CODE EXAMPLE -Bootstrap (Stats 230 - C4.7)

P.B.Matheson adopted from S.M.Liao

We'll use the dataset `FirstYearGPA` in this example. Recall that previously we determined that `SATV` has a significant relationship with `GPA` when `HSGPA` and `FirstGen` are in the model, using a **randomization test**. But what are *reasonable values* of that coefficient? The randomization test does NOT give us a range of reasonable values - instead, it gave us a range of values that would indicate NO relationship (as we constructed the interval using the **simulated "null" distribution**).

In this example we are going to use the **bootstrap** to obtain **bootstrap confidence intervals** for the coefficient of `SATV` for predicting `GPA` when `HSGPA` and `FirstGen` are in the model. The bootstrap distribution is a simulated sampling distribution based on a resampling with replacement using the sample as pseudo-population.

Let's first fit the model (as we did previously) and find out the actual estimate of the coefficient:

```r
m2 <- lm(GPA ~ SATV + HSGPA + FirstGen, data = FirstYearGPA)
msummary(m2)
```

```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.715988   0.295003    2.43    0.016 *
## SATV         0.001012   0.000348    2.91    0.004 **
## HSGPA        0.518540   0.074956    6.92  5.2e-11 ***
## FirstGen    -0.199837   0.089154   -2.24    0.026 *
##
## Residual standard error: 0.402 on 215 degrees of freedom
## Multiple R-squared:  0.263,  Adjusted R-squared:  0.253
## F-statistic: 25.6 on 3 and 215 DF,  p-value: 3.35e-14
```

```r
coef.actual <- m2$coefficients["SATV"]; coef.actual   #extract the estimated coeff. for `SATV`
```

```
##    SATV
## 0.00101
```

```r
#mplot(m2, which = 1)
#mplot(m2, which = 2)
confint(m2)            ## traditional 95% C.I.s (t-intervals) for coefficients
```

```
##                 2.5 %  97.5 %
## (Intercept)  0.134520  1.2975
## SATV         0.000327  0.0017
## HSGPA        0.370798  0.6663
## FirstGen    -0.375564 -0.0241
```

As mentioned previously, we would trust these p-values and CIs IF the conditions for regression are all met. Assuming that we have some concerns, let's see what the **bootstrap CIs** can tell us.

The key idea of **bootstrap** is that the data/sample itself is assumed to be randomly selected from the population, and should closely *resemble the population* (if collected correctly). Thus, we can create many new datasets with the same size by repeatedly sampling from our original dataset **WITH REPLACEMENT**, and construct the distribution of the estimate we are interested in. The bootstrap gives us a non-parametric understanding of the distribution of those estimates. Once again, the advantage to this method is that we can construct meaningful confidence intervals for, say, the slope coefficient of the regression line, without having to assume that the residuals are normally distributed and have the same standard deviations.

Here is how we create a new sample by sampling entire rows/cases *WITH replacement* from our original data.

```
sim1 <- FirstYearGPA[sample(nrow(FirstYearGPA), replace = TRUE), ]
```

`FirstYearGPA[sample(nrow(FirstYearGPA), replace = TRUE), ]` asks `R` to use the `FirstYearGPA` dataset and sample its rows with replacement (`replace = TRUE`) until the size of the original dataset is obtained. The comma (,) at the end and bracket pair are showing that we aren't selecting any variable subset from the `firstYearGPA` dataset; we are keeping ALL variables, but sampling the same number of rows with replacement. We call such a simulated sample **bootstrap sample**.

We'll want to calculate the new estimated coefficient of `SATV` on this bootstrap sample.

```
m.tmp1 <- lm(GPA ~ SATV + HSGPA + FirstGen, data = sim1)
#msummary(m.tmp)
m.tmp1$coefficients["SATV"]
```

```
##    SATV
## 0.00115
```

Let's try it again.

```
sim2 <- FirstYearGPA[sample(nrow(FirstYearGPA), replace = TRUE), ]
m.tmp2 <- lm(GPA ~ SATV + HSGPA + FirstGen, data = sim2)
#msummary(m.tmp)
m.tmp2$coefficients["SATV"]
```

```
##     SATV
## 0.000777
```

The 2 bootstrap estimates of the slope coefficient for `SATV` are similar but not quite the same. They are close to the actual estimated coefficient (i.e. $\widehat{\beta}_{SATV} = 0.001012$). The fact that they are different is showing us sampling variation. We can use these differences to estimate how much we expect results (sample statistics) to vary from one sample to another.

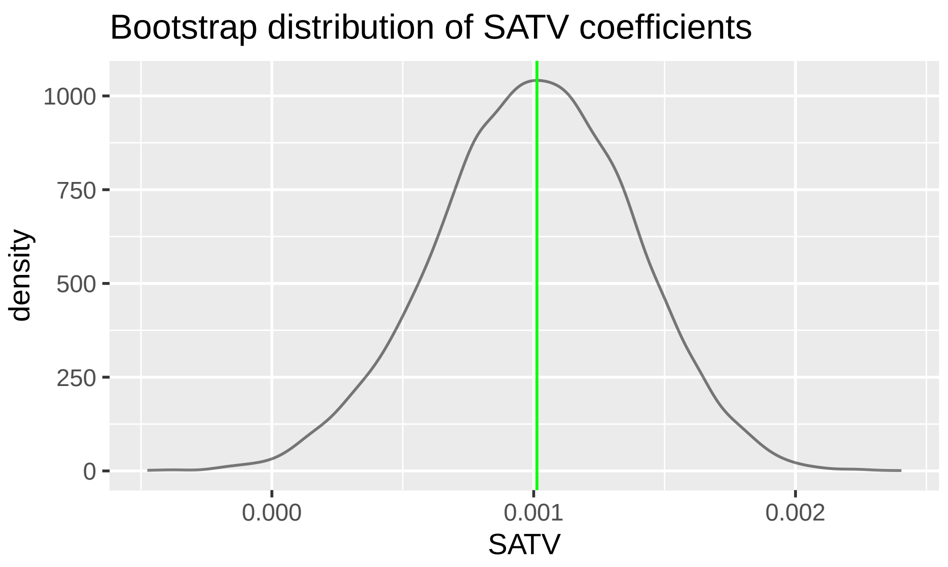
Let's now do this 10,000 times.

```
set.seed(500)
bootstrap <- do(10000)*lm(GPA ~ SATV + HSGPA + FirstGen,
                         data = FirstYearGPA[sample(nrow(FirstYearGPA), replace = TRUE), ])
names(bootstrap)
```

```
##  [1] "Intercept" "SATV"      "HSGPA"     "FirstGen"  "sigma"     "r.squared"
##  [7] "F"         "numdf"     "dendf"     ".row"      ".index"
```

```
gf_dens(~ SATV, data = bootstrap, title="Bootstrap distribution of SATV coefficients") %>%
  gf_vline(xintercept = coef.actual, color = "green")
```

```
## Warning: geom_vline(): Ignoring `mapping` because `xintercept` was provided.
```



This gives us a plot (densityplot) of the slope coefficient for `SATV` based on 10,000 bootstrap coefficients. In contrast to randomization test results, these are *slopes we WOULD expect to get* (though some values are more common than the others). We can then use this bootstrap distribution to construct **bootstrap confidence intervals**.

There are in fact 3 methods for constructing bootstrap CIs from the bootstrap sampling distribution of the statistic. Which one to use depends on the shape of the bootstrap distribution that you just created.

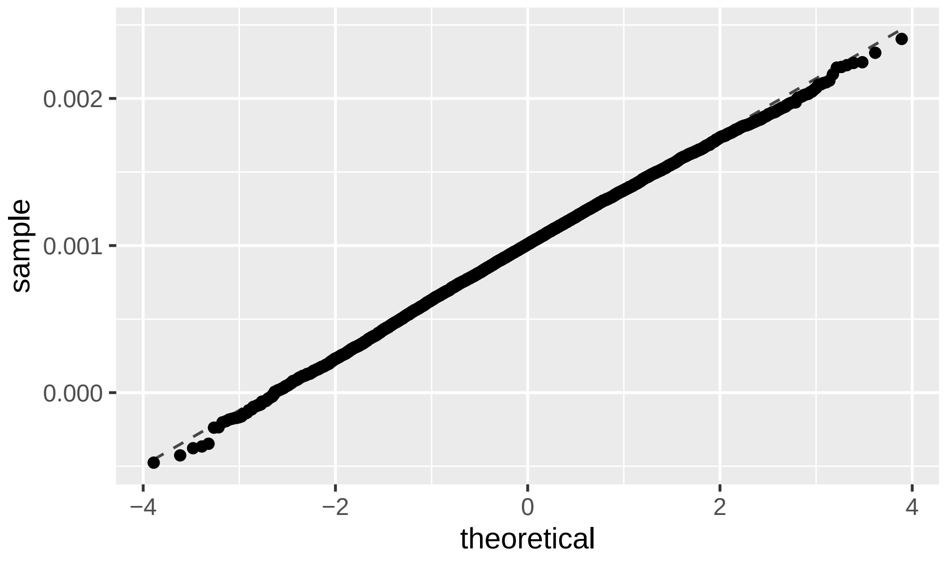
Method 1: For Normal-ish Distributions

THIS IS NUMBERED as Method 2 in the Stat2 book.

Confirm that the bootstrap distribution is approximately normal (see the qq plot to confirm). If yes, use the normal z distribution cutoffs (+/- 1.96) to identify the middle 95%. Then build a margin of error around the actual estimate using the bootstrap distribution to obtain the standard error (SE). Remember margin of error is the estimate +/- cutoff*SE

For a 95% bootstrap confidence interval for the coefficient for `SATV`, β_{SATV}:

```
gf_qq(~ bootstrap$SATV) %>%
  gf_qqline
```



```
coef.actual+c(-1, 1)*qnorm(0.975)*sd(bootstrap$SATV)
```

```
## [1] 0.000278 0.001747
```

Method 2: For Symmetric Distributions

THIS IS NUMBERED as Method 1 in the book.

If the distribution is roughly symmetric, but not quite normal the z scores might not apply. Since the distribution is symmetric, we could just use the actual 2.5th percentile and 97.5th percentile of the **bootstrap distribution**. We choose these cutoffs (not based on the normal distribution but rather on the bootstrap distribution) to locate the middle 95% of observations without relying on the normal z distribution. Remember the observations in the bootstrap distribution are the slope coefficients from 10,000 samples of our original sample (as if it were a psuedo-population).

```
qdata(bootstrap$SATV, c(0.025,0.975))
```

```
##     2.5%    97.5%
## 0.000243 0.001719
```

Method 3: For Skewed Distributions

For skewed bootstrap distributions, things are more complicated. If a bootstrap distribution is skewed, we can't not use normal distribution cutoffs like in Method #1. We also cannot use the standard deviation of a "skewed" distribution because it wouldn't be a good measure for spread.

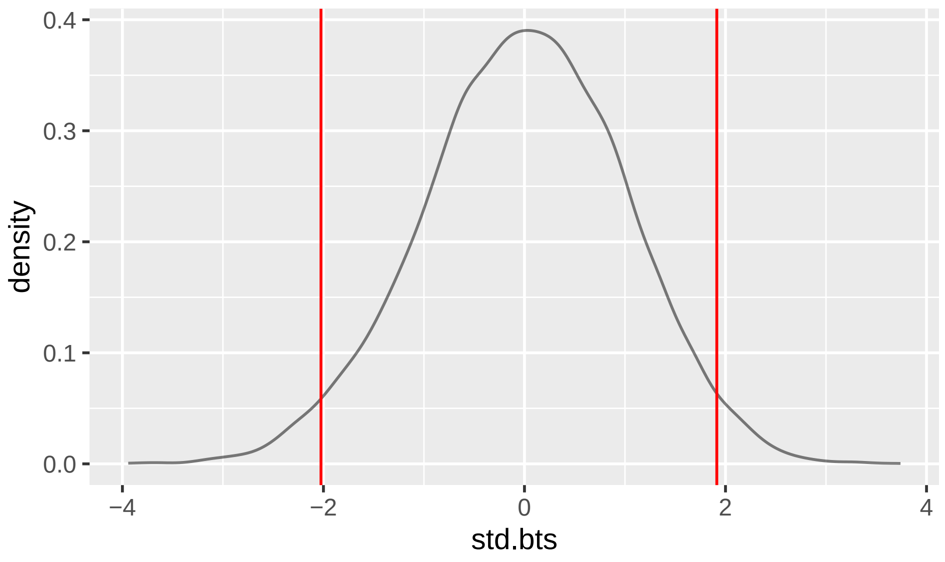
A common method is to use a combination of methods 1 and 2. We'll use the SE value from the *original* regression as an estimate of the standard deviation of the statistic of interest (here, it's $\hat{\beta}_{SATV}$), and use the cutoff values (quantiles) from a **standardized** (rescaled) bootstrap distribution (i.e centering at the sample estimate and dividing by the standard deviation of the bootstrap distribution).

Here, we'll need to somehow **reverse the standardized quantiles**. Why? If a bootstrap distribution is skewed, say **skewed to the left**, the standardized bootstrap distribution would be still skewed to the left, indicating those bootstrap estimates might *tend to go low*, lower than the true parameter value (i.e. the estimate $\hat{\beta}_{SATV}$ < the true β_{SATV} is an under-estimate). So when we construct the interval, we should intentionally **stretch more on the right side** (as that's the direction of where the true parameter might be).

A left-skewed standardized bootstrap distribution would make it (positive) upper quantile smaller than its (negative) lower quantile, in absolute value, i.e. $|qt_{0.975}| < |qt_{0.025}|$, so if we would like to stretch our interval more on the right side, we would need to use the larger $-qt_{0,025} \cdot SE$ as the upper margin of error (`minus` sign, because $qt_{0,025}$ is negative), and the smaller $qt_{0,025} \cdot SE$ as our lower margin of error. Thus, a 95% bootstrap confidence interval for a skewed bootstrap distribution would be:

$$(Estimate - qt_{0.975} \cdot SE, \quad Estimate - qt_{0.025} \cdot SE)$$

```
## standardized/rescaled the bootstrap distribution
std.bts <- (bootstrap$SATV - mean(bootstrap$SATV))/sd(bootstrap$SATV)
## find quantiles from the standardized bootstrap distribution
qtU <- qdata(std.bts, p = 0.975)
qtL <- qdata(std.bts, p = 0.025)
gf_dens(~ std.bts) %>%
  gf_vline(xintercept = c(qtL, qtU), color = "red")
```

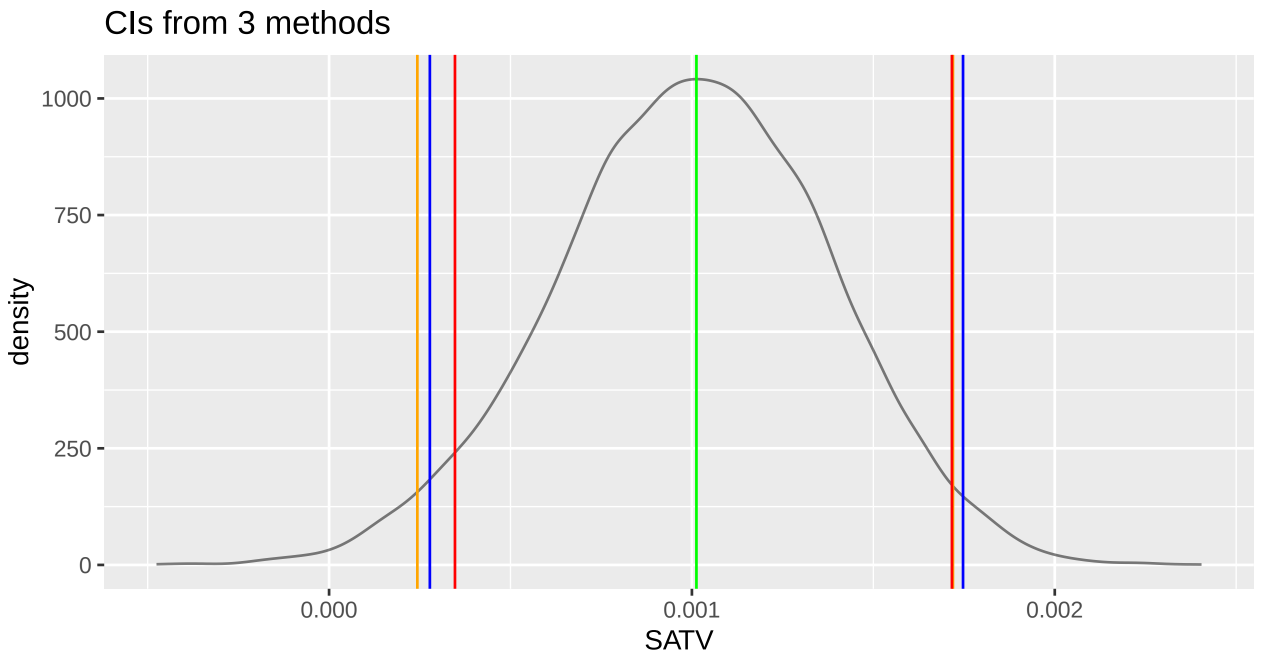


```
SE <- msummary(m2)$coefficients["SATV","Std. Error"]
## bootstrap CI for skewed distribution
c(coef.actual - qtU*SE, coef.actual - qtL*SE)
```

```
##     SATV     SATV
## 0.000347 0.001717
```

Plot 3 bootstrap intervals from above three methods:

```
gf_dens(~ SATV, data = bootstrap, title="CIs from 3 methods") %>%
  gf_vline(xintercept = coef.actual, color = "green") %>%
  gf_vline(xintercept = (coef.actual+c(-1, 1)*qnorm(0.975)*sd(bootstrap$SATV)),
           color = "blue") %>%
  gf_vline(xintercept = qdata(bootstrap$SATV, c(0.025,0.975)),
           color = "orange") %>%
  gf_vline(xintercept = c(coef.actual - qtU*SE, coef.actual - qtL*SE),
           color = "red")
```


CIs from 3 methods
density
1000
750
500
250
0
0.000
0.001
0.002
SATV