Introduction to Statistics and Data Analysis

Geoffrey M. Boynton
Department of Psychology
University of Washington

Latest build: November 05, 2024

Contents

Chapter 1

Introduction

This book contains materials for Psychology 522/524, the first quarter graduate statistics course in the Department of Psychology at the University of Washington. It's very much a work in progress.

A pdf version of this book can be found here: http://courses.washington.edu/psy524a/_book/Psych_524A_statistics_textbook.pdf This book was written using R's 'bookdown', and its Pdf format is finicky, so there may be some formatting issues with the pdf version. I'm still working on it.

I've been teaching statistics at the undergraduate and graduate level for a couple of decades now. To be honest, I took on the undergraduate stats course, Psychology 315, because teaching it was easy. I had TA'd undergraduate statistics in psychology back as a graduate student in the 90's and when I took on undergrad stats course at UW in 2010, and then graduate stats in 2013 the course material hadn't changed in 20 years. All textbooks were pretty much the same, covering hypothesis tests, binomial distributions, correlations, simple ANOVA, all using tables in the back of the book to get p-values from known standard distributions. I used to joke that teaching 315 year after year was easy because it didn't require me to keep up with the literature.

But then things started to change. Although it was created back in 1993, the statistical programming language R started to gain popularity in the 2010's probably due to the availability of cheap, fast laptops and the push toward open source languages and free data sets. When I took over the grad stats course in 2013 everything was done in SPSS. Back then I surveyed the faculty about whether they thought it'd be useful for me to teach using R there was a resounding vote of 'no'. So for the first few years I taught a hybrid class, each year emphasizing R more and SPSS less. By 2019 I had dropped SPSS entirely. The sounds of the students in lab course (522) has gradually switched from mouse clicking to keyboard typing.

Switching to R lead to two major changes in the way my students learn and use statistics. The first is the elimination of tables for looking up probabilities. A significant portion of my old notes and lectures involved explaining which table to use and where to find the answer in the table. With R, this has been replaced with a single command like 'pnorm','pt', or 'pf'. The second major change is the transition from teaching ANOVA using sums-of-squares to using regression. R's 'lm' and 'lmer' functions provide a natural way to conduct ANOVA tests (along with a bunch of other tests) using regression and the linear model. My lectures are now less filled with SS's all over the board.

By the way, statisticians have always emphasized (often snidely) that 'ANOVA is just regression', but I've never found a good resource that really explains why. Hopefully my chapters here on how ANOVA is just regression will help make this link more clear.

Chapter 2

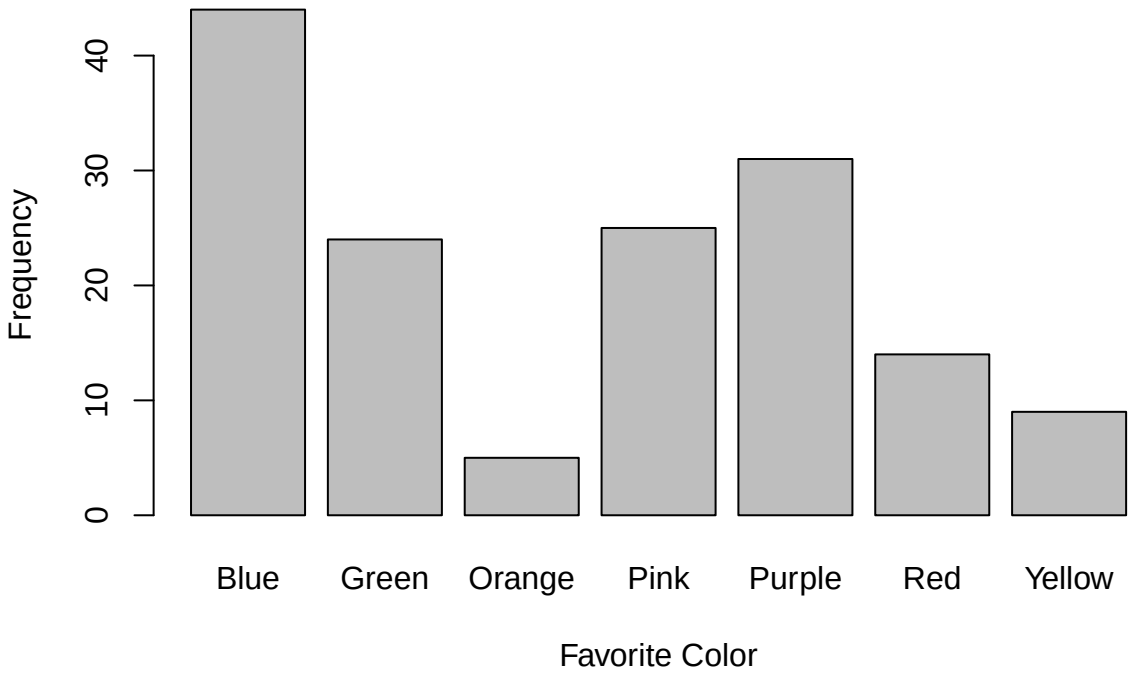
Frequency Distributions

The frequency distribution of a sample is a count of scores that fall into specific categories. Frequency distributions are very often shown as a bar graph, or histogram. You can make a histogram from nominal scale data or continuous data, but for continuous data you have to decided which ranges of scores fall into which 'class interval'. We'll first start with nominal scale data.

2.1 Nominal scale data

The easiest frequency distributions to understand and plot are for nominal scale data, where your measure falls into discrete categories. For example, in the survey, I asked students what their favorite color is. We can use R's 'table' function to tabulate the number of students that prefer each color, and 'barplot' to plot the resulting frequencies:

```
survey <- read.csv("http://www.courses.washington.edu/psy315/datasets/Psych315W21survey.csv")
color.distribution <- table(survey$color)
barplot(color.distribution,xlab = "Favorite Color",ylab = "Frequency")
```

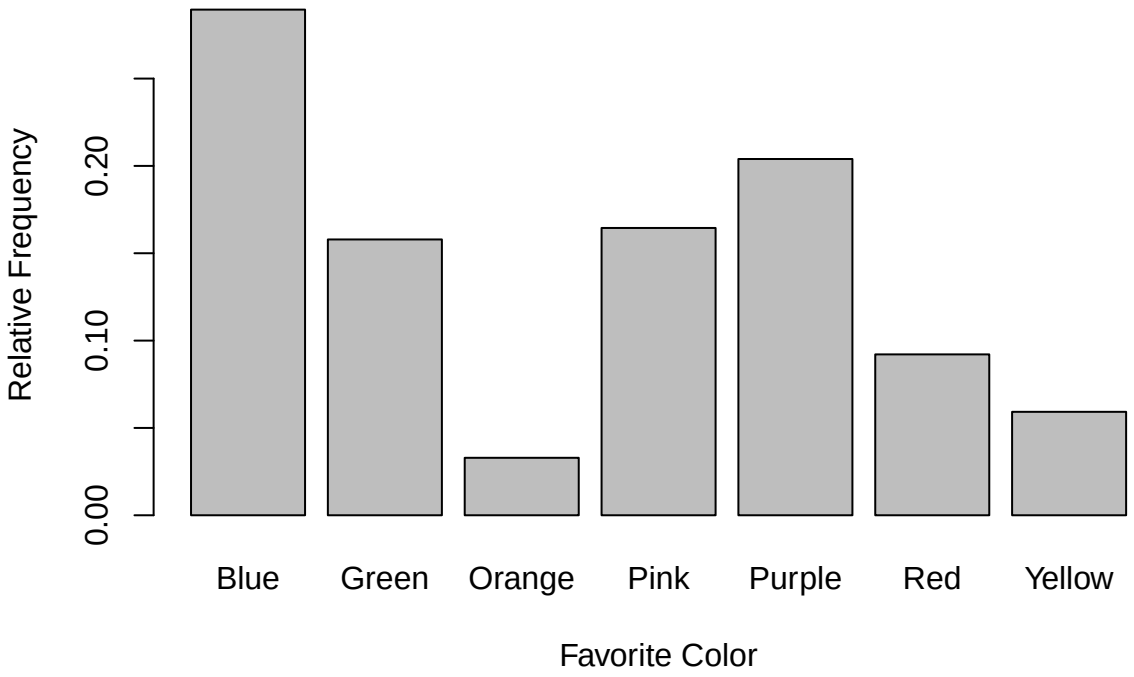


The y-axis is the number, or frequency, of students that prefer each color. For example, you can see that 31 students chose "Purple" as their favorite color (Go Huskies).

2.2 Relative frequency distributions

Sometimes you don't care about the actual frequency of things, but rather the proportion or percent of things that fall into each category. This is called a 'relative frequency distribution', or sometimes a 'probability distribution'. This is done by dividing the frequencies by the size of your sample (and multiplying by 100 if you want percent instead of proportion). For example, there are a total of 152 students in the class, so the proportion of students chose "Purple" as their favorite color is $\frac{31}{152} = 0.2039$. Here is the relative frequency distribution for favorite color:

```
color.relative.distribution = color.distribution/length(survey$color)
barplot(color.relative.distribution,xlab = "Favorite Color",ylab = "Relative Frequency")
```



The relative frequency distribution throws out the sample size, so it is often used as a way to generalize your sample to a larger population, or to compare the distributions of samples that have different sizes.

2.3 Probability distribution

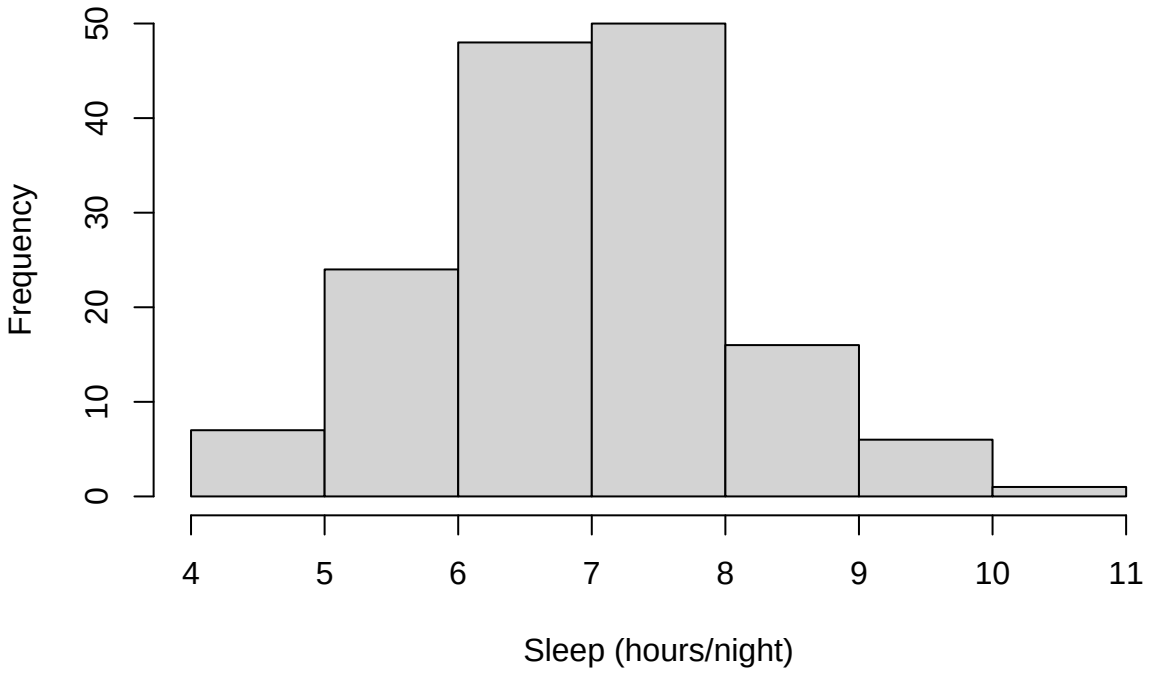
Since the relative frequency is the actual frequency divided by the sample size, the sum of all of the relative frequencies should add up to 1. This means that you can think of the relative frequencies as probabilities. For example, if your were to choose a student at random, the probability that that student chose "Purple" as their favorite color is 0.2039

This is why the relative frequency for discrete data is sometimes called the 'probability distribution'.

2.4 Continuous scale data

To plot the frequency distribution for continuous scale data we need to divide the continuous scale into discrete bins, called 'class intervals'. Plots of frequency distributions for continuous data are called 'histograms'. It's easy to make histograms in R using the 'hist' function. Here's a plot of the frequency distribution for the amount of sleep the students reported they get each night:

```
hist(survey$sleep,main = "",xlab = "Sleep (hours/night)",ylab = "Frequency")
```



Notice that the bars touch. This is a convention - for discrete data we often leave a gap between the bars, and with a continuous distribution (histogram) we have the bars touch.

2.5 Choosing class intervals

R 'hist' function chooses it's own set of class intervals based on some algorithm. It seems to chose around 8-1 intervals with widths and borders as integers (whole numbers).

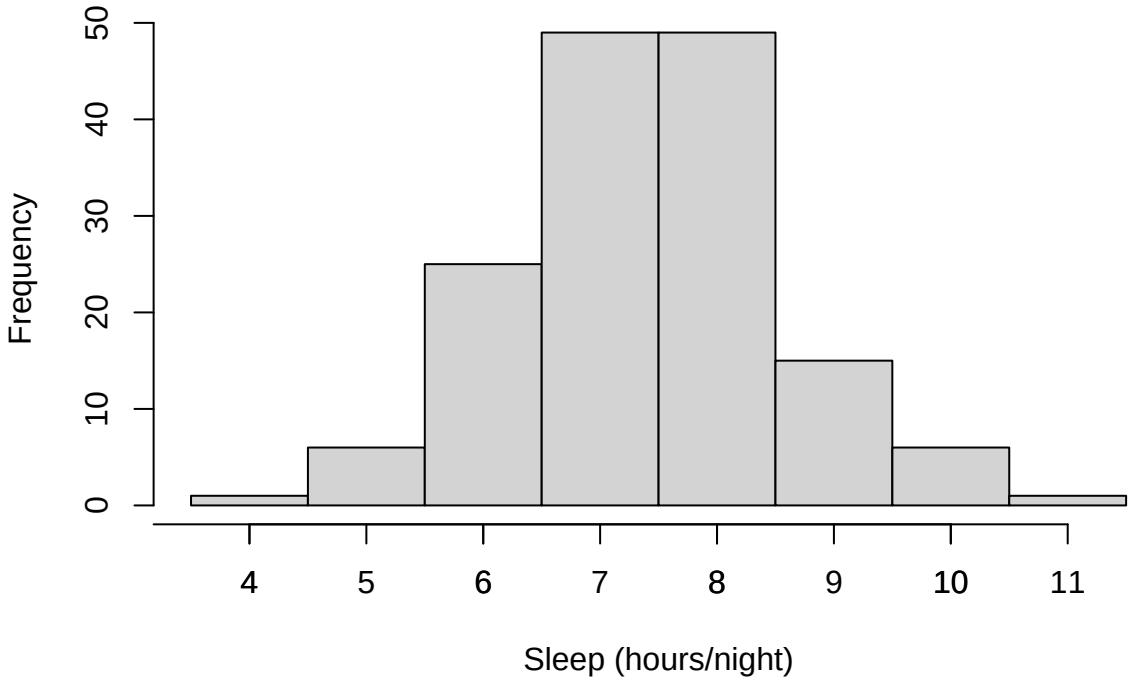
I tend to want control over these things. Next we'll discuss three factors that can help with your choice of class intervals: centering, the resolution of your data, and the size of your sample.

2.5.1 Centering the bars

Look at the histogram for sleep. Since the class interval borders are integers, it's hard to tell how many students got, say exactly 6 hours of sleep. The answer is 24 students. Since 6 lands exactly on the border, you can see that R decided to put those 24 students into the lower class interval (5-6 hours). Placing borderline scores on the lower class interval is the most common convention across software packages, by the way.

To avoid this ambiguity, we can set the class intervals to be centered on the integers using the 'breaks' option in 'hist'. Below there's a call to the function 'axis' which lets us modify the x-axis ticks to be steps of 1 hour (instead of 2)

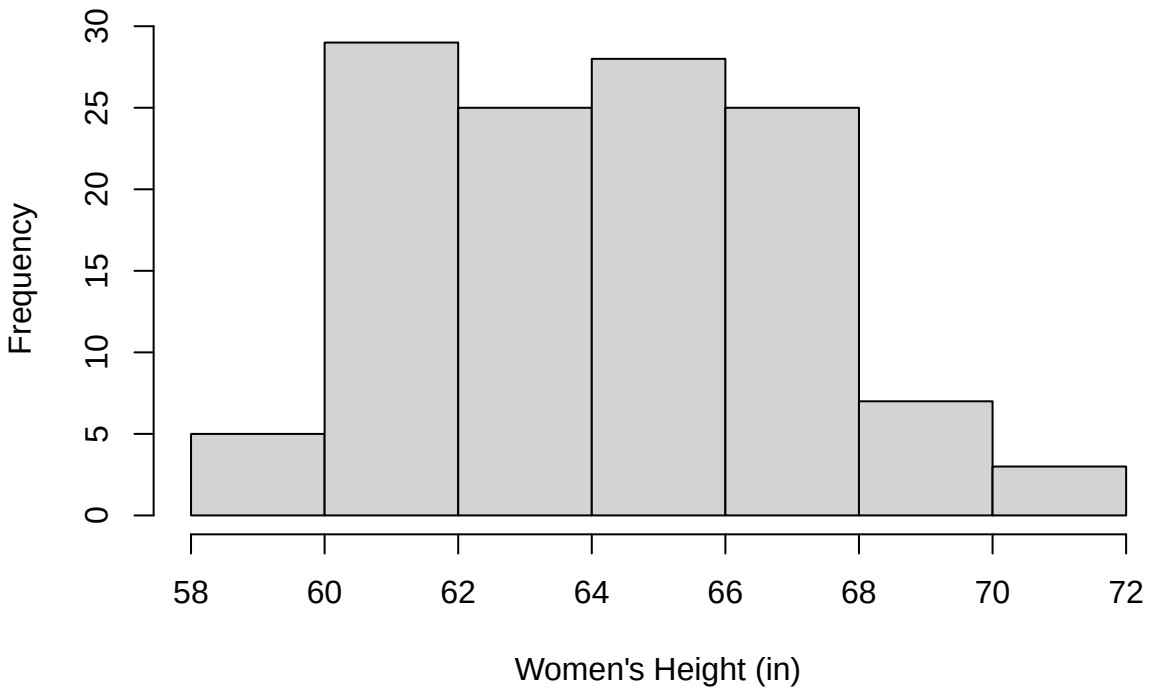
```
breaks <- seq(round(min(survey$sleep))-.5,round(max(survey$sleep))+.5)
hist(survey$sleep,main = "",breaks = breaks,xlab = "Sleep (hours/night)",ylab = "Frequency")
axis(side=1, at=seq(0,max(survey$sleep), 1))
```



2.5.2 Matching the resolution of your data

Consider R's choice for class intervals for plotting the heights of those who identified as women in the class:

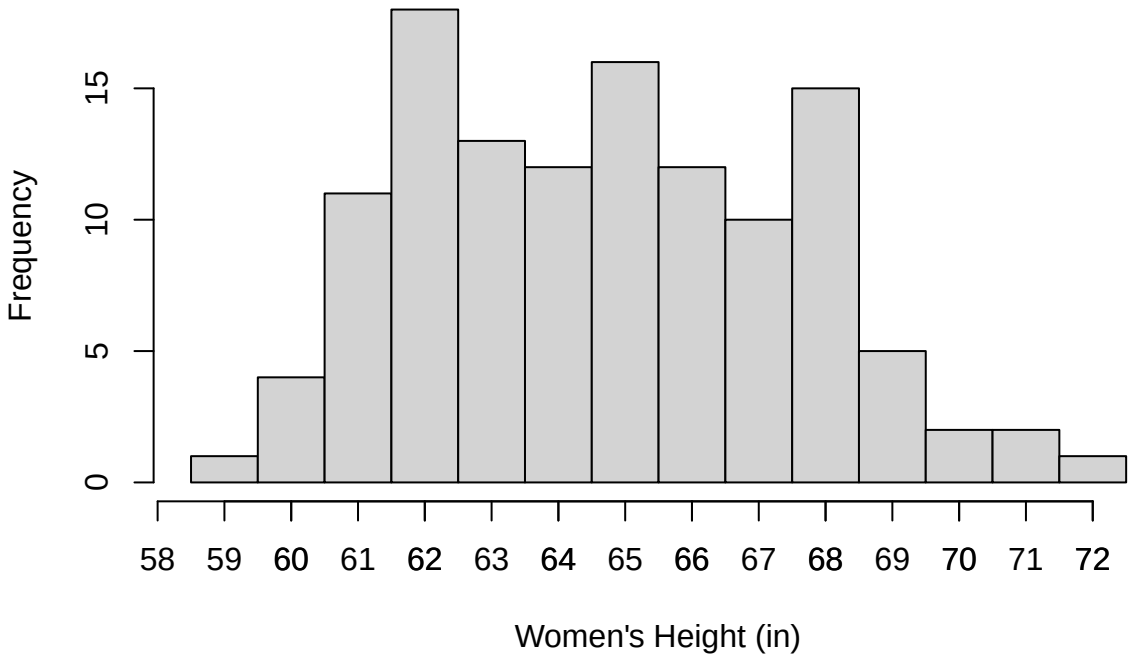
```
women.heights <- na.omit(survey$height[survey$gender=="Woman"])
hist(women.heights,main = "",xlab = "Women's Height (in)",ylab="Frequency")
```



In my opinion, R (and other programs) tend to choose class intervals that are too wide (2 inches here) so there aren't very many bars. Since the survey asked the students to report their heights to the nearest inch, it makes more sense to choose a class interval that matches the resolution of the data.

Here's the frequency distribution with one-inch intervals centered on the inch:

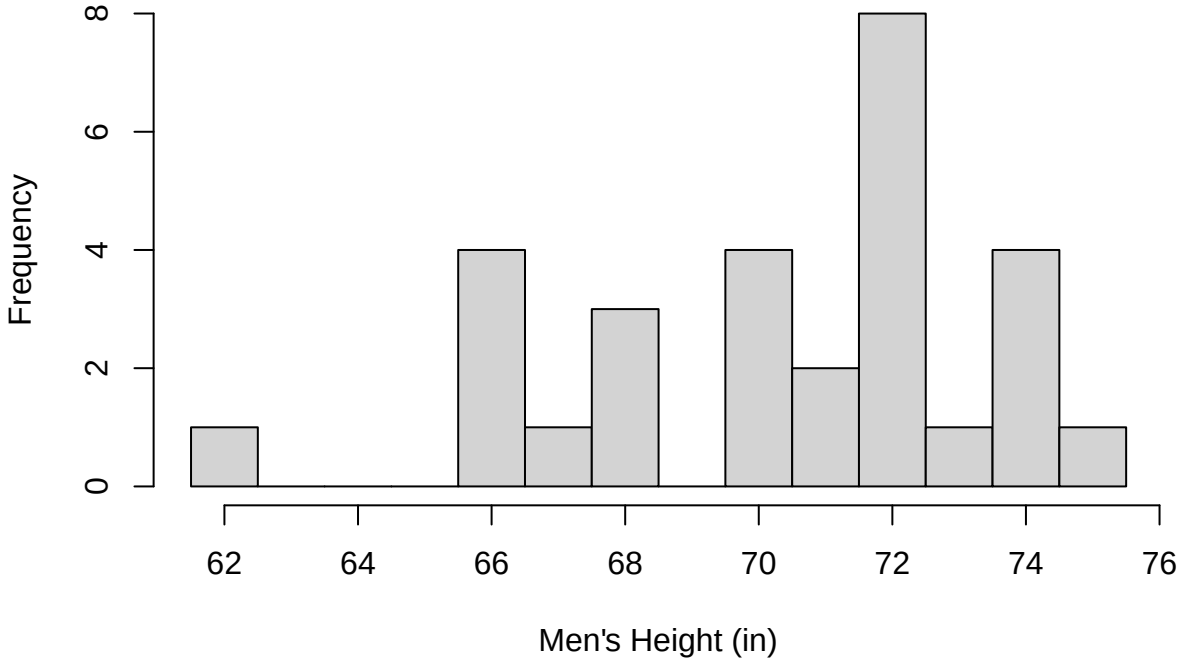
```
breaks <- seq(round(min(women.heights))-.5,round(max(women.heights))+.5)
hist(women.heights,breaks = breaks,main = "",xlab = "Women's Height (in)",ylab="Frequency")
axis(side=1, at=seq(min(women.heights),max(women.heights), 1))
```



2.5.3 Sample size

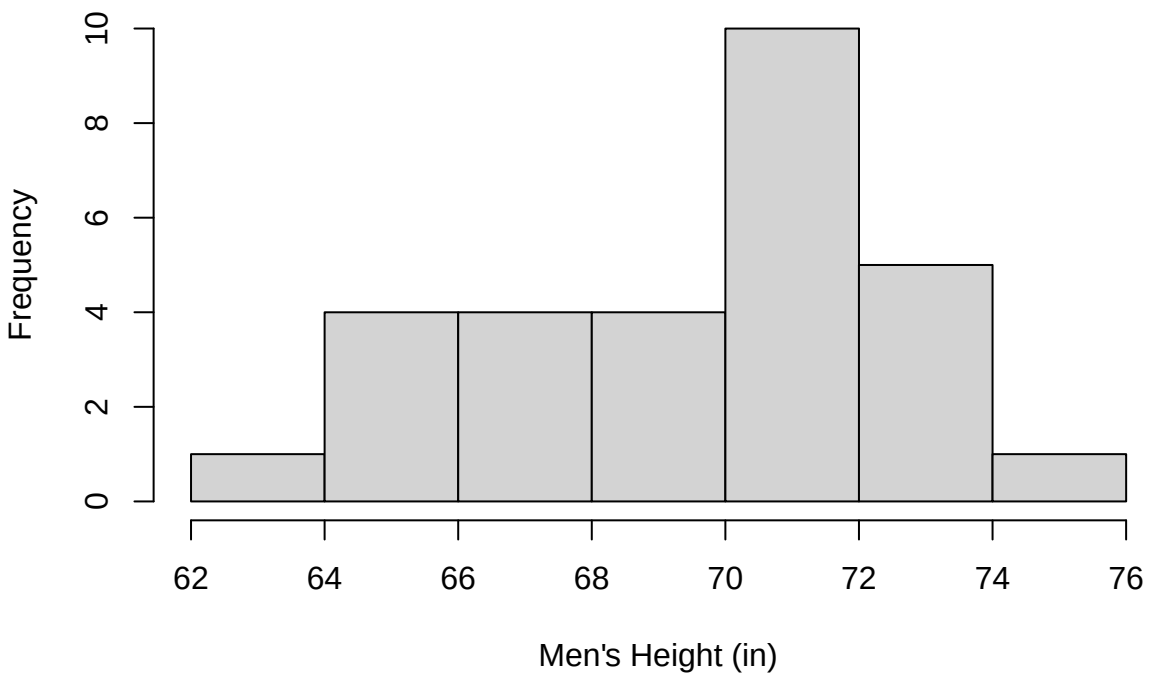
If you have a small sample size then you might have empty class intervals if your intervals are too narrow. For example, here's the frequency distribution for the 29 men in the class using one-inch wide intervals:

```
men.heights <- na.omit(survey$height[survey$gender=="Man"])
breaks <- seq(round(min(men.heights))-.5,round(max(men.heights))+.5)

hist(men.heights,main = "",breaks = breaks,xlab = "Men's Height (in)",ylab="Frequency")
```



Since there are fewer men than women we see gaps, or empty class intervals. In cases like this it makes sense to lower the number of intervals. With R's 'hist' function, if 'breaks' is set to a single number, then hist does it's best to use that number of class intervals:

```
breaks <- seq(round(min(men.heights)),round(max(men.heights)),2)
hist(men.heights,breaks = 8,main = "",xlab = "Men's Height (in)",ylab="Frequency")
```



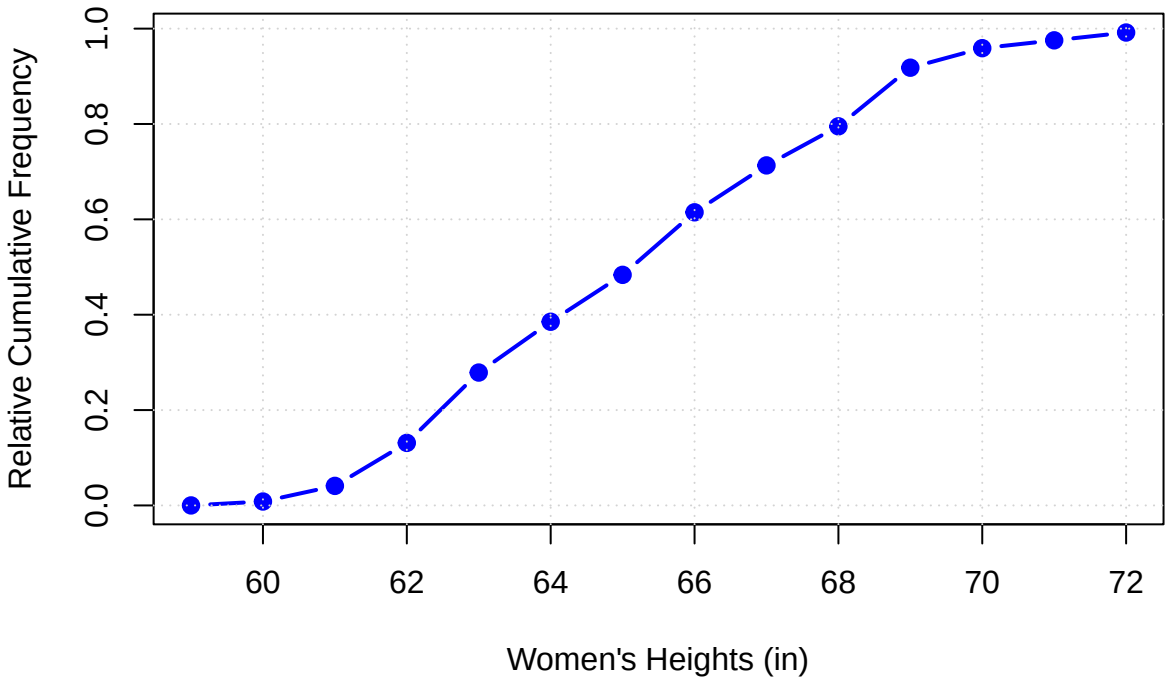
The choice of class intervals is a matter of taste, but it's good to realize that distributions look different to the eye with different class intervals. This emphasizes that frequency distributions are only a way of visualizing your data - they're a summary statistic. Please don't call these graphs 'data'!

2.6 Relative Cumulative Frequency

Frequency distribution plots are useful for visualizing where data lies along the distribution, but not as useful for estimating, say the median of the distribution or more generally the proportion of scores above and below some value.

Do to this we need the 'relative cumulative frequency' plot which is the proportion of samples that fall below a value as that value sweeps from the lowest to highest value in the sample. For example, from the survey, we can calculate and plot the cumulative frequency of women's heights like this:

```
breaks <- unique(sort(women.heights))
x.cut = cut(women.heights, breaks, right=FALSE)
x.freq = table(x.cut)

cumfreq0 = c(0, cumsum(x.freq))/length(women.heights)
plot(breaks,cumfreq0,type = "b",pch = 19, lwd = 2,col = "blue",xlab = "Women's Heights (in)",ylab = "Relat
grid()
```



Relative cumulative frequency plots are useful for eyeballing the proportion of heights above and below some value. From this plot we can see that, for example, you can see that 50 percent of the women are 65 inches or taller (or shorter). You can also see that the proportion of women that are shorter than 62 inches is about 0.15.

The shape - and specifically the slope - of the relative cumulative frequency plot tells you something about the shape of the distribution. The slope is flat in intervals where aren't many scores, and the slope is steep where the density of scores is high. The S-shaped cumulative frequency seen here for heights is the signature of a normal, or bell-shaped distribution. The curve is flat at the low and high ends, corresponding to the sparse number of samples in the tails, and the curve is steepest in the middle in the thick part of the bell-curve. You can see this by comparing this curve to the frequency distribution of woman's heights plotted earlier.

2.7 Percentile Points and Percentile Ranks

Rather then eyeballing these proportions, R has a couple of functions that calculate the proportion of scores below certain values. Back to our first example - what proportion of women's heights fall below 62 inches? We have specific names for these things. The height in this example (or more generally the 'score') is called the 'percentile point'. The corresponding proportion of scores below this percentile point is called the 'percentile rank'.

2.7.1 'Quantile': Percentile Ranks to Percentile Points

R's function 'quantile' give you percentile points from percentile ranks. For example, to find the height (percentile point) for which 50% (percentile rank) falls below is:

```
quantile(women.heights,.5,type = 5)
```

```
## 50%
##  65
```

Note the option 'type=5'. R allows for 9 different ways for computing percentile points! They're all very similar. Type 5 is the simplest and most commonly used.

If you want to calculate more than one percentile rank at a time, you can add a list of ranks using the 'c' command. Remember, 'c' allows you to concatenate a list of numbers together.

Let's generate the percentile points for the lowest 5%, first quartile, median, third quartile and highest 95%:

```
quantile(women.heights,c(.05,.25,.5,.75,.95),type = 5)
```

```
##  5% 25% 50% 75% 95%
##  61  62  65  67  69
```

2.7.2 Percentile Points to Percentile Ranks

For some reason, R doesn't have a good function for going from percentile points to percentile ranks in R. If you look around you'll be pointed to the 'ecdf' function ('Empirical Cumulative Distribution Function'). For our example converting a height (point) of 62 to rank is done like this:

```
ecdf(women.heights)(62)
```

```
## [1] 0.2786885
```

Notice that this doesn't agree with the cumulative frequency graph, where we estimated the rank to be about 0.15.

I don't know why this kind of wrong, and why there isn't a good inverse of the quantile function. Instead, I'm going to take a brief digression to show you how to find the inverse of a function like this using a procedure called 'binary search'.

2.7.3 Inverting the quantile function using 'Binary Search'

The binary search algorithm is an efficient way to find the inverse of a 'monotonically increasing' function - which is a function for which the output only increases with the input. 'quantile' is an example: as the percentile rank increases, the percentile point must also increase (or stay the same).

Let's use our example of finding the rank for a point of 62, that is the percentile rank for a height of 62 inches.

The trick behind the binary search is to start out with two values that we know will bracket the answer. For our example, we know that the percentile rank must be between zero and one. We start with finding the percentile point for the midpoint of our bracket which is a rank of 0.5. We already found that the corresponding percentile point is 65 inches. This is above the desired percentile point of 62 which means that our first guess of 0.5 is too high. The actual rank must be between 0 and .5, so we try the midpoint for the new bracket which is 0.25.

This procedure continues, dividing the bracket in half with either the upper and lower bounds being replaced by the midpoint of the bracket, depending on the percentile point at the midpoint. The bracket narrows quickly: after 20 iterations, the width of the bracket is only $(\frac{1}{2})^{20} = \frac{1}{1048576} = 0.000000953674$

Here's how to do it with a for loop in R:

```
desired_point <- 62  # desired percentile point
lo <- 0  # lower end of bracket
hi <- 1  # high end of bracket
for (i in 1:20) {
  mid <- (lo+hi)/2  # find the midpoint of the bracket
  # find the percentile point at the midpoint
  current_point <- quantile(women.heights,mid, type =5)
  if (current_point<desired_point) # if we're too low:
    {lo <- mid } # replace the lower end of the bracket with the midpoint
  else  # other wise
    {hi <- mid} # replace the higher end of the bracket with the midpoint
}
rank <- (lo+hi)/2  # split the bracket one more time.
rank
```

```
## [1] 0.1352458
```

This should be the rank for the percentile point of 62. Let's check:

```
quantile(women.heights,rank)
```

```
## 13.52458%
##        62
```

Chapter 3

Descriptive Statistics

This script will show you how to load in data from the Psych 315 survey and explore some of the data using basic descriptive statistics like measures of central tendency and variability, bar graphs and histograms.

First we'll clear the workspace and load in the survey data.
R's function `read.csv` loads in csv files and if there are headers in the first row uses the names in the headers to name the 'fields' that contain the data.

```
rm(list = ls())
survey <-read.csv("http://www.courses.washington.edu/psy315/datasets/Psych315W21survey.csv")
```

I've told `read.csv` to put all the data into a variable called 'survey'.
'survey' has a bunch of fields associated with it that correspond to your answers to each of the questions.

For example, I asked students about their heights (in inches). This can be found in the field 'height' and can be accessed = with the dollar sign:

```
survey$height
```

```
##   [1] 68 64 63 60 66 70 62 66 61 65 62 72 61 64 67 69 72 71 62 66 62 65 67 60 62
##  [26] 65 62 72 61 65 70 68 67 68 64 66 61 66 67 63 62 64 61 63 68 62 69 72 68 61
##  [51] 67 65 69 62 63 72 68 63 63 68 74 72 61 68 70 70 62 68 61 62 59 63 62 64 71
##  [76] 62 65 62 70 63 67 66 72 66 66 60 65 62 68 75 65 70 65 68 62 62 72 66 74 71
## [101] 68 68 67 64 67 63 65 66 66 65 66 60 67 64 64 62 61 68 66 64 64 67 67 62 65
## [126] 68 63 64 65 74 71 61 68 68 68 63 72 65 64 61 63 66 65 66 69 69 66 63 73 61
## [151] 65 74
```

To just look at the first few values, use `head`:

```
head(survey$height)
```

```
## [1] 68 64 63 60 66 70
```

How many students filled out the survey? This can be determined with the function `length` for any of the fields:

```
length(survey$height)
```

```
## [1] 152
```

Mother's heights are in the field 'mheight':

```
head(survey$mheight)
```

```
## [1] 64 62 61 60 61 60
```

These lists are in the same order across students, so the first student in the list (whoever it is) has a height of:

```
survey$height[1]
```

```
## [1] 68
```

and has a mother of height

```
survey$mheight[1]
```

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

Here are some basic statistics about your mother's heights:

mean:

```
mean(survey$mheight)
```

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

The answer might be 'NA' which means 'not available' That's because there is missing data in the list. Look back and you'll see that some of the entries are 'NA'.

To calculate the mean, while ignoring missing entries, use:

```
mean(survey$mheight,na.rm = TRUE)
```

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

Another way to deal with missing data is to create a new variable without the 'NA's.

To find the missing data, use 'is.na'

```
is.na(survey$mheight)
```

```
##   [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
##  [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [25] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [49] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [61] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [73] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [85] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [97] FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
## [109] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [121] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [133] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
## [145] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

gives you 'TRUE' for the locations of the missing data.

To find the data that's NOT missing, use '!' which switches TRUE to FALSE and vice versa:

```
!is.na(survey$mheight)
```

```
##   [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
##  [13]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [25]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [37]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [49]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [61]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [73]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [85]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [97]  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [109]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [121]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

```
## [133]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
## [145]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

Finally, we can use this list to pull out the non-missing data, and put it into a new variable 'mheight':

```
mheight <- survey$mheight[!is.na(survey$mheight)]
```

`mean` should now work:

```
mean(mheight)
```

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

The minimum mother's height with `min`:

```
min(mheight)
```

```
## [1] 51
```

and the maximum with `max`:

```
max(mheight)
```

```
## [1] 72
```

(sample) standard deviation with `sd`:

```
sd(mheight)
```

```
## [1] 2.976762
```

(sample) variance with `var`:

```
var(mheight)
```

```
## [1] 8.861111
```

`median`:

```
median(mheight)
```

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

or equivalently using `quantile`:

```
quantile(mheight,.5)
```

```
## 50%
##  64
```

Note the '50%' above the result. That's because `quantile` returns the result as a 'named number'. If you want just a regular number without the name, pass the named number through `as.numeric`:

```
as.numeric(quantile(mheight,.5))
```

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

The Semi-interquartile range can be calculated using `quantile`

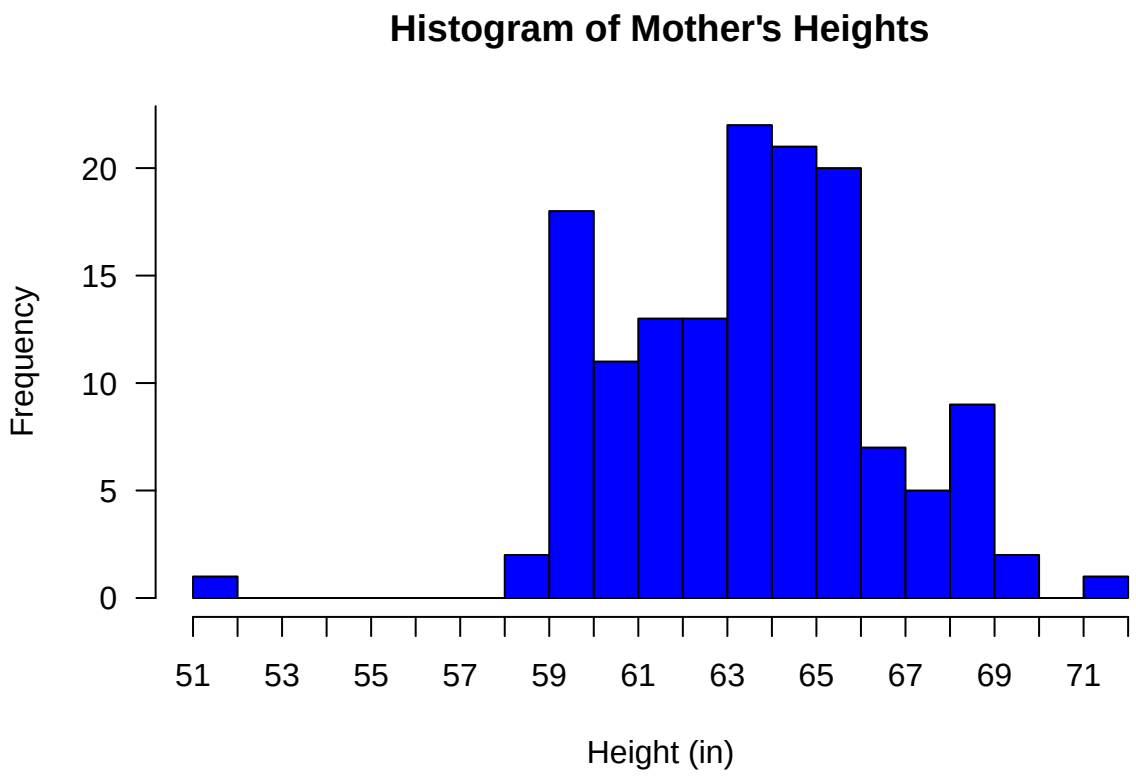
```
Q <- as.numeric(quantile(mheight,.75)-quantile(mheight,.25))/2
Q
```

```
## [1] 2
```

We can plot a histogram of mother's heights like this

```
# define class intervals based in the min and max:
class.interval <- seq(min(mheight),
                      max(mheight),
```

```
                      1)
hist(survey$mheight,
     main="Histogram of Mother's Heights",
     xlab="Height (in)",
     col="blue",
     xaxt='n',
     yaxt = 'n',
     breaks =class.interval
)

# and then adding your own axes with the 'axis' function
# Axis 1 is 'x' and 2 is 'y':
axis(1, at=class.interval)
axis(2, at=seq(0,100,5),las = 1)
```



What is ratio of men to women in this class? Remember, we can ask which students identified as "Woman" with:

```
survey$gender == "Woman"
```

```
##   [1] FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
##  [13]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [25]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [37]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE
##  [49]  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE
##  [61] FALSE  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [73] FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE
##  [85] FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
##  [97] FALSE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [109]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE
```

```
## [121]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE
## [133]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [145]  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE FALSE
```

The sum function will give you the number of TRUE's:

```
number.women <- sum(survey$gender == "Woman")
number.women
```

```
## [1] 122
```

Similarly for the men:

```
number.men <- sum(survey$gender == "Man")
number.men
```

```
## [1] 29
```

The total number of students is:

```
total.number <- length(survey$gender)
total.number
```

```
## [1] 152
```

And for women:

```
100*number.women/total.number
```

```
## [1] 80.26316
```

The percent of men is:

```
100*number.men/total.number
```

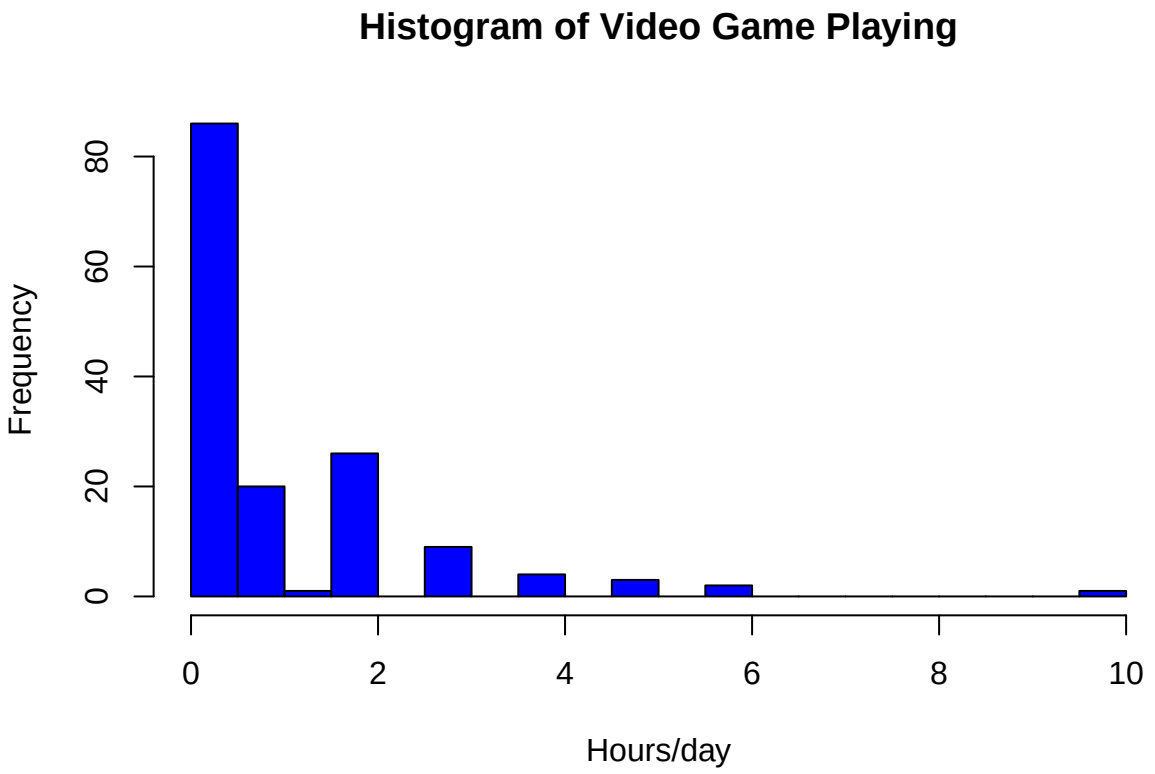
```
## [1] 19.07895
```

Note: these may not add up to 100%. This will happen if some students choose not to answer that question in the survey.

Try this on your own: Calculate the percent of left vs. right handers in the class.

How much do you all play video games? That's in the field 'games_hours'

```
hist(survey$games_hours,
     main="Histogram of Video Game Playing",
     xlab="Hours/day",
     col="blue",
     breaks =seq(0,max(survey$games_hours),.5)
)
```



Does this distribution look normal? If not is it positively or negatively skewed?

Does video game playing differ by gender?

```
mean(survey$games_hours[survey$gender == "Man"])
```

```
## [1] 1.568966
```

```
mean(survey$games_hours[survey$gender == "Woman"])
```

```
## [1] 0.8827869
```

Later on we'll see if this difference is 'statistically significant' using a 'paired sample t-test'.

What is the distribution of your favorite colors? You'd think we could type hist(survey$color)' but that doesn't work because `hist` needs a list of numbers, not a list of nominal category names.

Fortunately, R has a convenient function `table` that tabulates nominal scale data into frequencies:

```
color.freqs <- table(survey$color)
color.freqs
```

```
##
##   Blue  Green Orange   Pink Purple    Red Yellow
##     44     24      5     25     31     14      9
```

```
color.freqs
```

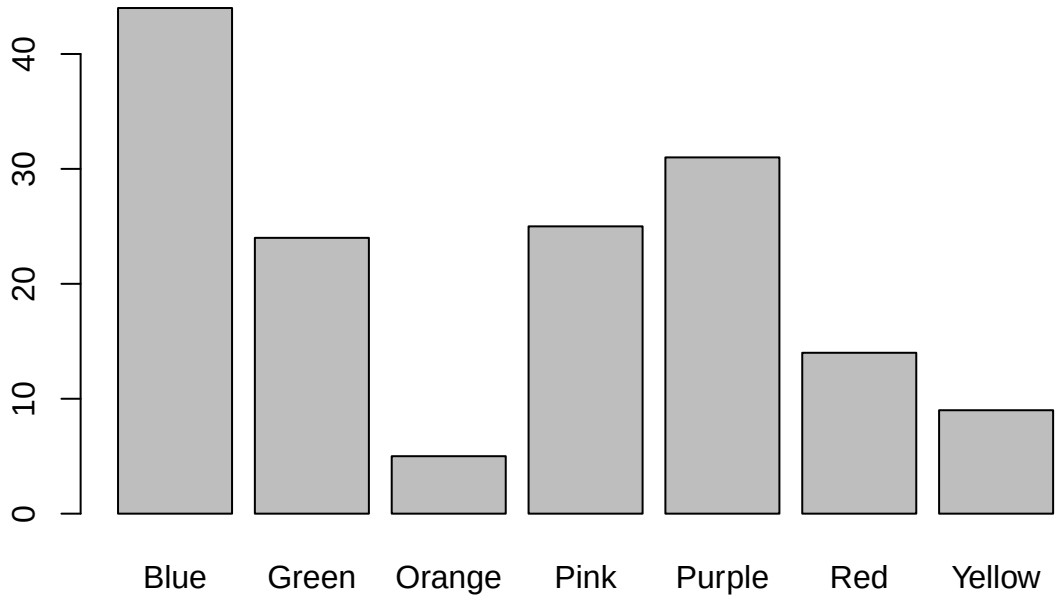
```
##
##   Blue  Green Orange   Pink Purple    Red Yellow
##     44     24      5     25     31     14      9
```

This variable color.freqs is something called a 'table'. It's like a list of numbers, except that the columns have names associated with them. In our case, the names of the columns are the color names.

Tables are convenient because they let you keep track of what the numbers mean.

You can then send this table into the function `barplot` to make a histogram:

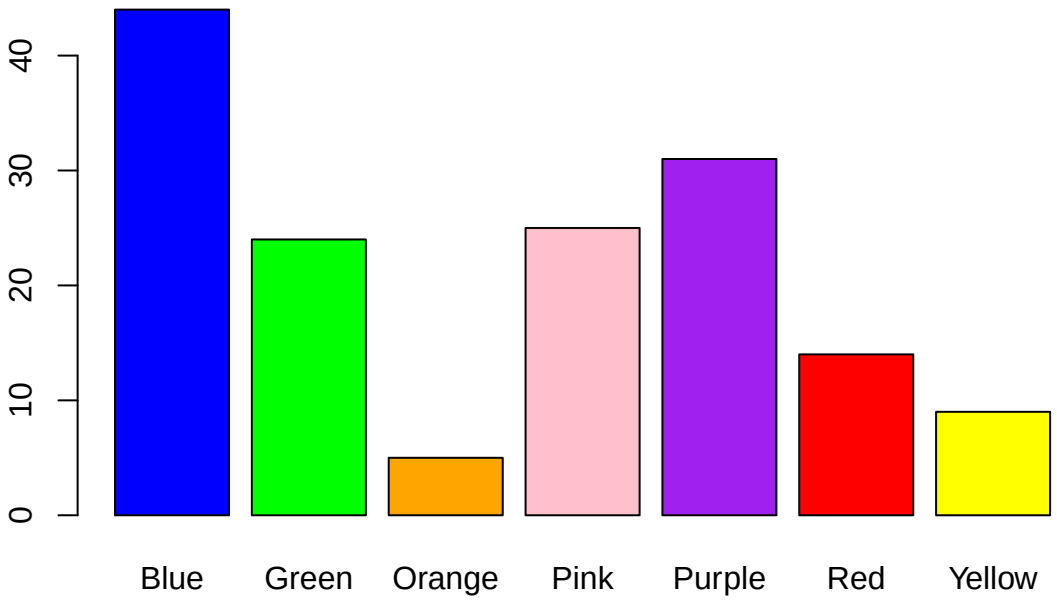
```r
barplot(color.freqs)
```



Got to love your love of 'purple'.

Want to get fancy? Let's use the option `col` to color the bars by their color names:

```r
barplot(color.freqs,
        col = c("blue","green","orange","pink","purple","red","yellow") )
```



Let's compare the frequency distribution of favorite colors by gender. First we need to create a table frequency distributions for each gender separately, using the 'table' function like before. But this time we'll index into the values for the associated gender:

```
color.freqs.men <- table(survey$color[survey$gender == "Man"])
color.freqs.women <- table(survey$color[survey$gender == "Woman"])
```

`table` will also create 2 (or more) dimensional tables by adding in each factor. Here's a table of preferred color for all genders:

Next we'll combine these two tables using the 'rbind' function which concatenates rows into a new table:

```
color.freqs.both <- table(survey$gender,survey$color)
color.freqs.both
```

```
##
##          Blue Green Orange Pink Purple Red Yellow
##             1     0      0    0      0   0      0
##   Man      11     5      0    0      6   6      1
##   Woman    32    19      5   25     25   8      8
```

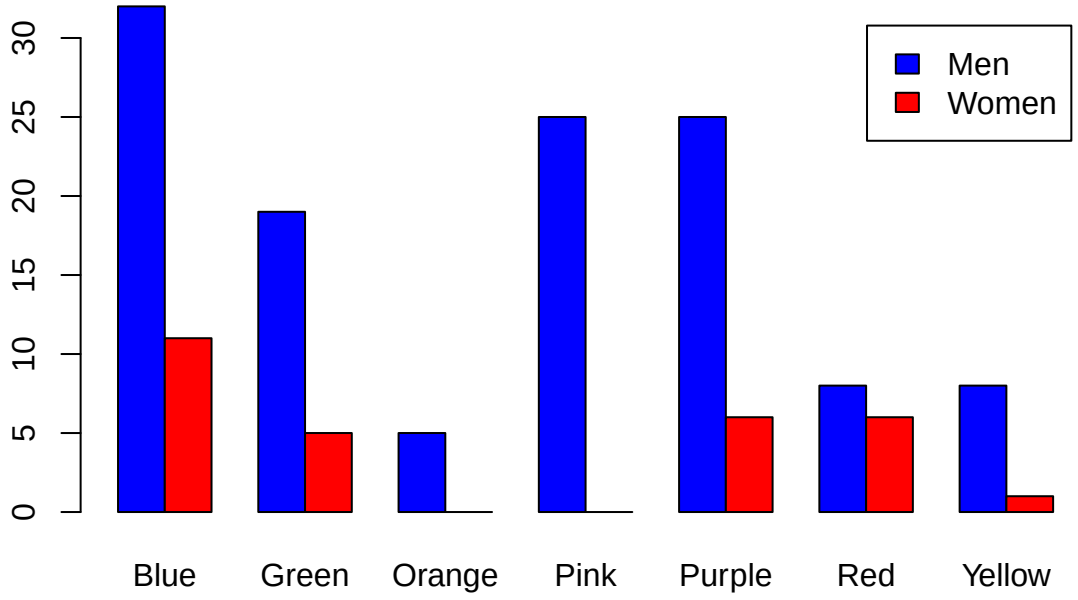
You can see that it contains a matrix of numbers along with names for the rows and columns.

This year, there were students that chose not to enter a gender. If we want to exclude this data from the table, we can pull out only the "Man" and "Woman" rows in the table (ignoring the thorny issue of only including students that provided a gender identity):

```
color.freqs.both <- color.freqs.both[c("Woman","Man"),]
```

This table is ready to be plotted using `barplot`. We'll use the option `beside = TRUE` so the "Men" and "Women" bars are plotted next to each other instead of on top of each other.

```
barplot(color.freqs.both,
        beside = TRUE,
        legend = c("Men","Women"),
        col = c("Blue","Red"))
```

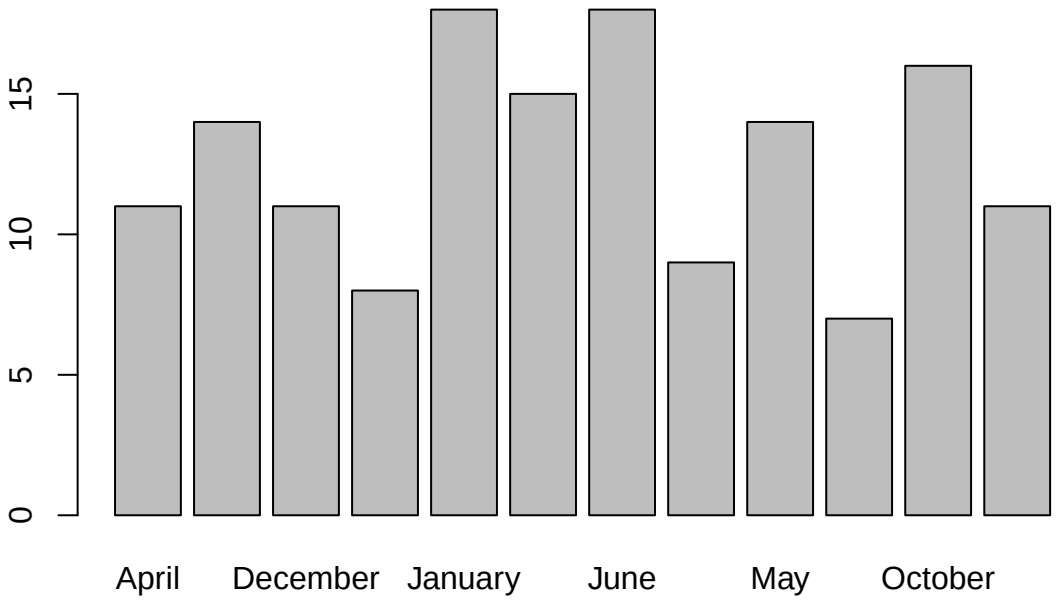


Do you think there is a difference in the distribution of color preference across gender? Later on we'll determine if these two distributions are significantly different from each other using a 'Chi-squared test for independence'.

As an exercise, see if you can create a histogram of the number of birthdays for each month The months you all were born in is in survey$month:

As before, we can make a table and plot it as a bar plot:

```
month.table <- table(survey$month)
barplot(month.table)
```



By default, table organizes the categories in alphabetical order. That's not what we want. Look at the table:

```
month.table
```

```
##
##     April    August  December  February   January      July      June     March
##        11        14        11         8        18        15        18         9
##       May  November   October September
##        14         7        16        11
```

You can see it's in alphabetical. To reorder the table, we can re-index the values by noting January, came 5th, Feburary came 4th, and so on.

To rearrange the order we can do this:

```
month.table <- month.table[c(5,4,8,1,9,7,6,2,12,11,10,3)]
```

Where the list 5,4,8,... are the locations for January, February, March, etc.

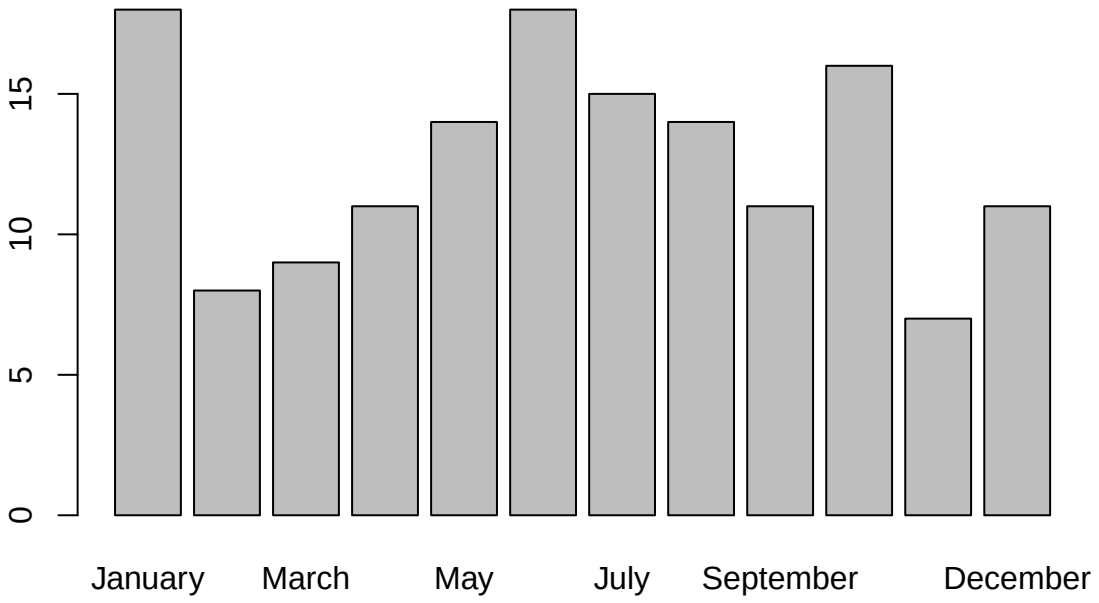
Now look:

```
month.table
```

```
##
##   January  February     March     April       May      June      July    August
##        18         8         9        11        14        18        15        14
## September   October  November  December
##        11        16         7        11
```

It's in the proper order. Now the bar plot will look right:

```
barplot(month.table)
```



There are more students born in some months than others. Do you think this happened just by chance? Or is this distribution particularly unsual? Later on we'll run a 'Chi-Squared' test to determine how likely we'd get a data set like this by chance.

Finally, let's find the average amount of sleep for the women that like to sit near the front of the class:

To find these students we need to find those for which BOTH their gender is Woman AND their favorite color is 'Purple'. This requires us to use `&` (and) to compare lists of TRUE and FALSE

`&` compares two TRUE/FALSE variables and returns 'TRUE' if both are TRUE

```
TRUE & TRUE
```

```
## [1] TRUE
```

```
TRUE & FALSE
```

```
## [1] FALSE
```

```
FALSE & TRUE
```

```
## [1] FALSE
```

```
FALSE & FALSE
```

```
## [1] FALSE
```

Here's how to use `&` to find the women that like to sit near the front of the class:

```
students.gender.front <- survey$gender== 'Woman' & survey$sit == 'Near the front'
```

And here's the average amount of sleep they get each night:

```
mean(survey$sleep[students.gender.front],na.rm = TRUE)
```

```
## [1] 7.204545
```

We use | for 'or'. This returns TRUE if either are TRUE:

```
TRUE | TRUE
```

```
## [1] TRUE
```

```
TRUE | FALSE
```

```
## [1] TRUE
```

```
FALSE | TRUE
```

```
## [1] TRUE
```

```
FALSE | FALSE
```

```
## [1] FALSE
```

Let's use | to find the favorite color for students that were either born in January or are left-handed:

```
students.january.left = survey$month == "January" | survey$hand == "Left"
survey$color[students.january.left]
```

```
##  [1] "Blue"   "Blue"   "Purple" "Blue"   "Green"  "Purple" "Green"  "Red"
##  [9] "Red"    "Purple" "Pink"   "Blue"   "Red"    "Red"    "Green"  "Purple"
## [17] "Pink"   "Green"  "Orange" "Purple" "Green"  "Blue"   "Pink"   "Blue"
```

Chapter 4

The Normal PDF

4.1 Sampling from the standard normal distribution

The normal distribution - the familiar bell-shaped distribution - shows up all over the place. In the next chapter on the Central Limit Theorem we'll see why this is the case but for now just realize that it's a fundamental part of statistics. R has a set of functions that let you sample from and work with the normal distribution. We'll start with `rnorm` (*r* for random, *norm* for normal) which draws random numbers from the normal distribution.

The following code generates a huge sample from a normal distribution with mean 0 and standard deviation of 1, which is called the *standard normal*:

```
n <- 10000    # huge sample size
set.seed(1)
x <- rnorm(n)
```

The command `set.seed` sets the random number generator seed so that the following sequence will always be the same. This way my numbers will look like your numbers if you run this code.

With this huge sample size, hopefully the mean and standard deviation will be close to 0 and 1 respectively:

```
mean(x)
```

```
## [1] -0.006537039
```

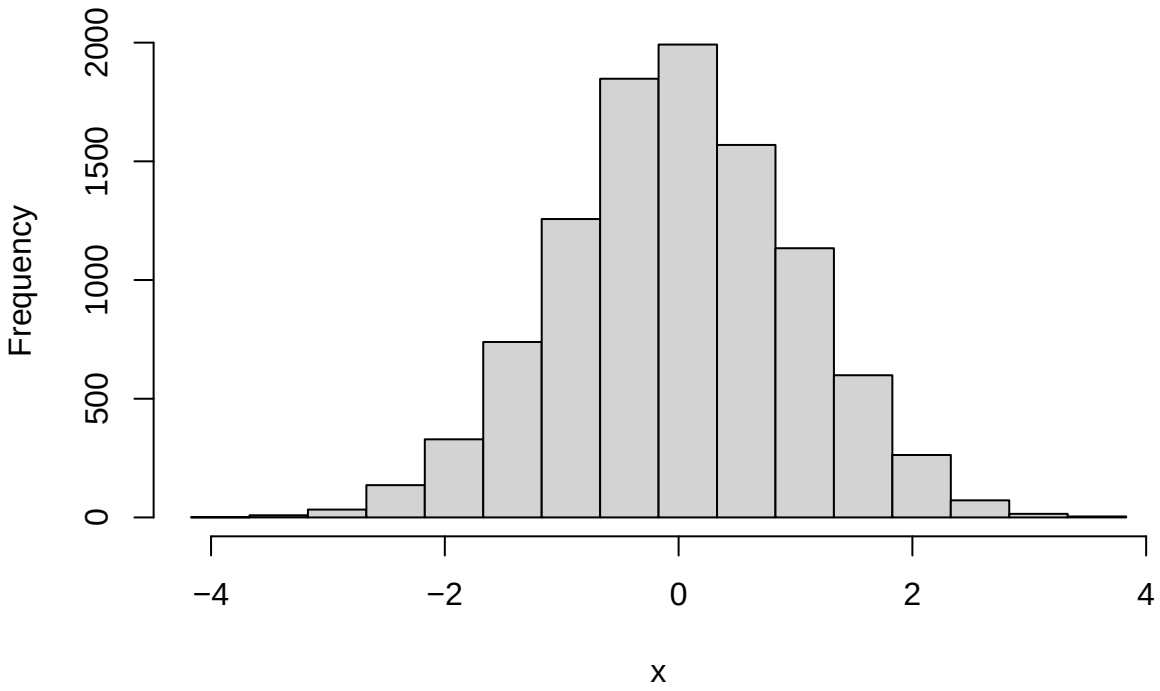
```
sd(x)
```

```
## [1] 1.012356
```

Good.

To visualize the distribution of this sample we can plot a histogram after choosing the class intervals. Here's how we did it in the last chapter on frequency distributions:

```
x.range <- round(max(abs(x)),.5)+.5
breaks = seq(min(x)-.5,max(x)+.5,.5)
hist(x,main = "",breaks = breaks,xlab = "x",ylab = "Frequency")
```

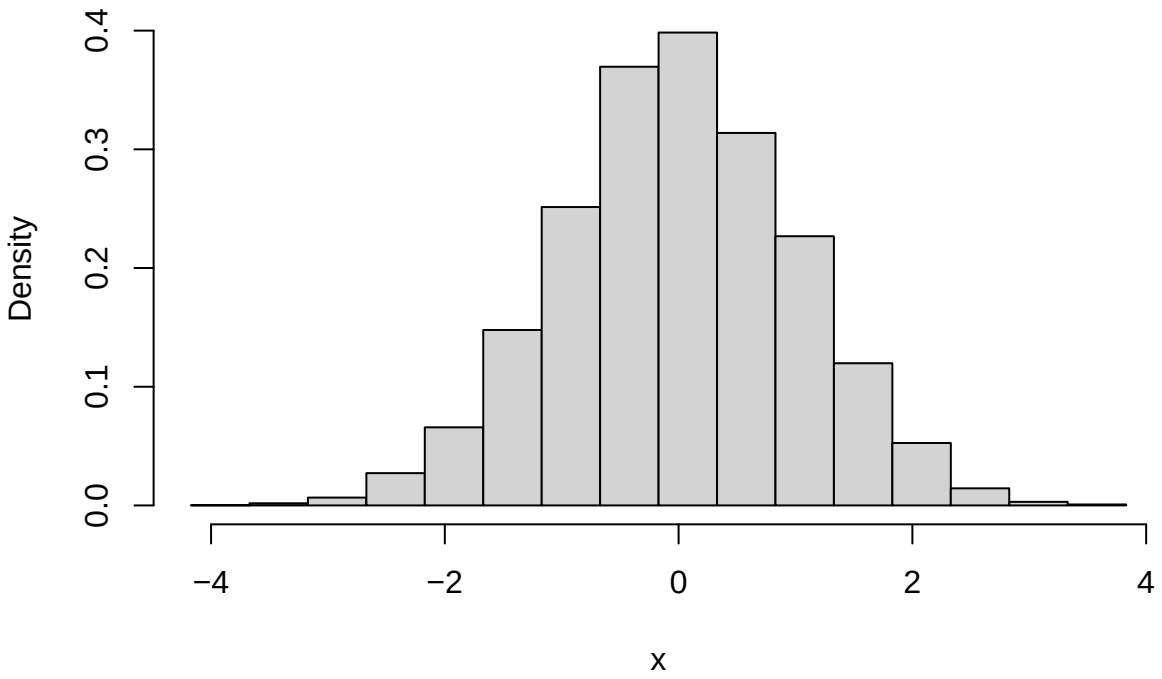


4.2 Density - the equivalent of relative frequency for continuous data

For discrete data, we calculated the relative frequency distribution by dividing by the sample size. If we do that here with continuous data we have the problem that we picked the class intervals ourselves. So, for example, doubling the width would roughly double the frequencies. To fix this, we divide the frequencies by *both* the sample size but then multiply by the interval width.

The analogue of the relative frequency for continuous data is called the 'density', and the graph is called the 'probability density function', or 'pdf'. You can plot the density instead of the frequency using 'hist' by setting 'freq = FALSE':

```
hist(x,main = "",breaks = breaks,xlab = "x",ylab = "Density",freq = FALSE)
```



With such a large sample size, we can use a very small class interval with to get a better look at the distribution:

```
breaks = seq(-x.range,x.range,.1)
hist(x,main = "",breaks = breaks,xlab = "x",ylab = "Density",freq = FALSE)
```

Notice that even though the intervals are narrow, the height of the graph still peaks at around the same height of about 0.4. This is because the calculation of density takes into account the width of the intervals.

Remember, the relative frequency distribution for discrete data tells you the probability that a given sample will fall into a given category. There is a related relation between density and probability for the pdf: Since the heights of the bars are scaled by their width, the *area*, and not the height, of each bar represents the probability of drawing a sample from that class interval.

The useful thing about this is that you can use the pdf to find the probability of sampling a value within a certain range of class intervals we just sum the areas of the bars that cover that range.

Let's use this to calculate the probability of obtaining a sample between the values of 0 and 0.5.

In the latest plot, the class interval width in the plot above was set to 0.1. There are 4 intervals that cover the range, and their heights add up to $0.395 + 0.381 + 0.371 + 0.333 = 1.48$.

Multiplying this sum by the class interval width gives us $(1.48)(0.1) = 0.148$

This means that about 15% of the scores fall between 0 and 0.5.

4.3 The normal pdf

Since we drew samples from the normal distribution, the pdf looks like the familiar bell-curve. With a finite sample we have to use a finite number of class intervals. We could use narrow class intervals because our sample size is large. In theory, with a big enough sample size we could get a smooth-looking pdf. This is a calculus thing: the sample sizes go to infinity but the widths go to zero. In the end, you get a smooth, continuous pdf that is the probability density function of the population that you're drawing from.

Since it's a smooth curve, we can describe it as a function of x. The standard normal distribution has a pdf of:

$$y = \frac{e^{-\frac{x^2}{2}}}{\sqrt{2\pi}}$$

Where e is the basis of the natural logarithm (one of my favorite numbers). Like π, it has a non-repeating decimal that goes on forever. You can use R's 'exp' function to see e to the first 80 decimal places:

```
sprintf('%2.50f',exp(1))
```

```
## [1] "2.71828182845904509079559829842764884233474731445312"
```

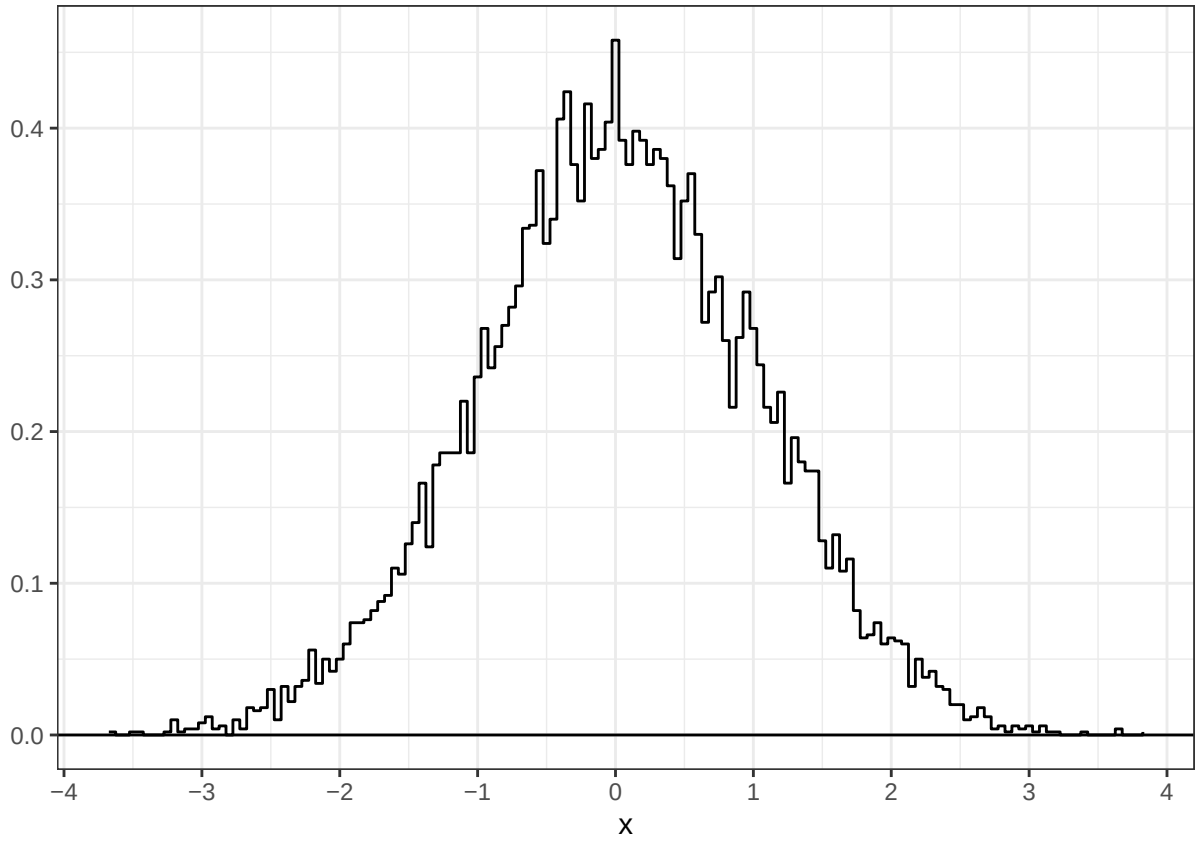
Sorry for the digression.

Let's see how well the pdf for the standard normal fits the distribution of our sample by overlaying it on the histogram:

We'll start plotting pdfs using the 'ggplot' function from the 'ggplot2' library. ggplot takes a little getting used to, but it's a very flexible function that allows us to do things like overlay plots.

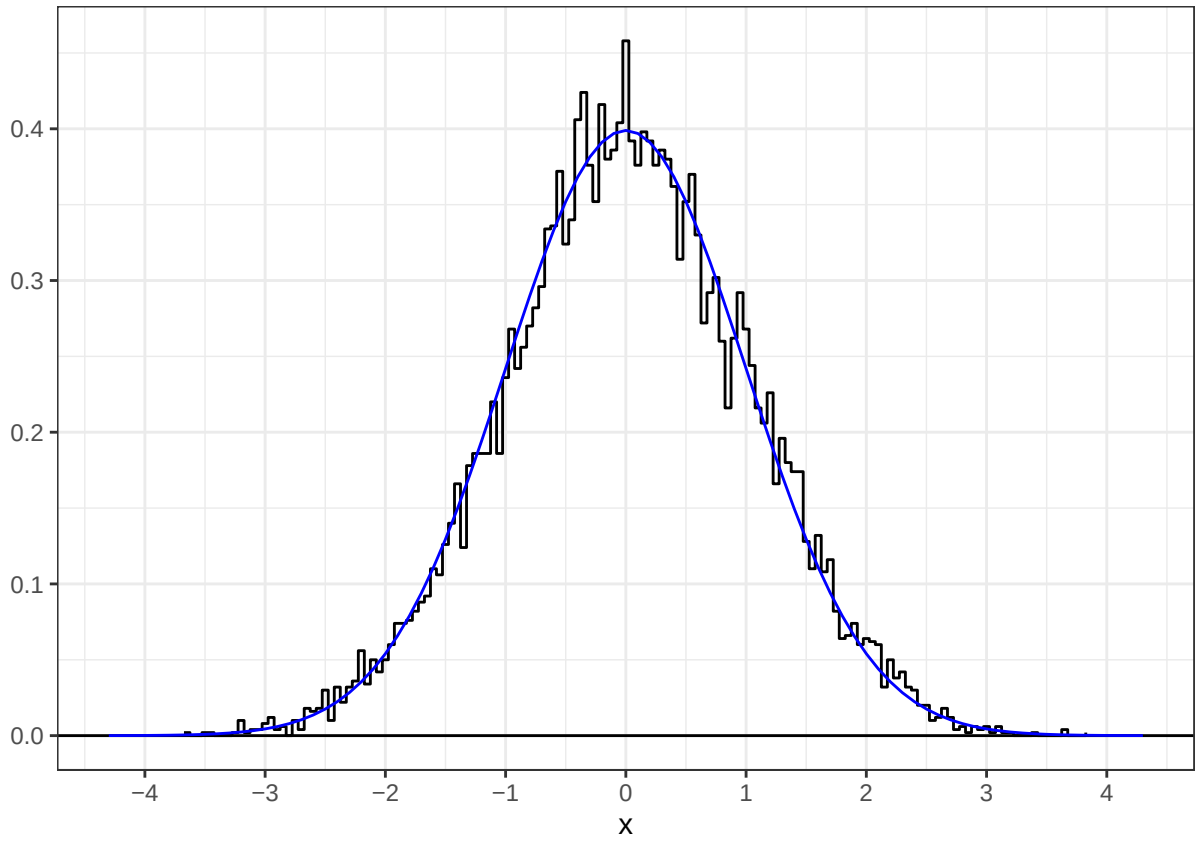
The following code first runs the `'hist'` function with `plot=FALSE` which returns the frequencies without plotting. 'plot' also returns the $density. We then use `ggplot` to plot the density. We'll use the `geom_step()` option which plots the bars without the vertical lines at the class intervals. This is cleaner for plots with lots of narrow class intervals. With a large sample size like this we can use narrow class intervals.

```
x.distribution <- hist(x,plot = FALSE,breaks = 200 )
big.data <- data.frame(x=x.distribution$mids,y=x.distribution$density)
p <- ggplot(big.data,aes(x=x,y=y)) + geom_step() + theme_bw() +
  scale_x_continuous(breaks = seq(-4,4)) +
  geom_hline(yintercept = 0) +
  ylab("")
p
```



R has it's own function, `dnorm`, that calculates the pdf as a function of x. Here's the smooth pdf for the standard normal on top of the sample pdf:

```
y <- dnorm(breaks,0,1)
norm.pdf <- data.frame(x=breaks,y=y)
p + geom_line(data =norm.pdf,aes(x=x,y=y),color = 'blue')
```



It's a nice fit, which demonstrates that R's random number generator is working well.

4.4 Probabilities for the standard normal

Given this smooth pdf, we can use the same trick to find probabilities that scores will fall within certain intervals by calculating the area under the pdf - which is integration in calculus. Frustratingly, although there is that function for the pdf, there is no closed-form integral for that function. But it doesn't really matter because there are algorithms for approximating the integrals to any degree of precision. R has the `pnorm` function for this. pnorm finds the area from minus infinity to a desired area. This result should make sense:

```
pnorm(0)
```

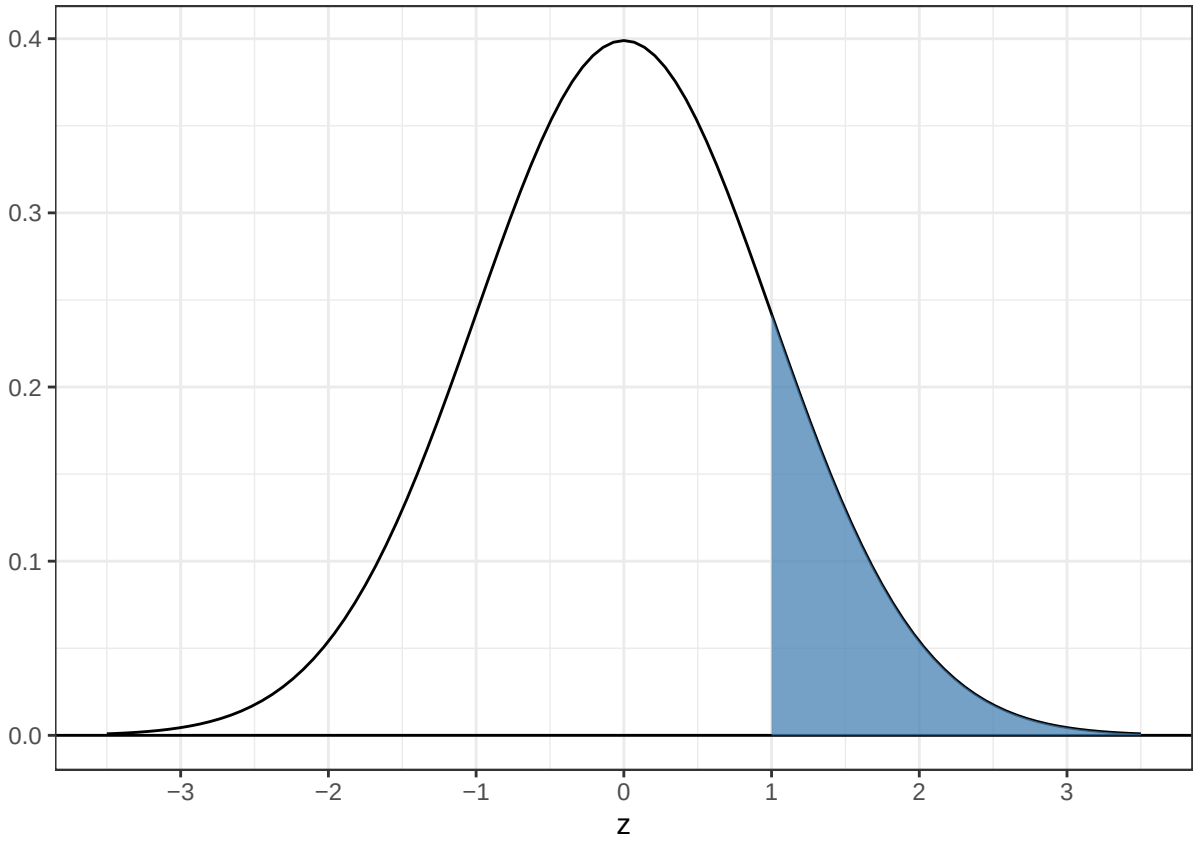
```
## [1] 0.5
```

Since the standard normal is symmetric around zero, half of the area falls below zero.

pnorm is all we need to find the area - and therefore the probability - between two values.

4.4.1 Example 1: Find the area above z=1 for the standard normal

We want to find the area of the shaded region:



The trick here is to find the area below 1 and subtract this from 1 since the total area is equal to 1:

```
1-pnorm(1)
```

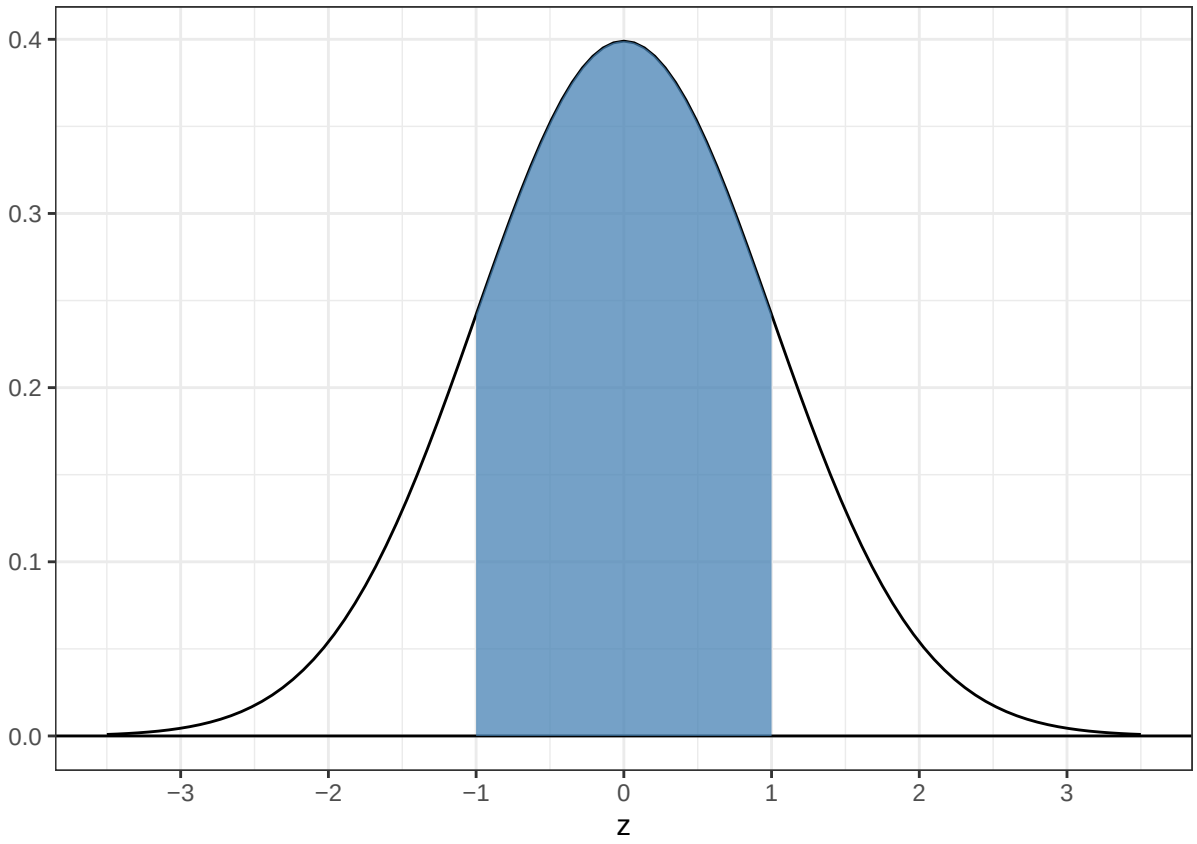
```
## [1] 0.1586553
```

You can get the same answer using the 'lower.tail' option which is 'TRUE' by default. If set lower.tail='FALSE' it finds the area above:

```
pnorm(1,lower.tail = FALSE)
```

```
## [1] 0.1586553
```

4.4.2 Example 2: Find the area between -1 and 1



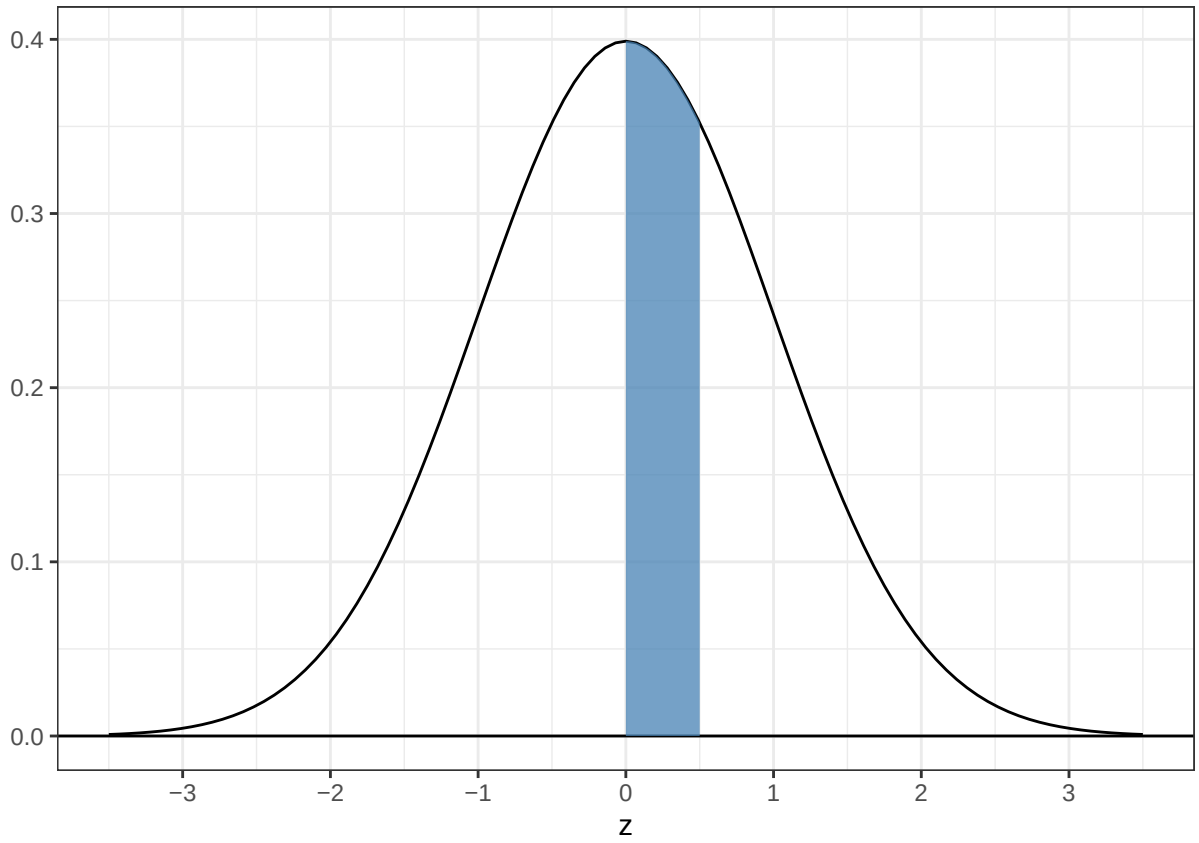
The trick here is to subtract the area below -1 from the area below 1:

```
pnorm(1)-pnorm(-1)
```

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

About 2/3 of the total area for the normal distribution falls within ± 1 standard deviation from the mean.

4.4.3 Example 3: Find the area between 0 and 1/2



Just like the last one:

```
pnorm(.5)-pnorm(0)
```

```
## [1] 0.1914625
```

Compare this number to 0.148, which is what we got when we added up the areas under the pdf from the sample. It's pretty close. It's not exact because our sample, although huge, is finite in size, so we didn't get the exact number of samples between 0 and 0.5 that's expected from the actual normal pdf.

4.4.4 Example 4: Find the z-score for which the area below is 0.75

This is the reverse of the previous examples: I gave you an area and you have to find the score. R does this with the `qnorm` function, which is the inverse of the `pnorm` function:

```
qnorm(.75)
```

```
## [1] 0.6744898
```

So about 67% of the area under the standard normal falls below .75. We can verify this with pnorm:

```
pnorm(0.6744898)
```

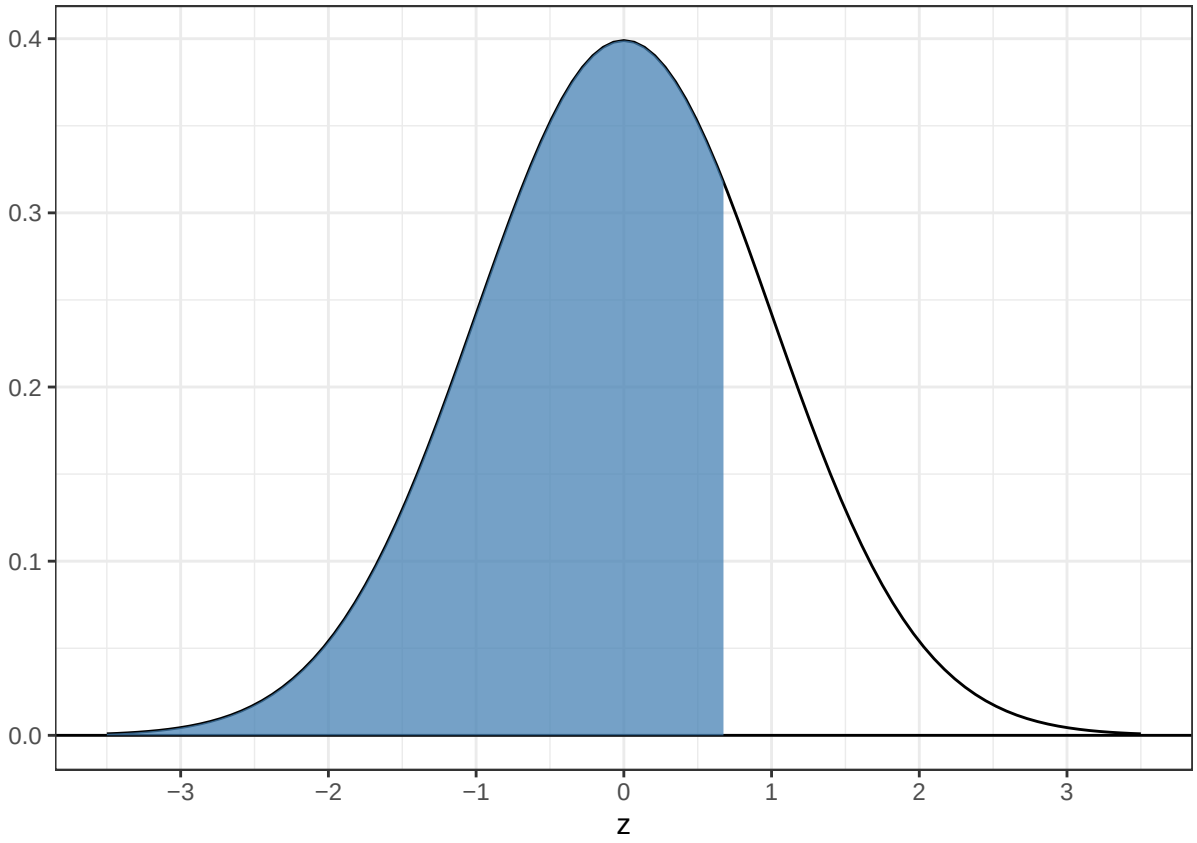
```
## [1] 0.75
```

Or to get fancy:

```
pnorm(qnorm(.75))
```

```
## [1] 0.75
```

The shaded region has an area of 0.75:



4.4.5 Example 5: What z-scores bracket the middle 95% of the area under the standard normal?

This is a fun one. If the middle contains 95% of the area, then the two tails each contain a total of 5% of the area. Splitting each tail in half means that each tails contains a proportion of $\frac{1-.95}{2} = 0.025$ of the area. The score for which the proportion of the area below is .025 is:

```
qnorm(.025)
```

```
## [1] -1.959964
```

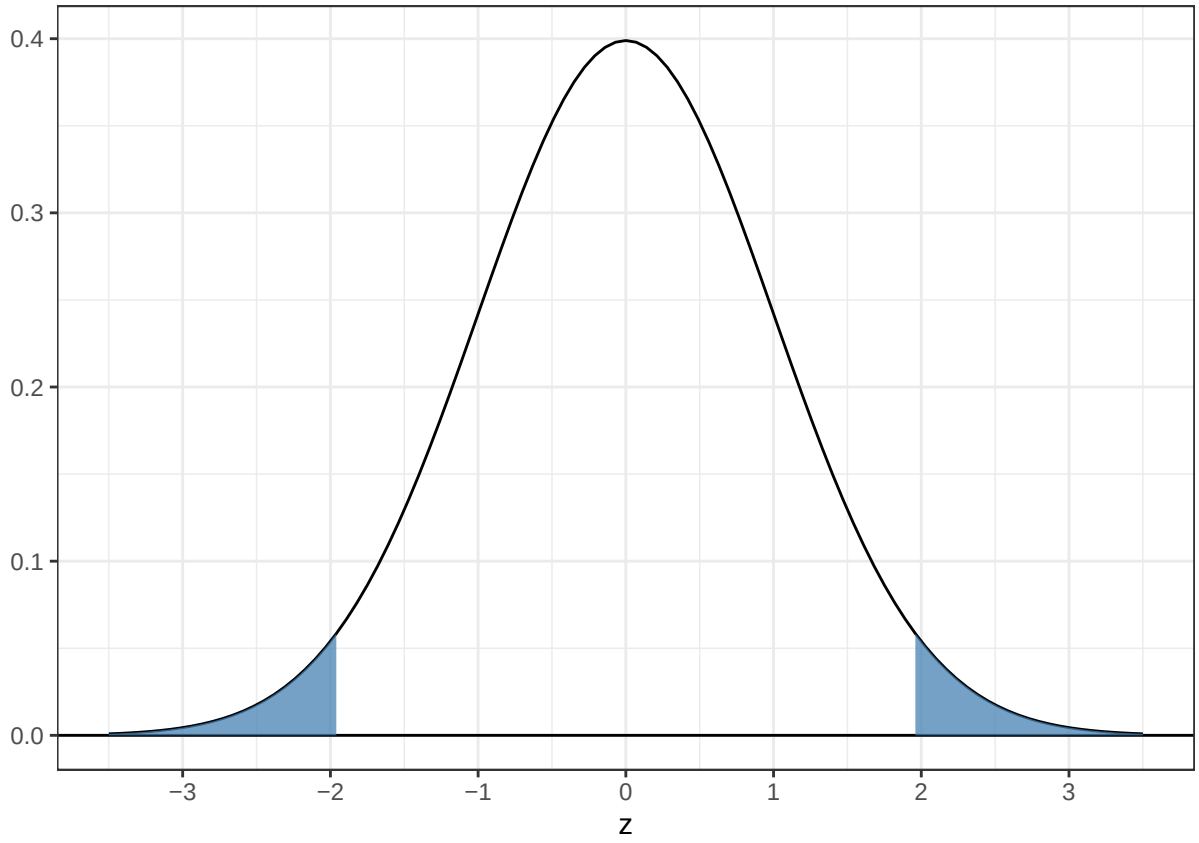
The boundary for the upper range is the score for which the proportion below is $1 - .025 = .975$:

```
qnorm(.975)
```

```
## [1] 1.959964
```

You could have guessed that because the standard normal distribution is symmetric around zero.

Here's the graph:



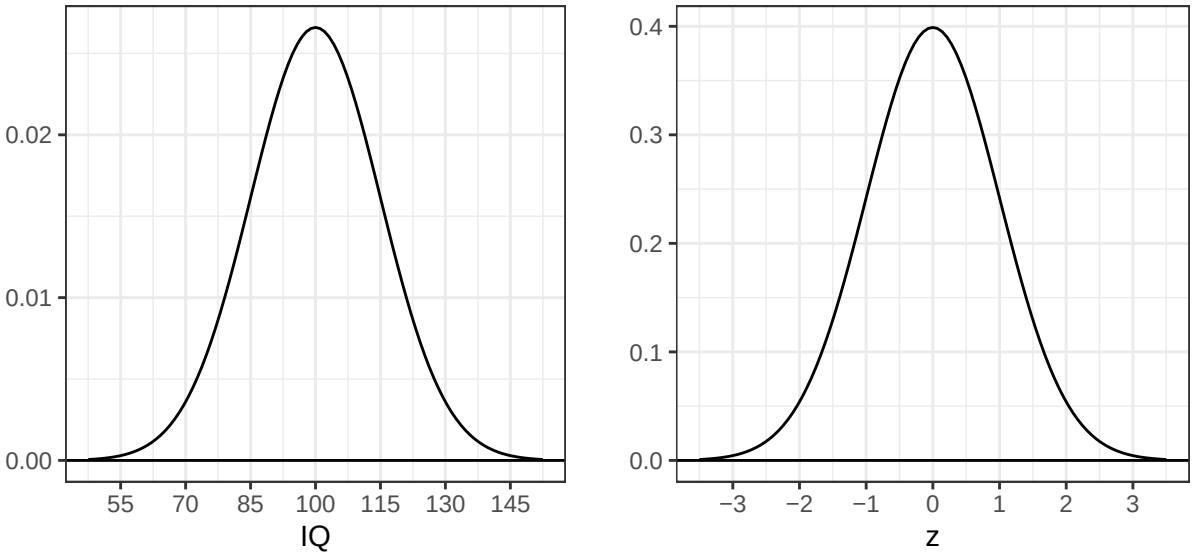
Remember this number - it's very close to 2. About 95% of the area of the standard normal falls between -2 and 2. This means that if you were to draw a random sample from the standard normal, it'll fall outside the range of -2 and 2 about 5% of the time. You'll see this number, 5%, a lot in the future.

4.5 Non-standard normal distributions

All normal distributions have the same shape - they only differ by their means and standard deviations. A normal distribution with mean μ and standard deviation σ has the probability density function:

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

Consider IQ scores, which are normalized to have a mean of 100 and a standard deviation of 15. The pdf for IQ's is just a shifted and scaled version of the standard normal pdf:



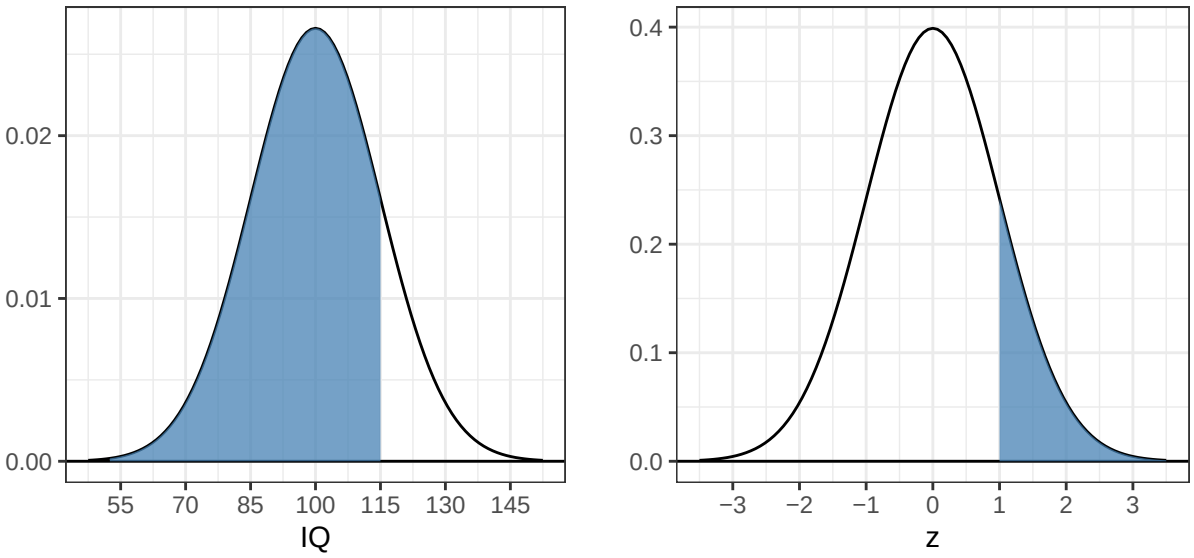
4.5.1 Example 6 What proportion of IQ's area above 115?

I picked this score because it's exactly one standard deviation above the mean. All normal distributions have the same shape, so the area is the same as the area above 1 for the standard normal, since the standard deviation of the standard normal is 1, which we calculated in Example 1 as 0.1586553.

It follows that probabilities for any IQ score can be found using the standard normal by converting the score into *standard deviation units*. If x is your score (IQ in this example), then the score in standard deviation units is:

$$z = \frac{x - \mu}{\sigma}$$

You can see that the areas are the same in this figure:



You might have noticed that the y-axis scales for these two plots are different. That's because the scales on the x-axis are different. Although the curves are smooth, you can think of the class interval width for IQs to be 15 times wider than for the z-scores. So in order for the bars to have the same heights, we need to divide the heights of the bars for the IQ scores by 15. That's why the height in the z-score of 0.3 corresponds to a height in the IQ scores of $\frac{.3}{15} = 0.02$.

The converted score is called *z-score*.

4.5.2 Example 7: What proportion of IQs fall between 100 and 118?

We need to convert the IQ scores to z-scores:

$$z_1 = \frac{100-100}{15} = 0, z_2 = \frac{118-100}{15} = 1.2$$

So the answer is:

```
pnorm(1.2)-pnorm(0)
```

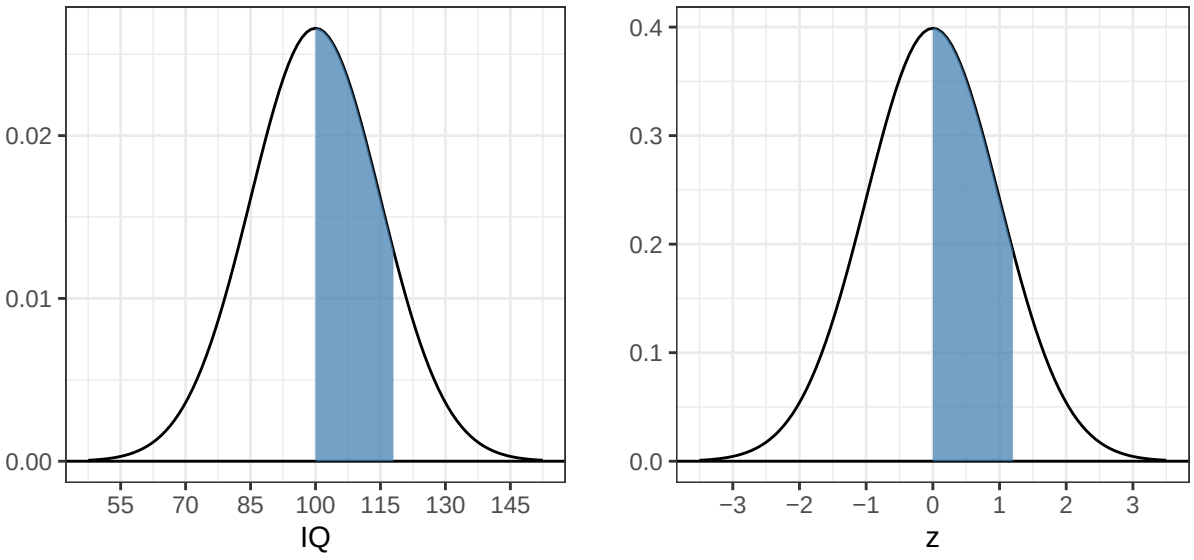
```
## [1] 0.3849303
```

There is a shortcut that bypasses the need to calculate the two z-scores. `pnorm` let you pass in the means and standard deviations of your population as arguments. Without them they're set to the default values of $\mu = 0$ and $\sigma = 1$. See how this gives you the same answer:

```
pnorm(118,100,15) - pnorm(100,100,115)
```

```
## [1] 0.3849303
```

Here's what the areas look like for both the IQ and the standard normal distributions:



4.5.3 Example 8: What IQ marks the top 1% of all IQ's?

Since you have a proportion (.01) we need to use `qnorm` to find the z-score:

```
z <- qnorm(1-.01)
z
```

```
## [1] 2.326348
```

Then we need to find the IQ that is 2.3263479 standard deviations above the mean. This is $IQ = 100+(2.3263479)(15)$:

```
100 + z*15
```

```
## [1] 134.8952
```

In general, if you have a z-score, we can rearrange the original formula for the z-score:

$$z = \frac{x-\mu}{\sigma}$$

and solve for x:

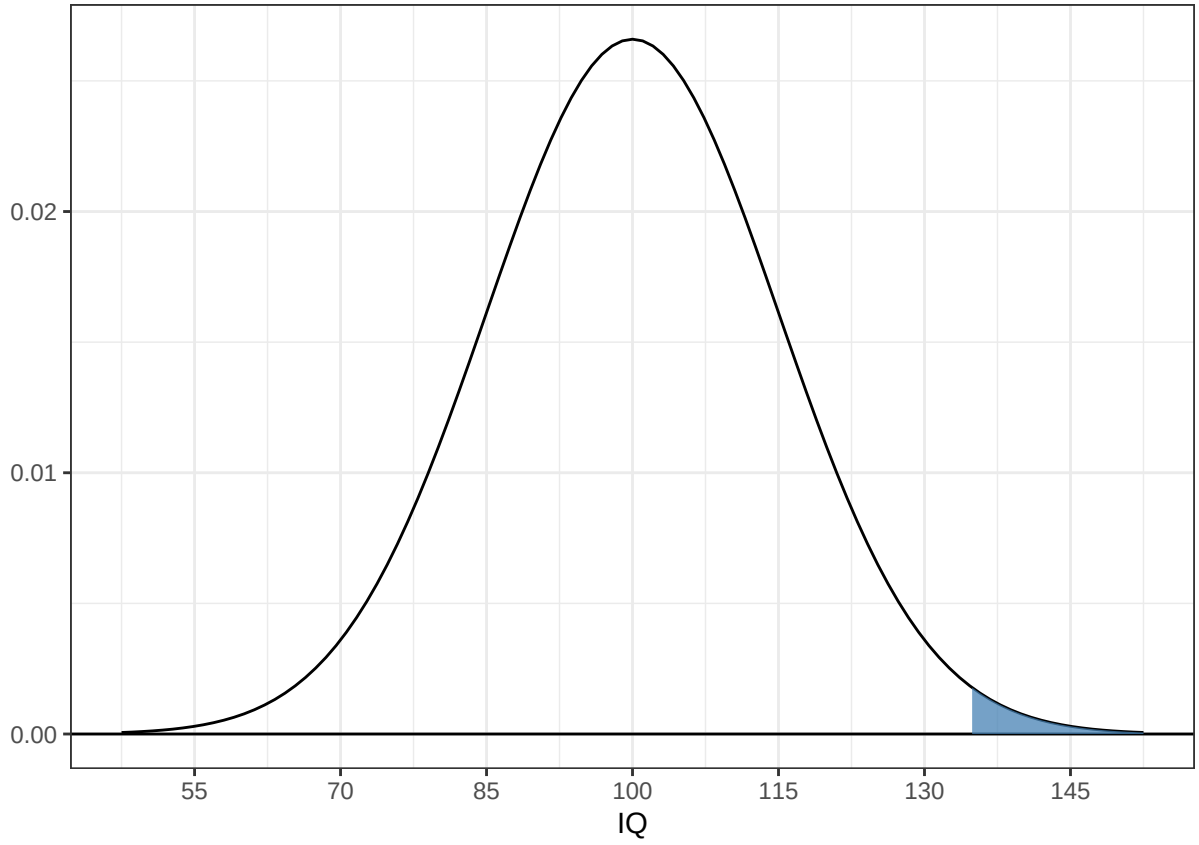
$$x = \mu + z\sigma$$

You might have guessed that `qnorm` also allows for the shortcut. So the quick way to answer this problem is:

```
qnorm(1-.01,100,15)
```

```
## [1] 134.8952
```

Here's what that top 1% (above an IQ of about 135) looks like:



4.5.4 Example 9: What proportion of women in the world are taller than 70 inches?

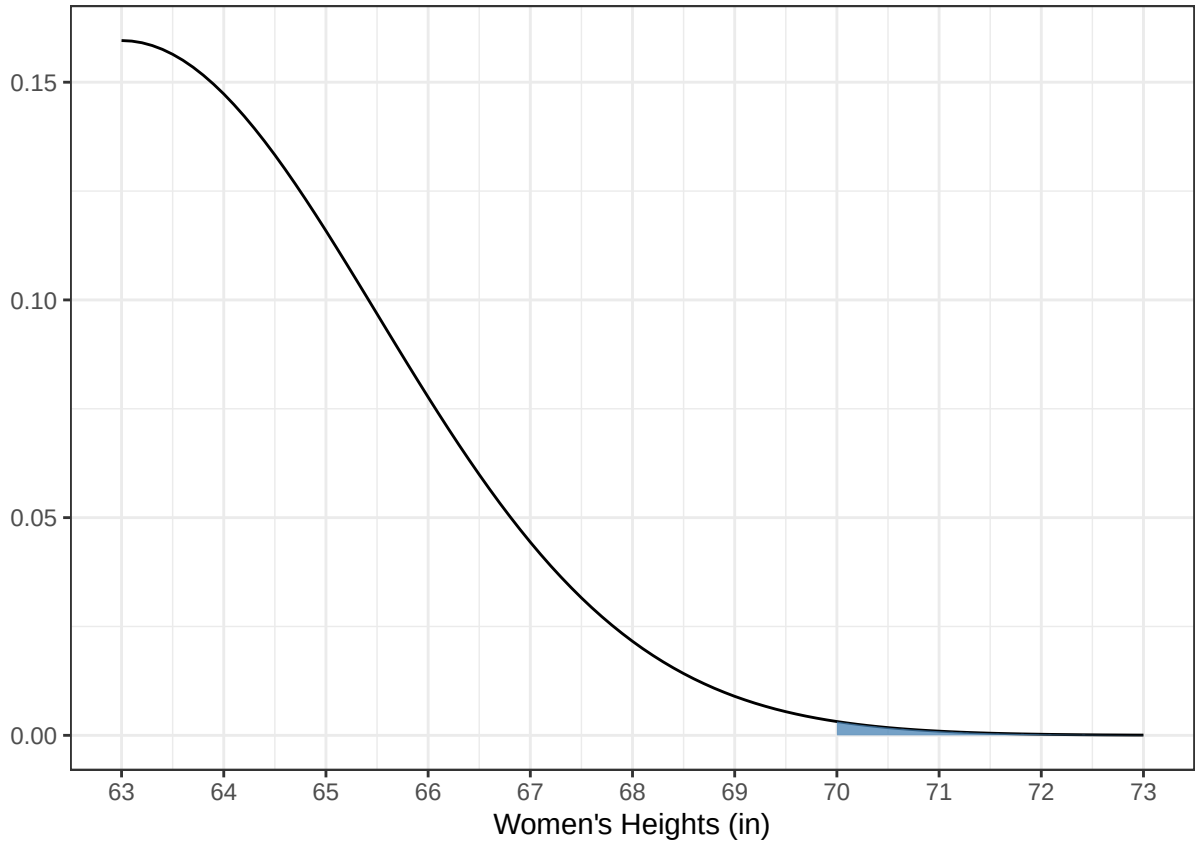
For this we need to know something about the distribution of heights of women in the world. Digging around the world wide web, I've come up with the estimate that the average height of women in the world globally is 63 inches, with a standard deviation of 2.5 inches.

To solve this problem we'll use the `pnorm` function:

```
mu <- 63
sigma <- 2.5
1-pnorm(70,mu,sigma)
```

```
## [1] 0.00255513
```

This is a tiny proportion. To visualize this we need to zoom in on the upper tail of the distribution:



4.5.5 Example 9: What heights cover the middle 50% of the population of the women's heights?

This can be done with the `qnorm` function, since we're given proportions. We know that the proportions in the two tails are 0.25:

```
qnorm(.25,mu,sigma)
```

```
## [1] 61.31378
```

```
qnorm(.75,mu,sigma)
```

```
## [1] 64.68622
```

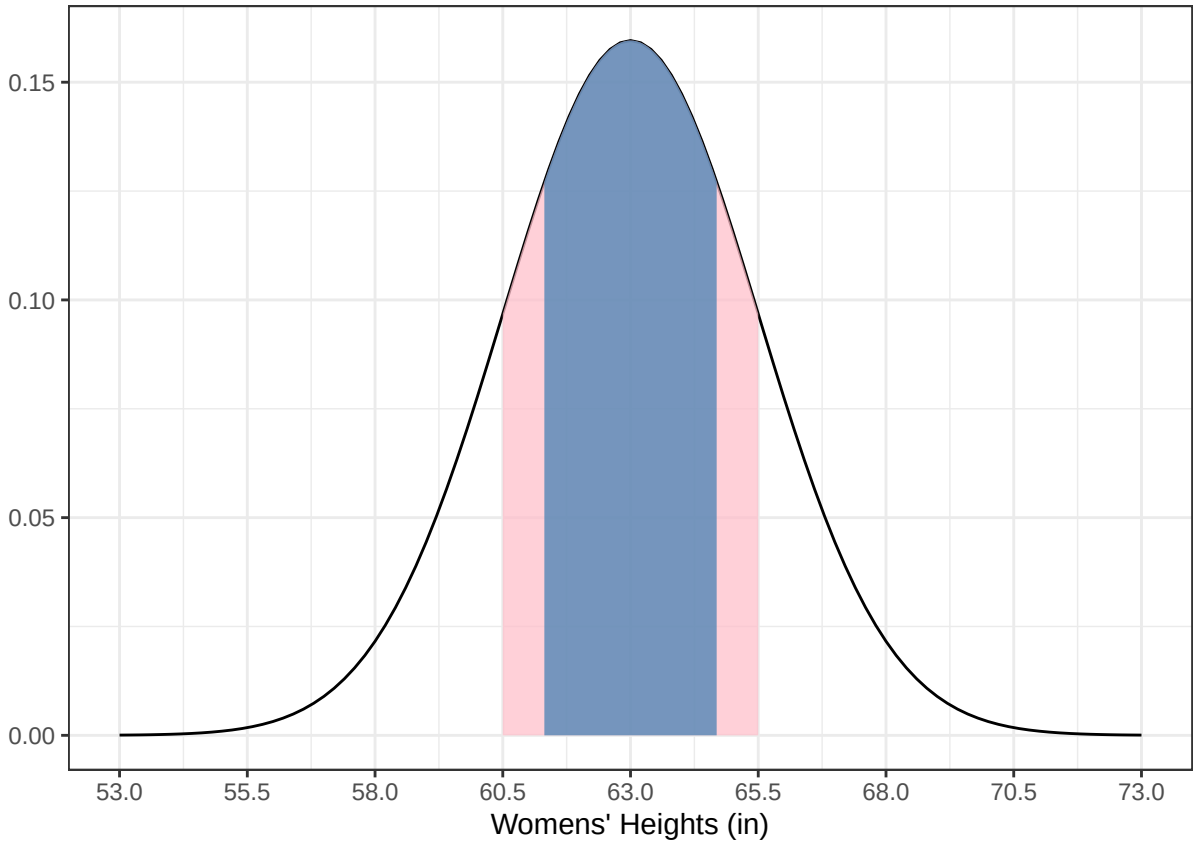
The range between the upper 75% and upper 25%, divided by 2, is called the *semi-interquartile-range*, and has it's own symbol, Q. Q for women's heights is:

```
Q <- (qnorm(.75,mu,sigma)-qnorm(.25,mu,sigma))/2
Q
```

```
## [1] 1.686224
```

From example 2 we calculated that about 2/3 of the area under the normal distribution falls between ± 1 standard deviation of the mean. Since ± Q covers only the middle 50% it makes sense that Q is smaller than the standard deviation of 2.5 inches.

Here's what the middle 50% looks like in blue overlaid on ± one standard deviation in pink:


0.15
0.10
0.05
0.00
53.0
55.5
58.0
60.5
63.0
65.5
68.0
70.5
73.0
Womens' Heights (in)

Chapter 5

The Central Limit Theorem

In science, we are typically interested in the properties of a certain population, like, for example, the average height of men in a population. But sampling the entire population is usually impossible. Instead, we obtain a random sample from the population and hope that the mean from this sample is a good estimate of the population mean. How well does a sample mean represent the population mean?

The mean is an *unbiased* statistic, which means that a typical sample mean won't be on average higher or lower than the population mean. But how close is a typical sample mean to the population mean? You probably have the intuition that this answer depends on the size of the sample. The larger the sample size, the closer the mean of your typical sample will be to the population mean. For example, it's not too unusual to meet a man that is 6 foot 2 inches. But a room full of 25 men with average of 6 foot 2 would be surprising. Unless you're in Holland.

The Central Limit Theorem is a formal description of this intuition. It's a theorem that tells you about the *distribution of sample means*.

5.1 The sampling distribution of the mean

Let's take a moment to think about that term "distribution of sample means". Every time you draw a sample from a population, the mean of that sample will be different. Some means will be more likely than other means. So it makes sense to think about the means drawn from a population as having their own distribution. This distribution is called the *sampling distribution of the mean*. The Central Limit Theorem tells us how the shape of the sampling distribution of the mean relates to the distribution of the population that these means are drawn from.

To define some terms, if samples from a population are labeled with the variable X, we define the parameters of mean as μ_x and the standard deviation as σ_x. Remember, the greek letter is the parameter, and the subscript is the name of the thing that we're talking about.

Now consider the sampling distribution of the mean. You know that sample means are written as $\bar{x}$. Using the same notation, the sampling distribution of the mean has its own mean, called $\mu_{\bar{x}}$, and its own standard deviation, called $\sigma_{\bar{x}}$.

There are three parts to the Central Limit Theorem:

1) The sampling distribution of the mean will have the same mean as the population mean. Formally, we state: $\mu_{\bar{x}} = \mu_x$.

This just means what I said earlier, that the mean is unbiased, so that sample means will be, on average, equal to the population mean.

2) For a sample size n, the standard deviation of the sampling distribution of the mean will be $\sigma_{\bar{x}} = \frac{\sigma_x}{\sqrt{n}}$

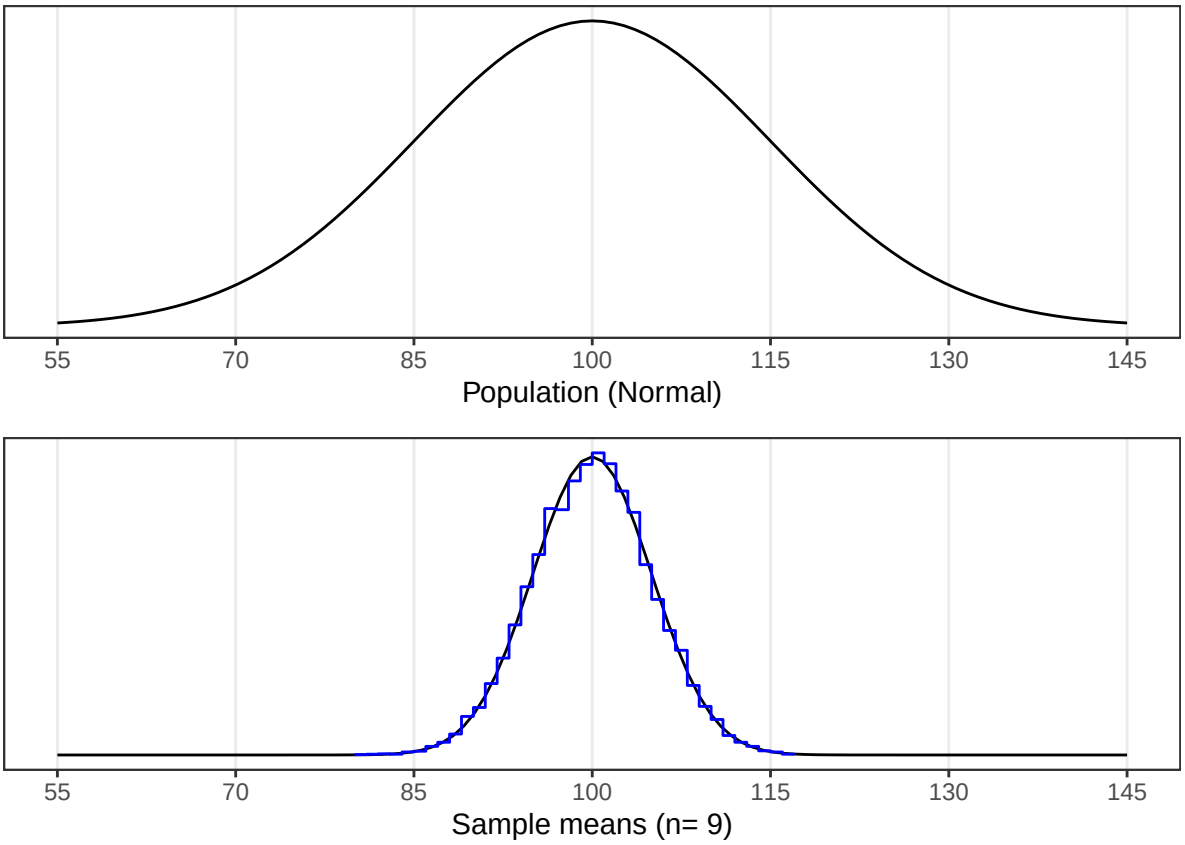
The name for $\sigma\bar{x}$ is sometimes shortened to the *standard error of the mean*, and sometimes shortened even more to 's.e.m.' or even just 'SE'.

This is a formalization of the intuition above. Since $\sqrt{n}$ is in the denominator, it means that as your sample size gets bigger, the standard deviation of the distribution of means, $\sigma_{\bar{x}}$, gets smaller. So as you increase sample size, any given sample mean will be on average closer to the population mean.

3) The sampling distribution of the mean will tend to be close to normally distributed. Moreover, the sampling distribution of the mean will tend towards normality as (a) the population tends toward normality, and/or (b) the sample size increases.

This last part is the most remarkable. It means that even if the population is not normally distributed, the sampling distribution of the mean will be roughly normal if your sample size is large enough.

Below is the results of a simulation demonstrating the Central Limit Theorem.



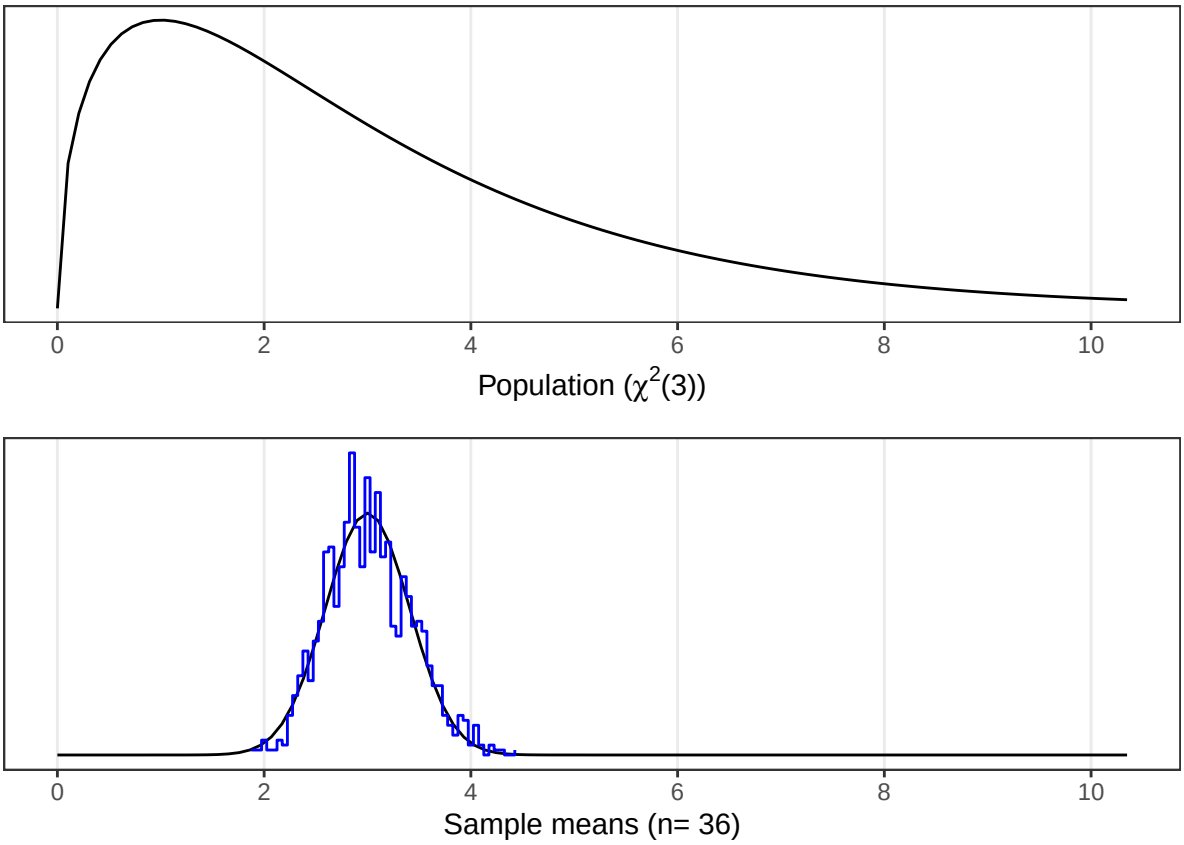
The graph on the top is the population distribution, which for this example is normal with a mean of $\mu_x = 100$ and a standard deviation of $\sigma_x = 15$. The simulation ran 10^4 samples of size n=9.

In the bottom graph in blue you can see a histogram of the means from these samples. Drawn on top of the histogram is the expected normal distribution of means according to the Central Limit Theorem:

$\mu_{\bar{x}} = \mu_x = 100$ and $\sigma_{\bar{x}} = \frac{\sigma_x}{\sqrt{n}} = \frac{15}{\sqrt{9}} = 5$

You can see that the smooth curve matches the distribution of actual means. The mean of sample means in this simulation is 99.97 and the standard deviation of the means is 5.01 - pretty close to what's expected from the Central Limit Theorem.

In this second demonstration we'll draw from a population that is clearly not normally distributed:



This time the population is a 'Chi-squared' (χ^2) distribution, which will show up later. The population has a mean of 3 and standard deviation of 2.4495.

Like before, the blue histogram in the bottom graph shows the distribution of 1000 means of sample size 36 from this skewed population distribution. Notice that sample means have a nice, symmetric normal-looking distribution.

The Central Limit Theorem predicts that the distribution of means should be roughly normal with a mean of $\mu_{\bar{x}} = \mu_x = 3$ and a standard deviation of $\sigma_{\bar{x}} = \frac{\sigma_x}{\sqrt{n}} = \frac{2.4495}{\sqrt{36}} = 0.4082$

In this simulation, the mean of these means is 2.99 which is close to the population mean, and a standard deviation of 0.4125, which is close to what's expected from the Central Limit Theorem. Like the first example, the smooth curve in the bottom graph is a normal distribution with the mean and standard deviation expected from the Central Limit Theorem. It clearly matches the distribution of sample means from the simulation.

The Central Limit Theorem is powerful because, as we've learned from previous chapters, if you know that a distribution is normal, and you know its mean and standard deviation, then you know everything about this distribution.

Next we'll work through some examples to show how we can use the Central Limit Theorem to make inferences about the population that a sample is drawn from.

5.2 Examples

5.2.1 Example 1

What is the probability that a mean drawn from a sample of 25 IQ scores will exceed 103 points?

IQs are normalized to have a mean of 100 and a standard deviation of 15. From the Central Limit Theorem, a mean from a sample size of 25 will come from its own distribution with a mean of 100 and a standard deviation of $\frac{15}{\sqrt{25}} = 3$

We can then use R's 'pnorm' function to find the area from this normal distribution above a mean of 103:

```
mu <- 100
sigma <-15
X <- 103
sem <- sigma/sqrt(n)
p <- 1-pnorm(X,mu,sem)
p
```

```
## [1] 0.1586553
```

So there is about a 16% chance that we'll draw a mean IQ of 103 or higher.

5.2.2 Example 2

In the last chapter we had examples using the fact that the average height of women in the world globally is 63 inches, with a standard deviation of 2.5 inches. Consider the mean height of 100 randomly sampled women this population. For what mean height will 5% of the means fall above?

Answer:

With a sample size of 100, the Central Limit Theorem states that the means will be distributed with a mean of 63 inches and a standard deviation of $\frac{2.5}{\sqrt{100}} = 0.25$ inches. We can find the height for which 5% falls above using R's 'qnorm' function to find the height for which the area below is 0.95:

```
mu <- 63
sigma <- 2.5
sem <- sigma/sqrt(n)
n <- 100
p <- 1-.05
x <- qnorm(p,mu,sem)
sprintf('The height for which %g%% of the means falls above is %5.2f inches.',100*(1-p),x)
```

```
## [1] "The height for which 5% of the means falls above is 63.41 inches."
```

Always check your answer to see if it makes sense. 63.41 inches is greater than the population mean of 63 inches, which makes sense since this mean is the cutoff for the upper 5%.

5.2.3 Example 3

From the survey, the mean height of the 122 women in Psych 315 had a height of 64.7 inches. If we assume that the women in Psych 315 are drawn randomly from the world population from Example 2, what is the probability of obtaining a mean this high or higher by chance?

Answer:

From the Central Limit Theorem, the means will be distributed with a mean of 63 inches and a standard deviation of $\frac{2.5}{\sqrt{122}} = 0.23$ inches. The probability of obtaining a mean of 64.7 or higher can be calculated from R's 'pnorm' function. Here's the code, which loads in the survey data:

```
mu <- 63
sigma <- 2.5

survey <- read.csv("http://www.courses.washington.edu/psy315/datasets/Psych315W21survey.csv")

women.height <- na.omit(survey$height[survey$gender=="Woman"])
x <- mean(women.height)
n <-length(women.height)
sem <- sigma/sqrt(n)

p <- 1-pnorm(x,mu,sem)  #sem from problem 2
sprintf('The probability of drawing a mean of %5.2f or higher is %g',x,p)
```

```
## [1] "The probability of drawing a mean of 64.70 or higher is 2.4869e-14"
```

This is an extremely low probability. That's because with a standard error of the mean of $\sigma_{\bar{x}} = 0.23$, a mean of 64.7 is $\frac{64.7-63}{0.23} = 7.53$ standard deviations above the population mean of 2.5.

What does this mean about the women in Psych 315? They seem to be impossibly tall. This is true assuming that these students are randomly sampled from the world's population. So logically, there seems to be something wrong with this assumption. It is more likely that this assumption is false, and the true population that we're drawing from has a mean taller than 63 inches.

‘

Chapter 6

Hypothesis Testing: the z-test

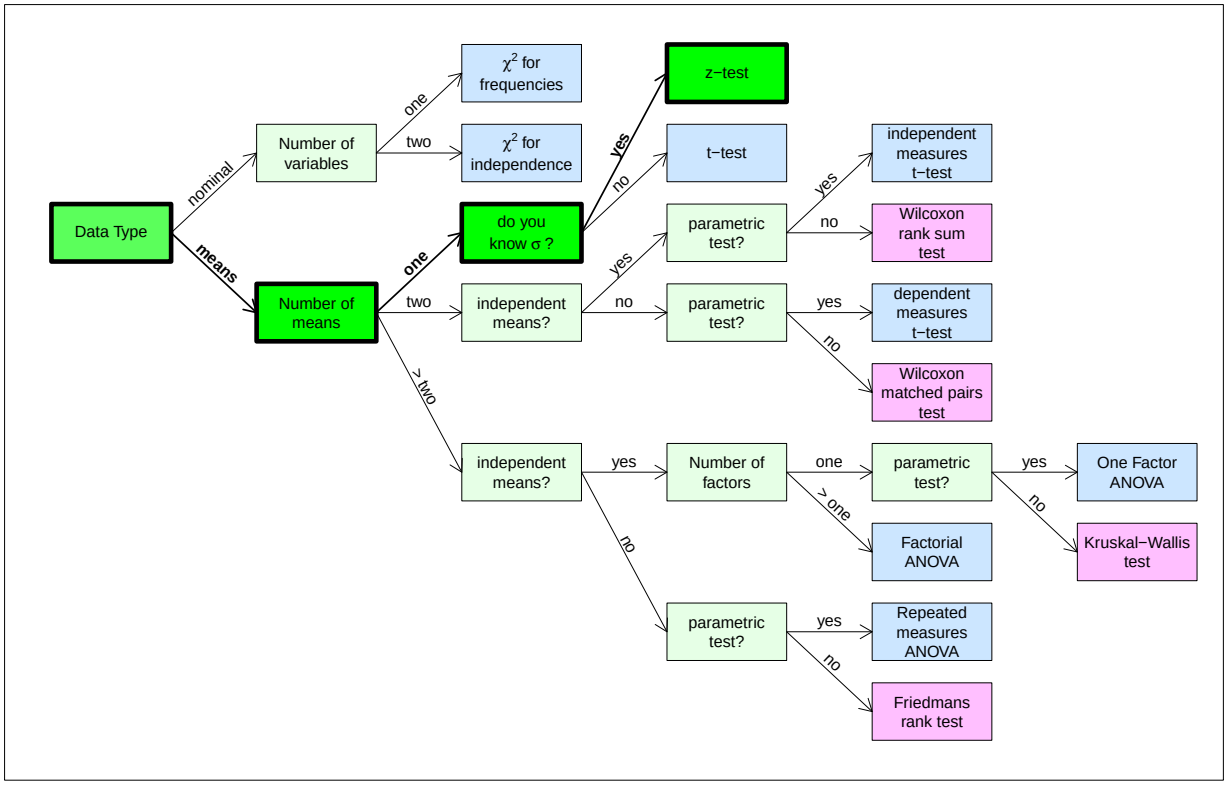
We've all had the experience of standing at a crosswalk waiting staring at a pedestrian traffic light showing the little red man. You're waiting for the little green man so you can cross. After a little while you're still waiting and there aren't any cars around. You might think 'this light is really taking a long time', but you continue waiting. Minutes pass and there's still no little green man. At some point you come to the conclusion that the light is broken and you'll never see that little green man. You cross on the little red man when it's clear.

You may not have known this but you just conducted a hypothesis test. When you arrived at the crosswalk, you assumed that the light was functioning properly, although you will always entertain the possibility that it's broken. In terms of hypothesis testing, your 'null hypothesis' is that the light is working and your 'alternative hypothesis' is that it's broken. As time passes, it seems less and less likely that light is working properly. Eventually, the probability of the light working given how long you've been waiting becomes so low that you reject the null hypothesis in favor of the alternative hypothesis.

This sort of reasoning is the backbone of hypothesis testing and inferential statistics. It's also the point in the course where we turn the corner from descriptive statistics to inferential statistics. Rather than describing our data in terms of means and plots, we will now start using our data to make inferences, or generalizations, about the population that our samples are drawn from. In this course we'll focus on standard hypothesis testing where we set up a null hypothesis and determine the probability of our observed data under the assumption that the null hypothesis is true (the much maligned p-value). If this probability is small enough, then we conclude that our data suggests that the null hypothesis is false, so we reject it.

In this chapter, we'll introduce hypothesis testing with examples from a 'z-test', when we're comparing a single mean to what we'd expect from a population with known mean and standard deviation. In this case, we can convert our observed mean into a z-score for the standard normal distribution. Hence the name z-test.

It's time to introduce the *hypothesis test flow chart*. It's pretty self explanatory, even if you're not familiar with all of these hypothesis tests. The z-test is (1) based on means, (2) with only one mean, and (3) where we know σ, the standard deviation of the population. Here's how to find the z-test in the flow chart:



6.1 Women's height example

Let's work with the example from the end of the last chapter where we started with the fact that the heights of US women has a mean of 63 and a standard deviation of 2.5 inches. We calculated that the average height of the 122 women in Psych 315 is 64.7 inches. We then used the central limit theorem and calculated the probability of a random sample 122 heights from this population having a mean of 64.7 or greater is $2.4868996 \times 10^{-14}$. This is a very, very small number.

Here's how we do it using R:

```
mu <- 63  # population mean
sigma <- 2.5 # population standard deviation
alpha <- .05

survey <- read.csv("http://www.courses.washington.edu/psy315/datasets/Psych315W21survey.csv")

women.height <- na.omit(survey$height[survey$gender=="Woman"])
x <- mean(women.height)    # sample mean
n <-length(women.height)   # sample size
sem <- sigma/sqrt(n)       # standard error of the mean

p <- 1-pnorm(x,mu,sem)     # p-value from normal distribution
print(sprintf('p = %1.15f',p))
```

```
## [1] "p = 0.000000000000025"
```

Let's think of our sample as a random sample of UW psychology students, which is a reasonable assumption since all psychology students have to take a statistics class. What does this sample say about the psychology students that are women at UW compared to the US population? It could be that these psychology students at UW have the

same mean and standard deviation as the US population, but our sample just happens to have an unusual number of tall women, but we calculated that the probability of this happening is really low. Instead, it makes more sense that the population that we're drawing from has a mean that's greater than the US population mean. Notice that we're making a conclusion about the whole population of women psychology students based on our one sample.

Using the terminology of hypothesis testing, we first assumed the null hypothesis that UW women psych students have the same mean (and standard deviation) as the US population. The null hypothesis is written as:

$$H_0 : \mu = 63$$

In this example, our alternative hypothesis is that the mean of our population is larger than the mean of null hypothesis population. We write this as:

$$H_A : \mu > 63$$

Next, after obtaining a random sample and calculate the mean, we calculate the probability of drawing a mean this large (or larger) from the null hypothesis distribution.

If this probability is low enough, we reject the null hypothesis in favor of the alternative hypothesis. When our probability allows us to reject the null hypothesis, we say that our observed results are 'statistically significant'.

In statistics terms, we never say we 'accept that alternative hypothesis' as true. All we can say is that we don't think the null hypothesis is true. I know it's subtle, but in science can never prove that a hypothesis is true or not. There's always the possibility that we just happened to grab an unusual sample from the null hypothesis distribution.

6.2 The hated p<.05

The probability that we obtain our observed mean or greater given that the null hypothesis is true is called the p-value. How improbable is improbable enough to reject the null hypothesis? The p-value for our example above on women's heights is astronomically low, so it's clear that we should reject H_0.

The p-value that's on the border of rejection is called the alpha (α) value. We reject H_0 when our p-value is less than α.

You probably know that the most common value of alpha is $\alpha = .05$.

The first publication of this value dates back to Sir Ronald Fisher, in his seminal 1925 book *Statistical Methods for Research Workers* where he states:

"It is convenient to take this point as a limit in judging whether a deviation is considered significant or not. Deviations exceeding twice the standard deviation are thus formally regarded as significant." (p. 47)

If you read the chapter on the normal distribution, then you should know that 95% of the area under the normal distribution lies within ± two standard deviations of the mean. So the probability of obtaining a sample that exceeds two standard deviations from the mean (in either direction) is .05.

6.3 IQ example

Let's do an example using IQ scores. IQ scores are normalized to have a mean of 100 and a standard deviation of 15 points. Because they're normalized, they are a rare example of a population which has a known mean and standard deviation. In the next chapter we'll discuss the t-test, which is used in the more common situation when we don't know the population standard deviation.

Suppose you have the suspicion that graduate students have higher IQ's than the general population. You have enough time to go and measure the IQ's of 25 randomly sampled grad students and obtain a mean of 105. Is this difference between our this observed mean and 100 statistically significant using an alpha value of $\alpha = 0.05$?

Here the null hypothesis is:

$$H_0 : \mu = 100$$

And the alternative hypothesis is:

$$H_A : \mu > 100$$

We know that the parameters for the null hypothesis are:

$$\mu = 100$$

and

$$\sigma = 15$$

From this, we can calculate the probability of observing our mean of 105 or higher using the central limit theorem and what we know about the normal distribution:

$$\sigma_{\bar{x}} = \frac{\sigma_x}{\sqrt{n}} = \frac{15}{\sqrt{25}} = 3$$

From this, we can calculate the probability of our observed mean using R's 'pnorm' function. Here's how to do the whole thing in R.

```
mu <- 100 # null hypothesis population mean
sigma <- 15 # null hypothesis population standard deviation
n <- 25  # sample size
x_bar <- 105  # sample mean

sem <- sigma/sqrt(n)  # standard error or the mean

p_value <- round(1-pnorm(x_bar,mu,sem),4)
print(sprintf('p = %5.4f',p_value))
```

```
## [1] "p = 0.0478"
```

Since our p-value of 0.0478 is (just barely) less than our chosen value of $\alpha = 0.05$ as our criterion, we reject H_0 for this (contrived) example and conclude that our observed mean of 105 is significantly greater than 100, so our study suggests that the average graduate student has a higher IQ than the overall population.

You should feel uncomfortable making such a hard, binary decision for such a borderline case. After all, if we had chosen our second favorite value of alpha, $\alpha = .01$, we would have failed to reject H_0. This discomfort is a primary reason why statisticians are moving away from this discrete decision making process. Later on we'll discuss where things are going, including reporting effect sizes, and using confidence intervals.

6.4 Alpha values vs. critical values

Using R's qnorm function, we can find the z-score for which only 5% of the area lies above:

```
qnorm(1-.05)
```

```
## [1] 1.644854
```

So the probability of a randomly sampled z-score exceeding 1.644854 is less than 5%. It follows that if we convert our observed mean into z-score values, we will reject H_0 if and only if our z-score is greater than 1.644854. This value is called the 'critical value' because it lies on the boundary between rejecting and failing to reject H_0.

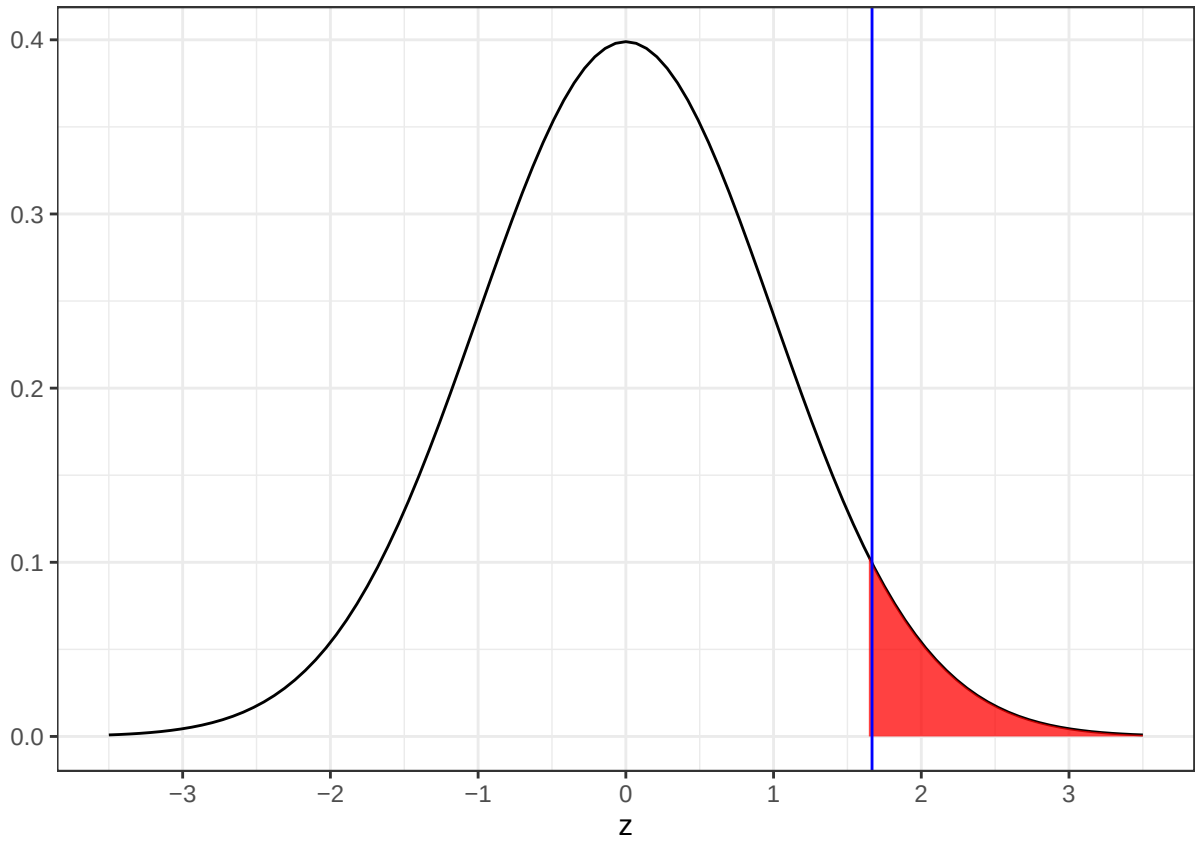
In our last example, the z-score for our observed mean is:

$$z = \frac{X - \mu}{\frac{\sigma}{\sqrt{n}}} = \frac{105 - 100}{3} = 1.67$$

Our z-score is just barely greater than the critical value of 1.644854, which makes sense because our p-value is just barely less than 0.05.

Sometimes you'll see textbooks will compare critical values to observed scores for the decision making process in hypothesis testing. This dates back to days were computers were less available and we had to rely on tables instead. There wasn't enough space in a book to hold complete tables which prohibited the ability to look up a p-value for any observed value. Instead only critical values for specific values of alpha were included. If you look at really old papers, you'll see statistics reported as $p < .05$ or $p < .01$ instead of actual p-values for this reason.

It may help to visualize the relationship between p-values, alpha values and critical values like this:



The red shaded region is the upper 5% of the standard normal distribution which starts at the critical value of z=1.644854. This is sometimes called the 'rejection region'. The blue vertical line is drawn at our observed value of z=1.67. You can see that the red line falls just inside the rejection region, so we Reject H_0!

6.5 One vs. two-tailed tests

Recall that our alternative hypothesis was to reject if our mean IQ was significantly greater than the null hypothesis mean: $H_A : \mu > 100$. This implies that the situation where $\mu < 100$ is never even in consideration, which is weird. In science, we're trying to understand the true state of the world. Although we have a hunch that grad student IQ's are higher than average, there is always the possibility that they are lower than average. If our sample came up with an IQ well below 100, we'd simply fail to reject H_0 and move on. This feels like throwing out important information.

The test we just ran is called a 'one-tailed' test because we only reject H_0 if our results fall in one of the two tails of the population distribution.

Instead, it might make more sense to reject H_0 if we get either an unusually large or small score. This means we need two critical values - one above and one below zero. At first thought you might think we just duplicate our critical value from a one-tailed test to the other side. But will double the area of the rejection region. That's not a good thing because if H_0 is true, there's actually a 2α probability that we'll draw a score in the rejection region.

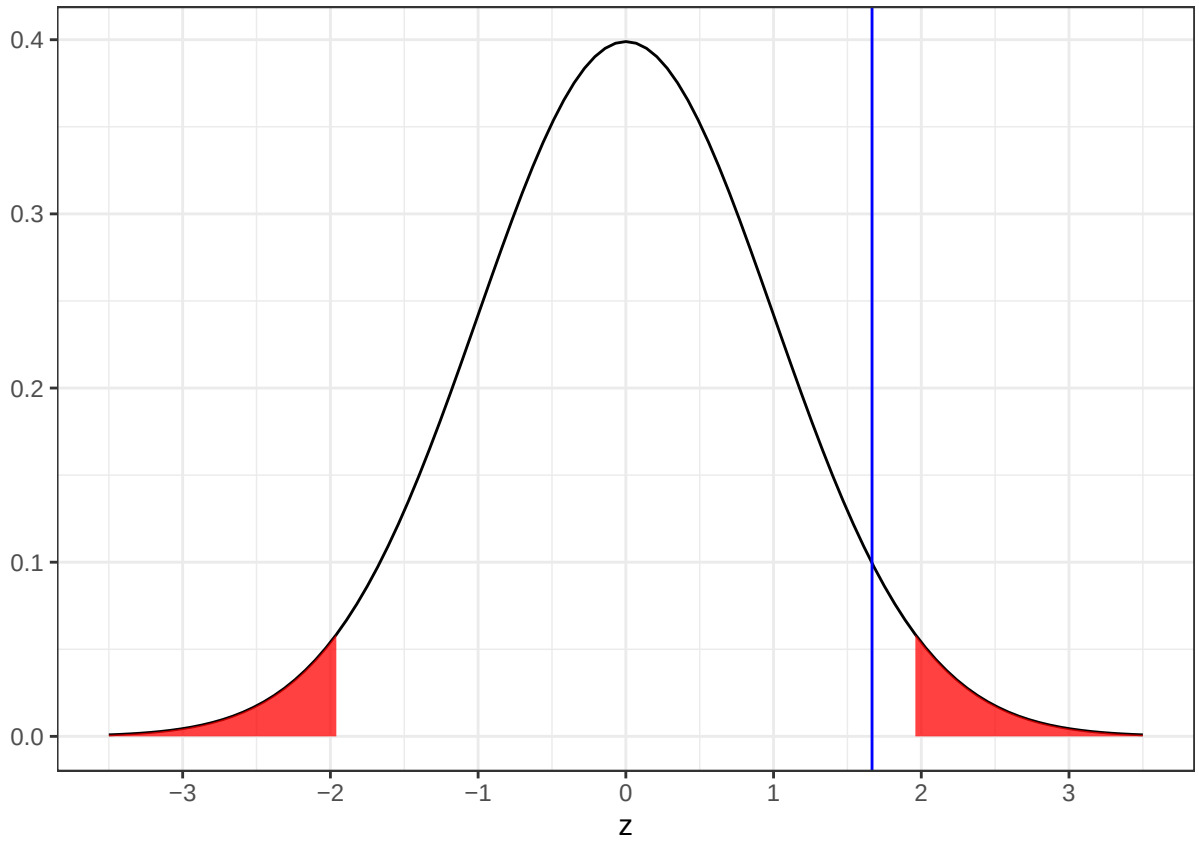
Instead, we divide the area into two tails, each containing an area of $\frac{\alpha}{2}$. For $\alpha = 0.05$, we can find the critical value of z with qnorm:

```
qnorm(1-alpha/2)
```

```
## [1] 1.959964
```

So with a two-tailed test at $\alpha = 0.05$ we reject H_0 if our observed z-score is either above z = 1.96 or less than -1.96. This is that value around 2 that Sir Ronald Fischer was talking about!

Here's what the critical regions and observed value of z looks like for our example with a two-tailed test:



You can see that splitting the area of $\alpha = 0.05$ into two halves increased the critical value in the positive direction from 1.64 to 1.96, making it harder to reject H_0. For our example, this changes our decision: our observed value of z = 1.67 no longer falls into the rejection region, so now we fail to reject H_0.

If we now fail to reject H_0, what about the p-value? Remember, for a one-tailed test, p = α if our observed z-score lands right on the critical value of z. The same is true for a two-tailed test. But the z-score moved so that the area above that score is $\frac{\alpha}{2}$. So for a two-tailed test, in order to have a p-value of α when our z-score lands right on the critical value, we need to double p-value hat we'd get for a one-tailed test.

For our example, the p-value for the one tailed test was $p = 0.0478$. So if we use a two-tailed test, our p-value is $(2)(0.0478) = 0.0956$. This value is greater than $\alpha = 0.05$, which makes sense because we just showed above that we fail to reject H_0 with a two tailed test.

Which is the right test, one-tailed or two-tailed? Ideally, as scientists, we should be agnostic about the results of our experiment. But in reality, we all know that the results are more interesting if they are statistically significant. So you can imagine that for this example, given a choice between one and two-tailed, we'd choose a one-tailed test so that we can reject H_0.

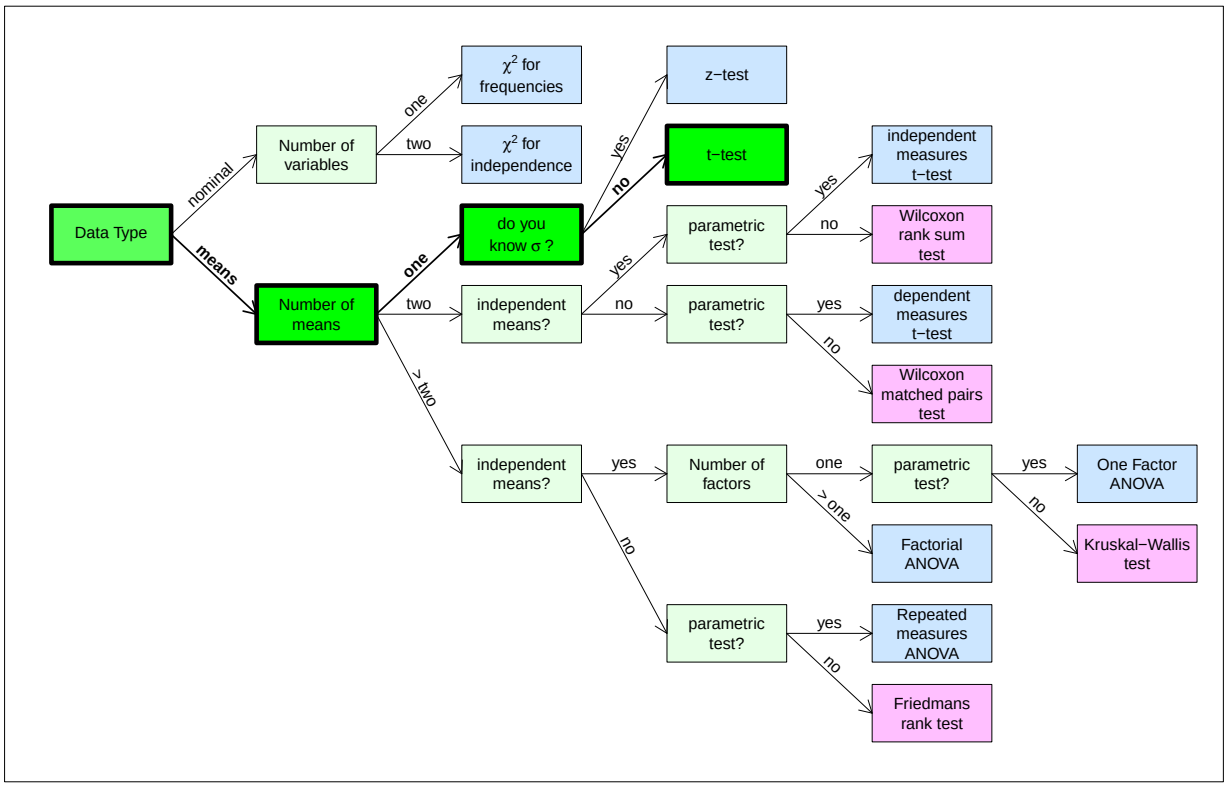
There are two problems with this. First, we should never adjust our choice of hypothesis test *after* we observe the data. That would be an example of 'p-hacking', a topic we'll discuss later. Second, most statisticians these days strongly recommend against one-tailed tests. The only reason for a one-tailed test is if there is no logical or physical possibility for a population mean to fall below the null hypothesis mean.

Chapter 7

One Sample T-Test

The one sample t-test is used to compare a *single mean* to an expected mean under the null hypothesis when you *don't know the standard deviation of the population.*

You can find this test in the flow chart here:



7.1 Using t.test to run a one-sample t-test

7.2 Simulating z-scores: replacing the population s.d. with the sample s.d.

See if you can follow what's going on in this R-code:

```r
nSamples <- 30000
mu <- 100
sigma <- 15
n <- 5  # sample size
t = numeric(nSamples)
z = numeric(nSamples)
for (i in 1:nSamples) {
  # draw a sample of size n from a normal distribution
  samp <- rnorm(n,mu,sigma)
  # calculate the z-statistic for this sample using the the population s.d.
  z[i] <- (mean(samp)-mu)/(sigma/sqrt(n))
  # calculate the t-statistic for this sample using the sample s.d.
  t[i] <- (mean(samp)-mu)/(sd(samp)/sqrt(n))
}
```

The code uses a for loop to generate 30000 samples of of size 5 drawn from a normal distribution using R's `dnorm` function. For each sample, a z-score is calculated by subtracting the population mean and dividing by the standard error of the mean using the known population standard deviation:

$$z = \frac{(\bar{x} - \mu_{hyp})}{\sigma_{\bar{x}}}$$

where:

$$\sigma_{\bar{x}} = \frac{\sigma}{\sqrt{n}}$$

But what if we don't know the population standard deviation? This is really the more common situation. After all, we're trying to make an inference about the population so it's unlikely that we know that much about it.

If we don't know the population standard deviation it might make sense to use the standard deviation of our sample instead. After all, the sample standard deviation s_x is an unbiased estimate of the population standard deviation σ_x.

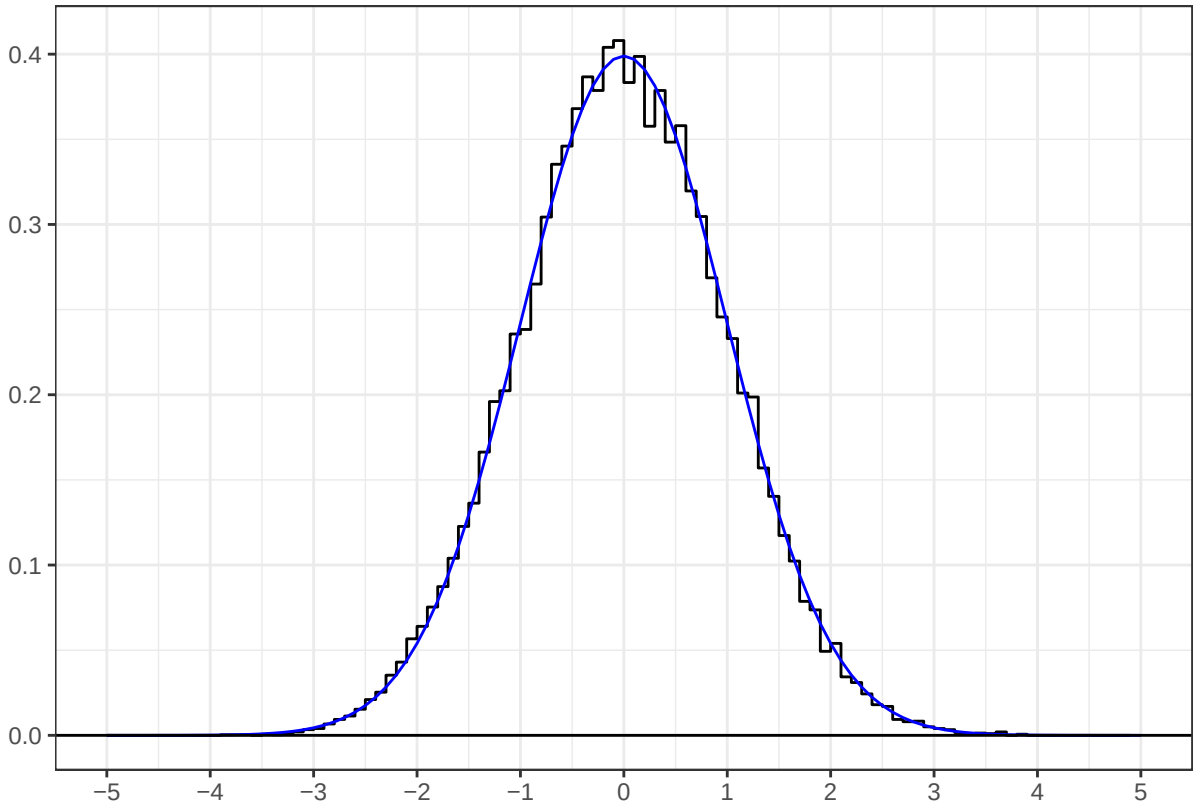
In the code, a score like the z-score is also calculated, but instead dividing by the standard error of the mean using the sample standard deviation. We call this score 't' instead of 'z' because you'll see in a second that it's not the same thing. Formally, 't' is calculated as:

$$t = \frac{(\bar{x} - \mu_{hyp})}{s_{\bar{x}}}$$

Where, just like for σ_x but using the sample standard deviation:

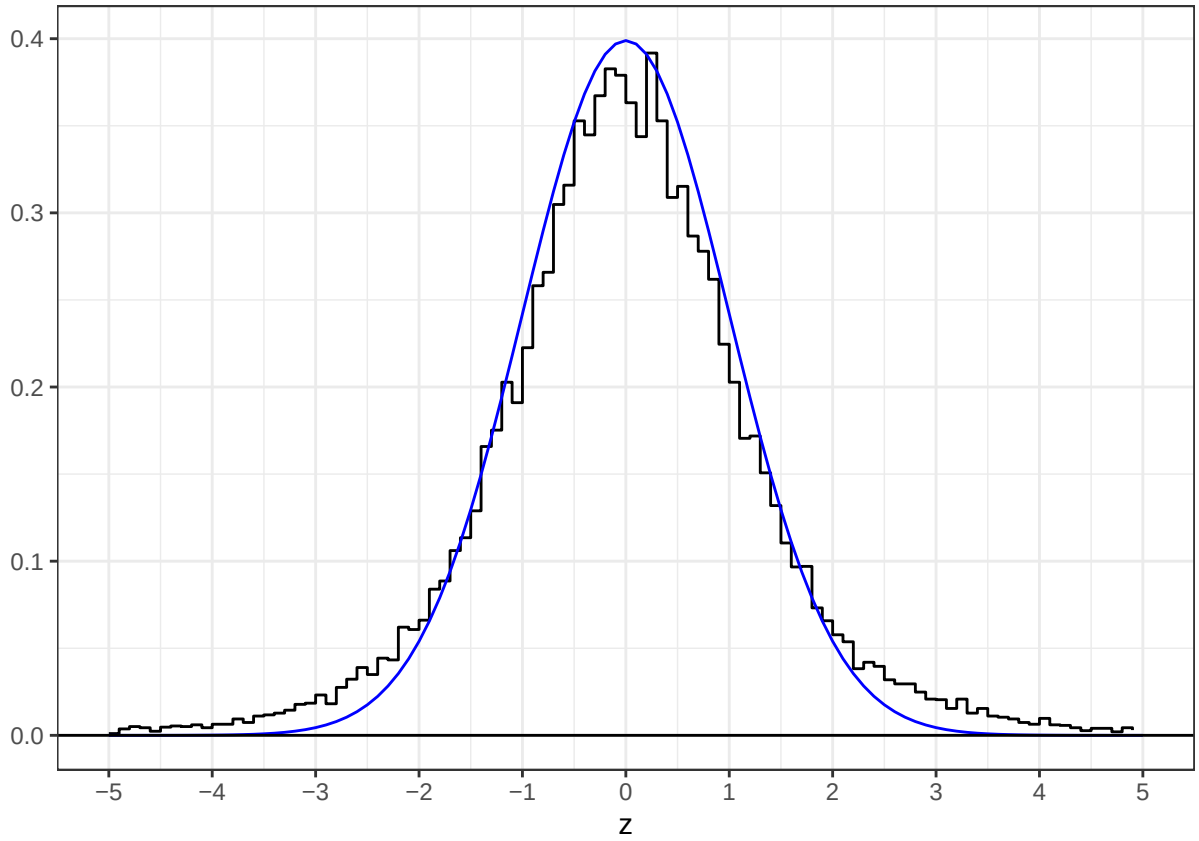
$$s_{\bar{x}} = \frac{s}{\sqrt{n}}$$

If all goes well, a histogram of the z-scores should match well with the standard normal distribution. Here's a histogram of the 30000 z-scores along with the smooth standard normal distribution:



It looks like a good fit. This means that if we use a known population standard deviation, and the null hypothesis is true, then when we convert our observed mean to a z-score we can use the standard normal distribution (pnorm in R) to calculate our p-values.

But here's a histogram of the 't' scores along with the standard normal distribution:

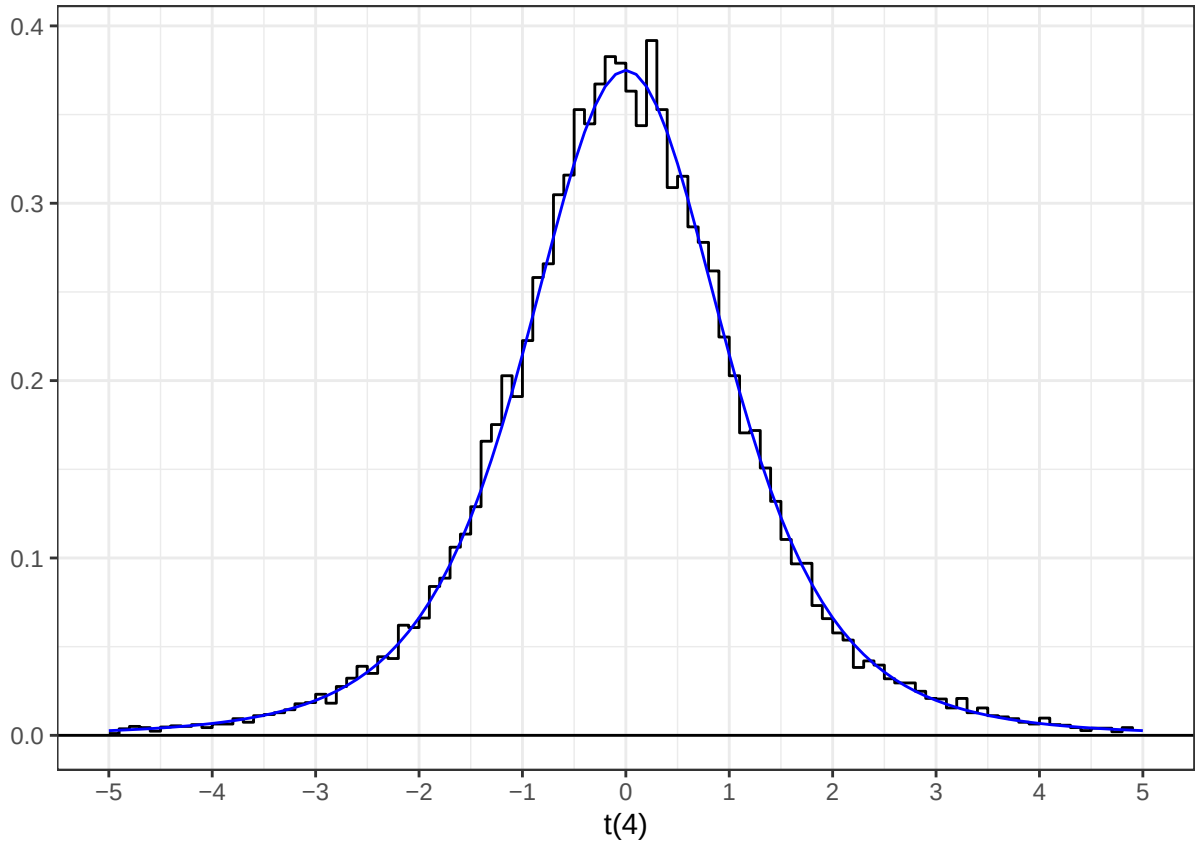


This is not a good fit. Notice the fat tails in the t-distribution compared to the standard normal, or z-distribution. This is because while the standard deviation of each sample may be an unbiased estimate of the population standard deviation, it is still a 'random variable', meaning that it varies from sample to sample. Sometimes, by chance, s_x will be small compared to σ_x so the t-statistic will end up large. If we were to use the standard normal distribution to calculate our p-values, we'd end up landing in the rejection zone way too often by accident.

7.3 The t-distribution

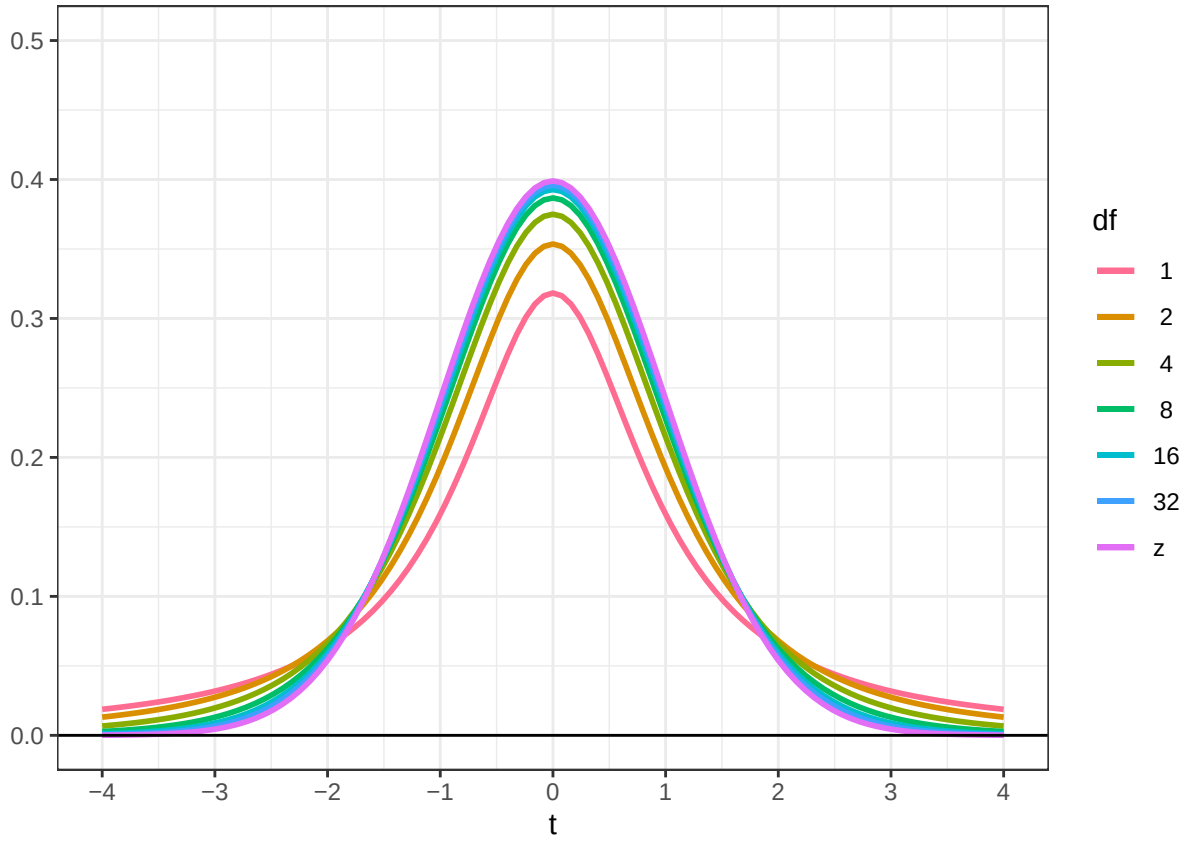
This distribution of 't' scores has a known shape. It's much like a normal distribution but it has longer tails. It also varies with sample size. As the sample size increases, the estimate of the population standard deviation gets more accurate (like a Central Limit Theorem for standard deviations). This makes the t-distribution look more like the z-distribution with larger sample sizes.

R's function for calculating areas or probabilities for the t-distribution is called 'pt' and it's inverse, 'qt', and have the same functionality as 'pnorm' and 'qnorm' for the normal distribution. The main difference is that 'pt' and 'qt' needs to know something about the sample size, or more specifically the 'degrees of freedom' which is the sample size minus one when we're dealing with one mean. Here's a histogram of our sampled t-scores and the known t-distribution for 5-1 = 4 degrees of freedom:



That's more like it. The 't(4)' x-axis label indicates that this is a t-distribution with 4 degrees of freedom.

Since the t-distribution depends on the sample size, the t-distribution is actually a 'family' of distributions, one for each sample size. Specifically, we define each separate t-distribution by its 'degrees of freedom', df, which for a single mean is df = n-1. Here are some example t-distributions:



You can see how the t-distribution starts out with fat tails for 4 degrees of freedom (df=4) and tightens up with increasing df. By df = 32 the t-distribution is nearly identical to the z-distribution.

You can also see this by comparing the area above, say 2, for z and various t-distributions using R:

```
1-pnorm(2)
```

```
## [1] 0.02275013
```

```
1-pt(2,c(4,8,16,32))
```

```
## [1] 0.05805826 0.04025812 0.03138598 0.02702409
```

About 2.3% of the area under the z-distribution lies above z=2, but 5.8% lies above the t-distribution for t=2 for df=4. But with df = 32, the area drops down to 2.7% which is getting close to the area for the z-distribution.

7.4 Example: Blood pressure and PTSD

Suppose you want to see if patients with PTSD have higher than normal systolic blood pressure. You sample 25 patients and obtain a sample mean Systolic BP of 124.22 and a sample standard deviation of 23.7527 mm Hg. Using an alpha value of 0.05 is this observed mean greater than a 'normal' Systolic BP of 120 mm Hg?

Like the z-test, we first compute the standard error of the mean, but using the sample standard deviation:

$s_{\bar{x}} = \frac{s_{\bar{x}}}{\sqrt{n}} = \frac{23.7527}{\sqrt{25}} = 4.7505$

We then calculate our t-statistic by subtracting the mean for the null hypothesis and divide by the estimated standard error of the mean:

$t = \frac{(\bar{x}-\mu_{H0})}{s_{\bar{x}}} = \frac{124.22-120}{4.7505} = 0.8876$

Since we're only rejecting H_0 for high values, this is a one-tailed test. We need to find the p-value, which is the probability of obtaining a score of 0.8876 or higher from the standard t-distribution. Since the sample size is 25, there are $df = 24$ degrees of freedom. Here's how to get this p-value:

```
1-pt(0.8876,24)
```

```
## [1] 0.1917825
```

Our p-value of 0.1918 is greater than $\alpha = 0.05$, so we fail to reject H_0 and conclude that our patients with PTSD do not have significantly higher than normal systolic blood pressure.

7.5 Example from the survey: Men's heights compared to 5'10"

Let's test the hypothesis that the average height of men in Psych 315 is different from an expected height of 70 inches (5'10") using a $\alpha = 0.05$. We'll do this with R in two ways.

7.5.1 By hand

First we'll conduct the test 'by hand' by calculating the t-statistic using the sample mean and standard deviation:

```
x <- na.omit(survey$height[survey$gender == "Man"])
n <- length(x)
df <- n-1
m <- mean(x)
sprintf('mean = %5.4f',m )
```

```
## [1] "mean = 70.3103"
```

```
s <- sd(x)
sprintf('standard devation = %5.4f',s)
```

```
## [1] "standard devation = 3.1521"
```

```
sem <- s/sqrt(n)
sprintf('standard error of the mean = %5.4f',sem)
```

```
## [1] "standard error of the mean = 0.5853"
```

```
t <- (m-70)/sem
sprintf('t(%d)=%5.4f',df,t)
```

```
## [1] "t(28)=0.5302"
```

The code above calculates that $n = 29$, $\bar{x} = 70.3103$, $s = 3.1521$ and $s_{\bar{x}} = 0.5853$.

From these values, the t-statistic was found to be:

$t = \frac{(\bar{x}-\mu_{H0})}{s_{\bar{x}}} = \frac{70.31-70}{0.5853} = 0.5302$

From t=0.5302 and df = 28 we can use R's 'tp' function to find our p-value. Note that we'll be rejecting H_0 if there is a significant *difference* between our mean and 70 inches, so this is a two-tailed test. That means we will be doubling the area under t-distribution beyond our observed value of t. Here's one way of doing it in R:

```
p.value <- 2*(1-(pt(abs(t),df)))
sprintf('p = %5.4f',p.value)
```

```
## [1] "p = 0.6002"
```

Let's unpack this working from the inside out. First we take the absolute value of t since with a two-tailed test we really only care about how far t is away from zero. This way, '1-(abs(t),df)' finds the area above the absolute value of t. We then double this area to find the p-value for a two-tailed test.

Our p-value of 0.6002 is greater than $\alpha = 0.05$, so we fail to reject H_0 and conclude that the average height of men is not significantly different from 70 inches.

7.5.2 Using t.test

Now we introduce our first function in R for conducting a hypothesis test using actual data (and not just summary statistics). R's 't.test' function goes through the same steps that we just did by hand. 't.test' is also used to compare two means, which we'll cover in the next chapter.

If you use R's help function (using, at the Console, '?'followed by 't.test') to investigate how to use t.test you'll see:

t.test(x, y = NULL, alternative = c("two.sided", "less", "greater"), mu = 0, paired = FALSE, var.equal = FALSE, conf.level = 0.95, ...)

The first argument sent into t.test,'x', is a vector containing the data, which is the minimum amount of information needed. The rest are set to defaults which you can set. The second argument 'y' is a second vector for a 'two-sample' t.test. When used, t.test will run a hypothesis test comparing the means of x and y. 'alternative' is used to run either a one or a two-tailed test (the default), and "less" vs. "greater" tells the function which direction to reject H_0. 'mu' is the mean for the null hypothesis - the value that will be compared to our sample mean. 'paired' and 'var.equal' are for two-sample t.tests which we'll cover in the next chapter.

Here's how to run the test on Man heights using t.test:

```
x <- na.omit(survey$height[survey$gender == "Man"])
t.test(x,mu = 70,alternative ='two.sided')
```

```
##
##  One Sample t-test
##
## data:  x
## t = 0.5302, df = 28, p-value = 0.6002
## alternative hypothesis: true mean is not equal to 70
## 95 percent confidence interval:
##  69.11134 71.50935
## sample estimates:
## mean of x
##  70.31034
```

You should see some familiar numbers here that match the values we calculated by hand including the p-value. Hopefully you'll appreciate that the 't.test' function isn't doing something too mysterious and is instead doing something we've done by hand.

Running 't.test' spits out the results into your console. It's sometimes useful to have access to these values as variables so you don't have to copy them from the console. This can be done by sending the output of t.test into an 'object' like this:

```
out <- t.test(x,mu = 70,alternative ='two.sided')
```

I've used the variable name 'out' but you can call it anything you'd like. When you run this the results are stored into 'out' and nothing is dumped into the console. To investigate what's in there, you can type 'out$' at the console and you'll see a list of fields that have been filled in. For example, the p-value is found in:

```
out$p.value
```

```
## [1] 0.6001541
```

7.6 APA style

APA has a specific style for reporting the results of a t.test. The style includes the mean, standard deviation, degrees of freedom and p.value:

t(degrees of freedom) = the t statistic, p = p value

We can use 'sprintf' to convert the output of the t.test in 'out' into a string containing the results in APA style:

```
str <- sprintf('t(%d)=%5.4f, p = %5.4f',out$parameter,out$statistic,out$p.value)
str
```

```
## [1] "t(28)=0.5302, p = 0.6002"
```

To report the results in the context of the experiment, including the mean and standard deviation. To start, let's pull out the important numbers that can be found in 'out':

```
m <- out$estimate     # mean
t <- out$statistic    # t-value
p <- out$p.value      # p-value
df <- out$parameter   # degrees of freedom
sem <- out$stderr     # standard error of the mean
H0 <- out$null.value  # null hypothesis mean
```

For some reason, t.test doesn't give you the standard deviation of the sample (s_x), but it does give you the standard error of the mean ($s_{\bar{x}}$). It also doesn't give you the sample size, but it does give you the degrees of freedom. The sample size is just $n = df + 1$, and since $s_{\bar{x}} = \frac{s_x}{\sqrt{n}}$, $s_x = s_{\bar{x}}\sqrt{n}$

```
n <- df +1          # sample size
s <- sem*sqrt(n)    # sample standard deviation
```

From all this, we can generate a string containing our results:

```
str <- sprintf('The height of the men in Psych 315 (M = %5.4f, S = %5.4f) is not significantly different f
```

```
str
```

```
## [1] "The height of the men in Psych 315 (M = 70.3103, S = 3.1521) is not significantly different from 7
```

Chapter 8

Confidence Intervals and Effect Size

The last example in the last chapter we tested the hypothesis that the heights of the men in Psych 315 was significantly different from 70 inches. We calculated that $n = 29$, $\bar{x} = 70.3103$, $s = 3.1521$ and $s_{\bar{x}} = 0.5853$.

From these values, the t-statistic was found to be:

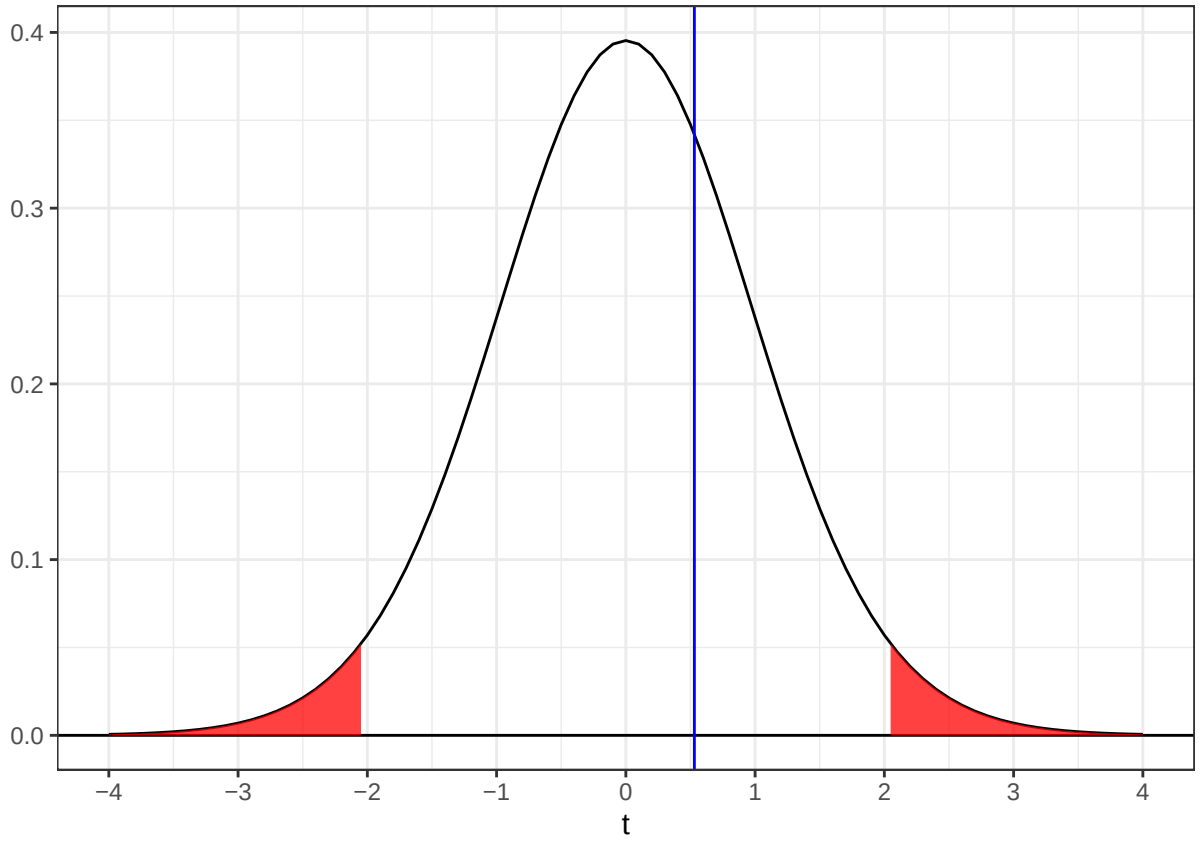
$t = \frac{(\bar{x} - \mu_{H0})}{s_{\bar{x}}} = \frac{70.31 - 70}{0.5853} = 0.5302$

and the p-value was:

$p = 0.6002$

The t-distribution, like the z-distribution, is the distribution of scores that you'd expect from an experiment if the null hypothesis is true. That's why it's centered around zero. On average your observed mean will be somewhere around the null hypothesis mean.

Let's look at where our t-statistic (t = 0.5302) falls compared to the rejection region (using a two-tailed test with $\alpha = 0.05$):

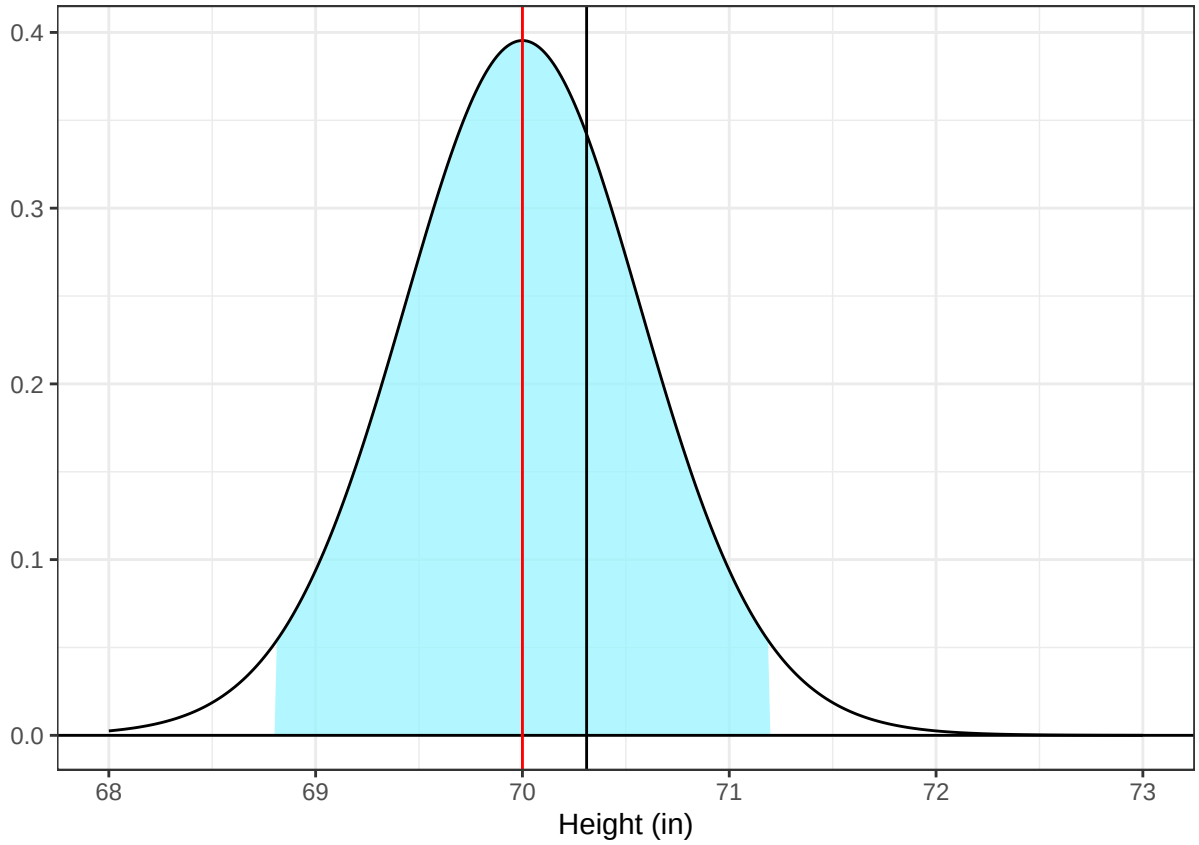


You can see why we failed to reject H_0. Our t-statistic (blue line) falls outside the rejection region (red tails).

With a little algebra we can rearrange the above equation to:

$$\bar{x} = \mu_{H_0} + s_{\bar{x}} \times t$$

This equation says that if the null hypothesis is true, then the expected distribution of means is shaped like a t-distribution scaled by $s_{\bar{x}} = 0.5853$ and centered at μ_{H_0}. Here's a plot of the distribution of means expected if the H_0 is true. Instead of shading the outer 5% in red, I've shaded the middle 95% in blue:

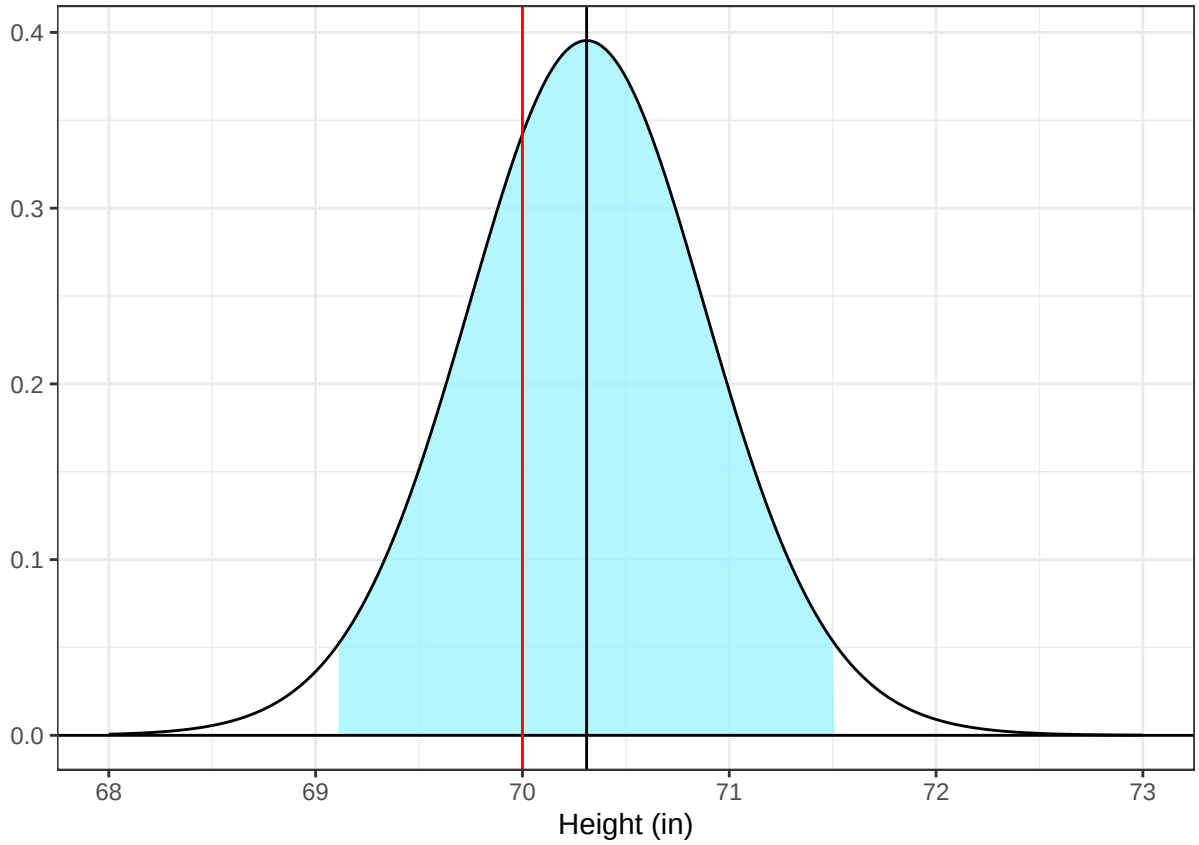


This looks just like the picture for the t-distribution, but shifted into our measured value of inches. Again you can see how the observed mean of 70.3103 falls with respect to the middle 95% of the distribution of means.

Now, let's forget about the null hypothesis. Instead, let's assume that our observed mean is the actual center of the population distribution. If we do this, then the expected distribution of means will look like the picture above, but will instead be centered at $\overline{x} = 70.3103$. The confidence interval can therefore be defines as:

$$CI = \bar{x} \pm s_{\bar{x}} \times t$$

This figure shows the 95% confidence interval for our example. It looks like the previous figure but with the scaled t-distribution now centered at $\bar{x}$. Again, the null hypothesis is shown as a vertical red line.



Assuming that $\overline{x}$ is the true population mean, then we expect the means to fall in the blue shaded region 95% of the time, so we call this region the *95% confidence interval*.

Notice that we can make the same decision about rejecting H_0 by comparing where H_0 falls with respect to the confidence interval. *If H_0 falls inside the confidence interval, we fail to reject H_0.*

Using R, we can calculate this interval by using qt to find the values of t covering the middle 95% and shifting and scaling these values by $s_{\bar{x}}$ and $\bar{x}$:

```
survey <- read.csv("http://www.courses.washington.edu/psy315/datasets/Psych315W21survey.csv")

alpha <- .05
x <- na.omit(survey$height[survey$gender == "Man"])
t.test.out <- t.test(x,mu = 70)
m <- mean(x)
n <- length(x)
sem <- sd(x)/sqrt(n)
df <- n-1
tRange <- qt(c(alpha/2,1-alpha/2),df)
CI <- m + sem*tRange
round(CI,2)
```

```
## [1] 69.11 71.51
```

This is our 95% confidence interval. It matches the values that came out of t.test which can be found in t.test.out$conf.int

```
t.test.out$conf.int
```

```
## [1] 69.11134 71.50935
## attr(,"conf.level")
## [1] 0.95
```

As we're encouraged to move away from p-values, journals more commonly asking for confidence intervals because they don't depend on a null hypothesis. But we can still use them to make our decision about rejecting H_0. Again, *we can therefore reject* H_0 if and only if *the null hypothesis mean falls outside the confidence interval.*

For this example 70 falls inside the confidence interval so we fail to reject H_0. You can see this in the plot above where the vertical line marking the null hypothesis mean (70) falls inside the blue shaded region.

A nice thing about confidence intervals compared to p-values and t-statistics is that they are in the units of our measurement (inches for our example). And, for this reason, they give us our critical value for rejecting H_0 in our measured units. We know that if your null hypothesis was either $H_0 : \mu = 69.11$ or $H_0 : \mu = 71.51$ - the lower and upper ranges of our confidence interval - our mean would be right on the border of rejecting, and therefore our p-value would be 0.05.

But, confidence intervals alone don't tell us the statistical significance of the test, which is why we're often asked to report both confidence intervals and p-values.

8.1 Effect Size

Remember, 'statistical significance', which is the p-value, is the probability that you'd obtain your observation given that H_0 is true. But even though the word 'significant' sounds like 'important', at statistically significant result may not be that interesting. Here's an example.

8.2 An Uninteresting Example

We'll go back to IQ's again so that we can use the simple z-test, but the idea holds for any hypothesis test.

Suppose you want to test the hypothesis that the IQ's of college students is different from 100. You go and sample the entire undergraduate population at UW which in 2021 had an enrollment of 36206 students and find that the mean IQ of UW undergraduates is 100.25. With our sample size, the standard error of the mean is $\frac{\sigma}{\sqrt{n}} = \frac{15}{\sqrt{36206}} = 0.0788$

Our z-score is therefore:

$$\frac{(\bar{x} - \mu)}{\sigma_{\bar{x}}} = \frac{(100.25 - 100)}{0.0788} = 3.1713$$

Using R's 'pnorm', our p-value for a two-tailed test is:

```
2*(1-pnorm(abs(3.1713)))
```

```
## [1] 0.001517583
```

We conclude that our mean of 100.25 is significantly different from 100 and that IQ's of college students is significantly different from the general population.

But how excited can you be about a $\frac{1}{4}$ of an IQ point difference? I don't think it's something you'd run out and tell your friends and send the result to Nature or something.

Of course, our results are statistically significant because our sample size is so large. To bend Archimedes, "Give me a sample size large enough and I can reject any H_0". Technically this is only true if H_0 is false, but it doesn't have to be *very* false.

So the problem with p-values is that they don't reflect the real size of the effect. The difference between the sample mean and the null hypothesis mean (100.25 - 100 = 0.25) is helpful, but it's hard to interpret because it doesn't take into account the variability in the population that we're sampling from.

We need something sort of in between the p-value and the difference between means. This is the *effect size*. It's defined as 'Cohen's d', which looks like the formula for a z or t-score, but has the standard deviation in the denominator instead of the standard error of the mean:

$$d = \frac{|\bar{x} - \mu_{hyp}|}{\sigma_x}$$

Or if you don't know σ:

$$d = \frac{|\bar{x} - \mu_{hyp}|}{s_x}$$

Cohen's d is the difference in the means in standard deviation units. For IQ's, a mean IQ of 115 would have an effect size of d=1, since 115 is one standard deviation above the mean. For our example:

$$d = \frac{|\bar{x} - \mu_{hyp}|}{\sigma_x} = \frac{|100.25 - 100|}{15} = 0.0167$$

Our difference in means is very small compared to the standard deviation of the population.

For the social sciences, almost all textbooks define the size of Cohen's d to be:

Table 8.1: Cohen's d

small	.2
medium	.5
large	.8

These values are a rough guide. We're allowed to embellish - for our IQ example it's fair to say that we have a 'very small' or 'teeny weeny' effect size.

There are two main advantages effect size. The first is that it doesn't depend on the sample size (although it does get more reliable with increasing sample size), and second, it's in standardized units. This means that effect sizes can be compared across experiments that have different units and sample sizes.

8.3 Summary

We now have four ways of telling people about the size of our results. First, there's simply the difference between our observed mean and null hypothesis mean. Then there is the p-value, the confidence interval, and the effects size. Each has their own advantages and disadvantages. The following table summarizes the attributes of each:

Table 8.2:

	Statistical significance	Real units	Compare across experiments	Requires H_0	Decision about H_0
Difference of Means	No	Yes	No	Yes	No
p-value	Yes	No	No	Yes	Yes
Confidence Interval	Yes (sort of)	Yes	No	No	Yes
Effect Size	No	No	Yes	Yes	No

Note that 'Yes' or 'No' aren't necessarily 'Pros' or 'Cons'. For example, the attribute 'in real units' is useful because it's easily interpreted, but can't easily be used to compare across experiments.

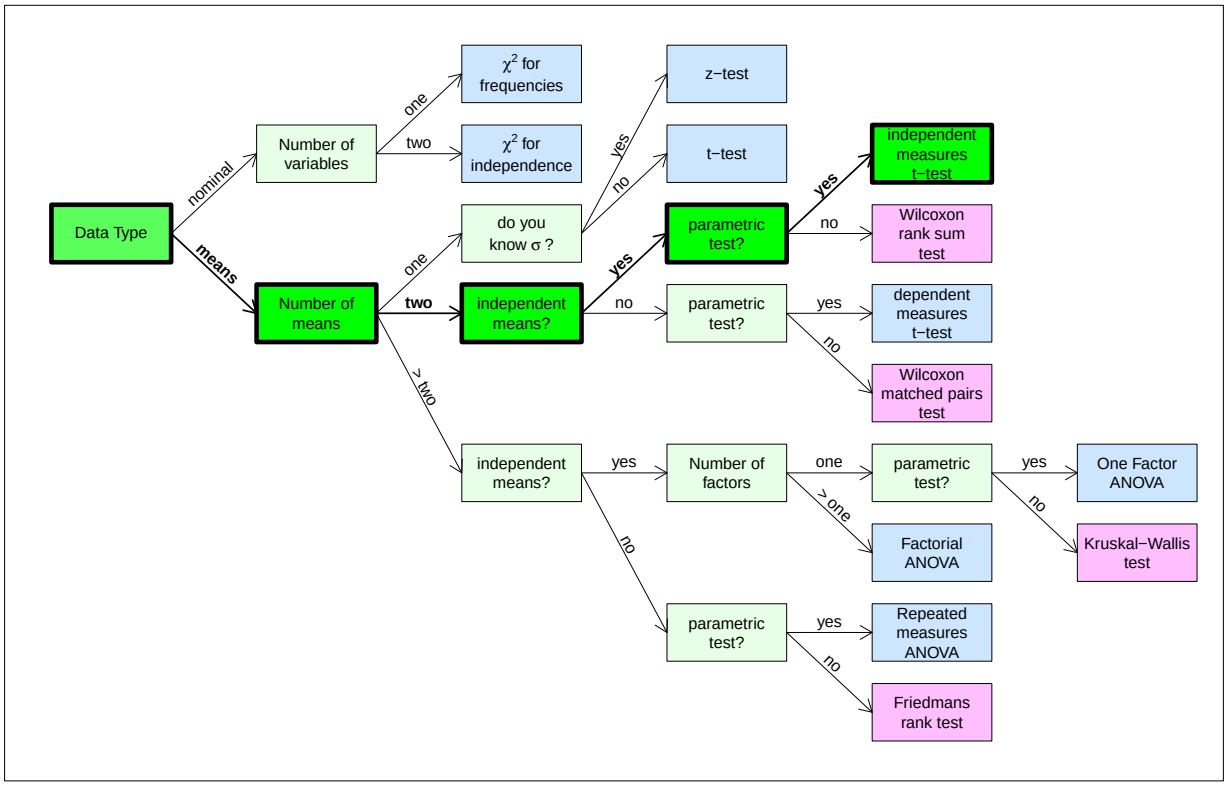
I've put 'sort of' for confidence intervals giving statistical significance because, as we discussed, you can see if your confidence interval includes the null hypothesis mean to make a binary decision, but it doesn't give you quantitative measure like the p-value.

Chapter 9

Two Sample Independent Measures t-test

This t-test compares two means that are drawn from separate, or independent samples.

You can find this test in the flow chart here:



The null hypothesis is that the two means are drawn from populations with the same mean (or means that differ by some fixed amount). Formally we write the null hypothesis as:

$$H_0 : (\mu_x - \mu_y) - \mu_{hyp} = 0$$

Where x and y are the two populations that our samples are drawn from. Formally, the two-sample t-test estimates the probability of obtaining a difference between the observed means (or some expected difference μ_{hyp}) if the null hypothesis is true. If this probability is small (less than α), then we reject the null hypothesis in support of the alternative hypothesis that the population means for the two samples are not the same (or differ by more than μ_{hyp}).

For the standard two-sample independent measures t-test, both the null and alternative hypotheses assume that the two population standard deviations are the same (even if we don't know what that standard deviation is). If we drop this assumption of 'equal variance' then we run a similar test called 'Welch's t-test', discussed later.

To conduct the test, we convert the two means and standard deviations into a test statistic which is drawn from a t-distribution under the null hypothesis:

$$t = \frac{\bar{x} - \bar{y} - \mu_{hyp}}{s_{\bar{x}-\bar{y}}}$$

Usually μ_{hyp} is zero, which is when we are simply testing if the two means are significantly different from each other. So the simple case when we are comparing means to each other the calculation for t simplifies to the more familiar:

$$t = \frac{\bar{x} - \bar{y}}{s_{\bar{x}-\bar{y}}}$$

The denominator is called $s_{\bar{x}-\bar{y}}$ because it represents the standard deviation of the difference between means, also called the *pooled standard error of the mean*. It is calculated by first calculating the *pooled standard deviation*:

$$s_p = \sqrt{\frac{(n_x - 1)s_x^2 + (n_y - 1)s_y^2}{(n_x - 1) + (n_y - 1)}}$$

s_p is a sort of complicated average of the two standard deviations. Technically, the calculation inside the square root is an average of the sample variances, weighted by their degrees of freedom. Importantly, s_p should always fall somewhere between the two sample standard deviations.

The denominator of the t-test, called the pooled standard error, is calculated from s_p by:

$$s_{\bar{x}-\bar{y}} = s_p \sqrt{\frac{1}{n_x} + \frac{1}{n_y}}$$

This is sort of like how we calculated the standard error from the sample standard deviation for the single sample t-test: $s_{\bar{x}} = \frac{s_x}{\sqrt{n}}$.

You can go straight to the pooled standard error of the mean in one step if you prefer:

$$s_{\bar{x}-\bar{y}} = \sqrt{\frac{(n_x - 1)s_x^2 + (n_y - 1)s_y^2}{(n_x - 1) + (n_y - 1)}(\frac{1}{n_x} + \frac{1}{n_y})}$$

Note: if the two sample sizes are the same ($s_x = s_y = n$), then the pooled standard error of the mean simplifies to:

$$s_{\bar{x}-\bar{y}} = \sqrt{\frac{s_x^2 + s_y^2}{n}}$$

The degrees of freedom of the independent means t-test is the sum of the degrees of freedom for each mean:

$$df = (n_x - 1) + (n_y - 1) = n_x + n_y - 2$$

9.1 Example 1: one-tailed test for independent means, equal sample sizes

Suppose you're a 315 Stats professor who has recently introduced a new textbook. You want to know if this new textbook are helping students learn. You do this by comparing Exam 2 scores from course taught this year with tutorials to the course taught last year without the textbook.

In this year, the 81 Exam 2 scores had a mean of 76.9345 and a standard deviation of 26.9491. The 81 Exam 2 scores in last year had a mean of 67.4122 and a standard deviation of 27.5612792. Let's run a hypothesis test to determine of the mean Exam 2 scores from this year is significantly greater than from last year. Use $\alpha = 0.05$.

First we calculate the pooled standard error of the mean. Since the sample sizes are the same ($n_x = n_y = 81$):

$$s_{\bar{x}-\bar{y}} = \sqrt{\frac{s_x^2 + s_y^2}{n}} = \sqrt{\frac{26.9491^2 + 27.5613^2}{81}} = 4.283$$

Our t-statistic is therefore:

$$t = \frac{\bar{x} - \bar{y}}{s_{\bar{x}-\bar{y}}} = \frac{76.9345 - 67.4122}{4.2830} = 2.2233$$

This is a one-tailed t-test with df $= 81 + 81 - 2 = 160$ and $\alpha = 0.05$. We can find our p-value using R's `pt` function:

```
## [1] "1-pt(2.2233,160)"
```

```
## [1] 0.0137983
```

Our p-value is less than $\alpha = 0.05$. We therefore reject H_0 and conclude that the mean Exam 2 scores from this year is significantly greater than from last year.

To state our conclusions using APA format, we'd state:

"The exam scores from this year for the Exam 2 scores (M = 76.9, SD = 26.9) are significantly greater than the exam scores from last year (M = 67.4, SD = 27.6), t(160) = 2.2, $p = 0.0138$."

9.2 Error Bars

For tests of independent means it's useful to plot our means as bars on a bar graph with error bars representing $\pm$ one standard error of the mean. We calculate each standard error for each mean the usual way by dividing the standard deviation by the square root of each sample size:

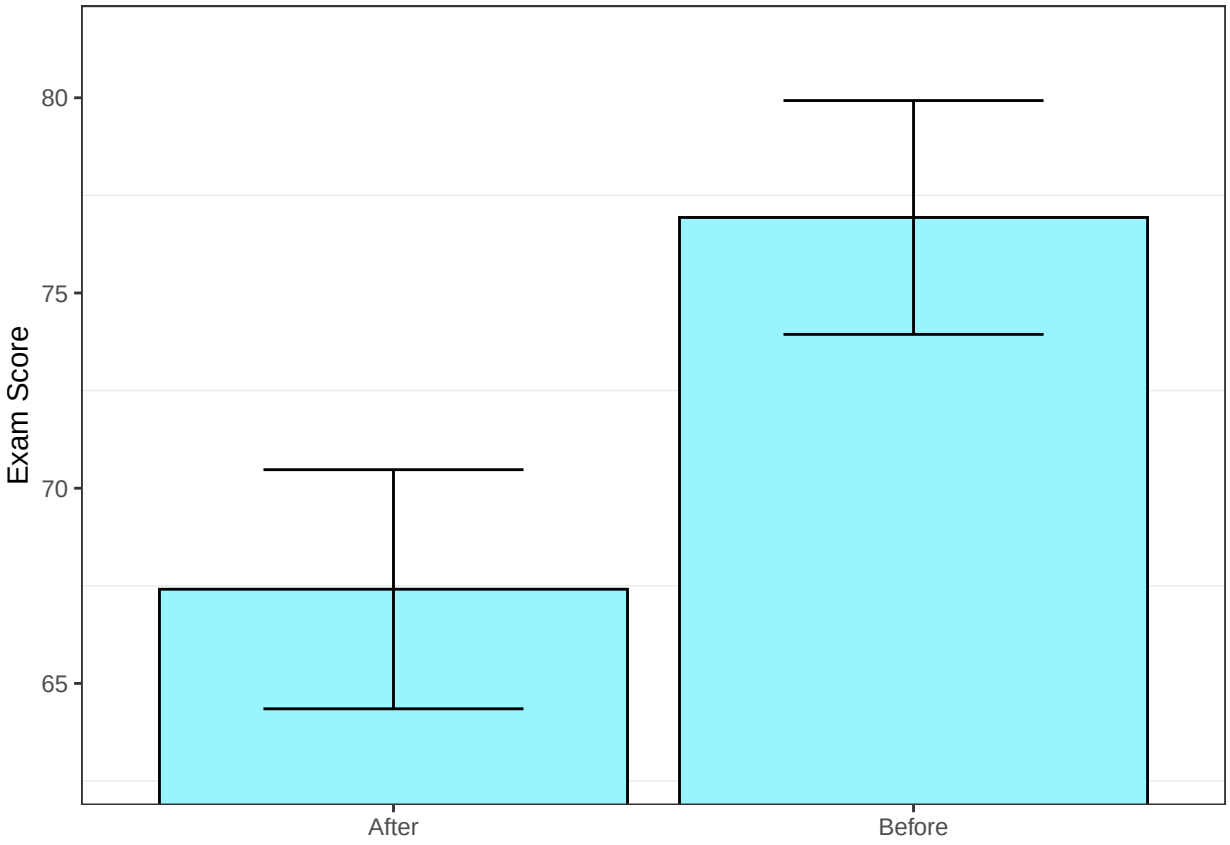
For this year,

$$s_{\bar{x}} = \frac{s_x}{\sqrt{n}} = \frac{26.9491}{\sqrt{81}} = 2.9943$$

and for last year,

$$s_{\bar{y}} = \frac{s_y}{\sqrt{n}} = \frac{27.5613}{\sqrt{81}} = 3.0624$$

The error bars are drawn by moving up and down one standard error of the mean ($s_{\bar{x}}$) for each mean ($\bar{x}$):



Bar graphs with error bars are useful for visualizing the significance of the difference between means.

Remember this rule of thumb: *If the error bars overlap, then a one-tailed t-test will fail to reject H_0 with $\alpha = .05$.* A one-tailed test with $\alpha = .05$ is the most liberal test, so you need a bigger gap between the error bars to reach significance for a two-tailed test, and/or for smaller values of α, like .01.

9.3 Example 2: Heights of women with tall and less tall mothers

Suppose you want to test the hypothesis that women with tall mothers are different than women with less tall mothers. We'll use our Psych 315 data and divide the students into women with mothers that are taller and shorter than the median of of 64 inches (5 feet 4 inches). The height of the 55 women with tall mothers has a mean of 65.9273 inches and a standard deviation of 2.5953 inches. The height of the 63 women with less tall mothers has a mean of 63.6349 inches and a standard deviation of 2.5546 inches. Are these heights significantly different? Use alpha = 0.01.

Since our sample sizes are different, we have to use the more complicated formula for the pooled standard error of the mean:

$$s_p = \sqrt{\frac{(n_x-1)s_x^2+(n_y-1)s_y^2}{(n_x-1)+(n_y-1)}} = \sqrt{\frac{(55-1)(2.5953)^2+(63-1)(2.5546)^2}{(55-1)+(63-1)}} = 2.5736$$

$$s_{\bar{x}-\bar{y}} = s_p\sqrt{\frac{1}{n_x}+\frac{1}{n_y}} = 2.5736\sqrt{\frac{1}{55}+\frac{1}{63}} = 0.4749$$

Our t-statistic is

$$t = \frac{\bar{x}-\bar{y}}{s_{\bar{x}-\bar{y}}} = \frac{65.9273-63.6349}{0.4749} = 4.8267$$

Using R's `pt` function, the p-value for this two-tailed test is:

```
## [1] "2*(1-pt(abs(4.8267),116))"
```

```
## [1] 4.268748e-06
```

Our p-value is less is than $\alpha = 0.01$. We therefore reject H_0 and conclude that the mean height height of the women with tall mothers height is statistically different from the heights of height of the women with less tall mothers height.

9.4 Using R's t.test from the data

This was all done by hand with the given means and standard deviations. Now we'll use R to load in the data set and conduct the test using R's `t.test` function. First we load in the survey data and pull out the heights of the mothers of the students that identify as women, which are in the field 'mheight'

```
survey <- read.csv("http://www.courses.washington.edu/psy315/datasets/Psych315W21survey.csv")
mheight <- survey$mheight[!is.na(survey$mheight) & survey$gender=="Woman" ]
```

Next, we'll find the median of the mother's height and divide the women's heights into two groups, the variable x will hold the heights of the women with mother's heights are greater than the median, and y will hold the heights of the women less than or equal to the median:

```
# Find the heights of the women students who's mother's aren't NA's:
height <- survey$height[!is.na(survey$mheight) & survey$gender=="Woman"]

# Find the heights of students that identify as women who's mothers are taller than the median.  Call them
x <- na.omit(height[mheight>median(mheight)])

# Find the heights of students that identify as women who's mother's heights are less than or equal to the
y <- na.omit(height[mheight<=median(mheight)])
```

Now we're ready for `t.test`. If you send in both x and y, `t.test` assumes that we're running a two-sample independent measures t-test. The `var.equal = TRUE` tells R to use the pooled standard deviation formula. If `var.equal = FALSE`, t.test runs the Welch Two Sample t-test I'll describe later.

```
out <- t.test(x,y,alternative = "two.sided",var.equal = TRUE,alpha = .01)
```

Here's where you can find the information needed to report your results in APA format:

```
sprintf('t(%g) = %4.2f, p = %5.7f',out$parameter,out$statistic,out$p.value)
```

```
## [1] "t(116) = 4.83, p = 0.0000043"
```

9.4.1 Alternate way to run t.test: formula and 'long format'

We ran the `t.test` function by inputting the two vectors, x and y as the fist two arguments. But often your data is not organized as two vectors like this. For example, if you're reading these two vectors from a csv file, the data will most likely be organized in 'long' format, where each student has a row, and a second column determines which category the student belongs to (e.g. 'tall mothers'). You can find the same data from the t-test above in a csv file organized in long format. Here's how to load it in:

```
heights.long <- read.csv("http://www.courses.washington.edu/psy524a/datasets/HeightsLongFormat.csv")
```

Here's what the data structure looks like:

Table 9.1:

mother	height
tall	65
tall	67
tall	69
tall	71
tall	66
tall	62
tall	67
tall	68
tall	67
tall	68
tall	66
tall	67
tall	63
tall	62
tall	61
tall	69
tall	68
tall	69
tall	68
tall	70
tall	62
tall	64
tall	62
tall	67
tall	68
tall	68
tall	62
tall	62
tall	68
tall	64
tall	67
tall	66
tall	66
tall	65
tall	66
tall	67
tall	64
tall	62
tall	67
tall	62
tall	68
tall	64
tall	65
tall	71
tall	68
tall	68
tall	63
tall	64

The data has two columns, one for what category the student is in (based on their mother's height: 'tall' and 'not tall') and the second column is the student's height. To run a t-test on data in long format we could pull out the numbers into two vectors, x and y. But an easier way is to define the test as a 'formula'.

What we're really doing with this t-test is trying to predict the student's heights based on their mother's heights, and measuring, with a p-value, the quality of this prediction. In R, we write this as `height ~ mother`, which reads as predicting the continuous variable 'height' from the categorical variable 'mother'. Here's how to run the t-test using this formula:

```
out <- t.test(height~mother ,data = heights.long,alternative = "two.sided",var.equal = TRUE,alpha = .01)
sprintf('t(%g) = %4.2f, p = %5.7f',out$parameter,out$statistic,out$p.value)
```

```
## [1] "t(116) = -4.83, p = 0.0000043"
```

You can verify that we get the same results as sending the data as 'x' and 'y' vectors. Later on when we get into regression and ANOVA we'll be using the formula method extensively.

9.5 Bar plots with Error Bars in R

To generate the bar plots, we'll use R's `ggplot` function from the 'ggplot2' package. The first step is to generate a 'data frame' that holds our statistics:

```
summary <- data.frame(
  mean = c(mean(x),mean(y)),
  n = c(length(x),length(y)),
  sd = c(sd(x),sd(y)),row.names = c('Tall','Less Tall'))
# use these to calculate the two standard errors of the mean
summary$sem <- summary$sd/sqrt(summary$n)
```

We've created a nice table of our summary statistics:

```
summary
```

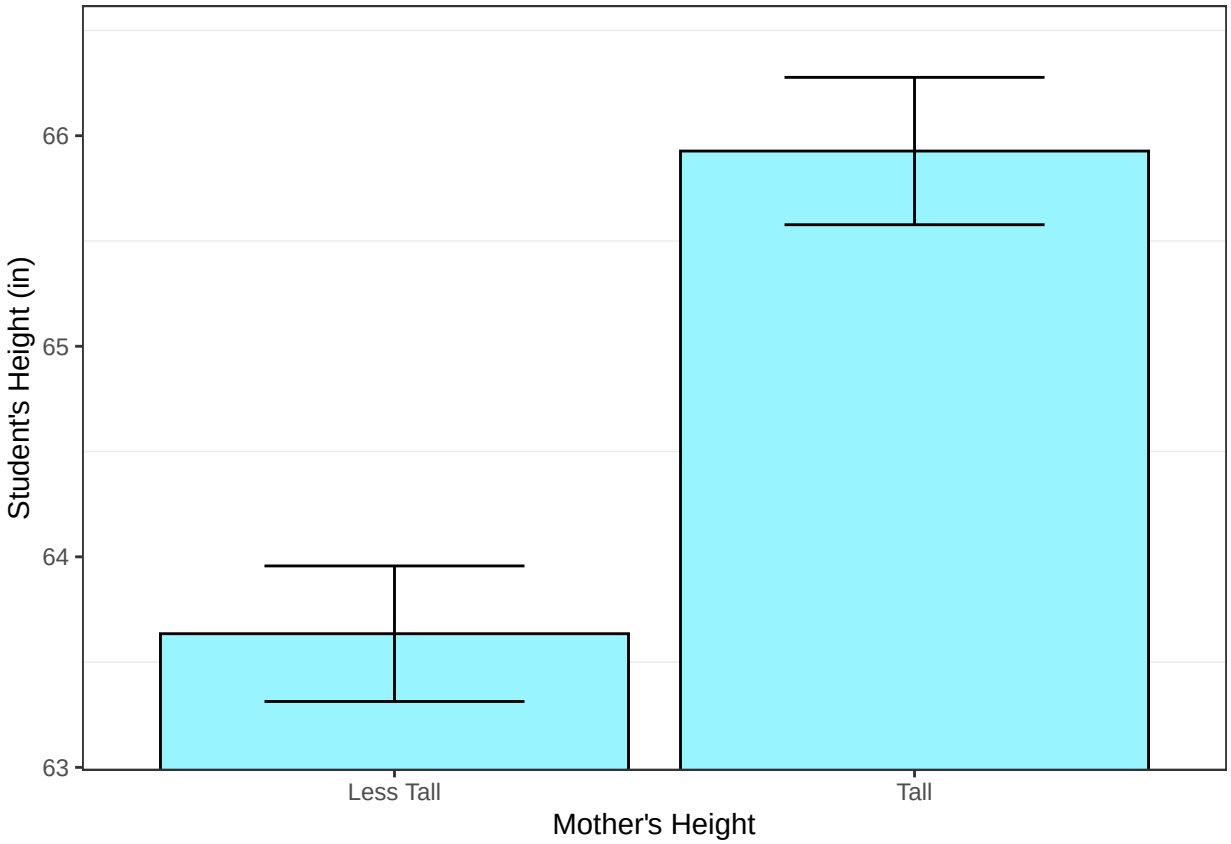
```
##               mean  n       sd       sem
## Tall      65.92727 55 2.595256 0.3499442
## Less Tall 63.63492 63 2.554576 0.3218463
```

This will define the y-axis limits for the bar graph. For most bar graphs with ratio scale measures we include zero, but bar graphs with error bars are an exception because we want to compare the differences between the means, not how far the means are from zero. The graph looks nice if we add about $\frac{1}{2}$ of a standard error above and below the error bars:

```
ylimit <-  c(min(summary$mean-1.5*summary$sem),
             max(summary$mean+1.5*summary$sem))
```

We're ready to call `ggplot`. `ggplot` is a very flexible function and takes a while to get used to. The first argument is the data frame that we created. The second defines what fields of the data frame correspond to the x and y-axis variables. `geom_col` tells `ggplot` that we're drawing a bar graph. `geom_errorbar` adds error bars of the specified lengths. I find it most useful to take existing `ggplot` code and editing them, rather than starting from scratch. So you can use this as a starting point for all of your bar graphs with error bars.

```
# Plot bar graph with error bar as one standard error (standard error of the mean/SEM)
ggplot(summary, aes(x = row.names(summary), y = mean)) +
  geom_col(position = position_dodge(), fill="cadetblue1",color = "black") +
  geom_errorbar(aes(ymin=mean-sem, ymax=mean+sem),width = .5) +
  xlab("Mother's Height") +
  theme_bw() +
  theme(panel.grid.major = element_blank()) +
  scale_y_continuous(name = "Student's Height (in)") +
  coord_cartesian(ylim=ylimit)
```



Notice the large gap between the error bars. Hence our small p-value.

9.6 Effect size

The effect size for an independent measures t-test is Cohen's d again. This time it's measured as:

$d = \frac{|\bar{x}-\bar{y}-(\mu_x-\mu_y)_{hyp}|}{s_p}$

Or more commonly when $(\mu_x - \mu_y)_{hyp} = 0$:

$d = \frac{|\bar{x}-\bar{y}|}{s_p}$

where, from above, the denominator is the pooled standard deviation:

$s_p = \sqrt{\frac{(n_x-1)s_x^2+(n_y-1)s_y^2}{(n_x-1)+(n_y-1)}}$

and for equal sizes:

$s_p = \sqrt{\frac{s_x^2+s_y^2}{2}}$

For the first example on test scores:

$d = \frac{|\bar{x}-\bar{y}|}{s_p} = \frac{|76.9345-67.4122|}{27.2569} = 0.3494$

This is a small effect size.

For the second example on student's heights:

$d = \frac{|\bar{x}-\bar{y}|}{s_p} = \frac{|65.9273-63.6349|}{2.5736} = 0.8907$

Which is considered to be a large effect size.

9.6.1 Calculating effect size using R's cohen's d

The package 'rstatix' provides some useful tools for calculating summary statistics, including cohen's d. You'll need to install the package using R's `install.packages` function, and then add the line `library(rstatix)` in your code. Once done, you can calculate Cohen's d from long-formatted data like this:

```
heights.long <- read.csv("http://www.courses.washington.edu/psy524a/datasets/HeightsLongFormat.csv")

cohen_out <- cohens_d(formula = height ~ mother,data = heights.long , var.equal = TRUE)
```

Note that the order of the arguments in `t.test` is 'formula' followed by 'data', but the order for Cohen's d is 'data' followed by 'formula'. This lack of standardization is a classic problem in open-source languages. So to be safe, you can always send in the arguments in your own order as long as you name them (e.g. `formula = height~mother`).

The field `effsize` holds Cohen's d, though it can be negative depending on the order of the nominal-scale values which are ordered alphabetically. ('not tall' comes before 'tall' in this example). So take the absolute value just in case:

```
cat(sprintf("Cohen's d: %0.4f",abs(cohen_out$effsize)))
```

```
## Cohen's d: 0.8907
```

9.7 Power for the two-sample independent measures t-test

Power calculations for the independent mean t-test are conceptually the same as for the t-test for one mean. It's still the probability of correctly rejecting H_0. Power for a two-sample independent measures t-test is easy to calculate in R using `power.t.test`. We just have to specify `type = 'two.sample'`. The only weird thing is that the sample size it needs is the average of the two sample sizes, not the total.

For our first example on exam scores, which was a one-sided test with an average sample size of 81, an effect size of d = 0.3494, and $\alpha = 0.05$, the observed power is found with:

```
power.out <- power.t.test(n=81,delta=0.3494,sig.level=0.05,type ='two.sample',alternative = 'one.sided')
```

```
power.out$power
```

```
## [1] 0.7154228
```

This means that with this observed effect size and sample size, there is about a 72% chance of rejecting H_0 for any given experiment.

For the second example, on women's heights, which was a two-sided test with an average sample size of 59, an effect size of d = 0.8907, and $\alpha = 0.01$, the observed power is found with:

```
power.out <- power.t.test(n=59,delta=0.8907,sig.level=0.01,type ='two.sample',alternative = 'two.sided')
```

```
power.out$power
```

```
## [1] 0.9858173
```

The set of power curves from the chapter on power include curves for independent means and can be found at:

http://courses.washington.edu/psy524a/pdf/PowerCurves.pdf

9.8 Welch's t-test for Unequal population variances

As described above, the pooled standard deviation, s_p, assumes that the variances of the two populations that we're drawing from are equal. But if the population variances are unequal, then the standard two-sample t-tests can lead to inflated type I errors.

If we want to drop this assumption we have to resort to an alternate version of the two-sample t-test most commonly called 'Welch's t-test'.

The logic is the same as the standard two-sample t-test, but there are two changes. First, the denominator of the t-test changes to:

$s_{\bar{x}-\bar{y}} = \sqrt{\frac{s_x^2}{n_x} + \frac{s_y^2}{n_y}}$

Which is actually simpler than for the regular t-test. The second change is to 'adjust' the degrees of freedom to:

$$df = \frac{\left(\frac{s_x^2}{n_x} + \frac{s_y^2}{n_y}\right)^2}{\frac{\left(\frac{s_x^2}{n_x}\right)^2}{n_x - 1} + \frac{\left(\frac{s_y^2}{n_y}\right)^2}{n_y - 1}}$$

From our second example about womens' heights:

$$df = \frac{\left(\frac{s_x^2}{n_x} + \frac{s_y^2}{n_y}\right)^2}{\frac{\left(\frac{s_x^2}{n_x}\right)^2}{n_x - 1} + \frac{\left(\frac{s_y^2}{n_y}\right)^2}{n_y - 1}} = \frac{\left(\frac{2.5953^2}{55} + \frac{2.5546^2}{63}\right)^2}{\frac{\left(\frac{2.5953^2}{55}\right)^2}{55-1} + \frac{\left(\frac{2.5546^2}{63}\right)^2}{63-1}} = 113.3523$$

The degrees of freedom for the Welch test isn't usually a whole number, which is weird. But R and other software can deal with calculating p-values for df's that are not integers.

The new t-statistic is:

$t = \frac{\bar{x}-\bar{y}}{s_{\bar{x}-\bar{y}}} = \frac{65.9273-63.6349}{0.4754} = 4.8215$

The p-value for the Welch test is 0.00000446

Recall that treated with equal variance, the original df was 116, the original t-statistic was 4.8267 and the original p-value was 0.00000427. Not much of a change.

9.8.1 Effect size for the Welch Test

The denominator for Cohen's d for the regular two-sample independent measures t-test is the pooled standard deviation. For the Welch test we skipped the pooling step and went straight to the denominator of the t-test, $s_{x,y}$.

For the denominator of Welch's test we pool the standard deviations by taking the square root of the mean of the variances:

$$\sqrt{\frac{s_x^2 + s_y^2}{2}}$$

So for our example, Cohen's d for the Welch test is:

$$d = \frac{|\bar{x} - \bar{y}|}{\sqrt{\frac{s_x^2+s_y^2}{2}}} = \frac{|65.9273 - 63.6349|}{\sqrt{\frac{2.5953^2+2.5546^2}{2}}} = 0.8902$$

This is really close to the value if we assume equal variance d = 0.8907.

9.8.1.1 Effect size for the Welch test using R's 'cohens_d'

All we need to do is change `var.equal` from TRUE to FALSE (or just leave it out, since 'FALSE' is the default.)

```
cohen_out <- cohens_d(formula = height ~ mother,data = heights.long , var.equal = FALSE)

cat(sprintf("Cohen's d: %0.4f",abs(cohen_out$effsize)))
```

```
## Cohen's d: 0.8902
```

9.8.2 When to use the Welch test?

Statisticians will tell you to use the Welch test when you can't assume that the variances of the two populations are equal. But then they'll tell you that you shouldn't run a hypothesis test on the difference between the sample variances (called a 'Levene's' test) to determine if you should run a Welch test. Instead, you should only choose the Welch test ahead of time when you somehow know that the variances of the populations might not be equal.

Interestingly, in simulations with unequal variances, the Welch test corrects for the inflated type-I errors, but there is no loss in the power when there is a true difference between the population means.

This seems like something-for-nothing, so real question is when should we *not* use the Welch test? Well, current opinion seems to be moving toward dropping the regular two-sample t-test and always using the Welch test. In future years I'll probably drop the standard t-test and teach with the Welch test entirely. But for now when I ask you to conduct an independent measures t-test we'll use the standard t-test method.

For an interesting discussion on when to (or always) use the Welch test see:

http://daniellakens.blogspot.com/2015/01/always-use-welchs-t-test-instead-of.html

You can test out the effect of unequal variances yourself by playing around with the parameters in the R-script:

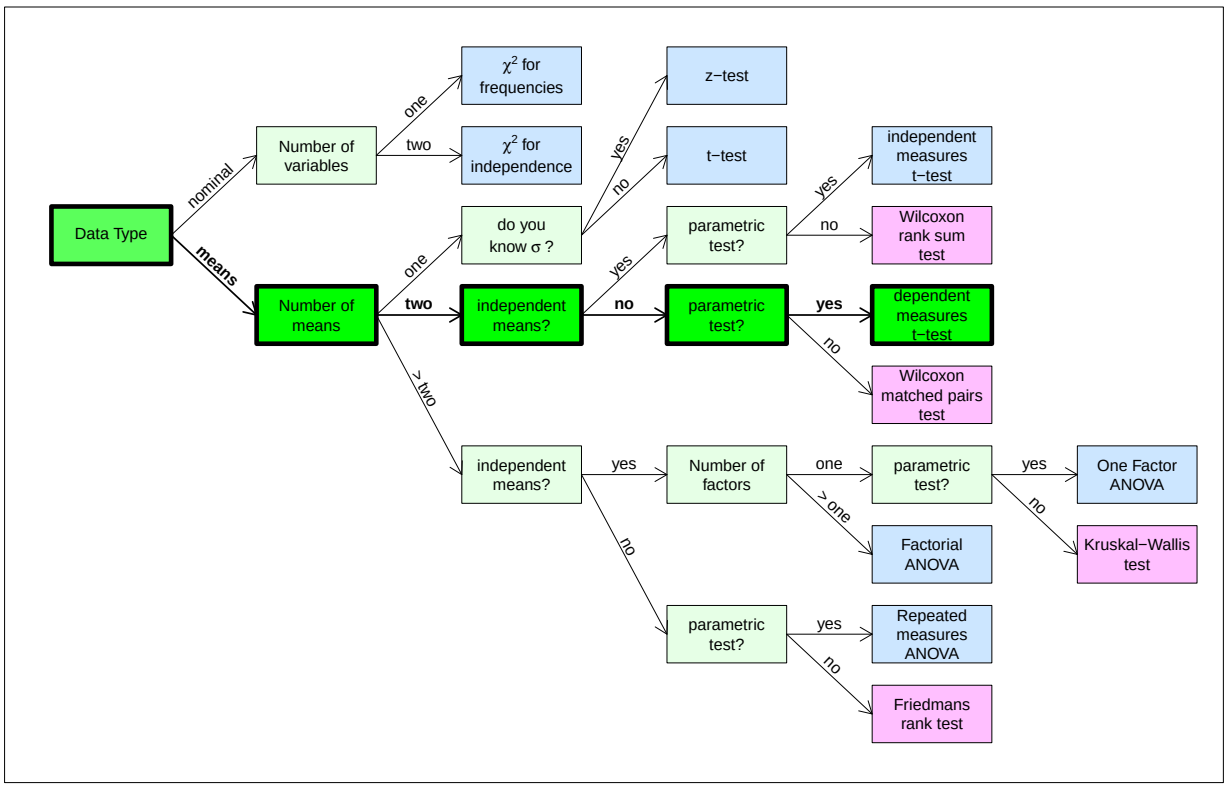
http://courses.washington.edu/psy524a/R/SimulatingPower.R

Chapter 10

Two Sample Dependent Measures t-test

This test is used to compare two means for two samples for which we have reason to believe are dependent or correlated. The most common example is a repeated-measure design where each subject is sampled twice- that's why this test is sometimes called a 'repeated measures t-test'.

You can find this test in the flow chart here:



Consider a weight-loss program where everyone lost exactly 10 pounds. Here's an example of weights before and after the program (in pounds) for 10 subjects:

Table 10.1:

Before	After
137	127
154	144
133	123
182	172
157	147
134	124
160	150
165	155
162	152
144	134

The mean weight before the program is 152.8 pounds and the mean after is 142.8 pounds. This should be considered a hugely successful program. But if you run an independent measures t-test on these two samples, you'd get a t-score of t(18) = 1.42, and p-value of p = 0.1725. You'd have to conclude that the program did not produce a significant change in weight.

But everyone lost 10 pounds! How could we not conclude that the weight loss program was effective? The problem is that there is a lot of variability in the weights across subjects. This variability ends up in the pooled standard deviation for the t-test.

But we don't care about the overall variability of the weights across subjects. *We only care about the change due to the weight-loss program.*

Experimental designs like this where we expect a correlation between measures are called 'dependent measures' designs. Most often they involve repeated measurements of the same subjects across conditions, so these designs are often called 'repeated measures' designs.

If you know how to run a t-test for one mean, then you know how to run a t-test for two dependent means. It's that easy.

The trick is to create a third variable, D, which is the pair-wise differences between corresponding scores in the two groups. You then simply run a t-test on the mean of these differences - usually to test if the mean of the differences, D, is different from zero.

10.1 High School vs. College GPAs

Let's go to the survey and see if there is a significant difference between the men students' high school and college GPAs. This is a repeated measures design because each student reported two numbers. We'll run a two-tailed test with $\alpha = 0.05$.

There are 28 students that identify as men in the class. The first step is to calculate the difference between the GPA's from high school and Uw for each student. Here's a table with a third column showing the differences which we'll call D:

An dependent measures t-test is done by simply running a t-test on that third column of differences. The mean of differences is $\overline{D} = 0.12$. The standard deviation of the differences is $S_D = 0.7013$.

You can verify that this mean of differences is the same as the difference of the means: the mean of the high school GPAs is 3.58 and the mean of the UW GPAs is 3.46. The difference between these two means is 3.5 - 3.46 = 0.12, which is the same as $\overline{D}$.

The standard error of the mean for D is:

Table 10.2:

HS	UW	D
3.40	3.23	0.17
4.00	3.65	0.35
3.95	3.80	0.15
3.85	3.66	0.19
3.65	3.30	0.35
3.87	3.68	0.19
2.90	3.83	-0.93
3.20	3.90	-0.70
3.70	3.43	0.27
4.60	2.60	2.00
3.70	3.80	-0.10
3.80	3.85	-0.05
3.00	3.00	0.00
3.83	3.35	0.48
3.80	3.31	0.49
3.60	2.00	1.60
3.89	3.84	0.05
4.00	3.91	0.09
2.18	2.89	-0.71
3.70	3.07	0.63
2.20	4.00	-1.80
3.95	3.66	0.29
3.88	3.53	0.35
3.50	3.51	-0.01
3.70	3.20	0.50
3.20	3.30	-0.10
3.20	3.65	-0.45
3.92	3.95	-0.03

$$s_{\overline{D}} = \frac{s_D}{\sqrt{n}} = \frac{0.7013}{\sqrt{28}} = 0.1325$$

Just like for a t-test for a single mean, we calculate our t-statistic by subtracting the mean for the null hypothesis and divide by the estimated standard error of the mean.

$$t = \frac{\overline{D}}{s_{\overline{D}}} = \frac{0.1168}{0.1325} = 0.8812$$

And since we're dealing with 28 pairs, our degrees of freedom is $df = n - 1 = 27$

Since this is a two-tailed test, we can use `pt` to find the p-value like this:

```
2*(1-pt(abs(0.8812),27))
```

```
## [1] 0.3859869
```

Our p-value is greater is than $\alpha = 0.05$. We therefore fail to reject H_0 and conclude that the high school GPA's are not statistically different from the college GPAs.

10.2 Effect size (d)

The effect size for the dependent measures t-test is just like that for the t-test for a single mean, except that it's done on the differences, D. Cohen's d is :

$$d = \frac{|\overline{D} - \mu_{hyp}|}{s_D}$$

For this example on GPA's

$$d = \frac{|\overline{D}|}{s_D} = \frac{|0.1168|}{0.7013} = 0.1665$$

This is considered to be a small effect size.

10.3 Power for the two-sample dependent measures t-test

Calculating power for the t-test with dependent means is just like calculating power for the single-sample t-test. For the power calculator, we just plug in our effect size, our sample size (size of each sample, or number of pairs), and alpha. For our example of an effect size of 0.1665273, sample size of 28, 28 and $\alpha = 0.05$, we can use `power.t.test`. For the dependent measures test we set `type = "paired"`.

```
power.out <- power.t.test(n=28,delta=0.1665,sig.level=0.05,type ="paired",alternative = "two.sided")
```

```
power.out$power
```

```
## [1] 0.1335137
```

Actually, since this is the same as a one-sample t-test, we get the same answer if we use `type = "one.sample"`:

```
power.out <- power.t.test(n=28,delta=0.1665,sig.level=0.05,type ="one.sample",alternative = "two.sided")
```

```
power.out$power
```

```
## [1] 0.1335137
```

To find out how many student's we'd need to get a power of 0.8 we use `n = NULL` and `power = 0.8`:

```
power.out <- power.t.test(n=NULL,delta=0.1665,power = 0.8,sig.level=0.05,type ="paired",alternative = "two
```

The answer is in `power.out$n`:

```
power.out$n
```

```
## [1] 285.0523
```

10.4 Example 2: Heights of Men in the Class vs. Their Father's Heights

Let's see if there's a statistically significant difference between the heights of the men students and their fathers. This time we'll go straight to the survey data and run the test using `t.test` rather than doing it by hand like the last example. This procedure is almost identical to what we used in for the t-test for independent means.

```
survey <- read.csv("http://www.courses.washington.edu/psy315/datasets/Psych315W21survey.csv")
x <- survey$height[survey$gender == "Man"]
y <- survey$pheight[survey$gender == "Man"]
```

We're ready for `t.test`. If you send in both x and y, `t.test`. To run a dependent measures t-test we use `paired = TRUE`.

```
out <- t.test(x,y,alternative = "two.sided",paired = T,alpha = .05)
```

Here's where you can find the information needed to report your results in APA format:

```
sprintf('t(%g) = %4.2f, p = %5.4f',out$parameter,out$statistic,out$p.value)
```

```
## [1] "t(25) = 1.60, p = 0.1219"
```

Our p-value is greater is than $\alpha = 0.05$. We therefore fail to reject H_0 and conclude that the heights of the men students are not statistically different from their father's heights.

Now that you know that a dependent measures t-test is the same as a one-sample t-test on the differences, see why we get the same result this way:

```
out <- t.test(x-y,alternative = "two.sided",alpha = .05)
sprintf('t(%g) = %4.2f, p = %5.4f',out$parameter,out$statistic,out$p.value)
```

```
## [1] "t(25) = 1.60, p = 0.1219"
```

One thing to note: There may be NA's in the data, but by default, `t.test` removes values that have a NA in rows of either x or y. If you want to be explicit, you can send in `na.action = "omit"`, which is the default.

Chapter 11

Power

11.1 Definition of power

If you retain anything from this class, it should be the following definition of power. I suggest you repeat this to yourself so this definition reflexively pops into your head whenever you read or hear about statistical power:

Power is the probability of correctly rejecting the null hypothesis.

The word 'correctly' is doing the heavy lifting here. If you correctly reject the null hypothesis, the null hypothesis must be false. So power is the probability of detecting an effect if it is truly out there.

In this chapter we'll define what this means in the context of a specific example: a z-test on the mean of a sample of IQ scores. This first part should be a review for you.

11.2 Z-test example

Suppose you want to test the hypothesis that the mean IQ of students at UW have a higher IQ than the population that has a mean IQ of 100 points. We also know that the standard deviation of the IQ's in the population is 15.

You measure the IQ in 9 UW students and obtain a mean IQ of 106. First we'll determine if this mean is significantly greater than 100 using $\alpha = 0.05$.

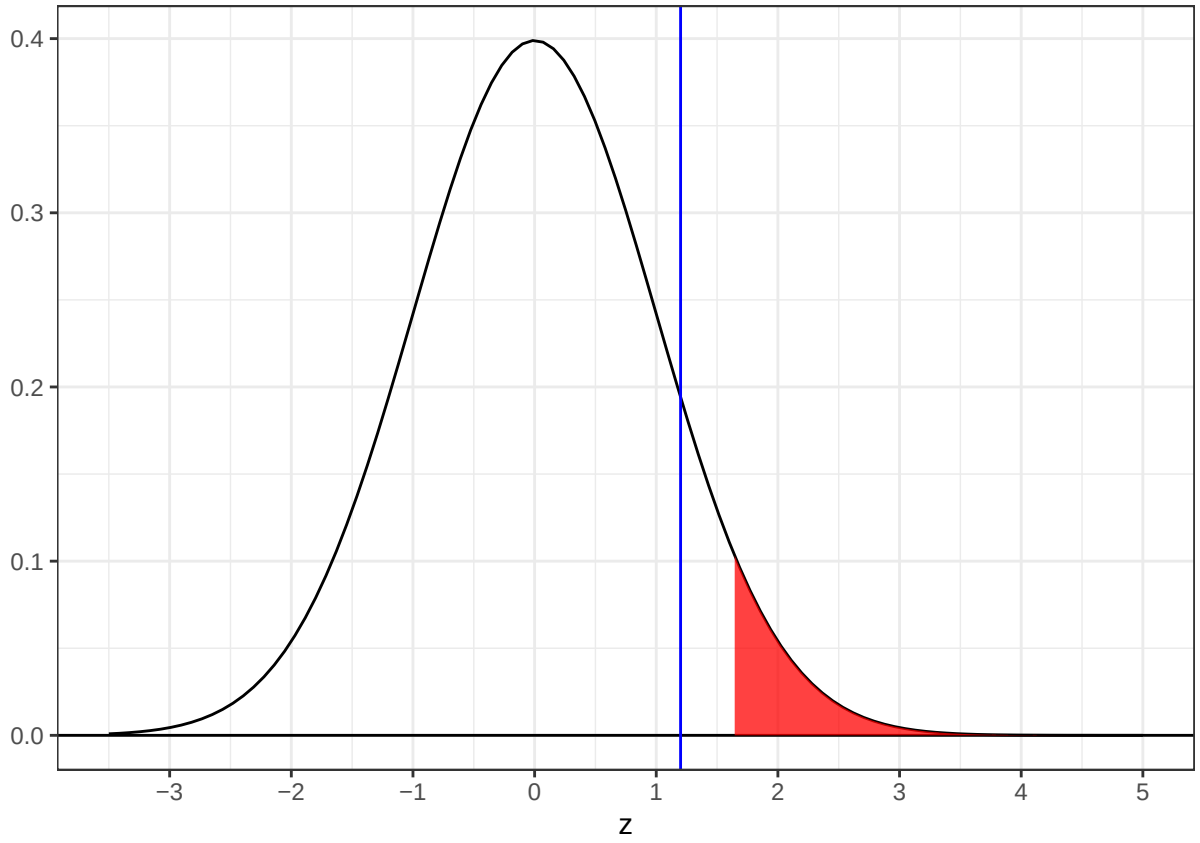
To conduct this test we need to calculate the standard error of the mean and then convert our observed mean into a z-score:

$$\sigma_{\bar{x}} = \frac{\sigma_x}{\sqrt{n}} = \frac{15}{\sqrt{9}} = 5$$

and

$$z = \frac{(\bar{x} - \mu_{H_0})}{\sigma_{\bar{x}}} = \frac{(106 - 100)}{5} = 1.2$$

With a one-tailed test and $\alpha = 0.05$, the critical value of z is $z_{crit} = 1.64$. Since our observed value of z=1.2 is less than the critical value of 1.64 we fail to reject H_0 and conclude that the mean IQ of UW students is not significantly greater than 100.



We never know if we're making a correct decision because we never actually know whether H_0 is true or false (if we did know this, we wouldn't need to run the experiment). If we happen to reject H_0 it could be that H_0 is true and we just happened to grab a sample with a high mean IQ. This kind of mistake, when we accidentally reject H_0 is called a *Type I Error*.

11.2.1 If the null hypothesis is true

α is the probability of rejecting H_0 if it is true. So α is the probability of a Type I error, which is 0.05 in this example.

If H_0 is true and we reject, then we've made a correct decision. Since we either reject or fail to reject H_0, the probability of failing to reject H_0 is $1 - \alpha = 1$ - $0.05 = 0.95$.

11.2.2 If the null hypothesis is false

Remember, if H_0 is false then the probability of rejecting H_0 is the power of our test. How do we find power?

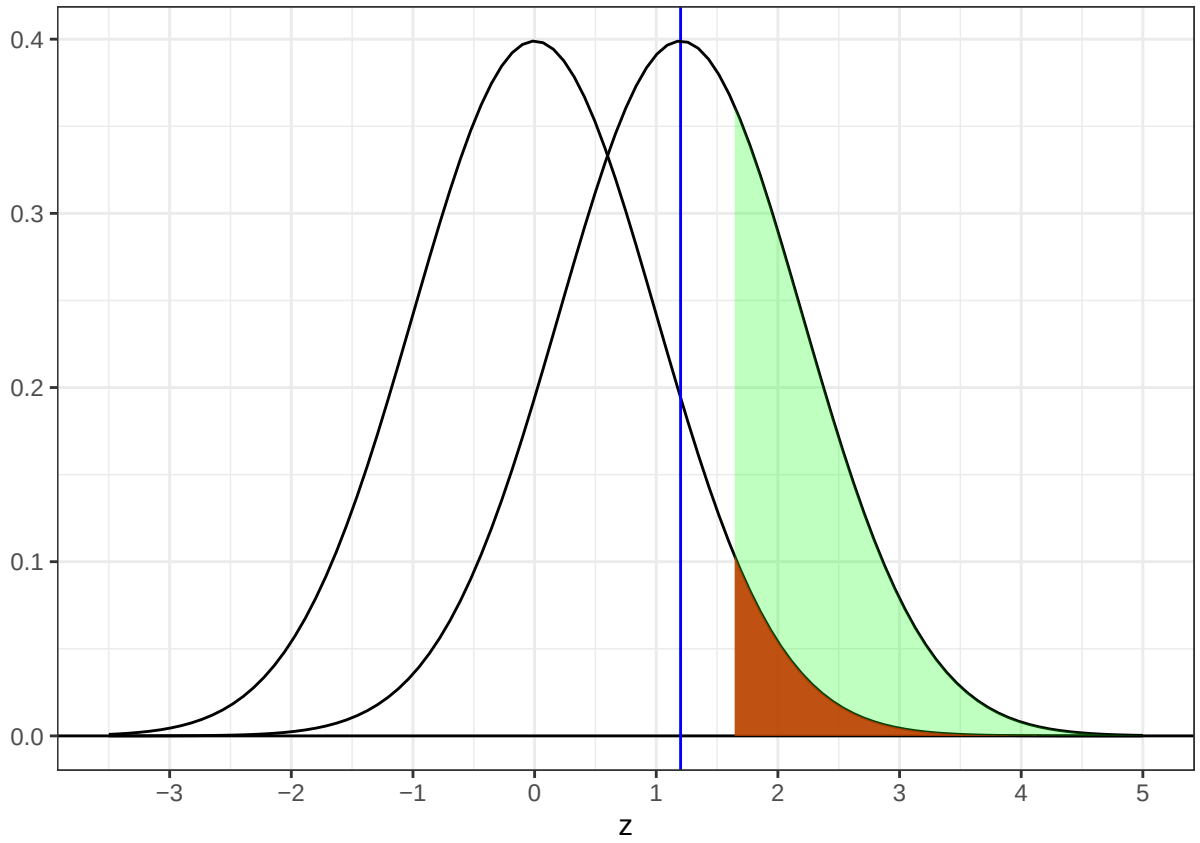
To calculate power we need to know the true mean of the population, which is different than the mean for H_0. You probably appreciate that this is weird. If we knew the true mean then we wouldn't have to run an experiment in the first place.

Since we don't know the true mean of the distribution we play a 'what if' game and calculate power for some hypothetical value for the true mean. For our example, let's say that the true mean is equal to the observed mean of $\bar{x} = 106$.

We just calculated that our observed mean of $\bar{x} = 106$ converts to a z-score of z = 1.2. So if we assume that the true mean IQ score is $\mu_{true} = 106$, then our z-scores will be drawn from a 'true' population with mean that I'll call z_{true}. We'll assume that the standard deviation σ_x is still 15:

$$z_{true} = \frac{(\mu_{true} - \mu_{H_0})}{\sigma_{\bar{x}}} = \frac{(\mu_{true} - \mu_{H_0})}{\frac{\sigma_x}{\sqrt{n}}} = \frac{(106 - 100)}{\frac{15}{\sqrt{9}}} = \frac{(106 - 100)}{5} = 1.2$$

Here's the true distribution drawn on top of the null hypothesis distribution:



Importantly, even though we're now drawing z-scores from the 'true' distribution our decision rule remains the same: if our observed value of z is greater than the critical value (z = 1.64 in this example), then we reject H_0. I've colored this region above z = 1.64 under the 'true' distribution in green. Note, the area above 1.64 also includes the red area.

It should be clear that *this green area (including the red area) is the power of our hypothesis test.* It's the probability of rejecting H_0 when H_0 is false. You can see that the green area (power) is greater than the red area (α). R's `pnorm` function will give you this area. It's the area above 1.64 for a normal distribution centered at 1.2 with a standard deviation of 1:

```
1-pnorm(1.64,1.2,1)
```

```
## [1] 0.3299686
```

So what does a power of 0.33 mean? It means that if the true mean of the population had an IQ of 106, then with our sample size of 9 and α=0.05 , the probability that we'd draw a mean is significantly greater than 100 is 0.33.

11.3 The 2x2 matrix of All Things That Can Happen

Out there in the world, the null hypothesis is either true or it is not true. To make a decision about the state of the world we run a hypothesis test and decide either to reject or to fail to reject the null hypothesis. Therefore, when we run a hypothesis test, there are four possible things that can happen. This can be described as a 2x2 matrix:

Table 11.1:

	H_0 True	H_0 False
Reject H_0	Error	Correct
Fail to Reject H_0	Correct	Error

Of the four things, two are correct outcomes (green) and two are errors (red). Consider the case where the null hypothesis is H_0 is true. We've made an error if we reject H_0. Rejecting H_0 when it's actually true is called a *Type I error*.

11.3.1 Pr(Type I error) = α

Type I errors happen when we accidentally reject H_0. This happens when we draw from the null hypothesis distribution, but happen to grab a mean that falls in the rejection region of the null hypothesis distribution. We know the probability of this happening because we've deliberately chosen it - it's α.

It follows that the probability of correctly rejecting H_0 is $1 - \alpha$.

11.3.2 Pr(Type II error) = β

Type II errors happen when we accidentally fail to reject H_0. This happens when we grab a mean from the 'true' distribution that happens to fall outside the rejection region of the null hypothesis distribution. We have a special greek letter for this probability, β ('beta').

11.3.3 power = 1-β

Remember, power is the probability of correctly rejecting H_0. This is the opposite of a Type II error, which is when we accidentally fail to reject H_0. Therefore, power = 1-Pr(Type II error) which is the same as power = $1 - \beta$.

We can summarize all of these probabilities in our 2x2 table of All Things that Can Happen:

Table 11.2:

	H_0 True	H_0 False
Reject H_0	Type I Error (α)	Power (1-β)
Fail to Reject H_0	1-α	Type II Error (β)

For our example, we can fill in the probabilities for all four types of events:

Table 11.3:

	H_0 True	H_0 False
Reject H_0	0.05	0.33
Fail to Reject H_0	0.95	0.67

11.4 Things that affect power

Figures like the one above with overlapping null and true distributions show up in all textbook discussion of power. Google 'statistical power' and check out the associated images. It's an ubiquitous image because it's very useful for visualizing how power is affected by various factors in your experimental design and your data.

Looking at the figure, you should see that power will increase if we separate the two distributions. There are a variety of ways this can happen.

11.4.1 Increasing effect size increases power

The most obvious way is for us to choose a larger value for the mean of the true distribution. In other words, if we have a larger effect size, we'll get a larger value for power.

The effect size in our current example on IQ's is

$$\frac{|\mu_{true} - \mu_{H_0}|}{\sigma} = \frac{|106 - 100|}{15} = 0.4$$

This is a medium effect size.

What if we think that the true distribution has a mean of $\mu_{true} = 112$ IQ points? This will increase the effect size to:

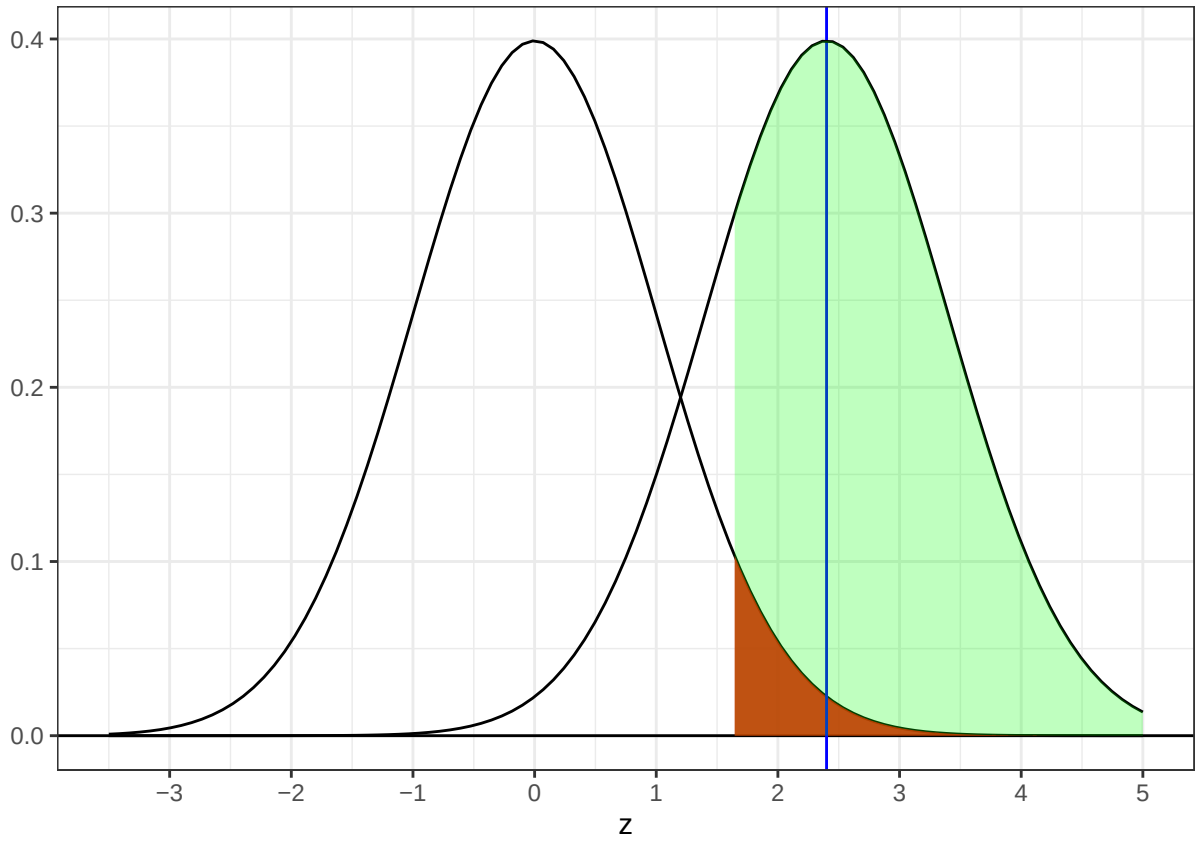
$$\frac{|\mu_{true} - \mu_{H_0}|}{\sigma} = \frac{|112 - 100|}{15} = 0.8$$

which is a large effect size.

Means drawn from this new true distribution will have z-scores that are normally distributed with a mean of

$$z_{true} = \frac{(\mu_{true} - \mu_{H_0})}{\sigma_{\bar{x}}} = \frac{(112 - 100)}{5} = 2.4$$

Here's that standard diagram for this new true distribution:



Notice what has changed and what has not changed. The true distribution is now shifted rightward to have a mean of $z_{true} = 2.4$. However, the critical region is unchanged - we still reject H_0 if our observed value of z is greater than 1.64.

Remember, power is the green area - the area under the true distribution above the critical value of z. You can now see how shifting the true distribution rightward increases power. The power of this hypothesis test is now:

```r
1-pnorm(1.64,2.4,1)
```

```
## [1] 0.7763727
```

Increasing the effect size from 0.4 to 0.8 increased the power from 0.33 to 0.7764.

11.4.2 Increasing sample size increases power

Another way to shift the true distribution away from the null distribution is to increase the sample size. For example, let's go back and assume that our true mean IQ is 106 points, but new let's increase our sample size from 9 to 36. The z-score for the mean of the true distribution is now:

$$z_{true} = \frac{(\mu_{true} - \mu_{H_0})}{\sigma_{\bar{x}}} = \frac{(\mu_{true} - \mu_{H_0})}{\frac{\sigma_x}{\sqrt{n}}} = \frac{(106 - 100)}{\frac{15}{\sqrt{36}}} = \frac{(106 - 100)}{2.5} = 2.4$$

This shift in z_{true} from 1.2 to 2.4 is the same as for the last example where μ_{true} was 112 IQ points and a sample size of 9. Thus, the power will also be 0.7749.

Note that increasing sample size increased the power without affecting the effect size (which doesn't depend on sample size).

effect size: $\frac{|\mu_{true} - \mu_{H_0}|}{\sigma} = \frac{|106-100|}{15} = 0.4$.

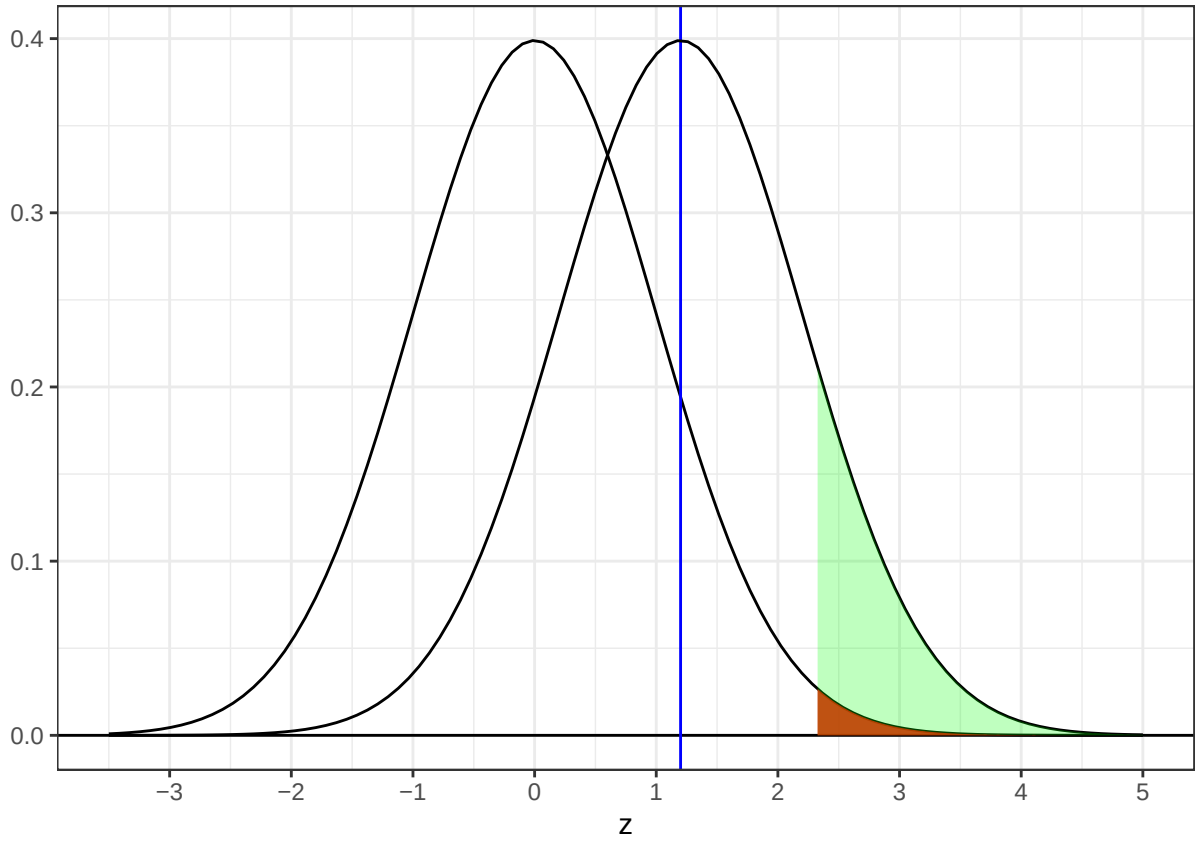
11.4.3 Decreasing α decreases power

A third thing that affects power is your choice of α. Recall that back in our original example with a mean of 9 IQ scores with $\mu_{true} = 106$ points, the power for $\alpha = 0.05$ was 0.33.

What if we keep the sample size at 9 so that effect size the same, but decrease our value of α to our second-favorite value, 0.01?

Decreasing α from 0.05 to 0.01 increases the critical value of z from 1.64 to 2.33.

Here's the new picture:



Notice that the true distribution still has a mean of $z_{true} = 1.2$. But shifting the critical value of z cuts into the greeen area. Power is now calculated as:

```
1-pnorm(2.33,1.2,1)
```

```
## [1] 0.1292381
```

The power decreased from 0.33 to 0.1292.

This illustrates a classic trade-off of Type I and Type II errors in decision making. Decreasing α by definition decreases the probability of making a Type I error. That is, decreasing α makes it harder to accidentally reject H_0. But that comes with the cost decreasing α also makes it harder to correctly reject H_0 if it was true. That's the same as a decrease in power. and a corresponding increase the probability of a Type II error (β), since power $= 1 - \beta$.

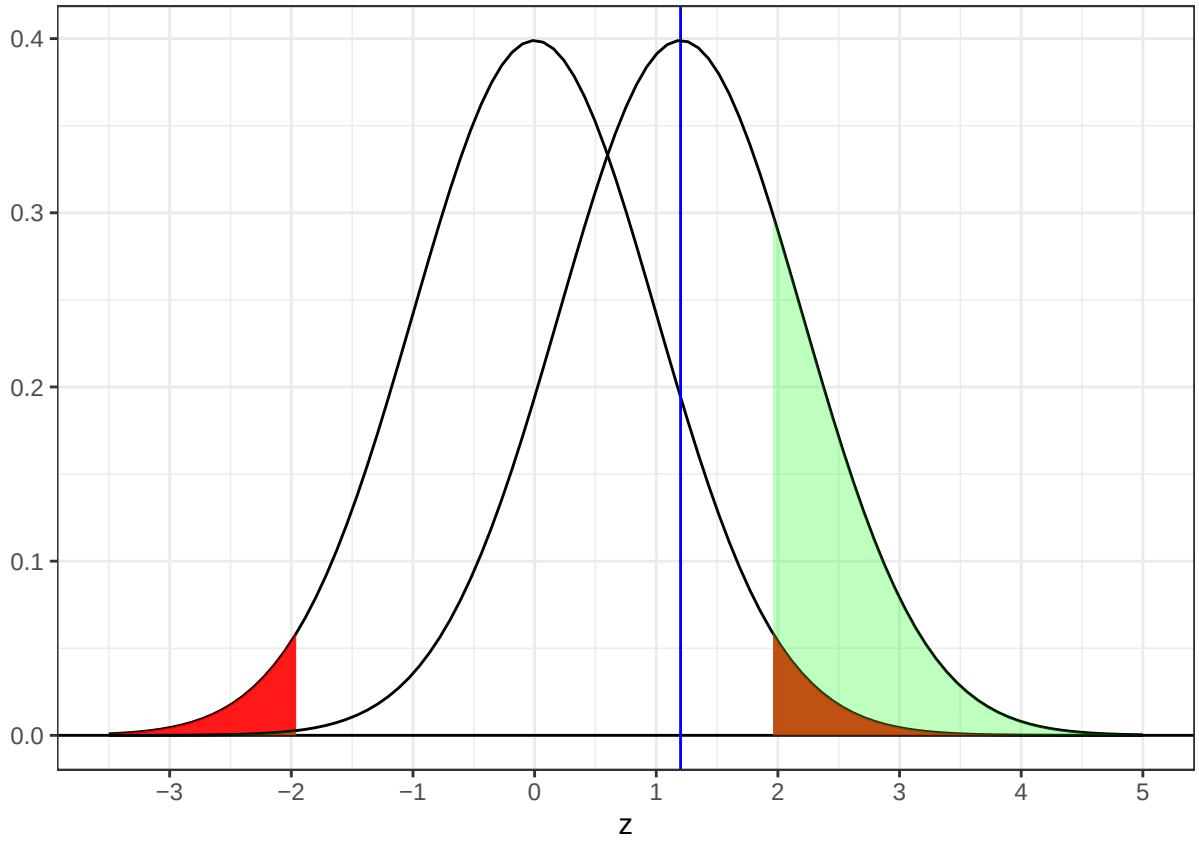
Get it? If you're still following things, when α goes down, z_{crit} goes up, power goes down and Pr(Type II error) goes up. **whew**

11.4.4 Power goes down with two-tailed tests

Let's look at our power calculation as we shift our original example from a one-tailed to a two-tailed test. Power calculations with two-tailed tests are a little more complicated because we have two rejection regions, but the concept is the same. We'll keep the true mean at 106 IQ points, the sample size n = 9 and $\alpha = 0.05$.

Recall that the critical value of z for a two-tailed test increases because we need to split the area in rejection region in half. You can therefore think of shifting to a two-tailed test as sort of like decreasing α, which as we know decreases power.

Here's the new picture:



Power is, as always, the area under the true distribution that falls in the rejection region. Now we have two rejection regions, one for z <1.96 and one for z>1.96. For a two-tailed test we need to add up two areas. For this example, the area in the negative region is so far below the true distribution that the area below z = -1.96 is very close to zero. This leaves the green area above z = 1.96.

The power of a two-tailed test is the sum of two areas:

```
pnorm(-1.96,1.2,1) + 1-pnorm(1.96,1.2,1)
```

```
## [1] 0.2244161
```

Our power is therefore 0.2244. This is lower than the power for the one-tailed test calculated above (0.33).

Here's a summary of how things affect power:

Table 11.4:

Thing	Pr(Type I error)	effect size	power
increasing effect size	same	increases	increases
increasing sample size	same	same	increases
decreasing alpha	decreases	same	decreases
two-tailed test	same	same	decreases

Go through the table and be sure to understand not only what happens with each thing, but also *why*.

11.5 Power for t-tests

This last example was using IQ's and therefore the z-distribution since we know the population standard deviation. Calculating power for t-tests is done conceptually the same way as described above, but with t-distributions instead of z-distributions. Since z's and t's look much the same, the pictures with the bell curves look almost identical and all the discussion about things that affect power are still true.

Fortunately, R has a function that does all of this calculation for you. It's called 'power.t.test'. Let's go back to PTSD and blood pressure example from the chapter on t-tests for one mean and discuss the power of that t-test.

11.5.1 PTSD and Blood Pressure Example

Recall that the test was to see if patients with PTSD have higher than normal systolic blood pressure. The sample size was 25 patients, the sample mean Systolic BP was 124.22 and the sample standard deviation was 23.7527 mm Hg. We used an alpha value of 0.05 to test if this observed mean was greater than a 'normal' Systolic BP of 120 mm Hg. We ended up failing to reject H_0 with a p-value of 0.1918. Using APA format we write:

(M = 124.22, S = 23.7527), t(24) = 0.89, p = 0.1918

We discussed above that to calculate power we need to know the 4 things that affect power: (1) the sample size, (2) α, (3) the effect size, and (4) whether it's a one or two tailed test.

This where things are a little weird. How do we know the effect size if we don't know the true mean of the population? After all, if we knew the true mean, we wouldn't have to run the experiment in the first place. One guess at the effect size is to use the effect size of our result. This is asking the question: 'What is the power of our test if the true mean is actually equal to the mean of our sample?'. When we use the effect size from our data we're calculating what's called 'observed power'. Let's calculate the observed power from our example. We first have to calculate the effect size from our results by hand:

$$d = \frac{|\overline{x} - \mu_{H0}|}{s_x} = \frac{124.22 - 120}{23.7527} = 0.1775$$

This is a small effect size.

Here's how to use 'power.t.test' find the observed power for this last example.

```
power.out <- power.t.test(n=25,delta=0.1775,sig.level=0.05,type ='one.sample',alternative = 'one.sided')
```

Most of the inputs are self explanatory. For some reason, pwer.t.test uses the name 'delta' for effect size. 'sig.level' is where we put in our value of α. The only new thing here is 'type' which is 'one.sample' for this example because this is a t-test for one mean. In the next chapter we'll comparing two means, so we'll use 'two.sample' for 'type' then.

The result is found in the field 'power'

```
power.out$power
```

```
## [1] 0.2170276
```

11.6 How much power do you need?

For some reason, the magic number for a desirable amount of power is 0.8. An experiment with a power of 0.8 means that there's an 80% chance of rejecting H_0 if H_0 is false.

For this last example, if we assume that the true mean is equal to our sample mean (124.2166 mm Hg) then with this sample size we only have a 22% chance of rejecting H_0. This is a pretty low amount of power.

If this is the true effect size, how could we modify the experiment to increase the power? All we really can do is increase the sample size.

power.t.test will calculate the sample size needed to obtain a certain level of power. This is done in a somewhat unconventional way. You enter all of the values that you know, and then enter 'NULL' for the value you want to find out. Here's how to find the sample size we'd need to get a power of 0.8:

```
power.out <- power.t.test(n=NULL,delta=0.1775,power = 0.8,sig.level=0.05,type ='one.sample',alternative =
```

We can find the sample size we'd need in the field 'n':

```
power.out$n
```

```
## [1] 197.5926
```

So we need to increase our sample size from 25 to 198 to get a power of 0.8.

Let's find out the effect size that we'd need (using our original sample size) to get a power of 0.8. This time we'll put the set 'delta = NULL':

```
power.out <- power.t.test(n=25,delta=NULL,power = 0.8,sig.level=0.05,type ='one.sample',alternative = 'one
```

```
power.out$delta
```

```
## [1] 0.5119381
```

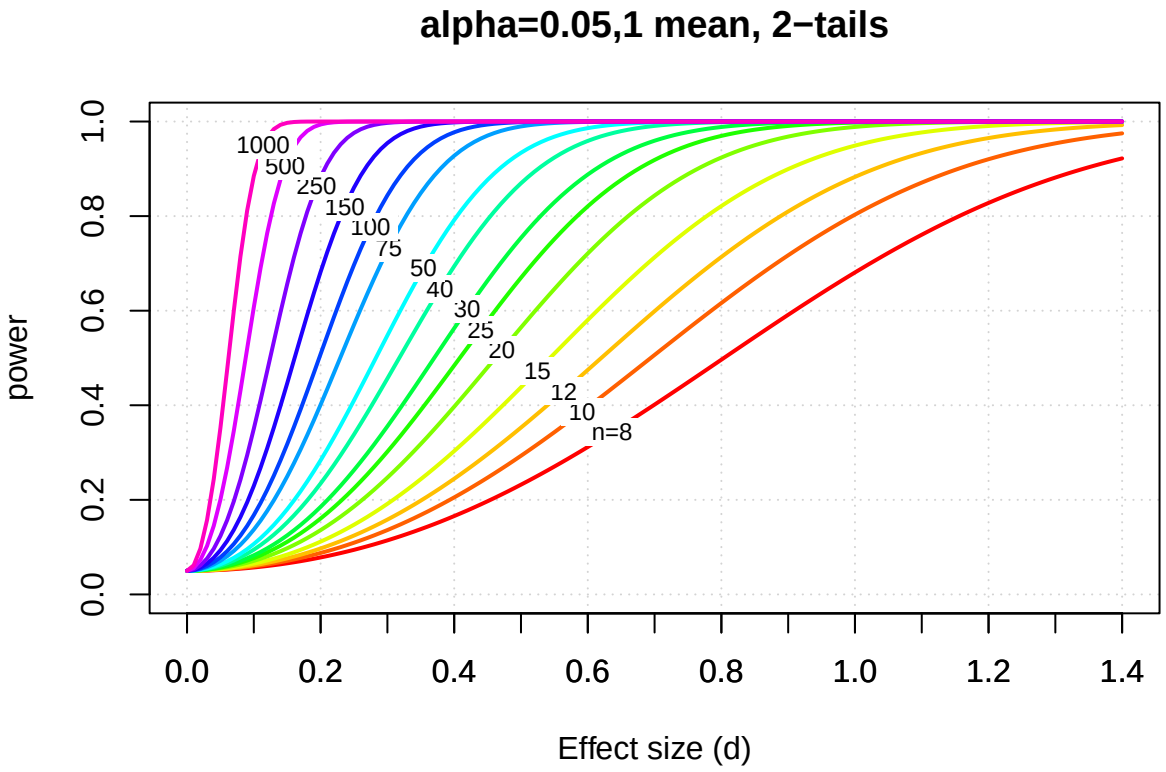
We'd have to increase our effect size from $d = 0.1775$ to a medium effect size of 0.5119 to get a power of 0.8.

If you wanted to, you could set 'alpha = NULL' to see what level of significance we'd need to get a power of 0.8. But that would be weird.

11.7 Power Curves

Finally, a nice way to visualize the things that affect power is with 'Power Curves', which are plots of power as a function of effect size, for different sample sizes.

Here's a set of power curves for a 2-tails test with $\alpha = 0.05$ and for 1 mean:



Power curves demonstrate the main things that affect power. First, you can see how power rises from left to right, as we increase the effect size. Second, you can see how power also rises with sample size.

They're also useful for estimating power without a computer in case your power goes out or something. For example, you can eyeball the curves and guess that if you'd like to design an experiment with a medium effect size of d=0.4 to have a power of 0.8, then you'd need to run about 50 subjects.

Can you see where all the power curves converge when the effect size is zero? That level is a $\alpha = 0.05$. This is because when the effect size is zero, the null hypothesis is true, and when the null hypothesis is true, the probability of rejecting H_0 is α.

The sets of power curves therefore depend on α. They also depend on whether you are comparing one or two means. A full set of power curves can be found in pdf format at:

http://courses.washington.edu/psy524a/pdf/PowerCurves.pdf

11.8 Estimating Power with Simulations

This is one of the most useful things you can do if you switch to a programming language to do your statistics.

Calculating power (the probability of correctly rejecting the null hypothesis) is usually done under very specific violations of the null hypothesis. For example, an independent measures t-test assumes that the population variances are equal and that only the means differ.

However, sometimes it's useful to calculate power for more complicated violations of the null hypothesis. In these cases there often is not a known distribution to draw p-values from.

A more general way to estimate power is to simulate data repeatedly under some specific assumption about the true distribution and calculate the proportion of times that the null hypothesis is rejected.

Here we'll do this for a two-tailed independent measures t-test.

First we'll define the population parameters

```
mu1 <- 100   # mean for group 1
mu2 <- 100   # mean for group 2

sigma1 <- 10  # s.d. for group 1
sigma2 <- 10  # s.d. for group 2

n1 <- 10  # sample size for group 1
n2 <- 10  # sample size for group 2

alpha <- .05
```

The great thing about simulations is that we know the state of the null hypothesis - we can make it true or false and see what happens. We're starting here with equal means and standard deviations, so H_0 is true.

This next chunk of code generates random data sets using `rnorm` based on the population parameters, runs a t-test on each set, and saves the p-value.

```
nReps <- 10000  # number of randomly sampled data sets and tests

p.values <- rep(NA,nReps)  # zero-out the vector that will hold the p-values

# start the loop
for (i in 1:nReps) {

  # generate a randomly sampled data set
  x1 <- rnorm(n1,mu1,sigma1)
  x2 <- rnorm(n2,mu2,sigma2)

  # run the hypothesis test
  out <- t.test(x1,x2,
                alternative = "two.sided",
                var.equal = F)

  # save the p-value for this iteration
  p.values[i] <- out$p.value
}
```

All we have to do now is calculate the proportion of times that our simulated p-values are less than $\alpha = 0.05$. This is the proportion of times that we're rejecting H_0:

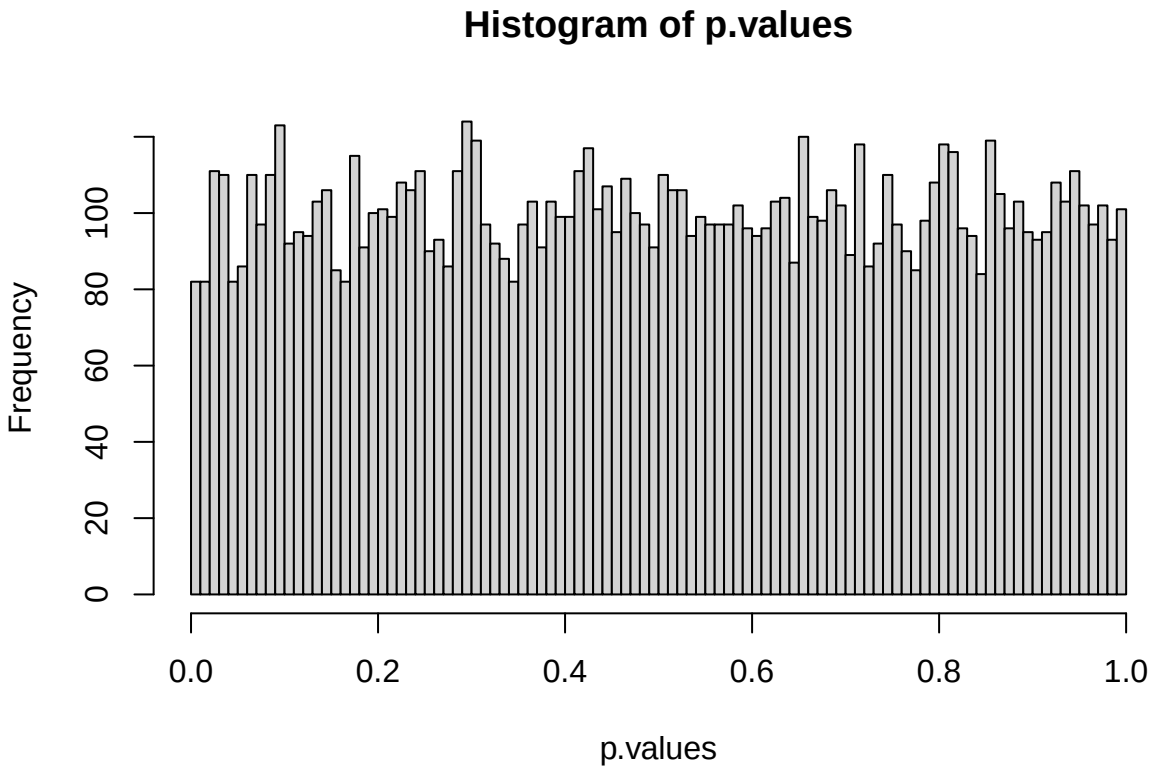
```
p.sim = sum(p.values<alpha)/nReps

sprintf('We rejected H0 for %g percent of the experiments with alpha = %g',p.sim*100,alpha)
```

```
## [1] "We rejected H0 for 4.67 percent of the experiments with alpha = 0.05"
```

Since we set H_0 to be true, hopefully this proportion is somewhere around $\alpha = 0.05$

Here's a histogram of the simulated p-values where H_0 is true:

```
hist(p.values,breaks = seq(0,1,by=.01))
```



It's flat! This is important: if H_0 is true, then the probability of obtaining any p-value between 0 and 1 is equally likely. It makes sense if you think about it. This is the only distribution for which the proportion of p-values falling below α is α.

Now lets' make H_0 be false. Here's the same script but we'll set `mu2 <- 110` instead of 100:

```
mu1 <- 100   # mean for group 1
mu2 <- 110   # mean for group 2

sigma1 <- 10  # s.d. for group 1
sigma2 <- 10  # s.d. for group 2

n1 <- 10  # sample size for group 1
n2 <- 10  # sample size for group 2

alpha <- .05

nReps <- 10000  # number of randomly sampled data sets and tests

p.values <- rep(NA,nReps)  # zero-out the vector that will hold the p-values

# start the loop
for (i in 1:nReps) {

  # generate a randomly sampled data set
  x1 <- rnorm(n1,mu1,sigma1)
  x2 <- rnorm(n2,mu2,sigma2)

  # run the hypothesis test
  out <- t.test(x1,x2,
```

```
                alternative = "two.sided",
                var.equal = F)

  # save the p-value for this iteration
  p.values[i] <- out$p.value
}
p.sim = sum(p.values<alpha)/nReps

sprintf('We rejected H0 for %g percent of the experiments with alpha = %g',p.sim*100,alpha)
```

```
## [1] "We rejected H0 for 55.81 percent of the experiments with alpha = 0.05"
```

Since H_0 is false, we're rejecting more often. You should appreciate that this number, 0.5581 is an estimation of the power of our hypothesis test for this particular violation of H_0.

We can compare our estimated power to the value we get from `power.t.test`. First we have to compute the effect size for our population parameters. We'll divide the difference between the μs by the calculated pooled standard deviation based on the population σ's:

```
# Calculate the pooled standard deviation
sigma.pooled <- sqrt( (sigma1^2*(n1-1)+sigma2^2*(n2-1))/(n1+n2-2))

# Calculate the effect size
d <- abs(mu1-mu2)/sigma.pooled

# Calculate power
out <- power.t.test(n = (n1+n2)/2,
                    d = d,
                    power = NULL,
                    sig.level =alpha,
                    alternative = "two.sided",
                    strict = T,
                    type ="two.sample" )

# Print out the simulated and calculated powers
sprintf('simulated power: %5.4f',p.sim)
```

```
## [1] "simulated power: 0.5581"
```

```
sprintf('observed power:  %5.4f',out$power)
```

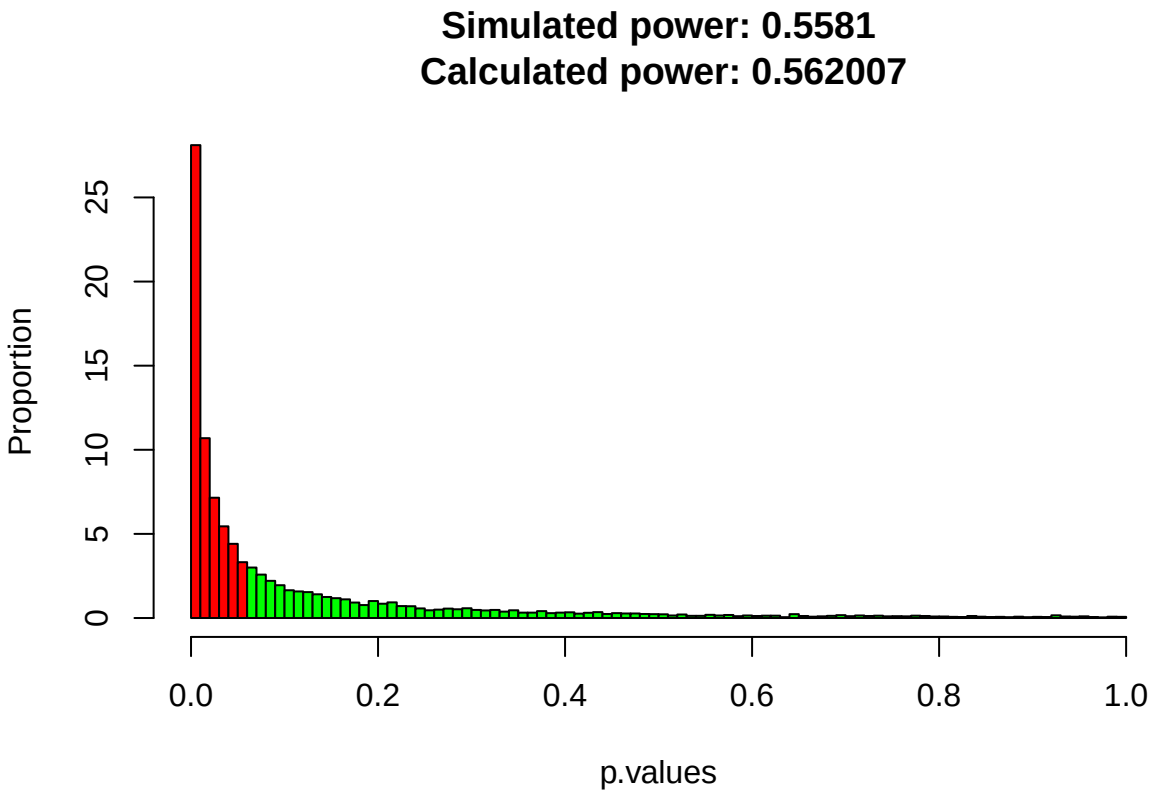
```
## [1] "observed power:  0.5620"
```

It's pretty close! If we increase the size of the simulations we'll get closer to the results from `power.t.test`.

Now let's look at a histogram of the p-values. We'll get fancy and color the range of the p-values red for when p<0.05

```
dx <- .01 # class interval width.
# Get the frequency distribution using 'hist'
h <- hist(p.values,
          breaks = seq(0,1,dx),
          plot = FALSE)

# Turn the frequencies into proportions
h$counts <- h$counts/(nReps*dx)

# Create a vector of colors for each bin
colors = rep("red",length(h$breaks))
colors[h$breaks>alpha] <- "green"
```

```
# Make a bar graph of proportions using 'colors' so p<alpha is red
plot(h,col=colors,
     ylab ="Proportion",
     main = sprintf('Simulated power: %g\n Calculated power: %g',p.sim,out$power))
```



Now the simulated p-values are bunched up to the left. That's because a higher proportion of p-values are now falling below α.

This distribution of p-values is sometimes called a 'p-curve'. We've learned here that the p-curve is flat when H_0 is true, and should bunch up toward small p-values when H_0 is false. Finally, the area under the p-curve below α is the power of our hypothesis test.

You can use the code above to run some interesting simulations. For example, what happens when you keep the population means the same but change the variances? What happens when you switch to a Welch t-test? What happens if you have radically different sample sizes? These questions can't be estimated with `power.t.test`.

Chapter 12

The Binomial Distribution

When you flip a coin there are only two possible outcomes - heads or tails. This is an example of a *dichotomous* event. Other examples are getting an answer right vs. wrong on a test, catching vs. missing a bus, or eating vs. not eating your vegetables. A roll of a dice, on other hand, is not a dichotomous event since there are six possible outcomes.

If you flip a coin repeatedly, say 10 times, and count up the number of heads, this number is drawn from what's called a *binomial distribution*. Other examples are counting the number of correct answers on an exam, or counting the number of days that your ten year old eats his vegetables at dinner. Importantly, each event has to be independent, so that the outcome of one event does not depend on the outcomes of other events in the sequence.

We can define a binomial distribution with three parameters:

P is the probability of a 'successful' event. That is the event type that you're counting up - like 'heads' or 'correct answers' or 'did eat vegetables'. For a coin flip, P = 0.5. For guessing on a 4-option multiple choice test, P = 1/4 = .25. For my twelve year old eating his vegetables, P = 0.05.

N is the number of repeated events.

k is the number of 'successful' events out of N.

The probability of obtaining k successful events out of N, with probability P is:

$$\frac{N!}{k!(N-k)!}P^k(1-P)^{N-k}$$

where N! = $N(N-1)(N-2)...$, or N 'factorial'.

12.1 Coin flip example

For example, if you flip a fair coin (P=0.5) 5 times, the probability of getting 2 heads is:

$$Pr(k=2) = \frac{5!}{2!(5-2)!}(0.5)^2(1-0.5)^{(5-2)} = (10)(0.5^2)(0.5)^3 = 0.3125$$

12.2 R's `dbinom` function

R has a function that works like: `dbinom(k,n,P)`. For our example it works like this:

```
dbinom(2,5,0.5)
```

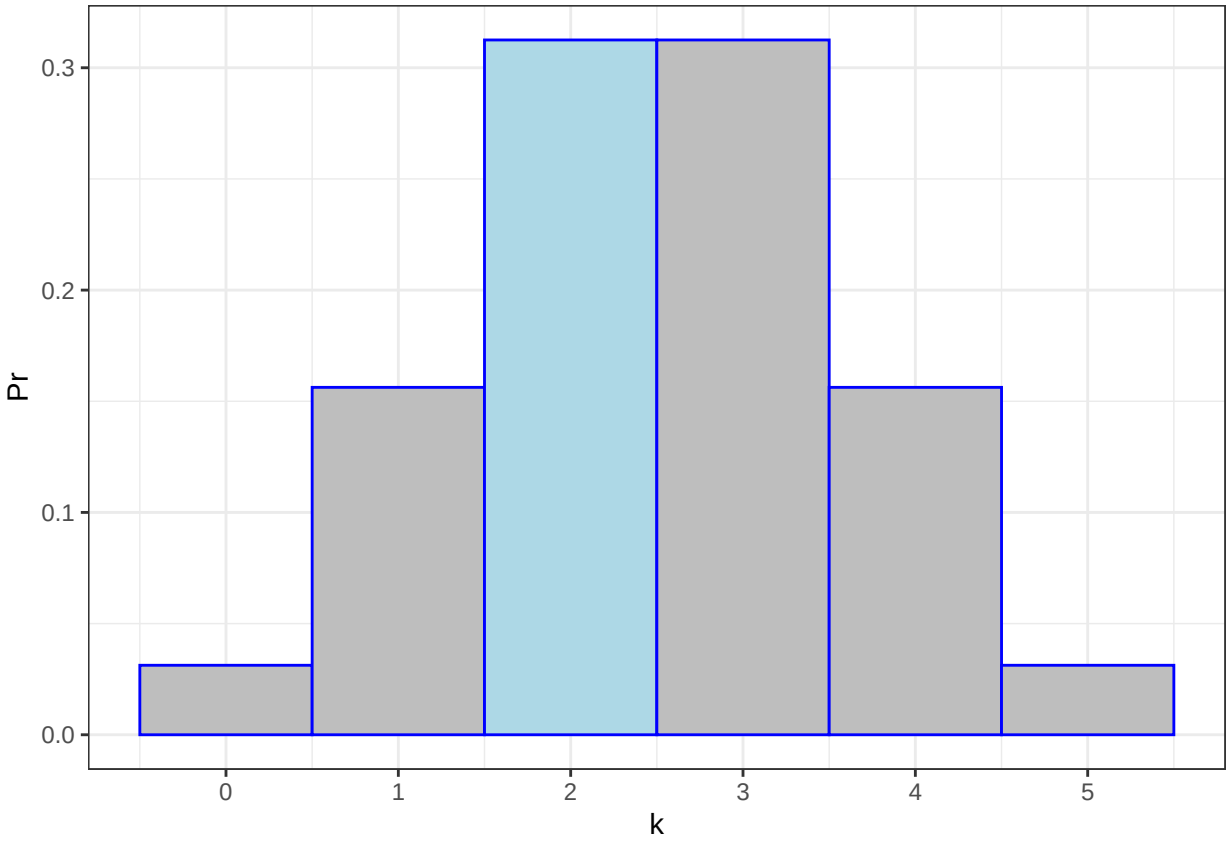
```
## [1] 0.3125
```

You can send in a list of values into `dbinom` to get a list of probabilities. Here's how to get the whole binary frequency distribution:

```
dbinom(seq(0,5),5,0.5)
```

```
## [1] 0.03125 0.15625 0.31250 0.31250 0.15625 0.03125
```

We can plot this binomial frequency distribution as a bar graph, highlighting k = 2 in blue:



The shape of the probability distribution for N=5 should look familiar. It looks normal! More on this later.

We just calculated the probability of getting exactly 2 heads out of 5 coin flips. What about the probability of calculating 2 *or more* heads out of 5? It's not hard to see that:

$$Pr(k \geq 2) = Pr(k = 2) + Pr(k = 3) + Pr(k = 4) + Pr(k = 5)$$

Which for our example is:

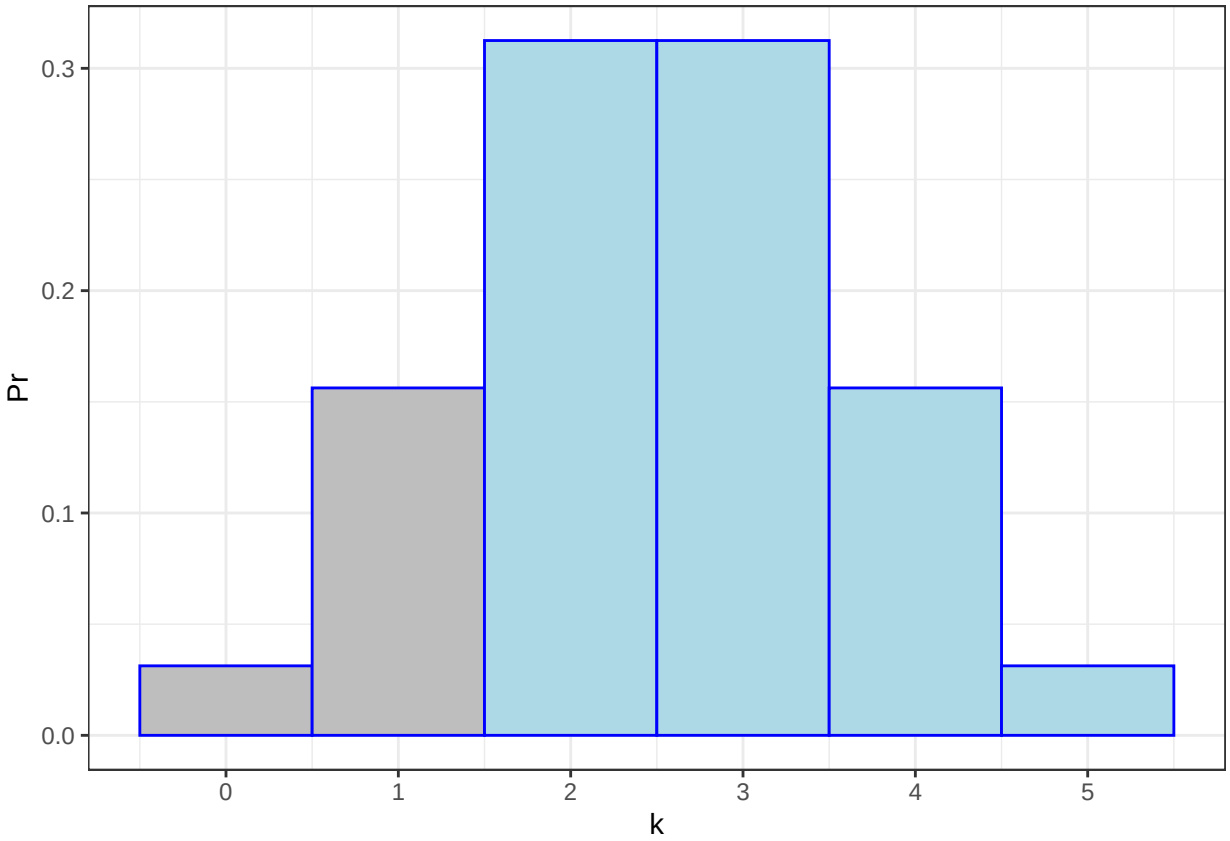
$$0.3125 + 0.3125 + 0.15625 + 0.03125 = 0.8125$$

This is called the 'cumulative binomial probability'. The easiest way to calculate this with R is to sum up the individual binomial probabilities:

```
sum(dbinom(seq(2,5),5,0.5))
```

```
## [1] 0.8125
```

Which can be shown as:



Since each of these bars has a width of 1 unit, the area of each bar represents the probability of each outcome. The total area shaded in blue is therefore $Pr(k \geq 2)$.

12.3 Exam guessing example

Suppose you're taking a multiple choice exam in PHYS 422, "Contemporary Nuclear And Particle Physics". There are 10 questions, each with 4 options. Assume that you have no idea what's going on and you guess on every question. What is the probability of getting 5 or more answers right?

Since there are 4 options for each multiple choice question, the probability of guessing and getting a single question right is P = $\frac{1}{4}$ = 0.25. The probability of getting 5 or more right is $Pr(k >= 5)$. We can find this answer using `dbinom` in R with N = 10, k = 5 and P = 0.25:

```
sum(dbinom(seq(5,10),10,0.25))
```

```
## [1] 0.07812691
```

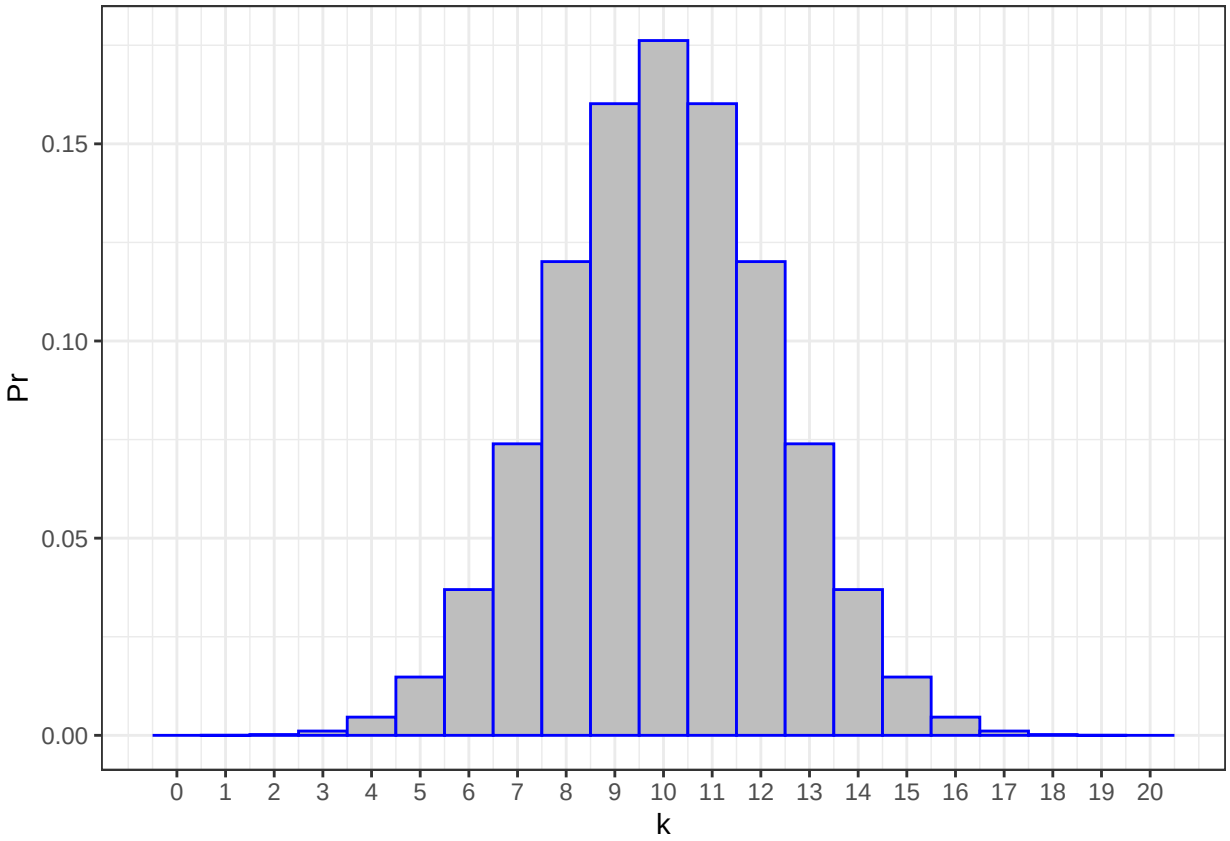
What is the probability of getting 6 or fewer *wrong*? The probability of getting any single problem wrong by chance is 1-0.25 = 1-0.25. We calculate Pr(k $\leq$ 6) with:

```
sum(dbinom(seq(0,6),10,0.75))
```

```
## [1] 0.2241249
```

12.4 Normal approximation to the binomial

Here's the probability distribution for P = 0.5 and n = 20:



This is a familiar shape! Yes, even though the binomial distribution is 'discrete', it's still shaped like the normal distribution. In fact, we know the mean of the distribution is:

$$\mu = nP$$

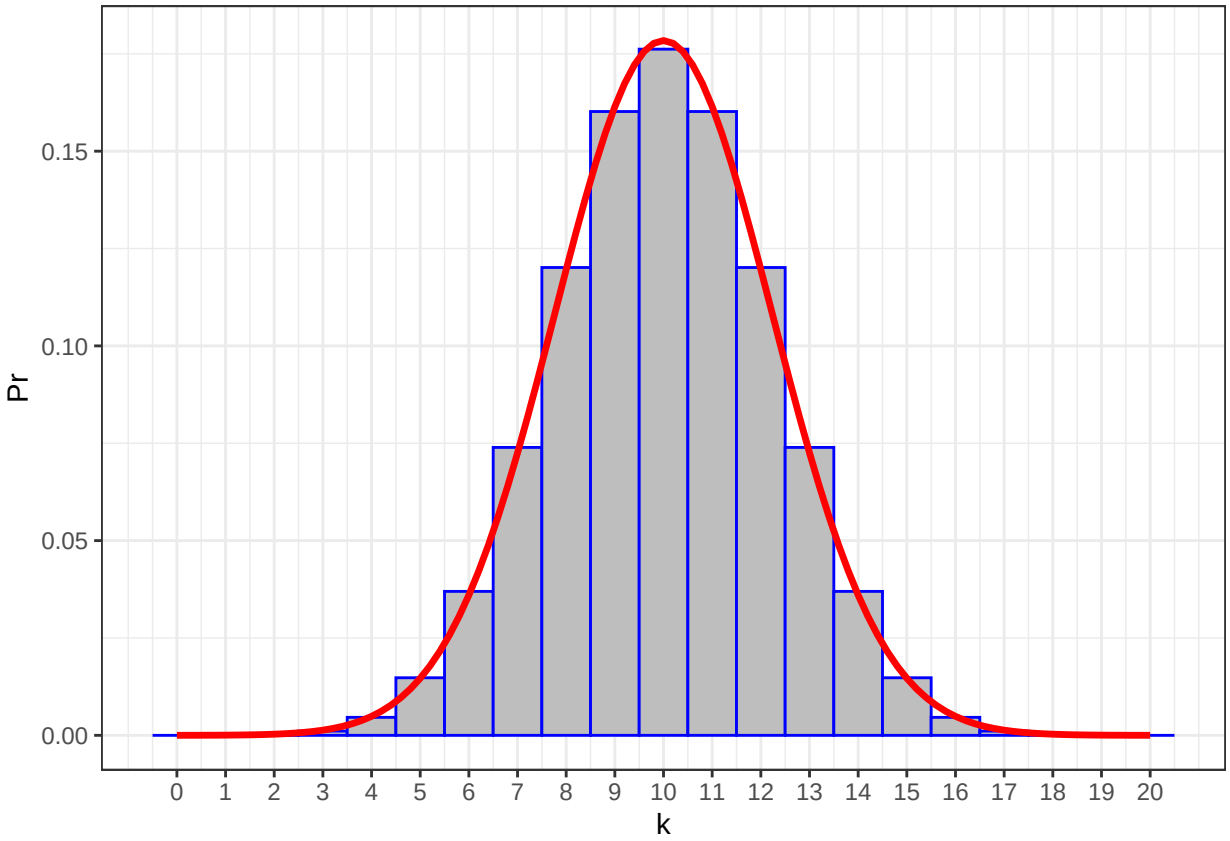
This should make sense. If, for example you flip a 50/50 coin 100 times, you expect to get $(100)(.5) = 50$ heads (or tails). For the example above with P = 0.5 and n = 20, $\mu = (20)(0.5) = 10$

Interestingly, the standard deviation is also known:

$$\sigma = \sqrt{nP(1-P)}$$

For example, $\sigma = \sqrt{(20)(0.5)(0.5)} = 2.2361$

Here's that binomial distribution drawn with a smooth normal distribution with $\mu = 10$ and $\sigma = 2.2361$:



You can see that it's a nice fit.

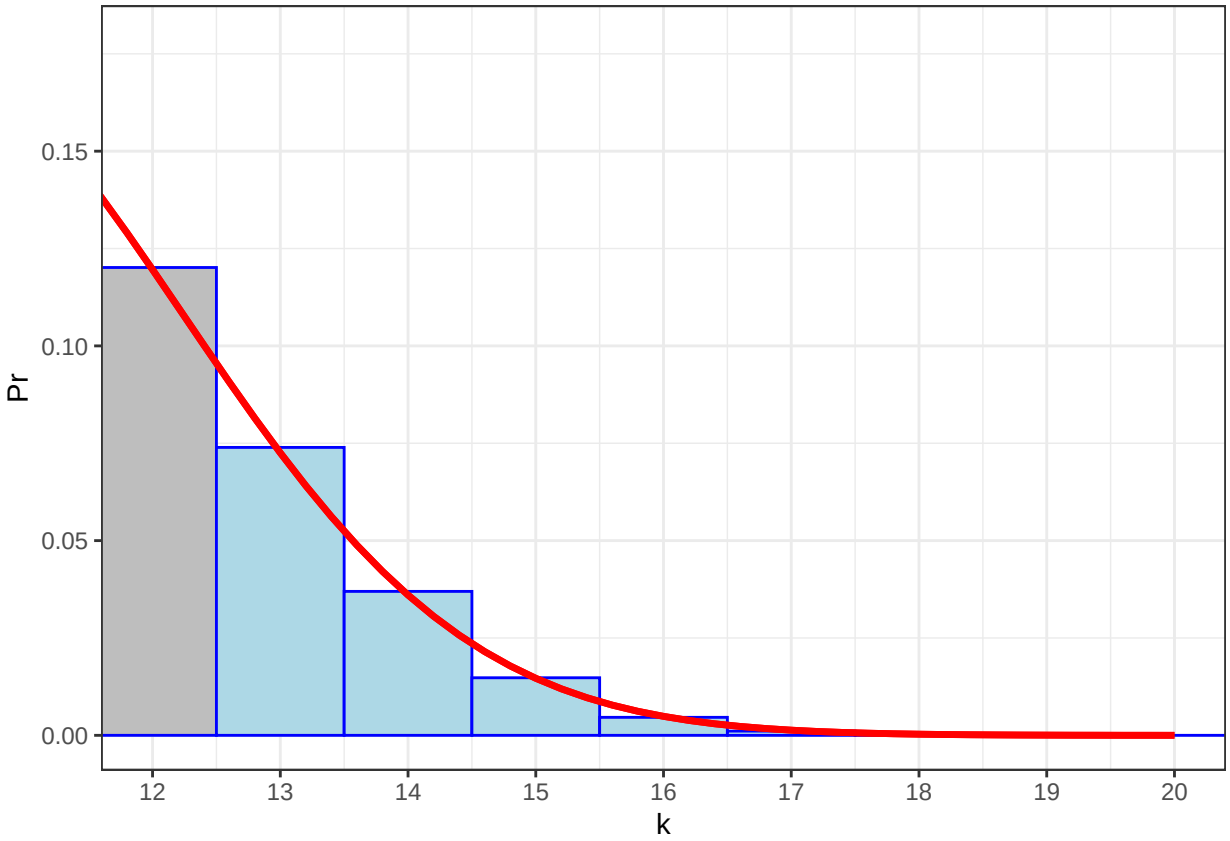
If we consider this as the distribution for obtaining k heads out of 20 coin flips, the probability of getting, say 13 or more heads is:

```
sum(dbinom(seq(13,20),20,0.5))
```

```
## [1] 0.131588
```

As we mentioned above, you can think of the areas of the gray bars as representing probabilities. So the area under the red normal distribution should be a good approximation to the binomial distribution.

There's one small hack, called the 'correction for continuity'. Let's zoom in on the upper tail, shading in the bars for $k \geq 13$:



In order to find the area under the (red) normal distribution that approximates the blue area, we need to actually find the area that covers the entire bar for k = 13. That means we need to back up one half of a unit and find the area above k = 12.5. Here's that area, using `pnorm`:

```
1-pnorm(13-.5,10,2.2361)
```

```
## [1] 0.1317797
```

This is really close to the result we got from `dinom`: 0.131588

Using the normal distribution to approximate binomial probabilities used to be really useful when we had to rely on tables for the binomial distribution. Tables typically went up to only n=15 or n=20. But now with computers we can calculate the 'exact' probabilities with `dinom` or its equivalent in other languages.

However, we still rely on this sort of approximation for some statistical tests, like the χ-squared test. So it's good to keep in mind that the χ-squared test is actually doing something like the approximation shown above - though often the correction for continuity is left out!

12.5 The binomial hypothesis test: Seattle Mariners season

The Seattle Mariners just completed a disappointing season. They were leading the division by 10 games in June, but they completely fell apart after the All Star break. They failed to make the playoffs.

At the end of the season they won 85 games and lost 77 games. How good is this? Specifically, let's answer the question: If the Mariners were a truly average team, meaning that there's a 50% chance of winning any game, what is the probability of having a season this good or better?

This is easy using `dbinom`.

```
wins <- 85 # wins
losses <- 77 # losses
P <- .5  # expected P(win)
```

```
n <- wins+losses  # no ties in baseball!

p.value <- sum(dbinom(seq(wins,n),n,P))
p.value
```

```
## [1] 0.2912379
```

A totally average team that has a 50/50 chance of winning any game will have a season this good or better 29 percent of the time. This is not a low probability, we can conclude that even with their great start, we can't conclude that the Mariners were any better than an average team.

Notice that even there are a large number of games in a season (162), we can still use the binomial distribution. We're adding up many small numbers, but computers don't complain about these things.

R has it's own function for computing a hypothesis test on binomial distributions, called `binom.test`. It works like this, using the variables from above:

```
binom.out <- binom.test(wins,n,P,alternative = 'greater')
binom.out$p.value
```

```
## [1] 0.2912379
```

This is probably a really short function, and we know exactly what it did. In fact, you can use `binom.test` to get your summed binomial probabilities instead of `sum(dbinom(..` like we've been doing this chapter.

You can set `alternative = 'two.sided`, which is the default, which simply doubles the p-value. It tests the hypothesis of obtaining a value of k that is far away from nP in either direction. If we want to see if the Mariner's season is truly mediocre, we really should have run a two-tailed test. In this case, the p-value is p = 0.5825. With a p-value around 0.5, this means that the 2024 Mariners aren't even unusually average, if that makes any sense.

12.6 The margin of error in polling

You've all seen the results of a poll where an article will state that the 'results are within the margin of error'. What does this mean? Consider a poll where there are two choices, candidate D and candidate R. Suppose that in the population, 50 percent of voters are planning on voting for candidate D and 50 percent of voters are planning on voting for candidate R.

If we sample 1000 voters from the population, then using the normal approximation to the binomial, we know that the the number of voters for candidate D will be distributed normally with a mean of

$$\mu = nP = (1000)(0.5) = 500$$

and a standard deviation of

$$\sigma = \sqrt{nP(1-P)} = \sqrt{(1000)(0.5)(0.5)} = 15.811$$

Polls, however, are reported in terms of the proportions or percentages, not in terms of the number of voters preferring a candidate. We can convert the counts of voters to percentage of voters by dividing the equations above by total sample size and multiplying by 100. Remember, when we multiply (or divide) the scores in a normal distribution by a constant, both the mean and standard deviation are multiplied (or divided) by that same constant. So (not surprisingly) the *percent* of voters from the poll will be drawn from a normal distribution with mean:

$$\mu_{percent} = 100\frac{nP}{n} = (100)(0.5) = 50$$

and a standard deviation of:

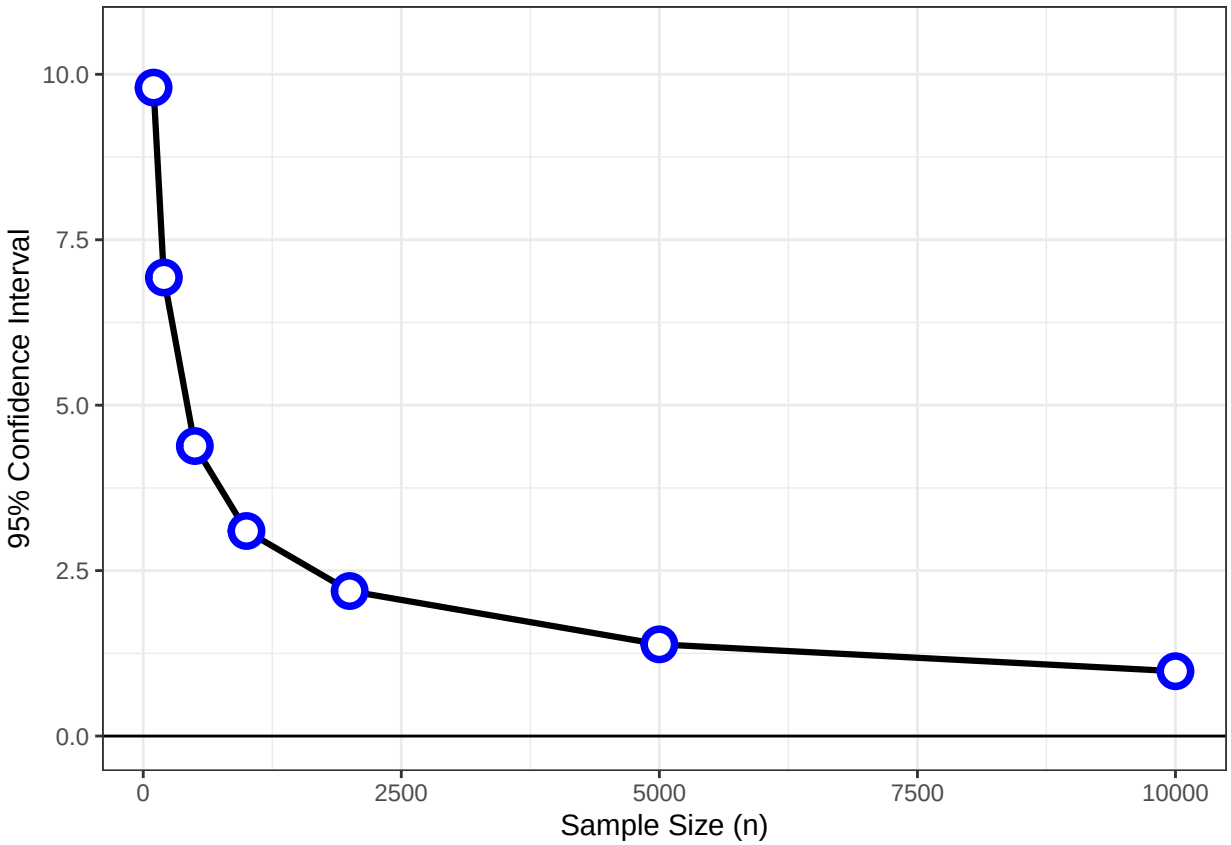
$$\sigma_{percent} = 100\frac{\sqrt{nP(1-P)}}{n} = 100\sqrt{\frac{P(1-P)}{n}} = 100\sqrt{\frac{(0.5)(1-0.5)}{1000}} = 0.0158$$

When pollsters talk about 'margin of error', they're talking about a confidence interval. Pretty much by default, unless otherwise defined, a confidence interval covers 95% of the probability distribution. Since this is a normal distribution, we can use our `qnorm` to calculate the 95% confidence interval for our poll:

```
n <- 1000
P <- .5
alpha <- .05
sigma_percent <- 100*sqrt((P*(1-P))/n)
CI = qnorm(1-alpha/2,0,sigma_percent)
sprintf('Our margin of error is plus or minus %0.2f percentage points',CI)
```

```
## [1] "Our margin of error is plus or minus 3.10 percentage points"
```

The margin of error depends both on P and on the sample size, N. For close races we can assume that P = 0.5. Here's a graph of how the margin of error depends on the sample size:



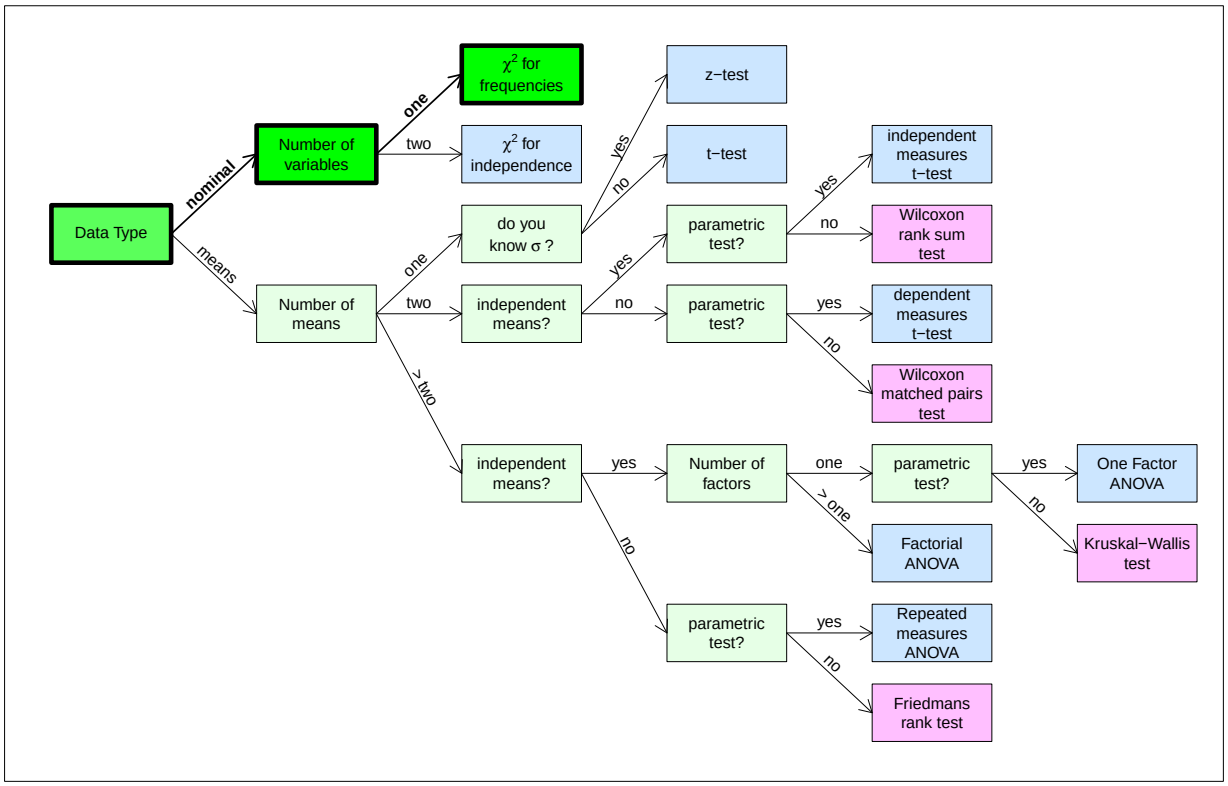
From this graph you can estimate the sample size of a poll from the reported margin of error. You've probably noticed that typical polls have a margin of error of around 2-3 percentage points. You can tell from the graph that this is because typical polls have around 1000-2000 voters.

Chapter 13

χ^2 Test For Frequencies

The Chi squared (χ^2) test for frequencies a hypothesis test on the frequency of samples that fall into different discrete categories. For example, are the number of left and right-handed people in Psych 315 distributed like you'd expect from the population? Or, is the frequency distribution of birthdays by month for the students in Psych 315 distributed evenly across months? For these tests the dependent measure is a frequency, not a mean.

You can find this test in the flow chart here:



Let's start with a simple example:

13.1 Example 1: left vs. right handers in Psych 315

According to Wikipedia, 10 percent of the population is left handed. For Psych 315, 7 students reported that they are left handed, while 145 reported right handedness. A χ^2 test determines if the frequency of our sampled observations are significantly different than the frequencies that you'd expect from the population. Specifically, the null hypothesis is that our observed frequencies are drawn from a population that has some expected proportions, and our alternative hypothesis is that we're drawing from a population that does not have these expected proportions.

Like all statistical tests, the χ^2 test involves calculating a statistic that measures how far our observations are from those expected under the null hypothesis.

The first step is to calculate the frequencies expected from the null hypothesis. This is simply done by multiplying the total sample size by each of the expected proportions. Since there are 152 students in the class, then we expect $(152)(0.1) = 15.2$ students to be left handed and $(152)(0.9) = 136.8$ to be right handed. Expected frequencies do not have to be rounded to the nearest whole number, even though frequencies are whole numbers. This is because we should think of these expected frequencies as the *average* frequency for each category over the long run - and averages don't have to be whole numbers.

The next step is to measure how far our observed frequencies are from the expected frequencies. Here's the formula, where χ^2 is pronounced "Chi-squared".

$$\chi^2 = \sum \frac{(f_{obs} - f_{exp})^2}{f_{exp}}$$

Where f_{obs} are the observed frequencies and f_{exp} are the expected frequencies. For our example, f_{obs} is 7, and 145 and f_{exp} is 15.2, and 136.8 for left and right-handers respectively:

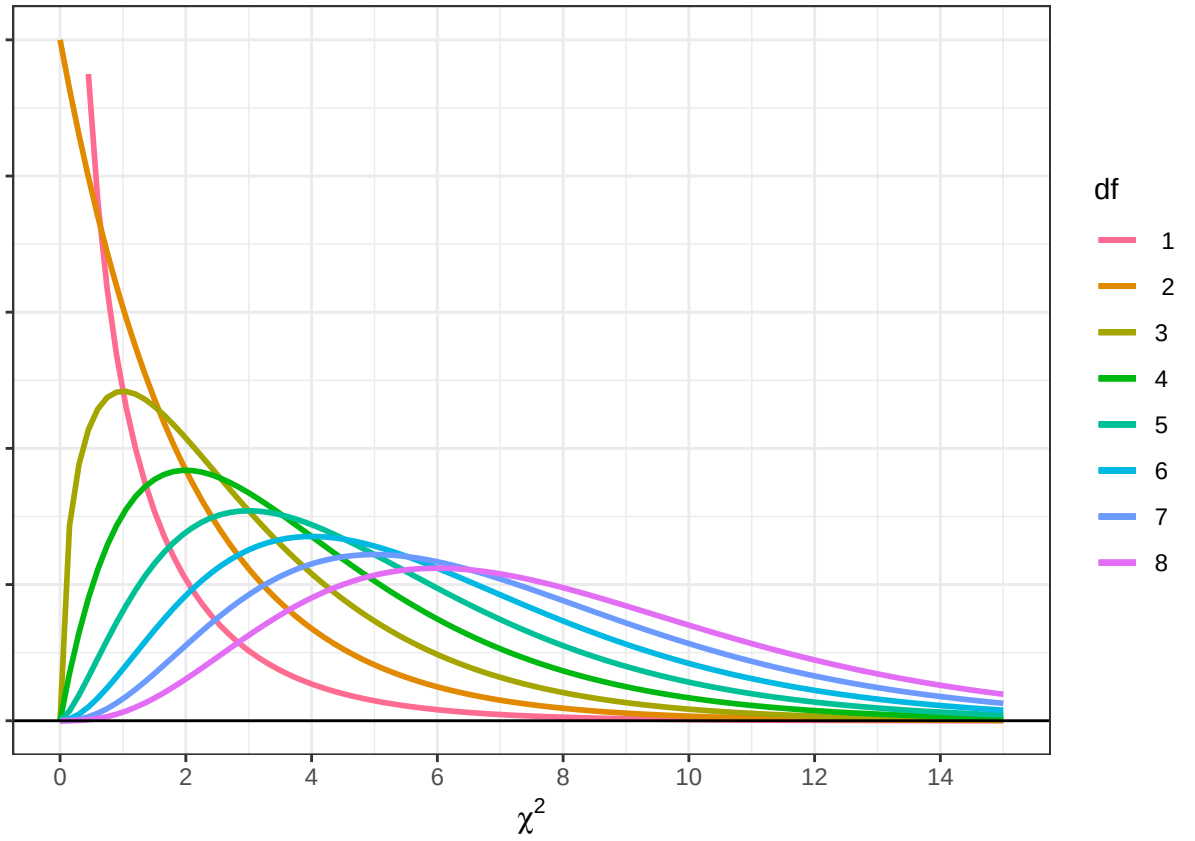
$$\chi^2 = \sum \frac{(f_{obs} - f_{exp})^2}{f_{exp}} = \frac{(7 - 15.2)^2}{15.2} + \frac{(145 - 136.8)^2}{136.8} = 4.9152$$

This measure, χ^2, is close to zero when the observed frequencies match the expected frequencies. Therefore, large values of χ^2 can be considered evidence against the null hypothesis.

13.2 The χ^2 distribution

Just like the z and t distributions, the χ^2 distribution has a known 'parametric' shape, and therefore has known probabilities and areas associated with it. Also, like the t-distribution, the χ^2 distribution is a family of distributions, with a different distribution for different degrees of freedom.

The χ^2 distribution for k degrees of freedom is the distribution you'd get if you draw k values from the standard normal distribution (the z-distribution), square them, and add them up. For more information on this, check out [this section][Simulating χ^2] in the chapter on how to generate all of our distributions from normal distributions. Here's what the probability distributions look like for different degrees of freedom:



Notice how the shape of the distributions spread out and change shape with increasing degrees of freedom. This is because as we increase df, and therefore the number of squared z-scores, the sum will on average increase too.

Since the χ^2 distribution is known we can calculate the probability of obtaining our observed value of χ^2 if null hypothesis is true. For a χ^2 test for frequencies, *The degrees of freedom is the number of categories minus one.* For this example, df = - 1 = 1.

13.3 R's pchisq function

Consistent with `pnorm`, `pt` and `pF`, R's `pchisq` function finds the area below a given value. So to find the area above $\chi^2 = 4.9152$ for df = 1 we subtract the result of `pchisq` from 1:

```
1-pchisq(4.9152,1)
```

```
## [1] 0.02662138
```

13.4 R's qchisq function

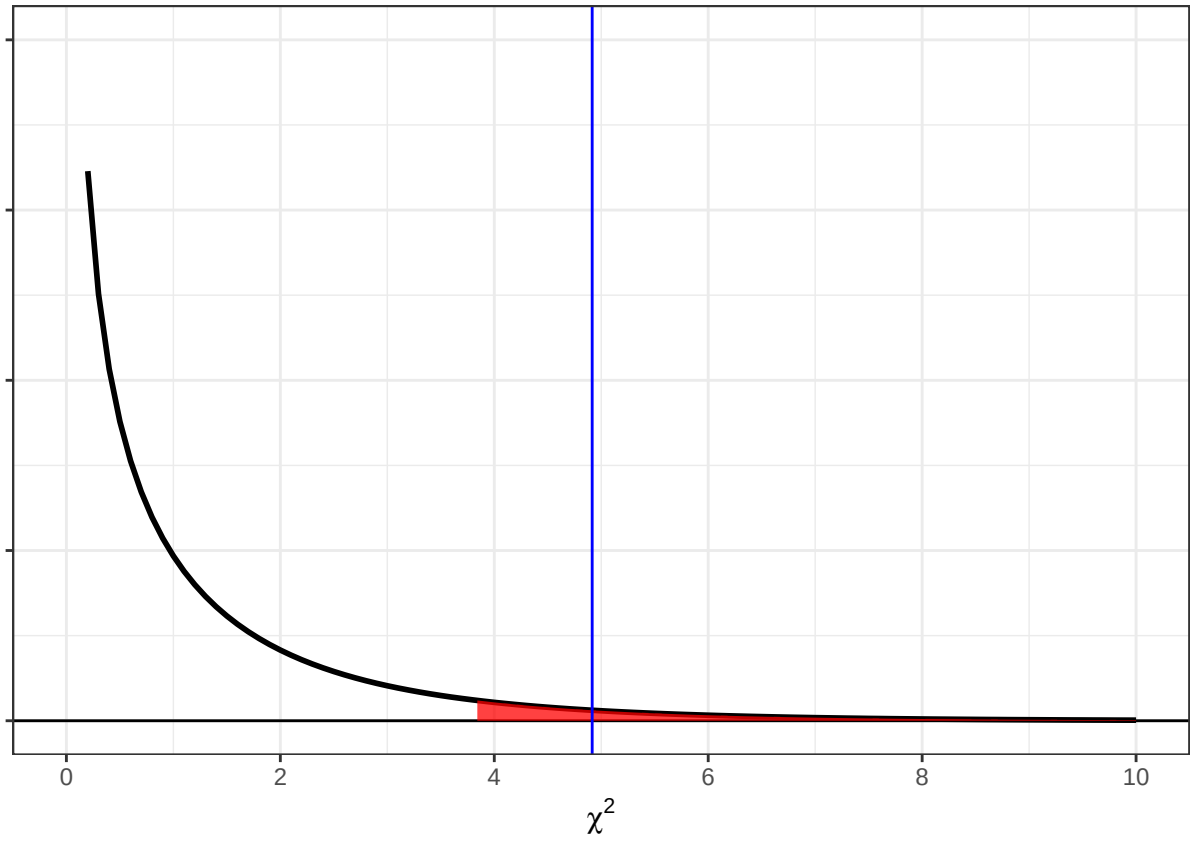
Like `qnorm`, `qt` and `qF`, R's `qchisq` function finds the value of $\chi 2$ for which a given proportion of the area lies below. So for our example, if $\alpha = 0.05$, the critical value of χ^2 for making our decision is:

```
qchisq(1-0.05,1)
```

```
## [1] 3.841459
```

We reject H_0 for any observed χ^2 value greater than 3.84.

Here's an illustration of the pdf for χ^2 for df = 1 with the top 5 percent shaded in red and our observed value of χ^2 = 4.9152 shown as a vertical line.



You can see that our observed value of χ^2 (4.9152) falls above the critical value of χ^2 (3.8415). Also, our p-value (0.0266) is less than $\alpha = 0.05$. We therefore reject H_0 and conclude that the proportion of left and right handers in our class is significantly different than what is expected from the population.

13.5 Using `chisq.test` to run a $\chi 2$ test for frequencies

R has the `chisq.test` function that computes the values that we've done by hand. For our example we'll load in the survey data and compute 'fobs' which is the frequency of the observed number of left and right handers in the class. This will be compared to `prob = c(.1,.9)` which is where we enter our population, or null hypothesis proportions.

```
survey <-read.csv("http://www.courses.washington.edu/psy315/datasets/Psych315W21survey.csv")
prob <- c(.1,.9)
fobs <- c(sum(survey$hand=="Left",na.rm = T),sum(survey$hand=="Right",na.rm = T))
chisq.out <- chisq.test(fobs,p=prob)
```

The important fields in the output of `chisq.test` are `statistic` which holds the observed χ^2 value, `parameter` which holds the degrees of freedom, `expected` which holds the computed expected frequencies, `observed` which holds the observed values (that we entered), and `p.value`.

13.6 APA format for the χ^2 test for independence

Here's how to extract the fields in the output of `chisq.test` into generate a string in APA format:

```
sprintf('Chi-Squared(%d,N=%d) = %5.4f, p = %5.4f',
        chisq.out$parameter,
        sum(chisq.out$observed),
        chisq.out$statistic,
        chisq.out$p.value)
```

```
## [1] "Chi-Squared(1,N=152) = 4.9152, p = 0.0266"
```

13.7 One or two tailed?

A common question for χ^2 tests is whether it is a one or a two tailed test. You might think it's a one-tailed test since we only reject H_0 for large values of χ^2. However, because the numerator of the χ^2 formula is $\sum (f_{obs} - f_{exp})^2$, we can get large values of χ^2 if our observed frequency, f_{obs} is either too small *or* too large compared to the expected frequency, f_{exp}. So really a χ^2 test is a two-tailed test.

13.8 Relationship between χ^2 test and the normal approximation to the binomial

You might have noticed that we already know a way of calculating the probability of obtaining 15.2 or more left-handers out of 152 students, assuming that 10 percent of the population is left-handed. That's right - the normal approximation to the binomial.

Let's compare:

Using the normal approximation with N = 152 and P = 0.1 , the number of left-handers will be distributed normally with a mean of

$$\mu = NP = (152)(0.1) = 15.2$$

and a standard deviation of

$$\sigma = \sqrt{NP(1-P)} = \sqrt{(152)(0.1)(0.9)} = 3.6986$$

.

The Chi-squared test is a two-tailed test, so to conduct a two tailed test with the normal approximation to the binomial we need to find the probability of obtaining an observed value more extreme than 7 compared to 15.2 .

If we ignore the business of correcting for continuity, we can use `pnorm` to find this probability. First, we convert our observed frequency of left-handers to a z-score:

```
z <- (7-15.2)/3.6986
```

Then we convert our z-score to a p-value like we did back in the chapter on the normal distribution:

```
2*(1-pnorm(abs(-2.2171)))
```

```
## [1] 0.02661626
```

This number is very close to the p-value we got from the χ^2 test: 0.0266. In fact, the difference is only due to rounding error. Thus *the χ^2 test with df = 1 is mathematically equivalent to the normal approximation to the binomial*. This means that the tests are the same if you don't correct for continuity, which is that process of including the whole interval by subtracting and adding 0.5.

The χ^2 test is therefore actually not quite right. Also, remember that the normal approximation to the binomial is really best for larger sample sizes, like for n>20. But we typically run χ^2 tests for smaller values of n. values of n.

13.9 Comparing the χ^2 test to the binomial test

For two groups like this example, we can use the binomial distribution, which we can get from `binom.test`:

```
binom.out <- binom.test(fobs[1],sum(fobs),prob[1],alternative = 'two.sided')
binom.out$p.value
```

```
## [1] 0.02140999
```

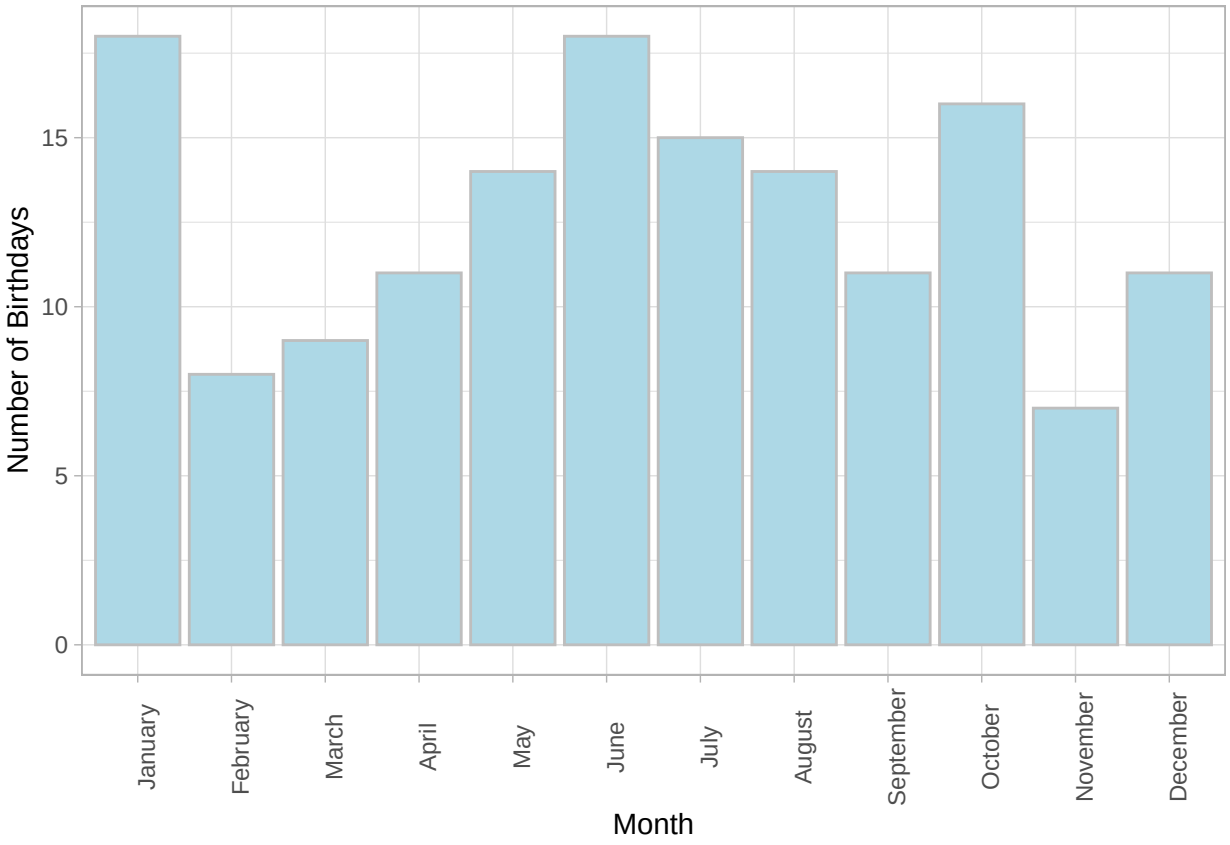
It's close, but not exact. In most cases, the χ^2 test is a good enough approximation to the 'exact' solution. You really only run into trouble when your expected frequencies drop too low, say below 5 or so.

13.10 Example 2: Birthdays by month

Let's see if the birthdays in this class are evenly distributed across months, or if there are some months for which students have significantly= more birthdays than others. For simplicity, we'll assume that all months have equal probability, even though they vary in length. We'll run a χ^2 test using an alpha value of 0.05.

Here's a table showing the number of birthdays for each month for all 152 students:

It looks kind of uneven. There are 18 students with birthdays in January and June but only 7 students with birthdays in November. A natural way to visualize this distribution is with a bar graph showing the frequency distribution:



To see if this distribution is significantly uneven we calculate the expected frequencies under the null hypothesis. Here we expect equal frequencies of $\frac{152}{} = 12.6667$ birthdays per month . Note equal frequencies assumes that each month has an equal number of days. This assumption is close enough for this example, but how would you correct the expected frequencies to account for this?

Using our χ^2 formula:

$$\chi^2 = \sum \frac{(f_{obs} - f_{exp})^2}{f_{exp}} = \frac{(18 - 12.6667)^2}{12.6667} ... + ... \frac{(11 - 12.6667)^2}{12.6667} = 12.0526$$

With months, the degrees of freedom is 11. Using `pchisq` we can find the p-value for our test:

```
1-pchisq(12.0526,11)
```

```
## [1] 0.3597025
```

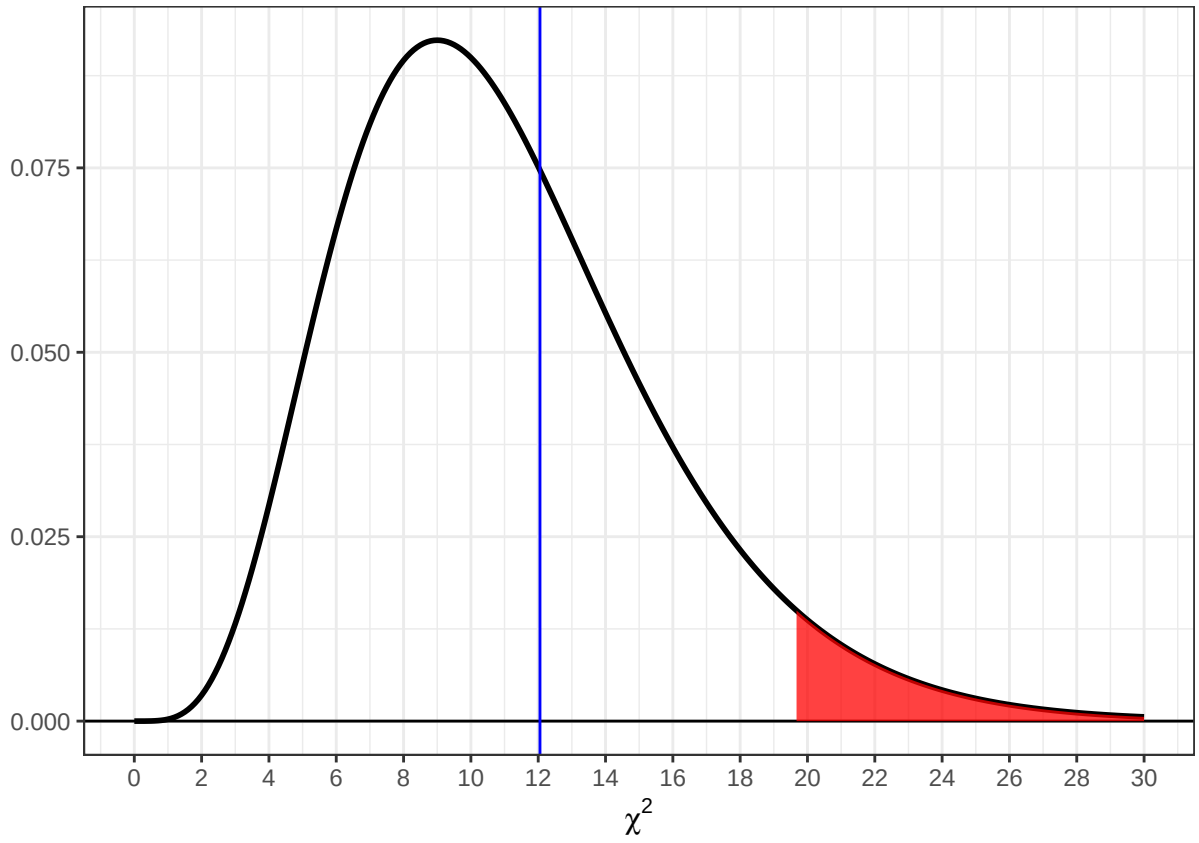
Or, we can find the critical value of χ^2 using `qnorm`:

```
qchisq(1-0.05,11)
```

```
## [1] 19.67514
```

We reject H_0 for any observed χ^2 value greater than 19.68.

Here's an illustration of the pdf for χ^2 for df = 11 with the top 5 percent shaded in red and our oberved value of χ^2 = 12.0526 shown as a vertical line.



You can see that our observed value of χ^2 (12.0526) falls below the critical value of χ^2 (19.6751). Also, our p-value (0.3597) is greater than $\alpha = 0.05$. We therefore fail to reject H_0 and conclude that the distribution of birthdays across months is not significantly different than what is expected from chance.

13.11 Conducting the same test with `chisq.test`

With the survey data loaded in from the previous example, running the χ^2 test for frequencies on the birth month data is pretty straightforward. The only new thing is the use of the function `table` to get a table of frequencies for birthdays across months, like we did in the chapter on descriptive statistics:

```
dat <- table(survey$month)
dat
```

```
##
##   January  February     March     April       May      June      July    August
##        18         8         9        11        14        18        15        14
## September   October  November  December
##        11        16         7        11
```

We can then send this table into `chisq.test`. We could send in a list of expected probabilities, but by default, `chisq.test` assumes equal probabilities across categories:

```
chisq.out <- chisq.test(dat)
chisq.out
```

```
##
##  Chi-squared test for given probabilities
##
## data:  dat
## X-squared = 12.053, df = 11, p-value = 0.3597
```

We can coerce the output of `chisq.test` into APA format just like we did for the first example:

```
sprintf('Chi-Squared(%d,N=%d) = %5.4f, p = %5.4f',
        chisq.out$parameter,
        sum(chisq.out$observed),
        chisq.out$statistic,
        chisq.out$p.value)
```

```
## [1] "Chi-Squared(11,N=152) = 12.0526, p = 0.3597"
```

13.12 Effect size

There are a couple of definitions for effect size for the χ^2 chi-squared test for frequencies. The one that is used most commonly to calculate power is is called ψ, or 'psi':

$$\psi = \sqrt{\frac{\chi^2}{N}}$$

Where N is the total sample size. From our first example (handedness):

$$\psi = \sqrt{\frac{\chi^2}{N}} = \sqrt{\frac{4.9152}{152}} = 0.1798$$

From our second example (birth months):

$$\psi = \sqrt{\frac{\chi^2}{N}} = \sqrt{\frac{12.0526}{152}} = 0.2816$$

ψ has the desirable quality that it is not influenced by sample size, so it can be used to compare effect sizes across experiments. However, since χ^2 increases with the number of categories (and df), so does ψ, so we can't use ψ on it's own to determine if an effect is 'small', 'medium', or 'large'.

An alternative measure of effect size is 'Cramer's V', which puts the degrees of freedom in the denominator to offset the increase in df with ψ:

$$V = \sqrt{\frac{\chi^2}{N \times df}}$$

If you look at the formulas you can see that there is a relation between ψ and V:

$$V = \frac{\psi}{\sqrt{df}}$$

For V, 0.1 is considered small, 0.3 medium, and 0.5 is a large effect size.

From our first example (handedness):

$$V = \sqrt{\frac{\chi^2}{N \times df}} = \sqrt{\frac{4.9152}{(152)(1)}} = 0.1798$$

Which would be considered to be a small effect size.

From our second example (birth months):

$$V = \sqrt{\frac{\chi^2}{N \times df}} = \sqrt{\frac{12.0526}{(152)(11)}} = 0.0849$$

Which would also be considered to be a small effect size.

13.13 Power for the χ^2 test for frequencies

Power can be calculated with R using 'pwr.chisq.test'. It requires the effect size ψ (called 'w' for some reason), the total sample size N, the degrees of freedom, and the alpha value.

`pwr.chisq.test` works much like `pwr.t.test` for the t-test. To find the power for the first test on handedness:

```
chisq.power.out <- pwr.chisq.test(w = 0.1798,N = 152,df = 1,sig.level = 0.05)
```

The observed power for the first test can be found:

```
chisq.power.out$power
```

```
## [1] 0.6013325
```

If you want to find the sample size that gives you a desired power of 0.8, pass in `NULL` for the sample size (N), and set `power = .8`:

```
chisq.power.out <- pwr.chisq.test(w = 0.1798,N = NULL,df = 1,sig.level = 0.05,power = 0.8)
```

The sample size needed is:

```
chisq.power.out$N
```

```
## [1] 242.788
```

Power for the second example, on birth months is:

```
chisq.power.out <- pwr.chisq.test(w = 0.2816,N = 152,df = 11,sig.level = 0.05)
```

```
chisq.power.out$power
```

```
## [1] 0.6220806
```

The sample sized needed to get a power of 0.8 is:

```
chisq.power.out <- pwr.chisq.test(w = 0.2816,N = NULL,df = 11,sig.level = 0.05,power = 0.8)
```

```
chisq.power.out$N
```

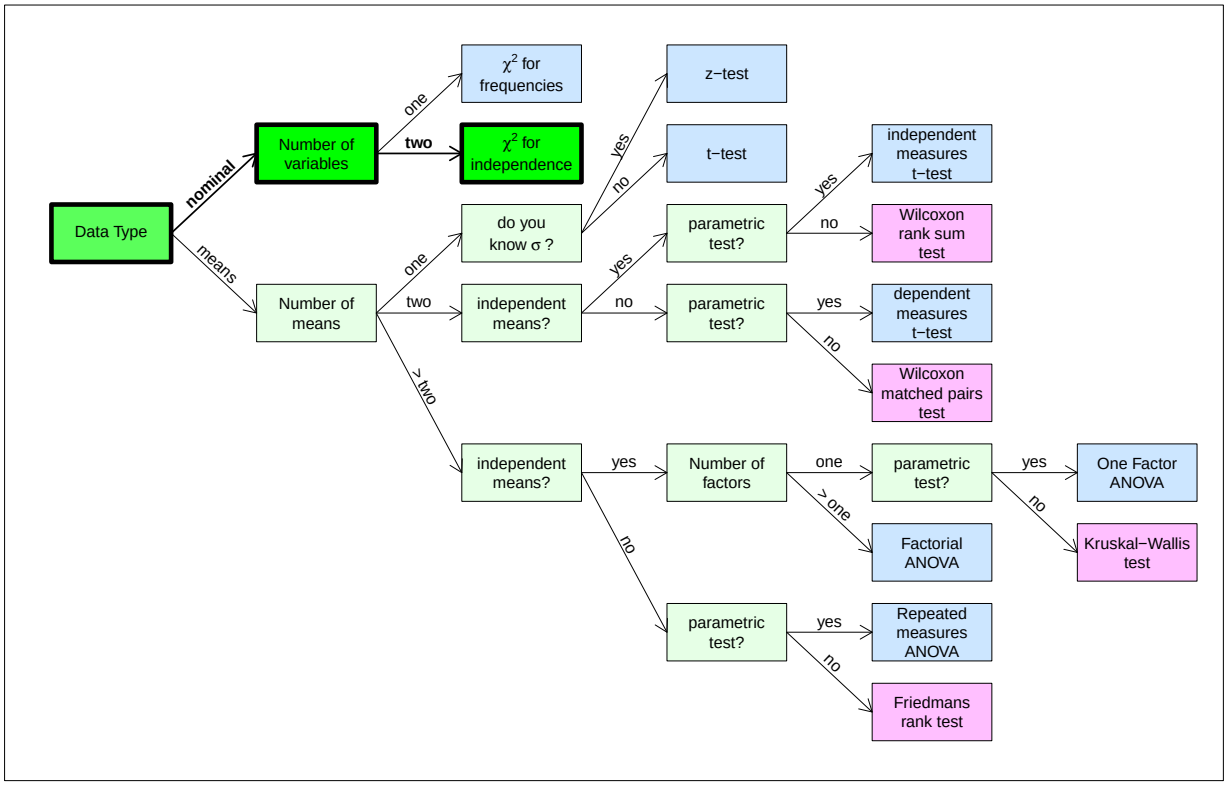
```
## [1] 211.8792
```

Chapter 14

χ^2 Test for Independence

This is the second kind of hypothesis test on frequency data. The test is to see if the frequency for which things fall into two nominal scale factors are independent of each other.

You can find this test in the flow chart here:



For the first example, we'll use this test to see if the choice of computers that students use varies with gender.

14.1 Example 1: Computer users by gender

To find out we'll run a χ^2 test for independence using alpha = 0.05.

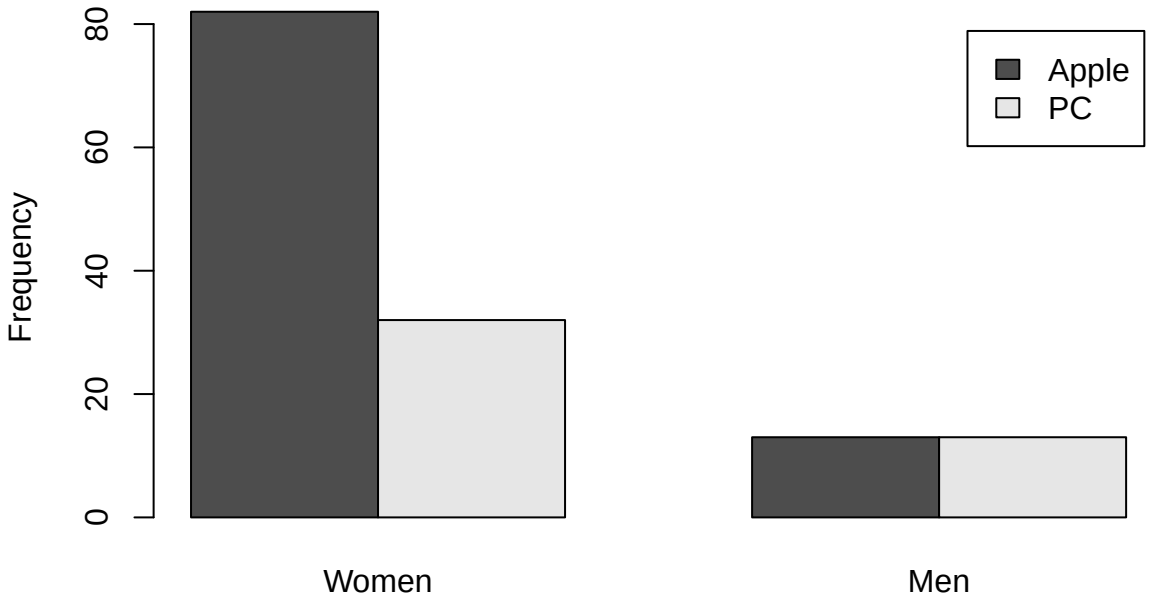
We'll use our survey data and count the number of students who use each kind of computer, depending upon their gender. This generates the following 2 X 2 table:

Table 14.1:

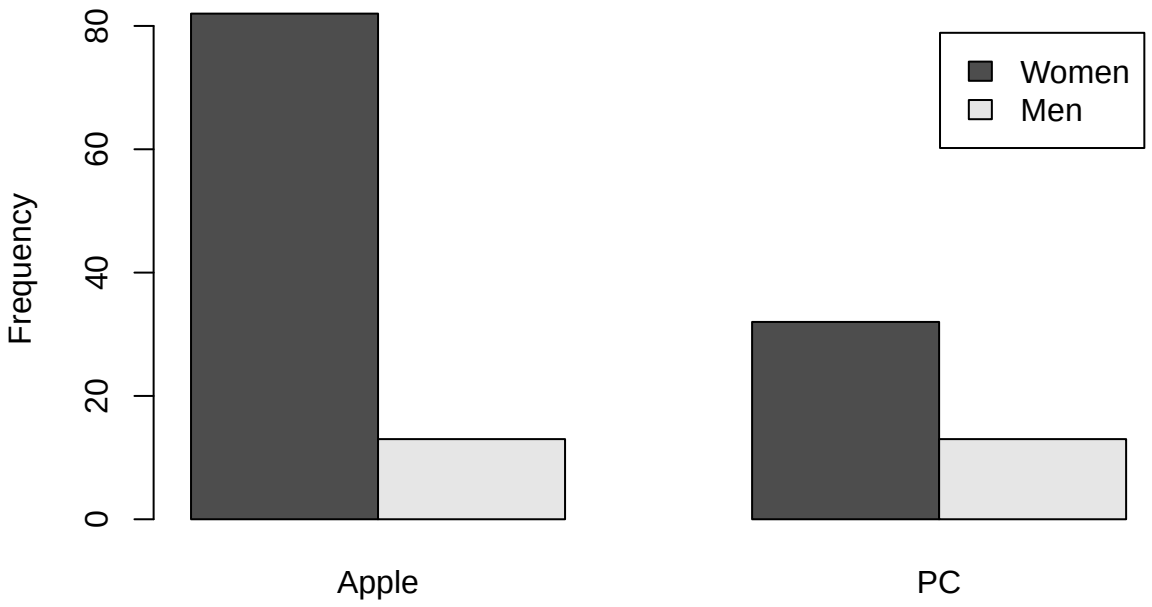
	Apple	PC
Women	82	32
Men	13	13

If you add up the rows and columns, you'll see that there are $82 + 13 = 95$ students who use Apple computers, and $32 + 13 = 45$ students who use PC computers. If computer choice did not depend on gender, then we should see a similar 95/45 ratio of computer use within the subsets of women and men in the class.

This type of frequency data can be visualized with a bar graph with a legend representing the second factor:



By the way, it's an arbitrary choice for which factor is on the x-axis and which factor is in the legend. Here's the same data plotted the other way around:



I find it interesting as a vision scientist that you can get different impressions of the data from these two graphs. Typically, for some reason, the implication is that the x-axis factor is *causing* any difference the legend factor. That's why the first graph seems more meaningful. It's more likely that your gender will determine the choice of computer rather than the other way around.

Anyway, to test the null hypothesis that computer use is independent of gender (and vice versa), we generate a table of expected frequencies. This table is set up so that the ratios of computer use is the same within each gender, but also ensures that the ratio of genders is the same across computer use.

An easy way to calculate the table of expected frequencies is to add up the rows and columns for observed frequencies. These are sometimes called *marginal sums* presumably because we put these numbers in the margins of the table. The total number of observations naturally ends up in the bottom right corner the table:

Table 14.2:

	Apple	PC	Row Sum
Women	82	32	114
Men	13	13	26
Column Sum	95	45	140

Then, to get the expected frequency for each cell we multiply the sum for that cell's row with the sum for that cell's column and divide by the total number of observations:

Table 14.3:

	Apple	PC
Women	$\frac{(114)(95)}{140} = 77.3571$	$\frac{(114)(45)}{140} = 36.6429$
Men	$\frac{(26)(95)}{140} = 17.6429$	$\frac{(26)(45)}{140} = 8.3571$

You should convince yourself that these expected frequencies have the right ratios under the null hypothesis. For example, for Women, the expected ratio of Apple to PC computer users is $\frac{77.3571}{36.6429} = 2.11$. For Men, the ratio is the same: $\frac{17.6429}{8.3571} = 2.11$. This is the same as the ratio of computer users for the whole class: $\frac{95}{45} = 2.11$

Going the other way, for Apple users, the expected ratio of Women to Men is $\frac{77.3571}{17.6429} = 4.38$. The ratio is the same for PC users: $\frac{36.6429}{8.3571} = 4.38$. This is the same as the ratio of genders for the whole class: $\frac{114}{26} = 4.38$

The math for calculating χ^2 is just like for the χ^2 test for frequencies. For each cell we calculate $\frac{(f_{obs}-f_{exp})^2}{f_{exp}}$ for each cell and sum up across all cells in the matrix:

$$\chi^2 = \sum \frac{(f_{obs} - f_{exp})^2}{f_{exp}}$$

Here's a table for each cell's χ^2 value:

Table 14.4:

	Apple	PC
Women	$\frac{(82-77.3571)^2}{77.3571} = 0.2787$	$\frac{(32-36.6429)^2}{36.6429} = 0.5883$
Men	$\frac{(13-17.6429)^2}{17.6429} = 1.2218$	$\frac{(13-8.3571)^2}{8.3571} = 2.5794$

The sum across all cells is $\chi^2 = \sum \frac{(f_{obs}-f_{exp})^2}{f_{exp}} = \frac{(82-77.3571)^2}{77.3571} + \frac{(13-17.6429)^2}{17.6429} + \frac{(32-36.6429)^2}{36.6429} + \frac{(13-8.35714)^2}{8.35714} = 4.6681$

The degrees of freedom for this test is (number of rows - 1)(number of columns - 1) = $df = (2-1)(2-1) = 1$. You should see that this measure, χ^2, is close to zero when the observed frequencies match the expected frequencies. Therefore, large values of χ^2 can be considered evidence against the null hypothesis of independence.

We can use `pchisq` to find the p-value for this example:

```
## [1] 0.0307279
```

We can report our results in APA format like this: "Chi-Squared(1,N=140) = 4.6681, p = 0.0307".

Using $\alpha = 0.05$ we can conclude that the choice of computers is not independent of gender.

14.2 Example 2 Does where you sit in class depend on gender?

When teaching my big classes, I notice that the women tend to sit disproportionately up front and the men in the back. This is a thing? This will be another χ^2 test for independence using alpha = 0.05.

This time we'll do it using `chisq.test` straight from the survey data. The first step is to make a table of frequencies.

```
survey <-read.csv("http://www.courses.washington.edu/psy315/datasets/Psych315W21survey.csv")
dat <- table(survey$gender,survey$sit)
```

We're almost there, but there are two issues, first, there are three rows for gender, but the first row - for those who chose not to answer - doesn't have enough people in it, so we'll need to leave this row out. The second issue is small - the order of the levels are alphabetical, but it'd be nice to turn them around so they read from front to back. Both of these issues can be fixed by selecting which rows and columns we want in the order that we want.

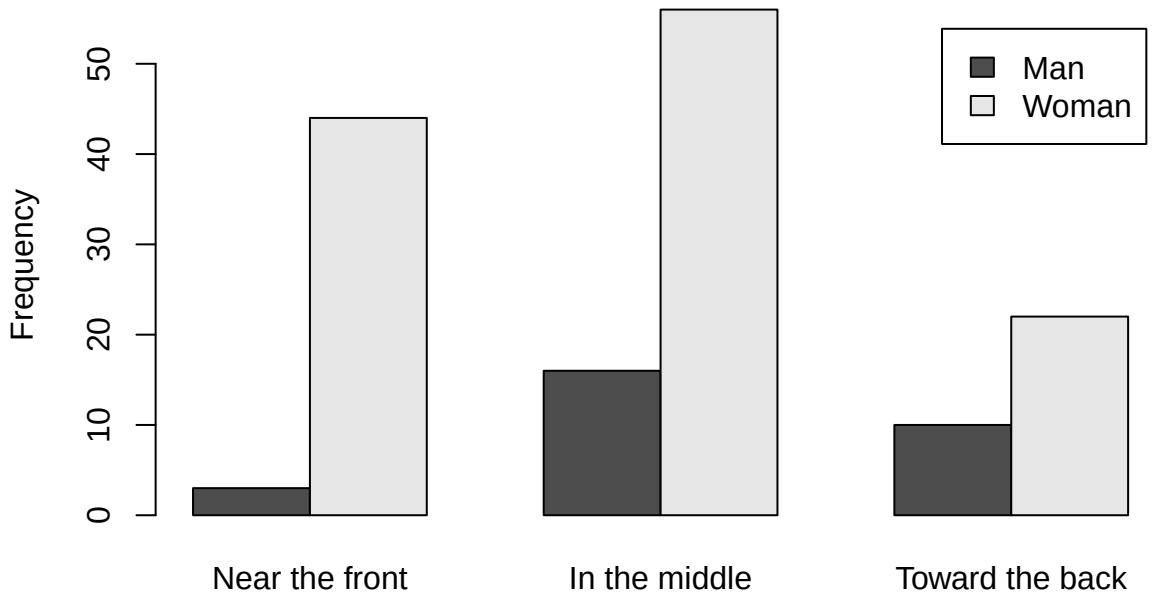
Tables are really two-dimensional matrices, where the first dimension is the row, and the second dimension is the column. We want to select the second and third row, and re-order the columns so that the second comes first, the first comes second, and the third stays the same. This can be done like this:

```
dat <- dat[c(2,3),c(2,1,3)]
dat
```

```
##
##         Near the front In the middle Toward the back
##   Man                3            16              10
##   Woman             44            56              22
```

The easiest way to plot this data is using R's `barplot` function. I set `beside = TRUE` to have the bars for Men and Women next to each other instead of the default which is stacked.

```
barplot(dat,ylab = "Frequency",beside = TRUE,legend = row.names(dat),width = .2)
```



We're now ready for the χ^2 test, using the same function as the χ^2 test for frequencies:

```
chisq.out <- chisq.test(dat)
```

The output is much like the output for the χ^2 test for frequencies. The expected frequencies are provided as a table, and can be found here:

```
chisq.out$expected
```

```
##
##         Near the front In the middle Toward the back
```

```
##    Man          9.02649      13.82781        6.145695
##    Woman       37.97351      58.17219       25.854305
```

We can extract the information we need to generate a string containing the APA style like we did for the χ^2 test for frequencies:

```
sprintf('Chi-Squared(%d,N=%d) = %5.4f, p = %5.4f',
        chisq.out$parameter,
        sum(chisq.out$observed),
        chisq.out$statistic,
        chisq.out$p.value)
```

```
## [1] "Chi-Squared(2,N=151) = 8.3941, p = 0.0150"
```

So, using $\alpha = 0.05$ it looks like where students like to sit in class is not independent of gender.

Notice that we have 2 degrees of freedom because we have 2 rows and 3 columns, so $df = (3-1)(2-1) = 2$.

This test only tells you whether or not the to factors are not independent. Typically you need to go back to the table of observations or the plot to make any inferences about what drove a significant result. From this example you can see that the ratio of Men to Women varies from front to back, with a more unbalanced ratio in the front compared to the back. The χ^2 test for independence tells you how likely you'd get something this unbalanced by chance.

14.3 Effect size and power

Calculating and interpreting effect size is exactly the same as for the χ^2 test for frequencies. ψ is used for calculating power in R:

$\psi = \sqrt{\frac{\chi^2}{N}}$

For our last example ψ is:

$\psi = \sqrt{\frac{\chi^2}{N}} = \sqrt{\frac{8.3941}{151}} = 0.2358$

And Cramer's V is:

$V = \sqrt{\frac{\chi^2}{N \times df}} = \sqrt{\frac{8.3941}{(151)(2)}} = 0.1667$

Using Cramer's V, we'd call this a small effect size.

Similarly, `power.chisq.test` works for the χ^2 test for independence:

Power for the second example is:

```
chisq.power.out <- pwr.chisq.test(w = 0.2358,N = 151,df = 2,sig.level = 0.05)
```

```
chisq.power.out$power
```

```
## [1] 0.7396396
```

The sample sized needed to get a power of 0.8 is:

```
chisq.power.out <- pwr.chisq.test(w = 0.2358,N = NULL,df = 2,sig.level = 0.05,power = 0.8)
```

```
chisq.power.out$N
```

```
## [1] 173.2807
```

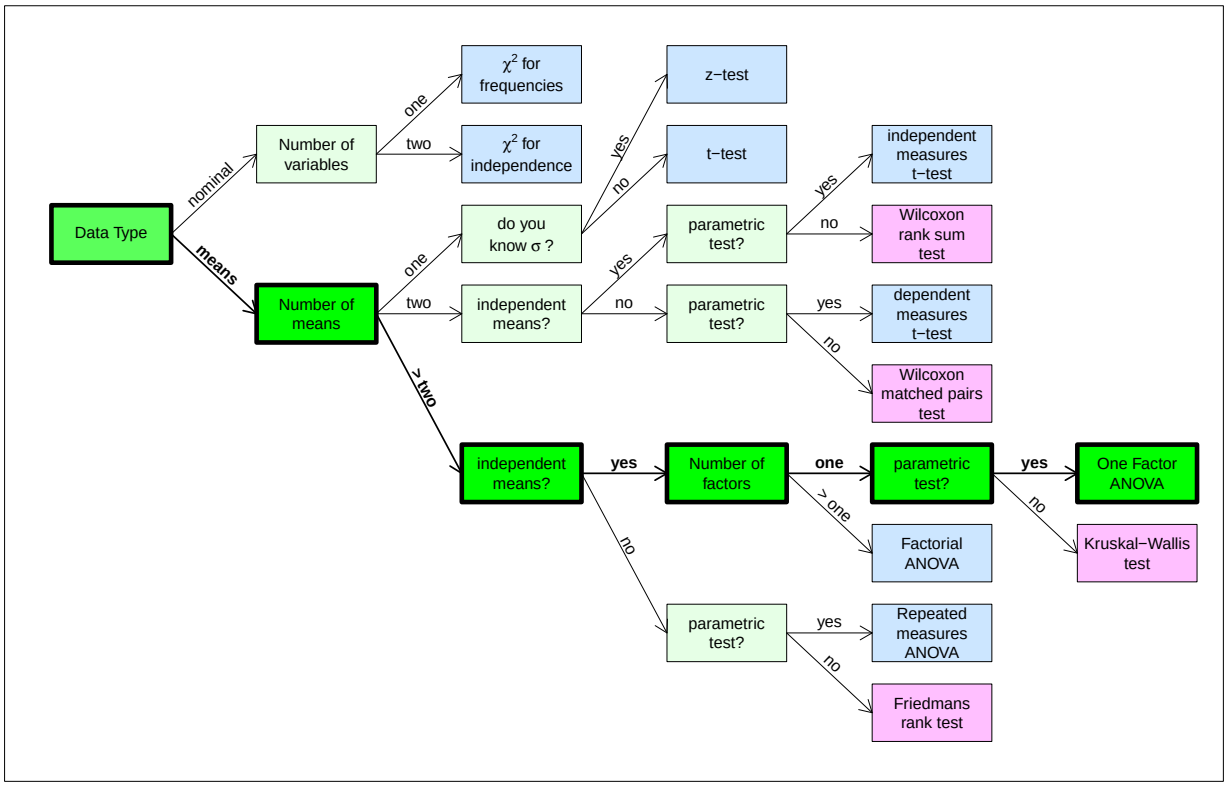
Chapter 15

ANOVA: Part 1 - The Ratio of Variances

In this chapter we'll introduce Analysis of Variance (ANOVA) the traditional way, which involves comparing ratios of variances and the F-distribution. Later on we'll get into how to conduct ANOVA tests using linear regression which is a different way of looking at the same thing.

The one-factor ANOVA is a way of comparing multiple means to each other. It's really an extension of the two-sample t-test which is for just two means.

You can find this test in the flow chart here:



In fact, you can run an ANOVA test with two means and you'll get the exact same results. You might be wondering why we even bothered with t-tests. I guess there are two reasons. First, z and t-tests are good ways of introducing the concepts of hypothesis testing, including effect sizes and power. Second, two-tailed tests are built into ANOVA, so you need a t-test for a one-tailed test. But of course you usually shouldn't run a one-tailed test anyway...

15.1 Generating a Fake Data Set

We're going to generate a random data set to work with. Creating fake data may seem like a weird (and wrong) thing to do, but it's a very powerful way of testing out your analysis methods. I recommend to my students that they create and analyze a random data set before they even start their experiment. It's not hard to do, and it lets you not only debug your code beforehand, but it also can reveal problems in your experimental design, and even let you estimate the power of your experiment.

For this example, suppose you think there might be a difference in final exam scores for a class that you've been teaching over the years. There were always the same number of students in the class, and each class had a different set of students. If there were only two years, we'd conduct a two-sample t-test for independence.

Here's some code that will generate a fake data set of scores by drawing from a normal distribution of known mean and standard deviation. We can think of the mean and standard deviation as the population parameters.

```
set.seed(10)
n <- 16  # number of students per class
k <- 5   # number of classes
mu <-70  # population mean
sigma <-8  # population standard deviation

dat <- as.data.frame(matrix(nrow = n,ncol = k))
colnames(dat) <- sprintf('Year %d',seq(1,k))
for (i in 1:k) {
  dat[,i] <- round(rnorm(n,mu,sigma))
}
```

There's a bit to unpack here. The command `set.seed(0)` sets the random number generator seed to some chosen number. Setting the seed (to any value) determines the list of randomly generated numbers to follow. If you set the seed before generating the data set you will always generate the same 'random' data set. You'll get the same list of numbers as me if you set the seed to zero.

The command `dat <- as.data.frame(matrix(nrow = n,ncol = k))` creates a data set of the desired size but full of NA's. It's away of setting aside space to be filled in later.

The command `colnames(dat) <- sprintf('Year %d',seq(1,k))` is a tricky way of generating a list of names 'Year 1', 'Year 2', etc. of length k which are used as the names of each column.

Finally, the for loop loops through k times, each time filling in the ith column with random numbers generated from a normal distribution with mean`mu` and standard deviation `sigma`. `dat[,i]` is a way of referring to every element of the ith column. `dat[i,]` will give you the elements of the ith row.

If you don't totally follow this that's OK. Just understand that it generates this data set:

Table 15.1:

Year 1	Year 2	Year 3	Year 4	Year 5
70	62	78	65	79
69	68	71	75	80
59	77	59	67	76
65	74	59	67	66
72	65	73	81	75
73	53	56	87	60
60	65	67	74	73
67	53	65	76	59
57	60	79	63	62
68	67	64	74	68
79	64	63	65	58
76	63	77	72	79
68	69	62	60	58
78	68	70	66	67
76	55	72	63	62
71	69	68	73	82

Each column was generated from a normal distribution with the same mean ($\mu_x = 70$) and standard deviation ($\sigma_x = 8$). So these differences between means are just due to random sampling. More formally, you can think of this data set as being drawn from a world where the null hypothesis is true. We write this as:

$H_0 : \mu_1 = \mu_2 = ... = \mu_k$

All population means are equal.

Let's see how different these test scores are by creating a table of summary statistics - the sample size, the mean, standard deviation and variance for each year. We'll get fancy and use the `apply` function. For example `sapply(dat,mean)` will calculate the mean of each column of dat. We'll also round to two decimal places to make things prettier. It's called 'apply' because it applies the function to every column.

```
summary <- data.frame(n = sapply(dat,length),
                      mean = round(sapply(dat,mean),2),
                      sd = round(sapply(dat,sd),2),
                      var = round(sapply(dat,var),2))
```

Here are our results:

Table 15.2:

	n	mean	sd	var
Year 1	16	69.25	6.61	43.67
Year 2	16	64.50	6.84	46.80
Year 3	16	67.69	7.06	49.83
Year 4	16	70.50	7.28	52.93
Year 5	16	69.00	8.66	75.07

The means differ by a few points. But how different are they?

We actually know how much they should differ, on average, thanks to the Central Limit Theorem. Since each sample

is from a normal distribution with a mean of $\mu_x = 70$ and standard deviation of $\sigma_x = 8$, each mean with sample size n = 16 is drawn from a normal distribution with standard deviation:

$$\sigma_{\bar{x}} = \frac{\sigma_x}{\sqrt{n}} = \frac{8}{4} = 2$$

We have 5 means. On average the standard deviation of these 5 means should be equal to 2. Let's calculate the standard deviation of our means, which we can call $s_{\bar{x}}$. We'll use `sapply` again to get the mean of each sample, and then take the standard deviation:

```
sd(sapply(dat,mean))
```

```
## [1] 2.290435
```

This number 2.2904 is kind of close to the expected value of $\sigma_{\bar{x}} = 2$.

Now, since

$$\sigma_{\bar{x}} = \frac{\sigma_x}{\sqrt{n}}$$

,

We know that

$$\sigma_x = \sigma_{\bar{x}}\sqrt{n}$$

It follows thast since $s_{\bar{x}}$ is an estimate of $\sigma_{\bar{x}}$, if we multiply $s_{\bar{x}}$ by $\sqrt{n}$ we should have an estimate of σ_x. This gives us $(2.2904)(4) = 9.1617$.

Which is kind of close to 8.

The 'V' in ANOVA is for 'Variance', so let's start thinking in terms of variance. The variance of the population, σ_x^2, is $8^2 = 64$. Instead of comparing $s_{\bar{x}}\sqrt{n}$ to estimate σ, we'll square it to get $n \cdot s_{\bar{x}}^2$ and compare it to σ_x^2. For our data set, $n \cdot s_{\bar{x}}^2$ can be calculated like this:

```
n*var(sapply(dat,mean))
```

```
## [1] 83.9375
```

Which is an estimate of the population variance $\sigma_x^2 = 64$.

This gives us a hint for how we can test our null hypothesis. When H_0 is true, each time we calculate $n \cdot s_{\bar{x}}^2$ we should get a number that is close to the population standard variance σ_x^2.

Now, suppose what happens when H_0 is false, and the population means are not all the same. This will make the variance the means larger, on average, so we should, on average, get values of $n \cdot s_{\bar{x}}^2$ that are larger than 64.

From our fake data set, we could run a hypothesis test comparing our observed value of $n \cdot s_{\bar{x}}^2 = 83.9375$ to 64 and reject H_0 if it gets big enough.

But we don't normally know the population variance σ_x^2. What can we use instead?

We have 5 samples of size 16 drawn from a normal distribution with variance $\sigma_x^2 = 64$. Look back at the table, are the variance of each of these samples around 64? They should be.

To combine these 5 variances into a single number we'll take the mean.

Using R, we first calculate the variance of each sample (using '`var` and`sapply(dat,var)`), then take the mean:

```
mean(sapply(dat,var))
```

```
## [1] 53.65917
```

This number is fairly close to $\sigma_x^2 = 64$. Notice, however, that this number shouldn't change, on average, with differences in the mean. This makes this number an estimate of the population standard deviation *even if* H_0 *is false*.

One way to think about ANOVA is that it's about comparing two numbers - the variance of the means (times n) to the mean of the variances.

The statistic we compute is the ratio of these two numbers. For our example, this ratio can be computed in one step:

```
F_obs <- n*var(sapply(dat,mean))/mean(sapply(dat,var))
F_obs
```

```
## [1] 1.564271
```

This ratio of variances is called an F statistic, which is why I've called it `F_obs` for 'F observed'.

If H_0 is true, then `F_obs` should around 1 on average since both are estimates of the same number, the population variance σ^2.

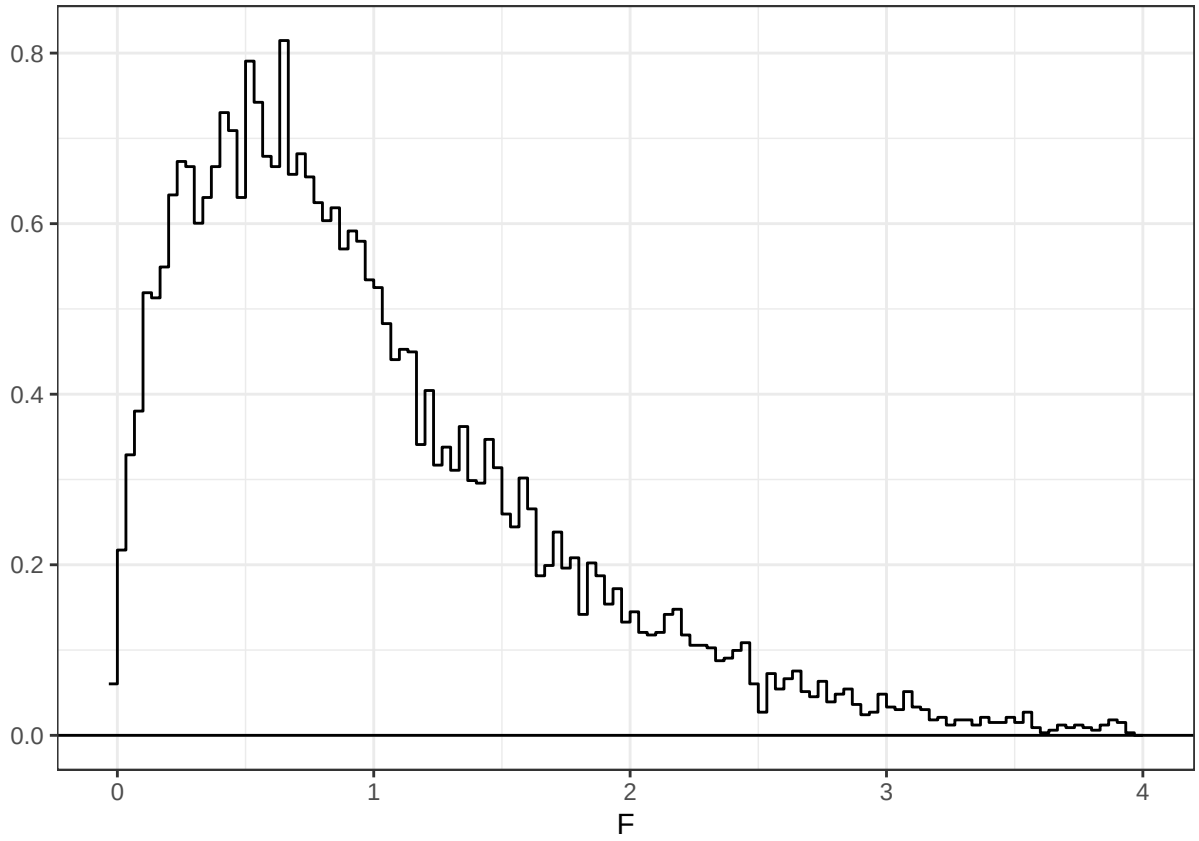
How unusual is our ratio of 1.56?

We can answer this by generating a huge set of fake data sets and calculate the F-statistic for each set.

```
nReps <- 10000  # number of fake data sets

datj <- as.data.frame(matrix(nrow = n,ncol = k))
colnames(datj) <- sprintf('Year %d',seq(1,k))

Fs <- numeric(nReps)  # allocate a vector of zeros to be filled in

for (j in 1:nReps) {
  for (i in 1:k) {
    datj[,i] <- round(rnorm(n,mu,sigma))
  }
  # Calculate the F statistic for this data set and save it in Fs[j]
  Fs[j] <- n*var(sapply(datj,mean))/mean(sapply(datj,var))
}
```

The code above loops through 10000 times, each generating a fake data set like before. For each fake data set, an F-statistic is calculated and is saved in the vector 'Fs'.

Let's look at a histogram of our 10000 F statistics. I'm hiding the code so we can keep things moving.

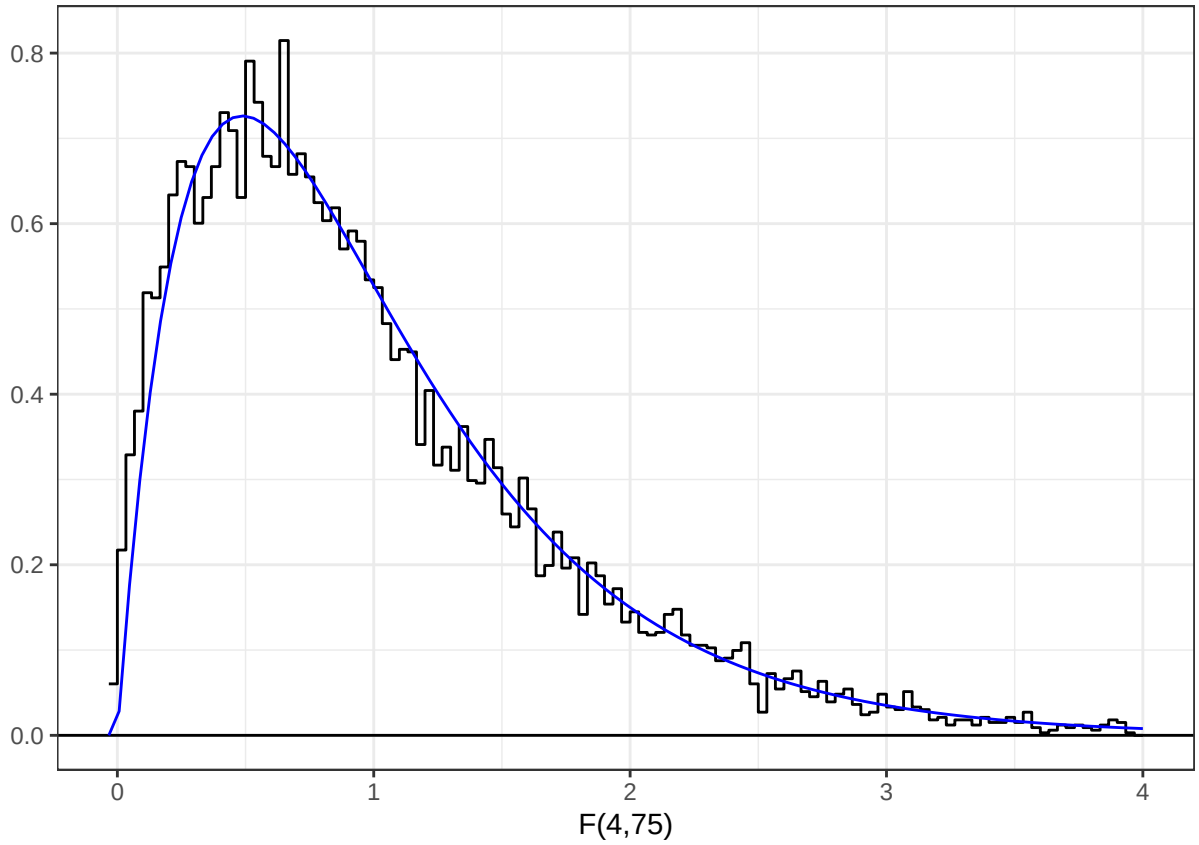


It's a strange looking distribution. First you'll notice that all values are positive - that's because we're working with variances which are always positive. Second, you'll notice that it is strongly positively skewed. Third, the mean of the distribution is around 1. This should make sense, since this is the expected ratio if H_0 is true.

This F-distribution has a known 'parameterised' shape, like the z-distribution and t-distribution. Like t, it's a family of distributions, but the F-distribution requires two separate degrees of freedom, one for the numerator and one for the denominator. Formally, the F distribution is the ratio of two χ^2 distributions. For more on this, check out this section on the chapter about how all distributions in this book are based on the normal distribution.

The df for the numerator is k-1, where k is the number of means: 5 - 1 = 4 for our example. For the denominator, which is the mean of the variances, each sample contributes n-1 degrees of freedom, so the total df for the denominator is $k(n-1) = 5(16-1) = 75$. This is the same as nk - k, which can be writen as N-k where N is the total number of scores in the experiment (nk).

You can find areas under the F distribution using R's `pf` function, which functions like the `pnorm` and the `pt` functions for normal and t-distributions. Here's the histogram of our generated F-statistics along with the known F-distribution with 4 and 75 degrees of freedom.



You can see that our randomly sampled F-statistics nicely fit the known F-distribtion. The F-statistic for our first randomly generated data set was 1.56. Where does this sit compared to the F-distribution?

R's `pf` function finds the area below a given value of F. So the area above our observed statistic is:

```
p_value <- 1-pf(F_obs,k-1,k*(n-1))
p_value
```

```
## [1] 0.1926133
```

This isn't a very unusual draw from the distribution. If we choose $\alpha = 0.05$, we wouldn't be suspicious that the null hypothesis is not true.

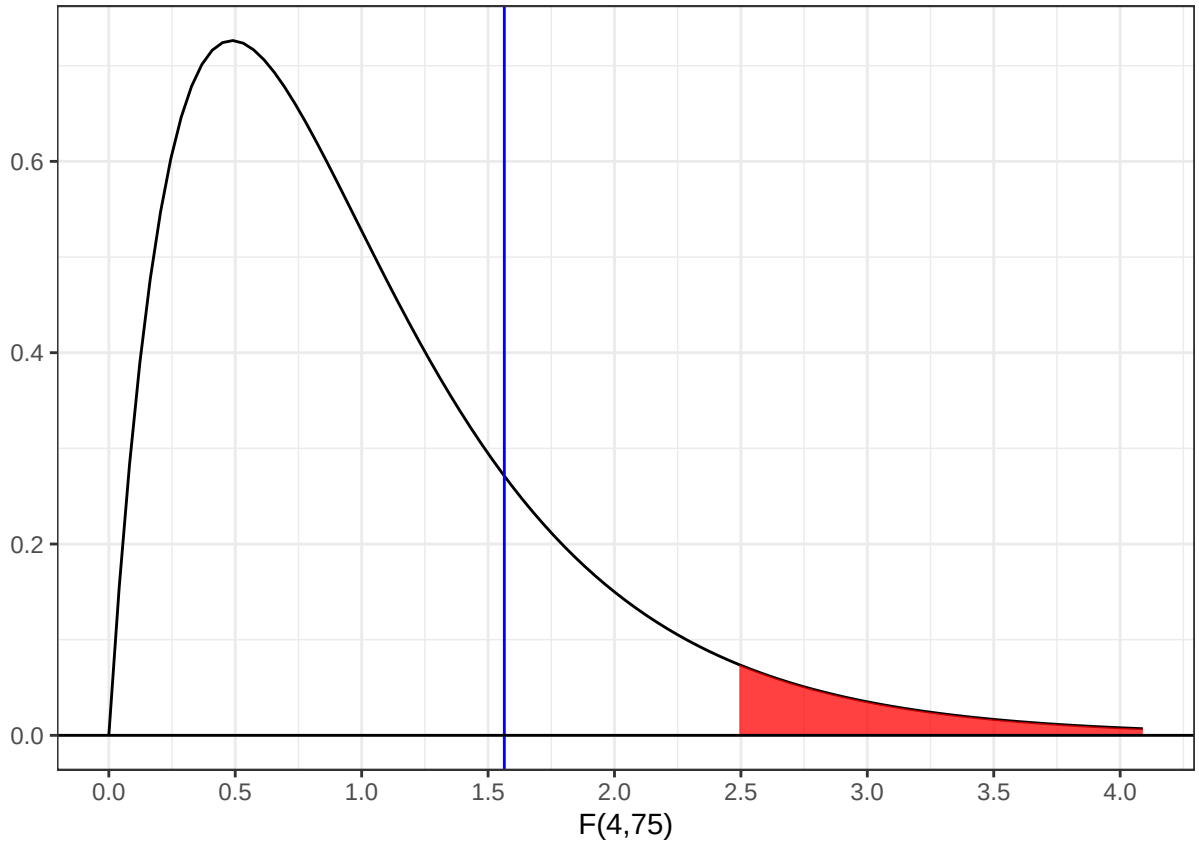
R's `qf` function is like `qnorm` and `qt` - it provides the value of F for the upper percentile. If we let $\alpha = 0.05$, the F-statistic that defines the top 5% is:

```
alpha <- .05
F_crit <- qf(1-alpha,k-1,k*(n-1))
F_crit
```

```
## [1] 2.493696
```

We need a F-statistic of 2.49 or more to reject H_0.

Here's the known F-distribution with the top 5% shaded in red, and our observed value of F as a vertical line:



For this random data set, our observed value of F does not fall in the critical region, so we'd fail to reject H_0 and we could not conclude that our samples were drawn from populations with different means.

If we were to change the random seed, we'd get a different F-statistic. One out of 20 times we'd expect to land in the rejection region by chance. We know that this would be a Type I error because we know that H_0 is true.

In fact, we did generate 10000 data sets and F-statistics. We can count the proportion of our randomly generated F-statistics that fall above the critical value:

```
mean(Fs > F_crit)
```

```
## [1] 0.0494
```

494 out of the 10000 F-statistics are greater than 2.49. This is very close to proportion of $\alpha = 0.05$.

15.2 Simulating data when H_0 is false

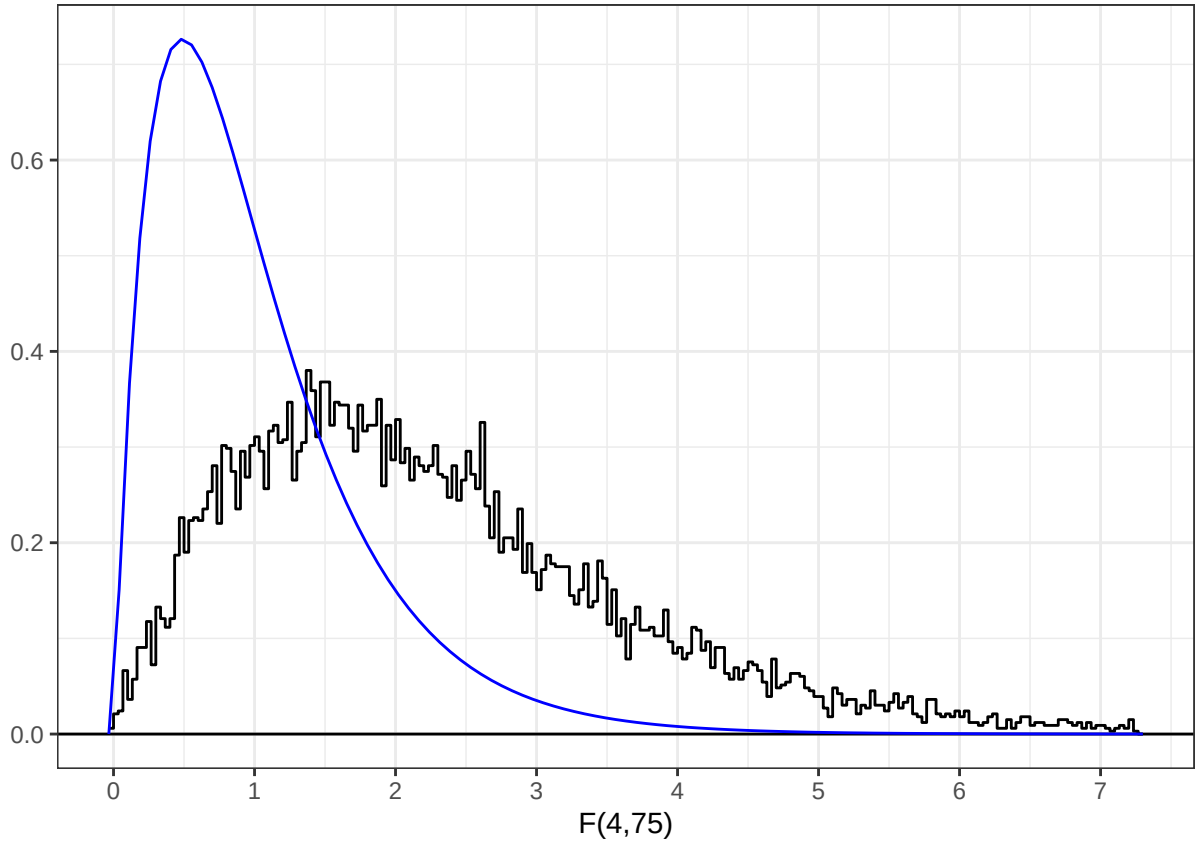
In this last part we'll generate a whole new set of fake data, but this time we'll make H_0 be false. This code is much like the code above:

```
# set mus and sigmas to be vectors for population means and sd's
mus <- rep(70,k)
sigmas <- rep(8,k)

# reset the first mean to be 75
mus[1] <- 75

datj <- as.data.frame(matrix(nrow = n,ncol = k))
colnames(datj) <- sprintf('Year %d',seq(1,k))

Fs <- numeric(nReps)  # allocate a vector of zeros to be filled in
```

```
for (j in 1:nReps) {
  for (i in 1:k) {
    # use each sample's mean and sd to generate the sample
    datj[,i] <- round(rnorm(n,mus[i],sigmas[i]))
  }
  # Calculate the F statistic for this data set and save it in Fs[j]
  Fs[j] <- n*var(sapply(datj,mean))/mean(sapply(datj,var))
}
```

The only difference is that we now have a vector of numbers for the population mean and standard deviations, called `mus` and `sigmas`. We set all of the sigmas to be the same, but we changed the first mean from 70 to 75.

Here's a histogram of our new set of randomly generated F-statistics along with the known F-distribution:



This no longer a good fit. We're generating way too many large F-statistics. This is because by changing that first mean, we've increased the numerator of F, the variance of the mean across our samples, without increasing denominator, the average mean of the variances.

Now if we calculate the proportion of F-statistics that fall above our critical value:

```
mean(Fs > F_crit)
```

```
## [1] 0.3773
```

We now reject H_0 much more often than 5% of the time.

Since H_0 is false, rejecting H_0 is the correct decision. Do you remember what we call the proportion of times that we correctly reject H_0?

It's *power*.

We just used a simulation to estimate the power of our ANOVA when we made H_0 false in a very specific way.

I hope you appreciate the 'power' of this sort of simulation. You don't need to know any math. All you need to do is generate fake data sets with some chosen set of population parameters and count the number of times that you reject H_0. This sort of procedure can be used for any sort of hypothesis test. It's also a sanity check for H_0, since if you set your population parameters so that H_0 is true (like we first did), your simulations should lead to rejections around a proportion of α.

Chapter 16

ANOVA Part 2: Partitioning Sums of Squares

The 1-Factor ANOVA compares means across at least two groups. In the last chapter we discussed the intuition that ANOVA is about comparing the variances between the means across the groups to the mean of the variances within each group. While this intuition is useful, it's not a practical way to calculate F-statistics from your data, mostly because it doesn't deal with differences in sample sizes between groups, and doesn't help to understand more complicated experimental designs.

This chapter covers the more traditional way to explain ANOVA, which is in terms of breaking down sums-of-squared deviations from means. This was the most popular way to think about ANOVA for years until recently when we've moved toward regression as a framework for ANOVA. But the sum-of-squares framework is useful to cover because it shows you how to calculate an F-statistic from scratch which you can't naturally do with regression.

In this chapter we'll define the formulas for calculating the numerator and denominators of the F-statistic and then go through the calculations in R. You can guess that R has it's own function for running ANOVAs, which we'll do at the end to check that we have the right answers.

We'll work with an example from the survey. I've always had the intuition that students in the front are more ambitious and engaged, so perhaps they are doing better in school compared to the students that sit in the middle or back of the class. Let's go to the survey data again and see if there is a difference in the student's GPA at UW when we divide the class into groups based on where they like to sit in the classroom.

First, some terminology. The thing that we're taking means of is the continuous scale (usually ratio or interval scale) 'independent measure', which is UW GPA for our example. The thing that we're using to group our students is the 'independent measure', and for ANOVA it's a nominal scale, which we call call a 'factor'. For our example, we have one factor: where students like to sit in class. The different options of our factor are called 'levels'.

Here's some code that generates a summary table based on the survey data. It uses `tapply` to calculate means, sums and standard deviations across the three levels of our factor. The slightly weird part is the call to the function `factor` which determines the order of the levels in our factor. By default, R puts things in alphabetical order, and we want to choose the order ourselves: front > middle > back

```
survey <-read.csv("http://www.courses.washington.edu/psy315/datasets/Psych315W21survey.csv")

survey$sit <- factor(survey$sit,levels = c(
  "Near the front", "In the middle", "Toward the back"), ordered = TRUE)

# define our statistics here - we'll be using them later.
ns <- tapply(survey$GPA_UW,survey$sit,function(x) sum(!is.na(x)))
means <- tapply(survey$GPA_UW,survey$sit,mean,na.rm = TRUE)
sds <- tapply(survey$GPA_UW,survey$sit,sd,na.rm = TRUE)
sems <- sds/sqrt(ns)
```

```
# stick them into a data frame for plotting
summary <- data.frame(n=ns,mean=means,sd=sds,sem=sems)
```

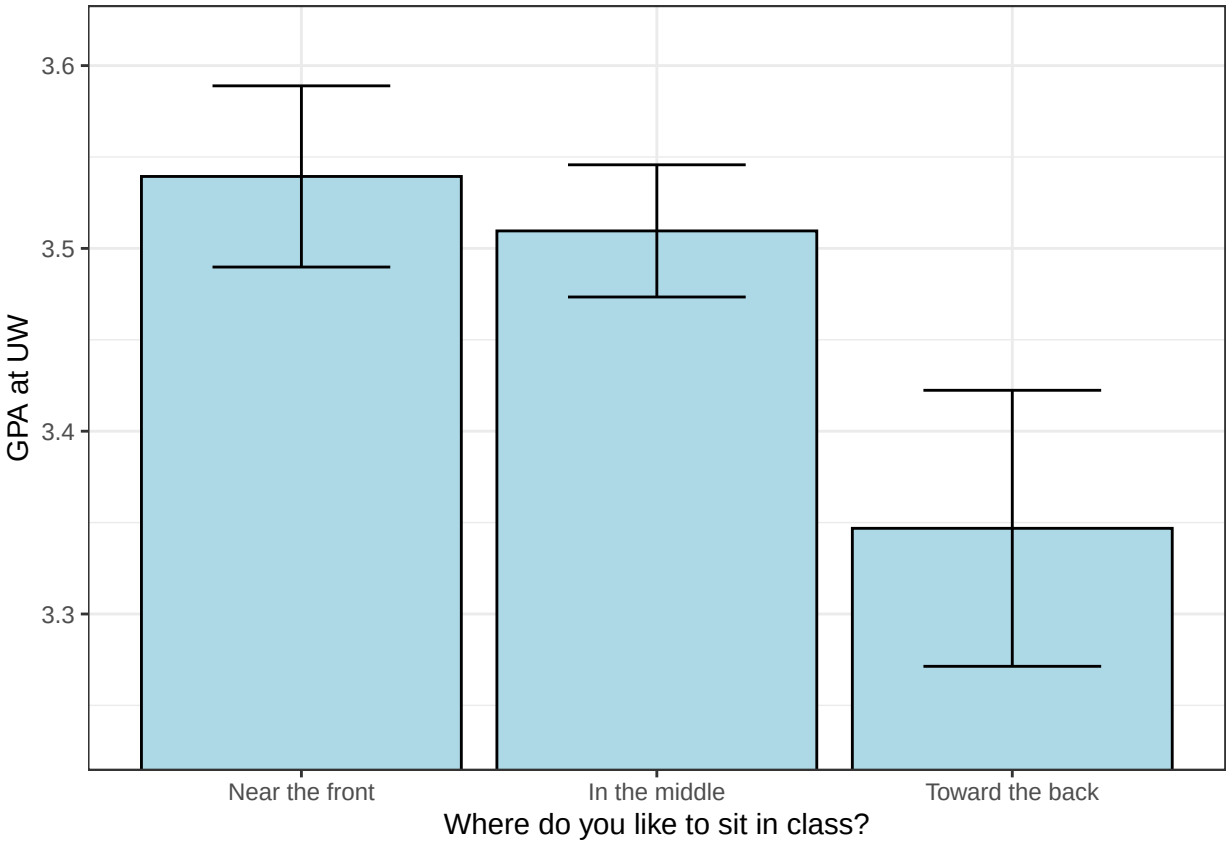
Here's the table:

Table 16.1:

	n	mean	sd	sem
Near the front	47	3.54	0.34	0.05
In the middle	71	3.51	0.30	0.04
Toward the back	32	3.35	0.43	0.08

The means do differ, but it's hard to see by how much compared to the standard errors. Let's use this table to make a bar plot of the means with error bars set to ± one standard error of the mean:

```
# Define y limits for the bar graph
ylimit <-  c(min(summary$mean-1.5*summary$sem),
             max(summary$mean+1.5*summary$sem))

# Plot bar graph with error bar as one standard error (standard error of the mean/SEM)
p <- ggplot(summary, aes(x = row.names(summary), y = mean)) +
  xlab("Where do you like to sit in class?") +
  geom_col(position = position_dodge(), color = "black", fill="lightblue")

p+geom_errorbar(aes(ymin=mean-sem, ymax=mean+sem),width = .5) +
  scale_y_continuous(name = "GPA at UW") +
  scale_x_discrete(limits = row.names(summary)) +
  coord_cartesian(ylim=ylimit) + theme_bw()
```



Remember the rule of thumb for the two-sample independent measures t-test: you need a gap in the error bars for a statistically significant difference. We use the same rule here to compare two means a time.

16.1 Familywise error

You might think that we could just run a bunch of independent measures t-tests to see which pairs of mean are significantly different. The problem is the issue of 'multiple comparisons'. If the null hypothesis is true and all three means are drawn from populations with the same mean, then the probability of rejecting any test is α. But if there are multiple tests, the probability of rejecting *one or more* of these tests becomes greater than α

If there are k levels, then there are $m = \frac{k(k-1)}{2}$ possible pairs. There are m = 6 pairs for our three-level example. If H_0 is true, then the probability of failing to reject any test is $1 - \alpha$. With m tests, the probability of failing to reject *all* of them (if they're independent) is $(1 - \alpha)^m$. So the probability of rejecting *one or more* is the opposite of failing rejecting all of them: $1 - (1 - \alpha)^m$

For our example, if we let $\alpha = 0.05$:

$$1 - (1 - \alpha)^m = 1 - (1 - 0.05)^3 = 0.1426$$

This is much higher than $\alpha = 0.05$. This means that if H_0 is true and we run all possible hypothesis tests, we'll make one or more type I errors about 14 percent of the time. This probability grows quickly with the number of levels. For an experiment with 10 levels,

$$1 - (1 - \alpha)^m = 1 - (1 - 0.05)^{10} = 0.4013$$

If we only report the significant outcomes, you can see how Type I errors creep in to the literature.

Technically this math isn't quite right because these tests aren't statistically independent. The chapter on Apriori and Post-Hoc Comparisons goes into excruciating detail about ways to deal with this issue of familywise error if we do want to make multiple hypothesis tests.

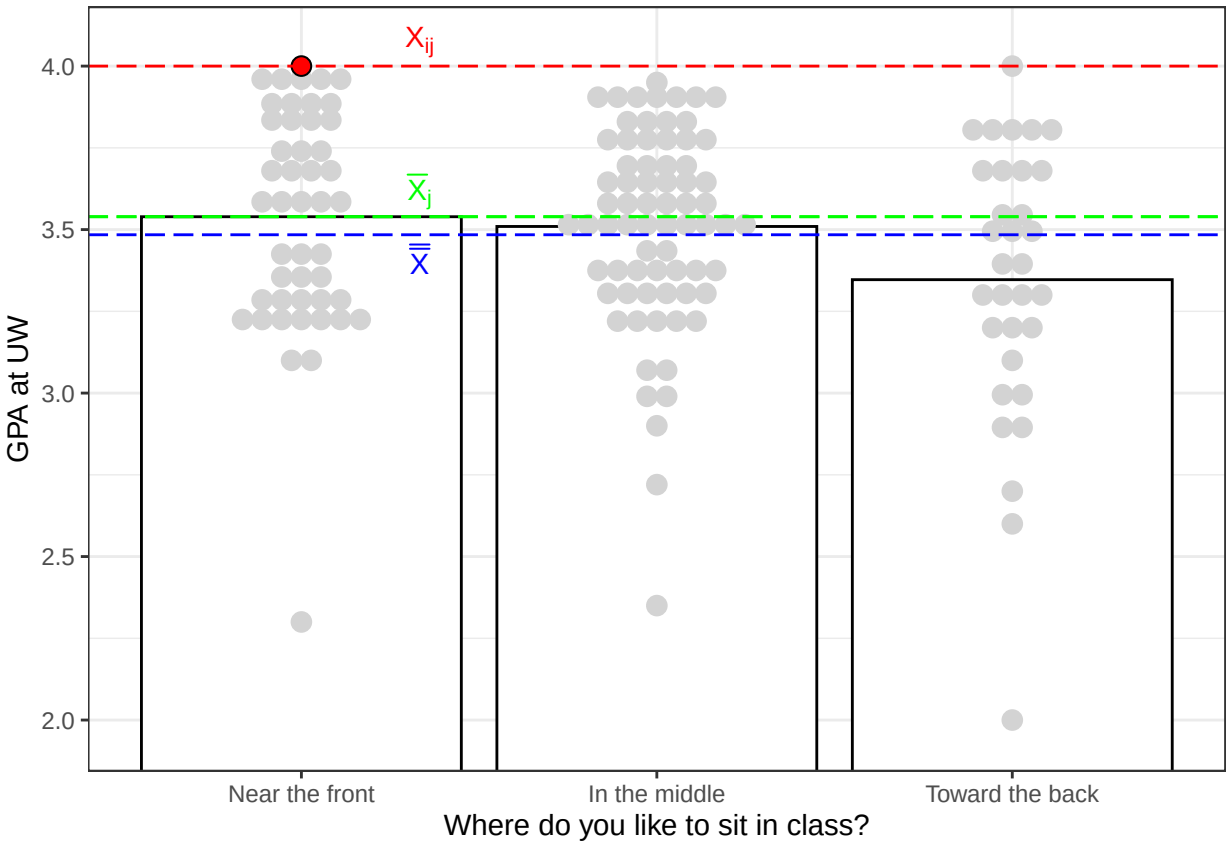
ANOVA is way of comparing all of the means to each other with one test, thereby avoiding familywise error. The drawback is that if there is a significant difference between the means, the test doesn't tell you *how* they are different. They all could be a little different from each other, or one of the means could be very different from the others.

16.2 Partitioning Sums of Squares

Back to our example. If we call our dependent measure X, let X_{ij} be the i th data point in level j. Let k be the number of levels, so j will range 1 to k. Let n_j be the sample size for level j, so i will range from 1 to n_j.

Let $\overline{X}_j$ be the mean for level j, and let $\overline{\overline{X}}$ be the *grand mean*, which is the mean of the entire data set. Note, despite it's symbol, $\overline{\overline{X}}$ will not necessarily be equal to the mean of the means unless the sample sizes are equal.

Consider the plot below which shows a bar graph of our survey data, but with each data point shown individually. I've highlighted a few values. First, I've picked out a single data point, X_{ij} which I've shown in red. The mean of the level that it came from, X_j is colored in green, and the grand mean, $\overline{\overline{X}}$ I've shown in is colored in blue



Consider how far our chosen data point is away from the grand mean: $X_{ij} - \overline{\overline{X}}$. We can separate this difference, or *deviation* into two parts:

$$(X_{ij} - \overline{\overline{X}}) = (X_{ij} - \overline{X}_j) + (\overline{X}_j - \overline{\overline{X}})$$

This is not advanced algebra, but it illustrates the point that the deviation between any data point and the grand mean is the sum of the deviation between that point the level's mean and the deviation between the level's mean and the grand mean. Where were going is that the term $(X_{ij} - \overline{X}_j)$ contributes to the mean of the variances within each level, and $(\overline{X}_j - \overline{\overline{X}})$ contributes to variance of the means between the levels.

While the above equation is trivially true, the following equation is also true:

$$\sum_{j=1}^{k}\sum_{i=1}^{n_j}(X_{ij}-\overline{\overline{X}})^2=\sum_{j=1}^{k}\sum_{i=1}^{n_j}(X_{ij}-\overline{X}_j)^2+\sum_{j=1}^{k}\sum_{i=1}^{n_j}(\overline{X}_j-\overline{\overline{X}})^2$$

The double sums means that we're summing across the entire data set by first summing across the samples within each level, and then summing that sum across the levels.

You can't normally just square terms on both sides of an equation and add them up, but it turns out that it works for this special case with means. Each of these three terms are called 'sums of squares' or *SS*, and each has it's own name and meaning.

The first term is called *sums of squares total*. We can be sloppy and drop the double sum, assuming that we're summing across the whole data set. It is written as:

$$SS_{total}=\sum_{ij}(X_{ij}-\overline{\overline{X}})^2$$

.

The second term is called *sums of squared_within*, written as:

$$SS_{within}=\sum_{ij}(X_{ij}-\overline{X}_j)^2$$

.

It's called *within* because it's the sum of squared of the deviations of the mean *within* each level.

The last term is called *sums of squared between*. If you look at the inner sum, it's the sum of n_j values of the same thing, so for each level, $\sum_{i=1}^{n_j}(\overline{X}_j-\overline{\overline{X}})^2=n_j(\overline{X}_j-\overline{\overline{X}})^2$

sums of squared between can therefore be written as:

$$SS_{between}=\sum_{j}n_j(\overline{X}_j-\overline{\overline{X}})^2$$

It's called *between* because it's the sums of squared of the means *between* the levels (multiplied by the sample size).

Using the formula above:

$$SS_{total}=SS_{within}+SS_{between}$$

Remember in the last chapter, where we discussed ANOVA as the ratio of the variance between groups and the variance within each group. The *between* partition is a measure of the variability across the means, which will be part of the numerator of the F-statistic. The *within* partition is a measure of the variability within each level and will contribute to the denominator of the F-statistic.

All we need to do is divide each of these sums of squares by their degrees of freedom and we can get variances. In the context of ANOVA, variance is called 'mean squared error', or *MS*, since it's (sort of) the mean of the sums of squared deviation.

If N is the total sample size ($\sum n_j$), then the total variance, which we call *mean squares total* is:

$$MS_{total}=\frac{\sum(X_{ij}-\overline{\overline{X}})^2}{N-1}$$

.

For the *within* partition, each mean contributes n_j-1 degrees of freedom, so the degrees of freedom added across all k levels is $\sum n_j-1=N-k$. *mean squares within* is therefore:

$$MS_{within}=\frac{\sum(X_{ij}-\overline{X}_j)^2}{N-k}$$

If the sample sizes are equal then the above equation simplifies to the mean of the variances within each group, which you might recognize this from the last chapter.

For the *between* partition, it's the variance of only k means, so the degrees of freedom is *k-1*. $MS_{between}$ is written as:

$$MS_{between} = \frac{\sum n_j(\overline{X}_j - \overline{\overline{X}})^2}{k-1}$$

You might recognize that this from the last chapter as the variance of the means multiplied by the sample size, except that this formula allows for different sample sizes.

16.2.1 Calculating F

The F-statistic is the ratio of $MS_{between}$ and MS_{within}:

$$F = \frac{SS_{between}/(k-1)}{SS_{within}/(N-k)} = \frac{MS_{between}}{MS_{within}}$$

and the degrees of freedom are $k-1$ and $N-k$.

Notice that just like the sums of squares, the degrees of freedom also add up:

$$df_{total} = df_{within} + df_{between}$$

$$N - 1 = (N-k) + (k-1)$$

It's convenient to write all of this in a summary table like this:

We're ready to calculate the F-statistic from our data 'by hand', meaning we'll use R to explicitly calculate each step.

16.2.2 Calculating SS_{total}

We often need to calculate the sums of squared deviation of things from their mean. When you do things often enough it's useful to create your own function to do this. Here's a quick way to create a function of your own using the `function` command:

```
SS = function(x)  sum((x-mean(x,na.rm = T))^2,na.rm = T)
```

We've defined a function called 'SS' which takes in a single variable, x, and calculates the sums of squared deviation of x from the mean of x. To run this function we can send in any variable - it doesn't need to be called 'x'. It's just called 'x' inside the function. For example, now that we've defined `SS`, here is the sums of squared deviation of the numbers 1, 2 and 3:

```
SS(c(1,2,3))
```

Table 16.2:

	df	SS	MS	F
Between	$k-1$	$\sum_j n_j(\overline{X}_j - \overline{\overline{X}})^2$	$\frac{SS_{between}}{df_{between}}$	$\frac{MS_{between}}{MS_{within}}$
Within	$N-k$	$\sum_{ij}(X_{ij} - \overline{X}_j)^2$	$\frac{SS_{within}}{df_{within}}$	
Total	$N-1$	$\sum_{ij}(X_{ij} - \overline{\overline{X}})^2$		

```
## [1] 2
```

Now that we've defined `SS`, SS_{total} can be calculated by:

```
SS_total <- SS(survey$GPA_UW)
SS_total
```

```
## [1] 18.25525
```

df_{total} can be calculated by:

```
N <-sum(ns)
df_total <- N-1
df_total
```

```
## [1] 149
```

We can start filling in our summary table, replacing our equations with numbers:

16.2.3 Calculating SS_{within}

We want to calculate the sums of squared deviation of each score from the mean of the level that it came from. Recall from above we used the `tapply` function to calculate the mean and standard deviation of each level, but there is no built-in function for sums of squares.

Now we can use `tapply` to get SS for each level of 'sit':

```
tapply(survey$GPA_UW,survey$sit,SS)
```

```
##  Near the front   In the middle Toward the back
##        5.311481        6.496487        5.655087
```

To calculate SS_{within} we add up these numbers:

```
SS_within <- sum(tapply(survey$GPA_UW,survey$sit,SS))
SS_within
```

```
## [1] 17.46306
```

The df_{within} and MS_{within} can be calculated as:

```
k <- length(ns)  # the number of levels
k
```

```
## [1] 3
```

```
df_within <- N-k
df_within
```

```
## [1] 147
```

```
MS_within = SS_within/df_within
MS_within
```

Table 16.3:

	df	SS	MS	F
Between	$k-1$	$\Sigma_j\, n_j(\overline{X}_j - \overline{\overline{X}})^2$	$\frac{SS_{between}}{df_{between}}$	$\frac{MS_{between}}{MS_{within}}$
Within	$N-k$	$\Sigma_{ij}\,(X_{ij} - \overline{X}_j)^2$	$\frac{SS_{within}}{df_{within}}$	
Total	149	18.2553		

```
## [1] 0.1187963
```

Here are the values for *within* in our summary table:

16.2.4 Calculating $SS_{between}$

$SS_{between}$ can be calculated by first calculating the grand mean, then calculating the sums of squared deviations of the means from the grand mean, then multiplying by each sample size and finally adding it all up.

With R, if you multiply two vectors of the same length as we do here: `ns*(means-grand_mean)^2`, you get the 'element by element' product, meaning that the first elements get multiplied together, followed by the second etc.

```
grand_mean = mean(survey$GPA_UW,na.rm = T)
SS_between = sum(ns*(means-grand_mean)^2)
SS_between
```

```
## [1] 0.7921983
```

And here's how to calculate $df_{between}$ and $MS_{between}$:

```
df_between <- k-1
df_between
```

```
## [1] 2
```

```
MS_between <- SS_between/df_between
MS_between
```

```
## [1] 0.3960992
```

Here's where these values go in the table:

16.2.5 Calculating F and the p-value

Our F-statistic is the ratio of $MS_{between}$ and MS_{within}. The p-value can be found with the `pf` function:

```
F_stat <- MS_between/MS_within
F_stat
```

```
## [1] 3.334272
```

```
p_value <- 1-pf(F_stat,df_between,df_within)
p_value
```

```
## [1] 0.03835556
```

Here's the completed table, with the p_value

You can check in the table that the SS's add up:

$$SS_{total} = SS_{between} + SS_{within}$$

$$18.2553 = 0.7922 + 17.4631$$

Table 16.4:

	df	SS	MS	F
Between	$k-1$	$\Sigma_j\, n_j(\overline{X}_j - \overline{\overline{X}})^2$	$\frac{SS_{between}}{df_{between}}$	$\frac{MS_{between}}{MS_{within}}$
Within	147	17.4631	0.1188	
Total	149	18.2553		

Table 16.5:

	df	SS	MS	F
Between	2	0.7922	0.3961	$\frac{MS_{between}}{MS_{within}}$
Within	147	17.4631	0.1188	
Total	149	18.2553		

Table 16.6:

	df	SS	MS	F	p-value
Between	2	0.7922	0.3961	3.3343	0.0384
Within	147	17.4631	0.1188		
Total	149	18.2553			

And that the df's add up:

$$df_{total} = df_{between} + df_{within}$$
$$149 = 2 + 147$$

16.3 APA format

Using APA format and $\alpha = 0.05$, we can say:

There is a significant difference between the GPAs at UW across the 3 levels of where students like to sit. F(2,147)=3.3343, p = 0.0384

16.4 Conducting ANOVA with R using `anova` and `lm`

You'd never go through this to conduct an ANOVA with your own data, although it would always work. Instead, the R command is very short. Here it is:

```
anova.out <- anova(lm(GPA_UW ~ sit,data = survey))
anova.out
```

```
## Analysis of Variance Table
##
## Response: GPA_UW
##            Df  Sum Sq Mean Sq F value  Pr(>F)
## sit         2  0.7922  0.3961  3.3343 0.03836 *
## Residuals 147 17.4631  0.1188
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

That's it. I suppose this book could be a lot shorter if I just skipped all of the intuition and explanatory stuff. But hopefully you'll appreciate what's going on under the hood before you run your next ANOVA.

The one line of code above actually runs two functions, first the `lm` function and then the output of that is passed into the `anova` function. `lm` stands for 'linear model' which is the basis of linear regression. In a later chapter we'll learn why ANOVA is just regression but for now we'll just hide that fact and pass the output of the regression into `anova` which generates the table that matches the one we did by hand.

The only thing different about the table from `anova` and the one we did by hand is that the independent variable, 'sit' is shown as the header for the between factor, and `Residuals` replaces `Within`. With regression, 'residuals' is the term used for stuff left over after fitting your data with a model. For now, just think of this as `within` level variability.

Also, 'Total' is missing. That's OK because 'Total' isn't actually used in the calculations, but it is useful to check your math and see if SS's and df's add up.

Here's how to pull out the numbers in the output of anova and convert them into APA format:

```
sprintf('F(%d,%d) = %5.4f, p = %5.4f',
        anova.out$Df[1],anova.out$Df[2],anova.out$`F value`[1],anova.out$`Pr(>F)`[1])
```

```
## [1] "F(2,147) = 3.3343, p = 0.0384"
```

By the way, 'lm' by default drops scores that have NA's in either the dependent or independent variable. The default is `na.action = na.omit`.

You can play with the data and compare any interval scale dependent variable to a nominal scale independent variable.

For example, This tests whether the ages of the student varies with handedness:

```
anova2.out <- anova(lm(age ~ hand,data = survey))
anova2.out
```

```
## Analysis of Variance Table
##
## Response: age
##            Df  Sum Sq Mean Sq F value Pr(>F)
## hand        1    2.18  2.1848  0.2924 0.5895
## Residuals 149 1113.48  7.4730
```

Not surprisingly, it doesn't.

Notice that $df_{between} = 1$. This means that there are two levels. How else could we have run this hypothesis test?

That's right, the independent measures t-test.

16.4.1 Comparing the t-test to ANOVA for two means

Here's how to run a t-test on the same data. ANOVAs are by design two-tailed tests so we'll set `alternative = 'two.sided'` (which is the default).

```
x <- survey$age[survey$hand=="Left"]
y <- survey$age[survey$hand=="Right"]
t.test.out <- t.test(x,y,alternative = 'two.sided',var.equal = T)
sprintf('t(%d)=%5.4f, p = %5.4f',t.test.out$parameter,t.test.out$statistic,t.test.out$p.value)
```

```
## [1] "t(149)=-0.5407, p = 0.5895"
```

We get the exact same p-value (0.5895). In fact,there's an interesting relationship between t and the F-distribution for 1 degree of freedom in the numerator:

$$F(1, df) = t(df)^2$$

Where df is the same degrees of freedom which, for two levels, is always N-1.

This means that if you generate a random set of t-statistics and square each sample, the histogram would look like the F-distribution with 1 degree of freedom in the numerator.

This is true for our example:

$$F = 0.2924 = (-0.5407)^2 = t^2$$

So now when you sit through a talk or read a results section and see "F(1,_) = ", you know they're comparing two means and could have run a t-test. Why not always run an ANOVA? The only good reason I can think of is that the Welch version of the t-test allows you to account for unequal variances in the populations. To my knowledge there isn't a Welch equivalent for ANOVA. I don't know of any other advantages of the t-test. You can even run a one-tailed t-test by doubling the p-value from ANOVA, right?

In a more mathematical statistics course you'd go through a bunch of derivations about probability distribution functions. It turns out that our parametric probability distributions, t, F, and χ^2 can all be derived by manipulating the standard normal, or z distribution. For example, χ^2 distributions come from squaring and summing values from the z-distribution. Variances are ratios of scaled χ^2 distributions. So it all goes back to our friend the normal distribution. For more on this, check out the chapter in this book that shows how to derive all of these distributions from the standard normal.

16.5 Effect Size for ANOVA

Remember, an effect size is a measure that can be use to compare results across experiments with different designs and different numbers of subjects. There are several measures of effect size for an ANOVA. Most software packages will spit out more than one. Each has their own advantage, and the field has not settled on any one in particular.

Here's a walk through the most common measures:

16.6 Eta squared, η^2

The simplest measure of effect size for ANOVA is η^2, or 'eta squared'. It's simply the ratio of $SS_{between}$ to SS_{total}:

$$\eta^2 = \frac{SS_{between}}{SS_{total}}$$

Remember, $SS_{total} = SS_{between} + SS_{within}$. So η^2 is the proportion of the total sums of squares that is attributed to the difference between the means. From our example of GPA's and choice of sitting in class:

$$\eta^2 = \frac{0.7922}{18.2553} = 0.0434$$

Since η^2 is a proportion it ranges between zero and one. If $\eta^2 = 0$, then $SS_{between} = 0$. This means that there is no variance between the means, which means that all of our means are the same.

If $\eta^2 = 1$, then $SS_{between} = SS_{total}$, which means that $SS_{within} = 0$. That means that there is no variance within the groups, and all of the total variance is attributed to the variance between groups.

η^2 is simple, commonly used, but tends to overestimate effect size for larger number of groups. That's because $SS_{between}$ grows with the number of groups which is a problem because effect sizes shouldn't depend on your experimental design.

16.7 Omega squared, ω^2

'omega squared', or ω^2, corrects for biases in η^2 by taking into account the number of groups. The formula isn't very intuitive:

$$\omega^2 = \frac{SS_{between} - df_{between} MS_{within}}{SS_{total} + MS_{within}}$$

ω^2 is always smaller than η^2.

For our example on GPAs:

$$\omega^2 = \frac{0.7922 - (2)(0.1188)}{17.4631 + 0.1188} = 0.0302$$

16.8 Cohen's *f*

A measure of effect size that is commonly used when talking about power is called 'Cohen's' *f*, which is:

$$f = \sqrt{\frac{SS_{between}}{SS_{within}}}$$

With a little algebra, and knowing that $SS_{total} = SS_{between} + SS_{within}$, you can show that Cohen's f is related to η^2 by:

$$f = \sqrt{\frac{\eta^2}{1-\eta^2}}$$

From our example:

$$f = \sqrt{\frac{0.7922}{17.4631}} = 0.213$$

Older publications don't report Cohen's f, but it is easily calculated from the reported value of F and its degrees of freedom. With a little algebra you can show that:

$$f = \sqrt{F\frac{df_{between}}{df_{within}}}$$

From our example:

$$f = \sqrt{3.3343\frac{2}{147}} = 0.213$$

Cohen's f is a monotonic transformation of η^2. That is, Cohen's f grows with η^2. Cohen's f has a minimum of zero but has no maximum.

16.9 Calculating Effect Sizes from the output of `anova`

R's `anova` function doesn't provide effect sizes, but they're easy to calculate once we've extracted the relevant numbers. It's easiest to first pull out the values into variables and then calculate:

```
SS_between <- anova.out$`Sum Sq`[1]
SS_within  <- anova.out$`Sum Sq`[2]
df_between <- anova.out$Df[1]
df_within <- anova.out$Df[2]
MS_between <- anova.out$`Mean Sq`[1]
MS_within <- anova.out$`Mean Sq`[2]
SS_total <- SS_between+SS_within # right?
k <- df_between + 1

eta_squared <- SS_between/SS_total
omega_squared <- (SS_between -df_between*MS_within)/(SS_total+MS_within)
cohens_f <- sqrt(SS_between/SS_within)

sprintf('eta_squared:   %5.4f',eta_squared)
```

```
## [1] "eta_squared:   0.0434"
```

```
sprintf('omega_squared: %5.4f',omega_squared)
```

```
## [1] "omega_squared: 0.0302"
```

```
sprintf('cohens_f:      %5.4f',cohens_f)
```

```
## [1] "cohens_f:      0.2130"
```

We say that effect sizes for Cohen's f around 0.1 are 'small', 0.25 are 'medium', and 0.4 and above are 'large'. These correspond to η^2 values of 0.01, 0.06, and 0.14:

Table 16.7: Cohen's f

small	0.10
medium	0.25
large	0.40

For our example on GPAs, our value of Cohen's f (0.213) is considered to be a medium effect size.

16.10 Relating η^2 and Cohen's f to the F-statistic

As we've discussed, η^2, Cohen's f and the F statistic are all related. Importantly, all measures of effect size for ANOVA can be computed from the same three pieces of information: The F statistic, $df_{between}$ and df_{within}.

This means that you can calculate the Cohen's f and η^2 from any reported F-statistic, even if the authors didn't explicitly provide it. This is useful because effect sizes weren't commonly reported until a couple of decades ago.

Here's a table showing the relation between η^2, Cohen's f, and the F statistic:

Table 16.8:

	η^2	f	F
η^2	$\eta^2 = \frac{SS_{between}}{SS_{total}}$	$\eta^2 = \frac{f^2}{1+f^2}$	$\eta^2 = \frac{F\frac{df_{between}}{df_{within}}}{1+F\frac{df_{between}}{df_{within}}}$
f	$f = \sqrt{\frac{\eta^2}{1-\eta^2}}$	$f = \sqrt{\frac{SS_{between}}{SS_{within}}}$	$f = \sqrt{F\frac{df_{between}}{df_{within}}}$

16.11 Power for ANOVA

The concept of power for ANOVA is the same as for the t-test or for the Chi-squared test: it's the probability of correctly rejecting the null hypothesis. Calculating power relies on calculating the area under the probability density function for the distribution of F values when the null hypothesis is false. This distribution is called the 'noncentral F distribution', and is a generalization of the regular F-distribution.

Software packages vary in what values are needed to calculate power for ANOVA, but they're all variants of the same information. For example, to calculate achieved power, the free app 'G*Power' requires the total sample size, the number of groups, and Cohen's f. Matlab requires df_{within} and $df_{between}$ and the 'non-centrality parameter' which is Cohen's F multiplied by the total sample size squared. And as you'll see below, R's `pwr.f2.test` requires $df_{between}$, df_{within} and Cohen's f squared.

Technically, using the noncentral F distribution to calculate power assumes a balanced design (equal sample sizes across groups). However the noncentral F distribution is commonly used to calculate power for unbalanced designs.

You should keep in mind that your power calculations will be inflated by an amount that depends on how unbalanced your design is.

Here's how `pwr.f2.test` works to calculate the observed power for our example on GPAs. For some reason u is used for $df_{between}$ and v is used for df_{within}. $f2$ is the square of Cohen's f.

```
power.anova.out <- pwr.f2.test(u = df_between,v = df_within,f2 = cohens_f^2,sig.level = .05)
sprintf('Observed power: %5.4f',power.anova.out$power)
```

```
## [1] "Observed power: 0.6327"
```

To find the sample size needed to get a desired level of power is a little different. The sample size isn't explicitly entered into `pwr.f2.test`, but since df_{within} = N-k, if we can set `v = NULL` and use the fact that N = df_{within} + k:

```
power.anova.out <- pwr.f2.test(u = df_between,v = NULL,f2 = cohens_f^2,sig.level = .05,power=0.8)

N_power = power.anova.out$v+k
sprintf('Total sample size needed for a power of 0.8: %d, which is %d observations per group.',round(N_pow
```

```
## [1] "Total sample size needed for a power of 0.8: 215, which is 72 observations per group."
```

As mentioned above, ANOVAs are useful because it's a single test for the difference between multiple means, which avoids familywise error. But usually we want to know more than the inference that the population means are different. We usually want to actually compare means or groups of means. The next chapter, Apriori and Post-Hoc Comparisons, covers how to conduct multiple tests while controlling for the number of Type I errors.

Chapter 17

Apriori and Post-Hoc Comparisons

This chapter is about how to test hypotheses on data from ANOVA designs that are more specific than the omnibus test which just tests if the means are significantly different from each other. Examples include comparing just two of the means, or comparing one mean (e.g. a control condition) to all of the other means.

The main issue here is familywise error, discussed in the last chapter, which is the fact that the probability of making one or more Type I errors increases with the number of hypothesis tests you make. For example, if you run 10 independent hypothesis tests on your results, each with an alpha value of 0.05, the probability of getting at least one false positive would be:

$$1 - (1 - 0.05)^{10} = 0.401$$

This number, 0.4013, is called the 'familywise error rate' or *FWER* and is clearly unacceptably high. The methods described in this chapter cover the various ways to control, or correct for, familywise error. The more tests you run, the greater the FWER.

Specific hypothesis tests on ANOVA data fall into two categories, 'A Priori' and 'post-hoc'.

A Priori tests are hypothesis tests that you planned on running before you started your experiment. Since there are many possible tests we could make, setting aside a list of just a few specific A Priori tests lets us correct for a much lower familywise error rate.

Post-hoc tests are hypothesis tests that you run after looking at your data. For example, you might want to go back and see if there is a significant difference between the highest and lowest means. Under the null hypothesis, the probability of rejecting a test on the most extreme difference between means will be much greater than α. Or, perhaps we want to go crazy and compare all possible pairs of means. Since there are many tests that you *could* have run, even if you only pick a few, you need to correct for a larger FWER for post-hoc tests.

17.1 One-Factor ANOVA Example:

We'll go through A Priori and post-hoc tests with an example. Suppose you want to study the effect of background noise on test score. You randomly select 10 subjects for each of 5 conditions and have them take a standardized reading comprehension test with the following background noise: silence, white noise, rock music, classical music, and voices.

Throughout this chapter we'll be referencing this same set of data. You can access it yourself at:

http://courses.washington.edu/psy524a/datasets/AprioriPostHocExample.csv

Your experiment generates the following statistics:

Table 17.1:

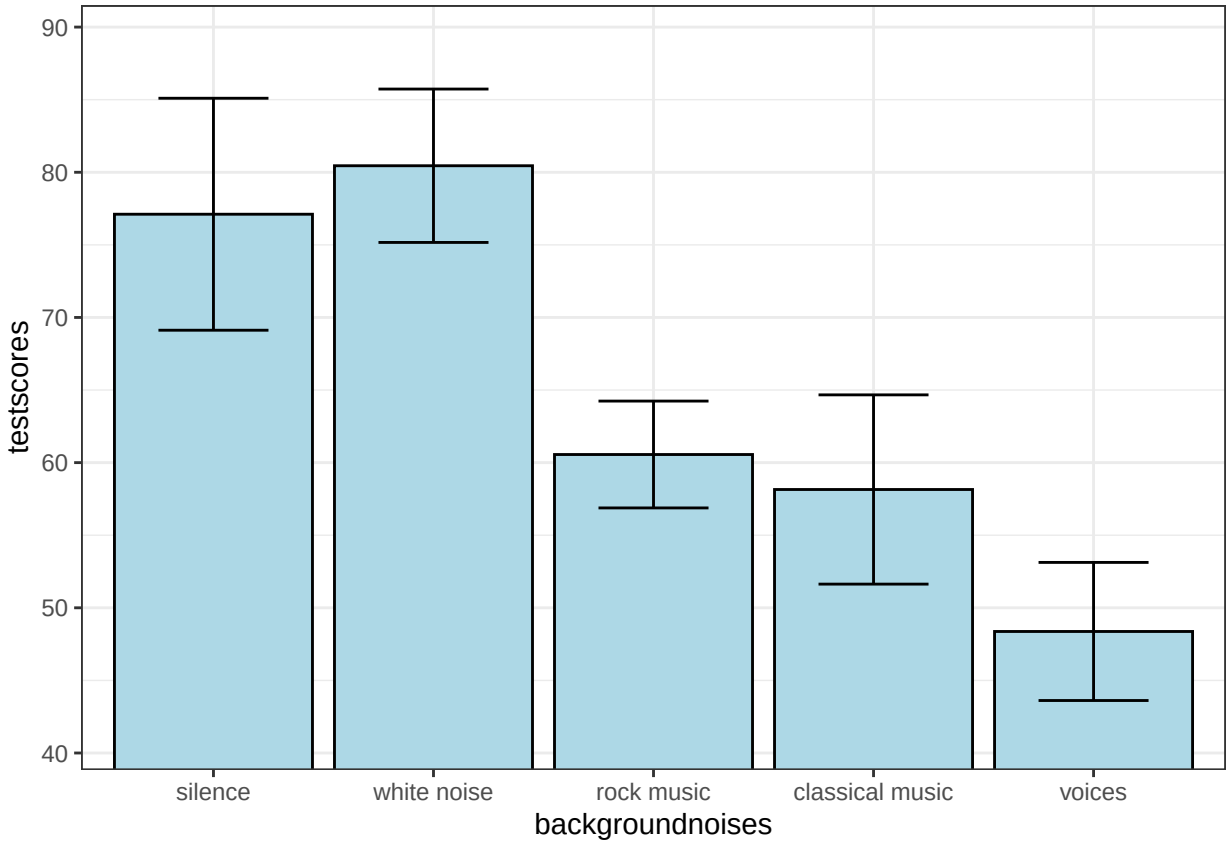
	mean	n	sd	sem
silence	77.11	10	25.26436	7.989291
white noise	80.45	10	16.70617	5.282955
rock music	60.56	10	11.63856	3.680435
classical music	58.15	10	20.61112	6.517809
voices	48.37	10	15.04113	4.756424

The results of the ANOVA are:

Table 17.2:

	df	SS	MS	F	
Between	4	7285.217	1821.3042	5.3445	p =
Within	45	15335.044	340.7788		
Total	49	22620.261			

Here's a plot of the means with error bars as the standard error of the mean:



17.2 A Priori Comparisons

An A Priori test is a hypothesis test that you had planned to make *before* you conducted the experiment. They're sometimes called *planned comparisons*.

17.2.1 t-test for two means

The simplest A Priori tests is a comparison between two means. In our example, suppose before we ran the experiment we had the prior hypothesis that there is a difference in mean test score between the *silence* and the *voices* conditions. This leads to comparing the means $\overline{X}_1 = 77.11$ and $\overline{X}_5 = 48.37$

You'd think that we would simply conduct an independent measures two-tailed t-test using these two group means and variances, while ignoring all of the other conditions (and sometimes this is what people do). But since we have MS_{within}, we should use this value since it's a better estimate of the population variance than the pooled variance from just two groups (assuming homogeneity of variance).

The old t-statistic was (since we have equal sample size, n):

$$t = \frac{\overline{X}_1 - \overline{X}_5}{\sqrt{\frac{s_1^2 + s_5^2}{n}}} = \frac{77.11 - 48.37}{\sqrt{\frac{25.3^2 + 15^2}{10}}} = 3.091$$

With $df = (n-1) + (n-1) = 2n - 2 = 18$.

However, we have mean-squared error within (MS_w) from the ANOVA, which is a better estimate of our population variance than s_1^2 and s_5^2. So we'll use it instead of the pooled variance:

$$t = \frac{\overline{X}_1 - \overline{X}_5}{\sqrt{\frac{2MS_w}{n}}}$$

The degrees of freedom is now N-k, since this is the df for SS_{within}.

For our example:

$$t = \frac{77.11 - 48.37}{\sqrt{\frac{(2)340.8}{10}}} = 3.4813$$

With df =N-k = 50 - 5 = 45, the p-value of t is 0.0011.

Since we planned on making this comparison ahead of time, and this is our only A Priori comparison, we can use this test to reject H_0 and say that there is a difference in the mean test score between the *silence* and the *voices* conditions.

As an exercise, make a planned comparison t-test between the *rock music* and *classical music* conditions. You should get a t-value of 0.2919 and a p-value of 0.7717.

17.2.2 'Contrast' for two means

Another way of thinking about the comparison we just made between the means from the *silence* and the *voices* conditions is to consider the numerator of our t-test as a 'linear combination' of means. A linear combination is simply a sum of weighted values. For this comparison, we assign a weight of 1 for the *silence* condition and a weight of -1 for the *voices* condition. All other means get zero weights. We use a lower case 'psi' (ψ) to indicate this weighted sum of means, with weights a_i for each mean, $\overline{X}_i$. For this example:

$$\psi = (1)(77.11) + (-1)(48.37) = 28.7400$$

The hypothesis test for contrasts can be done as either a t or an F test since when df_{bet} =1, F = t^2. We'll use the F test in this chapter. The numerator of the F test is calculated with the following sums of squared error:

$$SS_{contrast} = \frac{\psi^2}{\sum (a_i^2 / n_i)}$$

For equal sample sizes, like this example, this simplifies to:

$$SS_{contrast} = \frac{\psi^2}{(\sum a_i^2)/n}$$

Which for our example is:

$$SS_{contrast} = \frac{28.74^2}{(1^2 + (-1)^2)/10} = 4129.94$$

The mean squared error is always the sum of squared error divided by the degrees of freedom. The df for A Priori contrasts is always 1, so the numerator of the F test will be:

$$MS_{contrast} = \frac{SS_{contrast}}{1} = SS_{contrast}$$

The denominator of the F test for A Priori contrasts is the same denominator as for the omnibus F, or MS_w. So our F value is:

$$F(1, df_{within}) = \frac{MS_{contrast}}{MS_{within}}$$

Which for our example is:

$$F(1, 45) = \frac{4129.94}{340.78} = 12.12$$

The p-value for this value of F is 0.0011, which is the same p-value as for the t-test above. That's because our F statistic is equal to t^2 ($12.12 = 3.48^2$).

Note that our contrast weights, 1 and -1, can vary by a scale factor. If we used, for example, $\frac{1}{2}$ and $-\frac{1}{2}$ we'd get the same F statistic and p-value. But the contrasts do have to add up to zero in order to test the null hypothesis that the population means are the same.

17.2.3 Contrast for groups of means

Contrasts also allow us to compare groups of means with other groups of means. In our example, suppose we have the prior hypothesis that music in general has a different effect on test scores than white noise. That is, we want to compare the average of the two music conditions (rock and classical) with the white noise condition.

Our weights will be 1 for the white noise condition, and -.5 for the rock and -.5 for the classical conditions, and zero for the remaining conditions. The corresponding linear combination of means is:

$$\psi = (1)(80.45) + (-0.5)(60.56) + (-0.5)(58.15) = 21.0950$$

You should convince yourself that this value, ψ, is the difference between the white noise condition and the average of the two music conditions. It should have an expected value of zero for the null hypothesis because the weights add up to zero.

The mean squared error for this contrast is:

$$SS_{contrast} = \frac{\psi^2}{(\sum a_i^2)/n} = \frac{21.095^2}{\frac{(1)^2+(-0.5)^2+(-0.5)^2}{10}} = 2966.66$$

As always for contrasts, $df_{contrast} = 1$, so $MS_{contrast} = \frac{SS_{contrast}}{df_{contrast}} = SS_{contrast}$

so our F statistic is:

$$F(1,45) = \frac{2966.66}{340.78} = 8.71$$

The p-value for this value of F is 0.005

17.2.3.1 Orthogonal contrasts and independence

We have now made two contrasts. We just compared the effects of white noise to the average of the effects of rock and classical music on test score. Before that we compared the silence and voices conditions. You should appreciate that these two contrasts are *independent* simply because they don't share any groups in common.

Contrasts can be independent even if they share groups. Formally, two contrasts are independent if the sum of the products of their weights (the 'dot product') add up to zero. When this happens, the two contrasts are called *orthogonal*. In our example:

c1 = [1, 0, 0, 0, -1]

and our new contrast is

c2 = [0, 1, -1/2, -1/2, 0]

The sum of their products is 0:

(1)(0) + (0)(1) + (0)(-.5) + (0)(-.5) + (-1)(0) = 0

Another contrast that is orthogonal to the second one is:

c3 = [0, 0, 1, -1, 0]

This is because (0)(0) + (1)(0) + (-.5)(1) + (.5)(1) + (0)(0) = 0

What does this contrast test? It compares the test scores for the rock and classical music conditions. Notice that this contrast is also orthogonal to c1, the first contrast.

It turns out that there are exactly as many mutually orthogonal contrasts as there are degrees of freedom for the numerator of the omnibus (k-1). So there should be 4 orthogonal contrasts for our example (though this is not a unique set of 4 orthogonal contrasts). This leaves one more contrast. Can you think of it?

The answer is:

c4 = [1/2 -1/3 -1/3 -1/3 1/2]

Show that c4 is orthogonal to the other three. What is it comparing? It's the mean of the silence and voices conditions compared to the mean of the other three conditions (white noise, rock and classical). We probably wouldn't have had an A Priori hypothesis about this particular contrast.

Notice that since each contrast has one degree of freedom, the sum of degrees freedom across all possible contrasts is equal to the degrees of freedom of the omnibus. Likewise, it turns out that the sums of the $SS_{contrast}$ across all orthogonal contrasts adds up to the SS_{bet}.

If two contrasts are orthogonal, then the two tests are 'independent'. If two tests are independent, then the probability of rejecting one test does not depend of the probability of rejecting the other. An example of two contrasts that are *not* independent is comparing *silence* to *white noise* for the first contrast, and *silence* to *rock music* for the second contrast. You should see that if we happen to sample an extreme mean for the *silence*, then there will a high probability that *both* of the contrasts will be statistically significant. Even though both have a Type I error rate of α, there is will be a positive correlation between the probability of rejecting the two tests.

Testing orthogonal contrasts on the same data set is just like running completely separate experiments. Since orthogonal contrasts are independent, we can easily calculate the familywise error rate:

$$FWER = 1 - (1 - \alpha)^n$$

where n is the number of orthogonal contrasts.

If a set of tests are not independent, the familywise error rate still increases with the number of tests, but in more complicated ways that will be dealt with in the post-hoc comparison section below.

17.3 Contrasts with R

R doesn't have any libraries to conduct contrasts, but it's not too difficult to do them 'by hand'. I've supplied some code to do it for you. First, though, lets' load in the data set that we've been working with and compute the ANOVA. Although it's a bit lengthy, we've covered this in the chapter on ANOVA as sums of squares:

```
# choose your alpha
alpha <- .05

# load in the data
mydata<-read.csv("http://www.courses.washington.edu/psy524a/datasets/AprioriPostHocExample.csv")

# set the background noise levels as a 'factor' in a specific order
mylevels <-  unique(mydata$backgroundnoises)
mylevels <- factor(mylevels,levels =  mylevels[c(5,3,2,1,4)])
mydata$backgroundnoises <- factor(mydata$backgroundnoises,levels = mylevels)

# run the ANOVA
anova.out <- anova(lm(testscores ~ backgroundnoises,data=mydata))

# pull out values from the anova summary
dfbet <- anova.out$Df[1]
SSbet <- anova.out$`Sum Sq`[1]
MSbet <- anova.out$`Mean Sq`[1]
dfw <- anova.out$Df[2]
SSw <- anova.out$`Sum Sq`[2]
MSw <-  anova.out$`Mean Sq`[2]
F.value <- anova.out$`F value`[1]
p.value <- anova.out$`Pr(>F)`[1]

#  generate a data frame containing statistics for each level
mydata.summary <- data.frame(
       levels = mylevels,
       mean = tapply(mydata$testscores,mydata$backgroundnoises,mean),
       n = tapply(mydata$testscores,mydata$backgroundnoises,length),
       sd = tapply(mydata$testscores,mydata$backgroundnoises,sd))
mydata.summary$sem <- mydata.summary$sd/sqrt(mydata.summary$n)

# re-order the rows
mydata.summary <- mydata.summary[c(5,3,2,1,4),]

mydata.summary
```

```
##                          levels  mean  n       sd      sem
## silence                 silence 77.11 10 25.26436 7.989291
## white noise         white noise 80.45 10 16.70617 5.282955
## rock music           rock music 60.56 10 11.63856 3.680435
## classical music classical music 58.15 10 20.61112 6.517809
## voices                   voices 48.37 10 15.04113 4.756424
```

Now that we have all of our ANOVA results stored in variables, we're ready to compute contrasts. The coefficients for the four contrasts will be stored in a 4x5 matrix (4 rows by 5 columns since there are four contrasts and 5 'background noise' levels):

```
# Contrasts
contrast <- matrix(c(
 1, 0, 0, 0, -1,
 0, 1, -0.5, -0.5, 0,
 0, 0, 1, -1, 0,
 0.5, -0.33333, -0.33333, -0.33333, 0.5),nrow = 4, byrow = TRUE)
```

We're now ready to do the math to compute ψ, $SS_{contrast}$ and the corresponding F-statistics and p-values. I'm not expecting you to be able to program this yourself - but it's yours to have and will work on any set of contrasts that you define yourself. If your curious, the `%*%` means 'element by element' multiplication (like '.*' in Matlab if that helps) which is used to calculate ψ.

```
psi <- contrast %*%mydata.summary$mean
SScontrast <- psi^2/colSums(t(contrast^2)/as.vector(mydata.summary$n))
Fcontrast <- SScontrast/MSw
pcontrast <- 1-pf(Fcontrast,1,dfw)
contrast.result = data.frame(contrast,psi,round(SScontrast,4),Fcontrast,pcontrast)
colnames(contrast.result) <- c( rownames(mydata.summary),"psi","SS","F","p")
row.names(contrast.result) <- sprintf('c%d',1:nrow(contrast))
```

The resulting table `contrast.result` looks like this:

Table 17.3:

	silence	white noise	rock music	classical music	voices
c1	1.0	0.00	0.00	0.00	-1.0
c2	0.0	1.00	-0.50	-0.50	0.0
c3	0.0	0.00	1.00	-1.00	0.0
c4	0.5	-0.33	-0.33	-0.33	0.5

17.3.1 Breaking down $df_{between}$ with $SS_{contrast}$

For an ANOVA with k groups there will be $k-1$ independent contrasts. These contrasts are not unique - there can be multiple sets of $k-1$ orthogonal contrasts. But for any set, it turns out that $SS_{between}$ is the sum the $k-1$ $SS_{contrast}$ values:

$$4129.94 + 2966.66 + 29.0405 + 159.578 = 7285.22 = SS_{between}$$

The intuition behind this is that the 4 contrasts are breaking down the total amount of variability between the means ($SS_{between}$) into separate independent components, each producing their own hypothesis test.

17.4 Controlling for familywise error rates

The most common way to control for familywise error is to simply lower the alpha value when making multiple independent comparisons. This comes at the expense of lowering the power for each individual test because, remember, decreasing alpha decreases power.

There is a variety and growing number of correction techniques, we'll cover just a few here.

17.4.1 Bonferroni correction

Bonferroni correction is the easiest, oldest, and most common way to correct for FWER. All you do is reduce alpha by dividing it by the number of comparisons. For example, if you want to make 4 comparisons and want the FWER to be below 0.05, you simply test each comparison with an alpha value of $0.05/4 = 0.0125$.

Our contrasts produced p-values of 0.0011, 0.005, 0.7717, and 0.4973. If we were to make all four A Priori comparisons, we'd need to adjust alpha to be $0.05/4 = 0.0125$.

We'd therefore reject the null hypothesis for contrasts 1 and 2 but not for contrasts 3 and 4.

17.4.2 Šidák correction

Some software packages correct for familywise error using something called the Šidák correction. The result is almost exactly the same as the Bonferroni correction, but it's worth mentioning here so you know what it means when you see a button for it in software packages like SPSS.

Remember, the familywise error rate is the probability of making one or more false positives. The Bonferroni correction is essentially assuming that the familywise error rate grows in proportion to the number of comparisons so we scale alpha down accordingly. But we know that the family wise error rate is $1 - (1 - \alpha)^m$, where m is the number of comparisons.

For example, for a Bonferroni correction with $\alpha = .05$ and 4 comparisons, we need to reduce the alpha value for each comparison to $0.05/4 = 0.0125$. The familywise error rate is now

$$1 - (1 - 0.0125)^4 = 0.049$$

It's close to 0.05 but it's just a little lower. To bring the FWER up to exactly 0.05 we need to use:

$$\alpha' = 1 - (1 - \alpha)^{1/m}$$

With 4 comparisons and $\alpha = 0.05$, the corrected alpha is:

$$\alpha' = 1 - (1 - 0.05)^{1/4} = 0.01274$$

So instead of using an alpha of 0.0125 you'd use 0.01274. The difference between the Šidák correction and the Bonferroni correction is minimal but technically the Šidák correction sets the probability of getting one or more one false alarms to exactly alpha for independent tests.

17.4.3 Holm-Bonferroni Multistage Procedure

The Holm-Bonferroni procedure is a more forgiving way to correct for familywise error, and has been used more recently, especially for large number of comparisons. The procedure is best described by example. First, we rank-order our p-values from our multiple comparisons in from lowest to highest. From our four contrasts: 0.0011, 0.005, 0.4973, and 0.7717 for contrast numbers 1, 2, 4, and 3 respectively.

We start with the lowest p-value and compare it to the alpha that we'd use for the Bonferroni correction ($\frac{0.05}{4} = 0.0125$).

If our lowest p-value is less than this corrected alpha, then we reject the hypothesis for this contrast. If we fail to reject then we stop. For our example, p = 0.0011 is less than 0.0125, so we reject the corresponding contrast (number 1) and move on.

We then compare or next lowest p-value to a new corrected alpha. This time we divide alpha by 4-1=3 to get $\frac{0.05}{3} = 0.0167$, a less conservative value. If we reject this contrast, the we move on to the next p-value and the next corrected alpha $\frac{0.05}{2} = 0.025$).

This continues until we fail to reject a comparison, and then we stop.

The idea is that if you manage to reject with the lowest p-value using the full Bonferroni correction for m tests ($\frac{\alpha}{m}$), then you can move on to the next p-value and correct only for the remaining m-1 tests ($\frac{\alpha}{m-1}$), and so on.

There's not a clean package to do this in R, but here's how you can do it on your own:

```
# rank order the p-values from lowest to highest
pvalue.order <- order(contrast.result$p)

# set up the conditions for terminating the 'while' loop
failed.yet <- FALSE
i <- 1

while (i<=m && !failed.yet){
    current.alpha = alpha/(m-i+1)
    if (contrast.result$p[pvalue.order[i]]<current.alpha) {
        cat(sprintf("contrast %d: Reject H0, F(1,%d) = %5.4f, p = %5.4f < %5.4f\n",
        pvalue.order[i],dfw,contrast.result$F[pvalue.order[i]],
        contrast.result$p[pvalue.order[i]],current.alpha))
    } else {
        cat(sprintf("contrast %d: Fail to reject H0, F(1,%d) = %5.4f, p = %5.4f > %5.4f\n",
        pvalue.order[i],dfw,contrast.result$F[pvalue.order[i]],
        contrast.result$p[pvalue.order[i]],current.alpha))
        failed.yet <- TRUE
        cat("done, fail to reject the rest.\n")
    }
    i <- i+1
}
```

```
## contrast 1: Reject H0, F(1,45) = 12.1191, p = 0.0011 < 0.0125
## contrast 2: Reject H0, F(1,45) = 8.7055, p = 0.0050 < 0.0167
## contrast 4: Fail to reject H0, F(1,45) = 0.4681, p = 0.4974 > 0.0250
## done, fail to reject the rest.
```

This isn't a programming class, so I don't expect you to fully understand the code. But it will work for your own set of contrasts. If you're interested, however, see if you can follow how it works. It uses a `while` loop, which continues while the condition `i<=m && !failed.yet` is true. `m` is the number of contrasts, and `i` is an index that starts at 1 and increments after each rejected test. Each time through,the contrasts, ranked by their p-values, are compared to `alpha/(m-i+1)` which starts out at alpha/m for the first contrast, alpha/(m-1) for the second, etc.

The variable `failed.yet` is a logical (TRUE/FALSE) that starts out as `FALSE` and turns to true after the first contrast fails to reject. So `!failed.yet` starts out as true, allowing the while loop to continue until the first fail to reject, or until we run out of contrasts.

Multistage procedures like the Holm-Bonferroni are less conservative and therefore more powerful than the standard Bonferroni correction. They are less widely used probably because they're more complicated. But as you've seen, computers can easily do these things for us.

There are other variants of this sort of multistage procedure, including sorting from highest to lowest p-values, and using a Sidac correction for each test instead of a Bonferroni correction. They all produce similar results and the field has not settled on one procedure in particular.

17.5 Post Hoc Comparisons

Now let's get more exploratory and make some comparisons that we didn't plan on making in the first place. These are called *post hoc* comparisons. For A Priori comparisons, we only needed to adjust for the FWER associated with the number of planned comparisons. For post hoc comparisons, we need to adjust to not just the comparisons we feel like making, but for all possible comparisons of that type that we *could have made* (e.g all possible pairwise comparisons or all possible contrasts).

17.5.1 The Tukey Test

The Tukey Test is a way of correcting for FWER when testing pairs of means. For our example there are 3 pairs of means to test. But you can't use a Bonferroni correction and divide alphy by 3 because not all comparisons are

independent.

The test is based on the distribution that is expected when you specifically compare the largest and smallest mean. If the null hypothesis is true, the probability of a significant t-test for these most extreme means will be quite a bit greater than α, which is the probability of rejecting any random pair of means.

The way for correcting for this inflated false positive rate when comparing the most extreme means is to use a statistic called the *Studentized Range Statistic* and goes by the letter q.

The statistic q is calculated as follows:

$$q = \frac{\overline{X}_l - \overline{X}_s}{\sqrt{\frac{MS_{within}}{n}}}$$

Where $\overline{X}_l$ is the largest mean, $\overline{X}_s$ is the smallest mean, and n is the sample size of each group (assuming that all sample sizes are equal).

The q-statistic for our most extreme pairs of means is:

$$q = \frac{80.45 - 48.37}{\sqrt{\frac{340.7788}{10}}} = 5.4954$$

The q statistic has its own distribution. Like the t-statistic, it is broader than the normal distribution for small degrees of freedom. Also, q increases in width with increasing number of groups. That's because as the number of possible comparisons increases, the difference between the extreme means increases.

R has the functions `ptukey` and `qtukey` to test the statistical significance of this difference of extreme means. `ptukey` needs our value of q, the number of groups, and df_{within}. For our pair of extreme means:

```
1-ptukey(5.4954,5,45)
```

```
## [1] 0.002925262
```

Importantly, even though we selected these two means after running the experiment, this p-value accounts for this bias.

In fact, almost magically, we can use this procedure to compare any or all pairs of means, and we won't have to do any further correction for multiple comparisons.

17.5.2 Tukey's 'HSD'

Back in the chapter on effect sizes and confidence intervals we discussed how an alternative way to make a decision about H_0 is to see if the null hypothesis is covered by the confidence interval. The same thing is often done with the Tukey Test.

Instead of finding a p-value from our q-statistic with `ptukey`, we'll go the other way and find the value q_{crit} that covers the middle 95of the q distribution using `qtukey`. Since it's a two-tailed test, we need to find q for the top 97.5%

```
q_crit <- qtukey(1-0.05,5,45)
```

```
## [1] 4.018417
```

We can go through the same logic that we did when we derived the equation for the confidence interval. If we assume that the observed difference between a pair of means, $\overline{X}_i - \overline{X}_j$ is the true difference between the means, then we expect that in future experiments, we can expect to find a difference 95% of the time within:

$$\overline{X}_i - \overline{X}_j \pm q_{crit}\sqrt{\frac{MS_w}{n}}$$

For our example:

$$q_{crit}\sqrt{\frac{MS_w}{n}} = 4.0184\sqrt{\frac{340.7788}{10}} = 23.4579$$

This value, 23.4579 is called Tukeys' *honestly significant difference*, or *HSD*. Any pair of means that differs by more than this amount can be considered statistically significant at the level of α.

The 'H' in Tukey's HSD is for 'honestly' presumably because it takes into account all possible pairwise comparisons of means, so we're being honest and accounting for familywise error.

We can use Tukey's HSD to calculate a confidence interval by adding and subtracting it from the observed difference between means. Our most extreme means had a difference of

$$80.45 - 48.37 = 32.08$$

The 95% confidence interval for the difference between these means is:

$$(32.08 - 23.4579, 32.08 + 23.4579)$$

which is

$$(8.6221, 55.5379)$$

The lowest end of this interval is greater than zero, which is consistent with the fact that the p-value for the difference between means (0.0029) is less than $\alpha = 0.05$. If the difference between the means is significantly significant (with $\alpha = 0.05$), then the 95% confidence interval will never contain zero.

All this is easy to do in R using the `tukeyHSD` function. There's just a little hack. It's an old function which requires the output of an outdated ANOVA function called `aov`. But you can, instead, send in to `tukeyHSD` the output of `lm` after changing it's class to `aov`. Here's how you run the Tukey test on our example. We'll run the `lm` function on our data, but instead of passing it into `anova` like we did before, we'll change it's class to 'aov' and pass it into `TukeyHSD`:

```
lm.out <- lm(testscores ~ backgroundnoises,data =mydata)
class(lm.out) <- c("aov","lm")  # total hack
TukeyHSD(lm.out)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: lm(formula = testscores ~ backgroundnoises, data = mydata)
##
## $backgroundnoises
##                               diff        lwr       upr     p adj
## rock music-classical music    2.41 -21.048015 25.868015 0.9983541
## white noise-classical music  22.30  -1.158015 45.758015 0.0695439
## voices-classical music       -9.78 -33.238015 13.678015 0.7599945
## silence-classical music      18.96  -4.498015 42.418015 0.1649044
## white noise-rock music       19.89  -3.568015 43.348015 0.1315002
## voices-rock music           -12.19 -35.648015 11.268015 0.5826433
## silence-rock music           16.55  -6.908015 40.008015 0.2803943
## voices-white noise          -32.08 -55.538015 -8.621985 0.0029254
## silence-white noise          -3.34 -26.798015 20.118015 0.9941624
## silence-voices               28.74   5.281985 52.198015 0.0094156
```

The p-values in the column 'p adj' reflect the correction for FWER, so they don't need to be adjusted to compare to α. Any test for a pair of means with an adjusted p-value less than α can be rejected.

Here you'll see adjusted p-values for all possible $\frac{5\times4}{2} = 10$ comparisons. Can you find the row for the biggest difference between means?

You'll also see the columns 'lwr' and 'upr'. These are the ranges for the 95% confidence interval on the differences between the means. For all cases, if the p-value 'p adj' is less than .05, then the 95% confidence interval will not include zero. Is the confidence interval the same as our calculation? It's off by a $\pm$ sign flip because `tukeyHSD` decided to run the test with the opposite order of means (smallest - largest). Otherwise it's the same, and it doesn't matter anyway because it's a two-tailed test. You should notice that the range between 'lwr' and 'upr' is always the same value of Tukey's HSD = 23.4579.

17.5.3 Tukey-Kramer Test - for unequal sample sizes

The sample sizes must be equal when using the Tukey test. If the sample sizes varies across groups (called an 'unbalanced design') there is a modification of the Tukey test called the 'Tukey-Kramer method' which uses a different HSD depending upon which means are being compared.

Specifically, to compare means from group i to group j, with sample sizes n_i and n_j and variances s_i^2 and s_j^2, the q-statistic is:

$$q = \frac{\overline{X}_i - \overline{X}_j}{\sqrt{\frac{\frac{s_i^2}{n_i} + \frac{s_j^2}{n_j}}{2}}}$$

And replace df_{within} with:

$$df_{ij} = \frac{\left(\frac{s_i^2}{n_i} + \frac{s_j^2}{n_j}\right)^2}{\frac{\left(\frac{s_i^2}{n_i}\right)^2}{n_i - 1} + \frac{\left(\frac{s_j^2}{n_j}\right)^2}{n_j - 1}}$$

It's kind of ugly, but here's some R code that makes all possible paired comparisons based on our summary table `mydata.summary` that we computed above:

```
k <- nrow(mydata.summary)

# create a matrix of NA's to hold our p-values
pTable <- data.frame(matrix(nrow = k-1,ncol = k-1))

row.names(pTable) <- row.names(mydata.summary)[1:k-1]
colnames(pTable) <- row.names(mydata.summary)[2:k]

for (i in 1:(k-1)) {
  for (j in (i+1):k) {
    # compare means from group i to group j

    ni <- mydata.summary$n[i]
    nj <- mydata.summary$n[j]
    sni <- mydata.summary$sd[i]^2/ni
    snj <- mydata.summary$sd[j]^2/nj

    q <- (abs(mydata.summary$mean[i]-mydata.summary$mean[j]))/sqrt((sni+snj)/2)
    df <- (sni+snj)^2/(sni^2/(ni-1)+snj^2/(nj-1))
    pTable[i,j-1] <- round(1-ptukey(q,k,df),4)
  }
}
pTable[is.na(pTable)] <- ""
```

Which produces a nice table. I've rendered the table using the `kable` package, and colored the significant comparisons red.

Table 17.4:

	white noise	rock music	classical music	voices
silence	0.9965	0.3741	0.3844	0.0505
white noise		0.0475	0.1027	0.0022
rock music			0.9974	0.2955
classical music				0.7447

Even though we ran the Tukey-Kramer test on the same data set, which has equal sample sizes, the p-values aren't the same as for the regular Tukey Test. Usually, but not always, the Tukey-Kramer test will be less powerful (have larger p-values) than the Tukey Test because it is only using the variance for two means at a time which has a lower *df* than for the regular Tukey Test.

17.6 Dunnett's Test for comparing one mean to all others

This is a post-hoc test designed specifically for comparing all means to a single control condition. For our example, it makes sense to compare all of our conditions to the *silence* condition. Dunnett's test is a special case because (1) these comparisons are not independent and (2) there are fewer comparisons to correct for than for the Tukey Test since we're testing only a subset of all possible pairs of means.

Dunnett's test relies on finding critical values from a distribution that is related to the t-distribution called the *Dunnett's t Statistic*. It's easy to run Dunnett's test in R with the function `DunnetTest` which requires the 'DescTools' library.

Let's jump straight to the test, since it's not likely that you'll ever need to calculate p-values by hand with this test.

Let's let the the first condition, the "silence" condition, be the control condition, so we'll compare the other four means to this one ($\overline{X}_1 = 77.11$)

Here's how it works:

```
dunnett.out <- DunnettTest(testscores ~ backgroundnoises,
          data = mydata,control = "silence")
dunnett.out
```

```
##
##   Dunnett's test for comparing several treatments with a control :
##     95% family-wise confidence level
##
## $silence
##                             diff    lwr.ci    upr.ci   pval
## classical music-silence -18.96 -39.86108  1.941085 0.0852 .
## rock music-silence      -16.55 -37.45108  4.351085 0.1563
## white noise-silence       3.34 -17.56108 24.241085 0.9831
## voices-silence          -28.74 -49.64108 -7.838915 0.0040 **
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The table itself sits in the output under the field having the name of the control condition as a 'matrix'. It's useful to turn it into a data frame. For our example, use this:

```
myframe <- data.frame(dunnett.out$silence)
```

Here's the table formatted so that the significant values are in red:

Table 17.5:

	diff	lwr	upr	p adj
classical music-silence	-18.96	-39.8611	1.9411	0.0852
rock music-silence	-16.55	-37.4511	4.3511	0.1563
white noise-silence	3.34	-17.5611	24.2411	0.9831
voices-silence	-28.74	-49.6411	-7.8389	0.004

These p-values are all corrected for familywise error, so no further correction is needed.

In general, this test is more powerful (gives lower p-values) than the Tukey Test, since fewer comparisons are being made.

17.7 The Sheffe' Test: correcting for all possible contrasts

The Sheffe' test allows for post-hoc comparisons across all possible contrasts (including non-orthogonal contrasts), not just means. Since there are many more possible contrasts than pairs of means, the Sheffe' test has to control for more possible comparisons and is therefore more conservative and less powerful.

Remember, the F-test for a contrast is conducted by first computing the weighted sum of means:

$$\psi = \sum a_i \overline{X}_i$$

The $SS_{contrast}$ (and $MS_{contrast}$):

$$SS_{contrast} = MS_{contrast} = \frac{\psi^2}{\sum a_i^2/n_i}$$

and then the F statistic:

$$F(1, df_w) = \frac{MS_{contrast}}{MS_w}$$

The Sheffe' test adjusts for multiple comparisons by multiplying the critical value of F for the original 'omnibus' test by k-1 (the number of groups minus one). In our example, the critical value of F for the omnibus test can be calculated using R's `qf` function:

```
## [1] "F_crit <- qf(1-0.05,4,45)"
```

```
## [1] 2.578739
```

Our adjusted critical value for F is (4)(2.5787) = 10.315. Any contrast with an F-statistic greater than this can be considered statistically significant.

Instead of a critical value of F, we can convert F = 10.315 into a modified value of α. Using R's `pf` function we get:

```
## [1] "alpha_Sheffe <-1-pf(F_crit_Sheffe,1,45)"
```

```
## [1] 0.002438332
```

Any contrast with a p-value less than 0.0024 is considered statistically significant. This is about 1 in 400 contrasts.

Remember, our four contrasts had weights:

c1 = [1 0 0 0 -1]

c2 = [0 1 -1/2 -1/2 0]

c3 = [0 0 1 -1 0]

c4 = [1/2 -1/3 -1/3 -1/3 1/2]

The F values for these four contrasts are 12.12, 8.71, 0.09, and 0.47, and the four corresponding p-values are 0.0011, 0.005, 0.7717, and 0.4973.

Comparing these F values to our adjusted critical value of F, or comparing the p-values to 0.0024, lets us reject the null hypothesis for contrasts 1 but not for contrasts 2, 3 and 4. How does this compare to the A Priori Bonferroni test we used to compare these contrasts?

With the Sheffe' test, the door is wide open to test any post-hoc comparison we want. Looking at the bar graph, it looks like there is a difference between the average of the *silence, and white noise* conditions, and the average of the *rock music, classical music, and voices* conditions. This would be a contrast with weights:

[1/2, 1/2, -1/3, -1/3, -1/3]

If you work this out you get an F value of 18.7686. This exceeds the Sheffe'-corrected F-value of 10.315 so we can say that there is a significant difference between these groups of means.

It feels like cheating - that we're making stuff up. And yes, post-hoc tests are things you made up after you look at the data. I think if feels wrong because we're so concerned about replicability and preregistration. But with the proper correction for multiple comparisons, this is totally fine. In fact, I'll bet that most of the major discoveries on science have been made after noticing something interesting in the data that wasn't anticipated. I would even argue that sticking strictly to your preregistered hypotheses could slow down the progress of science.

17.8 Summary

These notes cover just a few of the many A Priori and post hoc methods for controlling for multiple comparisons. Statistics is a relatively new and developing field of mathematics, so the standards for these methods are in flux. Indeed, SPSS alone provides a confusing array of post-hoc methods and allows you to run many or all of them at once. It is clearly wrong to run a bunch of post-hoc comparisons and then pick the comparison that suits you. This sort of post hoc post hoc analysis can lead to a sort of meta-familywise error rate.

It's also not OK to treat a post hoc comparison with an A Priori method. You can't go back and say "Oh yeah, I meant to make that comparison" if it hadn't crossed your mind.

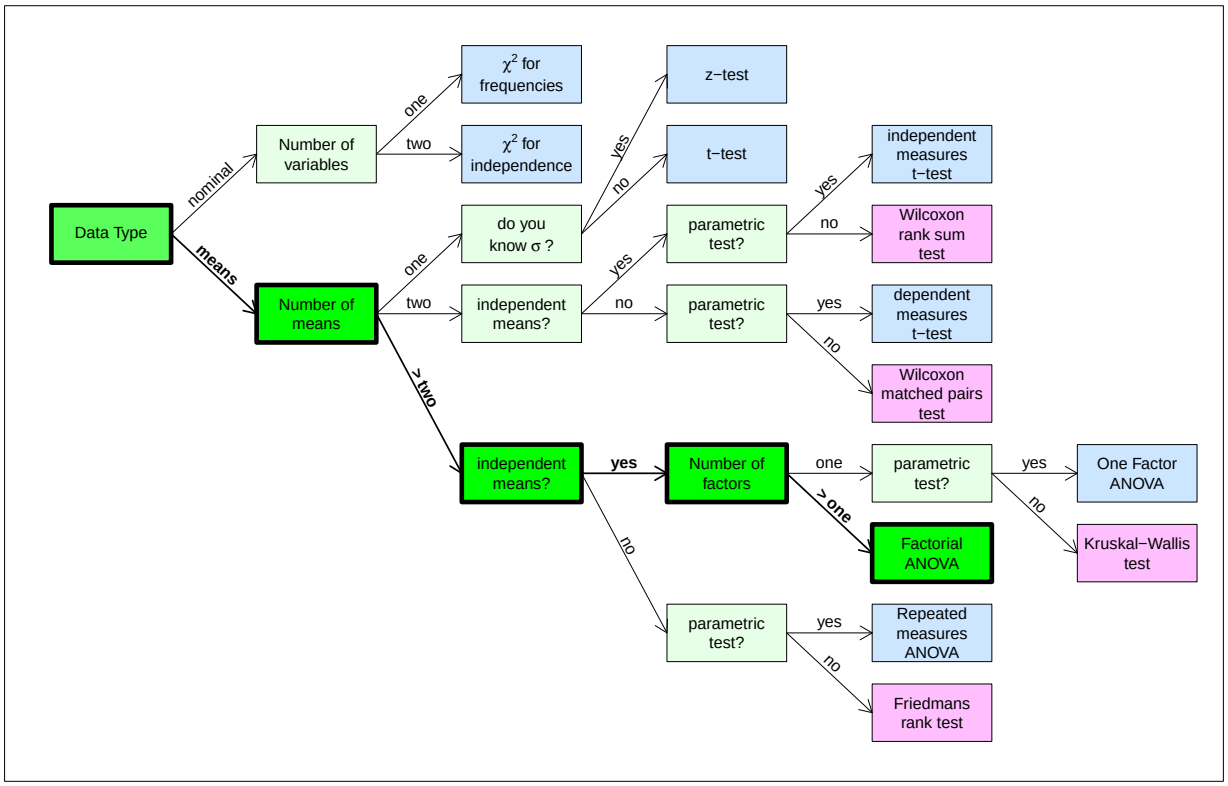
In the end, all of these methods differ by relatively small amounts. If the significance of your comparisons depend on which specific A Priori or post hoc test you choose, then you're probably taking the .05 or .01 criterion too seriously anyway.

Chapter 18

Two Factor ANOVA

The Two-factor ANOVA is a hypothesis test on means with a 'crossed design' which has two independent variables. Observations are made for all combinations of levels for each of the two variables.

You can find this test in the flow chart here:



We'll build up to the two-factor ANOVA by starting with what we already know - a 1-factor ANOVA experiment.

18.1 1-factor ANOVA Beer and Caffeine

Suppose you want to study the effects of beer and caffeine on response times for a simple reaction time task. One way to do this is to divide subjects in to four groups: a control group, a group with caffeine (and no beer), a group with beer (and without caffeine), and a lucky group with both beer and caffeine.

We'll load in this existing data from the course website:

```
data.1 <- read.csv('http://courses.washington.edu/psy524a/datasets/BeerCaffeineANOVA1.csv')
```

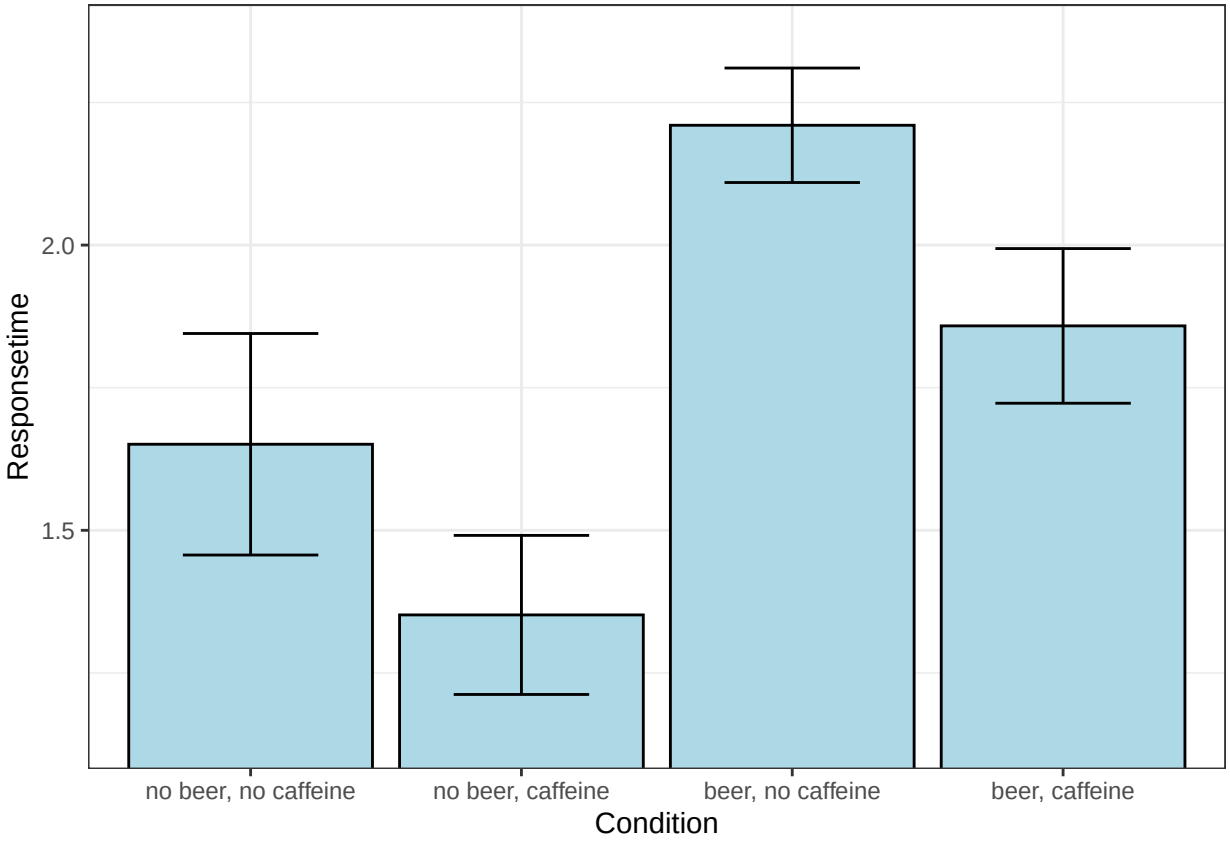
Here are the summary statistics from this data set:

The four conditions, or 'levels' are "no beer, no caffeine", "no beer, caffeine", "beer, no caffeine", and "beer, caffeine"

Table 18.1:

	mean	n	sd	sem
no beer, no caffeine	1.650833	12	0.6725455	0.1941472
no beer, caffeine	1.351667	12	0.4829235	0.1394080
beer, no caffeine	2.210000	12	0.3476153	0.1003479
beer, caffeine	1.858333	12	0.4696194	0.1355675

Here's a plot of the means with error bars as the standard error of the mean:



It looks like the means do differ from one another. Here's the 'omnibus' ANOVA result:

Table 18.2:

	df	SS	MS	F	p
Between	3	4.687	1.5623	6.0856	p = 0.0015
Within	44	11.296	0.2567		
Total	47	15.983			

Yup, some combination of beer and caffeine have a significant effect on response times.

In this design we actually manipulated two factors - beer, and caffeine. It'd be nice to be able to look at these two 'factors' separately.

18.2 Effect of Beer

Consider the effect of Beer on reaction times. We could just run a t-test on the 'no beer, no caffeine' condition vs. the 'beer, no caffeine' condition. But we also can compare the 'no beer, caffeine' condition to the 'beer, caffeine' condition. Or, perhaps even better, we can combine these two comparisons. This combined analysis can be done with a contrast with the following weights:

Table 18.3:

	no beer, no caffeine	no beer, caffeine	beer, no caffeine	beer, caffeine
Effect of Beer	1	1	-1	-1

This measures the effect of beer averaging across the two caffeine conditions.

The calculations for this contrast yields:

$$\psi = (1)(1.65) + (1)(1.35) + (-1)(2.21) + (-1)(1.86) = -1.0658$$

$$MS_{contrast} = \frac{(-1.0658)^2}{\frac{(1)^2}{12} + \frac{(1)^2}{12} + \frac{(-1)^2}{12} + \frac{(-1)^2}{12}} = 3.4080$$

$$F(1, 44) = \frac{3.4080}{0.2567} = 13.2748$$

$$p = 0.0007$$

It looks like there's a significant effect of beer on response times. Since we're subtracting the beer from the without beer conditions, our negative value of ψ indicates that the responses times for beer are greater than for without Beer. Beer increases response times.

18.3 Effect of Caffeine

To study the effect of caffeine, averaging across the two beer conditions, we use this contrast, which is independent of the first one:

Table 18.4:

	no beer, no caffeine	no beer, caffeine	beer, no caffeine	beer, caffeine
Effect of Caffeine	1	-1	1	-1

The calculations for this contrast yields:

$$\psi = (1)(1.65) + (-1)(1.35) + (1)(2.21) + (-1)(1.86) = 0.6508$$

$$MS_{contrast} = \frac{(0.6508)^2}{\frac{(1)^2}{12} + \frac{(-1)^2}{12} + \frac{(1)^2}{12} + \frac{(-1)^2}{12}} = 1.2708$$

$$F(1, 44) = \frac{1.2708}{0.2567} = 4.9498$$

$$p = 0.0313$$

Caffeine has a significant effect on response times - this time ψ is positive, so response times for without caffeine are greater than for with caffeine. Caffeine reduces response times.

18.4 The Third Contrast: Interaction

For four levels or groups, there should be three independent contrast. Here's the third contrast:

Table 18.5:

	no beer, no caffeine	no beer, caffeine	beer, no caffeine	beer, caffeine
Beer X Caffeine	1	-1	-1	1

What does that third contrast measure? Symbolically, the contrast combines the conditions as:

[without beer without caffeine] - [without beer with caffeine] - [with beer without caffeine] + [with beer with caffeine]

Rearanging the terms as a difference of differences:

([with beer without caffeine] - [without beer without caffeine]) - ([with beer with caffeine]-[without beer with caffeine])

The first difference is the effect of beer without caffeine. The second difference is the effect of beer with caffeine. The difference of the differences is a measure of how the effect of beer changes by adding caffeine. In statistical terms, we call this the *interaction* between the effects of beer and caffeine on response times. Interactions are labeled with an 'X', so this contrast is labeled as 'Beer X Caffeine'.

You might have noticed the parallel between this and the χ^2 test of independence. This is the same concept, but for means rather than frequencies.

The results of the F-tests for this third contrast is:

$$\psi = (1)(1.65) + (-1)(1.35) + (-1)(2.21) + (1)(1.86) = -0.0525$$

$$MS_{contrast} = \frac{(-0.0525)^2}{\frac{(1)^2}{12} + \frac{(-1)^2}{12} + \frac{(-1)^2}{12} + \frac{(1)^2}{12}} = 0.0083$$

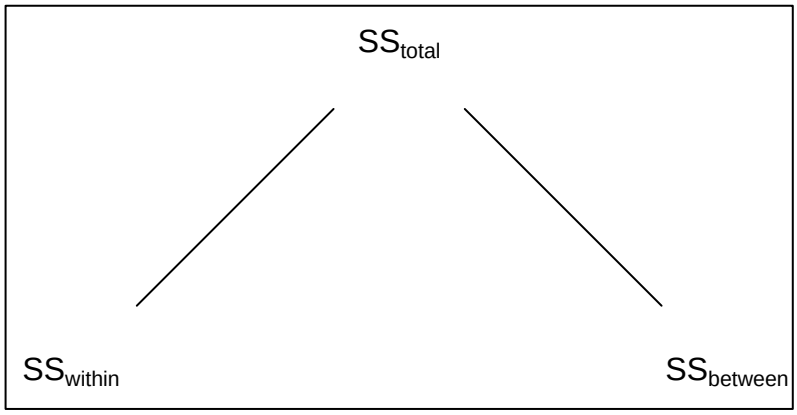
$$F(1, 44) = \frac{0.0083}{0.2567} = 0.0322$$

$$p = 0.8584$$

We fail to reject H_0, so there is no significant interaction between the effects of beer and caffeine on response times. This means that beer effectively increases response times the same amount, regardless of caffeine. Conversely, caffeine reduces response times effectively the same amount with or without beer. Notice the use of the word 'effectively' here. We should be careful about saying that 'beer increases response times the same amount, regardless of caffeine' because this isn't true. There is a slight numerical difference, but it is not statistically significant.

18.5 Partitioning $SS_{between}$

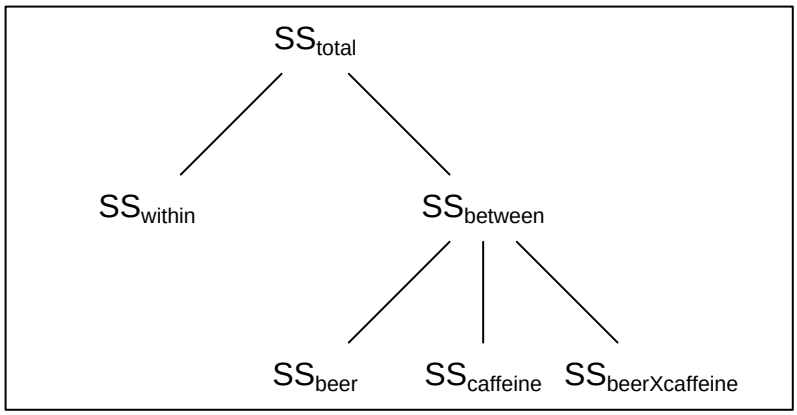
Recall that for a 1-factor ANOVA, SS_{total} is broken down in to two parts SS_{within} and $SS_{between}$:



In the chapter section 17.3.1 on APriori and post-hoc tests we discussed how the sums of squared for independent contrasts is a way of breaking down the total variability between the means, $SS_{between}$. The same is true here for our three orthogonal contrasts. Summing up the three $SS_{contrast}$ values gives us:

$$3.408 + 1.2708 + 0.0083 = 4.6871 = SS_{between}$$

So the three contrasts have partitioned the total variability between the means into three separate tests - each telling us something different about what is driving the significance of the 'omnibus' F-test. If we call the sums-of-squares for each of the three contrasts SS_{beer}, $SS_{caffeine}$, and $SS_{beerXcaffeine}$ (where the 'X' means 'interaction), we can expand the above diagram to this:



This experiment has what is called a 'factorial design', where there are conditions for each combination of levels for the two factors of beer and caffeine. This example is a 'balanced design', which means that the sample sizes are the same for all conditions.

A standard way to analyze a factorial design is to break the overall variability between the means into separate hypothesis tests - a *main effect* for each factor, and their interactions. In this section we'll show how treating the same data that we just discussed as a 2-factor ANOVA gives us the exact same result as treating the same results as a 1-factor ANOVA with three contrasts.

18.6 2-Factor ANOVA

I've saved the same data set but in a way that's ready to be analyzed as a factorial design experiment. We'll load it in here:

```
data.2 <- read.csv('http://courses.washington.edu/psy524a/datasets/BeerCaffeineANOVA2.csv')
head(data.2)
```

```
##   Responsetime    caffeine    beer
## 1         2.24 no caffeine no beer
## 2         1.62 no caffeine no beer
## 3         1.48 no caffeine no beer
## 4         1.70 no caffeine no beer
## 5         1.06 no caffeine no beer
## 6         1.39 no caffeine no beer
```

The data format has the same 'ResponseTime' column, but now it has two columns instead of one that define which condition for each measurement. The 'caffeine' column has two levels: 'caffeine' and 'no caffeine'. Similarly the 'beer' column has two levels 'beer' and 'no beer'. This way of storing the data is called 'long format', where which each row corresponds to a single observation.

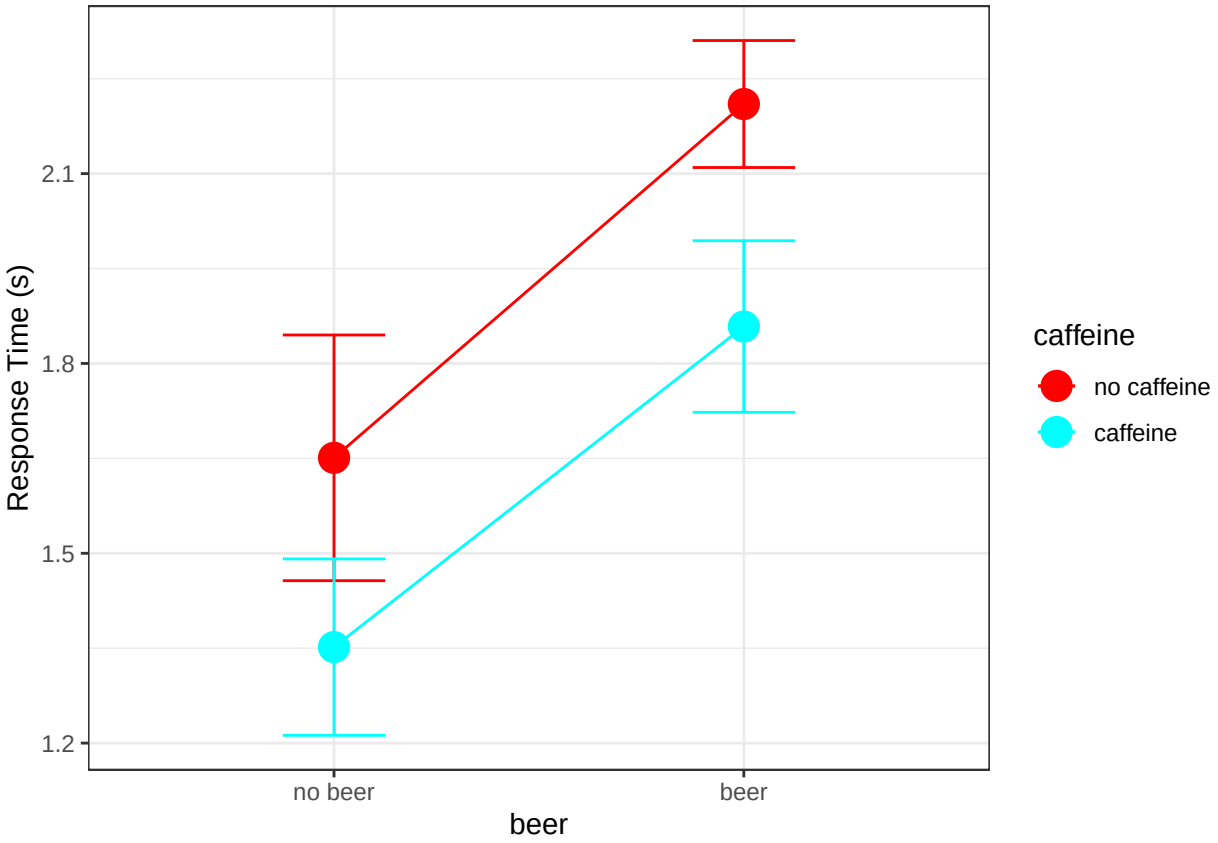
This experiment is called a *2x2 factorial design* because each of the two factors has two levels. We can summarize the results in the form of matrices with rows and columns corresponding to the two factors. We'll set the 'row factor' as 'caffeine' and the 'column factor' as 'beer'. That is, 'beer' varies across the rows and 'caffeine' varies across the column. Here's the 2x2 table for the means:

Table 18.6: Means

	no beer	beer
no caffeine	1.6508	2.2100
caffeine	1.3517	1.8583

Instead of bar graphs, it's common to plot results of factorial designs as data points with lines connecting them. By default, I plot the column factor along the x-axis and define the row factor in the legend. There are various ways of doing this in R. Here's an example for our data. It requires both 'ggplot2' and the 'dplyr' libraries. Both are part of the 'tidyverse' package.

```
# Do this to avoid a stupid useless error message
options(dplyr.summarise.inform = FALSE)

# order the levels for the two factors (alphabetical by default)
data.2$caffeine <- factor(data.2$caffeine,levels = c('no caffeine','caffeine'))
data.2$beer <- factor(data.2$beer,levels = c('no beer','beer'))

# Make a table (tibble) with generic names
summary.table <- data.2 %>%
  dplyr::group_by(caffeine,beer) %>%
  dplyr::summarise(
    m = mean(Responsetime),
    sem = sd(Responsetime)/sqrt(length(Responsetime))
  )

  # plot with error bars, replacing generic names with specific names
  ggplot(summary.table, aes(beer, m)) +
    geom_errorbar(
      aes(ymin = m-sem, ymax = m+sem, color = caffeine),
      position = position_dodge(0), width = 0.5)+
    geom_line(aes(group = caffeine,color = caffeine)) +
    geom_point(aes(group = caffeine,color = caffeine),size = 5) +
    scale_color_manual(values = rainbow(2)) +
    xlab('beer') +
    ylab('Response Time (s)') +
    theme_bw()
```



18.7 Within-Cell Variance (MS_{wc})

The three F-tests for a 2-factor ANOVA will use the same within-cell mean-squared error as the denominator. This is calculated the same way as for the 1-way ANOVA. We first add up the sums of squares for each condition.

The sums of squares within each group is called 'SS_{wc}' where 'wc' means 'within cell' since we're now talking about cells in a matrix. Here's the table for SS_{wc}:

Table 18.7: SS_{wc}

	no beer	beer
no caffeine	4.9755	1.3292
caffeine	2.5654	2.4260

SS_{wc} is the sum of these individual within-cell sums of squares:

$$SS_{wc} = 4.9755 + 2.5654 + 1.3292 + 2.426 = 11.2961$$

Each cell contributes n-1 degrees of freedom to SS_{wc}, so the degrees of freedom for all cells is N-k, where k is the total number of cells and N is the total sample size (n $\times$ k):

$$df_{wc} = 48 - 4 = 44$$

Mean-squared error is, as always, $\frac{SS}{df}$:

$$MS_{wc} = \frac{SS_{wc}}{df_{wc}} = \frac{11.296}{44} = 0.2567$$

This is the same value and df as MS_w from above when we treated the same data as a 1-factor ANOVA design.

The three contrasts that we used for the 1-factor ANOVA example correspond to what we call 'main effects' for the factors and the 'interaction' between the factors. To calculate the main effects by hand we need to calculate the means across the rows and columns of our factors. Here's a table with the row and sum means in the 'margins':

Table 18.8: Row and Column Means

	no beer	beer	means
no caffeine	1.6508	2.2100	1.9304
caffeine	1.3517	1.8583	1.6050
means	1.5012	2.0342	1.7677

The bottom-right number is the mean of the means, which is the grand mean ($\overline{\overline{X}} = 1.7677$)

18.7.1 Main Effect for Columns (Beer)

Calculating main effects is lot like calculating $SS_{between}$ for the 1-factor ANOVA. For the main effect for columns, we calculate the sums of squared deviations of the column means from the grand mean, and scale it by the number of samples that contributed to each column mean. For our example, the sums of square deviations is:

$$(1.5012 - 1.7677)^2 + (2.0342 - 1.7677)^2 = 0.071 + 0.071 = 0.142$$

There are $2 \times 12 = 24$ samples for each column mean, so the sums of squared for the columns, called SS_C is

$$SS_C = (24)(0.142) = 3.408$$

Since 2 means contributing to SS_R, so the degrees of freedom is df_R = 2 -1 = 1

MS_C is therefore

$$\frac{SS_C}{df_C} = \frac{3.4080}{1} = 3.4080$$

The F-statistic for this main effect is MS_C divided by our common denominator, MS_{wc}

$$F = \frac{MS_C}{MS_{wc}} = \frac{3.4080}{0.2567} = 13.2748$$

We can calculate the p-value for this main effect using `pf`:

```
1-pf(13.2748,1,44)
```

```
## [1] 0.0007060154
```

Notice that the F and p-values are the same as for the first contrast in the 1-way ANOVA above. If you work out the algebra you'll find that the math is the same. The main effect in a multi-factorial ANOVA is exactly the same as the appropriate contrast in a 1-factor ANOVA.

18.7.2 Main Effect for Rows (Caffeine)

The calculations for finding the main effect of rows (Caffeine) on response times is completely analogous to finding the main effect for columns. We use our row means in the table above, which are the averages across the two beer conditions.

The sums of squared deviations for the means for rows is:

$$(1.9304 - 1.7677)^2 + (1.605 - 1.7677)^2 = 0.0265 + 0.0265 = 0.053$$

There are $2 \times 12 = 24$ samples for each row mean, so the sums of squared for the row, called SS_R is

$$SS_R = (24)(0.0529) = 1.2708$$

There are 2 means contributing to SS_R, so the degrees of freedom is df_R = 2 -1 = 1

MS_R is therefore

$$\frac{SS_R}{df_R} = \frac{1.2708}{1} = 1.2708$$

The F-statistic for this main effect is MS_R divided by our common denominator, MS_{wc}

$$F = \frac{MS_R}{MS_{wc}} = \frac{1.2708}{0.2567} = 4.9498$$

The p-value for the main effect of Beer is:

```
1-pf(4.9498,1,44)
```

```
## [1] 0.03126914
```

18.7.3 Interaction Between Beer and Caffeine

The third contrast in the 1-factor ANOVA measured the differential effect of caffeine on response times across the two beer conditions (or vice versa). Recall that for a 1-factor ANOVA the sums of squares associated with three orthogonal conditions adds up to $SS_{between}$ for four groups. Also, recall that $SS_{between} + SS_{within} = SS_{total}$.

The easiest way to calculate the sums of square value for interaction is to appreciate that

$$SS_{total} = SS_{caffeine} + SS_{beer} + SS_{caffeineXbeer} + SS_{wc}$$

The total sums of squares is $SS_{total} = 15.983$.

Therefore,

$$SS_{caffeineXbeer} = SS_{total} - (SS_{wc} + SS_{caffeine} + SS_{beer})$$

so

$$SS_{RXC} = SS_{total} - SS_R - SS_C - SS_{wc} = 15.9830 - 1.2708 - 3.4080 - 11.2960 = 0.0083$$

The degrees of freedom for this interaction term is (n_{rows}-1)*(n_{cols}-1):

$$df_{RXC} = (n_{rows} - 1)(n_{cols} - 1) = (2 - 1)(2 - 1) = 1$$

So the mean-squared error for the interaction is

$$\frac{SS_{RXC}}{df_{RXC}} = \frac{0.0083}{1} = 0.0083$$

Using SS_{wc} for the denominator again, the F-statistic is:

$$

The p-value for the interaction is:

```
1-pf(0.0322,1,44)
```

```
## [1] 0.8584132
```

There is not a significant interaction between caffeine and beer on response times. Compare these numbers to the results of the third contrast in the 1-factor ANOVA above.

We typically summarize our calculations and results in a table like this:

Table 18.9:

	df	SS	MS	F	p
caffeine	1	1.2708	1.2708	4.9498	p = 0.0313
beer	1	3.4080	3.408	13.2748	p = 0.0007
interaction	1	0.0083	0.0083	0.0322	p = 0.8584
wc	44	11.2960	0.2567		
Total	47	15.9830			

Using APA format we state, for our three tests:

There is a main effect of caffeine. $F(1,44) = 4.9498$, $p = 0.0313$.

There is a main effect of beer. $F(1,44) = 13.2748$, $p = 0.0007$.

There is not a significant interaction between caffeine and beer. $F(1,44) = 0.0322$, $p = 0.8584$.

You might have noticed that didn't use any correction for familywise error for these three tests. There is a general consensus that the main effects and the interaction do not require familywise error correction. But if we treat the same data as a 1-factor design with three planned contrasts, we should apply error correction (like Bonferroni) even though the math and p-values are the same. If you find discrepancies like this baffling you are not alone.

18.8 The two-factor ANOVA in R

Conducting a two-factor ANOVA in R is a lot like for 1-factor ANOVA. We'll use the `lm` function and pass it through the `anova` function to get our table and statistics. The difference is the definition of the formula. Here we'll use: `Responsetime ~ caffieene*beer`. The use of '*' is the way to ask R to conduct not only the main effect of *caffeine* and *beer* but also their interaction:

```
anova2.out <- anova(lm(Responsetime ~ caffeine*beer,data = data.2))
anova2.out
```

```
## Analysis of Variance Table
##
## Response: Responsetime
##               Df  Sum Sq Mean Sq F value   Pr(>F)
## caffeine       1  1.2708  1.2708  4.9498 0.031269 *
## beer           1  3.4080  3.4080 13.2748 0.000706 ***
```

```
## caffeine:beer  1  0.0083  0.0083  0.0322 0.858395
## Residuals     44 11.2960  0.2567
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

All of these numbers should look familiar.

18.9 A 2x3 Factorial Example

Factorial designs let you study the effects of one factor across multiple levels of another factor (or factors). In this made-up example, we'll study the effect of two kinds of diets: "Atkins" and "pea soup" on the systolic blood pressure (BP, in mm Hg) for three exercise levels: "none", "a little", and "a lot". A systolic blood pressure less than 120 mm Hg is considered normal. 20 subjects participated in each of the 2x3 = 6 groups for a total of 120 subjects. Here's how to load in the data from the course website and order the levels:

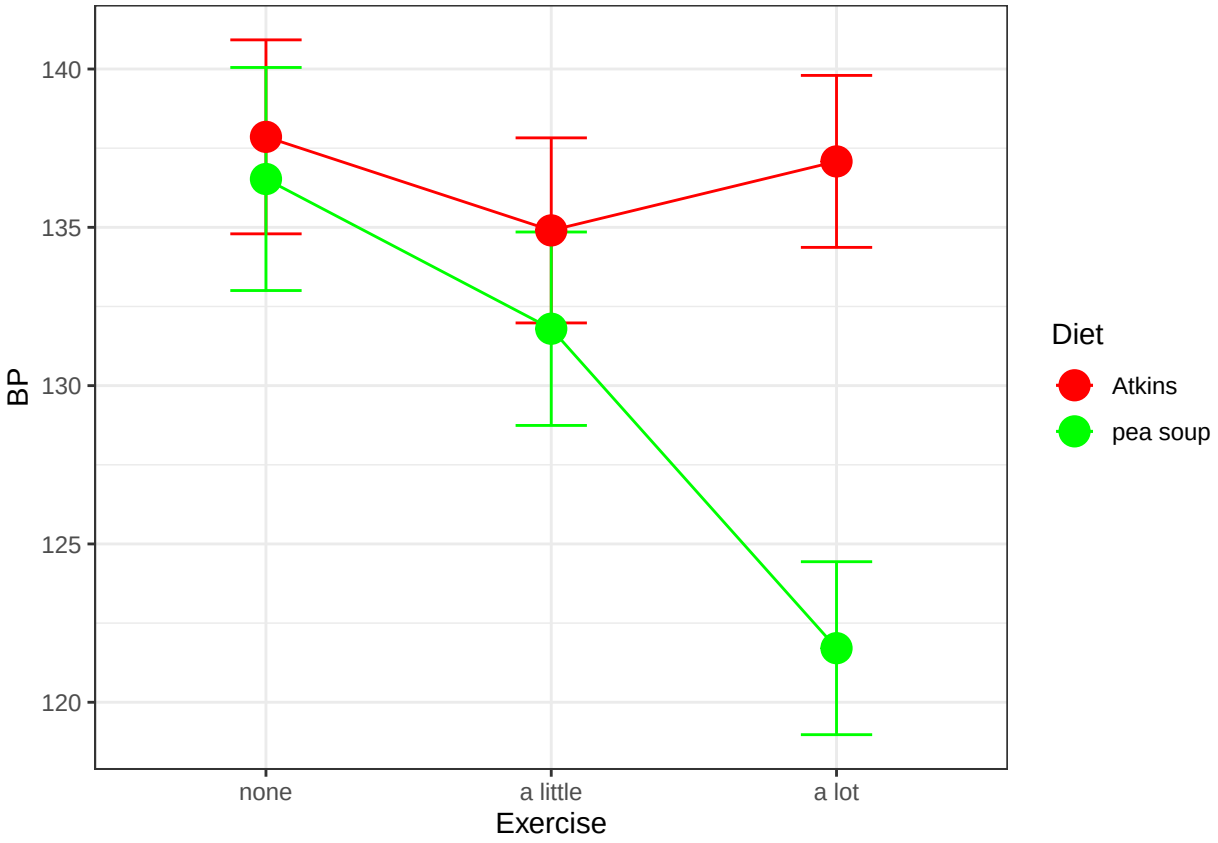
```
# data.3 <- read.csv('http://courses.washington.edu/psy524a/datasets/DietExercise.csv')
# data.3$Diet <- factor(data.3$Diet,levels = c('Atkins','pea soup'))
# data.3$Exercise <- factor(data.3$Exercise,levels = c('none','a little','a lot'))
```

The data is stored in 'long format' like this:

Table 18.10:

BP	Diet	Exercise
125.6032	Atkins	none
137.7546	Atkins	none
122.4656	Atkins	none
158.9292	Atkins	none
139.9426	Atkins	none
122.6930	Atkins	none
142.3114	Atkins	none
146.0749	Atkins	none
143.6367	Atkins	none
130.4192	Atkins	none
157.6767	Atkins	none
140.8476	Atkins	none
125.6814	Atkins	none
101.7795	Atkins	none
151.8740	Atkins	none
134.3260	Atkins	none
134.7571	Atkins	none
149.1575	Atkins	none
147.3183	Atkins	none
143.9085	Atkins	none
148.7847	Atkins	a little
146.7320	Atkins	a little
136.1185	Atkins	a little
105.1597	Atkins	a little
144.2974	Atkins	a little
134.1581	Atkins	a little
132.6631	Atkins	a little
112.9387	Atkins	a little
127.8277	Atkins	a little
141.2691	Atkins	a little
155.3802	Atkins	a little
133.4582	Atkins	a little
140.8151	Atkins	a little
134.1929	Atkins	a little
114.3441	Atkins	a little
128.7751	Atkins	a little
129.0857	Atkins	a little
134.1103	Atkins	a little
151.5004	Atkins	a little
146.4476	Atkins	a little
132.5321	Atkins	a lot
131.1996	Atkins	a lot

Here's a plot of the means with error bars:



Here we've defined the row factor to be *Diet* and the column factor to be *Exercise*.

From the graph it looks like the Atkins diet has little effect on systolic blood pressure across exercise levels, but the pea soup diet does seem to lead to lower BP for higher levels of exercise.

The math behind running a 2-factor ANOVA on this design is the same as for the 2x2 example above.

18.9.1 Calculating MS_{wc} for the 2x3 example

SS_{wc} and MS_{wc} are calculated the same way as for the 2x2 example. We sum up the sums of squared deviation of each score from the mean of the cell that each score came from. Here's the table of the SS for each of the cells:

Table 18.11: SS_{wc} for the 2x3 example

	none	a little	a lot
Atkins	3565.484	3245.913	2802.910
pea soup	4715.587	3546.061	2834.728

SS_{wc} is therefore

$$3565.48 + 4715.59 + 3245.91 + 3546.06 + 2802.91 + 2834.73 = 20710.7$$

Again, each cell contributes n- 1 degrees of freedom to SS_{wc}, so the degrees of freedom for all cells is N-k, where k is the total number of cells and N is the total sample size (n $\times$ k):

$$df_{wc} = 120 - 6 = 114$$

Mean-squared within-cell is:

$$MS_{wc} = \frac{SS_{wc}}{df_{wc}} = \frac{2.0710683 \times 10^4}{114} = 181.6727$$

Remember this number: $MS_{wc} = 181.6727$. It will be the common denominator for all of the F-tests for this data set.

Like for the 2x2 example, the main effects are done by computing the sums of squared deviation from the rows and column means from the grand mean, weighted by the total number of subjects contributing to each row or column mean.

Here's a table of the means, along with the row and column means:

Table 18.12: 2x3 Example: Row and Column Means

	none	a little	a lot	means
Atkins	137.8579	134.9029	137.0820	136.6142
pea soup	136.5260	131.7978	121.7074	130.0104
means	137.1920	133.3503	129.3947	133.3123

18.9.2 Main effect for columns (Exercise)

The main effect of columns (Exercise), the sums of squared deviation from the grand mean is:

$$(137.192 - 133.312)^2 + (133.35 - 133.312)^2 + (129.395 - 133.312)^2 = 15.0521 + 0.0014 + 15.3476 = 30.4011$$

Since there are 2 levels for the row factor, there are $20 \times 2 = 40$ subjects for each column mean. So SS_{col} is:

$$SS_C = (40)(30.4011) = 1216.032$$

There are 3 means contributing to SS_C, so the degrees of freedom is $df_C = 2 - 1 = 1$

MS_C is therefore

$$\frac{SS_C}{df_C} = \frac{1216.0320}{2} = 608.0160$$

The F-statistic for this main effect is MS_C divided by our common denominator, MS_{wc}

$$F = \frac{MS_C}{MS_{wc}} = \frac{608.0160}{181.6727} = 3.3468$$

The p-value for the main effect of Exercise is:

```
1-pf(3.3468,2,114)
```

```
## [1] 0.03868787
```

18.9.3 Main effect for rows (Diet)

The main effect of rows (Diet), the sums of squared deviation from the grand mean is:

$$(136.614 - 133.312)^2 + (130.01 - 133.312)^2 = 10.9025 + 10.9025 = 21.805$$

This time, since there are 3 levels for the column factor, there are $20 \times 3 = 60$ subjects for each row mean. So SS_{row} is:

$$SS_R = (60)(21.8051) = 1308.3198$$

There are 2 means contributing to SS_R, so the degrees of freedom is df_R = 3 -1 = 2

MS_R is therefore

$$\frac{SS_R}{df_R} = \frac{1308.3198}{1} = 1308.3198$$

The F-statistic for this main effect is MS_R divided by our common denominator, MS_{wc}

$$F = \frac{MS_R}{MS_{wc}} = \frac{1308.3198}{181.6727} = 7.2015$$

The p-value for the main effect of Diet is:

```
1-pf(7.2015,1,114)
```

```
## [1] 0.008368909
```

18.9.4 Interaction Between Diet and Exercise

The total sums of squares is 2.4404637×10^4, so we can calculate SS_{RXC} by;

$$SS_{RXC} = SS_{total} - SS_R - SS_C - SS_{wc} = 24404.6372 - 1308.3198 - 1216.0320 - 20710.6827 = 1169.6028$$

The degrees of freedom for this interaction term is:

$$df_{RXC} = (n_{rows} - 1)(n_{cols} - 1) = (2 - 1)(3 - 1) = 2$$

The mean-squared error for the interaction is

$$\frac{SS_{RXC}}{df_{RXC}} = \frac{1169.6028}{2} = 584.8014$$

Using SS_{wc} for the denominator again, the F-statistic is:

$$F = \frac{MS_{RXC}}{MS_{wc}} = \frac{584.8014}{181.6727} = 3.2190$$

The p-value for the interaction is:

```
1-pf(3.2190,2,114)
```

```
## [1] 0.04365711
```

Here's how to run the 2-factor ANOVA in R:

```
anova3.out <- anova(lm(BP ~ Diet*Exercise,data = data.3))
anova3.out
```

```
## Analysis of Variance Table
##
## Response: BP
##                Df  Sum Sq Mean Sq F value   Pr(>F)
## Diet            1  1308.3 1308.32  7.2015 0.008369 **
## Exercise        2  1216.0  608.02  3.3468 0.038689 *
## Diet:Exercise   2  1169.6  584.80  3.2190 0.043658 *
## Residuals     114 20710.7  181.67
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Using APA format we'd say:

There is a main effect of Diet. $F(1,114) = 7.2015$, $p = 0.0084$.

There is a main effect of Exercise. $F(2,114) = 3.3468$, $p = 0.0387$.

There is a significant interaction between Diet and Exercise. $F(2,114) = 3.2190$, $p = 0.0437$.

All three hypothesis tests are statistically significant, but this doesn't really tell us much about what's driving the effects of Diet and Exercise on BP. As discussed above when we plotted the results, what seems to be happening is that only the subjects on the pea soup diet are influenced by Exercise.

It might make sense, instead, to run two ANOVAs on the data, one for the Atkins diet and one for the pea soup diet. We expect to find that most of variability across the means is driven by the effect of Exercise for the pea soup dieters.

18.10 Simple Effects

Running ANOVAs on subsets of the data like this is called a *simple effects* analysis. Running separate ANOVAs on each level of Diet is studying the simple effects of *Exercise by Diet*. I remember this by replacing the word 'by' with 'for every level of'. That is, this simple effect analysis is studying the effect of Exercise on BP *for every level of* Diet.

Running simple effects is *almost* as simple as running separate ANOVA's for each level of Diet. In fact, let's start there. We can use the `subset` function to pull out the data for each of the two diets":

```
# Atkins diet:
anova3.out.Atkins <- anova(lm(BP ~ Exercise,data = subset(data.3,Diet == 'Atkins') ))
anova3.out.Atkins
```

```
## Analysis of Variance Table
##
## Response: BP
##           Df Sum Sq Mean Sq F value Pr(>F)
## Exercise   2   93.9  46.939  0.2783 0.7581
## Residuals 57 9614.3 168.672
```

```
# pea soup diet:
anova3.out.peasoup <- anova(lm(BP ~ Exercise,data = subset(data.3,Diet == 'pea soup') ))
anova3.out.peasoup
```

```
## Analysis of Variance Table
##
## Response: BP
##           Df  Sum Sq Mean Sq F value   Pr(>F)
## Exercise   2  2291.8 1145.88  5.8862 0.004744 **
## Residuals 57 11096.4  194.67
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

There's one more think we can do to increase the power of these tests. If we assume homogeneity of variance, then it makes sense to use the denominator of the two-factor ANOVA, $MS_{wc} = 181.6727$, for both of these F-tests since this should be a better estimate of the population variance - and it has a larger *df* which helps with power.

R doesn't have a function for simple effects, but it's not hard to do it by hand. All we need to do is pull out MS_{wc} from the output from the original two factor ANOVA and recalculate our F-statistics and p-values:

```
# from the 2-factor ANOVA, MS_wc is the fourth mean squared in the list
MS_wc <- anova3.out$`Mean Sq`[4]
df_wc <- anova3.out$Df[4]

# Atkins
MS_Atkins <- anova3.out.Atkins$`Mean Sq`[1]
df_Atkins <- anova3.out.Atkins$Df[1]
F_Atkins <- MS_Atkins/MS_wc
p_Atkins <- 1-pf(F_Atkins,df_Atkins,df_wc)

# pea soup
MS_peasoup <- anova3.out.peasoup$`Mean Sq`[1]
df_peasoup <- anova3.out.peasoup$Df[1]
F_peasoup <- MS_peasoup/MS_wc
p_peasoup <- 1-pf(F_peasoup,df_peasoup,df_wc)

sprintf('Atkins:    F(%d,%d)= %5.4f,p = %5.6f',df_Atkins,df_wc,F_Atkins,p_Atkins)
```

```
## [1] "Atkins:    F(2,114)= 0.2584,p = 0.772759"
```

```
sprintf('pea soup:  F(%d,%d)= %5.4f,p = %5.6f',df_peasoup,df_wc,F_peasoup,p_peasoup)
```

```
## [1] "pea soup:  F(2,114)= 6.3074,p = 0.002523"
```

The p-values didn't change much when substituting MS_{wc}, but every little bit of power helps.

18.11 Additivity of Simple Effects

These two simple effects have an interesting relation with the three tests from the original 2-factor ANOVA. It turns out that the SS associated with these two simple effects add up the SS associated with the main effects of Exercise plus the SS for the interaction between Diet and Exercise. In math terms:

$$SS_{exercisebyAtkinsdiet} + SS_{exercisebypeasoupdiet} = SS_{exercise} + SS_{exerciseXdiet}$$

You can see that here:

```
# Adding SS's for the two simple effects of Exercise by Diet:

anova3.out.Atkins$`Sum Sq`[1] + anova3.out.peasoup$`Sum Sq`[1]
```

```
## [1] 2385.635
```

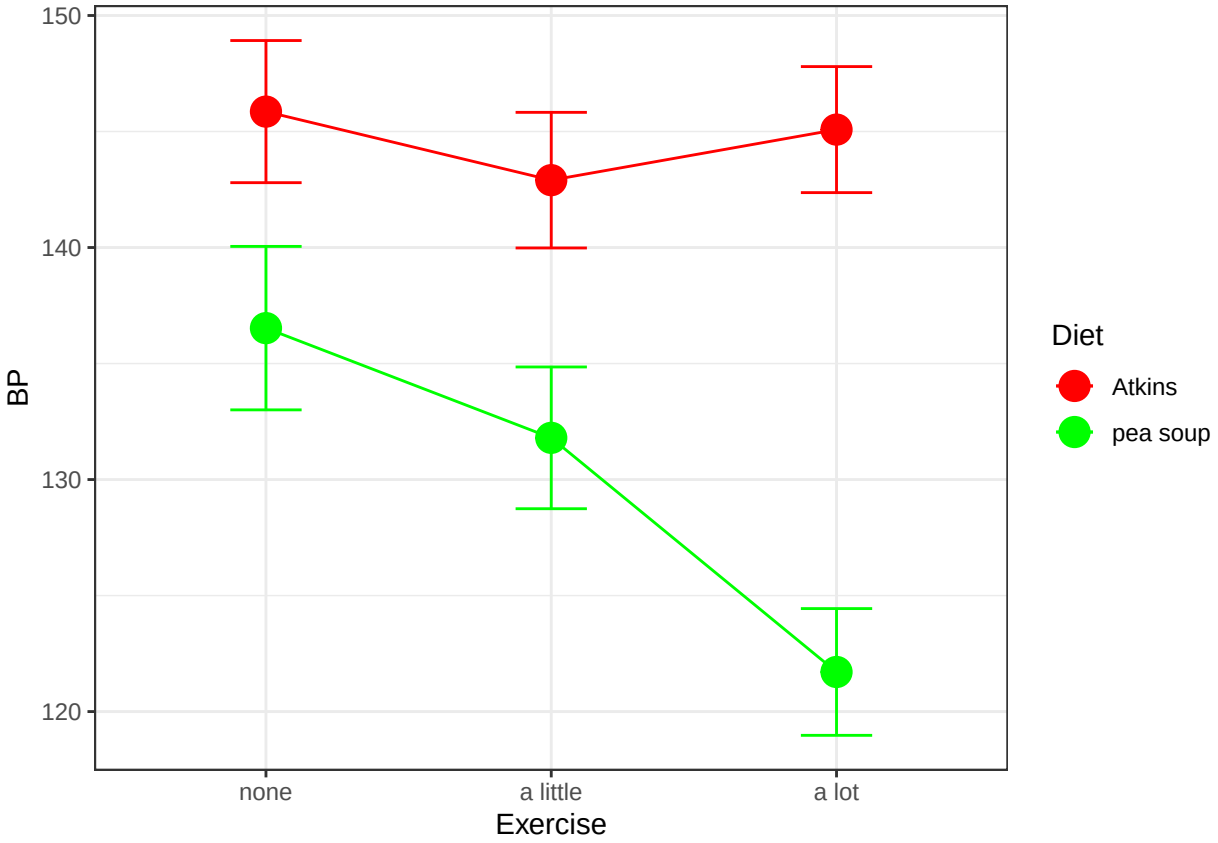
```
# Adding SS's for the main effect of Diet and the interaction
# (second and third in the list of SS's)

sum(anova3.out$`Sum Sq`[c(2,3)])
```

```
## [1] 2385.635
```

Note also that the degrees of freedom for both sets add up to 4

The pie charts below show how SS_{total} is divided up into the different sums of squares for the standard analysis (main effects and interaction) and for the simple effects analysis.

Our simple effects analysis is just another way of breaking down the SS associated with the main effects for columns and the interaction, since the main effect for Diet has no influence on this set of simple effects. You can visualize this by thinking about what would happen to our results of the shapes of the two effects were the same, but if one curve were to be shifted above or below another. For example if our results had come out like this, with an overall higher systolic blood pressure for the Atkins diet:



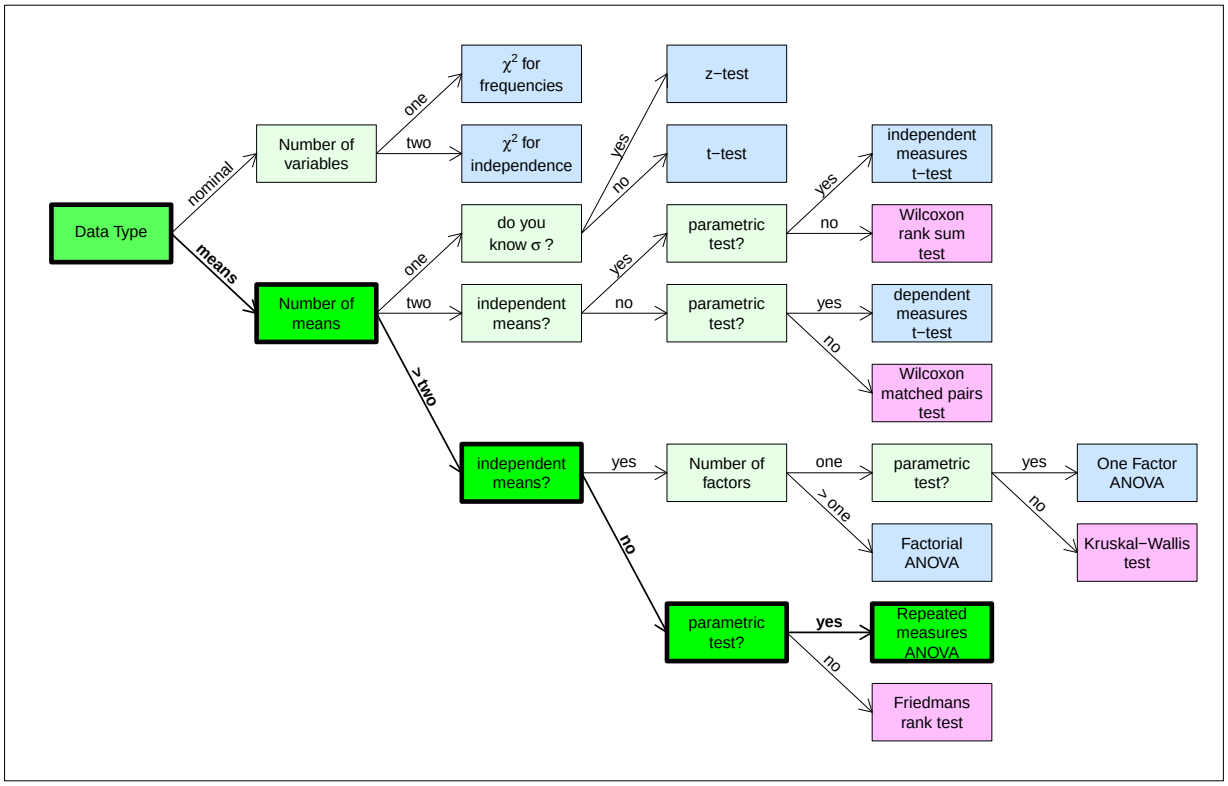
Our simple effects analysis of columns by row would have come out the same. This is because shifting up the Atkins group only increased the main effect of Diet, and not the main effects of Exercise by Diet, the main effect of Exercise, or the interaction between Exercise and Diet.

Chapter 19

Repeated Measures ANOVA

The repeated measures ANOA is an extension of the dependent (or repeated) measures t-test. The most common case for a one-factor repeated measures ANOVA is when each subject provides two or more measures.

You can find this test in the flow chart here:



We'll start with an example of a repeated measures t-test and build on that.

19.1 Review: Dependent measures t-test

Suppose you want to see if a specific exercise program affects body weight. You find 6 subjects and measure their weights (in kg) before the program and after three months. You can load in the (fake) data yourself in the course website:

```
data1 <- read.csv('http://courses.washington.edu/psy524a/datasets/SubjectExerciseANOVATtestdependent.csv')
```

Some of these notes are based on the website (https://statistics.laerd.com/) which has some nice examples at the level appropriate for our class. The example in this chapter comes from (https://statistics.laerd.com/statistical-guides/repeated-measures-anova-statistical-guide-2.php)

To run a dependent t-test on these results we create a new column of differences and run a one-sample t-test on that column. Here's a table of the results with the column of differences (after - before):

Table 19.1:

Before	After Three Months	D
45	50	5
42	42	0
36	41	5
39	35	-4
51	55	4
44	49	5

Here's how to run a dependent measures t-test on this data in R:

```
x <- data1$afterthreemonths
y <- data1$before
t.test.out <- t.test(x,y,paired = T,alternative = 'two.sided')
sprintf('t(%d) = %5.4f, p = %5.4f',t.test.out$parameter,t.test.out$statistic,t.test.out$p.value)
```

```
## [1] "t(5) = 1.6425, p = 0.1614"
```

19.2 Repeated measures with more than two levels

What if we want to measure each subject again after six months? This requires measuring the differences between means across three different groups. ANOVA!

Unfortunately, while a dependent measures t-test is easier to compute than an independent measures t-test, a dependent measures ANOVA (often called within subjects or repeated measures ANOVA) is actually a little more complicated than the standard independent measures ANOVA.

Here's how to load in the new data set from the course website:

```
data2 <- read.csv('http://courses.washington.edu/psy524a/datasets/SubjectExerciseANOVAdependent.csv')
```

This data is stored in 'long format', where each row in the table corresponds to a different observation as opposed to 'wide format' where each row corresponds to a subject.

With long format, the first column, 'subject', determines the subject for the observation. The second column 'time' determines the condition for that observation. You'll see that each subject has three observations, consistent with the repeated measures design:

Table 19.2:

Subject	Time	Weight
S1	before	45
S1	after three months	50
S1	after six months	55
S2	before	42
S2	after three months	42
S2	after six months	45
S3	before	36
S3	after three months	41
S3	after six months	43
S4	before	39
S4	after three months	35
S4	after six months	40
S5	before	51
S5	after three months	55
S5	after six months	59
S6	before	44
S6	after three months	49
S6	after six months	56

Long format may seem inefficient, but it's a very convenient way to store your results, especially for complicated experimental designs. This way, every time you acquire a new measurement, you just add a new row to your data with the fields determining exactly which condition/subject the data came from. Most of the functions in R are designed for long format. `t.test` is an exception since it takes in 'x' and 'y' variables, but `t.test` can also accept data in long format.

19.2.1 Converting long format to wide

Sometimes it's useful to see your results in 'wide' format, where each row is a subject. This can be done with R using a variety of functions. Here's how to convert to wide format with the `reshape` function. You have to tell it the subject variable `idvar = "subject"` and the independent variable `timevar = "time"`:

```
data2.wide <- reshape(data2, idvar ="subject", timevar = "time", direction = "wide")
```

The data in wide format looks like this:

Table 19.3:

Subject	Before	After Three Months	After Six Months
S1	45	50	55
S2	42	42	45
S3	36	41	43
S4	39	35	40
S5	51	55	59
S6	44	49	56

19.2.2 As an independent measures ANOVA

Let's first pretend that this is an independent measures design and run the ANOVA that way, ignoring the 'subject' field. We'll also pull out the SS's, MS', and df's from the table

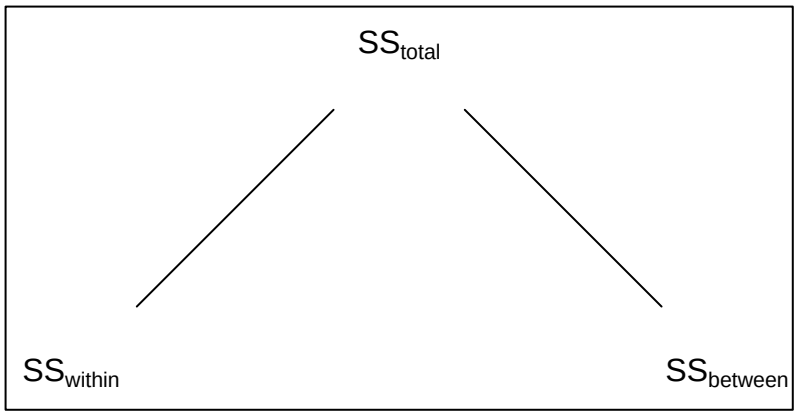
```
anova.out.1 <- anova(lm(weight ~ time,data2))
anova.out.1
```

```
## Analysis of Variance Table
##
## Response: weight
##           Df Sum Sq Mean Sq F value Pr(>F)
## time       2 143.44  71.722  1.5036  0.254
## Residuals 15 715.50  47.700
```

```
SS_between <- anova.out.1$`Sum Sq`[1]
df_between <- anova.out.1$Df[1]
MS_between <- anova.out.1$`Mean Sq`[1]

SS_within <- anova.out.1$`Sum Sq`[2]
df_within <- anova.out.1$Df[2]
MS_within <- anova.out.1$`Mean Sq`[2]

SS_total = SS_between+SS_within  # remember?
```

Remember, for a 1-factor ANOVA like this breaks down the total sums of squares so that

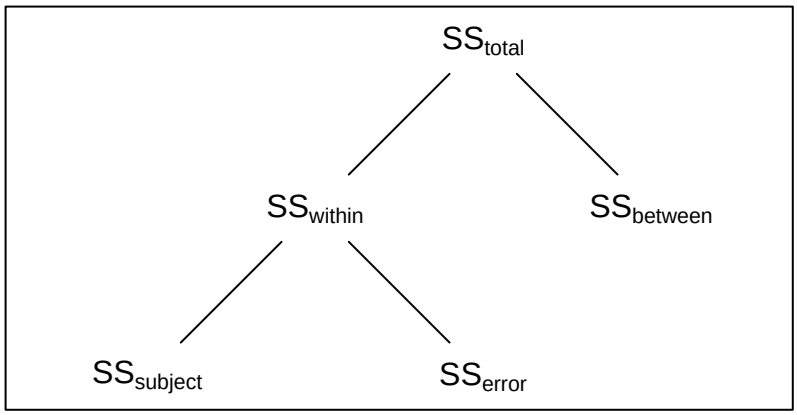
$$SS_{total} = SS_{between} + SS_{within}$$



Here, $SS_{between} = 143.4444$ reflects the variability across the 'time' factor, and $SS_{within} = 715.5$ is the variability within each level of time.

You wouldn't want to analyze a repeated measures design this way because you wouldn't be taking into account the variability in weights across the subjects. Treated as an independent measures ANOVA, this variability between subjects is part of SS_{within} (or 'Residuals' in R's output).

Thus, for the denominator of the ANOVA for a repeated measures design we only want the component of SS_{within} that's not associated with the variability across subjects. This is done by breaking down SS_{within} into two components: one component associated with within-subject variability, called $SS_{subject}$ and the remaining variability called SS_{error}:



For ANOVA, the way of accounting for this source of variability is to subtract it out of the SS_{within}, which will make the denominator of the F-test smaller, and therefore the F-statistic larger.

We can calculate $SS_{subject}$ in R by doing something a little weird - we'll run another one-factor ANOVA, but this time with 'subject' as our factor. We'll pull out the SS and Df from this analysis and call them $SS_{subject}$ and $df_{subject}$:

```
anova.out.2 <- anova(lm(weight ~ subject,data2))
anova.out.2
```

```
## Analysis of Variance Table
##
## Response: weight
##           Df Sum Sq Mean Sq F value   Pr(>F)
## subject    5 658.28 131.656  7.8731 0.001706 **
## Residuals 12 200.67  16.722
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
SS_subject <- anova.out.2$`Sum Sq`[1]
df_subject <- anova.out.2$Df[1]
```

The p-value isn't important here - it's just telling us the significance of the variability across the the subjects' weights after averaging across the three time conditions.

19.2.3 The denominator for the repeated measures ANOVA: SS_{error}

$SS_{subject}$ is the sums of squared deviation of each subject from the grand mean, and is exactly the source of variability that we want to remove from our analysis.

For repeated measures ANOVA we do this by literally subtracting $SS_{subject}$ from SS_{within} in the original independent measures ANOVA. We call this new sums of squared SS_{error}

$$SS_{error} = SS_{within} - SS_{subject}$$

which, for our example is:

```
SS_error <- SS_within - SS_subject
SS_error
```

```
## [1] 57.22222
```

SS_{error} will serve as the sums-of-squares for the denominator for the repeated-measures ANOVA. To calculate MS_{error} we need to divide by the degrees of freedom for SS_{error}. This is done in the same way as for sums-of-squared error:

$$df_{error} = df_{within} - df_{subject}$$

Which is, for our example:

```
df_error <- df_within-df_subject
```

The new denominator is:

```
MS_error <- SS_error/df_error
MS_error
```

```
## [1] 5.722222
```

The F-statistic for the repeated measures uses the $MS_{betweeen}$ from the original independent measures ANOVA for the numerator, but instead uses MS_{error} for the new denominator:

```
F_stat <- MS_between/MS_error
F_stat
```

```
## [1] 12.53398
```

Finally the p-value is calculated using the appropriate degrees of freedom:

```
p <- 1-pf(F_stat,df_between,df_error)
p
```

```
## [1] 0.001885591
```

After subtracting the between-subject variability for the subjects, our F-test has become statistically significant.

19.3 Repeated measures ANOVA with R

The `lm` function that we used for independent measures ANOVA can't deal with repeated measures designs. But there are a few packages out there that can. I'm going to use the most general version, called `lmer` from the `lme4` package. For reasons we'll discuss later, we also have to include the `lmerTest` package because the creators of `lmer` are opposed to providing p-values - again more on that later when we get into linear regression.

Once you have the packages installed, the use of `lmer` is much like it was for `lm` where we define the 'model' as `weight ~ time`. But now we have to tell the function which factor defines our subject with `+ (1|subject)`. It might help to think of (1|subject) as a fraction, where subject is in the denominator - like how we subtracted $SS_{subject}$ from SS_{within} to make the new denominator for the repeated measures ANOVA.

```
anova.out.3 <- anova(lmer(weight ~ time + (1|subject),data = data2))
anova.out.3
```

```
## Type III Analysis of Variance Table with Satterthwaite's method
##      Sum Sq Mean Sq NumDF DenDF F value   Pr(>F)
## time 143.44  71.722     2    10  12.534 0.001886 **
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

You should recognize the SS, MS, df F and p-value from the analysis that we did by hand.

Don't worry about the 'Satterthwaite's method' thing for now. It's a way of dealing with unbalanced designs. We'll get into this once we've learned more about linear regression.

Here's how to pull out the values in the output to make an APA formatted string:

```
sprintf('F(%g,%g) = %5.4f, p = %5.4f',
        anova.out.3$NumDF,anova.out.3$DenDF,anova.out.3$`F value`,anova.out.3$`Pr(>F)`)
```

```
## [1] "F(2,10) = 12.5340, p = 0.0019"
```

19.4 Sphericity

Remember, for ANOVA, we assume homogeneity of variance, which is the assumption that the samples of each of our cells are drawn from populations that have equal variance - even if the null hypothesis is false. We've discussed earlier that this assumption is fundamenental to the ANOVA hypothesis test.

For repeated measures ANOVA, a related but stronger assumption needs to be met. This is called *sphericity*. Sphericity is the condition where the variances of the *differences* between all combinations of related groups (levels) are equal. In our example, the asssumption of sphericity could be violated if perhaps weights stabilize within subjects bewteen three months and six months, so there is more variability in the differences between the first two time-points as the second and third time points in the experiment.

Let's look at our example.

Table 19.4:

	Before	After Three Months	After Six Months	2-1	3-1	3-2
S1	45	50	55	5	10	5
S2	42	42	45	0	3	3
S3	36	41	43	5	7	2
S4	39	35	40	-4	1	5
S5	51	55	59	4	8	4
S6	44	49	56	5	12	7

Here's a table of the variances for each column of this table:

Table 19.5:

	Before	After Three Months	After Six Months	2-1	3-1	3-2
s^2	26.97	53.07	63.07	13.9	17.37	3.07

With homogeneity of variance, the first three variance measures should be about the same. They vary by a factor of about two which seems large, but by now you should appreciate that it's actually not unusual for a sample size this small.

Under sphericity, the next three variances should be about the same. (Technically, sphericity is based on variability in the off-diagonals of the covariance matrix of the data, but the intuition is the same).

Violations of sphericity lead to inflated values of F, and therefore more type I errors than what is expected from the tables.

19.4.1 Mauchly's test of Sphericity

The standard way to test for a violation of sphericity is 'Mauchly's Test of Sphericity'. We won't go into the details here except to say that it is a $\chi 2$ test with k-1 degrees of freedom. If this test turns out statistically significant, it means that there's a violation of sphericity in your data, which can lead to inflated p-values.

There are at least three modifications of the ANOVA that correct for violations of sphericity: the 'Greenhouse-Geisser', the 'Huyn-Feldt', and the 'Lower-bound' (SPSS provides all three). All three work by decreasing degrees of freedom of both the numerator and the denominator by a multiplicative factor. You still use the F-statistic from the original repeated measures ANOVA. The result is that the F statistic stays the same but the critical values go down and p-values go up. The field seems to be settling on the 'Greenhouse-Geisser' correction as the default correction for sphericity.

I've been down some rabbit holes to find the most natural way to deal with sphericity (see this useful reference). I've settled on the `anova_test` function in the package 'rstatix'. You tell it your dependent variable `dv = weight`, the name for the subject field `wid = subject` and the within-subject variable `within = time`:

```
anova.out <- anova_test(data = data2, dv = weight, wid = subject, within = time)
anova.out
```

```
## ANOVA Table (type III tests)
##
## $ANOVA
##   Effect DFn DFd      F     p p<.05   ges
## 1   time   2  10 12.534 0.002     * 0.167
##
## $`Mauchly's Test for Sphericity`
##   Effect     W     p p<.05
## 1   time 0.434 0.188
##
## $`Sphericity Corrections`
##   Effect   GGe     DF[GG] p[GG] p[GG]<.05  HFe    DF[HF] p[HF] p[HF]<.05
## 1   time 0.638 1.28, 6.38 0.009         * 0.76 1.52, 7.6 0.005         *
```

Three separate tables are printed out. The first is the result of the repeated measures ANOVA which (thankfully) matches the results from anova(lmer(..)). The second is the result from Mauchly's test of sphericity. You can pull out the p-value from the table like this:

```
anova.out$`Mauchly's Test of sphericity`$p
```

```
## NULL
```

If this p-value is small, then it is suggested that you use a correction for sphericity, which could be either the Greenhouse-Geisser or the Huynh-Feldt correction. The results from both are in the third table. Both the Greenhouse-Geisser and the Huynh-Feldt corrections recalculate the p-value using the original repeated measures ANOVA F-value, but reducing both $df_{between}$ and df_{error} by a multiplicative factor.

19.4.2 Greenhouse-Geisser correction

Here's how to access the multiplicative factor from the Greenhouse-Geisser correction:

```
anova.out$`Sphericity Corrections`$GGe
```

```
## [1] 0.638
```

You can report the results of the Greenhouse-Geisser correction in APA format using this:

```
sprintf('F(%s) = %5.4f, p = %5.4f',
        anova.out$`Sphericity Corrections`$`DF[GG]`,
```

```
        anova.out$ANOVA$F,
        anova.out$`Sphericity Corrections`$`p[GG]`)
```

```
## [1] "F(1.28, 6.38) = 12.5340, p = 0.0090"
```

Just for sanity, let's check that we get our new p-value after adjusting the df's with the Greenhouse-Geisser factor:

```
GGe <- anova.out$`Sphericity Corrections`$GGe

# scale the original df's by G
df_between_new <- GGe*df_between
df_error_new <- GGe*df_error
F_orig <- anova.out.3$`F value`

 1-pf(F_orig,df_between_new,df_error_new)
```

```
## [1] 0.009000177
```

It matches the p-value from the Greenhouse-Geisser correction. So although this factor 'G' is mysterious, at least we know where the adjusted p-value comes from once we have it.

19.4.3 Huynh-Feldt correction

A similar procedure can be used to access the results of the Huynh-Feldt correction:

```
sprintf('F(%s) = %5.4f, p = %5.4f',
        anova.out$`Sphericity Corrections`$`DF[HF]`,
        anova.out$ANOVA$F,
        anova.out$`Sphericity Corrections`$`p[HF]`)
```

```
## [1] "F(1.52, 7.6) = 12.5340, p = 0.0050"
```

Which can be done manually by scaling the degrees of freedom by the 'HFe' factor:

```
Hfe <- anova.out$`Sphericity Corrections`$HFe

# scale the original df's by HFe
df_between_new <- Hfe*df_between
df_error_new <- Hfe*df_error
F_orig <- anova.out.3$`F value`

 1-pf(F_orig,df_between_new,df_error_new)
```

```
## [1] 0.005288238
```

Which matches the p-value from the Huynh-Feldt correction.

For our example, since Mauchly's test of sphericity was not significant, we can use the usual within-subjects method and reject our null hypothesis with the p-value from the regular repeated measures ANOVA. Otherwise, we'd base our decision on the p-value from one of the two corrections.

Which correction to use? According to Wikipedia: 'As a general rule of thumb, the Greenhouse–Geisser correction is the preferred correction method when the epsilon estimate is below 0.75. Otherwise, the Huynh–Feldt correction is preferred.'. I have no idea why.

19.5 Apriori contrasts for repeated measures ANOVA

There isn't a concensus on how Apriori and post hoc comparisons should be made with the repeated measures ANOVA. Since the difference between repeated measures and independent measures ANOVAs is the denominator, or 'error' term, you'd think that contrasts for repeated measures ANOVA would be the same as for indepenent, except

that you'd use MS_{error} instead of MS_{within} in the denominator. However, apparently, this approach is too sensitive to violations of the assumption of sphericity.

Instead, when comparing two means, statisticians (for example Howell) suggest that you simply run independent measures t-tests (or independent measures ANOVAS) on each pair of means. These tests will only use a subset of the data for the denominator which gives you less power (fewer degrees of freedom in the denominator), but will be less biased due to issues with sphericity. Of course, you will also have to correct for familywise error using your favorite method, like Bonferroni.

Chapter 20

Correlation and Regression

You'll often hear the words 'correlation' and 'regression' in the same context. They are essentially the same thing - both are about the relation between one or more independent variables and a dependent variable. We'll start with regression by introducing the 'general linear model' and apply it to some data from the survey.

20.1 The General Linear Model

A huge part of inferential statistics can be discussed in the framework of 'The General Linear Model', or the *GLM*. You can actually consider just about *all* inferential statistics in the context of the GLM, including correlations, regression, ANOVA and even χ^2 tests. Here we'll use a specific example to illustrate the relationship between these things.

There are two main assumptions for the GLM. The first is that you can predict a dependent variable as a linear combination of independent variables. A linear combination of a set of numbers is the sum of the numbers after multiplying each by a constant called a *weight* or *coefficient*. For now we'll assume that all variables are continuous, and not categorical. Formally, if X_i is your set of independent variables, called *regressors*, and β_i is a corresponding set of coefficients, then our dependent variable Y can be predicted as:

$$\hat{Y} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta k X_k$$

Putting a 'hat' ('^' symbol) on top of a variable means that this a prediction, or estimation, of that variable. So $\hat{Y}$, pronounced 'Y-hat', is the GLM's prediction of the dependent variable, Y.

The difference between the prediction $\hat{Y}$ and the data Y is called the *error* or *residual*. A second assumption of the GLM is called *homoscedasticity* which is that the residuals are distributed normally with mean zero and some standard deviation σ. We write this as:

$$Y - \hat{Y} \sim N(0, \sigma)$$

Putting this together, you'll sometimes see the GLM written as:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta k X_k + N(0, \sigma)$$

20.2 Example: predicting student's heights from their mother's heights

Let's work with a specific example with a single independent variable. With a single independent variable the GLM simplifies to:

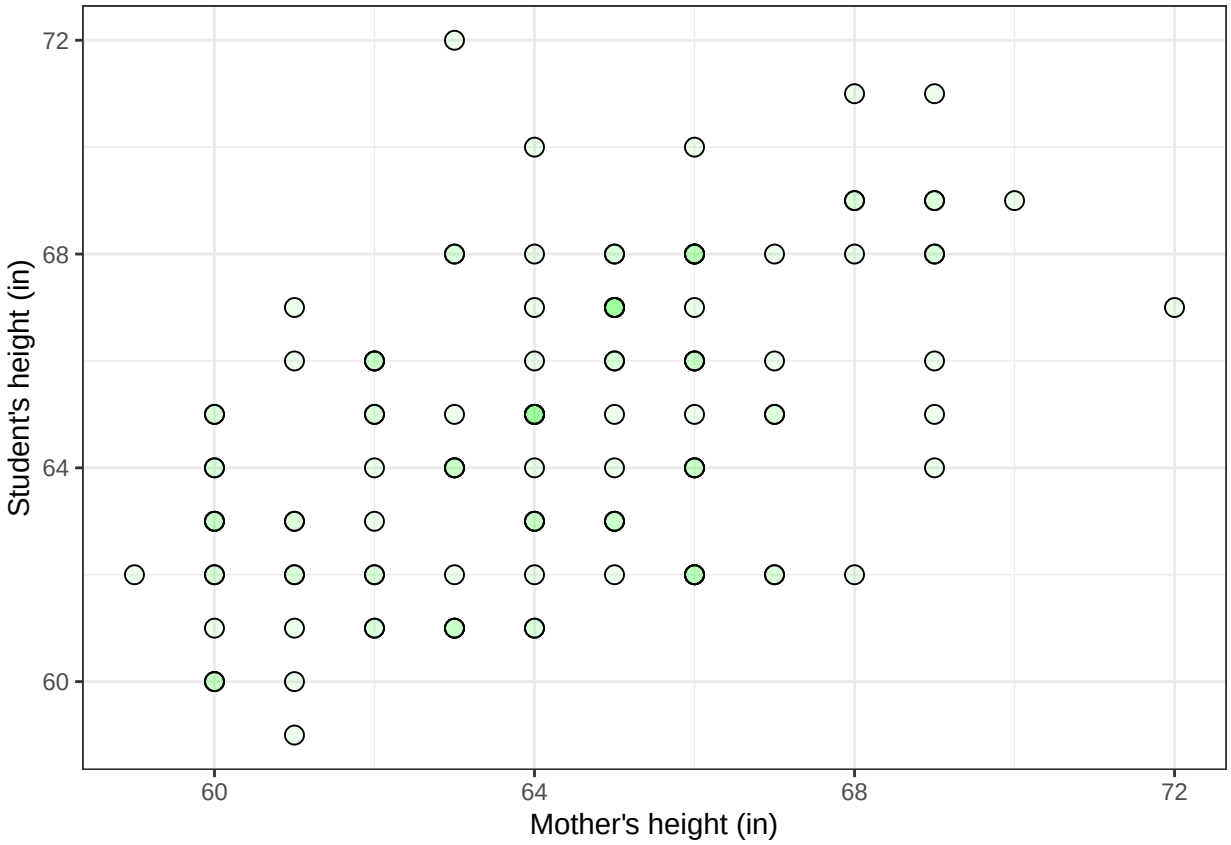
$$Y = \beta_0 + \beta_1 X_1 + N(0, \sigma)$$

We'll use survey and try to predict the height of the students that identify as women from their mother's heights.

```
survey <-
  read.csv("http://www.courses.washington.edu/psy315/datasets/Psych315W21survey.csv")

id <- survey$gender== "Woman" &
  !is.na(survey$height) & !is.na(survey$mheight) &!is.na(survey$pheight) &
  survey$mheight > 55  &  # remove outliers
  survey$height > 55

X <- survey$mheight[id]
Y <- survey$height[id]
Z <- survey$pheight[id]  # save fathers heights for later
heights <- data.frame(student = Y, mother = X, father = Z)
```

It helps to visualize the data as a scatterplot:

```
n <- length(X)
p <- ggplot(heights, aes(x=X, y=Y)) +
  geom_point(shape = 21,size= 3) +
  geom_point(shape = 21,size= 3,fill = "green",alpha = .075) + theme_bw() +
  xlab('Mother\'s height (in)') + ylab('Student\'s height (in)')
plot(p)
```



There is a clear tendency for taller mothers to have taller daughters.

If we let X be the mother's height and Y be the student's height, then the GLM predicts that:

$$\hat{Y} = \beta_0 + \beta_1 X$$

The GLM predicts a straight line through the scatterplot with a slope β_1 and intercept β_0.

What does it mean by 'fit the data'? For fitting the GLM, we define the goodness of fit as the sums of squares of the residuals:

$$SS_{YX} = \sum (Y - \hat{Y})^2$$

Warning: this chapter has a lot of similar looking symbols and formulas. See the glossary at the end of this chapter for reference.

We need to find the values of β_0 and β_1 that make SS_{YX} as small as possible.

A quick note about squaring - statisticians love squaring things. There are good reasons for this - it's an easy way to make things positive, and squaring has nice properties like having a continuous second derivative, and it's relationship to the mean. Squaring makes finding the best fitting parameters computationally easy and robust.

20.3 Regression

Finding the best-fitting coefficients for the GLM is called *regression*.

For a single independent variable, there is a closed-form solution. If we first calculate the mean of X ($\overline{X}$) and the mean of Y ($\overline{Y}$), then:

$$\beta_1 = \frac{\sum (X - \overline{X})(Y - \overline{Y})}{\sum (X - \overline{X})^2}$$

and

$$\beta_0 = \overline{Y} - \beta_1 \overline{X}$$

The Σ means to sum across all data points. With R, we can calculate these coefficients like this:

```
beta_1 <- sum((X-mean(X))*(Y-mean(Y)))/sum((X-mean(X))^2)
beta_0 <- mean(Y) - beta_1*mean(X)
beta_0
```

```
## [1] 30.78893
```

```
beta_1
```

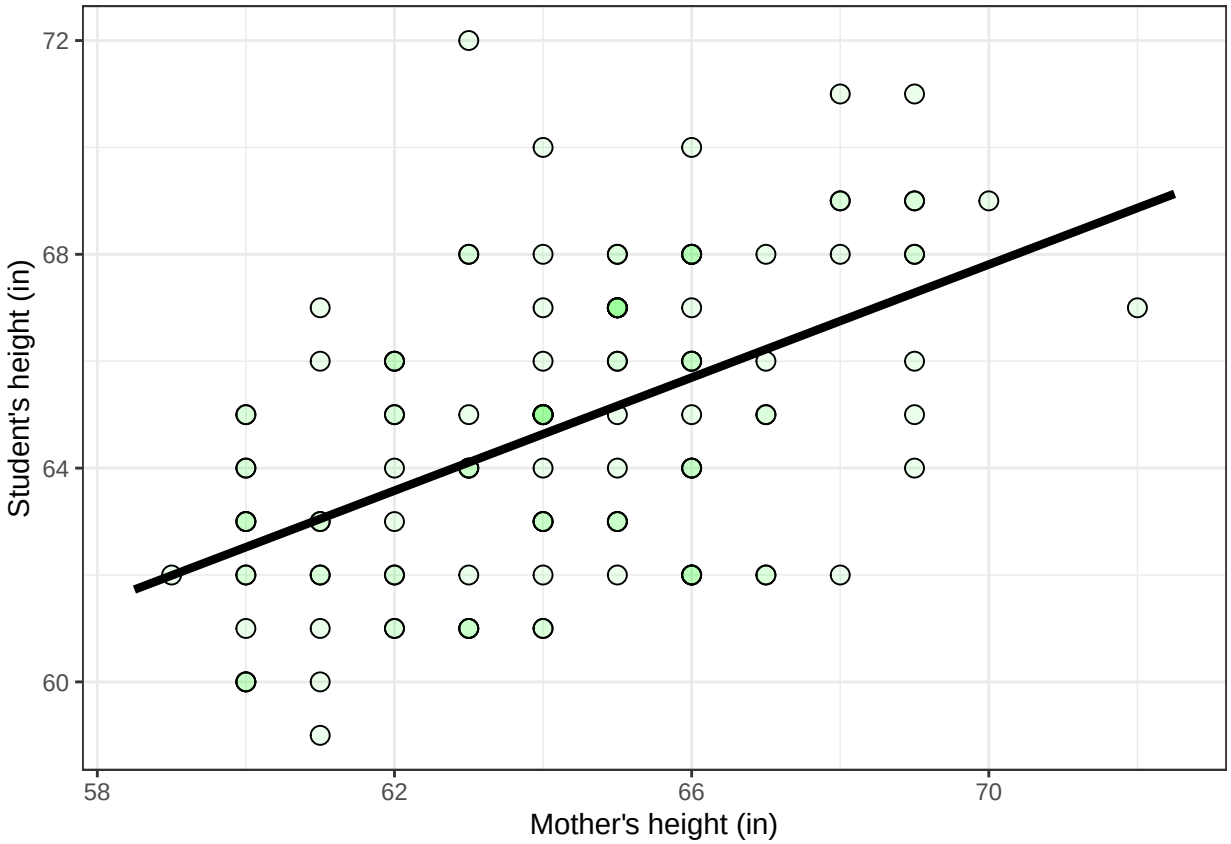
```
## [1] 0.5288607
```

So (rounding to 2 decimal places) the best fitting line through the scatterplot is:

$$\hat{Y} = 30.79 + 0.53X$$

This line is called the *regression line*. Here's the regression line drawn over the scatterplot of the data:

```
xplot <- c(min(X)-.5,max(X)+.5)
yplot <- beta_0 + beta_1*xplot
regression_data <- data.frame(x=xplot,y=yplot)
p_regression_line <- geom_line(data = regression_data,aes(x=x,y=y), linewidth = 1.5 )
p + p_regression_line
```



The sums of squares of the residuals, called SS_{YX}, can be calculated like this:

```
Y_hat <- beta_0 + beta_1*X
SS_YX <- sum((Y-Y_hat)^2)
SS_YX
```

```
## [1] 659.1687
```

No matter how hard you try, you'll never find a better pair of coefficients ($\beta_0 = 30.79$, $\beta_1 = 0.53$) that will produce a sums of squared error less than $SS_{YX} = 659.1687$

20.4 Regression using 'lm'

R has a very flexible function, `lm`, for calculating best-fitting regression coefficients. You've already used it in our previous chapters on ANOVA. It works by defining a 'model' based on the independent and dependent variables in your data frame. For our example, we write the model predicting student's heights (Y) from mother's heights (X) simply as $Y \sim X$. The result is an object that contains the best-fitting coefficients along with a lot of other stuff.

```
lm.out <- lm(student ~ mother, data = heights)
lm.out$coefficients
```

```
## (Intercept)      mother
##  30.7889348   0.5288607
```

There are our values of β_0 and β_1.

The residuals for every data point are in the field `$residuals`, so SS_{YX} can be calculated as:

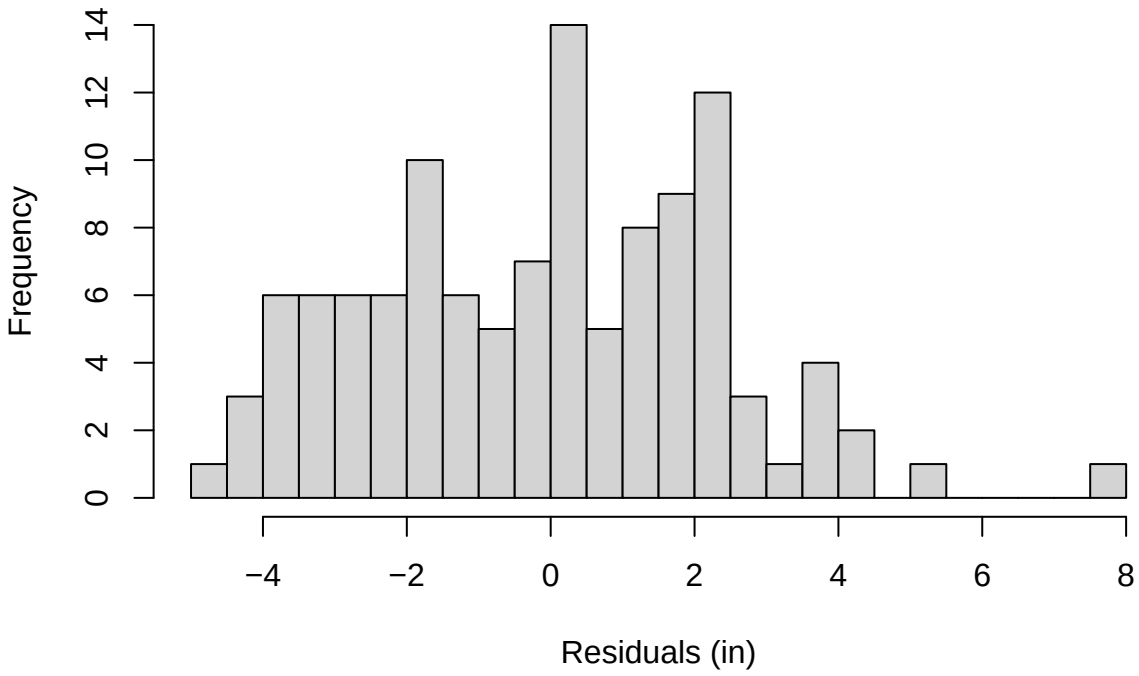
```
sum(lm.out$residuals^2)
```

```
## [1] 659.1687
```

20.5 Residuals

Remember, the second part of the GLM is that the residuals $(Y - \hat{Y})$ are distributed normally. Here's a plot of the residuals:

```
hist(Y-Y_hat,breaks = 20,xlab = 'Residuals (in)',main = NULL)
```



It's not perfect - but it never is. Later on we'll cover the chapter on test for normality Tests for Homogeneity of Variance and Normality that covers how to run a The Shapiro Wilk test or Lilliefors Test test for normality. The Shapiro Wilk tests for normality results in a p-value of p=0.094, which means we fail to reject the null hypothesis that this sample was drawn from a normal distribution. In other words, the residuals are normal enough.

20.6 The standard error of the estimate, S_{YX}

Since the residuals are the error left over after fitting the model, the standard deviation of these residuals is an estimate of the standard deviation of σ, the 'error' at the end of the GLM equation. This is why the standard deviation of the residuals is called the *standard error of the estimate*, and gets its own symbol S_{YX}. Using the formula for the sample standard deviation:

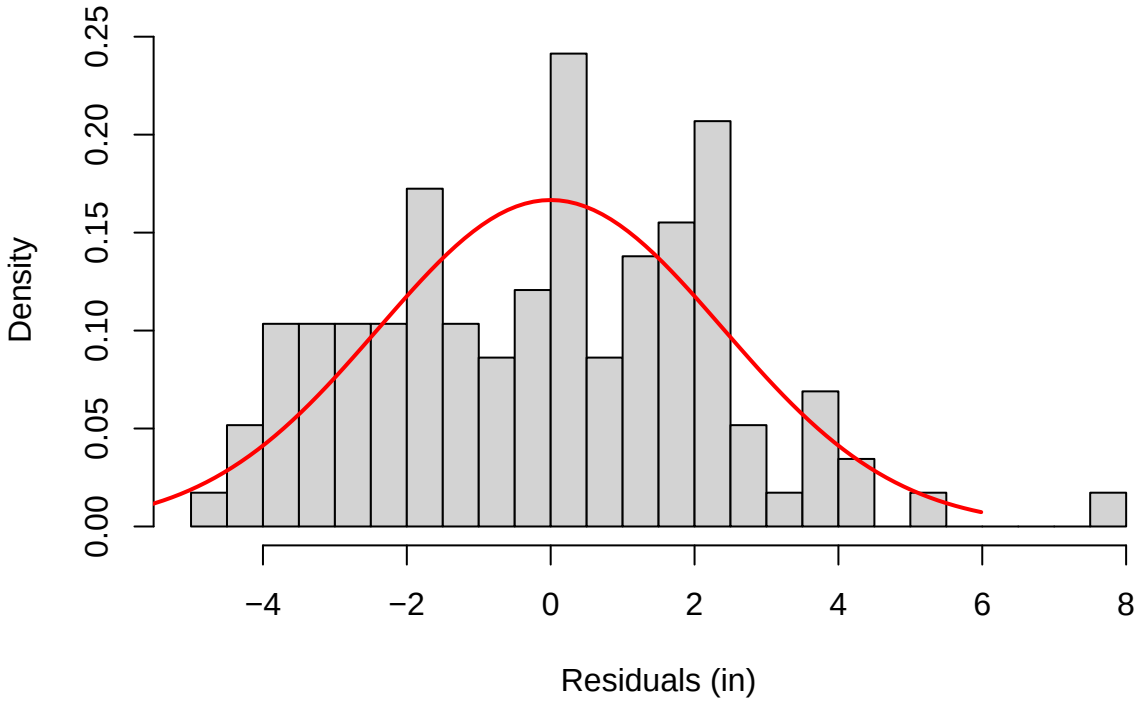
$$S_{YX} = \sqrt{\frac{\sum (Y - \hat{Y})^2}{n-1}}$$

You should recognize the formula for the sums of squares of the residuals in there $SS_Y = \sum (Y - \overline{Y})^2$, so equivalently:

$$S_{YX} = \sqrt{\frac{SS_{YX}}{n-1}}$$

The standard error of the estimate, S_{YX} is 2.4 inches for our example. Here's that same histogram drawn with a normal distribution centered at zero and standard deviation of 2.4 inches.

```
S_YX <- sd(Y-Y_hat)
hist(Y-Y_hat,prob = T,breaks = 20,xlab = 'Residuals (in)',main = NULL)
xnorm <- seq(-2.5*S_YX,2.5*S_YX,length = 101)
ynorm <- dnorm(xnorm,0,S_YX)
lines(xnorm,ynorm, col = "red", lwd = 2)
```

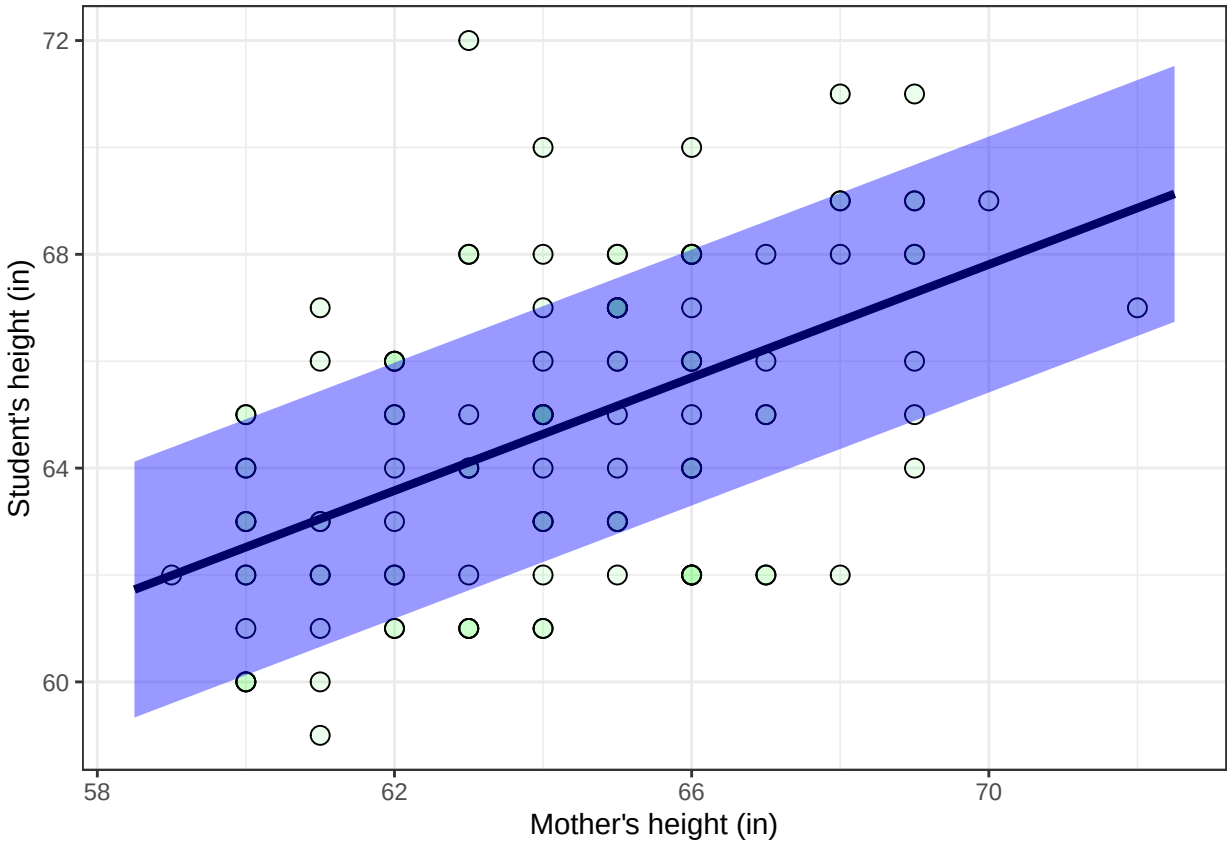


20.6.1 homoscedasticity

Another assumption that we need for regression is 'homoscedasticity', which means that not only are the residuals normally distributed, but also the variance of the distribution does not change with the dependent variable. Graphically, homoscedasticity means that the scatter above and below the regression line normally distributed with a constant standard deviation as you slide along the independent variable (the X-axis).

Here's the scatterplot a shaded region that covers $\pm$ one S_{YX}, or 2.4 inches above and below the regression line.

```
plot_poly <- data.frame(x = regression_data$x[c(1,2,2,1)],
                        y = regression_data$y[c(1,2,2,1)]+ c(S_YX,S_YX,-S_YX,-S_YX))

p +  p_regression_line  +
  geom_polygon(data = plot_poly,aes(x=x,y=y),fill = "blue",alpha = .4)
```



For a normal distribution, about 68% of the data points should fall within ± one standard deviation from the mean. With homoscedasticity, about 68% of the data points should fall within the shaded region in the plot above. It turns out that for this data set, about 85% of the data falls within ± 2.4 inches of the regression line. Not perfect, but pretty close.

20.7 Correlation

The strength of the relation between the student's and mother's heights can be quantified with *Pearson product-moment correlation coefficient*, sensibly referred to as just *correlation*. Though there are other measures of correlation, the Pearson correlation is really the default. The correlation, which ranges between -1 and 1, reflects how well the regression line fits the data. A correlation of -1 means that all the data points fall exactly on a downward sloping regression line, and a correlation of 1 means that all the points fall exactly on an upward sloping regression line. Values in between mean that there's some scatter around the regression line.

The correlation is calculated as:

$$r = \frac{\sum (X - \overline{X})(Y - \overline{Y})}{\sqrt{\sum (X - \overline{X})^2 \sum (Y - \overline{Y})^2}}$$

R has the 'cor' function for this:

```
r <- cor(X,Y)
r
```

```
## [1] 0.5217662
```

Just to show that 'cor' isn't doing something mysterious, here's how to do it by hand in R:

```
sum((X-mean(X))*(Y-mean(Y)))/sqrt(sum((X-mean(X))^2)*sum((Y-mean(Y))^2))
```

```
## [1] 0.5217662
```

20.8 The relationship between r and the slope, β_1

There's a simple relationship between the correlation, r, and the slope of the regression line, β_1:

$$\beta_1 = r\frac{S_Y}{S_X}$$

Where S_Y and S_X are the standard deviations of X and Y. Scaling the X or Y variable will affect the slope of the regression line, β_1. For example, converting the student's heights from inches to feet (dividing by 12) will divide the slope of the regression line by 12. But this conversion does not affect the correlation, r. In the formula above, see how dividing Y by 12 will divide S_Y by 12 which will then divide the slope, β_1 by 12.

20.9 The relationship between r^2 and S_{YX}

You can think of the residuals in the GLM as the variability in the dependent variable that you can't explain by the dependent variables. We therefore call the variance of the residuals, which is the square of the standard error of the estimate: S^2_{YX}, the *variance in Y not explained by X*. If S^2_{YX} is small, then the regression line fits the data well and there's not much variance left over to 'explain'. But what does 'small' mean? A good measure is the ratio of S^2_{YX} to the variance in Y alone, S^2_Y. This ratio has a special relationship to the correlation:

$$1 - r^2 = \frac{S^2_{YX}}{S^2_Y}$$

In words, $1 - r^2$ is the *proportion of the total variance in Y not explained by X*. Turning things around a bit we get:

$$r^2 = 1 - \frac{S^2_{YX}}{S^2_Y}$$

Which means that the square of the correlation, r^2, is the *proportion of variance in Y explained by X* and is a commonly reported number. If we have a 'perfect' correlation of $r = 1$ or $r = -1$, then $r^2 = 1$, and *all* of the variance in Y is explained by X. If there is no correlation ($r = 0$) then *none* of the variance in Y is explained by X.

We can verify the relationship between the correlation S^2_{YX} and S^2_Y with our example in R:

```
1-S_YX^2/var(Y)
```

```
## [1] 0.2722399
```

```
r^2
```

```
## [1] 0.2722399
```

For our example, about 27% of the variance in the student's heights can be explained by their mother's heights. The remaining 73% of the variance in the student's heights must be due to other factors (like their father's heights, random genetic factors, etc.)

20.10 Statistical significance of regression coefficients

Great, so we know that there is a positive regression coefficient, $\beta_1 = 0.53$ that describes how student's heights increase with their mother's heights. But do we really need this regressor to improve the fit? Specifically, does including mother's heights as a factor significantly improve the fit compared to a simpler model? The simpler model, called the 'null' model contains only the intercept:

$$Y = \beta_0 + N(0, \sigma)$$

The best-fitting value of β_0 is the number that minimizes the sums of squared deviations from that value. Remember what that value is? It's the mean! $\overline{Y}$ It's a silly model - it says that no matter what X is, our best guess of Y is the mean of Y. The sums of squares of the the residuals for the null model is $\sum (Y - \overline{Y})^2$, which is SS_Y. To specify the null model using `lm` in R we use `Y ~ 1`:

```
lm_null.out <- lm(Y ~ 1, data = heights)
lm_null.out$coefficients
```

```
## (Intercept)
##       64.75
```

This value is the mean of Y:

```
mean(Y)
```

```
## [1] 64.75
```

The sums of squares of the residuals is:

```
SS_Y <- sum((Y-mean(Y))^2)
SS_Y
```

```
## [1] 905.75
```

Which is the same as

```
sum(lm_null.out$residuals^2)
```

```
## [1] 905.75
```

By adding $\beta_1 X$ to the model, our sums of squared error went down from SS_Y =905.75 to SS_{YX} = 659.17, a difference of 246.58.

The model Y ~ X and Y ~ 1 are called *nested* because one is a simpler version of another. It is implied that the model Y ~ X is really Y ~ 1 + X (try it in R, you'll get the same thing). Two models are nested if one shares a subset of the regressors than the other.

We call the more complicated model the *complete* model and the null model the *restricted* model. Adding a regressor to a model will *always* improve the fit because one option for the new model is to have the coefficient equal to zero, which is the same as the restricted model.

You can run a hypothesis test to see if the complete model has a significantly smaller SS than the null model using a *nested F test*, where:

$$F = \frac{(SS_r - SS_c)/(k_c - k_r)}{SS_c/(n - k_c)}$$

SS_r and SS_c are the sums of squares errors for the restricted and complete models, k_r and k_c are the number of regressors for the two models, and n is the total number of data points. The F statistic has $k_c - k_r$ degrees of freedom in the numerator, and $n - k_c$ degrees of freedom in the denominator.

For our example, there are two regressors, β_0 and β_1 for the complete model, so $k_c = 2$, and one regressor for the restricted model, so $k_r = 1$. There are $n = 116$ data points, so:

$$F = \frac{(SS_Y - SS_{YX})/(2-1)}{SS_{YX}/(n-2)} = \frac{(905.75 - 659.17)/1}{659.17/(116-2)} = 42.65$$

The degrees of freedom are 1 and n-2 = 114. The p-value can be found using the pf function in R:

```
pf_value <- 1-pf(F_stat,1,n-2)
print(sprintf('F(%d,%d) = %5.3f, p = %g',1,n-2,F_stat,pf_value))
```

```
## [1] "F(1,114) = 42.645, p = 1.9006e-09"
```

This is a teeny tiny p-value, which means that adding the mother's height as a regressor significantly improves the prediction of the student's height.

This was just to show you how to do this by hand. R runs the same analysis by passing the output of `lm` into the `anova` function, like we did for ANOVA's in the previous chapters:

```
anova(lm.out)
```

```
## Analysis of Variance Table
##
## Response: student
##            Df Sum Sq Mean Sq F value    Pr(>F)
## mother      1 246.58 246.581  42.645 1.901e-09 ***
## Residuals 114 659.17   5.782
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Without telling you, 'lm' ran the null model on its own and did its own nested F test. You should recognize most of these numbers. The first column are the two degrees of freedom. The second column are sums of squares. The bottom row is the residual sums of squares for the complete model, $S_{YX} = 659.17$. The sums of squares for the row labeled 'X' is the difference in sums of squares between the complete and the restricted model, which we calculated before:

$$SS_Y - SS_{YX} = 905.75 - 659.17 = 246.58$$

The column labeled 'Mean Sq' is the sums of squares for each row divided by their degrees of freedom. And there's our F statistic and corresponding p-value.

20.11 Statistical significance of correlations

Another hypothesis test whether or not the correlation is significantly different from zero. The null hypothesis that there is no correlation in the population that you're drawing from. Formally, the parameter, ρ, is the correlation in the population and the null hypothesis is that $\rho = 0$.

Sir Ronald Fisher worked out that the distribution of correlations drawn from a population with zero correlation can be approximated by a t-distribution using the formula:

$$t = \frac{r}{\sqrt{\frac{1-r^2}{n-2}}}$$

with $df = n - 2$. For our example, using R, we can use pf to get a p-value:

```
n <- length(X)
t <- r/sqrt((1-r^2)/(n-2))
pt_value <- 2*(1-pt(t,n-2)) # 2* for a two-tailed test
print(sprintf('t(%d) = %5.4f, p = %g',n-2,t,pt_value))
```

```
## [1] "t(114) = 6.5303, p = 1.9006e-09"
```

R has it's own function, cor.test for this:

```
out.cor <- cor.test(X,Y)
out.cor
```

```
##
##  Pearson's product-moment correlation
##
## data:  X and Y
## t = 6.5303, df = 114, p-value = 1.901e-09
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.3751349 0.6429236
## sample estimates:
##       cor
```

```
## 0.5217662
```

You should see some familiar numbers there.

Compare this p-value to the nested F-test above. They're exactly the same! We've just demonstrated that these two hypotheses tests are identical. Notice, also, that if you square the t-statistic from the correlation test you get the F statistic from the nested F test: $6.53^2 = 42.645$. Recall that for an ANOVA with two levels, we also found that $t = F^2$.

20.12 Why two identical hypothesis tests?

We've covered two ways of testing the statistical significance of the effect of mother's heights on student's heights. The first was the t-test on the correlation and the second was an F-test on the sums of squared errors.

Why do we have two tests that get the same answer? It's probably because they look at the problem from different perspectives. The t-test on correlations is about testing if the relation between X and Y is significantly strong, and the F-test is about whether adding X to the null model significantly improves the goodness of fit. At least they get the same answer.

20.13 Glossary of terms

	description	formula
SS_X	Sums of squared deviation from the mean	$\sum (Y - \overline{Y})^2$
${S^2}_X$	Variance of Y	$\frac{SS_Y}{n-1}$
S_X	Standard deviation of Y	$\sqrt{{S^2}_Y}$
$\hat{Y}$	Predicted value of Y	$\hat{Y} = \beta_0 + \beta_1 X$
SS_{YX}	Sums of squared residuals	$\sum (Y - \hat{Y})^2$
${S^2}_{YX}$	Variance of Y not explained by X	$\frac{SS_{YX}}{n-1}$
S_{YX}	standard error of the estimate	$\sqrt{{S^2}_{YX}}$

Chapter 21

ANOVA is just regression

You'll hear the title of this chapter from certain statistics instructors and fans of linear regression. Sometimes ANOVA is taught completely in the context of linear regression, and not in the traditional way like we have in this book of ratios of mean squares and F distributions. I think the main difference between the two methods of understanding and teaching ANOVA is cultural; the traditional way emphasizes statistical significance in the differences of the means, and the regression way emphasizes how those means are calculated.

In any case, I've found a good description of why "ANOVA is just regression" lacking. So this is my best shot at it.

21.1 3 Equations and 3 Unknowns

First we'll start with a crash course on linear regression. We begin with basic matrix multiplication. For example, here's how you compute the product of a 3x3 matrix with a 3x1 column vector. We'll call the 3x3 matrix A, and the 3x1 column vector x and the product y. This is how to multiply A times x to get y:

A x y

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} (1)(0)+(1)(1)+(1)(2) \\ (1)(0)+(2)(1)+(3)(2) \\ (1)(0)+(0)(1)+(1)(2) \end{bmatrix} = \begin{bmatrix} 3 \\ 8 \\ 2 \end{bmatrix}$$

If it has been a while, take a look at how this works. When multiplying a matrix times a vector we perform dot products across the rows of the matrix A with the column of vector x.

Another way to think about it is that the product of A and x is equal to the sum of the columns of A scaled by the values in the vector x. In this example, the result is 0 times the first column plus 1 times the second column plus 2 times the third column of A. In other words the product of matrix A and a column vector x is a *linear combination* of the columns of A weighted by the values in x.

Suppose that you know the values in the matrix A and the answer y, but not the values in the vector x. If this were algebra with numbers, we'd just solve for x by dividing both sides of the equation by A. That is, if

$$Ax = y$$

then

$$x = \frac{y}{A}$$

But since A is a matrix, instead of dividing by A we compute the *inverse* of A which we then multiply on both sides of the equation $Ax = y$. The inverse of A, written as A^{-1} is the matrix for which $A^{-1}A = I$, where I is the identity matrix (a matrix with all zeros with 1's along the diagonal). Remember, I has as the property that $Ix = x$.

Multiplying A^{-1} on both sides of $Ax = y$ gives:

$$A^{-1}Ax = A^{-1}y$$

$$Ix = A^{-1}y$$

$$x = A^{-1}y$$

For our example, the inverse of A is:

A^{-1}

$$\begin{bmatrix} 1 & -0.5 & 0.5 \\ 1 & 0 & -1 \\ -1 & 0.5 & 0.5 \end{bmatrix}$$

You can verify that $A^{-1}A = I$:

A^{-1} A I

$$\begin{bmatrix} 1 & -0.5 & 0.5 \\ 1 & 0 & -1 \\ -1 & 0.5 & 0.5 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

So, if we multiply A^{-1} by y we get our original value of x:

A^{-1} y x

$$\begin{bmatrix} 1 & -0.5 & 0.5 \\ 1 & 0 & -1 \\ -1 & 0.5 & 0.5 \end{bmatrix} \times \begin{bmatrix} 3 \\ 8 \\ 2 \end{bmatrix} = \begin{bmatrix} (1)(3) + (-0.5)(8) + (0.5)(2) \\ (1)(3) + (0)(8) + (-1)(2) \\ (-1)(3) + (0.5)(8) + (0.5)(2) \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix}$$

This shows that you can use A^{-1} to solve $Ax = y$ for x.

Where did A^{-1} come from? You may have computed a matrix inverse in the past using methods like the *Gauss-Jordan method* or through numerical methods, which is how your computer does it.

21.2 5 Equations and 2 Unknowns

Now consider the case where A is not square, but has more rows than columns. For example, consider this equation for $Ax = y$:

A x y

$$\begin{bmatrix} 1 & 0 \\ 2 & 1 \\ 3 & 0 \\ 4 & 1 \\ 5 & 0 \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \\ 1.5 \\ 1.4 \\ 2.5 \end{bmatrix}$$

To find the values x_1 and x_2 we need to satisfy 5 equations with only 2 unknowns. This is an *overconstrained* set of equations and may not have an exact solution. Indeed, for this example there is no set of values x_1 and x_2 that satisfy this equation.

The best we can do is find values of $\hat{x}_1$ and $\hat{x}_2$ so that the product $A\hat{x}$ is close to the correct answer (we pronounce $\hat{x}$ as 'x-hat'). By close, we mean the answer that minimizes the sums of squared difference between $A\hat{x} = \hat{y}$ and y. This is called the *least squares* solution to the problem. The process of finding these best values is called *linear regression*.

Matrix notation is really just shorthand for writing out systems of equations. For this example, linear regression is the process of finding the values of $\hat{x_1}$ and $\hat{x_2}$ that make the following sum as small as possible:

$(1\hat{x_1} + 0\hat{x_2} - 0.5)^2 + (2\hat{x_1} + 1\hat{x_2} - 0.5)^2 + (3\hat{x_1} + 0\hat{x_2} - 1.5)^2 + (4\hat{x_1} + 1\hat{x_2} - 1.4)^2 + (5\hat{x_1} + 0\hat{x_2} - 2.5)^2$

In the first example, when A was a 3x3 matrix, we solved the equation by computing the inverse of A, A^{-1}. In this second example A is not a square matrix, so it doesn't have an inverse. Instead, there's something called a *pseudoinverse* (sometimes called the *Moore-Penrose generalized inverse*) that can be used to find the least squared solution. Like the real inverse, the pseudoinverse, denoted as A^+, also has the property that $A^+A = I$.

A^+ is calculated by:

$$A^+ = (A^T A)^{-1} A^T$$

where A^T is the transpose of A (the matrix A with rows and columns switched).

Calculating A^+ would be a nightmare by hand, but there are algorithms that allow computers to find it. For our example,

A^+ is:

A^+

$$\begin{bmatrix} 0.027 & -0.027 & 0.0811 & 0.027 & 0.1351 \\ -0.0811 & 0.5811 & -0.2432 & 0.4189 & -0.4054 \end{bmatrix}$$

Analogous to what we did when A was square, can use the pseudoinverse of A to find the least squared solution, $\hat{x}$, by multiplying A^+ on the left side both sides of the equation $A\hat{x} = y$:

$$A^+ A\hat{x} = A^+ y$$

Since $A^+A = I$,

$$\hat{x} = A^+ y$$

For our example:

$$\begin{bmatrix} 0.027 & -0.027 & 0.0811 & 0.027 & 0.1351 \\ -0.0811 & 0.5811 & -0.2432 & 0.4189 & -0.4054 \end{bmatrix} \times \begin{bmatrix} 0.5 \\ 0.5 \\ 1.5 \\ 1.4 \\ 2.5 \end{bmatrix} = \begin{bmatrix} 0.4973 \\ -0.5419 \end{bmatrix}$$

So, if $\hat{x}_1 = 0.4973$ and $\hat{x}_2 = -0.5419$, then the product $A\hat{x}$ gives us a vector, which we call $\hat{y}$, that is as close to y (in terms of least squares) as possible. For our example, $A\hat{x} = \hat{y}$:

A $\hat{x}$ $\hat{y}$

$$\begin{bmatrix} 1 & 0 \\ 2 & 1 \\ 3 & 0 \\ 4 & 1 \\ 5 & 0 \end{bmatrix} \times \begin{bmatrix} 0.4973 \\ -0.5419 \end{bmatrix} = \begin{bmatrix} 0.4973 \\ 0.4527 \\ 1.4919 \\ 1.4473 \\ 2.4865 \end{bmatrix}$$

Notice that the values in $\hat{y}$ are pretty close to the desired values of y:

y

$$\begin{bmatrix} 0.5 \\ 0.5 \\ 1.5 \\ 1.4 \\ 2.5 \end{bmatrix}$$

How close? In statistics we like to measure differences as sums of squared deviations. The sums of squared difference between y and $\hat{y}$ is:

$$(0.5 - 0.4973)^2 + (0.5 - 0.4527)^2 + ... + (2.5 - 2.4865)^2 = 0.00473$$

You can search as long and hard as you want, but you won't find a better pair of values than $x_1 = 0.4973$ and $x_2 = -0.5419$ that will give you an answer that gives you a smaller sums of squared deviation than 0.00473.

If you think of the matrix multiplication in Ax as computing a linear combination of the columns of A weighted by x, then linear regression is finding the 'weights' needed for each column of A that gives us an answer that is closest to y. Each column of A is called a *regressor*, and the values of x that give you the least-squares solution are called the *regression coeficients* (sometimes called *beta weights* for some reason, though the term *beta weight* should only be used for 'standardized' linear regression).

21.3 Predicting student's heights from mother's heights

When we think of linear regression we often think of fitting straight lines to a scatterplot like we did in the last chapter on correlation and regression. We'll go through a similar example from that chapter on the heights of women and their mothers, but in the context of linear algebra and the pseudoinverse described above. From the survey, I've chosen women in the class that were born in 1999 to keep the sample size smallish.

Here's some standard R code to pull out the student's heights into `student` and mother's heights into `mother`:

```
survey <-
  read.csv("http://www.courses.washington.edu/psy315/datasets/Psych315W21survey.csv")

survey$mheight <- as.numeric(survey$mheight)
id <- survey$year == "1999" & survey$gender== "Woman" &
  !is.na(survey$height) & !is.na(survey$mheight) & !is.na(survey$year)

student <- survey$height[id]
mother <- survey$mheight[id]
```

Here's a table of their heights:

Table 21.1:

student	mother
66	62
62	60
67	65
62	66
62	60
62	64
64	63
61	65
62	61
61	60
63	61
62	66
62	61
67	66
66	62
68	68
65	60
64	69
66	66
65	66
67	72
61	63
68	63
64	60
67	64
67	65
65	62
68	65
63	65
66	69
65	67
66	65
63	64

Suppose you want to come up with a model that predicts each student's height based on their mother's height. A simple model would be a *linear* model like this:

$$student = x_1 + x_2 * mother$$

Where x_1 and x_2 are constants, or 'free parameters'.

We can write this linear model as a matrix multiplication like this:

$A \quad x \quad student$

$$\begin{bmatrix} 1 & 62 \\ 1 & 60 \\ 1 & 65 \\ 1 & 66 \\ 1 & 60 \\ 1 & 64 \\ 1 & 63 \\ 1 & 65 \\ 1 & 61 \\ 1 & 60 \\ 1 & 61 \\ 1 & 66 \\ 1 & 61 \\ 1 & 66 \\ 1 & 62 \\ 1 & 68 \\ 1 & 60 \\ 1 & 69 \\ 1 & 66 \\ 1 & 66 \\ 1 & 72 \\ 1 & 63 \\ 1 & 63 \\ 1 & 60 \\ 1 & 64 \\ 1 & 65 \\ 1 & 62 \\ 1 & 65 \\ 1 & 65 \\ 1 & 69 \\ 1 & 67 \\ 1 & 65 \\ 1 & 64 \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 66 \\ 62 \\ 67 \\ 62 \\ 62 \\ 62 \\ 64 \\ 61 \\ 62 \\ 61 \\ 63 \\ 62 \\ 62 \\ 67 \\ 66 \\ 68 \\ 65 \\ 64 \\ 66 \\ 65 \\ 67 \\ 61 \\ 68 \\ 64 \\ 67 \\ 67 \\ 65 \\ 68 \\ 63 \\ 66 \\ 65 \\ 66 \\ 63 \end{bmatrix}$$

Here the first column of A is all 1's and the second column contains mother's heights. See how this matrix multiplication performs the calculation in the linear model. For each student,

$$x_1 + mother \times x_2 = student$$

Notice how this is a set of 33 (# of students) and 2 unknowns. We can use linear regression to find the values of x_1 and x_2 that minimize the sum of squares between the predicted heights and real student heights.

We can do this with the pseudoinverse, A^+, like we did before and then calculating $\hat{x} = (A^+)(mother)$. If you get your computer to do this you get these regression weights:

$\hat{x}$

$$\begin{bmatrix} 43.9715 \\ 0.3196 \end{bmatrix}$$

The equation that best predicts student heights is therefore: $student = 43.9715 + 0.3196 \times mother$

With this best fit we can generate the predicted student heights, $\hat{y}$ by multiplying A by $\hat{x}$:

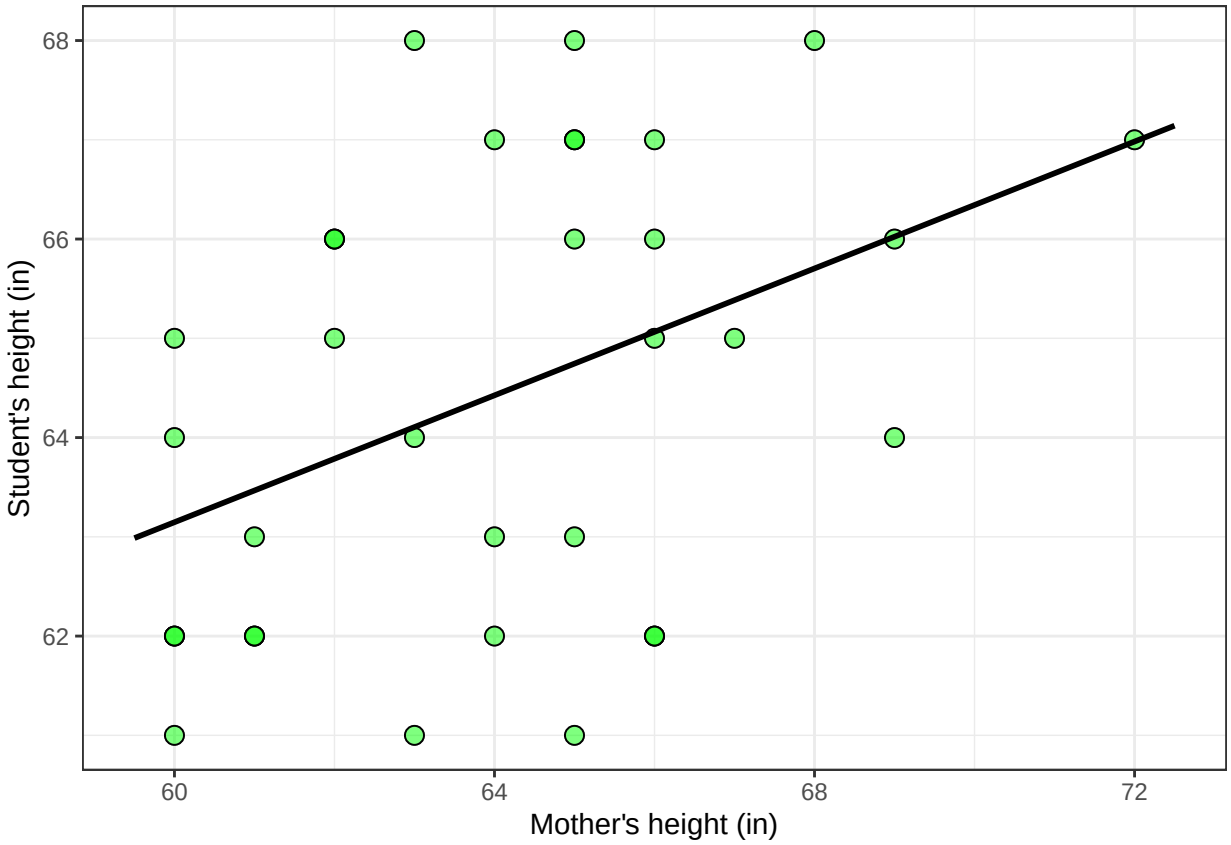
$A \qquad \hat{x} \qquad \hat{y}$

$$\begin{bmatrix} 1 & 62 \\ 1 & 60 \\ 1 & 65 \\ 1 & 66 \\ 1 & 60 \\ 1 & 64 \\ 1 & 63 \\ 1 & 65 \\ 1 & 61 \\ 1 & 60 \\ 1 & 61 \\ 1 & 66 \\ 1 & 61 \\ 1 & 66 \\ 1 & 62 \\ 1 & 68 \\ 1 & 60 \\ 1 & 69 \\ 1 & 66 \\ 1 & 66 \\ 1 & 72 \\ 1 & 63 \\ 1 & 63 \\ 1 & 60 \\ 1 & 64 \\ 1 & 65 \\ 1 & 62 \\ 1 & 65 \\ 1 & 65 \\ 1 & 69 \\ 1 & 67 \\ 1 & 65 \\ 1 & 64 \end{bmatrix} \times \begin{bmatrix} 43.9715 \\ 0.3196 \end{bmatrix} = \begin{bmatrix} 63.7863 \\ 63.1471 \\ 64.7451 \\ 65.0647 \\ 63.1471 \\ 64.4255 \\ 64.1059 \\ 64.7451 \\ 63.4667 \\ 63.1471 \\ 63.4667 \\ 65.0647 \\ 63.4667 \\ 65.0647 \\ 63.7863 \\ 65.7039 \\ 63.1471 \\ 66.0235 \\ 65.0647 \\ 65.0647 \\ 66.9822 \\ 64.1059 \\ 64.1059 \\ 63.1471 \\ 64.4255 \\ 64.7451 \\ 63.7863 \\ 64.7451 \\ 64.7451 \\ 66.0235 \\ 65.3843 \\ 64.7451 \\ 64.4255 \end{bmatrix}$$

The sums of squared deviation between the student's heights and predicted student's heights is

$$(66 - 63.7863)^2 + (62 - 63.1471)^2 + ... + (63 - 64.4255)^2 = 0.0047297$$

You'll recognize that our model is simply the equation of a line. Given a scatterplot with mother's heights on the x-axis and student height on the y-axis, solving for the regression weights finds the best-fitting line to the data in the scatterplot.



21.4 Statistical significance of adding regressors

Solving a linear regression problem will always give us a least-squares solution, even if the fit isn't very good. How can we evaluate the goodness of fit? More specifically, how can we determine if we need all of the regressors to predict the data?

One way to think about the fit of our regression line is to ask if if we really need a slope to predict the student's heights. Or in hypothesis test terms, is the slope significantly different from zero?

Let's see how well we can predict the student's heights with a simpler model that doesn't have a regressor for the slope. This is just finding the least square solution for $Ax = y$ where A is just a column of ones:

A x *student*

$$\begin{bmatrix}1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\end{bmatrix} \times \begin{bmatrix}x_1\end{bmatrix} = \begin{bmatrix}66\\62\\67\\62\\62\\62\\64\\61\\62\\61\\63\\62\\62\\67\\66\\68\\65\\64\\66\\65\\67\\61\\68\\64\\67\\67\\65\\68\\63\\66\\65\\66\\63\end{bmatrix}$$

The least-square solution to this is $x_1 = 64.4545$.

$$(66 - 64.4545)^2 + (62 - 64.4545)^2 + ... + (63 - 64.4545)^2 = 168.1818$$

Compare this to the sums of squared deviation for the regression with the slope: 138.8954.

The regression equation with the slope has a smaller sums of squared deviation. Does that mean it's a better predictor of the student's heights? Not necessarily. Adding regressors will always improve the fit, since the equation with the slope is a more flexible version of the equation without the slope. We call these two regression equations *nested*, with one a more general version of the other.

Since adding regressors always improves the fit, we need a hypothesis test to determine if the additional regressor improves the fit enough to justify it. This can be done by computing an F statistic and the corresponding p-value.

We define the model with more regressors as the 'complete' model,and the model with fewer regressors as the 'restricted' model. For our example, the complete model is the one with both an intercept and a slope, and the restricted model has just the one constant. The sums of squared deviation for the least-square fit for the two models are defined as SS_r and SS_c for the restricted and complete models respectively. For this example,

$$SS_r = 168.1818$$

and

$$SS_c = 138.8954$$

Let n be the total number of data points ($n = 33$ students in our example), k_r be the number of regressors in the restricted model ($k_r = 1$, the mean) and k_c be the number of regressors in the complete model ($k_c = 2$, intercept and slope).

The F-statistic is computed as:

$$F = \frac{(SS_r - SS_c)/(k_c - k_r)}{SS_c/(n - k_c)}$$

With $k_c - k_r$ degrees of freedom for the numerator and $n - k_c$ degrees of freedom for the denominator.

For our example,

$$F = \frac{(168.1818 - 138.8954)/(2 - 1)}{138.8954/(33 - 2)} = 6.5364$$

The p-value for this value of F, with 1 and 31 degrees of freedom is:

```
1-pf(6.5364,1,31)
```

```
## [1] 0.01568644
```

A small p-value indicates that adding the slope regressor significantly improves the fit of the linear regression model to the data. In other words, if p is small, then the slope is significantly different from zero, because if the slope is zero we have the restricted model.

If we use our favorite criterion value of $\alpha = 0.05$, we conclude that adding the slope regressor does significantly improve the fit. We therefore conclude that the slope is significantly different from zero. If we assume causality, it means that there is a significant influence of the mother's height on the student's height.

Recall that in the previous chapter we showed that if you use R's `cor.test` function to run a two-tailed test on the null hypothesis that the correlation in the population between student and mother's heights is zero you'll get the same p-value.

21.5 ANOVA as regression

We're ready to talk about ANOVA. We'll first show that given a 1-factor ANOVA data set you can calculate group means using linear regression. Consider the following data set which are UW GPA's for the men in the class grouped by whether they prefer to sit in the front, middle, or the back of the lecture hall. We can use a 1-factor ANOVA to determine if the means of each group are significantly different from each other.

Here are the summary statistics showing the group means, SS_{within} for each of the 3 groups:

And here's the result of the 'omnibus' F test:

The p-value for this hypothesis test is 0.2633, so using our magic value of $\alpha = 0.05$, it seems that GPA does not vary with where these students prefer to sit in class.

Just in case you're wondering, if we had included the whole class of 150, there is a significant difference in GPA at UW across where they like to sit.

Next we'll use regression to estimate the 3 means. Here we'll write a new equation $Ax = y$ where y is a column vector containing all of the GPA's. For this example, the first 3 values in y contain the GPAs in the 'Near the front' group. The second 15 values are GPA's from the second group and so on. We'll generate a funny looking matrix for A which contain zeros and ones like this:

A x y

Table 21.2:

Near the front	In the middle	Toward the back
3.66	3.23	3.30
3.84	3.65	3.68
3.95	3.80	3.83
	3.90	2.60
	3.43	3.80
	3.85	2.00
	3.00	2.89
	3.35	4.00
	3.31	3.66
	3.91	3.51
	3.07	
	3.53	
	3.20	
	3.30	
	3.65	

Table 21.3:

	mean	n	sd	sem
Near the front	3.8167	3	0.1464	0.0845
In the middle	3.4787	15	0.3025	0.0781
Toward the back	3.3270	10	0.6394	0.2022

Table 21.4:

	df	SS	MS	F	p
Between	2	0.5637	0.2819	1.4083	$p = 0.2633$
Within	25	5.0037	0.2001		
Total	27	5.5674			

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3.66 \\ 3.84 \\ 3.95 \\ 3.23 \\ 3.65 \\ 3.8 \\ 3.9 \\ 3.43 \\ 3.85 \\ 3 \\ 3.35 \\ 3.31 \\ 3.91 \\ 3.07 \\ 3.53 \\ 3.2 \\ 3.3 \\ 3.65 \\ 3.3 \\ 3.68 \\ 3.83 \\ 2.6 \\ 3.8 \\ 2 \\ 2.89 \\ 4 \\ 3.66 \\ 3.51 \end{bmatrix}$$

Think about what happens when A is multiplied by x to predict y. Notice that the first 3 predicted values are all equal to x_1: $(1x_1 + 0x_2 + 0x_2)$. What single value of x_1 is closest, in the least-squares sense, to all of the first 3 GPA's? It's the mean of those 3 GPA's from the 'Near the front' group. Similarly, the best-fitting value of x_2 will be the mean of the 'In the middle' group and x_3 will be the mean of the 'Toward the back' group. Sure enough, if we use the pseudoinverse to calculate the regression weights, we get our three group means:

$\hat{x}$

$$\begin{bmatrix} 3.8167 \\ 3.4787 \\ 3.327 \end{bmatrix}$$

The predicted GPA's based on these regression weights are just multiple copies of the group means:

A $\hat{x}$ $\hat{y}$

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 3.8167 \\ 3.4787 \\ 3.327 \end{bmatrix} = \begin{bmatrix} 3.8167 \\ 3.8167 \\ 3.8167 \\ 3.4787 \\ 3.4787 \\ 3.4787 \\ 3.4787 \\ 3.4787 \\ 3.4787 \\ 3.4787 \\ 3.4787 \\ 3.4787 \\ 3.4787 \\ 3.4787 \\ 3.4787 \\ 3.4787 \\ 3.4787 \\ 3.4787 \\ 3.4787 \\ 3.4787 \\ 3.327 \\ 3.327 \\ 3.327 \\ 3.327 \\ 3.327 \\ 3.327 \\ 3.327 \\ 3.327 \\ 3.327 \\ 3.327 \\ 3.327 \end{bmatrix}$$

The sums of squared deviations from $\hat{y}$ and y is 5.0036. Can you find this number in the ANOVA table above? It's the same as SS_{within}. That's because SS_{within} is the sums of squared deviation of each GPA from the mean of each group.

So far, this is just a complicated way of calculated the group means. What about the statistical significance of the test? The trick is to think of this regression model as the 'complete' model, which has a regressor for each group mean.

We can compare the fit to this model to a 'restricted' model that does not allow for a different mean for each group. Instead, we'll try to predict our GPA's from just one mean. Here's the restricted model:

$A \qquad x \qquad y$

$$\begin{bmatrix}1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\end{bmatrix} \times \begin{bmatrix}x_1\end{bmatrix} = \begin{bmatrix}3.66\\3.84\\3.95\\3.23\\3.65\\3.8\\3.9\\3.43\\3.85\\3\\3.35\\3.31\\3.91\\3.07\\3.53\\3.2\\3.3\\3.65\\3.3\\3.68\\3.83\\2.6\\3.8\\2\\2.89\\4\\3.66\\3.51\end{bmatrix}$$

Here's the least-squared solution:

$\hat{x}$

$$\begin{bmatrix}3.4607\end{bmatrix}$$

Can you guess what this number 3.4607 is? It's the grand mean.

The sums of squared deviation for this restricted model is 5.5674. Can you find this value in the ANOVA summary table? It's SS_{total}, which is the sums of squared deviation of all GPA's from the grand mean.

You may have guessed where we're going. A nested F-test will determine if the improvement of fit with the complete model justifies including a separate mean for each group. In other words, do we really need separate means to predict the data, or can we only conclude that the data came from a single population with a fixed mean?

For a nested F-test, the F-statistic is:

$$F = \frac{(SS_r - SS_c)/(k_c - k_r)}{SS_c/(n - k_c)}$$

With $k_c - k_r$ degrees of freedom for the numerator and k_c degrees of freedom for the denominator.

For this specific example, let's define k to be the number of groups. We can replace the terms in the equations with:

$$SS_C = SS_{within}$$

$$SS_R = SS_{total}$$

$$k_c = k$$

$$k_r = 1$$

And, for this example,

$$SS_r - SS_c = SS_{total} - SS_{within}$$

Remember,

$$SS_{total} - SS_{within} = SS_{between}$$

So we can re-write the F-statistic as:

$$F = \frac{SS_{between}/(k-1)}{SS_{within}/(n-k)}$$

With $k-1$ degrees of freedom for the numerator and $n-k$ degrees of freedom for the denominator.

This is exactly the formula for computing the F statistic and degrees of freedom for the 1-factor ANOVA. So we get the same F-statistic as the original ANOVA analysis:

$$F = \frac{(5.5674 - 5.0036)/(3-1)}{5.0036/(28-3)} = 1.4083$$

Which, with 2 and 25 degrees of freedom gives us a p-value of 0.2633, the same as in the 1-factor ANOVA.

In terms of the nested model, we conclude that including separate regressors for each group to predict group means does not significantly improve the fit of the regression model over the restricted model that just has a single mean.

One caveat. For this example, our complete model is not really a more general version of the restricted model because it's not the restricted model with added regressors (there is no column of 1's in the complete model). However, you can run a different version of the complete model, where the first column is all ones, you get a general version of the restricted model that fits exactly as well:

A x y

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3.66 \\ 3.84 \\ 3.95 \\ 3.23 \\ 3.65 \\ 3.8 \\ 3.9 \\ 3.43 \\ 3.85 \\ 3 \\ 3.35 \\ 3.31 \\ 3.91 \\ 3.07 \\ 3.53 \\ 3.2 \\ 3.3 \\ 3.65 \\ 3.3 \\ 3.68 \\ 3.83 \\ 2.6 \\ 3.8 \\ 2 \\ 2.89 \\ 4 \\ 3.66 \\ 3.51 \end{bmatrix}$$

The regression weights for this complete model are:

$\hat{x}$

$$\begin{bmatrix} 3.8167 \\ -0.338 \\ -0.4897 \end{bmatrix}$$

The sums of squared deviation for this regression is the same as for the previous complete model: 5.0036.

But what do these regression weights mean? The first weight, 3.8167 is still the mean for the first, 'Near the front', group of GPAs.

If you look at the matrix A you'll see that the predicted GPA for the second group is the sum of the first two regression weights. So the predicted GPA for the second group is the mean for the first group plus $\hat{x}_2$. Check that $\hat{x}_1 + \hat{x}_2 = 3.8167 - 0.338 = 3.4787$, which is the mean for the second group. Similarly, $\hat{x}_1 + \hat{x}_3 = 3.8167 - 0.4897 = 3.327$, which is the mean for the third group.

This new complete model has the same information in the regression weights, and since it's a true generalization of the restricted model, it's technically the one that should be used for the nested model F test. But since the SS and df's are the same, it will give you the same F statistic as for the first, more intuitive, complete model that gave us the group means.

In summary, when someone tells you that 'ANOVA is just regression', you should now understand that you can use regression to find group means using a least squares to to $Ax = y$ with a clever A matrix of zeroes and ones. This trick of using regressors consisting of zeros and ones to turn ANOVA into regression has many names, including 'dummy variable', 'indicator variables','design variable','Boolean indicator', 'binary variable', or 'qualitative variable'.

Using a nested F-test to compare the goodness of fit of this complete model with the fit to restricted model provides the same measure of statistical significance as the traditional ANOVA. In fact, most software packages use this sort

of dummy coding and regression to calculate F statistics for ANOVA. More complicated ANOVA tests, like main effects, interactions, simple effects, and random effects just require more complicated dummy variables that the computer figures out for you.

21.6 Effect Size

Here I'll show you that η^2, one of three measures of effect size for ANOVA, is the square of the correlation between the predicted and real data.

η^2 is defined as:

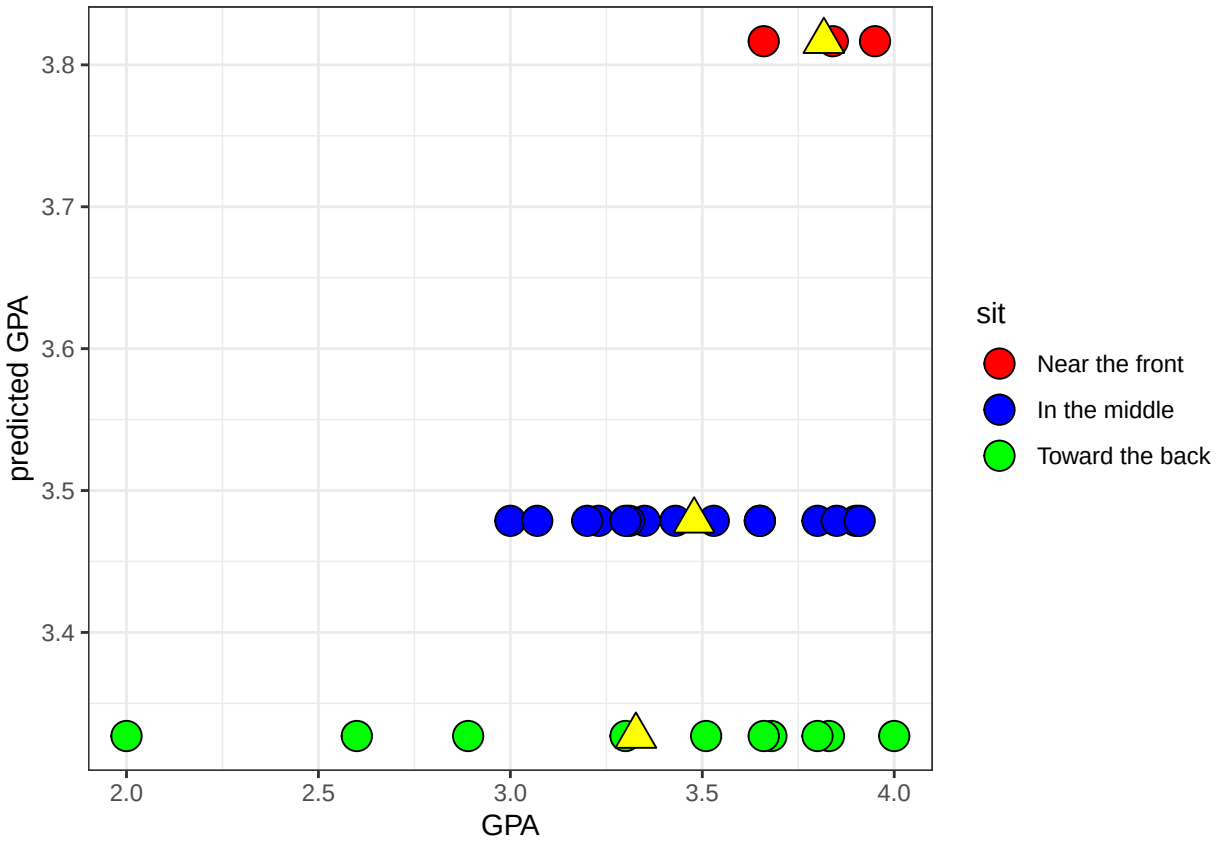
$$\eta^2 = \frac{SS_{between}}{SS_{total}}$$

which is the proportion of the total sums of squares that is associated with the differences between the means.

For our example:

$$\eta^2 = \frac{SS_{between}}{SS_{total}} = \frac{0.5637}{5.5674} = 0.1013$$

There's another interpretation of η^2 with respect to regression. Here's a scatterplot showing the real GPA's on the x-axis and the predicted GPA's from the complete model on the y-axis:



The scatterplot looks a little funny because there are only three levels of predicted GPA's. That's because, of course, the model predicts a specific GPA for each of the three levels of where students like to sit.

I've labeled the names of each group at the height of their means on the right side of the y-axis, and I've placed a yellow triangle at the location of the mean for each group.

The scatterplot lets you visualize each data point within each of the three groups. See how this scatterplot is a nice graphical representation of how the total variability of the points along the x-axis, SS_{total}, is broken down into SS_{within} and $SS_{between}$. SS_{within} is seen as the spread of points along the x-axis within each of the three groups (yellow triangles). $SS_{between}$ is associated with the differences in the three means, which are the three values along the y-axis.

Consider the correlation between the real and predicted GPAs. If there were no variability between the means, all the predicted values would be the same on the y-axis and the correlation would be zero. On the other hand, if there was no variability within each group, then all GPA's would be equal to the mean GPA for that group, so the points would all fall on a line defined by the three red boxes. The correlation would be equal to 1.

For this example, the correlation $r = 0.3182$ and its square is $r^2 = 0.1013$.

Note that $r^2 = \eta^2$

So another interpretation of the effect size measure, η^2, is that it's the square of the correlation between the predicted and real data. This is sometimes called the 'proportion of the variance in X *explained* by Y'. In our case, the proportion of the variance in GPA's *explained* by where students choose to sit is 0.1013.

Chapter 22

Linear Models

This chapter follows the chapter on ANOVA and regression which discusses the relationship between the traditional ANOVA and using regression to find the same results. This chapter takes the next step and shows how to use R's `lm` function to analyze results from ANOVA style experimental designs. We've already used the `lm` function in the previous chapters on 1 factor ANOVA and 2 factor ANOVA, but we always passed straight into the `anova` function to get our SS's MS's df's and p-values. In this chapter we'll explore the output of `lm` before it's passed into `anova`. We'll also show how `lm` can deal with unbalanced 2-factor designs with a discussion about Type I and Type II ANOVAs.

Here we focus on fixed effect factors. The next chapter, Linear Mixed Models, covers random effect factors including repeated measures designs.

22.1 One factor ANOVA example

We'll learn from examples, starting with the example from the 1 factor ANOVA chapter where we studied how GPA's at UW depended on where student's like to sit in the classroom. That chapter restricted the data to the students that identified as men, just to keep the data set small enough to show all of the numbers in tables. This time we'll include the whole class.

This is a classic one-factor ANOVA design, with sitting preference being the categorical independent variable, and GPA as the ratio-scale dependent variable.

First, we'll load in the survey data. I'm also going to set the levels of 'sit' to my chosen order, rather than the default alphabetical order:

```
survey <-
  read.csv("http://www.courses.washington.edu/psy315/datasets/Psych315W21survey.csv")
survey <- subset(survey,gender == "Man" | gender == "Woman")
survey$sit <- factor(survey$sit,levels = c('Near the front','In the middle','Toward the back'))
```

The independent variable is the field 'sit'. Let's see where people like to sit:

```
table(survey$sit)
```

```
##
##  Near the front   In the middle Toward the back
##              47              72              32
```

It looks like a lot of students prefer to sit near the front of the classroom. Here is the average UW GPA for each group, using the `tapply` function:

```
m1 <- tapply(survey$GPA_UW,survey$sit,mean,na.rm = TRUE)
m1
```

```
##  Near the front   In the middle Toward the back
```

```
##        3.539362        3.515857        3.346875
```

The GPA's differ, of course. But do they differ by a significant amount? We'll use R's `lm` function to test the model that we need three different means, one for each group, to fit the data.

```
lm1.out <- lm(GPA_UW ~ sit, data = survey)
```

Remember from the chapter, ANOVA is just regression, the first argument in to the `lm` function defines the model we're trying to fit. Here, 'GPA_UW ~ sit' means that we're predicting the dependent variable, GPA_UW, from the levels in the independent variable, 'sit'.

22.1.1 Warning about using numerical names for nominal data

In the chapter Correlation and Regression we used `lm` to predict student's heights from mother's heights, where the independent variable, mother's heights, is continuous. R's `lm` function is very flexible. If `lm` recognizes that the independent variable is nominal, then it switches to ANOVA mode. But be careful - if you labeled your levels with numbers for, 1,2 and 3 instead of Near the front, In the middle, and Toward the back then `lm` will treat those three levels as ratio scale values, like heights, so that In the middle will be considered twice the value as `level [1]`. You'll get a nice looking table with p-values and you might not notice that you fit your data with the wrong model.

You can fix this by setting the class to your independent level as 'factor' like we did above. But I suggest that you avoid this scary problem by labeling your levels with names (like 'S1','S2',... for subjects) instead of numbers (1, 2,...).

When the independent variable is nominal, `lm` knows to run an ANOVA, so it produces that matrix of ones and zeros that we discussed in ANOVA is just regression. In that matrix, called a *design matrix* is a column for each level of 'sit'. Each column is a *regressor* for that level, and `lm` finds the best *coefficient* (also called a *regression weight* or *beta weight*) for that regressor. For ANOVA, those coefficients are the means (or differences of means) for the GPA's within each level.

22.1.2 Interpreting coefficients

The result, lm1.out, is an 'object'. This means that it's not a number like a p-value or a table, but instead something that can be operated on when passed into other functions. For example, if we want to see the regression weights (or 'beta' weights) for this fit to the data we can pass lm1.out into R's `coef` function:

```
coef(lm1.out)
```

```
##        (Intercept)   sitIn the middle sitToward the back
##         3.53936170        -0.02350456        -0.19248670
```

These coefficients don't look quite like the three mean GPA's that we calculated above. Recall from the ANOVA is just regression chapter that the way ANOVA is done with regression is to let the first weight (the Intercept) be the mean for the first group and the next weights be the differences from this group. Here the intercept is 3.5393617, which is the mean GPA of the "Near the front" group.

The mean for the second group, "In the middle" is simply the intercept plus the second coefficient: (3.5393617 + -0.0235046 = 3.5158571). You can verify that adding the third coefficient to the intercept gives you the mean GPA for the "Toward the back" " group.

22.1.3 predictions

It's helpful to think of linear regression as a method of predicting your dependent variable, rather than a way to calculate statistical significance. For a one-factor ANOVA, the predicted GPA's are the means for each group. R will calculate these predicted GPAs (means) if you pass the output of `lm` into the `predict` function:

```
pred1 <- predict(lm1.out)
```

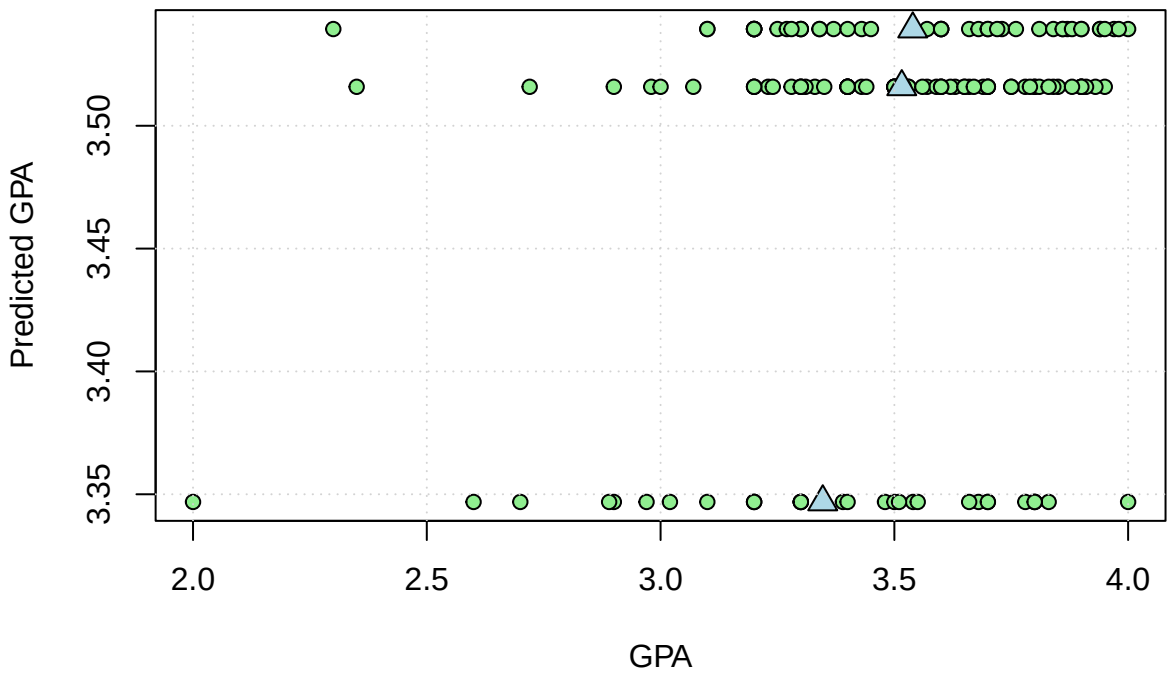
Here's a table of the actual and predicted GPA's.

Table 22.1:

sit	GPA	prediction
In the middle	3.23	3.5159
In the middle	3.20	3.5159
In the middle	3.50	3.5159
Near the front	2.30	3.5394
In the middle	3.30	3.5159
In the middle	3.65	3.5159
Near the front	3.30	3.5394
In the middle	3.90	3.5159
Toward the back	3.20	3.3469
In the middle	3.95	3.5159
In the middle	3.40	3.5159
In the middle	3.80	3.5159
Toward the back	2.97	3.3469
Near the front	3.60	3.5394
In the middle	3.52	3.5159
In the middle	3.90	3.5159
Near the front	3.66	3.5394
Near the front	3.87	3.5394
Near the front	3.20	3.5394
In the middle	3.50	3.5159
Near the front	3.68	3.5394
In the middle	3.60	3.5159
Near the front	3.73	3.5394
Near the front	3.60	3.5394
Near the front	3.37	3.5394
Near the front	3.97	3.5394
In the middle	3.40	3.5159
Near the front	3.70	3.5394
Toward the back	3.30	3.3469
Near the front	3.86	3.5394
In the middle	3.78	3.5159
Near the front	3.60	3.5394

If you scroll through the table you'll see that there are only three unique predicted GPAs, one for each level of 'sit'. In the chapter ANOVA is just regression we plotted the real vs. predicted GPAs. Let's do that again here but for the whole class. We'll also plot the means as blue triangles:

```
plot(na.omit(survey$GPA_UW),pred1,
     pch = 21,
     col = 'black',
     bg = 'lightgreen',
     xlab = 'GPA',
     ylab = 'Predicted GPA')

points(m1, m1, pch = 24,bg = 'lightblue',cex = 1.5)
grid()
```



22.2 Statistical Significance

For ANOVA, we think of statistical significance as determining if there is more variability across the three means than is expected by chance (based on the variability within means). For regression, we think of statistical significance as whether we can justify having three parameters to fit the data instead of one. With regression we can test for significance by comparing how good our *complete* model with three parameter fits compared to a *restricted* model that predicts all GPAs are the same (the grand mean). Recall from the ANOVA is just regression chapter that this turns out to be an F-test, with SS, MS, df, and p-values equal to those from the traditionally calculated ANOVA.

In R, we can conduct this test by passing the output of the model into the 'anova' function. In the ANOVA is just regression chapter we passed in the results from this 'full' model and a restricted model. If we send in only one model result the *restricted* model is implied and run as a comparison:

```
anova(lm1.out)
```

```
## Analysis of Variance Table
```

```
##
## Response: GPA_UW
##             Df  Sum Sq Mean Sq F value  Pr(>F)
## sit          2  0.8155 0.40774  3.4476 0.03444 *
## Residuals 146 17.2671 0.11827
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The table should look familiar. It should match the values done with the standard one-factor ANOVA.

In the plot above, the spread of the points horizontally around the means should be equal to SS_{within}. Let's check:

```
sum( sum((na.omit(survey$GPA_UW)-pred1)^2))
```

```
## [1] 17.26707
```

This matches the SS for 'Residuals' in the ANOVA table, which is SS_{within}

22.3 Two factor balanced ANOVA example

Next we'll revisit the data on the effects of Beer and Caffeine on reaction times that we analyzed in the Two Factor ANOVA chapter. This chunk loads in the data and orders the levels.

```
RTdata <- read.csv("http://www.courses.washington.edu/psy524a/datasets/BeerCaffeineANOVA2.csv")
RTdata$beer <- factor(RTdata$beer,levels = c('no beer','beer'))
RTdata$caffeine <- factor(RTdata$caffeine,levels = c('no caffeine','caffeine'))
```

Table 22.2:

Responsetime	caffeine	beer
2.24	no caffeine	no beer
1.62	no caffeine	no beer
1.48	no caffeine	no beer
1.70	no caffeine	no beer
1.06	no caffeine	no beer
1.39	no caffeine	no beer
2.69	no caffeine	no beer
0.28	no caffeine	no beer
2.24	no caffeine	no beer
1.15	no caffeine	no beer
1.53	no caffeine	no beer
2.43	no caffeine	no beer
1.71	no caffeine	beer
2.19	no caffeine	beer
2.27	no caffeine	beer
2.35	no caffeine	beer
2.47	no caffeine	beer
2.07	no caffeine	beer
2.56	no caffeine	beer
2.35	no caffeine	beer
1.50	no caffeine	beer
2.63	no caffeine	beer
2.48	no caffeine	beer
1.94	no caffeine	beer
0.62	caffeine	no beer
1.72	caffeine	no beer
1.75	caffeine	no beer
1.84	caffeine	no beer
1.30	caffeine	no beer
1.52	caffeine	no beer
1.31	caffeine	no beer
1.63	caffeine	no beer
1.91	caffeine	no beer
1.33	caffeine	no beer
0.84	caffeine	no beer
0.45	caffeine	no beer
2.05	caffeine	beer
1.51	caffeine	beer
1.65	caffeine	beer
2.68	caffeine	beer
2.06	caffeine	beer
1.80	caffeine	beer
2.68	caffeine	beer
1.93	caffeine	beer
1.29	caffeine	beer
1.93	caffeine	beer
1.35	caffeine	beer
1.37	caffeine	beer

This data is 'balanced', meaning that there are the same number of observations in each cell. You can see this using R's 'table' function:

```
table(RTdata$caffeine,RTdata$beer)
```

```
##
##               no beer beer
##   no caffeine      12   12
##   caffeine         12   12
```

With a balanced data set, the results from a linear regression analysis will be the same as for the traditional ANOVA. Here's the result using R's `lm` function:

```
lm2.1.out <- lm(Responsetime ~ caffeine*beer,data = RTdata)
anova(lm2.1.out)
```

```
## Analysis of Variance Table
##
## Response: Responsetime
##               Df  Sum Sq Mean Sq F value   Pr(>F)
## caffeine       1  1.2708  1.2708  4.9498 0.031269 *
## beer           1  3.4080  3.4080 13.2748 0.000706 ***
## caffeine:beer  1  0.0083  0.0083  0.0322 0.858395
## Residuals     44 11.2960  0.2567
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The model, `Responsetime ~ caffeine*beer` is telling `lm` to predict the dependent variable, 'Responsetime', by the nominal scale independent factors 'caffeine' and 'beer' and their interaction.

22.4 Two factor unbalanced design

This next example will use the survey data to study the effects of gender and math preference on students' predicted score in the first exam. The two (nominal scale) independent measures are 'gender' and 'math', and the ratio scale dependent measure is 'Exam1'. This is an unbalanced data set because there are unequal numbers of students that fall into the categories of math preference and gender. We can see that with another table:

```
table(survey$gender,survey$math)
```

```
##
##         A fair amount Just a little Not at all Very much
##   Man               7            14          5         3
##   Woman            25            62         33         2
```

You may have been taught that we can't use a two-factor ANOVA with an unbalanced design like this. But linear regression allows for it

```
lm3.out <- lm(Exam1 ~ math*gender, data = survey)
anova(lm3.out)
```

```
## Analysis of Variance Table
##
## Response: Exam1
##              Df Sum Sq Mean Sq F value  Pr(>F)
## math          3  571.7 190.569  3.0736 0.02974 *
## gender        1   41.8  41.769  0.6737 0.41314
## math:gender   3  350.2 116.721  1.8825 0.13520
## Residuals   143 8866.4  62.003
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

But here's where things get complicated. With an unbalanced design you'll find that the results are different if we switch the order of the independent variables:

```
lm4.out <- lm(Exam1 ~ gender*math, data = survey)
anova(lm4.out)
```

```
## Analysis of Variance Table
##
## Response: Exam1
##              Df Sum Sq Mean Sq F value  Pr(>F)
## gender        1  103.5 103.473  1.6688 0.19850
## math          3  510.0 170.001  2.7418 0.04549 *
## gender:math   3  350.2 116.721  1.8825 0.13520
## Residuals   143 8866.4  62.003
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

What's the deal here? It shouldn't matter which order you state your independent variables. This is true for a balanced data set (try it for the beer and caffeine example above). But for unbalanced designs it can matter. The reason has to do with the columns of the design matrix generated by `lm`. For a balanced design, the columns - or regressors - are mutually orthogonal, so that each column is pulling out independent sources of variability in our dependent variable. But for an unbalanced design, the columns are no longer orthogonal, so the order matters.

It might help to consider the case of predicting student's heights from parent's heights. Suppose we use both mother's and father's heights as regressors. There is a strong correlation between mother's and father's heights. So if we use mother's heights as the first factor, there isn't much remaining variability in student's height left for the father's heights to account for. But if we use father's heigths first, then there's not much variability left for the mother's heights to predict. So the order matters:

```
anova(lm(height ~ mheight + pheight,data = survey))
```

```
## Analysis of Variance Table
##
## Response: height
##            Df  Sum Sq Mean Sq F value    Pr(>F)
## mheight     1  184.14 184.137  20.406 1.326e-05 ***
## pheight     1  256.28 256.279  28.401 3.883e-07 ***
## Residuals 139 1254.27   9.024
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(lm(height ~ pheight + mheight,data = survey))
```

```
## Analysis of Variance Table
##
## Response: height
##            Df  Sum Sq Mean Sq F value    Pr(>F)
## pheight     1  368.44  368.44 40.8308 2.334e-09 ***
## mheight     1   71.98   71.98  7.9769  0.005435 **
## Residuals 139 1254.27    9.02
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

For an unbalanced anova, the regressors are also not orthogonal, so the order matters. I've found a useful discussion on this topic on (stack exchange)[https://stats.stackexchange.com/questions/552702/multifactor-anova-what-is-the-connection-between-sample-size-and-orthogonality]

It turns out that there three ways to deal with this issue with ANOVA. They are called Type I, II and III ANOVAs.

Notice that the SS, df, and MS for the 'Residuals' are the same (which serve as the denominator for the F test), regardless of the order. Thus, the fit of the complete model is the same, regardless of the order of factors. The

difference is in the way SS is calculated for the main effects. The 'anova' function runs a type I ANOVA, and with type I ANOVAs the order of variables does make a difference.

22.5 Type I ANOVA

Remember from the ANOVA is just regression chapter that with regression, the numerator of the F statistic is based on comparing a 'complete' model to a 'restricted' model. For type I ANOVAs, the SS for the numerator has to do with the order in which the model is increased with complexity. For the current example, consider the fit of the model in the following increasing levels of complexity. Note, the first model, 'Exam1 ~ 1', is a fit of a single parameter to the whole data set - leading to a calculation of the grand mean.

```
a <- lm(Exam1 ~ 1, data = survey)
b <- lm(Exam1 ~ math, data = survey)
c <- lm(Exam1 ~ gender + math, data = survey)
d <- lm(Exam1 ~ gender * math, data = survey)
```

The function 'deviance' takes in the output of 'lm' and gives you the sums of squared error for that model. For example, the sums of squared for the simplest model, model 'a' is:

```
deviance(a)
```

```
## [1] 9830.04
```

This is SS_{total} The SS for the second model, with just the 'math' factor is:

```
deviance(b)
```

```
## [1] 9258.333
```

It's smaller because it's a more complicated model. Let's look at how the SS decreases as we increase the complexity of the model in the order chosen above:

```
deviance(a)-deviance(b)
```

```
## [1] 571.7068
```

```
deviance(b)-deviance(c)
```

```
## [1] 41.76925
```

```
deviance(c)-deviance(d)
```

```
## [1] 350.1624
```

These numbers match the values in the 'Sum Sq' column when we ran the lm with math followed by gender. Thus the numerators for the type I ANOVA are based on adding each factor (and their interactions) sequentially. This is why type I ANOVAs are sometimes called 'sequential'. With a Type I ANOVA, putting math as the first factor tests the main effect of math first, followed by tghe main effect of gender, followed by their interaction. This is different than putting gender first followed by math.

This dependence on the order of factors is generally not a desirable thing. That's why Type II (and then type III) ANOVAs have been developed.

22.6 Type II ANOVA

The easiest way to esolve this problem is to run type I ANOVAs using all combinations of orders and then pulling out the SS that's most appropriate for the test we want to make. In the example above, the main effect of gender is best studied by comparing gender+math to math alone. So we should pull the SS from the lm fit with math first followed by gender. Conversely, the main effect of math should be pulled from the fit of gender first followed by math. That is, you should use the SS for your factor when it is added last to the list - after the other factors.

To run a type II ANOVA, we need to use the "car" package and use the 'Anova' function. Yes, the same name as 'anova', but with a capital 'A'. Welcome to the world of open source software.

Here's how to use 'Anova' to obtain result of the two-factor for gender and Exam 1 using a type II analysis:

```
Anova(lm3.out,type = "II")
```

```
## Anova Table (Type II tests)
##
## Response: Exam1
##             Sum Sq  Df F value  Pr(>F)
## math         510.0   3  2.7418 0.04549 *
## gender        41.8   1  0.6737 0.41314
## math:gender  350.2   3  1.8825 0.13520
## Residuals   8866.4 143
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This is exactly the same as using lm4.out, which had gender first:

```
Anova(lm4.out,type = "II")
```

```
## Anova Table (Type II tests)
##
## Response: Exam1
##             Sum Sq  Df F value  Pr(>F)
## gender        41.8   1  0.6737 0.41314
## math         510.0   3  2.7418 0.04549 *
## gender:math  350.2   3  1.8825 0.13520
## Residuals   8866.4 143
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Which is a good thing.

22.7 Type III ANOVA

Type II ANOVA tests for a main effect after the other main effect(s), but what about an interaction? Type III ANOVAs test for a main effect for a factor after accounting for other main effecs AND their interactions. Like type II ANOVAs, the order of the factors doesn't matter.

For a balanced design, the results of a type III anova will match those from a traditional ANOVA. For unbalanced designs, I often get errors running type III analysis. It happens with this regression example. You can try it for yourself.

Chapter 23

Linear Mixed Models

23.1 Using lmer for a Repeated Measures Design

In the previous chapter Linear Models we covered how to run one and two factor ANOVAs with R's 'lm' function. All of the examples in that chapter were independent measures designs, where each subject was assigned to a different condition. Now we'll move on to experimental designs with 'random factors' like repeated measures designs and 'mixed models' that have both within and between subject factors. R's 'lm' function can't handle design with random factors, so we'll use the most popular function 'lmer' from the 'lme4' package. Go ahead and install it if you haven't already.

23.2 Example: Effect of Exercise over Time on Body Weight

We'll start with the first example from the Repeated Measures ANOVA chapter. The first example was the study of exercise over time on weight (in kg). In this example, 6 subjects' weights were measured over 3 points in time: "before", "after three months", and "after six months" since starting the exercise program.

First we'll load in the data and take a look:

```
weightData<-
  read.csv("http://www.courses.washington.edu/psy524a/datasets/SubjectExerciseANOVAdependent.csv")

head(weightData)
```

```
##   subject               time weight
## 1      S1             before     45
## 2      S1 after three months     50
## 3      S1   after six months     55
## 4      S2             before     42
## 5      S2 after three months     42
## 6      S2   after six months     45
```

The advantage of a repeated measures design is that we can subtract out the individual differences in each subject's weight. 'subject' is a 'random effect factor', as opposed to 'time', which is a 'fixed effect factor'. A random effect factor has many possible levels, but the experiment only sampled a subset of them. By defining a factor as a random effect factor, you are seeing if your results generalize across all levels of that factor. 'subject' is a random effect factor because you have sampled a subset of subjects for your experiment, and you want to make conclusions about the entire population of subjects, not just the ones that you've sampled. Conclusions made from fixed effect factors can only be attributed to the specific levels that you included in your experiment.

In the context of ANOVA, we dealt with subject as a repeated measure factor by subtracting out the individual differences in weight across subjects by calculating $SS_{subjects}$ and subtracting that from SS_{within}. Mathematically it makes sense - you get to see how the systematic error is taken away from the SS for the denominator of the F-test, thereby increasing F and decreasing your p-value.

Another way to think about accounting for individual differences in weight is to add a regressor for each subject to your model so that each subject gets its own free parameter, called an 'intercept'. Here's how to use lmer to fit a regression model to the exercise data:

```
lmer1.out <- lmer(weight ~ time + (1|subject), data=weightData)
```

The first part of the model uses the same conventions as 'lm'. *weight time* should look familiar; we're predicting weight as a function of the factor, time. The new part is the stuff in parentheses which defines the random effect variable. (1|*subject*) means that we're going to allow for an intercept for each subject. When we pass the output of lmer into 'anova' you'll see the same results as in the Repeated Measures ANOVA chapter:

```
anova(lmer1.out)
```

```
## Type III Analysis of Variance Table with Satterthwaite's method
##      Sum Sq Mean Sq NumDF DenDF F value   Pr(>F)
## time 143.44  71.722     2    10  12.534 0.001886 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Let's use the 'coef' function to look at the values (coefficients) that best fit the data:

```
coef(lmer1.out)
```

```
## $subject
##    (Intercept) timeafter three months timebefore
## S1    53.54595              -4.333333  -6.833333
## S2    46.85020              -4.333333  -6.833333
## S3    43.98059              -4.333333  -6.833333
## S4    42.06752              -4.333333  -6.833333
## S5    58.32864              -4.333333  -6.833333
## S6    53.22711              -4.333333  -6.833333
##
## attr(,"class")
## [1] "coef.mer"
```

This looks different than the coefficients for the between subjects ANOVA discussed in the Linear Models chapter. Here there is a row for every subject. This is because with a repeated measure design like this, every subject gets their own 'intercept', which allows for each subject's weight to vary overall. The other two coefficients are the same across subjects. That's because 'time' is a fixed effect variable, so it affects all subjects the same way.

These coefficients are in the same form as for the independent measures ANOVA. Alphabetically, 'after six months' comes first, so this is the first regressor.

The two columns correspond to the difference between the 'after six months' condition

For a simple example like this, it turns out that we could have used the old 'lm' function to allow for a different parameter for each subject. This is simply done by letting our model be *weight time* + *subject*. Note the lack of interaction in the model:

```
lm1.out <- lm(weight ~ time + subject, data=weightData)
anova(lm1.out)
```

```
## Analysis of Variance Table
##
## Response: weight
##           Df Sum Sq Mean Sq F value    Pr(>F)
## time       2 143.44  71.722  12.534  0.001886 **
## subject    5 658.28 131.656  23.008 3.461e-05 ***
## Residuals 10  57.22   5.722
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We get the same F and p-values for the effect of time. We also get a measure of the differences in weights across subjects, which we don't care about. You can see how there's confusion about the definition of a random factor. Using 'lm' we're thinking of both time and subject as fixed factors, but with 'lmer' we're thinking of time as fixed but subject as random. But the results turn out the same.

You're now probably wondering why we have 'lmer' if we get the same thing using 'lm', treating all variables as fixed, and just ignoring the results from the variables we don't care about. The reason is that 'lmer' can handle much more complicated situations that 'lm' can't deal with. For example, mixed designs:

23.3 Mixed Design Example: Effect of Napping and Time on Perceptual Performance

A mixed-design has a mixture of independent and dependent (usually repeated) measures factors. The math behind mixed-designs are extensions of what we already know about independent and dependent measures, so although it sounds complicated, it's not that bad. The simplest mixed design has one independent and one dependent factor. Here's an example:

A former postdoc of mine, Sara Mednick, found that if you test the same subject on a texture discrimination task (finding a T amongst L's) your threshold (in milliseconds) increases (gets worse) across sessions throughout the day (when testing at 9am, 12pm, 3pm and then 6pm). In her experiment, half of the subjects took a nap after the 12pm session.

This is a mixed design because it has one within-subjects factor and one between-subjects factor. The time of testing is the within-subjects factor (each subject was tested four times), and the napping condition (nap vs. no nap) is the between-subjects factor.

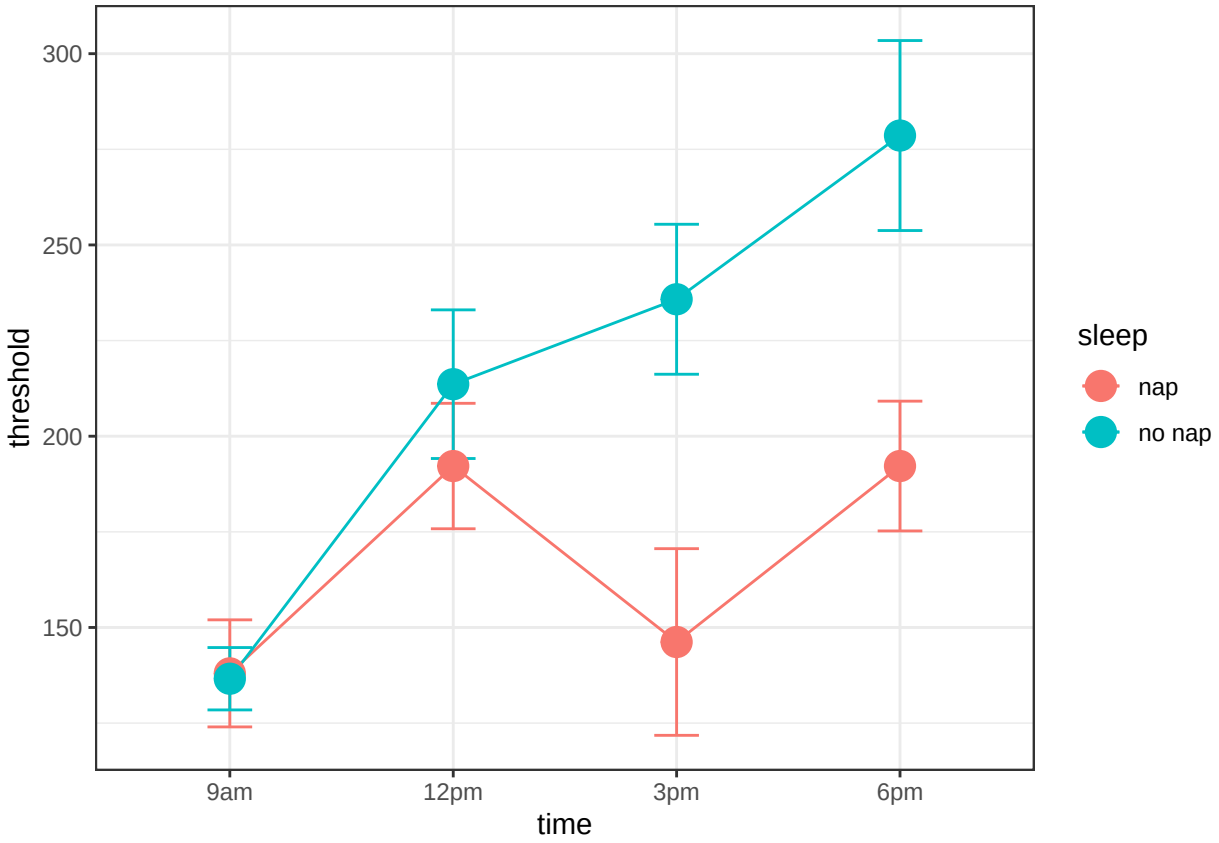
Let's load in the data and take a peek:

```
napData<-read.csv("http://www.courses.washington.edu/psy524a/datasets/napXtimemixed.csv")
head(napData)
```

```
##   subject  sleep time threshold
## 1      S1 no nap  9am       151
## 2      S1 no nap 12pm       285
## 3      S1 no nap  3pm       180
## 4      S1 no nap  6pm       356
## 5      S2 no nap  9am       105
## 6      S2 no nap 12pm       179
```

There are three independent factors: subject, sleep, and time. Subject is a random factor, sleep is a between subject factor and time is a within subject factor. The dependent factor is threshold.
Next we'll use 'summarySE' from the 'Rmisc' library and 'ggplot' from the 'ggplot2' library to compile the data and make plot of the means and standard errors over time.

```
# reorder the levels for time
napData$time <- ordered(napData$time, levels = c( "9am","12pm","3pm","6pm"))

# get the means and standard errors
napData.summary <- summarySE(napData, measurevar="threshold", groupvars=c("sleep","time"))

ggplot(napData.summary, aes(x=time,y=threshold,colour = sleep, group = sleep)) +
  geom_errorbar(aes(ymin=threshold-se, ymax=threshold+se), width=.2) +
  geom_line() +
  theme_bw() +
  geom_point(size = 5)
```



Notice how the thresholds decrease over time for the subject that didn't get a nap, but when subjects took a nap after 12pm, thresholds dropped back down (and then climbed back up for the 6pm session).

Now we'll fit the data with a linear model with the factors to see if there's a significant effect of time, sleep and and interaction. The analysis is straightforward using lmer. The model is simply $threshold\ time * sleep + (1|subject)$. We don't have to tell lmer which factor is between and which is within. It's all apparent in the data.

```
lmer2.out  <- lmer(threshold ~ time*sleep + (1|subject),data = napData)
anova(lmer2.out)
```

```
## Type III Analysis of Variance Table with Satterthwaite's method
##            Sum Sq Mean Sq NumDF DenDF F value    Pr(>F)
## time        49950 16649.9     3    24 11.7345 6.277e-05 ***
## sleep       12497 12497.4     1     8  8.8079   0.01793 *
## time:sleep  15873  5290.9     3    24  3.7289   0.02482 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The three hypothesis tests here - two main effects and an interaction - aren't necessarily the most interesting set of tests for this experiment. Yes, there is a main effect of sleep, so thresholds without a nap are higher than with a nap. But there's also an interaction between time and sleep, which makes the main effect of sleep more difficult to interpret. Another test we could run (it'd be an ariori test if we had thought of it before running the experiment) would be to look only at the 3pm and 6pm data. This can be done using the 'subset' function:

```
afternoonData <- subset(napData,time == "3pm" | time == "6pm")

lmer4.out <- lmer2.out  <- lmer(threshold ~ time*sleep + (1|subject),data = afternoonData)
anova(lmer4.out)
```

```
## Type III Analysis of Variance Table with Satterthwaite's method
##            Sum Sq Mean Sq NumDF DenDF F value  Pr(>F)
## time         9857    9857     1     8  4.3524 0.07043 .
```

```
## sleep       35846   35846     1     8 15.8281 0.00407 **
## time:sleep     13      13     1     8  0.0057 0.94192
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Here, consistent with the plot, for just the afternoon you can see that there's a main effect for time (threshold go up between 3pm and 6pm), there's a main effect of sleep (thresholds are higher without a nap), but no interaction between time and sleep. Note that running an analysis on a subset of the data is a little like running a simple effects analysis with ANOVAs.

23.4 Mixed Design Example: Effect of Donezepil on Test Scores

Suppose you want to test the efficacy of the Alzheimer's drug, Donepezil on various cognitive tests. 8 subjects are divided into 2 groups, "without Donepezil", and "with Donepezil", and each subject is given 3 tests: "memory", "reading", and "verbal". 8 subjects are divided into 2 groups, "without Donepezil", and "with Donepezil". Each subject repeats each test across 5 days.

This is a mixed design because 'drug' is a between subject factor and both 'test' and 'day' are within subject factors.

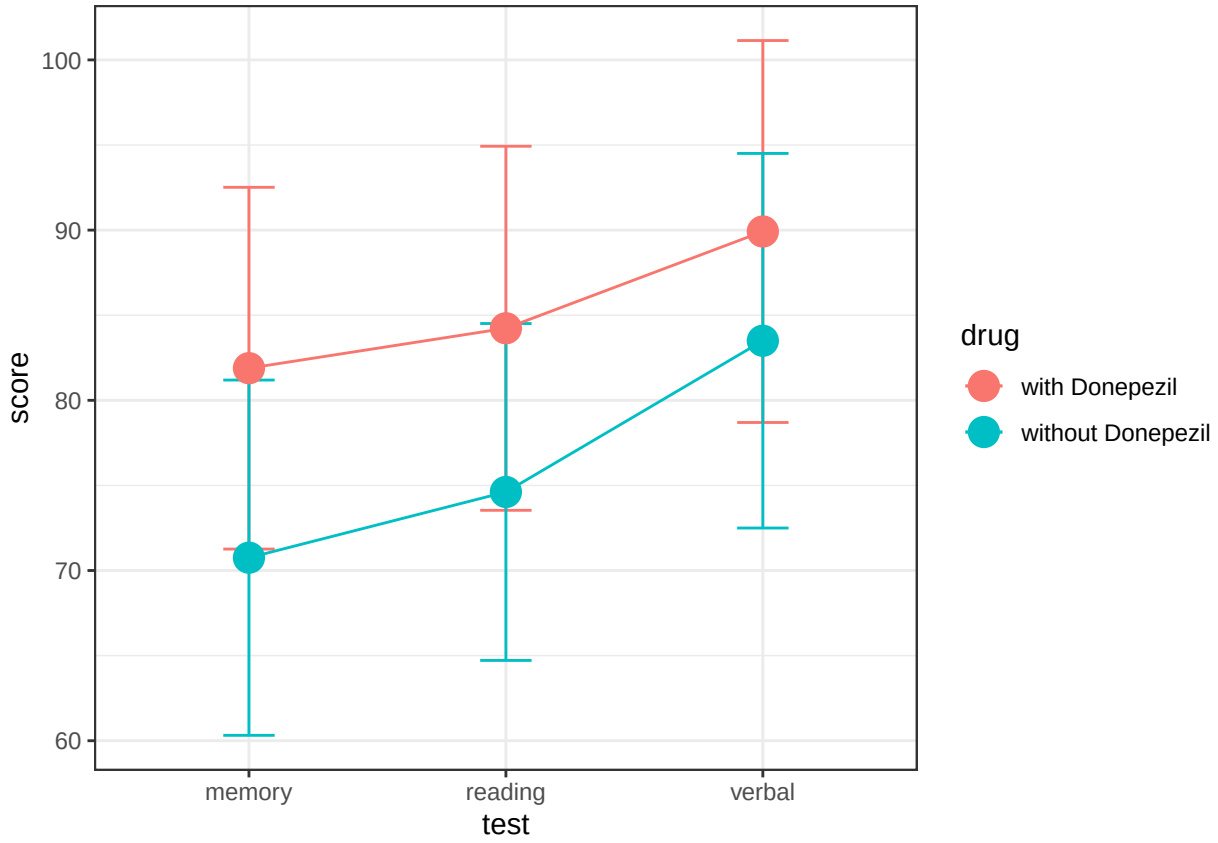
Let's load in the data file and take a look:

```
drugData<-read.csv("http://www.courses.washington.edu/psy524a/datasets/DrugTestMixed.csv")
head(drugData)
```

```
##   subject              drug day   test    score
## 1      S1 without Donepezil Mon memory 45.74021
## 2      S2 without Donepezil Mon memory 69.44595
## 3      S3 without Donepezil Mon memory 34.52847
## 4      S4 without Donepezil Mon memory 32.22222
## 5      S1 without Donepezil Tue memory 37.61943
## 6      S2 without Donepezil Tue memory 84.60970
```

Let's not worry about the 'day' factor for now and plot the test scores across groups:

```
# get the means and standard errors
drugData.summary <- summarySE(drugData, measurevar="score", groupvars=c("drug","test"))

ggplot(drugData.summary, aes(x=test,y=score,colour = drug, group = drug)) +
  geom_errorbar(aes(ymin=score-se, ymax=score+se), width=.2) +
  geom_line() +
  theme_bw() +
  geom_point(size = 5)
```



It looks like test scores are slightly higher with Donepezil, but the overlapping error bars mean that it might not be statistically significant. Also, there doesn't look like there's much difference across test scores. Let's run the regression analysis, leaving out the interaction. We'll use '(1|subject)' as the random effect.

```
# Run the ANOVA
lmer5.out <- lmer(score ~ test + drug + (1|subject),data =drugData)
anova(lmer5.out)
```

```
## Type III Analysis of Variance Table with Satterthwaite's method
##       Sum Sq Mean Sq NumDF DenDF F value Pr(>F)
## test 2275.24 1137.62     2   110  1.1939 0.3069
## drug   91.26   91.26     1     6  0.0958 0.7674
```

As predicted from the graph, the effects of the drug are not statistically significant, nor is there a significant difference in scores across the tests.

To understand how the regression model fit the data, it helps to look at the best fitting coeficents, or beta weights:

```
coef(lmer5.out)
```

```
## $subject
##    (Intercept) testreading testverbal drugwithout Donepezil
## S1    48.00598    3.103345     10.389              -9.056597
## S2   134.60783    3.103345     10.389              -9.056597
## S3    84.38692    3.103345     10.389              -9.056597
## S4    56.39305    3.103345     10.389              -9.056597
## S5    43.34021    3.103345     10.389              -9.056597
## S6   136.82312    3.103345     10.389              -9.056597
## S7    82.53079    3.103345     10.389              -9.056597
## S8    60.69966    3.103345     10.389              -9.056597
##
```

```
## attr(,"class")
## [1] "coef.mer"
```

The coeficients tell us how the model predicts each data point. Notice how using '(1|subject)' as the random effect in lmer allows for a different intercept for each subject. However, the coeficients for the other levels are all the same. That's because test and drug are fixed factors. We can use these coeficients to see the model predictions. Since 'memory' is the first level alphabetically in the test levels, the effects of the other levels are added to the 'memory' score. For example, subject S1, who is in the 'without Donepezil' group, has a predicted test score for 'memory' that is this subject's intercept plus the effect of being in the 'without Donepizil' group:

$$48.006 - 9.0566 = 38.9494$$

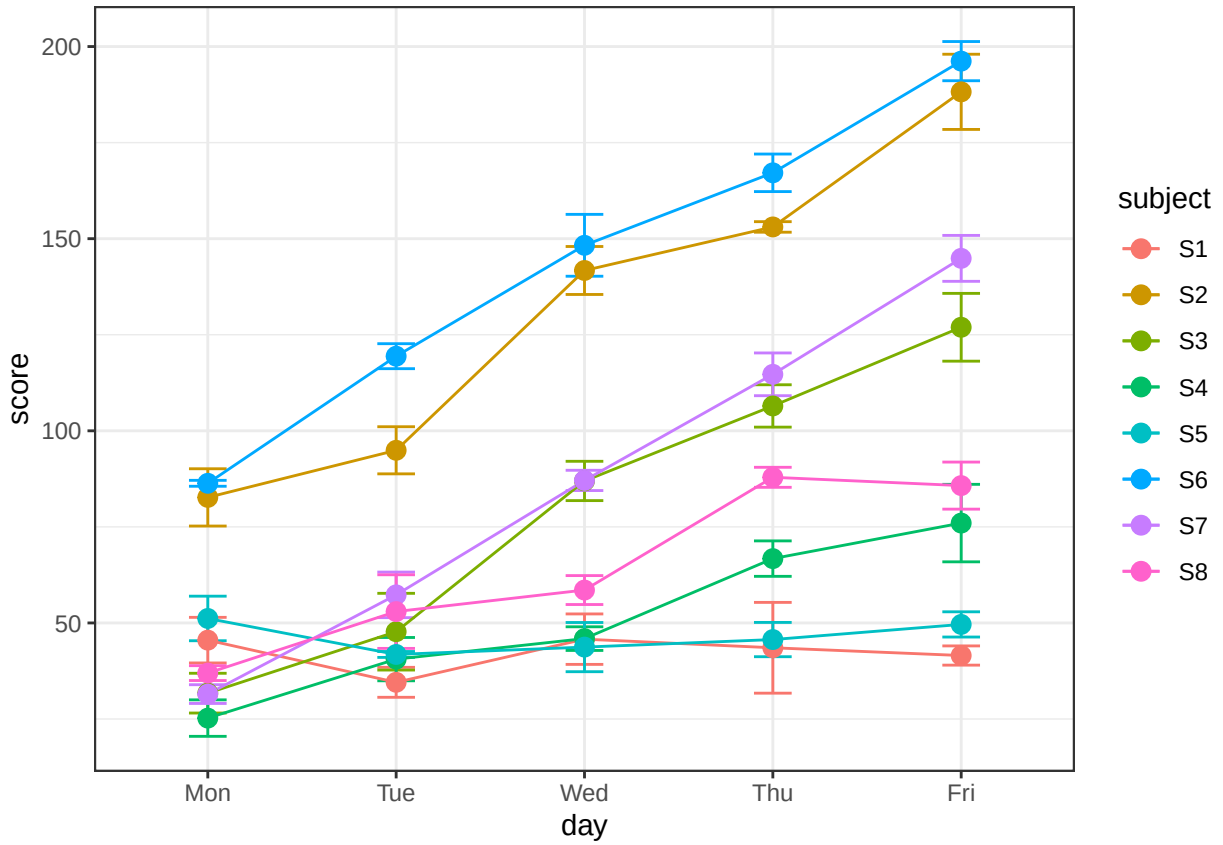
This subject's predicted score for the 'reading' group is this subject's intercept plus the effect of being in the 'without Donepizil' group plus the effect of being in the 'reading' group:

$$48.006 - 9.0566 + 3.10335 = 42.0527$$

What about the 'day' factor? It turns out that there is a significant amount of learning across days in this (fake) data set. We can see this by plotting the test scores across days (averaging across the three tests) for each subject:

But what about 'day'?

```
drugData.day.summary <- summarySE(drugData, measurevar="score", groupvars=c("day","subject"))
drugData.day.summary$day <- ordered(drugData.day.summary$day, levels = c("Mon","Tue","Wed","Thu","Fri"))
ggplot(drugData.day.summary, aes(x=day,y=score,colour = subject, group = subject)) +
  geom_errorbar(aes(ymin=score-se, ymax=score+se), width=.2) +
  geom_line() +
  theme_bw() +
  geom_point(size =3)
```



Can you see the improvement in scores across the week? 'day' is a random effect variable, since it's a variable that influences our results but we're not interested in it as part of our study. To account for it, we add five more parameters to the model: an intercept for each day by adding '(1|day)' as a random effect:

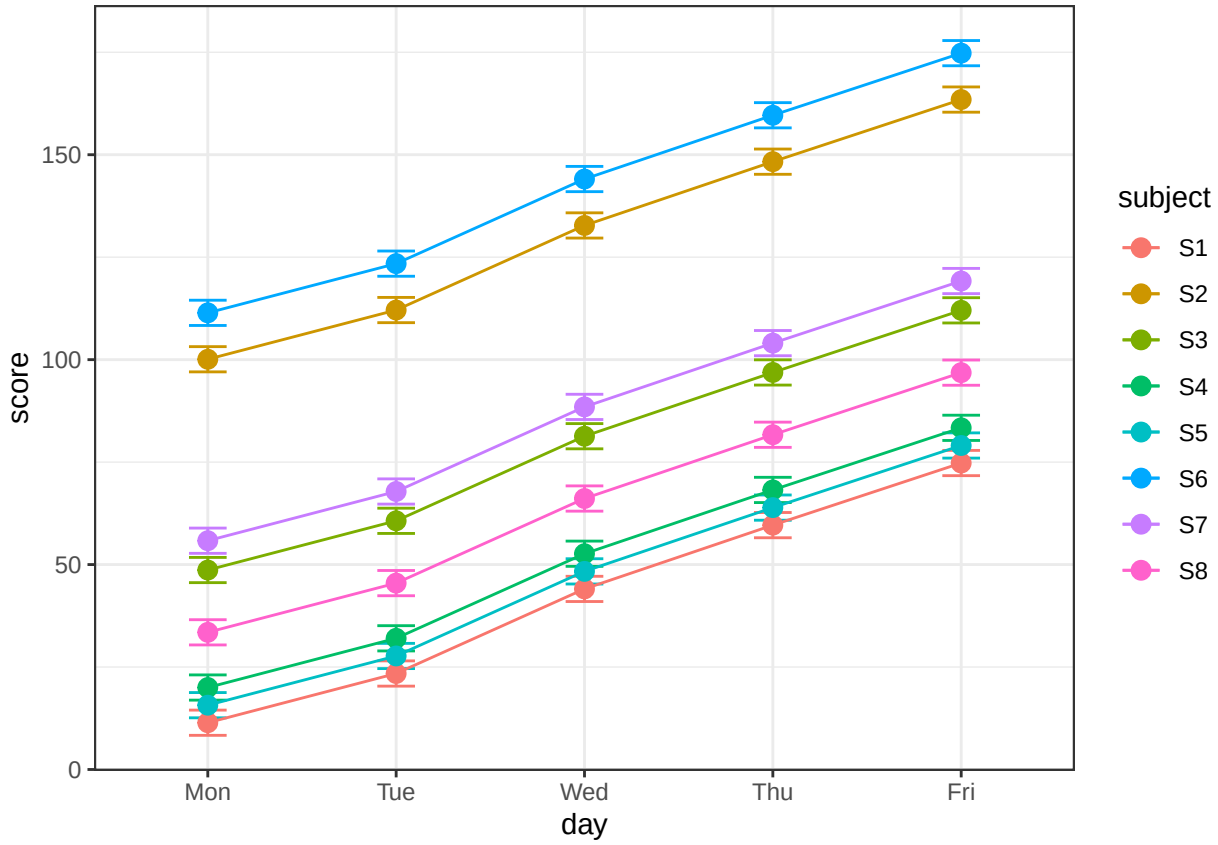
```
lmer6.out <- lmer(score ~ test + drug + (1|subject)+ (1|day),data =drugData)
anova(lmer6.out)
```

```
## Type III Analysis of Variance Table with Satterthwaite's method
##       Sum Sq Mean Sq NumDF DenDF F value  Pr(>F)
## test 2275.24 1137.62     2   106  3.1775 0.04569 *
## drug   34.29   34.29     1     6  0.0958 0.76742
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Now you can see that there is (barely) a significant difference across the three test scores. Note, however, that the results for the 'drug' factor hasn't changed. This is because 'drug' is a between subject factor for both day and subject. So repeated measures across tests and day doesn't help the test of drug. 'subject' as random effect variables only increases the power of the test for drug if each subject was tested both with and without the drug. Similarly, with 'day' as a random effect would only help the effect of drug if the drug was introduced across days.

One way to see how your model predicts the data is to plot the model's prediction instead of the real data. An easy way to do this is to create a new data set from the old one, and replace the real data with the model predictions using R's 'predict' function. Here's how:

```
# copy the original data into a new data frame
predDrugData <- drugData

# replace the real dv 'score' with the predicted scores
predDrugData$score <- predict(lmer6.out)

# plot the data like we did before
predDrugData.day.summary <- summarySE(predDrugData, measurevar="score", groupvars=c("day","subject"))
predDrugData.day.summary$day <- ordered(predDrugData.day.summary$day, levels = c("Mon","Tue","Wed","Thu","

ggplot(predDrugData.day.summary, aes(x=day,y=score,colour = subject, group = subject)) +
  geom_errorbar(aes(ymin=score-se, ymax=score+se), width=.2) +
  geom_line() +
  theme_bw() +
  geom_point(size =3)
```



It's clear now how using subject and day affects the model predictions. Each day and subject contribute to independent additive effects to the predicted data. What's not shown here are are the two fixed factors, drug and test, though we do know that subjects 1 through 4 are without Donepezil and 5 through 8 are with Donepezil. The predicted variability across tests are what leads to the error bars for each data point.

23.5 Random Slopes

If you look closely at the increase in the plot of test scores across days for the real data you'll notice that some subjects improved more rapidly than others throughout the week. You can think of this fact as another random factor, and this factor is not taken into account in the previous analysis since each day gets just one parameter that applies to all subjects. You can, howeer, increase the complexity of the model so that each subject gets its own parameter for each day. You can see why this is called 'random slopes' for this examle - each subject gets their own learning rate. The term is a bit misleading, though, because it doesn't really mean different 'slopes' across days - it really means that each subject has a different pattern of effects across days.

To include random slopes, the random term in parantheses becomes '(day|subject)', which technically means that 'subject is nested within day'. Here goes:

```
lmer7.out <- lmer(score ~ test + drug + (day|subject), data = drugData, REML = FALSE)
```

```
## boundary (singular) fit: see help('isSingular')
```

```
anova(lmer7.out)
```

```
## Type III Analysis of Variance Table with Satterthwaite's method
##       Sum Sq Mean Sq NumDF  DenDF F value    Pr(>F)
## test 2275.24 1137.62     2 94.446  16.204 8.909e-07 ***
## drug  122.37  122.37     1  9.105   1.743     0.219
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Now the test of the difference across tests is wildly significant. You can see how the model was fit to the data by looking at the coeficients:

```
coef(lmer7.out)
```

```
## $subject
##          dayMon      dayThu     dayTue      dayWed (Intercept) testreading
## S1   -0.0786833  -2.1509659  -3.997644   0.8478573    54.23680    3.103345
## S2 -107.8974240 -33.9344199 -89.408927 -45.4897148   198.17427    3.103345
## S3  -95.7243098 -23.1089724 -77.338508 -46.1666636   139.70227    3.103345
## S4  -49.4971665  -8.8052901 -35.714998 -25.5834076    85.84264    3.103345
## S5    1.0425001   0.2702627  -1.775658  -0.2092576    42.10551    3.103345
## S6 -108.7454111 -30.1980088 -82.170591 -47.3632989   192.28822    3.103345
## S7 -108.1418076 -22.6730908 -83.985819 -54.3656978   136.86652    3.103345
## S8  -52.2738003  -8.3899186 -36.533073 -27.5271591    84.74150    3.103345
##    testverbal drugwithout Donepezil
## S1     10.389             -15.49469
## S2     10.389             -15.49469
## S3     10.389             -15.49469
## S4     10.389             -15.49469
## S5     10.389             -15.49469
## S6     10.389             -15.49469
## S7     10.389             -15.49469
## S8     10.389             -15.49469
##
## attr(,"class")
## [1] "coef.mer"
```

With 5 days, there are really only 4 degrees of freedom since the intercept for each subject effectively absorbs the extra parameter. So to account for '(day|subject)' the model needs 4 parameters for each of the 8 subjects. Then there are 2 additional parameters for the 3 tests and another for the 'drug' factor.

This model has 64 parameters. There are only 120 observations in the entire data set. At some point a model can become too complex, leading to 'overfitting' of the data. There's a rule of thumb out there that you should have at least 30 observations per parameter. So for this model we really should have at least 1920 observations.

You might have noticed the 'Model failed to converge' error in this last model fit. This means that lmer couldn't find a unique solution for minimizing the sums of squared error. This can happen for a number of reasons, but the most common one - and the reason here - is when you have too many parameters compared to observations. When the model is overfitting like this, there are too many tradeoffs between parameters so that there isn't a good unique solution. If you get an error like this you shouldn't trust the result.

On the other hand, there is a trend toward larger models that include random slopes, influenced in part by a 2014 paper promoting the idea to "keep it maximal" Barr et al., 2014) where "Through theoretical arguments and Monte Carlo simulation, [they] show that LMEMs generalize best when they include the maximal random effects structure justified by the design."

23.6 'Anova' vs. 'anova'

Recall in the Linear Models chapter we used the 'Anova' function from the 'car' package to run a Type II ANOVA on the unbalanced data set to predict predicted Exam 1 scores from how much students like math and their gender. This was used because with an unbalanced design, the order of factors makes a difference when using 'anova' (lower case 'a').

This is just one example of the difference between 'Anova' and 'anova'. The differences are even more apparent given the ouptut of 'lmer'. Recall our first example in this tutorial - the repeated measures ANOVA predicting weight over time. The data set was balanced, with 6 subjects for each level of time.

Using 'anova', result of the ANOVA was:

```
anova(lmer1.out)
```

```
## Type III Analysis of Variance Table with Satterthwaite's method
##      Sum Sq Mean Sq NumDF DenDF F value   Pr(>F)
## time 143.44  71.722     2    10  12.534 0.001886 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Now look what happens if we use 'Anova' instead:

```
Anova(lmer1.out)
```

```
## Analysis of Deviance Table (Type II Wald chisquare tests)
##
## Response: weight
##       Chisq Df Pr(>Chisq)
## time 25.068  2  3.602e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

'Anova' is a very flexible program and will conduct various statistical tests on all kinds of model fits. By default, 'Anova' takes the output of 'lmer' model fits and conducts a 'Type II Wald chisquare' test. So the statistic is a chi-squared value instead of an F. This is a different way of comparing models than the nested F-test described in earlier tutorials, so the p-value is different too.

You can force 'Anova' to conduct an F-test instead like this:

```
Anova(lmer1.out,test.statistic = 'F')
```

```
## Analysis of Deviance Table (Type II Wald F tests with Kenward-Roger df)
##
## Response: weight
##           F Df Df.res   Pr(>F)
## time 12.534  2     10 0.001886 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Which gives you the same result as 'anova'.

The 'Wald chisquare test' is related to a 'Likelihood Ratio Test'. In short, it's a measure of how likely you'd expect your data given the complete model compared to the restricted model. Instead of dealing with sums-of-squared deviations, the Wald chisquare test is all about probabilities.

Under certain circumstances, a likelihood ratio test is a generalization of the F-test, so the two are related (see this reference for example.

Which to use? This is an ongoing debate in the statistics field. Digging around in the literature I don't see a consensus.

23.7 Treating *test* as a random effect factor

The analysis so far has treated *test* as a 'fixed effect' factor, meaning that we've been interested in these three specific tests. However, another way to consider the factor *test* is as a 'random effect' factor, which is to consider these three tests as a random sample from a larger set of possible tests.

We've already discussed *subject* as a random effect factor. When you treat *subject* as a random effect factor (using `+ (1|subject)` in lmer), you are telling `lmer` that your specific choice of subjects is a random sample from a larger population, which lets you generalize the interpretation of your results to the larger population of subjects. While *subject* is the most common random effect factor, it's possible to treat other factors as random effect factors too.

To do this with `lmer`, we take *test* out of the fixed effect side of the model and replace it with `(1|test)`. Here's the most complicated ANOVA that we'll run in this class. It studies the effect of *drug* on *score* using random slopes

with subject nested within day, and *test* as a random effect factor:

```
lmer8.out <- lmer(score ~ drug + (day|subject) + (1|test), data = drugData, REML = FALSE)
anova(lmer8.out)
```

```
## Type III Analysis of Variance Table with Satterthwaite's method
##      Sum Sq Mean Sq NumDF DenDF F value Pr(>F)
## drug 122.62  122.62     1 9.013  1.7082 0.2236
```

Notice the huge reduction in the denominator degrees of freedom, from 94.446018 to 9.0129885. This is similar to the reduction in the denominator degrees of freedom for a repeated measures ANOVA when we replaced SS_{within} with SS_{error}. In general, it's harder to reject H_0 when a factor is treated as a random effect factor, since you're making a stronger conclusion (generalizing across a greater population of levels for that factor).

23.8 Conclusions and References

We've only scraped the surface here on the evolving topic of linear mixed models. It is clear that the traditional ANOVA is on its way out due their lack of flexibility and inability to handle unbalanced designs including missing data. This tutorial is the result of reading a range of papers, blogs, and comments in sites like StackExchange. The following is a list of resources that I found most useful:

Bodo Winter a tutorial on lm and one on lmer that is a great introduction to the topic. I probably should have just had the class read these instead of my own notes. You can find them on our course website at http://courses.washington.edu/psy524a/pdf/Bodo_Winter_linear_mixed_model_tutorial.pdf

Referenced above is the paper on "keeping it maximal": 'Barr et al., 2014'

Reference on comparing F and likelihood ratio tests: 'Yu and Zhang, 2008'

Reference on the controversy about p-values and unbalanced mixed designs

https://featuredcontent.psychonomic.org/putting-ps-into-lmer-mixed-model-regression-and-statistical-significance/

Including some comments from the author of lmer:

https://stat.ethz.ch/pipermail/r-help/2006-May/094765.html

Chapter 24

Logistic Regression

Logistic regression is just like regular linear regression (and therefore like ANOVA) except that the dependent variable is binary - zeros and ones. Here we'll work with a specific example from research with Ione Fine and grad student Ezgi Yucel. Ezgi obtained behavioral data from patients with blindness that had received retinal implants. These patients, who had lost sight due to retinal disease, had an array of stimulating electrodes implanted on their retinas (Argus II, Second Sight inc.) When current is pulsed through these electrodes some of the remaining retinal cells are stimulated, leading to a visual percept.

Ideally, each electrode should lead to a single percept, like a spot of light. A desired image could then be generated by stimulating specific patterns of electrodes. But sometimes it doesn't work out this way. Simultaneously stimulating some pairs of electrodes leads to a single percept. The percepts induced by single electrodes are large enough that simulating neighboring electrodes can lead to a single 'blob'. Also, the size of the percept increases with the amplitude of the current, so high currents can also cause percepts to merge. The ability to see two separate spots therefore depends on how far the electrodes are physically far apart and the amplitude of the current.

24.1 Example data: visual percepts for retinal prostheses patients

Ezgi stimulated various pairs of electrodes simultaneously at a range of amplitudes and asked three patients to choose whether they saw one or two percepts. The data is loaded in from the csv file here:

```
myData<-read.csv("http://courses.washington.edu/psy524a/datasets/two_percept_data.csv")
```

Here's what the data structure looks like:

Table 24.1:

amp	dist	resp
331	1818.3097	0
299	2370.7857	0
565	3352.7973	1
387	2875.0000	1
387	2875.0000	1
452	2073.1920	1
299	2370.7857	1
412	3450.0000	1
266	2300.0000	1
533	1285.7391	0
266	2300.0000	0
387	2875.0000	1
500	4146.3840	1
266	2300.0000	1
419	813.1728	0
419	813.1728	0
299	2370.7857	1
533	1285.7391	0
565	3352.7973	1
419	813.1728	0
331	1818.3097	0
412	3450.0000	1
412	3450.0000	1
533	1285.7391	0
565	3352.7973	1
500	4146.3840	1
452	2073.1920	1
331	1818.3097	0
500	4146.3840	1
452	2073.1920	1
484	2073.1920	1
460	1818.3097	1
379	1818.3097	0
435	2571.4782	1
379	1818.3097	0
484	2073.1920	1
395	2073.1920	1
435	2571.4782	1
395	2073.1920	1
435	2571.4782	1
460	1818.3097	1
420	575.0000	1
484	2073.1920	1
444	1150.0000	0
379	1818.3097	1
420	575.0000	0
420	575.0000	0
428	1285.7391	1

Each row is a trial. We'll focus on two independent variables:

amp: the amplitude of the current

dist: the distance between the pairs of electrodes (μm)

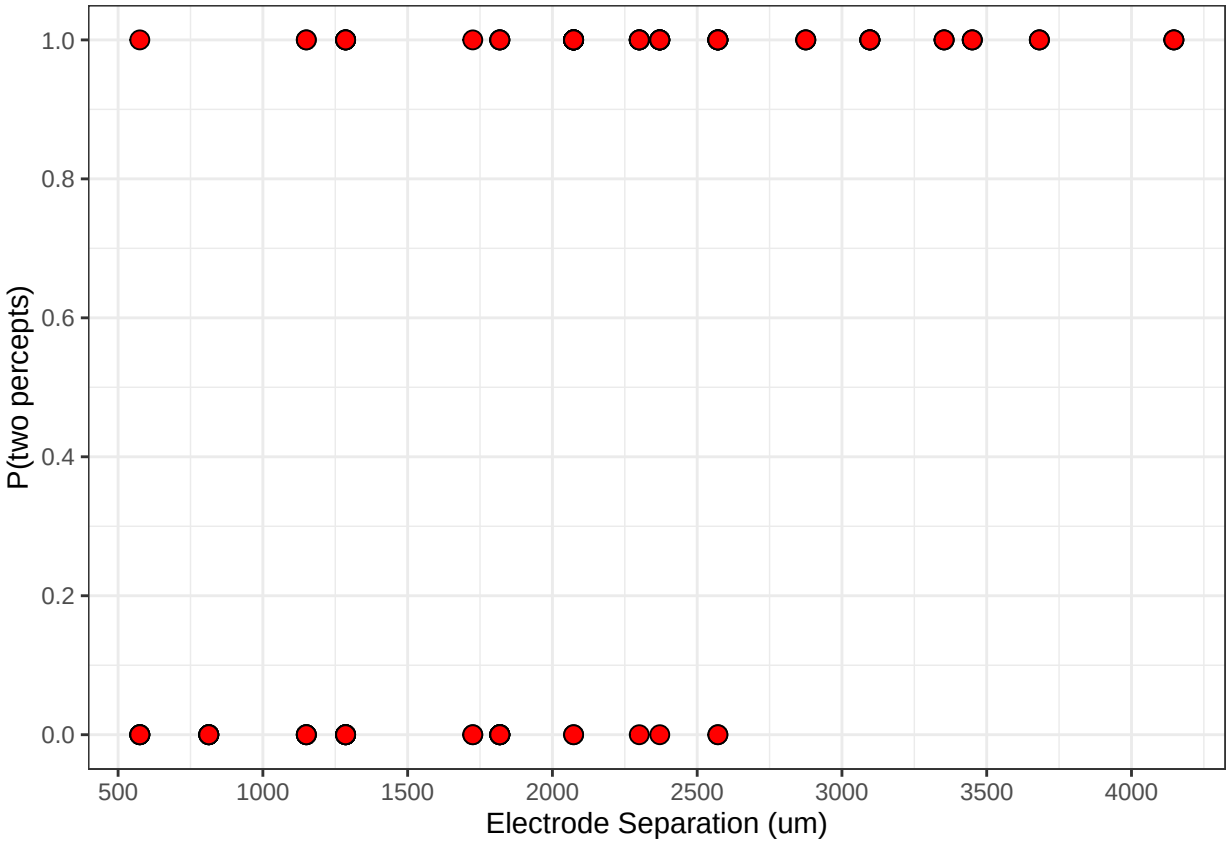
and the dependent variable:

resp: whether the patient saw one percept (resp=0) or two percepts (resp=1)

We'll be studying the effect of distance on the patient's reported percepts.

The easiest way to plot the data is to place distance on the x-axis and the patient's response on the y-axis. Remember, 0 is for trials when they saw one percept and 1 is when they saw two percepts.

```
p <- ggplot(myData, aes(x=dist, y=resp)) +
  scale_y_continuous(breaks = seq(0,1,by = 0.2)) +
  scale_x_continuous(breaks = seq(0,5000, by = 500)) +
  labs(x = 'Electrode Separation (um)',y='P(two percepts)') +
  theme_bw()
p +   geom_point(shape =21, size = 3, fill = 'red')
```

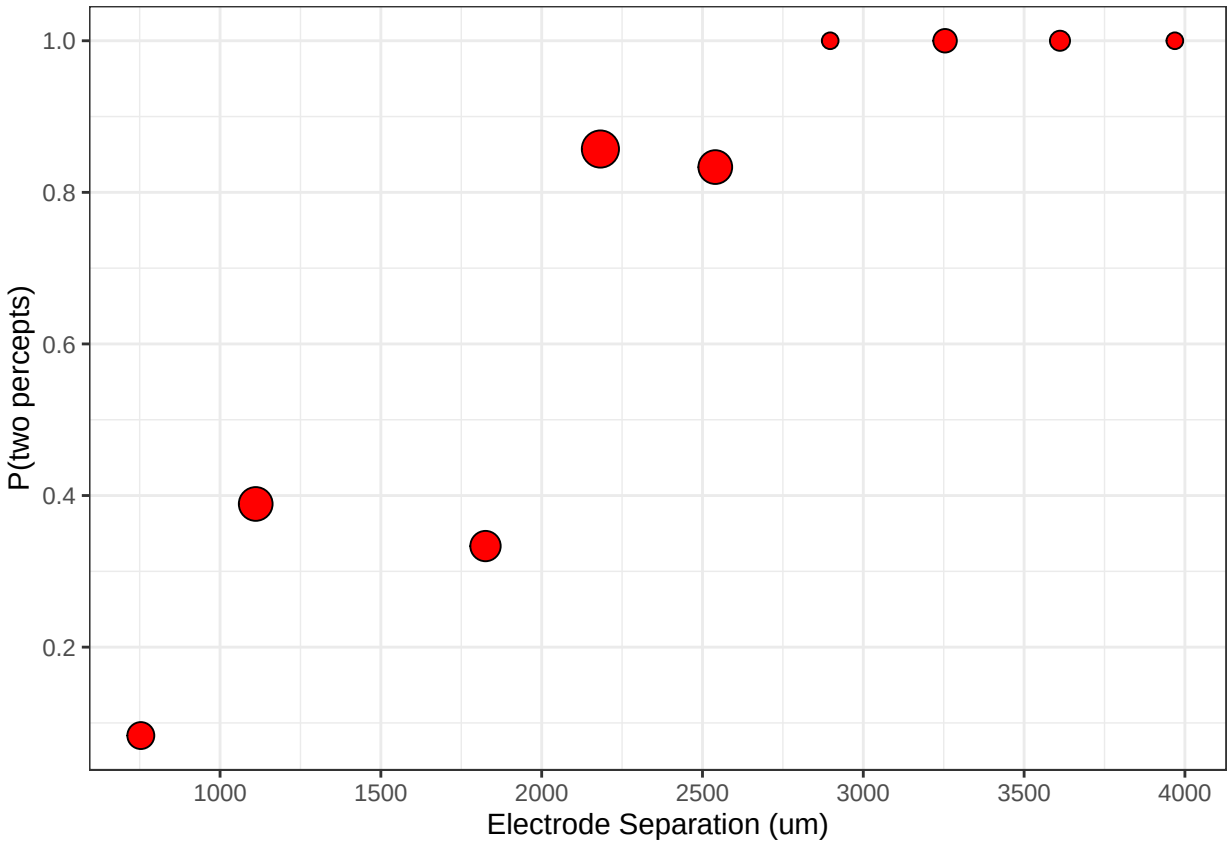


Since the response is binary, it's hard to see what's going on here, but you can probably tell that there are more '1' responses at larger distances.

A better way to visualize binary data is to bin the distances into class intervals and take the average number of 1's in each bin. Here's one way to do this using a for loop:

```
# define the bin boundaries
nBins <- 10
bins <- seq(min(myData$dist),
            max(myData$dist)+1,
            length.out = nBins+1)
```

```
# zero out the vectors to be filled in the loop
Presp <- numeric(nBins)
nresp <- numeric(nBins)
for (i in 1:nBins) {
  id <- myData$dist >= bins[i] & myData$dist < bins[i+1]
  nresp[i] <- sum(id)
  Presp[i] <- mean(myData$resp[id])
}
binCenter = (bins[1:nBins]+bins[2:(nBins+1)])/2
binned_data <- data.frame(distance = binCenter,Presp)
p <- p+ geom_point(aes(x=distance,y=Presp),data = binned_data, shape = 21,size= (nresp)/5+2,fill = "red")
plot(p)
```



Now you can see a clear effect of distance on the percept. Electrodes that are farther apart from each other have a higher probability of producing two percepts. I've scaled the size of the points to represent the number of trials that fall into each bin. Some bins may have no data, which is why you might see a 'missing values' warning from ggplot if you run this yourself.

If this were linear regression, we could study the effect of distance on the responses using 'lm' like this:

lm(resp ~ dist , data = myData)

This would fit a straight line through the data. This is clearly not appropriate. For one thing, the linear prediction can go below zero and above 1, and that's not right. But be careful - R will let you run linear regression on a binary dependent variable, and the results might look sensible.

Notice how the data points rise up in a sort of S-shaped fashion. Logistic regression is essentially fitting an S-shaped function to the binary data and running a hypothesis test on the best-fitting parameters.

Logistic regression differs from linear regression in two ways. The first difference is that while linear regression fits data with straight lines, logistic regression uses a specific parameterized S-shaped function called the 'logistic

function' which ranges between zero and one. The second difference is the 'cost function'. While linear regression uses sums-of-squared error as a measure of goodness of fit, logistic regression uses a cost function called 'log-likelihood'.

24.2 The logistic function

To predict the probability of a '1', the model first assumes a linear function of the independent variable (or variables), just like for linear regression. For a single factor:

$$y = \beta_0 + \beta_1 x$$

x is the independent variable, which is distance in our example, and β_0 and β_1 are free parameters that will vary to fit the data.

The result, y, is then passed through the S-shaped logistic function which produces the predicted probability that the dependent variable is a '1'.

$$p = \frac{1}{1 + e^{-y}}$$

Notice that these two parameters β_0 and β_1 are a lot like the intercept and slope in linear regression. They have similar interpretations for logistic regression.

Why do we use the logistic function? There are actually many parameterized S-shaped functions that would probably work just as well, such as the cumulative normal and the Weibul function.

24.3 logistic function parameters: the odds ratio and log odds ratio

One good reason is that the logistic function has special meaning in terms of the 'log odds ratio'.

If p is the probability of getting a '1', then (1-p) is the probability of getting a '0'. The odds ratio is the ratio of the two:

$$\frac{p}{1-p}$$

The log odds ratio is the log of the odds ratio (duh)

$$log(\frac{p}{1-p})$$

The odds ratio of getting heads on an unbiased coin flip is 1. The odds of rolling a 1 on a six sided die is 1/5 (sometimes read as 'one to five').

With a little algebra, you can show that the formula above can be worked around so that:

$$y = \beta_0 + \beta_1 x = log(\frac{p}{1-p})$$

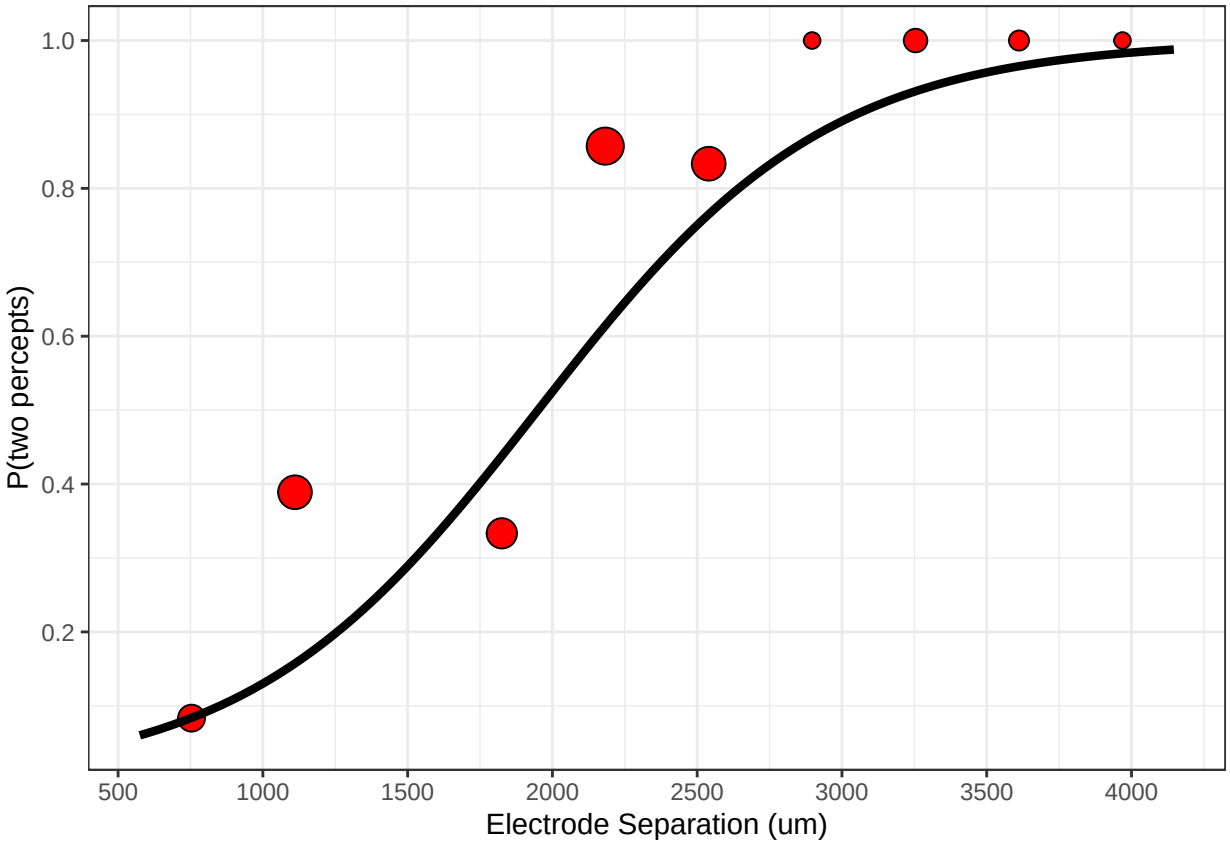
In words, logistic regression is like linear regression after converting your data into a log odds ratio.

The parameter β_0 predicts the log odds ratio when $x = 0$. For our data, patients should always see one percept when the distance is zero. If we assume that the probability of reporting two percepts is 2% with zero distance, then $\beta_0 = log(\frac{0.02}{1-0.02}) = -3.9$

The parameter β_1 determines how much the log odds ratio increases with each unit increase in x. Positive values of β_1 predict higher probabilities of '1' with increasing x (like our data). Negative values of β_1 predict that the probability of a 1 decreases with x. We'll start with a value of $\beta_1 = 0.002$.

Here is a plot of the binned data with the smooth parametrized logistic function:

```
beta_0 <- -3.9
beta_1 <- 0.002
nPlot <- 101
# x-axis values
xPlot <- seq(min(myData$dist),
             max(myData$dist),
            (max(myData$dist)-min(myData$dist))/nPlot)
# logistic function
y<- beta_0 + beta_1*xPlot
yPlot <- 1/(1+exp(-y))
plot_data <- data.frame(x = xPlot,y=yPlot)
p + geom_line(data = plot_data,aes(x=x,y=y), linewidth = 1.5 )
```



This seems like a good start, but probably not the best prediction. Logistic regression is all about finding the best parameters that predict the data. What does 'best' mean? We need to define a measure of goodness of fit - called a 'cost function' and then find the parameters that minimize it.

24.4 The log-likelihood cost function

For linear regression the cost function is the sum if squares of the deviation between the data and the predictions. But a least squares fit is not appropriate for binary data, even if we have an S-shaped prediction. Instead, we'll define a new cost function called 'log-likelihood'.

'likelihood' and probability are often used interchangeably, but they're not quite the same thing. Probability is the proportion of times an event will occur. In our example, the logistic function predicts the probability that a patient will report '2' as a function of the distance between electrodes. In this context, probability about predicting the binary dependent variable as we vary the independent variable.

Likelihood is about how well the data is predicted as we vary the model parameters. Likelihood is also a probability

- it's the probability that you'll obtain a given data set given a specific model.

Here's how the likelihood of obtaining our data is calculated from the parameters $\beta_0 = -3.9$ and $\beta_1 = 0.002$.

Consider the first few trials in our experiment:

Table 24.2:

dist	resp	predicted
1818.3097	0	0.4345
2370.7857	0	0.6988
3352.7973	1	0.9430
2875.0000	1	0.8641
2875.0000	1	0.8641
2073.1920	1	0.5613
2370.7857	1	0.6988
3450.0000	1	0.9526
2300.0000	1	0.6682
1285.7391	0	0.2094
2300.0000	0	0.6682
2875.0000	1	0.8641
4146.3840	1	0.9878
2300.0000	1	0.6682
813.1728	0	0.0933
813.1728	0	0.0933
2370.7857	1	0.6988
1285.7391	0	0.2094
3352.7973	1	0.9430
813.1728	0	0.0933
1818.3097	0	0.4345
3450.0000	1	0.9526
3450.0000	1	0.9526
1285.7391	0	0.2094
3352.7973	1	0.9430
4146.3840	1	0.9878
2073.1920	1	0.5613
1818.3097	0	0.4345
4146.3840	1	0.9878
2073.1920	1	0.5613
2073.1920	1	0.5613
1818.3097	1	0.4345
1818.3097	0	0.4345
2571.4782	1	0.7761
1818.3097	0	0.4345
2073.1920	1	0.5613
2073.1920	1	0.5613
2571.4782	1	0.7761
2073.1920	1	0.5613
2571.4782	1	0.7761
1818.3097	1	0.4345
575.0000	1	0.0601
2073.1920	1	0.5613
1150.0000	0	0.1680
1818.3097	1	0.4345
575.0000	0	0.0601
575.0000	0	0.0601
1285.7391	1	0.2094

For the first trial, the distance between the electrodes was 1818.309655 and the patient's response was 0. The logistic function model predicts that for this distance, the probability of the patient responding '1' (two percepts) is computed by:

$$y = \beta_0 + \beta_1 x = -3.9 + (0.002)(1818.309655) = -2.75$$

$$\frac{1}{1+e^{-y}} = \frac{1}{1+e^{-(-2.75)}} = 0.4345$$

Since the response was 0, the model predicts that the probability of this first response is $(1 - 0.4345) = 0.5655$

The response for the second trial was also 0, and the model predicts the probability of this happening is $(1-0.6988) = 0.3012$

The third trial had a response of 1, and the probability of this happening is 0.943

If we assume that the trials are independent, then the model predicts that the probability of the first three responses is the product of the three probabilities:

$$(0.5655)(0.3012)(0.943) = 0.1606199$$

We can continue this process and predict the probability - or technically the likelihood - of obtaining this whole 105 trial data set. If we define r_i as the response (0 or 1) on trial i and p_i as the predicted probability of a '1' on that trial, then the probability of our data given the model predictions can be written as:

$$likelihood = \prod_i p_i^{r_i}(1-p_i)^{1-r_i}$$

Where the π symbol means multiply. This is a clever little trick that takes advantage of the fact that a number raised to the 0th power is 1, and a number raised to the power of 1 is itself. The formula automatically performs the 'if' operation where $p_i^{r_i}(1-p_i)^{1-r_i} = p_i$ if $r_i = 1$ and $p_i^{r_i}(1-p_i)^{1-r_i} = 1 - p_i$ if $r_i = 0$

Our goal is to find the best parameters β_0 and β_1 that maximize this likelihood. For this example, the likelihood for $\beta_0 = -3.9$ and $\beta_1 = 0.002$ is an absurdly small number: $2.1841511 \times 10^{-22}$. That's because we're multiplying many numbers that are less than one together. Even a good fitting model will have likelihoods that are so small that computers have a hard time representing them accurately.

To deal with this this we take the log of the likelihood function. Remember, logarithms turn exponents in to products, and products into sums:

$$log(likelihood) = \sum_i r_i log(p_i) + (1-r_i) log(1-p_i)$$

For our data, the log-likelihood is -49.876, which is a much more reasonable number (it's negative because the log of a number less than 1 is negative). This number is as a measure of how well the logistic model fits our data assuming our first guess of parameters ($\beta_0 = -3.9, \beta_1 = 0.002$).

24.5 Finding the best fitting logistic regression parameters using R's glm function

With linear regression we found the slope and intercept parameters that minimized the sums of squared error between the predictions and the data. This was done using that psuedo-inverse trick from linear algebra.

For logistic regression, there isn't a clever trick for finding the best parameters. Instead, computers use a 'nonlinear optimization' algorithm that essentially fiddles with the parameters to find the maximum. In reality since optimization routines traditionally find the minimum of function, the best parameters for logistic regression are found by minimizing the negative of the log-likelihood function. Often you'll see the positive value - or twice the positive value - reported as the measure of goodness of fit.

The best-predicting parameters are found using R with the 'glm' function. The syntax is much like 'lm'. Don't forget the 'family = binomial' part.

```
model_dist <- glm(resp ~ dist, data = myData, family = binomial)
coef(model_dist)
```

```
##  (Intercept)         dist
## -3.713437286  0.002257211
```

The best-fitting parameters (coefficients) are $\beta_0 = -3.7134373$ and $\beta_1 = 0.0022572$.

The measure of goodness of fit is in the 'deviance' field of the output:

```
model_dist$deviance
```

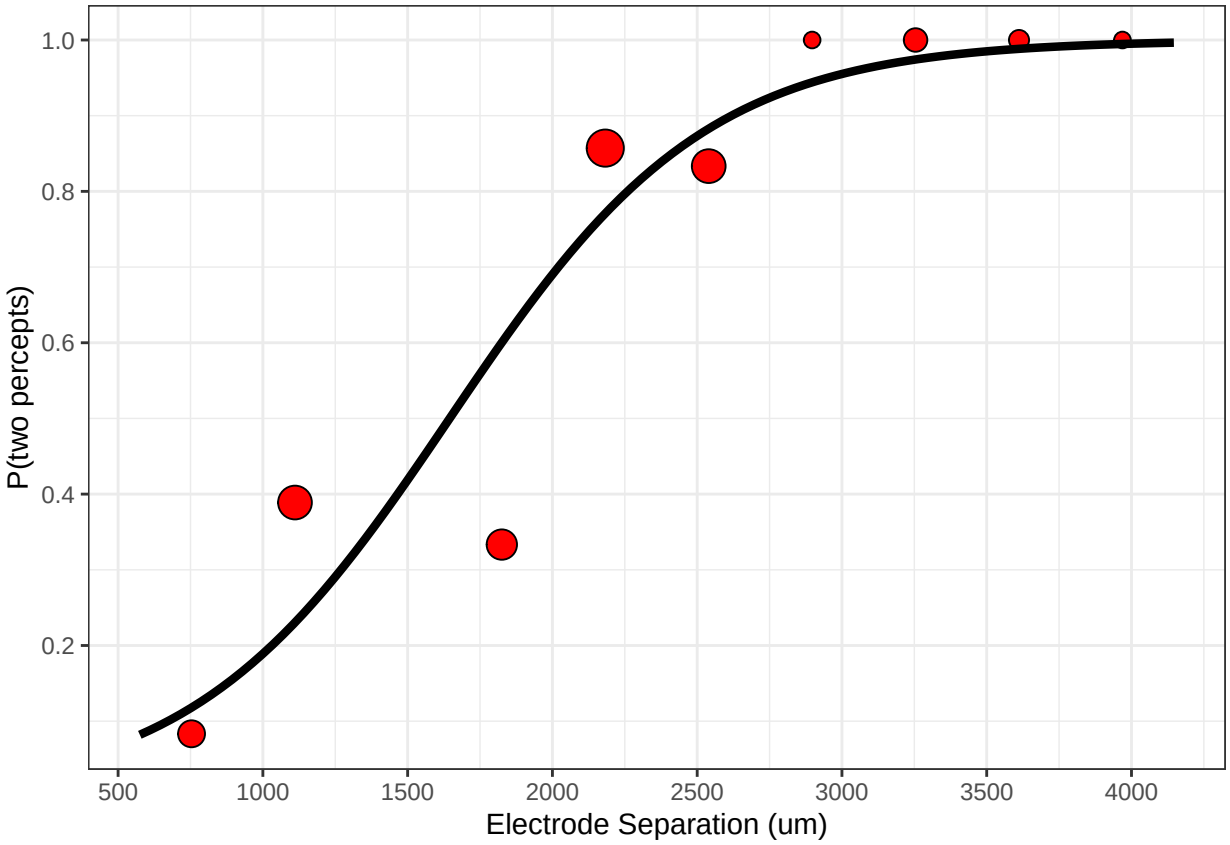
```
## [1] 92.39498
```

'deviance' is twice the negative of the log-likelihood. You'll see below why this the log-likelihood is doubled.

Halving this number: $\frac{92.3949825}{2} = 46.197$ gives us the negative of the log-likelihood. Note that this is smaller than our first guess, which was 49.876. No matter how hard you try, you'll never find a pair of parameters that make this value smaller.

Here's a plot of the binned data with the best-fitting logistical function:

```
# logistic function
y<- params[1] + params[2]*xPlot
yPlot <- 1/(1+exp(-y))
plot_data <- data.frame(x = xPlot,y=yPlot)
p <- p + geom_line(data = plot_data,aes(x=x,y=y), size = 1.5 )
plot(p)
```



The best-fitting model predicts the data nicely. You should appreciate that the binned data points are summary statistics of the actual binary data. A different choice of class intervals would produce a different set of data points.

Nevertheless, you can see how the fitting procedure 'pulled' the logistic function toward the binned data.

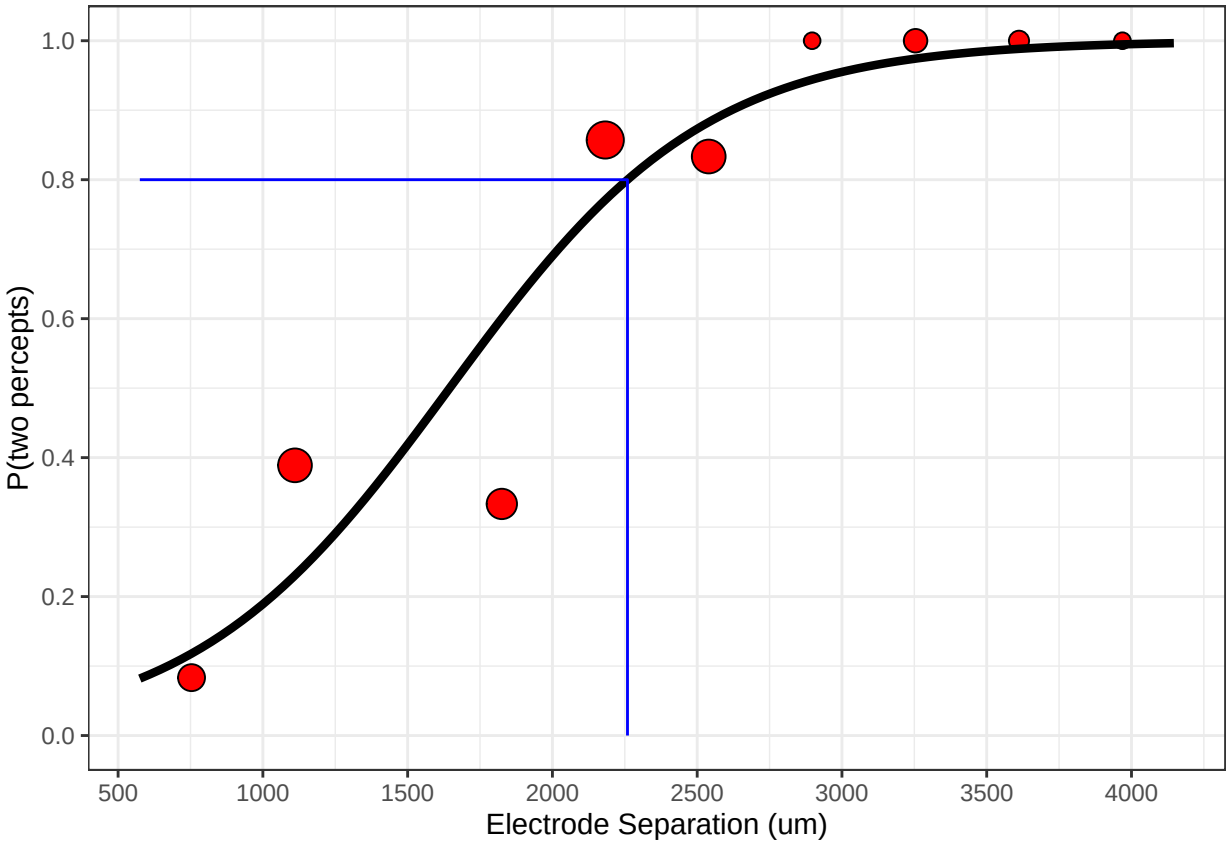
24.6 Interpreting the best-fitting parameters

In a moment we'll discuss the statistical significance of our best-fitting parameters. This will be similar to the nested model F-test discussion in the chapter on ANOVA and regression. For our example the null hypothesis is that electrode distance has no effect on the probability of seeing two percepts. Formally, this is testing the null hypothesis $H_0 : \beta_1 = 0$ because when $\beta_1 = 0$.

But before we think about p-values, let's think about what our best-fitting parameters mean. Looking at the figure above, it's very clear that distance has a strong influence on the probability of seeing two spots. What's perhaps more interesting is asking the question: "How far apart do the electrodes need to be for the subject to reliably see two spots?". To do this we need to define a 'threshold' performance level, which is typically something like $p_{thresh} =$ 80 percent of the time. From this, we can use the best-fitting logistic curve to find the distance that leads to this threshold performance level. With a little algebra:

$$thresh = \frac{log(\frac{p_{thresh}}{1-p_{thresh}}) - \beta_0}{\beta_1} = \frac{log(\frac{0.8}{1-0.8}) - (-3.71)}{0} = 2259.31$$

This means that we need two electrodes to be separated by at least 2250 micrometers to produce a percept that looks like more than one blob. You can see this by drawing a horizontal line at height pResp = 0.8 until it hits the logistic curve at distance = 2259.31 and then dropping own to the x-axis:



This distance on the retina works out to be about 8 degrees of visual angle, or about the width of about 4 fingers held out at arms-length. That's pretty big, indicating that this device probably can't produce high-resolution vision.

Turns out that Second Sight realized this too, and has given up on the Argus II, producing the ethical problem of hundreds of patients with unsupported electronics implanted in their eyes.

24.7 Testing statistical significance: the log-likelihood ratio test

Anyway, let's discuss statistical significance.

With linear regression we showed that the statistical significance of coefficients can be found using a 'nested model F test' which turned out to be numerically identical to the F-statistic from ANOVA. For logistical regression there are a few different methods for getting p-values from our coefficients.

We'll start with the 'log-likelihood ratio test'. Like the nested model F test it compares the goodness of fit between 'complete' and 'restricted' models. For our example, the complete model is the one we just fit with β_0 and $\beta 1$, and the restricted model just has the β_0 parameter. Sometimes this is called the 'null' model:

```
model_null <- glm(resp ~ 1 , data = myData,family = binomial)
model_null$deviance
```

```
## [1] 137.4463
```

If the null hypothesis is true, then the difference of the deviances for the two models (or twice the difference between the log-likelihoods) turns out to be distributed as a chi-squared statistic. The degrees of freedom is the difference between the number of parameters.

We can use 'pchis2' to get the p-value. There is one degree of freedom because the null model had just the β_0 parameter, and the full model is the null model plus the β_1 parameter.

```
chi_squared <- model_null$deviance - model_dist$deviance
sprintf('chi-squared(%d)= %5.3f',1,chi_squared)
```

```
## [1] "chi-squared(1)= 45.051"
```

```
pValue <- 1-pchisq(chi_squared,1)
sprintf('p = %g',pValue)
```

```
## [1] "p = 1.91936e-11"
```

There is a very significant effect of distance on the patient's probability of reporting 2 percepts.

R has it's own function for the log-likelihood test called 'lrtest'. If we just send in the complete model it automatically compares it to the null model:

```
 lrtest(model_dist)
```

```
## Likelihood ratio test
##
## Model 1: resp ~ dist
## Model 2: resp ~ 1
##   #Df  LogLik Df  Chisq Pr(>Chisq)
## 1   2 -46.197
## 2   1 -68.723 -1 45.051  1.919e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

You should recognize these numbers. You'll see the log-likelihood values (-46.197 and -68.723), the chi-squared value (45.051) which is twice the difference between the log-likelihood values, and the p-value for 1 degree of freedom.

24.8 Adding factors

Just as with linear regression we can fit more complicated models by adding independent variables to the logistic regression model.

A second potentially influential factor is the amplitude of the current used for stimulation, which is the field 'amp' in the data frame. Each factor has it's own coefficient, and each factor adds linearly to the predicted log odd ratio. The model incorporates the next factor β_2 like this:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

x_1 is distance, as before, and x_2 is current amplitude. As before, y is passed through the logistic function to predict the probability of getting '1'.

$$p = \frac{1}{1 + e^{-y}}$$

R's glm function incorporates addtional factors just like 'lm';

```
model_both <- glm(resp ~ dist+amp,data = myData,family = binomial)
```

To test the statistical significance of adding this second factor we use the log-likelihood test to compare this fit to the fit using the distance alone. This time we'll send in the results of both models, since we're not using the null model.

```
lrtest(model_both,model_dist)
```

```
## Likelihood ratio test
##
## Model 1: resp ~ dist + amp
## Model 2: resp ~ dist
##   #Df  LogLik Df Chisq Pr(>Chisq)
## 1   3 -44.819
## 2   2 -46.197 -1 2.756    0.09689 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

It turns out that current amplitude does not significantly improve the fit (after including distance as a factor).

Do you remember the issue with Type I and Type II ANOVA's? For unbalanced designs with multiple factors, the order that you add the factors can matter. The same is true for logistic regression. This means that if we want to test the significance of distance while including amplitude, we need to fit the model to amplitude alone first, and then compare that to the complete model:

```
model_amp <- glm(resp ~ amp , data = myData,family = binomial)
lrtest(model_both,model_amp)
```

```
## Likelihood ratio test
##
## Model 1: resp ~ dist + amp
## Model 2: resp ~ amp
##   #Df  LogLik Df  Chisq Pr(>Chisq)
## 1   3 -44.819
## 2   2 -67.711 -1 45.782  1.322e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Distance is still statistically significant - though the p-value is different than before where we simply added distance to the null model. With both amplitude an distance as factors, we should report this last p-value since it's the result with adding distance last.

You can get R to run a type II log-likelihood ratio test using the `Anova` function (with the capital `A`):

```
Anova(model_both)
```

```
## Analysis of Deviance Table (Type II tests)
##
## Response: resp
##      LR Chisq Df Pr(>Chisq)
## dist   45.782  1  1.322e-11 ***
## amp     2.756  1    0.09689 .
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Notice the p-values for dist and amp match those from our analyses when the factor is added last.

24.9 Wald test

There are other ways to calculate p-values with logistical regression. A common method is the Wald test. This is the default method in R using the 'summary' function:

```
kable(summary(model_both)$coef)
```

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-6.5436795	2.0399385	-3.207783	0.0013376
dist	0.0024604	0.0005141	4.785509	0.0000017
amp	0.0058554	0.0035981	1.627361	0.1036605

Notice that the p-values are similar to those from the log-likelihood ratio test. I'll confess that don't know exactly how to go from the raw data to the z-scores for Wald's test. I do know that Wald's test is an approximation for the likelihood ratio test. I have no idea why the more complicated Wald test is often the default for logistical regression when it's an approximation to a simpler likelihood ratio test. From my reading on the topic, we should be using the likelihood ratio test for logistical regression.

Chapter 25

Tests for Homogeneity of Variance and Normality

There are two assumptions for ANOVA that keep showing up - *homogeneity of variance* and *normality*.

Homogeneity of variance is the assumption that each population mean has the same variance - even if H_0 is false and the population means are different. Homogeneity of variance is needed so that we can average the variances from each sample to estimate the population variance. The assumption of normality means that the populations that each group is drawn from have normal distributions. Together, these two assumptions assume that for ANOVA, every sample is drawn from a normal distribution with the same population variance, even if the population means aren't the same.

This chapters covers some of the hypothesis tests for homogeneity of variance and normality. If a test is rejected, then this is evidence that your sample doesn't satisfy one of the two assumptions. If that's the case, then you might be advised to run one of the Nonparametric Tests that we cover later.

What if you fail to reject a test for homogeneity of variance or normality? You'll probably read somewhere that if your data fails to reject one of these tests, then your OK to go on with an ANOVA. But you're also told that you shouldn't interpret a null result from a hypothesis test as evidence that H_0 is true. And you can really drive yourself crazy if you start thinking about assumptions for the tests to see if you've satisfied the assumptions of the ANOVA test. What if you don't satisfy *those* assumptions? Is there a test for that? I wish I had something intelligent to say about this.

But a statistics course isn't complete without covering these tests. So here we go:

25.1 Homogeneity of Variance

The two most common tests for homogeneity of variance are Bartlett's test and Levene's test. Of the two, the *Levene's* test seems to be by far the most common so we'll only focus on this one here. Interestingly, Bartlett's test is more powerful, meaning that it is more likely to detect heterogeneity of variance if it's there. Could it be that we're favoring a less powerful test because where we kind of hope that we don't reject it? After all, if we reject a test for homogeneity of variance, then we're supposed to run a less powerful nonparametric test on our data. Nobody talks about these things.

25.1.1 Levene's test (by hand)

The formula for variance starts with calculating the deviation between scores and the mean of the scores. So if your data comes from a population that satisfies homogeneity of variance, then the deviation between each score and the mean of the group that it came from should be, on average, about the same.

This is where Levene's test starts - with a twist. R's `leveneTest` by default first calculates the deviation between each score and the *median* of the group that it came from. Let's demonstrate this with some fake data.

First, we define some population parameters - the number of groups (k), the sample sizes (n), and the population means (mu) and standard deviations (sigma). Notice that the first value of sigma is not like the others - we're violating homogeneity of variance. We should end up rejecting H_0 and conclude, correctly, that our population does not have equal variances.

```
set.seed(1)
k<-5
n <- rep(5,k)
mu <- rep(70,k)
sigma <- c(40,10,10,10,10)
```

This next step generates a single data set based on these parameters using `rnorm` to generate vectors x with the associated sample size, mean and standard deviation. The data is organized in long format, holding the (fake) independent variable x, and the corresponding group name (e.g. 'group 1').

I've also included, for each score, the median for the group that the score came from and a value Z which is the absolute value of the difference between the score and it's group median `abs(x-median(x))`:

```
dat <- as.data.frame(matrix(nrow=0,ncol=3))
for (i in 1:k) {
  x <- round(rnorm(n[i],mu[i],sigma[i]),2)  # this group's sample
  dat <- rbind(dat,data.frame(x,rep(median(x),n[i]),abs(x-median(x)),
                              factor(rep(sprintf('group %d',i),n[i]))))
}
colnames(dat) <- c('x','median','Z','group')
```

The fake data looks like this:

Table 25.1:

x	median	Z	group
44.94	77.35	32.41	group 1
77.35	77.35	0.00	group 1
36.57	77.35	40.78	group 1
133.81	77.35	56.46	group 1
83.18	77.35	5.83	group 1
61.80	74.87	13.07	group 2
74.87	74.87	0.00	group 2
77.38	74.87	2.51	group 2
75.76	74.87	0.89	group 2
66.95	74.87	7.92	group 2
85.12	73.90	11.22	group 3
73.90	73.90	0.00	group 3
63.79	73.90	10.11	group 3
47.85	73.90	26.05	group 3
81.25	73.90	7.35	group 3
69.55	75.94	6.39	group 4
69.84	75.94	6.10	group 4
79.44	75.94	3.50	group 4
78.21	75.94	2.27	group 4
75.94	75.94	0.00	group 4
79.19	76.20	2.99	group 5
77.82	76.20	1.62	group 5
70.75	76.20	5.45	group 5
50.11	76.20	26.09	group 5
76.20	76.20	0.00	group 5

You don't have to follow exactly how the code works. Just understand what each column holds.

If homogeneity of variance is true, then the average value of Z should be about the same for each group. How do we test if the values of Z vary significantly across the groups? ANOVA! Levene's test is just an ANOVA on Z (`Z ~ group`):

```
anova(lm(Z ~ group,data = dat))
```

```
## Analysis of Variance Table
##
## Response: Z
##           Df Sum Sq Mean Sq F value  Pr(>F)
## group      4 1822.4  455.60  2.8151 0.05288 .
## Residuals 20 3236.8  161.84
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

25.1.2 Levene's test (using R)

R's function for Levene's test requries the 'car' package. Once loaded it works like this:

```
leveneTest(x~group,data = dat)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##       Df F value  Pr(>F)
## group  4  2.8151 0.05288 .
##       20
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The numbers match.

25.1.2.1 We actually just ran the Brown-Forsythe test

If you want to impress your friends, because R's `LeveneTest` uses the median by default, you can tell them that is actually the Brown-Forsythe test. You can run the true Levene's test this way:

```
leveneTest(x~group,data = dat, center = "mean")
```

```
## Levene's Test for Homogeneity of Variance (center = "mean")
##       Df F value  Pr(>F)
## group  4  3.4503 0.02674 *
##       20
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Using the median is apparently more robust to violations of skewness and heavy tails in your sampling distributions. You can learn a little about the difference between using the median vs. mean from Wikipeida's entry on Levene's test.

25.1.2.2 Levene's test on Levene's test

Want to lose some sleep? Check this out:

```
leveneTest(Z~group,data = dat)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##       Df F value  Pr(>F)
## group  4  2.9673 0.04477 *
##       20
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

What's this? It's a test for homogeneity of variance on Z, which is Levene's test on the ANOVA used for Levene's test on our original data. For this data set, $p<.05$, which means that Levene's test failed its own test for homogeneity. Does this mean we shouldn't used Levene's test for this data set? Now what?

25.2 Tests for Normality

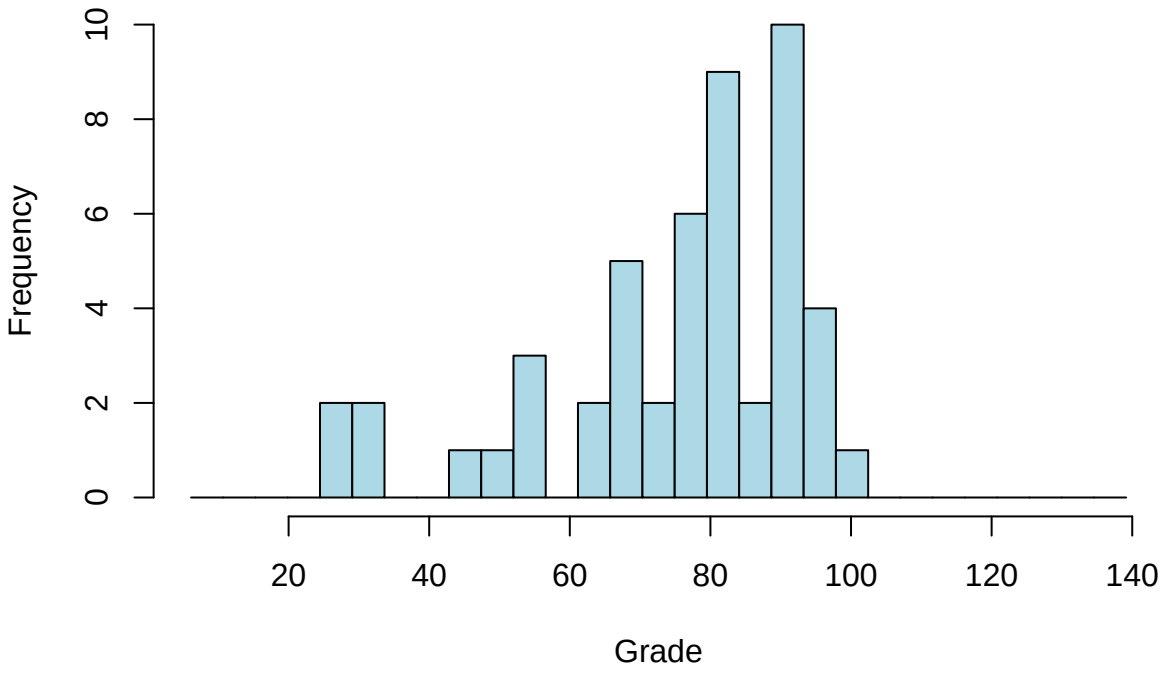
There is no single standard test the hypothesis that a sample is drawn from a normal distribution. Instead, there are a variety of tests, each having their own strengths and weaknesses. Here we'll discuss two of the most common tests, the *Lilliefors* test and the *Shapiro-Wilk* test. I chose these two because the first is one of the most common for historical reasons, and the second is considered the most powerful (most likely to correctly detect a deviation from normality).

There are many others, including specific tests for skeweness and/or kurtosis. I found a good pdf on this from NCSS.com that briefly covers a variety of tests. Here's a link to the pdf which I've copied to the course website: http://courses.washington.edu/psy524a/pdf/Normality_Tests.pdf

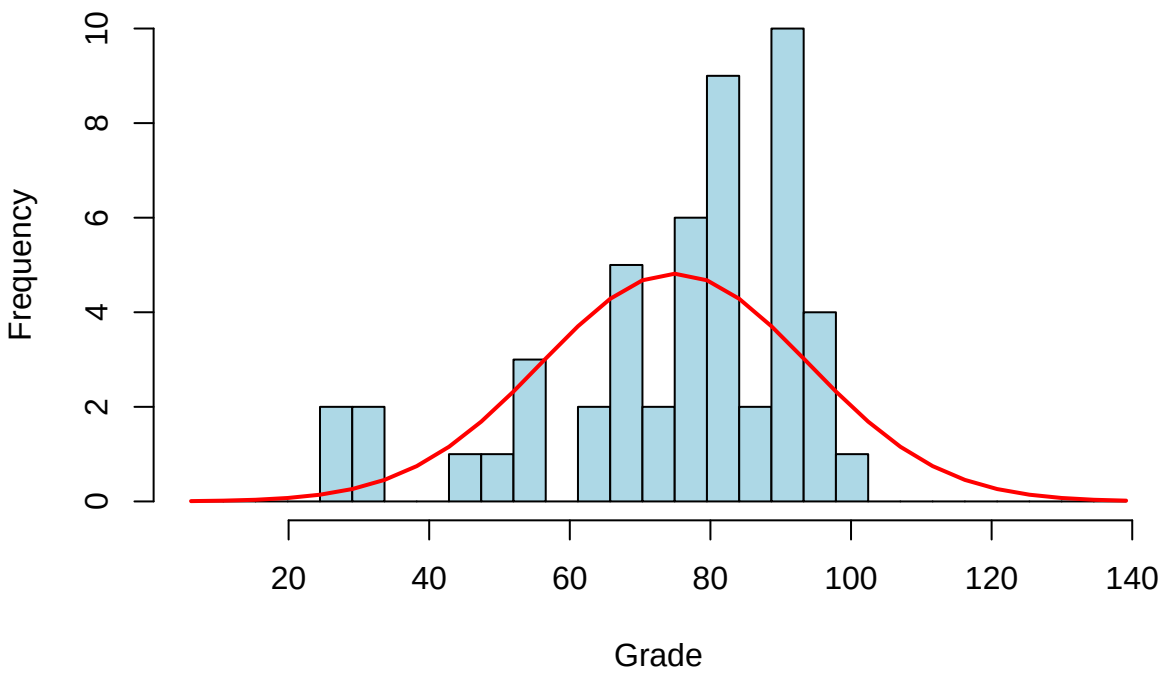
We'll use as an example a made-up sample of 50 Grades which can be loaded into R like this:

```
orig.data<-read.csv("http://www.courses.washington.edu/psy524a/datasets/Grades.csv")
# Pull out the single variable
grades <- orig.data$Grade
```

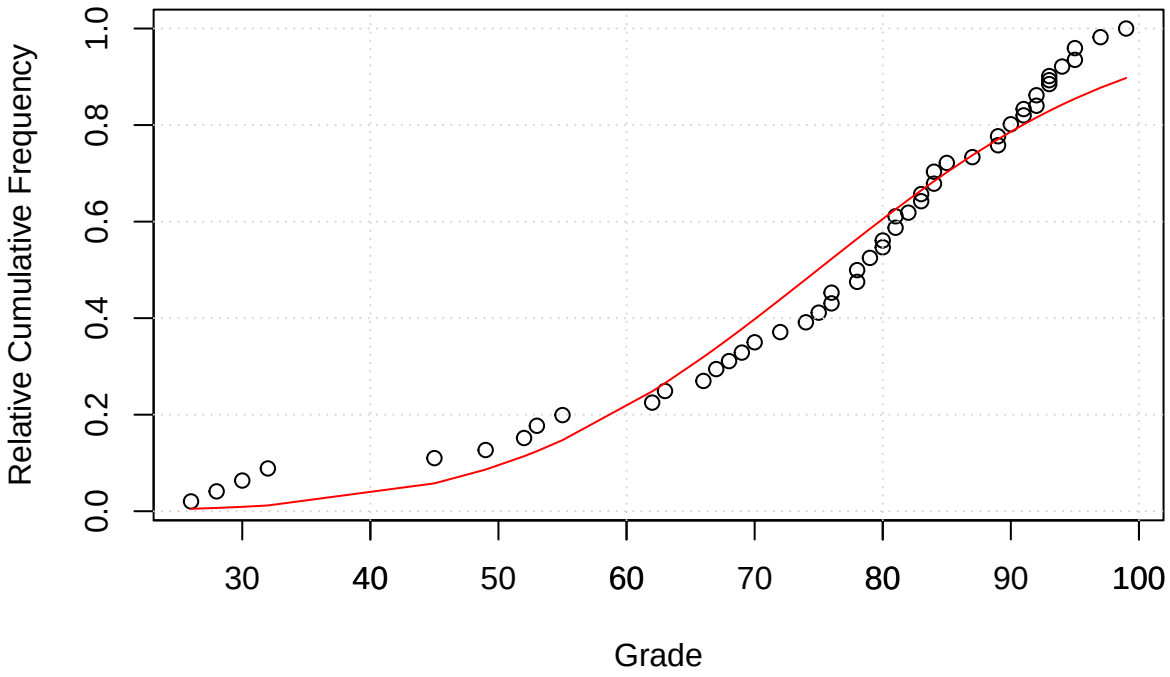
Before we run any tests for statistical significance, we should first look at the distribution of scores as a histogram. Often simply viewing the shape of the distribution will tell you enough. For this example it looks like our grades don't seem to be distributed as a familiar bell-shaped curve. Instead, it looks negatively skewed.



All tests for normality compare the distribution of your data to a normal distribution having the same mean and standard deviation of your sample. The actual mean of our scores is 74.9 and the standard deviation is 19. Here is a normal distribution with this mean and standard deviation drawn on top of our histogram:



A common way to compare the distribution of your data to a normal distribution is to compare *relative cumulative distributions*. As we've discussed before (see section 2.6), relative cumulative distributions are the cumulative sum of the histogram as you sweep from left to right, normalized so they sum to 1. Here are the corresponding relative cumulative distributions for the data and the normal curve:

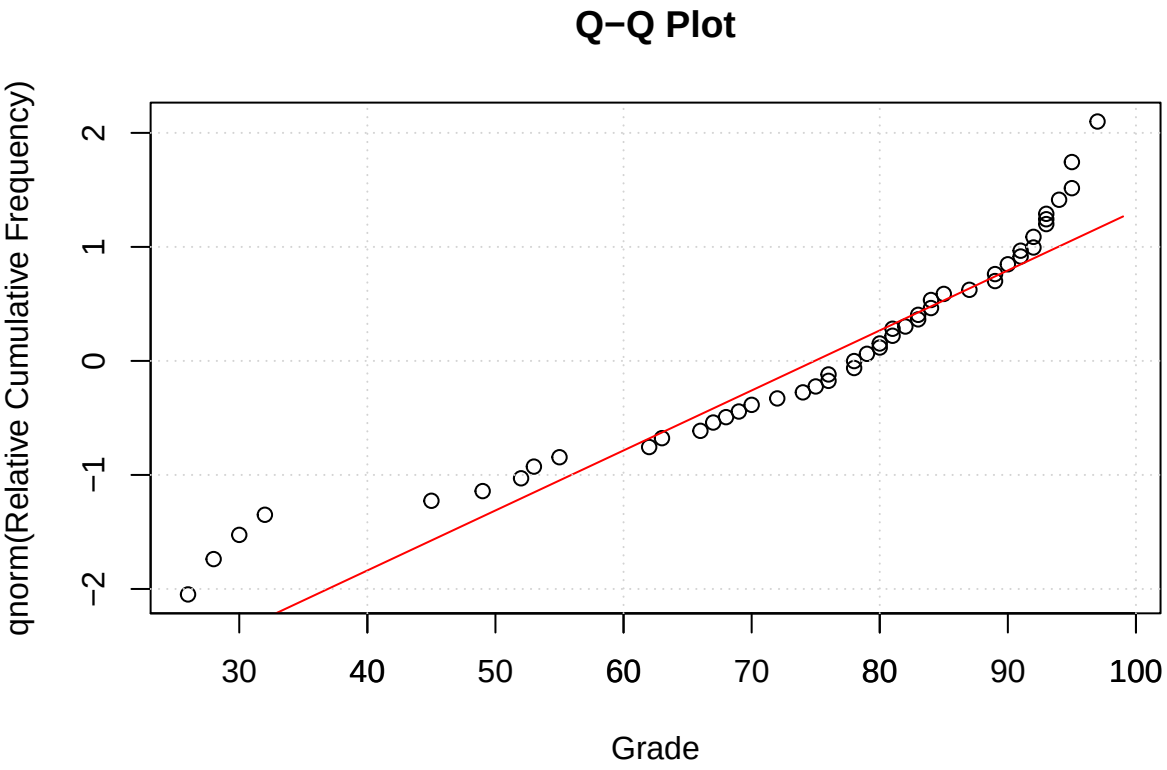


The closer the data points fit the red curve, the closer the data is to normal.

25.2.1 Q-Q Plot

It's common to convert these cumulative frequency into z-scores by passing them through the inverse of the normal distribution (`qnorm` in R). This generates what's called a *Q-Q plot*, where 'Q' stands for 'quantile'. This transforms the values on the y-axis in the previous plot to z-scores, so that for example a value of .05 becomes -1.6449, 0.5 becomes 0, and .95 becomes 1.6449.

This transformation turns the S-shaped cumulative normal into a straight line, which can be more easily compared to the data:



25.2.2 Lilliefors Test

Formally known as the *Kolmogrov-Smirnov test with the Lilliefors correction*, this is a hypothesis test on the maximum deviation between the cumulative frequency data and the corresponding curve for the normal distribution.

Lilliefors test is an extension on the *Kolmogrov-Smirnov*, or *KS* test which is a test to determine if two distributions came from the same population. However, as a test for normality the *KS* test is biased because the normal distribution we're comparing our data to is determined by the mean and standard deviation of our data. So the fit will generally be too good. We really should use the means and standard deviations of the population parameters from the null hypothesis, which we don't have.

As result, the p-value from the KS test for normality will be too large because a normal distribution based on the sample mean and standard deviation of the data will always be a better fit to the data than one based on the population parameters.

This bias can be fixed with the *Lilliefors Significance Correction* (the default with SPSS). There is no known formula for this corrected distribution. Tables are based on Monte Carlo simulations.

R has a function called `lillie.test` in the package *nortest*. Here's how to run Lilliefors test on our example data:

```
lillie.out = lillie.test(grades)
sprintf('Lilliefors test for normality, p = %5.6f',lillie.out$p.value)
```

```
## [1] "Lilliefors test for normality, p = 0.010845"
```

According to the Lilliefors test, we should reject the null hypothesis that this sample came from a normal distribution.

25.3 The Shapiro Wilk test

The Lilliefors test has the advantage of simplicity - it's easy to understand the statistic in terms of the fit of an observed to an expected cumulative frequency. However, it has been criticized as not having sufficient power for

detecting violations of normality. Later in this chapter we'll demonstrate this lack of power.

Instead, some statisticians prefer the *Shapiro-Wilk* test. The intuition is that the statistic returned from this test, W, is related to the correlation of the data in the Q-Q plot. That is, it can be thought of as a test of how well the Q-Q plot is fit by a straight line. The math behind it isn't very intuitive, and the statistic W doesn't come from one of our parametric friends (e.g. χ^2, t or F). For a deeper dive, see this reference.

R's function for the Shapiro Wilk test is called `shapiro.test` which requires loading the 'dplyr' package. Here's how it works:

```
shapiro.out <- shapiro.test(grades)
print(sprintf('Shapiro-Wilk test for normality, p = %5.6f',shapiro.out$p.value))
```

```
## [1] "Shapiro-Wilk test for normality, p = 0.000138"
```

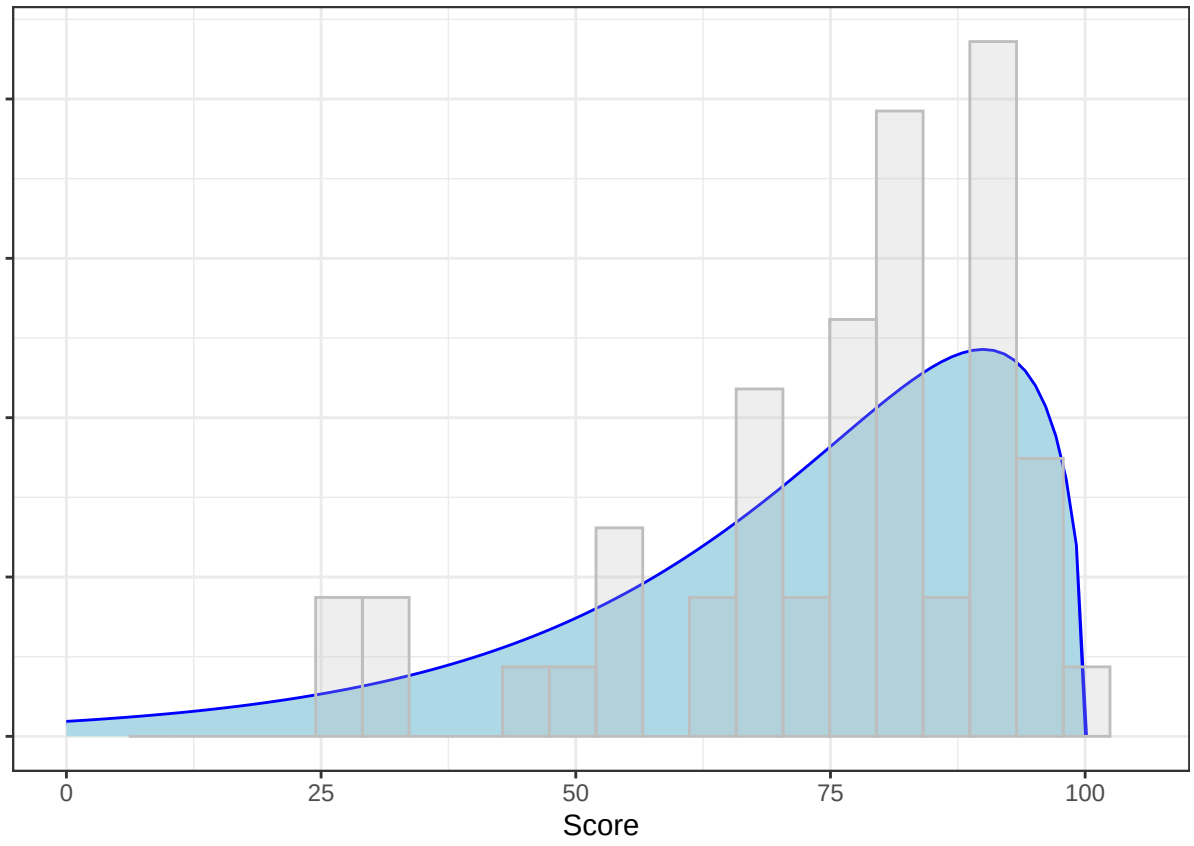
For this sample of scores the p-value for the Shapiro-Wilk test is way smaller than for the Lilliefors test.

This leads to a dilemma - with such large differences in p-values, which test should you use? After perusing many websites and textbooks, the common thread is that you should eyeball the data via histograms and Q-Q plots. I know that's not a very satisfying answer. Statisticians seem to be saying that non-normal distributions are like Supreme Court Justice Potter Stewart's famous 1964 test for obscenity in pornography - "I know it when I see it."

It turns out that both hypothesis tests led to the correct decision. We know this because I made this data of test scores by drawing 50 values from a χ^2 distribution with three degrees of freedom, multiplying the values by 10 and subtracting it from 100:

```
100-round(10*rchisq(50,3))
```

Here's the actual probability distribution of the population that our sample was drawn from along with the histogram of the sample:



This is not normal.

25.3.1 Power of tests for normality

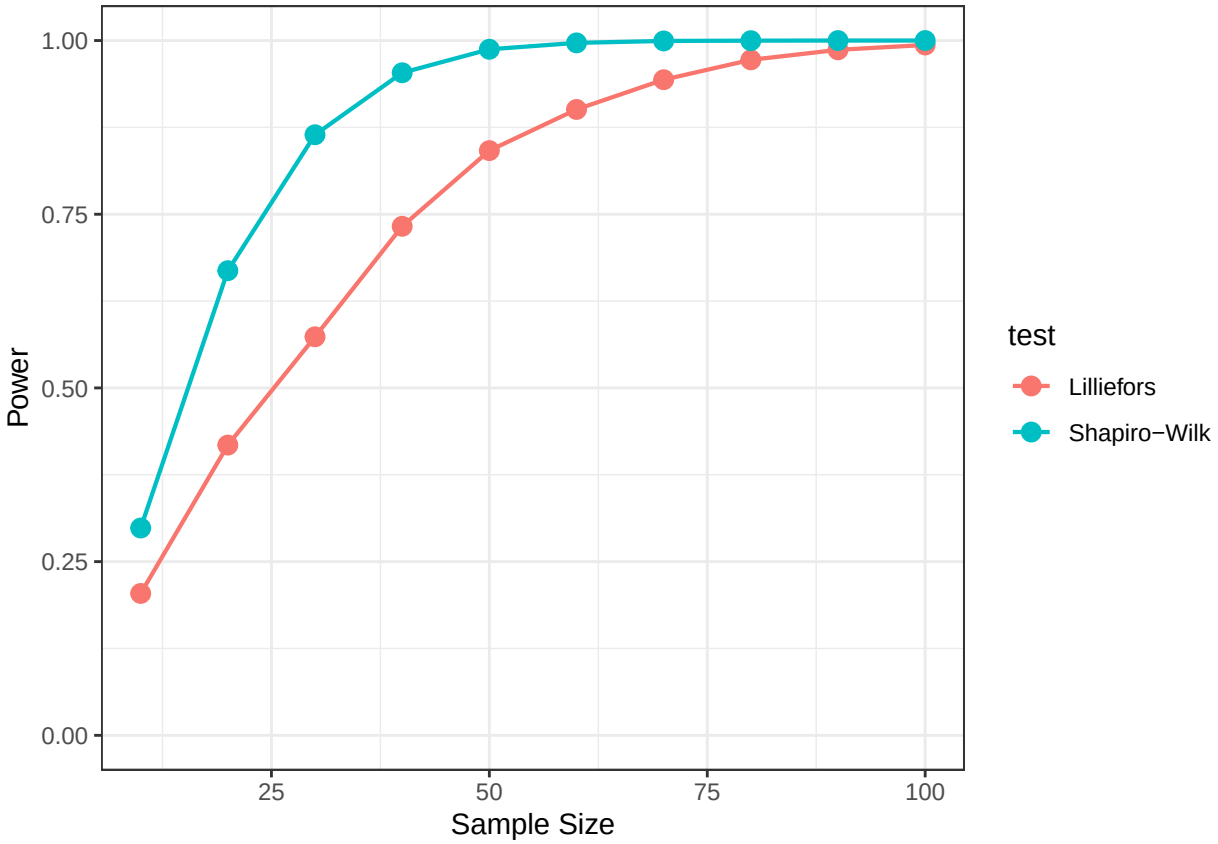
To my knowledge, there isn't a function that tests the power for these test for normality. In fact, there isn't even a measure for effect size. But never fear - we know how to estimate power without a function (or a math education). Simulation!

The following chunk of code generates multiple random samples from the distribution described above (which is not normal). This is done over a range of sample sizes. For each sample, both the Lilliefors test and the Shapiro-Wilk test is run and the p-values are stored:

```
# Simulate how power increases with sample size
n <- seq(10,100,10)  # list of sample sizes
nReps <- 5000        # number of simulations for each sample size
alpha <- .05
# zero out the things
p.lillie <-  matrix(data=NA,nrow=length(n),ncol=nReps)
p.shapiro <-  matrix(data=NA,nrow=length(n),ncol=nReps)
for (i in 1:length(n)) {
  for (j in 1:nReps) {
    x <- 100-round(10*rchisq(n[i],3))
    # x <- rnorm(n[i])  # use this to test type 1 error rate
    lillie.out = lillie.test(x)
    p.lillie[i,j] <- lillie.out$p.value
    shapiro.out <- shapiro.test(x)
    p.shapiro[i,j] <- shapiro.out$p.value
  }
}
```

After the simulation is run, the following chunk plots the proportion of rejected p-values for both tests across the range of sample sizes:

```
power.lillie <- rowMeans(p.lillie<alpha)
power.shapiro <- rowMeans(p.shapiro<alpha)
dat <- rbind(
  data.frame(n=n,power = power.lillie,test = rep('Lilliefors',length(n))),
  data.frame(n=n,power = power.shapiro,test = rep('Shapiro-Wilk',length(n))))

ggplot(data = dat,aes(x = n,y=power,group = test)) +
  geom_line(aes(color = test),size = .75) +
  geom_point(aes(color = test),size = 3) +
  ylab('Power') +
  ylim(0,1) +
  xlab('Sample Size') +
  theme_bw()
```



You'll notice a couple things from this simulation. First, power increases with sample size - which is always the case. Second, the Shapiro-Wilk test has more power than the Liliefors test. This is consistent with our original samle which showed a smaller p-value for the Shapiro-Wilk test than for the LIlllliefors test.

If you re-run the analysis with H_0 as true (drawing from a normal distribution) you'll see that the rejection rate hovers around 0.05 for both tests across all sample sizes, which is good.

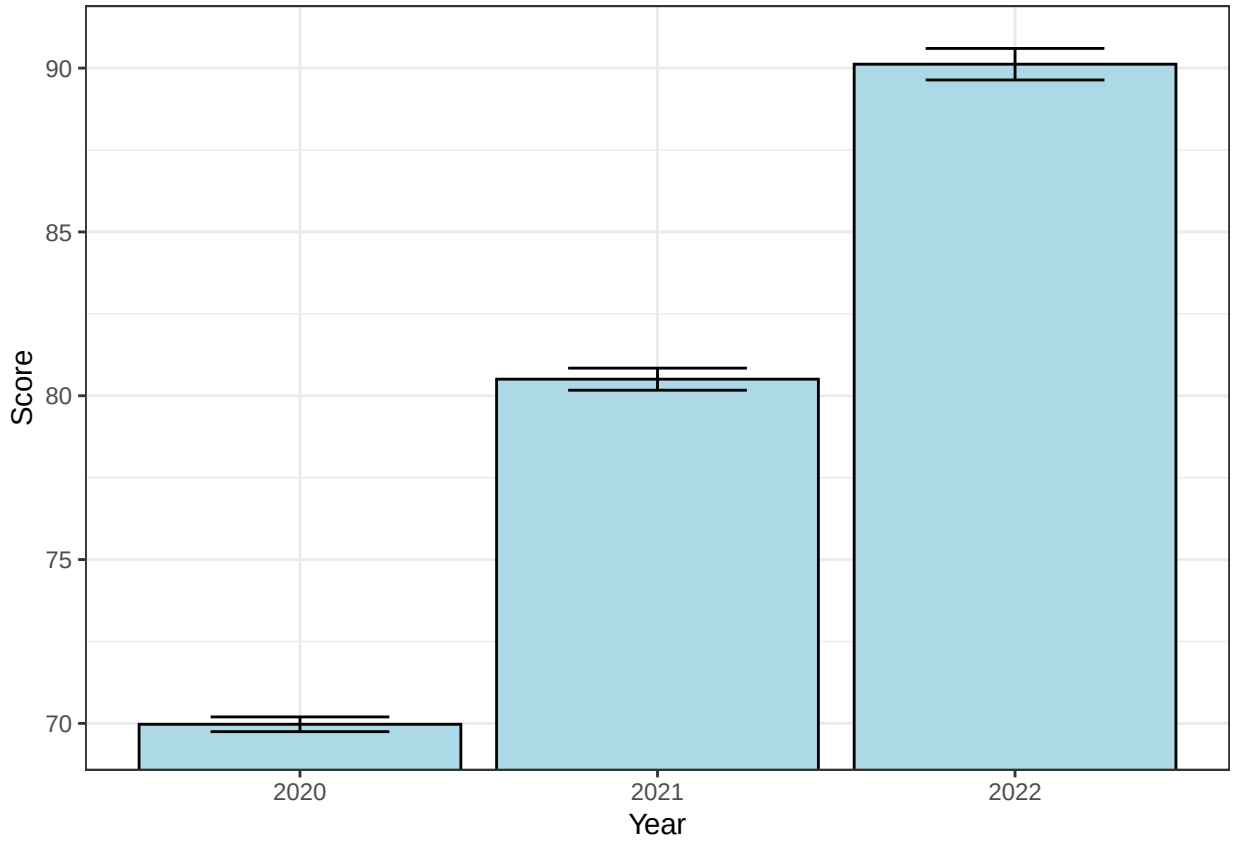
25.4 Why be normal?

It is a common misconception that your dependent variable in a hypothesis test, like an ANOVA or t-test, needs to be normal to make the test valid. There are two problems with this.

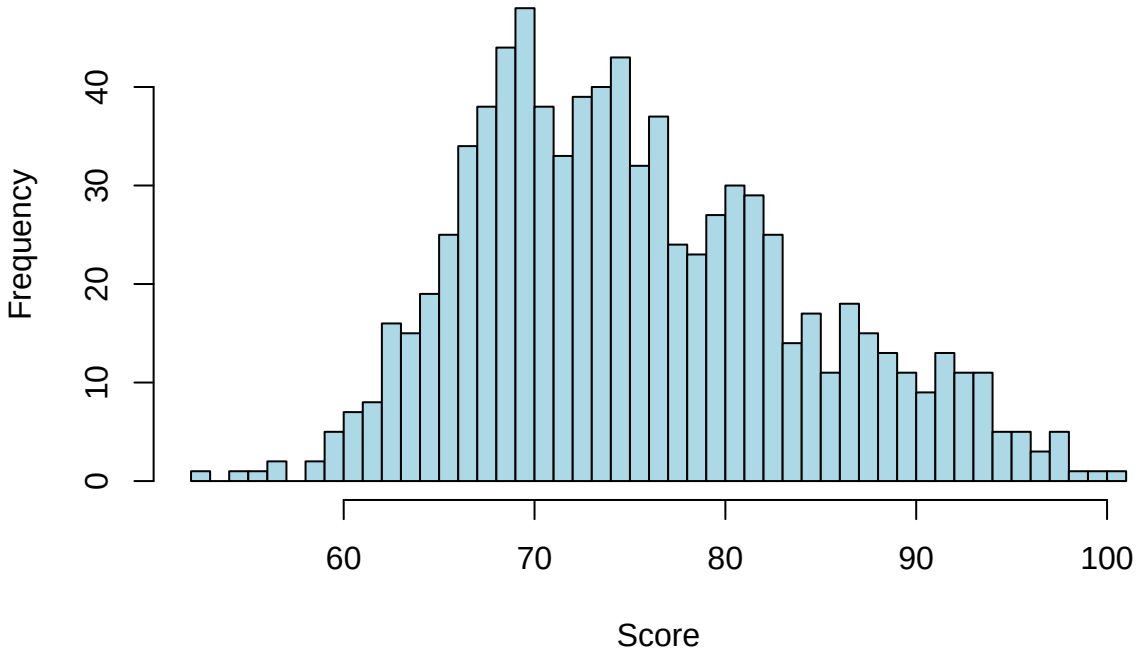
The first is that if we're running parametric tests on our data, our assumptions require that the *means* that we're measuring are coming from a population that is normally distributed. As we all know now, with a large enough sample, the Central Limit Theorem comes to the rescue. So if we're running a test on means (e.g. t-test or ANOVA) with a sample size of 20 or more, then the assumption of normality for the distributions of means is pretty safe.

Another issue with testing normality on your dependent variable is that what's actually important is that the *residuals* after fitting your model to the data are normally distributed.

Consider a one-factor fixed-effect ANOVA for exam scores over time. There are 500, 250, and 100 students that took exams in each of the years 2020, 2021, and 2022. Here's a plot of the means and standard errors:



Consider the overall distribution of our dependent variable, which a histogram of all test scores across all years:

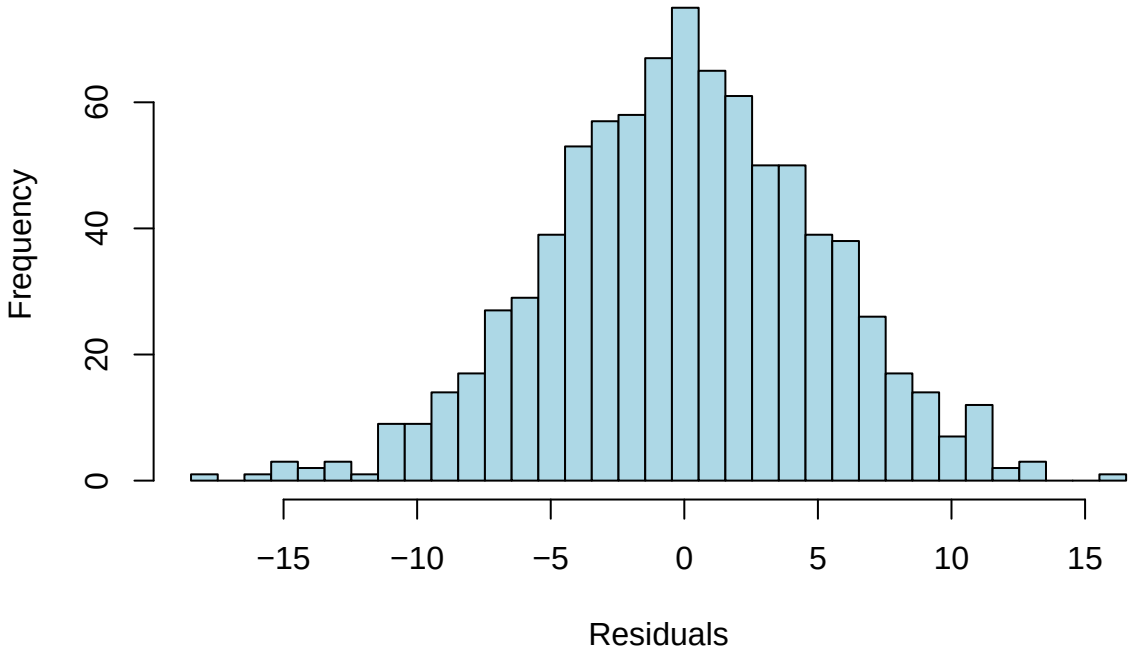


It looks like we're positively skewed. In fact, a Shapiro-Wilk test gives us a p-value of $p < 0.0001$.

But it shouldn't be surprising that the distribution of all scores is not normal. The overall distribution of scores is the combination of three separate distributions around the three group means. If the null hypothesis is false, and the means are different, then we would in fact *expect* a non-normal distribution of our dependent variable.

Instead, what's important to satisfy the conditions of our hypothesis test is that the *residuals* are normally distributed. Remember, residuals are the deviations between the model predictions and the data. In the case of ANOVA, the model predictions are the means for each group, so the deviations are the deviations from these means, which are the deviations that contribute to SS_w.

Here's a histogram of those deviations



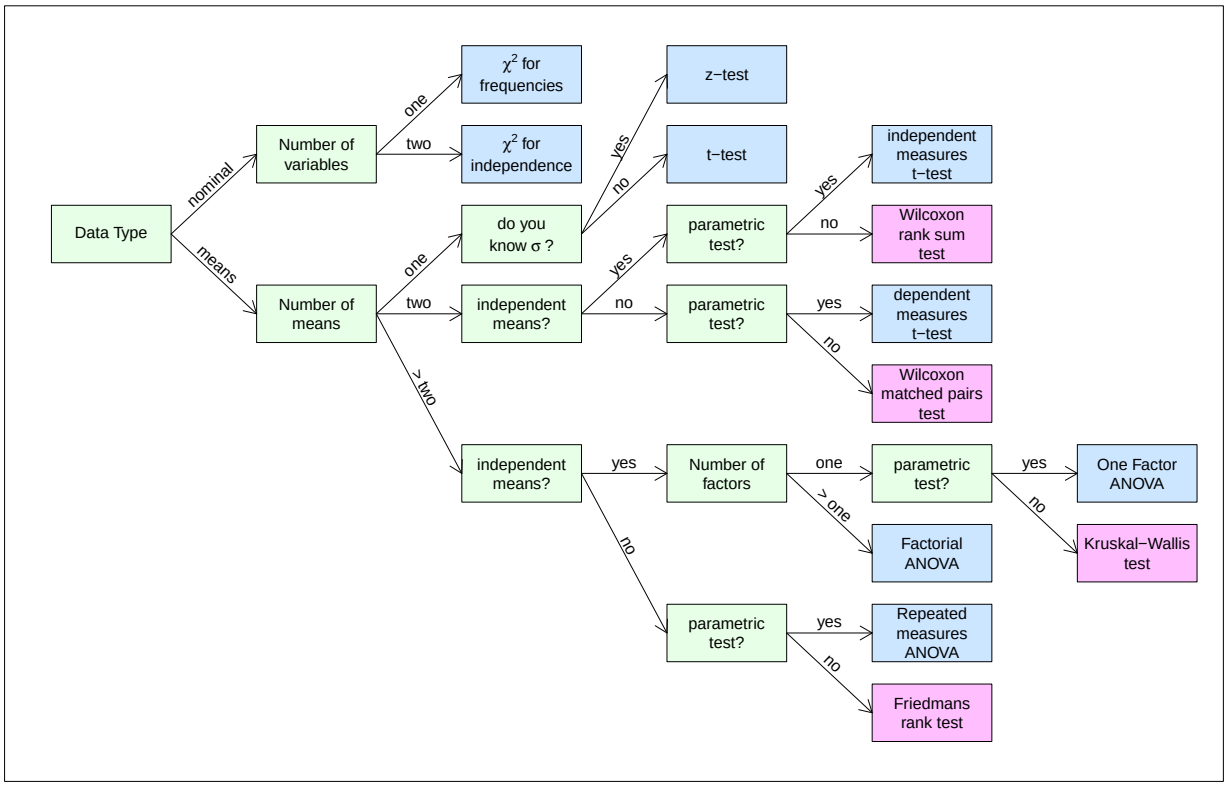
The residuals do look normally distributed. Indeed, a Shapiro-Wilk test gives us a p-value of $p = 0.7949$.

Chapter 26

Nonparametric Tests

Up to now all of the statistical tests we've done have involved calculating a statistic that we can look up in a table. These are *parametric* tests, because if the data satisfy assumptions such as normality, homogeneity of variance and sphericity, then we can assume that the computed statistic will be drawn from a known, parametric distribution such as a t, F, or χ^2. If you can't assume that your data satisfies the conditions of a parametric test you can run a nonparametric test.

Some of the nonparametric tests discussed in this tutorial are direct alternatives to parametric tests that we've covered in this class. Here's our familiar flow-chart:



The four tests in purple boxes are some of the tests we'll cover in this chapter Each of these four has a corresponding parametric test above it in the flow chart. For example the 'Friedman's rank test' is a direct substitute for the repeated measures ANOVA. This chapter walks through some of the standard nonparametric tests. It also covers alternatives such as bootstrapping, resampling and rank transformations.

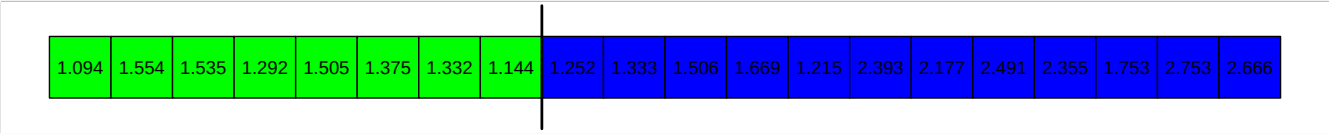
26.1 Example: RT for recognizing famous faces

Suppose we're interested in how experience affects the ability to recognize celebrity faces. We ask 8 experts and 12 novices to identify famous faces and measure their response times.

You can load in the (fake) data set here:

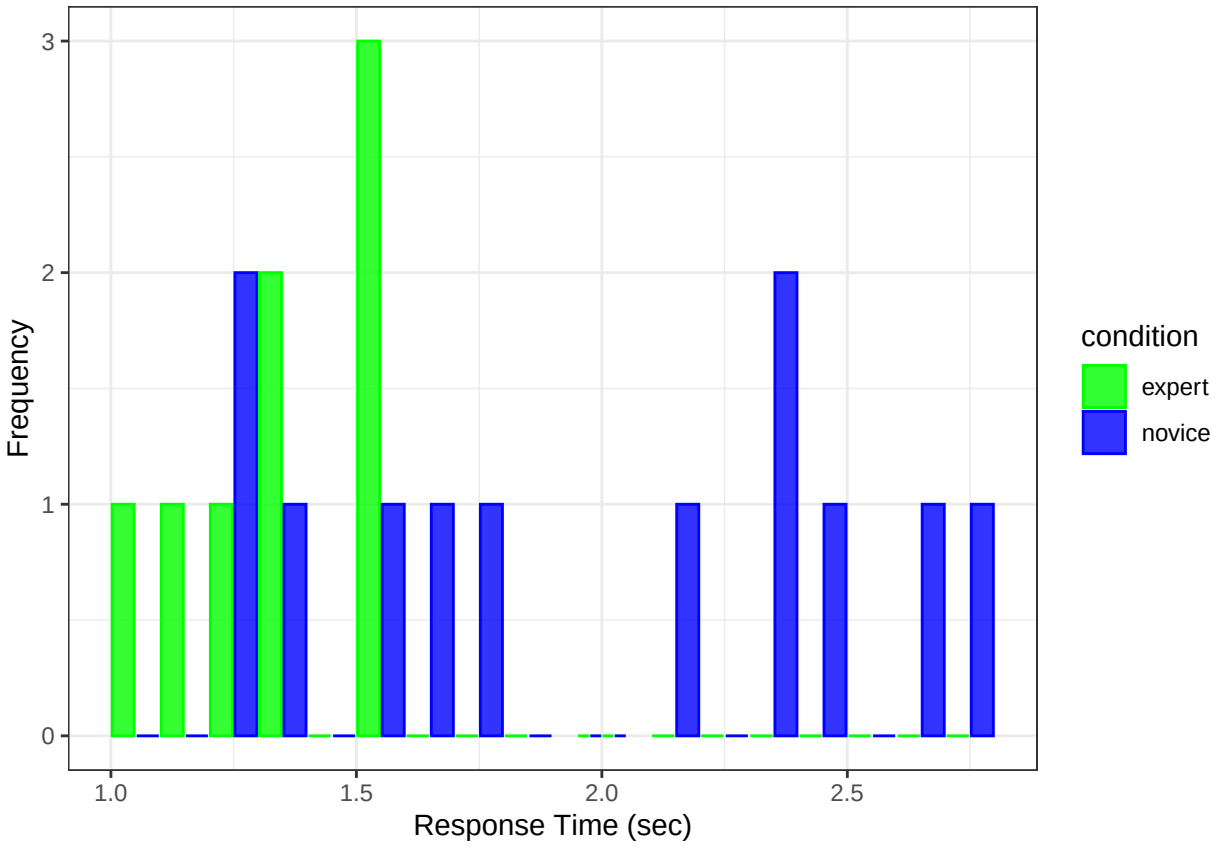
```
dat1<-read.csv("http://www.courses.washington.edu/psy524a/datasets/RTexpertnovice.csv")
```

Here are the response times (in seconds), with experts colored in green and novices colored in blue:



It's hard to tell from the raw numbers which group has a larger central tendency. Here's how to plot at a histogram of the two distributions:

```
ggplot(dat = dat1, aes(x = RT,fill = condition,col = condition)) +
  geom_histogram(position="dodge2", alpha=.8,breaks = seq(1,max(dat1$RT)+.1,by = .1))+
  scale_fill_manual(values=col)+
  scale_color_manual(values=col)+
  xlab("Response Time (sec)") + ylab("Frequency") +
  theme_bw()
```



It looks like the novices are slower.

Response times distributions are often positively skewed like this, so the median may be a better estimate of central tendency. This means that all of our statistics on means do not apply - so we can't use our standard independent measures t-test here.

In fact, a Shapiro-Wilk test will not reject the hypothesis test that these distributions came from a normal population. This is probably because our sample size is so small - so the test for normality is under powered. This is a good example of why we should be careful about 'accepting the null hypothesis' when a result is not statistically significant. I happen to know that the populations that these samples were drawn from non-normal distributions because I made the data up by sampling from shifted and scaled χ^2 distributions with $df = 3$.

The medians are:

```
medianRT <- tapply(dat1$RT,dat1$condition,median)
medianRT
```

```
## expert novice
## 1.3535 1.9650
```

```
diffObserved <- diff(medianRT)
```

The difference between these medians is 0.6115 seconds. We'd like determine if this difference is significantly different from zero.

26.2 The Permutation Test

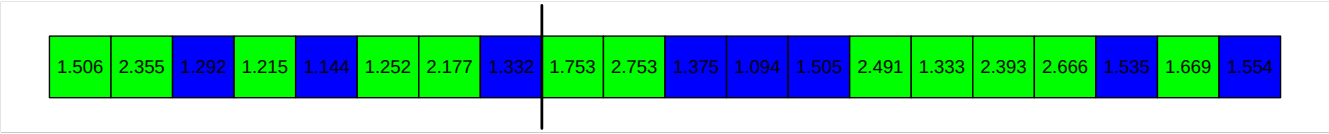
How can we calculate a p-value for this observed difference in medians? We'll start with a *permuation test*. Permutation simulations involve exchanging labels on data points to get an estimate of the sampling distribution under the null hypothesis. Permutation tests also go by the names of *exact tests*, *randomization tests*, or *re-randomization tests*.

Our null hypothesis is that the two samples of RT's are drawn from the same population (or two identical populations). This means that for our sample of 20 RTs there is nothing special about first 8 being drawn from the experts group the remaining 12 being drawn the from novice group. That is, the difference in medians that we observed shouldn't be any more extreme than for any arbitrary reordering of the samples.

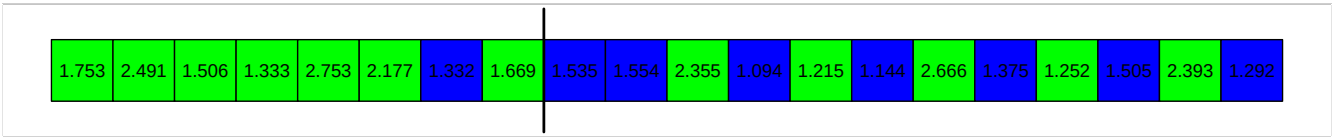
For example, let's reorder our 20 samples using the `sample` function:

```
permute.dat1 <- dat1[sample(nrow(dat1)),]
```

Here's a new table of RT's. I've kept the original color coding. Let's pretend that the first 8 came from experts, indicated by the vertical line:



The median of the first 8 response times is 1.312 and the median of the remaning 12 response times is 1.6115. The new differences is 0.2995 seconds. This is just one permutation. Here's another:



Now the median of the first 8 response times is 1.711 and the median of the remaning 12 response times is 1.44. This differences is -0.271 seconds.

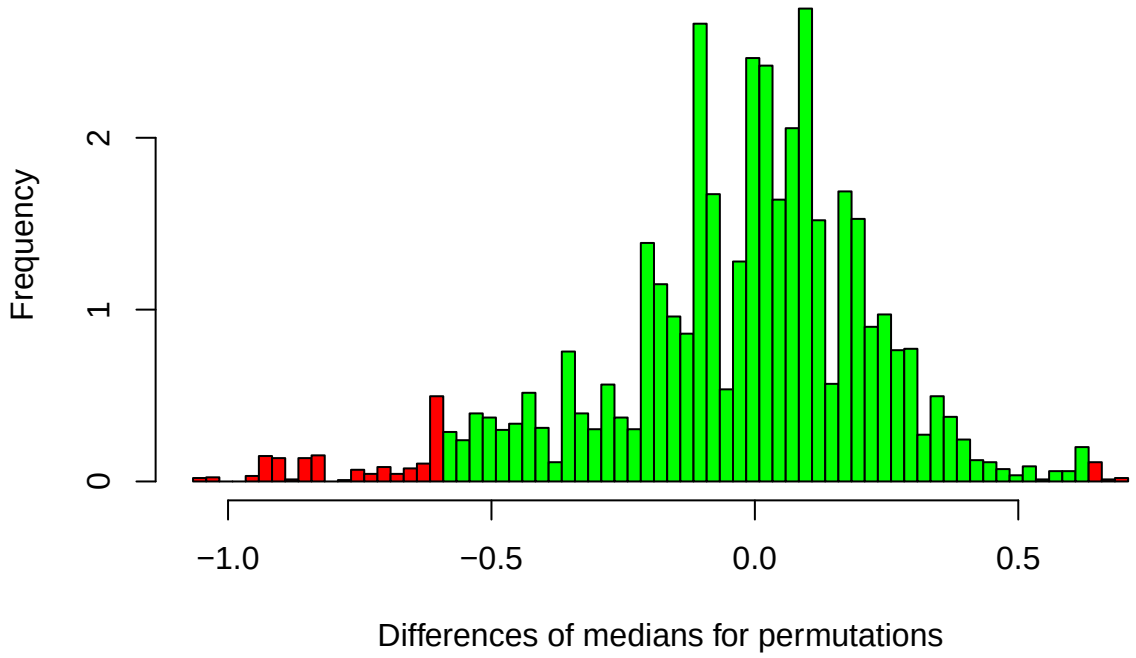
Permutation tests involve calculating our statistic on many, many permutations and see how unusual our observed statistic is compared to the statistics from the resampled data sets. You can think of the distribution of resampled statistics as our estimate of the population distribution of the null hypothesis, which is that there isn't anything special about how our 20 RT's are divided into the two categories.

Our p-value will be the proportion of resampled medians that are more extreme than our observed statistic.

The following chunk of R code repeatedly permutes the RT's and recalculates the differences between the median of the first 8 RTs and the median of the last 12 RTs:

```
nReps <- 10000
permute.median.diff <-  matrix(data=NA,ncol=nReps)
for (i in 1:nReps) {
  # 'sample' by default samples without replacement, which is a permutation
  permute.RT <- sample(dat1$RT)
  # calculate the median RT's for this permuted data
  # using the 'condition' coding from the real data:
  medians <- tapply(permute.RT,dat1$condition,median)
  # save the difference in medians for this permutation
  permute.median.diff[i] <- diff(medians)
}
```

Here's a histogram of the differences in medians for these 10000 permutations.



I've colored in red the permutations that generated differences in medians that are more extreme (in absolute value) than our observed difference of +/- 0.6115. The remaining are in green.

This is a complicated looking distribution, but it's our best guess at the sampling distribution of median differences under the null hypothesis.

Now all we do is calculate the proportion of these permutations that fell into the red zones. This serves as a p-value for the permutation test:

```
permute.p.value <- sum(abs(permute.median.diff) > abs(diffObserved))/nReps
sprintf('Permutation test p = %5.4f ',permute.p.value)
```

```
## [1] "Permutation test p = 0.0371 "
```

The proportion of differences that are colored red is 0.0371. This serves as our p-value. If this value is less than $\alpha = 0.05$ we should reject H_0 and conclude that there is a significant difference in medians between the two groups.

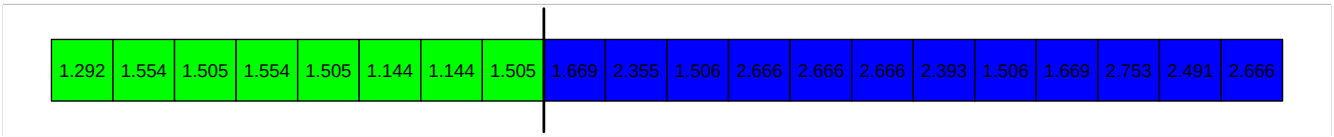
26.3 Bootstrapping - sampling with replacement

We can use another kind of resampling method called *bootstrapping* to estimate the likelihood of our observed statistic. Bootstrapping methods involve generating fake data sets based on your observed data and re-running your statistic. Fake data sets are obtained by *sampling with replacement* your observed data. The idea is that these resampled data sets reflect the statistical properties of your sample, and therefore the properties of the population they came from. For this example, we'll generate 8 fake RT's for the expert group by resampling with replacement the original 8 RT's, and do the same for the 12 RT's for the novice group:

Here's a nice use of `tapply` to resample RT's for each level of condition.

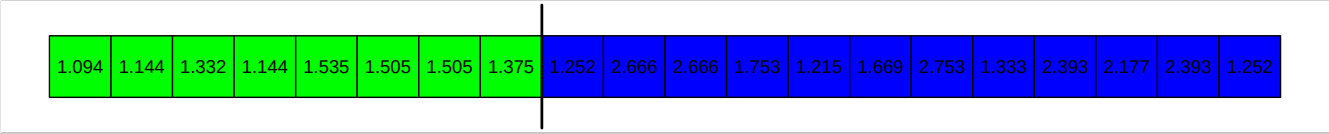
```
f <- function(x) sample(x,length(x),replace = TRUE)
resample.out <-   tapply(dat1$RT,dat1$condition,f)
```

Here's the resampled data:



Now the median of the 8 resampled expert response times is 1.505 and the median of the 12 resampled novice response times is 2.442. This differences is 0.937 seconds.

Here's another one:



For this resampled data set, the median expert response times is 1.3535 and the median of novice response times is 1.965. This differences is 0.6115 seconds.

Sampling with replacement means that some RT's will be sampled more than once, and others will be left out. It turns out that resampled data like this is good way of generating new data sets that have the same statistical properties (means, standard deviations, skewness...) as the population that the original data set was drawn from.

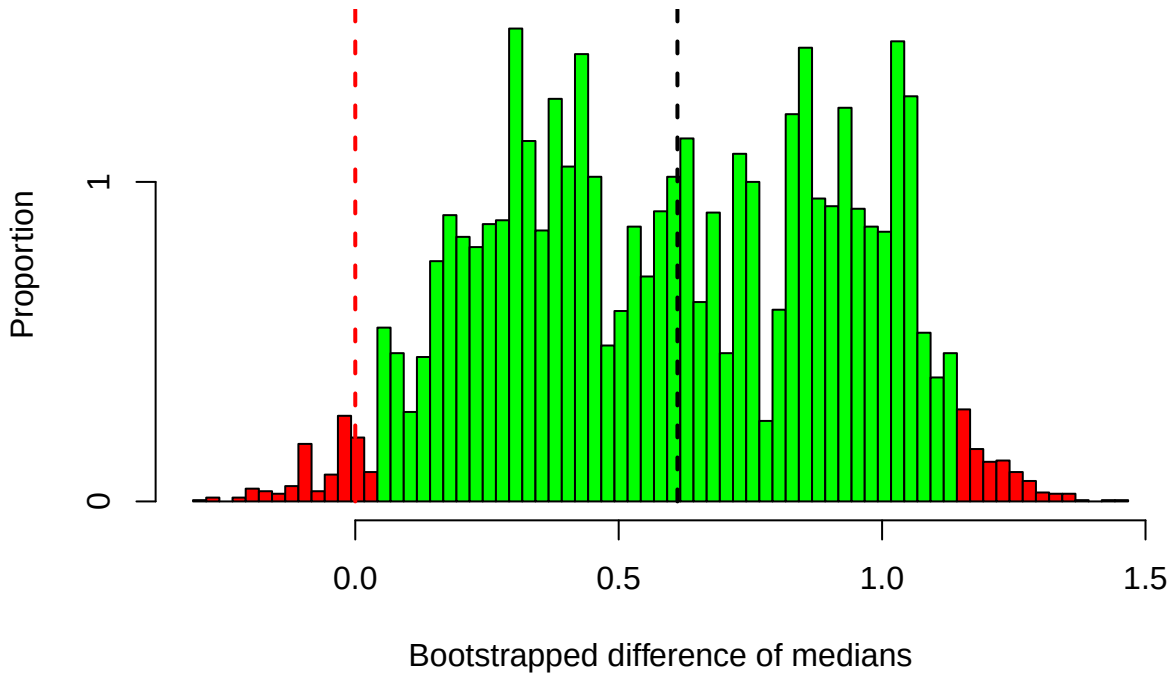
After resampling we recalculate our statistic, which is the absolute value of the difference between medians. On average the resampled statistic will tend toward the statistic from the original data. But it will vary each time we resample, and it's this variability that we use to estimate the likelihood of our observed statistic.

Unlike for the permutation test, were the differences in medians averages around zero, the typical difference in medians for a resampled data set is expected to be somewhere around differences in the medians for the real data set.

The following chunk of R code continues this resampling process, generating a distribution of differences of medians:

```
nReps <- 10000
bootstrap.median.diff <-  matrix(data=NA,ncol=nReps)
f <- function(x) sample(x,length(x),replace = TRUE)
for (i in 1:nReps) {
  resample.out <-   tapply(dat1$RT,dat1$condition,f)
  bootstrap.median.diff[i] <-
    median(resample.out$novice) -
    median(resample.out$expert)
}
```

Here's a histogram of the resampled differences of medians:



This histogram is our guess at what the population distribution looks like for the difference between medians. Remember, our observed median is was 0.6115, which I've indicated with a dashed black line. It's right in the middle of the distribution, but you can see that some of differences after resampling can be quite far away from our observed difference.

I've colored in green the bootstrapped statistics that fall within the middle 95 and red outside. These green samples fall within the 95 confidence interval. These values are:

```
quantile(bootstrap.median.diff,c(.025,.975))
```

```
##   2.5%  97.5%
## 0.0410 1.1385
```

We reject H_0 if the confidence interval does not contain the value for the null hypothesis (which is zero in our case).

This confidence interval does not contain zero (the green bars don't reach the red dashed line). This means that differences in the medians of our resampled data fall below zero less than 2.5 of the time. We therefore conclude that our observed difference of medians is significantly different from zero using an alpha value of $\alpha = 0.05$.

We can also calculate a p-value from the bootstrapped medians. This is the proportion of resampled differences that fall below zero, multiplied by two to get a two-tailed test:

```
2*mean(bootstrap.median.diff < 0)
```

```
## [1] 0.0386
```

This is just the tip of the iceberg for how bootstrapping works. Formally, we should be using a modification of this procedure that corrects for a bias with small samples. There are various ways proposed to do this. The most common is the *bias-corrected and accellerated (BCa)* method. For proper implementation, R has a package that does all the resampling and bias correction called boot.

26.4 Statistics on ranks

We now move onto the more traditional nonparametric tests, seen in the purple boxes in the flow chart above. All of these tests involve 'rank transformations' of the data in some way or another. Rank transformations simply replace the observations in a sample by their corresponding ranks starting with 1 as the lowest observation.

The ranks of a list have a known mean and standard deviation. A list of numbers counting from 1 to n has a sum of

$$\frac{n(n+1)}{2}$$

and therefore a mean of

$$\frac{n+1}{2}$$

and a (population) variance of

$$\frac{n^2-1}{12}$$

A rank transformation a set of n numbers will always have this mean and standard deviation since the order of the values don't matter.

Nonparametric tests that use rank transformations use these known means and transformations to calculate p-values. For example, if you have a data set with two groups of equal size and you rank order the entire set of numbers, then under the null hypothesis, we'd expect half of the ranks to fall randomly into each set. The sum of an arbitrary half of the ranks will therefore be, on average, half of the sum of all of the ranks:

$$\frac{n(n+1)}{4}$$

But the variance of an arbitrary half of ranks should, on average, be the same as the variance of all of the ranks. So now we have two means and variances which can be used to calculate a p-value. Note that the formula for the standard deviation above leads to strange constants like 4 and 12 in some of the equations for nonparametric tests that use ranks.

26.4.1 Dealing with ties

A note about ties: when ranking a list of numbers, if there are repeated values (ties) then the ranks for all of those ties are given the same number - the mean of the ranks that they'd have if they weren't ties. You can see how R does with the `rank` function for a list with ties:

```
v <- c(10,11,11,14,16)
rank(v)
```

```
## [1] 1.0 2.5 2.5 4.0 5.0
```

The second and third in the list are ties. So they each get a rank of 2.5. Ties mess up the expected means and standard deviations of a rank transformation a bit. Sometimes this is ignored. R's nonparametric test functions will give you scary warnings and run normal approximation versions of the tests instead, which we'll cover below.

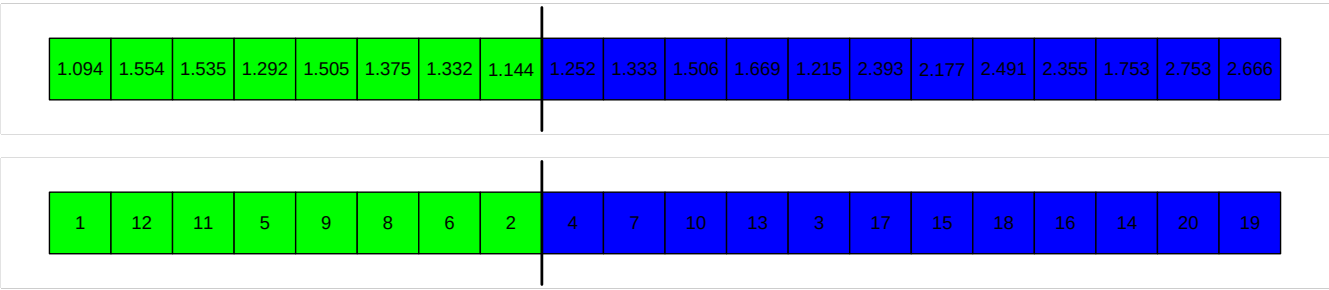
26.5 Wilcoxon's Rank-Sum Test

One well-known test that uses rank transformations is the *Wilcoxon's Rank-Sum test* which is the nonparametric alternative to the independent measures t-test. It tests the null hypothesis that two *medians* from two samples came from the same population. The Wilcoxson's Rank Sum is the same as the *Mann-Whitney U test*. Both give the same p-values, though with slightly different statistics (W for Wilcoxon and R for Mann-Whitney).

Later on we will introduce the Kruskal-Wallis nonparametric test which works for more than two independent means. Just like the 1-way ANOVA is a generalization of the two sample t-test, the Kruskal-Wallis is a generalization of the Wilcoxon's Rank-Sum Test. The Kruskal-Wallis test is actually a lot easier to implement than the Wilcoxon test. I'll say it here - we don't need the Wilcoxon's Rank-Sum Test, but I'm covering it because it is a common test that you'll see in the literature.

26.5.1 The procedure:

The procedure works by giving ranks for each of the 20 response times. Here is the original set of RT's and the rank transformation:



You should appreciate that if the experts have shorter RTs then the ranks for these 8 RTs should be smaller than for the novices. You should also appreciate that these ranks, like medians, are not dependent on the actual value of the extreme values - just their relative order.

The dependent measure in the Wilcoxon's Rank Sum test starts with adding up the ranks for each of the two groups:

For experts: $1+12+11+5+9+8+6+2=54$

for the novices: $4+7+10+13+3+17+15+18+16+14+20+19=156$

We then calculate W_1 and W_2 by subtracting the minimum value possible: $\frac{(n)(n+1)}{2}$, where n is the sample size for that group

$W_1 = R_1 - \frac{n_1(n_1+1)}{2}$

$W_2 = R_2 - \frac{n_2(n_2+1)}{2}$

For our example:

(expert) $W_1 = 54 - \frac{(8)(9)}{2} = 18$

(novice) $W_2 = 156 - \frac{(12)(13)}{2} = 78$

The W's have a known distribution. R has it's own function `pwilcox` that will convert your observed value of W to a p-value. Assuming a two-tailed test:

```
2*pwilcox(18,8,12)
```

```
## [1] 0.02013178
```

Note that sometimes, the values of R_1 and R_2 are used instead of W_1 and W_2. By default, R gives you the smaller of the two values of W.

26.5.2 Using R:

Here's how to run the Wilcoxon rank sum test in R and how to organize the results into an APA format. The 'formula' `RT ~ condition` is the way we define the independent and dependent variables in `lm`:

```
wrs.out.exact <- wilcox.test(RT ~ condition,data = dat1,exact = TRUE)
sprintf('W = %g, p = %5.4f',wrs.out.exact$statistic,wrs.out.exact$p.value)
```

```
## [1] "W = 18, p = 0.0201"
```

You might notice that the p-value we get from the Wilcoxon Rank Sum test is not the same as the permutation test (0.0371). I've found with simulations that under the null hypothesis the correlation between the two p-values is only about 0.75.

Technically this is the 'Exact' version of the Wilcoxon's rank-sum test, hence the `exact = TRUE` input.

26.6 Wilcoxon's Rank Sum Test: The Normal Approximation

R has trouble with the exact test when there are ties. Also, calculating the p-values for W (e.g. with `pwilcox`) can take up a large amount of time and computer memory if the sample sizes are large. If either there are ties or the sample sizes are large, R will run the inexact version automatically.

When you set `exact=FALSE` with `wilcox.test`, the test uses a normal approximation to the distribution of W based on the the means and standard deviations of ranks discussed earlier. Here's how it works:

26.6.1 The procedure:

For larger sample sizes (the rule is a total of about 25 observations) the distribution of the smallest of the two summed ranks tends toward a normal distribution that has a mean of:

$$\mu = \frac{n_1(n_1 + n_2 + 1)}{2}$$

And a standard deviation:

$$\sigma = \sqrt{\frac{n_1 n_2(n_1 + n_2 + 1)}{12}}$$

For our example:

$$\mu = \frac{8(21)}{2} = 84$$

$$\sigma = \sqrt{\frac{(8)(12)(21)}{12}} = 12.9615$$

26.6.2 By hand:

We can use `pnorm` to calculate the probability of obtaining a value of R drawn from a normal distribution with these means and standard deviations. Here's (presumably) how R computes the normal approximation to the Wilcoxon's rank sum test:

```
# calculate the sample sizes, sum of ranks and R (the smallest sum of ranks)
n <-  tapply(ranked.dat1$RT,ranked.dat1$condition,length)
sumRanks <- tapply(ranked.dat1$RT,ranked.dat1$condition,sum)
R <- min(sumRanks)
# calculate the mean and standard deviation of normal approximation to R for HO
mu <- as.numeric(n[1]*(n[1]+n[2]+1)/2)
sigma <- as.numeric(sqrt(n[1]*n[2]*(n[1]+n[2]+1)/12))
# use pnorm to get the p-value based on R - using correction for continuity (two-tailed)
p <- 2*(pnorm(R+.5,mu,sigma))
sprintf('p = %5.4f',p)
```

```
## [1] "p = 0.0228"
```

26.6.3 Using R:

Here's how to run the 'nonexact' version of the test by setting `exact = FALSE`:

```
wrs.out.exact <- wilcox.test(RT ~ condition,data = dat1,exact = FALSE)
sprintf('W = %g, p = %5.4f',wrs.out.exact$statistic,wrs.out.exact$p.value)
```

```
## [1] "W = 18, p = 0.0228"
```

The p-value matches the value we calculated by hand.

26.7 Wilcoxon's Matched-Pairs Signed-Ranks Test

The Wilcoxon Matched-Pairs test is a nonparametric alternative to the dependent-measures t-test.

Suppose now that you want to study the ability to recognize famous faces at a distance (Geoff Loftus was doing this experiment outside my office, using our long hallway as a laboratory) You find 10 subjects and ask each to identify famous faces from a near and far distance.

Here's how to load in the data:

```
dat2<-read.csv("http://www.courses.washington.edu/psy524a/datasets/acuitydistance2.csv")
```

26.7.1 The procedure:

The first two columns in the following table is our list of response times:

Table 26.1:

	one meter	10 meters	D	$\|D\|$	Rank of $\|D\|$
S1	1.539	2.849	-1.310	1.310	9
S2	2.207	4.049	-1.842	1.842	10
S3	1.359	2.552	-1.193	1.193	7
S4	3.725	3.902	-0.177	0.177	2
S5	3.025	3.408	-0.383	0.383	5
S6	3.298	3.375	-0.077	0.077	1
S7	5.360	5.025	0.335	0.335	4
S8	4.213	3.898	0.315	0.315	3
S9	2.003	3.297	-1.294	1.294	8
S10	1.377	2.055	-0.678	0.678	6

The third column is the difference between conditions, D, just as we would calculate for the repeated measures t-test. Notice that 2 of the 10 differences are positive.

The Wilcoxon's Matched-Pairs Signed-Ranks Test is based on a ranking of the absolute value of these differences. Absolute values are in the fourth column and corresponding ranks of the absolute values of D are in the fifth column.

Zeros values of D are ignored, so the first positive value is given a rank of 1 and so on.

I've colored the rows for the positive differences in green, and the negative differences in red. You can see visually the balance between the two. Our statistic, called V is the sum of the ranks for rows with positive differences, which is 7:

$V = 4 + 3 = 7$

Like W, the distribution for V under the null hypothesis can be calculated. The gist is that with the null hypothesis, half of the ranks should contribute to V on average. Earlier we showed that the average of half the ranks is $\frac{n(n+1)}{4} = 27.5$. We reject H_0 if V gets too far away from this value.

R has it's own function `psignrank` that will calculate a p-value from your observed value of V:

```
2*psignrank(7,10)
```

```
## [1] 0.03710938
```

26.7.2 Using R:

We use `wilcox.test` again to run this test but this time we'll set `paired = T` Here's how it's done:

```
wp2.out <- wilcox.test(RT ~ condition,dat2,paired = T,exact = T)
sprintf('Wilcoxon Matched Pair Sign Rank Test: V = %g, p = %5.4f ',
        wp2.out$statistic,wp2.out$p.value)
```

```
## [1] "Wilcoxon Matched Pair Sign Rank Test: V = 7, p = 0.0371 "
```

26.8 Wilcoxon's Matched-Pairs Signed-Ranks Test n>50: The Normal Approximation

Just like for the Wilcoxon Ranked-Sum Test, there is a normal approximation for larger number of pairs (typically greater than 50). Note: I've seen examples of the normal approximation used for only 10 or more pairs (e.g. https://en.wikipedia.org/wiki/Wilcoxon_signed-rank_test). Like for the Wilcoxon's rank sum test, calculating the distribution of V for large samples can be very time consuming, so R will switch to the normal approximation for a large sample size.

26.8.1 The procedure:

The mean of this distribution of T's is:

$$\mu = \frac{n(n+1)}{4}$$

And the standard deviation is:

$$\sigma = \sqrt{\frac{n(n+1)(2n+1)}{24}}$$

For our example (even though our sample size is less than 50)

$$\mu = \frac{10(11)}{4} = 27.5$$

and

$$\sigma = \sqrt{\frac{10(11)(21)}{24}} = 9.8107$$

26.8.2 By hand:

We can use R's `pnorm` function to find the probability of obtaining a value of V= 7 or more extreme when drawn from a normal distribution with these mean and standard deviations. Here's how to do it by hand. Note the calculation of the z-score - `abs(V-mu)-.5)` is a way of doing the correction for continuity, regardless of whether V is greater or less than μ.

```
n <- 10
mu <- n*(n+1)/4
sigma <- sqrt(n*(n+1)*(2*n+1)/24)
z <- (abs(V-mu)-.5)/sigma
p <- 2*(1-pnorm(abs(z)))
sprintf('p = %5.4f',p)
```

```
## [1] "p = 0.0415"
```

26.8.3 Using R:

You can run the 'inexact' version of the Wilcoxon's matched-pairs signed-ranks test like this:

```
wp2.out <- wilcox.test(RT~condition,dat2,paired = TRUE,exact=FALSE)
sprintf('Wilcoxon Matched Pair Sign Rank Test (inexact): V = %g, p = %5.4f ',
        wp2.out$statistic,wp2.out$p.value)
```

```
## [1] "Wilcoxon Matched Pair Sign Rank Test (inexact): V = 7, p = 0.0415 "
```

The p-values match what we calculated by hand.

26.9 Sign Test

Another nonparametric test that we can used for dependent-measures samples is the Sign Test. We look at the number of positive and negative values of D to see how likely this could happen by chance using the binomial distribution with P= .5. For our example, we want to calculate the probability of getting 8 out of 20 (or more) positive differences. This is really just a binomial test on the number of positive differences (see section 12.5)

26.9.1 Using R:

This can be done simply with `binom.test`, after counting the number of positive differences.

```
# count the number of positive differences
k <- sum(dat2$RT[dat2$condition=="ten_meters"]  > dat2$RT[dat2$condition=="one_meter"] )
n <- sum(dat2$condition=="one_meter")
sign.out <- binom.test(k,n,.5,alternative = 'two.sided')
sprintf('Sign test: k = %d, n = %d, p = %5.4f',
        sign.out$statistic,sign.out$parameter,sign.out$p.value)
```

```
## [1] "Sign test: k = 8, n = 10, p = 0.1094"
```

The sign test is the nonparametric test that uses the least amount of information in our data. It makes the fewest assumptions about how our data is distributed and is therefore the least powerful.

In the middle is the Wilcoxon's Matched-Pairs Signed-Ranks test, which takes into account the order of the differences. The dependent measures t-test assumes the most, and is the most powerful, as long as the conditions are satisfied.

26.10 Normal approximation to the Sign Test

Remember the normal approximation to the binomial distribution? I bet you do (section 12.4). The Sign Test is just a binomial test, so for sample sizes of 20 or more we can use the normal approximation to the binomial. We have only 10 differences in our example, but we can work out The procedure anyway:

26.10.1 The procedure:

Remember, the expected number of positive differences under the null hypothesis has a mean of:

$$\mu = NP$$

And the standard deviation is:

$$\sigma = \sqrt{NP(1-P)} = \sqrt{N(.5)(.5)}$$

For our example

$$\mu = (10)(.5) = 5$$

and

$$\sigma = \sqrt{10(.5)(.5)} = 1.58$$

26.10.2 By hand:

We can use `pnorm` to calculate the probability of getting 8 or more out of 10 positive differences (or more extreme). Here's how to do it. Note the correction for continuity again:

```
# quick way to get a list of RT's by condition
dat_list <-tapply(dat2$RT,dat2$condition,rbind)
D <- dat_list$ten_meters - dat_list$one_meter
V <- sum(rank(abs(D))[D>0])
n <- length(dat_list$one_meter)
mu <- n*.5
sigma <- sqrt(n*.5*.5)

z <- (abs(k-mu)-.5)/sigma
p <- 2*(1-pnorm(abs(z)))
sprintf('p = %5.4f',p)
```

```
## [1] "p = 0.1138"
```

This normal approximation is pretty close to the 'exact' result from the binomial test.

26.11 Kruskal-Wallis One-Way ANOVA

This is a nonparametric version of the independent measures ANOVA for cases where assumptions like normality and homogeneity of variance are violated. Like the Wilcoxon's Rank-Sum Test, it relies on rank-ordering the data.

Let's add one more group to our original study on recognizing famous faces for expert and novices and find 10 monkeys to participate in our study. You can load in the example data set like this:

```
dat3 <- read.csv("http://www.courses.washington.edu/psy524a/datasets/RTexpertnovicemonkey.csv")
```

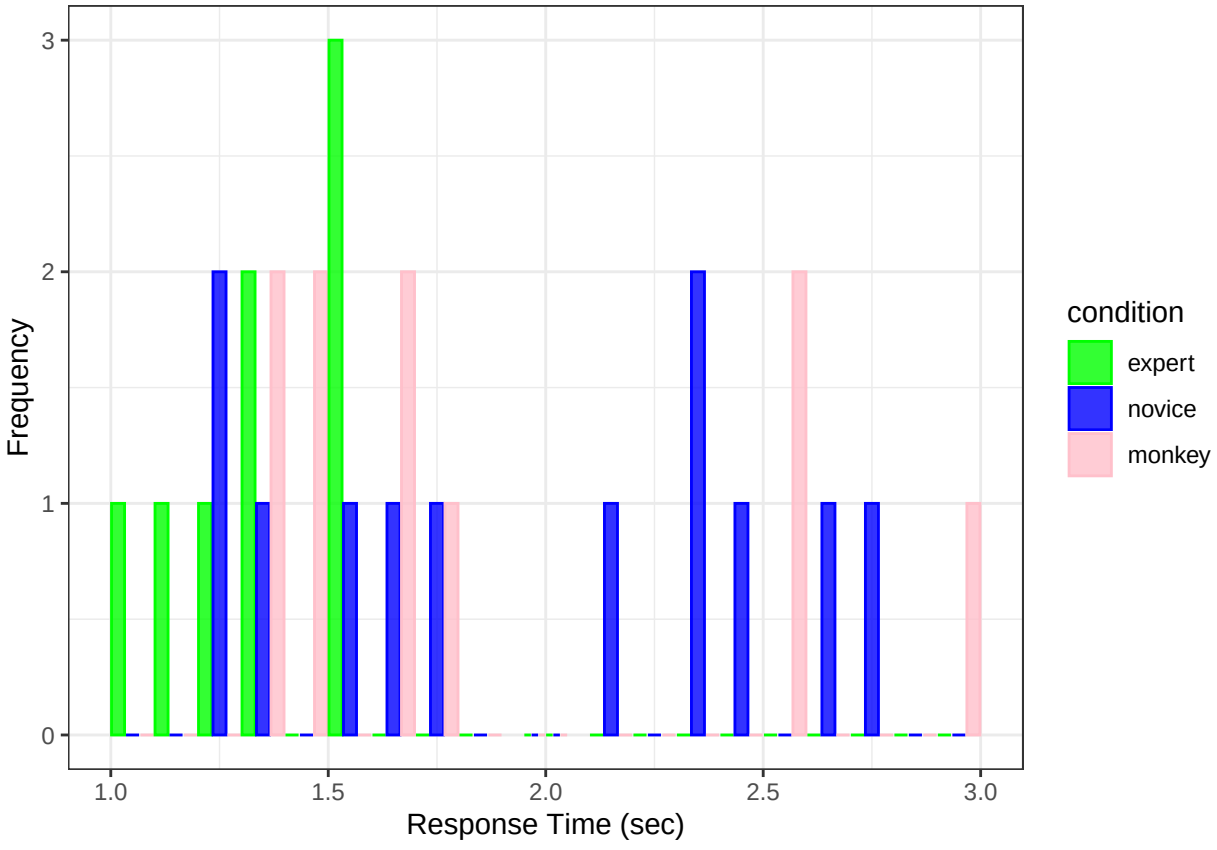
The data set now looks like this:

Table 26.2:

expert	novice	monkey
1.094	1.252	1.343
1.554	1.333	1.463
1.535	1.506	1.623
1.292	1.669	1.720
1.505	1.215	1.365
1.375	2.393	2.571
1.332	2.177	1.493
1.144	2.491	2.922
	2.355	2.527
	1.753	1.674
	2.753	
	2.666	

Here's a histogram with the new set of subjects:

```
ggplot(dat = dat3, aes(x = RT,fill = condition,col = condition)) +
  geom_histogram(position="dodge2", alpha=.8,breaks = seq(1,max(dat3$RT)+.1,by = .1))+
  scale_fill_manual(values=col)+
  scale_color_manual(values=col)+
  xlab("Response Time (sec)") + ylab("Frequency") +
  theme_bw()
```



26.11.1 The procedure:

For the Kruskal-Wallis test we find the ranks for each of these RT's as if they were all in one pile. Here's the rank transformation:

Table 26.3:

expert	novice	monkey
1	4	8
16	7	11
15	14	17
5	18	20
13	3	9
10	24	27
6	22	12
2	25	30
	23	26
	21	19
	29	
	28	

The null hypothesis is that the average rank for each group is the same. The statistic we calculate to measure this is called 'H' which under th null hypothesis is approximated by a χ^2 distribution with k-1 degrees of freedom (k is the number of groups, 3 for our example).

First we calculate the sum of the ranks for each column. For our example:

For expert: $R_1 = 1 + 16 + 15 + 5 + 13 + 10 + 6 + 2 = 68$

For novice: $R_2 = 4 + 7 + 14 + 18 + 3 + 24 + 22 + 25 + 23 + 21 + 29 + 28 = 218$

For monkey: $R_3 = 8 + 11 + 17 + 20 + 9 + 27 + 12 + 30 + 26 + 19 = 179$

The statistic H is then calculated with this crazy formula:

$$H = \frac{12}{N(N+1)}(\sum \frac{R_i^2}{n_i}) - 3(N+1)$$

Where N is the total number of measurements $(8 + 12 + 10 = 30)$.

For our example:

$$H = \frac{12}{30(31)}(578 + 3960.33 + 3204.1) - 3(30+1) = 6.9$$

26.11.2 By hand:

Here's how to do this by hand:

```
R <- tapply(rank(dat3$RT),dat3$condition,sum)
n <- table(dat3$condition)
N <- sum(n)
H <- 12/(N*(N+1)) * (sum(R^2/n)) - 3*(N+1)
df = length(n)-1
p <- 1-pchisq(H,df)
sprintf('chi-squared(%d) = %5.4f, p = %5.4f',df,H,p)
```

```
## [1] "chi-squared(2) = 6.9024, p = 0.0317"
```

26.11.3 Using R:

But of course, R has a function for this:

```
kruskal.out <- kruskal.test(RT ~ condition, data = dat3)
sprintf('chi-squared(%d) = %5.4f, p = %5.4f',
        kruskal.out$parameter,
        kruskal.out$statistic,
        kruskal.out$p.value)
```

```
## [1] "chi-squared(2) = 6.9024, p = 0.0317"
```

A perfect match.

26.12 Friedman's Rank Test

Our final nonparametric test of the book (are you sad?) is the nonparametric version of the repeated measures 1-way ANOVA, the Friedman's Rank Test. This is the same Milt Friedman, the U.S. economist and Nobel laureate who was possibly the most influential advocate of free-market capitalism and monetarism in the 20th century. All that and he got non parametric test named after him!

To extend our repeated measures data set on famous face recognition at a distance, suppose we add a third condition of one hundred meters to each subject's measurement of RT. Here's how to load in the data:

```
dat4<-read.csv("http://www.courses.washington.edu/psy524a/datasets/acuitydistance3.csv")
```

The new results look like this:

Table 26.4:

	one meter	ten meters	one hundred meters
S1	1.539	2.849	2.319
S2	2.207	4.049	2.345
S3	1.359	2.552	3.603
S4	3.725	3.902	4.329
S5	3.025	3.408	6.894
S6	3.298	3.375	6.041
S7	5.360	5.025	6.003
S8	4.213	3.898	5.029
S9	2.003	3.297	1.828
S10	1.377	2.055	1.520

26.12.1 The procedure:

The procedure is to rank the scores within each row: first, second and third. Here are the corresponding ranks:

Table 26.5:

	one meter	ten meters	one hundred meters
S1	1	3	2
S2	1	3	2
S3	1	2	3
S4	1	2	3
S5	1	2	3
S6	1	2	3
S7	2	1	3
S8	2	1	3
S9	2	3	1
S10	1	3	2

We then sum and square each column. This gives us:

Under the null hypothesis, the squared sum of each column should be approximately the same.

The statistic for the Friedman's Rank Test,χ^2_F is also approximated by a χ^2 distribution with k-1 degrees of freedom. But this time the equation is:

$$\chi^2_F = \frac{12}{nk(k+1)} \sum R_i^2 - 3n(k+1)$$

Which for our example is:

$$\chi^2_F = \frac{12}{(10)(3)(3+1)}(169+484+625) - (3)(10)(3+1) = 7.8$$

26.12.2 By hand:

Here's how to calculate Friedeman's rank test by hand:

```
# convert to wide format:
dat <- reshape(dat4,idvar = "subject",timevar = "condition",direction = "wide")
# get rid of 'subject' column
dat <- dat[,2:4]
# use the 'apply' function to get ranks for each row.
# The '1' means rows, and the 't' transposes the results
# so it's the same shape as the original data.
R <- t(apply(dat,1,rank))
Ri <- colSums(R)
n<-nrow(R)
k <- ncol(R)
Fr <- 12/(n*k*(k+1))*sum(Ri^2) - 3*n*(k+1)
p <- 1-pchisq(Fr,k-1)
sprintf('Chi-squared(%g) = %g, p = %5.4f',k-1,Fr,p)
```

```
## [1] "Chi-squared(2) = 7.8, p = 0.0202"
```

26.12.3 Using R:

And, again, R has the function `friedman.test` to do the work for us. It uses the formula `RT ~ condition|subject`, which is telling the function that subject is the random effect (subject) factor. I has the same function as using `RT ~`

`condition + (1|subject)` with the `lm` function.

```
friedman.out <- friedman.test(RT ~ condition|subject, data  = dat4)
sprintf('Chi-squared(%g) = %g, p = %5.4f',
        friedman.out$parameter,friedman.out$statistic,friedman.out$p.value)
```

```
## [1] "Chi-squared(2) = 7.8, p = 0.0202"
```

26.13 Rank Transformations

Relatively recently a hybrid hypothesis test has become popular called a *rank transformation* hypothesis test.

26.13.1 The procedure:

These tests are just parametric tests run on rank transformations of the data. For example, this last data set on face recognition for different viewing distances, a rank transformation repeated measures ANOVA is simply a standard repeated measures ANOVA run on the data after rank-ordering all of the scores.

Here are the RT's across the three distances, ranked:

Table 26.6:

	one meter	ten meters	one hundred meters
S1	4	12	9
S2	8	22	10
S3	1	11	18
S4	19	21	24
S5	13	17	30
S6	15	16	29
S7	27	25	28
S8	23	20	26
S9	6	14	5
S10	2	7	3

Now all we need to do is run the standard parametric test on the ranked data. For this experiment we use a repeated measures ANOVA. Here's how to rank the data and run the test in R with `lmer` and `anova` which require the `lme4` and `lmeTest` libraries:

26.13.2 Using R:

```
ranked.dat4 <- dat4
ranked.dat4$RT <- rank(dat4$RT)
anova.out <- anova(lmer(RT ~ condition + (1|subject),ranked.dat4))
sprintf('F(%g,%g) = %5.4f, p = %5.4f',
        anova.out$NumDF,anova.out$DenDF,anova.out$`F value`,anova.out$`Pr(>F)`)
```

```
## [1] "F(2,18) = 4.3811, p = 0.0282"
```

From this we'd state something like:

"A repeated measures ANOVA on the ranked response times shows that there is a significant difference in the RTs across the three distances. ($F(2,18) = 4.3811$, $p = 0.0282$).

There are several advantages of rank transformation tests. The most obvious is that they take advantage of all of the machinery that has been developed over the years for parametric tests, including contrasts, post-hoc tests,

random factors, and power. Rank ordering the data serves to fix outliers. But you can also see how rank ordering could increase variance - both within and between - since the scores must range from 1 to the total number of measurements. I think the jury is still out on these tests, but I've seen them more and more often.

Our example here is enlightening. The p-value for the rank transformation ANOVA is much smaller than for the Friedman Rank test. Any huge discrepancy like this makes you wonder what the correct answer is.

For an interesting discussion on them compared to the nonparametric tests like the Friedman Rank test see: (http://seriousstats.wordpress.com/2012/02/14/friedman/)

‘

```
source('libraries.R')
nbins <-80
```

Chapter 27

Everything is normal: generating χ^2, F and t from z-scores

Remember that normal distributions can be generated by averaging draws from any distribution, thanks to the Central Limit Theorem. So the normal distribution shows up everywhere that things are summed or averaged. But the normal distribution is just one of several distributions we've covered in this class. It turns out that all of these other distributions are intimately related to the normal distribution.

This bonus chapter demonstrates how the three probability distributions that we've used for hypothesis testing in this class (t, χ^2 and F) can be generated by manipulating draws from the standard normal (z) distribution. The point of this chapter is to show that these distributions are all related - and that tables and software giving probabilities associated with these distributions aren't coming out of nowhere.

27.0.1 Means are normally distributed

Thanks to the Central Limit Theorem, the distribution of means from a population tend toward a normal distribution, even if the population is not normal. Thinking in terms of variances, if the population has mean μ_x and standard deviation σ_x, then the mean of the distribution of means (called the sampling distribution of the mean) is just the mean of the population:

$$\mu_{\bar{x}} = \mu_x$$

And the variance of the sampling distribution of the mean is equal to the variance of the population divided by the sample size for each mean:

$$\sigma^2_{\bar{x}} = \frac{\sigma^2_x}{n}$$

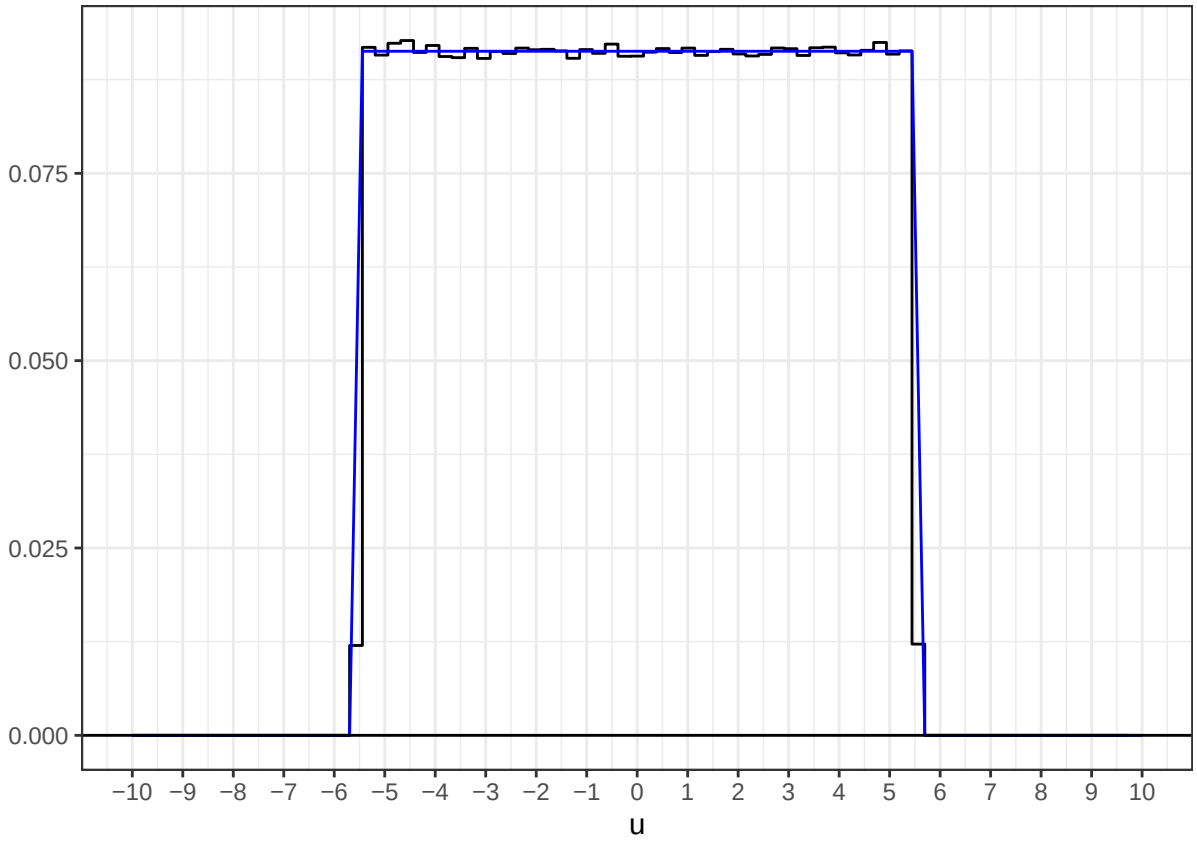
For example, a uniform distribution that has equal probability between a and b will have mean: $\frac{a+b}{2}$ and variance $\frac{(a-b)^2}{12}$.

A uniform distribution with values ranging from $-\sqrt{3n}$ to $\sqrt{3n}$ will therefore have a mean of zero and a variance of $\frac{(2\sqrt{3n})^2}{12} = n$.

This chunk of code generates a matrix with nSamples rows and n columns of values drawn from this distribution:

```
nSamples <- 100000
n <- 10

u <- matrix(runif(nSamples*n,-sqrt(3*n),sqrt(3*n)),ncol = n)
```

Here's a histogram of the entire set of numbers with the expected uniform distribution drawn with it (in blue:). The verifies that the function `runif` is generating samples as expected.

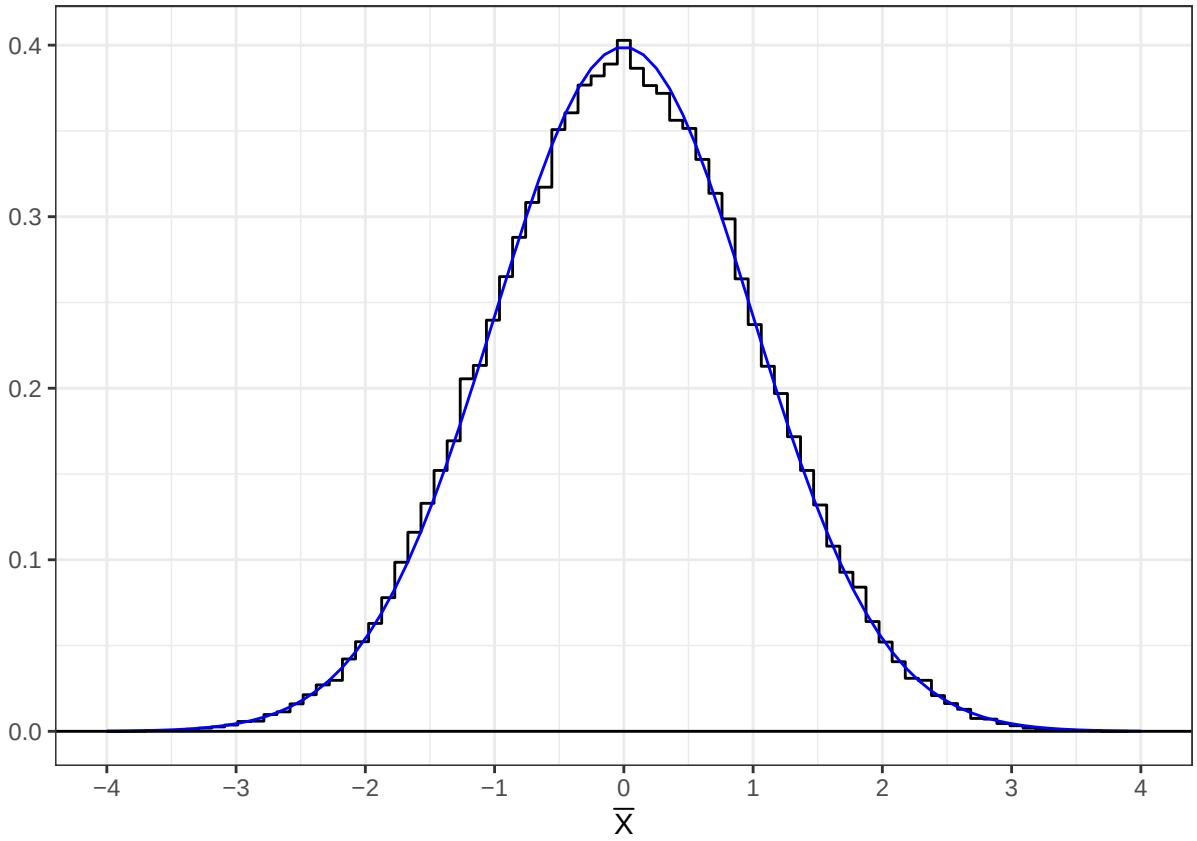


Since the variance of this distribution is n, the variance of the mean will be $\frac{n}{n} = 1$. So thanks to the Central Limit Theorem, the means drawn from this distribution should look like the z-distribution (mean 0, standard deviation 1)

This calculates the mean of each row, which the mean of 10 samples from this uniform distribution:

```
uvar <- apply(u,1,var)
norm <- rowSums(u)/n
```

Here's a histogram of those means with the standard normal (z) distribution drawn with it. It matches well.



The mean from any population with variance n and mean 0 will be distributed like a z-distribution. The closer the population is to normality, and the larger the n, the more normal the distribution of means.

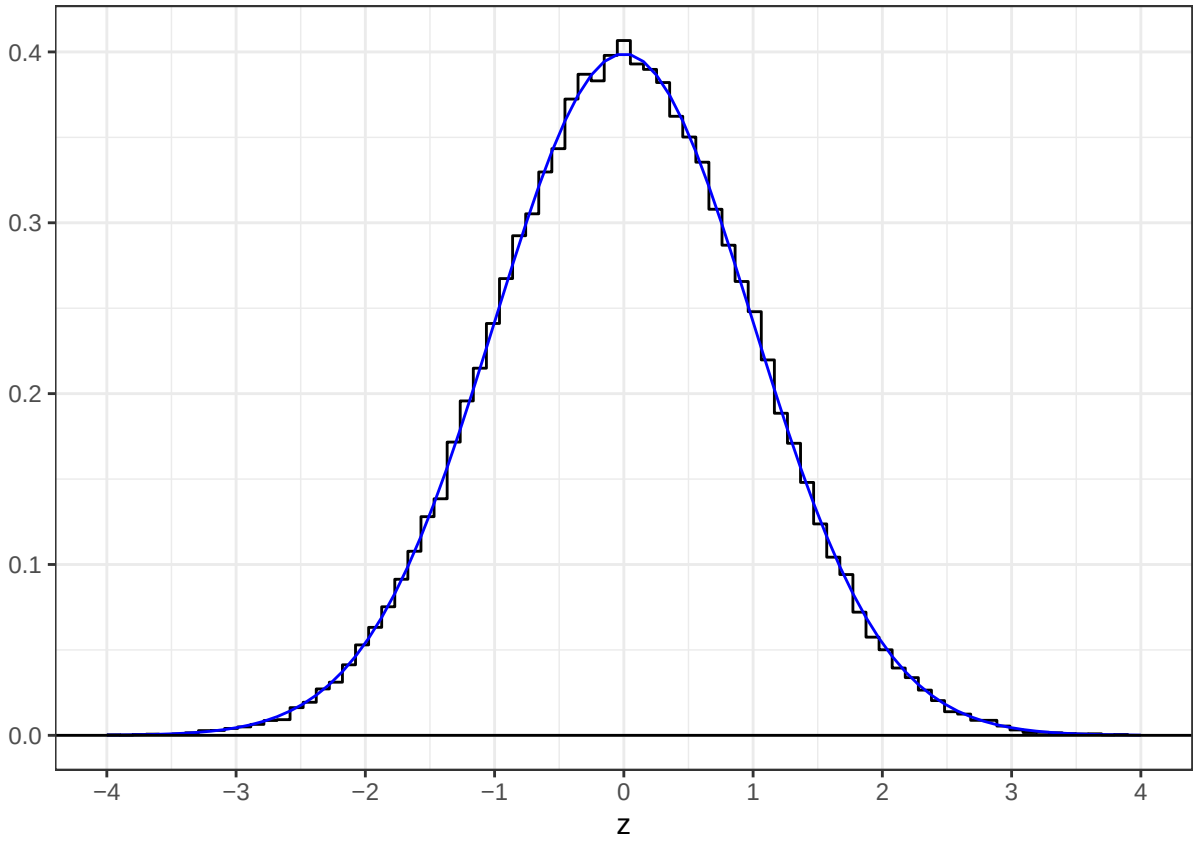
For the rest of this chapter we'll just draw from the z-distribution using `rnrorm` to simulate other distributions. But keep in mind we could always start by drawing means from any distribution as long as it has a variance of n and mean 0.

27.0.2 The z-distribution

Let's verify that a histogram of values drawn from `rnorm` match the probability distribution for z. Using 'rnorm' to sample from the standard normal distribution.

```
z <- rnorm(nSamples)
```

Here's a histogram of these z-scores with the standard normal pdf drawn with it (using `dnorm`)

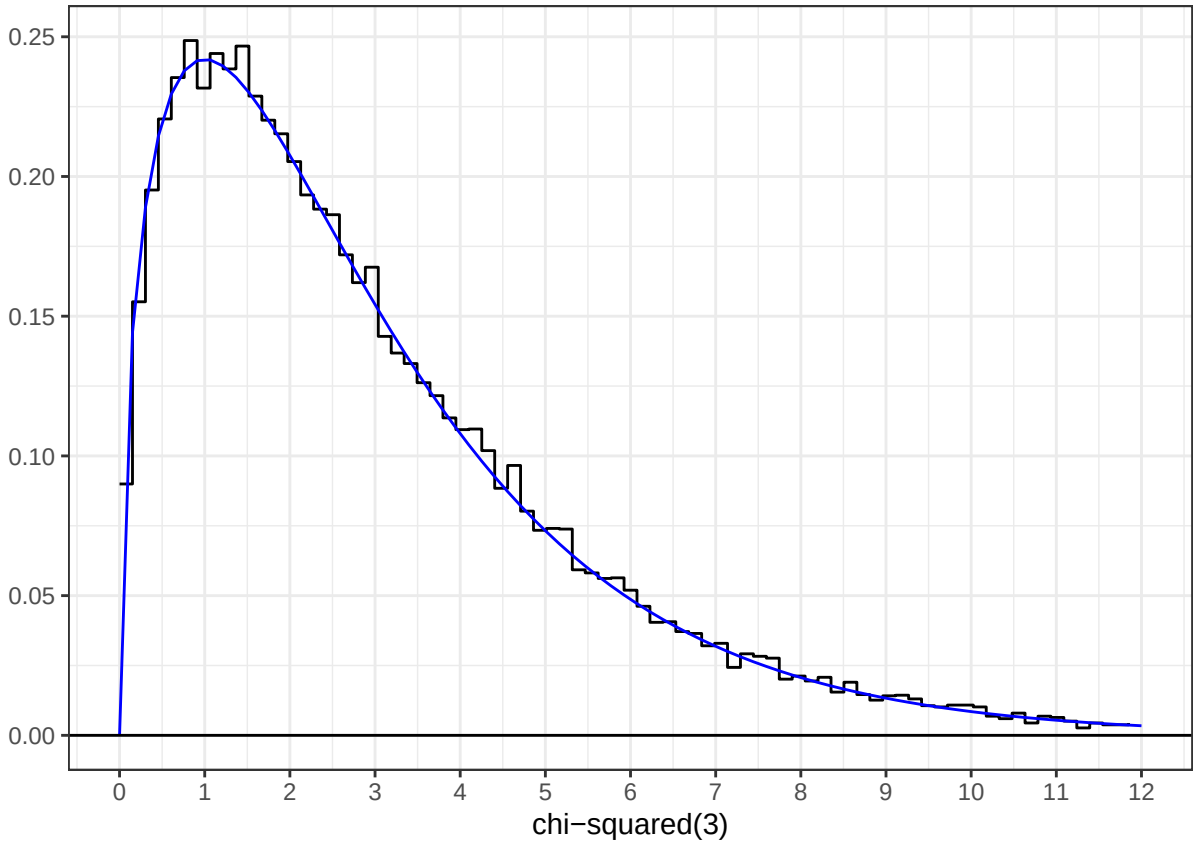


27.0.3 The χ^2 distribution

The χ^2 distribution can be generated by drawing *df* values from the z-distribution, squaring them and adding them up:

```
nSamples <- 30000
df <- 3

z <- matrix(rnorm(nSamples*df),ncol = df)
chi2 <- rowSums(z^2)
```

Here's a histogram of these simulated values with the χ^2 distribution of df = 3 drawn with it:



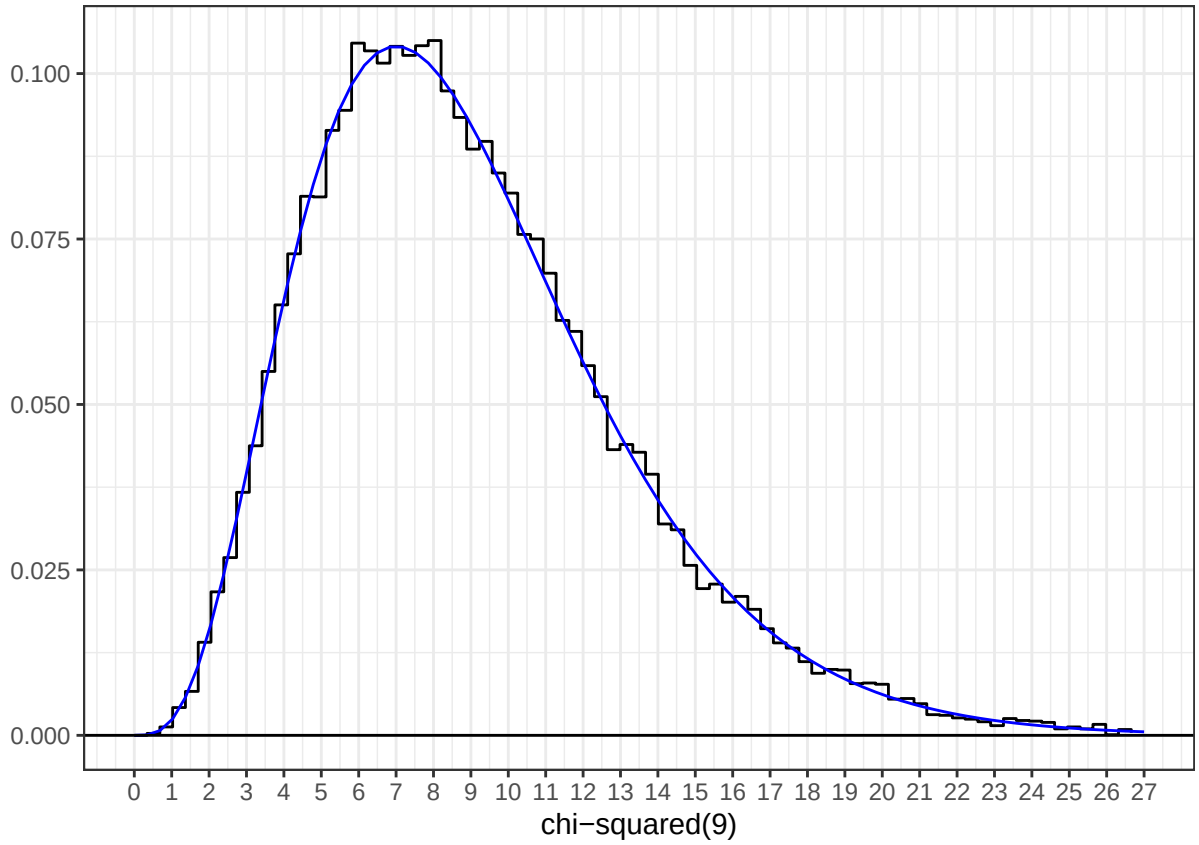
27.0.4 The distribution of variances

Variances of samples drawn from the z-distribution are distributed as χ^2 distributions with $df = n - 1$, divided by df. Equivalently, the variance of n values from the z-distribution times df is equal to the χ^2 distribution.

This calculates a list of length nSamples, each is the variance of n values drawn from the z distribution, multiplied by $df = n - 1$:

```
n <- 10
df <- n-1

z <- matrix(rnorm(nSamples*n),ncol = n)
dftimesvar <- df*apply(z,1,var)
```

Here's a histogram of these values with the χ^2 distribution of degree 9 drawn with it.

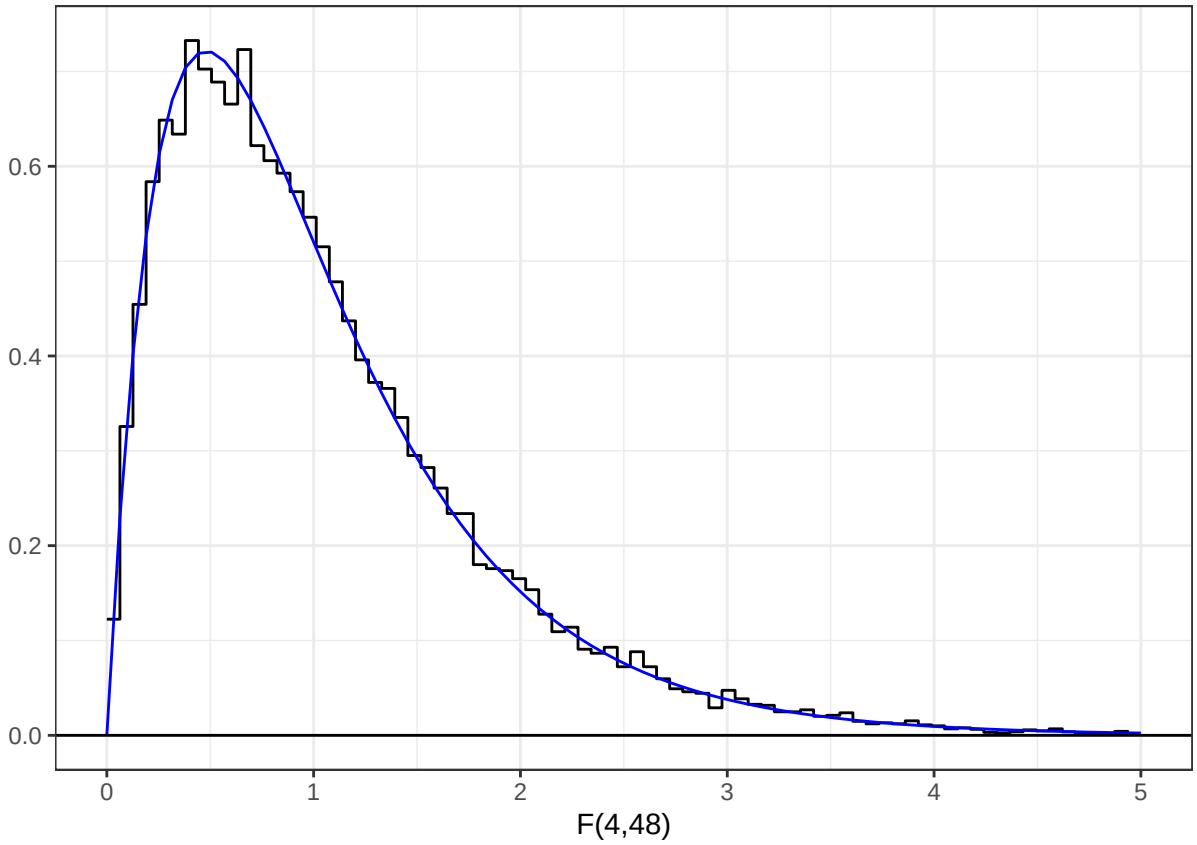


27.0.5 The F distribution

The F distribution is the ratio of two χ^2 distributions, each divided by their degrees of freedom. F is therefore the ratio of two variances (hence 'ANOVA'). Since χ^2 distributions can be generated from squared z-scores, it all starts with the z-distribution. This generates the ratio of two χ^2 distributions, each divided by their degrees of freedom:

```
df1 <- 4
df2 <-48

z1 <- matrix(rnorm(nSamples*df1),ncol = df1)  # To generate numerator chi-square
z2 <- matrix(rnorm(nSamples*df2),ncol = df2)  # To generate denominator chi-square
Fsim <- (rowSums(z1^2)/df1)/(rowSums(z2^2)/df2)
```

Here's a histogram of these values with the F-distribution having 4 and 48 degrees of freedom drawn with it:

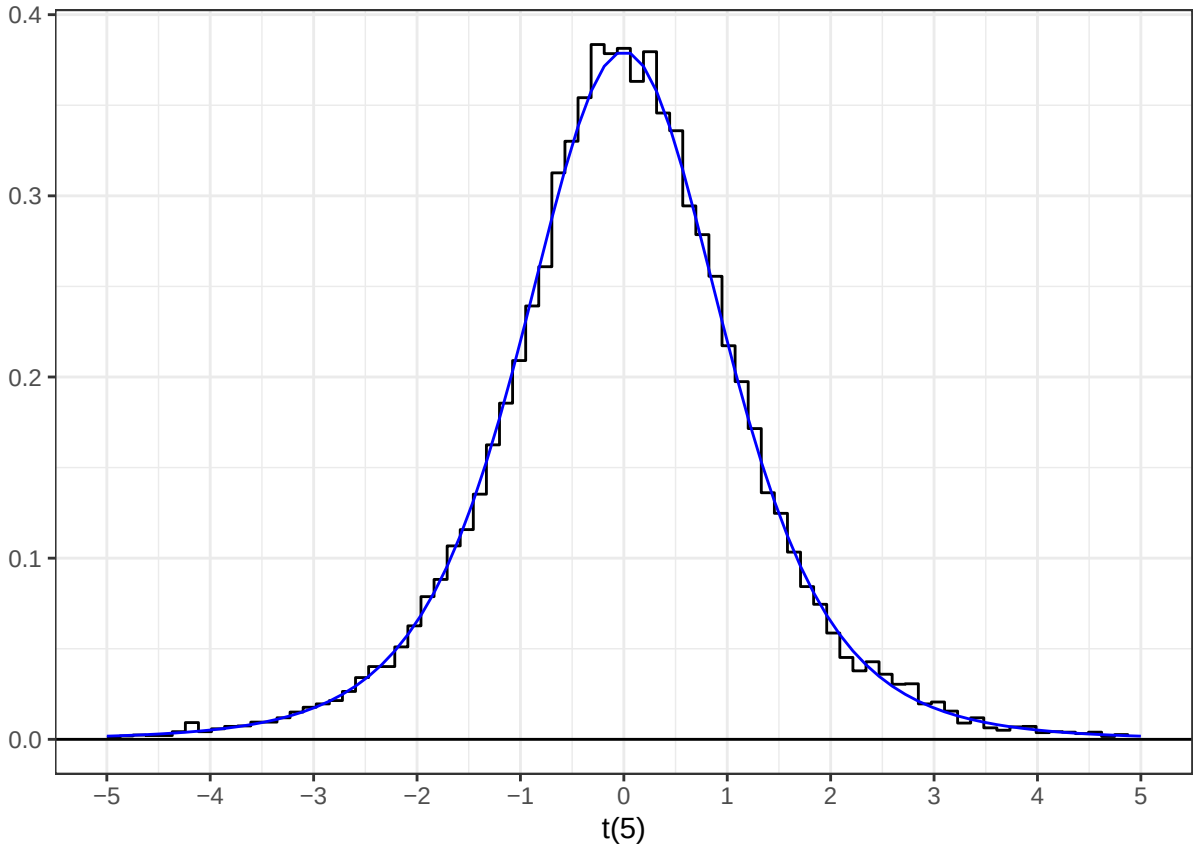


27.0.6 The t-distribution

The t-distribution is the ratio of a z-distribution divided by a distribution of standard deviations. Since variances are distributed as χ^2 divided by df, standard deviations are its square root. Again, the t-distribution can be generated by manipulating draws from the standard normal:

```
df <- 5

z1 <- rnorm(nSamples)
z2 <- matrix(rnorm(nSamples*df),ncol = df)

tsim <- z1/(sqrt(rowSums(z2^2)/df))
```

Here's a histogram of our values with the t-distribution with df = 5 drawn with it:



So that's it. I should point out that one other distribution that we've used, the binomial distribution, isn't directly generated by normal distributions but can be closely approximated by it. So the binomial distribution gets honorable mention in this chapter.

These simulations aren't particularly useful for solving statistical inference problems, but I hope they relieve some of the mystery behind the origins of the χ^2, F, and t distributions.

27.0.7 List of parametric distributions

R p, q, r, and d versions of a huge number of parametric distributions. Most, but not all of them can be derived from various manipulations of the standard normal (z) distribution. Here's a current list, many of these have been covered in this book:

Table 27.1:

	p	q	d	r
Beta	pbeta	qbeta	dbeta	rbeta
Binomial	pbinom	qbinom	dbinom	rbinom
Cauchy	pcauchy	qcauchy	dcauchy	rcauchy
Chi-Square	pchisq	qchisq	dchisq	rchisq
Exponential	pexp	qexp	dexp	rexp
F	pf	qf	df	rf
Gamma	pgamma	qgamma	dgamma	rgamma
Geometric	pgeom	qgeom	dgeom	rgeom
Hypergeometric	phyper	qhyper	dhyper	rhyper
Logistic	plogis	qlogis	dlogis	rlogis
Log Normal	plnorm	qlnorm	dlnorm	rlnorm
Negative Binomial	pnbinom	qnbinom	dnbinom	rnbinom
Normal	pnorm	qnorm	dnorm	rnorm
Poisson	ppois	qpois	dpois	rpois
Student t	pt	qt	dt	rt
Studentized Range	ptukey	qtukey	dtukey	rtukey
Uniform	punif	qunif	dunif	runif
Weibull	pweibull	qweibull	dweibull	rweibull
Wilcoxon Rank Sum Statistic	pwilcox	qwilcox	dwilcox	rwilcox
Wilcoxon Signed Rank Statistic	psignrank	qsignrank	dsignrank	rsignrank