HW boostrap Solutions

21.3 Fixed vs Random bootstrap

part (a): with Outlier

Mission: estimate slope of simple reg

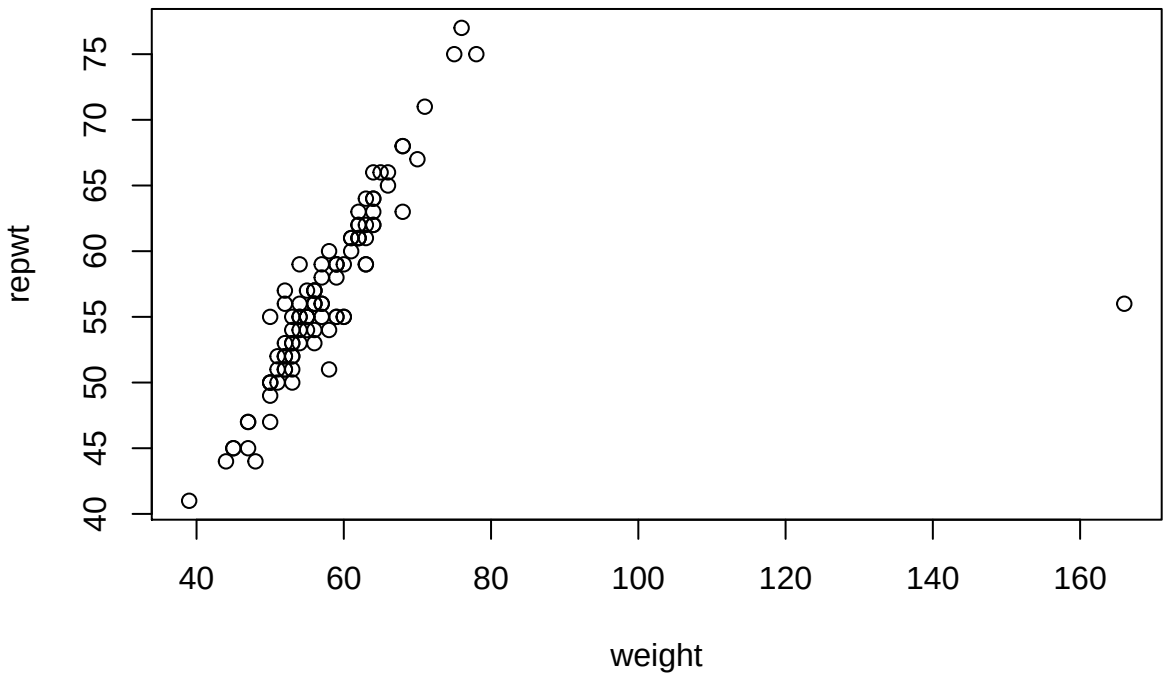
- Drawing r = 1000 bootstrap samples Using

1. random-X resampling
2. fixed-X resampling

In each case,

1. plot a histogram of 1000 bootstrap slopes
2. calculate the bootstrap estimate of **Standard Error** for the slope
3. explore how the influential outlier affect random resampling and fixed resampling

```
# read data from url
wdata = read.table("https://socialsciences.mcmaster.ca/jfox/Books/RegressionDiagnostics/Davis.txt")

wdata = subset(wdata, gender == "female" & !is.na(repwt) )

wFit = lm(repwt ~ weight, data = wdata)
plot(repwt ~ weight, data = wdata)
```

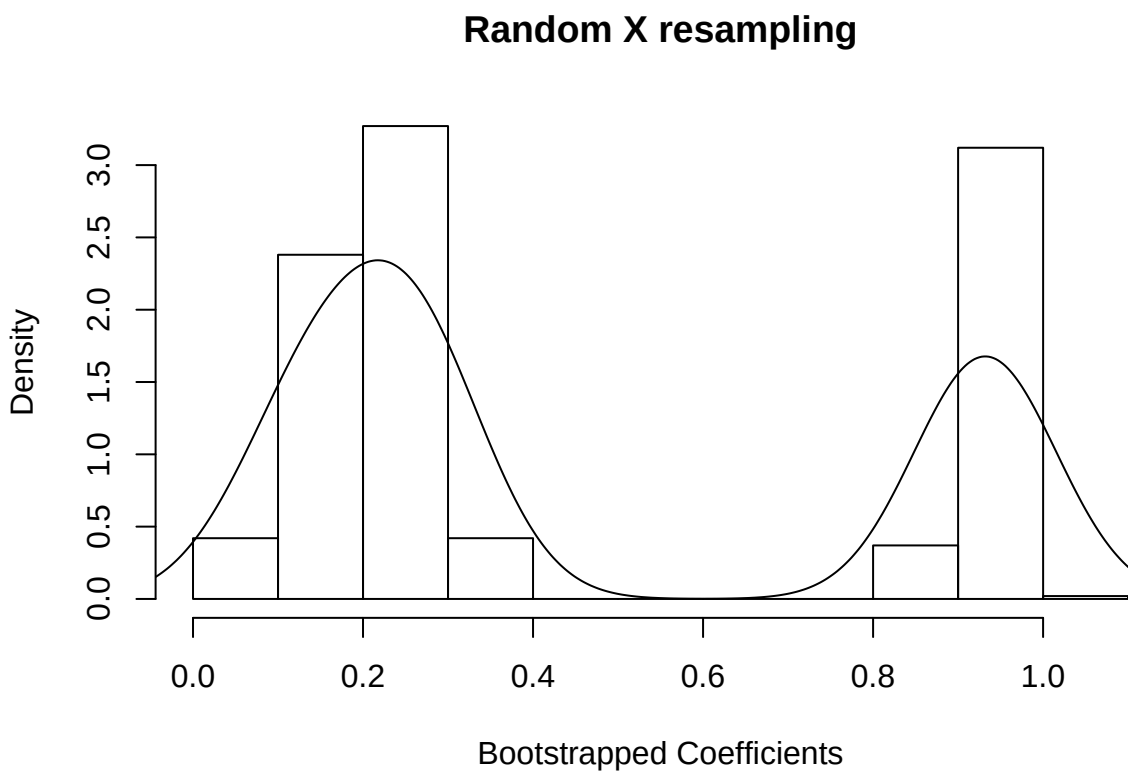


Choose the indices of the bootstrap samples

Important always sample from 1:100 indices, with replacement. (Never touch other observations). - The population vs the 101 sample - The 101 sample vs 100 bootstrap samples

```
set.seed(12345)
bootInd = replicate(1000, sample(101,100,replace=T)) # replicate sample func 1000 times
# to generate 1000 times, each time 100 bootstrap indices
```

```
# input(index), output(bootstarp slope(boostrap estimation) )
BootFit<-function(indices) {
  bootY = wdata$repwt[indices]
  bootX = wdata$weight[indices]
  bootFit = lm(bootY ~ bootX)
  bootFit$coefficients[2]
}

# PASS each column(100 index) of bootInd(matrix 100*1000), IN BootFit function as operand.
# store 1000 results
randomBootCoeffs = apply(bootInd, 2, BootFit)

# result of random bootstrap
hist(randomBootCoeffs,   freq = FALSE,
     main = "Random X resampling", xlab = "Bootstrapped Coefficients")
lines(density(randomBootCoeffs))
```



Random X resampling

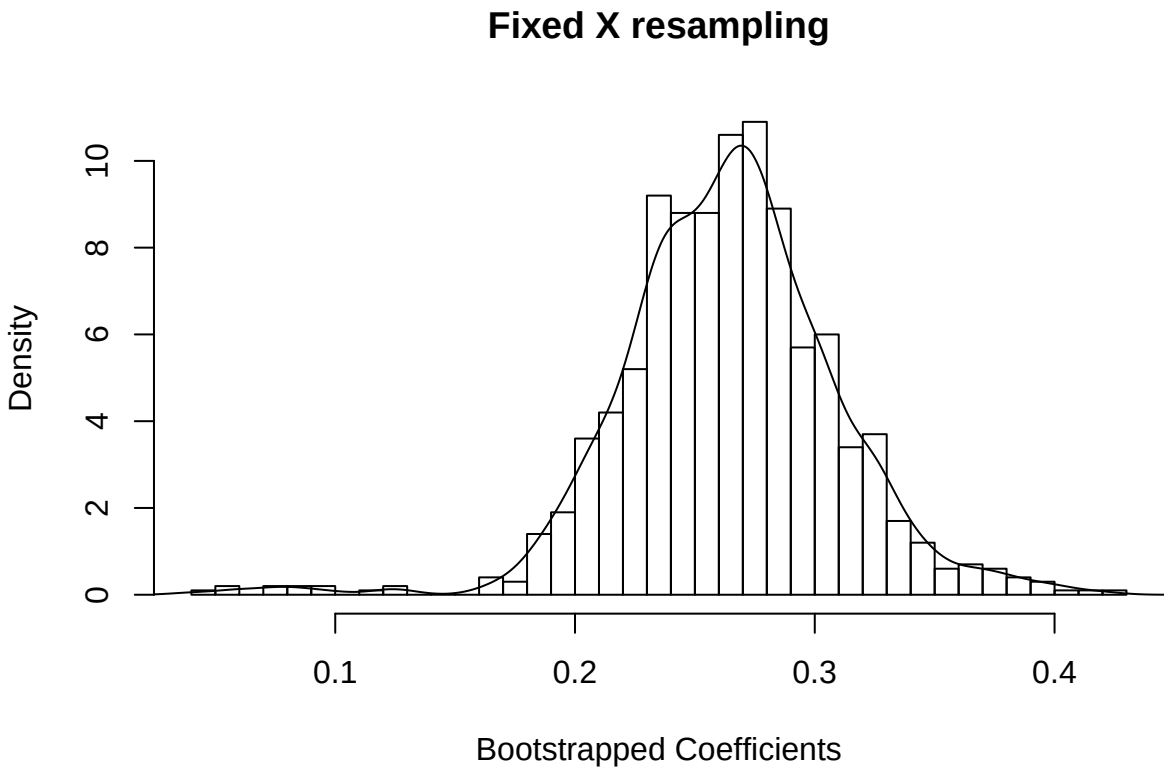
```
# bootstrap estimate of slope
mean(randomBootCoeffs)
```

```
## [1] 0.4613896
```

```
sd(randomBootCoeffs)
```

```
## [1] 0.3510458
```

```
wFittedVals = fitted(wFit) # y hat of lm object
wResidualVals = residuals(wFit)


FixBootFit<-function(indices) {
  bootY = wFittedVals + wResidualVals[indices]
  bootFit = lm(bootY ~ wdata$weight)
  bootFit$coefficients[2]
  }

fixedBootCoeffs = apply(bootInd, 2, FixBootFit)

hist(fixedBootCoeffs, breaks = 30, freq = FALSE,
     main = "Fixed X resampling", xlab = "Bootstrapped Coefficients")
lines(density(fixedBootCoeffs))
```



Fixed X resampling

```
mean(fixedBootCoeffs)
```

```
## [1] 0.2643282
```

```
sd(fixedBootCoeffs)
```

```
## [1] 0.04503113
```

- Estimate Random vs Fixed

The random X bootstrap coefficients have a bimodal distribution that reflects two types of samples that arise in resampling: those that contain or do not contain the influential observation. We see the tremendous difference in the estimated coefficient when the observation is included/excluded from the sample. The second mode is smaller, reflecting the chance that the influential observation is excluded from the bootstrap sample.

The fixed X estimate for the coefficient is close to the original estimate, because the x-value for the influential observation is always included in the sample, and its y-value is the original fitted value (i.e., from the data with the influential observation) plus a randomly chosen residual.
This means that impact of **the influential observation is always present in the bootstrap sample** (although the size of it may vary a bit depending on the size of the residual that is added to the fitted value).

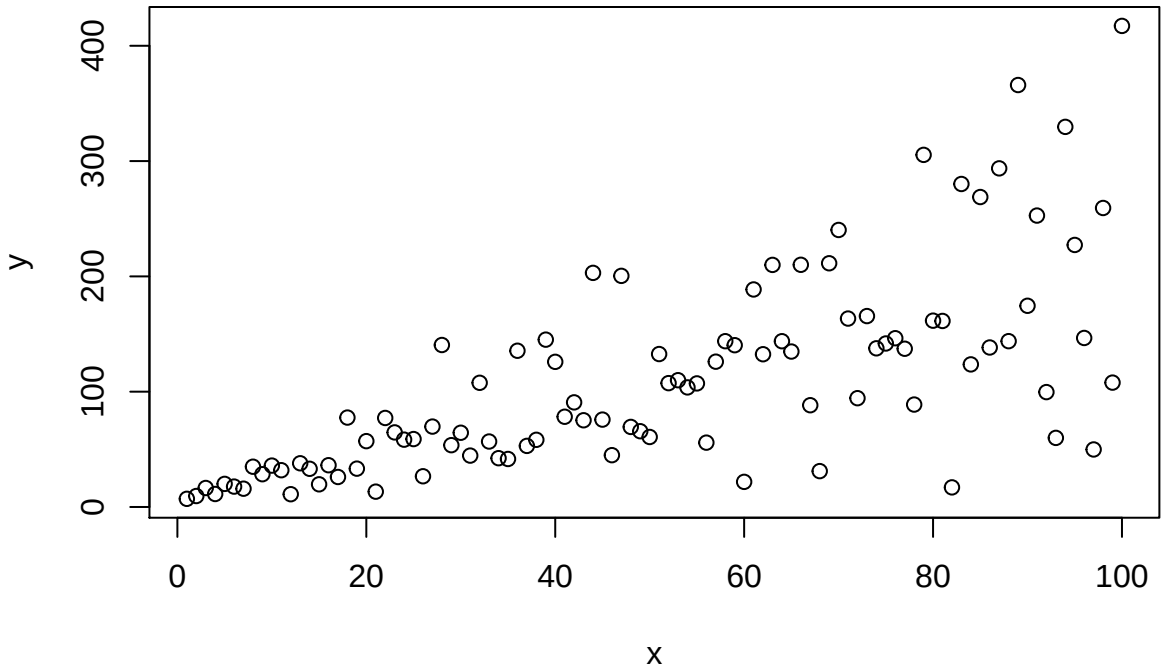
- Standard Error; Random vs Fixed The SD of the bootstrap coefficients for the random X resampling procedure is very large because the bootstrapped coefficient varies a lot depending on whether or not the influential observation is included in the bootstrap sample.

The SD of the bootstrap coefficients for the fixed X resampling procedure is close to the least squares estimate of the SE of the coefficient. This is because the high leverage x-value is always in the bootstrap sample and the accompanying y-value is close to the actual value.

part (b): with heteroscedasticity

We generate our sample as follows.

```
set.seed(110314)
x = 1:100
errs = rnorm(100, mean = 0, sd = x)
y = 5 + 2*x + errs

plot(y ~ x)
```



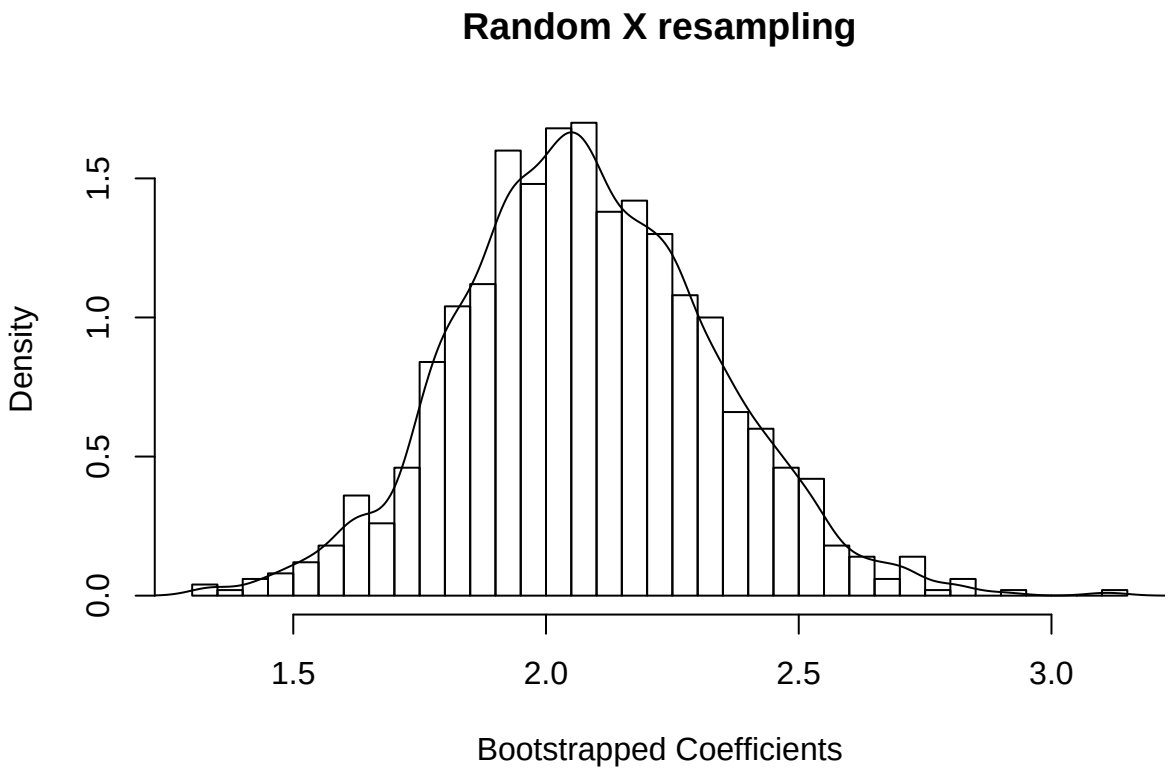
Notice that there is a linear association between y and x with slope 2. Also notice that the errors are heteroscedastic.

We can choose the bootstrap samples by selecting the same indices for both approaches as follows:

```
bootInd = replicate(1000, sample(100, 100, replace = TRUE))
```

Random X bootstrap For the random X type of bootstrap, for each bootstrap sample, we fit the model with

```
randomBootCoeffs = apply(bootInd, 2, function(indices) {
  bootY = y[indices]
  bootX = x[indices]
  bootFit = lm(bootY ~ bootX)
  bootFit$coefficients[2]
})

hist(randomBootCoeffs, breaks = 30, freq = FALSE,
     main = "Random X resampling", xlab = "Bootstrapped Coefficients")
lines(density(randomBootCoeffs, bw = .04))
```




```
mean(randomBootCoeffs)
```

```
## [1] 2.080766
```

```
sd(randomBootCoeffs)
```

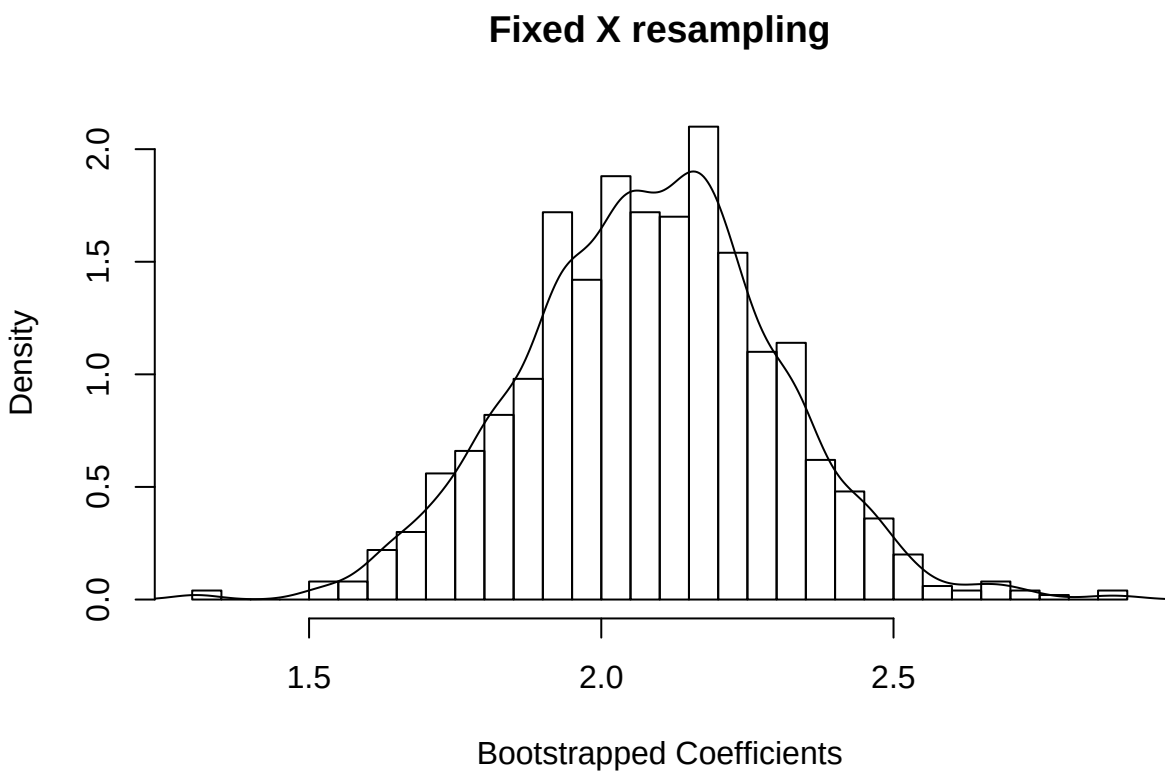
```
## [1] 0.251173
```

Fixed X bootstrap For the fixed X bootstap, we first fit the model to the original data and get the fitted values and the residuals with

```
origFit = lm(y ~ x)
origFittedVals = fitted(origFit)
origResidualVals = residuals(origFit)
```

Now we bootstrap the residuals from the fit and use these to create bootstrap Y values as follows.

```
fixedBootCoeffs = apply(bootInd, 2, function(indices) {
  bootY = origFittedVals + origResidualVals[indices]
  bootFit = lm(bootY ~ x)
  bootFit$coefficients[2]
})

hist(fixedBootCoeffs, breaks = 30, freq = FALSE,
     main = "Fixed X resampling", xlab = "Bootstrapped Coefficients")
lines(density(fixedBootCoeffs, bw = .04))
```




```
mean(fixedBootCoeffs)
```

```
## [1] 2.083938
```

```
sd(fixedBootCoeffs)
```

```
## [1] 0.2149278
```

Compare the results Both methods have the same average bootstrapped coefficient for the slope, i.e., 2.07. The random X version has a larger coefficient standard error, 0.25 compared to the fixed X version, 0.22. The fixed X version has a standard error that is similar to the least squares estimate of the SE.

This may seem a bit surprising, but recall that the standard error of the coefficients consists of a term involving the residual SD and a term involving the design. In the random X case, large values in the design are correlated wth large variation in the errors where as in the random design these are uncorrelated.

```
invisible(
  apply(bootInd[, 1:3], 2, function(indices) {
```

```
    bootY = y[indices]
    bootX = x[indices]
    bootFit1 = lm(bootY ~ bootX)
    bootY = origFittedVals + origResidualVals[indices]
    bootFit2 = lm(bootY ~ x)
    par(mfrow = c(1,2), mar = c(2,2,3,1))
    #plot(residuals(bootFit1) ~ fitted(bootFit1), main = "Random X" )
    #plot(residuals(bootFit2) ~ fitted(bootFit2), main = "Fixed X" )
    par(mfrow = c(1,1))
})
)
```

Plot the residuals for a few of the bootstraps: The residual plots for the random X indicate the problem with nonconstant variance. The residual plots for the fixed X show a higher variation for the residuals, but the variation is constant across the fitted values.

part (c): why random resampling is better

The random X approach is preferred because it can **uncover departures from the assumed model**(i.e. expose problems), e.g., structure in the residuals, influential observations, and non-constant variance.

21.4: Bootstrap estimates of bias

The bootstrap can be used to estimate ** the bias of an estimator $\hat{\sigma}$ of a parameter σ **, simply by comparing the mean of bootstrap distribution $\overline{\sigma}*$ with the sample estimate $\hat{\sigma}$. That is:

$\hat{bias} = \overline{\sigma}_* - \hat{\sigma}$

Mission Estimate the bias of the MLE of variance, that is :

$\hat{\sigma}^2 = \frac{\sum(Yi-\overline{Y})^2}{n}$

Notice that the unbiased estimate is:

$\hat{\sigma}^2 = \frac{\sum(Yi-\overline{Y})^2}{n-1}$

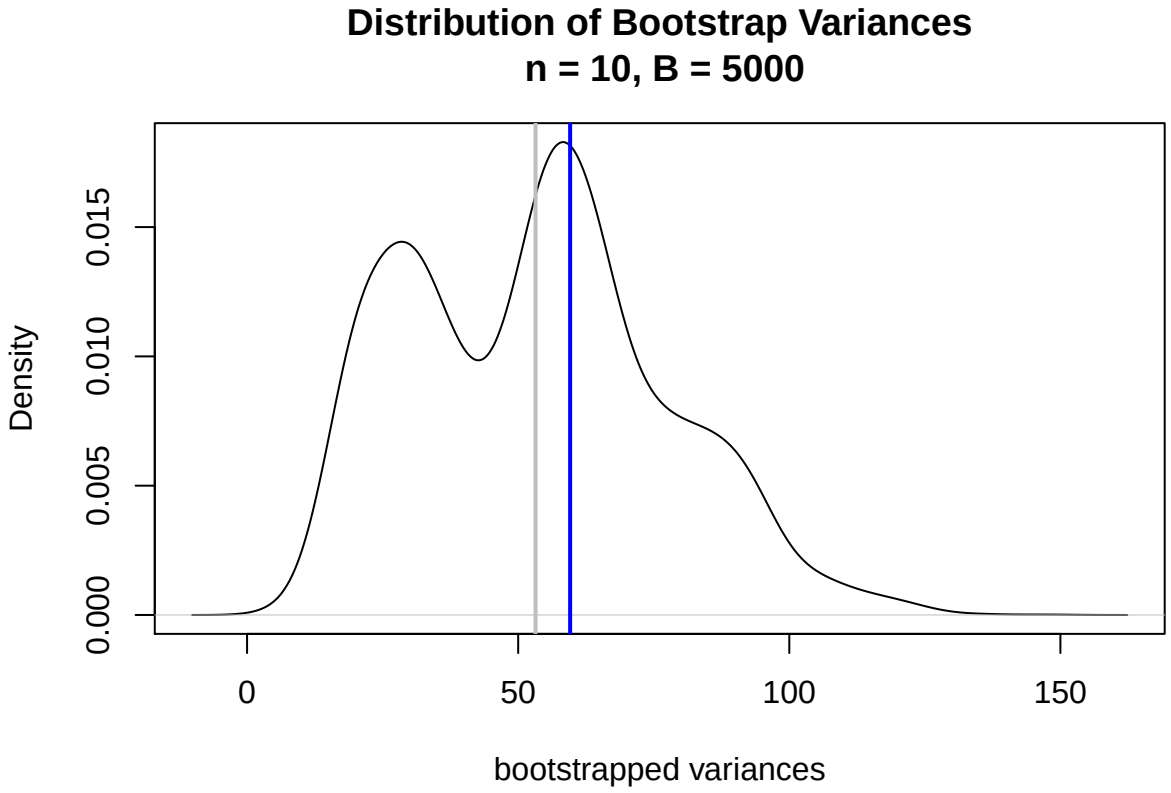
So the bias should theoretically be:

$-\frac{\sigma^2}{n}$

For a sample of n = 10, drawn from **N(0,100)** (True Distribution); Use r = 5000 as bootstrap replications.

```
set.seed(12345)
# generate 10 samples from N(0,100)
samp10 = rnorm(10, mean = 0, sd = 10)

#### sample estimate of MLE of variance(biased)
sigmaHat2 = sum( (samp10 - mean(samp10))^2 ) / 10

#### boot estimate
# single bootstrap estimate of variance
bootVar = function(samp) {
  bootSamp = sample(samp, 10, replace = TRUE)
  sum( (bootSamp - mean(bootSamp))^2 ) / 10
}
```

```r
bootVars = replicate(5000, bootVar(samp10))
# Bootstrap estimate:
boot_es <- mean(bootVars)


# Bootstrap estimate of bias
biasVar = boot_es - sigmaHat2



#
plot(density(bootVars, bw = 5),
     main = "Distribution of Bootstrap Variances\n n = 10, B = 5000",
     xlab = "bootstrapped variances")
abline(v = mean(bootVars), col = "grey", lwd = 2)
abline(v = sigmaHat2, col = "blue", lwd = 2)
```




```r
biasVar
```

```
## [1] -6.381189
```

The estimated bias is about -7, for 500 bootstrap replicates. When I increased the number of replicates to 5000, the estimated bias was about -8.2. (Note that we are bootstrapping a very small sample.)