# Shahab Geravesh, Statistical Computing 206 (002), Homework 1

1.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

```
rain.df<-read.table("http://www.faculty.ucr.edu/~jflegal/206/rnf6080.dat", header = FALSE)
```

1.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.)

```
nrow(rain.df)
```

```
## [1] 5070
```

```
ncol(rain.df)
```

```
## [1] 27
```

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

```
colnames(rain.df, do.NULL = TRUE, prefix = "col")
```

```
##  [1] "V1"  "V2"  "V3"  "V4"  "V5"  "V6"  "V7"  "V8"  "V9"  "V10" "V11" "V12"
## [13] "V13" "V14" "V15" "V16" "V17" "V18" "V19" "V20" "V21" "V22" "V23" "V24"
## [25] "V25" "V26" "V27"
```

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

```
rain.df[5,7]
```

```
## [1] 0
```

1.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 V21
## 2 60  4  2  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0
##   V22 V23 V24 V25 V26 V27
## 2   0   0   0   0   0   0
```

1.f Explain what this command does: names(rain.df) <- c("year","month","day",seq(0,23))

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

```
names(rain.df)
```

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

```
head(names(rain.df))
```

```
## [1] "year"  "month" "day"   "0"     "1"     "2"
```

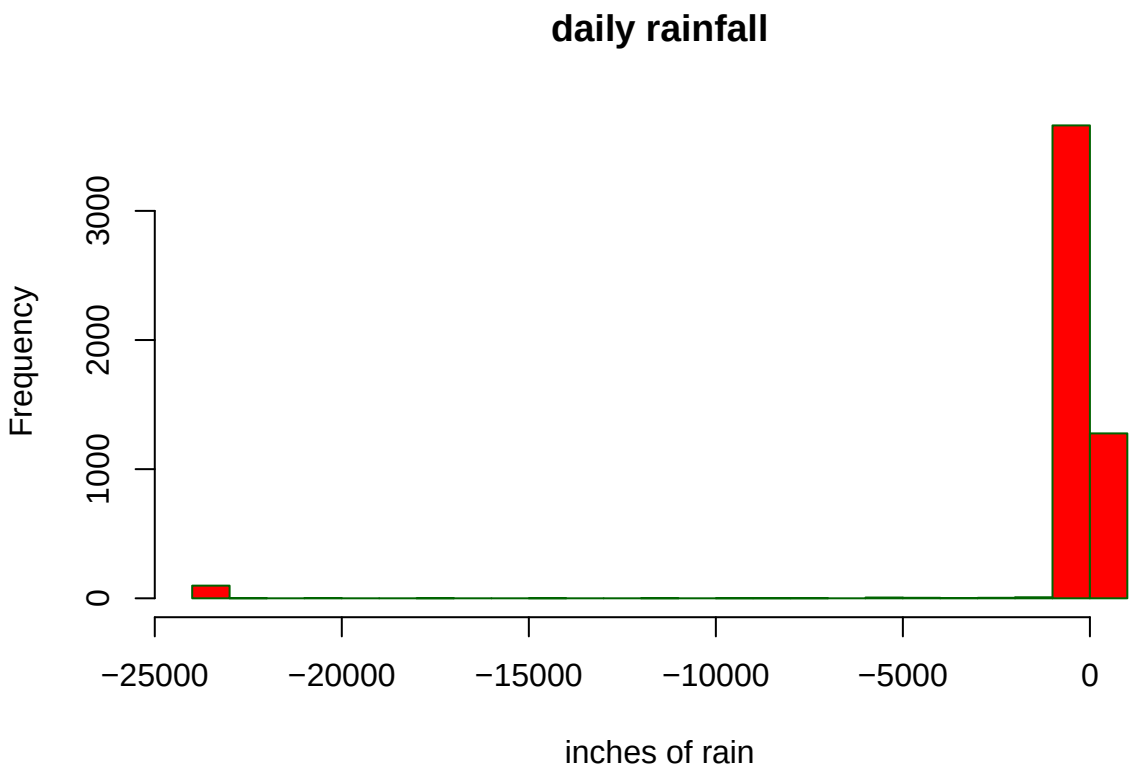
```
tail(names(rain.df))
```

```
## [1] "18" "19" "20" "21" "22" "23"
```

It named the columns as year, month and day and added zero through 23 sequential numbers as the rest of the columns. Yes, they are hours. From hour 0 to hour 23.

1.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?

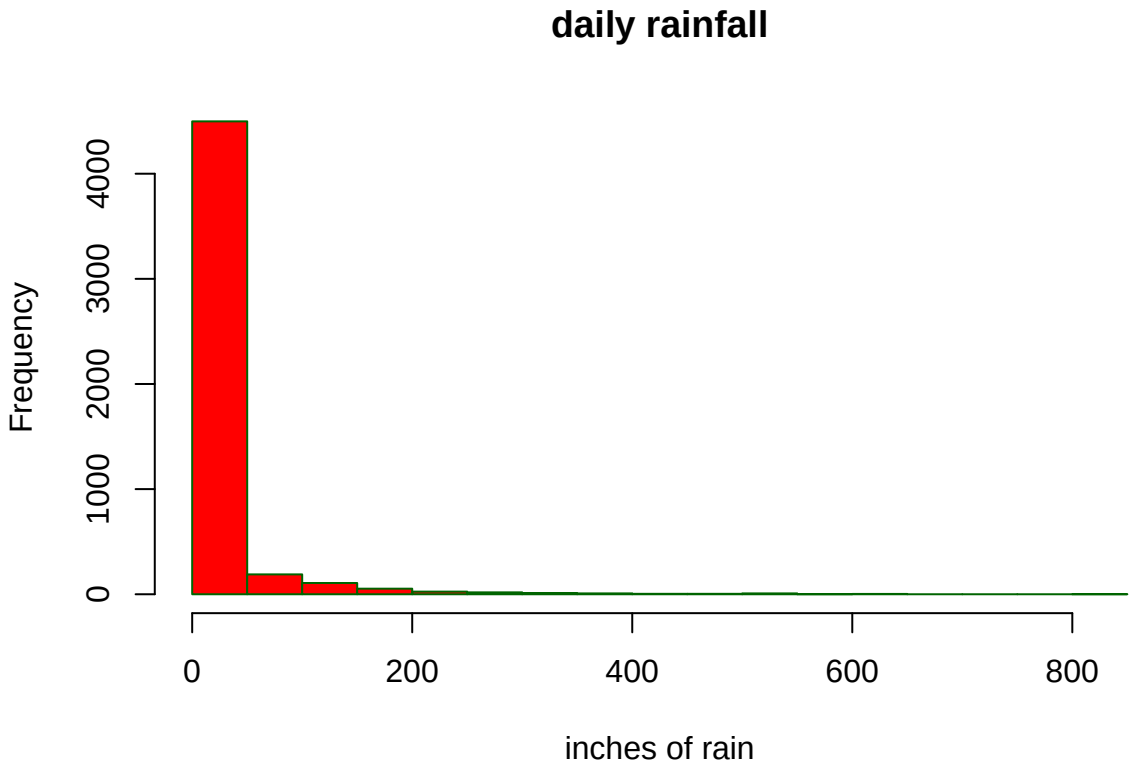
```
rain.df$daily <- apply(rain.df[,c(4:27)], 1, function(x) sum(x))
hist(rain.df[,28], breaks = 25,xlab = "inches of rain" ,col = " red", border = "dark Green", main = "da
```




The problem is that there are some negative values in daily rain amount which is not possible in real life!

1.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
is.na(rain.df.fixed) <- rain.df.fixed< 0
hist(rain.df.fixed[,28], breaks = 25, xlab = "inches of rain" ,col = "red", border = "dark green", main
```




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)
```

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

```
sort(vector1)
```

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

max(): The elements of vector1 are strings since they are all in quotation. The user probably was intending to get the maximum number between all elements, but since they are string, R is treating them as characters, not numbers.

sort(): Since the numbers are in quotation and R is treating them as characters and not numbers, R sorts them based on their first character.

sum(): Since the numbers are in quotation and R is treating them as characters and not numbers, sum() is not meaningful for characters. Sum function is for adding values.

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

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

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

vector2 <- c(“5”,7,12) vector2[2] + vector2[3]

error: non-numeric argument to binary operator. A Vector is a sequence of values, all of the same type. In this case, all integers will be treated as characters that’s why we can’t add up the characters.

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

The answer would be 19. In this case, we have 1 row and three columns in the dataframe. 7 and 12 are integers and 5 is a character.

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

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

The answer would be 168 because they are a list and in list we can have both characters and numbers. In this case 42 and 126 are numbers because [[]] drops both the structure and names and we can add them and the result would be 168.

list4[2]+list4[4]

error: non-numeric argument to binary operator. In this case, since we are using one bracket instead of two brackets the structures and names will stick with values. Therefore, we will get an error.

3. Working with functions and operators. 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 ?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, length.out=50)
```

```
##  [1]     1.0000   205.0612   409.1224   613.1837   817.2449  1021.3061
##  [7]  1225.3673  1429.4286  1633.4898  1837.5510  2041.6122  2245.6735
## [13]  2449.7347  2653.7959  2857.8571  3061.9184  3265.9796  3470.0408
## [19]  3674.1020  3878.1633  4082.2245  4286.2857  4490.3469  4694.4082
## [25]  4898.4694  5102.5306  5306.5918  5510.6531  5714.7143  5918.7755
## [31]  6122.8367  6326.8980  6530.9592  6735.0204  6939.0816  7143.1429
## [37]  7347.2041  7551.2653  7755.3265  7959.3878  8163.4490  8367.5102
## [43]  8571.5714  8775.6327  8979.6939  9183.7551  9387.8163  9591.8776
## [49]  9795.9388 10000.0000
```

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
```

When times=3, the sequence of 1 to 3 gets repeated 3 times.

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

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

When each=3, each number in 1 to 3 gets repeated 3 times.