HW: Week 5

36-350 – Statistical Computing

Week 5 – Fall 2020

Name: Kimberly Zhang

Andrew ID: kyz

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

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

```
## -- Attaching packages ------------------------------------------------------------------------------

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

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

Question 1

(12 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()`.

```r
header<-scan("simple.txt", what="character", nlines = 1)
df<-scan("simple.txt", what= list("", double(),double(),double(),double(),double(),double(),double()), s
df<-data.frame(matrix(unlist(df), ncol= 8), stringsAsFactors = FALSE)
colnames(df)<-header
df[, 2:8] <- sapply(df[, 2:8], as.numeric)
df
```

```
##       name       u       g       r       i       z       y redshift
## 1 galaxy.A 17.8313 16.9077 16.4431 16.2099 16.0613 15.8732 0.038356
## 2 galaxy.B 19.0731 17.7448 16.9789 16.5288 16.2551 15.9531 0.058309
## 3 galaxy.C 21.6380 21.0106 20.8286 20.6283 20.6552 20.5280 0.063701
## 4 galaxy.D 20.5474 19.5542 19.2387 19.0568 19.0887 18.9865 0.059006
## 5 galaxy.E 21.2378 20.6876 20.5661 20.4371 20.4799 20.4503 0.063202
## 6 galaxy.F 22.4627 21.4597 21.0484 20.8274 20.7639 20.6385 0.057773
## 7 galaxy.G 23.8221 22.8950 22.5779 22.3543 22.3225 22.2038 0.061548
## 8 galaxy.H 23.0491 22.1536 21.8791 21.6889 21.7044 21.6381 0.063769
## 9 galaxy.I 23.6742 23.0346 22.7857 22.6116 22.5813 22.5462 0.061427
```

Question 2

(12 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.

```r
df<-scan("branch.txt", what= "character", skip = 1, sep = "|")
df<-matrix(df, ncol= 14)
df<-apply(df, c(1,2), function(x){gsub("(\\\t)|( +)", "", x)})
df<-data.frame(df, stringsAsFactors = FALSE)
header<-scan("branch.txt", what= "character", sep = "\n", nlines = 1)
header<-gsub("(\\\t)|( +)", "", header)
header<-unlist(strsplit(header, split = "\\|_\\."))
header[13]<-gsub("\\|", "", header[13])
colnames(df)<-header
df %>% select(Subm_ID:Detection_image) %>% slice(1:6)
```

```
##    Subm_ID                            Score                         Sigma_s
## 1 A_SP_0.0 Alldetectedobjectswereconsidered                             1.0
## 2     80.9                                5                        Lanczos3
## 3     0.25                                                            47x47
## 4       No                         A_SP_1.0 Alldetectedobjectswereconsidered
## 5     Free                            120.4                              20
## 6 Modified                             0.25
##   Detection_image
## 1        Modified
## 2            None
## 3            None
## 4             1.0
## 5        Lanczos3
## 6           47x47
```

Question 3

(14 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()`.

```
df<-read.table("https://download.bls.gov/pub/time.series/ap/ap.data.0.Current",
         sep = "\t", header= TRUE, stringsAsFactors = TRUE)
df<-data.frame(df)
df$value<-as.numeric(df$value)
df %>% filter(df$year > 2009) %>% group_by(year) %>%
  summarize(Avg.Val = mean(value, na.rm= TRUE), Sd.Val = sd(value, na.rm= TRUE))
```

```
## # A tibble: 11 x 3
##     year Avg.Val Sd.Val
##    <int>   <dbl>  <dbl>
##  1  2010   5121.  6424.
##  2  2011   5418.  6328.
##  3  2012   5419.  6154.
##  4  2013   4351.  4981.
##  5  2014   2963.  1709.
##  6  2015   2683.  1730.
##  7  2016   2491.  1654.
##  8  2017   2595.  1626.
##  9  2018   2632.  1595.
## 10  2019   2589.  1603.
## 11  2020   2522.  1688.
```

Question 4

(12 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].

```
planets<-read_csv("planets.csv", skip = 73)
total<-nrow(planets)
res<-apply(planets, 2, function(x){sum(complete.cases(x)/total)})
length(res[which(res > 0.2)])/ncol(planets)
```

```
## [1] 0.8285714
```

Question 5

(14 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.

```
planets<-read.csv("planets.csv", skip=3, nrows=69, header = FALSE)
res<-apply(planets, 1, function(x){
                  tmp<-gsub("# COLUMN ", "", x)
                  strsplit(tmp, split= ": +")
                  })
res<-t(sapply(res, `[[`, 1))
df<-data.frame("variable"= res[,1], "definition"= res[,2],
               stringsAsFactors = FALSE)
head(df)
```

```
##        variable                        definition
## 1   pl_hostname                         Host Name
## 2     pl_letter                     Planet Letter
## 3 pl_discmethod                  Discovery Method
## 4       pl_pnum       Number of Planets in System
## 5     pl_orbper             Orbital Period [days]
## 6 pl_orbpererr1 Orbital Period Upper Unc. [days]
```

Question 6

(12 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

## Loading required package: xml2

##
## Attaching package: 'rvest'

## The following object is masked from 'package:purrr':
##
##     pluck

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

```r
site = read_html("https://www.baseball-reference.com/leagues/MLB-standings.shtml")
df.lst<-site %>% html_nodes("table")  %>% html_table()
df<-bind_rows(df.lst)
df<-select(df, Tm, W, L)
colnames(df)<-c("Team.Name", "Num.Wins", "Num.Losses")
arrange(df, desc(Num.Wins))
```

```
##                Team.Name Num.Wins Num.Losses
## 1    Los Angeles Dodgers       43         17
## 2         Tampa Bay Rays       40         20
## 3       San Diego Padres       37         23
## 4        Minnesota Twins       36         24
## 5      Oakland Athletics       36         24
## 6      Chicago White Sox       35         25
## 7      Cleveland Indians       35         25
## 8         Atlanta Braves       35         25
## 9           Chicago Cubs       34         26
## 10      New York Yankees       33         27
## 11     Toronto Blue Jays       32         28
## 12         Miami Marlins       31         29
## 13       Cincinnati Reds       31         29
## 14   St. Louis Cardinals       30         28
## 15        Houston Astros       29         31
## 16     Milwaukee Brewers       29         31
## 17  San Francisco Giants       29         31
## 18 Philadelphia Phillies       28         32
## 19      Seattle Mariners       27         33
## 20    Kansas City Royals       26         34
## 21    Los Angeles Angels       26         34
## 22         New York Mets       26         34
## 23  Washington Nationals       26         34
## 24       Colorado Rockies       26         34
## 25     Baltimore Orioles       25         35
## 26  Arizona Diamondbacks       25         35
## 27         Boston Red Sox       24         36
## 28         Detroit Tigers       23         35
## 29         Texas Rangers       22         38
## 30    Pittsburgh Pirates       19         41
```

Question 7

(12 points)

Your goal: to create a data frame with statistics course id's in one column (e.g., "36-350") and associated course titles in another (e.g., "Statistical Computing"). Google "cmu course catalog statistics" and go to the course catalog page. (Note: this is *not* the course schedule page.) The page should be entitled "Department of Statistics and Data Science Courses". View the HTML source and see if there is a single node delimiter that will return to you the individual course descriptions. (Hint: there is one.) Use string manipulation tools to extract the course id and course title only from the output of `html_text()`, and then place each into their own data frame columns. Display the final data frame. (Hint: you may need to employ the `[[` function here. See Q6.) (Hint 2: `substr()` might prove handy at the final string manipulation steps, handier than `strsplit()`.) As usual, in your call to `data.frame()`, set the argument `stringsAsFactors` to `FALSE`. Set the column names in your data frame to be "ID" and "Title". Display the first ten rows only.

```r
page<- read_html("http://coursecatalog.web.cmu.edu/schools-colleges/dietrichcollegeofhumanitiesandsocial
res<- page %>% html_nodes("dt") %>% .[-1] %>% html_text()
course.num<-substr(res, 1, 6)
course.title<-substr(res, 8, nchar(res))
df<-data.frame("ID"= course.num, "Title"= course.title,
               stringsAsFactors = FALSE)
head(df, n = 10)
```

```
##        ID                                              Title
## 1  36-201                   Statistical Reasoning and Practice
## 2  36-202                Methods for Statistics & Data Science
## 3  36-204                        Discovering the Data Universe
## 4  36-207 Probability and Statistics for Business Applications
## 5  36-208                                  Regression Analysis
## 6  36-217                 Probability Theory and Random Processes
## 7  36-218              Probability Theory for Computer Scientists
## 8  36-219                 Probability Theory and Random Processes
## 9  36-220              Engineering Statistics and Quality Control
## 10 36-225                      Introduction to Probability Theory
```

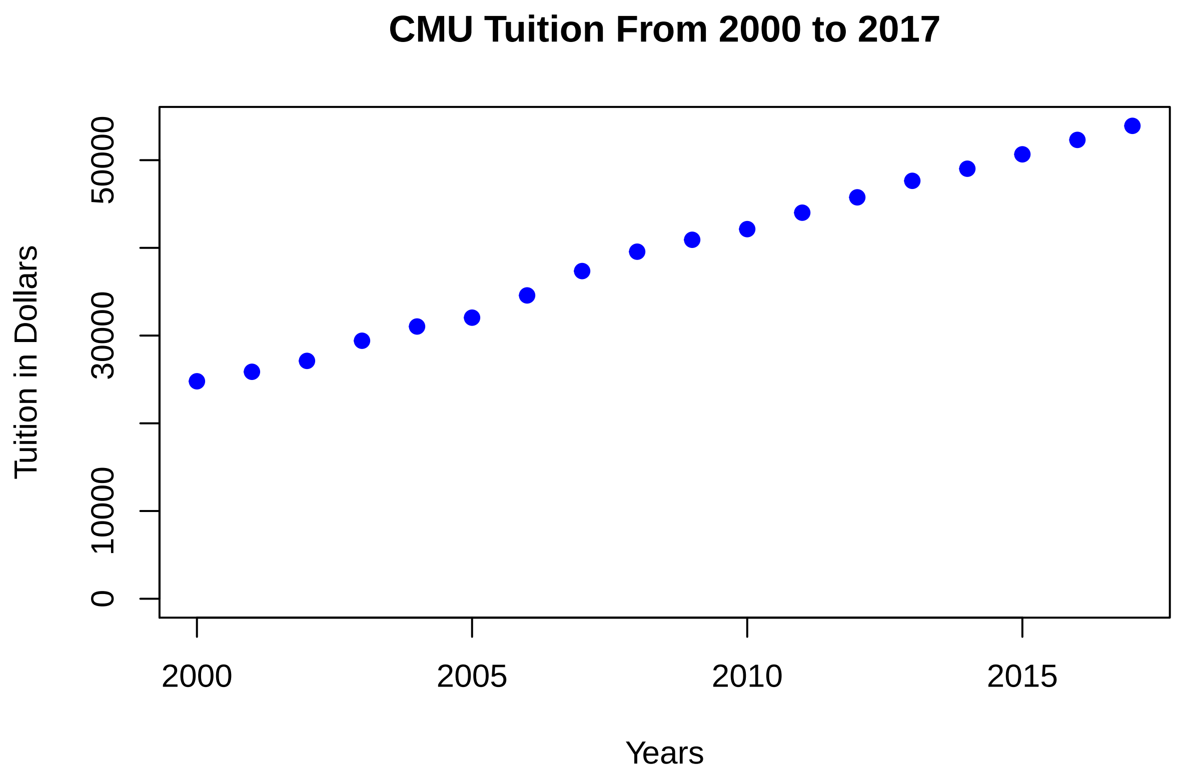
Question 8

(12 points)

Here you will build upon Q17 of Lab 5 and plot CMU's tuition for every year from 2000 to 2017 (18 values total). At this point in the course, I'm not going to be particularly picky about how you construct your data-extraction "engine"; for instance, you may find it easiest to run a `for()` loop after having set aside a vector called `tuition`. Or maybe you'll find it intuitive to use pipes. Do whatever works for you. Just get that plot built! Apply proper labeling to both the x and y axes, add a title to the plot, and set `pch=19`. Set the y-axis limits to be 0 to the maximum value along the y-axis. And add some color.

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

URL = "https://api.data.gov/ed/collegescorecard/v1/schools?"
key = "fBHXtkYTYPOjQCXvPnQf3kUqfD4RPnnys9i9sT49"
response<-GET(URL,
      query= list(api_key = key, school.name = "Carnegie Mellon University"))
df.lst<-fromJSON(content(response, "text"), simplifyVector = TRUE) %>%
  suppressMessages() %>% .$results
tuition = vector()
for(i in 2000:2017){
  val<-df.lst %>% .[[as.character(i)]] %>% .[["cost"]] %>%
       .[["tuition"]] %>% .[1, 1]
  tuition<-append(tuition, val)
}
plot(2000:2017, tuition, xlab= "Years",
     ylab= "Tuition in Dollars", main = "CMU Tuition From 2000 to 2017",
     pch= 19, ylim= c(0, max(tuition)), col = "blue")
```


CMU Tuition From 2000 to 2017
Tuition in Dollars
0
10000
30000
50000
2000
2005
2010
2015
Years