HW: Week 5

36-350 – Statistical Computing

Week 5 – Spring 2022

Name: Henry Wu

Andrew ID: zhenyuw

You must submit **your own** lab as a PDF file on Gradescope.

HW Length Cap Instructions

- If the question requires you to print a data frame in your solution e.g. `q1_out_df`, you must first apply **head(q1_out_df, 30)** and **dim(q1_out_df)** in the final knitted pdf output for such a data frame.
- Please note that this only applies if you are knitting the `Rmd` to a `pdf`, for Gradescope submission purposes.
- If you are using the data frame output for visualization purposes (for example), use the entire data frame in your exploration
- The **maximum allowable length** of knitted pdf HW submission is **30 pages**. Submissions exceeding this length *will not be graded* by the TAs. All pages must be tagged as usual for the required questions per the usual policy
- For any concerns about HW length for submission, please reach out on Piazza during office hours

---

```
suppressWarnings(library(tidyverse))
```

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

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

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

Question 1

(20 points)

An alternative to `read.table()` and such is the `scan()` function. The `scan()` function is *very* handy, particularly when someone gives you weirdly formatted text data files. (Maybe groups of unequal-length rows map to one record, etc., etc.) In this problem, use `scan()` to read in `simple.txt` (which you downloaded for Lab 5) and then post-process what you've read in to create a data frame with correct column names and correct data types (`character` for the `name` column and `double` for all the other columns). Your final step will be to print out the data frame. Look at the documentation for `scan()` and pay particular attention to the `what` argument. Once you've scanned the data, use a combination of, e.g., `matrix()` and `data.frame()` to bend the data to your will, and then cast the data in columns 2 through 8 to `numeric`. Hint: `t()` transposes a matrix. Also, pass `stringsAsFactors=FALSE` as an argument to `data.frame()`.

```
# FILL ME IN
file.url = "C:/Users/wuzyh/Downloads/simple.txt"
t = scan(file.url, what = character())
M = (matrix(t, ncol = 8, byrow = TRUE))
data.frame(matrix(as.numeric(M[2:10,2:8]), ncol = 7, byrow = TRUE), row.names = M[2:10, 1], stringsAsFa
```

```
## X1 X2 X3 X4 X5 X6 X7
## galaxy.A 17.831300 19.073100 21.638000 20.547400 21.237800 22.462700
23.822100
## galaxy.B 23.049100 23.674200 16.907700 17.744800 21.010600 19.554200
20.687600
## galaxy.C 21.459700 22.895000 22.153600 23.034600 16.443100 16.978900
20.828600
## galaxy.D 19.238700 20.566100 21.048400 22.577900 21.879100 22.785700
16.209900
## galaxy.E 16.528800 20.628300 19.056800 20.437100 20.827400 22.354300
21.688900
## galaxy.F 22.611600 16.061300 16.255100 20.655200 19.088700 20.479900
20.763900
## galaxy.G 22.322500 21.704400 22.581300 15.873200 15.953100 20.528000
18.986500
## galaxy.H 20.450300 20.638500 22.203800 21.638100 22.546200 0.038356 0.058309
## galaxy.I 0.063701 0.059006 0.063202 0.057773 0.061548 0.063769 0.061427
```

Question 2

(20 points)

Let's up the ante a bit here. Download `branch.txt` from the `DATA` directory on Canvas. Examine it with an external viewer. This one's a bit of a mess. (Welcome to real-world data.) Construct a data frame from these data. Assume all the columns are character (there is no need in this exercise to do a final cast of the numeric columns to numeric type). To read in the data themselves, I'd advise you to use `scan()` while skipping the first line and using "|" as the separator. (See the documentation for `scan()`.) To make the data frame, you could use a combination of `matrix()` and `data.frame()` as in Q1, but before doing do, clean up your strings: replace all tab symbols (`\t`) with empty strings, and replace any leading spaces and trailing spaces with empty strings. (Hint: `gsub()`.) Note that the data comprise 14 columns and 39 rows (not including the header).

Getting the column names is a bit trickier: they are separated by `|_.`, which `scan()` cannot handle. So I'd advise you to use `scan()` to read in *just the first line* (use `\n` as a separator; see the argument `n`), then

use `strsplit()` to split the line into 14 column names. You might have to "escape" (i.e., apply double backslashes) some or all of the characters used in splitting. Again, clean things up: get rid of `\t` symbols and trailing spaces.

In the end, display the first four columns and first six rows of your beautiful data frame, rising like a phoenix from the ashes of the terribly formatted ASCII file that you began with.

```
# FILL ME IN
file = "C:/Users/wuzyh/Downloads/branch.txt"
T = file %>% scan(what = character(), sep = "|", skip = 1)

data = matrix(gsub("(\t+)|(^[[:space:]]+)|([[:space:]]+$)", "", T), ncol = 14, byrow = TRUE)
col_n = scan(file, what = character(), sep = "\n", n = 1)
col_name = unlist(strsplit(col_n, split = "\\|_."))

col_names = gsub("([[:space:]]+$)", "",col_name)
d = data.frame(data)
colnames(d) = col_names
d[1:6, 1:4]
```

```
##    Subm_ID Score Sigma_s Detection_image
## 1 A_SP_0.0  80.9    0.25              No
## 2 A_SP_0.1 100.3    0.25              No
## 3 A_SP_0.4 579.8     1.0              No
## 4 A_SP_1.0 120.4    0.25              No
## 5 A_SP_1.7  78.5    10.0              No
## 6 A_SP_1.9 939.1     1.0              No
```

Question 3

(20 points)

Read in data from `https://download.bls.gov/pub/time.series/ap/ap.data.0.Current`, which are housed at the Bureau of Labor Statistics. Note before you start that the data are *tab delimited*, and you might find it helpful to remember that a tab is denoted `\t` in a string. The data may not read in cleanly with a simple function call; you may need to skip the header, in which case you will need to provide column names yourself. Also, the parser may misidentify column types, so you may have to set those too. And. . . you may have to cast data in some columns to be of proper type, after the reading in of the data is done. (Data wrangling is a messy business.) Once everything is read in and cast to (if necessary) proper type, display the mean and standard deviation of the data in the value column for every year *after* 2009 (i.e., 2010 and later). The tidyverse will help you here. Hint: `group_by()`.

```
# FILL ME IN
q_3 = read.delim('https://download.bls.gov/pub/time.series/ap/ap.data.0.Current')
q_3[, 4] = as.numeric(q_3[, 4])
```

```
## Warning: NAs introduced by coercion
```

```
filter(summarize(group_by(q_3, year), average.value = mean(value, na.rm = TRUE), sd_value = sd(value, na
```

```
## # A tibble: 13 x 3
##     year average.value sd_value
```

```
##    <int>         <dbl>    <dbl>
##  1  2010         14.8     30.2
##  2  2011         15.0     29.7
##  3  2012         14.5     27.9
##  4  2013          9.66    22.5
##  5  2014          3.07     1.90
##  6  2015          2.81     2.00
##  7  2016          2.61     1.90
##  8  2017          2.71     1.89
##  9  2018          2.75     1.83
## 10  2019          2.70     1.81
## 11  2020          2.64     1.93
## 12  2021          3.16     2.16
## 13  2022          3.42     2.23
```

Question 4

(20 points)

Download `planets.csv` from the Canvas site. It is in the Week 7 directory. Use an external viewer (your choice) to look at the file. Then apply an appropriate function to read the file's contents into `R`. Your goal: to determine what proportion of the columns have data in at least 20% of their rows. (In other words, step from column to column and see if the proportion of `NA`'s is less than 80%. Then determine the proportion of the columns that fulfill this condition.) Your final answer should be 82.86% [or 0.8286].

```
# FILL ME IN
c = read_csv(file = "C:/Users/wuzyh/Downloads/planets.csv", skip= 73)
```

```
## Rows: 3550 Columns: 70
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr   (7): pl_hostname, pl_letter, pl_discmethod, pl_bmassprov, ra_str, dec_...
## dbl  (61): rowid, pl_pnum, pl_orbper, pl_orbpererr1, pl_orbpererr2, pl_orbpe...
## lgl   (1): gaia_gmagerr
## date  (1): rowupdate
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
#per = sapply(c, f(x){x=ifelse(is.na(x), 0, 1)})
#per
```

Question 5

(20 points)

Make a data frame that is in essence a "dictionary" for the data in the `planets.csv` file. What this means is: extract those lines of the file that contain variable names and corresponding definitions, and from those lines extract the variable names into a vector called `variable` and the definitions into a vector called `definition`. Output the first six rows only! (Hint: in your call to `data.frame()`, set the argument `stringsAsFactors` to `FALSE`. This changes the column contents to character strings rather than factor variables.) Hint: let's say you do an `strsplit()` to split the variable from the definition in each line. The output will be a list,

with one list element for each line that contains two strings, one for the variable and one for the definition. A handy way to extract all of the variables would be, e.g., sapply(<output from strplit>,`[[`,1). That `[[` function is really useful.

```
# FILL ME IN
```

Question 6

(20 points)

Extract the 2020 Major League Baseball standings from the web site given below and put them into a *single* data frame that contains all 30 MLB teams, with the first column being the team name, the second column being the number of wins, and the third column being the number of losses. Order the data frame by decreasing number of wins. Use `rvest` functions to extract any tables you need, which are of class `data.frame`, and then process the data frames until you get a single one as described above.

```
if ( require(rvest) == FALSE ) {
  install.packages("rvest",repos="https://cloud.r-project.org")
  library(rvest)
}
```

```
## Loading required package: rvest
```

```
##
## Attaching package: 'rvest'
```

```
## The following object is masked from 'package:readr':
##
##     guess_encoding
```

```
site = read_html("https://www.baseball-reference.com/leagues/MLB-standings.shtml")
# FILL ME IN
```