less principal components than the number of original variables, we will not have a perfect recapitulation of the data.

But that's okay, because what such a plot does is summarize the 3 dimensional cloud of points by this two dimensional cloud, which captures most of the variablity of the data. Which is nice, since it's hard to plot in 3D.

5.4.3 Interpreting PCA

5.4.3.1 Loadings

The scatterplots doen't tell us how the original variables relate to our new variables, i.e. the coefficients a_j which tell us how much of each of the original variables we used. These a_j are sometimes called the **loadings**. We can go back to what their coefficients are in our linear combination



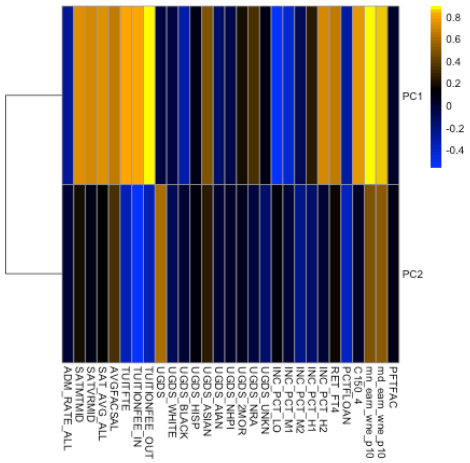
We can see that the first PC is a combination of variables related to the cost of the university (TUITFTE, TUITIONFEE_IN, TUITIONFEE_OUT are related to the tuition fees, and mn_earn_wne_p10/md_earn_wne_p10 are related to the total of amount financial aid students earn by working in aggregate across the whole school, so presumably related to cost of university); so it makes sense that in aggregate the public universities had lower PC1 scores than private in our 2-D scatter plot. Note all the coefficients in PC1 are positive, so we can think of this as roughly mean of these variables.

PC2, however, has negative values for the tuition related variables, and positive values for the financial aid earnings variables; and UGDS is the number of Undergraduate Students, which has also positive coefficients. So university with high tuition relative to the aggregate amount of financial aid they give and student size, for example, will have low PC2 values. This makes sense: PC2 is the variable that pretty cleanly divided private and public schools, with private schools having low PC2 values.

5.4.3.2 Correlations

It's often interesting to look at the *correlation* between the new variables and the old variables. Below, I plot the heatmap of the correlation matrix consisting of all the pair-wise correlations of the original variables with the new PCs

```
corPCACollege <- cor(pcaCollege$x, scale(scorecard[-whNACollege,
    -c(1:3, 12)], center = TRUE, scale = FALSE))
pheatmap(corPCACollege[1:2, ], cluster_cols = FALSE,
    col = seqPal2)
```

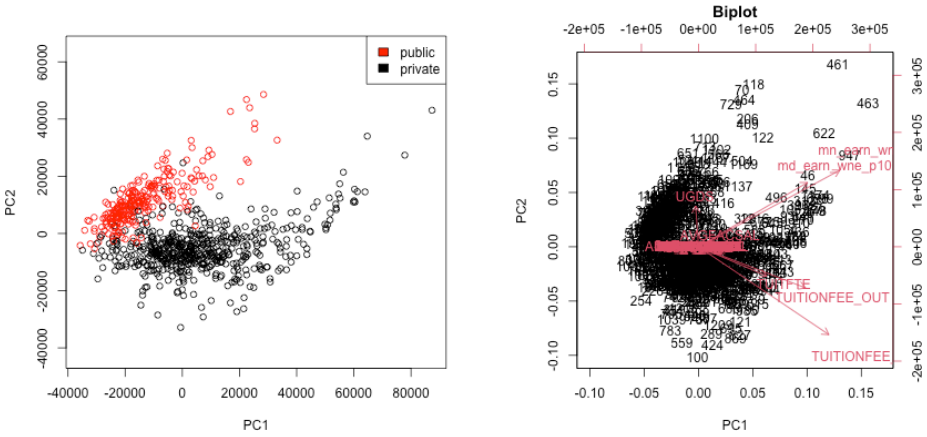


Notice this is not the same thing as which variables contributed to PC1/PC2. For example, suppose a variable was highly correlated with tuition, but wasn't used in PC1. It would still be likely to be highly correlated with PC1. This is the case, for example, for variables like SAT scores.

5.4.3.3 Biplot

We can put information regarding the variables together in what is called a biplot. We plot the observations as points based on their value of the 2 principal components. Then we plot the original variables as vectors (i.e. arrows).

```
par(mfrow = c(1, 2))
plot(pcaCollege$x[, 1:2], col = c("red", "black")[scorecard$CONTROL[-whNACollege]],
    asp = 1)
legend("topright", c("public", "private"), fill = c("red",
    "black"))
suppressWarnings(biplot(pcaCollege, pch = 19, main = "Biplot"))
```



Notice that the axes values are not the same as the basic scatterplot on the left. This is because biplot is scaling the PC variables.

Interpretation of the Biplot

The arrow for a variable points in the direction that is most like that variable.[10] So points that are in the direction of that vector tend to have large values of that variable, while points in the opposite direction of that vector have large negative values of that variable. Vectors that point in the same direction correspond to variables where the observations show similar patterns.

The length of the vector corresponds to how well that vector in this 2-dim plot actually represents the variable.[11] So long vectors tell you that the above interpretation I gave regarding the direction of the vector is a good one, while short vectors indicate that the above interpretation is not very accurate.

If we see vectors that point in the direction of one of the axes, this means that the variable is highly correlated with the principal component in that axes. I.e. the resulting new variable z that we get from the linear combination for that principal component is highly correlated with that original variable.

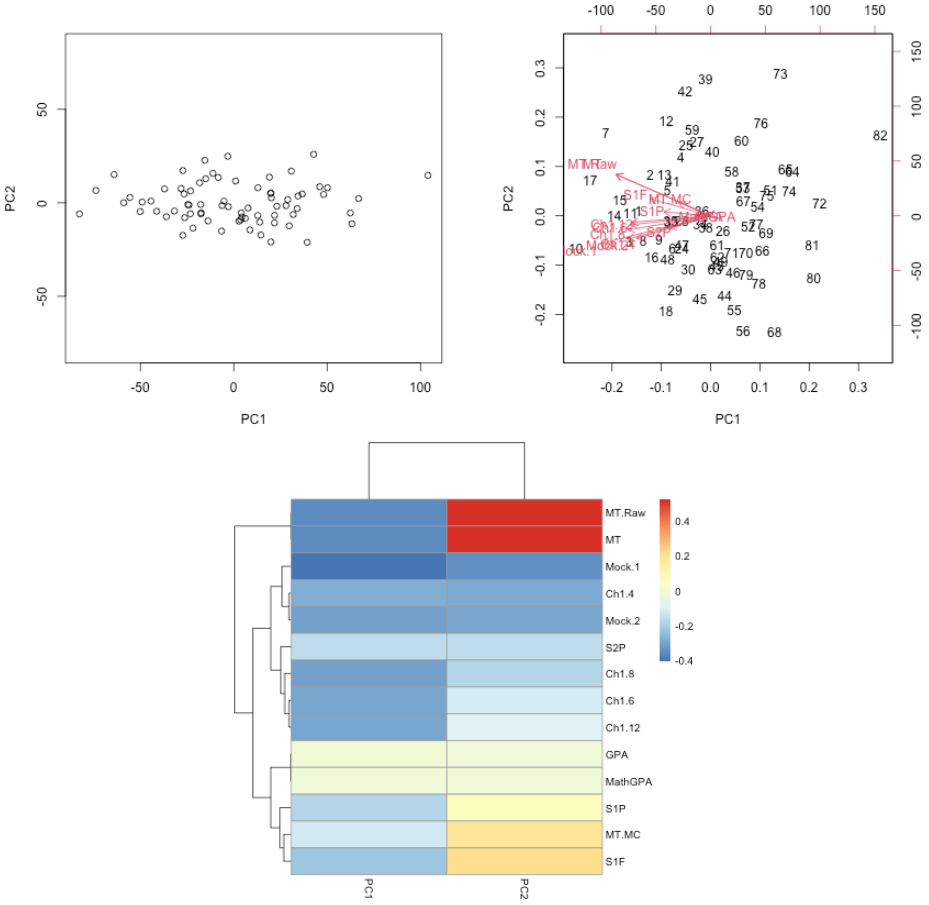
So the variables around tuition fee, we see that it points in the direction of large PC1 scores meaning observations with large PC1 scores will have higher values on those variables (and they tend to be private schools). We can see that the number of undergraduates (UGDS) and the aggregate amount of financial aid go in positive directions on PC2, and tuition are on negative directions on PC2. So we can see that some of the same conclusions we got in looking at the loadings show up here.

Example: AP Scores

[10] Specifically, if you projected the points in the biplot onto the line designated for that line, the values of the points on that line would be most correlated with the original variable.

[11] Specifically the size of the correlation of the points projected onto that vector and the actual variable.

We can perform PCA on the full set of AP scores variables and make the same plots for the AP scores. There are many NA values if I look at all the variables, so I am going to remove `Locus.Aug' (the score on the diagnostic taken at beginning of year) and`AP.Ave' (average on other AP tests) which are two variables that have many NAs, as well as removing categorical variables.

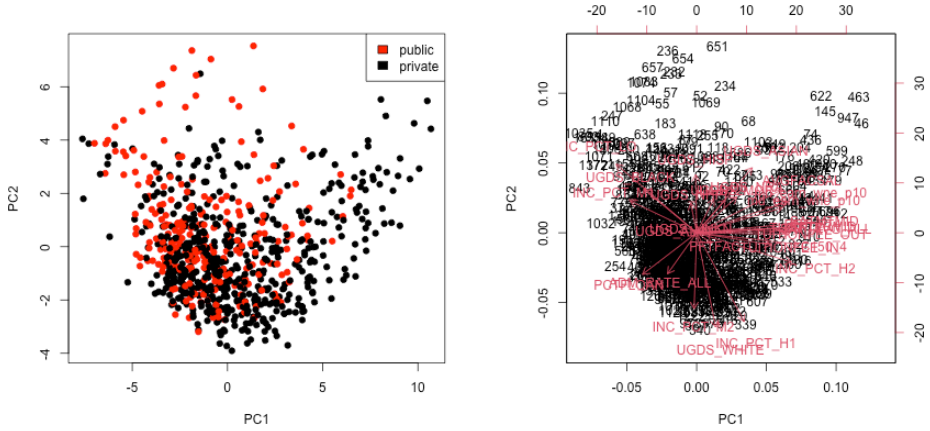


Not surprisingly, this PCA used all the variables in this first 2 PCs and there's no clear dominating set of variables in either the biplot or the heatmap of the loadings for the first two components. This matches the nature of the data, where all of the scores are measuring similar qualities of a student, and many are on similar scales.

5.4.3.4 Scaling

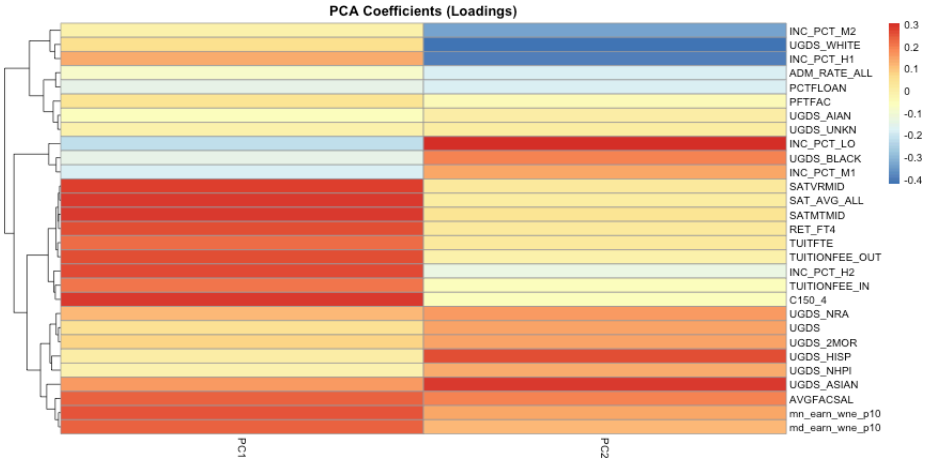
Even after centering our data, our variables are on different scales. If we want to look at the importance of variables and how to combine variables that are redundant, it is more helpful to scale each variable by its standard deviation.

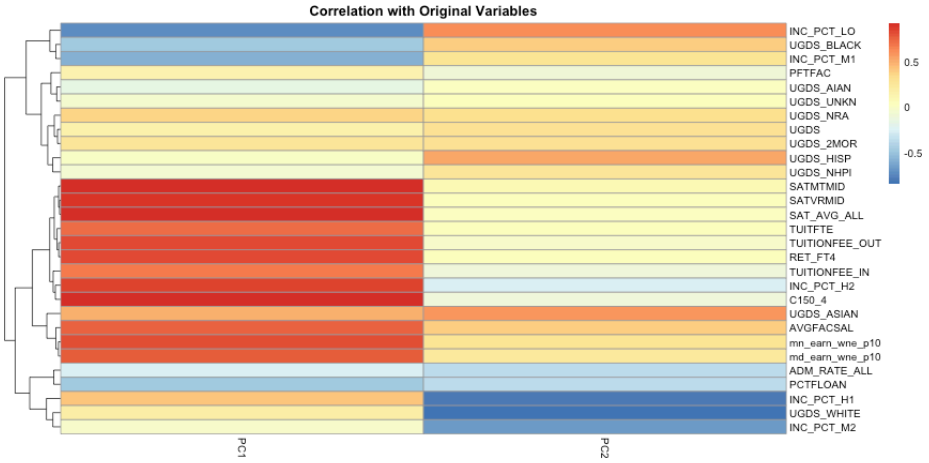
Otherwise, the coefficients a_k represent a lot of differences in scale of the variables, and not the redundancy in the variables. Doing so can change the PCA coordinates a lot.



There is still a slight preference for public schools to be lower on the 1st principal component, but its quite slight.

We see that many more variables contribute to the first 2 PCs after scaling them.

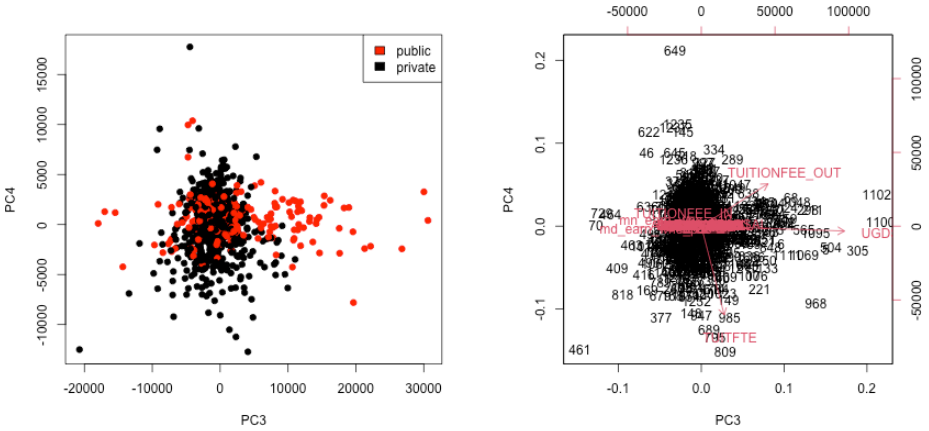




5.4.4 More than 2 PC coordinates

In fact, we can find more than 2 PC variables. We can continue to search for more components in the same way, i.e. the next best line, orthogonal to both of the lines that came before. The number of possible such principal components is equal to the number of variables (or the number of observations, whichever is smaller; but in all our datasets so far we have more observations than variables).

We can plot a scatter plot of the resulting third and 4th PC variables from the college data just like before.

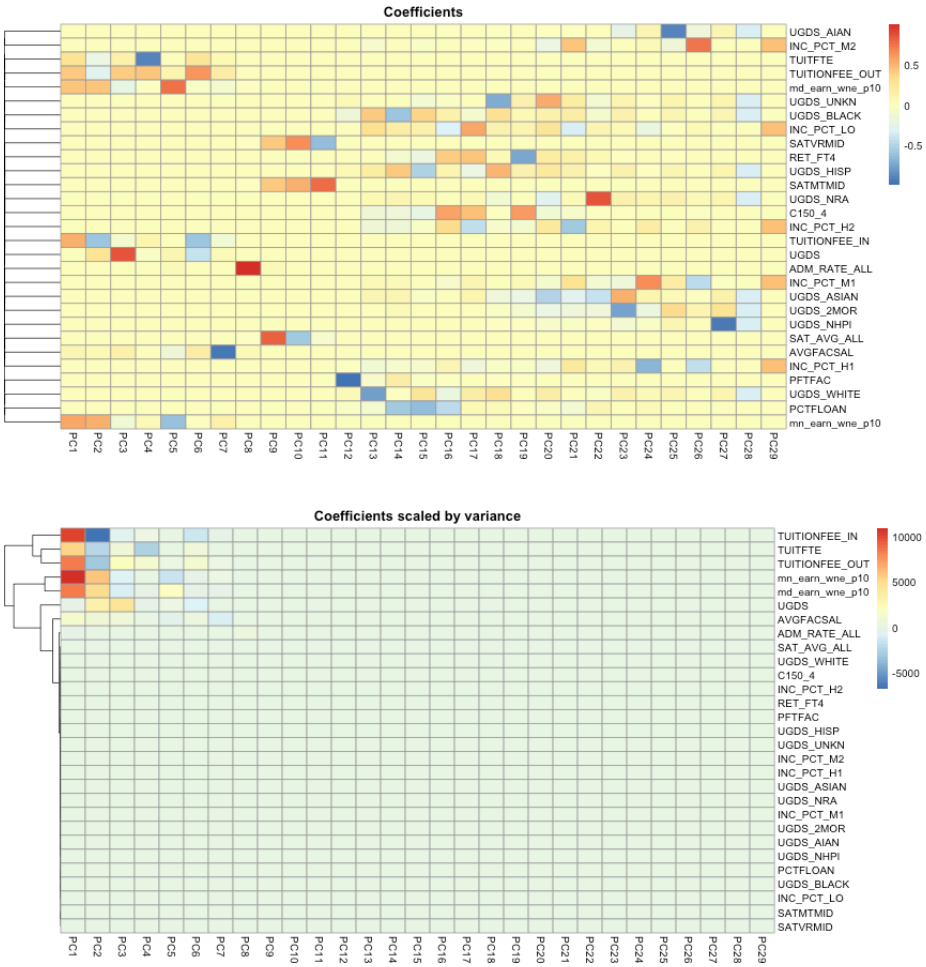


This is a very different set of coordinates for the points in 2 PCs. However, some of the same set of variables are still dominating, they are just different linear combinations of them (the two PCs lines are orthogonal to each other, but they can still just involve these variables because its such a high dimensional space).

In these higher dimensions the geometry becomes less intuitive, and it can be

helpful to go back to the interpretation of linear combinations of the original variables, because it is easy to scale that up in our minds.

We can see this by a heatmap of all the coefficients. It's also common to scale each set of PC coefficients by the standard deviation of the final variable z that the coefficients create. This makes later PCs not stand out so much. [12]



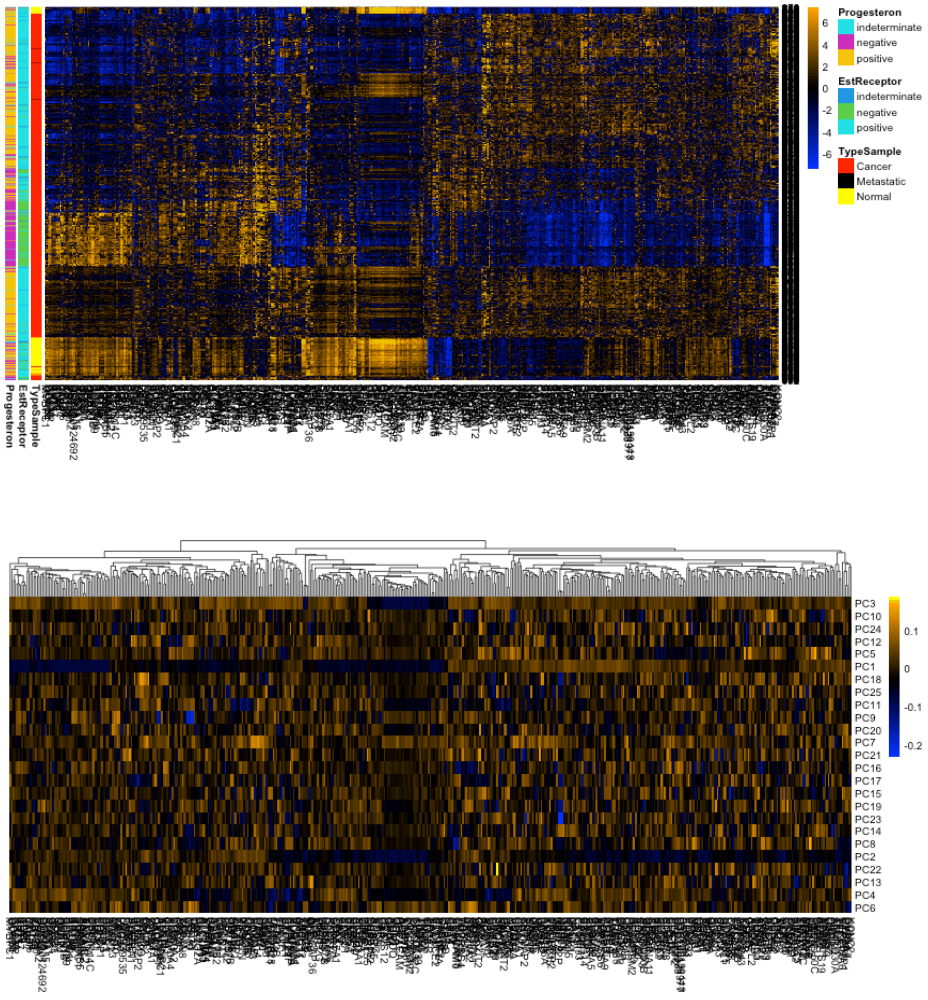
Breast data

We can also look at the higher PCs from the breast data (with the normal samples).

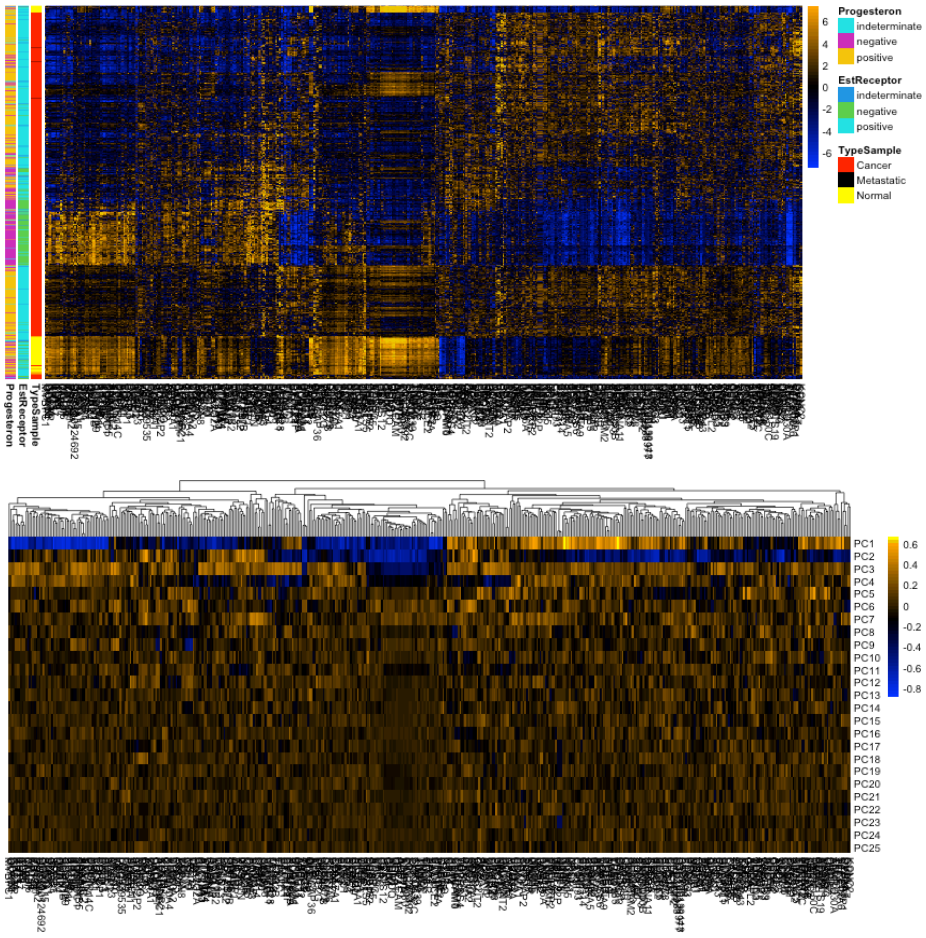
[12] We haven't discussed this, but in fact the coefficients are scaled so the sum of the square of the coefficients equal 1 (norm is one). Otherwise there's not a unique set of coefficients, since you could always just multiply all the coefficients by a number and get larger and larger variance. So the coefficients are all on the similar scale, regardless of the original variability or importance of the PC in explaining the data.

Question: If there are 500 genes and 878 observations, how many PCs are there?

We can see that there are distinct patterns in what genes/variables contribute to the final PCs (we plot only the top 25 PCs). However, it's rather hard to see, because there are large values in later PCs that mask the pattern.



This is an example of why it is useful to scale the variables by their variance

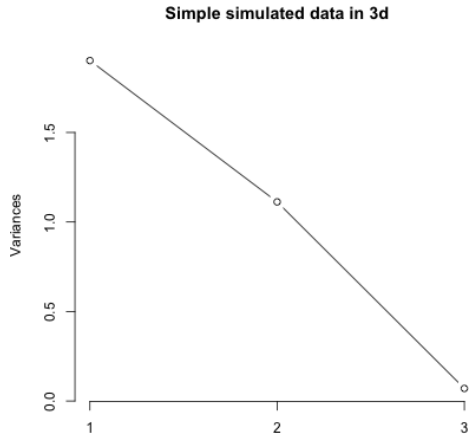


5.4.5 How many dimensions?

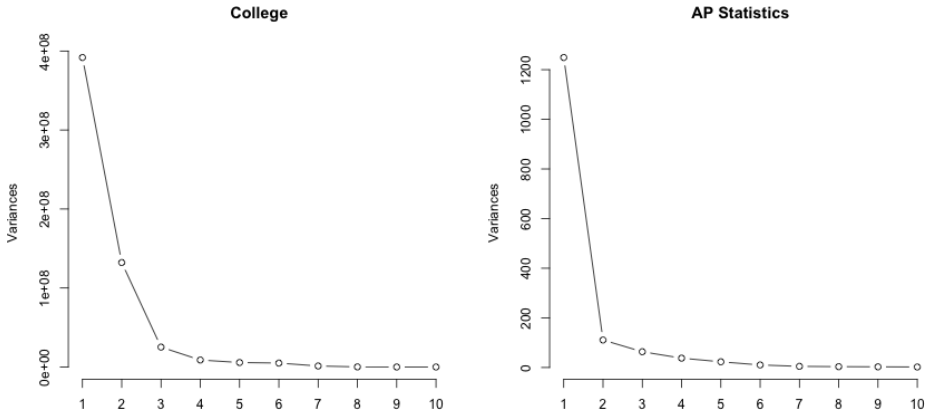
If I can draw my data in 3d, then I can guess what is the right number of coordinates – not 1 but 2 in our toy example case were needed. When I have a lot of coordinates, like the college data, how can I possibly know? One technique is to look at how much variability there is in each of the coordinates – how much variance is there in the new variable created by each linear combination. If there's not a lot of variability, then it indicates that when the points are projected onto that PC, they are huddled on top of each other, and its more likely to be noise than signal.

Consider our simple simulation example, where there was more or less a plane describing the data. If we look at the variance in each set of linear combinations we create, there is practically 0 left in the last variable, meaning that most of the representation of the points is captured in two dimensions. This is a measure

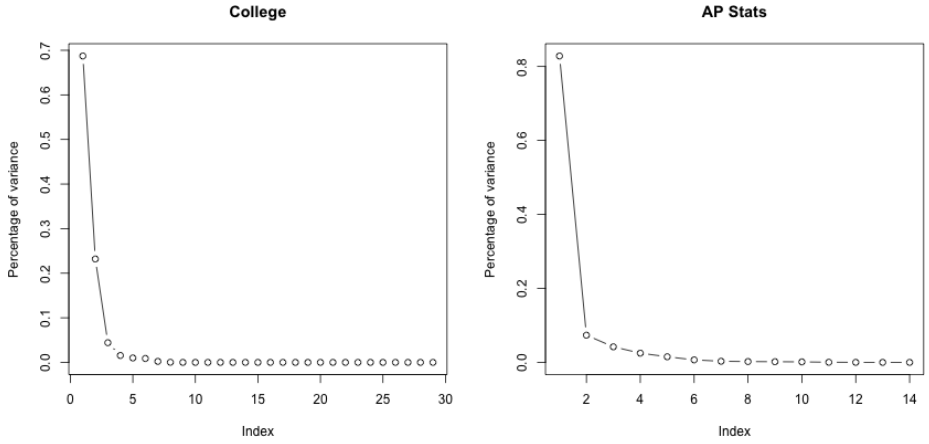
of how much we are "missing" by ignoring a coordinate.



For the college data, we similarly see that the first two dimensions both have much larger amounts compared to other dimensions. The AP Statistics data is strongly in just the first dimension.



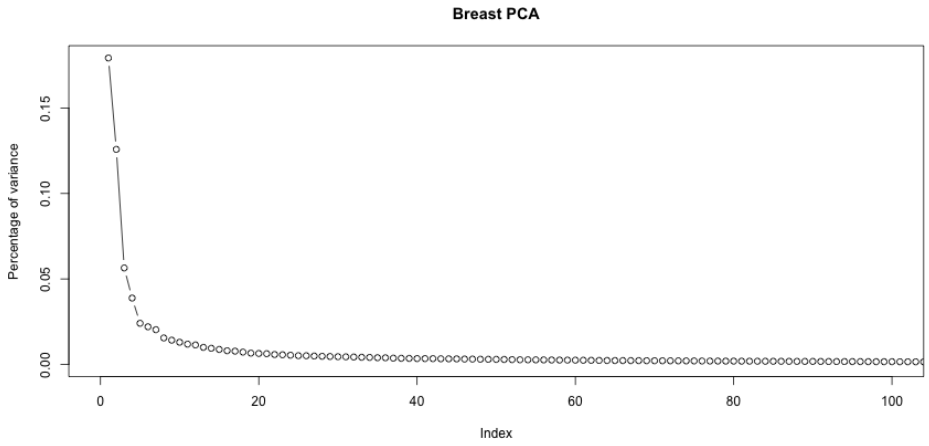
We can also plot this a percentage



2 dimensions is not always the answer

It is just a happenstance of this data that 1-2 dimensions is summarizing the data. There is nothing magical about two dimensions, other than the fact that they are easy to plot! Looking at the top two dimensions can be misleading if there is a lot of additional variability in the other dimensions (in this case, it can be helpful to look at visualizations like pairs plots on the principal components to get an idea of what you might be missing.)

We can do a similar plot for the breast cancer data.



Question: What does this tell you about the PCA?

Chapter 6

Multiple Regression

```
## Linking to ImageMagick 6.9.12.3
## Enabled features: cairo, fontconfig, freetype, heic, lcms, pango, raw, rsvg, webp
## Disabled features: fftw, ghostscript, x11
```

This chapter deals with the regression problem where the goal is to understand the relationship between a specific variable called the **response** or **dependent** variable (y) and several other related variables called **explanatory** or **independent** variables or more generally **covariates**. This is an extension of our previous discussion of simple regression, where we only had a single covariate (x).

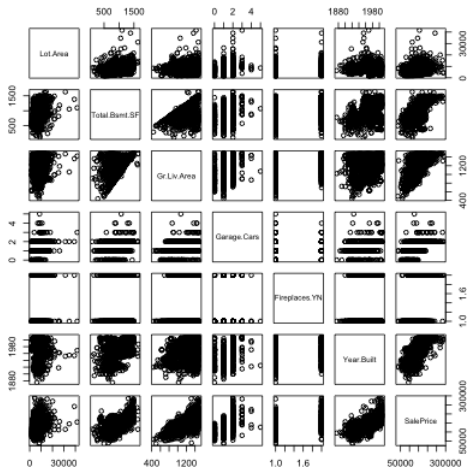
6.1 Examples

We will go through some specific examples to demonstrate the range of the types of problems we might consider in this chapter.

1. Prospective buyers and sellers might want to understand how the price of a house depends on various characteristics of the house such as the total above ground living space, total basement square footage, lot area, number of cars that can be parked in the garage, construction year and presence or absence of a fireplace. This is an instance of a regression problem where the response variable is the house price and the other characteristics of the house listed above are the explanatory variables.

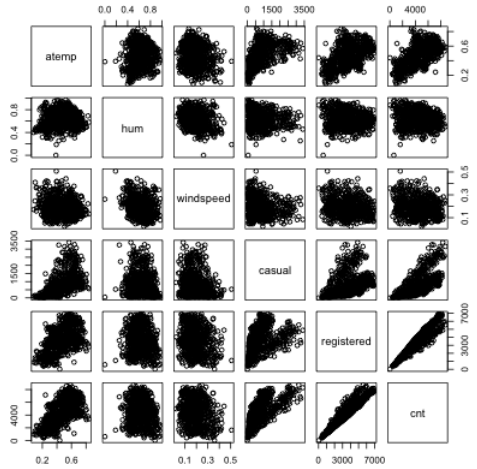
This dataset contains information on sales of houses in Ames, Iowa from 2006 to 2010. The full dataset can be obtained by following links given in the paper: (https://ww2.amstat.org/publications/jse/v19n3/decock.pdf)). I have shortened the dataset slightly to make life easier for us.

```
dataDir <- "../finalDataSets"
dd = read.csv(file.path(dataDir, "Ames_Short.csv"),
    header = T, stringsAsFactors = TRUE)
pairs(dd)
```



2. A bike rental company wants to understand how the number of bike rentals in a given hour depends on environmental and seasonal variables (such as temperature, humidity, presence of rain etc.) and various other factors such as weekend or weekday, holiday etc. This is also an instance of a regression problems where the response variable is the number of bike rentals and all other variables mentioned are explanatory variables.

```
bike <- read.csv(file.path(dataDir, "DailyBikeSharingDataset.csv"),
    stringsAsFactors = TRUE)
bike$yr <- factor(bike$yr)
bike$mnth <- factor(bike$mnth)
pairs(bike[, 11:16])
```

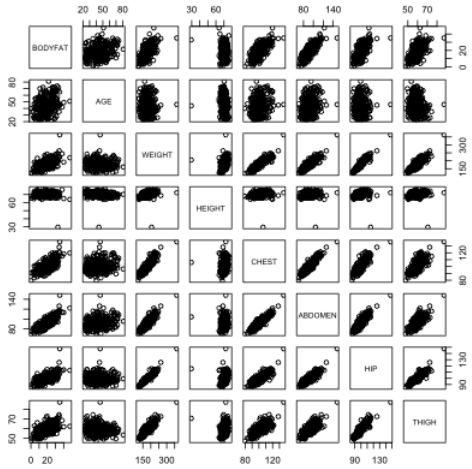


3. We might want to understand how the retention rates of colleges depend on various aspects such as tuition fees, faculty salaries, number of faculty members that are full time, number of undergraduates enrolled, number of students on federal loans etc. using our college data from before. This is again a regression problem with the response variable being the retention rate and other variables being the explanatory variables.

4. We might be interested in understanding the proportion of my body weight that is fat (body fat percentage). Directly measuring this quantity is probably hard but I can easily obtain various body measurements such as height, weight, age, chest circumeference, abdomen circumference, hip circumference and thigh circumference. Can we predict my body fat percentage based on these measurements? This is again a regression problem with the response variable being body fat percentage and all the measurements are explanatory variables.

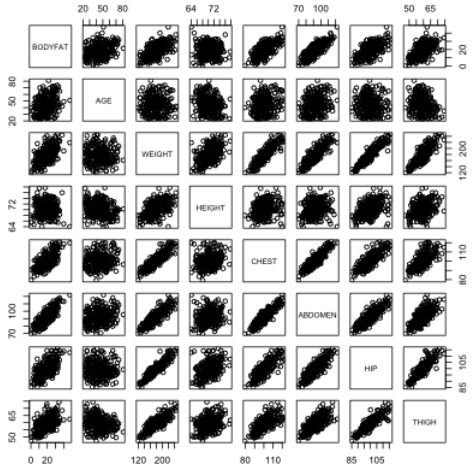
Body fat percentage (computed by a complicated underwater weighing technique) along with various body measurements are given for 252 adult men.

```
body = read.csv(file.path(dataDir, "bodyfat_short.csv"),
    header = T, stringsAsFactors = TRUE)
pairs(body)
```



There are outliers in the data and they make it hard to look at the relationships between the variables. We can try to look at the pairs plots after deleting some outlying observations.

```
ou1 = which(body$HEIGHT < 30)
ou2 = which(body$WEIGHT > 300)
ou3 = which(body$HIP > 120)
ou = c(ou1, ou2, ou3)
pairs(body[-ou, ])
```



6.2 The nature of the 'relationship'

Notice that in these examples, the *goals* of the analysis shift depending on the example from truly wanting to just be able to predict future observations

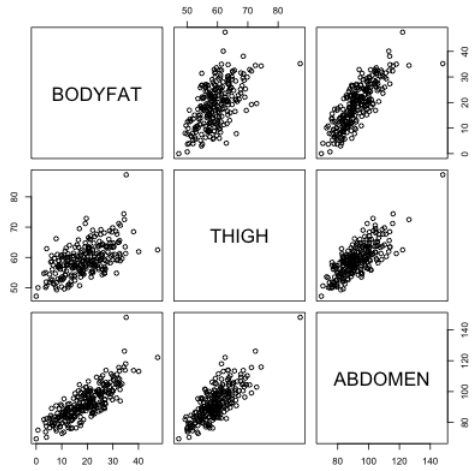
(e.g. body-fat), wanting to have insight into how to the variables are related to the response (e.g. college data), and a combination of the two (e.g. housing prices and bike sharing).

What do we mean by the relationship of a explanatory variable to a response? There are multiple valid interpretations that are used in regression that are important to distinguish.

- The explanatory variable is *a good predictor* of the response.
- The explanatory variable *is necessary* for good prediction of the response (among the set of variables we are considering).
- Changes in the explanatory variable *cause* the response to change (causality).

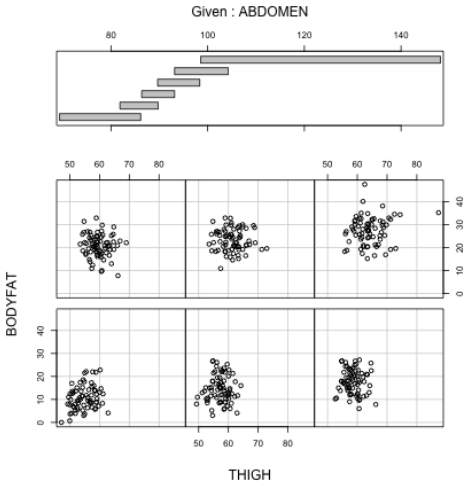
We can visualize the difference in the first and second with plots. Being a good predictor is like the pairwise scatter plots from before, in which case both thigh and abdominal circumference are good predictors of percentage of body fat.

```
pairs(body[, c("BODYFAT", "THIGH", "ABDOMEN")])
```



But in fact if we know the abdominal circumference, the thigh circumference does not tell us much more. A **coplot** visualizes this relationship, by plotting the relationship between two variables, conditional on the value of another. In otherwords, it plots the scatter plot of percent body fat against thigh, but only for those points for abdomen in a certain range (with the ranges indicated at the top).

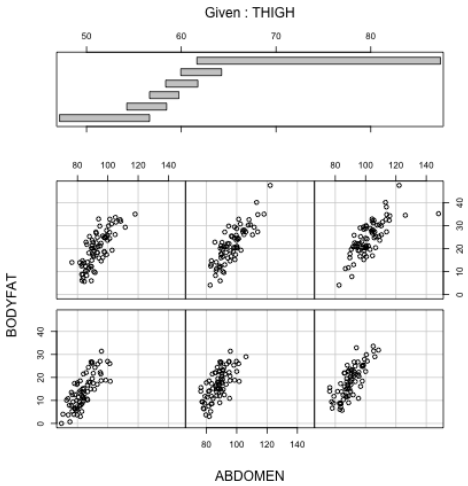
```
coplot(BODYFAT ~ THIGH | ABDOMEN, data = body)
```



We see there is no longer a strong relationship between percentage body fat and thigh circumference for specific values of abdomen circumference

The same is not true, however, for the reverse,

```
coplot(BODYFAT ~ ABDOMEN | THIGH, data = body)
```

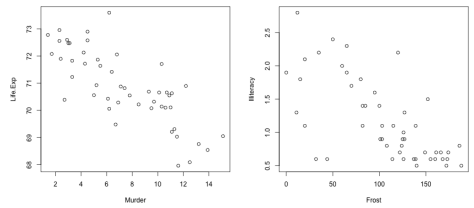


We will see later in the course when we have many variables the answers to these three questions are not always the same (and that we can't always answer all of them). We will almost always be able to say something about the first two, but the last is often not possible.

6.2.1 Causality

Often a (unspoken) goal of linear regression can be to determine whether something 'caused' something else. It is critical to remember that whether you can attribute causality to a variable depends on how your data was collected. Specifically, most people often have **observational data**, i.e. they sample subjects or units from the population and then measure the variables that naturally occur on the units they happen to sample. In general, you cannot determine causality by just collecting observations on existing subjects. You can only observe what is likely to naturally occur jointly in your population, often due to other causes. Consider the following data on the relationship between the murder rate and the life expectancy of different states or that of Illiteracy and Frost:

```
st <- as.data.frame(state.x77)
colnames(st)[4] = "Life.Exp"
colnames(st)[6] = "HS.Grad"
par(mfrow = c(1, 2))
with(st, plot(Murder, Life.Exp))
with(st, plot(Frost, Illiteracy))
```



Question: What do you observe in the plotted relationship between the murder rate and the life expectancy ? What about between frost levels and illiteracy? What would it mean to (wrongly) assume causality here?

It is a common mistake in regression to to jump to the conclusion that one variable causes the other, but all you can really say is that there is a strong relationship in the population, i.e. when you observe one value of the variable you are highly likely to observed a particular value of the other.

Can you ever claim causality? Yes, if you run an **experiment**; this is where you *assign* what the value of the predictors are for every observation *independently from any other variable*. An example is a clinical trial, where patients are randomly assigned a treatment.

It's often not possible to run an experiment, especially in the social sciences or working with humans (you can't assign a murder rate in a state and sit back and see what the effect is on life expectancy!). In the absence of an experiment, it is common to collect a lot of other variables that might also explain the response, and ask our second question – "how necessary is it (in addition to these other

variables)?" with the idea that this is a proxy for causality. This is sometime called "controlling" for the effect of the other variables, but it is important to remember that this is not the same as causality.

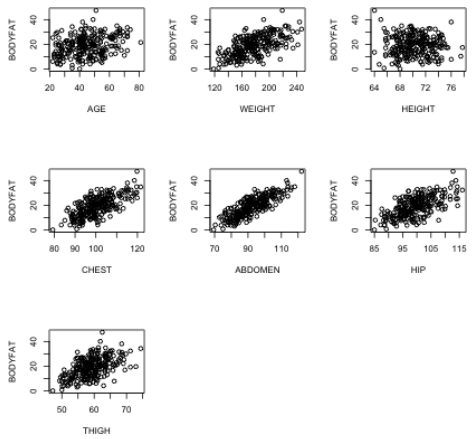
Regardless, the analysis of observational and experimental data often both use linear regression.[1] It's what conclusions you can draw that differ.

6.3 Multiple Linear Regression

The body fat dataset is a useful one to use to explain linear regression because all of the variables are continuous and the relationships are reasonably linear.

Let us look at the plots between the response variable (`bodyfat`) and all the explanatory variables (we'll remove the outliers for this plot).

```
par(mfrow = c(3, 3))
for (i in 2:8) {
    plot(body[-ou, i], body[-ou, 1], xlab = names(body)[i],
        ylab = "BODYFAT")
}
par(mfrow = c(1, 1))
```



Most pairwise relationships seem to be linear. The clearest relationship is between bodyfat and abdomen. The next clearest is between bodyfat and chest.

We can expand the simple regression we used earlier to include more variables.

$$y = \beta_0 + \beta_1 x^{(1)} + \beta_2 x^{(2)} + ...$$

[1] Note that there can be problems with using linear regression in experiments when only some of the explanatory variables are randomly assigned. Similarly, there are other methods that you can use in observational studies that can, within some strict limitations, get closer to answering questions of causality.

6.3.1 Regression Line vs Regression Plane

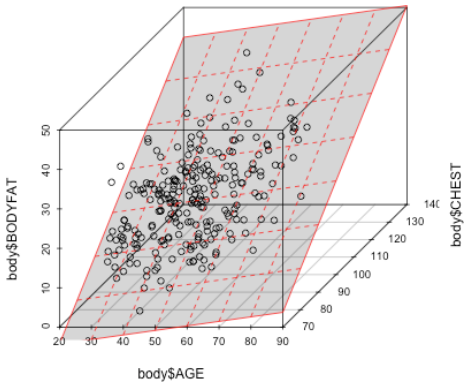
In simple linear regression (when there is only one explanatory variable), the fitted regression equation describes a line. If we have two variables, it defines a plane. This plane can be plotted in a 3D plot when there are two explanatory variables. When the number of explanatory variables is 3 or more, we have a general linear combination[2] and we cannot plot this relationship.

To illustrate this, let us fit a regression equation to bodyfat percentage in terms of age and chest circumference:

```
ft2 = lm(BODYFAT ~ AGE + CHEST, data = body)
```

We can visualize this 3D plot:

```
library(scatterplot3d)
sp = scatterplot3d(body$AGE, body$CHEST, body$BODYFAT)
sp$plane3d(ft2, lty.box = "solid", draw_lines = TRUE,
    draw_polygon = TRUE, col = "red")
```



6.3.2 How to estimate the coefficients?

We can use the same principle as before. Specifically, for any selection of our β_j coefficients, we get a predicted or fitted value $\hat{y}$. Then we can look for the β_j which minimize our loss

$$\sum_{i=1}^{n} \ell(y_i, \hat{y}_i)$$

Again, standard regression uses squared-error loss,

$$\sum_{i=1}^{n} (y_i - \hat{y}_i)^2.$$

[2] so defines a linear subspace

We again can fit this by using `lm` in R, with similar syntax as before:

```
ft = lm(BODYFAT ~ AGE + WEIGHT + HEIGHT + CHEST + ABDOMEN +
    HIP + THIGH, data = body)
summary(ft)
```

```
##
## Call:
## lm(formula = BODYFAT ~ AGE + WEIGHT + HEIGHT + CHEST + ABDOMEN +
##     HIP + THIGH, data = body)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.0729  -3.2387  -0.0782   3.0623  10.3611
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.748e+01  1.449e+01  -2.585  0.01031 *
## AGE          1.202e-02  2.934e-02   0.410  0.68246
## WEIGHT      -1.392e-01  4.509e-02  -3.087  0.00225 **
## HEIGHT      -1.028e-01  9.787e-02  -1.051  0.29438
## CHEST       -8.312e-04  9.989e-02  -0.008  0.99337
## ABDOMEN      9.685e-01  8.531e-02  11.352  < 2e-16 ***
## HIP         -1.834e-01  1.448e-01  -1.267  0.20648
## THIGH        2.857e-01  1.362e-01   2.098  0.03693 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.438 on 244 degrees of freedom
## Multiple R-squared:  0.7266, Adjusted R-squared:  0.7187
## F-statistic: 92.62 on 7 and 244 DF,  p-value: < 2.2e-16
```

In fact, if we want to use all the variables in a `data.frame` we can use a simpler notation:

```
ft = lm(BODYFAT ~ ., data = body)
```

Notice how similar the output to the function above is to the case of simple linear regression. R has fit a linear equation for the variable BODYFAT in terms of the variables AGE, WEIGHT, HEIGHT, CHEST, ABDOMEN, HIP and THIGH. Again, the summary of the output gives each variable and its estimated coefficient,

$$\begin{aligned} BODYFAT = &-37.48 + 0.012 * AGE - 0.139 * WEIGHT - 0.102 * HEIGHT \\ &- 0.0008 * CHEST + 0.968 * ABDOMEN - 0.183 * HIP + 0.286 * THIGH \end{aligned}$$

We can also write down explicit equations for the estimates of the $\hat{\beta}_j$ when we use squared-error loss, though we won't give them here (they are usually given in matrix notation).

6.3.3 Interpretation of the regression equation

Here the coefficient $\hat{\beta}_1$ is interpreted as the average increase in y for unit increase in $x^{(1)}$, *provided all other explanatory variables* $x^{(2)}, \ldots, x^{(p)}$ *are kept constant.* More generally for $j \geq 1$, the coefficient $\hat{\beta}_j$ is interpreted as the average increase in y for unit increase in $x^{(j)}$ provided all other explanatory variables $x^{(k)}$ for $k \neq j$ are kept constant. The intercept $\hat{\beta}_0$ is interpreted as the average value of y when all the explanatory variables are equal to zero.

In the body fat example, the fitted regression equation as we have seen is:

$$BODYFAT = -37.48 + 0.012 * AGE - 0.139 * WEIGHT - 0.102 * HEIGHT \\ - 0.0008 * CHEST + 0.968 * ABDOMEN - 0.183 * HIP + 0.286 * THIGH$$

The coefficient of 0.968 can be interpreted as the average percentage increase in bodyfat percentage per unit (i.e., 1 cm) increase in Abdomen circumference provided all the other explanatory variables age, weight, height, chest circumference, hip circumference and thigh circumference are kept unchanged.

Question: Do the signs of the fitted regression coefficients make sense?

6.3.3.1 Scaling and the size of the coefficient

It's often tempting to look at the size of the β_j as a measure of how "important" the variable j is in predicting the response y. However, it's important to remember that β_j is relative to the scale of the input $x^{(j)}$ – it is the change in y for *one unit change* in $x^{(j)}$. So, for example, if we change from measurements in cm to mm (i.e. multiply $x^{(j)}$ by 10) then we will get a $\hat{\beta}_j$ that is divided by 10:

```
tempBody <- body
tempBody$ABDOMEN <- tempBody$ABDOMEN * 10
ftScale = lm(BODYFAT ~ ., data = tempBody)
cat("Coefficients with Abdomen in mm:\n")
```

```
## Coefficients with Abdomen in mm:
```

```
coef(ftScale)
```

```
##   (Intercept)           AGE        WEIGHT        HEIGHT         CHEST
## -3.747573e+01  1.201695e-02 -1.392006e-01 -1.028485e-01 -8.311678e-04
##       ABDOMEN           HIP         THIGH
##  9.684620e-02 -1.833599e-01  2.857227e-01
```

```
cat("Coefficients with Abdomen in cm:\n")
```

```
## Coefficients with Abdomen in cm:
```

```
coef(ft)
```

```
##   (Intercept)           AGE        WEIGHT        HEIGHT         CHEST
## -3.747573e+01  1.201695e-02 -1.392006e-01 -1.028485e-01 -8.311678e-04
##       ABDOMEN           HIP         THIGH
##  9.684620e-01 -1.833599e-01  2.857227e-01
```

For this reason, it is not uncommon to scale the explanatory variables – i.e. divide each variable by its standard deviation – before running the regression:

```
tempBody <- body
tempBody[, -1] <- scale(tempBody[, -1])
ftScale = lm(BODYFAT ~ ., data = tempBody)
cat("Coefficients with variables scaled:\n")
```

```
## Coefficients with variables scaled:
```

```
coef(ftScale)
```

```
## (Intercept)         AGE      WEIGHT      HEIGHT       CHEST     ABDOMEN
## 19.15079365  0.15143812 -4.09098792 -0.37671913 -0.00700714 10.44300051
##         HIP       THIGH
## -1.31360120  1.50003073
```

```
cat("Coefficients on original scale:\n")
```

```
## Coefficients on original scale:
```

```
coef(ft)
```

```
##   (Intercept)           AGE        WEIGHT        HEIGHT         CHEST
## -3.747573e+01  1.201695e-02 -1.392006e-01 -1.028485e-01 -8.311678e-04
##       ABDOMEN           HIP         THIGH
##  9.684620e-01 -1.833599e-01  2.857227e-01
```

```
sdVar <- apply(body[, -1], 2, sd)
cat("Sd per variable:\n")
```

```
## Sd per variable:
```

```
sdVar
```

```
##       AGE    WEIGHT    HEIGHT     CHEST   ABDOMEN       HIP     THIGH
## 12.602040 29.389160  3.662856  8.430476 10.783077  7.164058  5.249952
```

```
cat("Ratio of scaled lm coefficient to original lm coefficient\n")
```

```
## Ratio of scaled lm coefficient to original lm coefficient
```

```
coef(ftScale)[-1]/coef(ft)[-1]
```

```
##       AGE    WEIGHT    HEIGHT     CHEST   ABDOMEN       HIP     THIGH
## 12.602040 29.389160  3.662856  8.430476 10.783077  7.164058  5.249952
```

Now the interpretation of the β_j is that per standard deviation change in the variable x^j, what is the change in y, again all other variables remaining constant.

6.3.3.2 Correlated Variables

The interpretation of the coefficient $\hat{\beta}_j$ depends crucially on the other explanatory variables $x^{(k)}, k \neq j$ that are present in the equation (this is because of the phrase "all other explanatory variables kept constant").

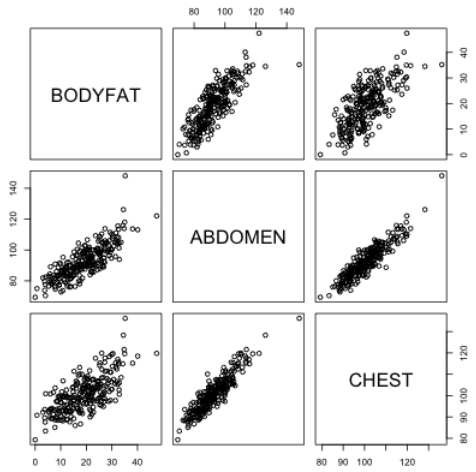
For the bodyfat data, we have seen that the variables chest thigh and hip and abdomen circumference are highly correlated:

```
cor(body[, c("HIP", "THIGH", "ABDOMEN", "CHEST")])
```

```
##                HIP     THIGH   ABDOMEN     CHEST
## HIP     1.0000000 0.8964098 0.8740662 0.8294199
## THIGH   0.8964098 1.0000000 0.7666239 0.7298586
## ABDOMEN 0.8740662 0.7666239 1.0000000 0.9158277
## CHEST   0.8294199 0.7298586 0.9158277 1.0000000
```

Notice that both `CHEST` and `ABDOMEN` are linearly related to `BODYFAT`

```
pairs(body[, c("BODYFAT", "ABDOMEN", "CHEST")])
```



Individual linear regressions would show *very* significant values for both `CHEST` and `ABDOMEN`:

```
summary(lm(BODYFAT ~ CHEST, data = body))$coef
```

```
##                Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) -51.1715853 4.51985295 -11.32152 2.916303e-24
## CHEST         0.6974752 0.04467377  15.61263 8.085369e-39
```

```
summary(lm(BODYFAT ~ ABDOMEN, data = body))$coef
```

```
##                Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) -39.2801847 2.66033696 -14.76512 6.717944e-36
## ABDOMEN       0.6313044 0.02855067  22.11172 9.090067e-61
```

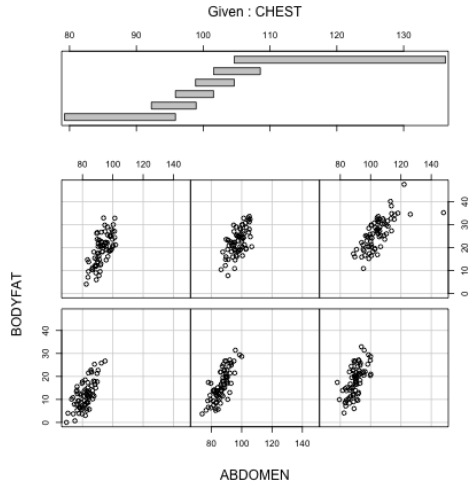
But when we look at the multiple regression, we see `ABDOMEN` is significant and not `CHEST`:

```
round(summary(ft)$coef, digits = 3)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -37.476     14.495  -2.585    0.010
## AGE            0.012      0.029   0.410    0.682
## WEIGHT        -0.139      0.045  -3.087    0.002
## HEIGHT        -0.103      0.098  -1.051    0.294
## CHEST         -0.001      0.100  -0.008    0.993
## ABDOMEN        0.968      0.085  11.352    0.000
## HIP           -0.183      0.145  -1.267    0.206
## THIGH          0.286      0.136   2.098    0.037
```

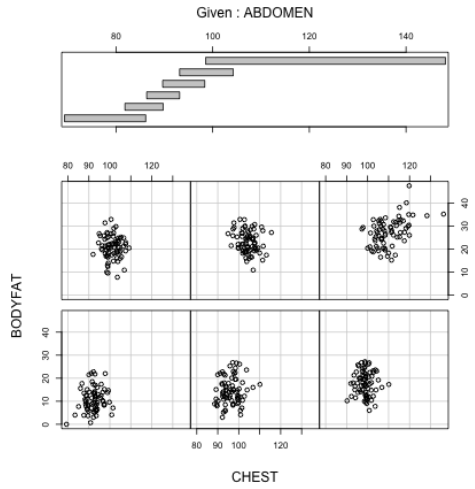
This is because coefficient assigned to `ABDOMEN` and `CHEST` tells us how the response changes *as the other variables stay the same.* This interpretation of β_j ties directly back to our coplots and can help us understand how this is different from an individual regression on each variable. A coplot plots the response (`BODYFAT`) against a variable for a "fixed" value of another variable (i.e. a small range of values). When we do this with `ABDOMEN` for fixed values of `CHEST` we still see a strong relationship between `ABDOMEN` and `BODYFAT`

```
coplot(BODYFAT ~ ABDOMEN | CHEST, data = body)
```



But the other way around shows us for a "fixed" value of `ABDOMEN`, `CHEST` doesn't have much relationship with `BODYFAT`

```
coplot(BODYFAT ~ CHEST | ABDOMEN, data = body)
```



This is the basic idea behind the interpretation of the coefficient β_j in a multiple regression, only for regression it is holding *all* of the other variables fixed, not just one.

What if we didn't include `ABDOMEN` and `THIGH` in our regression? (ie. a model based on age, weight, height, chest and hip):

```
ft1 = lm(BODYFAT ~ AGE + WEIGHT + HEIGHT + CHEST +
    HIP, data = body)
round(coef(ft), 4)
```

```
## (Intercept)         AGE      WEIGHT      HEIGHT       CHEST     ABDOMEN
##    -37.4757      0.0120     -0.1392     -0.1028     -0.0008      0.9685
##         HIP       THIGH
##     -0.1834      0.2857
```

```
round(coef(ft1), 4)
```

```
## (Intercept)         AGE      WEIGHT      HEIGHT       CHEST         HIP
##    -53.9871      0.1290     -0.0526     -0.3146      0.5148      0.4697
```

See now that the actually coefficient values are quite different from the previous one – and they have different interpretations as well. In this model, `CHEST` is now *very* significant.

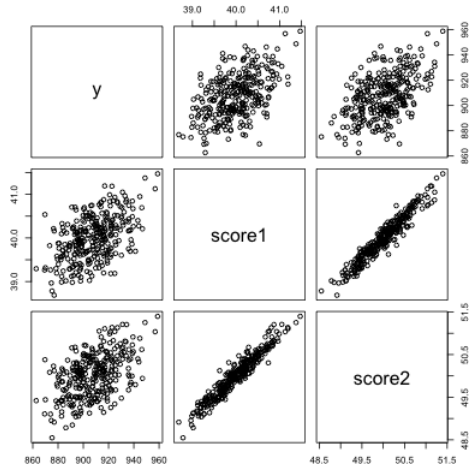
```
round(summary(ft1)$coeff, 4)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -53.9871    17.1362 -3.1505   0.0018
## AGE           0.1290     0.0308  4.1901   0.0000
## WEIGHT       -0.0526     0.0534 -0.9841   0.3260
## HEIGHT       -0.3146     0.1176 -2.6743   0.0080
## CHEST         0.5148     0.1080  4.7662   0.0000
## HIP           0.4697     0.1604  2.9286   0.0037
```

It's important to remember that the β_j are not a fixed, immutable property of the variable, but are only interpretable *in the context of the other variables.* So the interpretation of $\widehat{\beta}_j$ (and it's significance) is a function of the x data you have. If you only observe x^j large when $x^{(k)}$ is also large (i.e. strong and large positive correlation), then you have little data where $x^{(j)}$ is changing over a range of values while $x^{(k)}$ is basically constant. For example, if you fix `ABDOMEN` to be 100in, the range of values of `CHEST` is tightly constrained to roughly 95-112in, i.e. `CHEST` doesn't *actually* change much in the population if you fix `ABDOMEN`.

Here's some simulated data demonstrating this. Notice both variables are pretty correlated with the response y

```
set.seed(275382)
n <- 300
trueQuality <- rnorm(n)
score2 <- (trueQuality + 100) * 0.5 + rnorm(n, sd = 0.1)
score1 <- (trueQuality + 80) * 0.5 + rnorm(n, sd = 0.1)
y <- 8 + 10 * score1 + 10 * score2 + rnorm(n, sd = 15)
x <- data.frame(y, score1, score2)
pairs(x)
```



But if I look at the regression summary, I don't get any significance.

```
summary(lm(y ~ ., data = x))
```

```
##
## Call:
## lm(formula = y ~ ., data = x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -46.067 -10.909   0.208   9.918  38.138
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  110.246     97.344   1.133    0.258
## score1         8.543      6.301   1.356    0.176
## score2         9.113      6.225   1.464    0.144
##
## Residual standard error: 15.09 on 297 degrees of freedom
## Multiple R-squared:  0.2607, Adjusted R-squared:  0.2557
## F-statistic: 52.37 on 2 and 297 DF,  p-value: < 2.2e-16
```

However, individually, each score is highly significant in predicting y

```
summary(lm(y ~ score1, data = x))
```

```
##
## Call:
## lm(formula = y ~ score1, data = x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

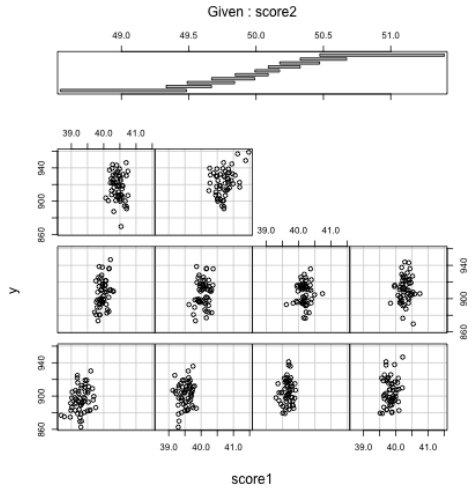
```
## -47.462 -10.471   0.189  10.378  38.868
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  211.072     68.916   3.063  0.00239 **
## score1        17.416      1.723  10.109  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.12 on 298 degrees of freedom
## Multiple R-squared:  0.2554, Adjusted R-squared:  0.2529
## F-statistic: 102.2 on 1 and 298 DF,  p-value: < 2.2e-16
```

```
summary(lm(y ~ score2, data = x))
```

```
##
## Call:
## lm(formula = y ~ score2, data = x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -44.483 -11.339   0.195  11.060  40.327
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   45.844     85.090   0.539     0.59
## score2        17.234      1.701  10.130   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.11 on 298 degrees of freedom
## Multiple R-squared:  0.2561, Adjusted R-squared:  0.2536
## F-statistic: 102.6 on 1 and 298 DF,  p-value: < 2.2e-16
```

They just don't add further information *when added to the existing variable already included*. Looking at the coplot, we can visualize this – for each bin of score 2 (i.e. as close as we can get to constant), we have very little further change in y.

```
coplot(y ~ score1 | score2, number = 10, data = x)
```



We will continually return the effect of correlation in understanding multiple regression.

What kind of relationship with y does β_j measure?

If we go back to our possible questions we could ask about the relationship between a single variable j and the response, then $\widehat{\beta}_j$ answers the second question: how necessary is variable j to the predition of y *above and beyond the other variables*? We can see this in our above description of "being held constant" – if when the other variables aren't changing, $\widehat{\beta}_j$ tells us how much y moves on average as only $x^{(j)}$ changes. If $\widehat{\beta}_j$ is close to 0, then changes in $x^{(j)}$ aren't affecting y much for fixed values of the other coordinates.

Why the β_j does not measure causality

Correlation in our variables is one important reason why the value of β_j does **not** measure causality, i.e whether a change in $x^{(j)}$ *caused* y to change. If $x^{(j)}$ is always large when $x^{(k)}$ is large, there is no (or little) data to evaluate whether a large $x^{(j)}$ in the presence of a *small* $x^{(k)}$ would result in a large y. [3]

Again, it can be helpful to compare what you would expect if you could create a randomized experiment. You would choose individuals with a particular value of `ABDOMEN` circumference, say 100cm. Then for some individuals you would change their `CHEST` size to be 80cm and for others 120cm, and then measure the resulting `BODYFAT`. Just writing it down makes it clear why such an experiment is impossible – and also why circumference on its own is a poor candidate for causing anything. Rather internal bodily mechanisms result in all three variables (`ABDOMEN`, `CHEST`, and `BODYFAT`) to be larger, without one causing another.

[3] Issues with making causal statements go beyond just whether variables are correlated, but correlation among the variables is a major issue.

6.4 Important measurements of the regression estimate

6.4.1 Fitted Values and Multiple R^2

Any regression equation can be used to predict the value of the response variable given values of the explanatory variables, which we call $\hat{y}(x)$. We can get a fitted value for any value x. For example, consider our original fitted regression equation obtained by applying `lm` with bodyfat percentage against all of the variables as explanatory variables:

$$BODYFAT = -37.48 + 0.01202 * AGE - 0.1392 * WEIGHT - 0.1028 * HEIGHT \\ - 0.0008312 * CHEST + 0.9685 * ABDOMEN - 0.1834 * HIP + 0.2857 * THIGH$$

Suppose a person X (who is of 30 years of age, weighs 180 pounds and is 70 inches tall) wants to find out his bodyfat percentage. Let us say that he is able to measure his chest circumference as 90 cm, abdomen circumference as 86 cm, hip circumference as 97 cm and thigh circumference as 60 cm. Then he can simply use the regression equation to predict his bodyfat percentage as:

```
bf.pred = -37.48 + 0.01202 * 30 - 0.1392 * 180 - 0.1028 *
    70 - 0.0008312 * 90 + 0.9685 * 86 - 0.1834 * 97 +
    0.2857 * 60
bf.pred
```

```
## [1] 13.19699
```

The predictions given by the fitted regression equation *for each of the observations} are known as **fitted values**, $\hat{y}_i = \hat{y}(x_i)$. For example, in the bodyfat dataset, the first observation (first row) is given by:

```
obs1 = body[1, ]
obs1
```

```
##   BODYFAT AGE WEIGHT HEIGHT CHEST ABDOMEN  HIP THIGH
## 1    12.3  23 154.25  67.75  93.1    85.2 94.5    59
```

The observed value of the response (bodyfat percentage) for this individual is 12.3 %. The prediction for this person's response given by the regression equation (**??**) is

```
-37.48 + 0.01202 * body[1, "AGE"] - 0.1392 * body[1,
    "WEIGHT"] - 0.1028 * body[1, "HEIGHT"] - 0.0008312 *
    body[1, "CHEST"] + 0.9685 * body[1, "ABDOMEN"] -
    0.1834 * body[1, "HIP"] + 0.2857 * body[1, "THIGH"]
```

```
## [1] 16.32398
```

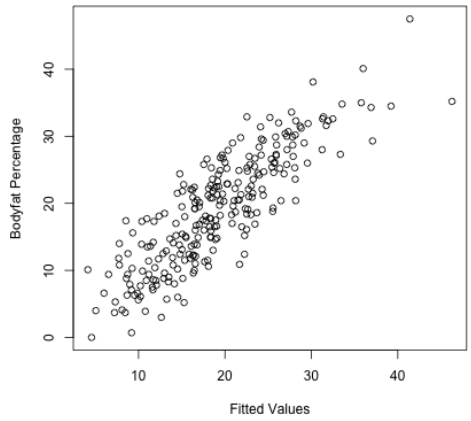
Therefore the *fitted value* for the first observation is 16.424%. R directly calculates all fitted values and they are stored in the $lm()$ object. You can obtain these via:

```
head(fitted(ft))
```

```
##        1        2        3        4        5        6
## 16.32670 10.22019 18.42600 11.89502 25.97564 16.28529
```

If the regression equation fits the data well, we would expect the fitted values to be close to the observed responses. We can check this by just plotting the fitted values against the observed response values.

```
plot(fitted(ft), body$BODYFAT, xlab = "Fitted Values",
    ylab = "Bodyfat Percentage")
```



We can quantify how good of a fit our model is by taking the correlation between these two values. Specifically, the square of the correlation of y and $\hat{y}$ is known as the **Coefficient of Determination** or **Multiple** R^2 or simply R^2:

$$R^2 = (cor(y_i, \hat{y}_i))^2 .$$

This is an important and widely used measure of the effectiveness of the regression equation and given in our summary the `lm` fit.

```
cor(body$BODYFAT, fitted(ft))^2
```

```
## [1] 0.7265596
```

```
summary(ft)
```

```
##
## Call:
## lm(formula = BODYFAT ~ ., data = body)
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -11.0729  -3.2387  -0.0782   3.0623  10.3611
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.748e+01  1.449e+01  -2.585  0.01031 *
## AGE          1.202e-02  2.934e-02   0.410  0.68246
## WEIGHT      -1.392e-01  4.509e-02  -3.087  0.00225 **
## HEIGHT      -1.028e-01  9.787e-02  -1.051  0.29438
## CHEST       -8.312e-04  9.989e-02  -0.008  0.99337
## ABDOMEN      9.685e-01  8.531e-02  11.352  < 2e-16 ***
## HIP         -1.834e-01  1.448e-01  -1.267  0.20648
## THIGH        2.857e-01  1.362e-01   2.098  0.03693 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.438 on 244 degrees of freedom
## Multiple R-squared:  0.7266, Adjusted R-squared:  0.7187
## F-statistic: 92.62 on 7 and 244 DF,  p-value: < 2.2e-16
```

A high value of R^2 means that the fitted values (given by the fitted regression equation) are close to the observed values and hence indicates that the regression equation fits the data well. A low value, on the other hand, means that the fitted values are far from the observed values and hence the regression line does not fit the data well.

Note that R^2 has no units (because its a correlation). In other words, it is scale-free.

6.4.2 Residuals and Residual Sum of Squares (RSS)

For every point in the scatter the error we make in our prediction on a specific observation is the **residual** and is defined as

$$r_i = y_i - \hat{y}_i$$

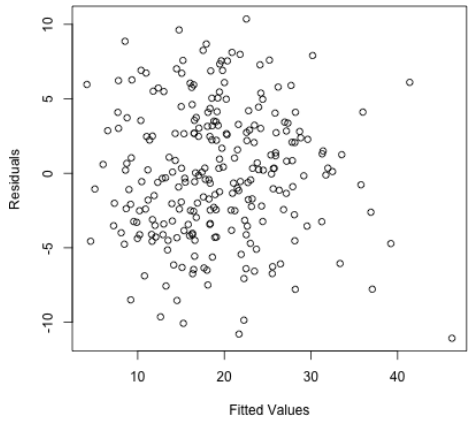
Residuals are again so important that $lm()$ automatically calculates them for us and they are contained in the lm object created.

```
head(residuals(ft))
```

```
##         1         2         3         4         5         6
## -4.026695 -4.120189  6.874004 -1.495017  2.724355  4.614712
```

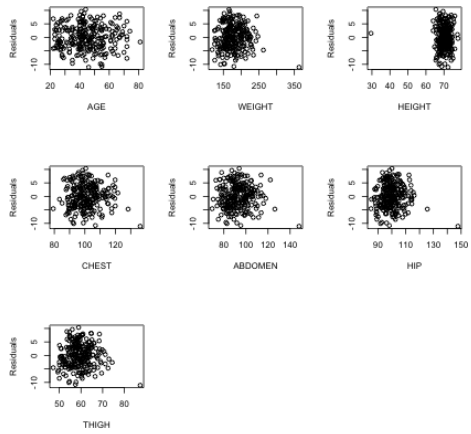
A common way of looking at the residuals is to plot them against the fitted values.

```
plot(fitted(ft), residuals(ft), xlab = "Fitted Values",
    ylab = "Residuals")
```



One can also plot the residuals against each of the explanatory variables (note we didn't remove the outliers in our regression so we include them in our plots).

```
par(mfrow = c(3, 3))
for (i in 2:8) {
    plot(body[, i], ft$residuals, xlab = names(body)[i],
        ylab = "Residuals")
}
par(mfrow = c(1, 1))
```



The residuals represent what is left in the response (y) after all the linear effects of the explanatory variables are taken out.

One consequence of this is that the residuals are *uncorrelated with every explanatory variable*. We can check this in easily in the body fat example.

```
for (i in 2:8) {
    cat("Correlation with", names(body)[i], ":\t")
    cat(cor(body[, i], residuals(ft)), "\n")
}
```

```
## Correlation with AGE :   -1.754044e-17
## Correlation with WEIGHT :    4.71057e-17
## Correlation with HEIGHT :    -1.720483e-15
## Correlation with CHEST : -4.672628e-16
## Correlation with ABDOMEN :   -7.012368e-16
## Correlation with HIP :   -8.493675e-16
## Correlation with THIGH : -5.509094e-16
```

Moreover, as we discussed in simple regression, the residuals always have mean zero:

```
mean(ft$residuals)
```

```
## [1] 2.467747e-16
```

Again, these are automatic properties of any least-squares regression. *This is not evidence that you have a good fit or that model makes sense!*

Also, if one were to fit a regression equation to the residuals in terms of the same explanatory variables, then the fitted regression equation will have all coefficients exactly equal to zero:

```
m.res = lm(ft$residuals ~ body$AGE + body$WEIGHT +
    body$HEIGHT + body$CHEST + body$ABDOMEN + body$HIP +
    body$THIGH)
summary(m.res)
```

```
##
## Call:
## lm(formula = ft$residuals ~ body$AGE + body$WEIGHT + body$HEIGHT +
##     body$CHEST + body$ABDOMEN + body$HIP + body$THIGH)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.0729  -3.2387  -0.0782   3.0623  10.3611
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.154e-14  1.449e+01       0        1
## body$AGE      1.282e-17  2.934e-02       0        1
## body$WEIGHT   1.057e-16  4.509e-02       0        1
## body$HEIGHT  -1.509e-16  9.787e-02       0        1
## body$CHEST    1.180e-16  9.989e-02       0        1
```

```
## body$ABDOMEN -2.452e-16  8.531e-02       0        1
## body$HIP     -1.284e-16  1.448e-01       0        1
## body$THIGH   -1.090e-16  1.362e-01       0        1
##
## Residual standard error: 4.438 on 244 degrees of freedom
## Multiple R-squared:  6.384e-32,  Adjusted R-squared:  -0.02869
## F-statistic: 2.225e-30 on 7 and 244 DF,  p-value: 1
```

If the regression equation fits the data well, the residuals are supposed to be small. One popular way of assessing the size of the residuals is to compute their sum of squares. This quantity is called the **Residual Sum of Squares (RSS)**.

```
rss.ft = sum((ft$residuals)^2)
rss.ft
```

```
## [1] 4806.806
```

Note that RSS depends on the units in which the response variable is measured.

Relationship to R^2

There is a very simple relationship between RSS and R^2 (recall that R^2 is the square of the correlation between the response values and the fitted values):

$$R^2 = 1 - \frac{RSS}{TSS}$$

where TSS stands for Total Sum of Squares and is defined as

$$TSS = \sum_{i=1}^{n} (y_i - \bar{y})^2 .$$

TSS is just the variance of y without the $1/(n-1)$ term.

It is easy to verify this formula in R.

```
rss.ft = sum((ft$residuals)^2)
rss.ft
```

```
## [1] 4806.806
```

```
tss = sum(((body$BODYFAT) - mean(body$BODYFAT))^2)
1 - (rss.ft/tss)
```

```
## [1] 0.7265596
```

```
summary(ft)
```

```
##
## Call:
## lm(formula = BODYFAT ~ ., data = body)
##
```

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.0729  -3.2387  -0.0782   3.0623  10.3611
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.748e+01  1.449e+01  -2.585  0.01031 *
## AGE          1.202e-02  2.934e-02   0.410  0.68246
## WEIGHT      -1.392e-01  4.509e-02  -3.087  0.00225 **
## HEIGHT      -1.028e-01  9.787e-02  -1.051  0.29438
## CHEST       -8.312e-04  9.989e-02  -0.008  0.99337
## ABDOMEN      9.685e-01  8.531e-02  11.352  < 2e-16 ***
## HIP         -1.834e-01  1.448e-01  -1.267  0.20648
## THIGH        2.857e-01  1.362e-01   2.098  0.03693 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.438 on 244 degrees of freedom
## Multiple R-squared:  0.7266, Adjusted R-squared:  0.7187
## F-statistic: 92.62 on 7 and 244 DF,  p-value: < 2.2e-16
```

If we did not have any explanatory variables, then we would predict the value of bodyfat percentage for any individual by simply the mean of the bodyfat values in our sample. The total squared error for this prediction is given by TSS. On the other hand, the total squared error for the prediction using linear regression based on the explanatory variables is given by RSS. Therefore $1 - R^2$ represents the reduction in the squared error because of the explanatory variables.

6.4.3 Behaviour of RSS (and R^2) when variables are added or removed from the regression equation

The value of RSS always increases when one or more explanatory variables are removed from the regression equation. For example, suppose that we remove the variable abdomen circumference from the regression equation. The new RSS will then be:

```
ft.1 = lm(BODYFAT ~ AGE + WEIGHT + HEIGHT + CHEST +
    HIP + THIGH, data = body)
rss.ft1 = summary(ft.1)$r.squared
rss.ft1
```

```
## [1] 0.5821305
```

```
summary(ft$r.squared)
```

```
## Length  Class   Mode
##      0   NULL   NULL
```

Notice that there is a quite a lot of increase in the RSS. What if we had kept ABDOMEN in the model but dropped the variable CHEST?

```
ft.2 = lm(BODYFAT ~ AGE + WEIGHT + HEIGHT + ABDOMEN +
    HIP + THIGH, data = body)
rss.ft2 = summary(ft.2)$r.squared
rss.ft2
```

```
## [1] 0.7265595
```

```
rss.ft
```

```
## [1] 4806.806
```

The RSS again increases but by a very very small amount. This therefore suggests that Abdomen circumference is a more important variable in this regression compared to Chest circumference.

The moral of this exercise is the following. The RSS always increases when variables are dropped from the regression equation. However the amount of increase varies for different variables. We can understand the importance of variables in a multiple regression equation by noting the amount by which the RSS increases when the individual variables are dropped. We will come back to this point while studying inference in the multiple regression model.

Because RSS has a direct relation to R^2 via $R^2 = 1 - (RSS/TSS)$, one can see R^2 decreases when variables are removed from the model. However the amount of decrease will be different for different variables. For example, in the body fat dataset, after removing the abdomen circumference variable, R^2 changes to:

```
ft.1 = lm(BODYFAT ~ AGE + WEIGHT + HEIGHT + CHEST +
    HIP + THIGH, data = body)
R2.ft1 = summary(ft.1)$r.squared
R2.ft1
```

```
## [1] 0.5821305
```

```
R2.ft = summary(ft)$r.squared
R2.ft
```

```
## [1] 0.7265596
```

Notice that there is a lot of decrease in R^2. What happens if the variable Chest circumference is dropped.

```
ft.2 = lm(BODYFAT ~ AGE + WEIGHT + HEIGHT + ABDOMEN +
    HIP + THIGH, data = body)
R2.ft2 = summary(ft.2)$r.squared
R2.ft2
```

```
## [1] 0.7265595
```

```
R2.ft
```

```
## [1] 0.7265596
```

There is now a very very small decrease.

6.4.4 Residual Degrees of Freedom and Residual Standard Error

In a regression with p explanatory variables, the residual degrees of freedom is given by $n - p - 1$ (recall that n is the number of observations). This can be thought of as the effective number of residuals. Even though there are n residuals, they are supposed to satisfy $p + 1$ exact equations (they sum to zero and they have zero correlation with each of the p explanatory variables).

The Residual Standard Error is defined as:

$$\sqrt{\frac{\text{Residual Sum of Squares}}{\text{Residual Degrees of Freedom}}}$$

This can be interpreted as the average magnitude of an individual residual and can be used to assess the sizes of residuals (in particular, to find and identify large residual values).

For illustration,

```
ft = lm(BODYFAT ~ AGE + WEIGHT + HEIGHT + CHEST + ABDOMEN +
    HIP + THIGH, data = body)
n = nrow(body)
p = 7
rs.df = n - p - 1
rs.df
```

```
## [1] 244
```

```
ft = lm(BODYFAT ~ AGE + WEIGHT + HEIGHT + CHEST + ABDOMEN +
    HIP + THIGH, data = body)
rss = sum((ft$residuals)^2)
rse = sqrt(rss/rs.df)
rse
```

```
## [1] 4.438471
```

Both of these are printed in the summary function in R:

```
summary(ft)
```

```
##
## Call:
```

```
## lm(formula = BODYFAT ~ AGE + WEIGHT + HEIGHT + CHEST + ABDOMEN +
##     HIP + THIGH, data = body)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.0729  -3.2387  -0.0782   3.0623  10.3611
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.748e+01  1.449e+01  -2.585  0.01031 *
## AGE          1.202e-02  2.934e-02   0.410  0.68246
## WEIGHT      -1.392e-01  4.509e-02  -3.087  0.00225 **
## HEIGHT      -1.028e-01  9.787e-02  -1.051  0.29438
## CHEST       -8.312e-04  9.989e-02  -0.008  0.99337
## ABDOMEN      9.685e-01  8.531e-02  11.352  < 2e-16 ***
## HIP         -1.834e-01  1.448e-01  -1.267  0.20648
## THIGH        2.857e-01  1.362e-01   2.098  0.03693 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.438 on 244 degrees of freedom
## Multiple R-squared:  0.7266, Adjusted R-squared:  0.7187
## F-statistic: 92.62 on 7 and 244 DF,  p-value: < 2.2e-16
```

6.5 Multiple Regression With Categorical Explanatory Variables

In many instances of regression, some of the explanatory variables are categorical (note that the response variable is always continuous). For example, consider the (short version of the) *college* dataset that you have already encountered.

```
scorecard <- read.csv(file.path(dataDir, "college.csv"),
    stringsAsFactors = TRUE)
```

We can do a regression here with the retention rate (variable name `RET-FT4`) as the response and all other variables as the explanatory variables. Note that one of the explanatory variables (variable name `CONTROL`) is categorical. This variable represents whether the college is public (1), private non-profit (2) or private for profit (3). Dealing with such categorical variables is a little tricky. To illustrate the ideas here, let us focus on a regression for the retention rate based on just two explanatory variables: the out-of-state tuition and the categorical variable `CONTROL`.

The important thing to note about the variable `CONTROL` is that its *levels* 1, 2 and 3 are completely arbitrary and have no particular meaning. For example,

we could have called its levels A, B, C or Pu, $Pr-np$, $Pr-fp$ as well. If we use the $lm()$ function in the usual way with TUITIONFEE and CONTROL as the explanatory variables, then R will treat CONTROL as a continuous variable which does not make sense:

```
req.bad = lm(RET_FT4 ~ TUITIONFEE_OUT + CONTROL, data = scorecard)
summary(req.bad)
```

```
##
## Call:
## lm(formula = RET_FT4 ~ TUITIONFEE_OUT + CONTROL, data = scorecard)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.69041 -0.04915  0.00516  0.05554  0.33165
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     6.661e-01  9.265e-03   71.90   <2e-16 ***
## TUITIONFEE_OUT  9.405e-06  3.022e-07   31.12   <2e-16 ***
## CONTROL        -8.898e-02  5.741e-03  -15.50   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08741 on 1238 degrees of freedom
## Multiple R-squared:  0.4391, Adjusted R-squared:  0.4382
## F-statistic: 484.5 on 2 and 1238 DF,  p-value: < 2.2e-16
```

The regression coefficient for CONTROL has the usual interpretation (if CONTROL increases by one unit, ...) which does not make much sense because CONTROL is categorical and so increasing it by one unit is nonsensical. So everything about this regression is wrong (and we shouldn't interpret anything from the inference here).

You can check that R is treating CONTROL as a numeric variable by:

```
is.numeric(scorecard$CONTROL)
```

```
## [1] TRUE
```

The correct way to deal with categorical variables in R is to treat them as factors:

```
req = lm(RET_FT4 ~ TUITIONFEE_OUT + as.factor(CONTROL),
    data = scorecard)
summary(req)
```

```
##
## Call:
```

```
## lm(formula = RET_FT4 ~ TUITIONFEE_OUT + as.factor(CONTROL), data = scorecard)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.68856 -0.04910  0.00505  0.05568  0.33150
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)          5.765e-01  7.257e-03  79.434  < 2e-16 ***
## TUITIONFEE_OUT       9.494e-06  3.054e-07  31.090  < 2e-16 ***
## as.factor(CONTROL)2 -9.204e-02  5.948e-03 -15.474  < 2e-16 ***
## as.factor(CONTROL)3 -1.218e-01  3.116e-02  -3.909 9.75e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08732 on 1237 degrees of freedom
## Multiple R-squared:  0.4408, Adjusted R-squared:  0.4394
## F-statistic:   325 on 3 and 1237 DF,  p-value: < 2.2e-16
```

We can make this output a little better by fixing up the factor, rather than having R make it a factor on the fly:

```
scorecard$CONTROL <- factor(scorecard$CONTROL, levels = c(1,
    2, 3), labels = c("public", "private", "private for-profit"))
req = lm(RET_FT4 ~ TUITIONFEE_OUT + CONTROL, data = scorecard)
summary(req)
```

```
##
## Call:
## lm(formula = RET_FT4 ~ TUITIONFEE_OUT + CONTROL, data = scorecard)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.68856 -0.04910  0.00505  0.05568  0.33150
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 5.765e-01  7.257e-03  79.434  < 2e-16 ***
## TUITIONFEE_OUT              9.494e-06  3.054e-07  31.090  < 2e-16 ***
## CONTROLprivate             -9.204e-02  5.948e-03 -15.474  < 2e-16 ***
## CONTROLprivate for-profit -1.218e-01  3.116e-02  -3.909 9.75e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08732 on 1237 degrees of freedom
## Multiple R-squared:  0.4408, Adjusted R-squared:  0.4394
## F-statistic:   325 on 3 and 1237 DF,  p-value: < 2.2e-16
```

Question: What do you notice that is different than our wrong output when the `CONTROL` variable was treated as numeric?

Question: Why is the coefficient of TUITIONFEE so small?

6.5.1 Separate Intercepts: The coefficients of Categorical/Factor variables

What do the multiple coefficients mean for the variable `CONTROL`?

This equation can be written in full as:

$$RET = 0.5765 + 9.4 \times 10^{-6} * TUITIONFEE - 0.0092 * I\left(CONTROL = 2\right) - 0.1218 * I\left(CONTROL = 3\right).$$

The variable $I\left(CONTROL = 2\right)$ is the indicator function, which takes the value 1 if the college has `CONTROL` equal to 2 (i.e., if the college is private non-profit) and 0 otherwise. Similarly the variable $I\left(CONTROL = 3\right)$ takes the value 1 if the college has `CONTROL` equal to 3 (i.e., if the college is private for profit) and 0 otherwise. Variables which take only the two values 0 and 1 are called indicator variables.

Note that the variable $I\left(CONTROL = 1\right)$ does not appear in the regression equation (**??**). This means that the level 1 (i.e., the college is public) is the baseline level here and the effects of -0.0092 and 0.1218 for private for-profit and private non-profit colleges respectively should be interpreted relative to public colleges.

The regression equation (**??**) can effectively be broken down into three equations. For public colleges, the two indicator variables in (**??**) are zero and the equation becomes:

$$RET = 0.5765 + 9.4 \times 10^{-6} * TUITIONFEE.$$

For private non-profit colleges, the equation becomes

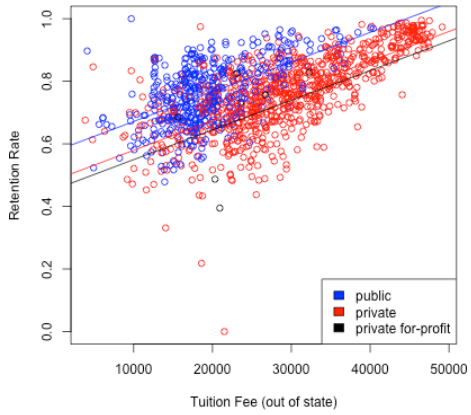
$$RET = 0.5673 + 9.4 \times 10^{-6} * TUITIONFEE.$$

and for private for-profit colleges,

$$RET = 0.4547 + 9.4 \times 10^{-6} * TUITIONFEE.$$

Note that the coefficient of `TUITIONFEE` is the same in each of these equations (only the intercept changes). We can plot a scatterplot together with all these lines.

```
cols <- c("blue", "red", "black")
plot(RET_FT4 ~ TUITIONFEE_OUT, data = scorecard, xlab = "Tuition Fee (out of state)",
    ylab = "Retention Rate", col = cols[scorecard$CONTROL])
baseline <- coef(req)[["(Intercept)"]]
slope <- coef(req)[["TUITIONFEE_OUT"]]
for (ii in 1:nlevels(scorecard$CONTROL)) {
    lev <- levels(scorecard$CONTROL)[[ii]]
    if (ii == 1) {
        abline(a = baseline, b = slope, col = cols[[ii]])
    }
    else {
        abline(a = baseline + coef(req)[[ii + 1]],
            b = slope, col = cols[[ii]])
    }
}
legend("bottomright", levels(scorecard$CONTROL), fill = cols)
```



6.5.2 Separate Slopes: Interactions

What if we want these regression equations to have different slopes as well as different intercepts for each of the types of colleges?

Intuitively, we can do separate regressions for each of the three groups given by the `CONTROL` variable.

Alternatively, we can do this in multiple regression by adding an **interaction variable** between `CONTROL` and `TUITIONFEE` as follows:

```
req.1 = lm(RET_FT4 ~ TUITIONFEE_OUT + CONTROL + TUITIONFEE_OUT:CONTROL,
    data = scorecard)
summary(req.1)
```

```
##
## Call:
## lm(formula = RET_FT4 ~ TUITIONFEE_OUT + CONTROL + TUITIONFEE_OUT:CONTROL,
##     data = scorecard)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.68822 -0.04982  0.00491  0.05555  0.32900
##
## Coefficients:
##                                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                                 5.814e-01  1.405e-02  41.372  < 2e-16
## TUITIONFEE_OUT                              9.240e-06  6.874e-07  13.441  < 2e-16
## CONTROLprivate                             -9.830e-02  1.750e-02  -5.617  2.4e-08
## CONTROLprivate for-profit                  -2.863e-01  1.568e-01  -1.826   0.0681
## TUITIONFEE_OUT:CONTROLprivate               2.988e-07  7.676e-07   0.389   0.6971
## TUITIONFEE_OUT:CONTROLprivate for-profit    7.215e-06  6.716e-06   1.074   0.2829
##
## (Intercept)                                ***
## TUITIONFEE_OUT                             ***
## CONTROLprivate                             ***
## CONTROLprivate for-profit                  .
## TUITIONFEE_OUT:CONTROLprivate
## TUITIONFEE_OUT:CONTROLprivate for-profit
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08734 on 1235 degrees of freedom
## Multiple R-squared:  0.4413, Adjusted R-squared:  0.4391
## F-statistic: 195.1 on 5 and 1235 DF,  p-value: < 2.2e-16
```

Note that this regression equation has two more coefficients compared to the previous regression (which did not have the interaction term). The two additional variables are the product of the terms of each of the previous terms: $TUITIONFEE * I(CONTROL = 2)$ and $TUITIONFEE * I(CONTROL = 3)$.

Question: The presence of these product terms means that three separate slopes per each level of the factor are being fit here, why?

Alternatively, this regression with interaction can also be done in R via:

```
summary(lm(RET_FT4 ~ TUITIONFEE_OUT * CONTROL, data = scorecard))
```

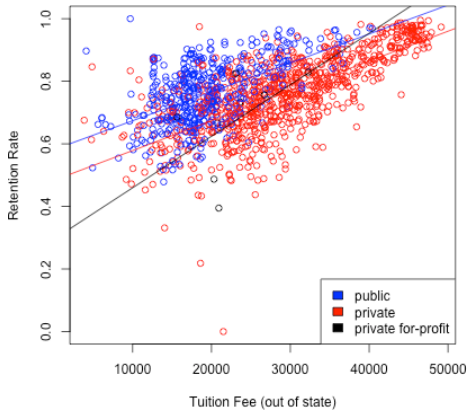
```
##
```

```
## Call:
## lm(formula = RET_FT4 ~ TUITIONFEE_OUT * CONTROL, data = scorecard)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.68822 -0.04982  0.00491  0.05555  0.32900
##
## Coefficients:
##                                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                                 5.814e-01  1.405e-02  41.372  < 2e-16
## TUITIONFEE_OUT                              9.240e-06  6.874e-07  13.441  < 2e-16
## CONTROLprivate                             -9.830e-02  1.750e-02  -5.617  2.4e-08
## CONTROLprivate for-profit                  -2.863e-01  1.568e-01  -1.826   0.0681
## TUITIONFEE_OUT:CONTROLprivate               2.988e-07  7.676e-07   0.389   0.6971
## TUITIONFEE_OUT:CONTROLprivate for-profit    7.215e-06  6.716e-06   1.074   0.2829
##
## (Intercept)                                ***
## TUITIONFEE_OUT                             ***
## CONTROLprivate                             ***
## CONTROLprivate for-profit                  .
## TUITIONFEE_OUT:CONTROLprivate
## TUITIONFEE_OUT:CONTROLprivate for-profit
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08734 on 1235 degrees of freedom
## Multiple R-squared:  0.4413, Adjusted R-squared:  0.4391
## F-statistic: 195.1 on 5 and 1235 DF,  p-value: < 2.2e-16
```

The three separate regressions can be plotted in one plot as before.

```
cols <- c("blue", "red", "black")
plot(RET_FT4 ~ TUITIONFEE_OUT, data = scorecard, xlab = "Tuition Fee (out of state)",
    ylab = "Retention Rate", col = cols[scorecard$CONTROL])
baseline <- coef(req.1)[["(Intercept)"]]
slope <- coef(req.1)[["TUITIONFEE_OUT"]]
for (ii in 1:nlevels(scorecard$CONTROL)) {
    lev <- levels(scorecard$CONTROL)[[ii]]
    if (ii == 1) {
        abline(a = baseline, b = slope, col = cols[[ii]])
    }
    else {
        abline(a = baseline + coef(req.1)[[ii + 1]],
            b = slope + coef(req.1)[[ii + 3]], col = cols[[ii]])
    }
}
```

```
legend("bottomright", levels(scorecard$CONTROL), fill = cols)
```



Interaction terms make regression equations complicated (have more variables) and also slightly harder to interpret although, in some situations, they really improve predictive power. In this particular example, note that the multiple R^2 only increased from 0.4408 to 0.4413 after adding the interaction terms. This small increase means that the interaction terms are not really adding much to the regression equation so we are better off using the previous model with no interaction terms.

To get more practice with regressions having categorical variables, let us consider the bike sharing dataset discussed above.

Let us fit a basic regression equation with `casual` (number of bikes rented by casual users hourly) as the response variable and the explanatory variables being `atemp` (normalized feeling temperature), `workingday`. For this dataset, I've already encoded the categorical variables as factors.

```
summary(bike$atemp)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.07907 0.33784 0.48673 0.47435 0.60860 0.84090
```

```
summary(bike$workingday)
```

```
##  No Yes
## 231 500
```

```
summary(bike$weathersit)
```

```
## Clear/Partly Cloudy     Light Rain/Snow               Misty
##                 463                  21                 247
```

We fit the regression equation with a different shift in the mean for each level:

```
md1 = lm(casual ~ atemp + workingday + weathersit,
    data = bike)
summary(md1)
```

```
##
## Call:
## lm(formula = casual ~ atemp + workingday + weathersit, data = bike)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1456.76  -243.97   -22.93   166.81  1907.20
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  350.31      55.11   6.357 3.63e-10 ***
## atemp                       2333.77      97.48  23.942  < 2e-16 ***
## workingdayYes               -794.11      33.95 -23.388  < 2e-16 ***
## weathersitLight Rain/Snow   -523.79      95.23  -5.500 5.26e-08 ***
## weathersitMisty             -150.79      33.75  -4.468 9.14e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 425.2 on 726 degrees of freedom
## Multiple R-squared:  0.6186, Adjusted R-squared:  0.6165
## F-statistic: 294.3 on 4 and 726 DF,  p-value: < 2.2e-16
```

Question: How are the coefficients in the above regression interpreted?

There are interactons that one can add here too. For example, I can add an interaction between `workingday` and `atemp`:

```
md3 = lm(casual ~ atemp + workingday + weathersit +
    workingday:atemp, data = bike)
summary(md3)
```

```
##
## Call:
## lm(formula = casual ~ atemp + workingday + weathersit + workingday:atemp,
##     data = bike)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1709.76  -198.09   -55.12   152.88  1953.07
##
```

```
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 -276.22      77.48  -3.565 0.000388 ***
## atemp                       3696.41     155.56  23.762  < 2e-16 ***
## workingdayYes                166.71      94.60   1.762 0.078450 .
## weathersitLight Rain/Snow   -520.78      88.48  -5.886 6.05e-09 ***
## weathersitMisty             -160.28      31.36  -5.110 4.12e-07 ***
## atemp:workingdayYes        -2052.09     190.48 -10.773  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 395.1 on 725 degrees of freedom
## Multiple R-squared:  0.6712, Adjusted R-squared:  0.6689
## F-statistic:   296 on 5 and 725 DF,  p-value: < 2.2e-16
```

Question: What is the interpretation of the coefficients now?

6.6 Inference in Multiple Regression

So far, we have learned how to fit multiple regression equations to observed data and interpret the coefficient. Inference is necessary for answering questions such as: "Is the observed relationship between the response and the explanatory variables real or is it merely caused by sampling variability?"

We will again consider both parametric models and resampling techniques for inference.

6.6.1 Parametric Models for Inference

There is a response variable y and p explanatory variables $x^{(1)}, \ldots, x^{(p)}$. The data generation model is similar to that of simple regression:

$$y = \beta_0 + \beta_1 x^{(1)} + \cdots + \beta_p x^{(p)} + e.$$

The numbers $\beta_0, \ldots, \beta_p$ are the parameters of the model and unknown.

The error e is the only random part of the model, and we make the same assumptions as in simple regression:

1. e_i are independent for each observation i
2. e_i all have the same distribution with mean 0 and variance σ^2
3. e_i follow a normal distribution

We could write this more succinctly as

$$e_i \text{ are i.i.d } N(0, \sigma^2)$$

but it's helpful to remember that these are separate assumptions, so we can talk about which are the most important.

This means that under this model,

$$y \sim N(\beta_0 + \beta_1 x^{(1)} + \cdots + \beta_p x^{(p)}, \sigma^2)$$

i.e. the observed y_i are normal and independent from each other, but each with a different mean, which depends on x_i (so the y_i are NOT i.i.d. because not identically distributed).

Estimates

The numbers $\beta_0, \ldots, \beta_p$ capture the true relationship between y and $x_1, \ldots, x_p$. Also unknown is the quantity σ^2 which is the variance of the unknown e_i. When we fit a regression equation to a dataset via $lm()$ in R, we obtain estimates $\hat{\beta}_j$ of the unknown β_j.

The residual r_i serve as natural proxies for the unknown random errors e_i. Therefore a natural estimate for the error standard deviation σ is the Residual Standard Error,

$$\hat{\sigma}^2 = \frac{1}{n-p-1} \sum r_i^2 = \frac{1}{n-p-1} RSS$$

Notice this is the same as our previous equation from simple regression, only now we are using $n-p-1$ as our correction to make the estimate unbiased.

6.6.2 Global Fit

The most obvious question is the global question: are these variables cummulatively any good in predicting y? This can be restated as, whether you could predict y just as well if didn't use any of the $x^{(j)}$ variables.

Question: If we didn't use any of the variables, what is our best "prediction" of y?

So our question can be phrased as whether our prediction that we estimated, $\hat{y}(x)$, is better than just $\overline{y}$ in predicting y.

Equivalently, we can think that our null hypothesis is

$$H_0 : \beta_j = 0, \text{for all } j$$

6.6.2.1 Parametric Test of Global Fit

The parametric test that is commonly used for assessing the global fit is a F-test. A common way to assess the fit, we have just said is either large R^2 or small $RSS = \sum_{i=1}^{n} r_i^2$.

We can also think our global test is an implicit test for comparing two possible prediction models

Model 0: No variables, just predict $\bar{y}$ for all observations

Model 1: Our linear model with all the variables

Then we could also say that we could test the global fit by comparing the RSS from model 0 (the null model), versus model 1 (the one with the variables), e.g.

$$RSS_0 - RSS_1$$

Question: This will always be positive, why?

We will actually instead change this to be a proportional increase, i.e. relative to the full model, how much increase in RSS do I get when I take out the variables:

$$\frac{RSS_0 - RSS_1}{RSS_1}$$

To make this quantity more comparable across many datasets, we are going to normalize this quantity by the number of variables in the data,

$$F = \frac{(RSS_0 - RSS_1)/p}{RSS_1/(n-p-1)}$$

Notice that the RSS_0 of our 0 model is actually the TSS. This is because

$$\hat{y}^{\text{Model 0}} = \bar{y}$$

so

$$RSS_0 = \sum_{i=1}^{n} (y_i - \hat{y}^{\text{Model 0}})^2 = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

Further,

$$RSS_1/(n-p-1) = \hat{\sigma}^2$$

So we have

$$F = \frac{(TSS - RSS)/p}{\hat{\sigma}^2}$$

All of this we can verify on our data:

```
n <- nrow(body)
p <- ncol(body) - 1
tss <- (n - 1) * var(body$BODYFAT)
rss <- sum(residuals(ft)^2)
sigma <- summary(ft)$sigma
(tss - rss)/p/sigma^2
```

```
## [1] 92.61904
```

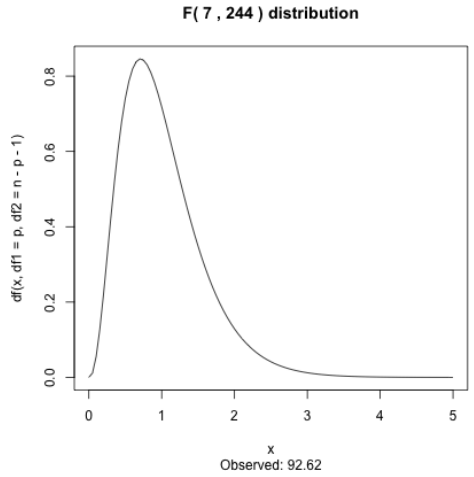
```
summary(ft)$fstatistic
```

```
##     value     numdf     dendf
##  92.61904   7.00000 244.00000
```

We do all this normalization, because under our assumptions of the parametric model, the F statistic above follows a F-distribution. The F distribution you have seen in a HW when you were simulating data, and has two parameters, the degrees of freedom of the numerator ($df1$) and the degrees of freedom of the denominator ($df2$); they are those constants we divide the numerator and denominator by in the definition of the F statistic. Then the F statistic we described above follows a $F(p, n - p - 1)$ distribution under our parametric model.

Here is the null distribution for our F statistic for the bodyfat:

```
curve(df(x, df1 = p, df2 = n - p - 1), xlim = c(0,
    5), main = paste("F(", p, ",", n - p - 1, ") distribution"),
    sub = paste("Observed:", round(summary(ft)$fstatistic["value"],
        2)))
```



This is a highly significant result, and indeed most tests of general fit are highly significant. It is rare that the entire set of variables collected have zero predictive value to the response!

6.6.2.2 Permutation test for global fit

Our null hypothesis to assess the global fit is that the x_i do not give us any information regarding the y. We had a similar situation previously when we considered comparing two groups. There, we measured a response y on two groups, and wanted to know whether the group assignment of the observation made a difference in the y response. To answer that question with permutation tests, we permuted the assignment of the y_i variables into the two groups.

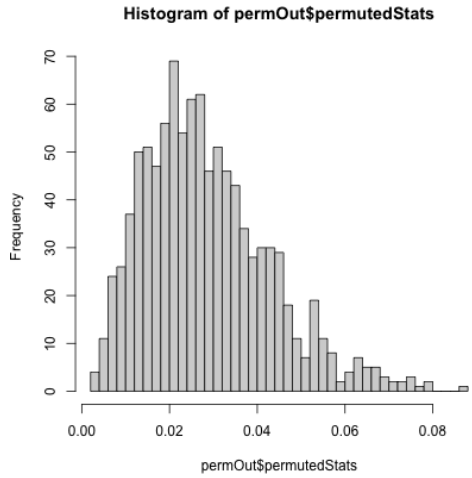
Then we can think of the global fit of the regression similarly, since under the null knowing x_i doesn't give us any information about y_i, so I can permute the assignment of the y_i to x_i and it shouldn't change the fit of our data.

Specifically, we have a statistic, R^2, for how well our predictions fit the data. We observe pairs (y_i, x_i) (x_i here is a vector of all the variables for the observation i). Then

1. Permute the order of the y_i values, so that the y_i are paired up with different x_i.
2. Fit the regression model on the permuted data
3. Calculate R_b^2
4. Repeat B times to get $R_1^2, \ldots, R_B^2$.
5. Determine the p-value of the *observed* R^2 as compared to the compute null distribution

We can do this with the body fat dataset:

```
set.seed(147980)
permutationLM <- function(y, data, n.repetitions, STAT = function(lmFit) {
    summary(lmFit)$r.squared
}) {
    stat.obs <- STAT(lm(y ~ ., data = data))
    makePermutedStats <- function() {
        sampled <- sample(y)
        fit <- lm(sampled ~ ., data = data)
        return(STAT(fit))
    }
    stat.permute <- replicate(n.repetitions, makePermutedStats())
    p.value <- sum(stat.permute >= stat.obs)/n.repetitions
    return(list(p.value = p.value, observedStat = stat.obs,
        permutedStats = stat.permute))
}
permOut <- permutationLM(body$BODYFAT, data = body[,
    -1], n.repetitions = 1000)
hist(permOut$permutedStats, breaks = 50)
```

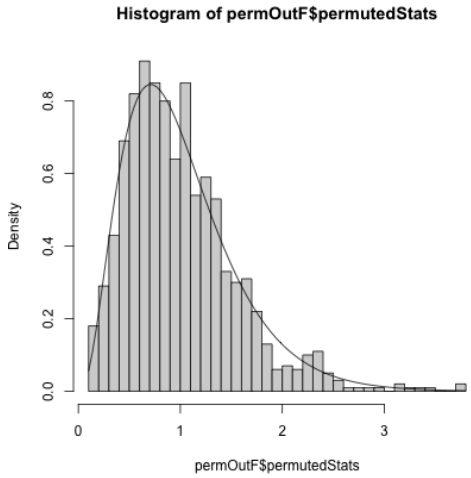



```
permOut[1:2]
```

```
## $p.value
## [1] 0
##
## $observedStat
## [1] 0.7265596
```

Notice that we could also use the F statistic from before too (here we overlay the null distribution of the F statistic from the parametric model for comparison),

```
n <- nrow(body)
p <- ncol(body) - 1
permOutF <- permutationLM(body$BODYFAT, data = body[,
    -1], n.repetitions = 1000, STAT = function(lmFit) {
    summary(lmFit)$fstatistic["value"]
})
hist(permOutF$permutedStats, freq = FALSE, breaks = 50)
curve(df(x, df1 = p, df2 = n - p - 1), add = TRUE,
    main = paste("F(", p, ",", n - p - 1, ") distribution"))
```




```
permOutF[1:2]
```

```
## $p.value
## [1] 0
##
## $observedStat
##    value
## 92.61904
```

6.6.3 Individual Variable Importance

We can also ask about individual variable, β_j. This is a problem that we have discussed in the setting of simple regression, where we are interested in inference regarding the parameter β_j, either with confidence intervals of β_j or the null hypothesis:

$$H_0 : \beta_j = 0$$

In order to perform inference for β_j, we have two possibilities of how to perform inference, like in simple regression: bootstrap CI and the parametric model.

6.6.3.1 Bootstrap for CI of $\hat{\beta}_j$

Performing the bootstrap to get CI for $\hat{\beta}_j$ in multiple regression is the exact same procedure as in simple regression.

Specifically, we still bootstrap pairs (y_i, x_i) and each time recalculate the linear model. For each β_j, we will have a distribution of $\hat{\beta}_j^*$ for which we can perform confidence intervals.

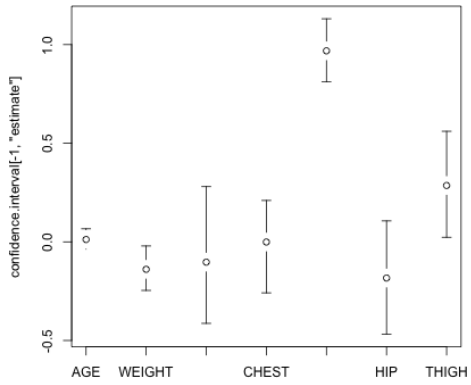
We can even use the same function as we used in the simple regression setting with little changed.

```
bootstrapLM <- function(y, x, repetitions, confidence.level = 0.95) {
    stat.obs <- coef(lm(y ~ ., data = x))
    bootFun <- function() {
        sampled <- sample(1:length(y), size = length(y),
            replace = TRUE)
        coef(lm(y[sampled] ~ ., data = x[sampled, ]))
    }
    stat.boot <- replicate(repetitions, bootFun())
    level <- 1 - confidence.level
    confidence.interval <- apply(stat.boot, 1, quantile,
        probs = c(level/2, 1 - level/2))
    return(list(confidence.interval = cbind(lower = confidence.interval[1,
        ], estimate = stat.obs, upper = confidence.interval[2,
        ]), bootStats = stat.boot))
}
```

```
bodyBoot <- with(body, bootstrapLM(y = BODYFAT, x = body[,
    -1], repetitions = 10000))
bodyBoot$conf
```

```
##                    lower      estimate       upper
## (Intercept) -75.68776383 -3.747573e+01 -3.84419402
## AGE          -0.03722018  1.201695e-02  0.06645578
## WEIGHT       -0.24629552 -1.392006e-01 -0.02076377
## HEIGHT       -0.41327145 -1.028485e-01  0.28042319
## CHEST        -0.25876131 -8.311678e-04  0.20995486
## ABDOMEN       0.81115069  9.684620e-01  1.13081481
## HIP          -0.46808557 -1.833599e-01  0.10637834
## THIGH         0.02272414  2.857227e-01  0.56054626
```

```
require(gplots)
with(bodyBoot, plotCI(confidence.interval[-1, "estimate"],
    ui = confidence.interval[-1, "upper"], li = confidence.interval[-1,
        "lower"], xaxt = "n"))
axis(side = 1, at = 1:(nrow(bodyBoot$conf) - 1), rownames(bodyBoot$conf)[-1])
```



Note, that unless I scale the variables, I can't directly interpret the size of the β_j as its importance (see commentary above under interpretation).

Assumptions of the Bootstrap

Recall that the bootstrap has assumptions, two important ones being that we have independent observations and the other being that we can reasonably estimate F with $\hat{F}$. However, the distribution F we need to estimate is not the distribution of an individual a single variable, but the entire *joint* distributions of all the variables. This gets to be a harder and harder task for larger numbers of variables (i.e. for larger p).

In particular, when using the bootstrap in multiple regression, it will not perform well if p is large relative to n.[4] In general you want the ratio p/n to be small (like less than 0.1); otherwise the bootstrap can give very poor CI.[5]

```
cat("Ratio of p/n in body fat: ", ncol(body)/nrow(body),
    "\n")
```

```
## Ratio of p/n in body fat:  0.03174603
```

6.6.3.2 Parametric models

Again, our inference on β_j will look very similar to simple regression. Using our parametric assumptions about the distribution of the errors will mean that each $\hat{\beta}_j$ is normally distributed [6]

$$\hat{\beta}_j \sim N(\beta_j, \nu_j^2)$$

where

$$\nu_j^2 = \ell(X)\sigma^2$$

[4] Of course, you cannot do regression *at all* unless $n > p$.

[5] The CI will tend to be *very* conservative...too wide to give meaningful inference

[6] again, the equation for $\hat{\beta}_j$ will be a linear combination of the y_i, and linear combinations of normal R.V. are normal, even if the R.V. are not independent.

($\ell(X)$ is a linear combination of all of the observed explanatory variables, given in the matrix X).[7]

Using this, we create t-statistics for each β_j by standardizing $\hat{\beta}_j$

$$T_j = \frac{\hat{\beta}_j}{\sqrt{\hat{var}(\hat{\beta}_j)}}$$

Just like the t-test, T_j should be normally distributed[8] This is exactly what `lm` gives us:

```
summary(ft)$coef
```

```
##                  Estimate  Std. Error      t value     Pr(>|t|)
## (Intercept) -3.747573e+01 14.49480190 -2.585460204 1.030609e-02
## AGE          1.201695e-02  0.02933802  0.409603415 6.824562e-01
## WEIGHT      -1.392006e-01  0.04508534 -3.087490946 2.251838e-03
## HEIGHT      -1.028485e-01  0.09787473 -1.050817489 2.943820e-01
## CHEST       -8.311678e-04  0.09988554 -0.008321202 9.933675e-01
## ABDOMEN      9.684620e-01  0.08530838 11.352484708 2.920768e-24
## HIP         -1.833599e-01  0.14475772 -1.266667813 2.064819e-01
## THIGH        2.857227e-01  0.13618546  2.098041564 3.693019e-02
```

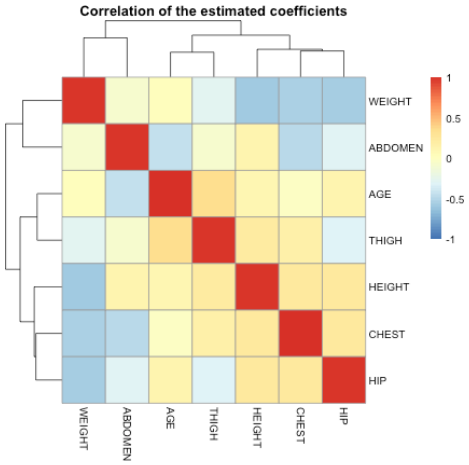
Correlation of estimates

The estimated $\hat{\beta}_j$ are themselves correlated with each other, unless the x^j and x^k variables are uncorrelated.

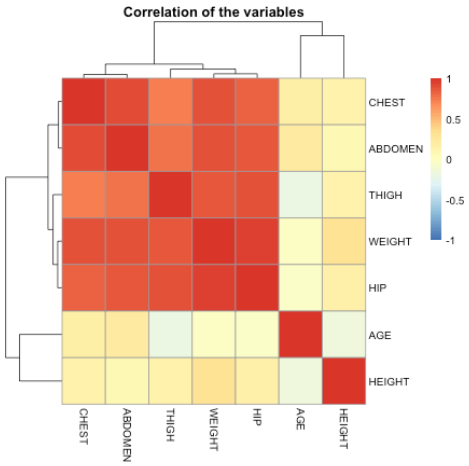
```
library(pheatmap)
pheatmap(summary(ft, correlation = TRUE)$corr[-1, -1],
    breaks = seq(-1, 1, length = 100), main = "Correlation of the estimated coefficients")
```

[7] Specifically, the vector of estimates of the β_j is given by $\hat{\beta} = (X'X)^{-1}Xy$ (a $p+1$ length vector) and the covariance matrix of the estimates $\hat{\beta}$ is given by $(X'X)^{-1}\sigma^2$

[8] with the same caveat, that when you estimate the variance, you affect the distribution of T_j, which matters in small sample sizes.




```
pheatmap(cor(body[, -1]), breaks = seq(-1, 1, length = 100),
    main = "Correlation of the variables")
```



6.6.4 Inference on $\hat{y}(x)$

We can also create confidence intervals on the prediction given by the model, $\hat{y}(x)$. For example, suppose now that we are asked to predict the bodyfat percentage of an individual who has a particular set of variables x_0. Then the same logic in simple regression follows here.

There are two intervals associated with prediction:

1. Confidence intervals for the **average** response, i.e. bodyfat percentage for **all individuals** who have the values x_0. The average (or expected values) at x_0 is
$$E(y(x_0)) = \beta_0 + \beta_1 x_0^{(1)} + ... + \beta_p x_0^{(p)}.$$

and so we estimate it using our estimates of β_j, getting $\hat{y}(x_0)$.

Then our $1-\alpha$ confidence interval will be[9]

$$\hat{y}(x_0) \pm t_{\alpha_2}\sqrt{\hat{var}(\hat{y}(x_0))}$$

2. Confidence intervals for a particular individual (**prediction interval**). If we knew β completely, we still wouldn't know the value of the particular individual. But if we knew β, we know that our parametric model says that all individuals with the same x_0 values are normally distributed as

$$N(\beta_0 + \beta_1 x_0^{(1)} + ... + \beta_p x_0^{(p)}, \sigma^2)$$

Question: So we could give an interval that we would expect 95% confidence that such an individual would be in, how?

We don't know β, so actually we have to estimate both parts of this,

$$\hat{y}(x_0) + \pm 1.96\sqrt{\hat{\sigma}^2 + \hat{var}(\hat{y}(x_0))}$$

Both of these intervals are obtained in R via the *predict* function.

```
x0 = data.frame(AGE = 30, WEIGHT = 180, HEIGHT = 70,
    CHEST = 95, ABDOMEN = 90, HIP = 100, THIGH = 60)
predict(ft, x0, interval = "confidence")
```

```
##        fit      lwr      upr
## 1 16.51927 15.20692 17.83162
```

```
predict(ft, x0, interval = "prediction")
```

```
##        fit      lwr      upr
## 1 16.51927 7.678715 25.35983
```

Note that the prediction interval is much wider compared to the confidence interval for average response.

% % ## Regression Diagnostics

Our next topic in multiple regression is regression diagnostics. The inference procedures that we talked about work under the assumptions of the linear regression model. If these assumptions are violated, then our hypothesis tests, standard errors and confidence intervals will be violated. Regression diagnostics enable us to diagnose if the model assumptions are violated or not.

The key assumptions we can check for in the regression model are:

[9]For those familiar with linear algebra, $\hat{var}(\hat{y}(x_0) = x_0^T(X^TX)^{-1}x_0\sigma^2$

1. Linearity: the mean of the y is linearly related to the explanatory variables.
2. Homoscedasticity: the errors have the same variance.
3. Normality: the errors have the normal distribution.
4. All the observations obey the same model (i.e., there are no outliers or exceptional observations).

These are particularly problems for the parametric model; the bootstrap will be relatively robust to these assumptions, but violations of these assumptions can cause the inference to be less powerful – i.e. harder to detect interesting signal.

These above assumptions can be checked by essentially looking at the residuals:

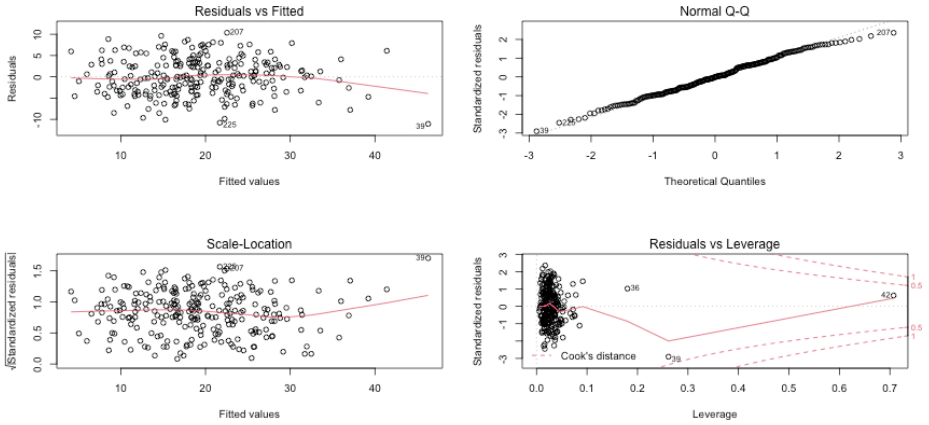
1. **Linearity**: The residuals represent what is left in the response variable after the linear effects of the explanatory variables are taken out. So if there is a non-linear relationship between the response and one or more of the explanatory variables, the residuals will be related non-linearly to the explanatory variables. This can be detected by plotting the residuals against the explanatory variables. It is also common to plot the residuals against the fitted values. Note that one can also detect non-linearity by simply plotting the response against each of the explanatory variables.
2. **Homoscedasticity**: Heteroscedasticity can be checked again by plotting the residuals against the explanatory variables and the fitted values. It is common here to plot the absolute values of the residuals or the square root of the absolute values of the residuals.
3. **Normality**: Detected by the normal Q-Q plot of the residuals.
4. **Outliers**: The concern with outliers is that they could be effecting the fit. There are three measurements we could use to consider whether a point is an outlier
   - Size of the residuals (r_i) – diagnostics often use standardized residuals to make them more comparable between different observations[10]
   - Leverage – a measure of how far the vector of explanatory variables of an observation are from the rest, and on average are expected to be about p/n.
   - Cook's Distance – how much the coefficients $\hat{\beta}$ will change if you leave out observation i, which basically combines the residual and the leverage of a point.

Outliers typically will have either large (in absolute value) residuals and/or large leverage.

Consider the bodyfat dataset. A simple way for doing some of the standard regression diagnostics is to use the `plot` command as applied to the linear model fit:

```
par(mfrow = c(2, 2))
plot(ft)
```

[10] in fact r_i is not a good estimate of e_i, in terms of not having constant variance and being correlated. Standardized residuals are still correlated, but at least have the same variance




```
par(mfrow = c(1, 1))
```

Let's go through these plots and what we can look for in these plots. There can sometimes be multiple issues that we can detect in a single plot.

Independence

Note that the most important assumption is independence. Violations of independence will cause problems for every inference procedure we have looked at, including the resampling procedures, and the problems such a violation will cause for your inference will be even worse than the problems listed above. Unfortunately, violations of independence are difficult to check for in a generic problem. If you suspect a certain kind of dependence, e.g. due to time or geographical proximity, there are special tools that can be used to check for that. But if you don't have a candidate for what might be the source of the dependence, the only way to know there is no dependence is to have close control over how the data was collected.

6.6.5 Residuals vs. Fitted Plot

The first plot is the residuals plotted against the fitted values. The points should look like a random scatter with no discernible pattern. We are often looking for two possible violations:

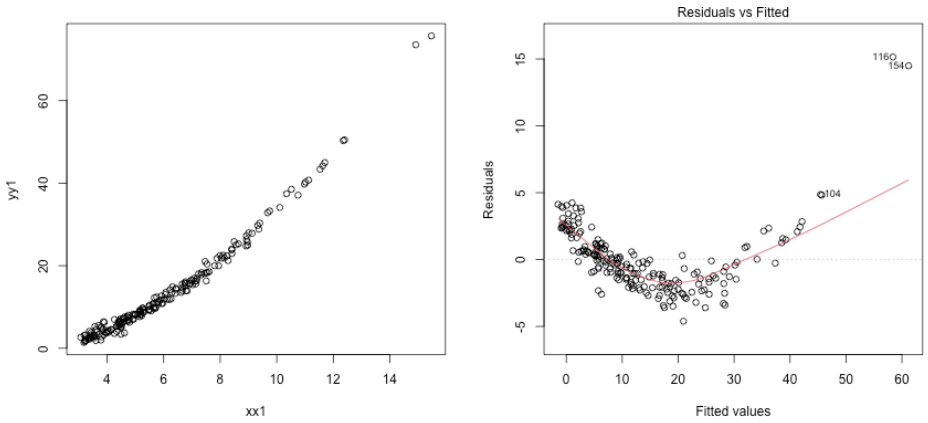
1. Non-linear relationship to response, detected by a pattern in the mean of the residuals. Recall that the correlation between $\hat{y}$ and the residuals must be numerically zero – but that doesn't mean that there can't be *non-linear* relationships.
2. Heteroscedasticity – a pattern in the variability of the residuals, for example higher variance in observations with large fitted values.

Let us now look at some simulation examples in the simple setting of a single predictor to demonstrate these phenomena.

Example: Non-linearity

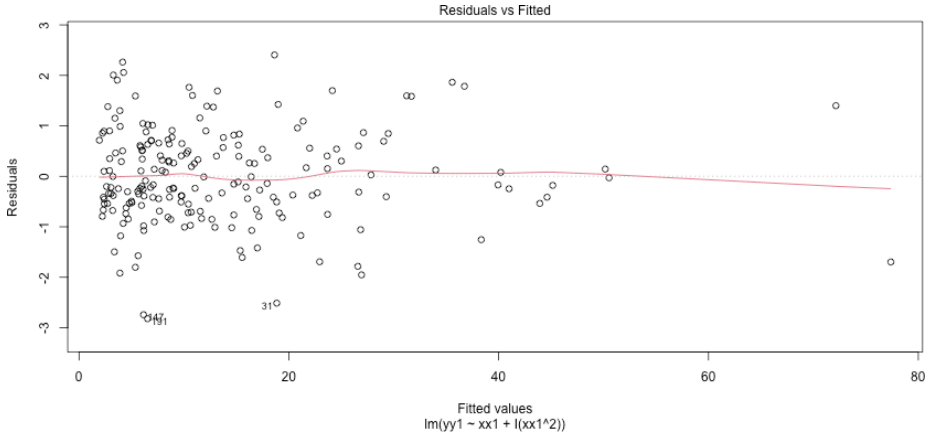
In the next example, the response is related non-linearly to x.

```
n = 200
xx1 = 3 + 4 * abs(rnorm(n))
yy1 = -2 + 0.5 * xx1^(1.85) + rnorm(n)
m1 = lm(yy1 ~ xx1)
par(mfrow = c(1, 2))
plot(yy1 ~ xx1)
plot(m1, which = 1)
```



Non-linearity is fixed by adding non-linear functions of explanatory variables as additional explanatory variables. In this example, for instance, we can add x^2 as an additional explanatory variable.

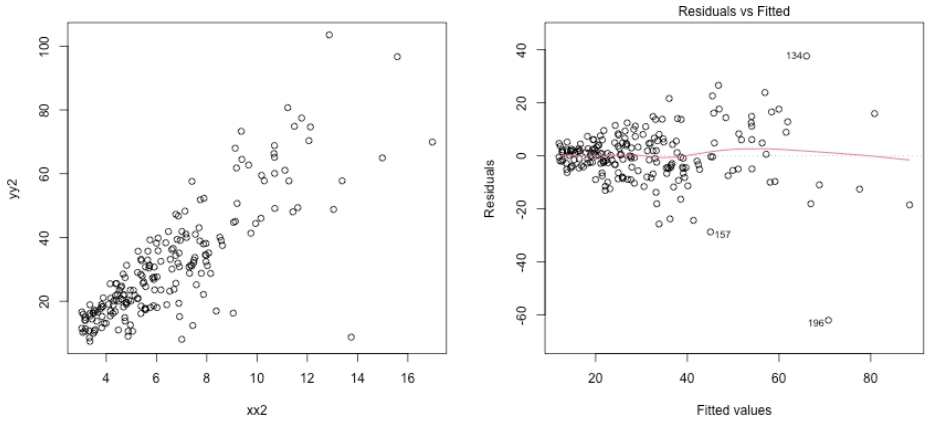
```
par(mfrow = c(1, 1))
m1.2 = lm(yy1 ~ xx1 + I(xx1^2))
plot(m1.2, which = 1)
```



Example: Heteroscedasticity

Next let us consider an example involving heterscedasticity (unequal variance).

```
set.seed(478912)
n = 200
xx2 = 3 + 4 * abs(rnorm(n))
yy2 = -2 + 5 * xx2 + 0.5 * (xx2^(1.5)) * rnorm(n)
m2 = lm(yy2 ~ xx2)
par(mfrow = c(1, 2))
plot(yy2 ~ xx2)
plot(m2, which = 1)
```



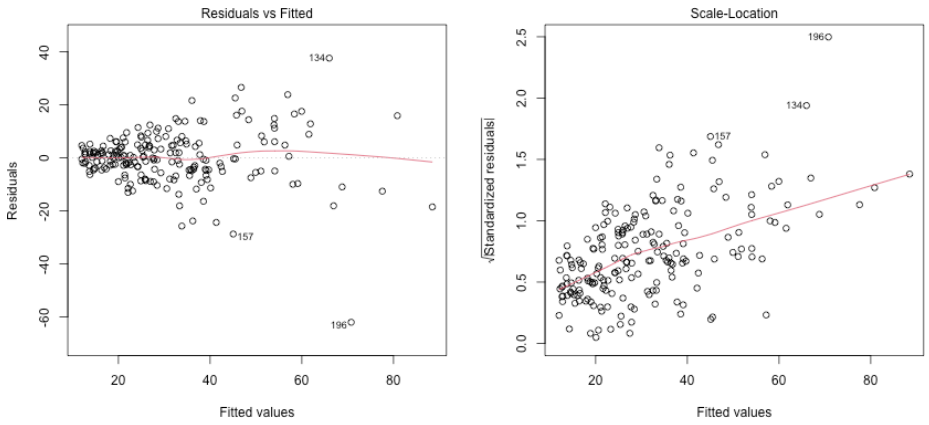
Notice that even with a single variable, it is easier to see the difference in variability with the residuals than in plotting y versus x (in the plot of y versus x, the fact that y is growing with x makes it harder to be sure).

Heteroscedasticity is a little tricky to handle in general. Heteroscedasiticity can

sometimes be fixed by applying a transformation to the response variable (y) before fitting the regression. For example, if all the response values are positive, taking the logarithm or square root of the response variable is a common solution.

The Scale-Location plot (which is one of the default plots of `plot`) is also useful for detecting heteroscedasiticity. It plots the square root of the absolute value of the residuals (actually standardized residuals but these are similar to the residuals) against the fitted values. Any increasing or decreasing pattern in this plot indicates heteroscedasticity. Here is that plot on the simulated data that has increasing variance:

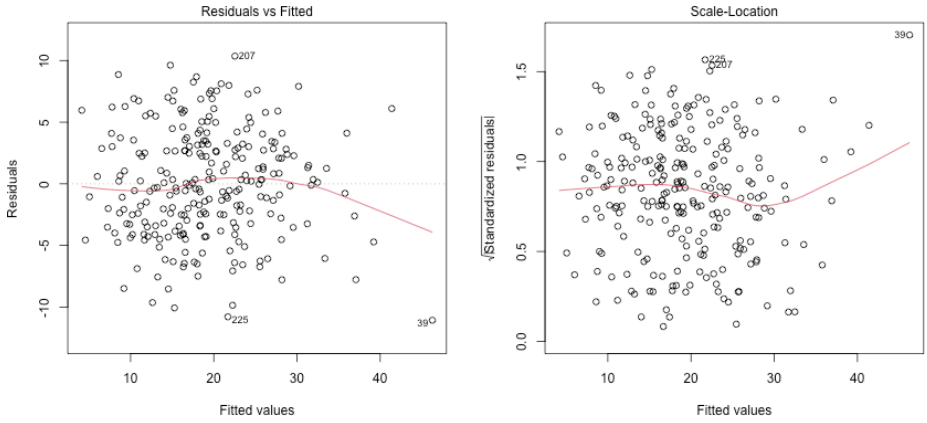
```
par(mfrow = c(1, 2))
plot(m2, which = c(1, 3))
```



Back to data

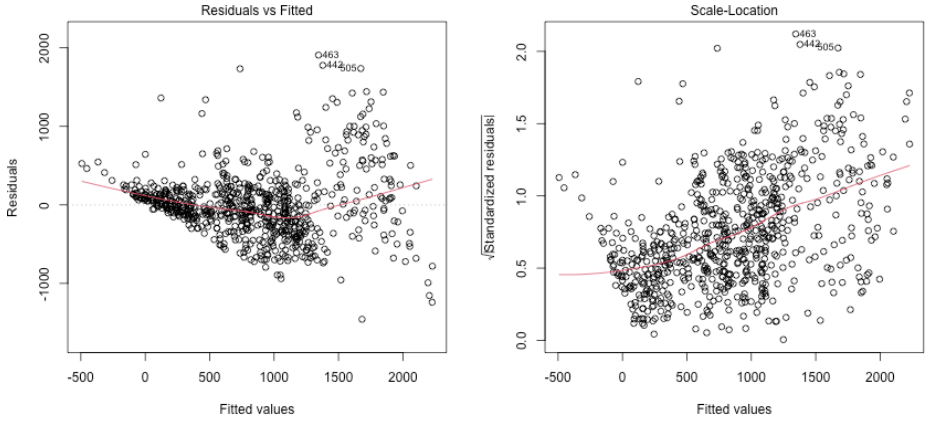
We don't see any obvious pattern in the fitted versus residual plot.

```
par(mfrow = c(1, 2))
plot(ft, which = c(1, 3))
```



We do the same plot from our bike regression from above:

```
md1 = lm(casual ~ atemp + workingday + weathersit,
    data = bike)
par(mfrow = c(1, 2))
plot(md1, which = c(1, 3))
```

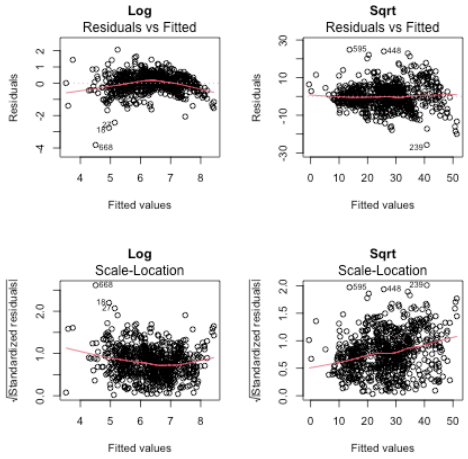


Here we see serious heteroskedasticity, where there is more variability in our residuals for larger fitted values than for smaller ones. There's also possibly signs that our residuals have a pattern to them (not centered at zero), possibly indicating that our linear fit is not appropriate.

The response here is counts (number of casual users) and it is common to transform such data. Here we show the fitted/residual plot after transforming the response by the log and square-root:

```
mdLog = lm(log(casual) ~ atemp + workingday + weathersit,
    data = bike)
mdSqrt = lm(sqrt(casual) ~ atemp + workingday + weathersit,
```

```
    data = bike)
par(mfrow = c(2, 2))
plot(mdLog, which = 1, main = "Log")
plot(mdSqrt, which = 1, main = "Sqrt")
plot(mdLog, which = 3, main = "Log")
plot(mdSqrt, which = 3, main = "Sqrt")
```



Why plot against $\hat{y}$?

If we think there is a non linear relationship, shouldn't we plot against the individual $x^{(j)}$ variables? We certainly can! Just like with $\hat{y}$, each $x^{(j)}$ is uncorrelated with the residuals, but there can be non-linear relationships that show up. Basically any plot we do of the residuals should look like a random cloud of points with no pattern, including against the explanatory variables.

Plotting against the individual $x^{(j)}$ can help to determine *which* variables have a non-linear relationship, and can help in determining an alternative model. Of course this is only feasible with a relatively small number of variables.

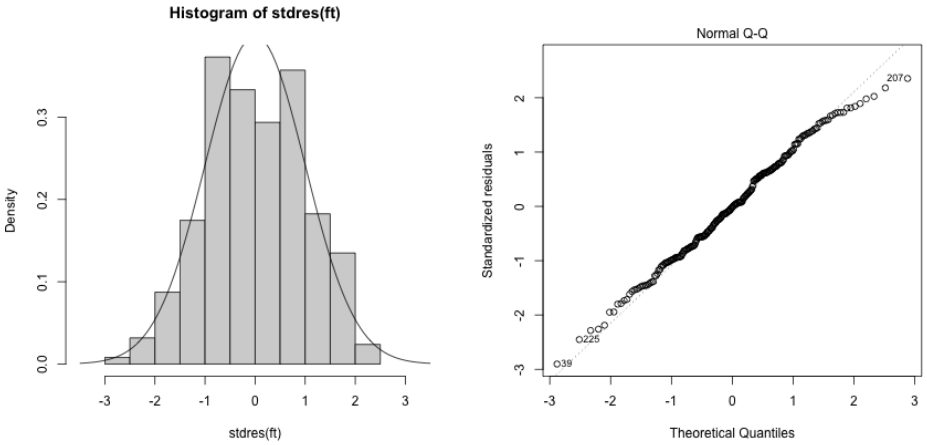
One reason that $\hat{y}$ is our default plot is that 1) there are often too many variables to plot against easily; and 2) there are many common examples where the variance changes as a function of the size of the response, e.g. more variance for larger y values.

6.6.6 QQ-Plot

The second plot is the normal Q-Q plot of the standardized residuals. If the normal assumption holds, then the points should be along the line here.

```
library(MASS)
par(mfrow = c(1, 2))
```

```
hist(stdres(ft), freq = FALSE, breaks = 10, xlim = c(-3.5,
    3.5))
curve(dnorm, add = TRUE)
plot(ft, which = 2)
```



A QQ-plot is based on the idea that every point in your dataset is a quantile. Specifically, if you have data $x_1, \ldots, x_n$ and you assume they are all in order, then the probability of finding a data point less than or equal to x_1 is $1/n$ (assuming there are no ties). So x_1 is the $1/n$ quantile of the observed data distribution. x_2 is the $2/n$ quantile, and so forth.[11]

```
quantile(stdres(ft), 1/nrow(body))
```

```
## 0.3968254%
##  -2.453687
```

Under our assumption of normality, then we also know what the $1/n$ quantile *should* be based on `qnorm` (the standardized residuals are such that we expect them to be $N(0, 1)$)

```
qnorm(1/nrow(body))
```

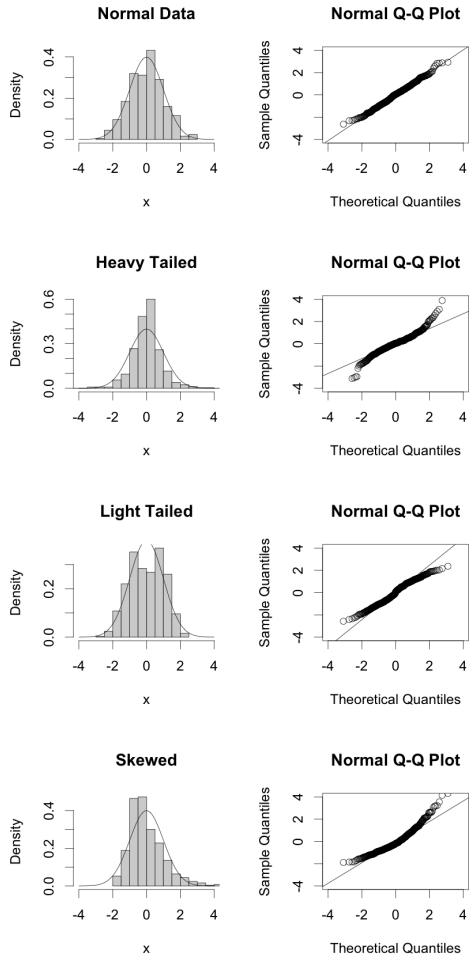
```
## [1] -2.654759
```

The idea with QQ-plots is that we can do this for all of the data, and compare whether our data has quantiles that match what we would expect for a normal distribution.

Here are some examples of QQ-plots for some simulated data, to give you a sense of how QQ-plots correspond to distributional properties:

[11] Actually, we estimate quantiles from data (called **empirical quantiles**), in a slightly more complex way that performs better, but this is the idea.

```
par(mfrow = c(4, 2), cex = 2)
n <- 500
qqlim <- c(-4, 4)
set.seed(302)
x <- rnorm(n)
x <- scale(x, scale = TRUE)
hist(x, freq = FALSE, breaks = 10, xlim = c(-4, 4),
    main = "Normal Data")
curve(dnorm, add = TRUE)
qqnorm(x, xlim = qqlim, ylim = qqlim)
qqline(x)
x <- rt(n, df = 3)
x <- scale(x, scale = TRUE)
hist(x, freq = FALSE, breaks = 30, xlim = c(-4, 4),
    main = "Heavy Tailed")
curve(dnorm, add = TRUE)
qqnorm(x, xlim = qqlim, ylim = qqlim)
qqline(x)
x <- rnorm(n, 0, 1)
x <- scale(sign(x) * abs(x)^{
    3/4
}, scale = TRUE)
hist(x, freq = FALSE, breaks = 10, xlim = c(-4, 4),
    main = "Light Tailed")
curve(dnorm, add = TRUE)
qqnorm(x, xlim = qqlim, ylim = qqlim)
qqline(x)
x <- rgamma(n, 5, 1)
x <- scale(x, scale = TRUE)
hist(x, freq = FALSE, breaks = 10, xlim = c(-4, 4),
    main = "Skewed")
curve(dnorm, add = TRUE)
qqnorm(x, xlim = qqlim, ylim = qqlim)
qqline(x)
```

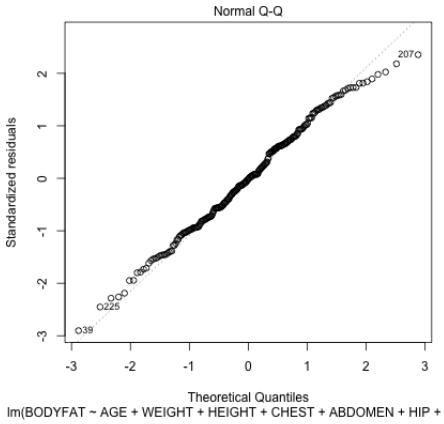


Back to body fat data

There are some signs in the right tail that the residuals are a little off normal.

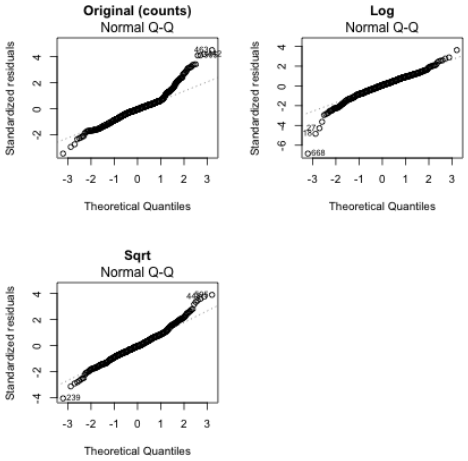
Question: Would you say that they are heavy or light tailed?

```
par(mfrow = c(1, 1))
plot(ft, which = 2)
```



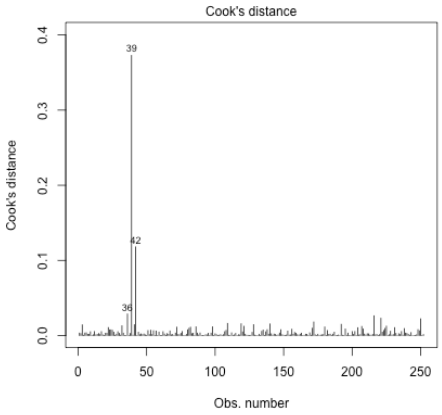
Looking at the bike model, we see the QQ plot shows serious problems in the residuals as well. We see that taking a transformation of the response not only helped with the heteroskedasticity, but also makes the residuals look closer to normal. This is not uncommon, that what helps create more constant variance can help the distributional assumptions as well.

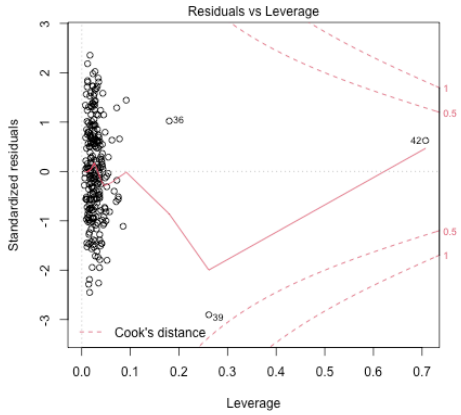
```
par(mfrow = c(2, 2))
plot(md1, which = 2, main = "Original (counts)")
plot(mdLog, which = 2, main = "Log")
plot(mdSqrt, which = 2, main = "Sqrt")
```



6.6.7 Detecting outliers

The final plots are used for detecting outliers and other exceptional observations. Large leverage or large residuals can indicate potential outliers, as does cooks distance, which is a combination of the two. The default plots give the index of potential outliers to help identify them.





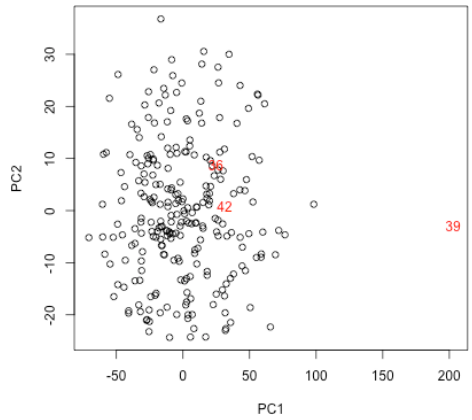
Three points flagged here are observations $i = 39, 42, 36$. Let us look at these observations separately, as well as plot some our visualizations highlighting these points:

```
## High leverage points:
```

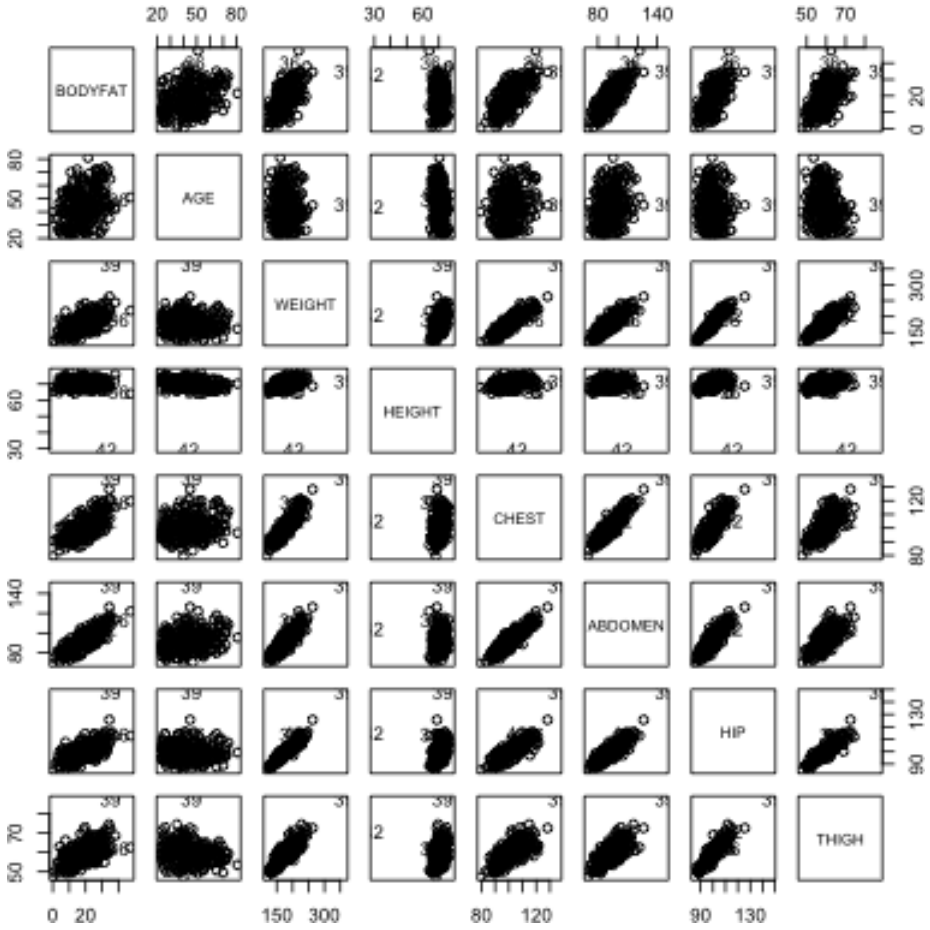
```
##    BODYFAT AGE WEIGHT HEIGHT CHEST ABDOMEN   HIP THIGH
## 39    35.2  46 363.15  72.25 136.2   148.1 147.7  87.3
## 42    32.9  44 205.00  29.50 106.0   104.3 115.5  70.6
## 36    40.1  49 191.75  65.00 118.5   113.1 113.8  61.9
```

```
## Mean of each variables:
```

```
##   BODYFAT       AGE    WEIGHT    HEIGHT     CHEST   ABDOMEN       HIP     THIGH
##  19.15079  44.88492 178.92440  70.14881 100.82421  92.55595  99.90476  59.40595
```




```
pairs(body, panel = function(x, y) {
    points(x[-whOut], y[-whOut])
    text(x[whOut], y[whOut], labels = whOut)
})
```



The observation 39 is certainly an outlier in many variables. Observation 42 seems to have an erroneous height recording. Observation 36 seems to have a high value for the response (percent bodyfat).

When outliers are detected, one can perform the regression analysis after dropping the outlying observations and evaluate their impact. After this, one needs to decide whether to report the analysis with the outliers or without them.

```
## Coefficients without outliers:

##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -22.902     20.297  -1.128    0.260
## AGE            0.021      0.029   0.717    0.474
## WEIGHT        -0.074      0.059  -1.271    0.205
## HEIGHT        -0.241      0.187  -1.288    0.199
## CHEST         -0.121      0.113  -1.065    0.288
## ABDOMEN        0.945      0.088  10.709    0.000
```

```
## HIP           -0.171      0.152  -1.124    0.262
## THIGH          0.223      0.141   1.584    0.114

##
## Coefficients in Original Model:

##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -37.476     14.495  -2.585    0.010
## AGE            0.012      0.029   0.410    0.682
## WEIGHT        -0.139      0.045  -3.087    0.002
## HEIGHT        -0.103      0.098  -1.051    0.294
## CHEST         -0.001      0.100  -0.008    0.993
## ABDOMEN        0.968      0.085  11.352    0.000
## HIP           -0.183      0.145  -1.267    0.206
## THIGH          0.286      0.136   2.098    0.037
```

We can see that WEIGHT and THIGH are no longer significant after removing these outlying points. We should note that removing observations reduces the power of all tests, so you may often see less significance if you remove many points (three is not really many!). But we can compare to removing three random points, and see that we don't have major changes in our results:

```
## Coefficients without three random points:

##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -36.732     14.620  -2.513    0.013
## AGE            0.008      0.030   0.287    0.774
## WEIGHT        -0.139      0.045  -3.070    0.002
## HEIGHT        -0.108      0.098  -1.094    0.275
## CHEST          0.002      0.100   0.016    0.987
## ABDOMEN        0.972      0.086  11.351    0.000
## HIP           -0.182      0.145  -1.249    0.213
## THIGH          0.266      0.136   1.953    0.052
```

6.7 Variable Selection

Consider a regression problem with a response variable y and p explanatory variables $x_1, \ldots, x_p$. Should we just go ahead and fit a linear model to y with all the p explanatory variables or should we throw out some unnecessary explanatory variables and then fit a linear model for y based on the remaining variables? One often does the latter in practice. The process of selecting important explanatory variables to include in a regression model is called variable selection. The following are reasons for performing variable selection:

1. Removing unnecessary variables results in a simpler model. Simpler models are always preferred to complicated models.

2. Unnecessary explanatory variables will add noise to the estimation of quantities that we are interested in.
3. Collinearity (i.e. strong linear relationships in the variables) is a problem with having too many variables trying to do the same job.
4. We can save time and/or money by not measuring redundant explanatory variables.

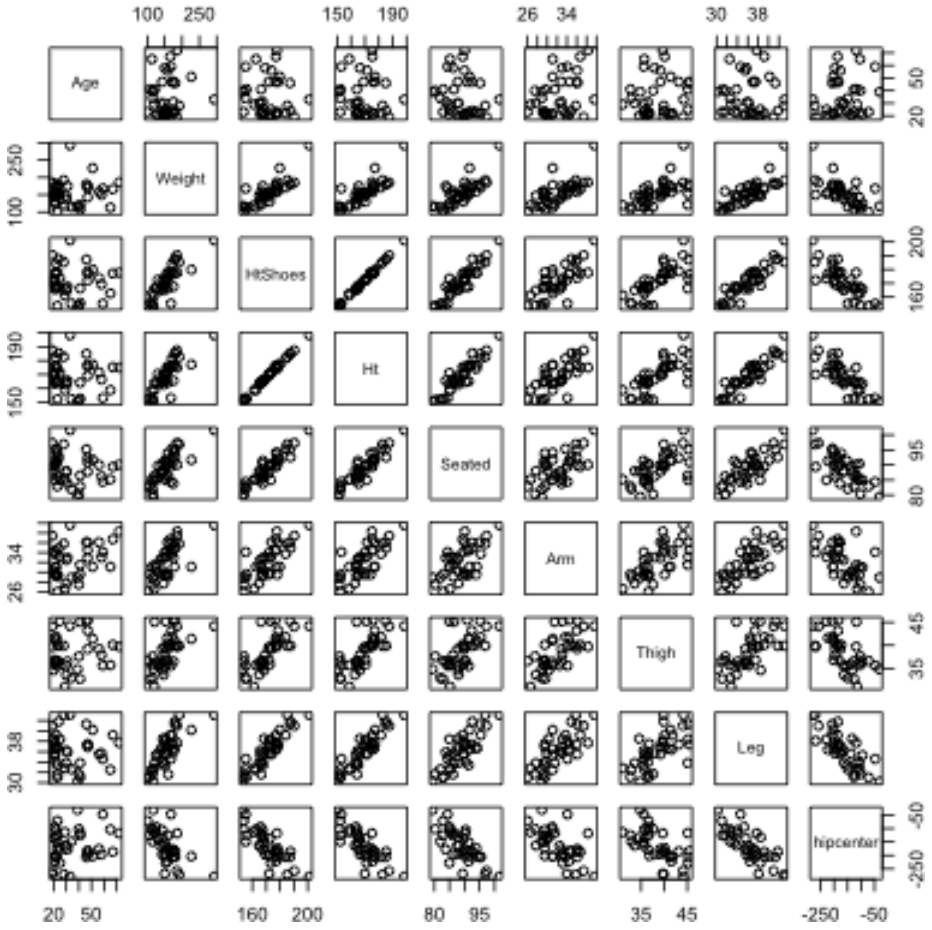
Several common, interrelated strategies for asking this question

1. Hypothesis testing on variables or submodels
2. Stepwise regression based on p-values
3. Criteria based Variable Selection

We shall illustrate variable selection procedures using the following dataset (which is available in R from the `faraway` package). This small dataset gives information about drivers and the seat position that they choose, with the idea of trying to predict a seat position from information regarding the driver (age, weight, height,...).

We can see that the variables are highly correlated with each other, and no variables are significant. However, the overall p-value reported for the F-statistic in the summary is almost zero (this is an example of how you might actually find the F statistic useful, in that it provides a check that even though no single variable is significant, the variables jointly do fit the data well)

```
library(faraway)
data(seatpos)
pairs(seatpos)
```




```
lmSeat = lm(hipcenter ~ ., seatpos)
summary(lmSeat)
```

```
##
## Call:
## lm(formula = hipcenter ~ ., data = seatpos)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -73.827 -22.833  -3.678  25.017  62.337
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 436.43213  166.57162   2.620   0.0138 *
## Age           0.77572    0.57033   1.360   0.1843
## Weight        0.02631    0.33097   0.080   0.9372
```

```
## HtShoes      -2.69241    9.75304  -0.276   0.7845
## Ht            0.60134   10.12987   0.059   0.9531
## Seated        0.53375    3.76189   0.142   0.8882
## Arm          -1.32807    3.90020  -0.341   0.7359
## Thigh        -1.14312    2.66002  -0.430   0.6706
## Leg          -6.43905    4.71386  -1.366   0.1824
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 37.72 on 29 degrees of freedom
## Multiple R-squared:  0.6866, Adjusted R-squared:  0.6001
## F-statistic:  7.94 on 8 and 29 DF,  p-value: 1.306e-05
```

6.7.1 Submodels and Hypothesis testing

We already saw that we can evaluate if we need *any* of the variables by setting up two models

Model 0: No variables, just predict $\bar{y}$ for all observations

Model 1: Our linear model with all the variables

Then we compare the RSS from these two models with the F-statistic,

$$F = \frac{(RSS_0 - RSS_1)/p}{RSS_1/(n-p-1)}$$

which the null hypothesis that these two models are equivalent (and assuming our parametric model) has a F distribution

$$H_0 : F \sim F(p, n-p-1)$$

We can expand this framework to compare any submodel to the full model, where a submodel means using only a specific subset of the p parameters. For example, can we use a model with only ABDOMEN, AGE, and WEIGHT?

For convenience lets say we number our variables so we have the first q variables are our submodel ($q = 3$ in our example). Then we now have two models:

Model 0: Just the first q variables (and the intercept) Model 1: Our linear model with all the p variables

We can do the same as before and calculate our RSS for each model and compare them. We can get a F statistic,

$$F = \frac{(RSS_0 - RSS_1)/(p-q)}{RSS_1/(n-p-1)}$$

and under the null hypothesis that the two models are equivalent,

$$H_0 : F \sim F(p-q, n-p-1)$$

Question: What does it mean if I get a non-significant result?

We can do this in R by fitting our two models, and running on the function `anova` on both models:

```
mod0 <- lm(BODYFAT ~ ABDOMEN + AGE + WEIGHT, data = body)
anova(mod0, ft)
```

```
## Analysis of Variance Table
##
## Model 1: BODYFAT ~ ABDOMEN + AGE + WEIGHT
## Model 2: BODYFAT ~ AGE + WEIGHT + HEIGHT + CHEST + ABDOMEN + HIP + THIGH
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    248 4941.3
## 2    244 4806.8  4     134.5 1.7069 0.1491
```

Question: What conclusion do we draw?

F-test is only valid for comparing submodels It is important to realize that the F test described here is only valid for comparing submodels, i.e. the smaller model has to be a set of variables that are a subset of the full model. You can't compare disjoint sets of variables with an F-test.

Single variable: test for β_j:

We could set up the following two models:

Model 0: All of the variables *except* for β_j Model 1: Our linear model with all the p variables

This is equivalent to

$$H_0 : \beta_j = 0$$

Question: How would you calculate the F statistic and null distribution of the F Statistic?

Here we run that leaving out just `HEIGHT`:

```
modNoHEIGHT <- lm(BODYFAT ~ ABDOMEN + AGE + WEIGHT +
    CHEST + HIP + THIGH, data = body)
anova(modNoHEIGHT, ft)
```

```
## Analysis of Variance Table
##
```

```
## Model 1: BODYFAT ~ ABDOMEN + AGE + WEIGHT + CHEST + HIP + THIGH
## Model 2: BODYFAT ~ AGE + WEIGHT + HEIGHT + CHEST + ABDOMEN + HIP + THIGH
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    245 4828.6
## 2    244 4806.8  1    21.753 1.1042 0.2944
```

In fact if we compare that with the inference from our standard t-test of $\beta_j = 0$, we see we get the same answer

```
summary(ft)
```

```
##
## Call:
## lm(formula = BODYFAT ~ AGE + WEIGHT + HEIGHT + CHEST + ABDOMEN +
##     HIP + THIGH, data = body)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.0729  -3.2387  -0.0782   3.0623  10.3611
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.748e+01  1.449e+01  -2.585  0.01031 *
## AGE          1.202e-02  2.934e-02   0.410  0.68246
## WEIGHT      -1.392e-01  4.509e-02  -3.087  0.00225 **
## HEIGHT      -1.028e-01  9.787e-02  -1.051  0.29438
## CHEST       -8.312e-04  9.989e-02  -0.008  0.99337
## ABDOMEN      9.685e-01  8.531e-02  11.352  < 2e-16 ***
## HIP         -1.834e-01  1.448e-01  -1.267  0.20648
## THIGH        2.857e-01  1.362e-01   2.098  0.03693 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.438 on 244 degrees of freedom
## Multiple R-squared:  0.7266, Adjusted R-squared:  0.7187
## F-statistic: 92.62 on 7 and 244 DF,  p-value: < 2.2e-16
```

In fact, in this case the F statistic is the square of the t statistic and the two tests are *exactly identical*

```
cat("F: \n")
```

```
## F:
```

```
print(anova(modNoHEIGHT, ft)$F[2])
```

```
## [1] 1.104217
```

```
cat("Square of t-statistic: \n")
```

```
## Square of t-statistic:
```

```
print(summary(ft)$coef["HEIGHT", "t value"]^2)
```

```
## [1] 1.104217
```

This again shows us that our inference on β_j is equivalent to asking if adding in this variable significantly improves the fit of our model – i.e. on top of the existing variables.

6.7.2 Finding the best submodel

The above method compares a specific defined submodel to the full model. But we might instead want to *find* the best submodel for prediction. Conceptually we could imagine that we would just fit all of possible subsets of variables for the model and pick the best. That creates two problems

1. How to compare all of these models to each other? What measure should we use to compare models? For example, we've seen that the measures of fit we've discussed so far (e.g. R^2 and RSS) can't be directly compared between different sized models, so we have to determine how much improvement we would expect simply due to adding another variable.

2. There often way too many possible submodels. Specifically, there are 2^p different possible submodels. That's 256 models for 8 variables, which is actually manageable, in the sense that you can run 256 regressions on a computer. But the number grows rapidly as you increase the number of variables. You quickly can't even enumerate all the possible submodels in large datasets with a lot of variables.

6.7.3 Criterion for comparing models

We are going to quickly go over different types of statistics for comparing models. By a model M, we mean a linear model using a subset of our p variables. We will find the $\widehat{\beta}(M)$, which gives us a prediction model, and we will calculate a statistic based on our observed data that measures how well the model predicts y. Once we have such a statistic, say $T(M)$, we want to compare across models M_j and pick the model with the smallest $T(M_j)$ (or largest depending on the statistic).

Notice that this strategy as described is not inferential – we are not generally taking into account the variability of the $T(M_j)$, i.e. how $T(M_j)$ might vary for different random samples of the data. There might be other models M_k that

have slightly larger $T(M_k)$ on this data than the "best" $T(M_j)$, but in a different dataset $T(M_k)$ might be slightly smaller.

6.7.3.1 RSS: Comparing models with same number of predictors (RSS)

We've seen that the RSS (Residual Sum of Squares) is a commonly used measure of the performance of a regression model, but will always decrease as you increase the number of variables. However, RSS is a natural criterion to use when comparing models having the *same number* of explanatory variables.

A function in R that is useful for variable selection is `regsubsets` in the R package `leaps`. For each value of $k = 1, \ldots, p$, this function gives the best model with k variables according to the residual sum of squares.

For the body fat dataset, we can see what variables are chosen for each size:

```
library(leaps)
bFat = regsubsets(BODYFAT ~ ., body)
summary(bFat)
```

```
## Subset selection object
## Call: eval(expr, envir, enclos)
## 7 Variables  (and intercept)
##         Forced in Forced out
## AGE         FALSE      FALSE
## WEIGHT      FALSE      FALSE
## HEIGHT      FALSE      FALSE
## CHEST       FALSE      FALSE
## ABDOMEN     FALSE      FALSE
## HIP         FALSE      FALSE
## THIGH       FALSE      FALSE
## 1 subsets of each size up to 7
## Selection Algorithm: exhaustive
##          AGE WEIGHT HEIGHT CHEST ABDOMEN HIP THIGH
## 1  ( 1 ) " " " "    " "    " "   "*"     " " " "
## 2  ( 1 ) " " "*"    " "    " "   "*"     " " " "
## 3  ( 1 ) " " "*"    " "    " "   "*"     " " "*"
## 4  ( 1 ) " " "*"    " "    " "   "*"     "*" "*"
## 5  ( 1 ) " " "*"    "*"    " "   "*"     "*" "*"
## 6  ( 1 ) "*" "*"    "*"    " "   "*"     "*" "*"
## 7  ( 1 ) "*" "*"    "*"    "*"   "*"     "*" "*"
```

This output should be interpreted in the following way. The best model with one explanatory variable (let us denote this by M_1) is the model with *ABDOMEN*. The best model with two explanatory variables (denoted by M_2) is the one involving ABDOMEN and WEIGHT. And so forth. Here "best" means

in terms of RSS. This gives us 7 regression models, one for each choice of k: $M_1, M_2, \ldots, M_7$. The model M_7 is the full regression model involving all the explanatory variables.

For the body fat dataset, there's a natural hierarchy in the results, in that for each time k is increased, the best model M_k is found by adding another variable to the set variables in M_{k-1}. However, consider the car seat position data:

```
bSeat = regsubsets(hipcenter ~ ., seatpos)
summary(bSeat)
```

```
## Subset selection object
## Call: eval(expr, envir, enclos)
## 8 Variables  (and intercept)
##         Forced in Forced out
## Age         FALSE      FALSE
## Weight      FALSE      FALSE
## HtShoes     FALSE      FALSE
## Ht          FALSE      FALSE
## Seated      FALSE      FALSE
## Arm         FALSE      FALSE
## Thigh       FALSE      FALSE
## Leg         FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: exhaustive
##          Age Weight HtShoes Ht  Seated Arm Thigh Leg
## 1  ( 1 ) " " " "    " "     "*" " "    " " " "   " "
## 2  ( 1 ) " " " "    " "     "*" " "    " " " "   "*"
## 3  ( 1 ) "*" " "    " "     "*" " "    " " " "   "*"
## 4  ( 1 ) "*" " "    "*"     " " " "    " " "*"   "*"
## 5  ( 1 ) "*" " "    "*"     " " " "    "*" "*"   "*"
## 6  ( 1 ) "*" " "    "*"     " " "*"    "*" "*"   "*"
## 7  ( 1 ) "*" "*"    "*"     " " "*"    "*" "*"   "*"
## 8  ( 1 ) "*" "*"    "*"     "*" "*"    "*" "*"   "*"
```

Question: Does the carseat data have this hiearchy?

Note though, that we cannot compare the models $M_1, \ldots, M_7$ with RSS because they have different number of variables. Moreover, for the car seat position dataset, we also cannot use the F statistic to compare the models because the sets of variables in the different models are not subsets of each other.

6.7.3.2 Expected Prediction Error and Cross-Validation

The best criterion for comparing models are based on trying to minimize the **predictive performance** of the model, meaning for a new observation (y_0, x_0), how accurate is our prediction $\hat{y}(x_0)$ in predicting y_0? In other words, how small is

$$y_0 - \hat{y}(x_0).$$

This is basically like the residual, only *with data we haven't seen*. Of course there is an entire population of unobserved (y_0, x_0), so we can say that we would like to minimize the average error across the entire population of unseen observations

$$minE(y_0 - \hat{y}(x_0))^2.$$

This quantity is the **expected prediction error**.

This seems very much like our RSS

$$RSS = \sum_{i=1}^{n}(y_i - \hat{y}(x_i))^2,$$

specifically, RSS/n seems like it should be a estimate of the prediction error.

The problem is that when you use the *same data* to estimate both the $\hat{\beta}$ and the prediction error, the estimate of the prediction error will underestimate the true prediction error (i.e. it's a biased estimate). Moreover, the more variables you add (the larger p) the more it underestimates the true prediction error of that model. That doesn't mean smaller models are always better than larger models – the larger model's true prediction error may be less than the true prediction error of the smaller model – but that comparing the fit (i.e. RSS) as measured on the data used to estimate the model gets to be a worse and worse estimate of the prediction error for larger and larger models. Moreover, the larger the underlying noise (σ) for the model, the more bias there is as well; you can think that the extra variables are being used to try to fit to the noise seen in the data, which will not match the noise that will come with new data points. This is often why larger models are considered to **overfit** the data.

Instead we could imagine estimating the error by not using all of our data to fit the model, and saving some of it to evaluate which model is better. We divide our data into **training** and **test** data. We can then fit the models on the training data, and then estimate the prediction error of each on the test data.

```
set.seed(1249)
nTest <- 0.1 * nrow(body)
whTest <- sample(1:nrow(body), size = nTest)
bodyTest <- body[whTest, ]
bodyTrain <- body[-whTest, ]
predError <- apply(summary(bFat)$which[, -1], 1, function(x) {
    lmObj <- lm(bodyTrain$BODYFAT ~ ., data = bodyTrain[,
```

```
        -1][, x, drop = FALSE])
    testPred <- predict(lmObj, newdata = bodyTest[,
        -1])
    mean((bodyTest$BODYFAT - testPred)^2)
})
cat("Predicted error on random 10% of data:\n")
```

```
## Predicted error on random 10% of data:
```

```
predError
```

```
##        1        2        3        4        5        6        7
## 25.29712 28.86460 27.17047 28.65131 28.96773 28.92292 29.01328
```

Question: What does this suggest is the model with the smallest prediction error?

Of course this is just one random subset, and 10% of the data is only 25 observations, so there is a lot of possible noise in our estimate of the prediction error. If we take a different random subset it will be different:

```
set.seed(19085)
nTest <- 0.1 * nrow(body)
whTest <- sample(1:nrow(body), size = nTest)
bodyTest <- body[whTest, ]
bodyTrain <- body[-whTest, ]
predError <- apply(summary(bFat)$which[, -1], 1, function(x) {
    lmObj <- lm(bodyTrain$BODYFAT ~ ., data = bodyTrain[,
        -1][, x, drop = FALSE])
    testPred <- predict(lmObj, newdata = bodyTest[,
        -1])
    mean((bodyTest$BODYFAT - testPred)^2)
})
cat("Predicted error on random 10% of data:\n")
```

```
## Predicted error on random 10% of data:
```

```
predError
```

```
##        1        2        3        4        5        6        7
## 22.36633 22.58908 22.21784 21.90046 21.99034 21.94618 22.80151
```

Question: What about this random subset, which is the best size model?

So a natural idea is to average over a lot of random training sets. For various

reasons, we do something slightly different. We divide the data into 10 parts (i.e. each 10%), and use 9 of the parts to fit the model and 1 part to estimate prediction error, and repeat over all 10 partitions. This is called **cross-validation**.

```
set.seed(78912)
permutation <- sample(1:nrow(body))
folds <- cut(1:nrow(body), breaks = 10, labels = FALSE)
predErrorMat <- matrix(nrow = 10, ncol = nrow(summary(bFat)$which))
for (i in 1:10) {
    testIndexes <- which(folds == i, arr.ind = TRUE)
    testData <- body[permutation, ][testIndexes, ]
    trainData <- body[permutation, ][-testIndexes,
        ]
    predError <- apply(summary(bFat)$which[, -1], 1,
        function(x) {
            lmObj <- lm(trainData$BODYFAT ~ ., data = trainData[,
                -1][, x, drop = FALSE])
            testPred <- predict(lmObj, newdata = testData[,
                -1])
            mean((testData$BODYFAT - testPred)^2)
        })
    predErrorMat[i, ] <- predError
}
predErrorMat
```

```
##           [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]
##  [1,] 18.72568 10.95537 11.68551 12.16354 11.83839 11.78985 11.93013
##  [2,] 21.41687 21.08760 21.53709 21.06757 21.10223 21.20400 21.62519
##  [3,] 32.47863 21.97477 22.48690 22.50871 22.97452 22.92450 24.05130
##  [4,] 21.05072 20.22509 19.16631 18.82538 18.90923 18.89133 18.94164
##  [5,] 26.47937 22.92690 23.76934 26.13180 26.17794 26.12684 26.28473
##  [6,] 26.60945 23.35274 22.06232 22.06825 22.15430 23.10201 25.29325
##  [7,] 25.65426 20.48995 19.95947 19.82442 19.53618 19.97744 20.29104
##  [8,] 17.54916 18.79081 18.14251 17.67780 17.74409 17.67456 17.71624
##  [9,] 33.52443 27.26399 25.83256 26.87850 27.80847 28.32894 28.41455
## [10,] 18.64271 14.11973 14.05815 14.53730 14.42609 14.36767 14.57028
```

We then average these estimates:

```
colMeans(predErrorMat)
```

```
## [1] 24.21313 20.11870 19.87002 20.16833 20.26714 20.43871 20.91184
```

6.7.3.3 Closed-form criterion for comparing models with different numbers of predictors

There are other theoretically derived measures that estimate the expected predicted error as well. These can be computationally easier, or when you have smaller datasets may be more reliable.

The following are all measures for a model M, most of which try to measure the expected prediction error (we're not going to go into where they come from)

- **Leave-One-Out Cross Validation Score** This is basically the same idea as cross-validation, only instead of dividing the data into 10 parts, we make each single observation take turns being the test data, and all the other data is the training data. Specifically, for each observation i, fit the model M to the $(n-1)$ observations obtained by **excluding** the i^{th} observation. This gives us an estimates of β, $\hat{\beta}^{(i)}$. Then we predict the response for the i^{th} observation using $\hat{\beta}^{(-i)}$,

  $$\hat{y}^{(-i)} = \hat{\beta}_0^{(-i)} + \hat{\beta}_1^{(-i)} x^{(1)} + ... \hat{\beta}_p^{(-i)} x^{(p)}$$

  Then we have the error for predicting y_i based on a model *that didn't use the data* (y_i, x_i). We can do this for each $i = 1, ..., n$ and then get our estimate of prediction error,

  $$LOOCV(M) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}^{(-i)})^2$$

  In fact, LOOCV can be computed very quickly in linear regression from our residuals of the model without a lot of coding using algebraic facts about regression that we won't get into.[12]

- **Mallows Cp**

  $$C_p(M) = RSS(M)/n + \frac{2\hat{\sigma}^2(p+1)}{n}$$

  There are other ways of writing C_p as well. $\hat{\sigma}^2$ in this equation is the estimate based on the *full* model (with all predictors included.)

  In fact, $C_p(M)$ becomes equivalent to the LOOCV as n gets large (i.e. asymptotically).

- **Akaike Information Criterion (AIC)**

  $$AIC(M) = nlog(RSS(M)/n) + 2(p+1)$$

  In regression, AIC is equivalent to using C_p above, only with $\hat{\sigma}^2(M)$, i.e. the estimate of σ based on the model M.

[12] $LOOCV = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{r_i^2}{1-h_i} \right)^2$ where h_i is the diagonal of $X(X'X)^{-1}X$

- **Bayes Information Criterion (BIC)**

$$BIC(M) = nlog(RSS(M)/n) + (p+1)log(n)$$

We would note that all of these measures, except for C_p can be used for models that are more complicated than just regression models, though AIC and BIC are calculated differently depending on the prediction model.

Relationship to comparing models with same size k Also, if we are comparing only models with the same number of predictors, C_p, AIC and BIC are simply picking the model with the smallest RSS, like we did before. So we can imagine using our results from running `regsubsets` to find the best model, and then running these criterion on just the best of each one.

Adjusted R^2 Another common measure is the adjusted R^2. Recall that $R^2(M) = 1 - \frac{RSS(M)}{TSS} = 1 - \frac{RSS(M)/n}{TSS/n}$. The adjusted R^2 is

$$R^2_{adj}(M) = 1 - \frac{RSS(M)/(n-p-1)}{TSS/(n-1)} = 1 - \frac{\hat{\sigma}^2(M)}{\hat{var}(y)},$$

i.e. it uses the "right" values to divide by (i.e. right degrees of freedom), rather than just n. You will often see it printed out on standard regression summaries. It is an improvement over R^2 ($R^2_{adj}(M)$ doesn't always get larger when you add a variable), but is not as good of a measure of comparing models as those listed above.

Example: Comparing our best k-sized models

We can compare these criterion on the best k-sized models we found above:

```
LOOCV <- function(lm) {
    vals <- residuals(lm)/(1 - lm.influence(lm)$hat)
    sum(vals^2)/length(vals)
}
```

```
calculateCriterion <- function(x = NULL, y, dataset,
    lmObj = NULL) {
    sigma2 = summary(lm(y ~ ., data = dataset))$sigma^2
    if (is.null(lmObj))
        lmObj <- lm(y ~ ., data = dataset[, x, drop = FALSE])
    sumlmObj <- summary(lmObj)
    n <- nrow(dataset)
    p <- sum(x)
    RSS <- sumlmObj$sigma^2 * (n - p - 1)
    c(R2 = sumlmObj$r.squared, R2adj = sumlmObj$adj.r.squared,
        `RSS/n` = RSS/n, LOOCV = LOOCV(lmObj), Cp = RSS/n +
            2 * sigma2 * (p + 1)/n, CpAlt = RSS/sigma2 -
            n + 2 * (p + 1), AIC = AIC(lmObj), BIC = BIC(lmObj))
```

```
}
cat("Criterion for the 8 best k-sized models of car seat position:\n")
```

```
## Criterion for the 8 best k-sized models of car seat position:
```

```
critSeat <- apply(summary(bSeat)$which[, -1], 1, calculateCriterion,
    y = seatpos$hipcenter, dataset = seatpos[, -9])
critSeat <- t(critSeat)
critSeat
```

```
##          R2     R2adj    RSS/n    LOOCV       Cp      CpAlt      AIC      BIC
## 1 0.6382850 0.6282374 1253.047 1387.644 1402.818 -0.5342143 384.9060 389.8188
## 2 0.6594117 0.6399496 1179.860 1408.696 1404.516 -0.4888531 384.6191 391.1694
## 3 0.6814159 0.6533055 1103.634 1415.652 1403.175 -0.5246725 384.0811 392.2691
## 4 0.6848577 0.6466586 1091.711 1456.233 1466.137  1.1568934 385.6684 395.4939
## 5 0.6861644 0.6371276 1087.184 1548.041 1536.496  3.0359952 387.5105 398.9736
## 6 0.6864310 0.6257403 1086.261 1739.475 1610.457  5.0113282 389.4782 402.5789
## 7 0.6865154 0.6133690 1085.968 1911.701 1685.051  7.0035240 391.4680 406.2062
## 8 0.6865535 0.6000855 1085.836 1975.415 1759.804  9.0000000 393.4634 409.8392
```

```
cat("\nCriterion for the 7 best k-sized models of body fat:\n")
```

```
##
## Criterion for the 7 best k-sized models of body fat:
```

```
critBody <- apply(summary(bFat)$which[, -1], 1, calculateCriterion,
    y = body$BODYFAT, dataset = body[, -1])
critBody <- t(critBody)
critBody <- cbind(critBody, CV10 = colMeans(predErrorMat))
critBody
```

```
##          R2     R2adj    RSS/n    LOOCV       Cp     CpAlt      AIC      BIC
## 1 0.6616721 0.6603188 23.60104 24.30696 23.91374 53.901272 1517.790 1528.379
## 2 0.7187981 0.7165395 19.61605 20.27420 20.08510  4.925819 1473.185 1487.302
## 3 0.7234261 0.7200805 19.29321 20.07151 19.91861  2.796087 1471.003 1488.650
## 4 0.7249518 0.7204976 19.18678 20.13848 19.96853  3.434662 1471.609 1492.785
## 5 0.7263716 0.7208100 19.08774 20.21249 20.02584  4.167779 1472.305 1497.011
## 6 0.7265595 0.7198630 19.07463 20.34676 20.16908  6.000069 1474.132 1502.367
## 7 0.7265596 0.7187150 19.07463 20.62801 20.32542  8.000000 1476.132 1507.896
##       CV10
## 1 24.21313
## 2 20.11870
## 3 19.87002
## 4 20.16833
## 5 20.26714
## 6 20.43871
## 7 20.91184
```

6.7.4 Stepwise methods

With a large number of predictors, it may not be feasible to compare all 2^p submodels.

A common approach is to not consider all submodels, but compare only certain submodels using **stepwise regression** methods. The idea is to iteratively add or remove a single variable – the one that most improves your model – until you do not get an improvement in your model criterion score.

For example, we can start with our full model, and iteratively remove the least necessary variable, until we don't get an improvement (Backward Elimination). Alternatively we could imagine starting with no variables and add the best variable, then another, until there's no more improvement (Forward Elimination).

The choice of which variable to add or remove can be based on either the criterion given above, or also by comparing p-values (since each step is a submodel), but the most common usuage is not via p-values.

The most commonly used methods actually combine backward elimination and forward selection. This deals with the situation where some variables are added or removed early in the process and we want to change our mind about them later. For example, in the car seat position data, if you want to add a single best variable you might at the beginning choose `Ht`. But having `Ht` in the model might keep you from ever adding Ht Shoes, which in combination with `Wt` might do better than just `Ht` – i.e. the best model might be `Ht Shoes + Wt` rather than `Ht`, but you would never get to it because once `Ht` is in the model, `Ht Shoes` never gets added.

The function `step` in R will perform a stepwise search based on the AIC. The default version of the `step` function only removes variables (analogous to backward elimination). If one wants to add variables as well, you can set the argument `direction`.

```
outBody <- step(ft, trace = 0, direction = "both")
outBody
```

```
##
## Call:
## lm(formula = BODYFAT ~ WEIGHT + ABDOMEN + THIGH, data = body)
##
## Coefficients:
## (Intercept)       WEIGHT      ABDOMEN        THIGH
##    -52.9631      -0.1828       0.9919       0.2190
```

We can compare this to the best k-sized models we got before, and their measured criterion.

```
summary(bFat)$out
```

```
##          AGE WEIGHT HEIGHT CHEST ABDOMEN HIP THIGH
## 1  ( 1 ) " " " "    " "    " "   "*"     " " " "
## 2  ( 1 ) " " "*"    " "    " "   "*"     " " " "
## 3  ( 1 ) " " "*"    " "    " "   "*"     " " "*"
## 4  ( 1 ) " " "*"    " "    " "   "*"     "*" "*"
## 5  ( 1 ) " " "*"    "*"    " "   "*"     "*" "*"
## 6  ( 1 ) "*" "*"    "*"    " "   "*"     "*" "*"
## 7  ( 1 ) "*" "*"    "*"    "*"   "*"     "*" "*"
```

```
critBody
```

```
##          R2     R2adj    RSS/n    LOOCV       Cp     CpAlt      AIC      BIC
## 1 0.6616721 0.6603188 23.60104 24.30696 23.91374 53.901272 1517.790 1528.379
## 2 0.7187981 0.7165395 19.61605 20.27420 20.08510  4.925819 1473.185 1487.302
## 3 0.7234261 0.7200805 19.29321 20.07151 19.91861  2.796087 1471.003 1488.650
## 4 0.7249518 0.7204976 19.18678 20.13848 19.96853  3.434662 1471.609 1492.785
## 5 0.7263716 0.7208100 19.08774 20.21249 20.02584  4.167779 1472.305 1497.011
## 6 0.7265595 0.7198630 19.07463 20.34676 20.16908  6.000069 1474.132 1502.367
## 7 0.7265596 0.7187150 19.07463 20.62801 20.32542  8.000000 1476.132 1507.896
##       CV10
## 1 24.21313
## 2 20.11870
## 3 19.87002
## 4 20.16833
## 5 20.26714
## 6 20.43871
## 7 20.91184
```

We see that stepwise picked the same model.

We can do the same for the car seat position data.

```
outCarseat <- step(lmSeat, trace = 0, direction = "both")
outCarseat
```

```
##
## Call:
## lm(formula = hipcenter ~ Age + HtShoes + Leg, data = seatpos)
##
## Coefficients:
## (Intercept)          Age      HtShoes          Leg
##    456.2137       0.5998      -2.3023      -6.8297
```

We can again compare to the best model we found before.

```
summary(bSeat)$out
```

```
##          Age Weight HtShoes Ht  Seated Arm Thigh Leg
## 1  ( 1 ) " " " "    " "     "*" " "    " " " "   " "
```

```
## 2  ( 1 ) " " " "    " "     "*" " "    " " " " "*"
## 3  ( 1 ) "*" " "    " "     "*" " "    " " " " "*"
## 4  ( 1 ) "*" " "    "*"     " " " "    " " "*"  "*"
## 5  ( 1 ) "*" " "    "*"     " " " "    "*" "*"  "*"
## 6  ( 1 ) "*" " "    "*"     " " "*"    "*" "*"  "*"
## 7  ( 1 ) "*" "*"    "*"     " " "*"    "*" "*"  "*"
## 8  ( 1 ) "*" "*"    "*"     "*" "*"    "*" "*"  "*"
```

```
critSeat
```

```
##          R2     R2adj    RSS/n    LOOCV       Cp      CpAlt      AIC      BIC
## 1 0.6382850 0.6282374 1253.047 1387.644 1402.818 -0.5342143 384.9060 389.8188
## 2 0.6594117 0.6399496 1179.860 1408.696 1404.516 -0.4888531 384.6191 391.1694
## 3 0.6814159 0.6533055 1103.634 1415.652 1403.175 -0.5246725 384.0811 392.2691
## 4 0.6848577 0.6466586 1091.711 1456.233 1466.137  1.1568934 385.6684 395.4939
## 5 0.6861644 0.6371276 1087.184 1548.041 1536.496  3.0359952 387.5105 398.9736
## 6 0.6864310 0.6257403 1086.261 1739.475 1610.457  5.0113282 389.4782 402.5789
## 7 0.6865154 0.6133690 1085.968 1911.701 1685.051  7.0035240 391.4680 406.2062
## 8 0.6865535 0.6000855 1085.836 1975.415 1759.804  9.0000000 393.4634 409.8392
```

Notice that for the carseat dataset, the stepwise procedure doesn't give us the same best model as we had when we compared the size-k best models – it uses `Ht Shoes` rather than `Ht`.

If we calculate all criterion on the model found by the stepwise method, we see that that the AIC for the model found by the stepwise method is actually slightly larger than the best AIC found by looking at all submodels.

```
calculateCriterion(lmObj = outCarseat, y = seatpos$hipcenter,
    dataset = seatpos[, -9])
```

```
##          R2        R2adj        RSS/n        LOOCV           Cp        CpAlt
##   0.6812662    0.6531427 1201.5776327 1412.6121485 1276.4629022   -3.9088387
##         AIC          BIC
##  384.0989931  392.2869239
```

Drawbacks of Stepwise Regression Stepwise procedures are relatively cheap computationally but they do have drawbacks because of the one-at-a-time nature of adding/dropping variables, it is possible to miss the optimal model. We've already mentioned that most stepwise methods use a combination of adding and dropping variables to allow to reach more possible combinations. But ultimately, there may be a best model that can't be "found" by adding or dropping a single variable.

6.7.5 Inference After Selection

After finding the best fitting model, it is tempting to then do inference on this model, e.g. by looking at the p-values given by `summary` on the reduced model:

```
summary(outBody)
```

```
##
## Call:
## lm(formula = BODYFAT ~ WEIGHT + ABDOMEN + THIGH, data = body)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.4832  -3.2651  -0.0695   3.2634  10.1647
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -52.96313    4.30641 -12.299  < 2e-16 ***
## WEIGHT       -0.18277    0.02681  -6.817 7.04e-11 ***
## ABDOMEN       0.99191    0.05637  17.595  < 2e-16 ***
## THIGH         0.21897    0.10749   2.037   0.0427 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.428 on 248 degrees of freedom
## Multiple R-squared:  0.7234, Adjusted R-squared:  0.7201
## F-statistic: 216.2 on 3 and 248 DF,  p-value: < 2.2e-16
```

However, these p-values are no-longer valid. Bootstrap inference would also no longer be valid. Once you start using the data to pick and choose between the variables, then you no longer have valid p-values. You can think of this as a multiple testing problem – we've implicitly run *many* tests to find this model, and so these p-values don't account for the many tests.

Another way of thinking about it is that every set of variables will have the "best" possible subset, even if they are just pure random noise. But your hypothesis testing is not comparing to the distribution you would expect of the best possible subset from random noise, so you are comparing to the wrong distribution. Note that this problem with the p-values are present whether you use the formal methods we described above, or just manually play around with the variables, taking some in and out based on their p-values.

The first question for doing inference after selection is "why"? You are getting the best prediction error (at least based on your estimates) with these variables, and there's not a better model. One reason you might want to is that there is noise in our estimates of prediction error that we are not worrying about in picking the minimum.

Solution 1: Don't look for submodels!

You should really think about why you are looking for a smaller number of variables. If you have a large number of variables relative to your sample size,

a smaller model will often generalize better to future observations (i.e. give better predictions). If that is the goal (i.e. predictive modeling) then it can be important to get concise models, but then often inference on the individual variables is not terribly important.

In practice, often times people look for small models to find only the variables that "really matter", which is sort of implicitly trying to infer causality. And then they want inferential results (p-values, CI) to prove that these particular variables are significant. This is hedging very close to looking for causality in your variables. A great deal of meaningful knowledge about causality has cummulatively been found in observational data (epidemiological studies on human populations, for example), but it's really important to keep straight the interpretation of the coefficients in the model and what they are *not* telling you.

Generally, if you have a moderate number of variables relative to your sample size, and you want to do inference on the variables, you will probably do well to just keep all the variables in. In some fields, researchers are actually required to state *in advance of collecting any data* what variables they plan to analyze precisely so they don't go "fishing" for important variables.

Solution 2: Use different data for finding model and inference

If you do want to do inference after selection of submodels the simplest solution is to use a portion of your dataset to find the best model, and then use the remaining portion of the data to do inference. Since you will have used completely different data for finding the model than from doing inference, then you have avoided the problems with the p-values. This requires, however, that you have a lot of data. Moreover, using smaller amounts of data in each step will mean both that your choice of submodels might not be as good and that your inference will be less powerful.

```
## Linking to ImageMagick 6.9.12.3
## Enabled features: cairo, fontconfig, freetype, heic, lcms, pango, raw, rsvg, webp
## Disabled features: fftw, ghostscript, x11
```

Chapter 7

Logistic Regression

In the previous chapter, we looked at regression, where the goal is to understand the relationship between a response variable y and many explanatory variables. Regression allowed for the explanatory variables to be categorical, but required that the response y be a continuous variable. Now we are going to consider the setting where our response is categorical – specifically where the response takes on one of two values (e.g. "Yes" and "No" or "Positive" and "Negative"). This is often called **classification.**

7.1 The classification problem

The setting for the classification problem is similar to that of the regression problem. We have a response variable y and p explanatory variables $x_1, \ldots, x_p$. We collect data from n subjects on these variables.

The only difference between regression and classification is that in classification, the response variable y is binary, meaning it takes only two values; for our purposes we assume it is coded as 0 and 1, though in other settings you might see -1 and 1. In contrast, for regression the response variable is continuous. (The explanatory variables, as before, are allowed to be both continuous and discrete.)

There are many examples for the classification problem. Two simple examples are given below. We shall look at more examples later on.

Frogs Dataset

This dataset consists of 212 sites of the Snowy Mountain area of New South Wales, Australia. Each site was surveyed to understand the distribution of the

Southern Corroboree frog. The variables are is available as a dataset in R via the package DAAG.

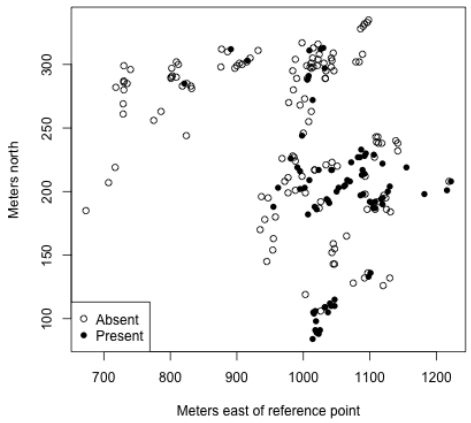
```
library(DAAG)
data(frogs)
```

The variables are:

1. `pres.abs` – 0/1 indicates whether frogs were found.
2. `easting` – reference point
3. `northing` – reference point
4. `altitude` – altitude in meters
5. `distance` – distance in meters to nearest extant population
6. `NoOfPools`– number of potential breeding pools
7. `NoOfSites`– number of potential breeding sites within a 2 km radius
8. `avrain` – mean rainfall for Spring period
9. `meanmin` – mean minimum Spring temperature
10. `meanmax` – mean maximum Spring temperature

The variable `easting` refers to the distance (in meters) east of a fixed reference point. Similarly `northing` refers to the distance (in meters) north of the reference point. These two variables allow us to plot the sites in terms of a map, where we color in sites where the frog was found:

```
presAbs <- factor(frogs$pres.abs, levels = c(0, 1),
    labels = c("Absent", "Present"))
plot(northing ~ easting, data = frogs, pch = c(1, 16)[presAbs],
    xlab = "Meters east of reference point", ylab = "Meters north")
legend("bottomleft", legend = levels(presAbs), pch = c(1,
    16))
```



A natural goal is to under the relation between the occurence of a frog (`pres.abs`) variable and the other geographic and environmental variables. This naturally falls under the classification problem because the response variable `pres.abs` is binary.

Email Spam Dataset

This dataset is from Chapter 10 of the book *Data Analysis and Graphics using R*. The original dataset is from the UC Irvine Repository of Machine Learning. The original dataset had 4607 observations and 57 explanatory variables. The authors of the book selected 6 of the 57 variables.

```
data(spam7)
head(spam7)
```

```
##   crl.tot dollar  bang money n000 make yesno
## 1     278  0.000 0.778  0.00 0.00 0.00     y
## 2    1028  0.180 0.372  0.43 0.43 0.21     y
## 3    2259  0.184 0.276  0.06 1.16 0.06     y
## 4     191  0.000 0.137  0.00 0.00 0.00     y
## 5     191  0.000 0.135  0.00 0.00 0.00     y
## 6      54  0.000 0.000  0.00 0.00 0.00     y
```

```
spam = spam7
```

The main variable here is `yesno` which indicates if the email is spam or not. The other variables are explanatory variables. They are:

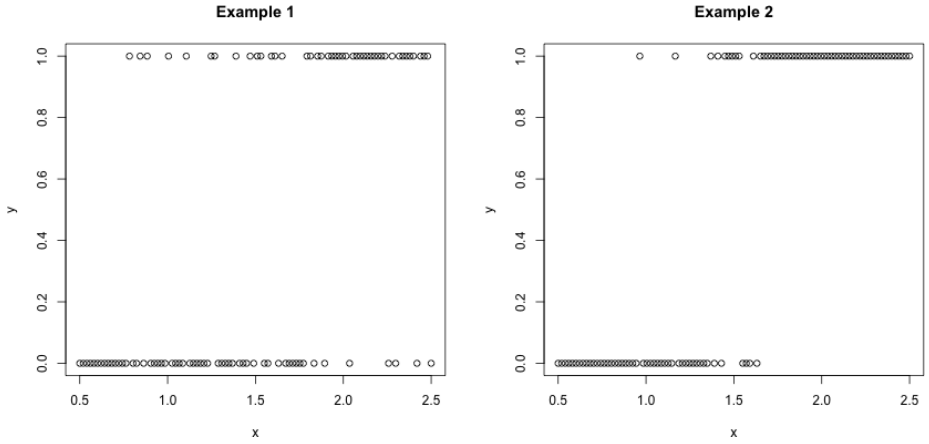
1. `crl.tot` - total length of words that are in capitals
2. `dollar` - frequency of the $ symbol, as percentage of all characters
3. `bang` - freqency of the ! symbol, as a percentage of all characters,
4. `money` - freqency of the word *money*, as a percentage of all words,
5. `n000` - freqency of the text string *000*, as percentage of all words,
6. `make` - freqency of the word *make*, as a percentage of all words.

The goal is mainly to predict whether a future email is spam or not based on these explanatory variables. This is once again a classification problem because the response is binary.

There are, of course, many more examples where the classification problem arises naturally.

7.2 Logistic Regression Setup

We consider the simple example, where we just have a single predictor, x. We could plot our y versus our x, and might have something that looks like these two examples plotted below (this is toy data I made up):



Our goal is to predict the value of y from our x.

Question: What do you notice about the relationship of y with x in the above two examples? In which of the two examples is it easier to predict y?

Question: This is a standard x-y scatterplot from regression, but is a pretty lousy plot for binary data! How could you better illustrate this data?

Prediction of Probability

Logistic regression does not directly try to predict values 0 or 1, but instead tries to predict the *probability* that y is 1 as a function of its variables,

$$p(x) = P(Y = 1|x)$$

Note that this can be thought of as our model for the random process of how the data were generated: for a given value of x, you calculate the $p(x)$, and then toss a coin that has probability $p(x)$ of heads. If you get a head, $y = 1$, otherwise $y = 0$.[1] The coin toss provides the randomness – $p(x)$ is an unknown but fixed quantity.

Note that unlike regression, we are not writing y as a function of x plus some noise (i.e. plus a random noise term). Instead we are writing $P(Y = 1)$ as a function of x; our randomness comes from the random coin toss we perform once we know $p(x)$.

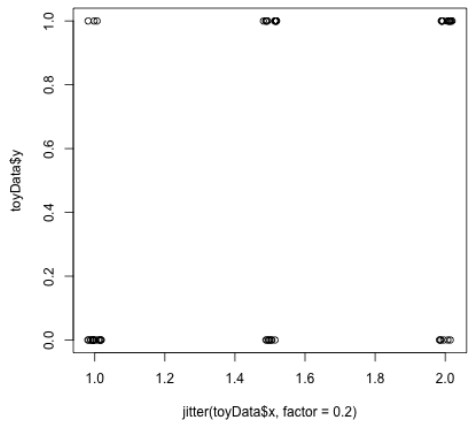
We can use these probabilities to try to predict what the actual observed y is more likely to be (the classification problem). For example, if $P(Y = 1|x) > 0.5$

[1] I.e. conditional on x, y is distributed $Bernoulli(p(x))$.

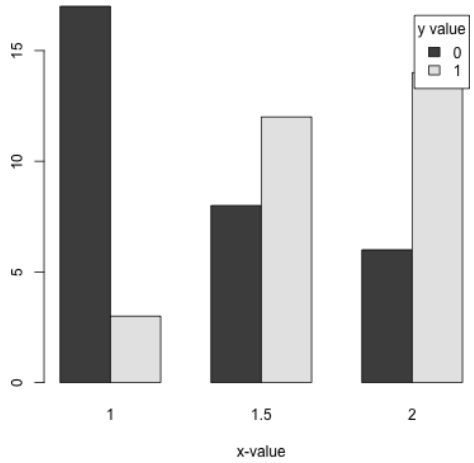
we could decide to predict that $y = 1$ and otherwise $y = 0$; we could also use the probabilities to be more stringent. For example, we could predict $y = 1$ if $P(Y = 1|x) > 0.7$ and $y = 0$ if $P(Y = 1|x) < 0.3$ and otherwise not make a prediction. The point is that if we knew the probabilities, we could decide what level of error in either direction we were willing to make. We'll discuss this more later in the chapter.

7.2.1 Estimating Probabilities

How can we estimate these probabilities? Let's think of a simpler example, where we observe many observations but only at three values x (1, 1.5, and 2), and their corresponding y. In the following plot I've added a small amount of noise to the x values so they don't overplot on each other, but there are actually only three values:

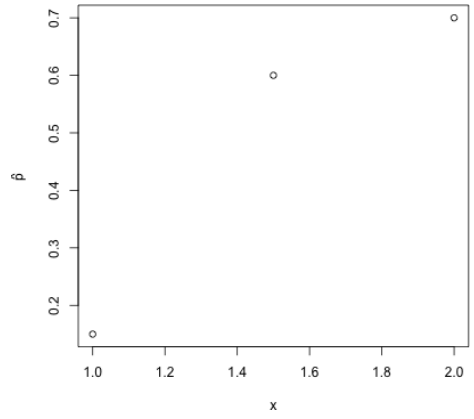


We can see this is still not entirely sufficient to see precisely the proportion of 0 versus 1 values at each x value, so I will plot each x value as a separate barplot so we can directly compare the proportion of 0 versus 1 values at each x value:



With this kind of data (i.e. with replication at each x-value), we could imagine estimating the probability that $y = 1$ if $x = 2$, for example: we could take the proportion of $y = 1$ values for the observations that have $x = 2$ (0.6. We could call this value $\hat{p}(2)$. We could do this for each of the three values of x, getting $\hat{p}(1)$,$\hat{p}(1.5)$,$\hat{p}(2)$.

We could also then try to see how the probability of $y = 1$ is changing with x. For example, if we plotted $\hat{p}(x)$ from above we would have:



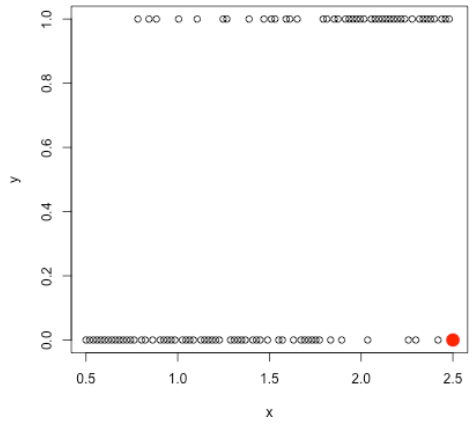
So in this example, we see an increase in the value $\hat{p}(x)$ as x increases, i.e. the probability of $y = 1$ increases with x. However, with only 3 values of x we couldn't describe $\hat{p}(x)$ as a function of x since we couldn't say much about how $\hat{p}(x)$ changes with x for other values.

But if we had this kind of data with a lot of different x values, we could think of a way to estimate how $\hat{p}(x)$ is changing with x, perhaps with curve fitting methods we've already considered.

Without repeated observations

More generally, however, we won't have multiple observations with the same x.

Returning to the more realistic toy data example I created earlier ("Example 1"), we only have one observation at each x value, but we have a lot of x values. Consider, for example, estimating the probability at $x = 2.5$ (colored in red). We only have one such value, and it's y values is 0:



We only have 1 observation, and $\hat{p}(2.5) = 0$ is a *very* bad prediction based on 1 observation. Indeed, looking at the surrounding x values around it, it is clear from the plot that the probability that $y = 1$ when $x = 2.5$ is probably pretty high, though not 1. We happen to get $y = 0$ at $x = 2.5$, but that was probably random chance.

We could do something, like try to bin together similar x values and estimate a single probability for similar values of x, like our local regression fitting in earlier chapters. Creating bins gets complicated when you have more than one explanatory variable (though we will see that we do something similar to this in decision trees, in the next chapter).

Instead, we will focus on predicting the function $p(x)$ by assuming $p(x)$ is a straightforward function of x.

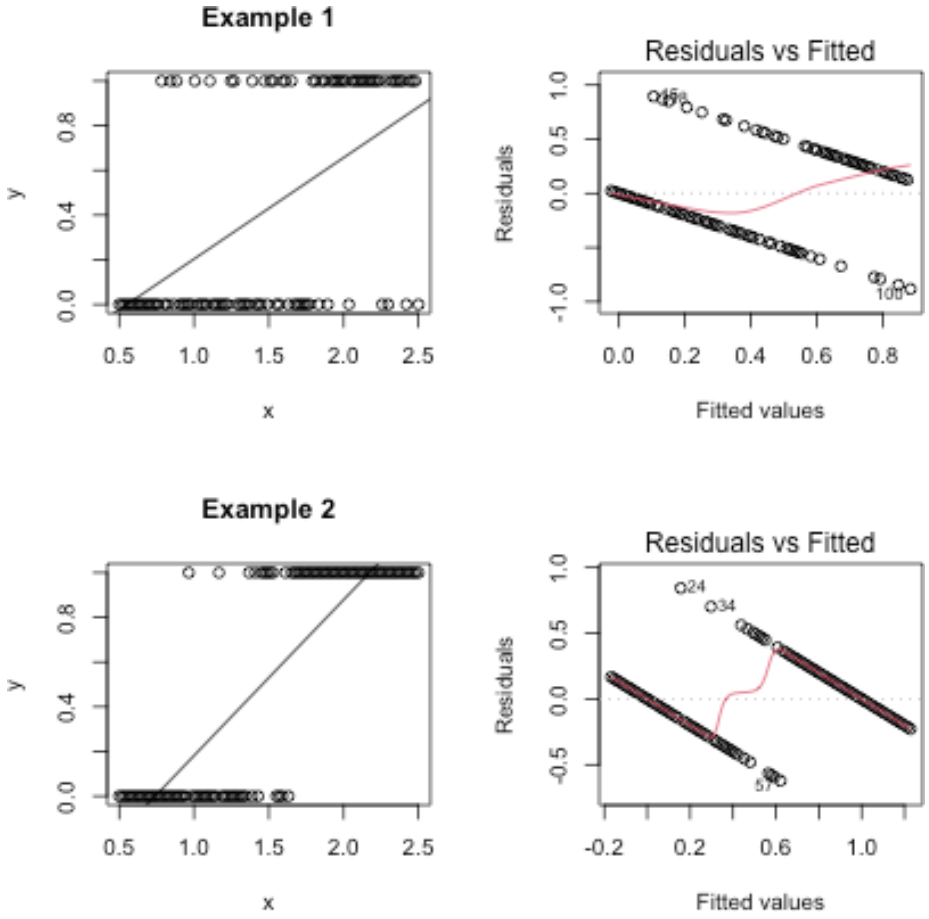
Why not regression?

We could ask, why don't we just do regression to predict y? In other words, we could do standard regression with the observed y values as the response.

Numerically, we *can* do it, in the sense that `lm` will give us an answer that vaguely seems reasonable. Here is the result from fitting to the two example data sets from above ("Example 1" and "Example 2" from above):

```
par(mfrow = c(2, 2))
plot(y ~ x, data = toyDataCont, main = "Example 1")
abline(lm(y ~ x, data = toyDataCont))
plot(lm(y ~ x, data = toyDataCont), which = 1)
```

```
plot(y ~ x, data = toyDataCont2, main = "Example 2")
abline(lm(y ~ x, data = toyDataCont2))
plot(lm(y ~ x, data = toyDataCont2), which = 1)
```



Question: What do you notice about using these predicted lines as an estimate $\hat{p}(x)$?

This result doesn't give us a prediction for probabilities (since values are outside of [0,1]).

What about forgetting about predicting the probability and just try to predict a 0 or 1 for y (i.e. classification)? The regression line doesn't give us an obvious way to create a classification. We'd have to make some decision (based on our data) on what value of x to decide to make a prediction of 1 versus zero. Since

we aren't correctly predicting probabilities, there's no obvious cutoff, though we could use our data to try to pick one.

7.2.2 Logit function / Log Odds

What do we do instead?

Instead, we want a function for $p(x)$ that is constrained within [0,1]. While we could try to figure out such a function, we are instead going to take another tack which will lead us to such a function. Specifically, we are going to consider transforming our $p(x)$ so that it takes on all real-valued values, say

$$z(x) = \tau(p(x))$$

Instead of trying to estimate $p(x)$, we will try to estimate $z(x)$ as a function of our x.

Why? Because it will mean that $z(x)$ is no longer constrained to be between 0 and 1. Therefore, we can be free to use any simple modeling we've already learned to estimate $z(x)$ (for example linear regression) without worrying about any constraints. Then to get $\hat{p}(x)$, we will invert to get

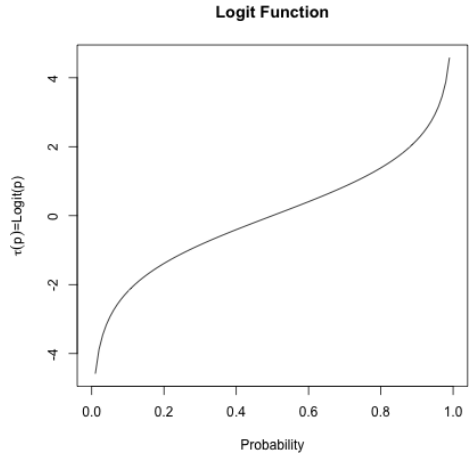
$$\hat{p}(x) = \tau^{-1}(\hat{z}(x)).$$

(Note that while $p(x)$ and $z(x)$ are unknown, we pick our function τ so we don't have to estimate it)

The other reason is that we are going to choose a function τ such that the value $z(x)$ is actually interpretable and makes sense on its own; thus the function $z(x)$ is actually meaningful to estimate on its own.

There are different reasonable choices for τ, but we are going to focus on the one traditionally used in logistic regression. The function we are going to use is called the **logit** function. It takes any value p in (0,1), and returns a real value:

$$\tau(p) = logit(p) = log(\frac{p}{1-p}).$$



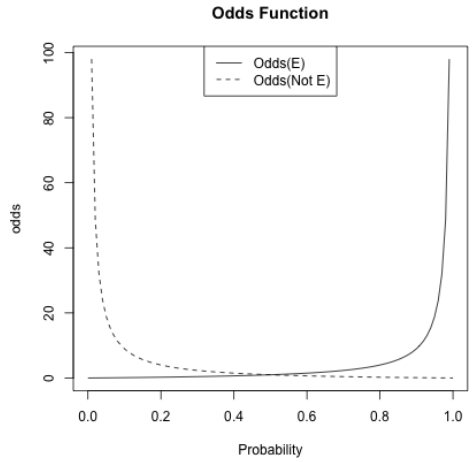
The value $z = logit(p)$ is interpretable as the log of the *odds*, a common measure of discussing the probability of something.

Odds

Let p be the probability of an event E, $p = P(E)$. For example, our event could be $E = \{y = 1\}$, and then $p = P(E) = P(y = 1)$). Then the **odds** of the event E is denoted by $odds(E)$ and defined as

$$odds(E) := \frac{P(E \text{ happens})}{P(E \text{ does not happen})} = \frac{P(E)}{1 - P(E)} = \frac{p}{1 - p}$$

An important thing to note is that while $p = P(E)$ lies between 0 and 1, the odds of E ($odds(E)$) is only restricted to be nonnegative – i.e. the odds takes on a wider range of values.



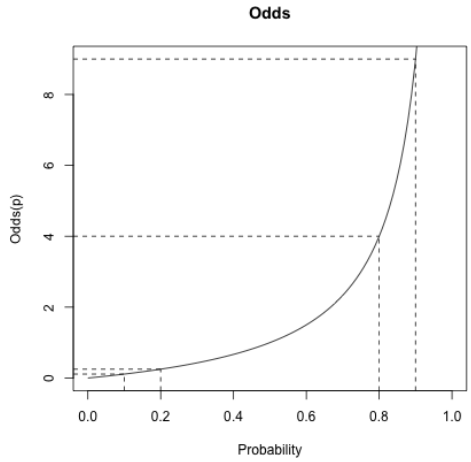
Note the following simple formulae relating probability and odds:

$$p = P(E) = \frac{odds(E)}{1 + odds(E)}$$

So if you know the odds of an event, you can also calculate the probability of the event.

Log Odds

From a modeling perspective, it is still akward to work with odds – for example to try to predict the odds – because it must be positive. Moreover, it's not symmetric in our choice of whether we consider $P(y = 1)$ versus $P(y = 0)$. Changing the probability of $y = 1$ from 0.8 to 0.9 create large differences in the odds, but similarly changing the probability from 0.2 to 0.1 create small changes in the odds.

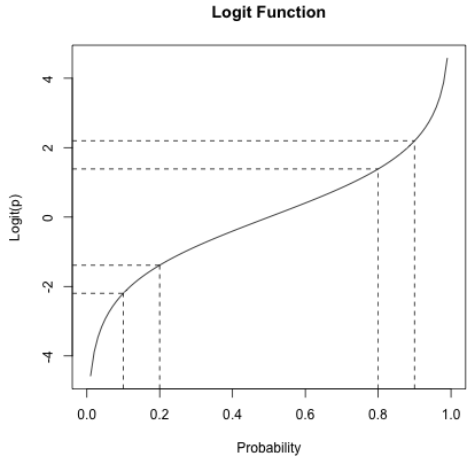


This is unfortunate, since the choice of trying to estimate $P(y = 1)$ is arbitrary – we could have just as easily considered $P(y = 0)$ and modeled that quantity.

However, if we take the log of the odds, there is no restriction on the value of $\log(odds(E))$ i.e.,

$$log\left(\frac{p}{1-p}\right)$$

As your probability p ranges between 0 and 1, the log-odds will take on all real-valued numbers. Moreover, the logit function is symmetric around 0.5, meaning that the difference between the log-odds of $p = 0.8$ versus $p = 0.9$ is the same difference as the log-odds of $p = 0.2$ versus $p = 0.1$:



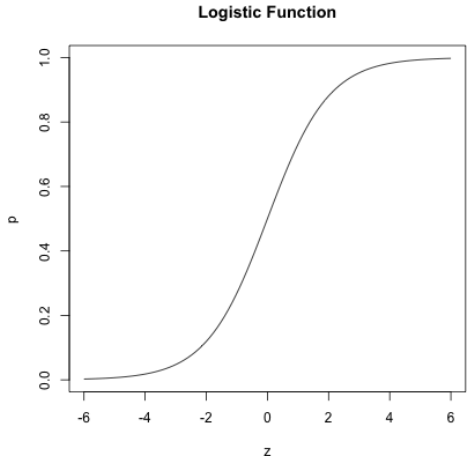
Converting from log-odds to probability

As we discussed above, our goal is going to be to model $z = logit(p)$, and then be able to transform from z back to p.

We have the simple relationship between z and the probability p

$$p = P(E) = \tau^{-1}(z) = \frac{e^z}{1+e^z} = \frac{1}{1+e^{-z}}.$$

This function τ^{-1} is called the **logistic function**. For any real-valued z, the logistic function converts that number into a value between 0-1 (i.e. a probability).



7.2.3 Logistic Regression Model

Logistic regression, then, is to model the $logit(p)$ (i.e. the log-odds of the event), as a linear function of the explanatory variable values x_i of the i^{th} individual. Again, this is a feasible thing to do, since $\log odds$ take on the full range of values, so that we won't have to figure out how to make sure we don't predict probabilities outside of 0-1. Then, because of the relationships above, this gives us a model for the probabilities with respect to our explanatory variables x.

The logistic regression model, for $p = P(y = 1)$, is given as:

$$\log(\frac{p}{1-p}) = \log\left(odds(y=1)\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p.$$

This means that we are modeling the probabilities as

$$p(x) = \frac{\exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p)}{1 + \exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p)}$$

Visualizing Logistic Model

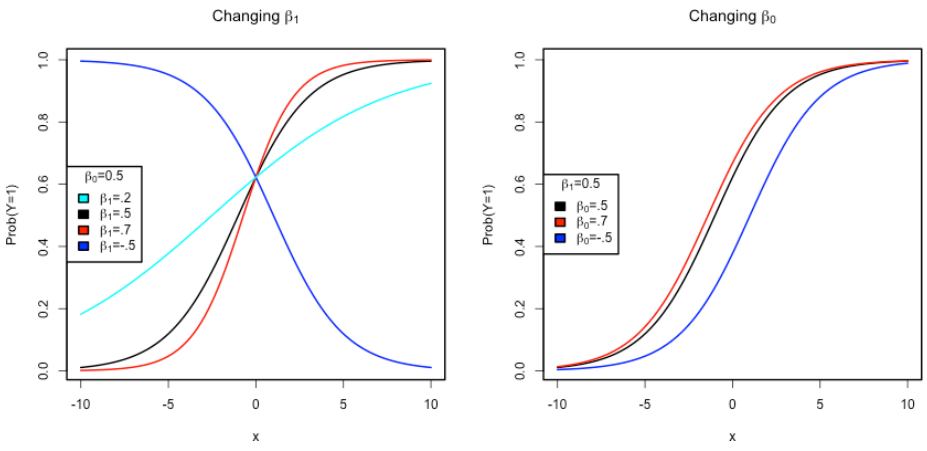
To understand the effect of these variables, let's consider our model for a single variable x:

$$log(\frac{p_i}{1-p_i}) = \log(odds(y_i = 1)) = \beta_0 + \beta_1 x_i$$

which means

$$p_i = \frac{\exp\left(\beta_0 + \beta_1 x_i\right)}{1 + exp(\beta_0 + \beta_1 x_i)}$$

We can visualize the relationship of the probability p of getting $y = 1$ as a function of x, for different values of β



Fitting the Logistic Model in R

The R function for logistic regression in R is `glm()` and it is not very different from `lm()` in terms of syntax.

For the frogs dataset, we will try to fit a model (but without the geographical variables)

```
frogsNoGeo <- frogs[, -c(2, 3)]
glmFrogs = glm(pres.abs ~ ., family = binomial, data = frogsNoGeo)
summary(glmFrogs)
```

```
##
## Call:
## glm(formula = pres.abs ~ ., family = binomial, data = frogsNoGeo)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7215  -0.7590  -0.2237   0.8320   2.6789
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.105e+02  1.388e+02   0.796  0.42587
## altitude    -3.086e-02  4.076e-02  -0.757  0.44901
## distance    -4.800e-04  2.055e-04  -2.336  0.01949 *
## NoOfPools    2.986e-02  9.276e-03   3.219  0.00129 **
## NoOfSites    4.364e-02  1.061e-01   0.411  0.68077
## avrain      -1.140e-02  5.995e-02  -0.190  0.84920
## meanmin      4.899e+00  1.564e+00   3.133  0.00173 **
## meanmax     -5.660e+00  5.049e+00  -1.121  0.26224
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 279.99  on 211  degrees of freedom
## Residual deviance: 198.74  on 204  degrees of freedom
## AIC: 214.74
##
## Number of Fisher Scoring iterations: 6
```

GLM is not just the name of a function in R, but a general term that stands for **Generalized Linear Model**. Logistic regression is a special case of a generalized linear model; the `family = binomial` clause in the function call above tells R to fit a logistic regression equation to the data – namely what kind of function to use to determine whether the predicted probabilities fit our data.

7.3 Interpreting the Results

7.3.1 Coefficients

The parameter β_j is interpreted as the change in *log-odds* of the event $y = 1$ for a unit change in the variable x_j provided all other explanatory variables are kept unchanged. Equivalently, e^{β_j} can be interpreted as the multiplicative change in *odds* due to a unit change in the variable x_j – provided all other explanatory variables are kept unchanged.

The R function provides estimates of the parameters $\beta_0, \ldots, \beta_p$. For example, in the frogs dataset, the estimated coefficient of the variable `NoOfPools` is 0.02986. This is interpreted as the change in log-odds of the event of finding a frog when the `NoOfPools` increases by one (provided the other variables remain unchanged). Equivalently, the odds of finding a frog get multiplied by $\exp(0.02986) = 1.03031$ when the `NoOfPools` increases by one.

P-values are also provided for each $\hat{\beta}_j$; they have a similar interpretation as in linear regression, namely evaluating the null hypothesis that $\beta_j = 0$. We are not going to go into how these p-values are calculated. Basically, if the model is true, $\hat{\beta}_j$ will be approximately normally distributed, with that approximation being better for larger sample size. Logistic regression significance statements rely much more heavily on asymptotics (i.e. having large sample sizes), *even if the data exactly follows the data generation model!*

7.3.2 Fitted Values and prediction

Now suppose a new site is found in the area for which the explanatory variable values are:

- `altitude=1700`
- `distance=400`
- `NoOfPools=30`
- `NoOfSites=8`
- `avrain=150`
- `meanmin=4`
- `meanmax=16`

What can our logistic regression equation predict for the presence or absence of frogs in this area? Our logistic regression allows us to calculate the log(*odds*) of finding frogs in this area as:

```
x0 = c(1, 1700, 400, 30, 8, 150, 4, 16)
sum(x0 * glmFrogs$coefficients)
```

```
## [1] -13.58643
```

Remember that this is $\log(odds)$. From here, the odds of finding frogs is calculated as

```
exp(sum(x0 * glmFrogs$coefficients))
```

```
## [1] 1.257443e-06
```

These are very low odds. If one wants to obtain an estimate of the **probability** of finding frogs at this new location, we can use the formula above to get:

```
exp(sum(x0 * glmFrogs$coefficients))/(1 + exp(sum(x0 *
    glmFrogs$coefficients)))
```

```
## [1] 1.257441e-06
```

Therefore, we will predict that this species of frog will not be present at this new location.

Similar to fitted values in linear regression, we can obtain fitted probabilities in logistic regression for each of the observations in our sample using the `fitted` function:

```
head(fitted(glmFrogs))
```

```
##         2         3         4         5         6         7
## 0.9994421 0.9391188 0.8683363 0.7443973 0.9427198 0.7107780
```

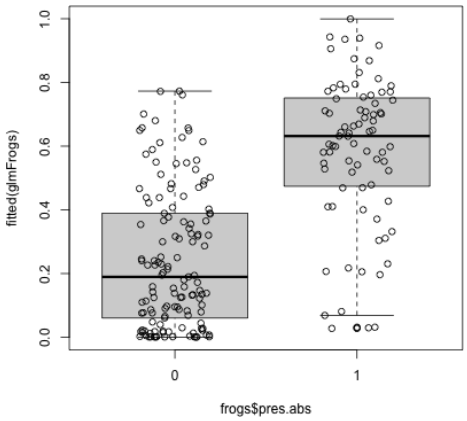
These fitted values are the *fitted probabilities* for each observation in our sample. For example, for $i = 45$, we can also calculate the fitted value manually as:

```
i = 45
rrg = c(1, frogs$altitude[i], frogs$distance[i], frogs$NoOfPools[i],
    frogs$NoOfSites[i], frogs$avrain[i], frogs$meanmin[i],
    frogs$meanmax[i])
eta = sum(rrg * glmFrogs$coefficients)
prr = exp(eta)/(1 + exp(eta))
c(manual = prr, FittedFunction = unname(glmFrogs$fitted.values[i]))
```

```
##         manual FittedFunction
##      0.5807378      0.5807378
```

The following plots the fitted values against the actual response:

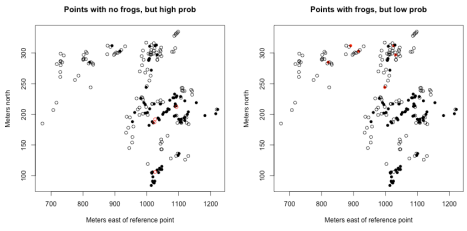
```
boxplot(fitted(glmFrogs) ~ frogs$pres.abs, at = c(0,
    1))
points(x = jitter(frogs$pres.abs), fitted(glmFrogs))
```



Question: Why do I plot this as a boxplot?

Some of the regions where frogs were present seems to have received very low fitted probability under the model (and conversely, some of the regions with high fitted probability did not actually have any frogs). We can look at these unusual points in the following plot:

```
high0 <- frogs$pres.abs == 0 & glmFrogs$fitted > 0.7
low1 <- frogs$pres.abs == 1 & glmFrogs$fitted < 0.2
par(mfrow = c(1, 2))
plot(northing ~ easting, data = frogs, pch = c(1, 16)[frogs$pres.abs +
    1], col = c("black", "red")[factor(high0)], xlab = "Meters east of reference point",
    ylab = "Meters north", main = "Points with no frogs, but high prob")
plot(northing ~ easting, data = frogs, pch = c(1, 16)[frogs$pres.abs +
    1], col = c("black", "red")[factor(low1)], xlab = "Meters east of reference point",
    ylab = "Meters north", main = "Points with frogs, but low prob")
```



Question: What do you notice about these points?

7.3.3 Fitting the model & Residual Deviance

We haven't discussed how `glm` found the "best" choice of coefficients β for our model. Like regression, the coefficients are chosen based on getting the best fit to our data, but how we measure that fit is different for logistic regression.

In regression we considered the squared residual as a measure of our fit for each observation i,

$$(y_i - \widehat{y}_i)^2,$$

and minimizing the average fit to our data. We will do something similar in logistic regression, but

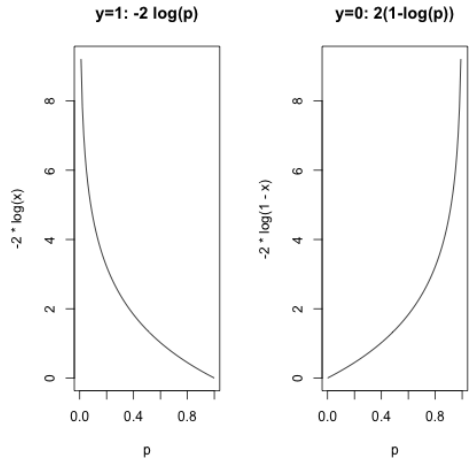
1. We will consider the fit of the fitted *probabilities*
2. The criterion we use to determine the best coefficients β is not the residual, but another notion of "fit" for every observation.

Let $\widehat{p}_1, \ldots, \widehat{p}_n$ denote the fitted probabilities in logistic regression for a possible vector of coefficients $\beta_1, \ldots, \beta_p$. The actual response values are $y_1, \ldots, y_n$ (remember our responses y are binary, i.e. 0-1 values). If the fit is good, we would expect $\widehat{p}_i$ to be small (close to zero) when y_i is 0 and $\widehat{p}_i$ to be large (close to one) when y_i is 1. Conversely, if the fit is not good, we would expect $\widehat{p}_i$ to be large for some y_i that is zero and $\widehat{p}_i$ to be small for some y_i that is 1.

A commonly used function for measuring if a probability p is close to 0 is

$$-2 \log p.$$

This quantity is always nonnegative and it becomes very large if p is close to zero. Similarly, one can measure if a probability p is close to 1 by $-2\log(1-p)$.



Using these quantities, we measure the quality of fit of $\widehat{p}_i$ to y_i by

$$Dev(\widehat{p}_i, y_i) = \begin{cases} -2\log \widehat{p}_i & : y_i = 1 \\ -2\log\left(1 - \widehat{p}_i\right) & : y_i = 0 \end{cases}$$

This is called the **deviance**.[2] If $Dev(\hat{p}_i, y_i)$ is large, it means that $\hat{p}_i$ is **not** a good fit for y_i.

Because y_i is either 0 or 1, the above formula for $Dev(\hat{p}_i, y_i)$ can be written more succinctly as

$$Dev(\hat{p}_i, y_i) = y_i \left(-2 \log \hat{p}_i\right) + (1 - y_i) \left(-2 \log(1 - \hat{p}_i)\right).$$

Note that this is the deviance for the i^{th} observation. We can get a measure of the overall goodness of fit (across all observations) by simply summing this quantity over all our observations. The resulting quantity is called the **Residual Deviance**:

$$RD = \sum_{i=1}^{n} Dev(\hat{p}_i, y_i).$$

Just like RSS, small values of RD are preferred and large values indicate lack of fit.

This doesn't have our β_j anywhere, so how does this help in choosing the best β? Well, remember that our fitted values $\hat{p}_i$ is a specific function of our x_i values:

$$\hat{p}_i = \hat{p}(x_i) = \tau^{-1}(\hat{\beta}_0 + ... + \hat{\beta}_p x_i^{(p)}) = \frac{\exp(\hat{\beta}_0 + ... + \hat{\beta}_p x_i^{(p)})}{1 + \exp(\hat{\beta}_0 + ... + \hat{\beta}_p x_i^{(p)})}$$

So we can put those values into our equation above, and find the $\hat{\beta}_j$ that maximize that quantity. Unlike linear regression, this *has* to be maximized by a computer – you can't write down a mathematical expression for the $\hat{\beta}_j$ that minimize the residual deviance.

Residual Deviance in R

The function `deviance` can be used in R to calculate deviance. It can, of course, also be calculated manually using the fitted probabilities.

RD (Residual Deviance) can be calculated from our `glm` object as

```
deviance(glmFrogs)
```

```
## [1] 198.7384
```

[2] This comes from assuming that the y_i follow a Bernoulli distribtion with probability p_i. Then this is the negative log-likelihood of the observation, and by minimizing the average of this over all observations, we are maximizing the likelihood.

7.4 Comparing Models

7.4.1 Deviance and submodels

Residual Deviance has parallels to RSS in linear regression, and we can use deviance to compare models similarly to linear regression.

RD decreases as you add variables

Just like the RSS in linear regression, the RD in logistic regression will always decrease as more variables are added to the model. For example, in the frogs dataset, if we remove the variable `NoOfPools`, the RD changes to:

```
m2 = glm(pres.abs ~ altitude + distance + NoOfSites +
    avrain + meanmin + meanmax, family = binomial,
    data = frogs)
deviance(m2)
```

```
## [1] 210.8392
```

```
deviance(glmFrogs)
```

```
## [1] 198.7384
```

Note that RD decreased from 210.84 to 198.7384 by adding `NoOfPools`.

The Null Model (No Variables)

We can similarly ask whether we need *any* of the variables (like the F test). The Null Deviance (ND) is the analogue of TSS (Total Sum of Squares) in linear regression. It simply refers to the deviance when there are no explanatory variables i.e., when one does logistic regression with only the intercept.

We can fit a model in R with no variables with the following syntax:

```
m0 <- glm(pres.abs ~ 1, family = binomial, data = frogs)
deviance(m0)
```

```
## [1] 279.987
```

Note, when there are no explanatory variables, the fitted probabilities are all equal to $\bar{y}$:

```
head(fitted(m0))
```

```
##         2         3         4         5         6         7
## 0.3726415 0.3726415 0.3726415 0.3726415 0.3726415 0.3726415
```

```
mean(frogs$pres.abs)
```

```
## [1] 0.3726415
```

Notice that this null deviance is reported in the summary of the full model we fit:

```
summary(glmFrogs)
```

```
##
## Call:
## glm(formula = pres.abs ~ ., family = binomial, data = frogsNoGeo)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7215  -0.7590  -0.2237   0.8320   2.6789
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.105e+02  1.388e+02   0.796  0.42587
## altitude    -3.086e-02  4.076e-02  -0.757  0.44901
## distance    -4.800e-04  2.055e-04  -2.336  0.01949 *
## NoOfPools    2.986e-02  9.276e-03   3.219  0.00129 **
## NoOfSites    4.364e-02  1.061e-01   0.411  0.68077
## avrain      -1.140e-02  5.995e-02  -0.190  0.84920
## meanmin      4.899e+00  1.564e+00   3.133  0.00173 **
## meanmax     -5.660e+00  5.049e+00  -1.121  0.26224
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 279.99  on 211  degrees of freedom
## Residual deviance: 198.74  on 204  degrees of freedom
## AIC: 214.74
##
## Number of Fisher Scoring iterations: 6
```

Significance of submodels

The deviances come with degrees of freedom. The degrees of freedom of RD is $n - p - 1$ (exactly equal to the residual degrees of freedom in linear regression) while the degrees of freedom of ND is $n - 1$.

Unlike regression, the automatic summary does not give a p-value as to whether this is a significant change in deviance. Similarly, the `anova` function for comparing submodels doesn't give a significance for comparing a submodel to the larger model

```
anova(m0, glmFrogs)
```

```
## Analysis of Deviance Table
```

```
##
## Model 1: pres.abs ~ 1
## Model 2: pres.abs ~ altitude + distance + NoOfPools + NoOfSites + avrain +
##     meanmin + meanmax
##   Resid. Df Resid. Dev Df Deviance
## 1       211     279.99
## 2       204     198.74  7   81.249
```

The reason for this that because for logistic regression, unlike linear regression, there are multiple tests that for the same test. Furthermore, the `glm` function can fit other models than just the logistic model, and depending on those models, you will want different tests. I can specify a test, and get back a significance value:

```
anova(m0, glmFrogs, test = "LRT")
```

```
## Analysis of Deviance Table
##
## Model 1: pres.abs ~ 1
## Model 2: pres.abs ~ altitude + distance + NoOfPools + NoOfSites + avrain +
##     meanmin + meanmax
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1       211     279.99
## 2       204     198.74  7   81.249 7.662e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Question: What are the conclusions of these tests?

Comparison with Tests of $\hat{\beta}_j$

Notice that unlike linear regression, you get slightly different answers testing the importance of leaving out `NoOfPools` using the anova above and test statistics that come with the summary of the logistic object:

```
anova(m2, glmFrogs, test = "LRT")
```

```
## Analysis of Deviance Table
##
## Model 1: pres.abs ~ altitude + distance + NoOfSites + avrain + meanmin +
##     meanmax
## Model 2: pres.abs ~ altitude + distance + NoOfPools + NoOfSites + avrain +
##     meanmin + meanmax
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       205     210.84
## 2       204     198.74  1   12.101 0.000504 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
cat("Summary results:\n")
```

```
## Summary results:
```

```
round(summary(glmFrogs)$coeff, 4)
```

```
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) 110.4935   138.7622  0.7963   0.4259
## altitude     -0.0309     0.0408 -0.7571   0.4490
## distance     -0.0005     0.0002 -2.3360   0.0195
## NoOfPools     0.0299     0.0093  3.2192   0.0013
## NoOfSites     0.0436     0.1061  0.4114   0.6808
## avrain       -0.0114     0.0599 -0.1901   0.8492
## meanmin       4.8991     1.5637  3.1329   0.0017
## meanmax      -5.6603     5.0488 -1.1211   0.2622
```

They are still testing the same null hypothesis, but they are making different choices about the statistic to use[3]; in linear regression the different choices converge to the same test (the F-statistic for ANOVA is the square of the t-statistic), but this is a special property of normal distribution. Theoretically these two choices are equivalent for large enough sample size; in practice they can differ.

7.4.2 Variable Selection using AIC

Although the Residual Deviance (RD) measures goodness of fit, it cannot be used for variable selection because the full model will have the smallest RD. The AIC however can be used as a goodness of fit criterion (this involves selecting the model with the smallest AIC).

AIC

We can similarly calculate the AIC , only now it is based on the residual deviance,

$$AIC = RD + 2\,(p+1)$$

```
AIC(glmFrogs)
```

```
## [1] 214.7384
```

Based on AIC, we have the same choices as in linear regression. In principle, one can go over all possible submodels and select the model with the smallest value of AIC. But this involves going over 2^p models which might be computationally

[3]The glm summary gives the Wald-statistics, while the anova uses the likelihood ratio statistic.

difficult if p is moderate or large. A useful alternative is to use stepwise methods, only now comparing the change in RD rather than RSS; we can use the same `step` function in R:

```
step(glmFrogs, direction = "both", trace = 0)
```

```
##
## Call:  glm(formula = pres.abs ~ distance + NoOfPools + meanmin + meanmax,
##     family = binomial, data = frogsNoGeo)
##
## Coefficients:
## (Intercept)     distance    NoOfPools      meanmin      meanmax
##  14.0074032   -0.0005138    0.0285643    5.6230647   -2.3717579
##
## Degrees of Freedom: 211 Total (i.e. Null);  207 Residual
## Null Deviance:       280
## Residual Deviance: 199.6     AIC: 209.6
```

7.5 Classification Using Logistic Regression

Suppose that, for a new site, our logistic regression model predicts that the probability that a frog will be found at that site to be $\hat{p}(x)$. What if we want to make a binary prediction, rather than just a probability, i.e. prediction $\hat{y}$ that is a 1 or 0, prediction whether there will be frogs found at that site. How large should $\hat{p}(x)$ be so that we predict that frogs will be found at that site? 0.5 sounds like a fair threshold but would 0.6 be better?

Let us now introduce the idea of a **confusion matrix**. Given any chosen threshold, we can form obtain predictions in terms of 0 and 1 for each of the sample observations by applying the threshold to the fitted probabilities given by logistic regression. The confusion matrix is created by comparing these predictions with the actual observed responses.

	$\hat{y} = 0$	$\hat{y} = 1$
$y = 0$	C_0	W_1
$y = 1$	W_0	C_1

- C_0 denotes the number of observations where we were correct in predicting 0: both the observed response as well as our prediction are equal to zero.
- W_1 denotes the number of observations where were wrong in our predictions of 1: the observed response equals 0 but our prediction equals 1.
- W_0 denotes the number of observations where were wrong in our predictions of 0: the observed response equals 1 but our prediction equals 0.
- C_1 denotes the number of observations where we were correct in predicting 0: both the observed response as well as our prediction are equal to 1.

For example, for the frogs data, if we choose the threshold 0.5, then the entries of the confusion matrix can be calculated as:

```
## Confusion matrix for threshold 0.5:
```

	$\widehat{y}=0$	$\widehat{y}=1$
$y=0$	$C_0=112$	$W_1=21$
$y=1$	$W_0=21$	$C_1=58$

On the other hand, if we use a threshold of 0.3, the numbers will be:

```
## Confusion for threshold 0.3:
```

	$\widehat{y}=0$	$\widehat{y}=1$
$y=0$	$C_0=84$	$W_1=49$
$y=1$	$W_0=10$	$C_1=69$

Note that C_0 and C_1 denote the extent to which the response agrees with our threshold. And W_1 and W_0 measure the extent to which they disagree. An optimal threshold can be chosen to be one which minimizes $W_1 + W_0$. We can compute the entries of the confusion matrix for a range of possible thresholds.

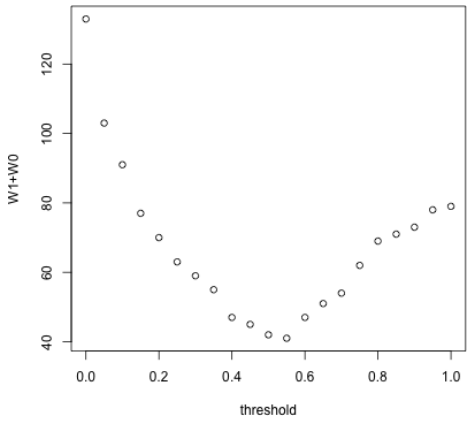
```
##       C0  W1 W0 C1
## 0      0 133  0 79
## 0.05  33 100  3 76
## 0.1   47  86  5 74
## 0.15  61  72  5 74
## 0.2   69  64  6 73
## 0.25  80  53 10 69
## 0.3   84  49 10 69
## 0.35  91  42 13 66
## 0.4  100  33 14 65
## 0.45 106  27 18 61
## 0.5  112  21 21 58
## 0.55 118  15 26 53
## 0.6  121  12 35 44
## 0.65 126   7 44 35
## 0.7  129   4 50 29
## 0.75 130   3 59 20
## 0.8  133   0 69 10
## 0.85 133   0 71  8
## 0.9  133   0 73  6
## 0.95 133   0 78  1
## 1    133   0 79  0
```

Notice that I can get either W_1 *or* W_0 exactly equal to 0 (i.e no misclassifications).

Question: Why is it not a good idea to try to get W_1 exactly equal to 0 (or alternatively try to get W_0 exactly equal to 0)?

We can then plot the value of $W_1 + W_0$ for each value of the threshold in the following plot:

```
plot(thr, conf[, "W1"] + conf[, "W0"], xlab = "threshold",
    ylab = "W1+W0")
```



The smallest value of $W_1 + W_0$ corresponds to the threshold 0.55. It is sensible therefore to use this threshold for predictions.

But there might be settings where you want to allow more of one type of mistake than another. For example, you might prefer to detect a higher percentage of persons with a communicable disease, so as to limit the chances that someone further transmits the disease, even if that means slightly more people are wrongly told that they have the disease. These are trade-offs that frequently have to be made based on domain knowledge.

7.5.1 Example of Spam Dataset

Let us now consider the email spam dataset. Recall the dataset:

```
head(spam7)
```

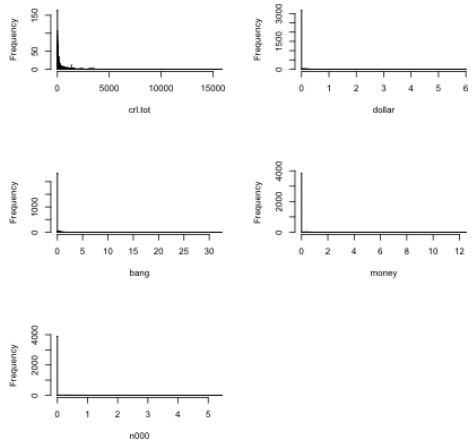
```
##   crl.tot dollar  bang money n000 make yesno
## 1     278  0.000 0.778  0.00 0.00 0.00     y
## 2    1028  0.180 0.372  0.43 0.43 0.21     y
## 3    2259  0.184 0.276  0.06 1.16 0.06     y
## 4     191  0.000 0.137  0.00 0.00 0.00     y
## 5     191  0.000 0.135  0.00 0.00 0.00     y
## 6      54  0.000 0.000  0.00 0.00 0.00     y
```

Before fitting a logistic regression model, let us first look at the summary and histograms of the explanatory variables:

```
summary(spam)
```

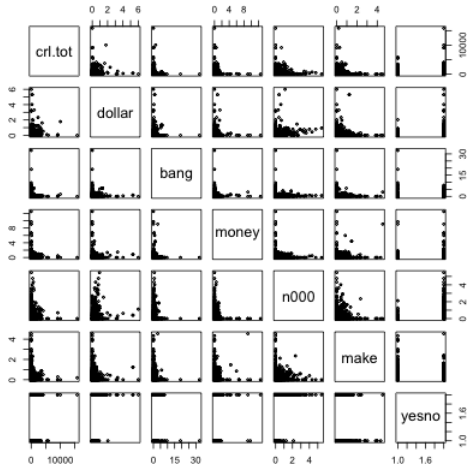
```
##     crl.tot            dollar             bang              money
##  Min.   :    1.0   Min.   :0.00000   Min.   : 0.0000   Min.   : 0.00000
##  1st Qu.:   35.0   1st Qu.:0.00000   1st Qu.: 0.0000   1st Qu.: 0.00000
##  Median :   95.0   Median :0.00000   Median : 0.0000   Median : 0.00000
##  Mean   :  283.3   Mean   :0.07581   Mean   : 0.2691   Mean   : 0.09427
##  3rd Qu.:  266.0   3rd Qu.:0.05200   3rd Qu.: 0.3150   3rd Qu.: 0.00000
##  Max.   :15841.0   Max.   :6.00300   Max.   :32.4780   Max.   :12.50000
##       n000              make        yesno
##  Min.   :0.0000   Min.   :0.0000   n:2788
##  1st Qu.:0.0000   1st Qu.:0.0000   y:1813
##  Median :0.0000   Median :0.0000
##  Mean   :0.1016   Mean   :0.1046
##  3rd Qu.:0.0000   3rd Qu.:0.0000
##  Max.   :5.4500   Max.   :4.5400
```

```
par(mfrow = c(3, 2))
for (i in 1:5) hist(spam[, i], main = "", xlab = names(spam)[i],
    breaks = 10000)
par(mfrow = c(1, 1))
```



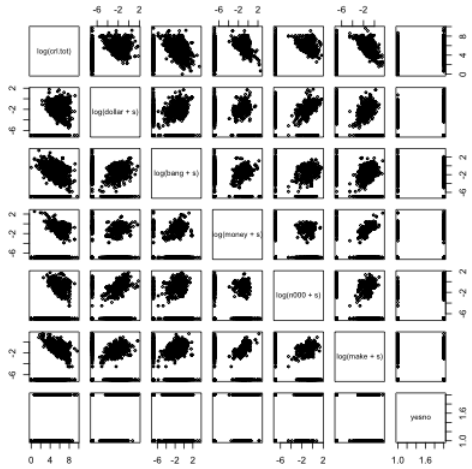
The following is a pairs plot of the variables.

```
pairs(spam, cex = 0.5)
```



It is clear from these plots that the explanatory variables are highly skewed and it is hard to see any structure in these plots. Visualization will be much easier if we take logarithms of the explanatory variables.

```
s = 0.001
pairs(~log(crl.tot) + log(dollar + s) + log(bang +
    s) + log(money + s) + log(n000 + s) + log(make +
    s) + yesno, data = spam, cex = 0.5)
```



We now fit a logistic regression model for *yesno* based on the logged explanatory variables.

```
spam.glm <- glm(yesno ~ log(crl.tot) + log(dollar +
    s) + log(bang + s) + log(money + s) + log(n000 +
    s) + log(make + s), family = binomial, data = spam)
summary(spam.glm)
```

```
##
## Call:
## glm(formula = yesno ~ log(crl.tot) + log(dollar + s) + log(bang +
##     s) + log(money + s) + log(n000 + s) + log(make + s), family = binomial,
##     data = spam)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1657  -0.4367  -0.2863   0.3609   2.7152
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)      4.11947    0.36342  11.335  < 2e-16 ***
## log(crl.tot)     0.30228    0.03693   8.185 2.71e-16 ***
## log(dollar + s)  0.32586    0.02365  13.777  < 2e-16 ***
## log(bang + s)    0.40984    0.01597  25.661  < 2e-16 ***
## log(money + s)   0.34563    0.02800  12.345  < 2e-16 ***
## log(n000 + s)    0.18947    0.02931   6.463 1.02e-10 ***
## log(make + s)   -0.11418    0.02206  -5.177 2.25e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6170.2  on 4600  degrees of freedom
## Residual deviance: 3245.1  on 4594  degrees of freedom
## AIC: 3259.1
##
## Number of Fisher Scoring iterations: 6
```

Note that all the variables are significant. We actually could have fitted a linear model as well (even though the response variable is binary).

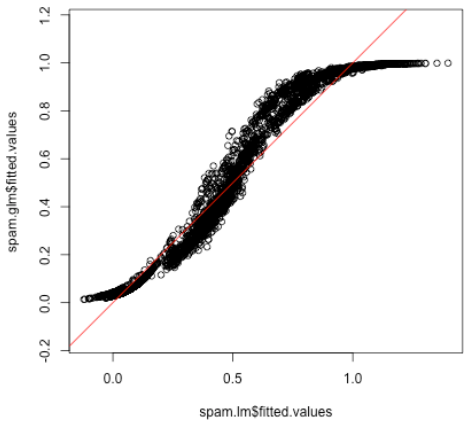
```
spam.lm <- lm(as.numeric(yesno == "y") ~ log(crl.tot) +
    log(dollar + s) + log(bang + s) + log(money + s) +
    log(n000 + s) + log(make + s), data = spam)
summary(spam.lm)
```

```
##
## Call:
## lm(formula = as.numeric(yesno == "y") ~ log(crl.tot) + log(dollar +
##     s) + log(bang + s) + log(money + s) + log(n000 + s) + log(make +
##     s), data = spam)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.10937 -0.13830 -0.05674  0.15262  1.05619
```

```
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       1.078531   0.034188  31.547  < 2e-16 ***
## log(crl.tot)      0.028611   0.003978   7.193 7.38e-13 ***
## log(dollar + s)   0.054878   0.002934  18.703  < 2e-16 ***
## log(bang + s)     0.064522   0.001919  33.619  < 2e-16 ***
## log(money + s)    0.039776   0.002751  14.457  < 2e-16 ***
## log(n000 + s)     0.018530   0.002815   6.582 5.16e-11 ***
## log(make + s)    -0.017380   0.002370  -7.335 2.61e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3391 on 4594 degrees of freedom
## Multiple R-squared:  0.5193, Adjusted R-squared:  0.5186
## F-statistic: 827.1 on 6 and 4594 DF,  p-value: < 2.2e-16
```

A comparison plot of the fitted values for the linear regression and logistic regression is given below.

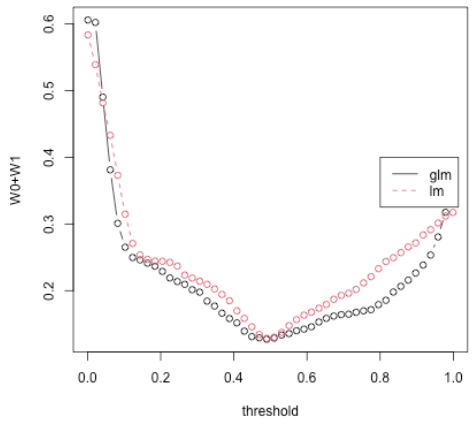
```
par(mfrow = c(1, 1))
plot(spam.lm$fitted.values, spam.glm$fitted.values,
    asp = 1)
abline(c(0, 1), col = "red")
```



Note that some of the fitted values for the linear model are less than 0 and some are more than one. We can formally compare the prediction performance of the linear model and the generalized linear model by the confusion matrix. For various thresholds on the fitted values, the confusion matrices of linear regression and logistic regression can be computed and we can compare their misclassification error

```
v <- seq(0.001, 0.999, length = 50)
y <- as.numeric(spam$yesno == "y")
```

```
glm.conf <- confusion(y, spam.glm$fitted, v)
lm.conf <- confusion(y, spam.lm$fitted, v)
matplot(v, cbind((glm.conf[, "W1"] + glm.conf[, "W0"])/4601,
    (lm.conf[, "W1"] + lm.conf[, "W0"])/4601), xlab = "threshold",
    ylab = "W0+W1", type = "b", pch = 1)
legend(0.8, 0.4, lty = 1:2, col = 1:2, c("glm", "lm"))
```



It is clear from this plot that 0.5 is the best threshold for both linear and logistic regression as the misclassification error is minimized at 0.5. Logistic regression seems to be slightly better than linear regression at other thresholds.

The log-transformation on the explanatory variables is quite important in this case.

To see this, let us perform a logistic regression without the transformations:

```
spam.glm.nolog <- glm(yesno ~ crl.tot + dollar + bang +
    money + n000 + make, family = binomial, data = spam)
summary(spam.glm)
```

```
##
## Call:
## glm(formula = yesno ~ log(crl.tot) + log(dollar + s) + log(bang +
##     s) + log(money + s) + log(n000 + s) + log(make + s), family = binomial,
##     data = spam)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1657  -0.4367  -0.2863   0.3609   2.7152
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)      4.11947    0.36342  11.335  < 2e-16 ***
## log(crl.tot)     0.30228    0.03693   8.185 2.71e-16 ***
```

```
## log(dollar + s)  0.32586    0.02365  13.777  < 2e-16 ***
## log(bang + s)    0.40984    0.01597  25.661  < 2e-16 ***
## log(money + s)   0.34563    0.02800  12.345  < 2e-16 ***
## log(n000 + s)    0.18947    0.02931   6.463 1.02e-10 ***
## log(make + s)   -0.11418    0.02206  -5.177 2.25e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6170.2  on 4600  degrees of freedom
## Residual deviance: 3245.1  on 4594  degrees of freedom
## AIC: 3259.1
##
## Number of Fisher Scoring iterations: 6
```

```
summary(spam.glm.nolog)
```

```
##
## Call:
## glm(formula = yesno ~ crl.tot + dollar + bang + money + n000 +
##     make, family = binomial, data = spam)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -8.4904  -0.6153  -0.5816   0.4439   1.9323
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.700e+00  5.361e-02 -31.717  < 2e-16 ***
## crl.tot      6.917e-04  9.745e-05   7.098 1.27e-12 ***
## dollar       8.013e+00  6.175e-01  12.976  < 2e-16 ***
## bang         1.572e+00  1.115e-01  14.096  < 2e-16 ***
## money        2.142e+00  2.418e-01   8.859  < 2e-16 ***
## n000         4.149e+00  4.371e-01   9.492  < 2e-16 ***
## make         1.698e-02  1.434e-01   0.118    0.906
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6170.2  on 4600  degrees of freedom
## Residual deviance: 4058.8  on 4594  degrees of freedom
## AIC: 4072.8
##
## Number of Fisher Scoring iterations: 16
```

```
spam.lglmFrogs = lm(as.numeric(yesno == "y") ~ crl.tot +
    dollar + bang + money + n000 + make, data = spam)
summary(spam.lglmFrogs)
```

```
##
## Call:
## lm(formula = as.numeric(yesno == "y") ~ crl.tot + dollar + bang +
##     money + n000 + make, data = spam)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.8650 -0.2758 -0.2519  0.4459  0.7499
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2.498e-01  7.488e-03  33.365   <2e-16 ***
## crl.tot     1.241e-04  1.058e-05  11.734   <2e-16 ***
## dollar      3.481e-01  2.733e-02  12.740   <2e-16 ***
## bang        1.113e-01  7.725e-03  14.407   <2e-16 ***
## money       1.765e-01  1.440e-02  12.262   <2e-16 ***
## n000        3.218e-01  1.891e-02  17.014   <2e-16 ***
## make        3.212e-02  2.101e-02   1.529    0.126
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4223 on 4594 degrees of freedom
## Multiple R-squared:  0.2543, Adjusted R-squared:  0.2533
## F-statistic: 261.1 on 6 and 4594 DF,  p-value: < 2.2e-16
```

```
summary(spam.lm)
```

```
##
## Call:
## lm(formula = as.numeric(yesno == "y") ~ log(crl.tot) + log(dollar +
##     s) + log(bang + s) + log(money + s) + log(n000 + s) + log(make +
##     s), data = spam)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.10937 -0.13830 -0.05674  0.15262  1.05619
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      1.078531   0.034188  31.547  < 2e-16 ***
## log(crl.tot)     0.028611   0.003978   7.193 7.38e-13 ***
## log(dollar + s)  0.054878   0.002934  18.703  < 2e-16 ***
```

```
## log(bang + s)    0.064522   0.001919  33.619  < 2e-16 ***
## log(money + s)   0.039776   0.002751  14.457  < 2e-16 ***
## log(n000 + s)    0.018530   0.002815   6.582 5.16e-11 ***
## log(make + s)   -0.017380   0.002370  -7.335 2.61e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3391 on 4594 degrees of freedom
## Multiple R-squared:  0.5193, Adjusted R-squared:  0.5186
## F-statistic: 827.1 on 6 and 4594 DF,  p-value: < 2.2e-16
```

There is a noticeable difference between the two R-squared values.

7.5.2 Trading off different types of errors

We used the quantity $W_0 + W_1$ to quantify how many errors we make. However, these are combining together two different types of errors, and we might care about one type of error more than another. For example, if $y = 1$ if a person has a disease and $y = 0$ if they do not, then we might have different ideas about how much of the two types of error we would want. W_1 are all of the times we say someone has a disease when they don't, while W_0 is the reverse (we say someone has the disease, but they don't).

We've already seen that it's not a good idea to try to drive either W_0 or W_1 to zero (if that was our goal we could just ignore any data and always says someone has the disease, and that would make $W_0 = 0$ since we would never have $\hat{y} = 0$).

Alternatively, we might have a prediction procedure, and want to quantify how good it is, and so we want a vocabulary to talk about the types of mistakes we make.

Recall our types of results:

	pred = 0	pred = 1
obs = 0	C_0	W_1
obs = 1	W_0	C_1

There are two sets of metrics that are commonly used.

1. Precision/Recall

   - **Precision**

     $$P(y = 1|\hat{y} = 1)$$

     We estimate it with the proportion of predictions of $\hat{y} = 1$ that are correct

     $$\frac{\#\text{ correct } \hat{y} = 1}{\#\hat{y} = 1} = \frac{C_1}{C_1 + W_1}$$

- **Recall**

  $$P(\hat{y} = 1|y = 1)$$

  Estimated with proportion of $y = 1$ that are correctly predicted

  $$\frac{\#\text{ correct } \hat{y} = 1}{\#y = 1} \frac{C_1}{C_1 + W_0}$$

2. Sensitivity/Specificity

   - **Specificity** (or true negative rate)

     $$P(\hat{y} = 0|y = 0)$$

     Estimated with the proportion of all $y = 0$ that are correctly predicted

     $$\frac{\#\text{ correct } \hat{y} = 0}{\#\hat{y} = 0} = \frac{C_0}{C_0 + W_1}$$

   - **Sensitivity** (equivalent to Recall or true positive rate)

     $$P(\hat{y} = 1|y = 1)$$

     Estimated with the proportion of all $y = 1$ that are correctly predicted

     $$\frac{\#\text{ correct } \hat{y} = 1}{\#y = 1} = \frac{C_1}{C_1 + W_0}$$

Example: Disease classification

If we go back to our example of $y = 1$ if a person has a disease and $y = 0$ if they do not, then we have:

- **Precision** The proportion of patients classified with the disease that actually have the disease.
- **Recall/Sensitivity** The proportion of diseased patients that will be correctly identified as diseased
- **Specificity** The proportion of non-diseased patients that will be correctly identified as non-diseased

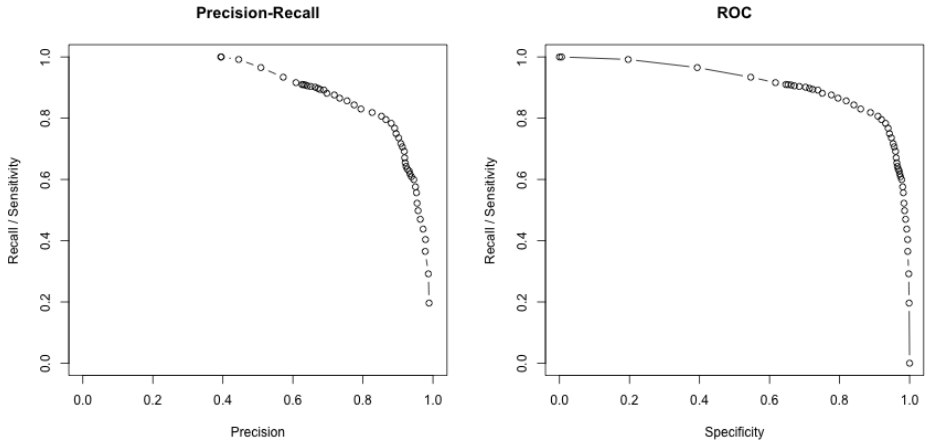
7.5.3 ROC/Precision-Recall Curves

These metrics come in pairs because you usually consider the two pairs together to decide on the right cutoff, as well as to generally compare techniques.

These measures are usually done plotted: Sensitivity plotted against specificity is called a ROC curve ("Receiver operating characteristic" curve); the other plot is just the precision-recall plot.

Here are these curves estimated for our glm model on the spam dataset (note that the points are where we actually evaluated it, and we draw lines between those points to get a curve)

```
spamGlm.precision <- glm.conf[, "C1"]/(glm.conf[, "C1"] +
    glm.conf[, "W1"])
spamGlm.recall <- glm.conf[, "C1"]/(glm.conf[, "C1"] +
    glm.conf[, "W0"])
spamGlm.spec <- glm.conf[, "C0"]/(glm.conf[, "C0"] +
    glm.conf[, "W1"])
par(mfrow = c(1, 2))
plot(x = spamGlm.precision, y = spamGlm.recall, xlab = "Precision",
    ylab = "Recall / Sensitivity", type = "b", xlim = c(0,
        1), ylim = c(0, 1), main = "Precision-Recall")
plot(x = spamGlm.spec, y = spamGlm.recall, ylab = "Recall / Sensitivity",
    xlab = "Specificity", type = "b", xlim = c(0, 1),
    ylim = c(0, 1), main = "ROC")
```

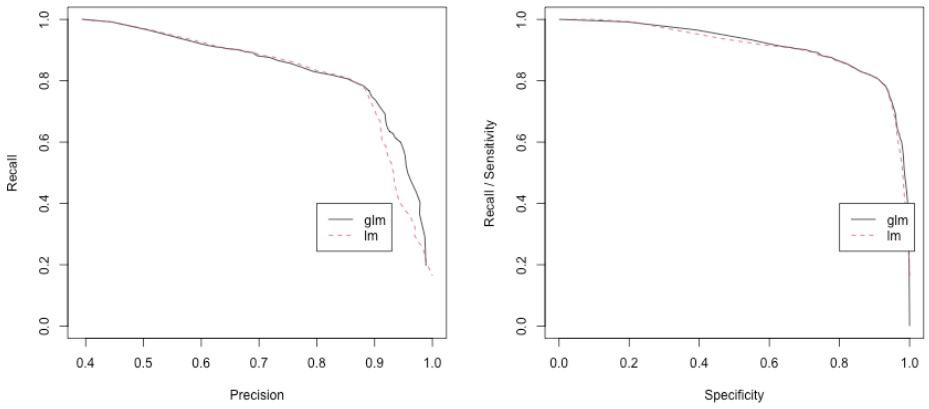


Question: Why would there be a NaN?

We can compare linear and logistic regressions in one plot as follows.

```
spamlm.precision <- lm.conf[, "C1"]/(lm.conf[, "C1"] +
    glm.conf[, "W1"])
spamlm.recall <- lm.conf[, "C1"]/(lm.conf[, "C1"] +
    glm.conf[, "W0"])
spamlm.spec <- lm.conf[, "C0"]/(lm.conf[, "C0"] + glm.conf[,
    "W1"])
par(mfrow = c(1, 2))
matplot(x = cbind(spamGlm.precision, spamlm.precision),
    y = cbind(spamGlm.recall, spamlm.recall), xlab = "Precision",
    ylab = "Recall", type = "l")
legend(0.8, 0.4, lty = 1:2, col = 1:2, c("glm", "lm"))
```

```
matplot(x = cbind(spamGlm.spec, spamlm.spec), y = cbind(spamGlm.recall,
    spamlm.recall), ylab = "Recall / Sensitivity",
    xlab = "Specificity", type = "l")
legend(0.8, 0.4, lty = 1:2, col = 1:2, c("glm", "lm"))
```



Why one versus the other?

Notice that Precision-Recall focuses on the cases of $y = 1$ or $\hat{y} = 1$. This can be useful in cases where your focus is really on how well you detect someone and you are not concerned about how well you are detecting $y = 0$. This is most common if the vast majority of the population has $y = 0$, and you want to detect very rare events when $y = 1$ – these are often problems when you are "trying to find a needle in the haystack", and only care about your ability to find positive results. For example, suppose you want to consider how well a search engine lists of links are correctly related to the topic requested. You can imagine all websites in the world have a true $y_i = 1$ if it would be correct to be listed and the search engine gives a $\hat{y}_i = 1$ to those websites that they will return. Then precision is asking (on average) what proportion of the search engine's list of websites are correct; recall/sensitivity is asking what proportion of all of the $y_i = 1$ websites are found. Both are reasonable questions to try to trade off. Specificity, however, is what proportion of all the other (billion?) websites that are not related (i.e. $y_i = 0$) are NOT given in the search engine's list of good websites. You are not concerned about this quantity at all.

However, in other contexts it matters very much how good you are at separating the negative results (i.e. $y = 0$). Consider predicting if a patient has a disease ($\hat{y}_i = 1$), and then y_i is whether the patient actually has a disease. A negative result tells a patient that the patient doesn't have the disease – and is a serious problem if in fact the patient does have the disease, and thus doesn't get treatment.

Note that the key distinction between these two contexts the reprecussions to

being negative. There are many settings where the use of the predictions is ultimately as a recommendation system (movies to see, products to buy, best times to buy something, potential important genes to investigate, etc), so mislabeling some positive things as negative (i.e. not finding them) isn't a big deal so long as what you do recommend is high quality.

Indeed, the cases where trade-offs lead to different conclusions tend to be cases where the overall proportion of $y_i = 1$ in the population is small. (However, just because they have a small proportion in the population doesn't mean you don't care about making mistakes about negatives – missing a diagnosis for a rare but serious disease is still a problem)

Chapter 8

Regression and Classification Trees

```
## Linking to ImageMagick 6.9.12.3
## Enabled features: cairo, fontconfig, freetype, heic, lcms, pango, raw, rsvg, webp
## Disabled features: fftw, ghostscript, x11
```

Now we are going to turn to a very different statistical approach, called decision trees. This approach is focused on prediction of our outcome y based on covariates x. Unlike our previous regression and logistic regression approaches, decision trees are a much more flexible model and are primarily focused on accurate *prediction* of the y, but they also give a very simple and interpretable model for the data y.

Decision trees, themselves, are not very powerful for predictions. However, when we combine them with ideas of resampling, we can combine together many decision trees (run on different samples of the data) to get what are called **Random Forests**. Random Forests are a pretty powerful and widely-used prediction tool.

8.1 Basic Idea of Decision Trees.

The basic idea behind decision trees is the following: Group the n subjects in our observed data (our **training data**) into a bunch of groups. The groups are defined based on binning the explanatory variables (x) of the observed data, and the bins are picked so that the observed data in a bin have similar outcomes y.

Prediction for a future subject is then done in the following way. Look at the explanatory variable values for the future subject to figure into which binned

values of the x the observation belongs. Then predict the future response based on the responses in the observed data that were in that group.

When the output y is continuous, we call it **regression trees**, and we will predict a future response based on the mean of the training data in that group/bin. If the outcome y is binary we call this technique **classification trees**. Just like with regression and logistic regression, there are important distinctions in how the model is built for continuous and binary data, but there is a general similarity in the approach.

8.2 The Structure of Decision Trees

The main thing to understand here is how the grouping of the data into groups is constructed. Let's return to the `bodyfat` data from our multiple regression chapter.

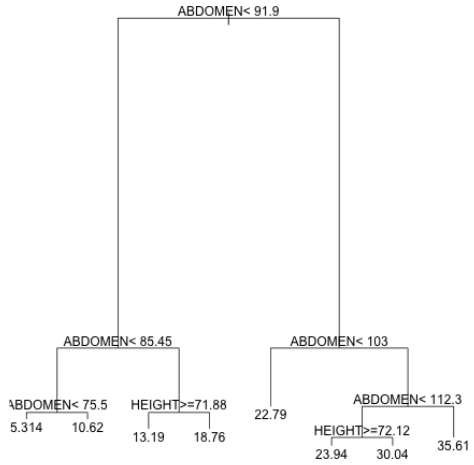
The groups of data are from partitioning (or binning) the x covariates in the training data. For example, one group of data in our training data could be observations that meet all of the following criterion:

- `HEIGHT`>72.5
- $91<$`ABDOMEN`<103
- $180<$`WEIGHT`<200

Notice that this group of observations is constructed by taking a simple range for each of the variables used. This is the partitioning of the x data, and decision trees limit themselves to these kind of groupings of the data. The particular values for those ranges are picked, as we said before, based on what best divides the training data so that the response y is similar.

Why Trees?

The reason these are called decision trees, is that you can describe the rules for how to put an observation into one of these groups based on a simple **decision tree**.



How to interpret this tree? This tree defines all of the possible groups based on the explanatory variables. You start at the top node of the tree. At each node of the tree, there is a condition involving a variable and a cut-off. If the condition is met, then we go left and if it is not met, we go right. The bottom "terminal nodes" or "leaves" of the tree correspond to the groups. So for example, consider an individual who is 30 years of age, 180 pounds in weight, 70 inches tall and whose chest circumference is 95 cm, abdomen circumference is 90 cm, hip circumference is 100 cm and thigh circumference is 60 cm. The clause at the top of the tree is "`ABDOMEN` < 91.9" which is met for this person, so we move left. We then encounter the clause "`ABDOMEN` < 85.45" which is not met so we move right. This leads to the clause "`HEIGHT` $>= 71.88$" which is true for this person. So we move left. We then hit a terminal node, so this defines the group for this individual. Putting all those conditions together, we have that individuals in this group are defined by

- $85.45 \leq$ `ABDOMEN` < 91.9
- `HEIGHT` ≥ 71.88

There is a displayed value for this group of 13.19 – this is the predicted value for individuals in this group, namely the mean of the training data that fell into this group.

Consider another terminal node, with the displayed (predicted) value of 30.04. What is the set of conditions that describes the observations in this group?

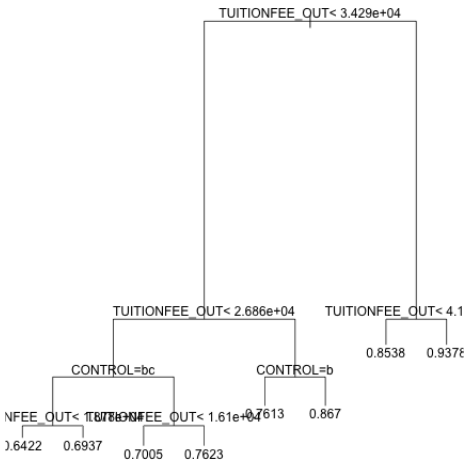
8.2.1 How are categorical explanatory variables dealt with?

For a categorical explanatory variable, it clearly does not make sense to put a numerical cut-off across its value. For such variables, the groups (or splits) are created by the possible combinations of the levels of the categorical variables.

Specifically, suppose X_j is a categorical variable that one of k values given by: $\{a_1, \ldots, a_k\}$. Then possible conditions in the node of our tree are given by subsets of these k values; the condition is satisfied if the value of X_j for the observation is in this subset: go left if $X_j \in S$ and go right if $X_j \notin S$.

Here is an example with categorical explanatory variables from our college dataset. The variable `CONTROL` corresponded to the type of college (private, public, or for profit)

Note that `CONTROL` is a categorical variable. Here is a decision tree based on the college data:



Note the conditions `CONTROL = bc` and `CONTROL = b` appearing in the tree. Unfortunately the plotting command doesn't actually give the names of the levels in the tree, but uses "a","b",... for the levels. We can see the levels of `CONTROL`:

```
levels(scorecard$CONTROL)
```

```
## [1] "public"     "private"    "for-profit"
```

So in `CONTROL = b`, "b" corresponds to the second level of the variable, in this case "private". So it is really `CONTROL="private"`. `CONTROL = bc` corresponds `CONTROL` being *either* "b" or "c", i.e. in either the second OR third level of `CONTROL`. This translates to observations where `CONTROL` is either "private"‘ OR "for-profit". (We will also see when we look at the R command that is creating these trees below that you can work with the output to see this information better, in case you forget.)

What are the set of conditions that define the group with prediction 0.7623?

8.3 The Recursive Partitioning Algorithm

Finding the "best" such grouping or partitioning is a computationally challenging task, regardless of how we define "best". In practice, a greedy algorithm, called *Recursive Partitioning*, is employed which produces a reasonable grouping, albeit not guaranteeing to be the best grouping.

8.3.1 Fitting the Tree in R

Let's first look at how we create the above trees in R. Recursive Partitioning is done in R via the function `rpart` from the library `rpart`.

Let us first use the *rpart* function to fit a regression tree to the bodyfat dataset. We will use `BODYFAT` as our response, and explanatory variables Age, Weight, Height, Chest, Abdomen, Hip and Thigh. This is, in fact, the code that gave us the tree above.

```
library(rpart)
rt = rpart(BODYFAT ~ AGE + WEIGHT + HEIGHT + CHEST +
    ABDOMEN + HIP + THIGH, data = body)
```

Notice we use a similar syntax as `lm` and `glm` to define the variable that is the response and those that are the explanatory variables.

In addition to plotting the tree, we can look at a textual representation (which can be helpful if it is difficult to see all of the tree or you want to be sure you remember whether you go right or left)

```
print(rt)
```

```
## n= 252
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 252 17578.99000 19.150790
##    2) ABDOMEN< 91.9 132  4698.25500 13.606060
##      4) ABDOMEN< 85.45 66  1303.62400 10.054550
##        8) ABDOMEN< 75.5 7   113.54860  5.314286 *
##        9) ABDOMEN>=75.5 59  1014.12300 10.616950 *
##      5) ABDOMEN>=85.45 66  1729.68100 17.157580
##       10) HEIGHT>=71.875 19   407.33790 13.189470 *
##       11) HEIGHT< 71.875 47   902.23110 18.761700 *
##    3) ABDOMEN>=91.9 120  4358.48000 25.250000
##      6) ABDOMEN< 103 81  1752.42000 22.788890 *
##      7) ABDOMEN>=103 39  1096.45200 30.361540
##       14) ABDOMEN< 112.3 28   413.60000 28.300000
```

```
##         28) HEIGHT>=72.125 8    89.39875 23.937500 *
##         29) HEIGHT< 72.125 20   111.04950 30.045000 *
##       15) ABDOMEN>=112.3 11   260.94910 35.609090 *
```

Note that the tree here only uses the variables Abdomen and Height even though we gave many other variables. The other variables are not being used. This is because the algorithm, which we will discuss below, does variable selection in choosing the variables to use to split up observations.

Interaction Terms

You should note that `rpart` gives an error if you try to put in interaction terms. This is because interaction is intrinsically included in decision trees trees. You can see this by thinking about what an interaction is in our regression framework: giving a different coefficient for variable X_j based on what the value of another variable X_k is. For example, in our college data, a different slope for the variable `TUITIONFEE_OUT` based on whether the college is private or public is an interaction between `TUITIONFEE_OUT` and `CONTROL`.

Looking at our decision trees, we can see that the groups observations are put in also have this property – the value of `TUITIONFEE_OUT` that puts you into one group will also depend on the value of `CONTROL`. This is an indication of how much more flexible decision trees are in their predictions than linear regression.

8.3.2 How is the tree constructed?

How does `rpart` construct this tree? Specifically, at each node of the tree, there is a condition involving a variable and a cut-off. How does `rpart` choose the variable and the cut-off?

The first split

Let us first understand how the first condition is selected at the top of the tree, as this same process is repeated iteratively.

We're going to assume that we have a continuous response (we'll discuss variations to this procedure for binary response in the next section).

Our condition is going to consist of a variable and a cutoff c, or if the variable is categorical, a subset S of levels. For simplicity of notation, let's just assume that we are looking only at numerical data so we can assume for each condition we need to find a variable j and it's corresponding cutoff c, i.e. the pair (j, c). Just remember for categorical variables it's really (j, S).

We are going to consider each possible variable and a possible cutoff c find the best (j, c) pair for dividing the data into two groups. Specifically, each (j, c) pair, can divide the subjects into two groups:

- G_1 given by observations with $X_j \leq c$

- G_2 given by observations with $X_j > c$.

Then we need to evaluate which (j, c) pair gives the best split of the data.

For any particular split, which defines groups G_1 and G_2, we have a predicted value for each group, $\hat{y}_1$ and $\hat{y}_2$, corresponding to the mean of the observations in group G_1 and G_2 (i.e. $\bar{y}_1$ and $\bar{y}_2$). This means that we can calculate the loss (or error) in our prediction for each observation. Using standard squared-error loss, this gives us the RSS for the split defined by (j, c):

$$RSS(j, c) := \sum_{i \in G_1} (y_i - \bar{y}_1)^2 + \sum_{i \in G_2} (y_i - \bar{y}_2)^2.$$

To find the best split, then, we compare the values $RSS(j, c)$ and pick the value of j and c for which $RSS(j, c)$ is the smallest.

Further splits

The above procedure gives the first node (or split) of the data. The same process continues down the tree, only now with a smaller portion of the data.

Specifically, the first node split the data into two groups G_1 and G_2. The next step of the algorithm is to repeat the same process, only now with only the data in G_1 and G_2 separately. Using the data in group G_1 you find the variable X_j and cutoff c that divides the observations in G_1 into two groups G_{11} and G_{12}. You find that split by determining the pair (j, c) with the smallest RSS, just like before. And similarly the observations in G_2 are split into two by a different pair (j, c), obtaining groups G_{21} and G_{22}.

This process continues, continuing to split the current sets of groups into two each time.

Measuring the improvement due to the split

Just like in regression, the improvement in fit can be quantified by comparing the error you get from adding variables (RSS) to the error you would have if you just used the group mean (TSS). This same principle applies here. For each split (j, c), the smallest RSS,

$$\min_{j,c} RSS(j, c)$$

can be compared with the to the total variability in the data before splitting

$$TSS = \sum_i (y_i - \bar{y})^2.$$

Notice that TSS here is only calculated on the current set of observations in the group you are trying to split.

The ratio

$$\frac{\min_{j,c} RSS(j, c)}{TSS}$$

is always smaller than 1 and the smaller it is, the greater we are gaining by the split.

For example, for the bodyfat dataset, the total sum of squares before any splitting is 17578.99. After splitting based on "Abdomen < 91.9", one gets two groups with residuals sums of squares given by 4698.255 and 4358.48. Therefore the reduction in the sum of squares is:

```
(4698.255 + 4358.48)/17578.99
```

```
## [1] 0.5152022
```

The reduction in error due to this split is therefore 0.5152. This is the greatest reduction possible by splitting the data into two groups based on a variable and a cut-off.

In the visualization of the decision tree, the length of the branches in the plot of the tree are proportional to the reduction in error due to the split. In the bodyfat dataset, the reduction in sum of squares due to the first split was 0.5152. For this dataset, this is apparently a big reduction compared to subsequence reductions and this is why it is plotted with such a long branch down to subsequent splits (a common phenomena).

For every regression tree T, we can define its global RSS in the following way. Let the *final* groups generated by T be $G_1, \dots, G_k$. Then the RSS of T is defined as

$$RSS(T) := \sum_{j=1}^{m} \sum_{i \in G_j} \left(y_i - \bar{y}_j\right)^2$$

where $\bar{y}_1, \dots, \bar{y}_m$ denote the mean values of the response in each of the groups.

We can also define a notion of R^2 for the regression tree as:

$$R^2(T) := 1 - \frac{RSS(T)}{TSS}.$$

```
1 - (sum(residuals(rt)^2))/(sum((body$BODYFAT - mean(body$BODYFAT))^2))
```

```
## [1] 0.7354195
```

8.3.3 Tree Size and Pruning

Notice that as we continue to recursively split our groups, we have less and less data each time on which to decide how to split the data. In principle we could keep going until each group consisted of a single observation! Clearly we don't want to do that, which brings us to the biggest and most complicated issue for decision trees. How large should the tree be "grown"? Very large trees obviously lead to over-fitting, but insufficient number of splits will lead to poor prediction. We've already seen a similar over-fitting phenomena in regression,

where the more variables you include the better the fit will be on your training data. Decision trees are have a similar phenomena only it is based on how big the tree is – bigger trees will fit the training data better but may not generalize to new data well creating over-fitting.

How is `rpart` deciding when to stop growing the tree?

In regression we saw that we could make this choice via cross-validation – we fit our model on a portion of the tree and then evaluated it on the left out portion. This is more difficult to conceptualize for trees. Specifically, with regression, we could look at different a priori submodels (i.e. subset of variables), fit the submodels to our random subsets of data, and calculate the cross-validation error for each submodel to choose the best one. For our decision trees, however, what would be our submodels be? We could consider different variables as input, but this wouldn't control the size of the tree, which is a big source of over-fitting.

One strategy is to instead stop when the improvement

$$\frac{\min_{(j,c)} RSS(j,c)}{TSS}$$

is not very large. This would be the case when we are not gaining all that much by splitting further. This is actually not a very smart strategy. Why? Because you can actually sometimes split the data and get small amount of improvements, but because you were able to split the data there, it allows you to make another split later that adds a lot of improvement. Stopping the first time you see a small improvement would keep you from discovering that future improvement.

Regression and classification trees were invented by Leo Breiman from UC Berkeley. He also had a different approach for the tree size issue. He advocated against stopping the recursive partitioning algorithm. Instead, he recommends growing a **full tree** (or a very large tree), $T_{\max}$, and then "pruning" back $T_{\max}$ by cutting back lower level groupings. This allows you to avoid situations like above, where you miss a great split because of an early unpromising split. This "pruned" tree will be a subtree of the full tree.

How to Prune

The idea behind pruning a tree is to find a measure of how well a smaller (prunned) tree is fitting the data that doesn't suffer from the issue that larger trees will always fit the training data better. If you think back to variable selection in regression, in addition to cross-validation, we also have measures in addition to cross-validation, like CP, AIC, and BIC, that didn't involve resampling, but had a form

$$R(\alpha) = RSS + \alpha k$$

In other words, use RSS as a measure of fit, but penalize models with a large number of variables k by adding a term αk. Minimizing this quantity meant

that smaller models with good fits could have a value $R(\alpha)$ that was lower than bigger models.

Breiman proposed a similar strategy for measuring the fit of a potential subtree for pruning, but instead penalizing for the size of the tree rather than the number of variables. Namely, for a possible subtree T, define

$$R_\alpha(T) := RSS(T) + \alpha(TSS)|T|$$

where $|T|$ is the number of terminal nodes of the tree T. $R_\alpha(T)$ is evaluated for all the possible subtrees, and the subtree with the smallest $R_\alpha(T)$ is chosen. Since it depends on α, we will call the subtree that minimizes $R_\alpha(T)$ $T(\alpha)$.

Obviously the number of possible subtrees and possible values of α can be large, but there is an algorithm (*weakest link cutting*) that simplifies the process. In fact it can be shown that only a fixed number of α_k values and the corresponding optimal $T(\alpha_k)$ subtrees need to be considered. In other words you don't need to consider all α values, but only a fixed set of α_k values and compare the fit of their optimal $T(\alpha_k)$ subtree. After obtaining this sequence of trees $T(\alpha_1), T(\alpha_2), ...$, the default choice in R is to take $\alpha* = 0.01$ and then generating the tree $T(\alpha_k)$ for the α_k closest to $\alpha*$.[1] The value of α^* is set by the argument `cp` in `rpart`.

The *printcp*() function in R gives those fixed α_k values for this data and also gives the number of splits of the subtrees $T(\alpha_k)$ for each k:

```
printcp(rt)
```

```
##
## Regression tree:
## rpart(formula = BODYFAT ~ AGE + WEIGHT + HEIGHT + CHEST + ABDOMEN +
##     HIP + THIGH, data = body)
##
## Variables actually used in tree construction:
## [1] ABDOMEN HEIGHT
##
## Root node error: 17579/252 = 69.758
##
## n= 252
##
##         CP nsplit rel error  xerror     xstd
## 1 0.484798      0   1.00000 1.00618 0.081161
## 2 0.094713      1   0.51520 0.56445 0.048861
## 3 0.085876      2   0.42049 0.50963 0.045123
## 4 0.024000      3   0.33461 0.39540 0.033100
## 5 0.023899      4   0.31061 0.40685 0.033483
## 6 0.012125      5   0.28672 0.37517 0.031735
## 7 0.010009      6   0.27459 0.38122 0.030388
```

[1] specifically the α_k such that $\alpha_k \leq \alpha* < \alpha_{k-1}$

```
## 8 0.010000       7   0.26458 0.38243 0.030431
```

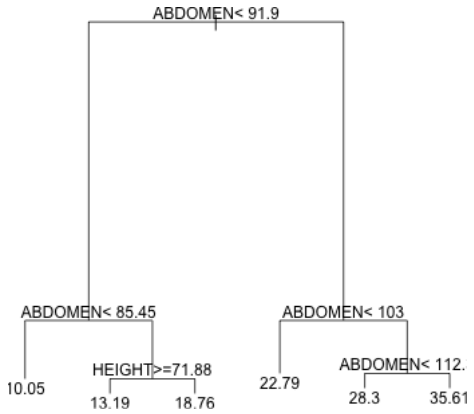
Each row in the `printcp` output corresponds to a different tree $T(\alpha_k)$. Note that each tree has an increasing number of splits. This is a property of the $T(\alpha_k)$ values, specifically that the best trees for each α_k value will be nested within each other, so going from α_k to α_{k+1} corresponds to adding an additional split to one of the terminal nodes of $T(\alpha_k)$.

Also given in the `printcp` output are three other quantities:

- `rel error`: for a tree T this simply $RSS(T)/TSS$. Because more deep trees have smaller RSS, this quantity will always decrease as we go down the column.
- `xerror`: an accuracy measure calculated by 10-fold cross validation (and then divided by TSS). Notice before we mentioned the difficult in conceptualizing cross-validation. But now that we have the complexity parameter α, we can use this for cross-validation. Instead of changing the number of variables k and comparing the cross-validated error, we can change values of α, fit the corresponding tree on random subsets of the data, and evaluate the cross-validated error as to which value of α is better. Notice that this quantity will be random (i.e., different runs of *rpart*() will result in different values for `xerror`); this is because 10-fold cross-validation relies on randomly partitioning the data into 10 parts and the randomness of this partition results in `xerror` being random.
- `xstd`: The quantity `xstd` provides a standard deviation for the random quantity `xerror`. If we do not like the default choice of 0.01 for α, we can choose a higher value of α using *xerror* and *xstd*.

For this particular run, the `xerror` seems to be smallest at $\alpha = 0.012125$ and then `xerror` seems to increase. So we could give this value to the argument `cp` in `rpart` instead of the default `cp = 0.01`.

```
rtd = rpart(BODYFAT ~ AGE + WEIGHT + HEIGHT + CHEST +
    ABDOMEN + HIP + THIGH, data = body, cp = 0.0122)
plot(rtd)
text(rtd)
```



We will then get a smaller tree. Now we get a tree with 5 splits or 6 terminal nodes.

However, we would also note that `xstd` is around $0.032 = 0.033$, so it's not clear that the difference between the `xerror` values for the different α values is terribly meaningful.

8.3.4 Classification Trees

The partitioning algorithm for classification trees (i.e. for a 0-1 response) is the same, but we need to make sure we have an appropriate measurement for deciding on which split is best at each iteration, and there are several to choose from. We can still use the RSS for binary response, which is the default in R, in which case it has a useful simplification that we will discuss.

Specifically, as in the case of regression trees, we need to find the pair (j, c) corresponding to a variable X_j and a cut-off c. (or a pair (j, S) for variables X_j that are categorical). Like regression trees, the pair (j, c) divides the observations into the two groups G_1 (where $X_j \leq c$) and G_2 (where $X_j > c$), and we need to find the pair (j, c) that gives the best fit to the data. We will go through several measures.

8.3.4.1 RSS / Gini-Index

We can use the RSS as before,

$$RSS(j, c) := \sum_{i \in G_1} (y_i - \bar{y}_1)^2 + \sum_{i \in G_2} (y_i - \bar{y}_2)^2$$

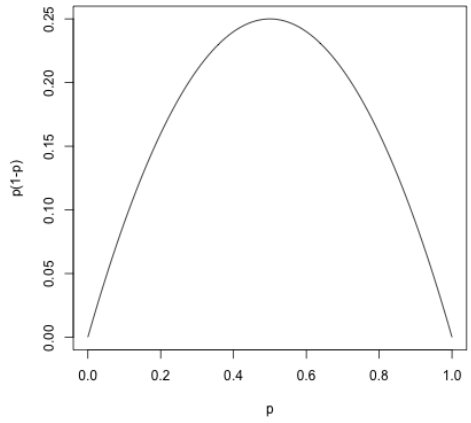
where $\bar{y}_1$ and $\bar{y}_2$ denote the mean values of the response in the groups G_1 and G_2 respectively. Since in classification problems the response values are 0 or 1, $\bar{y}_1$ equals the proportion of ones in G_1 while $\bar{y}_2$ equals the proportion of ones in

G_2. It is therefore better to denote $\bar{y}_1$ and $\bar{y}_2$ by $\hat{p}_1$ and $\hat{p}_2$ respectively, so that the formula for $RSS(j, c)$ then simplifies to:

$$RSS(j, c) = n_1\hat{p}_1(1 - \hat{p}_1) + n_2\hat{p}_2(1 - \hat{p}_2).$$

This quantity is also called the **Gini index** of the split corresponding to the pair (j, c).

Notice that the Gini index involves calculating the function $p(1 - p)$ for each group's proportion of 1's:



This function takes its largest value at $p = 1/2$ and it is small when p is close to 0 or 1.

Therefore the quantity

$$n_1\hat{p}_1(1 - \hat{p}_1)$$

is small if either most of the response values in the group G_1 are 0 (in which case $\hat{p}_1$ is close to 0) or when most of the response values in the group are 1 (in which case $\hat{p}_1 \approx 1$).

A group is said to be pure if either most of the response values in the group are 0 or if most of the response values are 1. Thus the quantity $n_1\hat{p}_1(1 - \hat{p}_1)$ measures the impurity of a group. If $n_1\hat{p}_1(1-\hat{p}_1)$ is low, then the group is pure and if it is high, it is impure. The group is maximally impure if $\hat{p}_1 = 1/2$.

The Gini Index (which is $RSS(j, c)$), is the sum of the impurities of the groups defined by the split given by $X_j \leq c$ and $X_j > c$. So that for binary data, the recursive partitioning algorithm determines j and c as the one that divides the observations into two groups with high amount of purity.

8.3.4.2 Other measures

The quantity $n\hat{p}(1-\hat{p})$ is not the only function used for measuring the impurity of a group in classification. The key property of the function $p(1 - p)$ is that

it is symmetric about 1/2, takes its maximum value at 1/2 and it is small near the end points $p = 0$ and $p = 1$. Two other functions having this property are also commonly used:

- **Cross-entropy or Deviance**: Defined as

$$-2n\left(\hat{p}\log\hat{p} + (1-\hat{p})\log(1-\hat{p})\right).$$

  This also takes its smallest value when $\hat{p}$ is 0 or 1 and it takes its maximum value when $\hat{p} = 1/2$. We saw this value when we did logistic regression, as a measure of the fit.

- **Misclassification Error**: This is defined as

$$n\min(\hat{p}, 1-\hat{p}).$$

  This quantity equals 0 when $\hat{p}$ is 0 or 1 and takes its maximum value when $\hat{p} = 1/2$.

  This is value is called misclassification error based on prediction using a **majority rule** decision for prediction. Specifically, assume we predict the response for an observation in group G based on the which response is seen the most in group G. Then the number of observations that are misclassified by this rule will be equal to $n\min(\hat{p}, 1-\hat{p})$.

One can use Deviance or Misclassification error instead of the Gini index while growing a classification tree. The default in R is to use the Gini index.

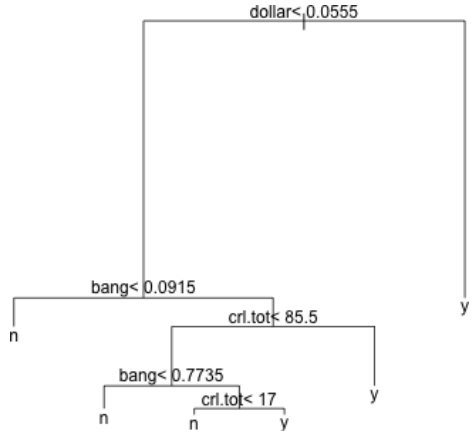
8.3.4.3 Application to spam email data

Let us apply the classification tree to the email spam dataset from the chapter on logistic regression.

```
library(DAAG)
data(spam7)
spam = spam7
```

The only change to the `rpart` function to classification is to use the argument `method = "class"`.

```
sprt = rpart(yesno ~ crl.tot + dollar + bang + money +
    n000 + make, method = "class", data = spam)
plot(sprt)
text(sprt)
```



The tree construction works exactly as in the regression tree. We can look at the various values of the α_k parameter and the associated trees and errors using the function `printcp`.

```
printcp(sprt)
```

```
##
## Classification tree:
## rpart(formula = yesno ~ crl.tot + dollar + bang + money + n000 +
##     make, data = spam, method = "class")
##
## Variables actually used in tree construction:
## [1] bang    crl.tot dollar
##
## Root node error: 1813/4601 = 0.39404
##
## n= 4601
##
##         CP nsplit rel error  xerror     xstd
## 1 0.476558      0   1.00000 1.00000 0.018282
## 2 0.075565      1   0.52344 0.56977 0.015611
## 3 0.011583      3   0.37231 0.39548 0.013570
## 4 0.010480      4   0.36073 0.39217 0.013523
## 5 0.010000      5   0.35025 0.38610 0.013437
```

Notice that the `xerror` seems to decrease as `cp` decreases. We might want to set the `cp` to be lower than 0.01 so see how the `xerror` changes:

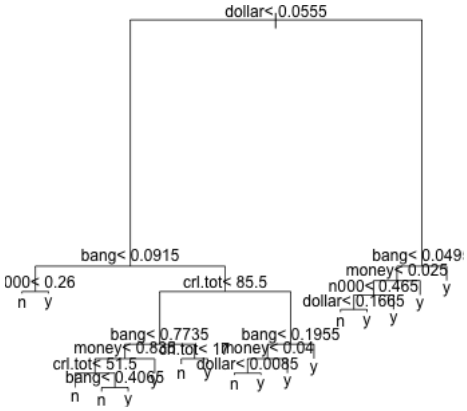
```
sprt = rpart(yesno ~ crl.tot + dollar + bang + money +
    n000 + make, method = "class", cp = 0.001, data = spam)
printcp(sprt)
```

```
##
```

```
## Classification tree:
## rpart(formula = yesno ~ crl.tot + dollar + bang + money + n000 +
##     make, data = spam, method = "class", cp = 0.001)
##
## Variables actually used in tree construction:
## [1] bang    crl.tot dollar  money   n000
##
## Root node error: 1813/4601 = 0.39404
##
## n= 4601
##
##           CP nsplit rel error  xerror     xstd
## 1  0.4765582      0   1.00000 1.00000 0.018282
## 2  0.0755654      1   0.52344 0.54881 0.015403
## 3  0.0115830      3   0.37231 0.38941 0.013484
## 4  0.0104799      4   0.36073 0.38555 0.013429
## 5  0.0063431      5   0.35025 0.36569 0.013139
## 6  0.0055157     10   0.31660 0.35025 0.012904
## 7  0.0044126     11   0.31109 0.34253 0.012784
## 8  0.0038610     12   0.30667 0.33867 0.012723
## 9  0.0027579     16   0.29123 0.32377 0.012482
## 10 0.0022063     17   0.28847 0.32212 0.012455
## 11 0.0019305     18   0.28627 0.32488 0.012500
## 12 0.0016547     20   0.28240 0.31771 0.012381
## 13 0.0010000     25   0.27413 0.31826 0.012391
```

Now the minimum `xerror` seems to be the tree with 16 splits (at `cp = 0.0027`). A reasonable choice of `cp` here is therefore 0.0028. We can refit the classification tree with this value of `cp`:

```
sprt = rpart(yesno ~ crl.tot + dollar + bang + money +
    n000 + make, method = "class", cp = 0.0028, data = spam)
plot(sprt)
text(sprt)
```



Predictions for Binary Data

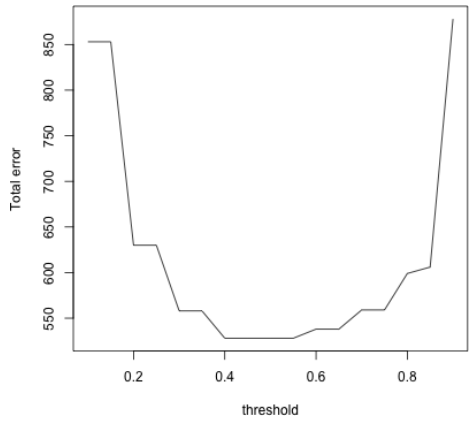
Let us now talk about getting predictions from the classification tree. Prediction is obtained in the usual way using the `predict` function. The `predict` function results in predicted probabilities (not 0-1 values). Suppose we have an email where `crl.tot = 100`, `dollar = 3`, `bang = 0.33`, `money = 1.2`, `n000 = 0` and `make = 0.3`. Then the predicted probability for this email being spam is given by:

```
x0 = data.frame(crl.tot = 100, dollar = 3, bang = 0.33,
    money = 1.2, n000 = 0, make = 0.3)
predict(sprt, newdata = x0)
```

```
##             n        y
## 1 0.04916201 0.950838
```

The predicted probability is 0.950838. If we want to convert this into a 0-1 prediction, we can do this via a confusion matrix in the same way as for logistic regression.

```
y = as.numeric(spam$yesno == "y")
y.hat = predict(sprt, spam)[, 2]
v <- seq(0.1, 0.9, by = 0.05)
tree.conf = confusion(y, y.hat, thres = v)
plot(v, tree.conf[, "W1"] + tree.conf[, "W0"], xlab = "threshold",
    ylab = "Total error", type = "l")
```



It seems that it is pretty equivalent between $0.4 - 0.6$, so it is seems the simple choice of 0.5 is reasonable. This would give the following confusion matrix:

	$\hat{y} = 0$	$\hat{y} = 1$
$y = 0$	$C_0 = 2624$	$W_1 = 164$
$y = 1$	$W_0 = 364$	$C_1 = 1449$

8.4 Random Forests

Decision trees are very simple and intuitive, but they often do not perform well in prediction compare to other techniques. They are too variable, with the choice of variable X_j and the cutoff c changing a good deal with small changes in the data. However, decisions trees form the building blocks for a much better technique called **random forests**. Essentially a random forest is a collection of decision trees (either regression or classification trees depending on the type of response).

The idea behind random forests is to sample from your training data (like in the bootstrap) to create new datasets, and fit decision trees to each of these resampled data. This gives a large number of decision trees, from similar but not exactly the same data. Then the prediction of a new observation is based on combining the predictions of all these trees.[2]

8.4.1 Details of Constructing the Random Trees

We will construct B total trees. The method for constructing the b^{th} tree (for $b = 1, ..., B$) is the following:

[2]This is an example of an *ensemble method*, where many models are fit on the data, or variations of the data, and combined together to get a final prediction.

1. Generate a new dataset having n observations by resampling uniformly at random with replacement from the existing set of observations. This resampling is the same as in bootstrap. Of course, some of the original set of observations will be repeated in this bootstrap sample (because of with replacement draws) while some other observations might be dropped altogether. The observations that do not appear in the bootstrap are referred to as *out of bag* (o.o.b) observations.
2. Construct a decision tree based on the bootstrap sample. This tree construction is almost the same as the construction underlying the `rpart` function but with two important differences:

- **Random selection of variables** At each stage of splitting the data into groups, k number of variables are selected at random from the available set of p variables and only splits based on these k variables are considered. In contrast, in `rpart`, the best split is chosen by considering possible splits from all p explanatory variables and all thresholds.

  So it can happen, for example, that the first split in the tree is chosen from variables $1, 2, 3$ resulting in two groups G_1 and G_2. But then in splitting the group G_1, the split might chosen from variables $4, 5, 6$ and in further splitting group G_2, the next split might be based on variables $1, 5, 6$ and so on.

  The rationale behind this random selection of variables is that often covariates are highly correlated with each other, and the choice of using one X_j versus a variable X_k is likely to be due to training observations you have. Indeed we've seen in the body fat data, that the variables are highly correlated with each other. On future data, a different variable X_k might perform better. So by actually forcing the tree to explore not always relying on X_j, you are more likely to give good predictions for future data that may not match your training data.

- **No "pruning" of trees** We discussed above how to choose the depth or size of a tree, noting that too large of a tree results in a tree with a lot of variability, as your groups are based on small sample sizes. However, the tree construction in random forests the trees are actually grown to full size. There is no pruning involved, i.e. no attempt to find the right size tree. More precisely, each tree is grown till the number of observations in each terminal node is no more than a size m. This, of course, means that each individual tree will overfit the data. However, each individual tree will overfit in a different way and when we average the predictions from different trees, the overfitting will be removed.

At the end, we will have B trees. These B trees will all be different because each tree will be based on a different bootstrapped dataset and also because of our randomness choice of variables to consider in each split. The idea is that these different models, that might be roughly similar, when put together will fit future data more robustly.

Prediction now works in the following natural way. Given a new observation with explanatory variable values $x_1, \ldots, x_p$, each tree in our forest will yield a prediction for the response of this new observation. Our final prediction will simply take the average of the predictions of the individual trees (in case of regression trees) or the majority vote of the predictions of the individual trees in case of classification.

8.4.2 Application in R

We shall use the R function `randomForest` (in the package `randomForest`) for constructing random forests. The following important parameters are

- `ntree` corresponding to B, the number of trees to fit. This should be large (default choice is 500)
- `mtry` corresponding to k, the number of random variables to consider at each split (whose default choice is $p/3$)
- `nodesize` corresponding to m, the maximum size allowed for any terminal node (whose default size is 5)

Let us now see how random forests work for regression in the bodyfat dataset.

The syntax for the `randomForest` function works as follows:

```
library(randomForest)
ft = randomForest(BODYFAT ~ AGE + WEIGHT + HEIGHT +
    CHEST + ABDOMEN + HIP + THIGH, data = body, importance = TRUE)
ft
```

```
##
## Call:
##  randomForest(formula = BODYFAT ~ AGE + WEIGHT + HEIGHT + CHEST +      ABDOMEN + HI
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 23.26881
##                     % Var explained: 66.64
```

R tells us that `ntree` is 500 and `mtry` (number of variables tried at each split) is 2. We can change these values if we want.

The square of the mean of squared residuals roughly indicates the size of each residual. These residuals are slightly different from the usual residuals in that for each observation, the fitted value is computed from those trees where this observation is out of bag. But you can ignore this detail.

The percent of variance explained is similar to R^2. The `importance = TRUE` clause inside the randomForest function gives some variable importance measures. These can be seen by:

```
importance(ft)
```

```
##          %IncMSE IncNodePurity
## AGE     10.26600      1086.051
## WEIGHT  14.15797      2252.906
## HEIGHT  11.97533      1204.042
## CHEST   15.96507      3120.295
## ABDOMEN 34.62901      5858.290
## HIP     14.45801      2050.055
## THIGH   11.03569      1369.466
```

The exact meaning of these importance measures is nicely described in the help entry for the function importance. Basically, large values indicate these variables were important for the prediction, roughly because many of the trees built as part of the random forest used these variables.

The variable `ABDOMEN` seems to be the most important (this is unsurprising given our previous experience with this dataset) for predicting bodyfat.

Now let us come to prediction with random forests. The R command for this is exactly the same as before. Suppose we want to the body fat percentage for a new individual whose `AGE = 40`, `WEIGHT = 170`, `HEIGHT = 76`, `CHEST = 120`, `ABDOMEN = 100`, `HIP = 101` and `THIGH = 60`. The prediction given by random forest for this individual's response is obtained via the function `predict`

```
x0 = data.frame(AGE = 40, WEIGHT = 170, HEIGHT = 76,
    CHEST = 120, ABDOMEN = 100, HIP = 101, THIGH = 60)
predict(ft, x0)
```

```
##        1
## 24.34849
```

Now let us come to classification and consider the email spam dataset. The syntax is almost the same as regression.

```
sprf = randomForest(as.factor(yesno) ~ crl.tot + dollar +
    bang + money + n000 + make, data = spam)
sprf
```

```
##
## Call:
##  randomForest(formula = as.factor(yesno) ~ crl.tot + dollar +      bang + money + n000 + make,
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 11.78%
## Confusion matrix:
##      n    y class.error
```

```
## n 2643  145  0.05200861
## y  397 1416  0.21897408
```

The output is similar to the regression forest except that now we are also given a confusion matrix as well as some estimate of the misclassification error rate.

Prediction is obtained in exactly the same was as regression forest via:

```
x0 = data.frame(crl.tot = 100, dollar = 3, bang = 0.33,
    money = 1.2, n000 = 0, make = 0.3)
predict(sprf, x0)
```

```
## 1
## y
## Levels: n y
```

Note that unlike logistic regression and classification tree, this directly gives a binary prediction (instead of a probability). So we don't even need to worry about thresholds.