Midterm 2

Thu Nguyen

08 June, 2019

Libraries

```
library('quantreg', lib='C:/temp')
library("reshape", lib = "C:/temp")
library('ggplot2', lib='C:/temp')
library('tidyverse', lib='C:/temp')
```

Problem 1: Bootstrap tests for goodness-of-fit

We saw in lecture that when it comes to goodness-of-fit (GOF) testing, it is quite "natural" to obtain a p-value by permutation. It is also possible, however, to use the bootstrap for that purpose. Consider the two-sample situation for simplicity, although this generalizes to any number of samples. Thus assume a situation where we observe $X_1, \ldots, X_m$ iid from F and (independently) $Y_1, \ldots, Y_n$ iid from G, where F and G are two distributions on the real line. We want to test $F = G$ versus $F \neq G$. We may want to use a statistic $T = T(X_1, \ldots, X_m, Y_1, \ldots, Y_n)$ for that purpose, and the question is how to obtain a p-value for T via a bootstrap. The idea is, as usual, to estimate the "best" null distribution and bootstrap from that distribution. A natural approach to estimate the null distribution is to simply combine the two samples as one, and estimate the corresponding distribution via the empirical distribution. We thus use the empirical distribution from the combined sample to bootstrap from.

A. Write a function `bootGOFdiff( x, y, B = 2000)` that takes in two samples as vectors x and y, and a number of replicates B (Monte Carlo samples from the estimated null distribution), and returns the bootstrap GOF p-value for the difference in means $T = |\bar{X} = \bar{Y}|$.
B. Apply your function to the FIFA dataset to compare the wages of players ≤ 29 years old with older players (≥ 30 years old).

Part A: `bootGOFdiff( x, y, B = 2000)` function

```
# A: function
bootGOFdiff <- function( x, y, B = 2000 ) {
  # Combining vectors x and y into 1 single vector z
  z <- c(x,y)
  # Observed test-statistic from the original data
  t_obs <- abs( mean(x) - mean(y) )
  # Counter: to keep track of number of times bootstrapped stat > stat obs.
  counter <- 0
  # Bootstrapping
  for (b in 1:B) {
    # Generating new samples by bootstrap
    x_b <- sample(z, length(x), replace = TRUE)
```

```
    y_b <- sample(z, length(y), replace = TRUE)
    # Current bootstrap test-statistic
    t_b <- abs( mean(x_b) - mean(y_b) )
    # Checking for if stat_boot > stat_obs
    if (t_b > t_obs) {
      counter <- counter + 1
    }
  }
  p_value <- (counter+1)/(B+1)
  return(p_value)
}
```

Part B: Applying the function on the `fifa19` dataset

```
# B: applying on FIFA dataset
data <- read.csv('fifa19.csv')
# Clean up Wage column
data <- data %>%
  mutate(WageMultiplier = ifelse(str_detect(Wage, "K"), 1000,
                                 ifelse(str_detect(Wage, "M"), 1000000, 1))) %>%
  mutate(Wages = as.numeric(str_extract(Wage, "[[:digit:]]+\\.*[[:digit:]]*"))
         * WageMultiplier)
# Selecting Wages and Age columns
data <- data[,c('Wages', 'Age')]
# Filtering Wages ~ Ages
over29 <- subset(data, Age > 29)$Wages
below29 <- subset(data, Age <= 29)$Wages
# Applying the function on FIFA19 dataset
p <- bootGOFdiff(below29, over29)
mes <- 'p-value by bootstrapping: '
print(paste0(mes, p))
```

```
## [1] "p-value by bootstrapping: 0.000499750124937531"
```

Problem 2. (Local Absolute Linear Regression)

Local linear regression is a popular smoother. However, based on the squared errors, it is not robust. To make it more robust, one option is to use the absolute errors instead.

A. Write a function `localAbsLinearRegression( x, y, h, xnew = x)` that takes in paired vectors x (predictor) and y (response), and a bandwidth h, and computes the local absolute linear regression (use any kernel of your liking). The function is evaluated at the vector x_{new} (equalto x by default).
B. Apply your function to the Boeing stock closing prices from 1/01/2018 to 6/01/2019 - seethe `BA.csv` file, which was downloaded from here (some dates are missing for some unknown reason). Plot the data and overlay the fitted curve for a few choices of bandwidth (identified in a legend).
C. Choose the bandwidth by 10-fold cross-validation.

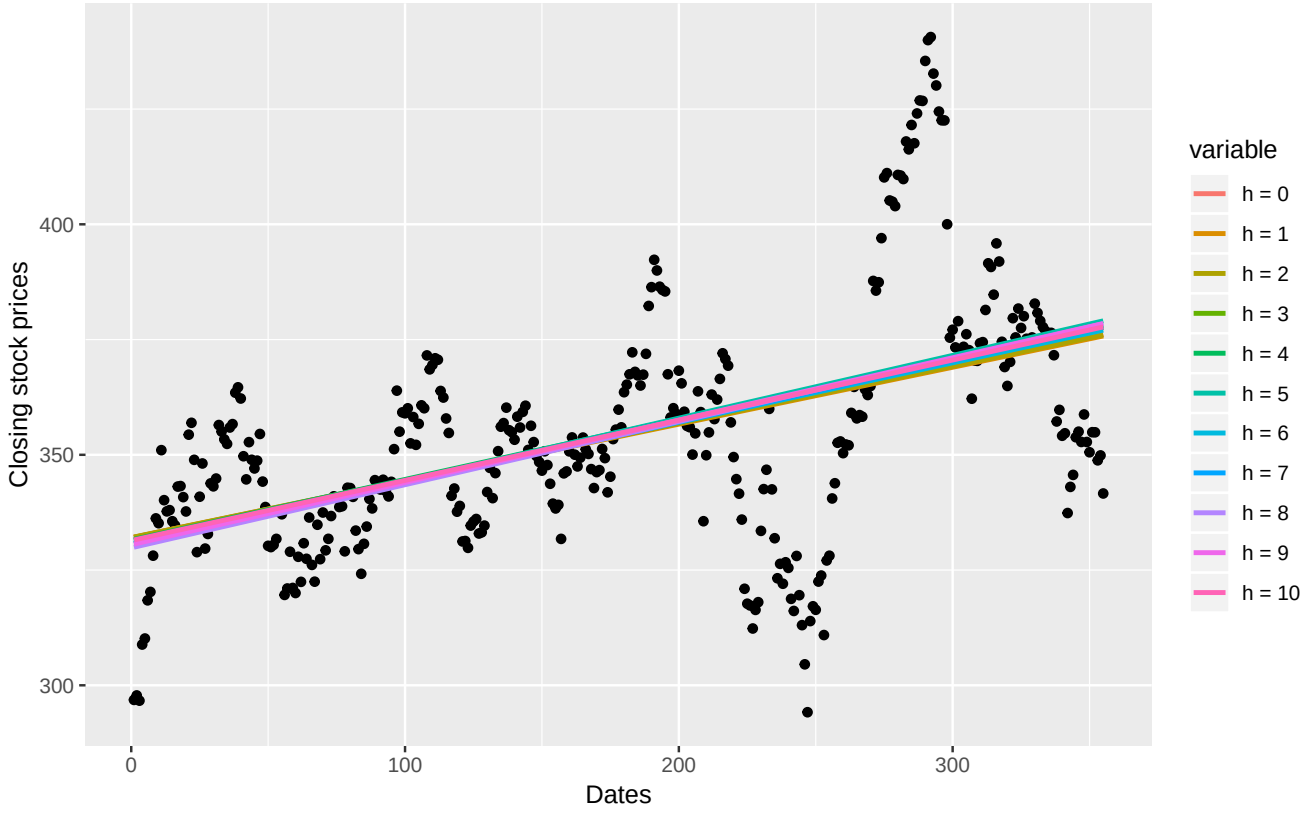
Part A: `localAbsLinearRegression( x, y, h, xnew = x)` function

```
# A: Function
localAbsLinearRegression <- function( x, y, h, xnew) {
  dat <- data.frame(x,y)
  # Local Average vector
  yhat <- numeric(length(y))
  # Computing Local Averages
  for (i in 1:length(x)) {
    temp <- dat %>%
      filter(abs(x - i) <= h ) %>%
      summarise(localAve = mean(y))
    yhat[i] <- temp[[1]]
  }
  # Local Absolute Linear Regression model
  fit <- rq(yhat ~ x)
  # Model evaluated at xnew
  xnew <- xnew
  pred <- predict(fit, data.frame(x = xnew))
  return(pred)
}
```

Part B: Applying the function on the BA dataset

```r
# B: Apply function on Boeing stock prices
data <- read.csv('BA.csv')
# Vector of Closing prices
prices <- data$Close
# Vector of Dates: sequence of 1,2,...
dates <- c(1:length(prices))

# Data.frame of predictions on different bandwidth h
predictions <- data.frame(dates, prices)
# h = 0,1,2,3,4,5,6,7,8,9,10
h <- seq(0, 10, by = 1)
# Running predictions on each bandwidth h
for (j in h) {
  # current bandwidth h
  temp <- localAbsLinearRegression(dates, prices, j, dates)
  predictions <- cbind(predictions, temp)
}
# Renaming columns of predictions
colnames(predictions) <- c('dates', 'prices', 'h = 0', 'h = 1',
                           'h = 2', 'h = 3', 'h = 4', 'h = 5',
                           'h = 6', 'h = 7', 'h = 8', 'h = 9', 'h = 10')
pred_melted <- melt(predictions[,-2], id = 'dates')
ggplot(data = predictions, aes(x = dates, y = prices), lwd = 2) +
  geom_point() +
  xlab('Dates') + ylab('Closing stock prices') +
  geom_line(data = pred_melted, aes(x = dates, y = value, col = variable), lwd = 1)
```



Part C: 10-fold Cross-Validation

```
# C: 10-fold Cross-Validation
# Shuffling the data
set.seed(185)
dat <- data.frame(dates, prices)
dat_random <- dat[sample(nrow(dat)),]
# Creating 10 equally sized folds
folds <- cut(seq(1,nrow(dat)), breaks=10, labels=FALSE)
# Different bandwidth h
h <- seq(0, 10, by = 1)
# Errors matrix to store errors for each bandwidth h
errors <- matrix(ncol = length(h), nrow = 10)
# Perform 10 fold cross validation
for(i in 1:10) {
  # Partitioning into train and test sets
  idx <- which(folds==i,arr.ind=TRUE)
  test <- dat_random[idx, ]
  train <- dat_random[-idx, ]
  # Running predictions and associated errors for each bandwidth h
  for (j in h) {
    h_j <- j
    j <- match(j, h)
    temp <- localAbsLinearRegression(train$dates, train$prices, h_j, test$dates)
    errors[i,j] <- sum(abs(temp - test$prices))
  }
}

print('Average Errors for each bandwidth h:')
```

```
## [1] "Average Errors for each bandwidth h:"
```

```
data.frame(bandwidth_h = h, average_errors = colMeans(errors))
```

```
##    bandwidth_h average_errors
## 1            0       696.8871
## 2            1       688.6502
## 3            2       692.3843
## 4            3       698.1904
## 5            4       699.0272
## 6            5       697.4861
## 7            6       695.9208
## 8            7       693.9861
## 9            8       692.5172
## 10           9       692.4759
## 11          10       690.9762
```

```
k <- which.min(colMeans(errors))[[1]]
```

From the table, after doing 10-fold Cross-Validation, $h = 1$ returns a model with the least average errors, and thus should be chosen.
