Lab 4: Foundations for Inference

Alex Davis

May 21, 2018

This lab will give us the tools for the foundations of inference. We will revisit normality, and then move on to exploring stadard errors and confidence intervals, and end with learning about single sample t-tests. For this lab you will need the following packages:

1. car
2. psych
3. sm
4. HistData

Part One: Testing for Normality

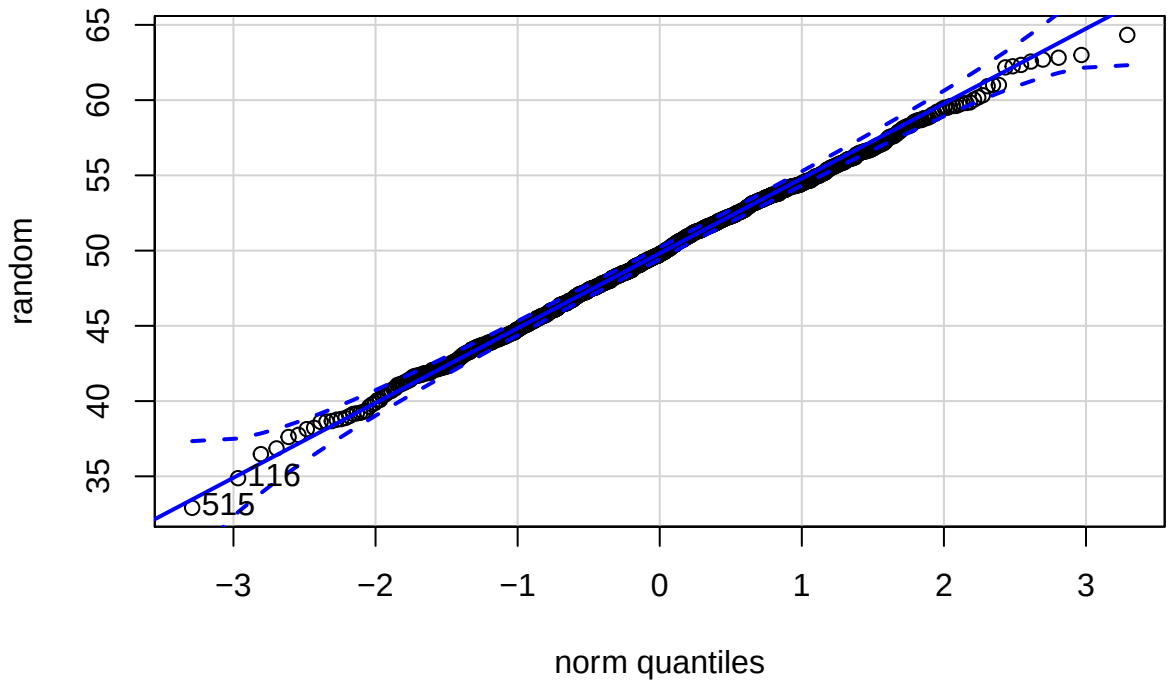
In lab three we started exploring the basics of visualizing normality. Much of the statistical work we do is built on the condition that our data is normally distributed. This topic is so important that it merits further discssion. In this section we will address ways to check if your data is normally distributed.

Recall that we can generate normally distributed data within R using the rnorm function. Let's generate 1000 random values with a mean of 50 and a standard deviation of 5 and call it "random":

```
random <- rnorm(1000, mean = 50, sd = 5)
```

There are many different ways to test for normality, one of which is to visualize the data using a particular visualization. In the last lab we went over density plots, box plots, and QQ plots. Let's visualize our new data using a QQ plot. Remember that a QQ plot graphs the quantiles of your data against the qunaitiles of what normally-distributed data should be given the mean and standard deviation of the data.

```
qqPlot(random)
```



```
## [1] 515 116
```

We see that this data is obviously normally distributed. But let's turn to an empirical means of testing if your data is normally distributed. One such test is the Shapiro-Wilk test.

Shapiro-Wilk Test

The Shapiro-Wilk test allows us to test and see if our data is normally-distributed. In R, you simply using the shapiro.test() function. There are some important things to note before using it, though:

1. The Shapiro-Wilk test should only be used on univariate, continuous data
2. The Shapiro-Wilk Test does not provide accurate results for very large sample sizes. As a rule of thumb you shouldn't use it on an n-size over 5000.
3. The Shapiro-Wilk Test tests against the null hypothesis that **the data already follows the normal distributed**. If you are able to reject the null, that means that your data is **not normally-distributed**. Smaller test scores indicate that the data is not normal.
4. The test returns a W score. W = 1 would indicate perfect normality.

Let's perform a Shapiro-Wilk test on our random data:

```
shapiro.test(random)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  random
## W = 0.99894, p-value = 0.8427
```

Notice how this data has a W score of almost 1. If we had been working with data that was not generated to already be normally-distributed, this would be more than enough to indicate that the data is normaly-distributed. Remember, the S-W test is testing against the null that the data is already normally-distributed. Looking at the p-value, this makes sense. If we want our data to be normal, we actually do not want to reject the null when using the Shapiro-Wilk test. In this case, we have a very high p value, meaning that if the null hypothesis were true (in this case, if the data was normally distributed), we would have a good probability of observing the same result. All in all, we don't reject the null here and we can assert that our data is normally distributed.

Testing Normality

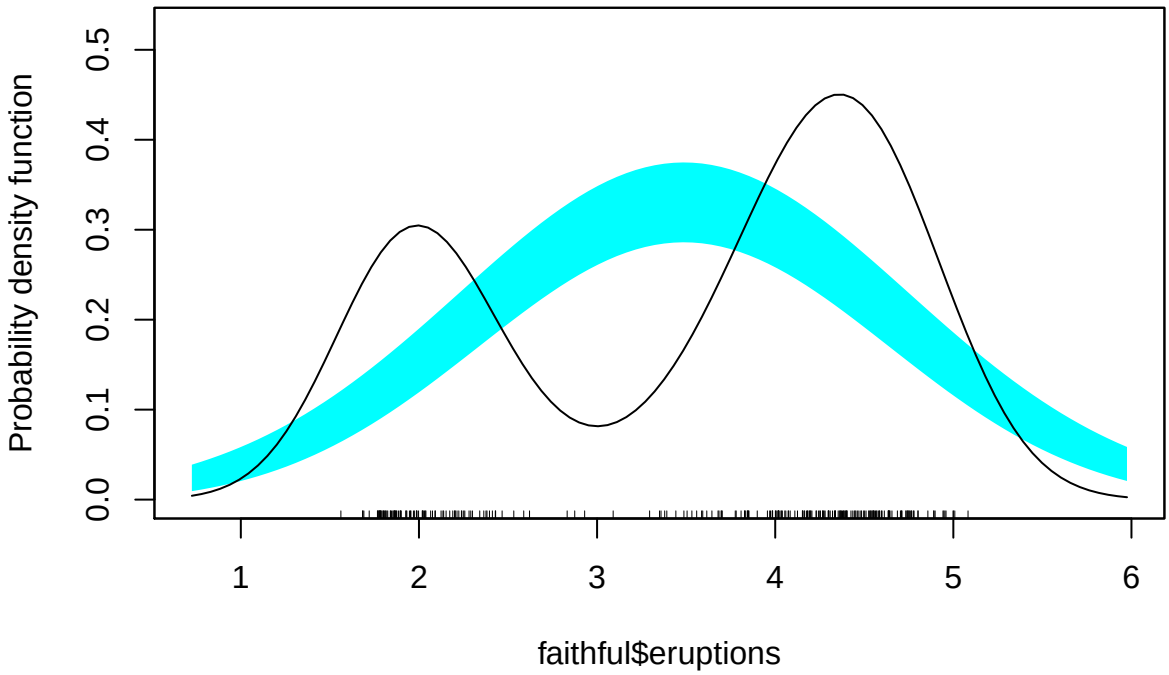
Now let's use a variety of methods to test for normality on data that we don't know is already normally-distributed. There is a data set about Old Faithful eruptions that comes pre-loaded in R, so let's use that. It provides data on the duration of eruptions in minutes, and the wait time in between eruptions in miunutes. Let's use the eruption length variable. First we'll look at the basics of it using the describe() function.

```
describe(faithful$eruptions)
```

```
##    vars   n mean   sd median trimmed  mad min max range  skew kurtosis
## X1    1 272 3.49 1.14      4    3.53 0.95 1.6 5.1   3.5 -0.41    -1.51
##      se
## X1 0.07
```

Now let's use the sm.density() function to look at the actualy density distribution of the data what a normal distribution of the data should look like given its mean adn standard deviation:

```
sm.density(faithful$eruptions, model = "Normal")
```



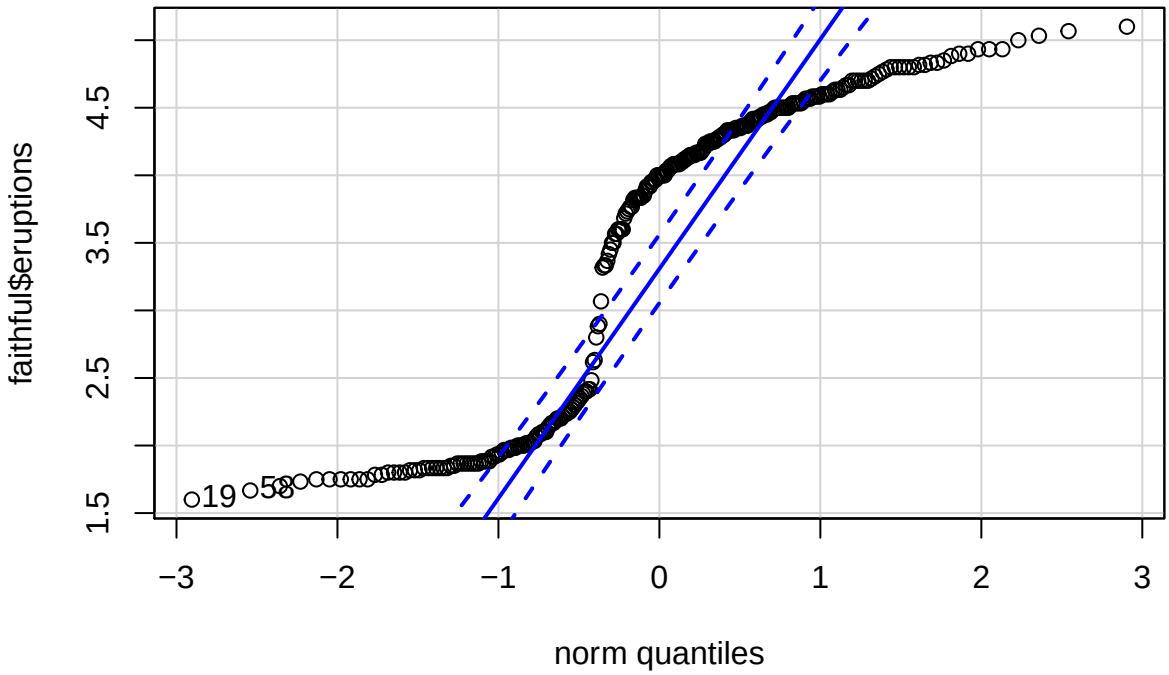
Notice that the actual distribution is bimodal (there are two peaks). This does not mean that the data is not normally-distributed, but it does mean that a simple visualization will not be enough for us to assess the normality of the data. This would be a prime situation to use the Shapiro-Wilk test. Let's do so:

```
shapiro.test(faithful$eruptions)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  faithful$eruptions
## W = 0.84592, p-value = 0.0000000000000009036
```

This result is a little more interesting. It looks as the if the data is not extremely close to the normal-distribution. The extremely small p value indicates strong certainty of this. To be sure, though, we should use a QQplot to further validate this:

```
qqPlot(faithful$eruptions)
```

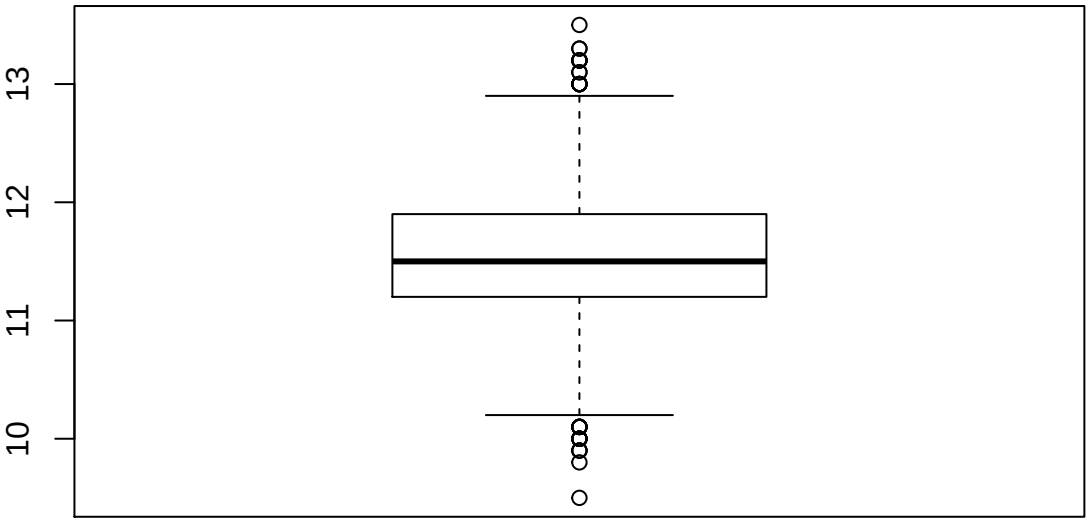



```
## [1] 19 58
```

After looking at the QQplot, it is pretty safe to reject the null hypothesis tested by the Shapiro-Wilk test and assert that the data is not normally-distributed.

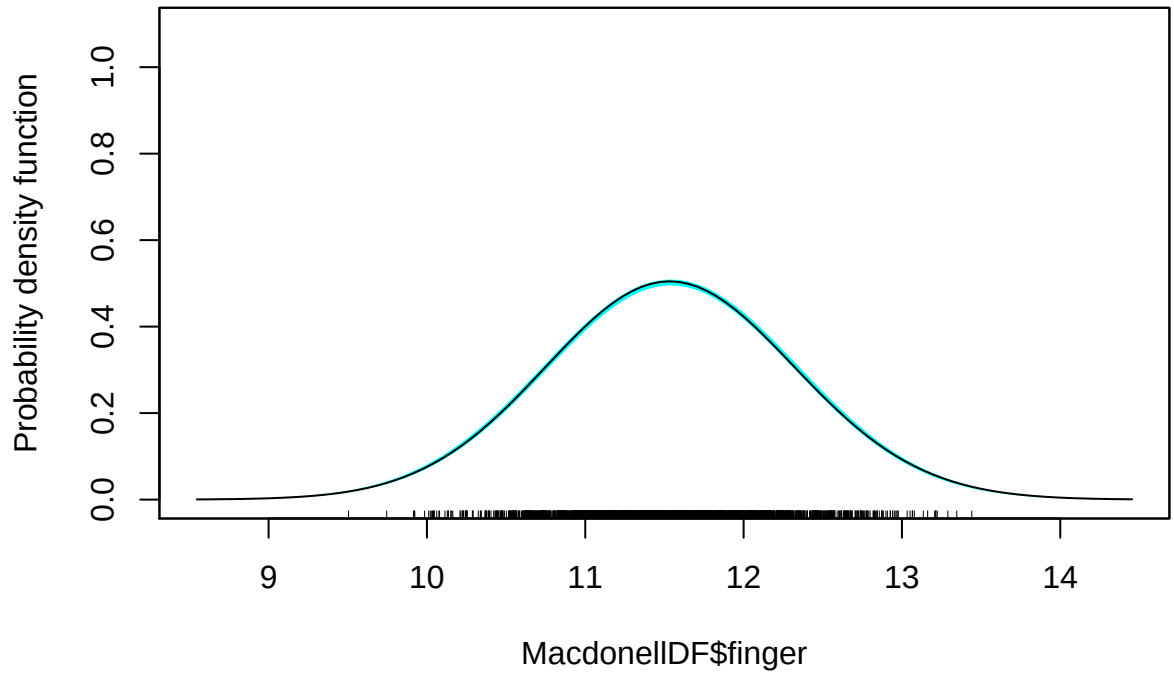
Let's do one more test, this time with different data. The HistData package includes different datasets we can use. Let's use a variable that measures finger length and see if the distribution of finger length is normal. Let's start with a boxplot:

```
boxplot(MacdonellDF$finger)
```



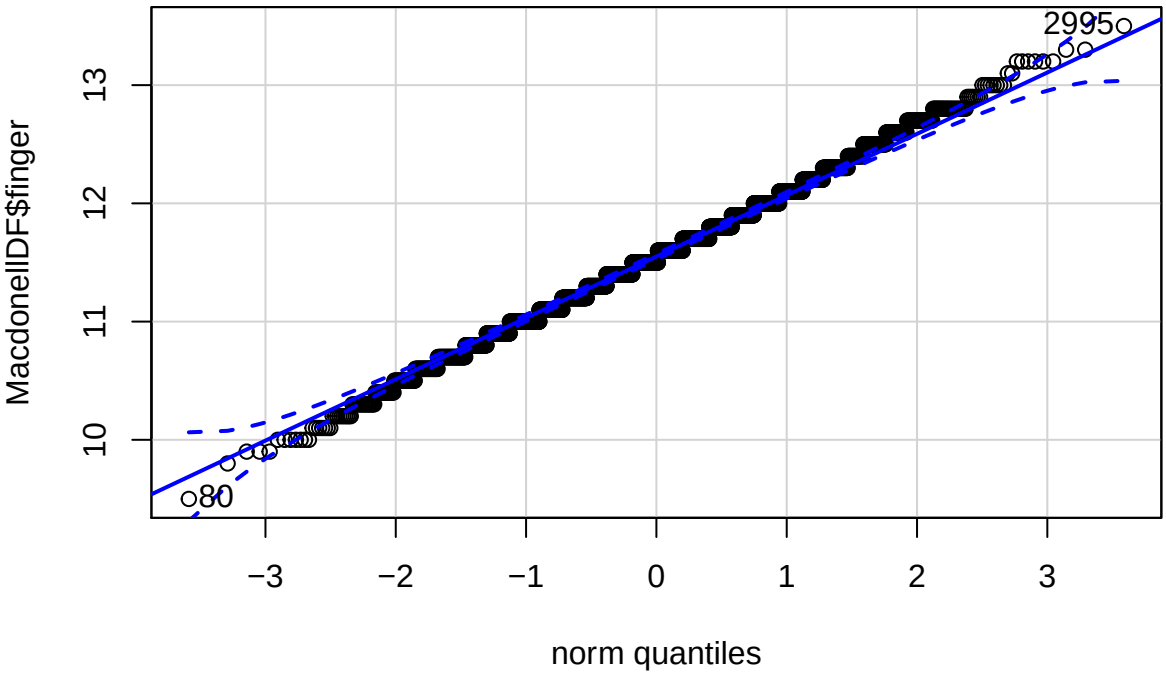
This is a pretty balanced boxplot. Now let's use the sm.density() function to look at the actual distribution vs projected normal distribution:

```
sm.density(MacdonellDF$finger, model = "Normal")
```



And now let's use a QQplot:

```
qqPlot(MacdonellDF$finger)
```




```
## [1]   80 2995
```

Based on all the visualizations, it looks like this data is very normal. However, let's run a Shapiro-Wilk test just to be sure.

```
shapiro.test(MacdonellDF$finger)
```

```
##
##  Shapiro-Wilk normality test
```

```
##
## data:  MacdonellDF$finger
## W = 0.99646, p-value = 0.000001666
```

As we can see, the W score is extremely close to one. Taking all of this into account, we can surely assert that this data is normally-distributed.

Part Two : Standard Errors

Recall that the standard error can be thought of as the standard deviation of the sample distribution of the sample mean. That might be confusing, so just remember that it is the square root of the variance of the population divided by the n size of the sample population. You could also think of it as the standard deviation of the population divided by the square root of the n size of the sample population. However, we often don't know the standard deviation of the whole population, so we use the standard deviation of the sample population.

We can calculate the standard error of the age variable in our class dataset like this:

```
sd(ds$age, na.rm=T)/sqrt(length(ds$age)-sum(is.na(ds$age)))
```

```
## [1] 0.2815446
```

Fortunately for us, R can do all these complicated calculations for us. Using the describe() function, we can more easily find the standard error:

```
describe(ds$age)
```

```
##    vars    n  mean    sd median trimmed   mad min max range  skew kurtosis
## X1    1 2547 60.37 14.21     62   61.01 13.34  18  99    81 -0.39    -0.24
##      se
## X1 0.28
```

Notice these answers are the same!

Part Three: Confidence Intervals

Knowing the standard error in the abstract might not intuitively seem very useful, but the standard error is vital to inferential statistics. The standard error is needed in order to calculate confidence intervals. Because we are trying to infer from a sample population something about a larger population, we have to understand that there will be some uncertainty in our calculations. Confidence intervals tell us just how much uncertainty there is.

Calculating confidence intervals should bring together and synthesize our knowledge of standard errors and z scores. Recall that under the normal distribution, a z score is how many standard deviations something is away from the mean. In order to calculate some level of confidence (90%, 95%, 99%), we need to know the z scores associated with those levels of confidence. Here they are:

- 90% = 1.645
- 95% = 1.960
- 99% = 2.576

Recall that the formula to calculate standard errors is:

$$CI = \bar{x} \pm z\frac{s}{\sqrt{n}},$$

Or, more simply, the sample mean, plus (and minus) the z score of the level of confidence you want multipled by the standard error.

This makes more sense when you actually calculate it, so let's do so. Let's find the true mean of age, using our age variable, with a level of 95% confidence:

First find the mean:

```
mean(ds$age, na.rm = T)
```

```
## [1] 60.36749
```

We are going for 95% confidence, so the z score we use will be 1.96 (rounded). We also need to know the standard error of the mean:

```
describe(ds$age)$se ## this is a way to pull the standard error out without having to look at everythin
```

```
## [1] 0.2815446
```

So now we have everything we need to find a confidence interval for the true mean of age. Let's find the upper level of the confidence interval:

```
60.38 + 1.96*.28
```

```
## [1] 60.9288
```

Now the lower end:

```
60.38 - 1.96*.28
```

```
## [1] 59.8312
```

Putting all this together, we can say that with 95% certainty, the true mean of age if between 59.83 and 60.93 (rounded).

If we wanted the confidence intervals to be a little more accurate, we could just tell R to use the mean and the standard error of the population without having to calculate them ourselves. Let's do that, but this time let's find the true mean of age at the 99% confidence level (we will use a z score of 2.58). First let's pull the standard error and assign it to an object called "se":

```
se <- describe(ds$age)$se
```

Now let's do the calculation

```
mean(ds$age, na.rm = T) + 2.58*se
```

```
## [1] 61.09388
```

```
mean(ds$age, na.rm = T) - 2.58*se
```

```
## [1] 59.64111
```

Now we know that we are 99% certain that the true mean of age is between 59.64 and 61.09. Notice that this interval is larger than the 95% confidence level. This should intuitively make sense because we should be more sure that the true mean is somewhere in a larger interval. The small the interval, the less certain we become.

To get an evenmore accurate idea of the true mean, R can calculate the confidence intervals for us. To do this, we need to recall the t distribution. The t distribution is used when you're working with a smaller n size, or when the data is not known to be normal. However, the t distribution also begins to approximate the z distribution as you have a larger n size. Since our n size is already large (over 2500), we can use the t distribution and know that it approximates the z distribution (what we've bene using so far). The function t.test() can be used to find the true mean. It tests whether the true mean is 0 and returns a p value for that. It will also give us 95% confidnece intervals for what the true mean actually is.

```r
t.test(ds$age)
```

```
##
##  One Sample t-test
##
## data:  ds$age
## t = 214.42, df = 2546, p-value < 0.00000000000000022
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  59.81541 60.91957
## sample estimates:
## mean of x
##  60.36749
```

Notice that this is very similar to what we already calculated for 95% CI, but this is more accurate. Notice the extremely small p value? That is because the one sample t test by default tests whether the mean is equal to zero. Though confusing, the p value tells us the probability of observing this particular calculation, given the null hypothesis is true. In this case, the probabilty that we would observe the a mean of 60.38 when the true mean of age is 0. Obviously that is not going to happen, hense the p value of essentailly zero. We can change the null hypothesis, though. Let's now test whether the true mean of the population is 50:

```r
t.test(ds$age, mu=50)
```

```
##
##  One Sample t-test
##
## data:  ds$age
## t = 36.824, df = 2546, p-value < 0.00000000000000022
## alternative hypothesis: true mean is not equal to 50
## 95 percent confidence interval:
##  59.81541 60.91957
## sample estimates:
## mean of x
##  60.36749
```

Again, we are veyr sure that the mean is not 50. Notice that 50 does fall in the confidence interval. But what if we were testing with an age that does? Let's test whether the true mean of age is 60:

```r
t.test(ds$age, mu=60)
```

```
##
##  One Sample t-test
##
## data:  ds$age
## t = 1.3053, df = 2546, p-value = 0.1919
## alternative hypothesis: true mean is not equal to 60
## 95 percent confidence interval:
##  59.81541 60.91957
## sample estimates:
## mean of x
##  60.36749
```

Notice now how there is more uncertainty! There is much higher p value, because it is much more likely we would observe a 60.36 given the true mean is 60. The p value of .19 does not meet the traditional threshold of .05 needed to reject the null hypothesis.

If we wanted to do this test using the normal z distribution instead of the t distribution, we can calculate

the p values ourselves. Let's say we wanted to test whether the true mean of the age is 60, using the z distribution. We would need to

1. Find the sample mean (xbar), in this case the mean of the variable.
2. Calculate confidence intervals (we already did this earlier).
3. Find the p value associated with our null (the true mean is 60).

- To calculate the p vale we'll need to know our n size. In this case it is 2547.

Let's create some objects to make this calcution simpler:

```
xbar <- mean(ds$age, na.rm = T)
a <- 60
s <- sd(ds$age, na.rm = T)
n <- 2547
```

Let's calculate the p value for testing if the true mean is not 60:

```
2*(1-pnorm(xbar, mean=a, sd=s/sqrt(n)))
```

```
## [1] 0.1918016
```

Now recall that our 95% confidence interval from earlier went from 59.83 to 60.93. Now we have everything we need to know in order to test the hypothesis that the true mean of age is not 60. We can say that with 95% confidence, the true mean is between 59.83 and 60.93. Our p value of .191 does not allow us to reject the null hypothesis that the true mean is 60. Notice how this is extremely similar to the calculations from the t test.

Part Four: More on Single Sample T-tests

Above we went over the basics of confidence intervals and p values, and included a preliminary discussion of t-tests. Let's continue with that exploration. Whereas the normal distribution uses z scores, the t distribution uses t scores. The t distribution is based on sample estimates, wheres as the normal distribution is properly used when we know the population variance and population mean.

We'll now establish the building blocks of the single sample t test. Let's test whether the true mean of age is 58. We will need to identify the n size, the sample standard deviation, the sample mean, and the t statistic.

```
nAge <- length(ds$age)-sum(is.na(ds$age))
sdAge <- sd(ds$age, na.rm=TRUE) # Standard deviation of age
seAge <- sdAge/(sqrt(nAge)) # Standard error of age
meanAge <- mean(ds$age, na.rm = TRUE)
```

To find the t value we take the sample mean, subtract the population mean we're testing (58 in this case) and divide it by the standard error:

```
t_value <- (meanAge - 58)/seAge
t_value
```

```
## [1] 8.408938
```

Now to find the two-sided p value under the t distribution:

```
2*(1-pt(abs(t_value),df=nAge-1))
```

```
## [1] 0
```

We're getting a returned p value of 0. This should lend some certainty to what we know from previous tests, that the true mean is not 58. Let's use the t.test function to test this for sure:

```r
t.test(ds$age, mu=58)
```

```
##
##  One Sample t-test
##
## data:  ds$age
## t = 8.4089, df = 2546, p-value < 0.00000000000000022
## alternative hypothesis: true mean is not equal to 58
## 95 percent confidence interval:
##  59.81541 60.91957
## sample estimates:
## mean of x
##  60.36749
```

We know for sure that the true mean is not 58!