Lecture 5

R "Programming" + Statistics + Data Analysis Example

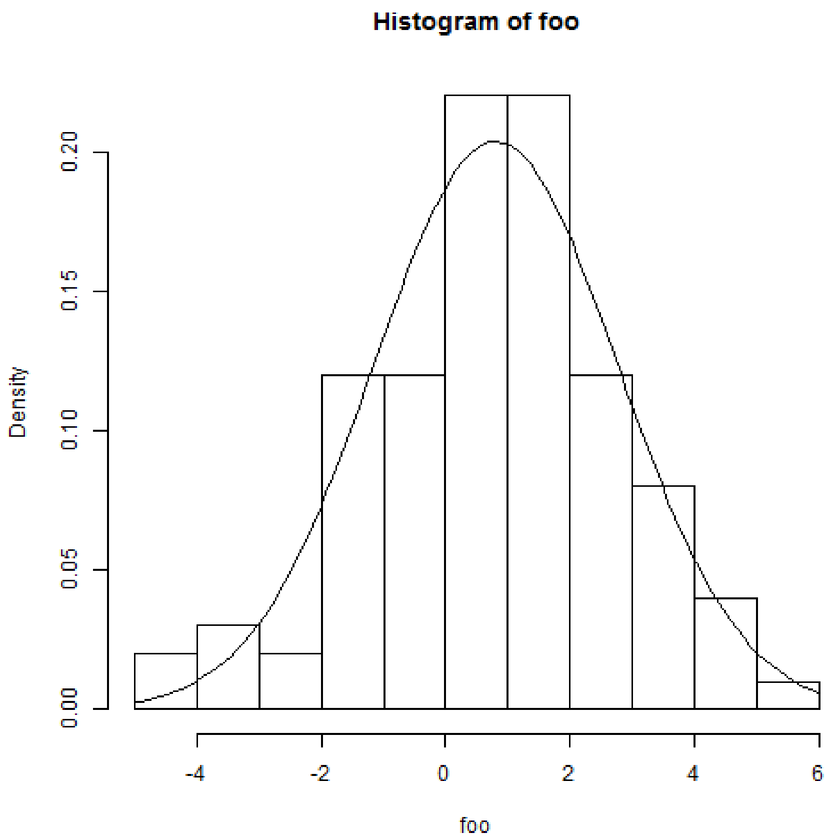
Andrew Jaffe
Instructor

Summary of functions yesterday

- plot() - multi-use function/workhorse that plots
- points/lines() - add points/lines to a plot that's already been made
- abline() - just makes a line, horizontal (h), vertical (v), or mx + b (using a, b)
- axis function - can add an axis to an R plot, if you plot with xaxt ="n", then run axis(1, options) to make a customized axis
- boxplot, hist, plot(density(x)), barplot - self-explanatory
- scatter.smooth - smooths x and y, plots the smoothed line and points
  - Similar to lowess in stata
- lattice graphics, like levelplot (levels/heat map)
- text() - adds text to a plot using x, y coordinates
- legend() - creates a legend for your plot
- RColorBrewer - better Colors

Curve

```
> foo <- rnorm(100, mean = 1, sd = 2)
> hist(foo, prob = TRUE)
> curve(dnorm(x, mean = mean(foo), sd = sd(foo)), add = TRUE)
```



Aggregate

- So we didn't cover some other "by" functions, but aggregate is good:
  - aggregate - another "by" function, maybe more intuitive/powerful than tapply
- There are others - but each may output different data types

```
> Sal$AnnualSalary <- as.numeric(gsub(Sal$AnnualSalary, pattern = "$", replacement = "",
+     fixed = TRUE))
> Sal$GrossPay <- as.numeric(gsub(Sal$GrossPay, pattern = "$", replacement = "",
+     fixed = TRUE))
> head(aggregate(Sal[, c("AnnualSalary", "GrossPay")], by = list(Sal$Agency),
+     mean, na.rm = TRUE))
```

```
 AnnualSalary GrossPay
1                Circuit Court        53425    45404
2                 City Council        45528    33303
3          Community Relations        52630    43193
4                  COMP-Audits        63587    54902
5 COMP-Communication Services        36801    27819
6    COMP-Comptroller's Office        68681    59226
```

Categorical Statistics

- Let's look at some tests for categorical data
- fisher.test - fisher's exact test
- chisq.test - chi-squared test
  - Of those that were adult, and didn't survive the titanic, did gender play a role

```
> DF <- data.frame(Titanic)
> print(tab <- xtabs(Freq ~ Sex + Survived, DF))
```

```
        Survived
Sex        No  Yes
  Male   1364  367
  Female  126  344
```

Categorical Statistics

- Let's look at what the summary is for Fisher's test:

```
> fisher.test(tab)
```

```

	Fisher's Exact Test for Count Data

data:  tab
p-value < 2.2e-16
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
  7.977 12.929
sample estimates:
odds ratio 
     10.13 
```

Categorical Statistics

- Let's look at what the summary is for a Chi-square test:

```
> chisq.test(tab)
```

```

    Pearson's Chi-squared test with Yates' continuity correction

data:  tab
X-squared = 454.5, df = 1, p-value < 2.2e-16
```

Parametric tests

- t.test - t.test(df$Y[df$Trt == 1], df$Y[df$Trt == 0]) or t.test(Y~ Trt, data=df)
- the second syntax is an expression or formula
- These are the basic syntax for models and can be used many other places

```
> twolanes <- bike[bike$type %in% c("BIKE LANE", "SHARE THE ROAD"), ]
> twolanes$type <- factor(twolanes$type)
> t.test(twolanes$length[twolanes$type == "BIKE LANE"], twolanes$length[twolanes$type ==
+     "SHARE THE ROAD"])
```

```

    Welch Two Sample t-test

data:  twolanes$length[twolanes$type == "BIKE LANE"] and twolanes$length[twolanes$type == "SHARE THE ROA
t = 0.7458, df = 122.3, p-value = 0.4572
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -33.40  73.78
sample estimates:
mean of x mean of y
    260.2     240.0
```

T-test - not so much typing

- That's a lot of typing for one t-test, let's just use formula syntax
- Also has better output - note the levels in the mean

```
> t.test(length ~ type, data = twolanes)
```

```

    Welch Two Sample t-test

data:  length by type
t = 0.7458, df = 122.3, p-value = 0.4572
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -33.40  73.78
sample estimates:
     mean in group BIKE LANE mean in group SHARE THE ROAD
                       260.2                        240.0
```

ANOVA

- aov - it does ANOVA!
- Is bike lane length different by type?

```
> bike$type[bike$type == ""] <- NA
> bike$type <- factor(bike$type)
> print(bike.anova <- aov(length ~ type, data = bike))  ## WHAT!! NO P-values!
```

```
Call:
   aov(formula = length ~ type, data = bike)

Terms:
                     type Residuals
Sum of Squares    1059082 159753294
Deg. of Freedom         8      2910

Residual standard error: 234.3
Estimated effects may be unbalanced
7 observations deleted due to missingness
```

Summary - very useful

- Summary summarizes the model/data depending what's passed in.
- It gives you a lot of relevant statistics that you want, and have some better formatted objects

```
> summary(bike.anova)
```

```
              Df   Sum Sq Mean Sq F value Pr(>F)
type           8 1.06e+06  132385    2.41  0.014 *
Residuals   2910 1.60e+08   54898
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
7 observations deleted due to missingness
```

```
> summary(aov(log(length) ~ type, data = bike))  #log transformed
```

```
              Df Sum Sq Mean Sq F value Pr(>F)
type           8     16   2.020    2.95 0.0027 **
Residuals   2910   1989   0.684
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
7 observations deleted due to missingness
```

ANOVA

Note this gets returned in the summary of a linear regression model:

```
> summary(lm(length ~ type, data = bike))$fstat
```

```
   value     numdf     dendf
   2.411     8.000  2910.000
```

Nonparametric Tests

- Nonparametric tests do not assume normality or a distribution of the statistic usually (many times have exact p-values)
  - wilcox.test - performs rank-sum and signed rank tests
  - kruskal.test - nonparametric ANOVA equivalent (wilcox.test for more than 2 groups)
  - Friedman's test - a nonparametric repeated measures ANOVA
- Use paired=TRUE for signed rank, just like paired t-test.

Wilcox Rank Sum vs T-tests

```
> t.test(length ~ type, data = twolanes)
```

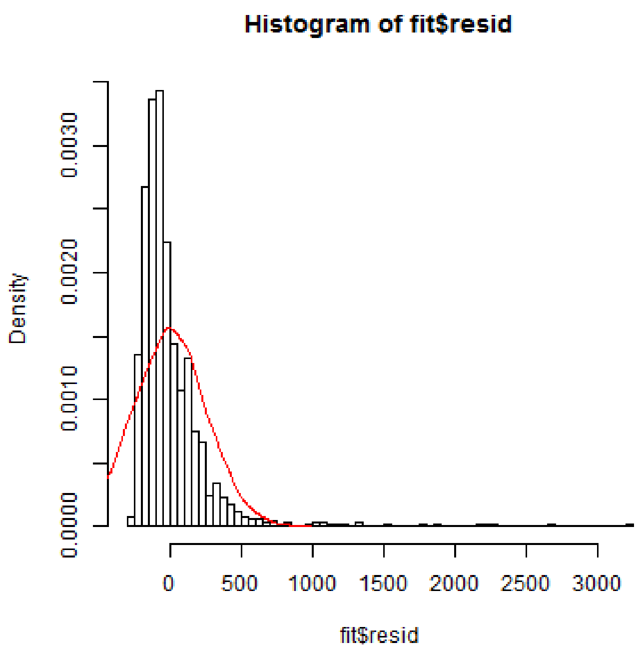
```

    Welch Two Sample t-test

data:  length by type
t = 0.7458, df = 122.3, p-value = 0.4572
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -33.40  73.78
sample estimates:
     mean in group BIKE LANE mean in group SHARE THE ROAD
                       260.2                        240.0
```

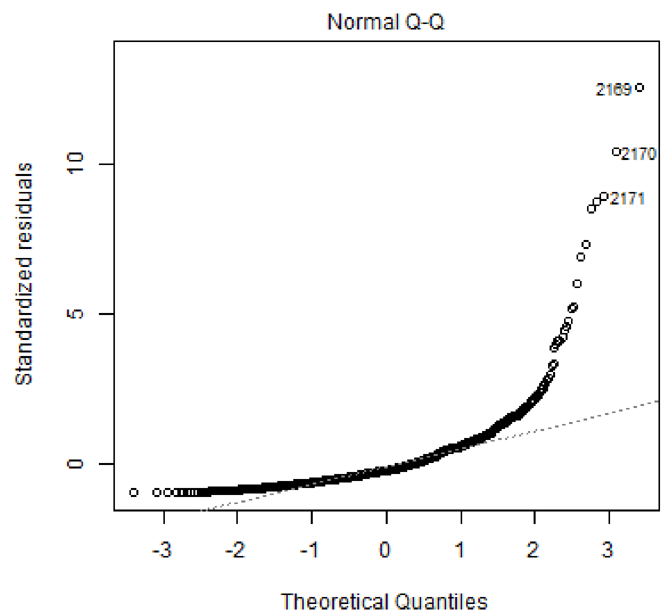
```
> wilcox.test(length ~ type, data = twolanes)
```

```

    Wilcoxon rank sum test with continuity correction

data:  length by type
W = 83990, p-value = 0.0222
alternative hypothesis: true location shift is not equal to 0
```

Wilcox Rank Sum vs T-tests

```
> par(mfrow = c(1, 2))
> fit = lm(length ~ type, data = twolanes)
> hist(fit$resid, breaks = 50, freq = F)
> lines(density(rnorm(10000, mean = mean(fit$resid), sd = sd(fit$resid))), col = "red")
> plot(fit, 2)
```





Wilcox Rank Sum vs T-tests

- Only 2 groups? Wilcox.test and kruskal.test are the same!

```
> kruskal.test(length ~ type, data = twolanes)
```

```

    Kruskal-Wallis rank sum test

data:  length by type
Kruskal-Wallis chi-squared = 5.231, df = 1, p-value = 0.02219
```

```
> wilcox.test(length ~ type, data = twolanes)
```

```

    Wilcoxon rank sum test with continuity correction

data:  length by type
W = 83990, p-value = 0.0222
alternative hypothesis: true location shift is not equal to 0
```

Nonparametric Tests

- Is bike lane length different by type?

```
> kruskal.test(length ~ type, data = bike)
```

```

	Kruskal-Wallis rank sum test

data:  length by type
Kruskal-Wallis chi-squared = 27.45, df = 8, p-value = 0.0005915
```

```
> kruskal.test(log(length) ~ type, data = bike)  ## same because only based on ranks
```

```

	Kruskal-Wallis rank sum test

data:  log(length) by type
Kruskal-Wallis chi-squared = 27.45, df = 8, p-value = 0.0005915
```

Friedman's Test -

Again, non-Parametric repeated measures ANOVA

```
> ## from friedman.test documentation
> head(RoundingTimes <- matrix(c(5.4, 5.5, 5.55, 5.85, 5.7, 5.75, 5.2, 5.6, 5.5,
+     5.55, 5.5, 5.4, 5.9, 5.85, 5.7, 5.45, 5.55, 5.6, 5.4, 5.4, 5.35, 5.45, 5.5,
+     5.35, 5.25, 5.15, 5, 5.85, 5.8, 5.7, 5.25, 5.2, 5.1, 5.65, 5.55, 5.45, 5.6,
+     5.35, 5.45, 5.05, 5, 4.95, 5.5, 5.5, 5.4, 5.45, 5.55, 5.5, 5.55, 5.55, 5.35,
+     5.45, 5.5, 5.55, 5.5, 5.45, 5.25, 5.65, 5.6, 5.4, 5.7, 5.65, 5.55, 6.3,
+     6.3, 6.25), nrow = 22, byrow = TRUE, dimnames = list(1:22, c("Round Out",
+     "Narrow Angle", "Wide Angle"))), 2)
```

```
  Round Out Narrow Angle Wide Angle
1      5.40          5.5       5.55
2      5.85          5.7       5.75
```

```
> friedman.test(RoundingTimes)
```

```

    Friedman rank sum test

data:  RoundingTimes
Friedman chi-squared = 11.14, df = 2, p-value = 0.003805
```

Generalized Linear Models

- Generalized Linear models allow you to model non-normal data
  - Poisson, Logistic, Negative Binomial, etc.
- Use the glm function!

```
> cars = read.csv("http://biostat.jhsph.edu/~ajaffe/files/kaggleCarAuction.csv",
+     as.is = T)
> summary(glm(IsBadBuy ~ VehBCost, data = cars, family = "binomial"))
```

```

Call:
glm(formula = IsBadBuy ~ VehBCost, family = "binomial", data = cars)

Deviance Residuals:
   Min       1Q   Median       3Q      Max
-0.783   -0.550   -0.484   -0.430    4.268

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -7.68e-01   4.43e-02   -17.3   <2e-16 ***
VehBCost    -1.83e-04   6.78e-06   -27.1   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 54421  on 72982  degrees of freedom
Residual deviance: 53660  on 72981  degrees of freedom
AIC: 53664
```

Stepwise Regression

- There is a function called `step` that does stepwise regression
  - We do NOT recommend this for analysis
  - Does stepwise regression based on AIC (Akaike Information Criterion), not p-values! (Or any loss function)
  - It may be interesting to see what model is chosen, but not as your main analysis tool
- Also a function called leaps, in library(leaps) that does all subsets regression
- You should consult someone if you don't what to model, unless you're doing exploratory data analysis

R 'programming'

Now we are going to switch gears a little bit, and talk about some of the more traditional programming that you can do in R.

You can do very flexible things, but at a cost of more difficult notation, and having to actually write programming statements. There are slight notation differences as well, including the use of curly '{}' brackets

We are going to cover 'for' loops and 'if' statements

'for' Loops

These allow you to iterate over certain observations or subsets of observations

The syntax is:

```
for(*var* in seq) {
do something
}
```

Typically they look something like:

```
for(i in 1:nrow(dat)) {
  something(dat[i,])
}
```

'for' loops

These are essentially fancier 'apply' statements

For example,

```
> for (i in 1:10) {
+     print(i)
+ }
```

```
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
[1] 7
[1] 8
[1] 9
[1] 10
```

'for' loops

Here's how they can be more flexible:

```
> Index = c(3, 6, 7, 20, 32, 100, 234, 1000, 6543)
> for (i in 1:length(Index)) {
+     print(Index[i])
+ }
```

```
[1] 3
[1] 6
[1] 7
[1] 20
[1] 32
[1] 100
[1] 234
[1] 1000
[1] 6543
```

Note that the first time through the body of the loop, 'i' takes the value 1, then evaluates the body. Then, 'i' takes the value 2, and evaluates the body, until i = length(Index), then it stops.

'for' loops

They are essentially more useful than apply statements when you are working with two sets of matching datasets or vectors.

```
> myList = vector("list", length = 4)
> mat1 = matrix(rnorm(8), nc = 4)
> mat2 = matrix(rnorm(8), nc = 4)
> mat1
```

```
        [,1]    [,2]    [,3]    [,4]
[1,]  1.3446 -0.1687  0.0594  0.7942
[2,] -0.6142  0.2090 -0.3965 -0.3027
```

```
> mat2
```

```
        [,1]   [,2]   [,3]     [,4]
[1,] -0.7344 -0.481 0.2722 0.008351
[2,] -0.4861  1.668 1.5507 1.146078
```

```
> for (i in seq(along = myList)) {
+     myList[[i]] = cbind(mat1[, i], mat2[, i])
+ }
> myList
```

```
[[1]]
        [,1]    [,2]
[1,]  1.3446 -0.7344
[2,] -0.6142 -0.4861

[[2]]
        [,1]   [,2]
[1,] -0.1687 -0.481
[2,]  0.2090  1.668

[[3]]
        [,1]   [,2]
[1,]  0.0594 0.2722
[2,] -0.3965 1.5507

[[4]]
        [,1]     [,2]
[1,]  0.7942 0.008351
[2,] -0.3027 1.146078
```

'for' loops

```
> i = 1
> cbind(mat1[, i], mat2[, i])
```

```
        [,1]    [,2]
[1,]  1.3446 -0.7344
[2,] -0.6142 -0.4861
```

```
> i = 2
> cbind(mat1[, i], mat2[, i])
```

```
        [,1]   [,2]
[1,] -0.1687 -0.481
[2,]  0.2090  1.668
```

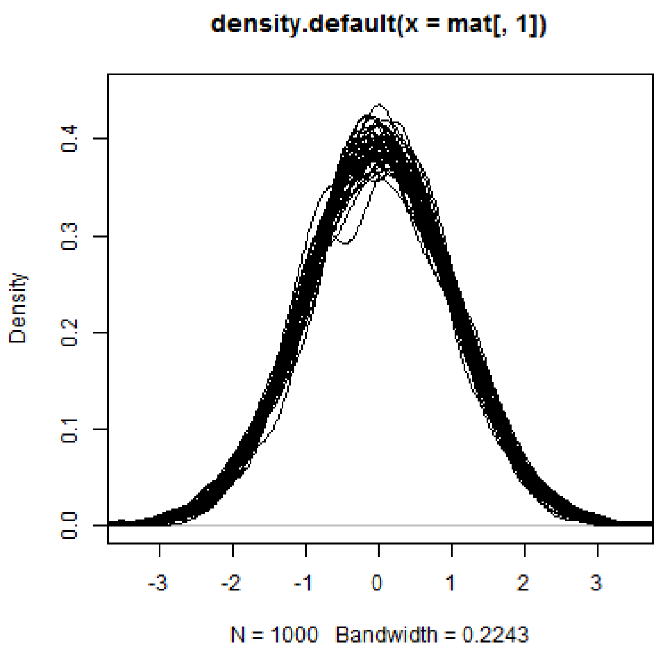
```
> i = 3
> cbind(mat1[, i], mat2[, i])
```

```
        [,1]   [,2]
[1,]  0.0594 0.2722
[2,] -0.3965 1.5507
```

'for' loops

These are useful for making many columns worth of density plots

```
> mat = matrix(rnorm(1000 * 50), nc = 50)
> plot(density(mat[, 1]), ylim = c(0, 0.45))
> for (i in 2:ncol(mat)) {
+     lines(density(mat[, i]))
+ }
```



'for' loops

You can also integrate with lists.

```
> outList = vector("list", 10)
> start = 1:10
> end = sample(1:100, 10)
> for (i in seq(along = outList)) {
+     outList[[i]] = start[i]:end[i]
+ }
> outList
```

```
[[1]]
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
[24] 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46
[47] 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69
[70] 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84

[[2]]
 [1]  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
[24] 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
[47] 48 49 50 51 52 53 54 55 56 57 58 59 60 61

[[3]]
 [1]  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
[24] 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48
[47] 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71
[70] 72 73 74 75 76 77

[[4]]
[1] 4

[[5]]
```

'if' statements

You can put 'if' statements inside of 'for' loops

```
for(i in 1:nrow(dat)) {
  if(dat$x > num) {
    dat$y[i] = something
  } else {
    dat$y[i] = something else
  }
}
```

Break/stop

You can use break; and stop; to end loops early, typically combined with a logical statement using if. However, errors can be signs that the loop is not doing what you want, so it might be good to avoid these until you get more comfortable using loops.

```
> dummy <- FALSE
> for (ii in 1:5) {
+     for (jj in 2:5) {
+         cat("ii=", ii, "; jj=", jj, "\n", sep = "")
+         if (ii == jj) {
+             dummy <- TRUE
+             break
+         }
+     }
+     if (dummy)
+         break
+ }
```

```
ii=1; jj=2
ii=1; jj=3
ii=1; jj=4
ii=1; jj=5
ii=2; jj=2
```

Report generation

Now we are going to combine some "programming" with making automated tables/reports.

In the 'Reports.zip' folder on the webpage, there are 36 tables, one table per month, of new individuals joining a study. We are going to practice flexibly reading in many similarly-formatted tables at once.

Report generation

Suppose you have many files of the same general format in one or more folders across your computer (or a server somewhere). We can use apply statements and for loops to automate the process of handling many datasets identically.

```
> files = list.files("Reports", full.names = T)
> length(files)
```

```
[1] 36
```

```
> head(files)
```

```
[1] "Reports/April_2009_Report.txt"  "Reports/April_2010_Report.txt"
[3] "Reports/April_2011_Report.txt"  "Reports/August_2009_Report.txt"
[5] "Reports/August_2010_Report.txt" "Reports/August_2011_Report.txt"
```

Report generation

Now it's going to be useful to name the character vector `files` :

```
> name = sapply(strsplit(files, "/"), function(x) x[2])
> name = sapply(strsplit(name, "\\."), function(x) x[1])
> head(name)
```

```
[1] "April_2009_Report"  "April_2010_Report"  "April_2011_Report"
[4] "August_2009_Report" "August_2010_Report" "August_2011_Report"
```

```
> names(files) = name
> head(files)
```

```
               April_2009_Report                April_2010_Report
 "Reports/April_2009_Report.txt"  "Reports/April_2010_Report.txt"
               April_2011_Report               August_2009_Report
 "Reports/April_2011_Report.txt" "Reports/August_2009_Report.txt"
              August_2010_Report               August_2011_Report
"Reports/August_2010_Report.txt" "Reports/August_2011_Report.txt"
```

Report generation

For this example, it's probably easier to use lapply, which performs a function on each element of a list or vector, and returns a list.

```
> fileList = lapply(files, read.delim, header = T, as.is = T)
> head(names(fileList))
```

```
[1] "April_2009_Report"  "April_2010_Report"  "April_2011_Report"
[4] "August_2009_Report" "August_2010_Report" "August_2011_Report"
```

```
> head(fileList[[1]])
```

```
    id    sex   treat   age bgDrugs height weight block recruitDate   bmi
1 1072 Female Control 51.00  asprin  63.84  131.3     d          21 22.64
2 1073 Female Control 54.81 tylenol  66.10  117.2     b           1 18.85
3 1074 Female    Case 43.54  asprin  64.39  145.0     a          28 24.59
4 1075   Male    Case 52.52    none  70.36  170.0     b           8 24.13
5 1076   Male    Case 43.12   advil  68.38  180.1     a          18 27.08
6 1077   Male    Case 37.54  asprin  70.16  172.5     b          24 24.63
```

```
> fileList = lapply(files, read.delim, header = T, as.is = T)
> head(names(fileList))
```

```
[1] "April_2009_Report"  "April_2010_Report"  "April_2011_Report"
[4] "August_2009_Report" "August_2010_Report" "August_2011_Report"
```

```
> lapply(fileList, head, 2)
```

```
$April_2009_Report
    id    sex   treat   age bgDrugs height weight block recruitDate   bmi
1 1072 Female Control 51.00  asprin  63.84  131.3     d          21 22.64
2 1073 Female Control 54.81 tylenol  66.10  117.2     b           1 18.85

$April_2010_Report
    id    sex treat   age bgDrugs height weight block recruitDate   bmi
1 4337 Female  Case 46.91    none  64.95  140.6     f          25 23.43
2 4338 Female  Case 47.95    none  66.47  143.3     f          14 22.81

$April_2011_Report
    id  sex   treat   age bgDrugs height weight block recruitDate   bmi
1 7780 Male    Case 53.93  asprin  70.12  175.0     f          29 25.02
2 7781 Male Control 62.77 tylenol  71.02  153.1     b          29 21.34

$August_2009_Report
    id  sex   treat   age bgDrugs height weight block recruitDate   bmi
1 2051 Male Control 56.76 tylenol  70.47  168.0     f           2 23.78
2 2052 Male    Case 50.14  asprin  69.56  172.3     c           1 25.04

$August_2010_Report
    id    sex   treat   age bgDrugs height weight block recruitDate   bmi
1 5481   Male Control 40.97  asprin  71.15  168.0     b           7 23.34
2 5482 Female Control 41.10    none  65.78  137.1     c          23 22.27
```

Report generation

Now we have 36 tables in a list. We can order that list chronologically, instead of alphabetically.

```
> month = sapply(strsplit(name, "_"), function(x) x[1])
> month = factor(month, levels = c("January", "February", "March", "April", "May",
+     "June", "July", "August", "September", "October", "November", "December"))
> year = as.integer(sapply(strsplit(name, "_"), function(x) x[2]))
> fileList = fileList[order(year, month)]
> names(fileList)
```

```
 [1] "January_2009_Report"   "February_2009_Report"
 [3] "March_2009_Report"     "April_2009_Report"
 [5] "May_2009_Report"       "June_2009_Report"
 [7] "July_2009_Report"      "August_2009_Report"
 [9] "September_2009_Report" "October_2009_Report"
[11] "November_2009_Report"  "December_2009_Report"
[13] "January_2010_Report"   "February_2010_Report"
[15] "March_2010_Report"     "April_2010_Report"
[17] "May_2010_Report"       "June_2010_Report"
[19] "July_2010_Report"      "August_2010_Report"
[21] "September_2010_Report" "October_2010_Report"
[23] "November_2010_Report"  "December_2010_Report"
[25] "January_2011_Report"   "February_2011_Report"
[27] "March_2011_Report"     "April_2011_Report"
[29] "May_2011_Report"       "June_2011_Report"
[31] "July_2011_Report"      "August_2011_Report"
[33] "September_2011_Report" "October_2011_Report"
[35] "November_2011_Report"  "December_2011_Report"
```

Report generation

How many entries are in each list? How many overall entries are there?

For this, sapply is very useful, because it is applied to a list, but tries to return a matrix.

```
> sapply(fileList, nrow)[1:10]  # number of entries
```

```
  January_2009_Report  February_2009_Report     March_2009_Report
                  328                   359                   384
    April_2009_Report       May_2009_Report      June_2009_Report
                  287                   226                   264
     July_2009_Report    August_2009_Report September_2009_Report
                  202                   353                   225
  October_2009_Report
                  341
```

```
> sum(sapply(fileList, nrow))  # all reports
```

```
[1] 10438
```

Report generation

We can also tabulate variables across reports.

```
> sapply(fileList, function(x) table(x$sex))
```

```
       January_2009_Report February_2009_Report March_2009_Report
Female                 152                  189               197
Male                   176                  170               187
       April_2009_Report May_2009_Report June_2009_Report July_2009_Report
Female               152             110              132              119
Male                 135             116              132               83
       August_2009_Report September_2009_Report October_2009_Report
Female                167                   117                 151
Male                  186                   108                 190
       November_2009_Report December_2009_Report January_2010_Report
Female                  124                  158                 152
Male                    108                  117                 161
       February_2010_Report March_2010_Report April_2010_Report
Female                  150               101               168
Male                    177               119               156
       May_2010_Report June_2010_Report July_2010_Report
Female             118              185              134
Male               106              165              112
       August_2010_Report September_2010_Report October_2010_Report
Female                156                   149                 137
Male                  213                   131                 152
       November_2010_Report December_2010_Report January_2011_Report
Female                  140                  141                 115
Male                    145                  136                 105
       February_2011_Report March_2011_Report April_2011_Report
Female                  179               123               175
Male                    179                98               184
```

```
> sapply(fileList, function(x) table(x$treat))
```

```
        January_2009_Report February_2009_Report March_2009_Report
Case                    176                  184               178
Control                 152                  175               206
        April_2009_Report May_2009_Report June_2009_Report
Case                  154             104              133
Control               133             122              131
        July_2009_Report August_2009_Report September_2009_Report
Case                  91                176                   113
Control              111                177                   112
        October_2009_Report November_2009_Report December_2009_Report
Case                    166                  115                  141
Control                 175                  117                  134
        January_2010_Report February_2010_Report March_2010_Report
Case                    142                  161               122
Control                 171                  166                98
        April_2010_Report May_2010_Report June_2010_Report
Case                  161             108              188
Control               163             116              162
        July_2010_Report August_2010_Report September_2010_Report
Case                 131                179                   147
Control              115                190                   133
        October_2010_Report November_2010_Report December_2010_Report
Case                    160                  138                  128
Control                 129                  147                  149
        January_2011_Report February_2011_Report March_2011_Report
Case                    121                  161               112
Control                  99                  197               109
        April_2011_Report May_2011_Report June_2011_Report
Case                  173              98              186
Control               186             107              175
        July_2011_Report August_2011_Report September_2011_Report
Case                 126                150                   141
```

```
> sapply(fileList, function(x) table(x$bgDrugs))
```

```
        January_2009_Report February_2009_Report March_2009_Report
advil                    62                   84                83
asprin                  107                   95                88
none                     82                   85               105
tylenol                  77                   95               108
        April_2009_Report May_2009_Report June_2009_Report
advil                  74              45               50
asprin                 60              62               77
none                   81              55               64
tylenol                72              64               73
        July_2009_Report August_2009_Report September_2009_Report
advil                 57                 87                    52
asprin                50                 82                    65
none                  45                 86                    61
tylenol               50                 98                    47
        October_2009_Report November_2009_Report December_2009_Report
advil                   107                   51                   53
asprin                   78                   70                   66
none                     79                   49                   78
tylenol                  77                   62                   78
        January_2010_Report February_2010_Report March_2010_Report
advil                    88                   81                66
asprin                   82                   76                51
none                     67                   92                51
tylenol                  76                   78                52
        April_2010_Report May_2010_Report June_2010_Report
advil                  81              52               87
asprin                 74              63               96
none                   77              47               93
tylenol                92              62               74
        July_2010_Report August_2010_Report September_2010_Report
advil                 62                 89                    76
```

```
> sapply(fileList, function(x) table(x$block))
```

```
  January_2009_Report February_2009_Report March_2009_Report
a                  52                   45                75
b                  64                   82                59
c                  64                   66                60
d                  43                   64                65
e                  56                   46                71
f                  49                   56                54
  April_2009_Report May_2009_Report June_2009_Report July_2009_Report
a                40              33               59               38
b                45              39               48               25
c                44              35               41               35
d                52              36               32               27
e                56              46               40               33
f                50              37               44               44
  August_2009_Report September_2009_Report October_2009_Report
a                 71                    40                  67
b                 49                    36                  51
c                 57                    39                  71
d                 55                    44                  47
e                 56                    35                  54
f                 65                    31                  51
  November_2009_Report December_2009_Report January_2010_Report
a                   39                   41                  37
b                   42                   52                  55
c                   46                   46                  60
d                   37                   39                  49
e                   44                   53                  67
f                   24                   44                  45
  February_2010_Report March_2010_Report April_2010_Report May_2010_Report
a                   56                29                53              36
b                   56                41                58              33
c                   57                38                50              40
```

```
> sapply(fileList, function(x) quantile(x$age))
```

```
     January_2009_Report February_2009_Report March_2009_Report
0%                 24.51                24.48             23.29
25%                44.61                44.88             44.81
50%                50.16                50.60             50.51
75%                55.17                56.30             56.85
100%               67.49                75.50             82.73
     April_2009_Report May_2009_Report June_2009_Report July_2009_Report
0%               27.41           30.84            28.93            27.37
25%              43.99           44.27            44.16            44.65
50%              49.66           50.13            50.03            49.94
75%              55.03           55.88            55.41            54.80
100%             71.70           72.81            70.36            73.26
     August_2009_Report September_2009_Report October_2009_Report
0%                23.16                 32.96               21.76
25%               44.60                 44.89               44.80
50%               49.48                 49.66               49.85
75%               54.59                 55.50               55.41
100%              73.93                 67.81               73.14
     November_2009_Report December_2009_Report January_2010_Report
0%                  26.84                28.18               25.64
25%                 43.04                44.09               44.69
50%                 49.49                49.89               50.46
75%                 54.47                54.75               54.57
100%                72.64                68.19               72.46
     February_2010_Report March_2010_Report April_2010_Report
0%                  26.39             19.84             18.34
25%                 44.16             44.73             43.54
50%                 49.83             49.46             48.90
75%                 55.57             55.01             54.99
100%                69.39             71.69             75.65
     May_2010_Report June_2010_Report July_2010_Report August_2010_Report
0%             25.25            26.65            27.56              22.14
```

```
> sapply(fileList, function(x) quantile(x$height))
```

```
     January_2009_Report February_2009_Report March_2009_Report
0%                 62.76                62.47             62.09
25%                65.05                65.09             65.07
50%                68.41                66.55             67.13
75%                70.13                70.07             70.12
100%               73.53                72.54             72.73
     April_2009_Report May_2009_Report June_2009_Report July_2009_Report
0%               61.91           63.02            62.90            61.77
25%              64.88           64.92            65.01            64.73
50%              66.67           68.01            67.73            66.06
75%              69.97           70.09            70.07            69.85
100%             72.86           73.01            74.01            72.91
     August_2009_Report September_2009_Report October_2009_Report
0%                62.32                 62.75               62.00
25%               65.20                 64.94               65.03
50%               68.29                 66.60               68.77
75%               70.01                 69.73               70.05
100%              72.56                 72.30               72.52
     November_2009_Report December_2009_Report January_2010_Report
0%                  62.15                62.77               62.27
25%                 64.82                64.78               64.91
50%                 66.46                66.04               68.21
75%                 69.92                69.89               70.15
100%                72.04                72.31               72.88
     February_2010_Report March_2010_Report April_2010_Report
0%                  61.61             62.53             62.59
25%                 65.09             64.87             64.97
50%                 68.59             68.56             66.97
75%                 70.08             70.25             69.80
100%                72.21             73.16             72.25
     May_2010_Report June_2010_Report July_2010_Report August_2010_Report
0%             62.91            61.76            61.19              62.13
```

```
> sapply(fileList, function(x) quantile(x$bmi))
```

```
     January_2009_Report February_2009_Report March_2009_Report
0%                 18.34                18.51             18.12
25%                22.72                22.63             22.53
50%                23.96                23.77             23.79
75%                25.04                25.12             25.08
100%               28.11                29.09             29.43
     April_2009_Report May_2009_Report June_2009_Report July_2009_Report
0%               18.71           17.94             19.05             17.74
25%              22.41           22.75             22.78             22.45
50%              23.72           24.03             23.85             23.67
75%              24.99           24.99             24.97             25.10
100%             30.42           28.86             28.52             28.58
     August_2009_Report September_2009_Report October_2009_Report
0%                17.42                 18.09               17.98
25%               22.71                 22.69               22.91
50%               23.85                 23.85               23.99
75%               25.16                 24.99               25.24
100%              29.33                 28.83               28.88
     November_2009_Report December_2009_Report January_2010_Report
0%                  18.33                19.66               18.58
25%                 22.59                22.65               22.73
50%                 24.01                23.87               23.83
75%                 25.29                24.89               25.01
100%                28.74                29.25               30.32
     February_2010_Report March_2010_Report April_2010_Report
0%                  18.85             19.04             18.77
25%                 22.64             22.52             22.56
50%                 23.82             23.68             23.92
75%                 25.06             25.09             25.08
100%                29.31             28.86             29.37
     May_2010_Report June_2010_Report July_2010_Report August_2010_Report
0%             18.07            18.84            18.52              17.99
```

"Table 1"

We can now use R to make a "table 1" containing each report. Let's use the first report as an example.

```
> y = fileList[[1]]
> y[1:5, ]
```

```
  id    sex   treat   age bgDrugs height weight block recruitDate   bmi
1  1   Male Control 52.68    none  70.24  173.4     f          25 24.70
2  2 Female Control 47.10    none  63.84  139.9     f          24 24.13
3  3   Male Control 62.84  asprin  69.47  174.5     c           8 25.42
4  4 Female Control 49.51 tylenol  65.39  132.3     b          24 21.75
5  5   Male Control 54.42   advil  70.87  161.8     d           7 22.64
```

```
> cIndexes = split(1:nrow(y), y$treat)  # splits 1st vector by levels of the 2nd
> lapply(cIndexes, head)  # indices for each outcome
```

```
$Case
[1]  6  9 13 14 15 19

$Control
[1] 1 2 3 4 5 7
```

We can use sapply() again here.

```
> mCont = sapply(cIndexes, function(x) colMeans(y[x, c("age", "weight", "height",
+     "bmi")]))
> mCont  # mean of continuous variables by outcome
```

```
          Case Control
age      49.45   50.34
weight  153.94  158.45
height   67.32   68.17
bmi      23.83   23.91
```

```
> sdCont = sapply(cIndexes, function(x) apply(y[x, c("age", "weight", "height",
+     "bmi")], 2, sd))
> sdCont  # sd of continuous variables by outcome
```

```
          Case Control
age      7.912   8.067
weight  17.854  17.833
height   2.793   2.587
bmi      1.820   1.711
```

Note that we now have the mean and sd for the continuous traits. Now we need to do some formatting, basically putting the SDs in parentheses.

```
> mat1 = matrix(paste(signif(mCont, 4), " (SD=", signif(sdCont, 2), ")", sep = ""),
+     nc = 2)
> dimnames(mat1) = dimnames(mCont)  # copies row and column names
> mat1
```

```
       Case              Control
age    "49.45 (SD=7.9)" "50.34 (SD=8.1)"
weight "153.9 (SD=18)"  "158.4 (SD=18)"
height "67.32 (SD=2.8)" "68.17 (SD=2.6)"
bmi    "23.83 (SD=1.8)" "23.91 (SD=1.7)"
```

Now we can tabulate the binary sex variable.

```
> sex = sapply(cIndexes, function(x) table(y$sex[x]))
> sex
```

```
       Case Control
Female   93      59
Male     83      93
```

```
> sexF = signif(prop.table(sex, 2), 3)
> sexF
```

```
        Case Control
Female 0.528   0.388
Male   0.472   0.612
```

And we can add the row to our existing 'table 1'

```
> mat1 = rbind(mat1, sexF[1, ])
> rownames(mat1)[nrow(mat1)] = "Sex (Female)"
> mat1
```

```
             Case              Control
age          "49.45 (SD=7.9)" "50.34 (SD=8.1)"
weight       "153.9 (SD=18)"  "158.4 (SD=18)"
height       "67.32 (SD=2.8)" "68.17 (SD=2.6)"
bmi          "23.83 (SD=1.8)" "23.91 (SD=1.7)"
Sex (Female) "0.528"          "0.388"
```

Now we add the p-values. For continuous variables we will use a t-test and for sex we will use a chi-sqaured test.

```
> pv = apply(y[, c("age", "weight", "height", "bmi")], 2, function(x) t.test(x ~
+     y$treat)$p.value)
> pv
```

```
    age  weight  height     bmi 
0.31571 0.02324 0.00436 0.69091 
```

```
> pv = paste("p=", signif(pv, 3), sep = "")
> pv
```

```
[1] "p=0.316"   "p=0.0232"  "p=0.00436" "p=0.691"
```

```
> sexp = chisq.test(table(y$sex, y$treat))$p.value
> sexp = paste("p=", signif(sexp, 3), sep = "")
> sexp
```

```
[1] "p=0.0151"
```

And now we bind the p-values as a column to the current 'table 1'

```
> pv = c(pv, sexp)
> mat1 = cbind(mat1, pv)
> colnames(mat1)[ncol(mat1)] = "p-value"
> mat1
```

```
              Case                Control             p-value
age           "49.45 (SD=7.9)"  "50.34 (SD=8.1)"  "p=0.316"
weight        "153.9 (SD=18)"   "158.4 (SD=18)"   "p=0.0232"
height        "67.32 (SD=2.8)"  "68.17 (SD=2.6)"  "p=0.00436"
bmi           "23.83 (SD=1.8)"  "23.91 (SD=1.7)"  "p=0.691"
Sex (Female) "0.528"            "0.388"           "p=0.0151"
```

Lastly, we will add the total N as the last row

```
> mat1 = rbind(mat1, c(sapply(cIndexes, length), nrow(y)))
> rownames(mat1)[nrow(mat1)] = "Number"
> mat1
```

```
              Case               Control            p-value
age           "49.45 (SD=7.9)" "50.34 (SD=8.1)" "p=0.316"
weight        "153.9 (SD=18)"  "158.4 (SD=18)"  "p=0.0232"
height        "67.32 (SD=2.8)" "68.17 (SD=2.6)" "p=0.00436"
bmi           "23.83 (SD=1.8)" "23.91 (SD=1.7)" "p=0.691"
Sex (Female) "0.528"           "0.388"           "p=0.0151"
Number        "176"             "152"             "328"
```

Ta-da!

But that's not the best part. We can now do this to every element of the fileList list, using two different ways. The first way is to build a 'for' loop.

```
tableList=fileList # copy format/structure/names
for(i in seq(along=fileList)) {
  y = fileList[[i]]
  < copy all of the table making coding inside here, that starts with 'y' >
  tableList[[i]] = mat1
}
```

This would essentially make tableList a list of tables, one per report.

```
> # or we can write this as a general function
> makeTable1 = function(y) {
+     cIndexes = split(1:nrow(y), y$treat)
+     mCont = sapply(cIndexes, function(x) colMeans(y[x, c("age", "weight", "height",
+         "bmi")]))
+     sdCont = sapply(cIndexes, function(x) apply(y[x, c("age", "weight", "height",
+         "bmi")], 2, sd))
+     mat1 = matrix(paste(signif(mCont, 4), " (SD=", signif(sdCont, 2), ")", sep = ""),
+         nc = 2)
+     dimnames(mat1) = dimnames(mCont)
+     sex = sapply(cIndexes, function(x) table(y$sex[x]))
+     sexF = signif(prop.table(sex, 2), 3)
+     apply(sexF, 2, function(x) paste(x[1], "M/", x[2], "F", sep = ""))
+     mat1 = rbind(mat1, sexF[1, ])
+     rownames(mat1)[nrow(mat1)] = "Sex (Female)"
+     pv = apply(y[, c("age", "weight", "height", "bmi")], 2, function(x) t.test(x ~
+         y$treat)$p.value)
+     pv = paste("p=", signif(pv, 3), sep = "")
+     sexp = chisq.test(table(y$sex, y$treat))$p.value
+     sexp = paste("p=", signif(sexp, 3), sep = "")
+     pv = c(pv, sexp)
+     mat1 = cbind(mat1, pv)
+     colnames(mat1)[ncol(mat1)] = "p-value"
+     mat1 = rbind(mat1, c(sapply(cIndexes, length), nrow(y)))
+     rownames(mat1)[nrow(mat1)] = "Number"
+     return(mat1)
+ }
```

With our general function, it's really easy to lapply this to our list of reports.

```
> tabList = lapply(fileList, makeTable1)
> lapply(tabList, head, 2)
```

```
$January_2009_Report
       Case                Control             p-value
age    "49.45 (SD=7.9)" "50.34 (SD=8.1)" "p=0.316"
weight "153.9 (SD=18)"  "158.4 (SD=18)"  "p=0.0232"

$February_2009_Report
       Case                Control             p-value
age    "50.68 (SD=8.5)" "50.37 (SD=7.5)" "p=0.71"
weight "154.7 (SD=19)"  "154.7 (SD=18)"  "p=0.997"

$March_2009_Report
       Case               Control             p-value
age    "50.2 (SD=8.6)" "50.53 (SD=8.4)" "p=0.698"
weight "155.8 (SD=18)" "154 (SD=18)"    "p=0.306"

$April_2009_Report
       Case                Control             p-value
age    "49.58 (SD=8.1)" "49.59 (SD=7.6)" "p=0.989"
weight "154.2 (SD=18)"  "152.7 (SD=18)"  "p=0.491"

$May_2009_Report
       Case                Control            p-value
age    "48.93 (SD=8.5)" "51.22 (SD=8)"  "p=0.0398"
weight "157.6 (SD=17)"  "153.3 (SD=20)" "p=0.0818"

$June_2009_Report
       Case                Control            p-value
age    "50.05 (SD=8.2)" "49.53 (SD=8)"  "p=0.603"
weight "155.1 (SD=17)"  "155.8 (SD=18)" "p=0.768"
```

Now we can write out each 'Table 1' to a new file. Create a new folder in your current working directory called 'Tables'.

```
> for (i in seq(along = tabList)) {
+     fn = paste("Tables/", names(tabList)[i], "_table1.txt", sep = "")
+     write.table(tabList[[i]], fn, quote = F, sep = "\t")
+ }
```

So we now have 36 tab-delimited tables written to our Tables/ directory

Ta-da!

'Table 1'

We can also make one big data frame, combining each report. The do.call() function is very useful here, which 'constructs and executes a function call from a name or a function and a list of arguments to be passed to it'.

While the definition is a little confusing, you can see how it works in practice. This will row bind all of the list elements together into 1 data frame.

```
> bigTab = do.call("rbind", fileList)
> dim(bigTab)
```

```
[1] 10438    10
```

```
> class(bigTab)
```

```
[1] "data.frame"
```

Note that 'rbind' will only work here if EVERY element of fileList has the same number of columns and likely the same column names.

```
> bigTab[1:10, ]
```

```
                        id    sex   treat   age bgDrugs height weight block
January_2009_Report.1    1   Male Control 52.68    none  70.24  173.4     f
January_2009_Report.2    2 Female Control 47.10    none  63.84  139.9     f
January_2009_Report.3    3   Male Control 62.84  asprin  69.47  174.5     c
January_2009_Report.4    4 Female Control 49.51 tylenol  65.39  132.3     b
January_2009_Report.5    5   Male Control 54.42   advil  70.87  161.8     d
January_2009_Report.6    6 Female    Case 46.02  asprin  63.94  150.5     c
January_2009_Report.7    7 Female Control 60.98 tylenol  65.68  133.5     b
January_2009_Report.8    8   Male Control 45.93    none  69.39  183.9     a
January_2009_Report.9    9 Female    Case 50.37   advil  64.80  144.5     c
January_2009_Report.10 10   Male Control 50.08 tylenol  70.68  169.2     b
                       recruitDate   bmi
January_2009_Report.1           25 24.70
January_2009_Report.2           24 24.13
January_2009_Report.3            8 25.42
January_2009_Report.4           24 21.75
January_2009_Report.5            7 22.64
January_2009_Report.6            5 25.88
January_2009_Report.7            8 21.75
January_2009_Report.8           13 26.84
January_2009_Report.9           13 24.19
January_2009_Report.10           9 23.81
```

'Table 1'

And now we can use our custom function on the full data frame.

```
> makeTable1(bigTab)
```

```
              Case                Control             p-value
age           "49.85 (SD=8.2)"  "50.07 (SD=8)"    "p=0.169"
weight        "155 (SD=18)"     "154.7 (SD=18)"   "p=0.409"
height        "67.5 (SD=2.7)"   "67.49 (SD=2.7)"  "p=0.87"
bmi           "23.85 (SD=1.8)"  "23.82 (SD=1.8)"  "p=0.3"
Sex (Female)  "0.502"           "0.504"           "p=0.921"
Number        "5234"            "5204"            "10438"
```

Data Formatting

Let's fix up the row names from our big table.

```
> ss = function(x, pattern, slot = 1, ...) sapply(strsplit(x, pattern, ...), function(y) y[slot])
> month = ss(rownames(bigTab), "_", 1)
> year = as.integer(ss(rownames(bigTab), "_", 2))
> rownames(bigTab) = NULL
> head(bigTab)
```

```
  id    sex   treat   age bgDrugs height weight block recruitDate   bmi
1  1   Male Control 52.68    none  70.24  173.4     f          25 24.70
2  2 Female Control 47.10    none  63.84  139.9     f          24 24.13
3  3   Male Control 62.84  asprin  69.47  174.5     c           8 25.42
4  4 Female Control 49.51 tylenol  65.39  132.3     b          24 21.75
5  5   Male Control 54.42   advil  70.87  161.8     d           7 22.64
6  6 Female    Case 46.02  asprin  63.94  150.5     c           5 25.88
```

```
> head(month)
```

```
[1] "January" "January" "January" "January" "January" "January"
```

Data Formatting

We can clean up the date as well, and coerce it to the 'Date' class. See more information about formatting here: http://www.statmethods.net/input/dates.html

```
> date = paste(month, " ", bigTab$recruitDate, ", ", year, sep = "")
> bigTab$Date = as.Date(date, format = "%B %d, %Y")
> bigTab = bigTab[, names(bigTab) != "recruitDate"]
> head(bigTab)
```

```
  id    sex   treat   age bgDrugs height weight block   bmi       Date
1  1   Male Control 52.68    none  70.24  173.4     f 24.70 2009-01-25
2  2 Female Control 47.10    none  63.84  139.9     f 24.13 2009-01-24
3  3   Male Control 62.84  asprin  69.47  174.5     c 25.42 2009-01-08
4  4 Female Control 49.51 tylenol  65.39  132.3     b 21.75 2009-01-24
5  5   Male Control 54.42   advil  70.87  161.8     d 22.64 2009-01-07
6  6 Female    Case 46.02  asprin  63.94  150.5     c 25.88 2009-01-05
```

Data Formatting

And we can order by date.

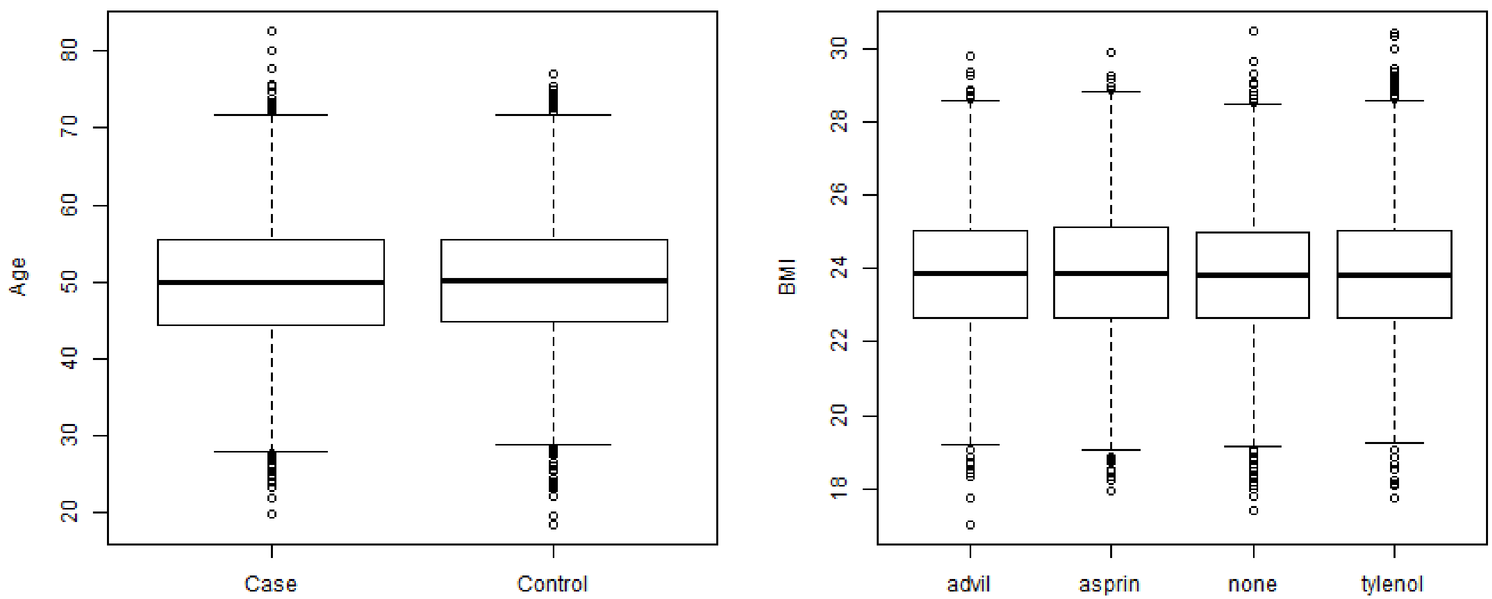
```
> bigTabDate = bigTab[order(bigTab$Date), ]
> head(bigTabDate)
```

```
     id    sex   treat   age bgDrugs height weight block   bmi       Date
29   29   Male    Case 54.56 tylenol  70.94  164.4     b 22.97 2009-01-01
56   56 Female    Case 53.97 tylenol  64.58  147.7     b 24.91 2009-01-01
68   68 Female    Case 51.81   advil  63.58  137.8     c 23.97 2009-01-01
70   70   Male Control 43.70   advil  69.00  169.0     c 24.95 2009-01-01
82   82 Female Control 53.88    none  66.01  136.6     b 22.04 2009-01-01
134 134   Male    Case 57.16    none  71.16  170.2     c 23.63 2009-01-01
```

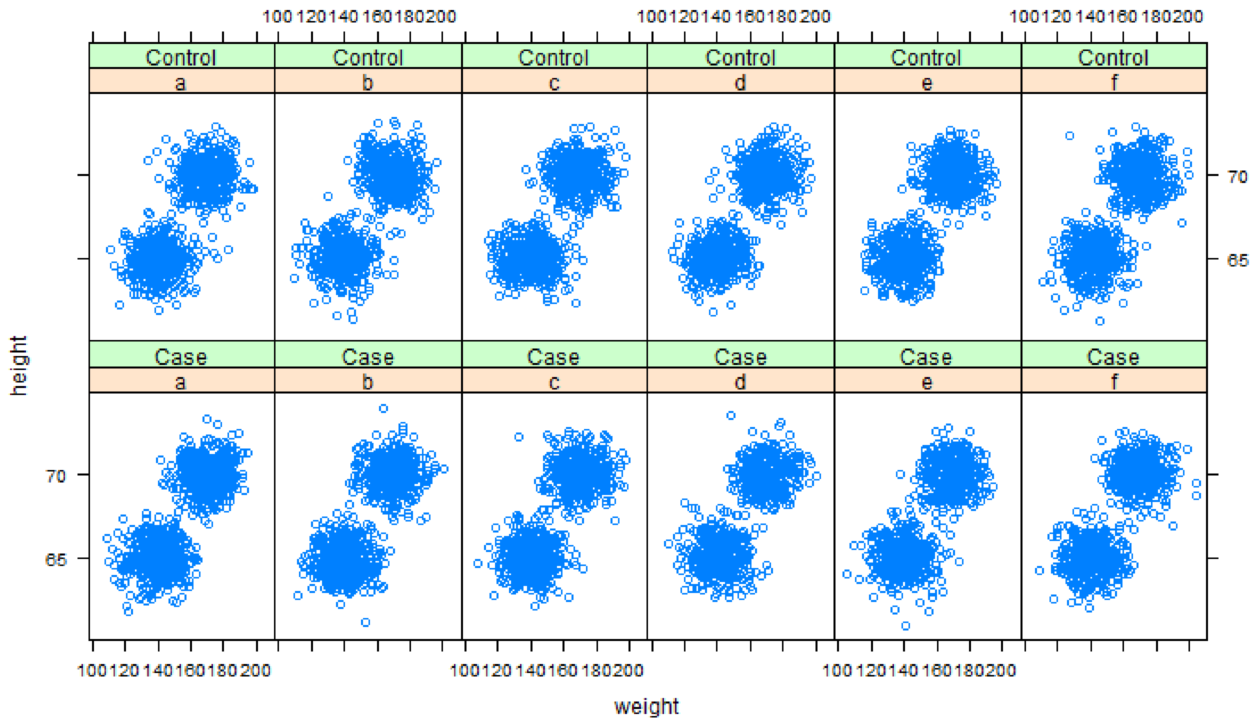
Data Exploration

Now we explore this data frame.

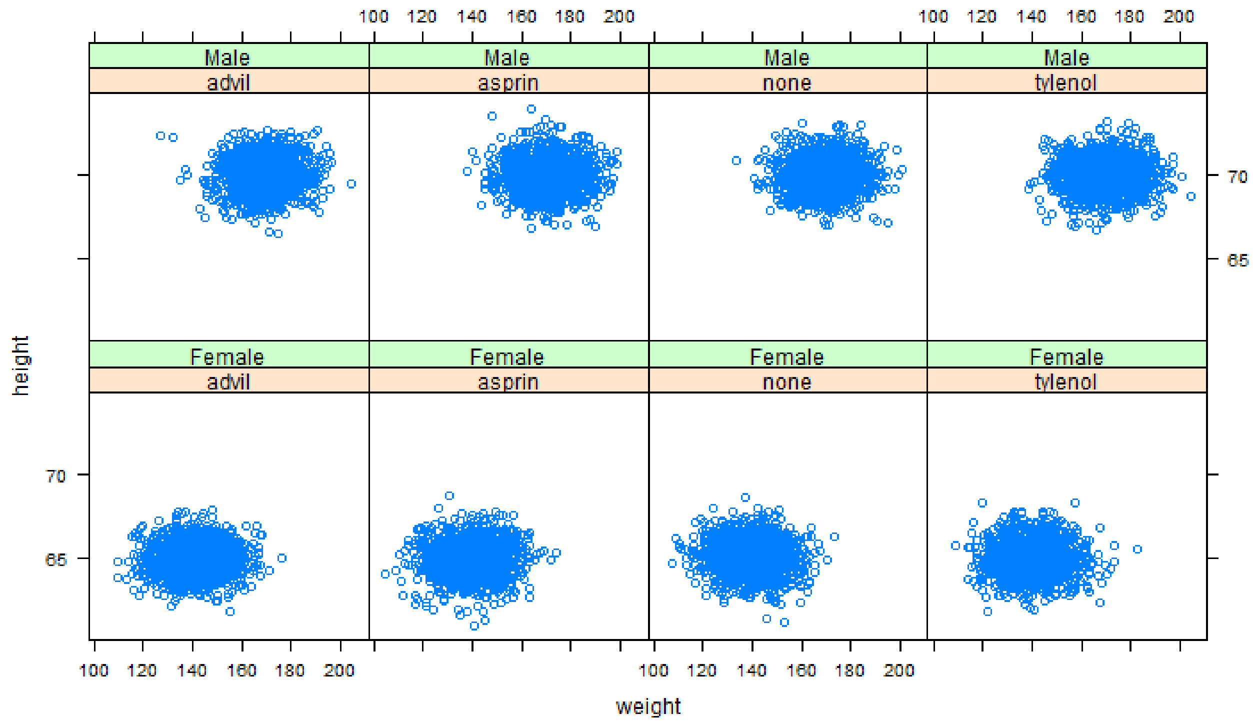
```
> par(mfrow = c(1, 2))
> boxplot(age ~ treat, data = bigTab, ylab = "Age")
> boxplot(bmi ~ bgDrugs, data = bigTab, ylab = "BMI")
```



```
> par(mfrow = c(1, 1))
> library(lattice)
> xyplot(height ~ weight | block * treat, data = bigTab)
```



```
> par(mfrow = c(1, 1))
> library(lattice)
> xyplot(height ~ weight | bgDrugs * sex, data = bigTab)
```



Lab

Play around with this publicly available Cervical Dystonia Dataset, specifically focusing on visualizing the data:

load('http://biostat.mc.vanderbilt.edu/wiki/pub/Main/DataSets/cdystonia.sav')

- Randomized to placebo (N=36), 5000 units of BotB (N=36), 10,000 units of BotB (N=37)
- Response variable: total score on Toronto Western Spasmodic Torticollis Rating Scale (TWSTRS), measuring severity, pain, and disability of cervical dystonia (high scores mean more impairment)
- TWSTRS measured at baseline (week 0) and weeks 2, 4, 8, 12, 16 after treatment began

Then start working on your projects. We will go over some longitudinal plotting at the end of class.