STAT341 A2 Markdown

QUESTION 1: The Three-Hump Camel Function

a) By taking appropriate derivatives, determine the 2 x 1 gradient vector

$$\mathrm{g} = \bigtriangledown \rho(\theta, \mathcal{P})$$

$$\mathrm{g} = \begin{bmatrix} 4\theta_1 - 4.2\theta_1^2 + \theta_1^3 + \theta_2 \\ \theta_1 + 2\theta_2 \end{bmatrix}$$

b) Write rho and gradient functions for the Three-Hump Camel Function which take a single vector-valued input theta. Use the partial derivatives calculated in part (a) in your definition of gradient.

```
rho <- function(theta) {
  (2 * theta[1]^2) - (1.05 * theta[1]^4) + (theta[1]^6 / 6) +
    (theta[1] * theta[2]) + theta[2]^2
}

g <- function(theta) {
  c((4 * theta[1]) - (4.2 * theta[1]^3) + theta[1]^5 + theta[2],
    theta[1] + (2 * theta[2]))
}
```

c) In this question you will explore the surface of the Three-Hump Camel Function using gradient descent. In particular you will consider 5 different starting values and explore the impact of changing one's starting location. Using the `gradientDescent` function (from class) together with the `gridLineSearch` and `testConvergence` functions (from class) as well as the `rho` and `gradient` functions from part (b), find the solution to

$$\underset{\boldsymbol{\theta} \in \mathbb{R}^2}{\operatorname{argmin}}\, \rho(\boldsymbol{\theta})$$

for each of the following five starting values. In each case state which minima you've converged to (A, B, or C) and be sure to include the output from the `gradientDescent` function.

```
gradientDescent <- function(theta = 0, rhoFn, gradientFn, lineSearchFn, testConvergenceFn,
                            maxIterations = 100, tolerance = 1e-06, relative = FALSE, lambdaStepsize = 0
                            lambdaMax = 0.5) {

  converged <- FALSE
  i <- 0

  while (!converged & i <= maxIterations) {
```

```
    g <- gradientFn(theta)  ## gradient
    glength <- sqrt(sum(g^2))  ## gradient direction
    if (glength > 0)
      g <- g/glength

    lambda <- lineSearchFn(theta, rhoFn, g, lambdaStepsize = lambdaStepsize,
                           lambdaMax = lambdaMax)

    thetaNew <- theta - lambda * g
    converged <- testConvergenceFn(thetaNew, theta, tolerance = tolerance,
                                   relative = relative)
    theta <- thetaNew
    i <- i + 1
  }

  ## Return last value and whether converged or not
  list(theta = theta, converged = converged, iteration = i, fnValue = rhoFn(theta))
}

### line searching could be done as a simple grid search
gridLineSearch <- function(theta, rhoFn, g, lambdaStepsize = 0.01, lambdaMax = 1) { ## grid of lambda v
  lambdas <- seq(from = 0, by = lambdaStepsize, to = lambdaMax)
  ## line search
  rhoVals <- sapply(lambdas, function(lambda) {
    rhoFn(theta - lambda * g)
  })
  ## Return the lambda that gave the minimum
  lambdas[which.min(rhoVals)]
}

testConvergence <- function(thetaNew, thetaOld, tolerance = 1e-10,
                            relative = FALSE) {
  sum(abs(thetaNew - thetaOld)) < if (relative)
    tolerance * sum(abs(thetaOld)) else tolerance
}
```

i. $\widehat{\boldsymbol{\theta}} = (-1.5, 1.5)$

```
gradientDescent(theta = c(-1.5, 1.5), rhoFn = rho, gradientFn = g,
                lineSearchFn = gridLineSearch, testConvergenceFn =
                testConvergence, maxIterations = 1000)
```

```
## $theta
## [1] -1.7473565  0.8760595
##
## $converged
## [1] TRUE
##
## $iteration
## [1] 9
##
## $fnValue
## [1] 0.2986443
```

Solution: (-1.7473565 0.8760595)
Coverges to point A

ii. $\widehat{\boldsymbol{\theta}} = (1.5, 1.5)$

```
gradientDescent(theta = c(1.5, 1.5), rhoFn = rho, gradientFn = g,
                lineSearchFn = gridLineSearch, testConvergenceFn = testConvergence,
                maxIterations = 1000)
```

```
## $theta
## [1]  1.7443172 -0.8634349
##
## $converged
## [1] TRUE
##
## $iteration
## [1] 12
##
## $fnValue
## [1] 0.2987752
```

Solution: (1.7443172 -0.8634349)
Coverges to point C

iii. $\widehat{\boldsymbol{\theta}} = (1.5, -1.5)$

```
gradientDescent(theta = c(1.5, -1.5), rhoFn = rho, gradientFn = g,
                lineSearchFn = gridLineSearch, testConvergenceFn = testConvergence,
                maxIterations = 1000)
```

```
## $theta
## [1]  1.7473565 -0.8760595
##
## $converged
## [1] TRUE
##
## $iteration
## [1] 9
##
## $fnValue
## [1] 0.2986443
```

Solution: (1.7473565 -0.8760595)
Coverges to point C

iv. $\widehat{\boldsymbol{\theta}} = (-1.5, -1.5)$

```
gradientDescent(theta = c(-1.5, -1.5), rhoFn = rho, gradientFn = g,
                lineSearchFn = gridLineSearch, testConvergenceFn = testConvergence,
                maxIterations = 1000)
```

```
## $theta
## [1] -1.7443172  0.8634349
```

```
##
## $converged
## [1] TRUE
##
## $iteration
## [1] 12
##
## $fnValue
## [1] 0.2987752
```

Solution: (-1.7443172 0.8634349)
Coverges to point A

v. $\widehat{\boldsymbol{\theta}} = (-1, -1)$

```
gradientDescent(theta = c(-1, -1), rhoFn = rho, gradientFn = g,
                lineSearchFn = gridLineSearch, testConvergenceFn = testConvergence,
                maxIterations = 1000)
```

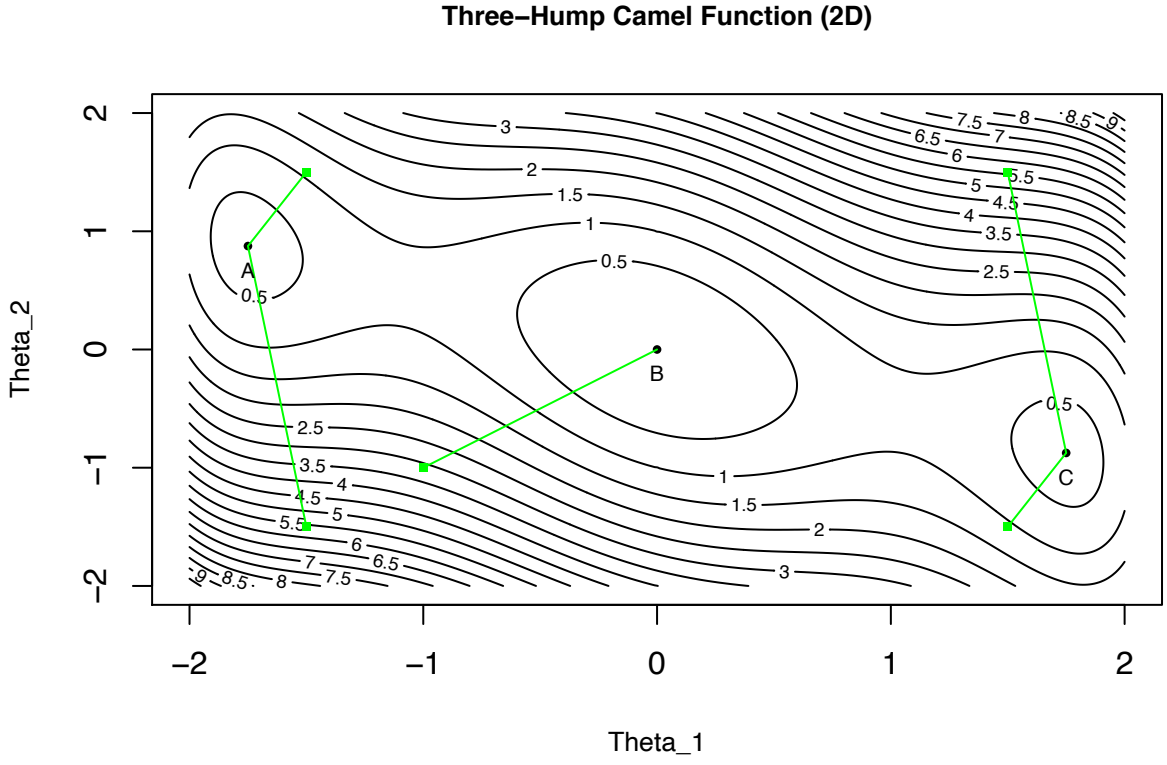
```
## $theta
## [1] -0.001546561 -0.003261060
##
## $converged
## [1] TRUE
##
## $iteration
## [1] 5
##
## $fnValue
## [1] 2.046164e-05
```

Solution: (-0.001546561 -0.003261060)
Coverges to point B

d) Recreate the contour plot shown above. You may find the functions `outer`, `image`, and `contour` useful for this task. Include on this plot `green` squares at each of the starting points specified in (c) as well as `green` line `segments` connecting these starting points with their respective points of convergence.

```
# values for theta1 and theta2
xaxs <- yaxs <- seq(-2, 2, by=0.01)
# matrix to contain values of p(theta1,ptheta2)
z <- matrix(NA, nrow=length(xaxs), ncol=length(yaxs))
for (i in 1:length(xaxs)) {
  for (j in 1:length(yaxs))
    z[i,j]=rho(c(xaxs[i],yaxs[j]))
}

# local minimas of three-hump camel function
xpnt <- c(-1.75,0,1.75)
ypnt <- c(0.875,0,-0.875)

# points from part c)
sxpnt <- c(-1.5,1.5,1.5,-1.5,-1)
```

```
sypnt <- c(1.5,1.5,-1.5,-1.5,-1)

# contour plot with local minimas and line segements
# connected with points from part c)
conlevels <- seq(0.5,10,by=0.5)
contour(xaxs, yaxs, z, levels=conlevels,
        xlab="Theta_1",
        ylab="Theta_2",
        main="Three-Hump Camel Function (2D)",
        cex.main=0.8,cex.lab=0.8)
points(xpnt,ypnt,pch=16,cex=0.6)
points(sxpnt,sypnt,pch=15,col="green",cex=0.6)
segments(-1.75,0.875,-1.5,1.5,col="green")
segments(1.75,-0.875,1.5,1.5,col="green")
segments(1.75,-0.875,1.5,-1.5,col="green")
segments(-1.75,0.875,-1.5,-1.5,col="green")
segments(0,0,-1,-1,col="green")
text(xpnt,ypnt,labels=c("A","B","C"),pos=c(1,1,1),cex=0.65)
```



e) Based on what you found in part (c), and visualized in part (d), explain the importance of the starting value when performing non-convex optimization (when locating a global optimum is desired).

The starting values help identify the global/local optimums in non-convex test functions. In this case, the starting points help identify which local minimas are closest

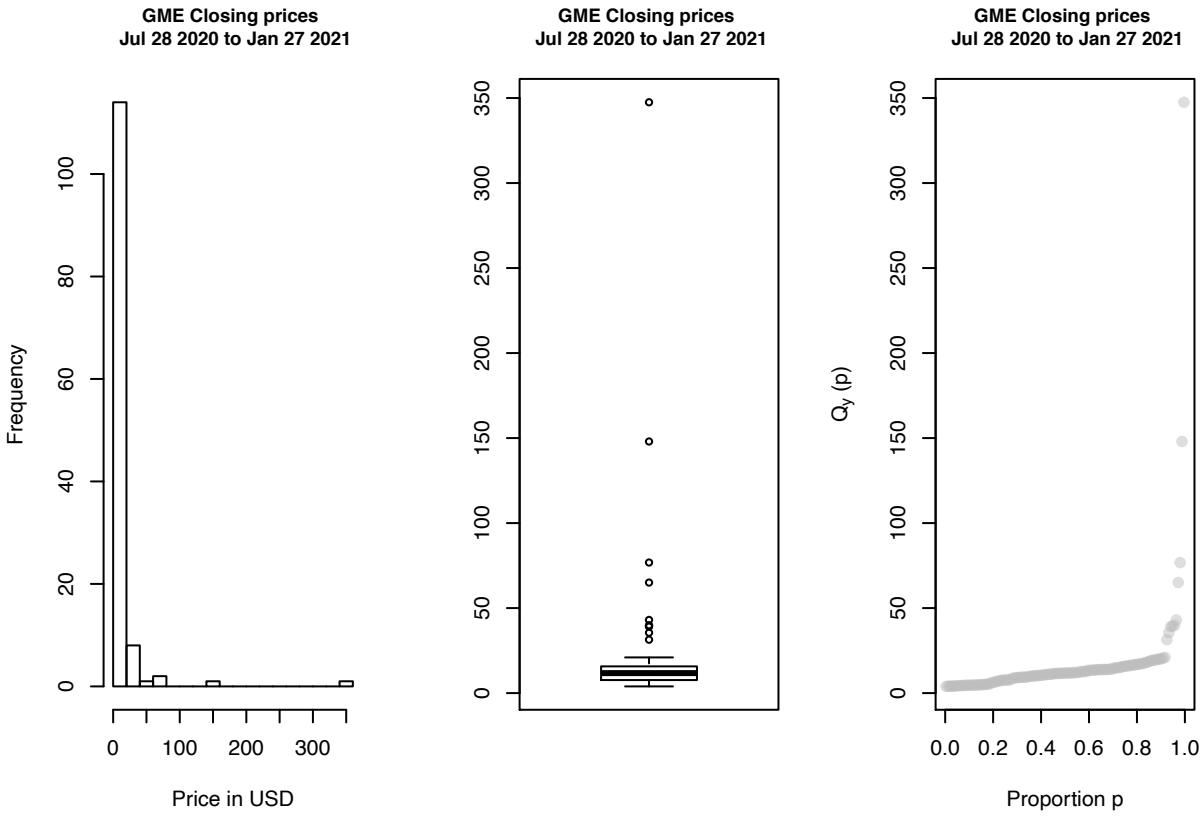
QUESTION 2: The GME Short Squeeze

a) Construct a 1×3 grid of plots of `Adj_Close` where

- the left plot is a histogram of `Adj_Close`, constructed using Scott's rule for binning,
- the middle plot is a boxplot of `Adj_Close`, and
- the right plot is a quantile plot of `Adj_Close`.

On each plot clearly label all axes and provide an informative title. Based on these plots, describe the shape of the distribution and provide an interval of daily adjusted closing prices that might be considered "typical". Justify your choice of range.

```
gme <- read.csv("GME.csv")
adjc <- gme["Adj_Close"]
par(mfrow = c(1, 3))
# histogram
hist(adjc[1:127,],
     xlab = "Price in USD",
     main = "GME Closing prices \n Jul 28 2020 to Jan 27 2021",
     cex.main=0.9,breaks="Scott")
# boxplot
boxplot(adjc[1:127,],
        main = "GME Closing prices \n Jul 28 2020 to Jan 27 2021",
        cex.main=0.9)
# quantile plot
qvals <- sort(adjc[1:127,])
pvals <- ppoints(127)
plot(pvals,qvals,pch=19,col=adjustcolor("grey", alpha = 0.5),
     xlim =c(0,1),
     xlab = "Proportion p",
     ylab = bquote("Q"["y"]~"(p)"),
     main = "GME Closing prices \n Jul 28 2020 to Jan 27 2021",
     cex.main=0.9)
```



Based on the plots:

The shape of the distribution looks similar to an exponential distribution as it is increasing throughout the period based on the quantile plot

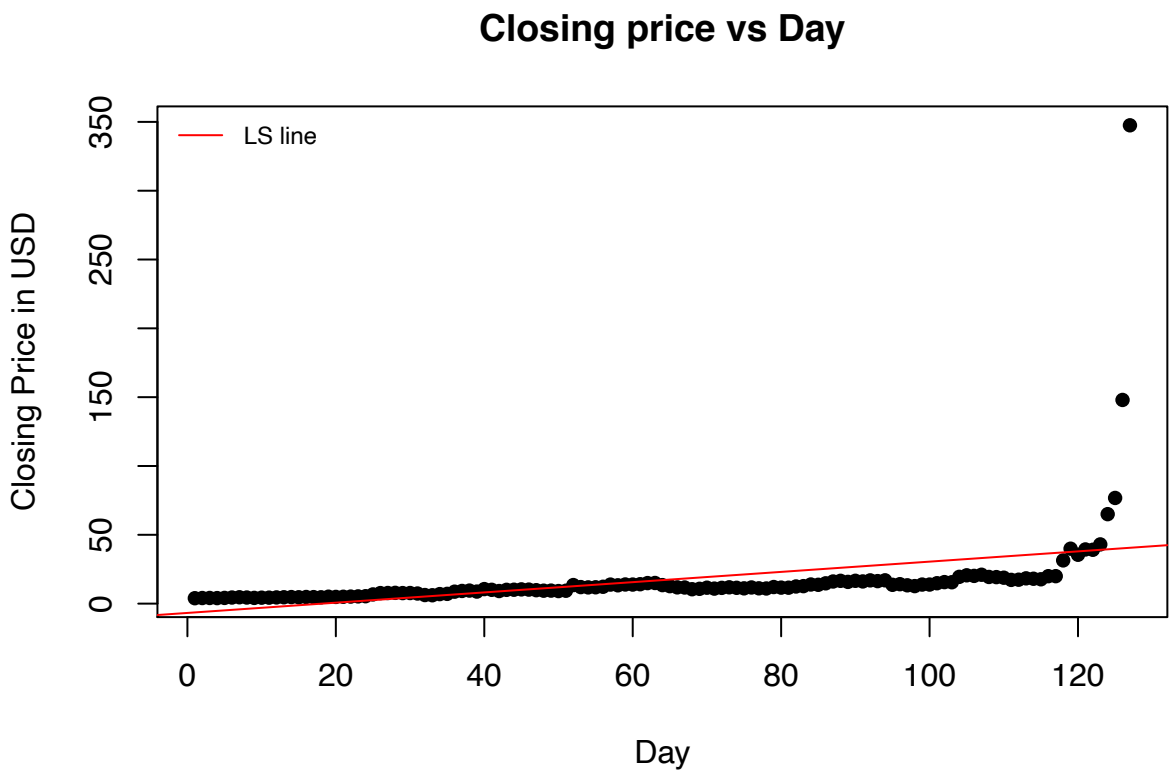
A typical closing price interval would be [0,25] since majority of the prices are between 0 and 25 based on the histogram and boxplot.

b) Visualizations such as those in part (a) tell us a lot about the distribution of a variate. I claim that no matter the data, among the histogram, the boxplot, and the quantile plot, the quantile plot is always the most useful of the three. Now, I want you to pick a side and defend your position. In one paragraph, state whether you *agree* or *disagree* with my claim and provide concrete supporting arguments. If you *disagree*, state which plot you think is superior, and explain why. You will be graded on the accuracy and clarity of your argument and the quality of your writing in accordance with the rubric below. **Hint:** Brainstorm a long list of properties of a distribution that you might wish to summarize and then think about which plot(s) can provide those summaries and which one(s) cannot.

I agree with the statement. The quantile plot visualizes quantiles of data the best which helps identify the measures of locations also measures spread and concentration. Ideally a boxplot measures spread and histograms measures concentrations, but a quantile plot can measure both spread and concentrations making it a superior plot.

c) Construct a scatter plot of `Adj_Close` versus `Day_Num`, and add the least squares regression line to this plot. Note that you may use the `lm` function to determine the equation of this line. Be sure to add a title and informative axis labels.

```
adjc <- gme["Adj_Close"]
day <- gme["Day_Num"]
plot(day[1:127,],adjc[1:127,],
     xlab = "Day",
     ylab = "Closing Price in USD",
     main = "Closing price vs Day", pch=16)
abline(lm(adjc[1:127,] ~ day[1:127, ]), col = "red")
legend("topleft",legend="LS line",col="red",
       cex=0.75,bty = "n", lty = 1)
```



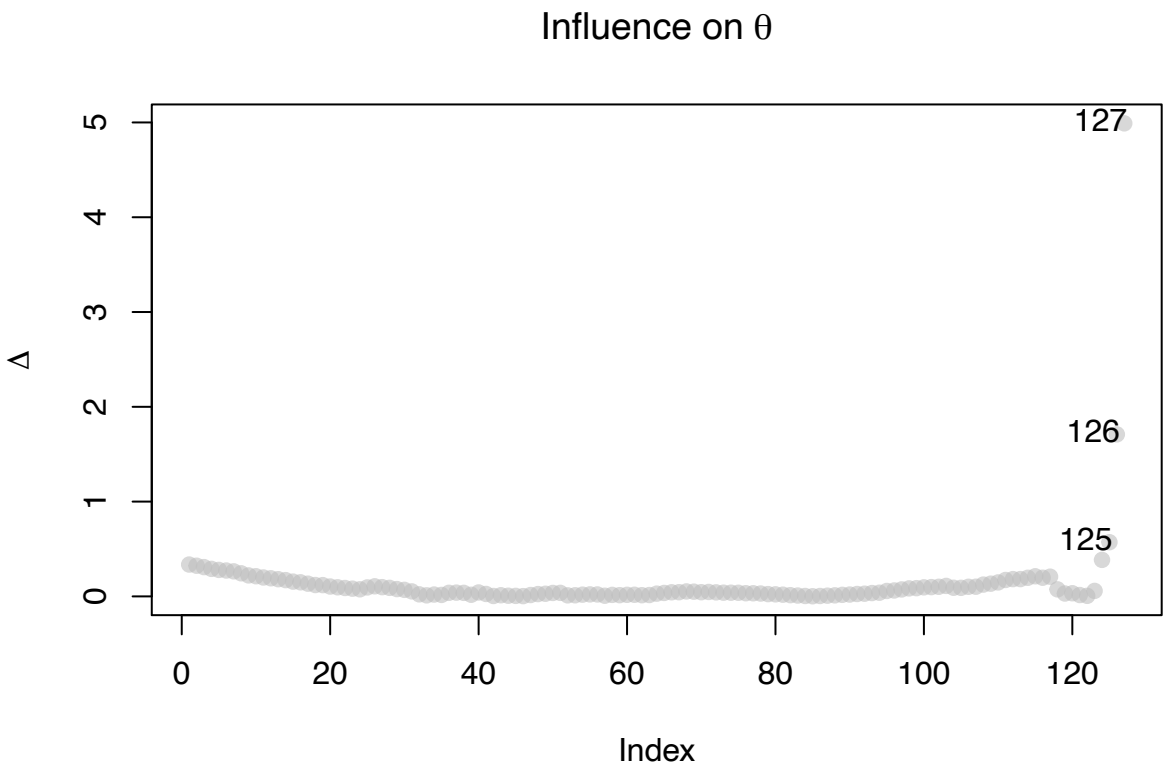
d) For each unit in the population, calculate the influence that it has on the fitted regression line from part (c), using the following definition of influence:

$$\Delta(\boldsymbol{\theta}, u) = ||\widehat{\boldsymbol{\theta}} - \widehat{\boldsymbol{\theta}}_{[-u]}||_2$$

where $\widehat{\boldsymbol{\theta}} = \left(\widehat{\alpha}, \widehat{\beta}\right)^T$ are the regression coefficients estimated from all of the data, $\widehat{\boldsymbol{\theta}}_{[-u]} = \left(\widehat{\alpha}_{[-u]}, \widehat{\beta}_{[-u]}\right)^T$ are the regression coefficients estimated from all of the data excluding unit u, and $||\cdot||_2$ is the Euclidean norm. Construct a scatter plot of all influence values and determine on which `Dates` the three most influential closing prices occurred.

```
# calculate influence of units on fitted regression line
gmed <- read.csv("GME.csv", header=T,
```

```
                colClasses = c("character", "numeric", "numeric"))
model.pop <- lm(adjc[1:127,] ~ day[1:127, ])
theta.hat <- model.pop$coef

N = nrow(adjc)
delta = matrix(0, nrow = N, ncol = 2)

for (i in 1:N) {
  temp.model = lm(Adj_Close ~ Day_Num,data=gme[-i, ])
  delta[i, ] = abs(theta.hat - temp.model$coef)
}

delta2 = apply(X = delta, MARGIN = 1, FUN = function(z) {
  sqrt(sum(z^2)) })

# create scatter plot to display influence on regression
plot(delta2, ylab = bquote(Delta), main = bquote("Influence on" ~ theta),
     pch = 19, col = adjustcolor("grey", 0.6))

obs = c(125,126,127)
text(obs + 3, delta2[obs], obs, pos=2)
```

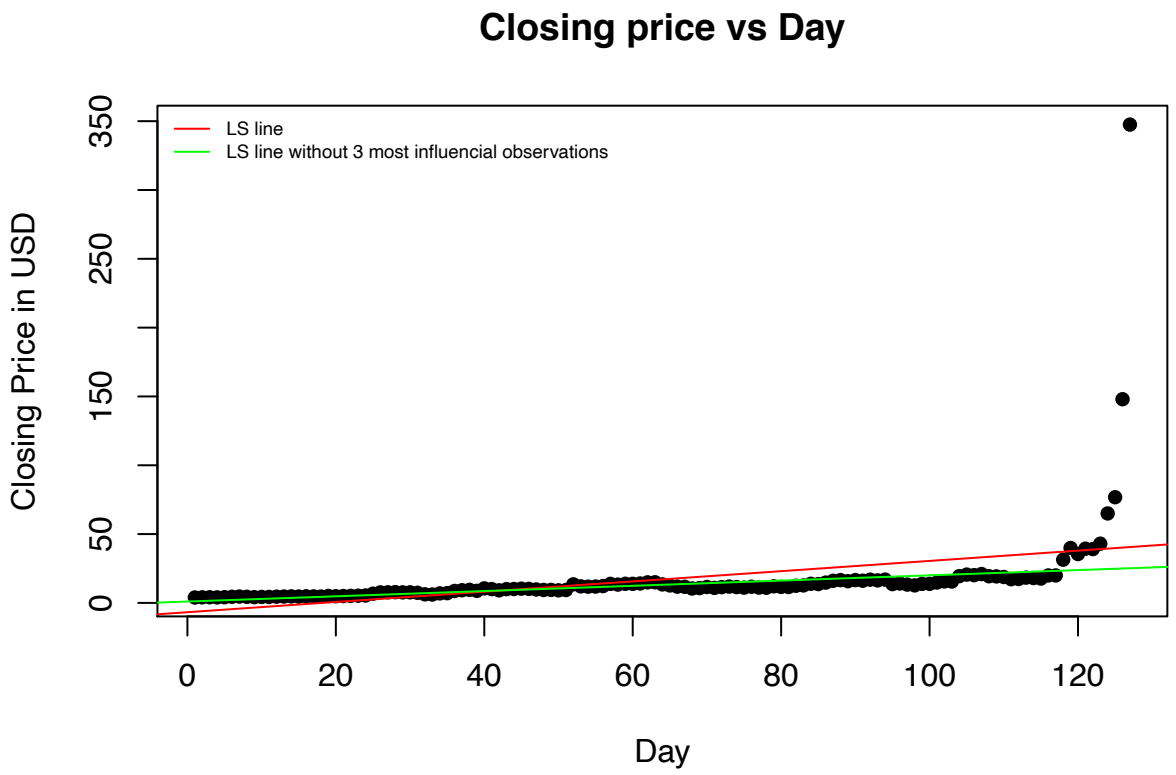


Based on the scatter plots, the three most influencial closing prices occured on days 125, 126, and 127 which is: Jan 25 2021, Jan 26 2021, Jan 27 2021

e) Re-construct the scatter plot from part (c). Remove the three most influential observations from the population and calculate the least squares regression line using the observations that remain (you may

use `lm` again). Add this line to the plot in a colour different from what you used on the first line. Add a legend to the plot that distinguishes between the two lines. Which line best represents the relationship between `Adj_Close` and `Day_Num`? Briefly justify your choice.

```
adjc <- gme["Adj_Close"]
day <- gme["Day_Num"]
adjc2 <- adjc[1:124, ]
day2 <- day[1:124, ]

# scatter plot of closing price vs day with LS line and LS line
# without the 3 most influences observations
plot(day[1:127,], adjc[1:127,],
     xlab = "Day",
     ylab = "Closing Price in USD",
     main = "Closing price vs Day", pch=16)
abline(lm(adjc[1:127,] ~ day[1:127, ]), col = "red")
abline(lm(adjc2 ~ day2), col = "green")
legend("topleft",legend=c("LS line","LS line without 3 most influencial observations"),
       col=c("red","green"), cex=0.60, bty = "n", lty = 1)
```



Based on the scatter plot, the line without the 3 most influencial observations fit better with the typical data hence it best represent the relationship between Adj_Close and Day_Num

f)

i. By taking appropriate derivatives, determine the 2×1 gradient vector $\boldsymbol{g} = \nabla\rho(\boldsymbol{\theta}; \mathcal{P})$. Show your work.

note that

$$\frac{\mathsf{d}\rho_k(r)}{\mathsf{d}r} = \begin{cases} -r + \frac{2r^3}{k^2} - \frac{r^5}{k^4} & \text{for } |r| \leq k \\ 0 & \text{for } |r| > k \end{cases}$$

and

$$\frac{\mathsf{d}r_u}{\mathsf{d}\theta} = \begin{bmatrix} \frac{\mathsf{d}r_u}{\mathsf{d}\alpha} \\ \frac{\mathsf{d}r_u}{\mathsf{d}\beta} \end{bmatrix} = -\begin{bmatrix} 1 \\ x_u \end{bmatrix}$$

thus the gradient can be written as

$$\boldsymbol{g} = \sum_{u \in \mathcal{P}} \frac{\mathsf{d}\rho_k(r_u)}{\mathsf{d}\theta} = \sum_{u \in \mathcal{P}} \frac{\mathsf{d}\rho_k(r_u)}{\mathsf{d}r_u} \times \frac{\mathsf{d}r_u}{\mathsf{d}\theta} = -\begin{bmatrix} \sum_{u \in \mathcal{P}} \frac{\mathsf{d}\rho_k(r_u)}{\mathsf{d}r} \\ \sum_{u \in \mathcal{P}} \frac{\mathsf{d}\rho_k(r_u)}{\mathsf{d}r} x_u \end{bmatrix}$$

ii. Write *factory functions* `createRobustTukeyRho(x, y, kval)` and `createRobustTukeyGradient(x, y, kval)` which take in as inputs the data and a value for the constant k, and which respectively return as output the Tukey Biweight objective function and the corresponding gradient function. **Hint:** Use the `createRobustHuberRho` and `createRobustHuberGradient` functions from the lecture notes as a guide. You may also use the following two functions as necessary.

```
tukey.fn <- function(r, k) {
  val = (r^2)/2 - (r^4)/(2 * k^2) + (r^6)/(6 * k^4)
  subr = abs(r) > k
  val[subr] = (k^2)/6
  return(val)
}

createRobustTukeyRho <- function(x, y, kval) { ## local variable
  ## Return this function
  function(theta) {
    alpha <- theta[1]
    beta <- theta[2]
    sum(tukey.fn(y - alpha - beta * x, k = kval))
  }
}

tukey.fn.prime <- function(r, k) {
  val = r - (2 * r^3)/(k^2) + (r^5)/(k^4)
  subr = abs(r) > k
  val[subr] = 0
  return(val)
}
```

```
createRobustTukeyGradient <- function(x, y, kval) { ## local variables
  ## Return this function
  function(theta) {
    alpha <- theta[1]
    beta <- theta[2]
    ru = y - alpha - beta * x
    rhok = tukey.fn.prime(ru, k = kval)
    -1 * c(sum(rhok * 1), sum(rhok * x))
  }
}
```

iii. Using the `nlminb` function with the `rho` and `gradient` functions created by your factory functions from part ii., find $\widehat{\boldsymbol{\theta}} = \left(\widehat{\alpha}, \widehat{\beta}\right)^T$, the solution to

$$\underset{\boldsymbol{\theta} \in \mathbb{R}^2}{\operatorname{argmin}} \, \rho(\boldsymbol{\theta}; \mathcal{P}).$$

Start the optimization at $\widehat{\boldsymbol{\theta}}_0 = (0, 1)^T$ and use $k = 4.685$. For full points be sure to include the output from the `nlminb` function. **Hint:** Your results should be similar to the coefficients determined by `rlm(Adj_Close ~ Day_Num, data = gme, psi = "psi.bisquare")`, where the function `rlm` can be found in the `MASS` package.

```
adjc <- gme["Adj_Close"]
day <- gme["Day_Num"]

nlminb(c(0,1), objective=createRobustTukeyRho(day,adjc,4.685),
       gradient=createRobustTukeyGradient(day,adjc,4.685))
```

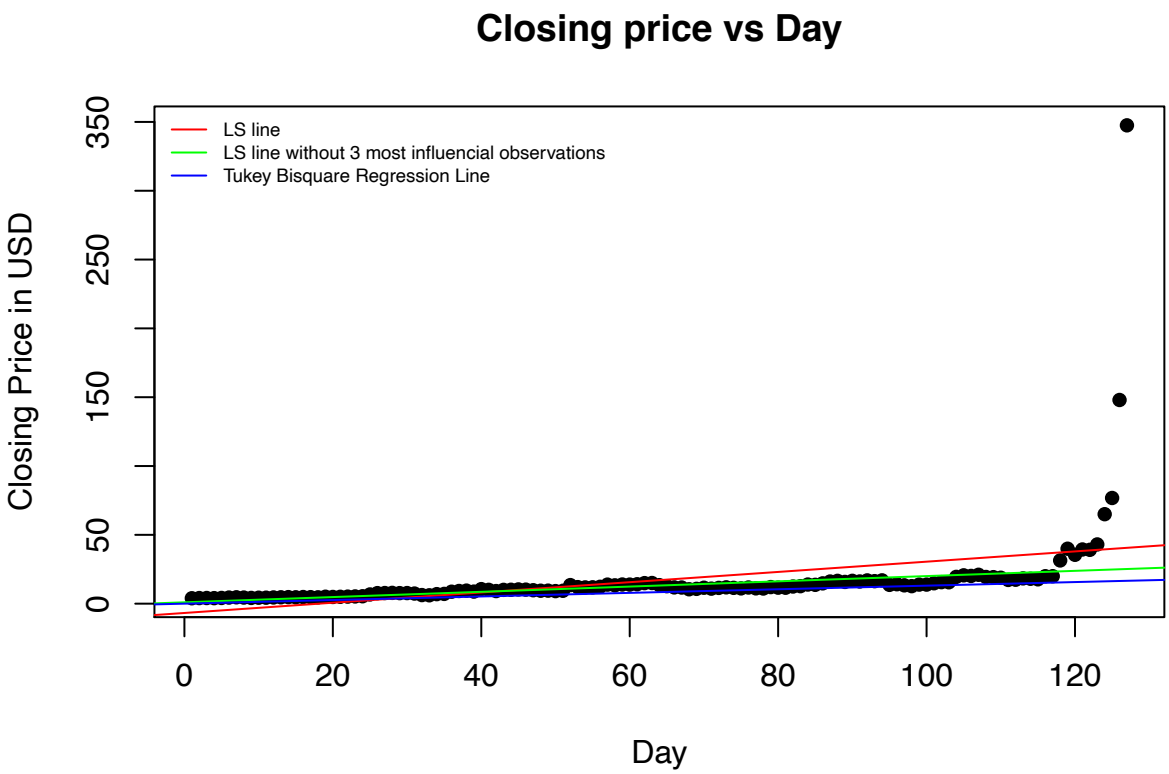
```
## $par
## [1] 3.3106361 0.1307125
##
## $objective
## [1] 142.3393
##
## $convergence
## [1] 0
##
## $iterations
## [1] 12
##
## $evaluations
## function gradient
##       21       13
##
## $message
## [1] "relative convergence (4)"
```

Based on the output, the solution is:

$\widehat{\boldsymbol{\theta}} = (3.3106361, 0.1307125)^T$

iv. Re-construct the scatter plot from part (e). In a third colour, add the Tukey Bisquare regression line that you calculated in part iii. Be sure to include a legend that distinguishes among the three lines. Does this robust regression line do a better job at representing the relationship between `Adj_Close` and `Day_Num` than do the other two? State YES or NO and briefly justify your choice.

```
adjc <- gme["Adj_Close"]
day <- gme["Day_Num"]
adjc2 <- adjc[1:124, ]
day2 <- day[1:124, ]

# create Tukey Bisquare Regression Line
tukeyd <- 0:127
tukeycp <- 3.3106361 * tukeyd + 0.1307125

plot(day[1:127,], adjc[1:127,],
     xlab = "Day",
     ylab = "Closing Price in USD",
     main = "Closing price vs Day", pch=16)
abline(lm(adjc[1:127,] ~ day[1:127, ]), col = "red")
abline(lm(adjc2 ~ day2), col = "green")
abline(tukeyd,tukeycp, col = "blue")
legend("topleft",
       legend=c("LS line",
                "LS line without 3 most influencial observations",
                "Tukey Bisquare Regression Line"),
       col=c("red","green", "blue"), cex=0.60, bty = "n", lty = 1)
```



YES, this regression line does a better job in representing the relationship between Adj_Close and Day_Num since it best fits the typical data as it looks that it puts least emphasis on the outliers.