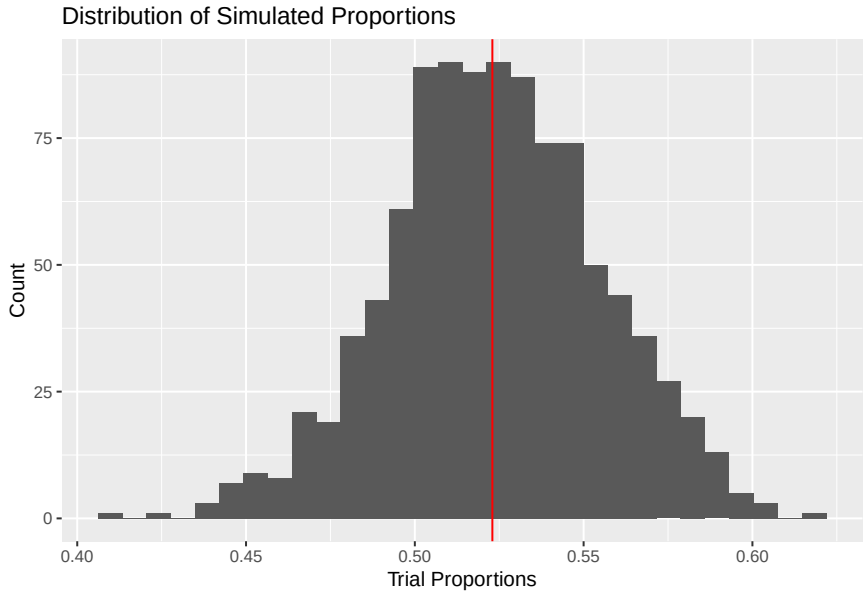
Lesson 6 Example Code

Are you have read in the book, confidence intervals (CI) are all about "plausible values" of the parameter we are researching. The book presents the *2SD method* of calculating the 95% CI as well as presenting some "multiplier values" for common confidence levels. I aim to show you quick way to do this in *R* so you both understand where these numbers are coming and don't have to memorize (or look them up).

First, we calculate a range of plausible values for a parameter because we can't calculate the exact parameter value. If we could, we wouldn't need to do all this statistics stuff. This should seem fairly obvious, but I state it because I see a lot of students forget why we are calculating a CI or what the CI tells us. We **do not** calculate a CI for the sample statistic: we don't need a range of plausible values for that... we have the actual value! We do use the sample statistic, however, because we need a starting (or center) point for our range. Yes, we are assuming that the sample statistic gives some us some information about the population parameter. If you're not ready to make that assumption better pack it up and head home.

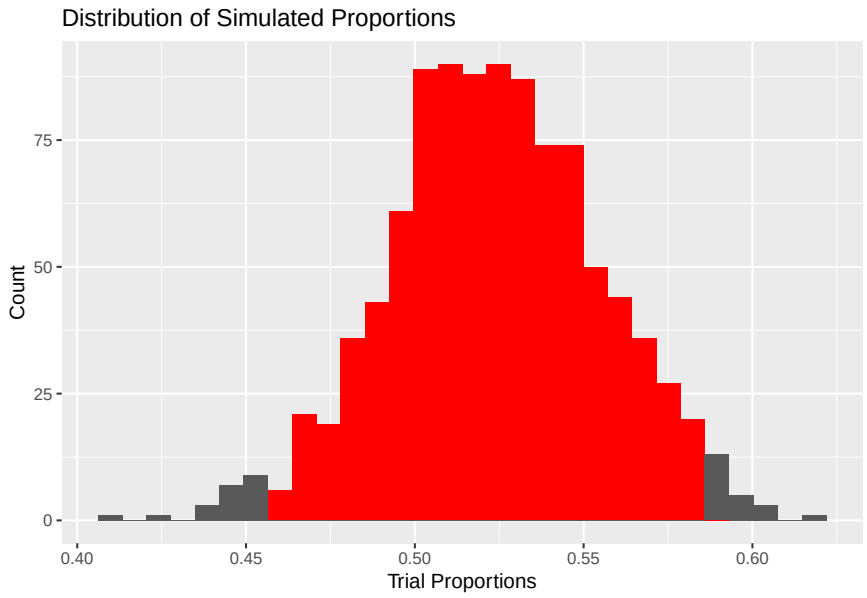
Let's use the Halloween treats example from Chapter 1. I'm going to first simulate a bunch of these experiments. This should look a lot like our code that builds a null distribution, however, this is **not** a null distribution. You should notice it is centered at our sample statistic. There is really no reason you would build this distribution in the normal course of events but I'm doing it here for instructional purposes.

```
library(tidyverse)

replications_dataframe = NULL

num_reps = 1000

sample_size = 135 + 148

sample_stat =  148 / (135 + 148)

for (i in 1:num_reps){

  trial = sample(x = c(1,0),
                 size = sample_size,
                 prob = c(sample_stat, 1-sample_stat),
                 replace = TRUE)

  trial_proportion = sum(trial)/sample_size

  replications_dataframe = rbind(replications_dataframe, data.frame(trial_proportion))
}

replications_dataframe %>%
  ggplot(aes(x = trial_proportion)) +
  geom_histogram() +
  labs(x = "Trial Proportions", y = "Count", title = "Distribution of Simulated Proportions") +
  geom_vline(xintercept = sample_stat, color = "red")
```



If we are looking for a 95% CI, we are looking for the range of values in which we capture the center 95% of these values.

```
#A lot of "instructional purpose" code here... and a little bit of 2SD cheating
below <- replications_dataframe %>%
  filter(trial_proportion <= (mean(trial_proportion) - 2*sd(trial_proportion)))

middle <- replications_dataframe %>%
  filter(trial_proportion >= (mean(trial_proportion) - 2*sd(trial_proportion))) %>%
  filter(trial_proportion <= (mean(trial_proportion) + 2*sd(trial_proportion)))

above <- replications_dataframe %>%
  filter(trial_proportion >= (mean(trial_proportion) + 2*sd(trial_proportion)))

ggplot(data = below, aes(x = trial_proportion)) +
  geom_histogram() +
  geom_histogram(data = middle, fill = "red") +
  geom_histogram(data = above) +
  labs(x = "Trial Proportions", y = "Count",
       title = "Distribution of Simulated Proportions")
```



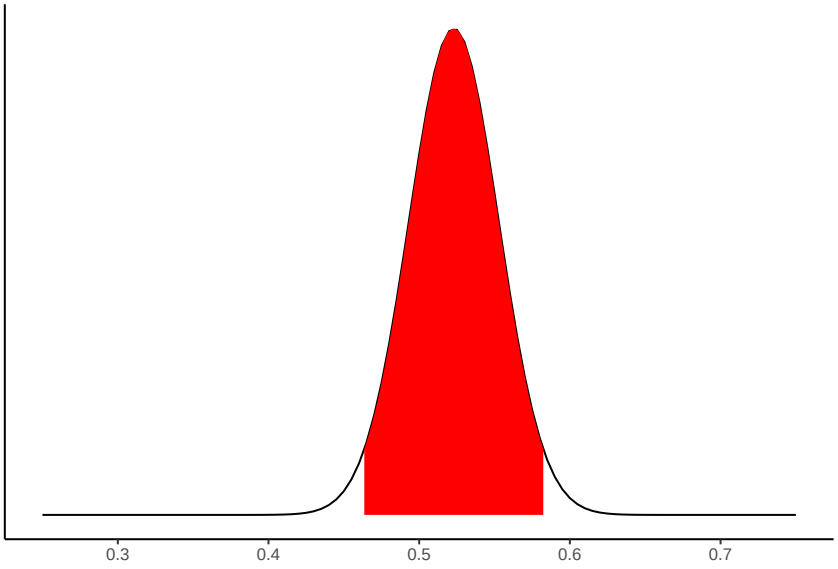
Using the *2SD Method* we could calculate our range of plausible values:

```
#Calculate "SD of Statistic"
standard_error = sqrt((sample_stat * (1-sample_stat))/sample_size)

lower = sample_stat - 2 * standard_error

upper = sample_stat + 2 * standard_error

#Fancier than you need
paste("(",lower, ",", upper, ")")
```

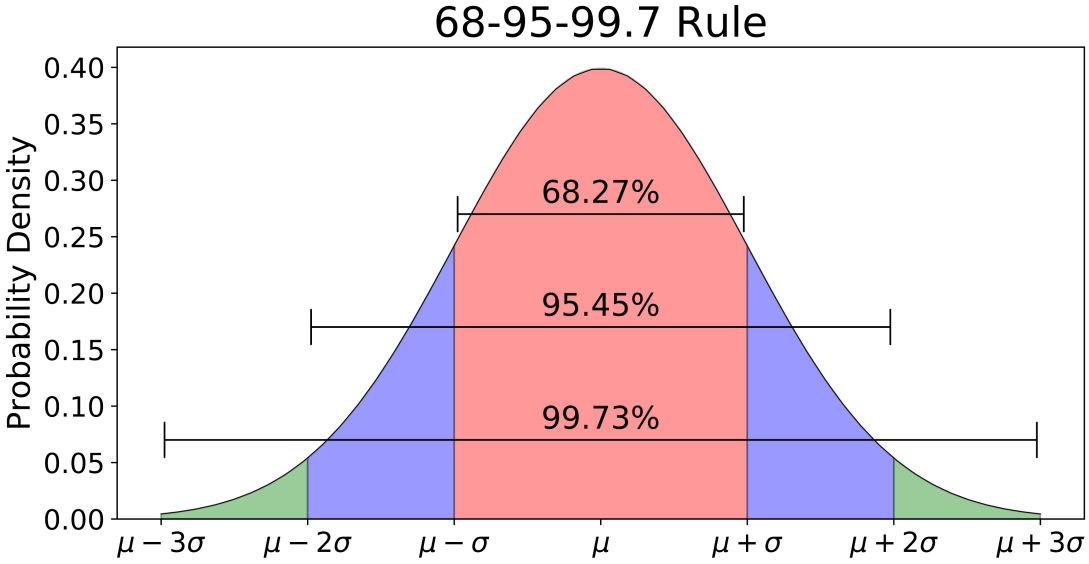
```
## [1] "( 0.463587118944786 , 0.582349276814931 )"
```

However, since we have learned that the central limit theory applies to sample proportions (as long as we meet the validity conditions), we could use the normal distribution to represent the above distribution and make our life a lot easier.

```
ggplot(NULL, aes(x = c(0.25, 0.75))) +
  stat_function(fun = dnorm,
                args = list(mean = sample_stat, sd = standard_error)) +
    stat_function(fun = dnorm,
                args = list(mean = sample_stat, sd = standard_error),
                xlim = c(lower, upper),
                geom = "area", fill = "red") +
  labs(x = "", y = "") + theme_classic() +
  scale_y_continuous(breaks = NULL)
```



(We probably talked about this in class so I'm going to pretend like we did.) You recall that there is a useful property of the normal distribution that allows us to estimate how much of a population falls within the first few standard deviations of the mean as depicted in the image below:



Hopefully you see where the *2SD Method* came from for estimating a 95% CI. We can use **pnorm**'s brother (sister, friend, etc.) **qnorm** for a very similar task however. My saying for **qnorm()** is: "give me an area and I'll give you a x-value." If we want to find the critical value (or multipler) we can use **qnorm()** by giving is the area we want in the left tail for a 95% CI: 0.025.

```
#Left critical value
qnorm(0.025)
```

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

```
#Right critical value
qnorm(0.975)
```

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

```
lower = sample_stat + qnorm(0.025) * standard_error

upper = sample_stat + qnorm(0.975) * standard_error

paste("(",lower, ",", upper, ")")
```

```
## [1] "( 0.464775809841923 , 0.581160585917795 )"
```

Our critical values are slightly less than 2... which is what you should expect based on our *Rule* above. You'll also notice that the "lower" calculation has a plus-sign because the critical value is already negative. Clearly there is no need to actually calculate both critical values as you can just calculate one and switch the signs. To take this one step further, we can actuall provide **qnorm()** with our sample mean and standard error to remove one of the steps here. This, in essence, shifts our normal distribution to reflect the simulation distribution I built above.

```
lower = qnorm(0.025, mean = sample_stat, sd = standard_error)

upper = qnorm(0.975, mean = sample_stat, sd = standard_error)

paste("(",lower, ",", upper, ")")
```

```
## [1] "( 0.464775809841923 , 0.581160585917795 )"
```

```
replications_dataframe %>%
  ggplot(aes(x = trial_proportion)) +
  geom_histogram(aes(y = ..density..)) +
  stat_function(fun = dnorm,
                args = list(mean = sample_stat, sd = standard_error),
                color = "blue") +
  geom_vline(xintercept = lower, color = "red") +
  geom_vline(xintercept = upper, color = "red") +
  labs(x = "Trial Proportions", y = "Density",
       title = "Distribution of Simulated Proportions")
```