STAT 206 Homework–Lihua Xu

Due Monday, October 9, 5:00 PM

General instructions for homework: Homework must be completed as an R Markdown file. Be sure to include your 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. (Examining your various objects in the "Environment" section of RStudio is insufficient – you must use scripted commands.)

1. The data set at [http://www.stats.uwo.ca/faculty/braun/data/rnf6080.dat] records hourly rainfall at a certain location in Canada, every day from 1960 to 1980.

a. First, we need to load the data set into R using the command `read.table()`. Use the help function to learn what arguments this function takes. Once you have the necessary input, load the data set into R and make it a data frame called `rain.df`.

```
hourly_rain.matrix <- read.table("http://www.stats.uwo.ca/faculty/braun/data/rnf6080.dat")
rain.df <- data.frame(hourly_rain.matrix)
```

b. How many rows and columns does `rain.df` have? (If there are not 5070 rows and 27 columns, something is wrong; check the previous part to see what might have gone wrong in the previous part.)

```
# There are 5070 rows and 27 columns.
```

c. What are the names of the columns of `rain.df`?

```
# The names of the columns of `rain.df` are V1, V2,V3,V4,V5......V24,V25,V26,V27, separately.
```

d. What is the value of row 5, column 7 of `rain.df`?

```
#The value for row 5:
#There are only three columns for row 5 have values.
#They are column V1, value 60. Column V2, value 4. Column V3, value 5.

#The value for row 7:
#There are only three columns for row 7 have values.
#They are column V1, value 60. Column V2, value 4. Column V3, value 7.
```

e. Display the second row of `rain.df` in its entirety.

```
rain.df[2,]
```

```
##   V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20
## 2 60  4  2  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   0
##   V21 V22 V23 V24 V25 V26 V27
## 2   0   0   0   0   0   0   0
```

f. Explain what this command does:

`names(rain.df) <- c("year","month","day",seq(0,23))`

by running it on your data and examining the object. (You may find the display functions `head()` and `tail()` useful here.) Is it clear now what the last 24 columns represent?

```
names(rain.df) <- c("year","month","day",seq(0,23))
names(rain.df)
```

```
##  [1] "year"  "month" "day"   "0"     "1"     "2"     "3"     "4"
##  [9] "5"     "6"     "7"     "8"     "9"     "10"    "11"    "12"
## [17] "13"    "14"    "15"    "16"    "17"    "18"    "19"    "20"
## [25] "21"    "22"    "23"
```

```r
head(rain.df)
```

```
##   year month day 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
## 1   60     4   1 0 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0
## 2   60     4   2 0 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0
## 3   60     4   3 0 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0
## 4   60     4   4 0 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0
## 5   60     4   5 0 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0
## 6   60     4   6 0 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0
##   22 23
## 1  0  0
## 2  0  0
## 3  0  0
## 4  0  0
## 5  0  0
## 6  0  0
```

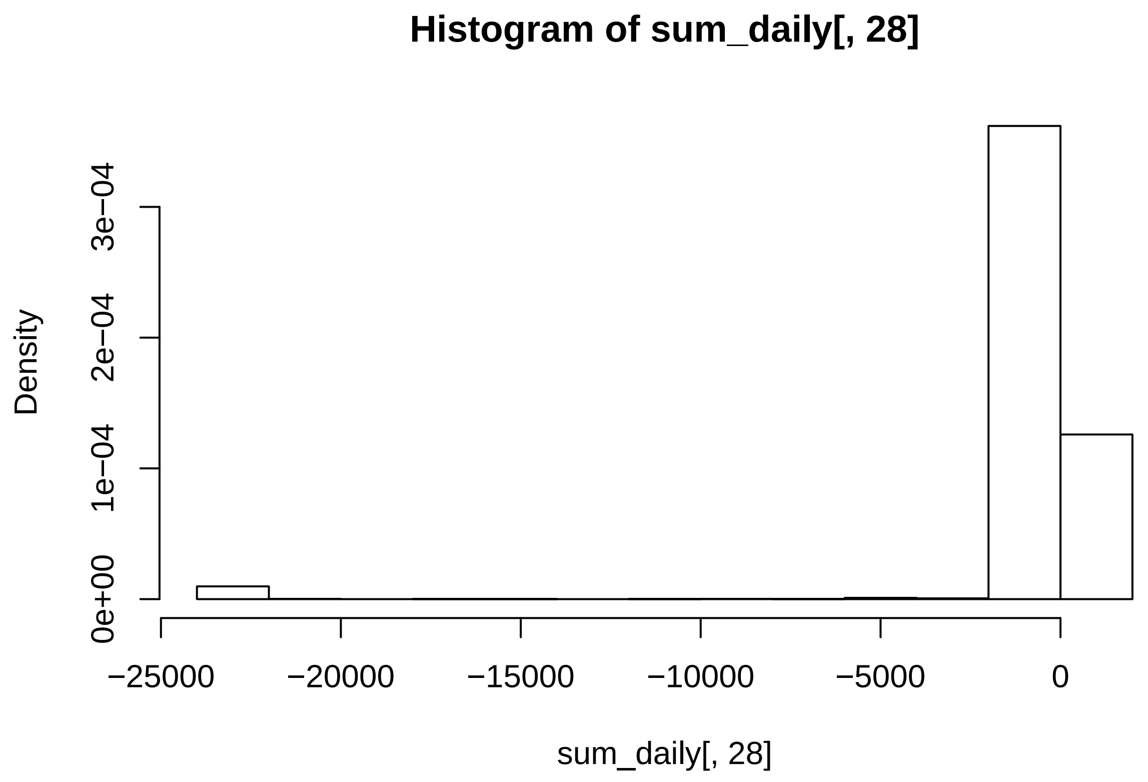
```r
tail(rain.df)
```

```
##      year month day 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
## 5065   80    11  25 0 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0
## 5066   80    11  26 0 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0
## 5067   80    11  27 0 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0
## 5068   80    11  28 0 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0
## 5069   80    11  29 0 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0
## 5070   80    11  30 0 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0
##      21 22 23
## 5065  0  0  0
## 5066  0  0  0
## 5067  0  0  0
## 5068  0  0  0
## 5069  0  0  0
## 5070  0  0  0
```

```r
#It becomes very clear after labeling the first three columes.
#The remaining 24 columes are related to the 24 different hours per day.
```

g. Create a new column in the data frame called `daily`, which is the sum of the rightmost 24 columns. With this column, create a histogram of the values in this column, which are supposed to be daily rainfall values. What is wrong with this picture?

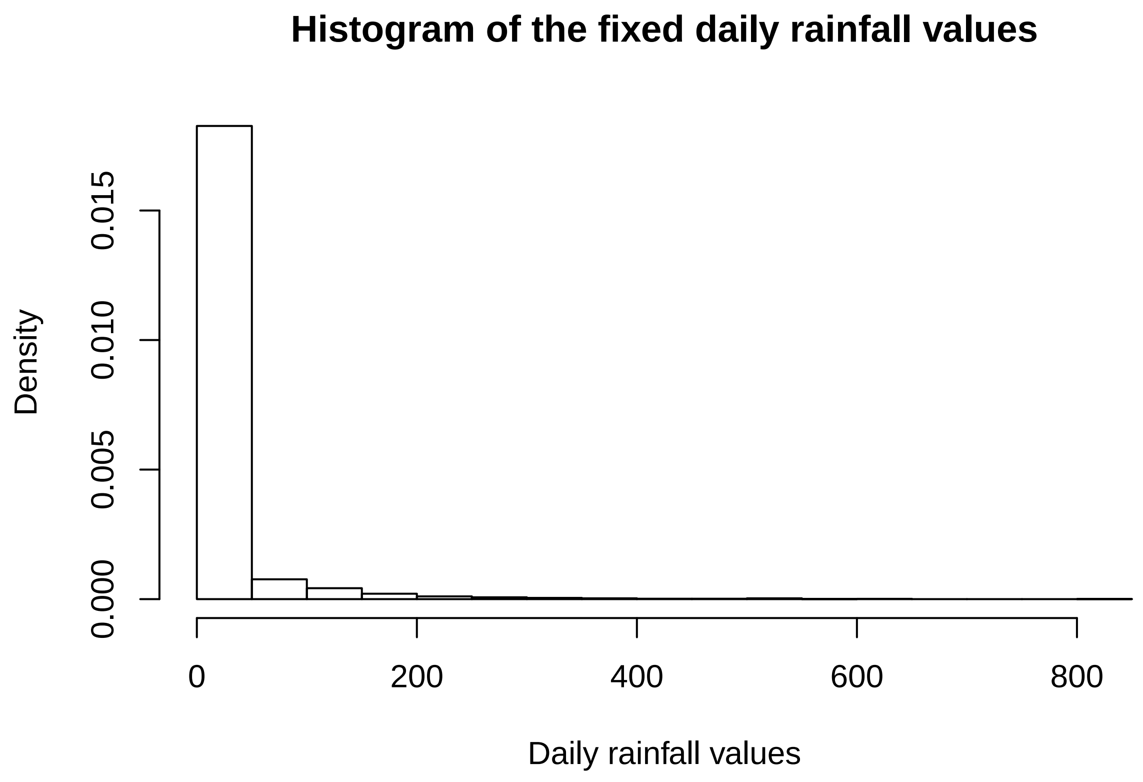
```r
sum_daily<-transform(rain.df,daily=rowSums(rain.df[,4:27]))
hist(sum_daily[,28],freq = FALSE)
```



```
#It appears the negative value for the x axis.
#It is impossible because daily rainfal values could not be negetive values.
```

h. Create a new data frame `rain.df.fixed` that takes the original and fixes it for the apparent flaw you have discovered. Having done this, produce a new histogram with the corrected data and explain why this is more reasonable.

```
rain.df.fixed <- rain.df
rain.df.fixed[rain.df.fixed<0] = 0
#Replace all -999 in 'rain.df.fixed' to 0.
sum_daily_new <- transform(rain.df.fixed,daily=rowSums(rain.df.fixed[,4:27]))
hist(sum_daily_new[,28],freq = FALSE,
     main="Histogram of the fixed daily rainfall values",
     xlab="Daily rainfall values")
```



```
#Because the rainfall value should always be larger or equal to 0.
#In the formal dataframe, we see some value equal to -999, which is impossibel.
#So I use the the command to replace negative values to 0.
# This help to make sure there are no negative value in dataframe.
```

2. Syntax and class-typing.

a. For each of the following commands, either explain why they should be errors, or explain the non-erroneous result.

```
vector1 <- c("5", "12", "7", "32")
max(vector1)
sort(vector1)
sum(vector1)
```

```
vector1 <- c("5", "12", "7", "32")
vector1
```

```
## [1] "5"  "12" "7"  "32"
```

```
#This command has no errors.
#Because it shows that make a new vector and in the vector contains four different charaters.

max(vector1)
```

```
## [1] "7"
```

```
#This command has no errors.
#In the vector1, is contains the charaters, not numbers.
#When referring to the maximum of the character in vector1, it is 7.
```

```r
sort(vector1)
```

```
## [1] "12" "32" "5"  "7"
```

```r
#This command has no errors.
#In the vector1, is contains the charaters, not numbers.
#The command sort an R object with character vector within an increasing order.

#sum(vector1)

#This command has an error.
#In the vector1, is contains the charaters, not numbers.
#sum command can only be used in numeric or complex or logical vectors.
```

b. For the next series of commands, either explain their results, or why they should produce errors.

```
vector2 <- c("5",7,12)
vector2[2] + vector2[3]

dataframe3 <- data.frame(z1="5",z2=7,z3=12)
dataframe3[1,2] + dataframe3[1,3]

list4 <- list(z1="6", z2=42, z3="49", z4=126)
list4[[2]]+list4[[4]]
list4[2]+list4[4]
```

```r
vector2 <- c("5",7,12) #first command
vector2
```

```
## [1] "5"  "7"  "12"
```

```r
#vector2[2] + vector2[3] #second command

#The fist command has no errors. As the new vector2 contains 3 characters.
#The second command has errors, Characters cannot be added.

dataframe3 <- data.frame(z1="5",z2=7,z3=12) #first command
dataframe3[1,2] + dataframe3[1,3] #second command
```

```
## [1] 19
```

```r
#The fist command shows that making a dataframe3 comtains a row with three columns
#The numbers for each column are 5, 7and 12, seperately.
#The second command show that adding the numbers from the first row second and third colums.
#The final adding vlue is 19.

list4 <- list(z1="6", z2=42, z3="49", z4=126) #first command
list4[[2]]+list4[[4]] #second column
```

```
## [1] 168
```

```r
#list4[2]+list4[4] #third column

#The first command shows that making a list with two characters 6, 49 and two numbers 42 and 126.
#The second command shows that adding the two numbers in the list and the result is 168.,
#which is also a number.
#The thisd command has errors.
#Because 'list4[2]' comtains not only the number but also the name related to the number.
```

```
#And the name could not be added.
```

3. Working with functions and operators.

a. The colon operator will create a sequence of integers in order. It is a special case of the function `seq()` which you saw earlier in this assignment. Using the help command `help(seq)` to learn about the function, design an expression that will give you the sequence of numbers from 1 to 10000 in increments of 372. Design another that will give you a sequence between 1 and 10000 that is exactly 50 numbers in length.

```
seq(1,10000,372)
```

```
##  [1]    1  373  745 1117 1489 1861 2233 2605 2977 3349 3721 4093 4465 4837
## [15] 5209 5581 5953 6325 6697 7069 7441 7813 8185 8557 8929 9301 9673
```

```
new_length <- seq(1,10000,200)
length(new_length)
```

```
## [1] 50
```

b. The function `rep()` repeats a vector some number of times. Explain the difference between ‘rep(1:3, times=3) and rep(1:3, each=3).

```
rep(1:3, times=3)
```

```
## [1] 1 2 3 1 2 3 1 2 3
```

```
rep(1:3, each=3)
```

```
## [1] 1 1 1 2 2 2 3 3 3
```

```
#The differences between the two commands are:
#For 'rep(1:3, times=3)'
#Repeating the number order {1,2,3} three times.
#So it gives the result '1 2 3 1 2 3 1 2 3'
#For 'rep(1:3, each=3)'
#Repeating the each of the number in order {1,2,3} three times.
#So it gives the result '1 1 1 2 2 2 3 3 3'.
```