# Introduction to Statistical Analyses in R: OSOS 2024

Matthew Marano

08/24/2024

## Introduction

There are many ways to analyze data, and the tests you will choose to run depend on the type of data you have (i.e. numbers, factors, whether normally distributed, etc.) and the questions you want to answer. This Module will highlight how we can run basic statistical tests in R.

For simplicity, we are going to start by simulating a normalized dataset to use, then we will move on later in the workshop to using mock datasets that R has openly available.

### Setting a seed in R

- The way that R generates "random numbers" is with an algorithm that uses a seed to initialize.
- Therefore, setting a seed will allow you to initialize at this same location and reproduce the analyses and output. This is particularly important when performing simulation studies or when doing a workshop like this where we will all get the same data to work with.

### Simulating a normalized dataset in R

- Random normal distribution in R: rnorm()
- n = how many random numbers we want to generate
- mean = mean of the normalized random dataset
- sd = standard deviation of the normalized random dataset

### Other types of datasets that we can simulate in R but won't cover in this workshop

- Random uniform distribution: runif()
- Random Poisson distribution: rpois()
- As part of the stats package, it's possible to create many other distributions: binomial, F, log normal, beta, exponential, Gamma, etc.
- Even more datasets are possible with additional packages.

```r
#set the seed of the environment to 1
set.seed(1)

#simulate and store a dataset with a normalized distribution
x <- rnorm(100,
           mean = 10,
           sd = 2)
```

### Visualizing your data

It's always important to get a feel for the data that you are working with prior to performing analyses, as it can help you to understand if the results that you are getting make sense.

Let's look at the raw data.

```
#let's simply print out the dataset that we have created
x
```

```
##   [1]  8.747092 10.367287  8.328743 13.190562 10.659016  8.359063 10.974858
##   [8] 11.476649 11.151563  9.389223 13.023562 10.779686  8.757519  5.570600
##  [15] 12.249862  9.910133  9.967619 11.887672 11.642442 11.187803 11.837955
##  [22] 11.564273 10.149130  6.021297 11.239651  9.887743  9.688409  7.058495
##  [29]  9.043700 10.835883 12.717359  9.794425 10.775343  9.892390  7.245881
##  [36]  9.170011  9.211420  9.881373 12.200051 11.526351  9.670953  9.493277
##  [43] 11.393927 11.113326  8.622489  8.585010 10.729164 11.537066  9.775308
##  [50] 11.762215 10.796212  8.775947 10.682239  7.741274 12.866047 13.960800
##  [57]  9.265557  7.911731 11.139439  9.729891 14.803236  9.921520 11.379479
##  [64] 10.056004  8.513454 10.377585  6.390083 12.931110 10.306507 14.345223
##  [71] 10.951019  8.580107 11.221453  8.131805  7.492733 10.582892  9.113416
##  [78] 10.002211 10.148683  8.820958  8.862663  9.729643 12.356174  6.952866
##  [85] 11.187892 10.665901 12.126200  9.391632 10.740038 10.534198  8.914960
##  [92] 12.415736 12.320805 11.400427 13.173667 11.116973  7.446816  8.853469
##  [99]  7.550775  9.053199
```

```
#let's look at the mean, median, standard deviation, and range of our normalized dataset
mean(x) #10.21777
```

```
## [1] 10.21777
```

```
median(x) #10.22782
```

```
## [1] 10.22782
```

```
sd(x) #1.796399
```

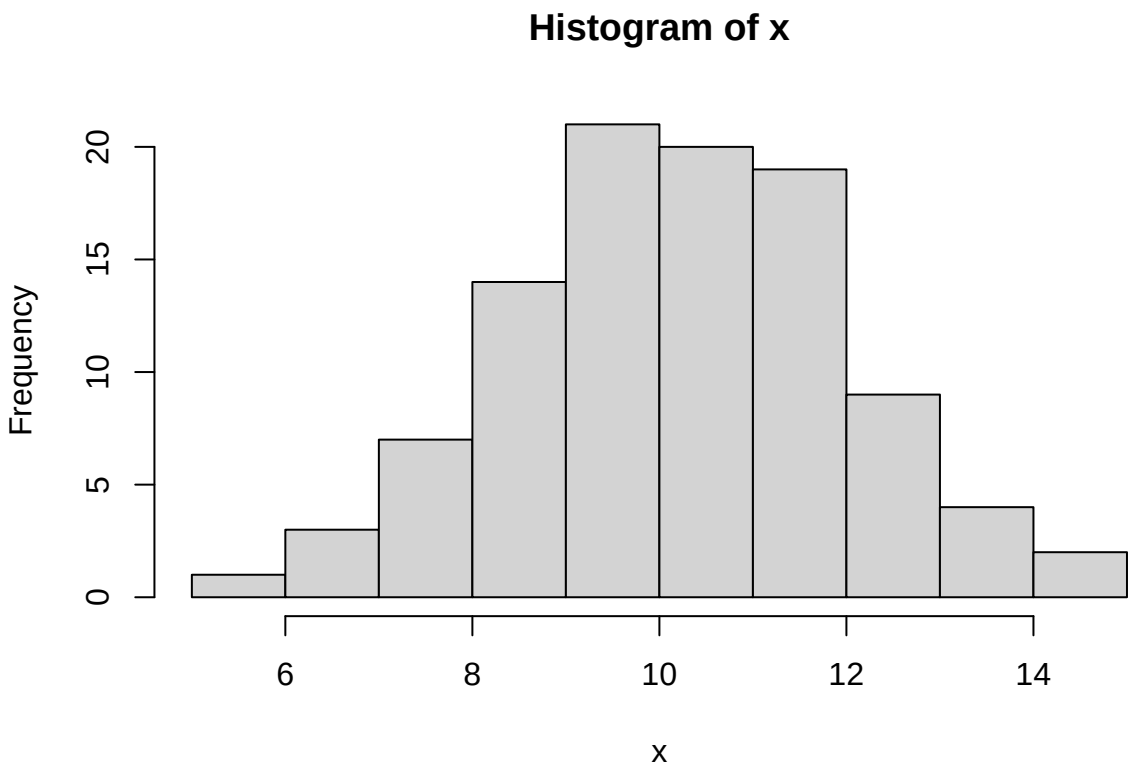
```
## [1] 1.796399
```

```
range(x) #(5.57060, 14.80324)
```

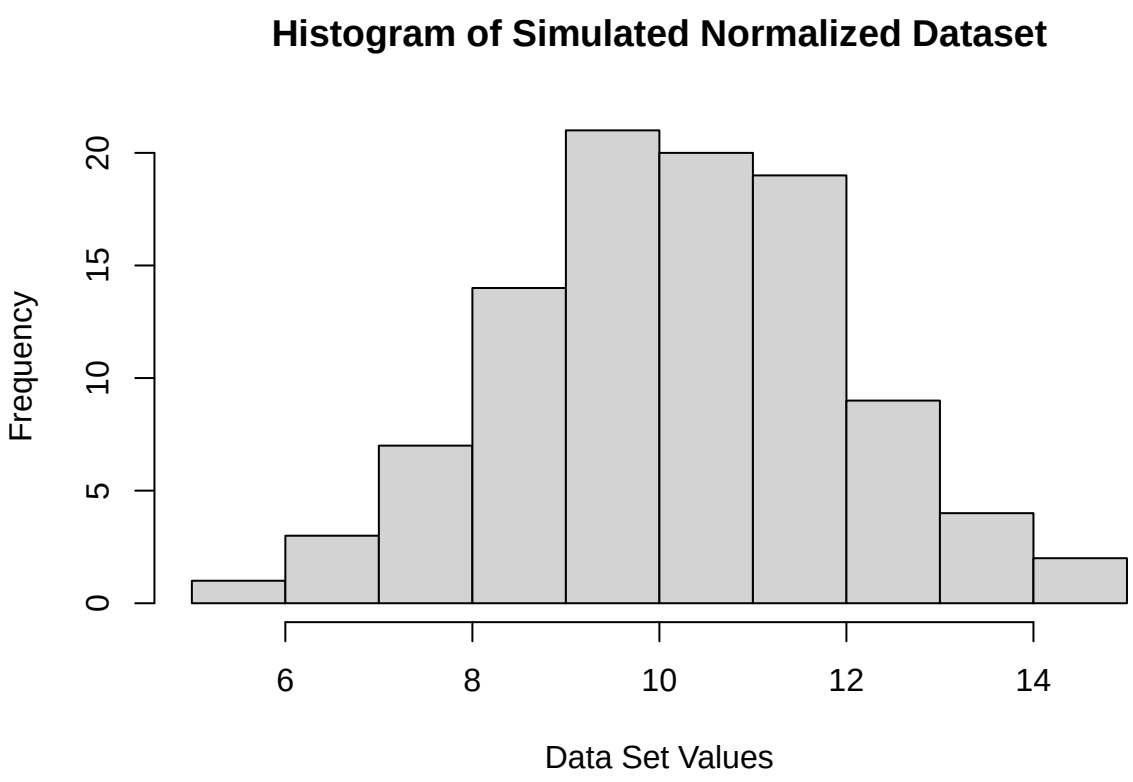
```
## [1]  5.57060 14.80324
```

Next, we can plot the data and see what it looks like visually.

```
#plot a basic histogram of the data
hist(x)
```




```
#Using skills that will be elaborated in the following workshop, we can even improve this
#plot by changing the main plot label and axes labels
hist(x,
     xlab = "Data Set Values",
     ylab = "Frequency",
     main = "Histogram of Simulated Normalized Dataset")
```




## Testing for Normality

Does the prior histogram look like a normal distribution to you?

What can we do if we don't know if our data is normalized? Test for normality.

### Parametric statistical tests

- Parametric tests will make assumptions about the distribution of the data and their validity relies upon the assumptions of the data being correct.
- Examples: Correlation, regression, t-test, and analysis of variance ANOVA)
- Normality and other assumptions that are associated with parametric tests are important to draw reliable interpretation and conclusions of the research.
- Before using parametric tests, we should perform some tests to ensure assumptions are met. Otherwise, when assumptions are violated, a non-parametric test is recommended.
- You should use your biological knowledge of your system to decide whether or not you expect your data to be normally distributed (or violate other assumptions). Often times your sample size will be too small to determine what the true distribution of your data is.

### Shapiro-Wilk's Test for Normality

- Visual inspection isn't the most reliable way to determine normality.
- Significance tests can allow us to compare our sample distribution to a normal distribution to determine whether our dataset deviates from normality

- The Shapiro-Wilk's test is a widely recommended test for normality based on the correlation between the data and the corresponding normal scores.
- The R function Shapiro.test() can be used to perform a Shapiro-Wilk's test of normality for one variable (univariate).

```
#perform a Shapiro-Wilk test for normality
shapiro.test(x) #p-value p = 0.9876
```

```
##
##  Shapiro-Wilk normality test
##
## data:  x
## W = 0.9956, p-value = 0.9876
```

```
#If the p-value is significant (p<0.05) the data is significantly different from a normal
#distribution. If the p-value is not significant (p>0.05) the data is not significantly
#different from a normal distribution.
```

The p-value is not significant, so it's safe to assume that the data are normal. We are more confident in our application of parametric statistical tests using this data, becaue the assumption of normally distributed data is not violated.

### Simulating another normalized dataset

What if we want to compare two different distributions that are normal and numeric?

Let's make a second dataset that we can compare to the first.

```
#setting a different seed than the first dataset
set.seed(5)
#create a second normalized dataset
y <- rnorm(n = 100,
           mean = 11,
           sd = 2)
```

We can again visualize different statistics from the second dataset.

```
#let's simply print out the dataset that we have created
y
```

```
##   [1]  9.318289 13.768719  8.489016 11.140286 14.422882  9.794184 10.055667
##   [8]  9.729257 10.428453 11.276216 13.455261  9.396441  8.839215 10.684931
##  [15]  8.856480 10.722028  9.805374  6.632066 11.481635 10.481289 12.801024
##  [22] 12.883739 13.935924 12.413522 12.638018 10.413036 13.837178 13.997548
##  [29]  9.685836  9.294409 11.631830 13.219388 15.430921 13.434207 13.958444
##  [36] 12.903148  8.980935  6.999055  7.475628 10.714784 14.100121  9.395154
##  [43] 10.850842 14.791336 10.086862 12.124447  9.225983 10.079511  9.551343
##  [50] 10.861578 13.926497 11.375452 13.044046  9.816330 10.775599  9.150094
##  [57] 12.506610 10.774782 10.871818 11.466551  8.726834 12.709661  9.843259
##  [64] 11.992723  9.479884 10.317227  6.795342 10.396595  8.455233 10.440668
##  [71] 10.591805 10.548772 11.694057 11.064736 11.827063 10.689303 12.946971
##  [78] 11.242180 11.378347  9.874230 11.996832  7.515395 12.951058 10.951834
```

```
##  [85] 12.351369  9.579381 15.774465 10.053136 10.848455  9.956320 12.852094
##  [92]  8.875178 12.114068 12.801461 12.979891 11.767216 10.306832  9.919621
##  [99] 10.634889 10.881401
```

```
#let's look at the mean, median, standard deviation, and range of our normalized dataset
mean(y) #11.06327
```

```
## [1] 11.06327
```

```
median(y) #10.81203
```

```
## [1] 10.81203
```

```
sd(y) #1.890569
```

```
## [1] 1.890569
```

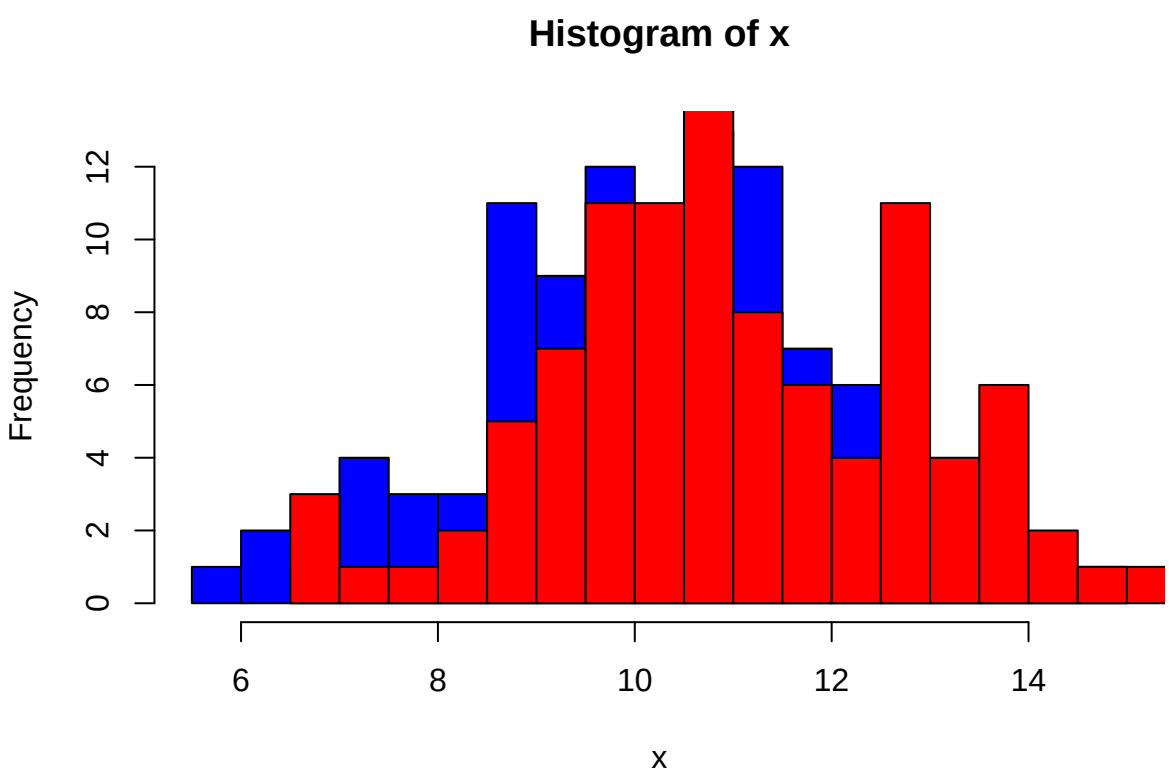
```
range(y) #(6.632066, 15.774465)
```

```
## [1]  6.632066 15.774465
```

We can also visualize the two datasets together to see how they deviate from each other.

```
#plot a histogram of the first set of data
hist(x = x,
     col = "blue",
     breaks = 20)

#plot a histogram of the second set of data
hist(x = y,
     col = "red",
     breaks = 20,
     add = T)
```




```
#These two lines of code may need to be run together to see the plots on top of each other.
#Take note of the "add = T" argument that tells R to generate graphics on top of the
#previous plot, rather than generate a new plot. Additional graphical arguments can be used
#to implement partial transparency or other effects, such that the red histogram doesn't
#completely block the blue histogram: reference the following "Base Graphics in R" module.
```

## Independent Samples (T-Test and Wilcoxon-Mann-Whitney)

### T-Test (Parametric)

- T tests help you to determine whether the means of two groups are equal to each other
- Assumptions of this test are that both groups are sampled from normal distributions with equal variances
- Null hypothesis: The means of the data sets are equal
- Alternative hypothesis: The means of the data sets are not equal

```
#perform a t test of the two datasets

#x = numeric vector of dataset 1
#y = numeric vector of dataset 2

t.test(x = x,
       y = y)
```

```
##
```

```
##  Welch Two Sample t-test
##
## data:  x and y
## t = -3.242, df = 197.49, p-value = 0.001393
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.3597919 -0.3311987
## sample estimates:
## mean of x mean of y
##  10.21777  11.06327
```

We see the above output which shows the p-value: 0.001393. This p-value is significant indicating that the means of the data sets are not equal. This is expected with the numbers we used to create each dataset, where the mean of the first data set is specified as 10 and the mean of the second data set is specified as 11.

If the data weren't normally distributed, we could run a non-parametric test instead.

### Wilcoxon-Mann-Whitney Test (Non-Parametric)

- A non-parametric alternative to the unpaired two-samples t-test
- This test can be used to compare two independent groups of samples that are not normally distributed.
- Null hypothesis: The means of the data sets are equal
- Alternative hypothesis: The means of the data sets are not equal

```
#perform a wilcox test of the two datasets

#x = numeric vector of dataset 1
#y = numeric vector of dataset 2

wilcox.test(x = x,
            y = y)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  x and y
## W = 3814, p-value = 0.003772
## alternative hypothesis: true location shift is not equal to 0
```

We still see a significant p-value ($p = 0.003772$), but this p-value is not as low as the previous p-value from the parametric t-test. This is common when comparing the statistical power of parametric vs non-parametric tests. What if our samples were instead paired or related?

## Dependent Samples (T-Test and Wilcoxon-Mann-Whitney)

### Paired T-Test (Parametric)

- The paired samples t-test compares the means between two related groups of samples or pairs of observations
- Paired t-test can only be used when the difference is normally distributed
- Null hypothesis: The true mean difference between the paired samples is 0
- Alternative hypothesis: The true mean difference between the paired samples is not equal to 0

```r
#perform a paired t test of the two datasets

#x = numeric vector of dataset 1
#y = numeric vector of dataset 2
#paired = true is the data is paired data

t.test(x = x,
       y = y,
       paired = TRUE)
```

```
##
##  Paired t-test
##
## data:  x and y
## t = -3.2742, df = 99, p-value = 0.00146
## alternative hypothesis: true mean difference is not equal to 0
## 95 percent confidence interval:
##  -1.3578724 -0.3331182
## sample estimates:
## mean difference
##      -0.8454953
```

We see the above output which shows the p-value: 0.00146. This p-value is different than we see above from the independent t-test. What else is different? Our degrees of freedom have also changed. We previously had 200 observations, but now we have 2 measurements of 100 observations reducing the degrees of freedom.

### Paired Wilcoxon-Mann-Whitney Test (Non-Parametric)

- The non-parametric alternative to the paired t-test is the paired Wilcoxon-Mann-Whitney test

```r
#perform a paired wilcox test of the two datasets

#x = numeric vector of dataset 1
#y = numeric vector of dataset 2
#paired = true is the data is paired data

wilcox.test(x = x,
            y = y,
            paired = TRUE)
```

```
##
##  Wilcoxon signed rank test with continuity correction
##
## data:  x and y
## V = 1612, p-value = 0.001704
## alternative hypothesis: true location shift is not equal to 0
```

We see the above output which shows the p-value: 0.001704. Similar to the t-test, this p-value from the paired Wilcoxon-Mann-Whitney is different than what we see above from the independent Wilcoxon-Mann-Whitney.

## Simulating Character/Group Data

So far we have been working with only numerical data, but in your research you may also have character or group data.

```
#letters funtion allows you to print out characters in the alphabet
rep(letters[1:2])
```

```
## [1] "a" "b"
```

```
#we can create a group vector that will allow us to add group labels to our dataset
#length.out specifies how many total observations you want in the vector
group <- rep(letters[1:2], length.out = 200)

#we're going to set our seed again, this time with a different seed
set.seed(6)

#create a normalized data set with values at two different means
response <- rnorm(n = 100, mean = c(10,11), sd = 2)

#place the normalized data and the group labels all together into a data frame
test.df <- data.frame(group, response)

#we can take a look at the top of this data frame without looking at the dentire data frame
#this is especially useful when you have really large datasets that you are loading in
head(test.df)
```

```
##   group  response
## 1     a 10.539212
## 2     b  9.740029
## 3     a 11.737320
## 4     b 14.454391
## 5     a 10.048375
## 6     b 11.736050
```

```
#we can use a function to look at the structure of our data frame
#this can help you to see the types of data that you are working with

#For example, when we look at our data frame that we have just created, we can see the
#number of observations and variables that are in our dataset. We can also see that we have
#a column of character data that is labelled as group and a column of numeric data that is
#labelled as response.

str(test.df)
```

```
## 'data.frame':    200 obs. of  2 variables:
##  $ group   : chr  "a" "b" "a" "b" ...
##  $ response: num  10.54 9.74 11.74 14.45 10.05 ...
```

## T-Test and Wilcoxon-Mann-Whitney with Groups

### T-Test by Group

- We can also perform a basic t-test by group

- We use the formula, X~Y, or the response or dependent variable (X) as a function of ("~") the independent variable (Y)

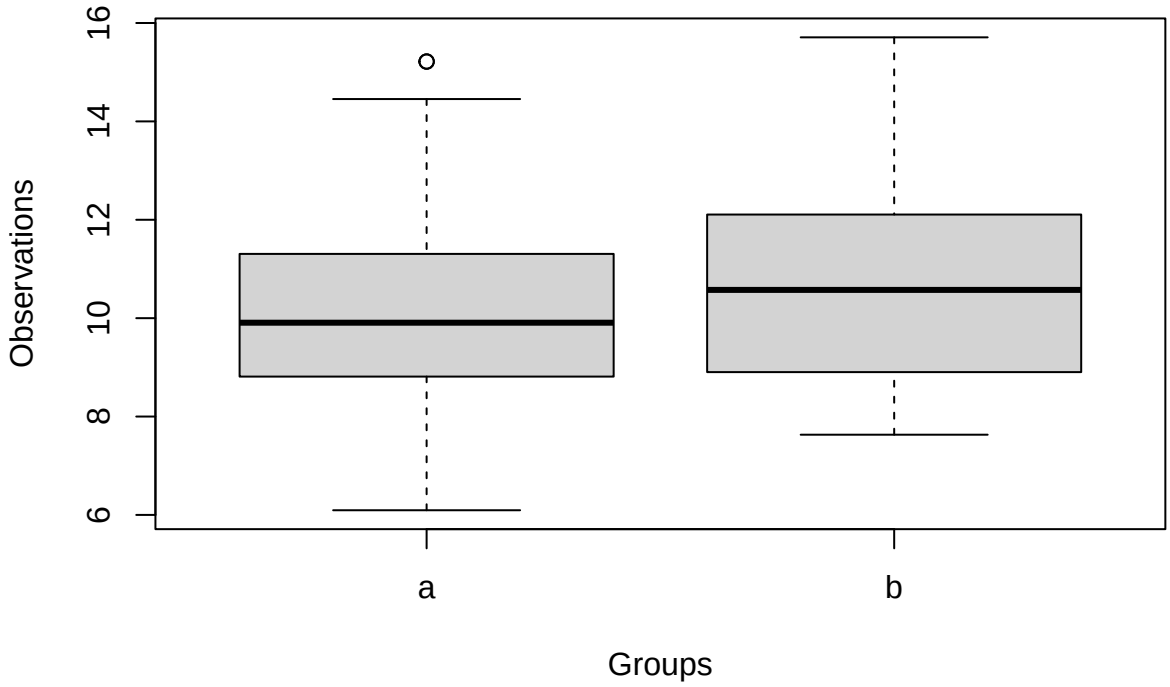
```
t.test(response~group,
       data = test.df)
```

```
##
##  Welch Two Sample t-test
##
## data:  response by group
## t = -2.1976, df = 197.55, p-value = 0.02914
## alternative hypothesis: true difference in means between group a and group b is not equal to 0
## 95 percent confidence interval:
##  -1.21415080 -0.06567404
## sample estimates:
## mean in group a mean in group b
##        10.15963        10.79954
```

### Visualizing the Grouped Data with a Boxplot

We can make a boxplot using a similar syntax and format to the grouped t-test above, and adding in labels for the X and Y axis

```
boxplot(response~group,
        data = test.df,
        xlab = "Groups",
        ylab = "Observations")
```




### Wilcoxon-Mann-Whitney by Group

- We can also perform a basic Wilcoxon-Mann-Whitney by group
- We use the formula as we did for grouped t-test above

```
wilcox.test(response~group,
            data = test.df)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  response by group
## W = 4176, p-value = 0.0442
## alternative hypothesis: true location shift is not equal to 0
```

## ANOVA (Analysis of Variance)

Now, let's think about if we had more than 2 groups.

### Simulating Data for ANOVA

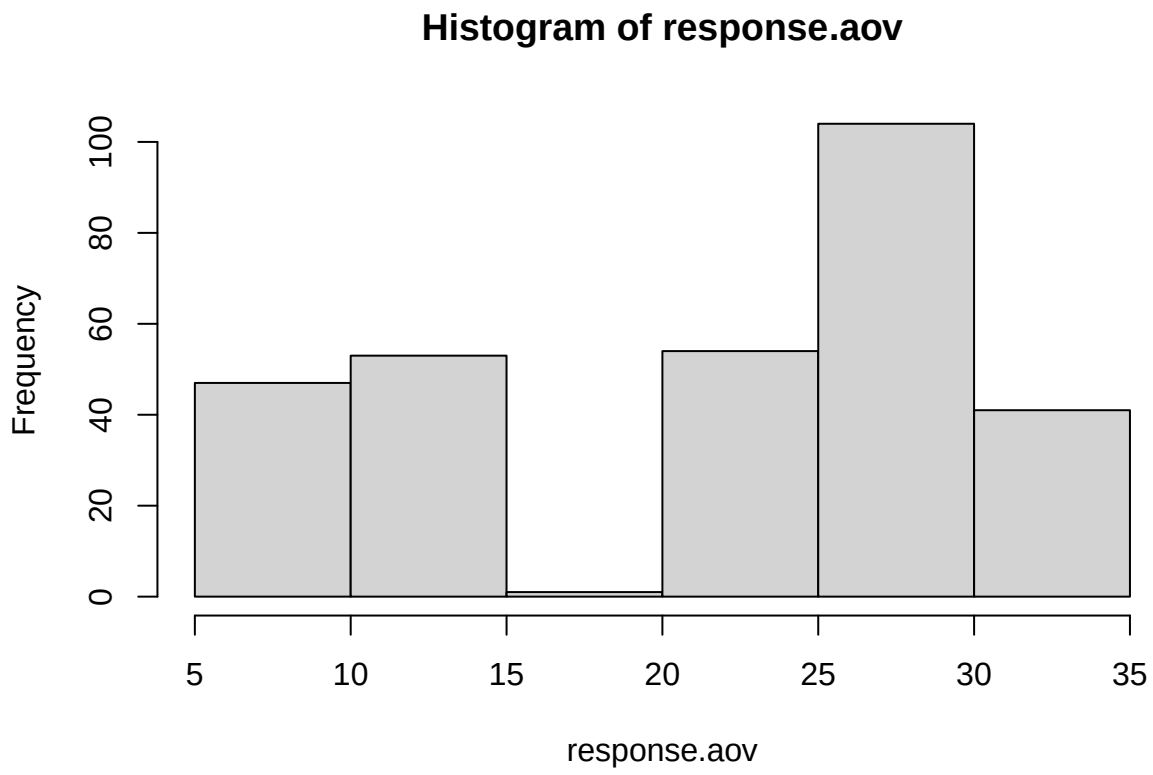
We are again going to simulate groups and responses, but with three groups this time.

```r
#create a vector of the group data for the ANOVA
group.aov <- as.factor(rep(letters[1:3], length.out = 300))

#create a vector of response data for the ANOVA
response.aov <- rnorm(n = 300,
                      mean = c(10,25,30),
                      sd = 2)
```

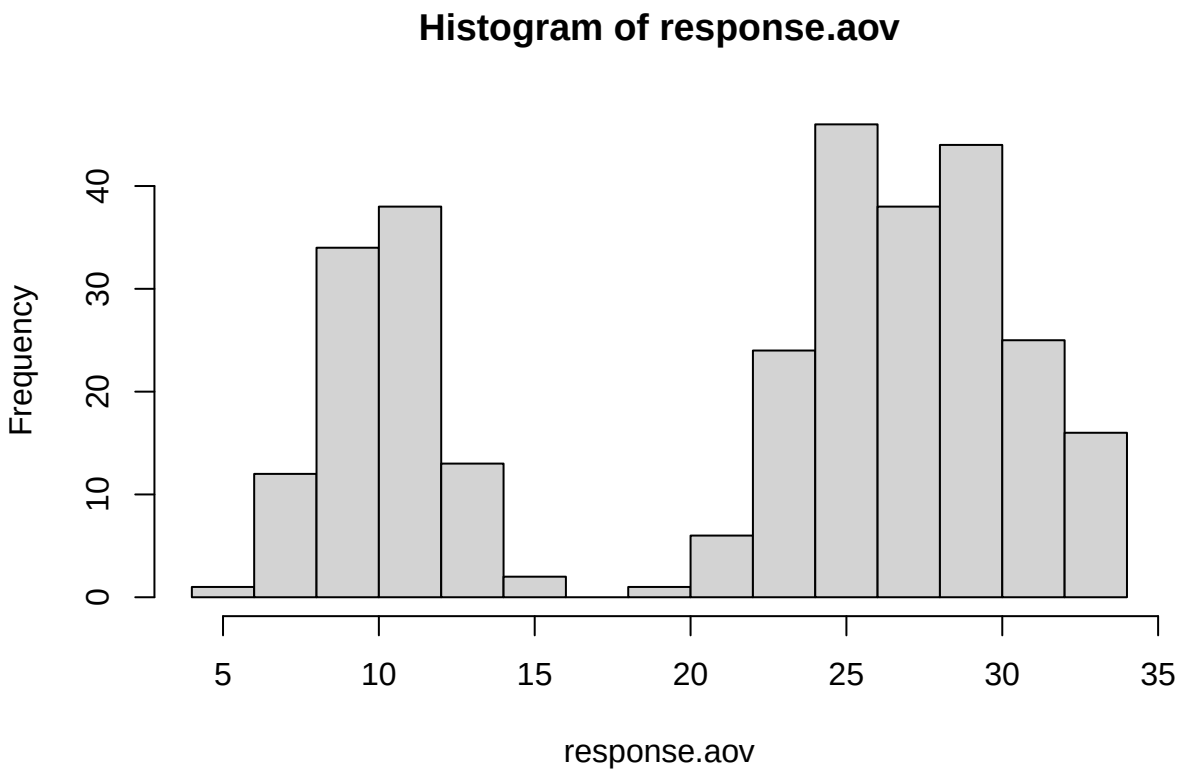
Let's again, visualize the data we have created.

```r
hist(response.aov)
```




We can't really see much when we visualize the data without breaking it up more.

```r
hist(response.aov,
     breaks = 20)
```




What did this do? Now we can see a lot more bins along the x-axis. The breaks function allows us to control the number of bars or bins of the histogram.

We are going to place the data we have created into a data frame.

```
test.aov <- data.frame(group.aov, response.aov)
```

Now, let's look at the type of data that we are working with by using the structure function.

```
str(test.aov)
```

```
## 'data.frame':    300 obs. of  2 variables:
##  $ group.aov   : Factor w/ 3 levels "a","b","c": 1 2 3 1 2 3 1 2 3 1 ...
##  $ response.aov: num  8.12 25.19 30.13 8.02 24.86 ...
```

You could also similarly head the data frame to see the first few rows.

```
head(test.aov)
```

```
##   group.aov response.aov
## 1         a     8.118466
## 2         b    25.187821
## 3         c    30.132507
## 4         a     8.018688
## 5         b    24.855770
## 6         c    29.553486
```

**ANOVA**

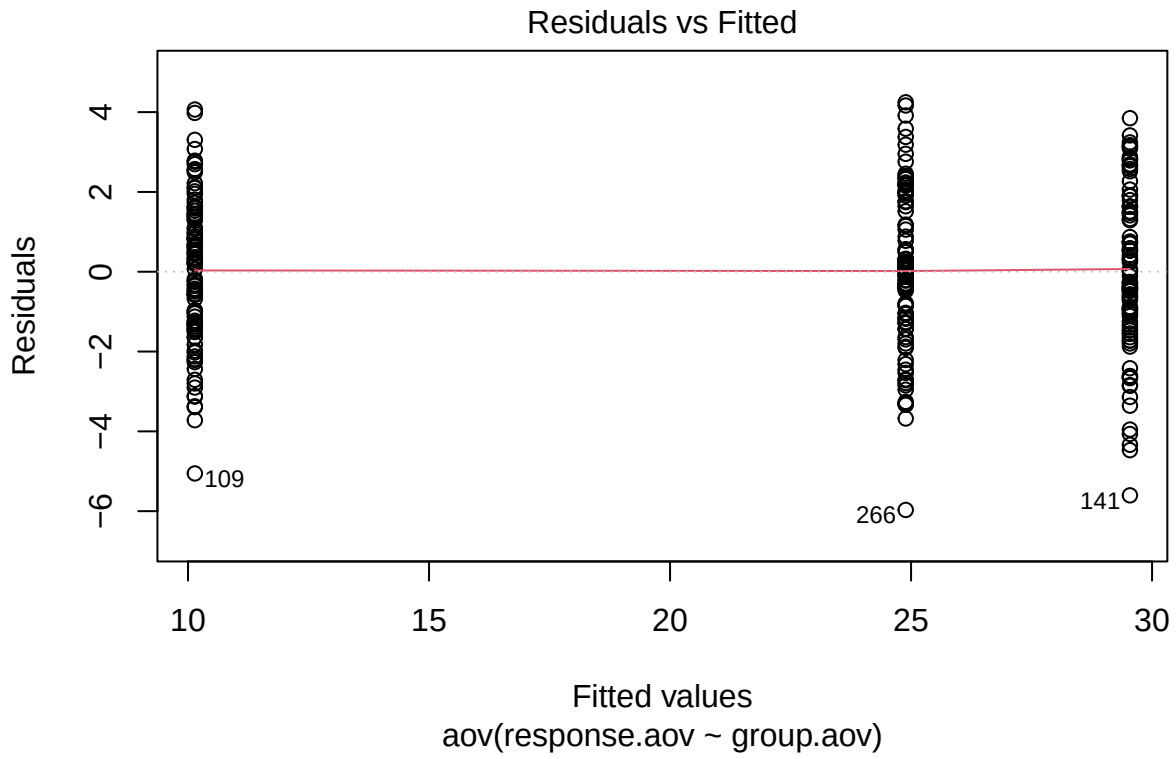
Let's set up the ANOVA (Analysis of Variance) test.

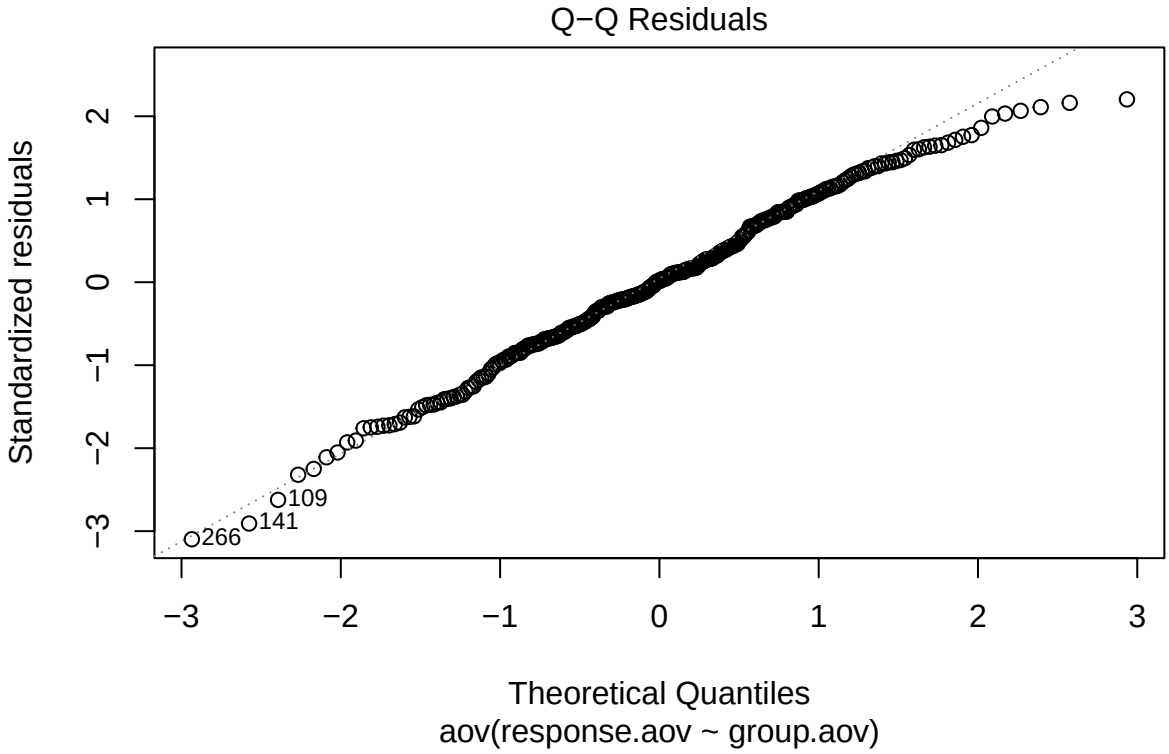
We again are going to use that same formula format: dependent ~ independent.

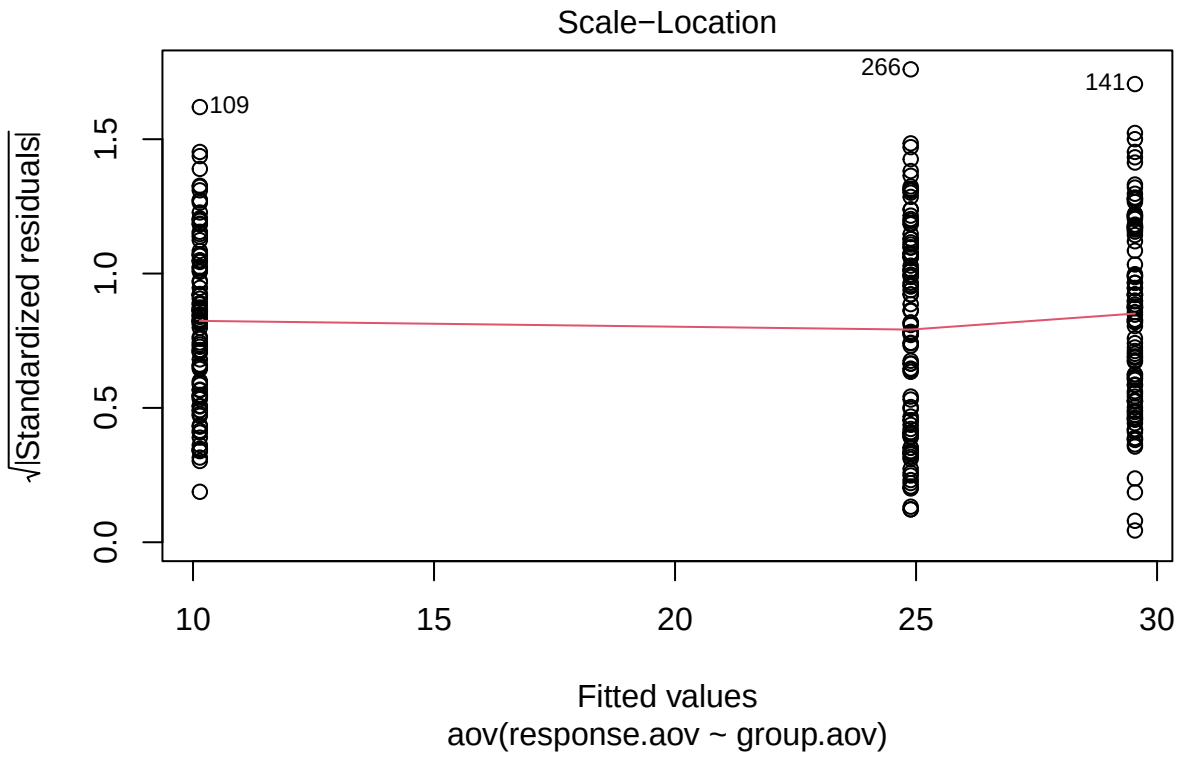
```
test.aov.fit <- aov(response.aov ~ group.aov,
                    data = test.aov)
```

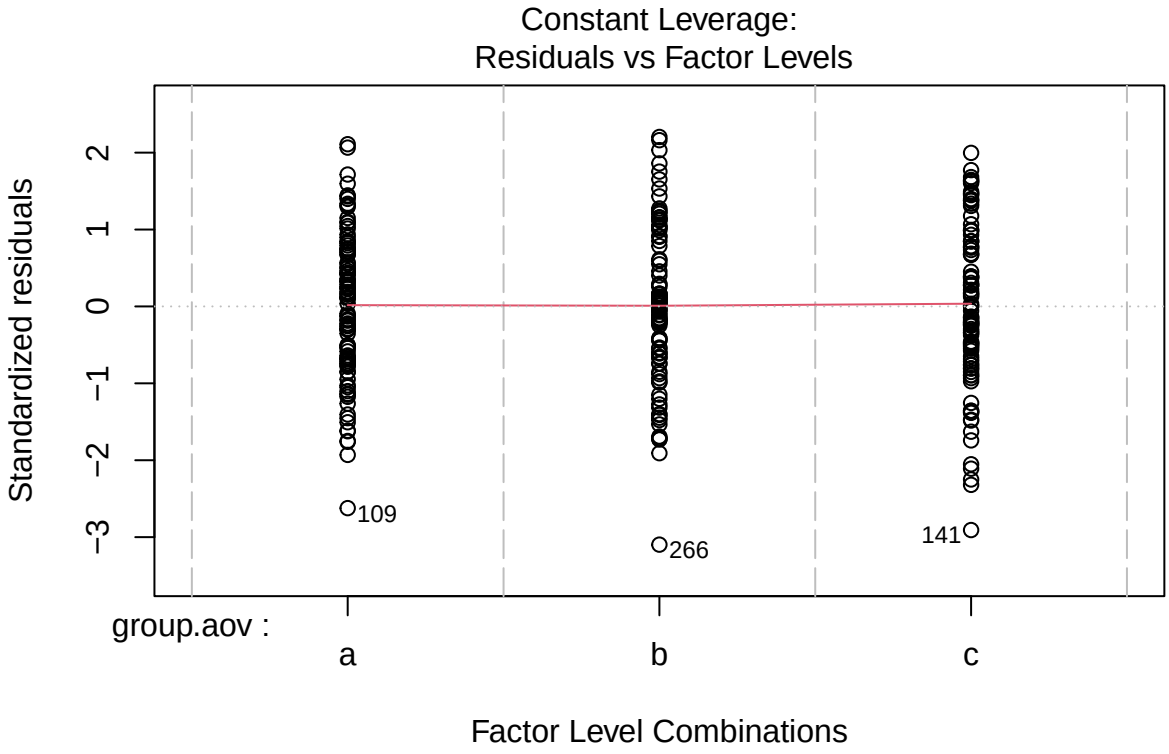
Let's visualize the aov function output using the plot function.

```
plot(test.aov.fit)
```




Q−Q Residuals
Standardized residuals
Theoretical Quantiles
aov(response.aov ~ group.aov)
266
141
109
−3
−2
−1
0
1
2
3


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
109
266
141
10
15
20
25
30
Fitted values
aov(response.aov ~ group.aov)




While these plots can be helpful, we noticeably don't see a p-value or other information associated with the ANOVA.

```
test.aov.fit
```

```
## Call:
##    aov(formula = response.aov ~ group.aov, data = test.aov)
##
## Terms:
##                 group.aov Residuals
## Sum of Squares   20516.80   1114.08
## Deg. of Freedom         2       297
##
## Residual standard error: 1.936779
## Estimated effects may be unbalanced
```

Let's use summary to get a summary of the statistical test we ran.

```
summary(test.aov.fit)
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## group.aov     2  20517   10258    2735 <2e-16 ***
## Residuals   297   1114       4
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We see that it's really significant, but this shouldn't be surprising given the values that we told the test to use.

### Tukey Post Hoc Test

An ANOVA can tell you if there is significance in your results overall, but it can't tell you where exactly those differences lie. After running an ANOVA and finding significance, you can run a Tukey test to compare all possible pairs of means and find out which specific group's means are different.

```
TukeyHSD(test.aov.fit)
```

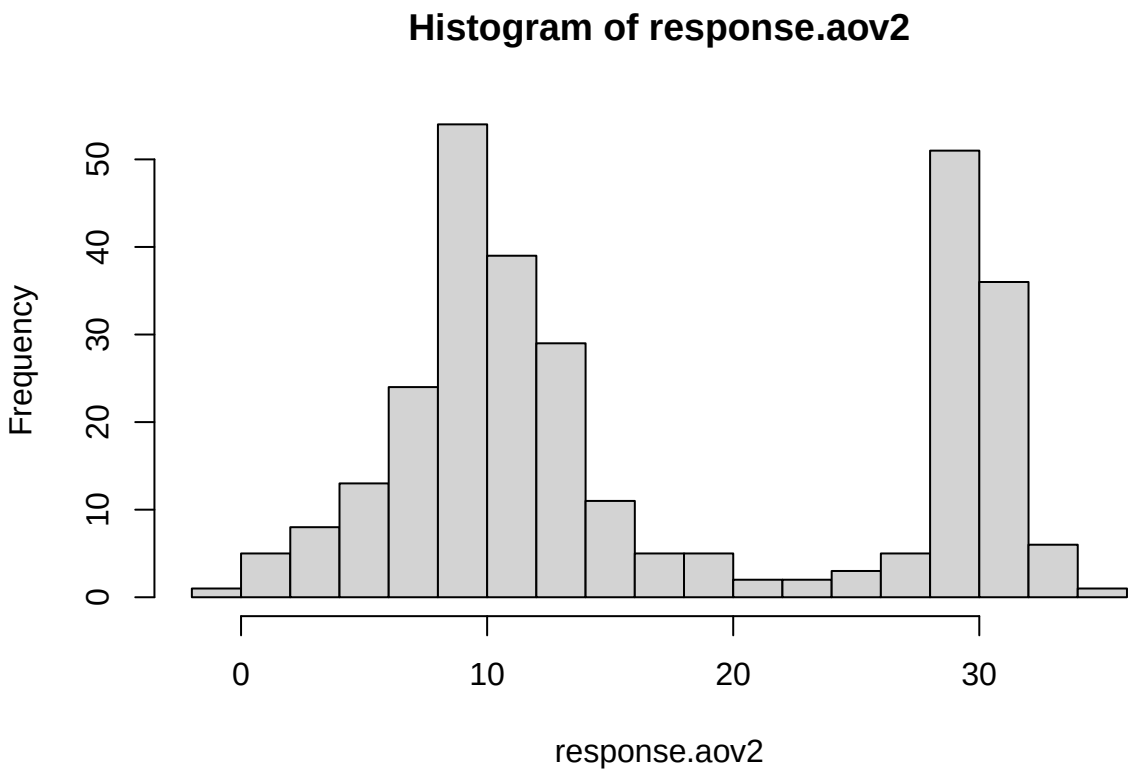
```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = response.aov ~ group.aov, data = test.aov)
##
## $group.aov
##          diff       lwr       upr p adj
## b-a 14.748184 14.103002 15.393367     0
## c-a 19.399908 18.754726 20.045091     0
## c-b  4.651724  4.006541  5.296907     0
```

This output will show us all of the comparisons between the pairs of means and we can see that all of them are different from one another. But what if that wasn't the case?

###ANOVA and Post Hoc (only some significant differences) This time we are going to simulate groups and responses again, but with three groups where two are very similar, but with different standard deviations.

Let's first create this data and then look at it with a histogram.

```
group.aov2<-rep(letters[1:3], length.out=300)
response.aov2<-rnorm(n=300, mean=c(10, 10.5, 30), sd=c(2,6,1.5))
hist(response.aov2, breaks=20)
```




Let's place our group and response data into a data frame to run the ANOVA.

```
test.aov2<-data.frame(group.aov2, response.aov2)
```

ANOVA time.

```
test.aov.fit2<-aov(response.aov2~group.aov2, data=test.aov2)
summary(test.aov.fit2)
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## group.aov2    2  25880   12940   989.6 <2e-16 ***
## Residuals   297   3883      13
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This result is still very significant, but what do we see if we run the Tukey test?

```
TukeyHSD(test.aov.fit2)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = response.aov2 ~ group.aov2, data = test.aov2)
##
```

```
## $group.aov2
##          diff        lwr       upr     p adj
## b-a  0.664413 -0.5401587  1.868985 0.3967147
## c-a 20.026745 18.8221736 21.231317 0.0000000
## c-b 19.362332 18.1577606 20.566904 0.0000000
```

We can see from the above Tukey output that we do not find a significant difference between groups A and B. However, this isn't surprising given the means that we simulated for these two groups.

### Non parametric Alternative to ANOVA

Is there a non parametric alternative to an ANOVA?

```
nonpara.aov<-kruskal.test(response.aov2~group.aov2, data=test.aov2)
nonpara.aov
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  response.aov2 by group.aov2
## Kruskal-Wallis chi-squared = 199.36, df = 2, p-value < 2.2e-16
```

There are additional options for post-hoc tests for the Kruskal Wallis Test, but those are beyond the scope of this workshop and we will not explore those right now.

### Multiple Response Variables Using Available Data sets in R

So far we have been working with pretty simple simulated datasets.

What is we are interested in more than one repose variable (dependent variable)?

We are going to use the iris flower dataset in R.

```
data("iris")
```

Let's be sure to look at the data that we are going to be using to see the format.

```
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

### MANOVA with Iris Dataset

We can see that in our iris dataset there are a few different numerical traits (sepal length, sepal width, petal length, petal width) and also species. We can ask ourselves if these dependent traits are impacted by one independent trait. For example, is sepal length and petal length significantly different between species?

To answer this question, we can use a MANOVA.

In order to do this, we need to bind together our two dependent variables of interest (sepal length and petal length).

```
results.manova<-manova(cbind(iris$Sepal.Length, iris$Petal.Length)~iris$Species)
summary(results.manova)
```

```
##               Df Pillai approx F num Df den Df    Pr(>F)
## iris$Species   2 0.9885   71.829      4    294 < 2.2e-16 ***
## Residuals    147
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We can also look at each individual result separately.

```
summary.aov(results.manova)
```

```
##  Response 1 :
##               Df Sum Sq Mean Sq F value    Pr(>F)
## iris$Species   2 63.212  31.606  119.26 < 2.2e-16 ***
## Residuals    147 38.956   0.265
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##  Response 2 :
##               Df Sum Sq Mean Sq F value    Pr(>F)
## iris$Species   2 437.10 218.551  1180.2 < 2.2e-16 ***
## Residuals    147  27.22   0.185
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From these results, we can see that both traits are significantly different by species.

### Continuous variables and correlation data

So far we have covered basic response by group, but what about continuous variables and correlation data?

We are again going to call in another dataset that is already preloaded into R.

```
data("mtcars")
```

Let's be sure to look at the data that we are going to be using to see the format.

```
str(mtcars)
```

```
## 'data.frame':    32 obs. of  11 variables:
##  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
##  $ disp: num  160 160 108 258 360 ...
##  $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
```

```
##  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec: num  16.5 17 18.6 19.4 17 ...
##  $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
##  $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
##  $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
##  $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

### Setting up linear models

There is a lot of data to look at in this dataset, but let's keep it simple.

What is directly impacting the fuel efficiency (mpg) of these cars?

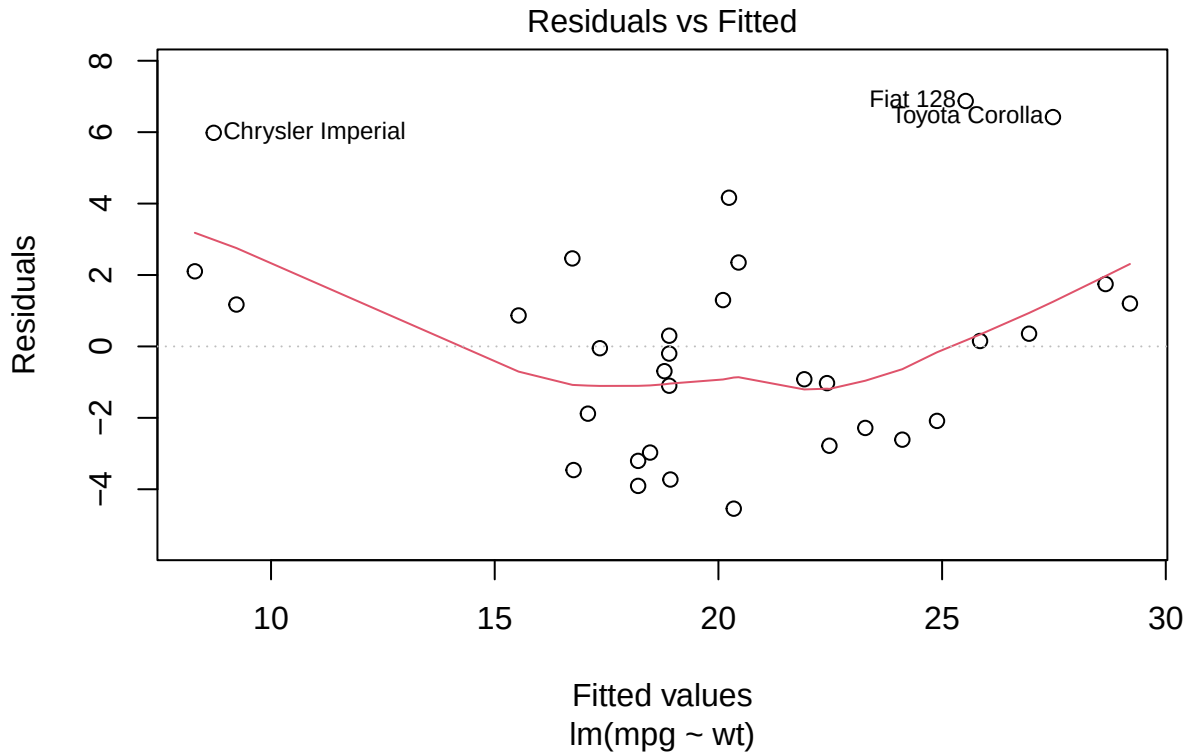
We are going to set up a linear model and use the same formula that we were using before (dependent~independent).

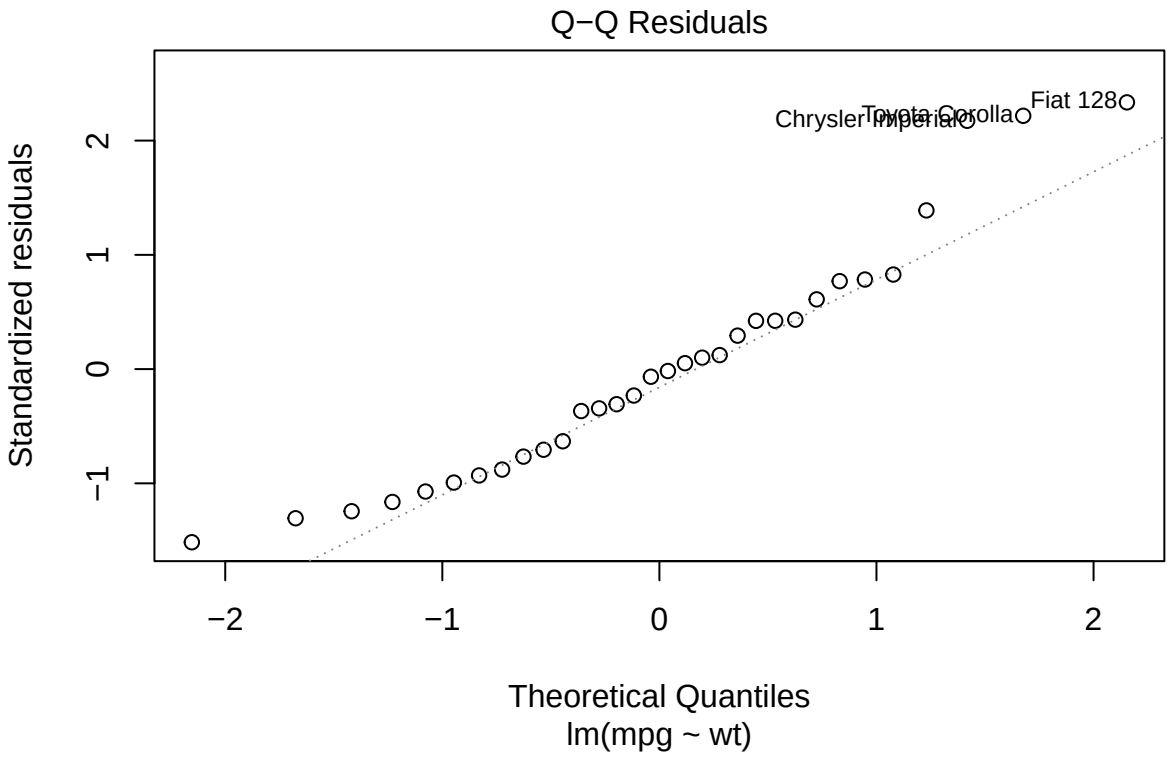
Let's start with a single variable, weight, and ask if it impacts fuel efficiency (mpg).

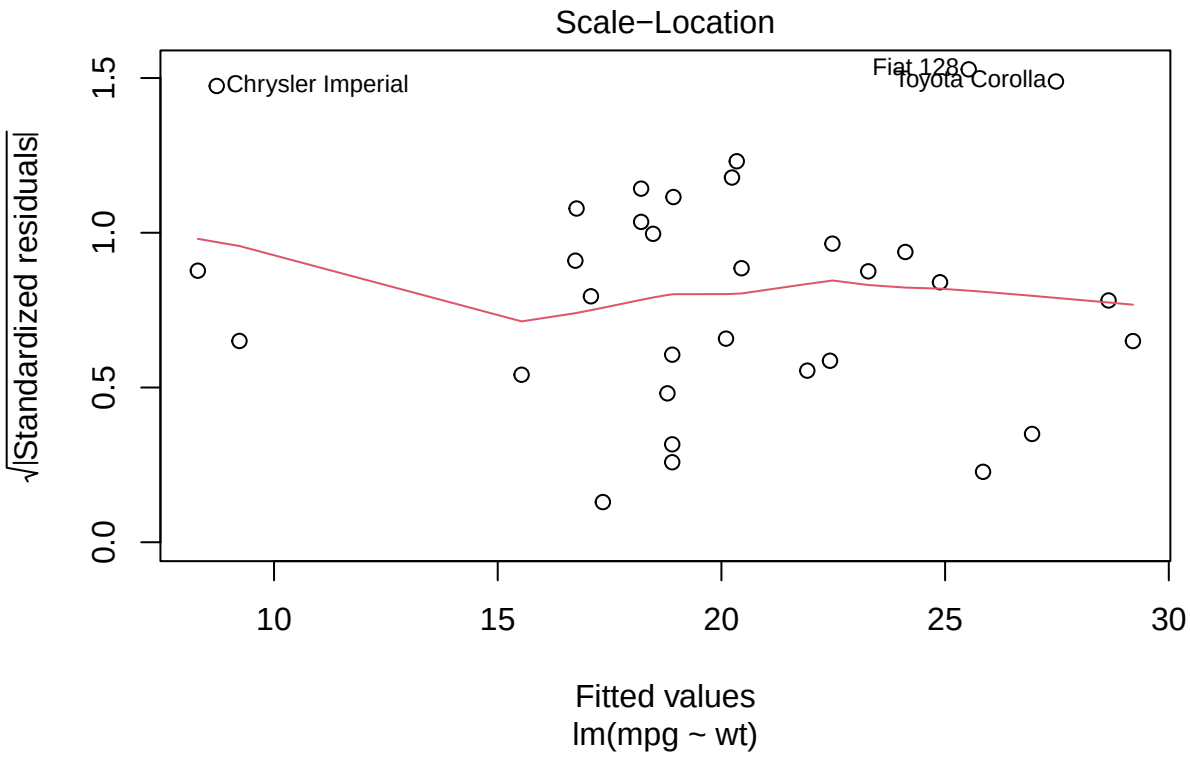
```
carfit<-lm(mpg~wt, data=mtcars)
```

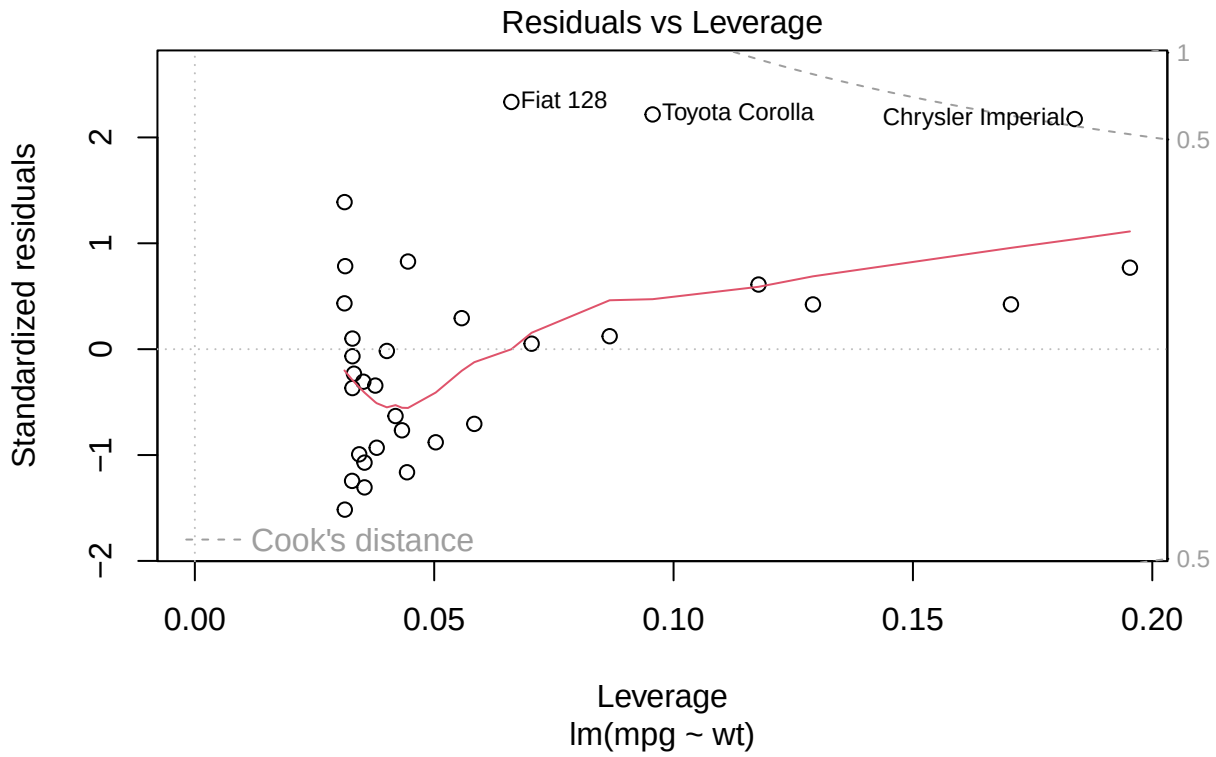
If you want, you can check a plot of the residuals.

```
plot(carfit)
```




Q−Q Residuals
Standardized residuals
−1
0
1
2
Chrysler Imperial
Toyota Corolla
Fiat 128
−2
−1
0
1
2
Theoretical Quantiles
lm(mpg ~ wt)


Scale−Location
Chrysler Imperial
Fiat 128
Toyota Corolla
√|Standardized residuals|
0.0
0.5
1.0
1.5
10
15
20
25
30
Fitted values
lm(mpg ~ wt)




What does the output of the linear model look like?

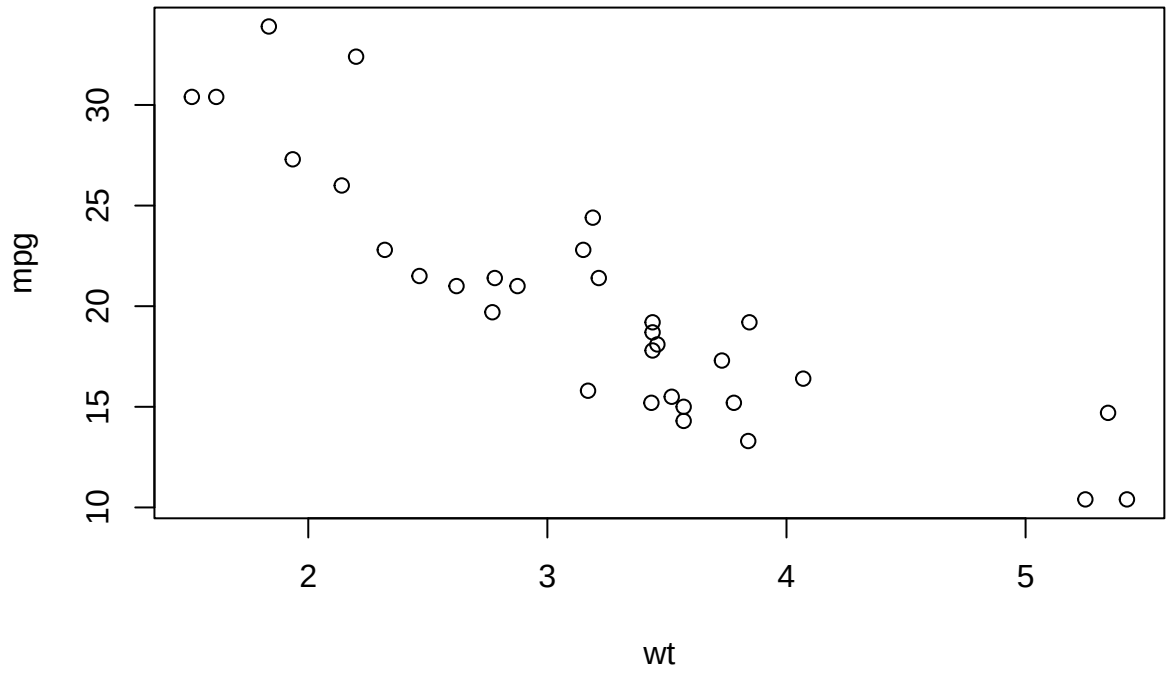
```
summary(carfit)
```

```
##
## Call:
## lm(formula = mpg ~ wt, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5432 -2.3647 -0.1252  1.4096  6.8727
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  37.2851     1.8776  19.858  < 2e-16 ***
## wt           -5.3445     0.5591  -9.559 1.29e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.046 on 30 degrees of freedom
## Multiple R-squared:  0.7528, Adjusted R-squared:  0.7446
## F-statistic: 91.38 on 1 and 30 DF,  p-value: 1.294e-10
```

When we look at the model output we can see a highly significant result, with a strong R^2 value (% of variation in mpg explained by weight)

What do the data look like when we plot them?

```r
plot(mpg~wt, data=mtcars)
```




Now, let's see how our model looks if we add another variable in addition to the weight.

Let's also add in cylinders to ask if weight and cylinder number affect fuel efficiency (mpg).

```r
carfit2<-lm(mpg~wt+cyl, data=mtcars)
```

We see that the results is significant and has a higher R^2 value. Does this matter?

```r
summary(carfit2)
```

```
##
## Call:
## lm(formula = mpg ~ wt + cyl, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.2893 -1.5512 -0.4684  1.5743  6.1004
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  39.6863     1.7150  23.141  < 2e-16 ***
## wt           -3.1910     0.7569  -4.216 0.000222 ***
## cyl          -1.5078     0.4147  -3.636 0.001064 **
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.568 on 29 degrees of freedom
## Multiple R-squared:  0.8302, Adjusted R-squared:  0.8185
## F-statistic: 70.91 on 2 and 29 DF,  p-value: 6.809e-12
```

How can we determine which model is better?

```
anova(carfit, carfit2)
```

```
## Analysis of Variance Table
##
## Model 1: mpg ~ wt
## Model 2: mpg ~ wt + cyl
##   Res.Df    RSS Df Sum of Sq     F   Pr(>F)
## 1     30 278.32
## 2     29 191.17  1     87.15 13.22 0.001064 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This output shows that there is a significantly better fit with the more complex model (p = 0.001064)

If the resulting p-value is sufficiently low (usually less than 0.05), we conclude that the more complex model is significantly better than the simpler model, and thus favor the more complex model. If the p-value is not sufficiently low (usually greater than 0.05), we should favor the simpler model

### Interaction Data

But what if there is an interaction effect between our predictor variables? If a car's weight can be mostly attributed to the engine, and if engine size (and weight) is correlated to the cylinder number, maybe there is an effect? An interaction is denoted with an * instead of a + in the linear model function, lm().

```
carfit3<-lm(mpg~wt+cyl+wt*cyl, data=mtcars)
summary(carfit3)
```

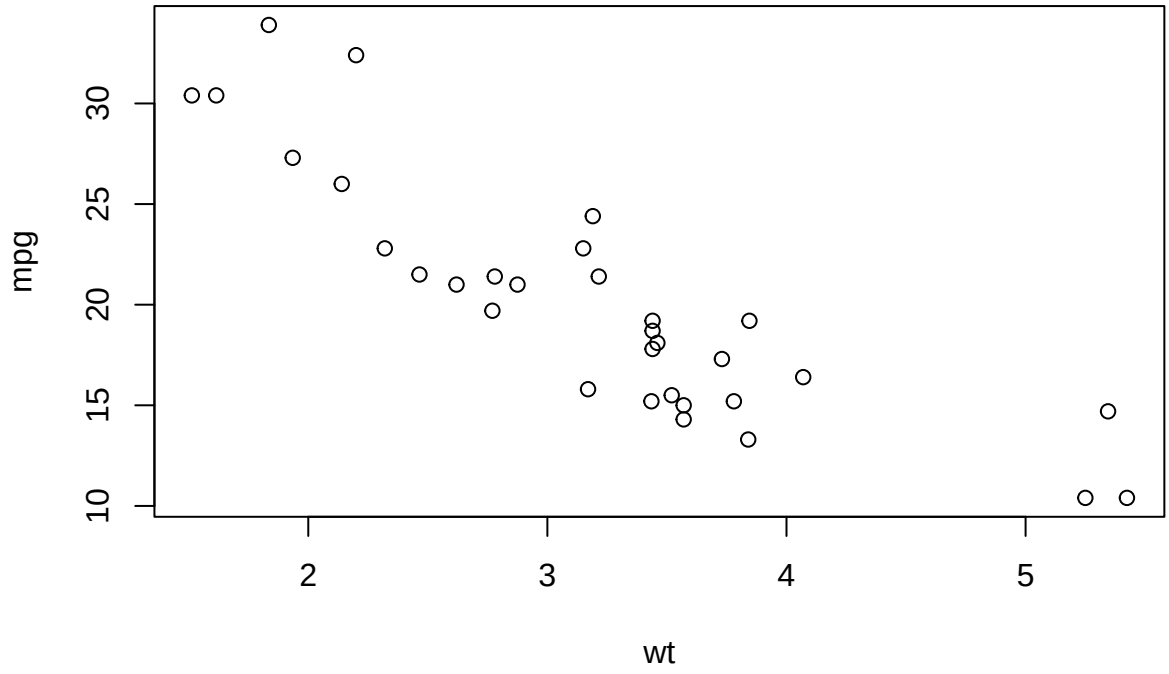
```
##
## Call:
## lm(formula = mpg ~ wt + cyl + wt * cyl, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.2288 -1.3495 -0.5042  1.4647  5.2344
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  54.3068     6.1275   8.863 1.29e-09 ***
## wt           -8.6556     2.3201  -3.731 0.000861 ***
## cyl          -3.8032     1.0050  -3.784 0.000747 ***
## wt:cyl        0.8084     0.3273   2.470 0.019882 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 2.368 on 28 degrees of freedom
## Multiple R-squared:  0.8606, Adjusted R-squared:  0.8457
## F-statistic: 57.62 on 3 and 28 DF,  p-value: 4.231e-12
```

We see that wight is significant in the model, as is cylinder number, BUT we also see a significant interaction between cylinder number and weight.

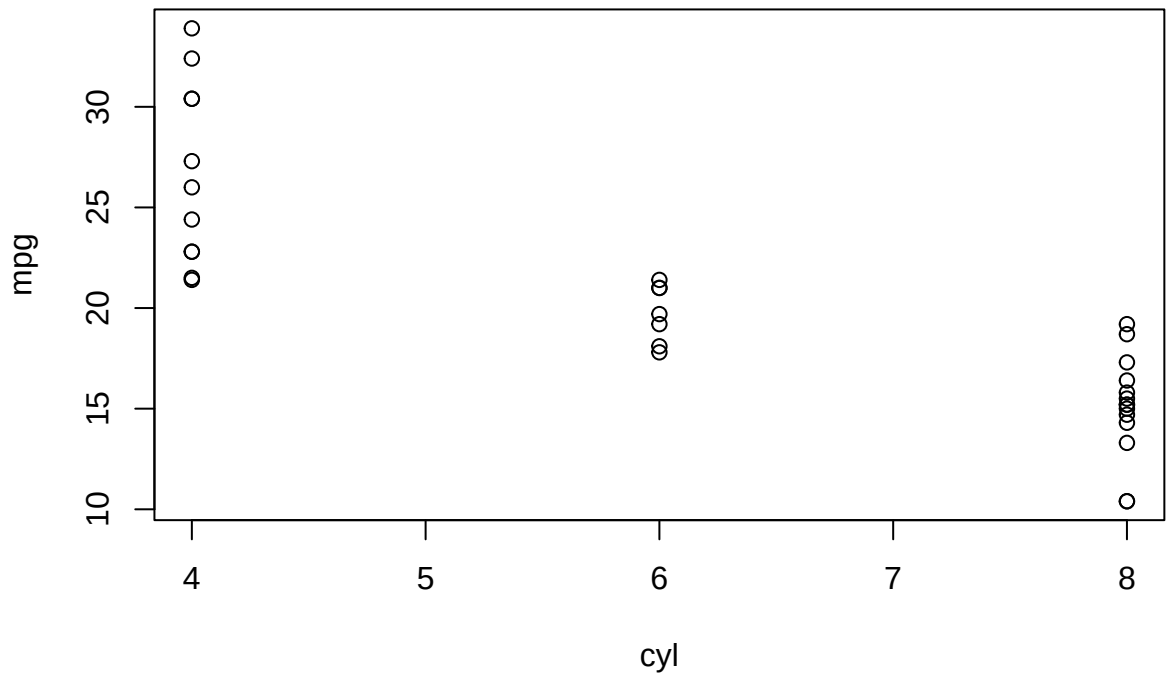
What is going on here? We know that as weight goes up, the mpg goes down.

```
plot(mpg~wt, data=mtcars)
```



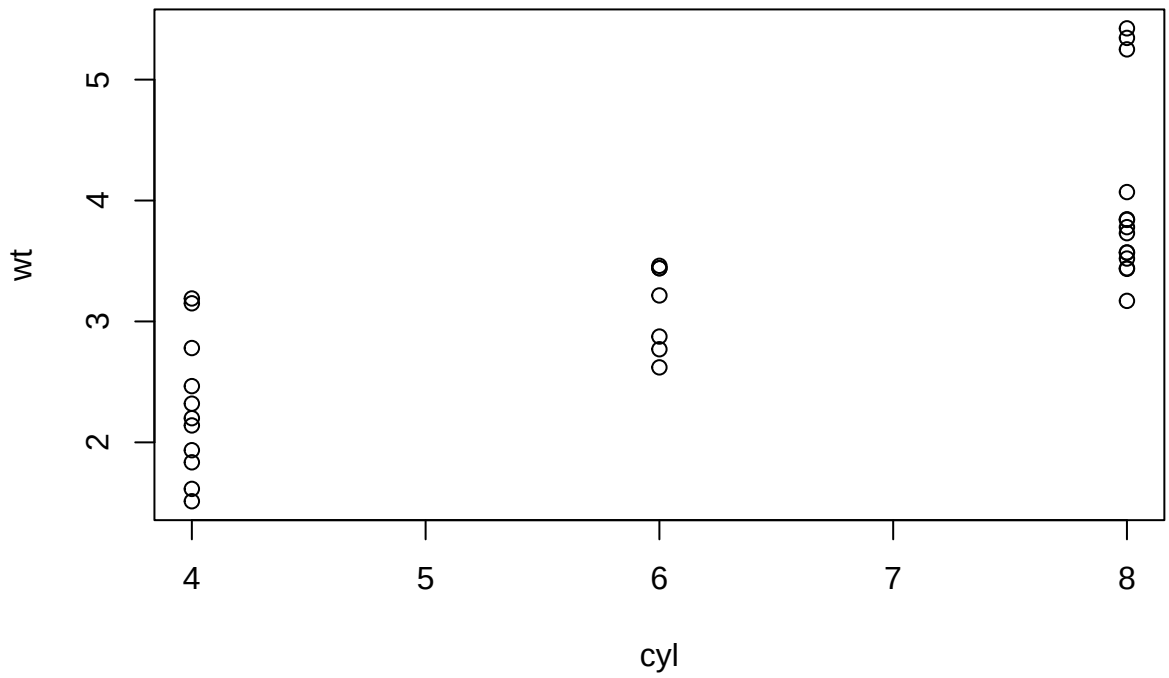

And we also know that as the cylinders go up, the mpg goes down too.

```
plot(mpg~cyl, data=mtcars)
```




What happens to the weight as cylinders go up?

```
plot(wt~cyl, data=mtcars)
```




If we look at our summary again, we can see that the estimate for the interaction is positive, whereas the estimates for the wt and cyl independently are negative. This means, while wt and cyl are both negatively impacting mpg, as cylinder number increases, the effect of wt on mpg decreases.

There is an interaction of these variables, and if we compare the simpler wt+cyl model to our more complex model, the more complex model is significantly better.

```
anova(carfit2, carfit3)
```

```
## Analysis of Variance Table
##
## Model 1: mpg ~ wt + cyl
## Model 2: mpg ~ wt + cyl + wt * cyl
##   Res.Df    RSS Df Sum of Sq      F  Pr(>F)
## 1     29 191.17
## 2     28 156.98  1    34.196 6.0995 0.01988 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

### Marginal Frequency Tables

What do we do if we have categorical data? Data from a survey or phenotypes from genetic crosses? What if we wanted to know if there is a more common combination of hair and eye color?

Let's again use an available dataset in R

```
data("HairEyeColor")
```

And let's look at the data that we are working with.

```
str(HairEyeColor)
```

```
##  'table' num [1:4, 1:4, 1:2] 32 53 10 3 11 50 10 30 10 25 ...
##  - attr(*, "dimnames")=List of 3
##   ..$ Hair: chr [1:4] "Black" "Brown" "Red" "Blond"
##   ..$ Eye : chr [1:4] "Brown" "Blue" "Hazel" "Green"
##   ..$ Sex : chr [1:2] "Male" "Female"
```

```
HairEyeColor
```

```
## , , Sex = Male
##
##        Eye
## Hair    Brown Blue Hazel Green
##   Black    32   11    10     3
##   Brown    53   50    25    15
##   Red      10   10     7     7
##   Blond     3   30     5     8
##
## , , Sex = Female
##
##        Eye
## Hair    Brown Blue Hazel Green
##   Black    36    9     5     2
##   Brown    66   34    29    14
##   Red      16    7     7     7
##   Blond     4   64     5     8
```

As you can see from above, the data is in a bit of an odd format as it is separated by sex. However, for our question we are not interested in sex as a factor, so we can use the function margin.table to give us all of our output for a marginal frequency table

```
HairEyeNew<-margin.table(HairEyeColor, margin=c(1,2))
```

Let's see what this new table looks like.

```
HairEyeNew
```

```
##        Eye
## Hair    Brown Blue Hazel Green
##   Black    68   20    15     5
##   Brown   119   84    54    29
##   Red      26   17    14    14
##   Blond     7   94    10    16
```

We can see that all of the data that we need to answer our question is in a usable table now. However, we are not interested in the raw numbers.

### Chi Square Test

What kind of test do we use to see if something is occurring more or less frequently than expected? Chi square is a common statistical test used to determine whether there is a statistically significant different between expected and observed frequencies.

We can use a Chi Square test to answer our question with the marginal frequency table that we created from above.

```
chisq.test(HairEyeNew)
```

```
##
##  Pearson's Chi-squared test
##
## data:  HairEyeNew
## X-squared = 138.29, df = 9, p-value < 2.2e-16
```

From this output, we can see that something is significant, but we want to see what frequencies we have. In order to calculate frequency, we also need to divide what we have by the totals.

```
HairEyeNew/sum(HairEyeNew)
```

```
##        Eye
## Hair          Brown        Blue       Hazel       Green
##   Black 0.114864865 0.033783784 0.025337838 0.008445946
##   Brown 0.201013514 0.141891892 0.091216216 0.048986486
##   Red   0.043918919 0.028716216 0.023648649 0.023648649
##   Blond 0.011824324 0.158783784 0.016891892 0.027027027
```

We can also check what the expected frequencies are from the chi square test output.

```
chisq.test(HairEyeNew)$expected
```

```
##        Eye
## Hair        Brown      Blue    Hazel     Green
##   Black  40.13514  39.22297 16.96622 11.675676
##   Brown 106.28378 103.86824 44.92905 30.918919
##   Red    26.38514  25.78547 11.15372  7.675676
##   Blond  47.19595  46.12331 19.95101 13.729730
```

```
chisq.test(HairEyeNew)$expected/sum(HairEyeNew)
```

```
##        Eye
## Hair         Brown       Blue      Hazel      Green
##   Black 0.06779584 0.06625502 0.02865915 0.01972243
##   Brown 0.17953342 0.17545311 0.07589367 0.05222790
##   Red   0.04456949 0.04355654 0.01884074 0.01296567
##   Blond 0.07972288 0.07791100 0.03370104 0.02319211
```