Solutions: Introduction to Outliers

STA 325: Lab 2, Fall 2018

Today's agenda: finding outliers

Programming partners: You should have a programming partner for each lab, and you should switch off who is programming, and use each other for help. We will spend about 30–50 minutes per week on lab exercises and you will be expected to bring your laptops to class to work on these exercises in class. Myself and the TA will be in class to help you.

Background

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 $Q_3 + 1.5IQR$.

In this lab, we will consider the following data

```
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.

Lab Tasks

1. (5) 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`.

```
(Q1 <- quantile(x,0.25)) #first quartile
```

```
##  25%
## -4.4
```

```
(Q3 <- quantile(x,0.75)) #third quartile
```

```
## 75%
## 4.9
```

```
(iqr.x <- Q3-Q1)  #inter-quartile range
```

```
## 75%
## 9.3
```

2. (10) 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(x) {
  q1<-quantile(x,0.25,names=FALSE)
  q3<-quantile(x,0.75,names=FALSE)
  quartiles <- c(first=q1,third=q3,iqr=q3-q1)
  return(quartiles)
}
```

Let's check that our function applied to the vector x returns the answer from task 1. The code below illustrates that the first, third, and iqr of our new function matches the computations from task 1.

```
quartiles(x)
```

```
## first third   iqr
##  -4.4   4.9   9.3
```

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

Recall that 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 $Q_3 + 1.5IQR$.

We can see below that the only value that is an outlier by Tukey's rule is 28.7.

```
Q1 - 1.5*iqr.x
```

```
##    25%
## -18.35
```

```
Q3 + 1.5*iqr.x
```

```
##   75%
## 18.85
```

```
x[x>=18.5]
```

```
## [1] 28.7
```

```
x[x <= -18.5]
```

```
## numeric(0)
```

4. (20) 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`.

```
# Input: data
# Output: outliers according to Tukey's rule
tukey.outlier <- function(x) {
  quartiles <- quartiles(x)
  lower.limit <- quartiles[1]-1.5*quartiles[3]
  upper.limit <- quartiles[2]+1.5*quartiles[3]
  outliers <- ((x < lower.limit) | (x > upper.limit))
  return(outliers)
}
tukey.outlier(x)
```

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

5. (20) Write a function, `test.tukey.outlier`, which tests the function `tukey.outlier` against your answer. This function should return `TRUE` if `tukey.outlier` works properly; otherwise, it can either return `FALSE`, or an error message, as you prefer.

```
# Input: Nothing
# Output: Boolean
test.tukey.outlier <- function() {
  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)
  x.pattern <- rep(FALSE,length(x)); x.pattern[12] <- TRUE
  stopifnot(all(tukey.outlier(x) == x.pattern))
  return(TRUE)
```

```
}
test.tukey.outlier()
```

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

Remark: since `tukey.outlier(x)` and `x.pattern` are both vectors, `==` will compare them element by element, giving us yet another Boolean vector. We want to summarize this in a single TRUE/FALSE value, hence we use the `all` command.

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

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

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

The same value as before, 28.7, is an outlier to the symmetry of x and -x.

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

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

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

Multiplying all the values by 100 also multiplies the quartiles and the IQR by 100, so once again only the next to last value is an outlier in `100*x`.

8. (10) Let's modify `test.tukey.outlier` to includes two test cases for tasks 6 and 7.

```
# Inputs: none
# Output: TRUE if all tests pass, else stops with an error
test.tukey.outlier <- function() {
  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)
  x.pattern <- rep(FALSE,length(x)); x.pattern[12] <- TRUE
  stopifnot(all(tukey.outlier(x) == x.pattern))
  stopifnot(all(tukey.outlier(-x) == tukey.outlier(x)))
  stopifnot(all(tukey.outlier(100*x) == tukey.outlier(x)))
  return(TRUE)
}
test.tukey.outlier()
```

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

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

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

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

10. (15) According to Tukey's rule, which points in the next vector y 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)
```

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

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

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

```
## [1]  3  4  5 10
```

We can see that our simple santity checks on our function have been passed for both data vectors that we have used, so we should feel comfortable using this code again in other exercises.