STAT 341: Assignment 4

DUE: Friday April 3 by 11:59pm EST

QUESTION 1: Bootstrap Confidence Intervals [33 points]

The Toronto Raptors are a Canadian professional basketball team and the reigning NBA champions. In this question you will analyze data associated with their 2018-2019 championship season. The `raptors.csv` file contains information on the $N = 82$ regular season games. In particular, for each game we have the recorded the variates listed in the table below. For more information on these variates and how they are calculated, you may find this Wikipedia article helpful.

Variate	Value
`DATE`	The date of the game, recorded `"MM/DD/YYYY"`
`MATCHUP`	The opponent and location of the game
`W/L`	Whether the Raptors won (`W`) or lost (`W`) the gameThe runner's bib number
`MIN`	The number of minutes played
`PTS`	The number of points scored by the raptors
`FGM`	The number of field goals made
`FGA`	The number of field goals attempted
`FG%`	The percentage of field goals made
`3PM`	The number of three-point field goals made
`3PA`	The number of three-point field goals attempted
`3P%`	The percentage of three-point field goals made
`FTM`	The number of free throws goals made
`FTA`	The number of free throws attempted
`FT%`	The percentage of free throws made
`OREB`	The number of offensive rebounds made
`DREB`	The number of defensive rebounds made
`REB`	The number of rebounds made (the sum of `OREB` and `DREB`)
`AST`	The number of assists
`STL`	The number of steals
`BLK`	The number of blocks
`TOV`	The number of turnovers
`PF`	The number of personal fouls

Particular attention will be paid to the `PTS` variate (y_u). In particular, throughout this question you will summarize the `PTS` variate with a variety of attributes and you will calculate bootstrap-based confidence intervals for each of them.

(a) [1 point] Load the data and save it in a variable called `rap` and, using the indices contained in the `sampIndex.txt` file, take a sample $\mathcal{S}$ of size $n = 20$ from the population. Print out the `PTS` values for this sample.

SOLUTION:

```r
setwd("/Users/nstevens/Dropbox/Teaching/STAT_341/Assignments/Assignment4/")
rap <- read.csv("raptors.csv", header = TRUE)
sampIndex <- read.table("sampIndex.txt")$V1
S <- rap[sampIndex, "PTS"]
print(S)
```

```
##  [1] 111 129 131 109 125 104 120 105 106 121  87 123 108 111  92 112  95
## [18] 119  99 120
```

(b) [1 point] By resampling S with replacement, construct $B = 1000$ bootstrap samples $S_1^\star, S_2^\star, \ldots, S_{1000}^\star$.

SOLUTION:

```
B <- 1000
n <- 20
Sstar <- sapply(1:B, FUN = function(b) {
    sample(S, n, replace = TRUE)
})
```

(c) [7 points] This question concerns the average points scored per game

$$a(\mathcal{P}) = \overline{y} = \frac{1}{N}\sum_{u\in\mathcal{P}} y_u$$

i. [2 points] Calculate $a(\mathcal{S})$ and $a(\mathcal{P})$

SOLUTION:

```r
# Sample Mean:
mean(S)
```

```
## [1] 111.35
```

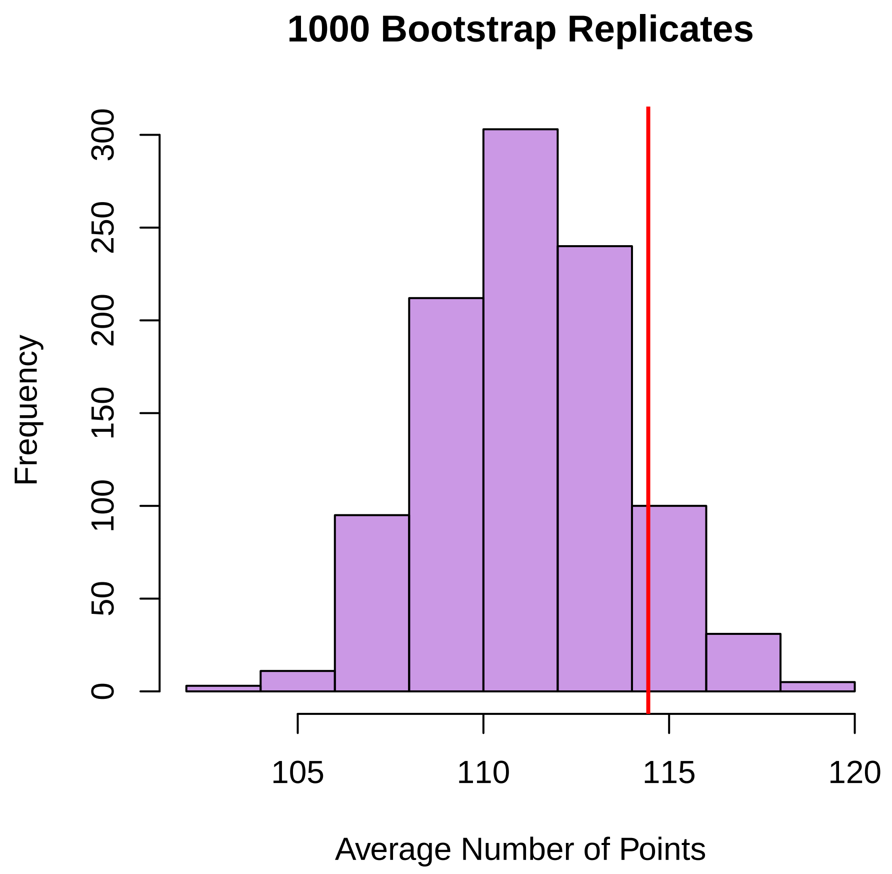
```r
# Population Mean:
mean(rap$PTS)
```

```
## [1] 114.439
```

ii. [2 points] Calculate $a(\mathcal{S}_b^\star)$ for each bootstrap sample $b = 1, 2, \ldots, B$ from part (b) and construct a histogram of these values. Be sure to include a vertical line representing $a(\mathcal{P})$, and also be sure to informatively label your plot.

SOLUTION:

```r
avg_star <- apply(X = Sstar, MARGIN = 2, FUN = mean)
hist(avg_star, col = adjustcolor("darkorchid", 0.5), xlab = "Average Number of Points",
    main = "1000 Bootstrap Replicates")
abline(v = mean(rap$PTS), col = "red", lwd = 2)
```



iii. [1 point] Calculate a 95% confidence interval for $a(\mathcal{P})$ using the naive normal theory approach.

SOLUTION:

```
mean(S) + qnorm(0.975) * c(-1, 1) * sd(avg_star)
```

```
## [1] 106.3475 116.3525
```

iv. [1 point] Calculate a 95% confidence interval for $a(\mathcal{P})$ using the percentile method.

SOLUTION:

```
c(quantile(avg_star, 0.025), quantile(avg_star, 0.975))
```

```
##     2.5%    97.5%
## 106.6000 116.4013
```

v. [1 point] Calculate a 95% confidence interval for $a(\mathcal{P})$ using the bootstrap-t approach. **Note** that you may find it helpful to use the `bootstrap_t_interval` function appended at the end of the assignment. Please use $B = 1000$ and $D = 100$.

SOLUTION:

```
bootstrap_t_interval(S = S, a = mean, confidence = 0.95, B = 1000, D = 100)
```

```
##    lower   middle    upper
## 105.8467 111.3500 116.5189
```

(d) [7 points] This question concerns the median points scored per game

$$a(\mathcal{P}) = \underset{u \in \mathcal{P}}{\text{median}}\, y_u$$

i. [2 points] Calculate $a(\mathcal{S})$ and $a(\mathcal{P})$

SOLUTION:

```r
# Sample Median:
median(S)
```

```
## [1] 111
```

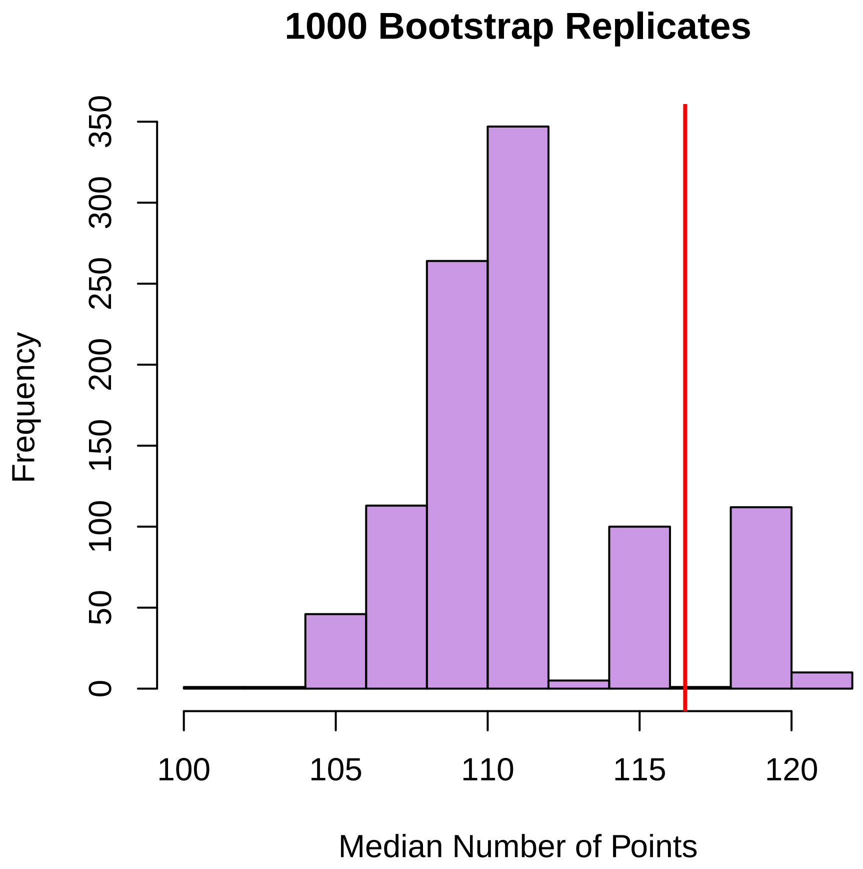
```r
# Population Median:
median(rap$PTS)
```

```
## [1] 116.5
```

ii. [2 points] Calculate $a(\mathcal{S}_b^\star)$ for each bootstrap sample $b = 1, 2, \ldots, B$ from part (b) and construct a histogram of these values. Be sure to include a vertical line representing $a(\mathcal{P})$, and also be sure to informatively label your plot.

SOLUTION:

```r
med_star <- apply(X = Sstar, MARGIN = 2, FUN = median)
hist(med_star, col = adjustcolor("darkorchid", 0.5), xlab = "Median Number of Points",
    main = "1000 Bootstrap Replicates")
abline(v = median(rap$PTS), col = "red", lwd = 2)
```



iii. [1 points] Calculate a 95% confidence interval for $a(\mathcal{P})$ using the naive normal theory approach.

SOLUTION:

```
median(S) + qnorm(0.975) * c(-1, 1) * sd(med_star)
```

```
## [1] 103.4735 118.5265
```

iv. [1 points] Calculate a 95% confidence interval for $a(\mathcal{P})$ using the percentile method.

SOLUTION:

```
c(quantile(med_star, 0.025), quantile(med_star, 0.975))
```

```
##  2.5% 97.5%
## 105.5 120.0
```

v. [1 points] Calculate a 95% confidence interval for $a(\mathcal{P})$ using the bootstrap-t approach. **Note** that you may find it helpful to use the `bootstrap_t_interval` function appended at the end of the assignment. Please use $B = 1000$ and $D = 100$.

SOLUTION:

```
bootstrap_t_interval(S = S, a = median, confidence = 0.95, B = 1000, D = 100)
```

```
##    lower   middle    upper
## 100.0390 111.0000 118.7562
```

(e) [7 points] This question concerns the standard deviation of the number points scored per game

$$a(\mathcal{P}) = SD_{\mathcal{P}}(y) = \sqrt{\frac{\sum_{u \in \mathcal{P}} (y_u - \overline{y})^2}{N}}$$

i. [2 points] Calculate $a(\mathcal{S})$ and $a(\mathcal{P})$

SOLUTION:

Here it will be useful to define a function that calculates the standard deviation with N as the divisor.

```r
sdn <- function(S) {
    N <- length(S)
    sqrt((N - 1) * var(S)/N)
}
```

```r
# Sample Standard Deviation:
sdn(S)
```

```
## [1] 11.9343
```

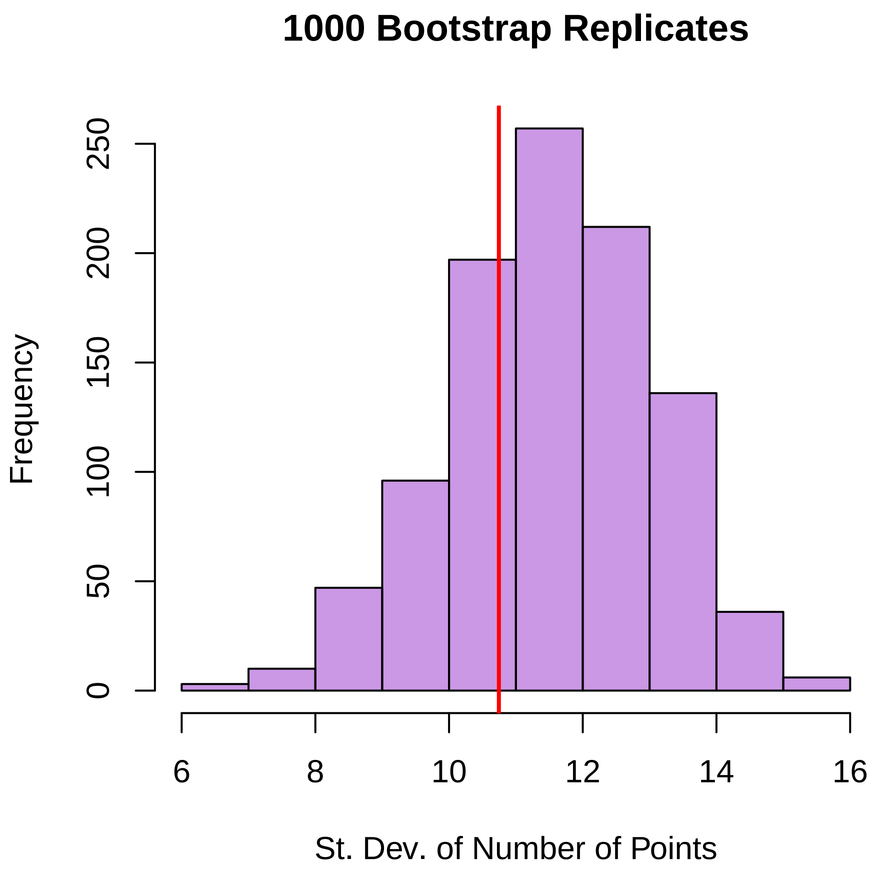
```r
# Population Standard Deviation:
sdn(rap$PTS)
```

```
## [1] 10.7455
```

ii. [2 points] Calculate $a(\mathcal{S}_b^\star)$ for each bootstrap sample $b = 1, 2, \ldots, B$ from part (b) and construct a histogram of these values. Be sure to include a vertical line representing $a(\mathcal{P})$, and also be sure to informatively label your plot.

SOLUTION:

```r
sdn_star <- apply(X = Sstar, MARGIN = 2, FUN = sdn)
hist(sdn_star, col = adjustcolor("darkorchid", 0.5), xlab = "St. Dev. of Number of Points",
    main = "1000 Bootstrap Replicates")
abline(v = sdn(rap$PTS), col = "red", lwd = 2)
```



iii. [1 points] Calculate a 95% confidence interval for $a(\mathcal{P})$ using the naive normal theory approach.

SOLUTION:

```
sdn(S) + qnorm(0.975) * c(-1, 1) * sd(sdn_star)
```

```
## [1]  8.952684 14.915915
```

iv. [1 points] Calculate a 95% confidence interval for $a(\mathcal{P})$ using the percentile method.

SOLUTION:

```
c(quantile(sdn_star, 0.025), quantile(sdn_star, 0.975))
```

```
##      2.5%     97.5%
##  8.372839 14.300046
```

v. [1 points] Calculate a 95% confidence interval for $a(\mathcal{P})$ using the bootstrap-t approach. **Note** that you may find it helpful to use the `bootstrap_t_interval` function appended at the end of the assignment. Please use $B = 1000$ and $D = 100$.

SOLUTION:

```
bootstrap_t_interval(S = S, a = sdn, confidence = 0.95, B = 1000, D = 100)
```

```
##     lower    middle     upper
##  9.179127 11.934299 16.841844
```

(f) [7 points] This question concerns the skewness of the number points scored per game

$$a(\mathcal{P}) = 3 \times \frac{(\overline{y} - \text{median}_{u \in \mathcal{P}}\, y_u)}{SD_{\mathcal{P}}(y)}$$

i. [2 points] Calculate $a(\mathcal{S})$ and $a(\mathcal{P})$

SOLUTION:

Here it will be useful to define a function that calculates the standard deviation with N as the divisor.

```r
skew <- function(S) {
    3 * (mean(S) - median(S))/sdn(S)
}
```

```r
# Sample Skewness:
skew(S)
```

```
## [1] 0.0879817
```

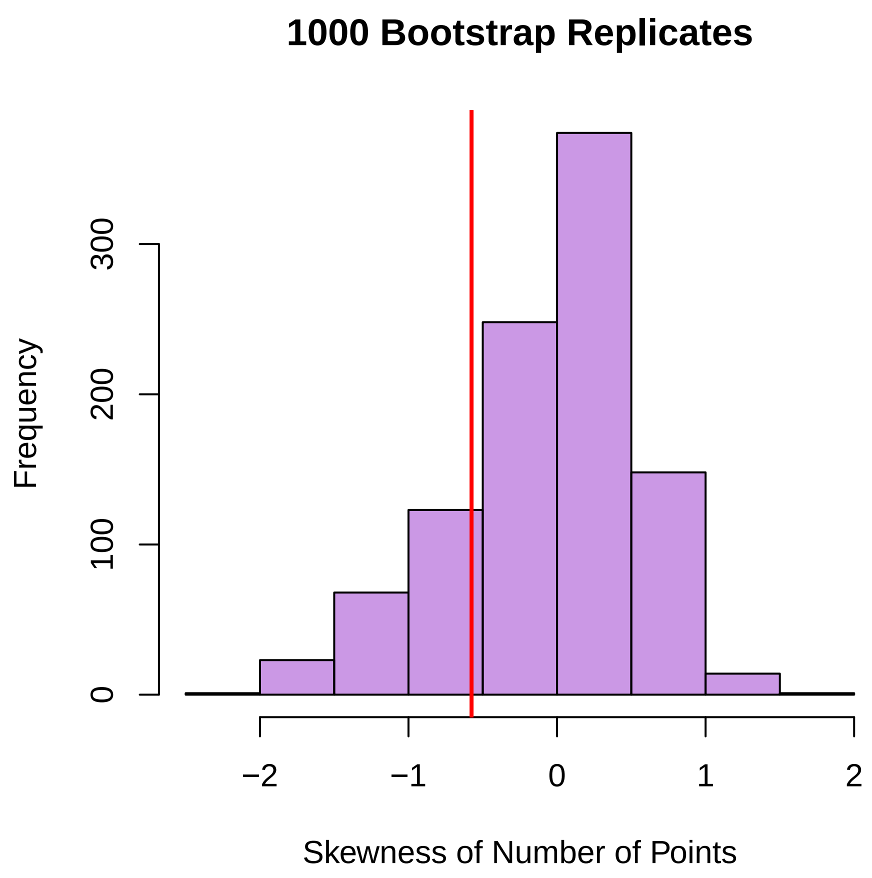
```r
# Population Skewness:
skew(rap$PTS)
```

```
## [1] -0.5753968
```

ii. [2 points] Calculate $a(\mathcal{S}_b^\star)$ for each bootstrap sample $b = 1, 2, \ldots, B$ from part (b) and construct a histogram of these values. Be sure to include a vertical line representing $a(\mathcal{P})$, and also be sure to informatively label your plot.

SOLUTION:

```r
skew_star <- apply(X = Sstar, MARGIN = 2, FUN = skew)
hist(skew_star, col = adjustcolor("darkorchid", 0.5), xlab = "Skewness of Number of Points",
    main = "1000 Bootstrap Replicates")
abline(v = skew(rap$PTS), col = "red", lwd = 2)
```



iii. [1 points] Calculate a 95% confidence interval for $a(\mathcal{P})$ using the naive normal theory approach.

SOLUTION:

```
skew(S) + qnorm(0.975) * c(-1, 1) * sd(skew_star)
```

```
## [1] -1.109644  1.285607
```

iv. [1 points] Calculate a 95% confidence interval for $a(\mathcal{P})$ using the percentile method.

SOLUTION:

```
c(quantile(skew_star, 0.025), quantile(skew_star, 0.975))
```

```
##       2.5%      97.5%
## -1.4872115  0.9085112
```

v. [1 points] Calculate a 95% confidence interval for $a(\mathcal{P})$ using the bootstrap-t approach. **Note** that you may find it helpful to use the `bootstrap_t_interval` function appended at the end of the assignment. Please use $B = 1000$ and $D = 100$.

SOLUTION:

```
bootstrap_t_interval(S = S, a = skew, confidence = 0.95, B = 1000, D = 100)
```

```
##      lower     middle      upper
## -0.6533252  0.0879817  1.4405163
```

(g) [3 points] This question concerns advantages and disadvantages associated with the various methods of confidence interval calculation you've explored.

i. [1 point] List one advantage and one disadvantage of naive normal theory intervals.

SOLUTION:

- Advantage: Simple calculation.
- Disadvantage: Not widely applicable.

ii. [1 point] List one advantage and one disadvantage of percentile method intervals.

SOLUTION:

- Advantage: Simple calculation and widely applicable (don't need both).
- Disadvantage: May not be accurate if the attributes sampling distribution is very asymmetric.

iii. [1 point] List one advantage and one disadvantage of bootstrap-*t* intervals.

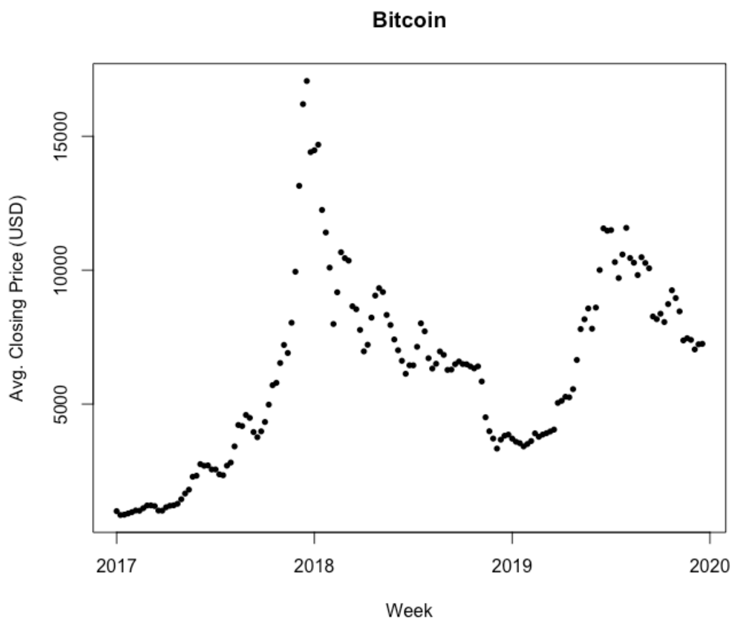
SOLUTION:

- Advantage: Tends to have good coverage when the attribute of interest is an average.
- Disadvantage: Not widely applicable, and computationally intensive if no closed form expression exists for the standard deviation of the estimator (don't need both).

QUESTION 2: Evaluating Prediction Accuracy [22 points]

Bitcoin (ticker symbol: BTC) is a cryptocurrency – a means to engage in decentralized digital transactions. In recent years, much attention has been paid to cryptocurrencies in general and bitcoin in particular. As an investment, bitcoin is highly volatile – going through periods of enormous highs and devastating lows. Using polynomials, you will attempt to model the average weekly closing prices of BTC during the period 2017-2019 inclusive. The data available in `bitcoin.csv` is explained in the following table and visualized in the figure below.

Variate	Value
`Week`	A numeric value indicating week number, counting up from the first week of 2017
`Avg_Closing_Price`	The average BTC closing price (in US dollars) for a given week



DISCLAIMER: If you are interested in modeling time series data you should, in general, not rely on polynomials. Many more sophisticated and useful time series models exist for this purpose. If you have interest in this I recommend taking STAT 443: Forecasting.

(a) [1 point] Load the data and save it in a variable called `btc`.

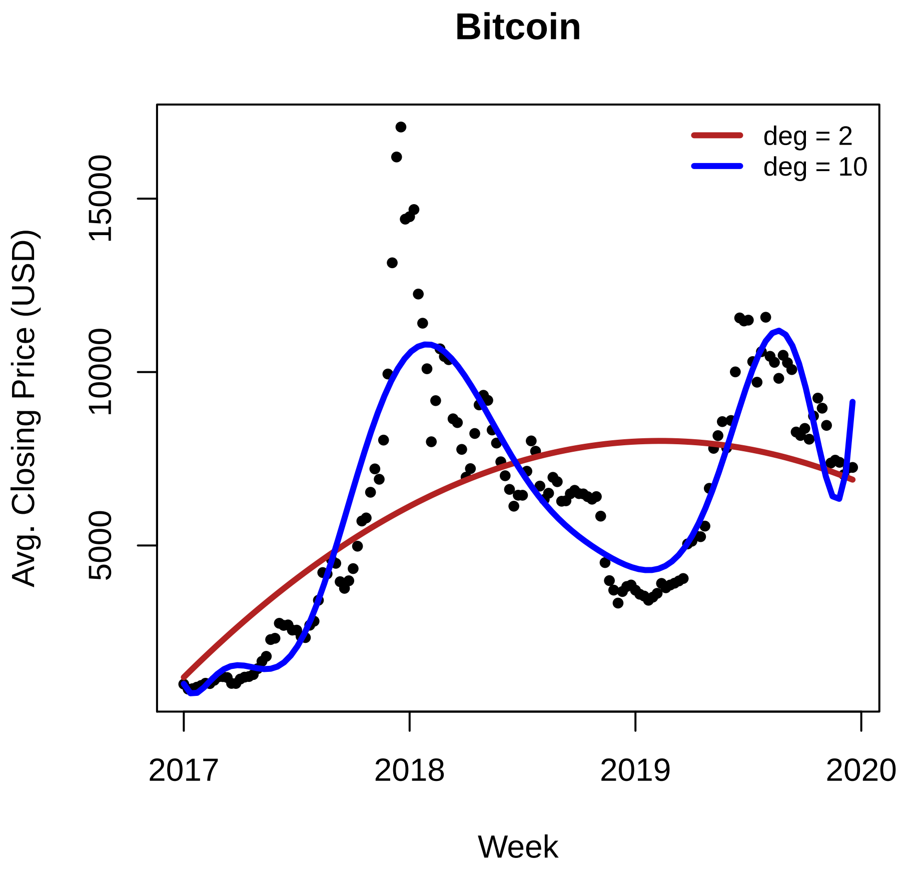
SOLUTION:

```
btc <- read.csv("bitcoin.csv", header = TRUE)
N <- dim(btc)[1]
```

(b) [5 points] Recreate the scatter plot from above and overlay fitted polynomials with degrees 2 and 10 to the data (in different colours distinguished with a legend). **Note:** Use the `getmuhat` function appended at the bottom of the assignment.

SOLUTION:

```r
# Make the plot
plot(x = btc$Week, y = btc$Avg_Closing_Price, pch = 16, cex = 0.75, ylab = "Avg. Closing Price (USD)",
    xlab = "Week", main = "Bitcoin", xaxt = "n")
axis(side = 1, at = c(1, 53, 105, 157), labels = 2017:2020)

# Fit a degree 2 polynomial and add it to the plot
muhat2 <- getmuhat(data.frame(x = btc$Week, y = btc$Avg_Closing_Price),
    2)
curve(muhat2, from = 1, to = N, add = TRUE, col = "firebrick", lwd = 3)

# Fit a degree 10 polynomial and add it to the plot
muhat10 <- getmuhat(data.frame(x = btc$Week, y = btc$Avg_Closing_Price),
    10)
curve(muhat10, from = 1, to = N, add = TRUE, col = "blue", lwd = 3)

# Add a legend
legend("topright", legend = c("deg = 2", "deg = 10"), col = c("firebrick",
    "blue"), lwd = 3, cex = 0.8, bty = "n")
```



(c) [2 points] Use the following code to generate $N_S = 50$ samples of size $n = 75$. Note that you will also need to run the `getSampleComp` and `getXYSample` functions appended at the end of the assignment. (This is not for points).

```r
N_S <- 50
n <- 75
set.seed(341)
samps <- lapply(1:N_S, FUN = function(i) {
    getSampleComp(btc, n)
})
Ssamples <- lapply(samps, FUN = function(Si) {
    getXYSample("Week", "Avg_Closing_Price", Si, btc)
})
Tsamples <- lapply(samps, FUN = function(Si) {
    getXYSample("Week", "Avg_Closing_Price", !Si, btc)
})
```

Fit polynomials of degree 2 and 10 to every sample.

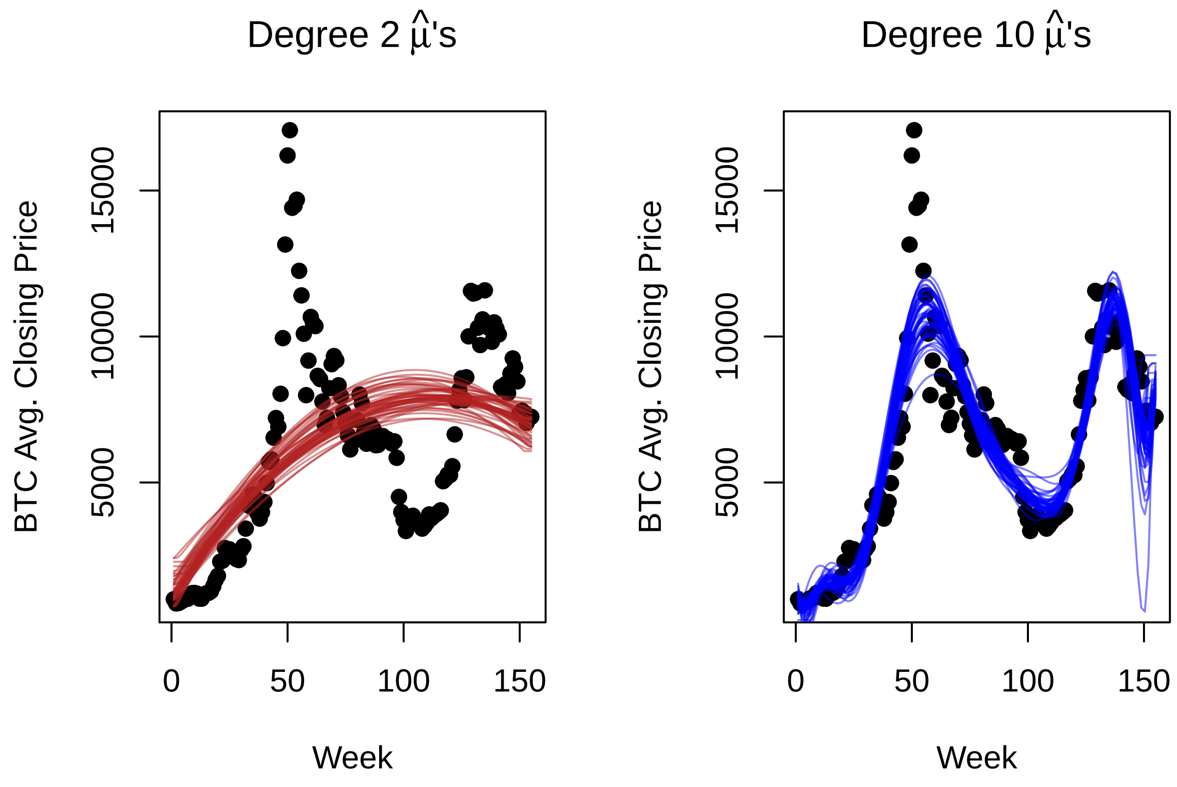
SOLUTION:

```r
muhats2 <- lapply(Ssamples, getmuhat, complexity = 2)
muhats10 <- lapply(Ssamples, getmuhat, complexity = 10)
```

(d) [4 points] Using `par(mfrow=c(1,2))` plot the data and overlay all the fitted polynomials from part (c) with degree 2 depicted in the left plot and degree 10 in the right plot. Use colours that are consistent with part (b).

SOLUTION:

```r
par(mfrow = c(1, 2))
xlim <- range(btc$Week)
plot(btc$Week, btc$Avg_Closing_Price, pch = 19, xlab = "Week", ylab = "BTC Avg. Closing Price",
    main = bquote("Degree 2" ~ hat(mu) * "'s"))

for (i in 1:N_S) {
    curveFn <- muhats2[[i]]
    curve(curveFn, from = 1, to = 155, add = TRUE, col = adjustcolor("firebrick",
        0.5))
}

plot(btc$Week, btc$Avg_Closing_Price, pch = 19, xlab = "Week", ylab = "BTC Avg. Closing Price",
    main = bquote("Degree 10" ~ hat(mu) * "'s"))

for (i in 1:N_S) {
    curveFn <- muhats10[[i]]
    curve(curveFn, from = 1, to = 155, add = TRUE, col = adjustcolor("blue",
        0.5))
}
```



(e) [3 points] Using the $N_S = 50$ samples of size $n = 75$ generated in part (c), calculate the APSE (and each of its components) for degrees in `0:20`. In particular, print out a table that shows for each degree `apse`, `var_mutilde`, `bias2` and `var_y`. **Note:** Use the `apse_all` function appended at the bottom of the assignment. Doing so will require the `getmubar` and `getmuFun` functions also appended at the bottom of the assignment.

SOLUTION:

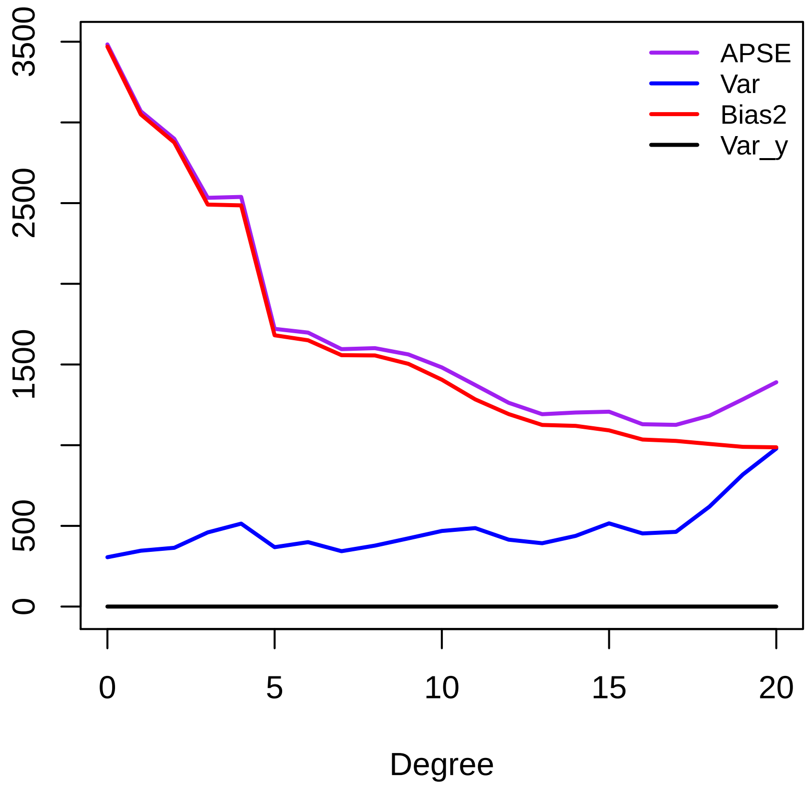
```r
tau.week <- gettauFun(btc, "Week", "Avg_Closing_Price")
degrees <- 0:20
apse_vals <- sapply(degrees, FUN = function(complexity) {
    apse_all(Ssamples, Tsamples, complexity = complexity, tau = tau.week)
})

# Print out the results
t(rbind(degrees, apse = round(apse_vals, 5)))
```

```
##       degrees     apse var_mutilde      bias2 var_y
##  [1,]       0 12135281    93539.31 12041741.8     0
##  [2,]       1  9419123   119775.56  9299347.3     0
##  [3,]       2  8402154   132667.79  8269486.5     0
##  [4,]       3  6415690   211209.86  6204479.7     0
##  [5,]       4  6444325   264147.89  6180176.7     0
##  [6,]       5  2961362   135385.70  2825975.9     0
##  [7,]       6  2882192   159410.95  2722780.9     0
##  [8,]       7  2544404   117743.31  2426660.4     0
##  [9,]       8  2564073   142716.71  2421356.6     0
## [10,]       9  2440974   178744.00  2262230.3     0
## [11,]      10  2195670   219476.25  1976193.6     0
## [12,]      11  1884217   236328.36  1647888.3     0
## [13,]      12  1593913   171720.65  1422192.3     0
## [14,]      13  1420645   154009.37  1266635.7     0
## [15,]      14  1445599   191749.02  1253850.0     0
## [16,]      15  1457910   265877.69  1192032.4     0
## [17,]      16  1276581   205276.20  1071304.7     0
## [18,]      17  1267785   214269.42  1053515.2     0
## [19,]      18  1398607   383106.61  1015500.6     0
## [20,]      19  1649143   669438.42   979704.3     0
## [21,]      20  1932223   958357.65   973864.9     0
```

(f) [4 points] Using your results from part (e) construct a plot whose x-axis is degree and which has four lines: one for each of `apse`, `var_mutilde`, `bias2` and `var_y`. Specifically, and for interpretability, plot `sqrt(apse)`, `sqrt(var_mutilde)`, `sqrt(bias2)` and `sqrt(var_y)` vs. `degree`. Be sure to distinguish the lines with different colours and a legend. Briefly describe the trends you see in the plot.

SOLUTION:

```r
plot(degrees, sqrt(apse_vals[1, ]), xlab = "Degree", ylab = "", type = "l",
    ylim = c(0, max(sqrt(apse_vals))), col = "purple", lwd = 2)
lines(degrees, sqrt(apse_vals[2, ]), col = "blue", lwd = 2)
lines(degrees, sqrt(apse_vals[3, ]), col = "red", lwd = 2)
lines(degrees, sqrt(apse_vals[4, ]), col = "black", lwd = 2)
legend("topright", legend = c("APSE", "Var", "Bias2", "Var_y"), col = c("purple",
    "blue", "red", "black"), lwd = 2, bty = "n", cex = 0.8)
```



(g) [3 points] Based on your findings in parts (e) and (f), which degree polynomial has the best predictive accuracy? Repeat part (b) but just include the fitted polynomial with the degree you found to be best.

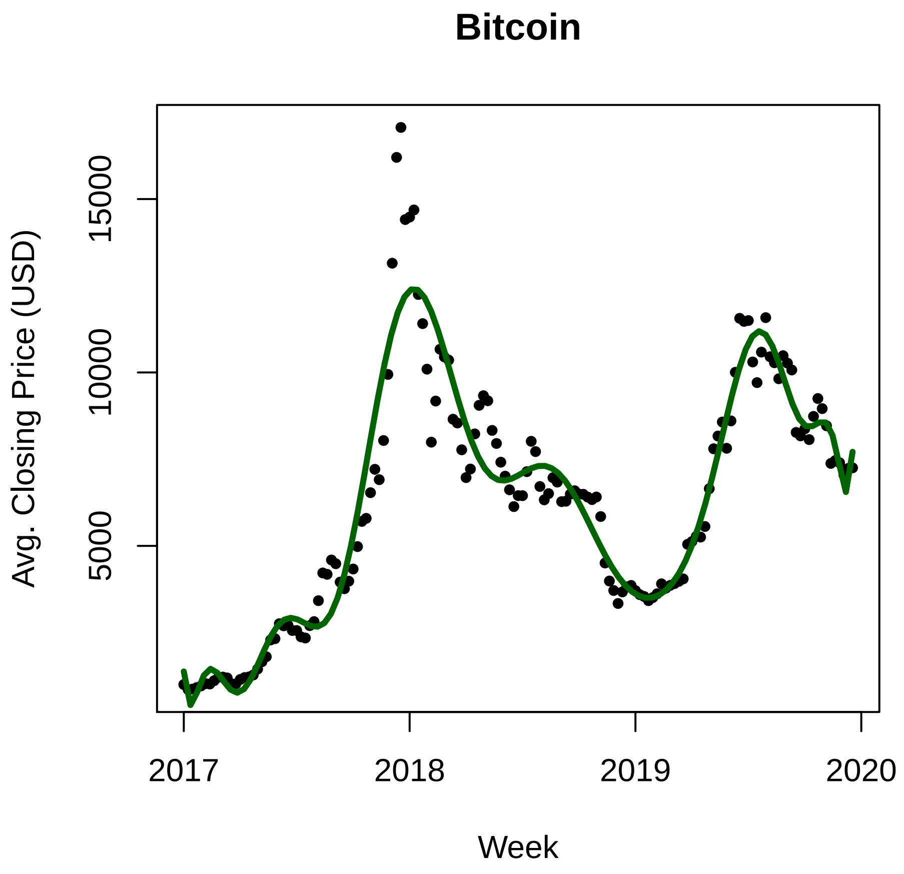
SOLUTION:

```r
best.deg <- degrees[which.min(apse_vals[1, ])]
print(best.deg)
```

```
## [1] 17
```

The best degree is the one which yields the smallest APSE, which in this case is 17

```r
muhat <- getmuhat(data.frame(x = btc$Week, y = btc$Avg_Closing_Price),
    best.deg)

plot(x = btc$Week, y = btc$Avg_Closing_Price, pch = 16, cex = 0.75, ylab = "Avg. Closing Price (USD)",
    xlab = "Week", main = "Bitcoin", xaxt = "n")
axis(side = 1, at = c(1, 53, 105, 157), labels = 2017:2020)
curve(muhat, from = xlim[1], to = xlim[2], add = TRUE, col = "darkgreen",
    lwd = 3)
```


Bitcoin
Avg. Closing Price (USD)
Week