Data Wrangling in R

Aakash Kothapally, STA 360: Homework 1

Due Friday August 27, 5 PM EDT

Today's agenda: Manipulating data objects; using the built-in functions, doing numerical calculations, and basic plots; reinforcing core probabilistic ideas.

General instructions for homeworks: Please follow the uploading file instructions according to the syllabus. You will 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. Your code must be completely reproducible and must compile.

Advice: Start early on the homeworks and it is advised that you not wait until the day of. While the professor and the TA's check emails, they will be answered in the order they are received and last minute help will not be given unless we happen to be free.

Commenting code Code should be commented. See the Google style guide for questions regarding commenting or how to write code https://google.github.io/styleguide/Rguide.xml. No late homework's will be accepted.

R Markdown Test

0. Open a new R Markdown file; set the output to HTML mode and "Knit". This should produce a web page with the knitting procedure executing your code blocks. You can edit this new file to produce your homework submission.

Working with data

Total points on assignment: 10 (reproducibility) + 22 (Q1) + 9 (Q2) + 3 (Q3) = 44 points

Reproducibility component: 10 points.

```
# useful for filter function, which I make use of in 1k
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v tibble  3.0.6     v purrr   0.3.4
## v tidyr   1.1.2     v dplyr   1.0.4
## v readr   1.4.0     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x lubridate::as.difftime() masks base::as.difftime()
## x lubridate::date()        masks base::date()
## x dplyr::filter()          masks stats::filter()
## x readr::guess_encoding()  masks rvest::guess_encoding()
## x lubridate::intersect()   masks base::intersect()
## x dplyr::lag()             masks stats::lag()
## x purrr::pluck()           masks rvest::pluck()
## x lubridate::setdiff()     masks base::setdiff()
## x lubridate::union()       masks base::union()
```

1. (22 points total, equally weighted) The data set **rnf6080.dat** records hourly rainfall at a certain location in Canada, every day from 1960 to 1980.

a. Load the data set into R and make it a data frame called `rain.df`. What command did you use?

```
rain.df <- read.table("./data/rnf6080.dat")
rain.df <- data.frame(rain.df)
```

I used the read.table command to load the dat file into R and the data.frame command to specifically ensure it is a data frame.

b. How many rows and columns does `rain.df` have? How do you know? (If there are not 5070 rows and 27 columns, you did something wrong in the first part of the problem.)

I can use the nrow and ncol commands to retrieve the number of rows and columns in the `raind.df` dataframe.

```
row_count <- nrow(rain.df)
col_count <- ncol(rain.df)
```

Here are the values:

```
row_count
```

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

```
col_count
```

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

So, we have that the number of rows is 5070, and the number of columns is 27.

c. What command would you use to get the names of the columns of `rain.df`? What are those names?

I would use the names command to retrieve the names of columns in `rain.df`.

```
column_names <- names(rain.df)
```

Below are a list of the column names:

```
column_names
```

```
##  [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"
```

d. What command would you use to get the value at row 2, column 4? What is the value?

To extract the value of a single cell, we can use the [r, c] notation to get the value of the rth row and cth column. For row 2, column 4, we can do...

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

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

The value is 0.

e. What command would you use to display the whole second row? What is the content of that row?

To extract the values of the entire row, we can use the [r,] notation to get the values of the rth row. For the second row, we can do...

```
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
```

f. What does the following command do?

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

It is modifying names(rain.df), which as we answered in 1c, refers to the column names.

seq(0, 23) creates a sequence of numerical values, starting with 0, ending at 23, and (by default) incrementing by 1, something like (0, 1, 2, . . . , 23).

So c("year","month","day",seq(0,23)) is combining the values to create a list composed of 'year', 'month', 'day', then 0, 1, 2, . . . , 23.

Essentially, the column names are being modified to be "year" for the 1st column, "month" for the 2nd column, "day" for the 3rd column, 0 for the 4th column, 1 for the 5th column, 2 for the 6th column, . . . , 23 for the 27th column. The 0 to 23 presumably represent the hourly columns with numbers associated with each hour of a given day.

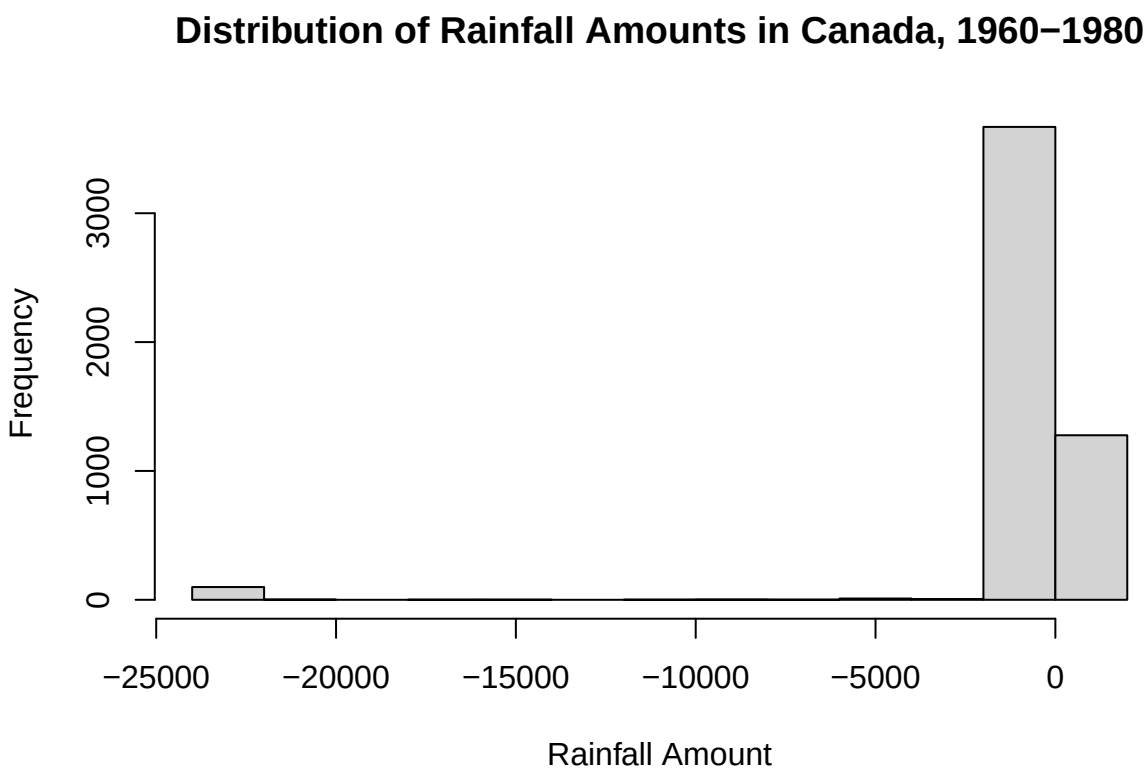
g. Create a new column called `daily`, which is the sum of the 24 hourly columns.

We are summing the hourly columns, which make up the 4th to 27th columns in each row. We can use rowSums to do this summation over those specific columns of each row and place the values for each row in the `daily` column.

```
rain.df$daily <- rowSums(rain.df[,4:27])
```

h. Give the command you would use to create a histogram of the daily rainfall amounts. Please make sure to attach your figures in your .pdf report.

```
hist(rain.df$daily,
     main="Distribution of Rainfall Amounts in Canada, 1960-1980",
     xlab="Rainfall Amount")
```



i. Explain why that histogram above cannot possibly be right.

This histogram cannot possibly be correct because it has days with negative rainfall. After inspecting the data frame more closely, it appears negative values are being placed in certain hours (obviously not meant to indicate that a negative amount of rainfall occurred). I presume these are hours where the amount of rainfall is not available.

j. Give the command you would use to fix the data frame.

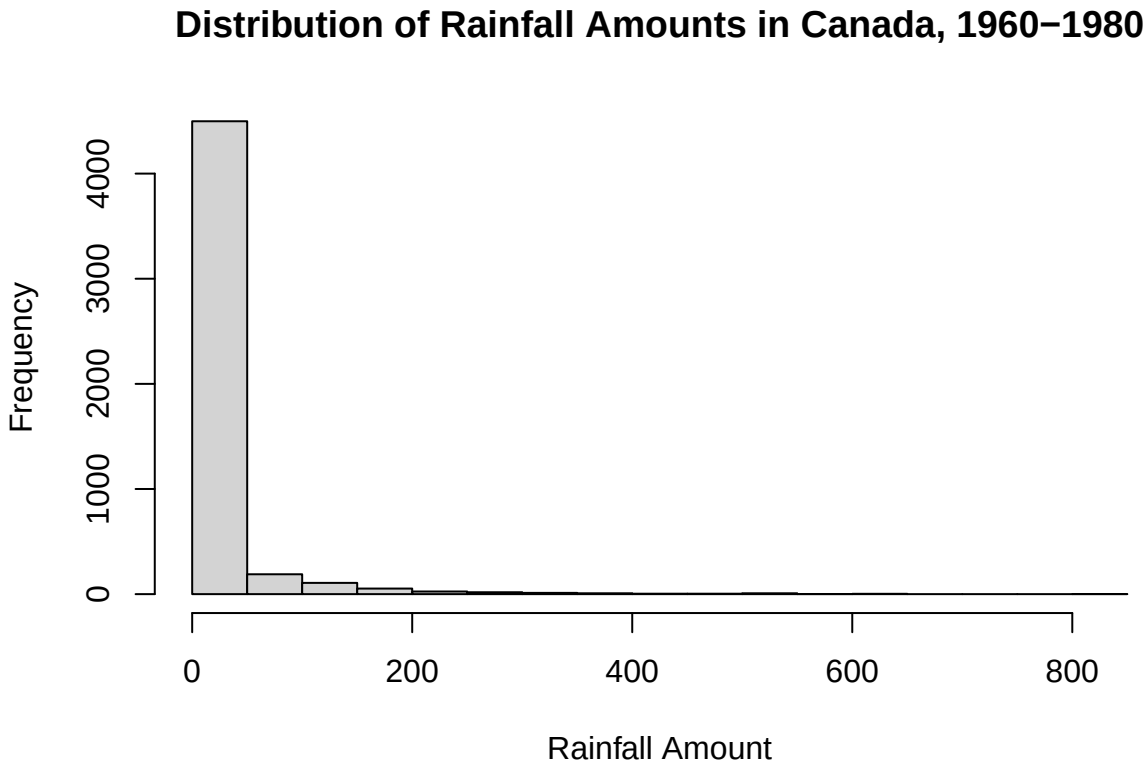
I would exclude all days with negative rain values in the data frame since it seems fairer to not record days where we don't have positive data for it. I can store it in a different variable to ensure both copies are available for use. I will use the filter function and keep daily values that are non-negative.

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

k. Create a corrected histogram and again include it as part of your submitted report. Explain why it is more reasonable than the previous histogram.

Now, we can look at the updated histogram.

```
hist(rain_wo_neg.df$daily,
     main="Distribution of Rainfall Amounts in Canada, 1960-1980",
     xlab="Rainfall Amount")
```



This histogram makes more sense, as there are no negative rainfall numbers and no extremely high rainfall numbers here. And as a sanity check, the most common rainfall amount is the values close to or equal to 0, which also makes sense.

Data types

2. (9 points, equally weighted) Make sure your answers to different parts of this problem are compatible with each other.

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

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

The first command takes a sequence of strings into a vector, storing it in the variable x. The following commands make use of this, computing max, sort, and sum on the sequence of strigs.

When comparing a sequence of strings for max and sort, it utilizes ASCII values to make the comparison. Specifically each string is treated as a sequence of ASCII values (each character has a unique ASCII value). The first ASCII values of each string are compared, and if two strings share the same value, it follows with a comparison of the second value, and so on.

max(x) evaluates with the vector x by returning the highest ASCII value. So in this case, it looks at the first characters of the string, and the ASCII value of "1" < "5" < "7", so "7" is the max. max(x) = "7".

sort(x) evaluates with the vector x by returning them in increasing ASCII value. So in this case, it looks at the first characters of the string, and the ASCII value of "1" < "5" < "7", so "12", "5", "7" is the order.

sum(x) returns an error, because you cannot perform an addition using characters in R and x is a sequence of characters, it needs to use numeric/complex/logical values in order to sum something up.

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

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

The first command combines values into a vector of length 3 and stores it in y. Importantly, since the first value is a character, all other values in the vector are also treated as characters. The first element is "5", the second element is "7", the third element is "12".

y[2] + y[3] returns an error, because y[2] is "7" and y[3] is "12", and you cannot perform an addition using characters in R, it needs to use numeric/complex/logical values in order to sum something up.

c. For the next two commands, either explain their results, or why they should produce errors.

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

The first command combines values into a data frame of length 3 and stores it in z. Importantly, even though the first value is a character, the other values in the data frame are numbers and are stored as numbers. The first element is "5", the second element is 7, the third element is 12, all in the first row of the data frame.

z[1, 2] + z[1, 3] evaluates with a sum of 19, because z[1, 2] is row 1, column 2 of the data frame, so it's 7, and z[1, 3] is row 1, column 3 of the data frame, so it's 12. The addition of 7 and 12 is 19.

3. (3 pts, equally weighted).

a.) What is the point of reproducible code?

Code at times can rely on taking a random sample, and if that code is re-run over and over again, it can produce differing results with a new run. It can also involve one-time use of variables that are later removed from the code. If either of these things happen, it makes it difficult for another person to use your code to replicate/validate your results. Reproducible code is vital for sharing and communicating work with others.

b.) Given an example of why making your code reproducible is important for you to know in this class and moving forward.

Code is often used for analysis, so an example could be answering a question where I have to take a random sample then provide a statistical description of it. If my code wasn't reproducible, I could be describing a result that wouldn't occur when the grader re-runs the code. This also matters moving forward, since analysis of results is important and code being reproducible is important to making that analysis remain relevant to others.

c.) On a scale of 1 (easy) – 10 (hard), how hard was this assignment. If this assignment was hard (> 5), please state in one sentence what you struggled with.

It felt like around a 3, since I needed to refresh my R skills, but nothing was particularly troublesome to work out.