Load necessary packages

```
library(tidyverse)
library(broom)
library(xlsx)
library(infer)
```

Load the file and prepare the data for analysis

```
# only matched data for simplicity
df <- read.csv('C:/Users/Cole/Documents/DATA/PLIC_DATA/Collective_Surveys/Merged/Merged_Concat.csv') %>%
  select(Q6b_y, Q6b.i_y, Q6e_1_y, Q6e_2_y, PreScores, PostScores) %>%
  mutate(Gender = case_when( # convert gender and major columns to something readable
    Q6e_1_y == 1 ~ 'M',
    Q6e_2_y == 1 ~ 'F'
  ),
  Major = case_when(
    Q6b_y < 4 ~ 'Physics',
    Q6b.i_y == 1 ~ 'Engineering',
    Q6b.i_y == 2 | Q6b.i_y == 3 ~ 'Other Science',
    Q6b.i_y == 4 ~ 'Other'
  )) %>%
  select(Gender, Major, PreScores, PostScores)

glimpse(df)
```

```
## Observations: 5,050
## Variables: 4
## $ Gender     <chr> "M", "M", "M", "M", "M", "M", "M", "M", "M", "F", "...
## $ Major      <chr> "Other Science", "Engineering", "Engineering", "Eng...
## $ PreScores  <dbl> 5.071825, 4.161111, 2.746474, 5.476863, 5.380827, 3...
## $ PostScores <dbl> 0.125000, 2.468254, 1.650000, 1.440293, 5.851564, 5...
```

```
head(df, 20)
```

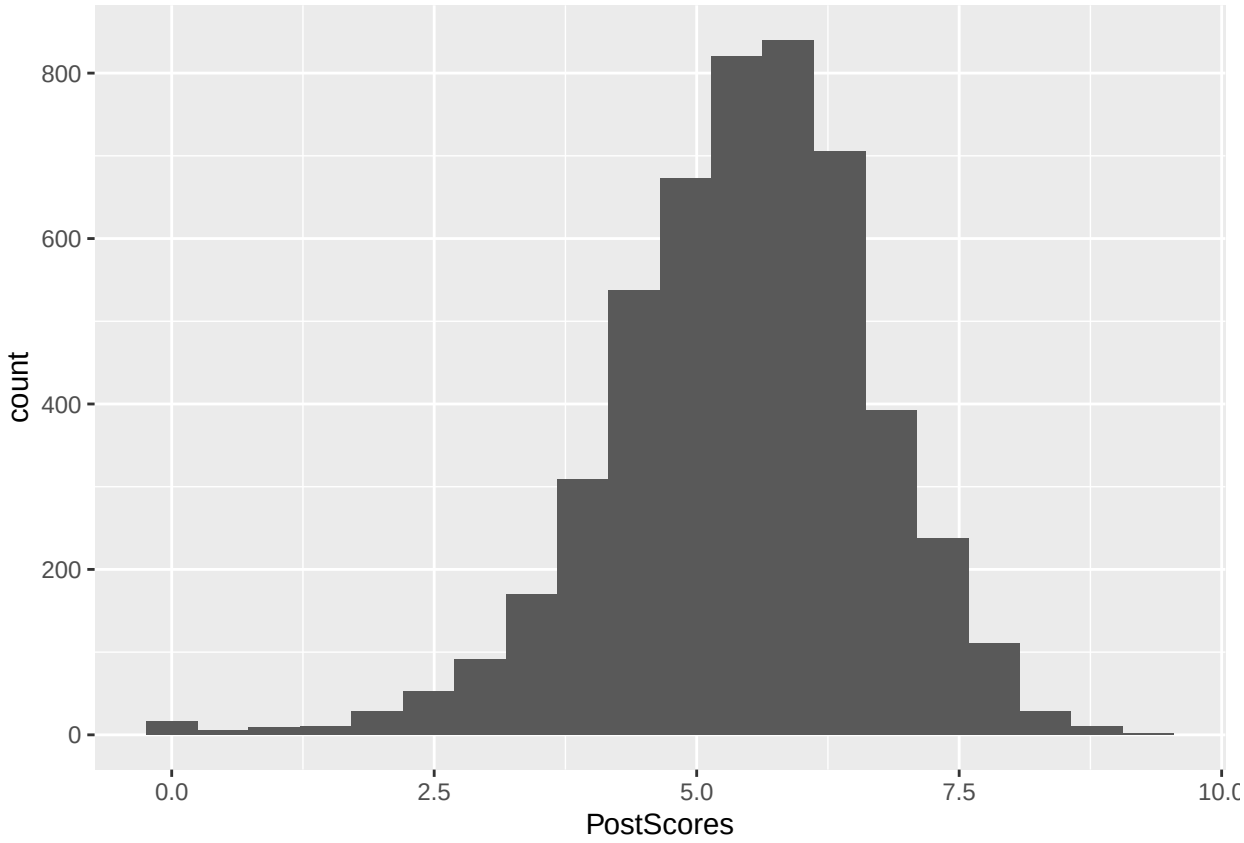
```
##    Gender         Major PreScores PostScores
## 1       M Other Science  5.071825   0.125000
## 2       M   Engineering  4.161111   2.468254
## 3       M   Engineering  2.746474   1.650000
## 4       M   Engineering  5.476863   1.440293
## 5       M          <NA>  5.380827   5.851564
## 6       M Other Science  3.628388   5.001851
## 7       M   Engineering  4.523034   7.316132
## 8       M       Physics  6.051302   7.932403
## 9       M   Engineering  3.761529   5.699128
## 10      F Other Science  4.935374   2.869963
## 11      M   Engineering  3.532468   7.225925
## 12      F   Engineering  3.335150   0.100000
## 13      M   Engineering  2.480037   4.951305
```

```
## 14      M Other Science  3.940760   2.825000
## 15      F   Engineering  2.587970   6.752665
## 16      M   Engineering  6.008516   6.225950
## 17      F   Engineering  3.660652   4.851578
## 18      F   Engineering  5.832683   5.649935
## 19      F   Engineering  4.709117   3.691192
## 20      F   Engineering  6.697073   6.355873
```

Bootstrapping means, medians, and other descriptive stats

Summary statistics

```
ggplot(df, aes(x = PostScores)) +
  geom_histogram(bins = 20)
```




```
Sample_Mean <- mean(df$PostScores)

# Standard Error
Sample_Mean_sd <- sd(df$PostScores)/sqrt(nrow(df))

Sample_Mean
```

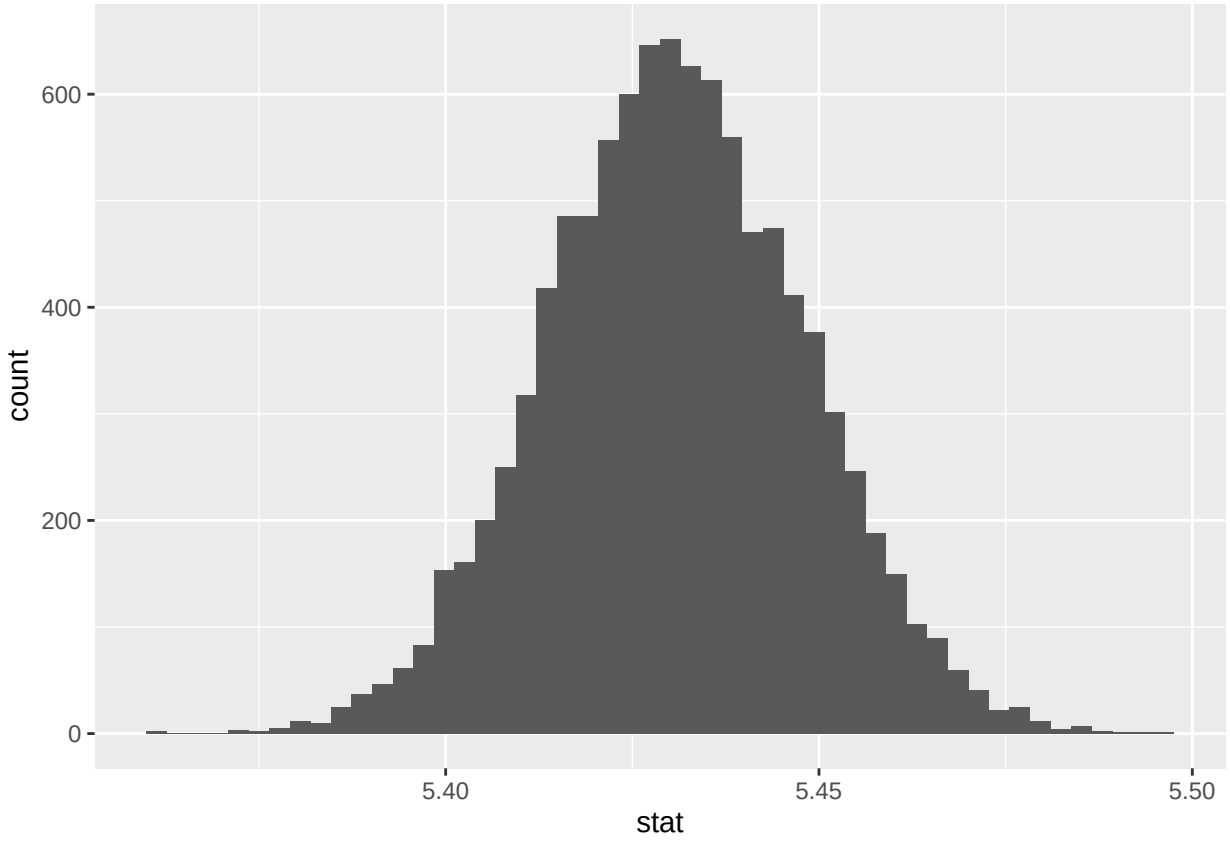
```
## [1] 5.430845
```

```
Sample_Mean_sd
```

```
## [1] 0.01756105
```

Bootstrap mean

```
set.seed(11) # for reproducibility

Bootstrap_Means <- df %>%
  specify(response = PostScores) %>%
  generate(reps = 10000, type = "bootstrap") %>%
  calculate(stat = "mean")

ggplot(Bootstrap_Means, aes(x = stat)) +
  geom_histogram(bins = 50)
```



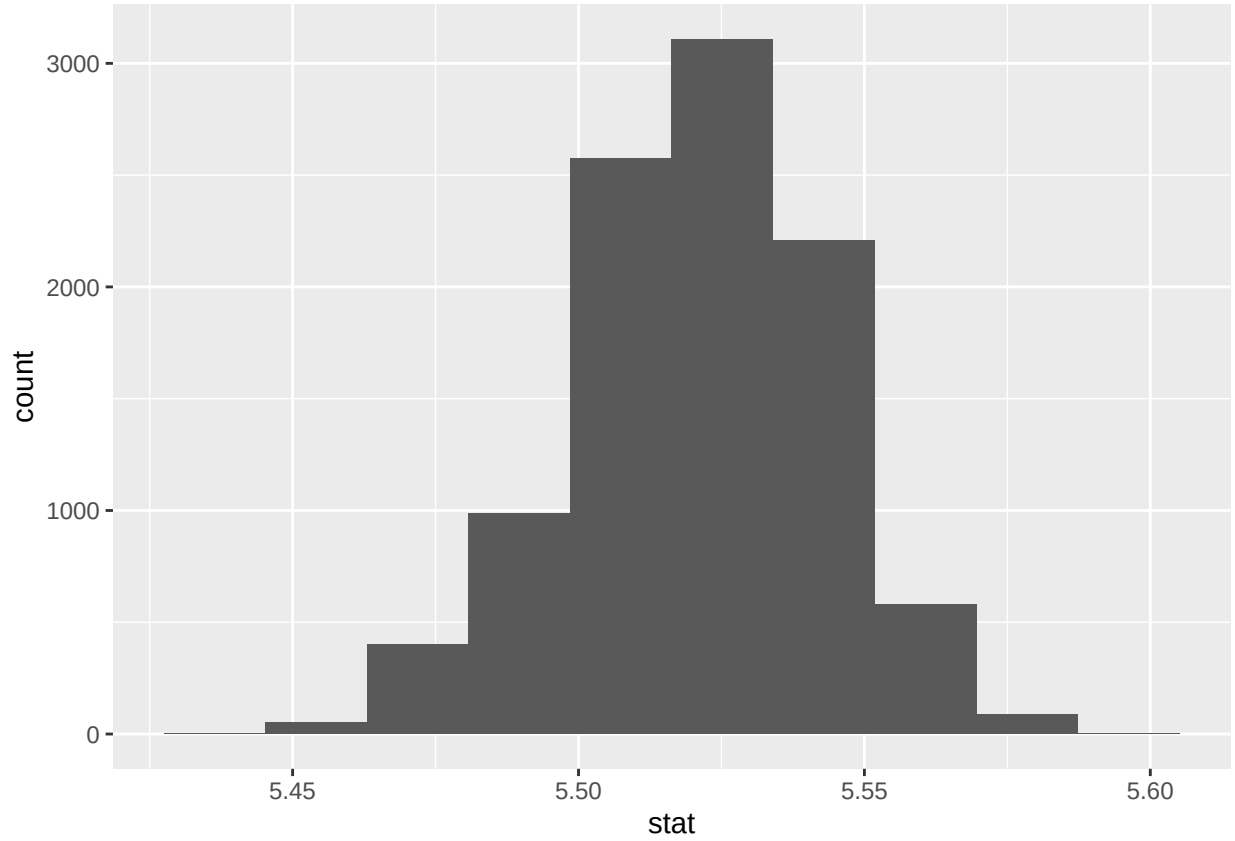

```
Bootstrap_Means %>%
  summarize(Sample_Mean,
            Sample_Mean_sd,
            l = quantile(stat, 0.025),
            u = quantile(stat, 0.975))
```

```
## # A tibble: 1 x 4
##   Sample_Mean Sample_Mean_sd     l     u
##         <dbl>          <dbl> <dbl> <dbl>
## 1        5.43         0.0176  5.40  5.46
```

The mean and standard deviation of the distribution of bootstrapped means is pretty much identical to the mean and standard error of the sample.

Bootstrap median

```
set.seed(11)

Sample_Median <- median(df$PostScores)

Bootstrap_Medians <- df %>%
  specify(response = PostScores) %>%
  generate(reps = 10000, type = "bootstrap") %>%
  calculate(stat = "median")

ggplot(Bootstrap_Medians, aes(x = stat)) +
  geom_histogram(bins = 10)
```



```
Bootstrap_Medians %>%
  summarize(Sample_Median,
            se = sd(stat),
            l = quantile(stat, 0.025),
            u = quantile(stat, 0.975))
```

```
## # A tibble: 1 x 4
##   Sample_Median     se     l     u
##           <dbl>  <dbl> <dbl> <dbl>
## 1          5.53 0.0222  5.47  5.56
```

The central limit theorem ensures that the distribution of any aggregated measure (such as mean or meadian) with finite variance will be normally distributed. Bootstrapping (and analagous resampling methods) allows us to take advantage of this theorem to calculate confidence intervals and conduct hypothesis tests with many aggregate measures that are not necessarily normally distributed.

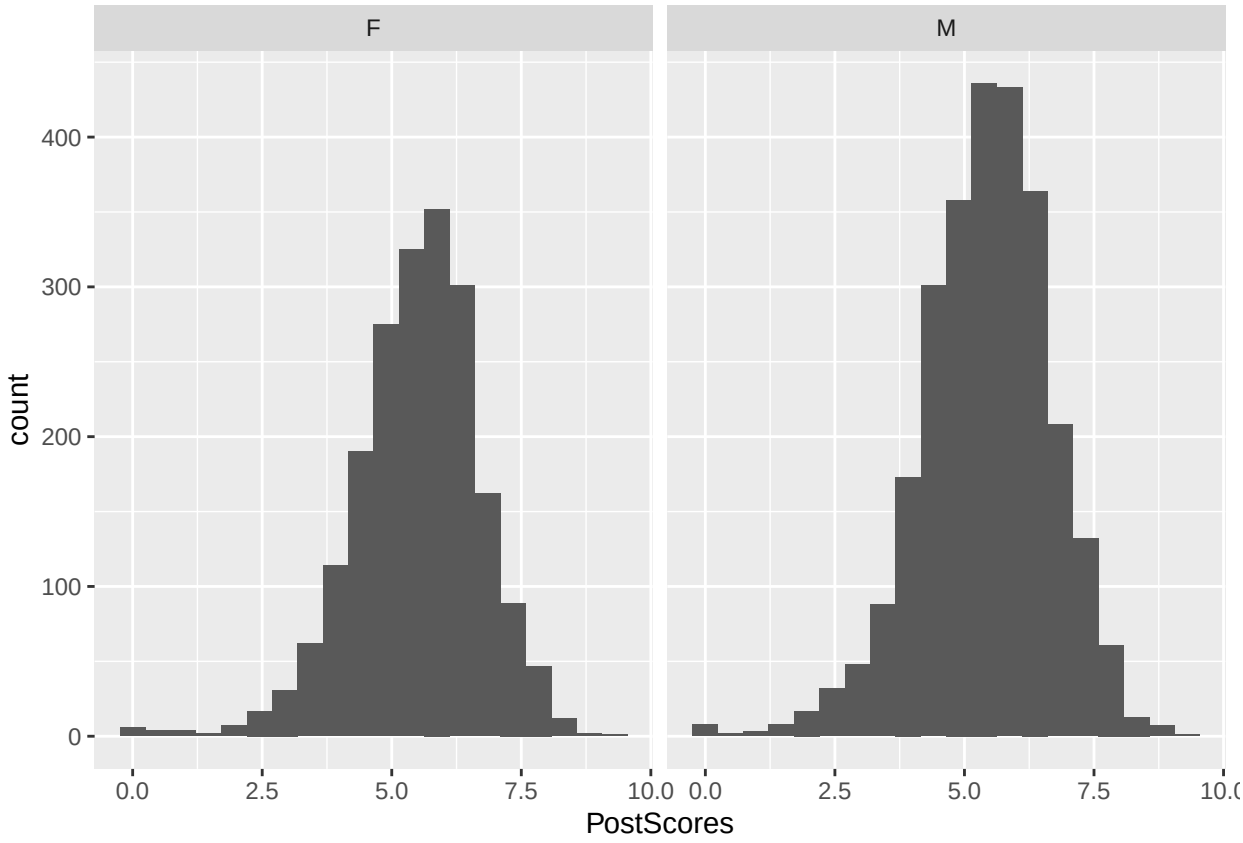
Bootstrapping and linear regression

Summary statistics and histograms

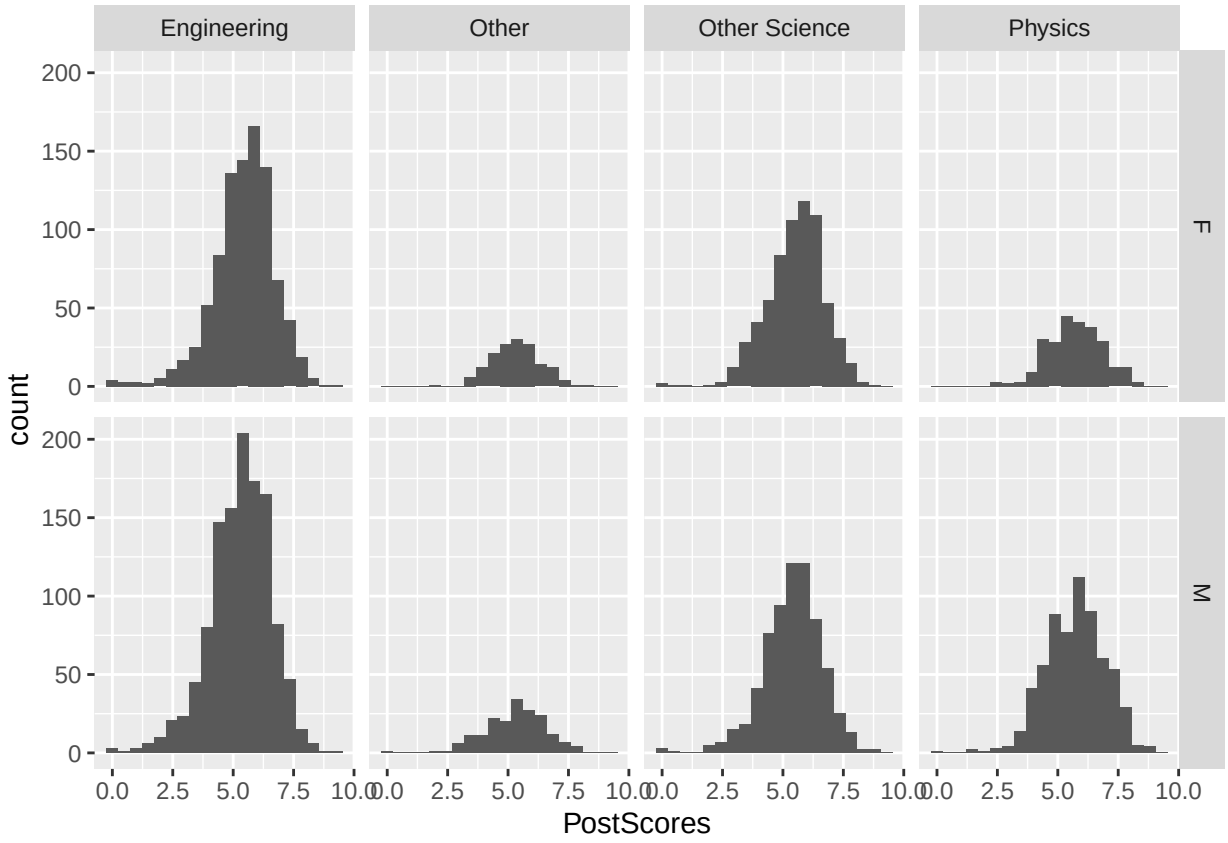
```
df_filtered <- df %>%
  filter(!is.na(Gender) & !is.na(Major)) # get only complete demographic data

glimpse(df_filtered)
```

```
## Observations: 4,696
## Variables: 4
## $ Gender     <chr> "M", "M", "M", "M", "M", "M", "M", "M", "F", "M", "...
## $ Major      <chr> "Other Science", "Engineering", "Engineering", "Eng...
## $ PreScores  <dbl> 5.071825, 4.161111, 2.746474, 5.476863, 3.628388, 4...
## $ PostScores <dbl> 0.125000, 2.468254, 1.650000, 1.440293, 5.001851, 7...
```

```
ggplot(df_filtered, aes(x = PostScores)) +
  geom_histogram(bins = 20) +
  facet_wrap(~Gender)
```



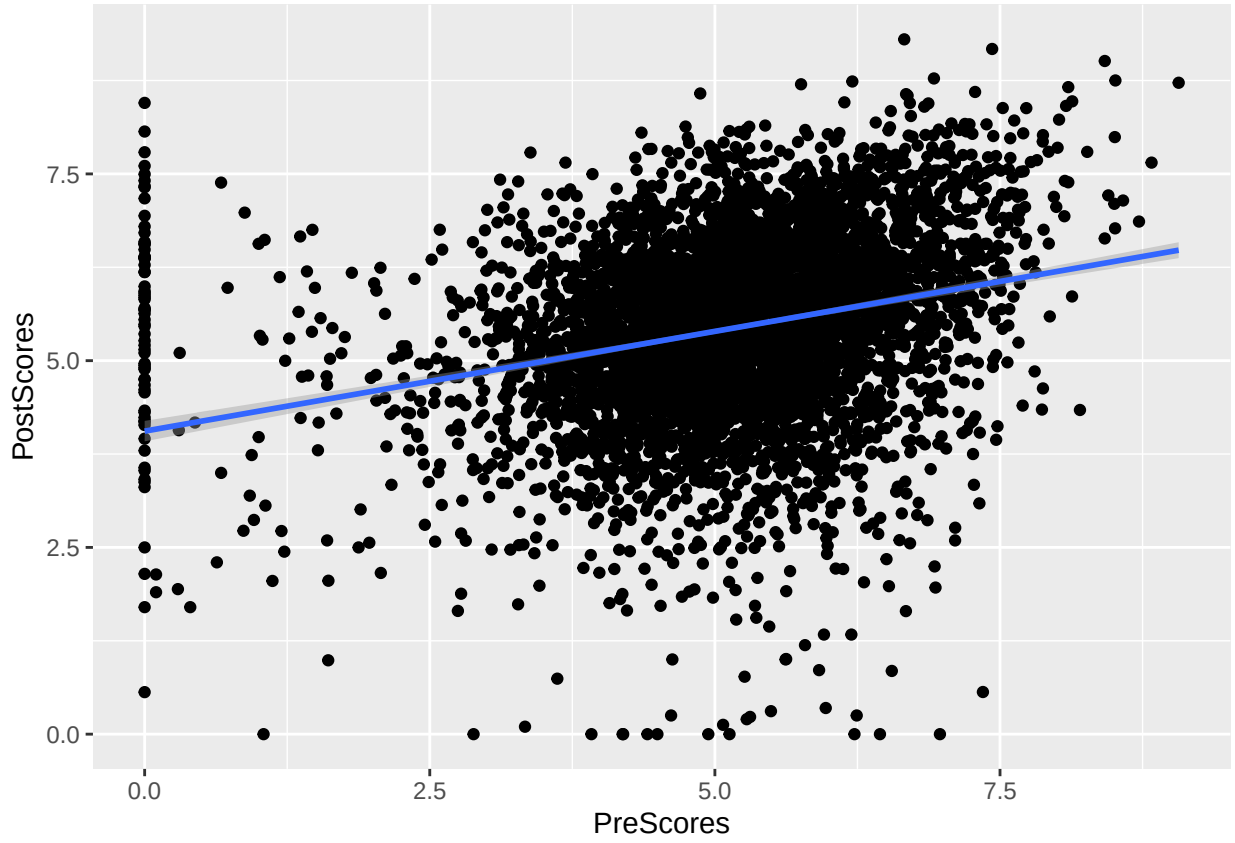

```
ggplot(df_filtered, aes(x = PostScores)) +
  geom_histogram(bins = 20) +
  facet_grid(Gender ~ Major)
```



Regression with sample data

```
ggplot(df, aes(x = PreScores, y = PostScores)) +
  geom_point() +
  geom_smooth(method = 'lm')
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
summary(lm(PostScores ~ PreScores, data = df))
```

```
##
## Call:
## lm(formula = PostScores ~ PreScores, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.9195 -0.6911  0.0872  0.7829  4.3941
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.05648    0.06984   58.08   <2e-16 ***
## PreScores    0.26717    0.01317   20.28   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.2 on 5048 degrees of freedom
## Multiple R-squared:  0.07534,    Adjusted R-squared:  0.07515
## F-statistic: 411.3 on 1 and 5048 DF,  p-value: < 2.2e-16
```

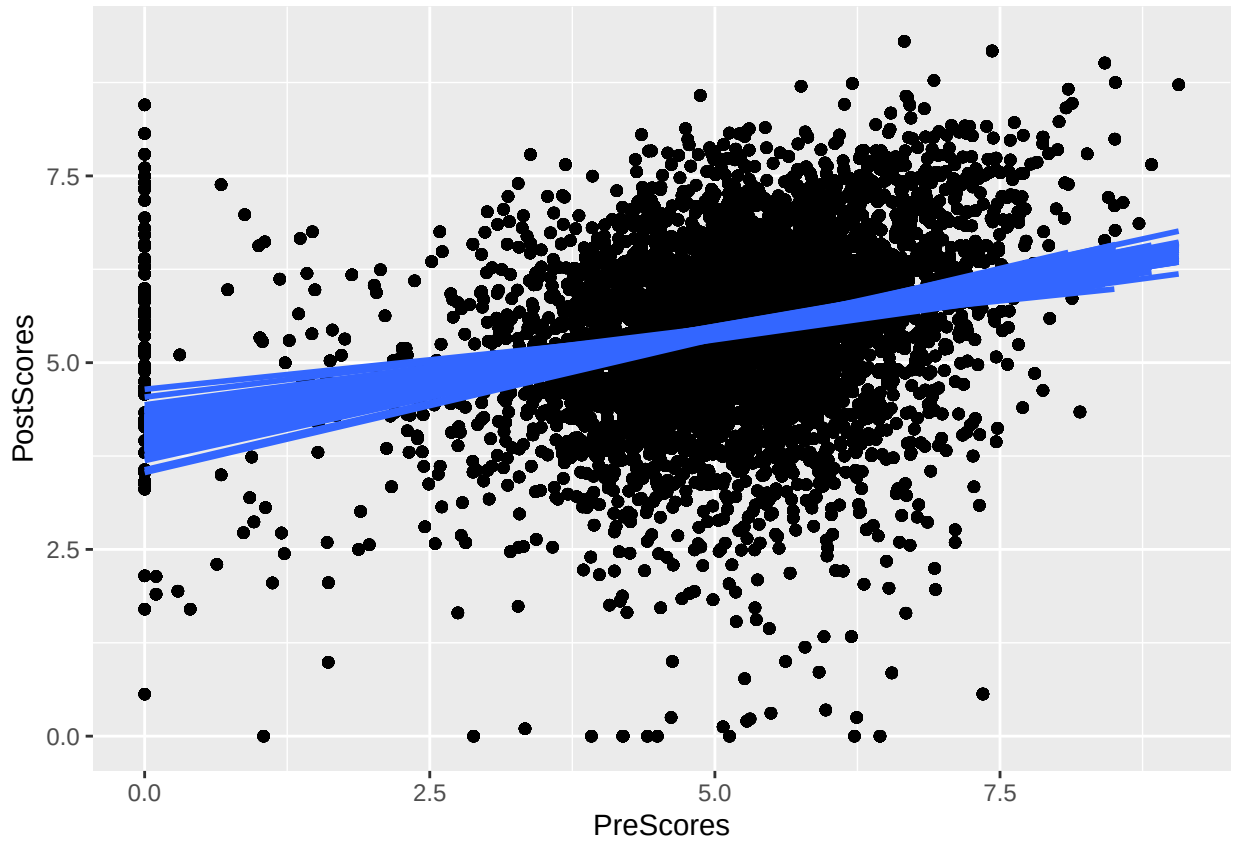
Bootstrapped regression

```
set.seed(11)

# sample sizes of 300, no replacement...I don't want this to run forever
Many_Samples <- rep_sample_n(df_filtered, 1000, reps = 100)

glimpse(Many_Samples)
```

```
## Observations: 100,000
## Variables: 5
## Groups: replicate [100]
## $ replicate  <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ Gender     <chr> "F", "M", "M", "F", "F", "M", "M", "M", "M", "F", "...
## $ Major      <chr> "Other Science", "Engineering", "Other Science", "E...
## $ PreScores  <dbl> 7.307555, 2.746474, 5.353167, 5.459148, 6.502381, 3...
## $ PostScores <dbl> 6.352426, 1.650000, 5.419178, 3.256746, 3.706906, 4...
```

```
# Plot the regression lines
ggplot(Many_Samples, aes(x = PreScores, y = PostScores, group = replicate)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE)
```

```
## `geom_smooth()` using formula 'y ~ x'
```

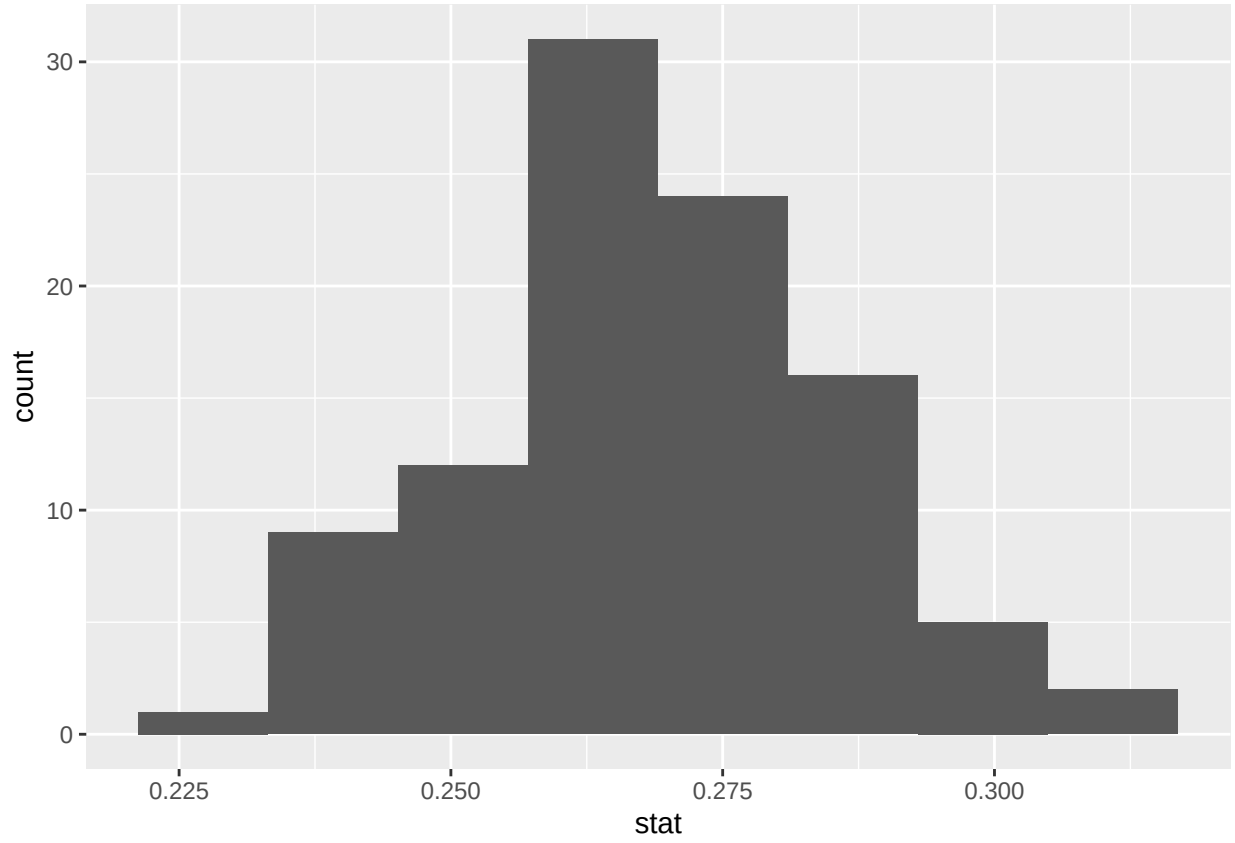


Distribution of bootstrapped slopes

```
Sample_Slope <- lm(PostScores ~ PreScores, data = df_filtered) %>%
  tidy() %>%
  filter(term == "PreScores") %>%
  summarize(estimate, std.error)

Sample_Slope
```

```
## # A tibble: 1 x 2
##   estimate std.error
##      <dbl>     <dbl>
## 1    0.265    0.0136
```

```
Bootstrap_Slopes <- df_filtered %>%
  specify(PostScores ~ PreScores) %>%
  generate(reps = 100, type = "bootstrap") %>%
  calculate(stat = "slope")

ggplot(Bootstrap_Slopes, aes(x = stat)) +
  geom_histogram(bins = 8)
```



```r
Bootstrap_Slopes %>%
  summarize(slope = Sample_Slope$estimate,
            se = sd(stat),
            l = quantile(stat, 0.025),
            u = quantile(stat, 0.975))
```

```
## # A tibble: 1 x 4
##   slope     se     l     u
##   <dbl>  <dbl> <dbl> <dbl>
## 1 0.265 0.0167 0.238 0.302
```

The mean and standard deviation of the slope distribution is again pretty much identical to the slope and standard error for our regression above.