STA 602: Homework 1

Zach Calhoun

1/7/2022

1. Working with data

a. Load the data.

```
# This command will read the data into the dataframe.
rain.df <- read.table("data/rnf6080.dat")
```

I used the command `read.table` to load this data set. There are no headers provided, so there are no extra parameters required, except for the file path.

b. How many rows and columns?

```
# Print out the number of rows and columns.
cat("This has", nrow(rain.df),"rows and", ncol(rain.df),"columns.")
```

```
## This has 5070 rows and 27 columns.
```

I can print the number of rows using the `cat` command with `nrow` and `ncol`, or, since I am in R studio, I can look at the environmental variables to see the dimensionality of the data.

c. How to get the column names?

```
# Print out the column names.
colnames(rain.df)
```

```
##  [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"
```

I can use `colnames` to get the column names. In the case of this data set, there are no headers, so the default headers are provided, using "V#" as the format (where # refers to the column index).

d. Command to get the value at row 2, column 4?

```
# This will select the second row, fourth column.
rain.df[2,4]
```

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

I can simply index using `[2,4]` to get the value at row 2, column 4. The value is 0.

e. Command to display the whole second row.

```
# What is the second row?
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
```

I can just index on the entire second row using `[2,]` on the data frame. The content of the row only has non-zero values in the first three columns, which indicates that there was zero rainfall for that day.

f. What does the following command do?

```
# This will add the column names to the data frame.
names(rain.df) <- c("year", "month","day", seq(0,23))
```

This command adds the column names to the data frame. In this case, we have the year, month, day as the first three column names, then the hour indicated by the integer values between 0 and 23.

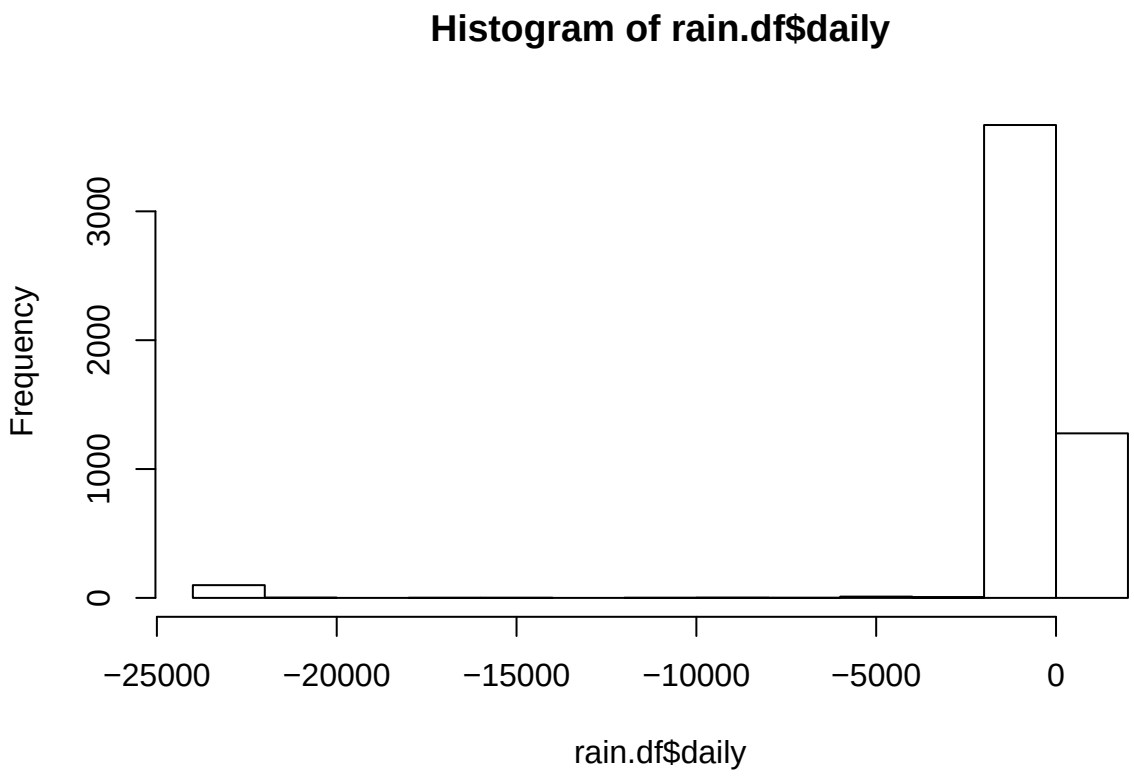
g. Create a new column called "daily."

```
# Calculate the sum of the rows, omitting the first three columns in
# the summation.
rain.df$daily <- rowSums(rain.df[c(-1,-2,-3)])
```

h. Create a histogram.

We can create a histogram using the `hist` function.

```
# Create a histogram of the rainfall.
hist(rain.df$daily)
```



i. Explain why this cannot be right.

There are negative values in this histogram, which doesn't make sense. We should not have negative rainfall amounts.

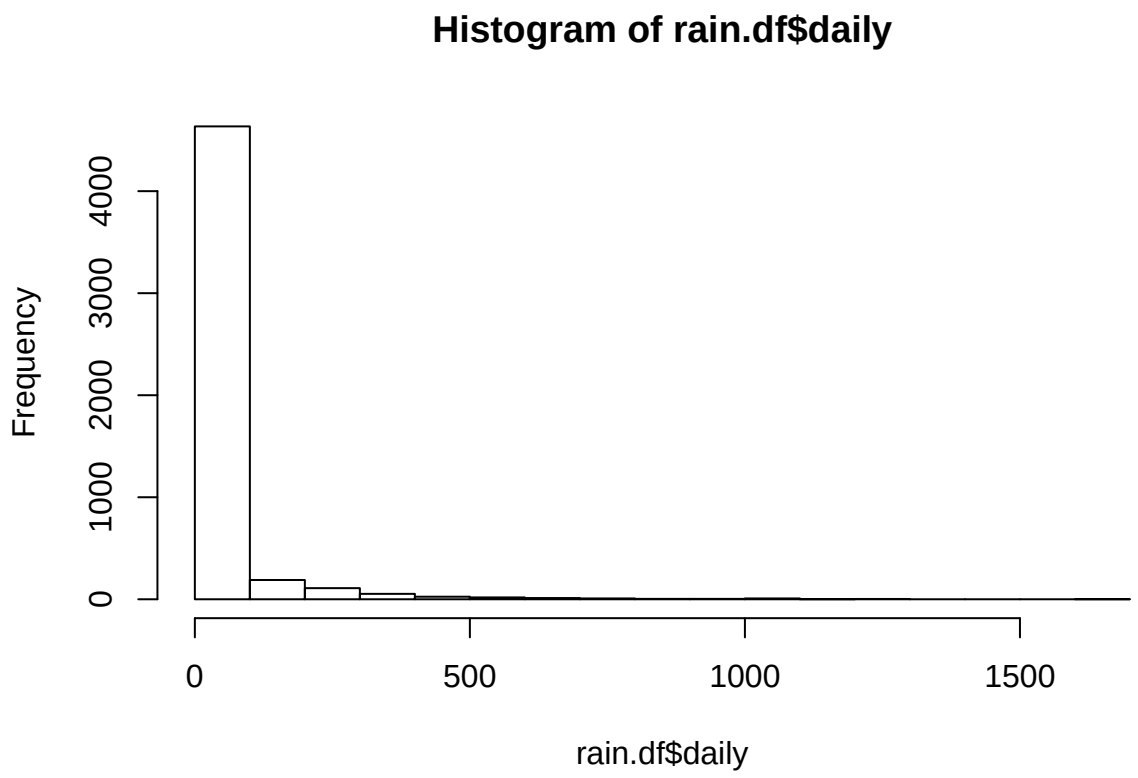
j. Fix the data issue.

We can fix the data issue by changing any negative values to equal zero.

```
# Set values to zero
rain.df[rain.df < 0] <- 0
# Recalculate the sum for each row.
rain.df$daily <- rowSums(rain.df[c(-1,-2,-3)])
```

k. Create the corrected histogram.

```
# Now, create the histogram again.
hist(rain.df$daily)
```



This is a much more reasonable histogram. All of the values are above zero, and most days have low rainfall amounts, which is to be expected.

2. Data types

a. For each of the following commands. . . explain the results, or the resulting error.

```
# This creates an array of characters.
x <- c("5","12","7")
```

This will execute fine, just creating an array of characters.

```
max(x)
```

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

This function looks at the first character to calculate the maximum value, since this maximum value is basically sorting in alphabetical order, then selecting the maximum value when sorted.

```
sort(x)
```

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

As alluded to in the previous response, R sorts the values alphabetically, which is the result provided here. Since "1" is the first alphabetically, "12" comes first.

```
# Commenting out the below code because it gets an error.
# sum(x)
```

In this case, we get an error, because there is no logical way to sum up this vector of characters.

b. Explain their results or why they should produce errors.

```
y <- c("5",7, 12)
```

This command executes fine, but casts the integer values as strings.

```
# Attempting to add these yields and error, so I am going to comment it out.
# y[2]+y[3]
```

This creates an error, because again, we are adding two strings together (since the vector casts the second and third values as strings, since the first value in the vector is a string).

c. For the next two commands...

```
z <- data.frame(z1="5", z2=7, z3=12)
```

This executes fine, too, but instead of casting all of the values as strings, this command will keep the data types as typed, since we are creating a data frame, rather than a vector.

```
z[1,2] + z[1,3]
```

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

Since we have a data frame, the data types are preserved, meaning addition will work as expected with the values in the second and third columns.

3.

a. The point of reproducible code is the same as reproducible science! If our results cannot be reproduced, then our results are meaningless or invalid. Thus, we should strive to have easily understood and repeatable code.

b. Making reproducible code is important to me because I work with others in my lab, and I want them to be able to use the code I write. Thus, if I make my code reproducible, I can better work with others in my field.

c. This assignment was a 4 for me. I have not used R in a while, but I have used Python a lot recently, so I had a good idea of what to google to figure out what functions to use, so I was able to quickly find the answers I needed to complete the assignment.