Dial-Jackson-homework1

Jackson Dial

2023-01-11

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Working with data

Q1

```
rain.df <- read.table("rnf6080.dat.txt")
# rain.df1 <- read.table("HW1_data/rnf6080.dat.txt", header = TRUE) #I think the header statement messe
# rain.df2 <- read.table(url("homeworks/homework-1/data/rnf6080.dat"))
```

A

I used the *read.table()* function to read in the data.

B

```
dim(rain.df)
```

```
## [1] 5070   27
```

There are 5070 rows and 27 columns. I used the *dim()* function to find the length and width of the data frame.

C

```
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"
```

To get the names of the columns, I used the *colnames()* function, and they are named with 'V' followed by the index of the column (1:27).

D

```
rain.df[2,4]
```

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

I indexed the value by specifying the row number and column number inside brackets.

E

```
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 uses the same function as before, the brackets, but left the column argument blank as to indicate I want every column, while specifying the row.

F

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

This function replaces the column names with the strings "year", "month", and "day", followed by the sequence of numbers 0:23.

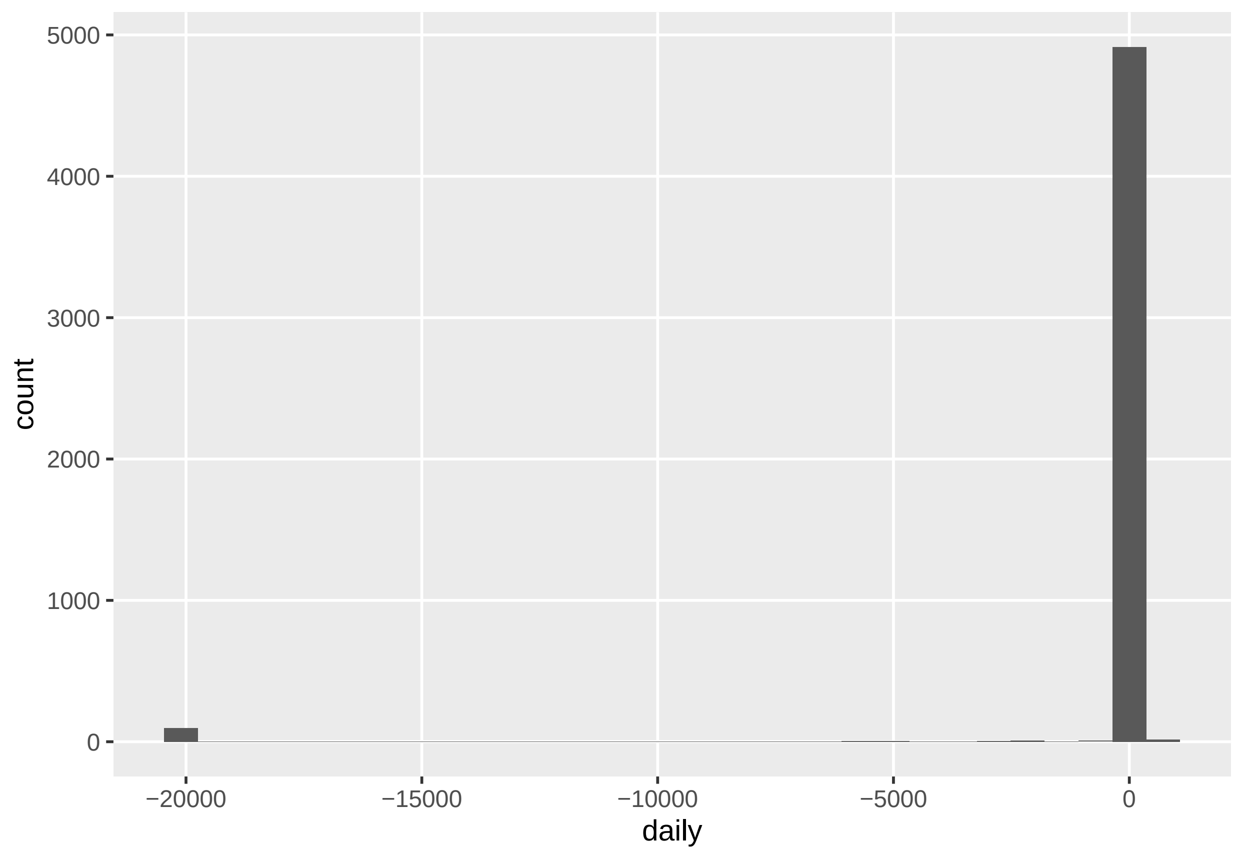
G

```
rain.df <- rain.df %>%
  mutate(
    daily = rowSums(across(4:23))
  )
```

H

```
ggplot(rain.df, aes(x = daily))+
  geom_histogram()+
  theme(panel.grid.minor = element_blank())
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



I

The histogram cannot be correct because there are a few observations with negative values for rainfall, which is not possible.

J

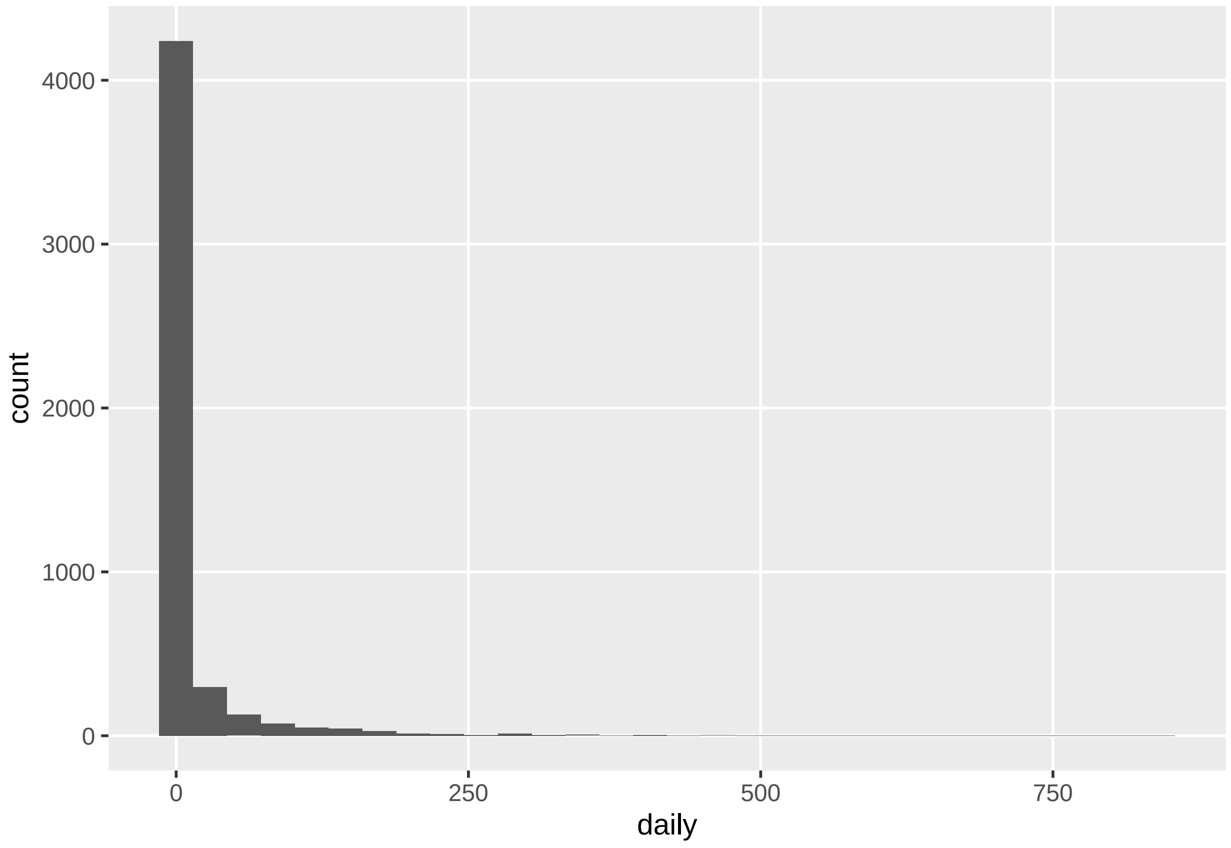
```
rain.df_cleaned <- rain.df %>% filter(daily >= 0)
```

I used the *filter()* command.

K

```
ggplot(rain.df_cleaned, aes(x = daily))+
  geom_histogram()+
  theme(panel.grid.minor = element_blank())
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



This is more reasonable than the other histogram because it only includes positive or 0 values.

Q2

A

The *x <- c("5","12","7")* command works because we are just creating a vector of the 3 arguments in the concatenate function.

The *max()* function works but does not give the correct output because the values in the vector are characters, and thus mathematical operations are not correctly applied to them. The 7 is output because it is the last value in the vector.

The *sort()* function does sort the values, but again they are as characters and thus sorts based off of the first character in the string, essentially like alphabetical order but with numbers.

The *sum()* function does not work because it is a direct mathematical computation but the values are still characters and thus cannot be used in mathematical operations.

B

The *y <- c("5",7,12)* command does work, but only 1 data type is allowed for the vector which is set by the first argument in the function. Thus, all values are saved as character type.

The *y[2] + y[3]* command does not work because again, mathematical operations cannot be applied to character objects.

C

The *z <- data.frame(z1="5",z2=7,z3=12)* command works because we are just assigning values to a dataframe and specifying columns and values in those columns.

The *z[1,2] + z[1,3]* operation works correctly by adding the values in the first row, second and third columns together. They are both numeric data types.

Q3

A

The point of reproducible code is to allow for others to easily use the code you wrote to either attempt to solidify your findings by replicating it, or to understand what you did in order to use it in a similar or different way.

B

In my desired career of a data scientist, it will be important to be able to look back at certain techniques or models that I coded during my studies, and understand why I did certain things or what types of questions I can answer with certain types of models.

C

3