STAT 341: Assignment 4

DUE: Friday April 3 by 11:59pm EST

NOTES

Your assignment must be submitted by the due date listed at the top of this document, and it must be submitted electronically in .pdf format via Crowdmark. This means that your responses for different question parts should begin on separate pages of your .pdf file. Note that your .pdf solution file must have been generated by R Markdown. Additionally:

- For mathematical questions: your solutions must be produced by LaTeX (from within R Markdown). Neither screenshots nor scanned/photographed handwritten solutions will be accepted – these will receive zero points.
- For computational questions: R code should always be included in your solution (via code chunks in R Markdown). If code is required and you provide none, you will received zero points.
- For interpretation questions: plain text (within R Markdown) is required. Text responses embedded as comments within code chunks will not be accepted.

Organization and comprehensibility is part of a full solution. Consequently, points will be deducted for solutions that are not organized and incomprehensible.

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 game
`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.

(b) [1 point] By resampling S with replacement, construct $B = 1000$ bootstrap samples $S^\star_1, S^\star_2, \ldots, S^\star_{1000}$.

(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 $a(\mathcal{P})$

ii. [2 points] Calculate $a(\mathcal{S}^\star_b)$ 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.

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

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

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 $D = 100$.

(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 $a(\mathcal{P})$

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.

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

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

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 $D = 100$.

(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 $a(\mathcal{P})$

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.

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

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

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 $D = 100$.

(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 $a(\mathcal{P})$

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.

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

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

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 $D = 100$.

(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.

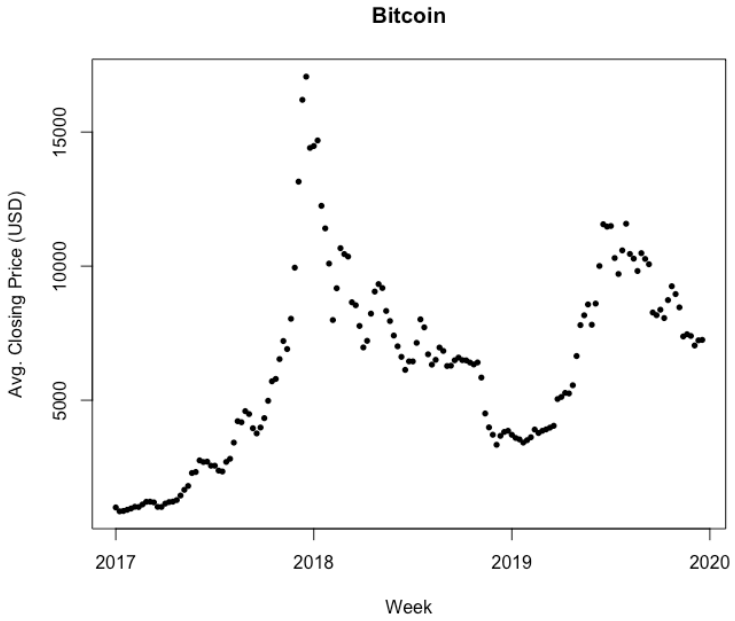
ii. [1 point] List one advantage and one disadvantage of percentile method intervals.

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

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`.

(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.

(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).

```
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.

(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).

(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 `gettauFun` functions also appended at the bottom of the assignment.

(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.

(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.

USEFUL FUNCTIONS

```
bootstrap_t_interval <- function(S, a, confidence, B, D) {
    ## Inputs: S = an n element array containing the variate values in the
    ## sample a = a scalar-valued function that calculates the attribute a()
    ## of interest confidence = a value in (0,1) indicating the confidence
    ## level B = a numeric value representing the outer bootstrap count of
    ## replicates (used to calculate the lower and upper limits) D = a
    ## numeric value representing the inner bootstrap count of replicates
    ## (used to estimate the standard deviation of the sample attribute for
    ## each (outer) bootstrap sample)

    Pstar <- S
    aPstar <- a(Pstar)
    sampleSize <- length(S)
    ## get (outer) bootstrap values
    bVals <- sapply(1:B, FUN = function(b) {
        Sstar <- sample(Pstar, sampleSize, replace = TRUE)
        aSstar <- a(Sstar)
        ## get (inner) bootstrap values to estimate the SD
        Pstarstar <- Sstar
        SD_aSstar <- sd(sapply(1:D, FUN = function(d) {
            Sstarstar <- sample(Pstarstar, sampleSize, replace = TRUE)
            ## return the attribute value
            a(Sstarstar)
        }))
        z <- (aSstar - aPstar)/SD_aSstar
        ## Return the two values
        c(aSstar = aSstar, z = z)
    })
    SDhat <- sd(bVals["aSstar", ])
    zVals <- bVals["z", ]
    ## Now use these zVals to get the lower and upper c values.
    cValues <- quantile(zVals, probs = c((1 - confidence)/2, (confidence +
        1)/2), na.rm = TRUE)
    cLower <- min(cValues)
    cUpper <- max(cValues)
    interval <- c(lower = aPstar - cUpper * SDhat, middle = aPstar, upper = aPstar -
        cLower * SDhat)
    interval
}
```

```
getmuhat <- function(sampleXY, complexity = 1){
  formula <- paste0("y ~ ",
                    if (complexity==0) {
                      "1"
                    } else
                      paste0("poly(x, ", complexity, ", raw = FALSE)")
                    #paste0("bs(x, ", complexity, ")")
  )

  fit <- lm(as.formula(formula), data = sampleXY)
  tx = sampleXY$x
  ty = fit$fitted.values
```

```
  range.X = range(tx)
  val.rY  = c( mean(ty[tx == range.X[1]]),
               mean(ty[tx == range.X[2]]) )

  ## From this we construct the predictor function
  muhat <- function(x){
    if ("x" %in% names(x)) {
      ## x is a dataframe containing the variate named
      ## by xvarname
      newdata <- x
    } else
      ## x is a vector of values that needs to be a data.frame
    { newdata <- data.frame(x = x) }
    ## The prediction
    ##
    val = predict(fit, newdata = newdata)
    val[newdata$x < range.X[1]] = val.rY[1]
    val[newdata$x > range.X[2]] = val.rY[2]
    val
  }
  ## muhat is the function that we need to calculate values
  ## at any x, so we return this function from getmuhat
  muhat
}
```

```
getSampleComp <- function(pop, size, replace = FALSE) {
    N <- nrow(as.data.frame(pop))
    samp <- rep(FALSE, N)
    samp[sample(1:N, size, replace = replace)] <- TRUE
    samp
}
```

```
getXYSample <- function(xvarname, yvarname, samp, pop) {
    sampData <- pop[samp, c(xvarname, yvarname)]
    names(sampData) <- c("x", "y")
    sampData
}
```

```
apse_all <- function(Ssamples, Tsamples, complexity, tau) {
    ## average over the samples S
    N_S <- length(Ssamples)
    muhats <- lapply(Ssamples, FUN = function(sample) getmuhat(sample,
        complexity))
    ## get the average of these, mubar
    mubar <- getmubar(muhats)

    rowMeans(sapply(1:N_S, FUN = function(j) {
        T_j <- Tsamples[[j]]
        S_j <- Ssamples[[j]]
        muhat <- muhats[[j]]
        ## Take care of any NAs
        T_j <- na.omit(T_j)
        y <- c(S_j$y, T_j$y)
        x <- c(S_j$x, T_j$x)
```

```
        tau_x <- tau(x)
        muhat_x <- muhat(x)
        mubar_x <- mubar(x)

        apse <- (y - muhat_x)
        bias2 <- (mubar_x - tau_x)
        var_mutilde <- (muhat_x - mubar_x)
        var_y <- (y - tau_x)

        squares <- rbind(apse, var_mutilde, bias2, var_y)^2

        ## return means
        rowMeans(squares)
    }))
}
```

```
getmubar <- function(muhats) {
    function(x) {
        Ans <- sapply(muhats, FUN = function(muhat) {
            muhat(x)
        })
        apply(Ans, MARGIN = 1, FUN = mean)
    }
}
```

```
gettauFun <- function(pop, xvarname, yvarname) {
    pop = na.omit(pop[, c(xvarname, yvarname)])

    # rule = 2 means return the nearest y-value when extrapolating, same as
    # above.  ties = mean means that repeated x-values have their y-values
    # averaged, as above.
    tauFun = approxfun(pop[, xvarname], pop[, yvarname], rule = 2, ties = mean)
    return(tauFun)
}
```