R Fundamentals Part 3: Data exploration and analysis

Rochelle Terman, Evan Muzzall, Dillon Niederhut

October 23, 2016

Table of Contents

Learning objectives

1. Day 2 subsetting review
2. Data exploration and analysis in R
3. Data exploration and plotting
4. Statistical testing
5. ggplot2

1. Day 2 review

Set your working directory to the R-Fundamentals folder

```
getwd()
setwd("/Users/E/Desktop/R-Fundamentals-master")
```

Read in the `animals` data frame you created on Day 1

```
animals <- read.csv("/Users/E/Desktop/R-Fundamentals-
master/animals.csv", header=TRUE, stringsAsFactors=FALSE)
str(animals)

## 'data.frame':    20 obs. of  5 variables:
##  $ Name    : chr  "Dog" "Pig" "Dog" "Cat" ...
##  $ Healthy : logi  FALSE TRUE TRUE FALSE FALSE TRUE ...
##  $ Weight  : num  4.06 4.49 5.29 6.63 3.81 ...
##  $ Height  : int  10 6 7 6 8 6 7 9 5 10 ...
##  $ Progress: num  1.512 0.39 -0.621 -2.215 1.125 ...

head(animals)

##   Name Healthy   Weight Height    Progress
## 1  Dog   FALSE 4.062035     10  1.51178117
## 2  Pig    TRUE 4.488496      6  0.38984324
## 3  Dog    TRUE 5.291413      7 -0.62124058
## 4  Cat   FALSE 6.632831      6 -2.21469989
## 5  Pig   FALSE 3.806728      8  1.12493092
## 6  Cat    TRUE 6.593559      6 -0.04493361

animals
```

2. Data exploration and analysis in R

Research design, data collection, exploration, visualization, and analysis are fundamental aspects of research. Learning how to explore and analyze data will help you "think with the data" so that you become able to formulate better research designs and more efficient data collection protocols for your own research.

Furthermore, when you read peer-reviewed work in your field, having an understanding of basic summaries, plots, and statistical tests will help you better grasp arguments that the authors are making and will allow you to more critically evaluate their rationale.

Our focus is on data analysis in R, which usually proceeds in two steps: 1) exploration/graphing 2) statistical testing/inference

Data exploration often means graphing your data to see if it is possible to visualize any obvious (or not so obvious) trends.

Statistical inference is employed to test relationships that might be illustrated by your plots.

2. Data exploration and analysis in R/ hypothesis testing

You will frequently encounter hypotheses in the peer-reviewed literature and in test results in R. The simplest way to think about hypothesis testing is that you have two hypotheses: the null (Ho) and the alternative (Ha).

The null hypothesis can be thought of as ***NO DIFFERENCE***.

The alternative hypothesis can be thought of as ***SOME SORT OF DIFFERENCE***.

Should your test fail to achieve statistical significance you ***FAIL TO REJECT THE NULL HYPOTHESIS***. Should your test achieve statistical significance, you ***REJECT THE NULL HYPOTHESIS***. These are your only two options! (more on this in section # 4 below).

3. Data exploration and plotting/

3. Data exploration and plotting/ `summary()`

It is often useful to first summarize your data. In R, this can be done in a variety of ways.

`summary()` provides a six-number summary of a data frame

```
?summary

summary(animals)

##      Name              Healthy            Weight          Height
Progress
##  Length:20          Mode :logical   Min.   :3.247   Min.   : 5.0
Min.   :-2.21470
##  Class :character   FALSE:9         1st Qu.:4.382   1st Qu.: 7.0
1st Qu.:-0.23638
##  Mode  :character   TRUE :11        Median :5.404   Median : 7.0
```

```
Median : 0.23220
##                     NA's :0         Mean   :5.221   Mean   : 7.9
Mean   : 0.05759
##                                     3rd Qu.:6.087   3rd Qu.:10.0
3rd Qu.: 0.79191
##                                     Max.   :6.968   Max.   :10.0
Max.   : 1.51178
```

3. Data exploration and plotting/ `describe()` and `describeBy()`

`describe()` and `describeBy()` from the 'psych' R package provide more in-depth summary information. However, we are going to subset the data so that it only includes the numeric variables within the `describe()` call.

Remember, we installed the 'psych' package on Day 1, so all we have to do is call it into our environment.

```
library(psych)

## Warning: package 'psych' was built under R version 3.2.5

?describe

str(animals)

## 'data.frame':    20 obs. of  5 variables:
##  $ Name    : chr  "Dog" "Pig" "Dog" "Cat" ...
##  $ Healthy : logi  FALSE TRUE TRUE FALSE FALSE TRUE ...
##  $ Weight  : num  4.06 4.49 5.29 6.63 3.81 ...
##  $ Height  : int  10 6 7 6 8 6 7 9 5 10 ...
##  $ Progress: num  1.512 0.39 -0.621 -2.215 1.125 ...

describe(animals[,c(3:5)])

##          vars  n mean   sd median trimmed  mad   min   max range
skew kurtosis   se
## Weight      1 20 5.22 1.14   5.40    5.23 1.33  3.25  6.97  3.72 -
0.10    -1.40 0.26
## Height      2 20 7.90 1.68   7.00    7.94 1.48  5.00 10.00  5.00
0.09    -1.54 0.38
## Progress    3 20 0.06 1.01   0.23    0.17 0.84 -2.21  1.51  3.73 -
0.86    -0.19 0.23
```

We can also subset these summary statistics to create simpler tables. Let's assign this output to an object first:

```
animals_describe <- describe(animals[,c(3:5)])
animals_describe

##          vars  n mean   sd median trimmed  mad   min   max range
skew kurtosis   se
## Weight      1 20 5.22 1.14   5.40    5.23 1.33  3.25  6.97  3.72 -
```

```
0.10    -1.40 0.26
## Height      2 20 7.90 1.68   7.00    7.94 1.48  5.00 10.00  5.00
0.09    -1.54 0.38
## Progress    3 20 0.06 1.01   0.23    0.17 0.84 -2.21  1.51  3.73 -
0.86    -0.19 0.23
```

If we just want "median", "sd", skew", "kurtosis", and "se", we could subset our data frame like this:

```
animals_simple <- animals_describe[c(5, 4, 11:13)]
animals_simple

##          median   sd  skew kurtosis   se
## Weight     5.40 1.14 -0.10    -1.40 0.26
## Height     7.00 1.68  0.09    -1.54 0.38
## Progress   0.23 1.01 -0.86    -0.19 0.23
```

Saving summary tables is convenient in this fashion:

```
write.csv(animals_simple, "animals_simple.csv", row.names=TRUE)
#`row.names=TRUE` ensures that row names "Weight", "Height", and
"Progress" labels are printed.
```

Check your working directory for the new .CSV file!

We can also describe our data by a grouping variable using `describeBy()`:

```
?describeBy

#Output summary statistics by one grouping variable:
summary_sub <- describeBy(animals[,c(3:5)], animals$Name)
summary_sub

## $Cat
##          vars n mean   sd median trimmed  mad   min   max range
skew kurtosis   se
## Weight      1 7 5.02 1.41   5.52    5.02 1.66  3.25  6.63  3.39 -
0.06    -2.00 0.53
## Height      2 7 7.57 2.07   7.00    7.57 2.97  5.00 10.00  5.00
0.11    -1.99 0.78
## Progress    3 7 0.26 1.14   0.78    0.26 0.24 -2.21  0.94  3.16 -
1.36     0.19 0.43
##
## $Dog
##          vars n  mean   sd median trimmed  mad   min   max range
skew kurtosis   se
## Weight      1 6  5.67 0.96   5.81    5.67 0.61  4.06  6.97  2.91 -
0.37    -1.15 0.39
## Height      2 6  8.50 1.64   8.50    8.50 2.22  7.00 10.00  3.00
0.00    -2.31 0.67
## Progress    3 6 -0.04 1.00  -0.04   -0.04 0.77 -1.47  1.51  2.98
0.12    -1.33 0.41
```

```
##
## $Pig
##          vars n  mean   sd median trimmed  mad   min   max range
skew kurtosis   se
## Weight      1 7  5.03 1.04   4.54    5.03 0.67  3.81  6.78  2.97
0.55    -1.39 0.39
## Height      2 7  7.71 1.38   7.00    7.71 1.48  6.00 10.00  4.00
0.43    -1.45 0.52
## Progress    3 7 -0.06 1.00  -0.02   -0.06 0.68 -1.99  1.12  3.11 -
0.75    -0.67 0.38
##
## attr(,"call")
## by.data.frame(data = x, INDICES = group, FUN = describe, type =
type)

#If we just want to view Cats, Dogs, or Pigs, we can type:
summary_sub$Cat

##          vars n mean   sd median trimmed  mad   min   max range
skew kurtosis   se
## Weight      1 7 5.02 1.41   5.52    5.02 1.66  3.25  6.63  3.39 -
0.06    -2.00 0.53
## Height      2 7 7.57 2.07   7.00    7.57 2.97  5.00 10.00  5.00
0.11    -1.99 0.78
## Progress    3 7 0.26 1.14   0.78    0.26 0.24 -2.21  0.94  3.16 -
1.36     0.19 0.43

summary_sub$Dog

##          vars n  mean   sd median trimmed  mad   min   max range
skew kurtosis   se
## Weight      1 6  5.67 0.96   5.81    5.67 0.61  4.06  6.97  2.91 -
0.37    -1.15 0.39
## Height      2 6  8.50 1.64   8.50    8.50 2.22  7.00 10.00  3.00
0.00    -2.31 0.67
## Progress    3 6 -0.04 1.00  -0.04   -0.04 0.77 -1.47  1.51  2.98
0.12    -1.33 0.41

summary_sub$Pig

##          vars n  mean   sd median trimmed  mad   min   max range
skew kurtosis   se
## Weight      1 7  5.03 1.04   4.54    5.03 0.67  3.81  6.78  2.97
0.55    -1.39 0.39
## Height      2 7  7.71 1.38   7.00    7.71 1.48  6.00 10.00  4.00
0.43    -1.45 0.52
## Progress    3 7 -0.06 1.00  -0.02   -0.06 0.68 -1.99  1.12  3.11 -
0.75    -0.67 0.38
```

```
#We can also view just the means for Dogs, we can type:
summary_sub$Dog[[3]] #Here, double bracket subsetting is used to return
just the value from a list.

## [1]  5.67490313  8.50000000 -0.04058346

#If we just want the Height mean for Dogs (8.5), we can type:
summary_sub$Dog[[3]][2]

## [1] 8.5

#Or the medians for Pigs:
summary_sub$Pig[[5]]

## [1]  4.53641487  7.00000000 -0.01619026
```

3. Data exploration and plotting/ `table()`

We can view frequencies for of categorical data like Cat, Dog, and Pig with `table()`

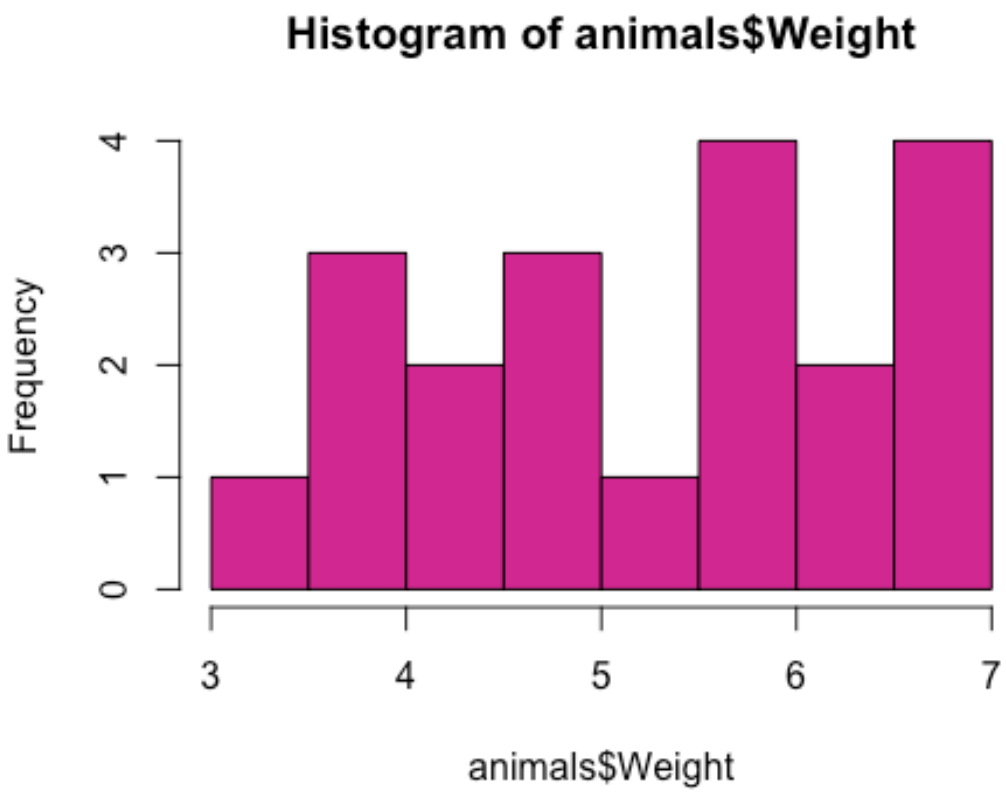
```
?table

table(animals$Name) # get frequencies for Cat, Dog, and Pig

##
## Cat Dog Pig
##   7   6   7

table(animals$Name, animals$Healthy) #get frequencies for Cat, Dog, and
Pig by column "Healthy".

##
##       FALSE TRUE
##   Cat     5    2
##   Dog     3    3
##   Pig     1    6
```

Wow! Only 2/7 cats are healthy.
3/6 dogs are healthy.
6/7 pigs are healthy!

3. Data exploration and plotting/ `hist()`

Histograms are a handy way to quickly illustrate the distribution shape of a particular variable. Let's visualize the "Height" variable from the `animals` data frame:

```
?hist

hist(animals$Weight, col="violetred")
```

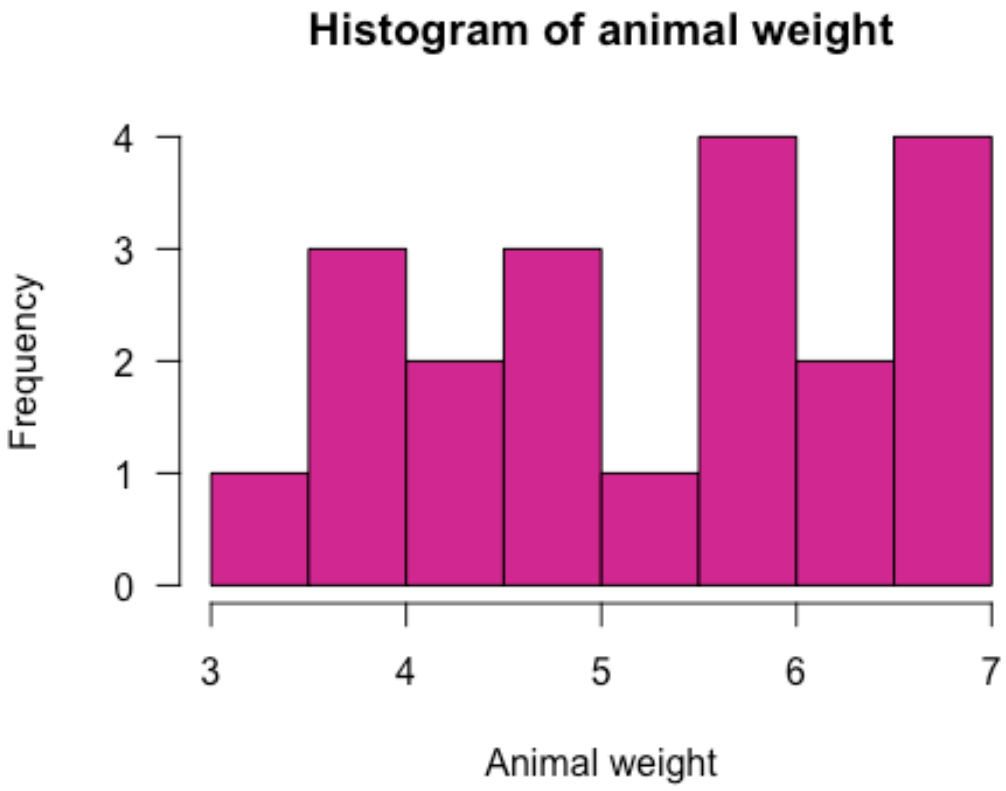


The `col=` argument specifies the color of the bars. Look at the colors you can choose from in base R using `colors()`

```
colors()
```

We can change the title and the x-axis label with `main` and `xlab`, respectively. `las` can be used to change the orientation of the axis text:

```
hist(animals$Weight,
     col="violetred",
     main="Histogram of animal weight",
     xlab="Animal weight",
     las=1)
```



Challenge 1

1. Using the `iris` dataset, create a histogram of one of the numeric variables.
2. Use `table()` to count the numbers of the three different iris flower species.

3. Data exploration and plotting/ `plot()`

What if we want to know more about the relationships between two numeric variables? Here, base `plot()` is useful. This creates a scatterplot.

```
?plot
```

First, let's coerce "Name" from character to factor data type:

```
animals$Name <- factor(animals$Name)
str(animals)
```

```
## 'data.frame':    20 obs. of  5 variables:
##  $ Name    : Factor w/ 3 levels "Cat","Dog","Pig": 2 3 2 1 3 1 3 1 1
1 ...
##  $ Healthy : logi  FALSE TRUE TRUE FALSE FALSE TRUE ...
##  $ Weight  : num  4.06 4.49 5.29 6.63 3.81 ...
```

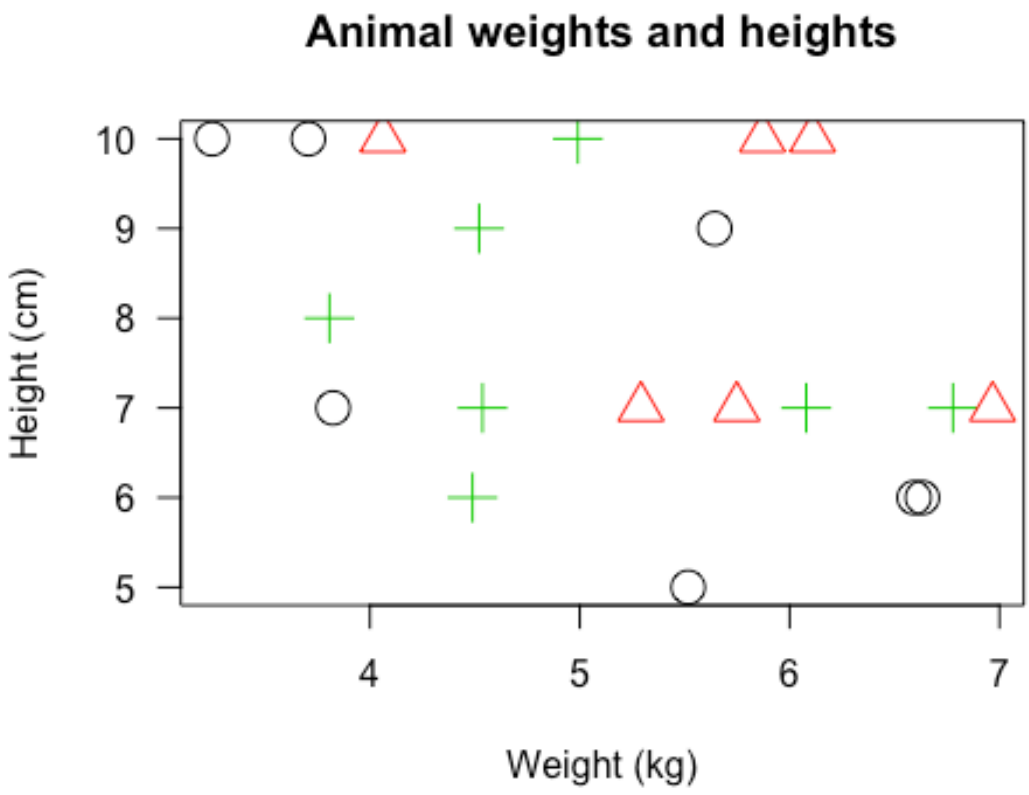
```
##  $ Height  : int  10 6 7 6 8 6 7 9 5 10 ...
##  $ Progress: num  1.512 0.39 -0.621 -2.215 1.125 ...
```

Then, identify two variables you want to plot on the x and y axes:

```
animals$Weight
animals$Height
```

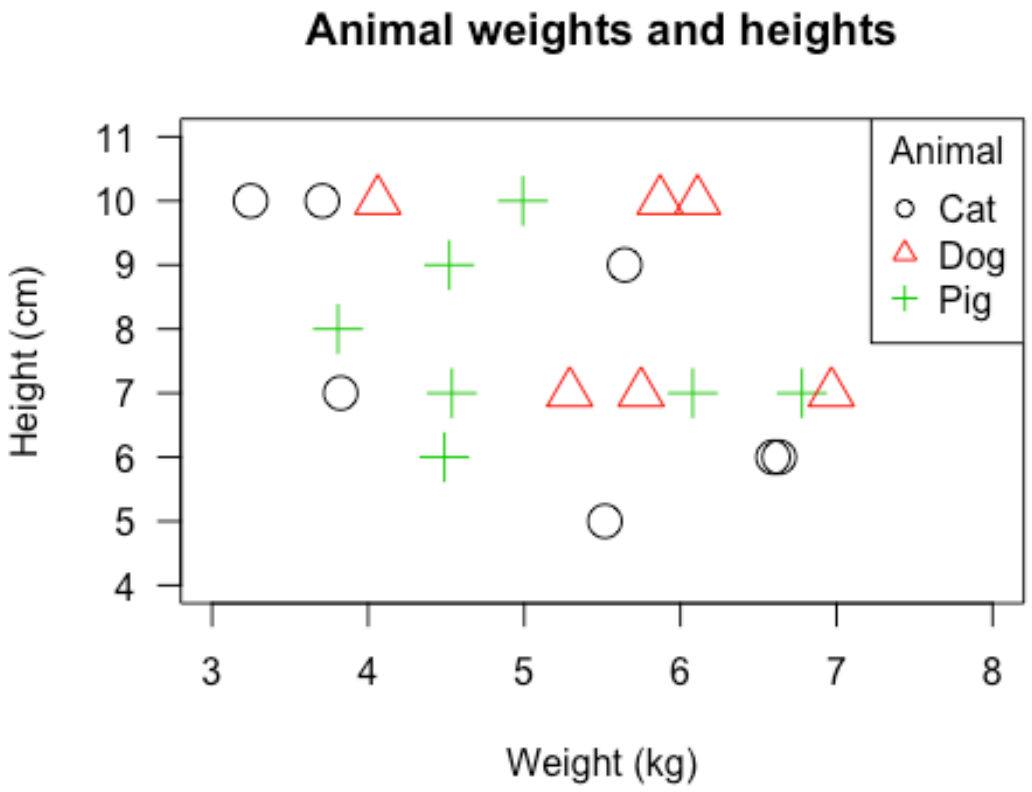
Now, we can plot animal "Weight" versus "Height":

```
plot(animals$Weight, animals$Height,
    xlab="Weight (kg)",   # change x axis label
    ylab="Height (cm)",   # change y axis label
    main="Animal weights and heights",   # add plot title
    las=1, # make all axis text parallel to x-axis
    col=as.integer(animals$Name),   # change point colors to correspond
to animal names
    pch=as.integer(animals$Name),   # change point symbols to
correspond to animal names
    cex=2)   # change point size
```



However, our points are falling ourside the plot boundaries! We can change that with `xlim` and `ylim`:

```r
plot(animals$Weight, animals$Height,
    xlab="Weight (kg)",   # change x axis label
    ylab="Height (cm)",   # change y axis label
    main="Animal weights and heights",   # add plot title
    las=1, # make all axis text parallel to x-axis
    col=as.integer(animals$Name),   # change point colors to match
animal names
    pch=as.integer(animals$Name),   # change point symbols to match
animal names
    cex=2,   # change point size
    xlim=c(3,8),  # adjust range of x axis
    ylim=c(4,11))  # adjust range of y axis

# That looks better! We can also add a legend:

legend("topright", inset=.0, title="Animal", cex=1,
       c("Cat","Dog","Pig"), col=c(1,2,3), pch=c(1,2,3),
       horiz=FALSE)
```



NOTE: One drawback of R Studio is that its in-environment graphics look a little strange sometimes. However, to save it as a beautiful .PNG file, we wrap our plotting and legend instructions by `png()` and `dev.off()` lines of code. We can specify figure characteristics inside `png()` such as the dimensions and resolution of the image, while `dev.off()` writes the file to disk.

```
?png
```

Highlight and run all of this code at once:

```
png("Animal weights and heights.png", height=6, width=6, units="in",
res=300)
plot(animals$Weight, animals$Height,
    xlab="Weight (kg)", ylab="Height (cm)",
    main="Animal weights and heights", las=1,
    col=as.integer(animals$Name), pch=as.integer(animals$Name),
    cex=2, xlim=c(3,8), ylim=c(3,11))
legend("topright", inset=.0, title="Animal", cex=1,
       c("Cat","Dog","Pig"), col=c(1,2,3), pch=c(1,2,3),
       horiz=FALSE)
dev.off()
```

Voila! Check your working directory - we now have a publishable quality figure that can be copy and pasted into your manuscript :)

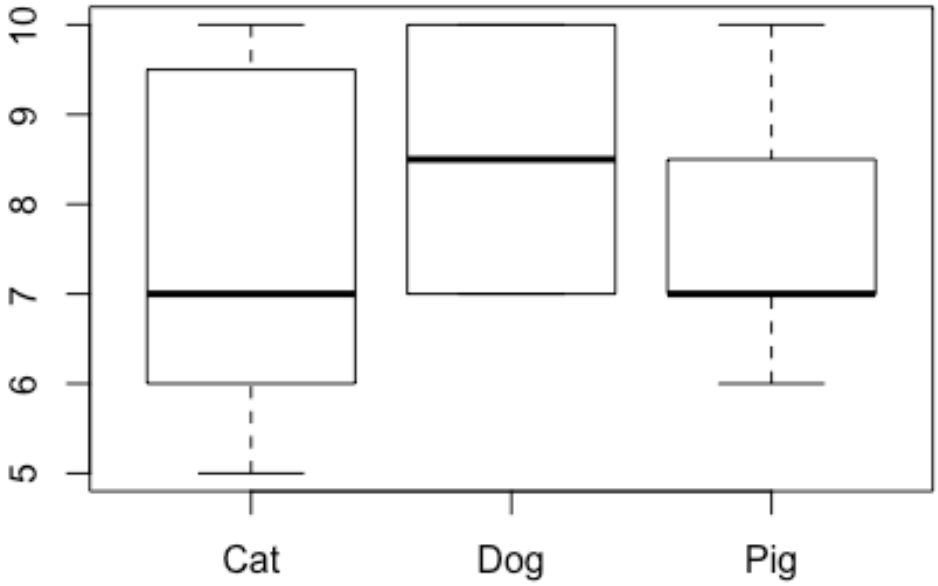
Challenge 2

1. Using the `iris` dataset, plot two numeric variables and export the graph to your working directory as a .PNG file.

3. Data exploration fand plotting/ `boxplot()`

Boxplots also are useful because you can graphically represent a numeric variable as parsed by a factor. The idea is similar to `hist()` and `plot()` except the syntax is slightly different. These are often referred to as "Tukey boxplots":

```
?boxplot

boxplot(animals$Height ~ animals$Name)
```



The tilde ~ means "as parsed by" or "as divided by" some category (in our case Height as parsed by Cat, Dog, and Pig). It means we want to look at animal Height by Name.

Remember your six number summary from Day 2? A Tukey boxplot represents a 5 number summary (minus the mean).

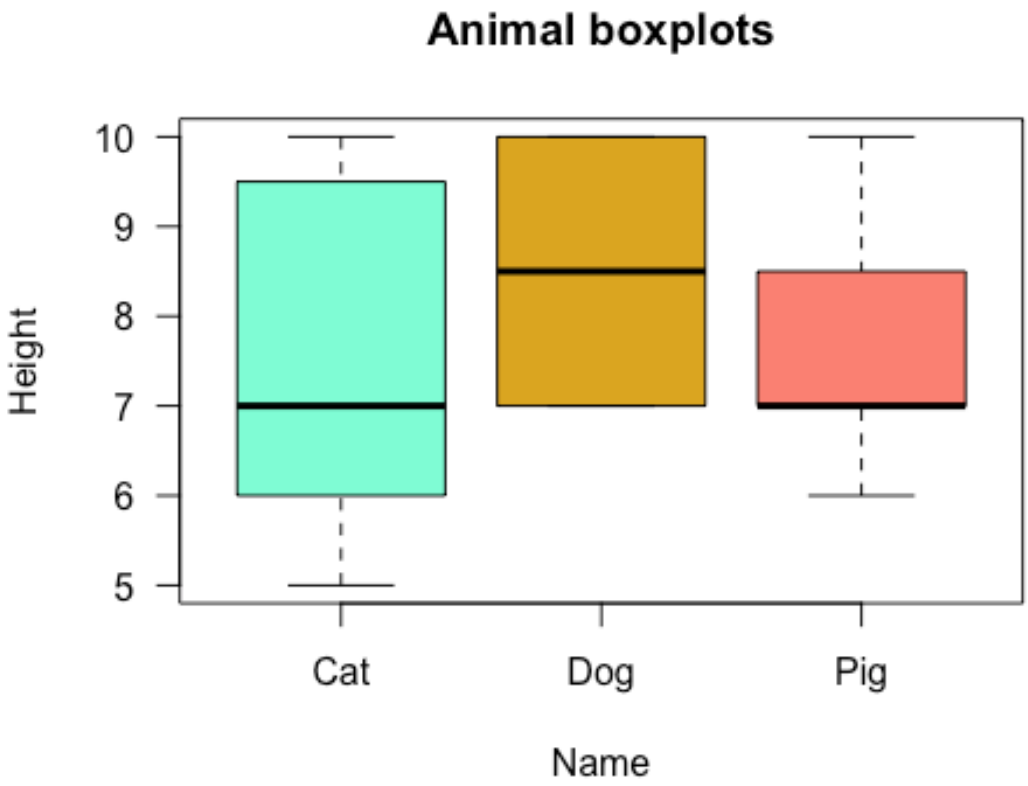
The "whiskers" of these box and whisker plots represent the minimum and maximum values.

The lower and upper borders of the rectangle represent the first and third quartiles.

The thick horizontal black bar represents the median!

We can also change colors by specifying a vector of color names with a length of 3 (one for each box). Other base plotting features are applicable as well:

```
boxplot(animals$Height ~ animals$Name,
        col=c("aquamarine", "goldenrod", "salmon"),
        las=1, main="Animal boxplots", xlab="Name", ylab="Height")
```




```
#That looks better!
```

We can also save our figures as .PDF files:

```
?pdf

pdf("animals boxplot.pdf", 6, 6)
boxplot(animals$Height ~ animals$Name,
        col=c("aquamarine", "goldenrod", "salmon"),
        las=1, main="Animal boxplots", xlab="Name", ylab="Height")
dev.off()
```

Check your working directory again - you now have a .PDF of your boxplots!

Challenge 3

1. Create boxplots for one of the numeric variables in the `iris` dataset and export it to your working directory as a .PDF.

4. Statistical testing

Now that we have sufficiently explored our data, it is time to test some of the relationships identified by it.

Hypothesis testing is central to research. How can you tell if the differences you observe are real or not? We answer the question through hypothesis formulation and significance testing as measured by p-values.

Suppose we want to see if Height differenecs between Cat, Dog, and Pig are significantly different between the three groups.

We can define our ***NULL HYPOTHESIS*** to state that there are no real Height differences between our animals.

We can also define our ***ALTERNATIVE HYPOTHESIS*** to state that there are *some kind of* differences between our animals. We do not specify the nature of the differences, but just that they exist.

These hypotheses are gauged by statistical significance as indicated by a "p-value". A p-value signifies how likely differences are actually real instead of due to random chance. The standard cutoff value is $p<0.05$.

If $p>0.05$, you ***FAIL TO REJECT THE NULL HYPOTHESIS***! This means that the null hypothesis holds true and that there is "no difference" between your groups.

If $p<0.05$, you ***REJECT THE NULL HYPOTHESIS*** of "no difference" and by default you accept the alternative hypothesis and whatever "differences" it specified. This is a "statistically significant" difference.

For example, look at your boxplots. Cat and Pig medians appear similar for Height, but their standard deviations differ greatly (the position of the median and the distribution shape of the boxes). However, they both seem different from Dog...

Might there be formal significant statistical differences between the means of Cat/Pig and Dog?

Well, we can test those assumptions with "mean comparisons" such as t-tests and analysis of variance (ANOVA)!

4. Statistical testing/ `t.test()`

A "t.test" formally compares two (and only two) group means for statistically significant differences at a standard p-value cutoff of $p<0.05$.

The null hypothesis states that there are no actual mean differences between the two groups.

Let's first subset our data into `Cat`, `Dog`, and `Pig` data frames.

```
Cat <- animals[animals$Name=="Cat",]
Cat
```

```
##    Name Healthy   Weight Height    Progress
## 4   Cat   FALSE 6.632831      6 -2.21469989
## 6   Cat    TRUE 6.593559      6 -0.04493361
## 8   Cat    TRUE 5.643191      9  0.94383621
## 9   Cat   FALSE 5.516456      5  0.82122120
## 10  Cat   FALSE 3.247145     10  0.59390132
## 11  Cat   FALSE 3.823898      7  0.91897737
## 12  Cat   FALSE 3.706227     10  0.78213630
```

```
Dog <- animals[animals$Name=="Dog",]
Dog
```

```
##    Name Healthy   Weight Height    Progress
## 1   Dog   FALSE 4.062035     10  1.51178117
## 3   Dog    TRUE 5.291413      7 -0.62124058
## 13  Dog    TRUE 5.748091      7  0.07456498
## 17  Dog   FALSE 5.870474     10 -0.15579551
## 18  Dog    TRUE 6.967624      7 -1.47075238
## 20  Dog   FALSE 6.109781     10  0.41794156
```

```
Pig <- animals[animals$Name=="Pig",]
Pig
```

```
##    Name Healthy   Weight Height    Progress
## 2   Pig    TRUE 4.488496      6  0.38984324
## 5   Pig   FALSE 3.806728      8  1.12493092
## 7   Pig    TRUE 6.778701      7 -0.01619026
## 14  Pig    TRUE 4.536415      7 -1.98935170
## 15  Pig    TRUE 6.079366      7  0.61982575
## 16  Pig    TRUE 4.990797     10 -0.05612874
## 19  Pig    TRUE 4.520141      9 -0.47815006
```

Now, we can compare Height means between two of the three animals.

```
?t.test
```

```
t.test(Cat$Height, Pig$Height)
```

```
##
##  Welch Two Sample t-test
##
## data:  Cat$Height and Pig$Height
## t = -0.15191, df = 10.454, p-value = 0.8821
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -2.225942  1.940227
## sample estimates:
## mean of x mean of y
##  7.571429  7.714286
```

```
# The p-value is 0.88 - we FAIL TO REJECT the null hypothesis (aka,
there is no actual difference between Cat and Pig means for Height). We
accept the conditions of the null hypothesis that state there are no
between-group differences.
```

What about between Cat and Dog?

```
t.test(Cat$Height, Dog$Height)

##
##  Welch Two Sample t-test
##
## data:  Cat$Height and Dog$Height
## t = -0.90095, df = 10.958, p-value = 0.387
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -3.198090  1.340947
## sample estimates:
## mean of x mean of y
##  7.571429  8.500000

# p=0.39 so again we FAIL TO REJECT the null hypothesis
```

Challenge 4

1. Subset the `iris` dataset by species and perform a t-test between two of the species on one of the numeric variables.

4. Statistical testing/ `aov()` and `TukeyHSD()`

What about if we want to compare more than two groups? Doing a bunch of pairwise t-tests (i.e., "multiple comparisons" for cat v. dog, cat v. pig, dog v. pig) is not ideal because it becomes more difficult to adjust for the "family wise error rate", or the influence of the other variables that are not being tested.

When we want to compare *more than two group means at once* (i.e., Cat, Dog, *and* Pig), we can use an analysis of variance (ANOVA). This is called by `aov()` in R.

We can follow `aov()` with a "Tukey test of Honest Significant Difference" to find out between exactly which groups the differences (if any) exist. This is called with `TukeyHSD` in R.

Let's try! Note the slightly different syntax - we now specify only the column headings, and enter the name of our data frame in the "data" argument. Let's call our object `aov1`:

```
aov1 <- aov(Height ~ Name, data = animals)

#Use `summary()` to access the useful information from our `aov1`
```

```
model:
summary(aov1)

##             Df Sum Sq Mean Sq F value Pr(>F)
## Name         2   3.16   1.579    0.53  0.598
## Residuals   17  50.64   2.979
```

We receive a non-significant p-value. However, we can still follow this up with a TukeyHSD test to see between which between-group differences are responsible for the differences!

```
TukeyHSD(aov1)

##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = Height ~ Name, data = animals)
##
## $Name
##                diff       lwr      upr     p adj
## Dog-Cat  0.9285714 -1.534798 3.391940 0.6068162
## Pig-Cat  0.1428571 -2.223871 2.509586 0.9868784
## Pig-Dog -0.7857143 -3.249083 1.677655 0.6971138

#This allows us to see mean Height differences bewteen the three
animals.
```

Challenge 5

1. Using the `iris` dataset, perform an ANOVA and TukeyHSD test for one of the numeric variables between the three flower species.

4. Statistical testing/ `cor.test()`

Correlation is a useful way to see if two numeric variables are related. "Pearson's r" is the default coefficient in `cor.test()`.

Investigate if Cat Height and Cat Weight are correlated:

```
?cor.test

cor.test(Cat$Height, Cat$Weight)

##
##  Pearson's product-moment correlation
##
## data:  Cat$Height and Cat$Weight
## t = -2.1585, df = 5, p-value = 0.08334
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.9504702  0.1227167
```

```
## sample estimates:
##        cor
## -0.6945243
```

Almost! Cat Height and Weight are NEGATIVELY correlated (r = -0.69) and are almost statistically significant (p=0.08).

4. Statistical testing/ `lm()`

So, Cat Height and Weight seem to be related at an adjusted p-value cutoff of p<0.10 for example.

Now what? Linear regression is a convenient way to see if one numeric variable can be used to predict another. Do you think Cat Weight can be used to predict Cat Height? Again, note the altered syntax:

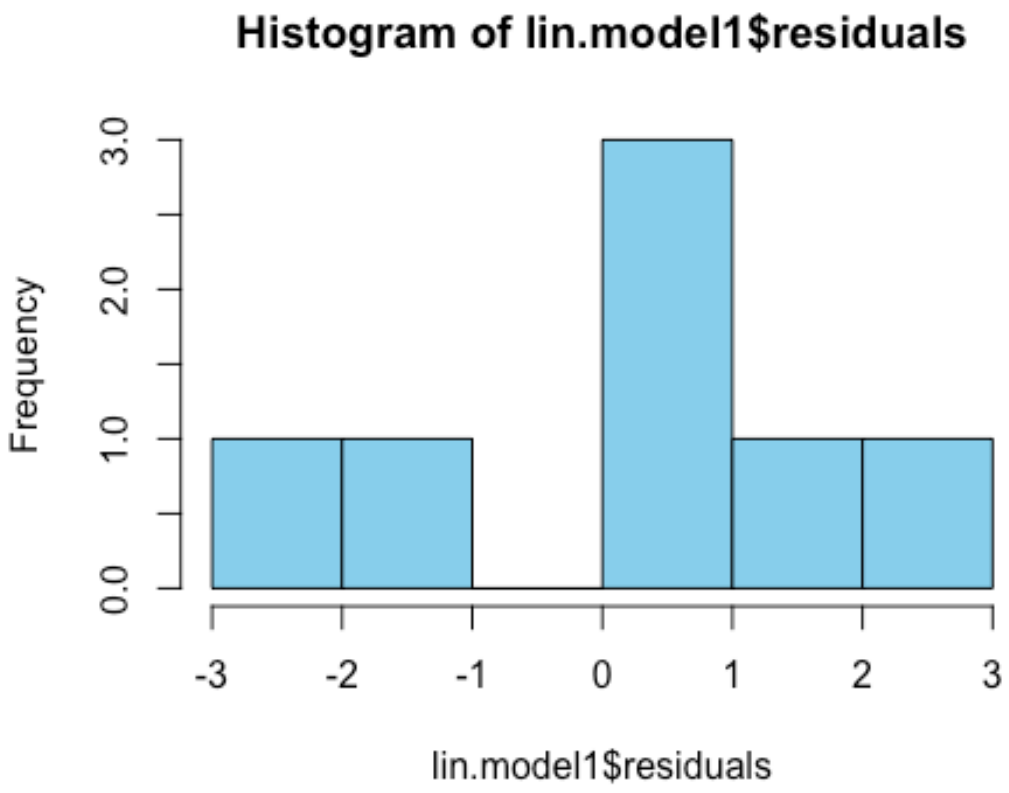
```
?lm

lin.model1 <- lm(Height ~ Weight, data=Cat)
summary(lin.model1)

##
## Call:
## lm(formula = Height ~ Weight, data = Cat)
##
## Residuals:
##        4        6        8        9       10       11       12
##  0.06412  0.02421  2.05846 -2.07032  0.62364 -1.79027  1.09015
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  12.6760     2.4439   5.187  0.00351 **
## Weight       -1.0162     0.4708  -2.159  0.08334 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.632 on 5 degrees of freedom
## Multiple R-squared:  0.4824, Adjusted R-squared:  0.3788
## F-statistic: 4.659 on 1 and 5 DF,  p-value: 0.08334
```

`lm()` output is dense! Check out the yhat blog for fitting and interpreting linear models

You can also pull items out of return objects using `names()`. To extract the residuals we would use the dollar sign operator `$` and type:

```
names(lin.model1)

##  [1] "coefficients"  "residuals"     "effects"       "rank"
"fitted.values" "assign"        "qr"
```

```
##  [8] "df.residual"   "xlevels"       "call"          "terms"
"model"

lin.model1$residuals

hist(lin.model1$residuals, col="skyblue") # residuals should be
approximately normal when plotted
```



Challenge 6

1. Using the `mtcars` dataset, create boxplots for one numeric variable as parsed by a factor variable.

(hint: use `?mtcars` to view the variable names)

(hinthint: can you surmise a relationship about something like engine size and miles per gallon?)

(hinthinthint: you can also use `cor.test()` to see if two variables are related)

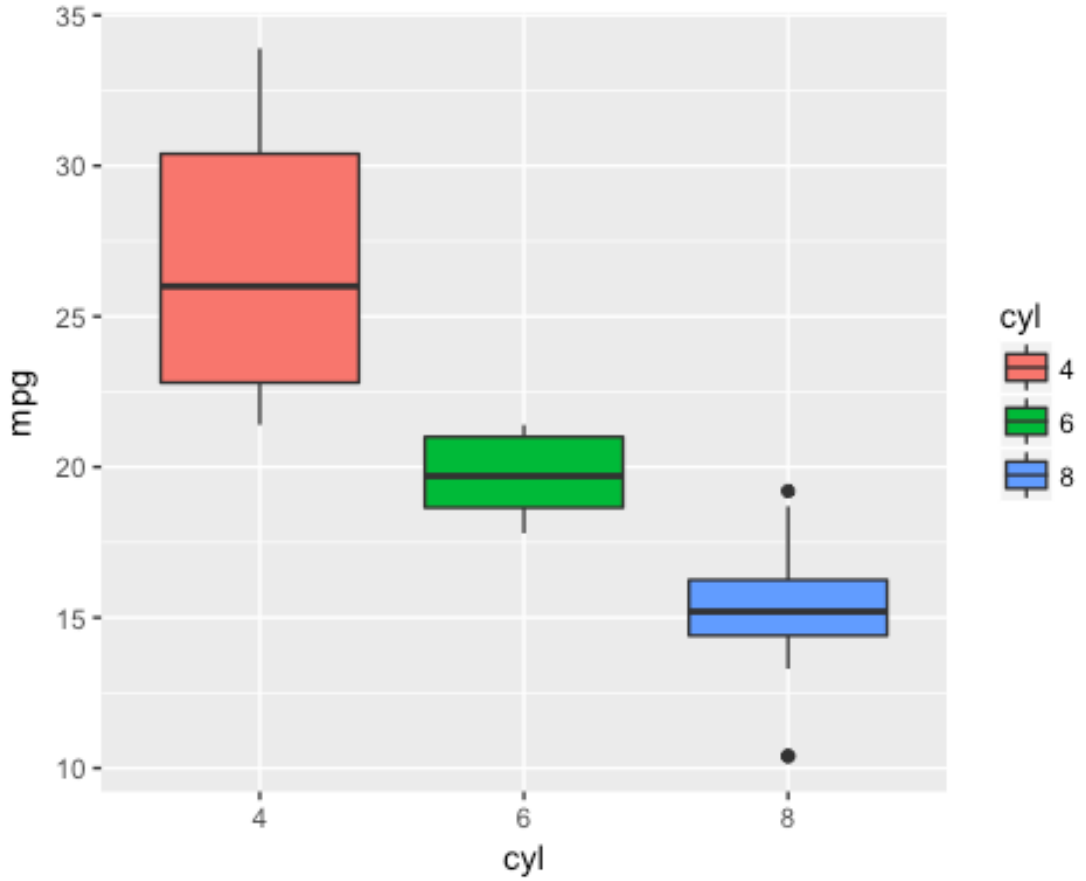
2. Create a scatterplot of two variables using `plot()`.

3. Can one of these variables be used to predict another in a linear regression model?

5. ggplot2

The "ggplot2" R package is another powerful way to graph your data. The syntax is similar to what we used in "dplyr", except instead of a pipe %>%, we use a plus-symbol +:

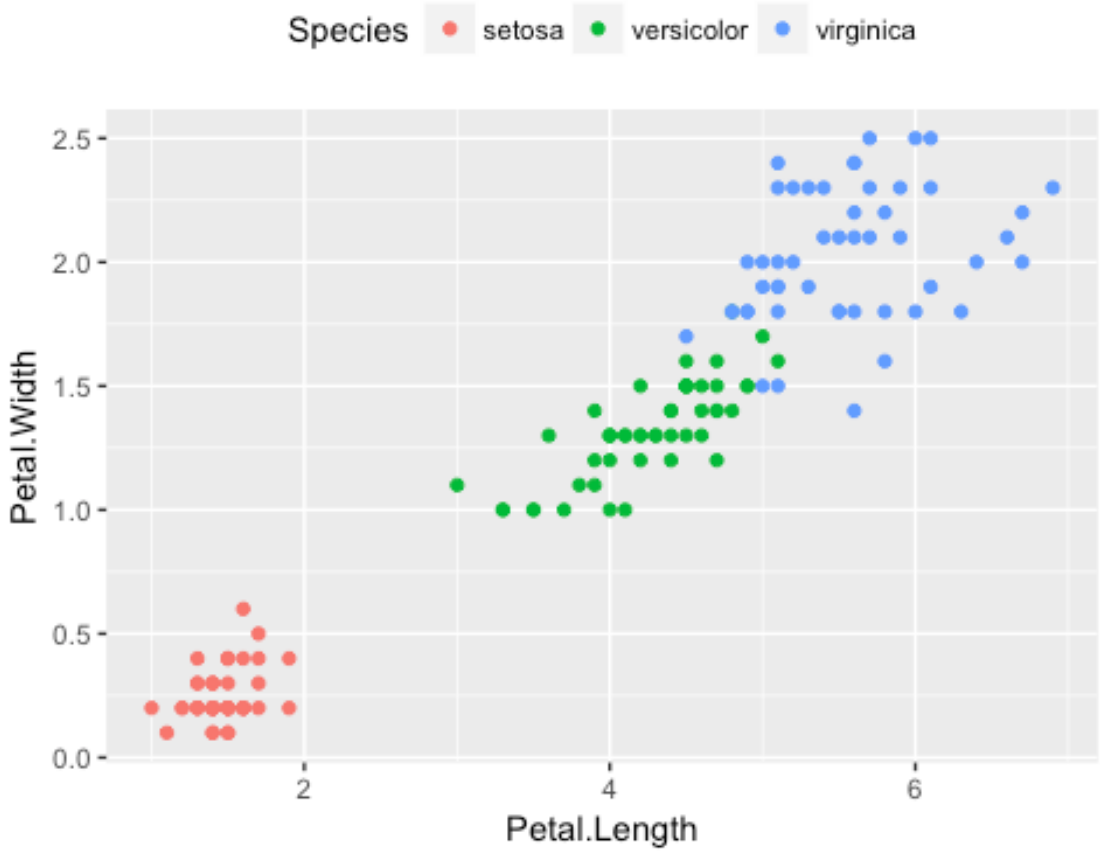
```
install.packages("ggplot2", dependencies=TRUE)
?ggplot

library(ggplot2)

## Warning: package 'ggplot2' was built under R version 3.2.5

##
## Attaching package: 'ggplot2'

## The following objects are masked from 'package:psych':
##
##     %+%, alpha

mtcars$cyl <- factor(mtcars$cyl) # first, coerce "cyl" to factor data
type
ggplot(mtcars, aes(x=cyl, y=mpg, fill=cyl)) +
         geom_boxplot()
```



Note the different syntax again. You need three things to make a ggplot: 1. A dataset (`mtcars`)
2. aesthetics `aes()` - this is where you specify your axes and colors 3. A geom - this is where you tell R if you want a scatterplot, boxplots, etc.

Let's create a scatterplot using the `iris` data:

```
data(iris)
ggplot(iris, aes(x=Petal.Length, y=Petal.Width, color=Species)) +
  geom_point() + theme(legend.position="top")
```



Challenge 7

1. Use "ggplot2" to create a boxplot or scatterplot using the `mtcars` or `iris` dataset.
2. Export this plot to your working directory as a .PNG or .PDF file.