STAT 206 Lab 7_Lihua_Xu

Due Monday, November 20, 5:00 PM

General instructions for labs: You are encouraged to work in pairs to complete the lab. Labs must be completed as an R Markdown file. Be sure to include your lab partner (if you have one) and your own name in the file. Give the commands to answer each question in its own code block, which will also produce plots that will be automatically embedded in the output file. Each answer must be supported by written statements as well as any code used.

Agenda: Debugging and testing

Testing for Outliers

Identifying outliers in data is an important part of statistical analyses. One simple rule of thumb (due to John Tukey) for finding outliers is based on the quartiles of the data: the first quartile Q_1 is the value $\geq 1/4$ of the data, the second quartile Q_2 or the median is the value $\geq 1/2$ of the data, and the third quartile Q_3 is the value $\geq 3/4$ of the data. The interquartile range, IQR, is $Q_3 - Q_1$. Tukey's rule says that the outliers are values more than 1.5 times the interquartile range from the quartiles – either below $Q_1 - 1.5IQR$, or above $Q_3 + 1.5IQR$. Consider the data values

```
x <- c(2.2, 7.8, -4.4, 0.0, -1.2, 3.9, 4.9, 2.0, -5.7, -7.9, -4.9,  28.7,  4.9)
```

We will use these as part of writing a function to identify outliers according to Tukey's rule. Our function will be called `tukey.outlier`, and will take in a data vector, and return a Boolean vector, `TRUE` for the outlier observations and `FALSE` elsewhere.

1. Calculate the first quartile, the third quartile, and the inter-quartile range of `x`. Some built-in R functions calculate these; you cannot use them, but you could use other functions, like `sort` and `quantile`.

```
#The first and third quantile are as following:
quantile(x, probs = c(0.25,0.75))
```

```
##  25%  75%
## -4.4  4.9
```

```
First_third <- quantile(x, probs = c(0.25,0.75))
#The inter-quartile range of "x" is:
4.9-(-4.4)
```

```
## [1] 9.3
```

2. Write a function, `quartiles`, which takes a data vector and returns a vector of three components, the first quartile, the third quartile, and the inter-quartile range. Show that it gives the right answers on `x`. (You do not have to write a formal test for quartiles.)

```
quartiles <- function(v){
 V <- sort(v)
 len <- length(V)
 A <- round((len+1)/4)
 B <- round((len+1)*3/4)
   quantile_fisrt <-  V[A]
   quantile_third <-  V[B]
 inter_quartile_range <- quantile_third-quantile_fisrt
 list_three_data <- list(first_quartile=quantile_fisrt,third_quartile=quantile_third,
```

```
                          inter_quartile_range=inter_quartile_range)
 return(list_three_data)
}

data <- quartiles(x)
data
```

```
## $first_quartile
## [1] -4.4
##
## $third_quartile
## [1] 4.9
##
## $inter_quartile_range
## [1] 9.3
```

```
#It  gives the right answers.
```

3. Which points in `x` are outliers, according to Tukey's rule, if any?

```
Max <- list(MAX=data$third_quartile+1.5*data$inter_quartile_range)
Max
```

```
## $MAX
## [1] 18.85
```

```
Min <- list(MIN=data$first_quartile-1.5*data$inter_quartile_range)
Min
```

```
## $MIN
## [1] -18.35
```

```
#The data should be within the range (-18.35,18.85), so in "x" vector, 28.7 is the outlier.
```

4. Write a function, `test.tukey.outlier`, which tests the function `tukey.outlier` against your answer in the previous question. This function should return `TRUE` if `tukey.outlier` works properly; otherwise, it can either return `FALSE`, or an error message, as you prefer. (You can do the next problem first, if you find that easier.)

```
test.tukey.outlier <- function (x){
  Outlier_point <- 28.7
  new_test <- tukey.outlier(x)
  len <- length(x)
  for (i in 1:len){
    if (new_test$x[i]!=Outlier_point&new_test$Boolean_vextor[i]==FALSE){
      a <- TRUE}
      else if (new_test$x[i]==Outlier_point&new_test$Boolean_vextor[i]==TRUE){
        a <- TRUE}
      else {a <- FALSE
      break}
  }
  return(logic=a)
}
```

5. Write `tukey.outlier`, using your `quartiles` function. The function should take a single data vector, and return a Boolean vector, take in a data vector, and return a Boolean vector, `TRUE` for the outlier observations and `FALSE` elsewhere. Show that it passes `test.tukey.outlier`.

```
tukey.outlier <- function(x){
  data <- quartiles(x)
  Max <- data$third_quartile+1.5*data$inter_quartile_range
  Min <- data$first_quartile-1.5*data$inter_quartile_range
  data_frame <- as.data.frame(x)
  data_frame_new <- transform(data_frame, Boolean_vextor=(x<Min|x>Max))
  return(as.list(data_frame_new))
}

tukey.outlier(x)
```

```
## $x
##  [1]  2.2  7.8 -4.4  0.0 -1.2  3.9  4.9  2.0 -5.7 -7.9 -4.9 28.7  4.9
##
## $Boolean_vextor
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [12]  TRUE FALSE
```

6. Which data values should be outliers in `-x`?

```
tukey.outlier(-x)
```

```
## $x
##  [1]  -2.2  -7.8   4.4   0.0   1.2  -3.9  -4.9  -2.0   5.7   7.9   4.9
## [12] -28.7  -4.9
##
## $Boolean_vextor
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [12]  TRUE FALSE
```

```
#The outlier point in "-x" should be -28.7.
```

7. Which data values should be outliers in `100*x`?

```
tukey.outlier(100*x)
```

```
## $x
##  [1]  220  780 -440    0 -120  390  490  200 -570 -790 -490 2870  490
##
## $Boolean_vextor
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [12]  TRUE FALSE
```

```
#The outlier point in "-x" should be 2870.
```

8. Modify `test.tukey.outlier` to include tests for these cases.

```
#modification for "-x" case
test.tukey.outlier_x <- function (x){
  Outlier_point <- -28.7
  new_test <- tukey.outlier(x)
  len <- length(x)
  for (i in 1:len){
    if (new_test$x[i]!=Outlier_point&new_test$Boolean_vextor[i]==FALSE){
      a <- TRUE}
      else if (new_test$x[i]==Outlier_point&new_test$Boolean_vextor[i]==TRUE){
        a <- TRUE}
      else {a <- FALSE
```

```
      break}
  }
  return(logic=a)
}

#modification for "100*x" case
test.tukey.outlier_100x <- function (x){
  Outlier_point <- 2870
  new_test <- tukey.outlier(x)
  len <- length(x)
  for (i in 1:len){
    if (new_test$x[i]!=Outlier_point&new_test$Boolean_vextor[i]==FALSE){
      a <- TRUE}
      else if (new_test$x[i]==Outlier_point&new_test$Boolean_vextor[i]==TRUE){
        a <- TRUE}
      else {a <- FALSE
      break}
  }
  return(logic=a)
}
```

9. Show that your `tukey.outlier` function passes the new set of tests, or modify it until it does.

```
test.tukey.outlier_x(-x)
```

```
## [1] TRUE
```

```
test.tukey.outlier_100x(100*x)
```

```
## [1] TRUE
```

```
#It passes the new set of tests
```

10. According to Tukey's rule, which points in the next vector are outliers? What is the output of your function? If they differ, explain why.

```
y <- c(11.0, 14.0, 3.5, 52.5, 21.5, 12.7, 16.7, 11.7, 10.8, -9.2, 12.3, 13.8, 11.1)
```

```
data <- quartiles(y)
Max <- list(MAX=data$third_quartile+1.5*data$inter_quartile_range)
Max
```

```
## $MAX
## [1] 18.5
```

```
Min <- list(MIN=data$first_quartile-1.5*data$inter_quartile_range)
Min
```

```
## $MIN
## [1] 6.5
```

```
#The outliers should be 3.5, 52.5, 21.5 and -9.2.
```

```
tukey.outlier(y)
```

```
## $x
##  [1] 11.0 14.0  3.5 52.5 21.5 12.7 16.7 11.7 10.8 -9.2 12.3 13.8 11.1
##
## $Boolean_vextor
```

```
##  [1] FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE
## [12] FALSE FALSE
#They are the same.
```