Visualizing Multivariate Data

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.

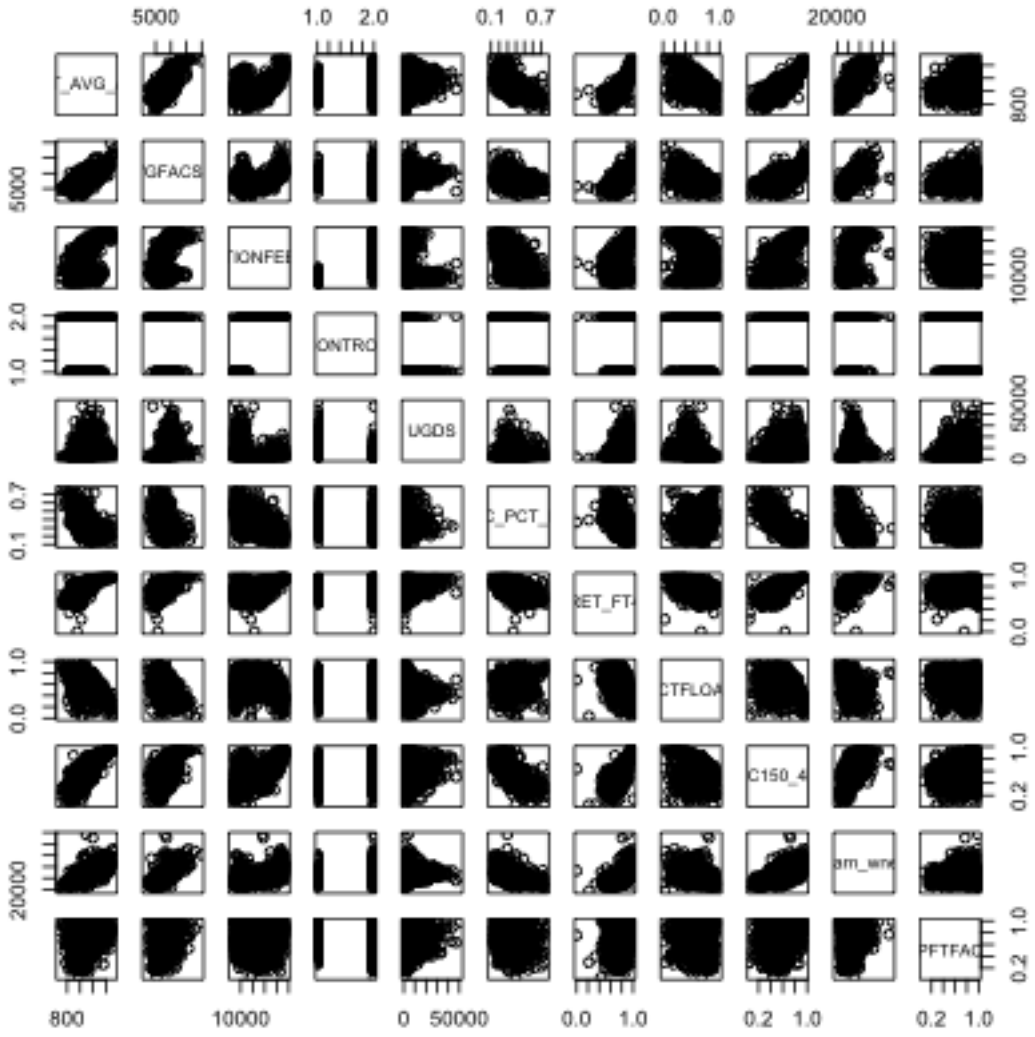
This chapter will focus on visualization of the relationship between many variables and using these tools to explore your data. This is often called **exploratory data analysis** (EDA)

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)]
pairs(smallScores)
```



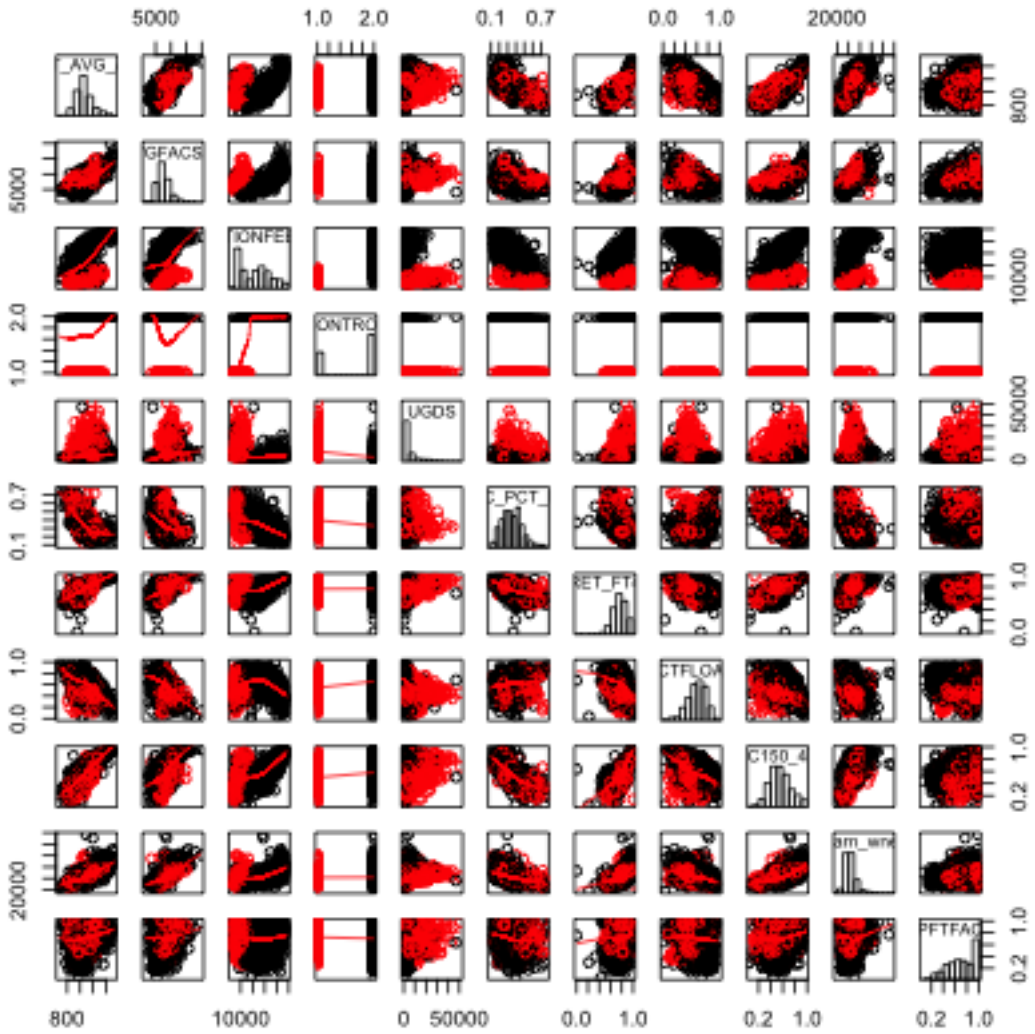
What kind of patterns can you see? What is difficult about this plot?

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")
```

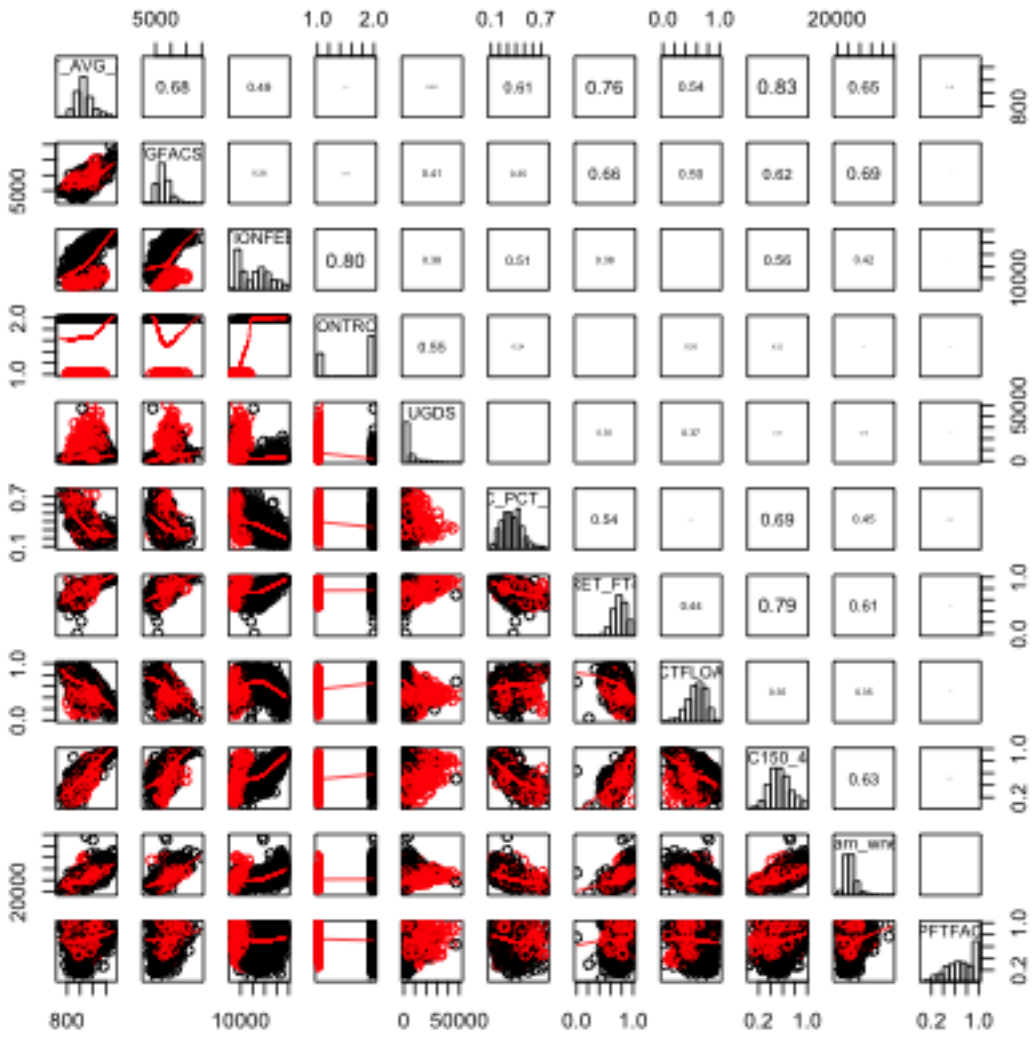
```
    on.exit(par(usr))
    par(usr = c(usr[1:2], 0, 1.5))
    h <- hist(x, plot = FALSE)
    breaks <- h$breaks
    nB <- length(breaks)
    y <- h$counts
    y <- y/max(y)
    rect(breaks[-nB], 0, breaks[-1], y)
}
pairs(smallScores, lower.panel = panel.smooth, col = c("red",
    "black")[smallScores$CONTROL], diag.panel = panel.hist)
```



In fact double plotting on the upper and lower diagonal is often a waste of space.

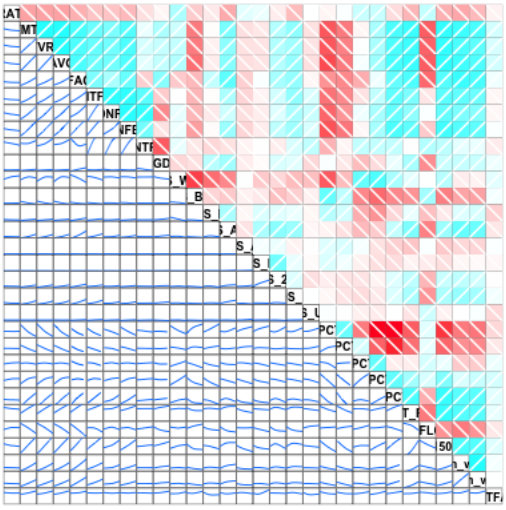
Here is code to plot the correlation value instead.

```
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, lower.panel = panel.smooth, upper.panel = panel.cor,
    col = c("red", "black")[smallScores$CONTROL], diag.panel = panel.hist)
```



For many variables, we can look at the correlations using colors and a summary of the data via loess smoothing curves. This is implemented in the `gpairs` function that offers a lot of the above features we programmed in a easy format.

```
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.

What do you see in this plot?

2 Categorical Variable

Let's consider now how we would visualize categorical variables, starting with the simplest, a single categorical variable.

Single Categorical Variable For a single categorical variable, how have you learn how you might visualize the data?

Let's demonstrate this 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.

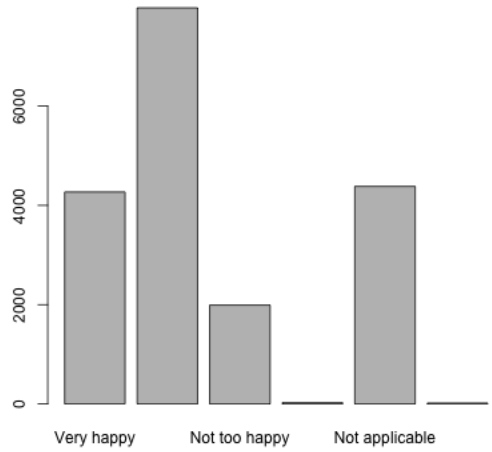
Here we look at a dataset where we have pulled out variables related to reported measures of well-being (based on a report about trends in psychological well-being `https://gssdataexplorer.norc.org/documents/903/display`). Like many surveys, the variables of interest are categorical.

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))
```

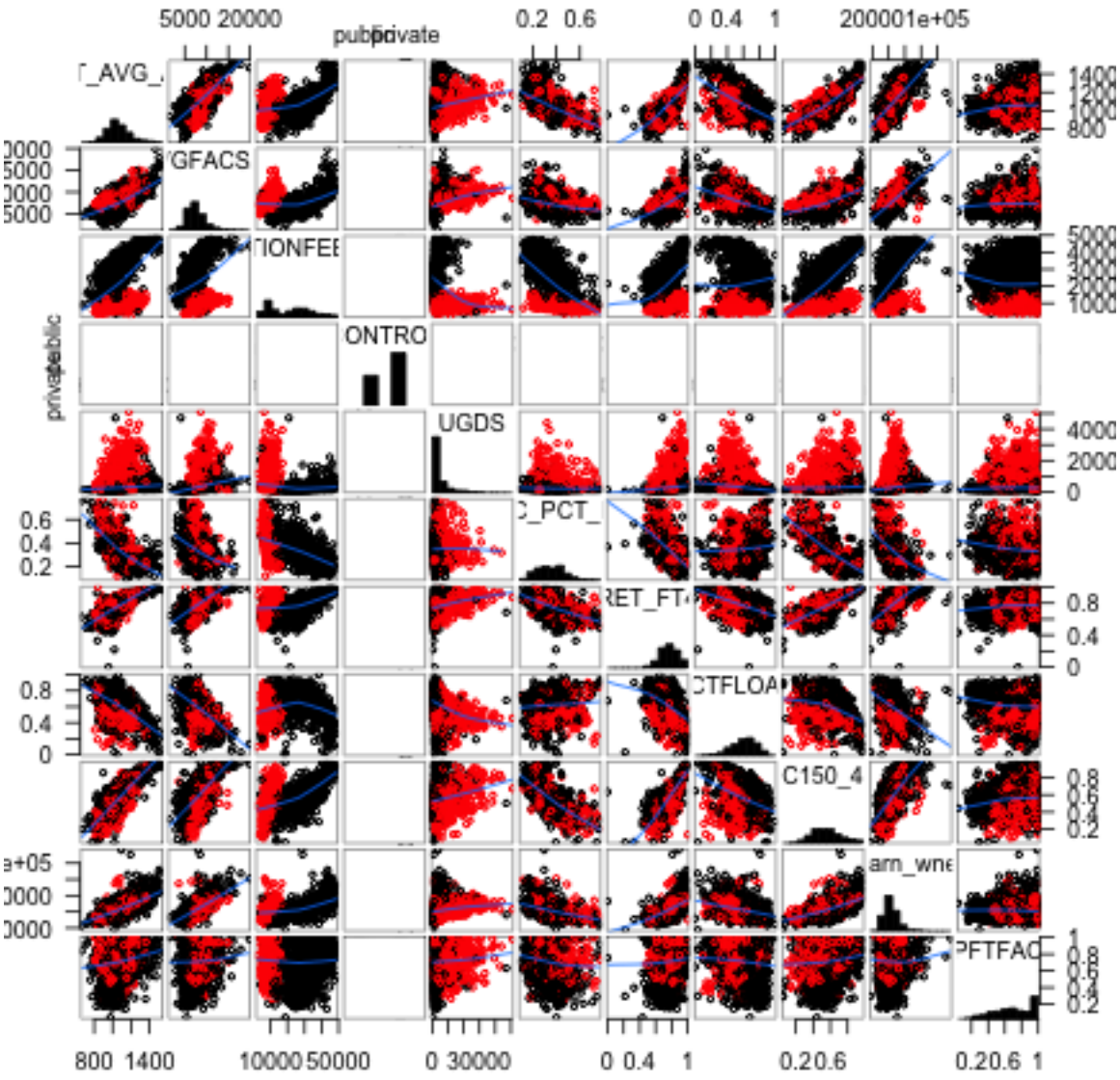


What would we do for relationship between a categorical and continuous variable?

Going back to our pairs plot of college, we can use the function `gpairs` (in the package `gpairs`) to incorporate 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]))
```



2.1 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.

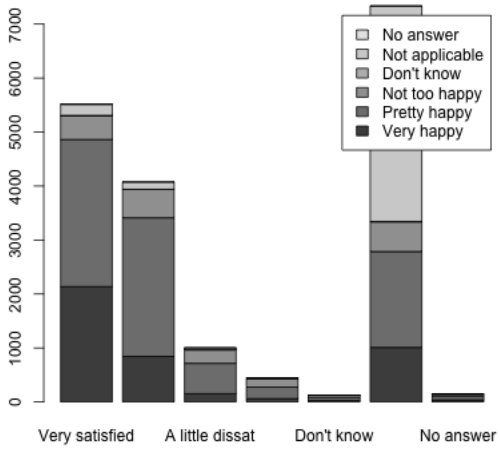
Cross-tabulations You have seen that **contingency tables** are a table that give the cross-tabulation of two categorical variables.

```
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
```

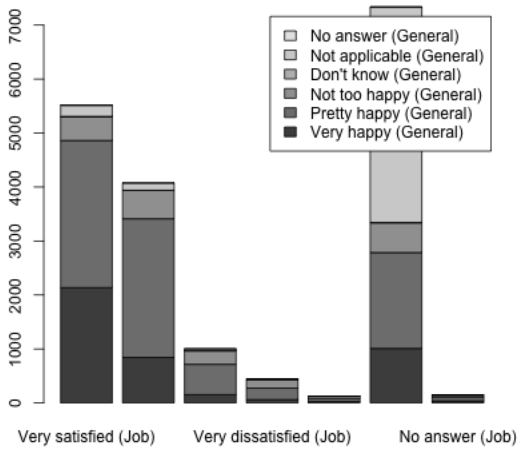
We can similarly make barplots to demonstrate these relationships.

```
barplot(tabGeneralJob, legend = TRUE)
```

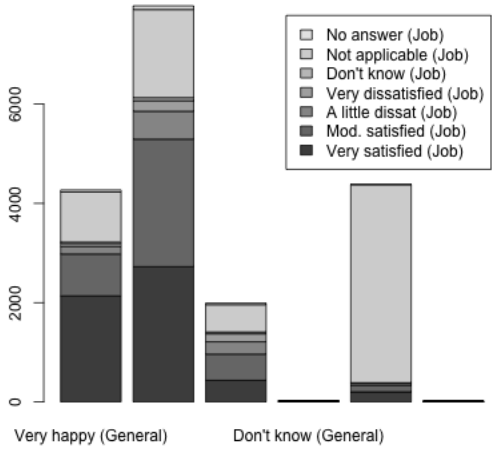


This barplot is not very satisfying. In particular, since the two variables have the same names for their levels, we don't know which is which!

```
colnames(tabGeneralJob) <- paste(colnames(tabGeneralJob),
    "(Job)")
rownames(tabGeneralJob) <- paste(rownames(tabGeneralJob),
    "(General)")
barplot(tabGeneralJob, legend = TRUE)
```

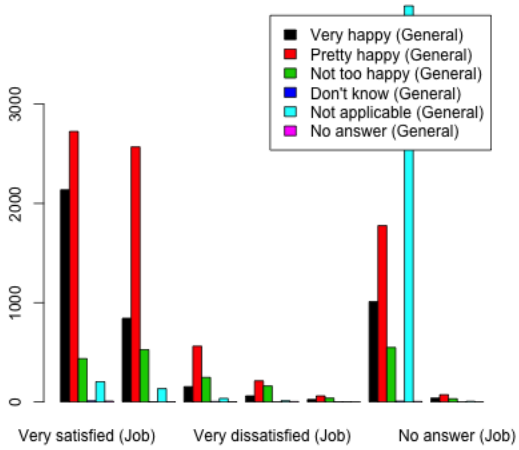



```
barplot(t(tabGeneralJob), legend = TRUE)
```



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])
```



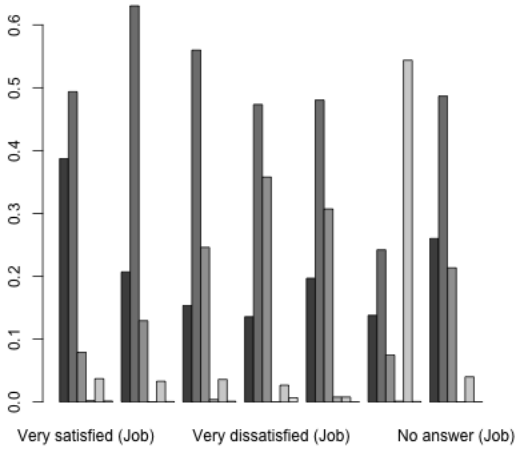
Conditional Distributions from Contingency Tables When we look at the contingency table, a natural question we ask is whether the distribution of the data changes across the different categories. For example, for people answering 'Very Satisfied' for their job, there is a distribution of answers for 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.

Looking at the barplot, what would you say? Are the variables related?

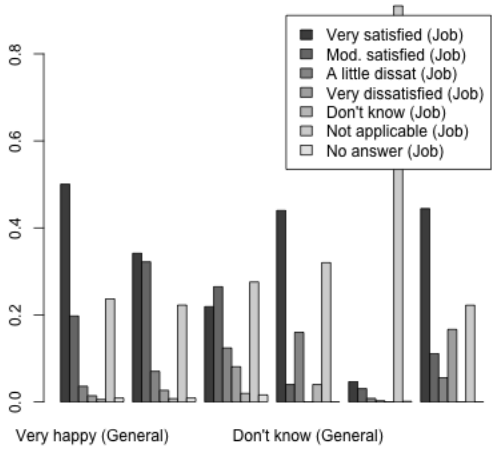
We can, of course, flip the variables around.

```
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?

2.2 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
```

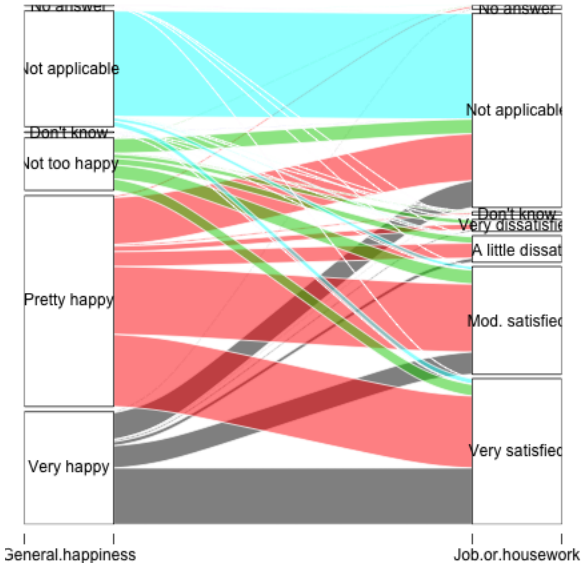
This format extends more easily to more variables:

```
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])
```

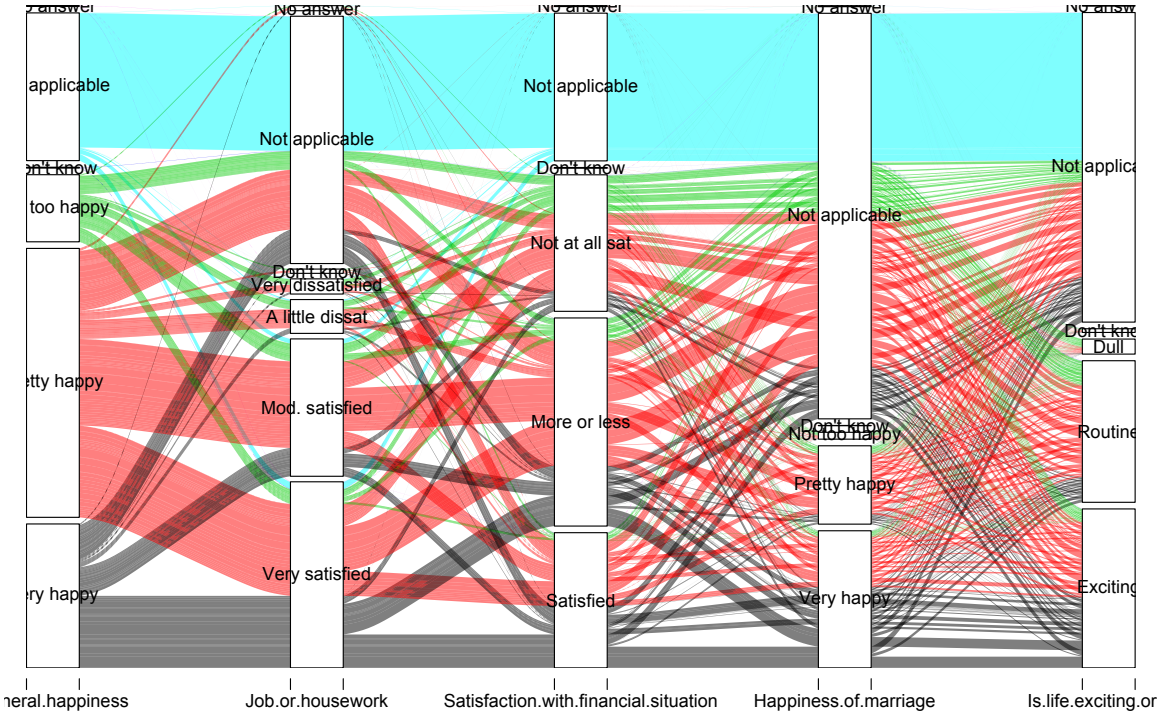


Notice how you can see the relative numbers that go through each category.

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$Freq, col = palette()[wellbeingAggregatesBig$Genera
```



Putting aside the messiness, we can at least see some big things about the data. For example, what would you say about people's willingness to answer these questions, and more generally about non-response?

What other things can you see about the data?

These are obviously **self-reported** measures of happiness, meaning only what the respondent says is their state; these are not external, objective measures (and indeed, with happiness, that is hard!).

What are the 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.

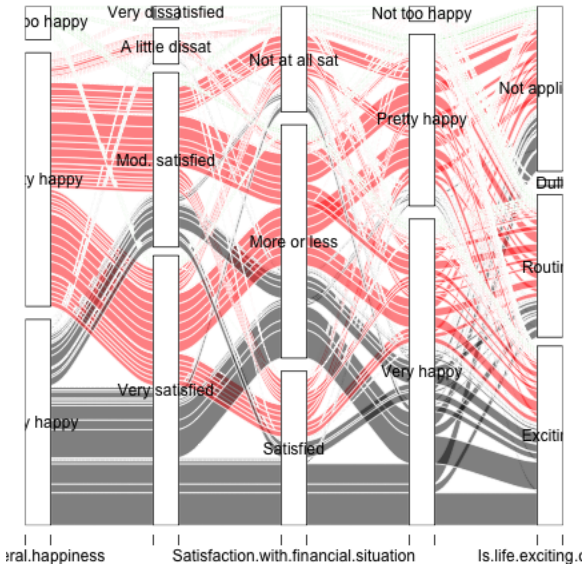
```
wh <- with(wellbeingRecent, which(General.happiness ==
    "Not applicable" | Job.or.housework == "Not applicable" |
    Satisfaction.with.financial.situation == "Not applicable"))
wellbeingCondenseGroups <- wellbeingRecent[-wh, ]
wellbeingCondenseGroups <- subset(wellbeingCondenseGroups,
    !General.happiness %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)
```

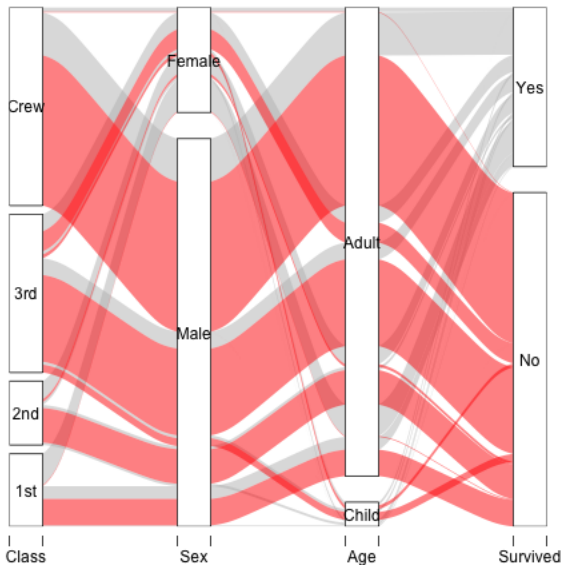
```
alluvial(wellbeingMarriedAggregates[, -ncol(wellbeingMarriedAggregates)],
    freq = wellbeingMarriedAggregates$Freq, hide = wellbeingMarriedAggregates$Freq <
        quantile(wellbeingMarriedAggregates$Freq, 0.5),
    col = palette()[wellbeingMarriedAggregates$General.happiness])
```



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"))
```



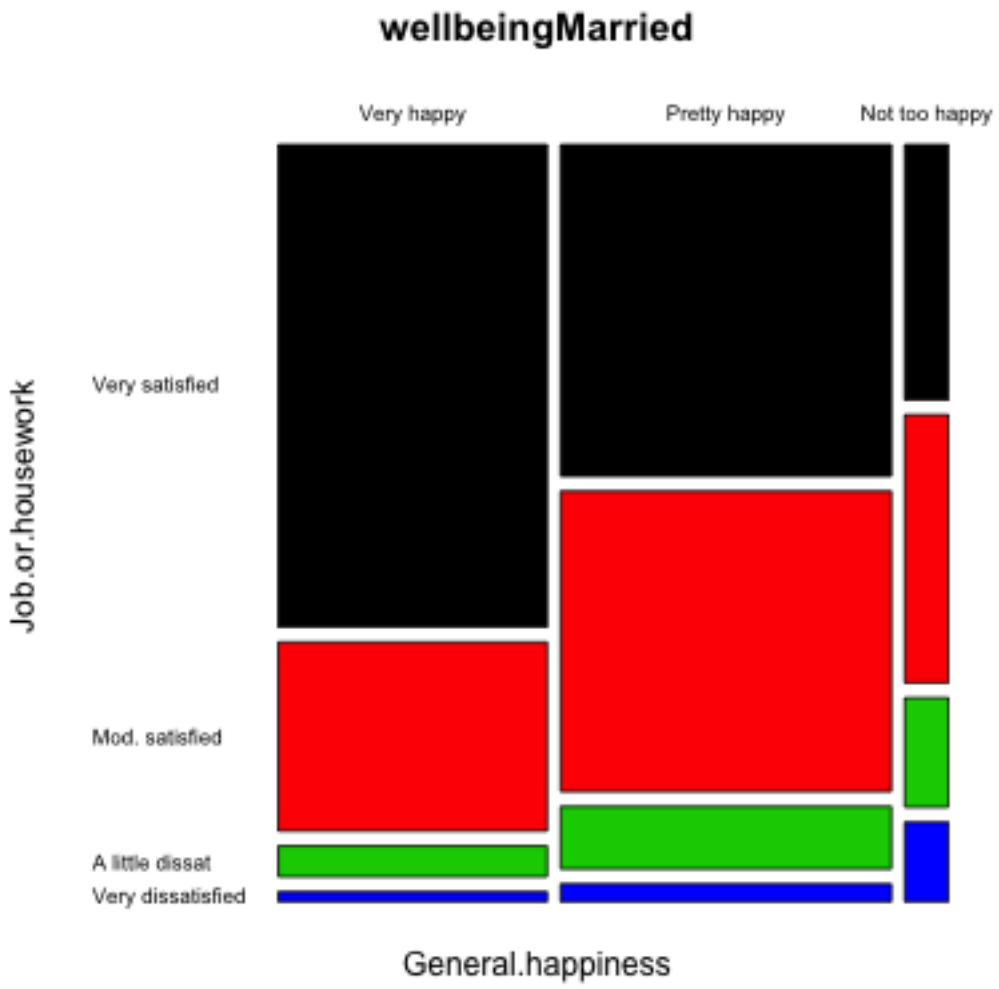
Like so many visualization tools, the effectiveness of a particular plot depends on the dataset.

2.3 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.

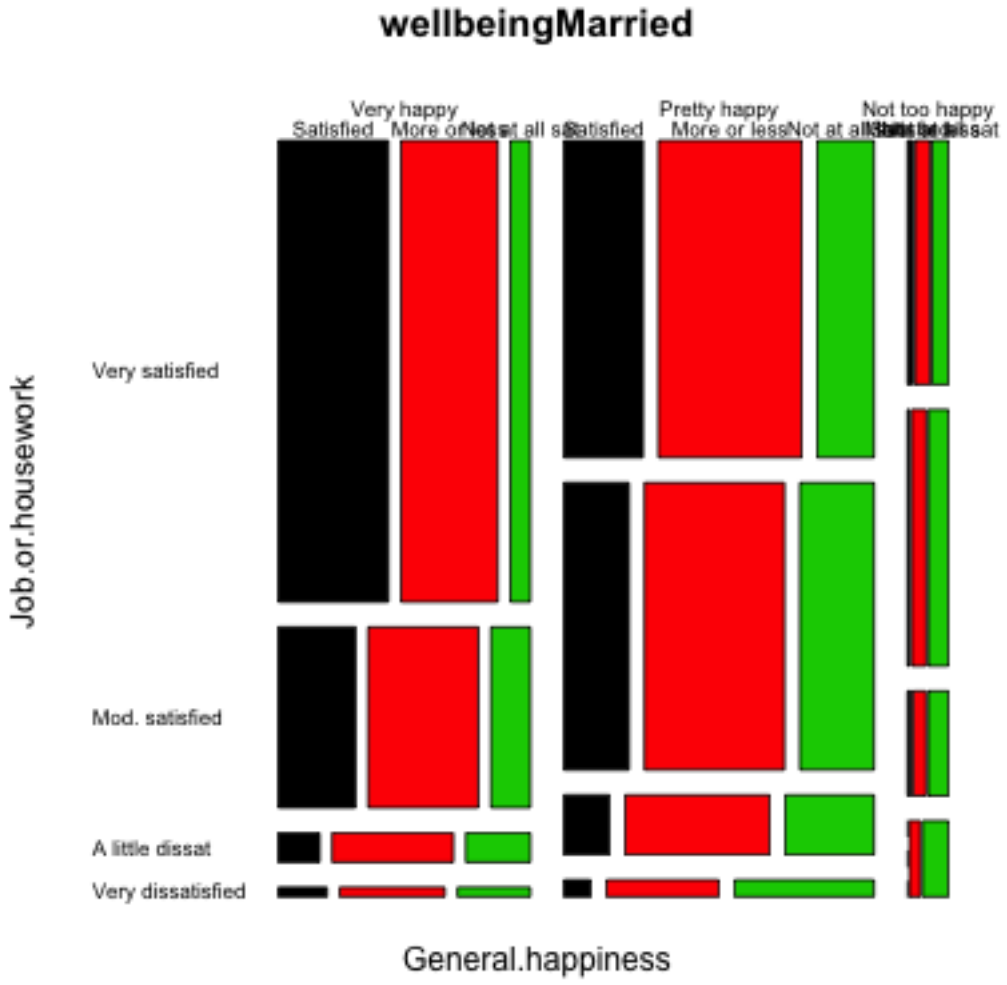
```
mosaicplot(~General.happiness + Job.or.housework, data = wellbeingMarried,
    las = 1, col = palette())
```



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 responded 'Very 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 for 'Job 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.

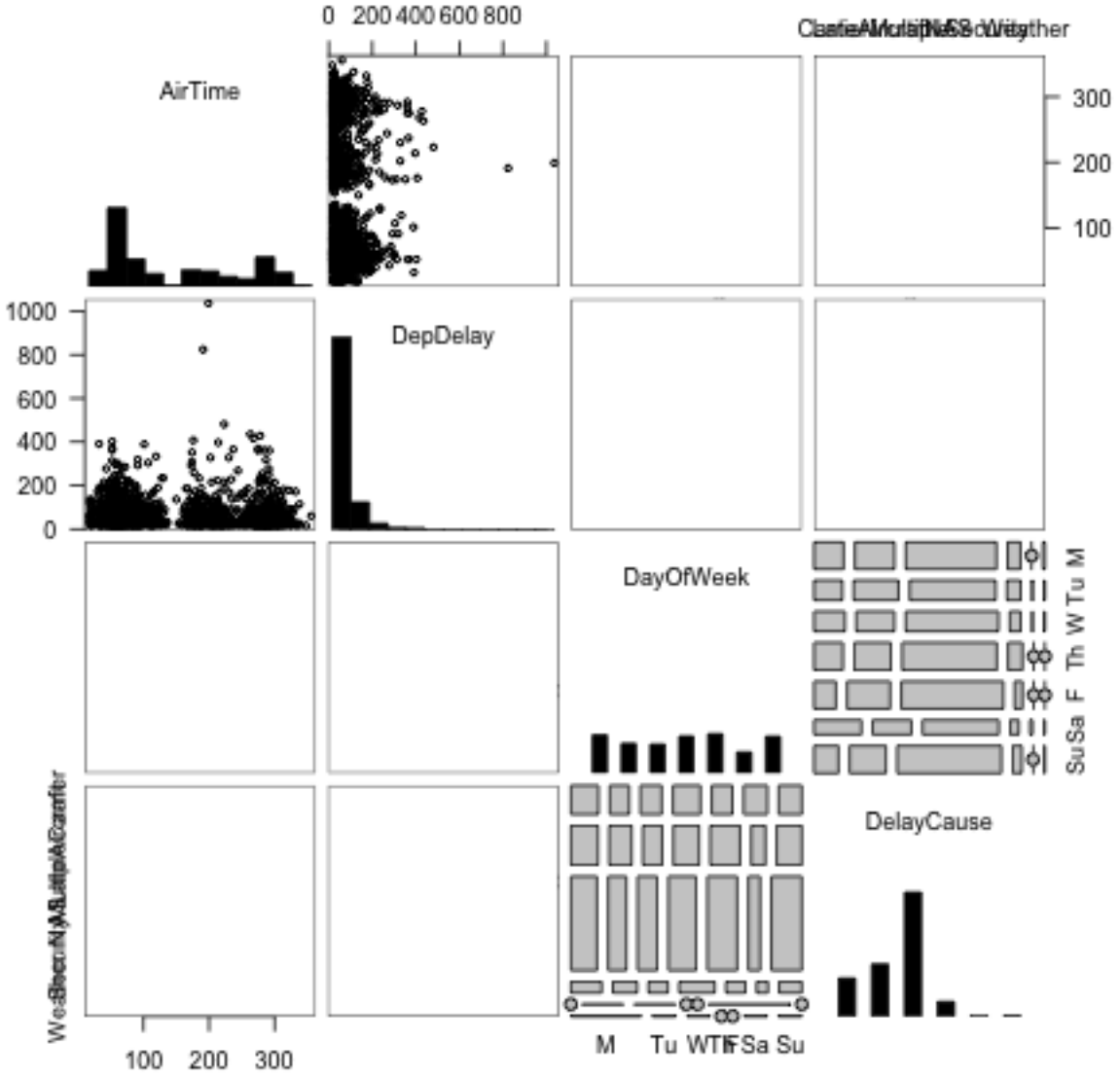
Using this information, how do you interpret this plot? What does this tell you about people who are 'Very Happy' in general happiness?

2.4 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 = "boxplo
```



How do you interpret the different elements of this pairs plot?

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. There are measurements for 19,000 genes but we limited ourselves to around 275 genes.

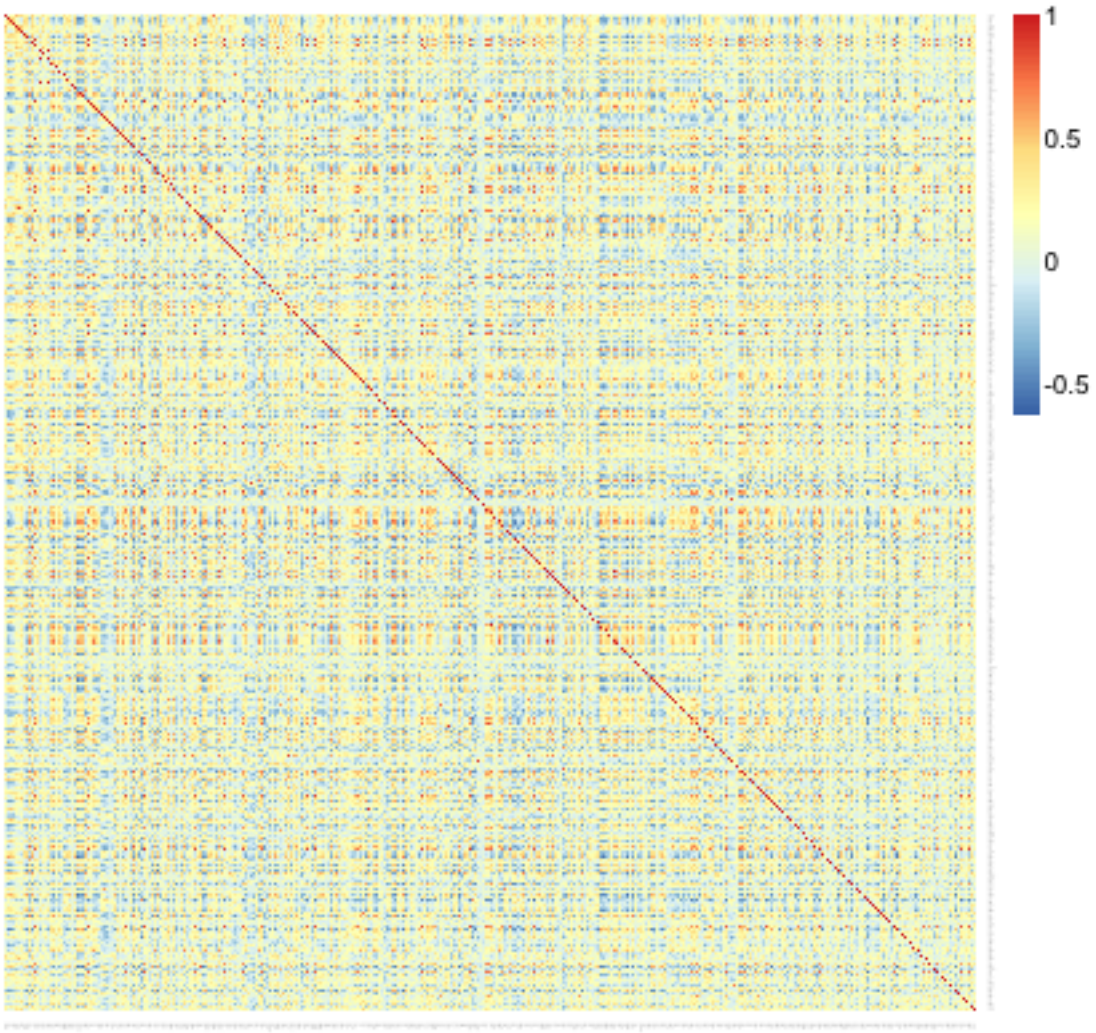
```
breast <- read.csv(file.path(dataDir, "highVarBreast.csv"))
```

One common goal of this kind of data is to be able to identify what different types of breast cancers. The idea is that by looking at the genes in the tumor, we can discover similarities, which might lead to

We have so many variables, that we might consider visualizing the correlations like before. We will use a different type of plot called a **heatmap** to do so. A heatmap works like the correlation picture we looked at before. 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(NMF)
nmf.options(grid.patch = TRUE)
corMat <- cor(breast[, -c(1:7)])
aheatmap(corMat, Rowv = NA, Colv = NA)
```



Why is the diagonal all dark red?

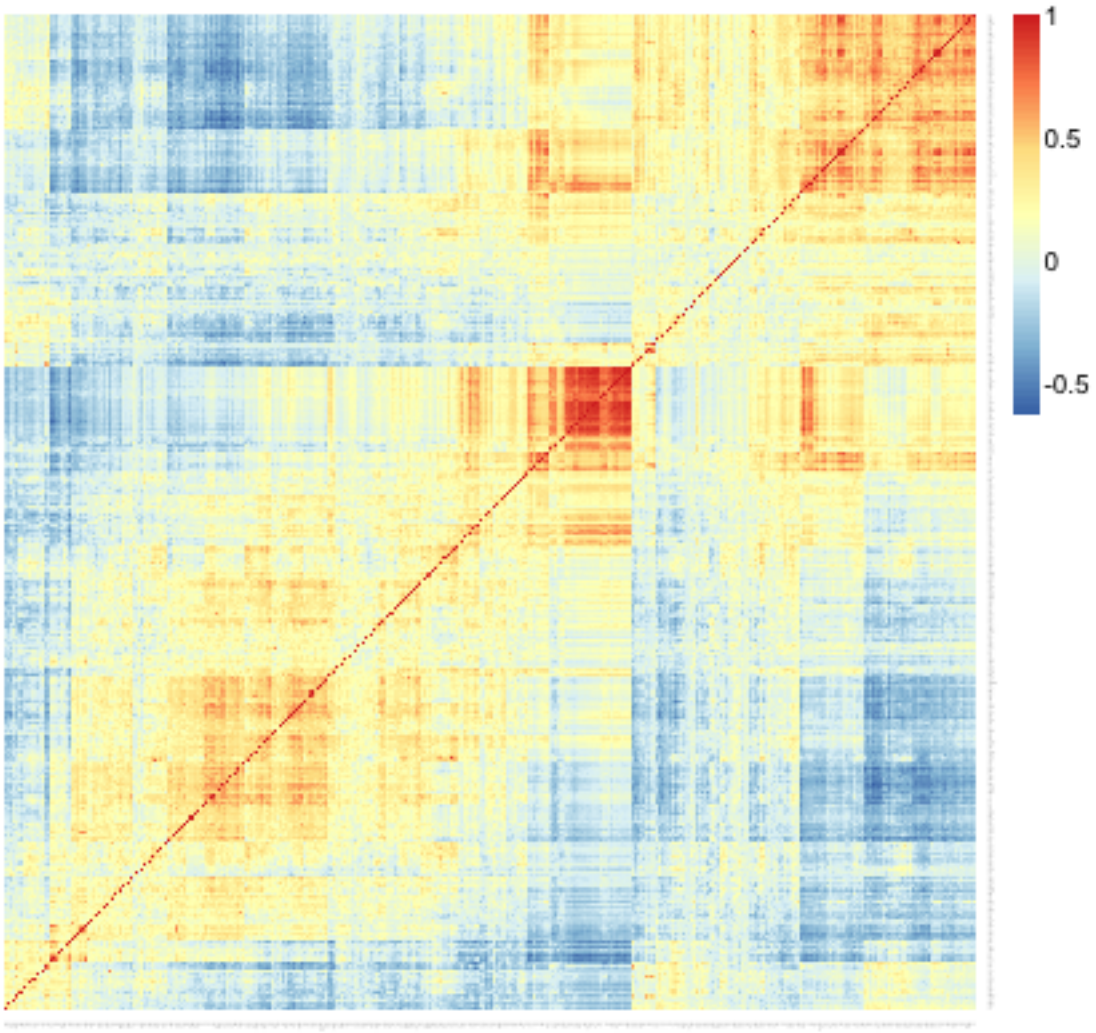
This is not an informative picture, however – there are so many variables (genes) that we can't discover anything here.

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.

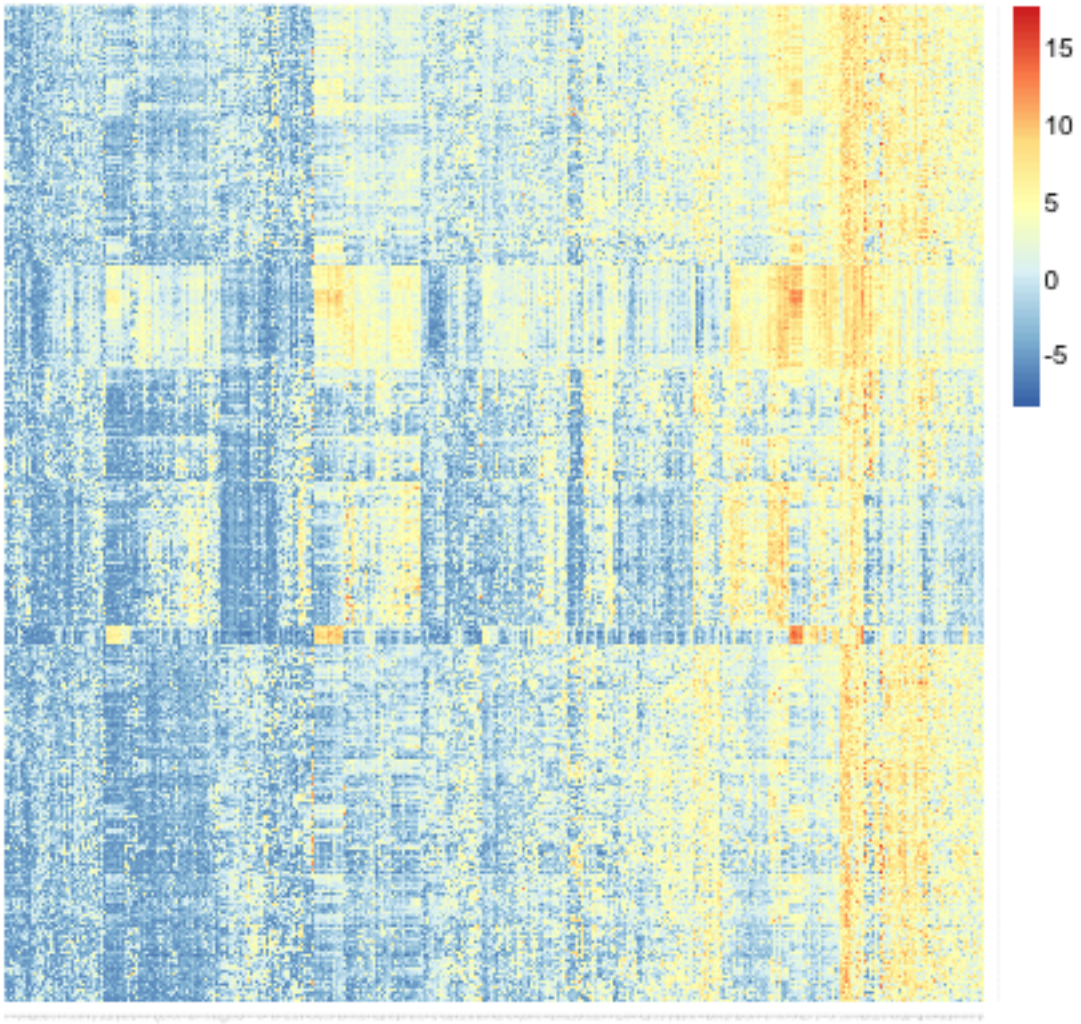
```
aheatmap(corMat, Rowv = FALSE, Colv = FALSE)
```



What do we see in this heatmap?

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.

```
aheatmap(breast[, -c(1:7)], Rowv = FALSE, Colv = FALSE,
    scale = "none")
```



What do we see in this heatmap?

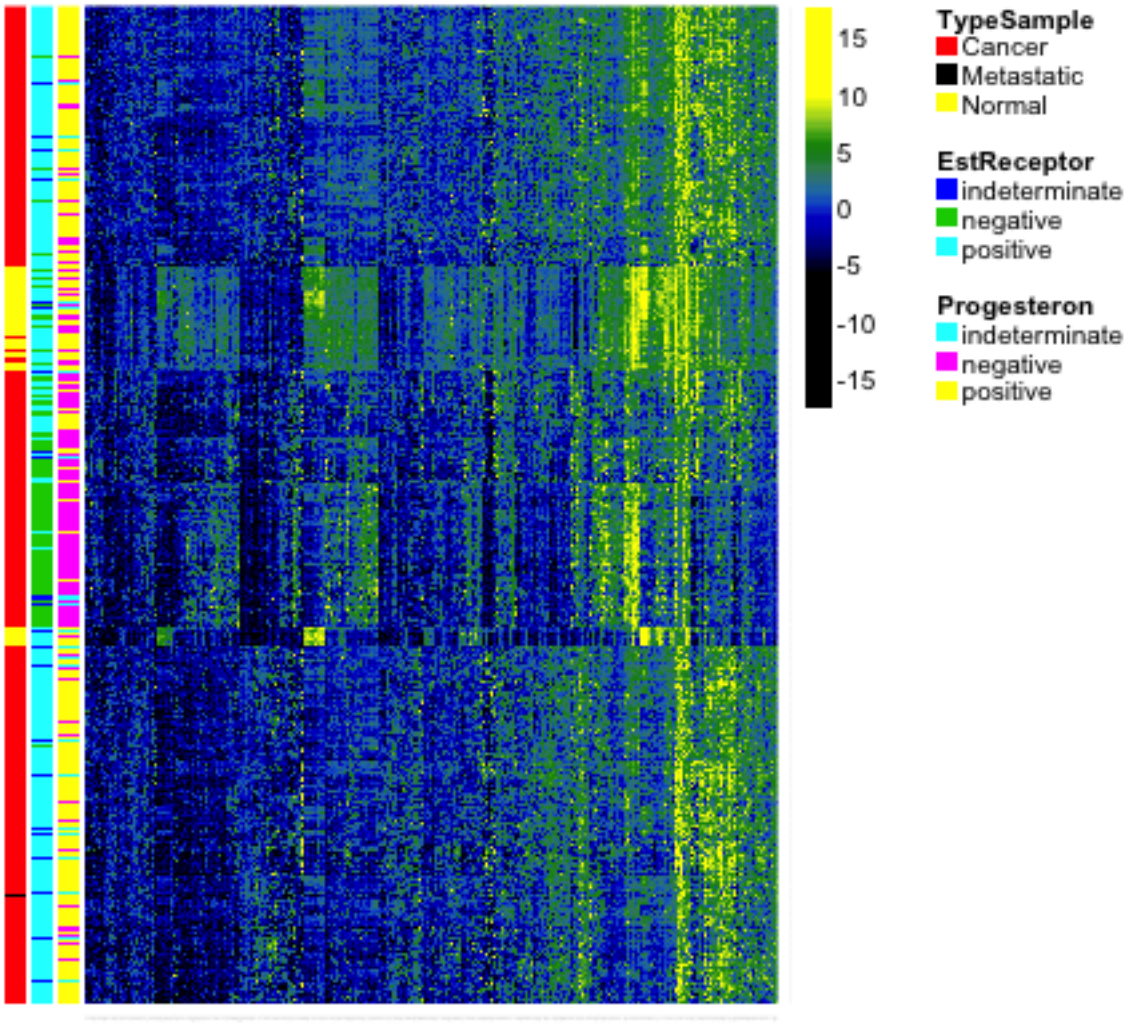
We can improve upon this heatmap. I prefer different colors for this type of data, and we can add some information we have about these samples. I am also going to change how the heatmap assigns colors to the data. Specifically, heatmap gives a color for data by binning it and all data within a particular 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(estCol) <- levels(breast$Progesteron)
seqPal5 <- colorRampPalette(c("black", "navyblue",
    "mediumblue", "dodgerblue3", "aquamarine4", "green4",
    "yellowgreen", "yellow"))(16)
m <- max(abs(breast[, -c(1:7)]))
qnt <- quantile(as.numeric(data.matrix((breast[, -1]))),
    c(0.01, 0.99))
brks <- c(-m, seq(qnt[1], qnt[2], length = 50), m)
head(brks)
```

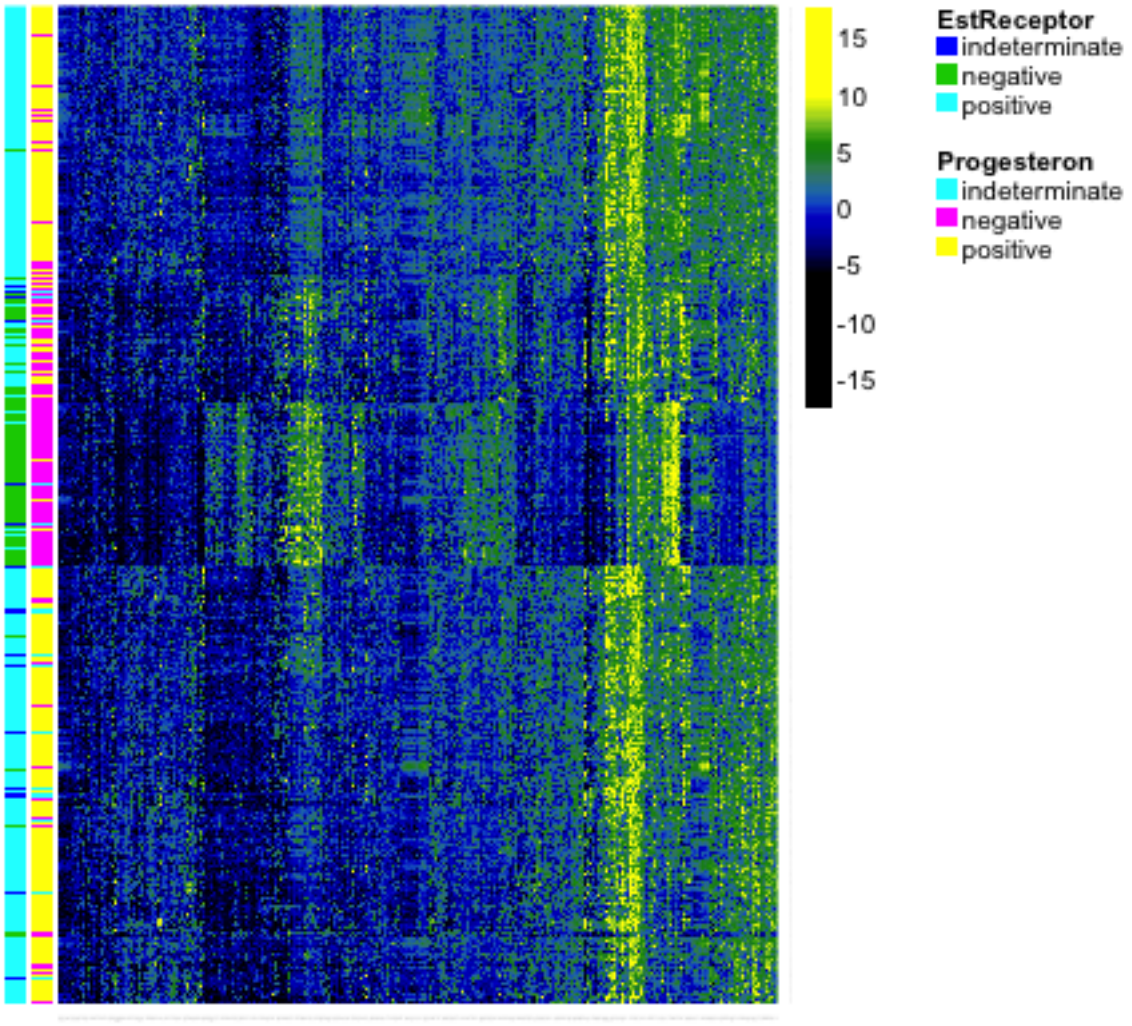
```
## [1] -17.651678  -5.742913  -5.428952  -5.114990  -4.801028  -4.487066
```

```
fullHeat <- aheatmap(breast[, -c(1:7)], Rowv = FALSE,
    Colv = FALSE, breaks = brks, col = seqPal5, annRow = breast[,
        5:7], annColors = list(TypeSample = typeCol,
        EstReceptor = estCol, Progesteron = proCol))
```



What do we see now?

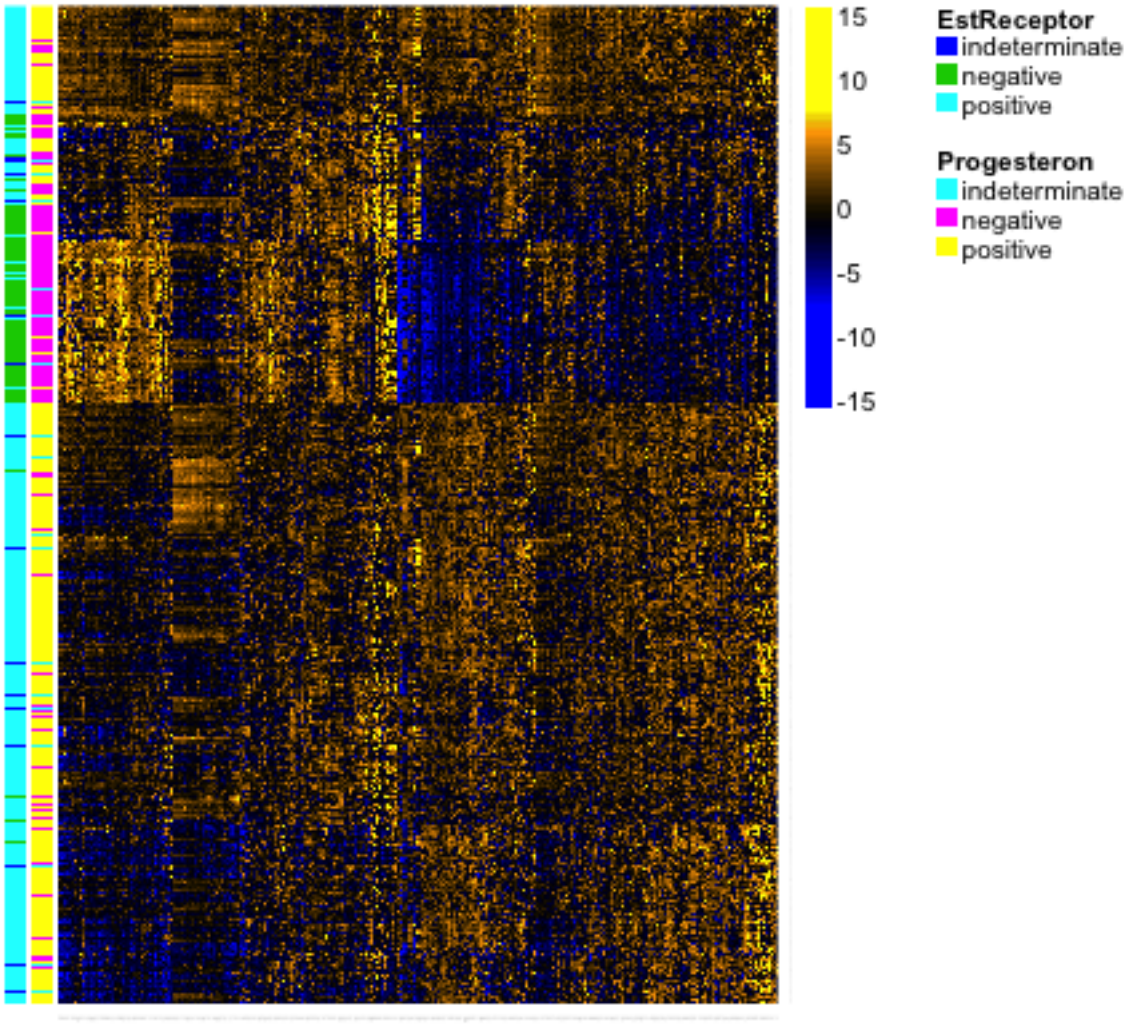
```
whCancer <- which(breast$Type != "Normal")
aheatmap(breast[whCancer, -c(1:7)], Rowv = FALSE, Colv = FALSE,
    breaks = brks, col = seqPal5, annRow = breast[whCancer,
        6:7], annColors = list(TypeSample = typeCol,
        EstReceptor = estCol, Progesteron = proCol))
```



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. Why this focus on 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, "-")
m <- max(abs(breastCenteredMed[, -1]))
qnt <- max(abs(quantile(as.numeric(data.matrix((breastCenteredMed[,
    -1])))), c(0.01, 0.99))))
brksCentered <- c(-m, seq(-qnt, qnt, length = 50),
    m)
seqPal2 <- colorRampPalette(c("orange", "black", "blue"))(16)
seqPal2 <- (c("yellow", "gold2", seqPal2))
seqPal2 <- rev(seqPal2)
aheatmap(breastCenteredMed[whCancer, -c(1:7)], Rowv = FALSE,
    Colv = FALSE, breaks = brksCentered, annRow = breast[whCancer,
        6:7], labRow = NA, col = seqPal2, annColors = 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.

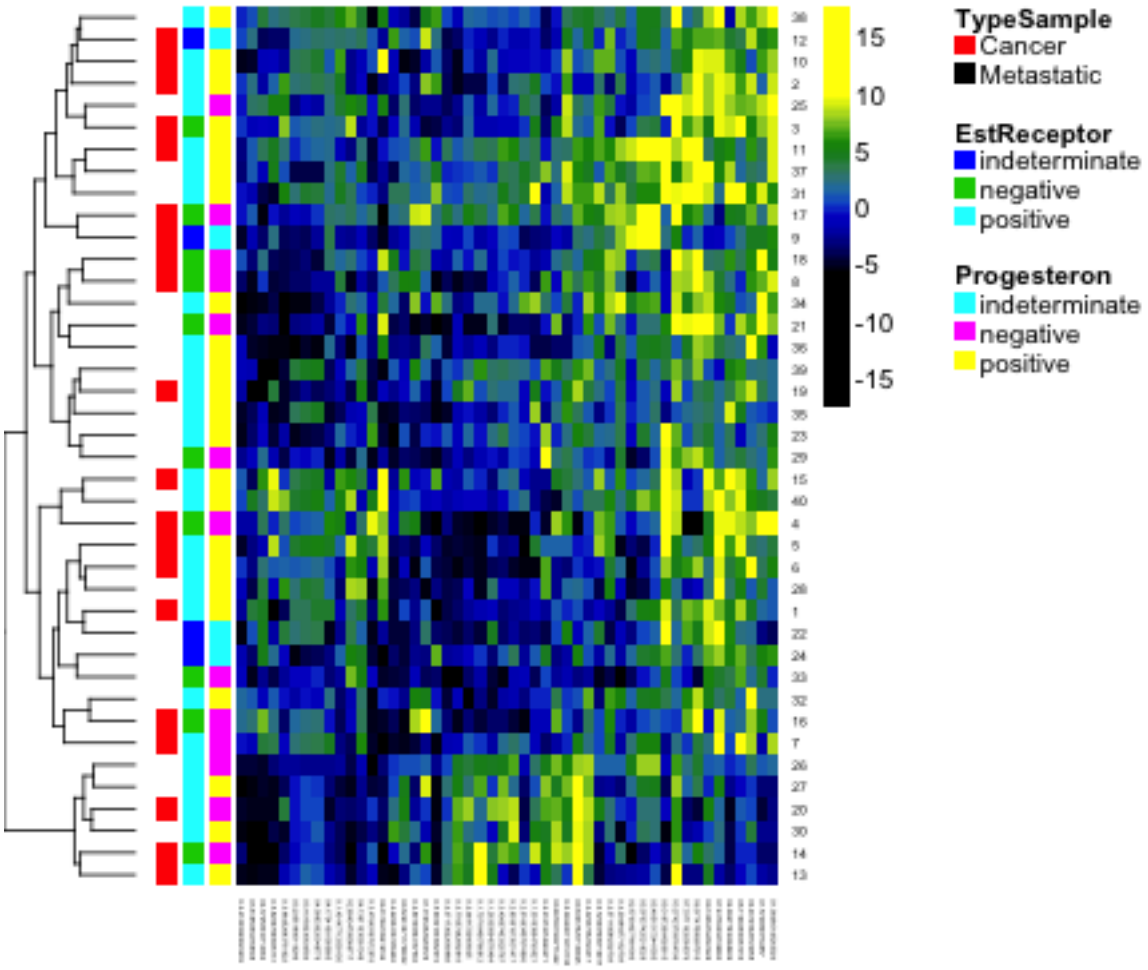
3.1 Clustering

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, 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 rely on examining the distances between points and putting together 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** for the samples.

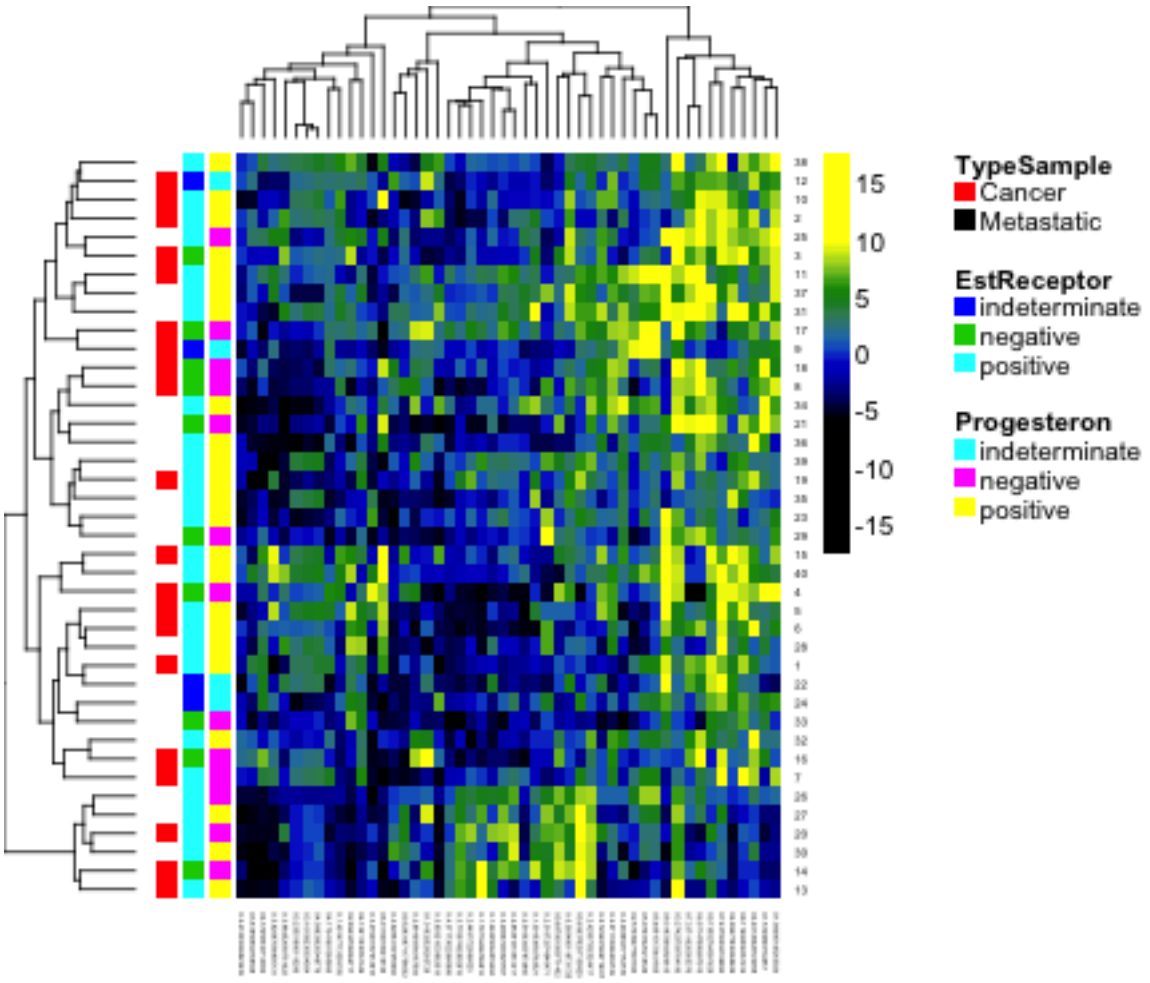
Here we show this on a small subset of the samples and genes.

```
smallBreast <- read.csv(file.path(dataDir, "smallVarBreast.csv"),
    header = TRUE)
aheatmap(smallBreast[, -c(1:7)], Rowv = TRUE, Colv = FALSE,
    breaks = brks, col = seqPal5, annRow = smallBreast[,
        5:7], annColors = list(TypeSample = typeCol,
        EstReceptor = estCol, Progesteron = proCol))
```



We can use the same principle for clustering the variables:

```
aheatmap(smallBreast[, -c(1:7)], Rowv = TRUE, Colv = TRUE,
    breaks = brks, col = seqPal5, annRow = smallBreast[,
        5:7], annColors = 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.

How Hierarchical Clustering Works The idea behind hiearchical clustering is to first calculate all of the pairwise distances between all of the samples. For example, if you have one variable, say $y_1, \ldots, y_n$ the standard distance between samples i and j is $d_{ij} = |y_i - y_j|$ or equivalently squared distance, $d_{ij} = (y_i - y_j)^2$. So we can get all of the pairwise distances between all of the samples.

Then find the two points that are closest together, and join them together. Then you repeat this process of joining together to build up the tree.[1]

[1]This is called an agglomerative method, where you start at the bottom of the tree and build up.

Of course this is easier said than done, because how do you join together two samples that have been joined to yet another sample so as to 'build' the tree? Specifically, you have joined together samples i and j to make the first join. How do you decide if sample k should be joined? You need to have a way of defining the distance of this joined group to the other samples. 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 join of i and j, then the distance of that join to sample k is the maximum distance of i and j to k. And similarly higher up in the tree, when you have more points. More generally, if you have a set of points A that have been joined to create one branch of a tree, and a set of different points B that have been joined to create a branch, then the distance between those two branches is the maximum distance of all the points in A to all the points in B.[2]

Higher Dimension Distances What is the equivalent distance when you have more variables? Now 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:

$$(y_i^{(1)}, \ldots, y_i^{(p)})$$

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

$$d_{ij} = \sqrt{\sum_{\ell=1}^{p} (y_i^{(p)} - y_j^{(p)})^2}$$

So its the cummulative 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

$$d_{ij} = \sum_{\ell=1}^{p} (y_i^{(p)} - y_j^{(p)})^2$$

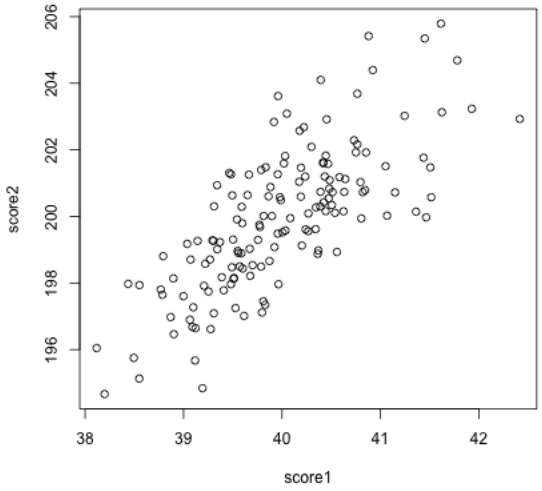
There are also divisive method for creating a hiearchical tree that starts at the top by dividing the samples.

[2] This is called complete linkage.

4 Principal Components Analysis

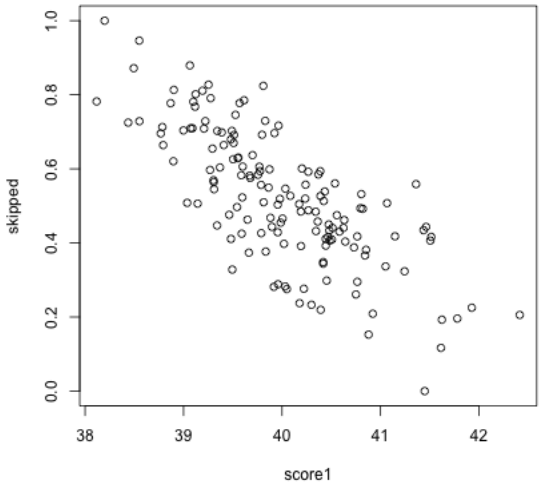
In looking at both the college data and the gene expression data, it is clear that there is a lot of redundancy in our variables, meaning that several variables are often giving us the same information about the patterns in our observations. We could see this by looking at their correlations, or by seeing 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.

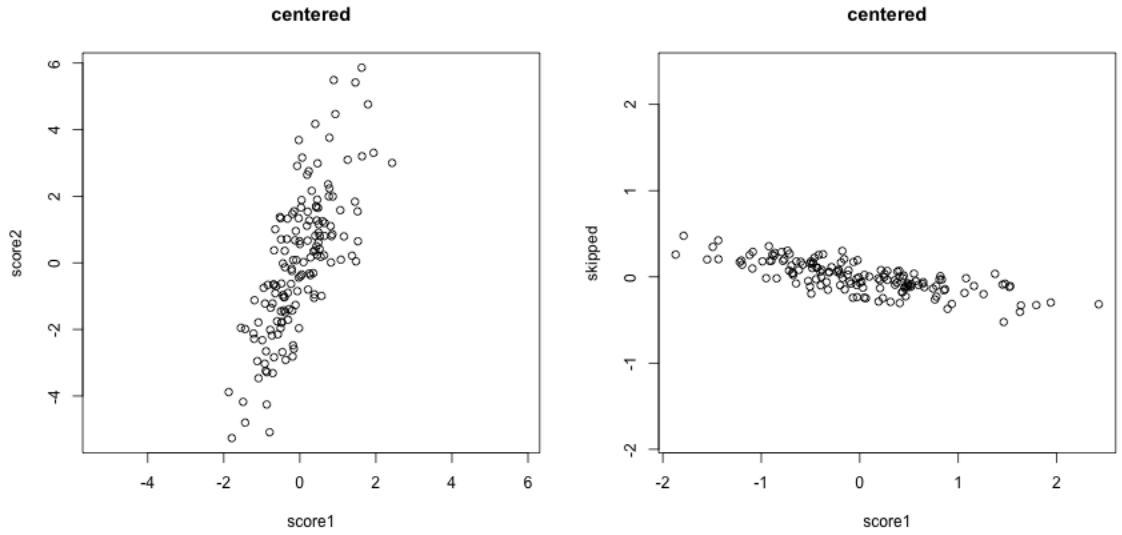
There are two *equivalent* ways to think about how principal components analysis does this.

4.1 Linear combinations of existing variables

You want to find a single score to give a final grade.

What is the problem with taking the mean of our two exam scores?

Let's assume we make them have the same mean, what is the problem?



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.

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

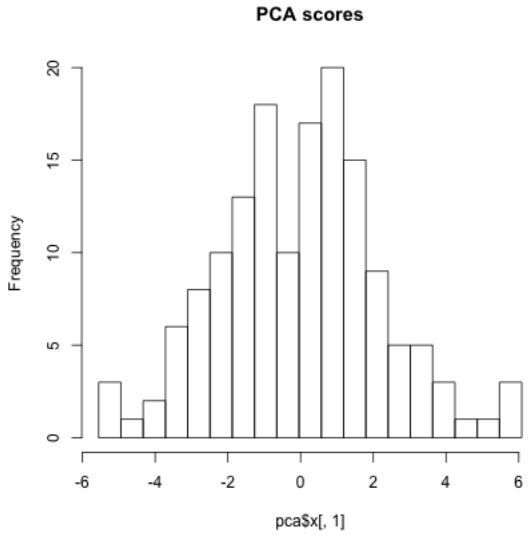
$$\hat{var}(z) = \frac{1}{n-1}\sum_{i=1}^{n}(z_i - \bar{z})^2$$

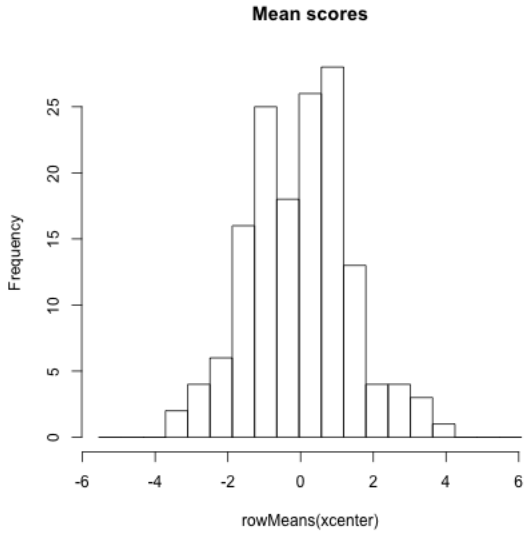
Of course, $z_i = a_1 x_i^{(1)} + a_2 x_i^{(2)}$, this is actually

$$\hat{var}(z) = \frac{1}{n-1}\sum_{i=1}^{n}(a_1 x_i^{(1)} + a_2 x_i^{(2)} - \bar{z})^2$$

(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.

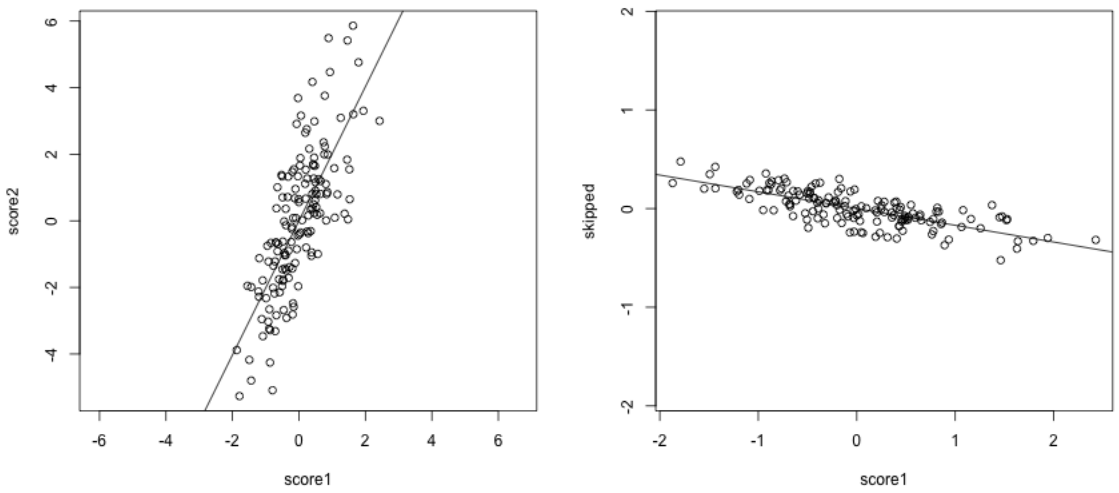




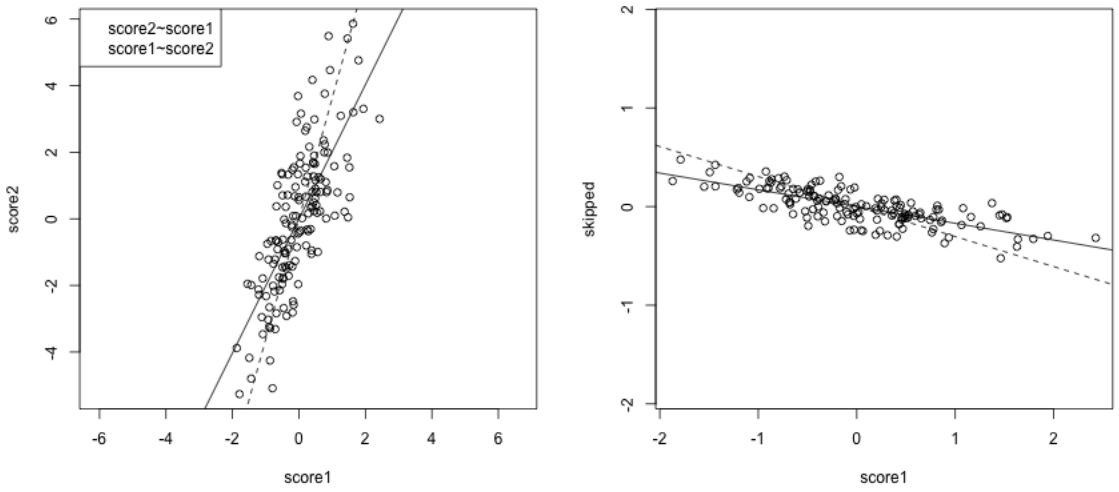
We'll return to considering this criteria more but first let's look at the other interpretation of summarizing the data.

4.2 Geometric Interpretation

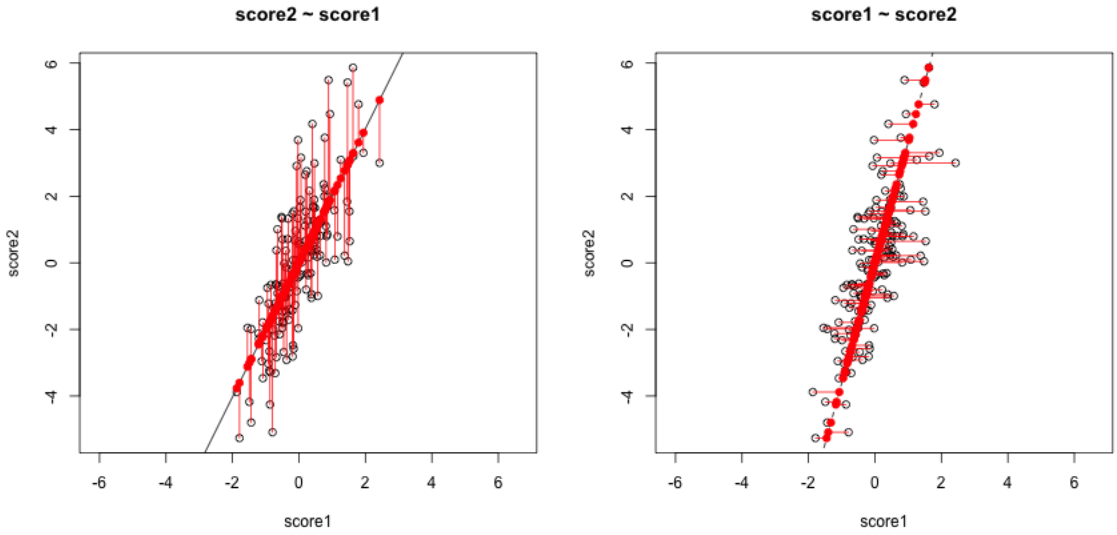
Another way to consider our redundancy is geometrically. If this was a regression problem we would "summarize" the relationship betweeen our variables by the regression line:



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.



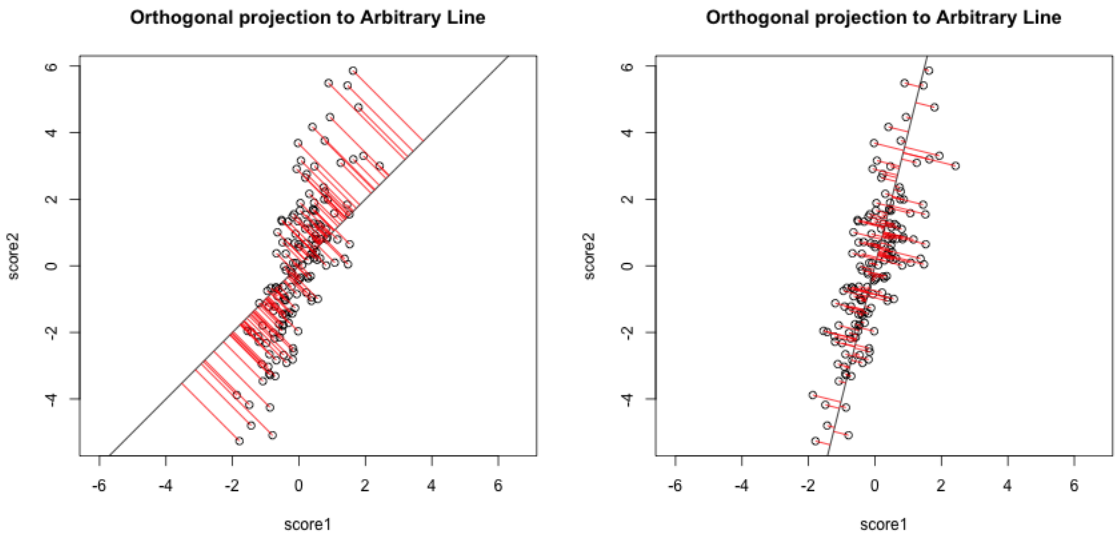
The problem here is that our definition of what is the best line summarizing this relationship is not symmetric in regression. Our best line minimizes error in the y direction. Specifically, for every observation i, we project our data onto the line so that the error in the y direction is minimized.



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.

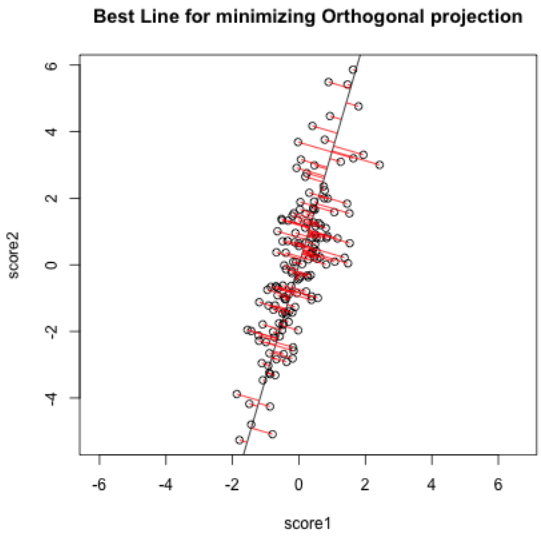
By distance of a point to a line, we mean the minimimum distance of any point to the line. This is found by drawing another line that goes through the point and is orthogonal to the line. Then the length of that line segment from the point to the line is the distance of a point to the line.

Just like for regression, we can consider all lines, and for each line, calculate the average distance of the points to the line.

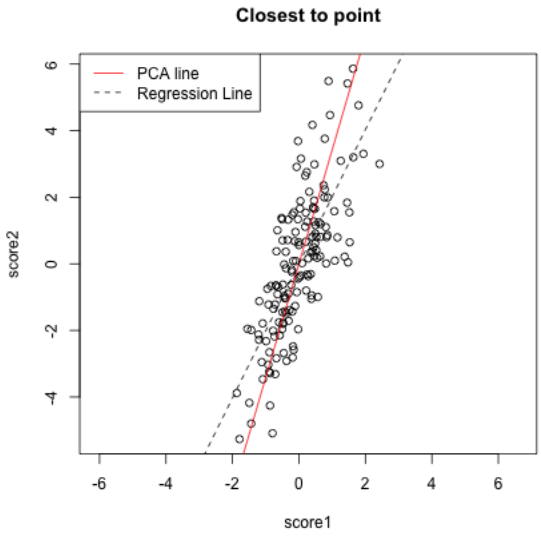


So to pick a line, we now find the line that minimizes the average distance to the

line across all of the points. This is the PCA line:



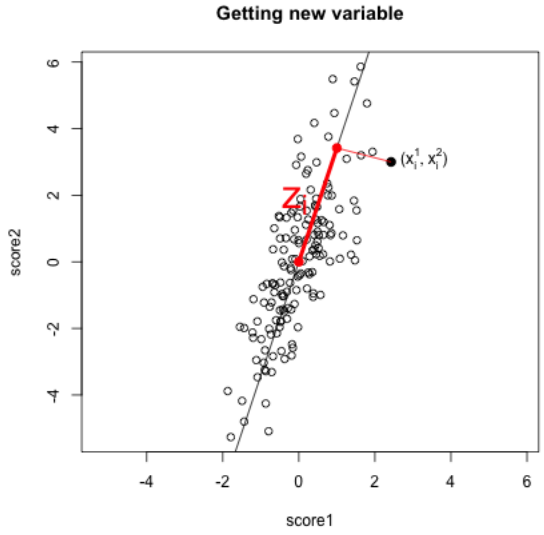
Compare this to our regression line:



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.

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. But if we consider the PCA line as our axis, then the distance of the projected point along that axis (from zero) is our new variable z_i.[3]



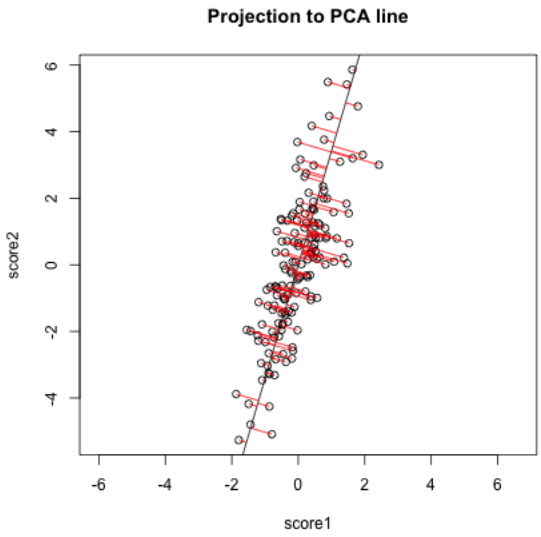
Relationship to linear combinations Note if we consider all points such that $a_1x^{(1)} + a_2x^{(2)} = c$ where c is some constant, this difines a line through the space, where each different c gives a different intercept. We know we want our line to go through (0,0), so that means $a_1x^{(1)} + a_2x^{(2)} = 0$ is the line corresponding to our linear combination. So coefficients of a linear combination of our variables define a line through our space.

Finding the best line as described above (minimum average distance of points to line), 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 that results in the greatest variance of our points. This is beyond the scope of the class to explain.

However, this means that if we minimize the distance to the line, we *also* find the line so that the projected points have the most variance – so we can spread out the

[3]From zero, because I centered the data, so the center of the points is at (0,0).

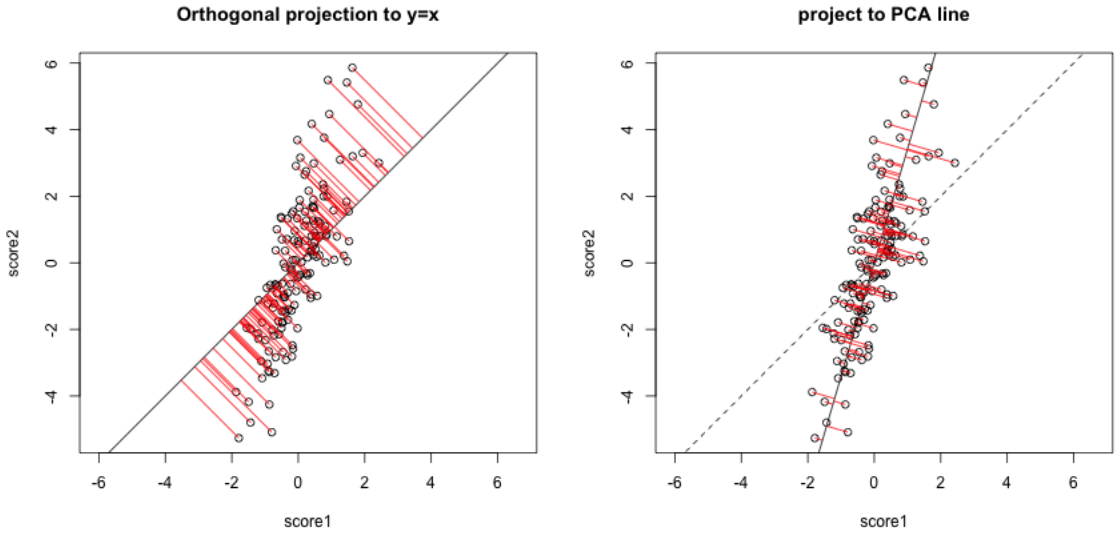
points the most.



However, we can use this information to consider what is the line corresponding to the linear combination defined by the mean,

$$\frac{1}{2}x^{(1)} + \frac{1}{2}x^{(2)}$$

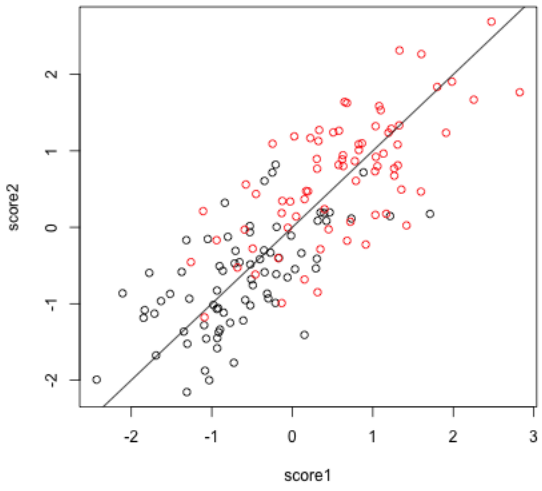
It is the line $y = x$,



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 the same thing from the PCA linear combination and the mean.[4] 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 think about combining 2 variables.

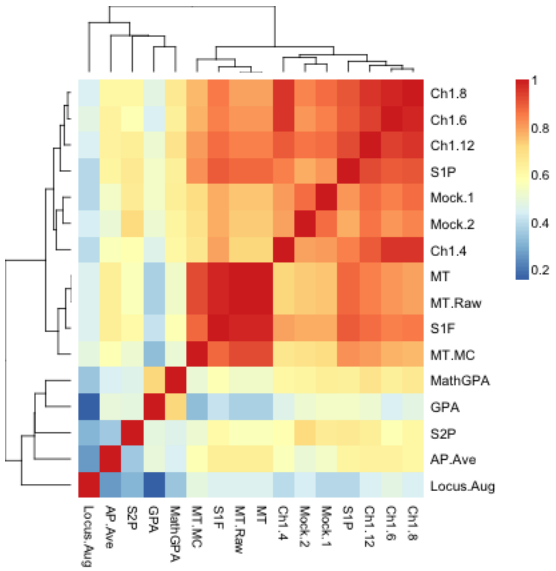
Why *maximize* variance – isn't that wrong? Usually we think we want low variability because we think of variability as noise. But that is only the case if we have remove the interesting patterns. Otherwise we can have variability in our variables due to patterns in the data. Consider this simple simulated 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.



Example on real data Here is data on scores of students taking AP statistics.

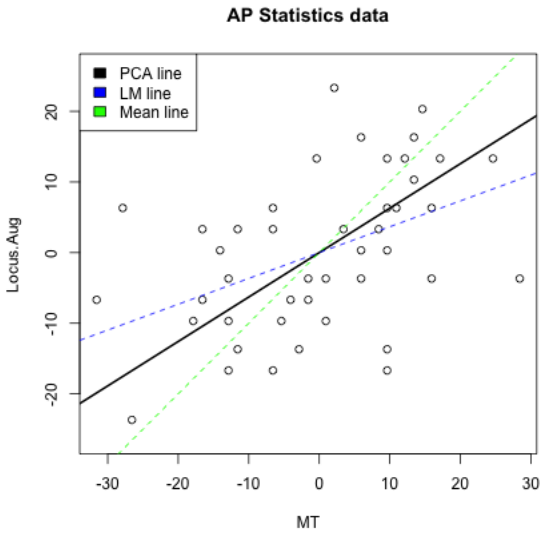
```
apscores <- read.csv(file = file.path(dataDir, "AP_Statistics_Predictions_2013-16.csv
    header = TRUE, stringsAsFactors = TRUE)
library(NMF)
aheatmap(cor(apscores[, -c(1, 3, 10, 13, 21:29)], use = "pairwise.complete.obs"))
```

[4] Up to a constant; PCA uses $\frac{1}{\sqrt{2}}$ rather than $\frac{1}{2}$ for the constant



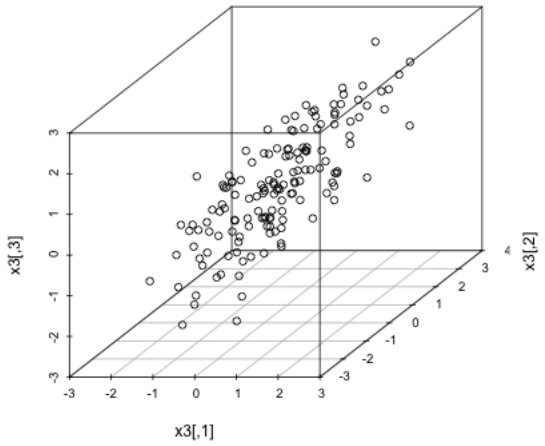
Not surprisingly, many of these measures are highly correlated.

Let's look at 2 scores, the midterm score and the pre-class evaluation and consider how to summarize them using PCA.



4.3 More than 2 variables

We could similarly combine three measurements



Now a good summary of our data would be a line that goes through the cloud of points. Just as in 2 dimensions, this corresponds to a linear combination of the three variables.

$$z_i = a_1 x_i^{(1)} + a_2 x_i^{(2)} + a_3 x_i^{(3)}$$

The exact same principles hold. Namely, that we look for the line with the smallest average distance to the line from the points. Only now distance is in 3 dimensions, rather than 2. This is given by the Euclidean distance, that we discussed earlier.

Many variables We can of course expand this to as many variables as we want. Specifically, any observation i is a vector of values, $(x_i^{(1)}, \ldots, x_i^{(p)})$ where p is the number of variables. With PCA, I am looking for a **linear combination** of these p variables. This means some set of adding and subtracting of these variables to get a new variable z,

$$z_i = a_1 x_i^{(1)} + \ldots + a_p x_i^{(p)}$$

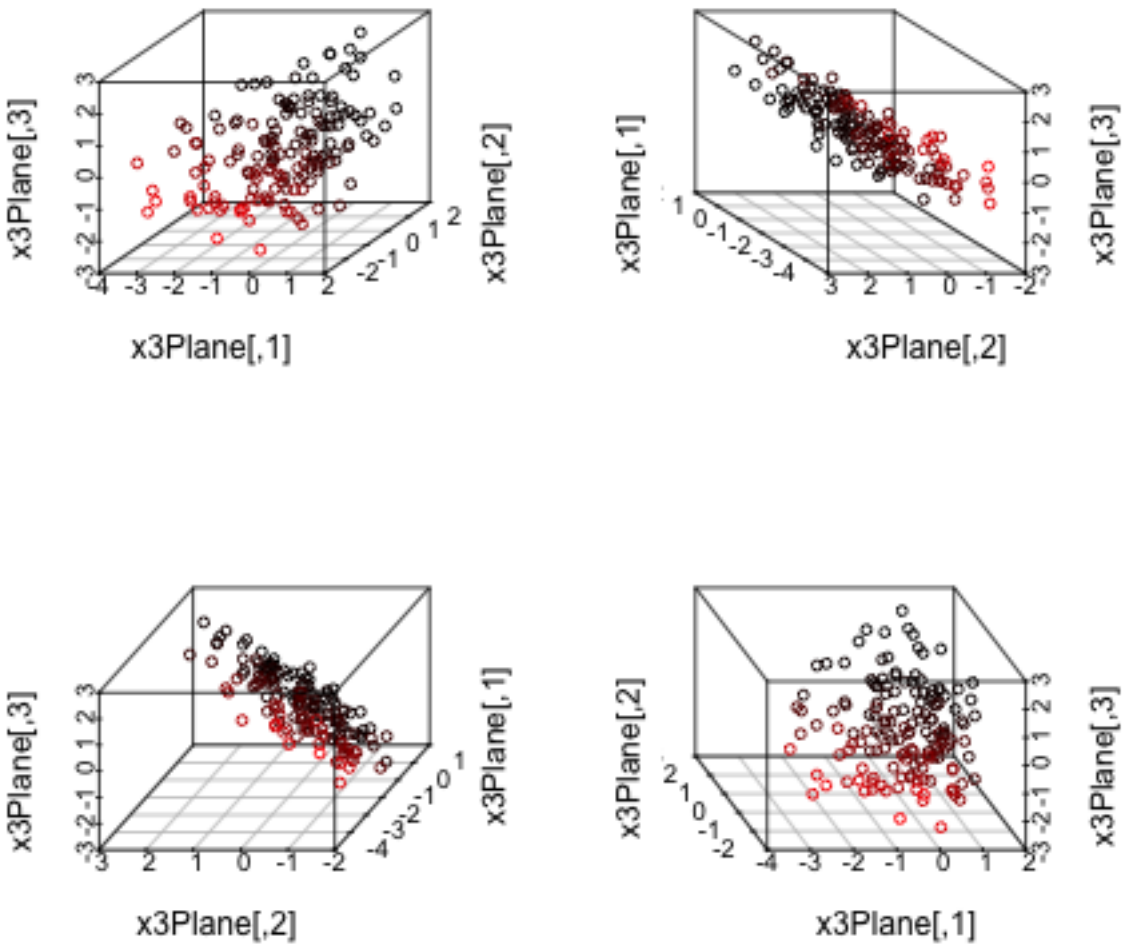
So a linear combination is just a set of p constants that I will multiply my variables by.

Q: 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. The geometric story stays the same too, though it's harder to visualize in higher dimensions than 3.

4.4 Adding another principal component

What if instead my three scores look like this?



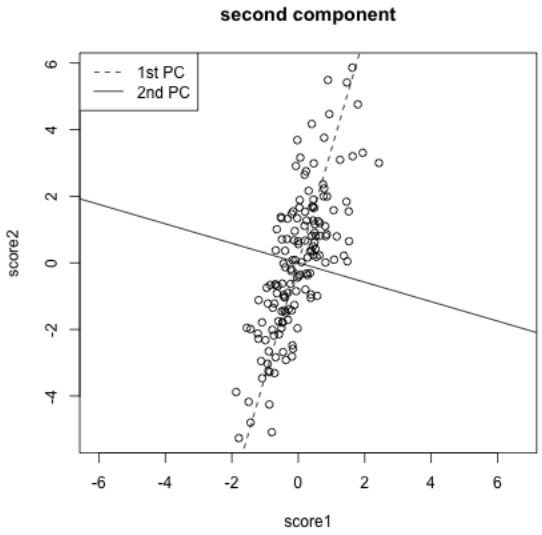
I would can get one line through the cloud of points, corresponding to my best linear combination of the three variables. But I might worry whether this really represented my data, since as we rotate the plot around we can see that my points appear to be closer to a lying near a plane than a single line. So there's some redundancy, but it's not clear there's only 1 variable (i.e. line) that can describe this cloud of data geometrically.

Then I might ask whether I could better summarize these three variables by two variables. But I have to be careful. In the geometric sense, if I draw my best line, I

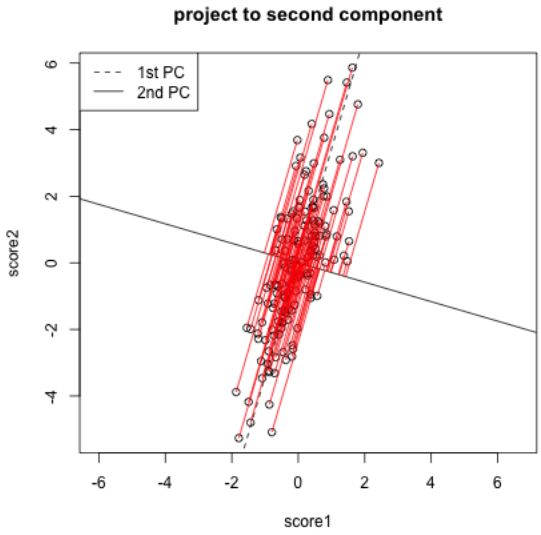
need to define what is "next best".

Example of 2 dimensions Let's return to our 2-dim example to consider this. 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.

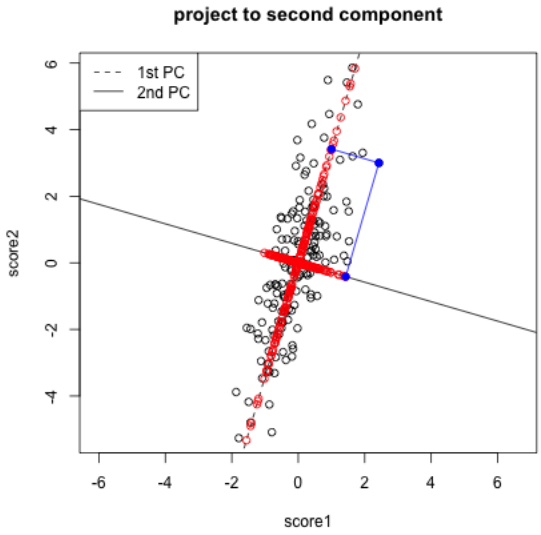
Moreover, such a line that is close to the best doesn't give me very different information from my best line. So I need to force "next best" to be separated and distinct from my best line. How do we do that? We make the requirement that the next best line be orthogonal from the best line. In two dimensions that's a pretty strict constraint – there's only 1 such line! (at least that goes through the center of the points).



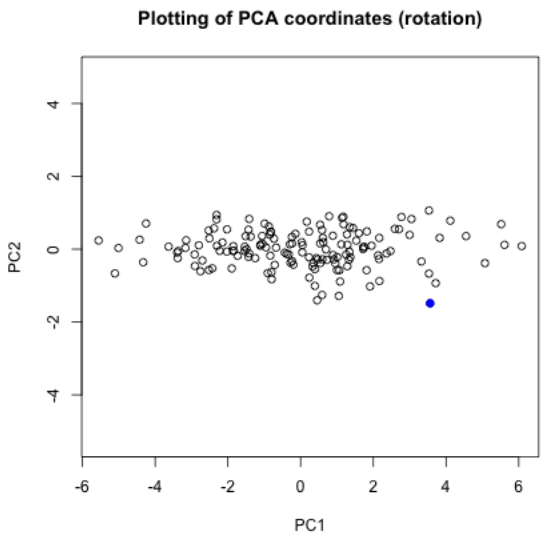
Two Principal Components define two variables z Then we can carry the idea forward, and and projecting our data onto this next best line.



This means we now have two variables: $z_i^{(1)}$ is our best, and $z_i^{(2)}$ is our second best.



So going from 1 principal component to 2 principal components is like defining two variables. We can imagine that now we have a new dataset, with the same set of observations, only now with the two variables defined by $z_i^{(1)}$ and $z_i^{(2)}$. 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.

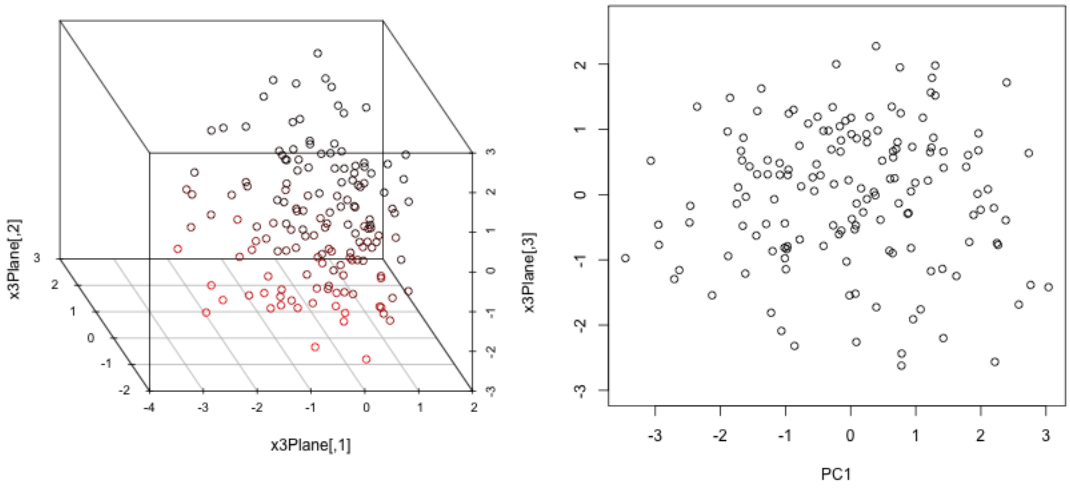


Preserving distances In two dimensions, we completely recapture the pattern of the data with 2 principal components – we've just rotated the picture, but the relationship of the points to each other (i.e. their distances to each other), are exactly the same. Of course this wasn't true if I looked at one dimension; the distances in the 1st PC variable are not the same as the distances in the whole dimension space.

So plotting the 2 PC variables instead of the 2 original variables doesn't tell us anything new about our data

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?

In 3 dimensions, the two PCs also define coordinates and we can make a scatter plot of the 2 top PCs. But unlike our 2D example, 2 top PCs don't preserve the entire dataset.

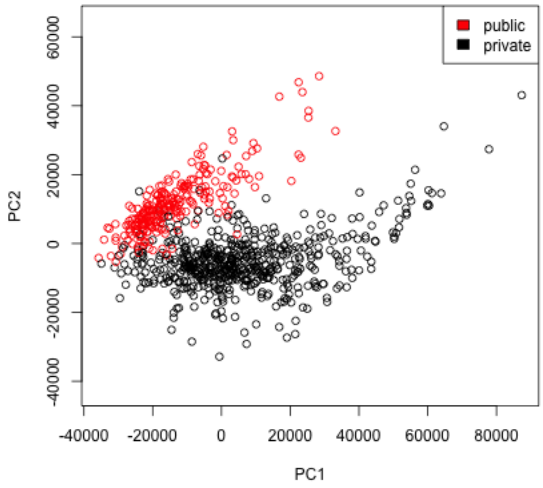


What they do show is the points as represented by that plane in 3d that we discussed above. So we can summarize the 3 dimensional cloud of points by this two dimensional cloud, which captures most of the variablity of the data. This is now a *summary* of the 3D data. Which is nice, since it's hard to plot in 3D. We will turn now to illustrating this on some real data examples.

4.5 Return to real data (2 PCs)

We can find the top 2 PCs for our real data examples and plot the scatter plot of these points. Consider the college dataset.

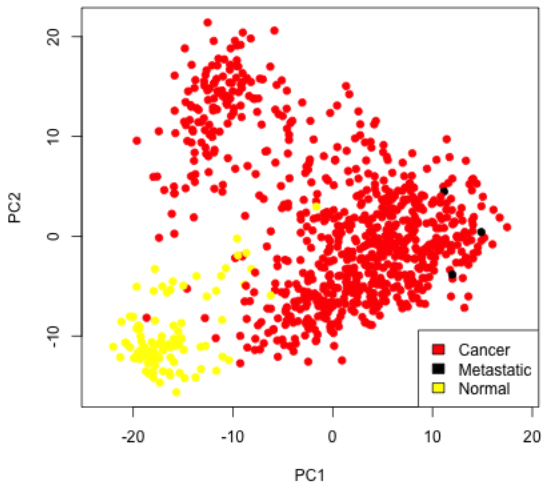
```
whNACollege <- attributes(na.omit(scorecard[, -c(1:3,
    12)]))[["na.action"]]
pcaCollege <- prcomp(scorecard[-whNACollege, -c(1:3,
    12)], center = TRUE, scale = FALSE)
plot(pcaCollege$x[, 1:2], col = c("red", "black")[scorecard$CONTROL[-whNACollege]],
    asp = 1)
legend("topright", c("public", "private"), fill = c("red",
    "black"))
```



This certainly shows us patterns among the data – what?

Similarly we can see a big difference between cancer and normal observations in the first two principal components.

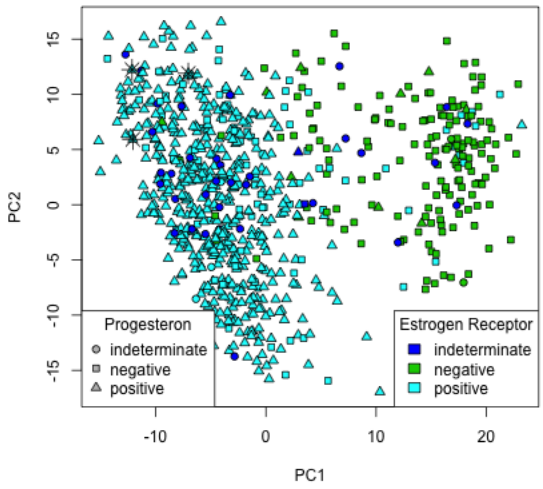
```
pcaBreast <- prcomp(breastCenteredMed[, -c(1:7)], center = FALSE,
    scale = TRUE)
plot(pcaBreast$x[, 1:2], col = typeCol[breast$TypeSample],
    asp = 1, pch = 19)
legend("bottomright", levels(breast$TypeSample), fill = typeCol)
```



Does PC1 separate normal from cancer? What about PC2?

If we remove the normal samples,

```
pcaBreastCancer <- prcomp(breastCenteredMed[whCancer,
    -c(1:7)], center = FALSE, scale = TRUE)
whMets <- which(breast$Type[whCancer] == "Metastatic")
plot(pcaBreastCancer$x[, 1:2], col = "black", bg = estCol[breast$EstReceptor[whCancer
    asp = 1, pch = c(21, 22, 24)[breast$Progesteron[whCancer]])
points(pcaBreastCancer$x[whMets, 1:2], pch = 8, cex = 2)
legend("bottomright", levels(breast$EstReceptor), fill = estCol,
    title = "Estrogen Receptor")
legend("bottomleft", levels(breast$Progesteron), pch = c(21,
    22, 24), col = "black", pt.bg = "grey", title = "Progesteron")
```



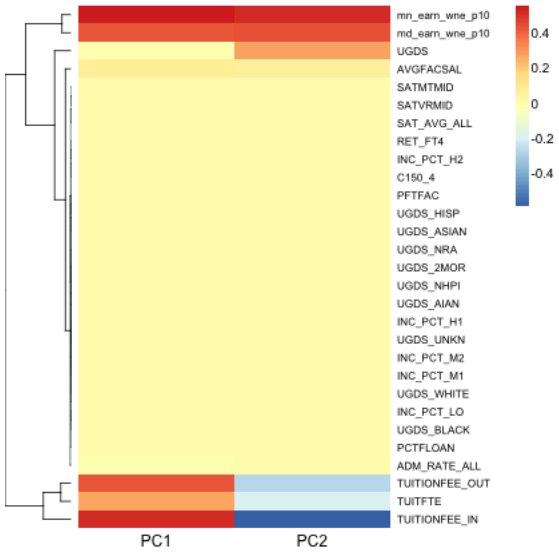
Does PC1 or PC2 separate the estrogen receptor or progesteron pathologies?

What about metastastic samples?

Loadings The scatterplots doen't tell us how the original variables relate to our new variables. These are sometimes called the loadings. We can go back to what

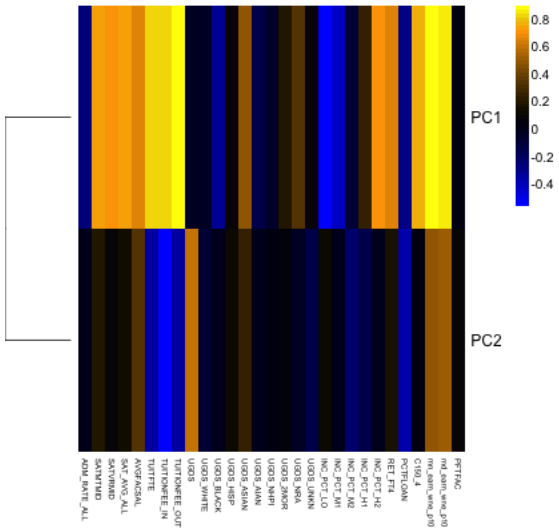
their coefficients are in our linear combination

```
aheatmap(pcaCollege$rotation[, 1:2], Colv = NA)
```



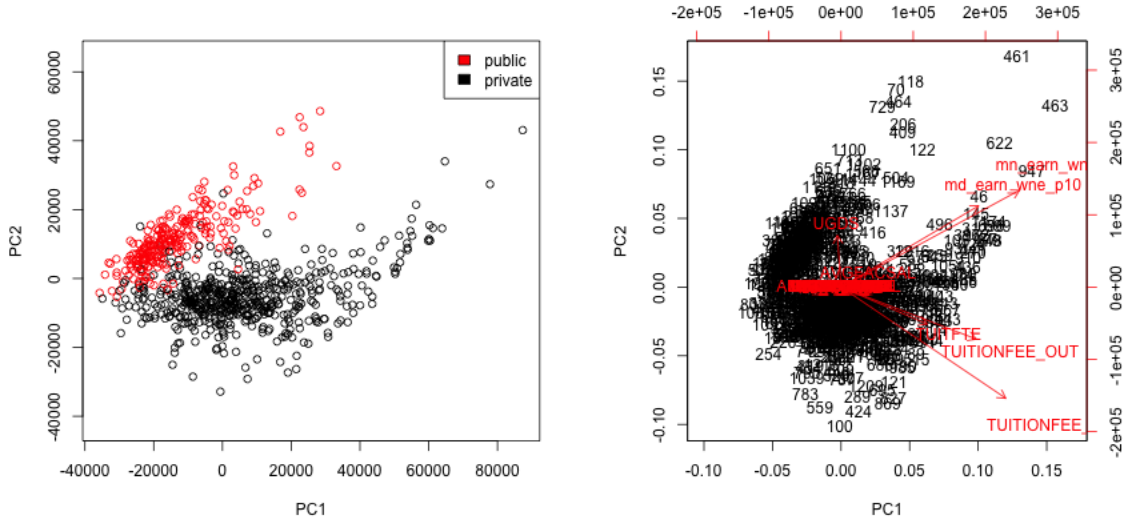
It's often interesting to look at the *correlation* between the new variables and the old variables

```
corPCACollege <- cor(pcaCollege$x, scale(scorecard[-whNACollege,
    -c(1:3, 12)], center = TRUE, scale = FALSE))
aheatmap(corPCACollege[1:2, ], Colv = NA, col = seqPal2)
```



Biplot We can put information regarding the variables together in what is called a biplot. Here the variables are plotted with vectors (i.e. arrows), while the observations are plotted as points based on their value of the 2 principal components.

```
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))
```



Notice that the axes values are not the same. This is because biplot is scaling the PC variables.

The arrow for a variable points in the direction that is most like that variable.[5] 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.[6] 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.

[5]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.

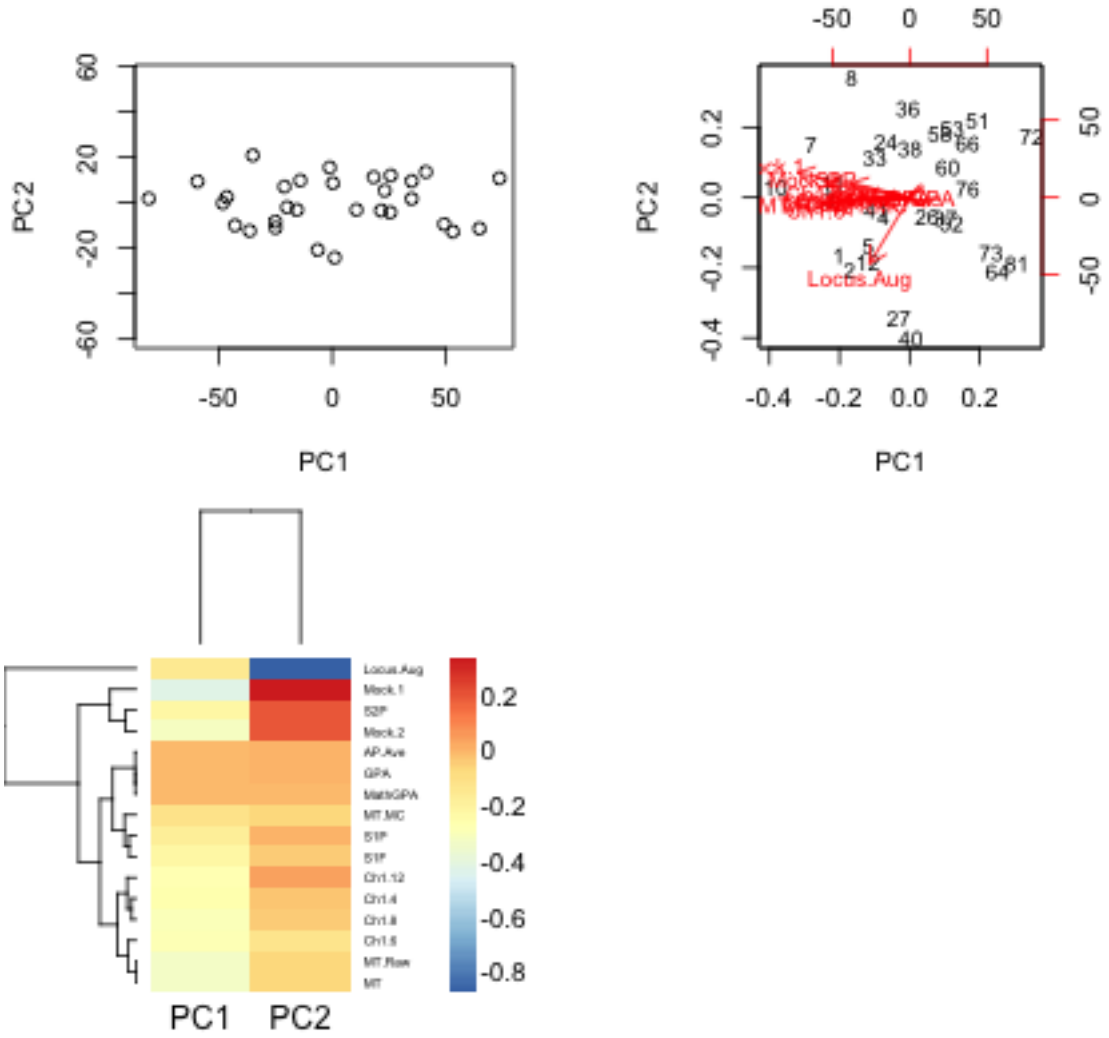
[6]Specifically the size of the correlation of the points projected onto that vector and the actual variable.

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 if we track in state tuition fee, we see that it points in the direction of the private universities, meaning higher values on that variable. The vector corresponding to in-state tuition fee is like an axis that separates public and private – not surprisingly since that's a big difference between them on these metrics.

AP Scores We can make the same plots for the AP scores

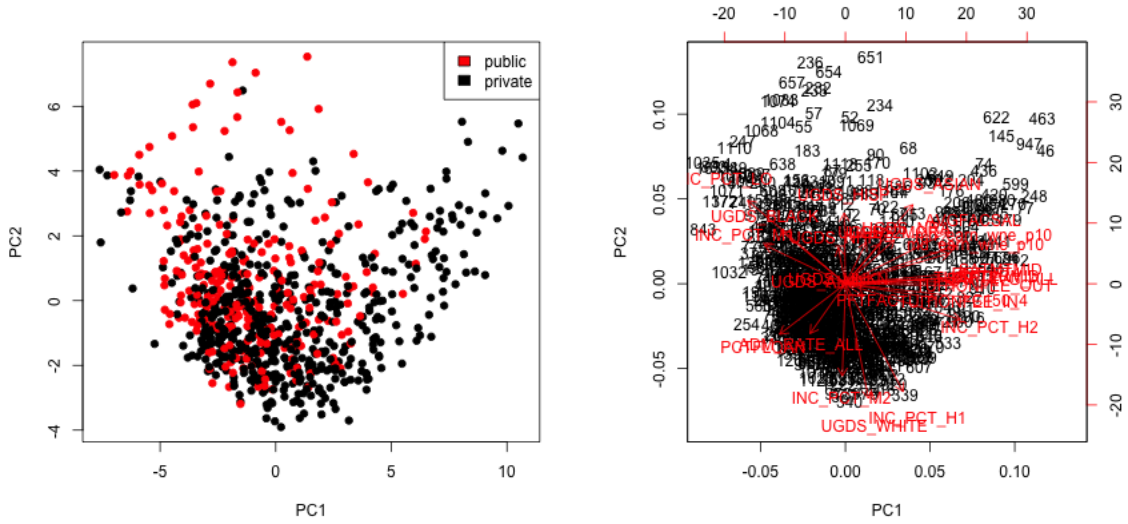
```
whNAAP <- as.numeric(attributes(na.omit(apscores[,
    -c(1, 3, 10, 13, 21:29)]))[["na.action"]])
pcaAP <- prcomp(apscores[-whNAAP, -c(1, 3, 10, 13,
    21:29)], center = TRUE, scale = FALSE)
par(mfrow = c(2, 2))
plot(pcaAP$x[, 1:2], asp = 1)
biplot(pcaAP)
aheatmap(pcaAP$rotation[, 1:2])
```



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 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.

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.

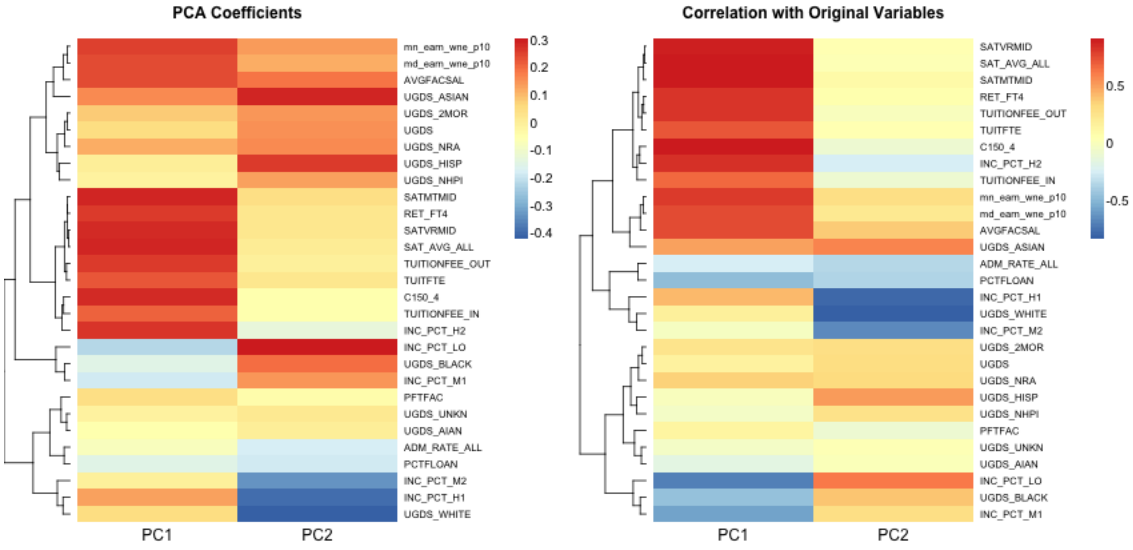
```
pcaCollegeScale <- prcomp(scorecard[-whNACollege, -c(1:3,
    12)], center = TRUE, scale = TRUE)
par(mfrow = c(1, 2))
plot(pcaCollegeScale$x[, 1:2], pch = 19, col = c("red",
    "black")[scorecard$CONTROL[-whNACollege]])
legend("topright", c("public", "private"), fill = c("red",
    "black"))
biplot(pcaCollegeScale, choices = c(1, 2), pch = 19)
```



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.

```
par(mfrow = c(1, 2))
aheatmap(pcaCollegeScale$rotation[, 1:2], Colv = NA,
    main = "PCA Coefficients")
aheatmap(cor(scorecard[-whNACollege, -c(1:3, 12)],
    pcaCollegeScale$x[, 1:2]), Colv = NA, main = "Correlation with Original Variables
```

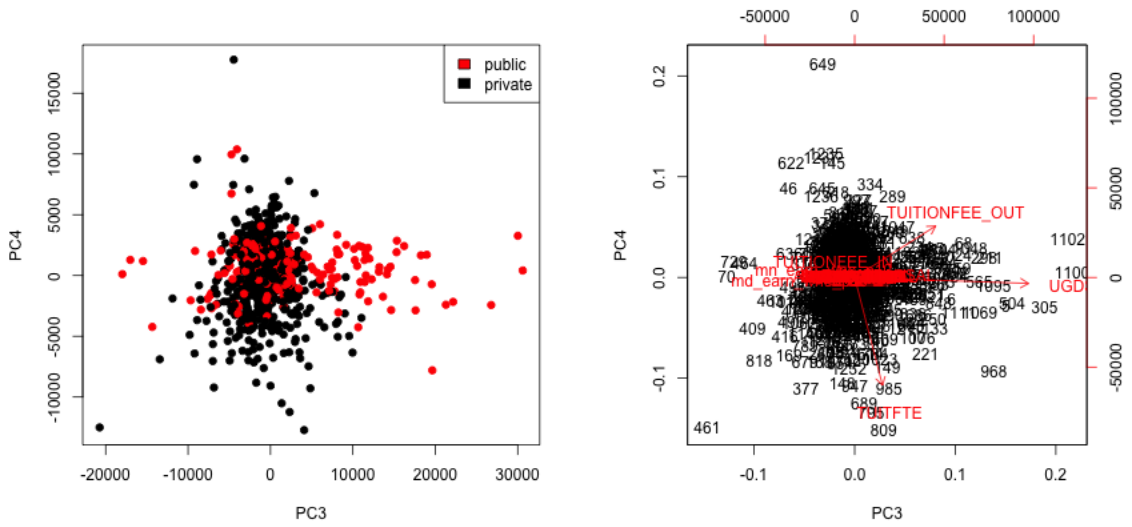


4.6 More than 2 PC coordinates

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.

```
par(mfrow = c(1, 2))
plot(pcaCollege$x[, 3:4], pch = 19, col = c("red",
    "black")[scorecard$CONTROL[-whNACollege]])
legend("topright", c("public", "private"), fill = c("red",
    "black"))
suppressWarnings(biplot(pcaCollege, choices = 3:4))
```



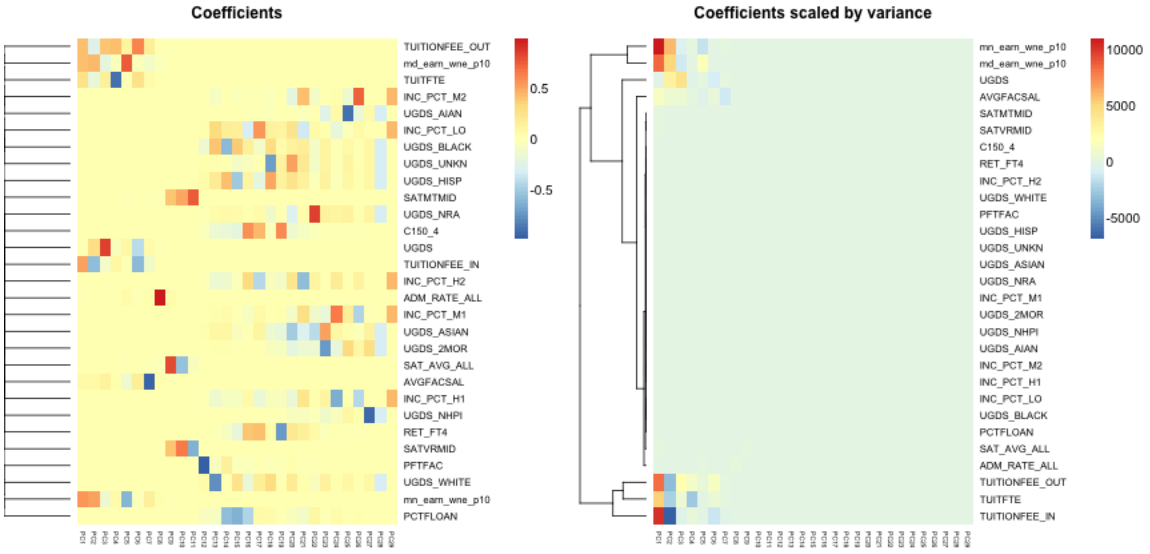
This is a very different set of coordinates for the points in 2 PCs. However, some of the same set of variables are still dominating, they are just different linear combinations of them (the two PCs lines are orthogonal to each other, but they can still just involve these variables because its such a 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. [7]

```
par(mfrow = c(1, 2))
aheatmap(pcaCollege$rotation, Colv = NA, main = "Coefficients")
aheatmap(sweep(pcaCollege$rotation, 2, pcaCollege$sdev,
    "*"), Colv = NA, main = "Coefficients scaled by variance")
```

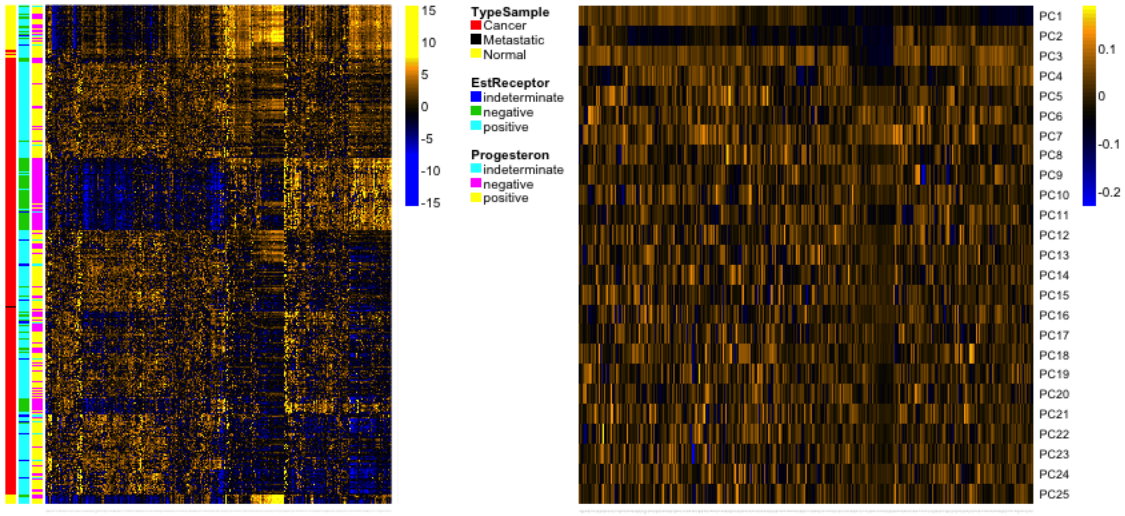
[7] We haven't discussed this, but in fact the coefficients are scaled so the sum of the square of the coefficients equal 1 (norm is one). Otherwise there's not a unique set of coefficients, since you could always just multiply all the coefficients by a number and get larger and larger variance. So the coefficients are all on the similar scale, regardless of the original variability or importance of the PC in explaining the data.



Breast data We can also look at the higher PCs from the breast data (with the normal samples). If there are 500 genes and 878 observations, how many PCs are there?

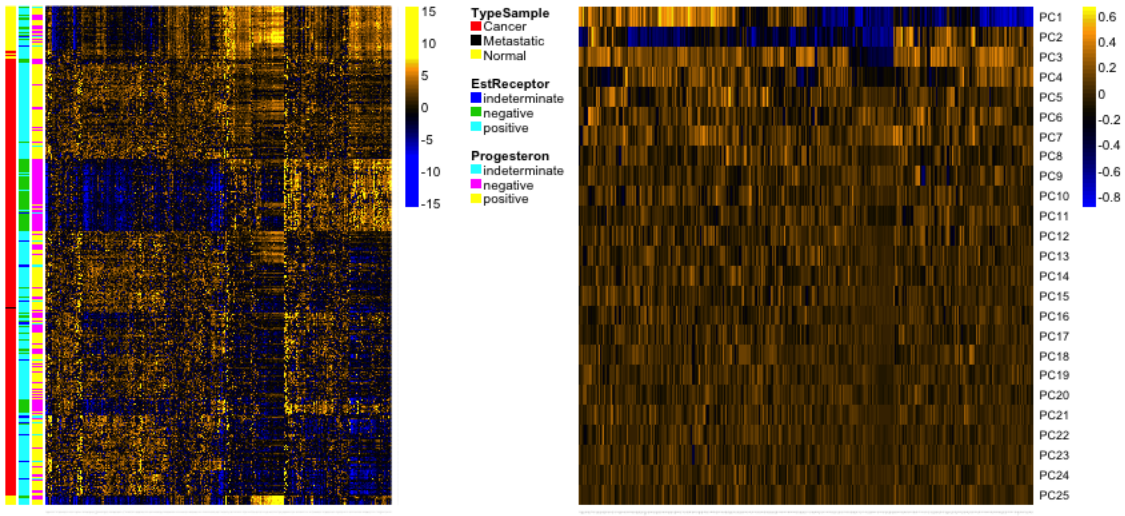
We can see that there are distinct patterns in what genes/variables contribute to the final PCs (we plot only the top 25 PCs). However, it's rather hard to see, because there are large values in later PCs that mask the pattern.

```
par(mfrow = c(1, 2))
breastHeat <- aheatmap(breastCenteredMed[, -c(1:7)],
    Rowv = FALSE, Colv = FALSE, breaks = brksCentered,
    annRow = breast[, 5:7], labRow = NA, col = seqPal2,
    annColors = list(TypeSample = typeCol, EstReceptor = estCol,
        Progesteron = proCol))
aheatmap(t(pcaBreast$rotation[, 1:25]), Colv = breastHeat$colInd,
    col = seqPal2, Rowv = NA)
```



This is an example of why it is useful to scale the variables by their variance

```
par(mfrow = c(1, 2))
aheatmap(breastCenteredMed[, -c(1:7)], Rowv = FALSE,
    Colv = FALSE, breaks = brksCentered, annRow = breast[,
        5:7], labRow = NA, col = seqPal2, annColors = list(TypeSample = typeCol,
        EstReceptor = estCol, Progesteron = proCol))
scaledPCs <- sweep(pcaBreast$rotation, 2, pcaBreast$sdev,
    "*")
aheatmap(t(scaledPCs[, 1:25]), Colv = breastHeat$colInd,
    col = seqPal2, Rowv = NA)
```

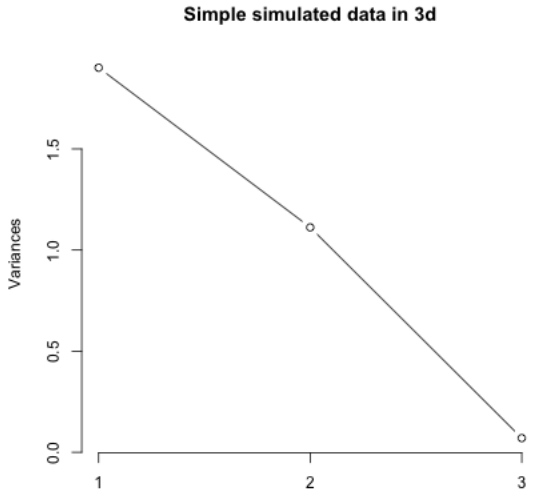


4.7 How many dimensions?

If I can draw my data in 3d, then I can guess what is the write number of coordinates – not 1 but 2 in this case are 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.

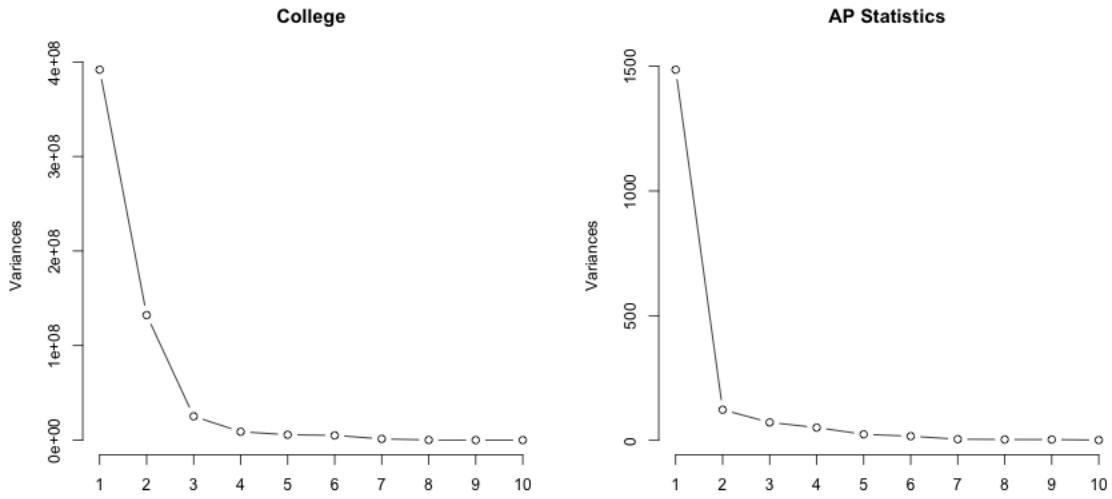
Consider our simple simulation example, where there was more or less a plane describing the data. If we look at the variance in each set of linear combinations we create, there is practically 0 left in the last variable, meaning that most of the representation of the points is captured in two dimensions. This is a measure of how much we are "missing" by ignoring a coordinate.

```
screeplot(pca3d, type = "lines", main = "Simple simulated data in 3d")
```



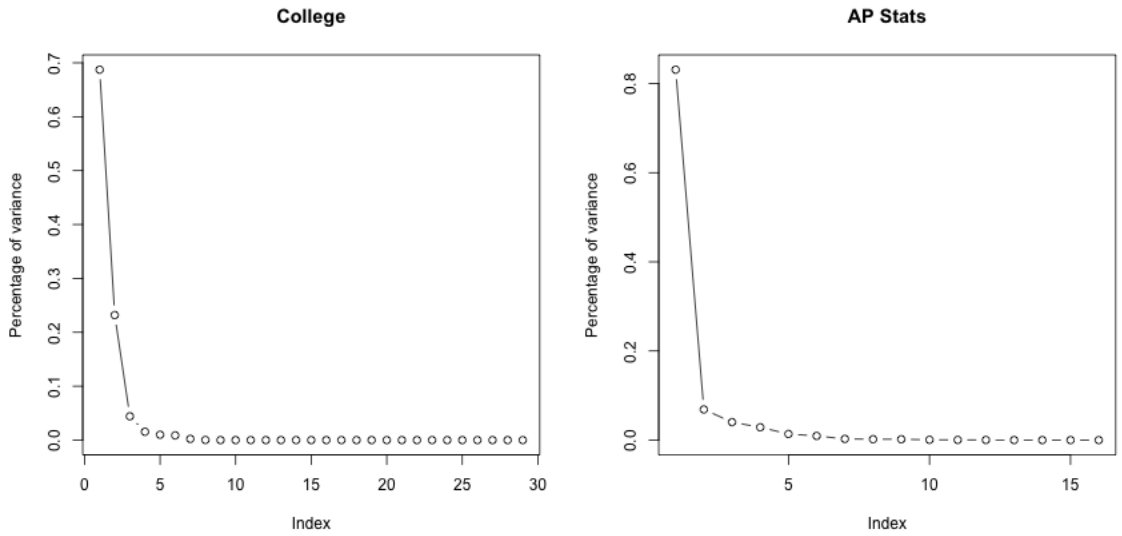
For the college data, we similarly see that the first two dimensions both have much larger amounts compared to other dimensions. The AP Statistics data is strongly in just the first dimension.

```
par(mfrow = c(1, 2))
screeplot(pcaCollege, type = "lines", main = "College")
screeplot(pcaAP, type = "lines", main = "AP Statistics")
```



We can also plot this a percentage

```
par(mfrow = c(1, 2))
plot(pcaCollege$sdev^2/sum(pcaCollege$sdev^2), type = "b",
    main = "College", ylab = "Percentage of variance")
plot(pcaAP$sdev^2/sum(pcaAP$sdev^2), type = "b", main = "AP Stats",
    ylab = "Percentage of variance")
```



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. What does this tell you about the PCA?

```
par(mfrow = c(1, 2))
plot(pcaBreast$sdev^2/sum(pcaBreast$sdev^2), type = "b",
    main = "Breast PCA", ylab = "Percentage of variance",
    xlim = c(0, 100))
plot(pcaBreastCancer$sdev^2/sum(pcaBreastCancer$sdev^2),
    type = "b", main = "Breast, Cancer only, PCA",
    ylab = "Percentage of variance", xlim = c(0, 100))
```