Using the computer for statistical inference

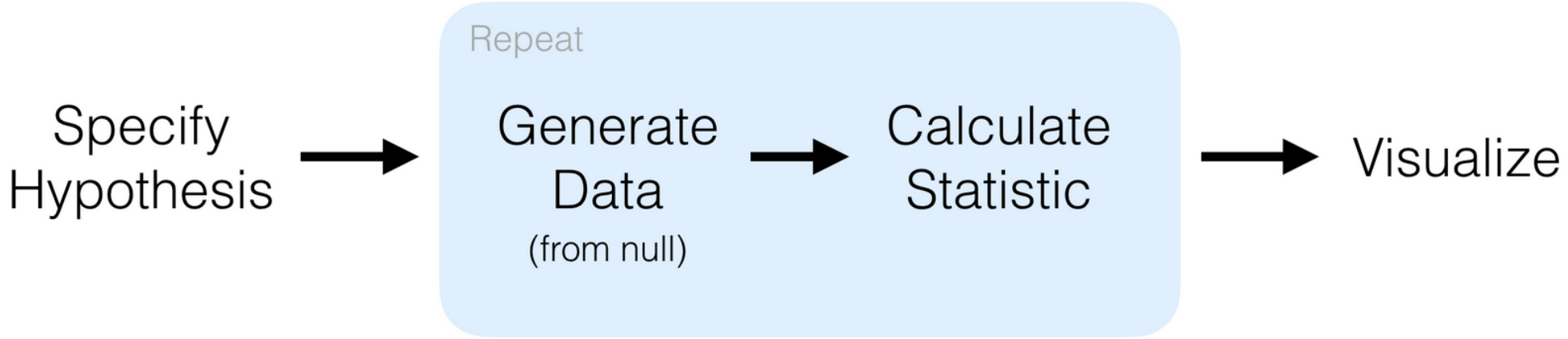
Abhijit Dasgupta

BIOF 339

A bit of computational inference

Computational inference

- Using the computer and simulation to simulate null distributions or sampling distributions of statistics
  - *sampling distribution* is the distribution of values we'd see for a statistics if we repeated the experiment over and over
  - *null distribution* is the distribution of values we'd expect to see if the null hypothesis we're using happens to be true.



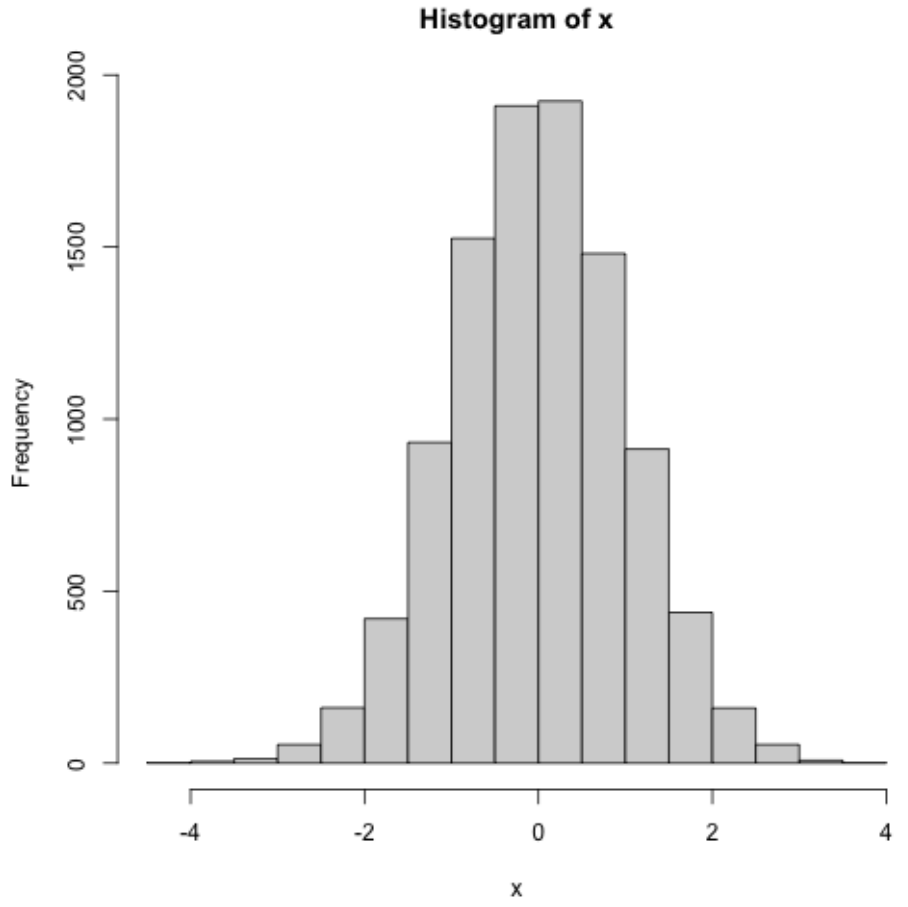
Simulation

R has several standard distributions programmed in, from which random numbers can be drawn and distributions visualized

Simulation

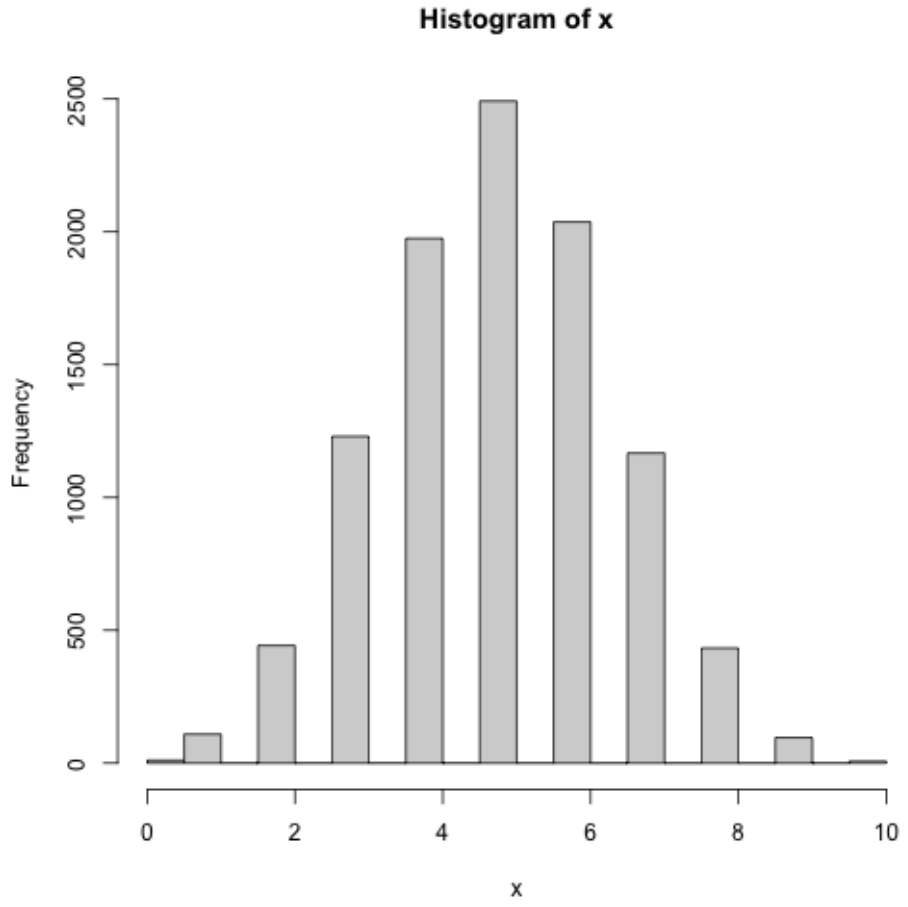
The Gaussian or normal distribution

```
x <- rnorm(10000) # 10,000 random numbers from standa
hist(x) # This is the base R way to create a histogra
```



Simulation

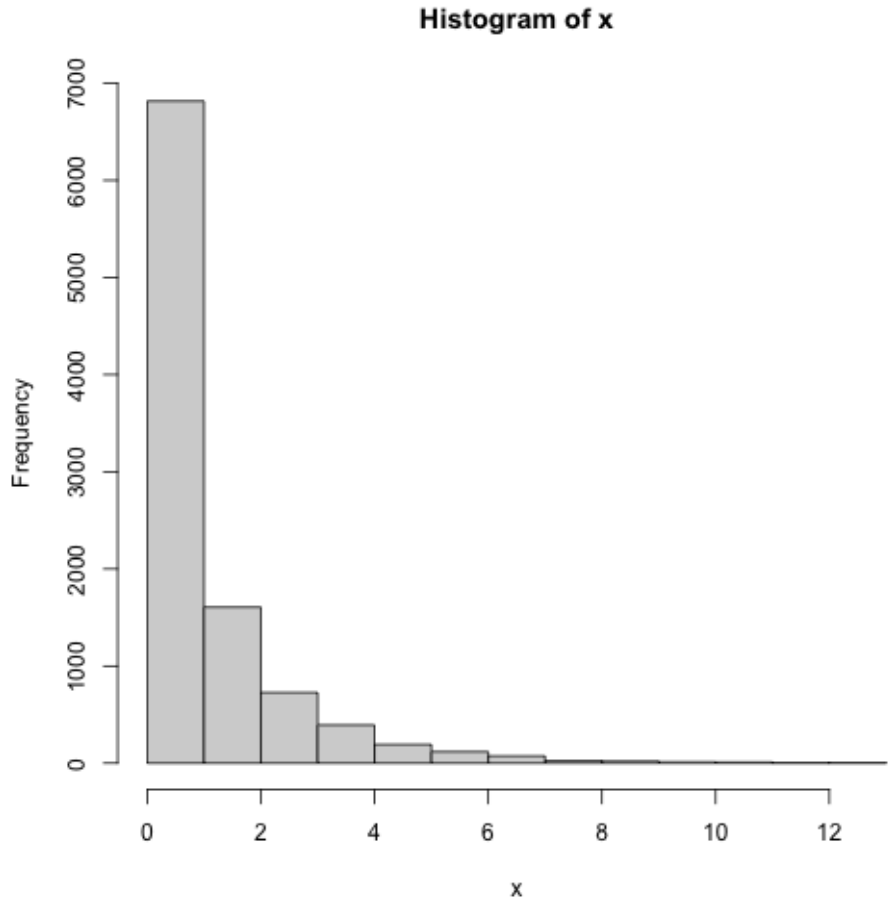
```r
# Toss a fair coin 10 times, count number of heads
# Repeat 10,000 times
x <- rbinom(10000, size = 10, prob = 0.5)
hist(x)
```



Simulation

```
# 10,000 random dumbers from a chi-square distributio
# with 1 d.f.

x <- rchisq(10000, 1)
hist(x)
```



Simulations

Simulations on a computer aren't exactly random, but *pseudo-random*. They form a *complex, deterministic* series of numbers which have some properties.

You can set the starting point of the series. It's called the **seed**.

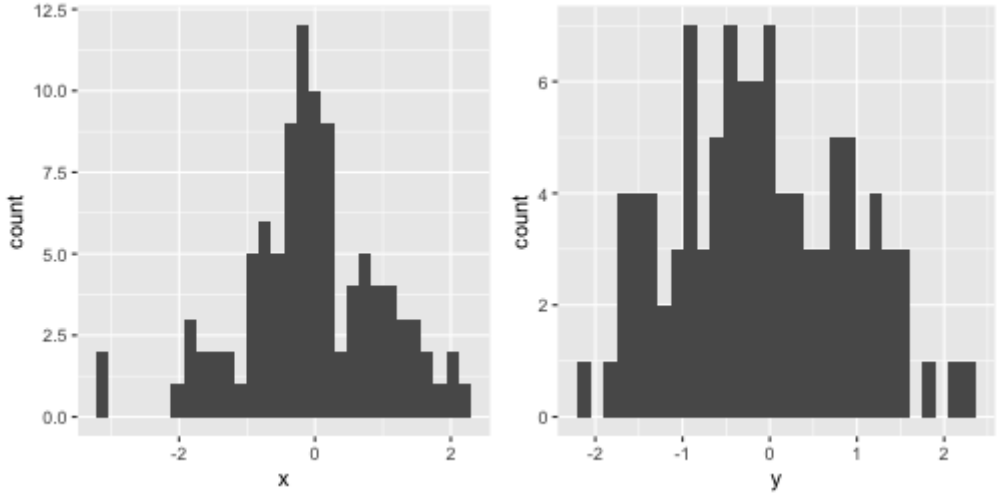
```r
set.seed(28954)

dd <- data.frame(x = rnorm(100))

dd$y <- rnorm(100)

plt1 <- ggplot(dd, aes(x))+geom_histogram()
plt2 <- ggplot(dd, aes( y)) + geom_histogram()

cowplot::plot_grid(plt1, plt2, nrow=1)
```

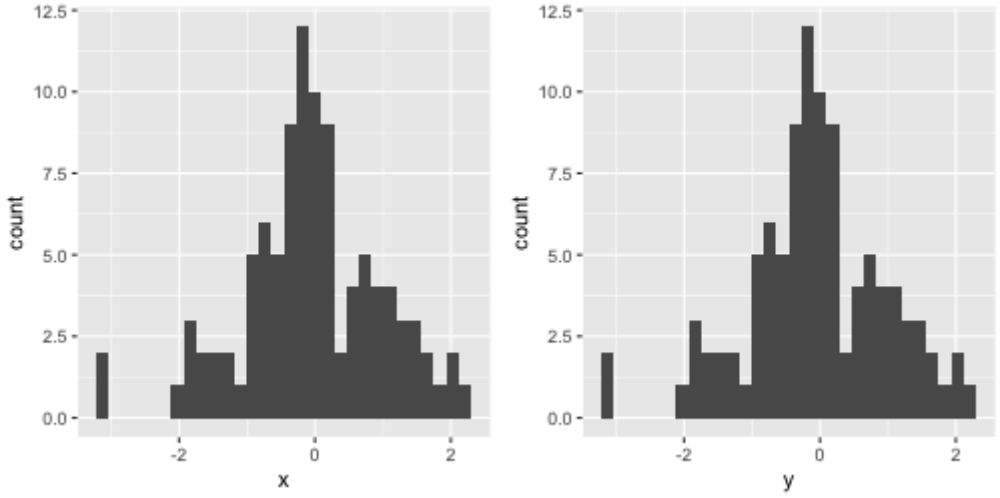


```r
set.seed(28954)

dd <- data.frame(x = rnorm(100))
set.seed(28954)
dd$y <- rnorm(100)

plt1 <- ggplot(dd, aes(x))+geom_histogram()
plt2 <- ggplot(dd, aes( y)) + geom_histogram()

cowplot::plot_grid(plt1, plt2, nrow=1)
```



Always set a seed to ensure replicability of simulation experiments

Permutation tests

We will use the R package `infer` for this section, as well as the `pbc` data. We'll first do a permutation test to see if bilirubin is significantly different by treatment group.

A classical thing to do would be a `t.test` or a `wilcox.test`.

```
library(survival)

t.test(bili~trt, data=pbc)
```

```
    Welch Two Sample t-test

data:  bili by trt
t = -1.5074, df = 270.39, p-value = 0.1329
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.7878314  0.2372643
sample estimates:
mean in group 1 mean in group 2
       2.873418        3.648701
```

Permutation tests

In a permutation test, we assume the null hypothesis of no difference between the groups, which means

- if we shuffled the group memberships (re-assigned individuals to different treatments) nothing should change.

If we compute the test statistic over different permutations, we'll get the distribution of test statistic values we'd see if the null hypothesis was true.

Permutation tests

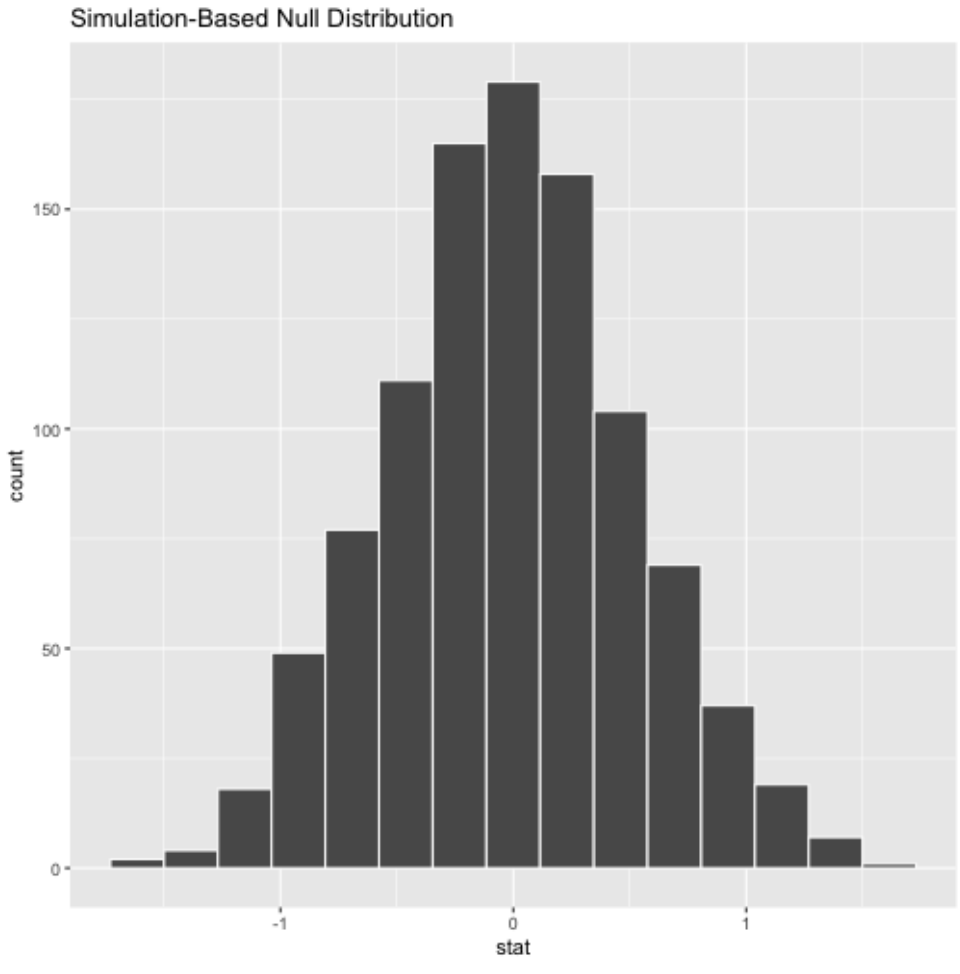
```
library(infer)

set.seed(10193)
sims <- pbc %>%
  mutate(trt = as.factor(trt)) %>%
  specify(bili ~ trt) %>%
  hypothesize(null = 'independence') %>%
  generate(reps = 1000, type = 'permute') %>%
  calculate(stat = 'diff in means', order = c('1','2'

visualize(sims)
```

```
library(infer)
(obs_stat <- pbc %>% mutate(trt=as.factor(trt)) %>%
  specify(bili ~ trt) %>%
  calculate(stat='diff in means',order = c('1','2')))
```

```
# A tibble: 1 x 1
    stat
   <dbl>
1 -0.775
```



Permutation tests

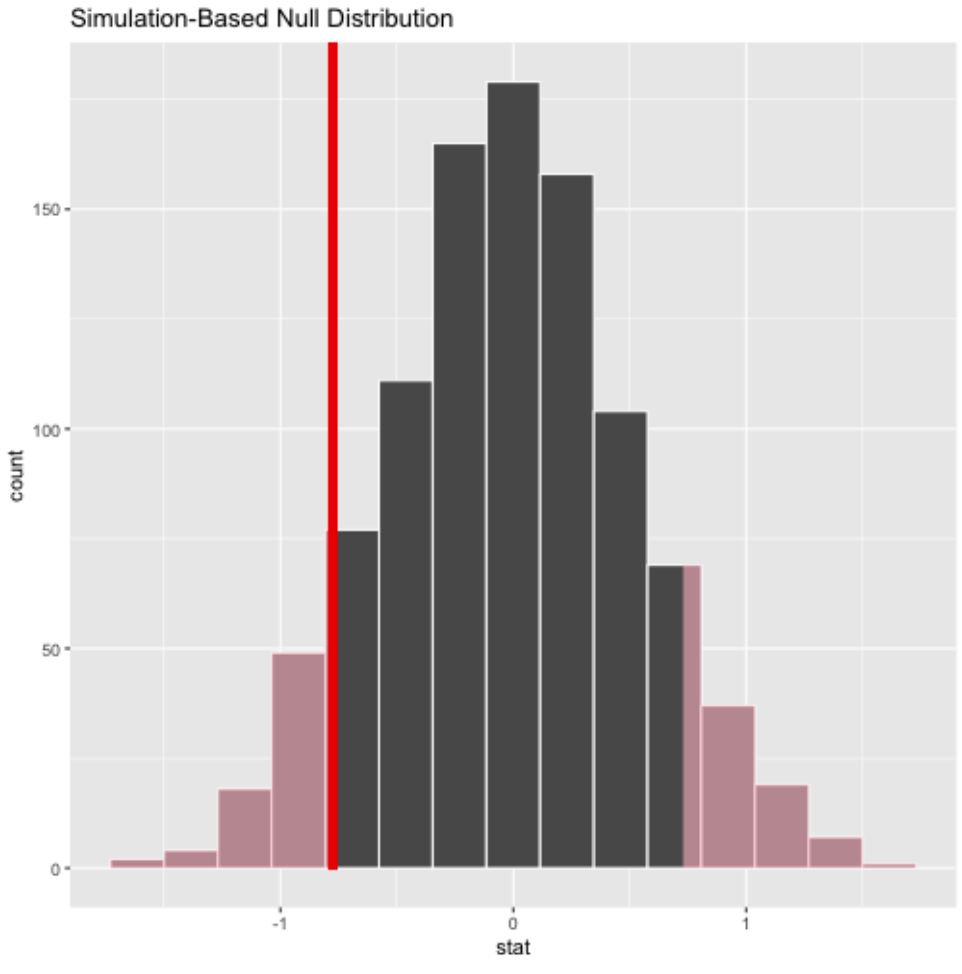
```
library(infer)

set.seed(10193)
sims <- pbc %>%
  mutate(trt = as.factor(trt)) %>%
  specify(bili ~ trt) %>% # trt must be a factor
  hypothesize(null = 'independence') %>%
  generate(reps = 1000, type = 'permute') %>%
  calculate(stat = 'diff in means', order = c('1','2'

visualize(sims) + shade_p_value(obs_stat, direction =
```

```
sims %>% get_pvalue(obs_stat, direction ='both')
```

```
# A tibble: 1 x 1
  p_value
    <dbl>
1   0.158
```



Bootstrapping for confidence intervals

Suppose we want to get a confidence interval for the mean bilirubin level overall.

The bootstrap samples the original data **with replacement** to get a dataset of the same size.

Since sampling is with replacement, some observations are repeated, some are omitted.

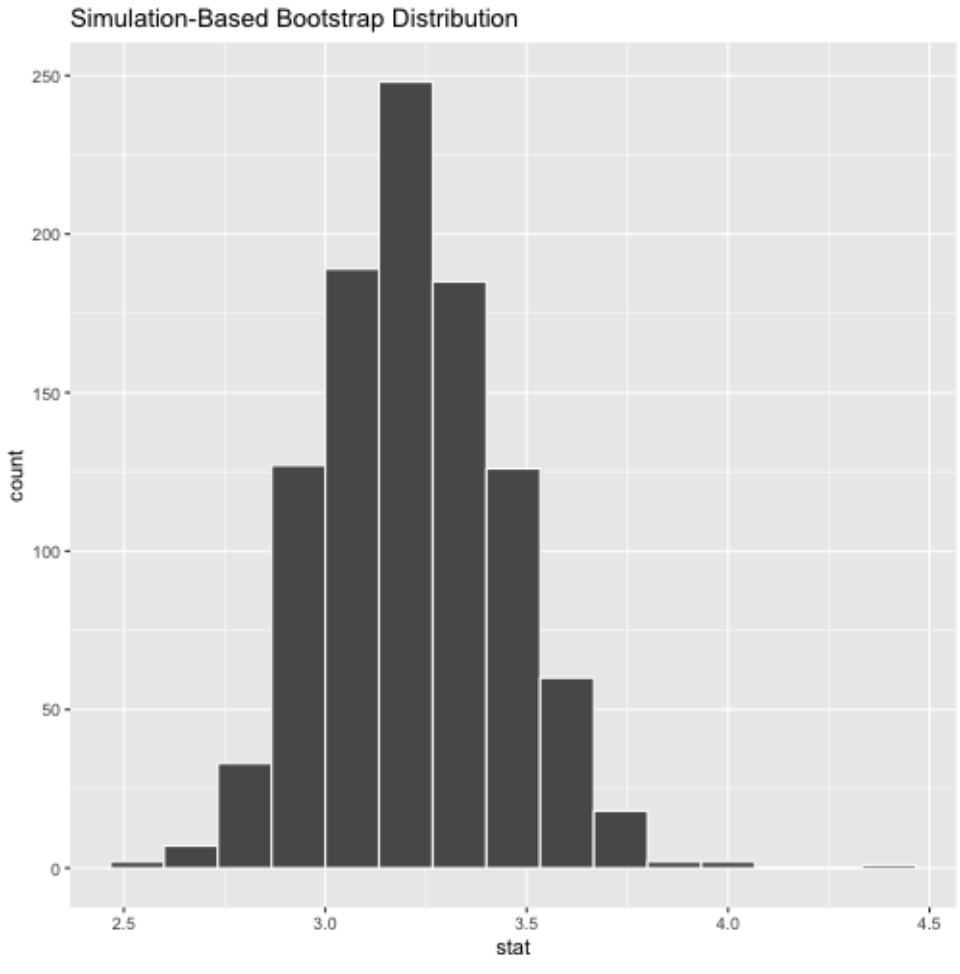
Strong theory from the 80s and 90s says that if we repeatedly take bootstrap samples of our data and compute the sample means, their distribution will be very close to the true sampling distribution of the sample mean.

Bootstrapping for confidence intervals

```
x <- pbc %>%
  specify(response = bili) %>%
  generate(reps = 1000, type = 'bootstrap') %>%
  calculate('mean')
```

```
(ci <- x %>% get_confidence_interval())
```

```
# A tibble: 1 x 2
  lower_ci upper_ci
     <dbl>    <dbl>
1     2.84     3.66
```



Resources

1. Several infer examples
2. The R Companion chapter on permutation tests
3. This site gives alternative methods for doing permutation tests in R
4. The coin package provides a richer set of permutation tests, but `infer` covers what you need most often.