STAT 206 Lab 7

Xin Feng(Vanessa)

11/16/2019

Due Monday, November 18, 5:00 PM

General instructions for labs: Labs must be completed as a pdf 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

```r
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`.

```r
qt_x <- quantile(x)
qt_1 <- qt_x[[2]]
qt_3 <- qt_x[[4]]
iqr <- qt_3-qt_1
cat("The first quartile =", qt_1, '\n')
```

```
## The first quartile = -4.4
```

```r
cat("The third quartile =", qt_3, '\n')
```

```
## The third quartile = 4.9
```

```r
cat("The interquartile range of x =", iqr, '\n')
```

```
## The interquartile range of x = 9.3
```

```r
# IQR(x)
```

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.)

```r
quartiles <- function(x){
  qt <- quantile(x)
  qt_1 <- qt[[2]]
  qt_3 <- qt[[4]]
  return(c(qt_1, qt_3, qt_3-qt_1))
}
quartiles(x)
```

```
## [1] -4.4  4.9  9.3
```

```r
# The answer of testing given x is right.
```

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

```r
tukey.ot <- function(x){
  qt <- quartiles(x)
  rg_max <- qt[2] + 1.5*qt[3]
  rg_min <- qt[1] - 1.5*qt[3]
  index <- which(x>rg_max | x<rg_min)
  data <- x[index]
  if(!length(data)) print("There isn't outlier in the vector.")
  else return(cbind(index, data))
}
tukey.ot(x)
```

```
##      index data
## [1,]    12 28.7
```

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.)

```r
library(testthat)
test.tukey.outlier <- function(x){
  expect_type(x, "double")
  expect_type(tukey.outlier(x), "logical")

  result <- tukey.outlier(x)
  qt <- quartiles(x)
  rg_max <- qt[2] + 1.5*qt[3]
  rg_min <- qt[1] - 1.5*qt[3]
  judge = 0
  for(i in 1:length(x)){
    if(x[i] < rg_min | x[i] > rg_max){
      if(result[i]==FALSE){
        judge = 1
        cat("Error: function tukey.outlier() is wrong. The ", i, "-th data
            in x is outlier.")
      }
    }else{
      if(result[i]==TRUE){
        judge = 1
        cat("Error: function tukey.outlier() is wrong. The ", i, "-th data
            in x is not outlier.")
      }
    }
  }
  ifelse(!judge, return(TRUE), return(FALSE))
}
# Test: initial vector x
try(test.tukey.outlier(x))
```

```
## Error in tukey.outlier(x) : could not find function "tukey.outlier"
```

```
# Test: if vector x contains character item.
try(test.tukey.outlier(c("a",1,4,5,7,9)))
```

```
## Error : `x` has type `character`, not `double`.
```

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){
  qt <- quartiles(x)
  rg_max <- qt[2] + 1.5*qt[3]
  rg_min <- qt[1] - 1.5*qt[3]
  judge <- vector(mode="logical", length=length(x))
  judge[which(x>rg_max |x<rg_min)]=TRUE
  return(judge)
}
result <- tukey.outlier(x)
result
```

```
##  [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.ot(-x)
```

```
##      index  data
## [1,]    12 -28.7
```

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

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [12]  TRUE FALSE
```

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

```r
tukey.ot(100*x)
```

```
##      index data
## [1,]    12 2870
```

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

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [12]  TRUE FALSE
```

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

```r
test.tukey.outlier <- function(x){
  expect_type(x, "double")
  expect_type(tukey.outlier(x), "logical")

  judge.x <- function(x){
    result <- tukey.outlier(x)
    qt <- quartiles(x)
    rg_max <- qt[2] + 1.5*qt[3]
    rg_min <- qt[1] - 1.5*qt[3]
    judge = 1
    for(i in 1:length(x)){
      if(x[i] < rg_min | x[i] > rg_max){
        if(result[i]==FALSE){
          judge = 0
          cat("Error: function tukey.outlier() is wrong. The ", i, "-th data
              in",x," is outlier.")
        }
      }else{
        if(result[i]==TRUE){
          judge = 0
          cat("Error: function tukey.outlier() is wrong. The ", i, "-th data
              in ",x," is not outlier.")
        }
      }
    }
    ifelse(judge, return(TRUE), return(FALSE))
  }
  judge <- judge.x(x) + judge.x(-x) + judge.x(100*x)
  ifelse(judge, return(TRUE), return(FALSE))
}
```

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

```r
tukey.outlier <- function(x){
  expect_type(x, "double")
  qt <- quartiles(x)
  rg_max <- qt[2] + 1.5*qt[3]
  rg_min <- qt[1] - 1.5*qt[3]
  judge <- vector(mode="logical", length=length(x))
  judge[which(x>rg_max |x<rg_min)]=TRUE
  return(judge)
}
tukey.outlier(x)
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [12]  TRUE FALSE
```

```r
try(test.tukey.outlier(x))
```

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

```r
# Test, input character
try(tukey.outlier(c('a',1,3,4,5,6)))
```

```
## Error : `x` has type `character`, not `double`.
```

```r
try(test.tukey.outlier(c('a',1,2,3,4)))
```

```
## Error : `x` has type `character`, not `double`.
```

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.

```r
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)
```

```r
tukey.ot(y)
```

```
##      index data
## [1,]     3  3.5
## [2,]     4 52.5
## [3,]     5 21.5
## [4,]    10 -9.2
```

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

```
##  [1] FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE
## [12] FALSE FALSE
```

```r
# The output of function tukey.outlier is a Boolean vector, `TRUE` for the
# outlier observations and `FALSE` for the data which is not outlier.
# Due to the different output format, they seem difference, in fact, the
# results are same.
```