Lab: Week 5

36-350 – Statistical Computing

Week 5 – Fall 2020

Name: Kyle Wagner

Andrew ID: kowagner

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

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

Question 1

(5 points)

Notes 5A (2-4)

Download `simple.txt` from the Canvas site. It is in the `DATA` 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`. Show the names of the columns. Make sure the names are correct, and that there are eight columns. (Note: you may find that your first choice of function does not provide optimal results. If so, try another function. Note that `read_delim()` with a properly formatted regex might help here. If you use base `R` functionality, consider including the argument `stringsAsFactors=FALSE`.)

```
simple.df = suppressMessages(read_delim("simple.txt", delim = " "))
colnames(simple.df)
```

```
## [1] "name" "u" "g" "r" "i" "z" "y"
## [8] "redshift"
```

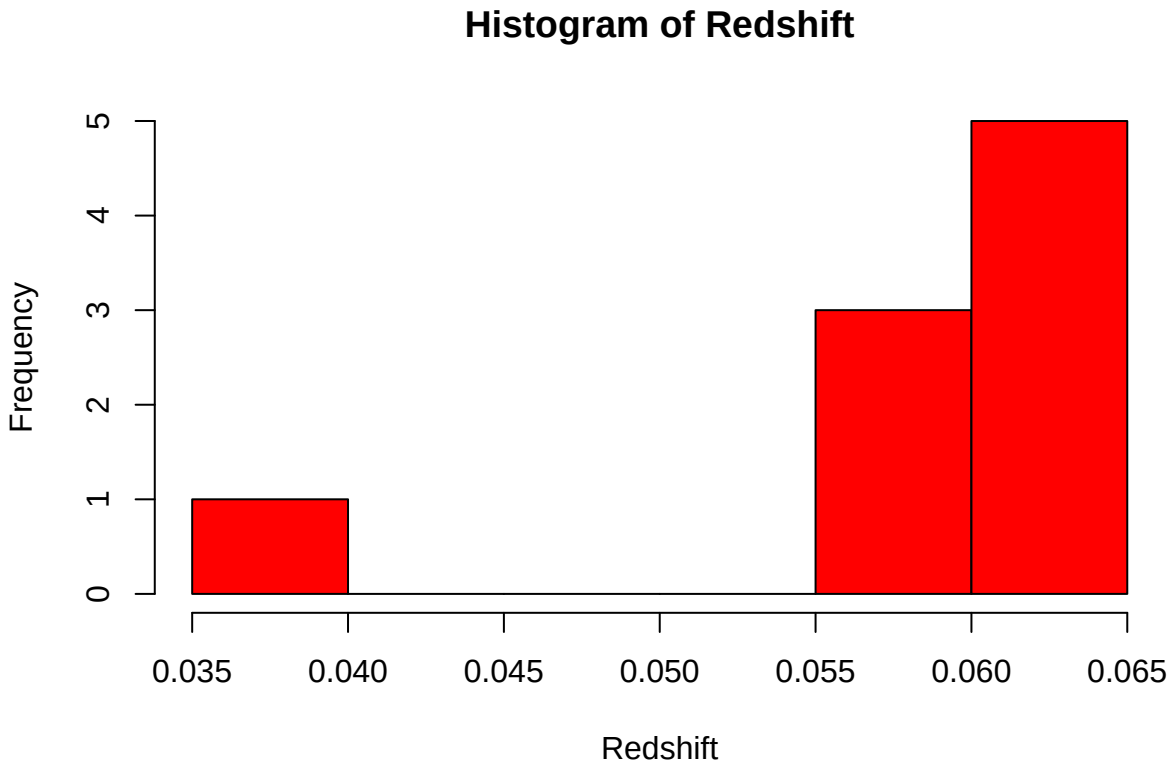
Question 2

(5 points)

Notes 5D (4-5)

Create a histogram for the `redshift` variable in the data frame input in Q1. Change the x-axis label to "Redshift". Add some color too.

```
galaxy_hist <- hist(simple.df$redshift,breaks = 4,
                    xlab = "Redshift",main = "Histogram of Redshift",col = "red")
```



Question 3

(5 points)

Notes 5A (2-4) Read in the data file from Q1 but skip the header. Display the names that `R` gives to the columns. For `readr`, you may find sub-optimal results; if so, you would need to specify the column names as an argument. (You need not actually do that here.)

```
simple.df = read.table("simple.txt",header = FALSE)
colnames(simple.df)
```

```
## [1] "V1" "V2" "V3" "V4" "V5" "V6" "V7" "V8"
```

Question 4

(5 points)

Notes 5A (2-4)

Read in the data file from Q1 but only read in the first four lines, while retaining the header.

```
simple.df = read.table("simple.txt",header = TRUE,nrows = 4)
simple.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
```

Question 5

(5 points)

Notes 5A (2-4)

Download `students.txt` from the Canvas site. It is in the `DATA` 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`. Use an appropriate argument that ensures that each non-numerical column is treated as a vector of strings, and not a factor variable. (You may have done this back up in Q1.) Show that you've done this by displaying the type of the variable the column `Last Name`. (Use `typeof()`.)

```
students.df <- suppressMessages(read_delim("students.txt",delim = " "))
typeof(students.df$Last.Name)
```

```
## [1] "character"
```

Question 6

(5 points)

Notes 5A (2-4)

Download `emline.csv` from the Canvas site. It is in the `DATA` 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`. When you are done, show the mean and median values of the `sfr` column. Hint: if they are wildly different, you may need to adjust how you read in the data. Hint: look for numbers that represent missing data, and use an appropriate argument to tell `R` that those numbers should be converted to `NA`.

```
emline <- suppressMessages(read_csv("emline.csv",na = "-9999"))
mu <- mean(emline$sfr,na.rm = TRUE)
m <- median(emline$sfr,na.rm = TRUE)
cat("sfr avg:",mu,"\nsfr median:",m)
```

```
## sfr avg: -0.71412
## sfr median: -0.6436
```

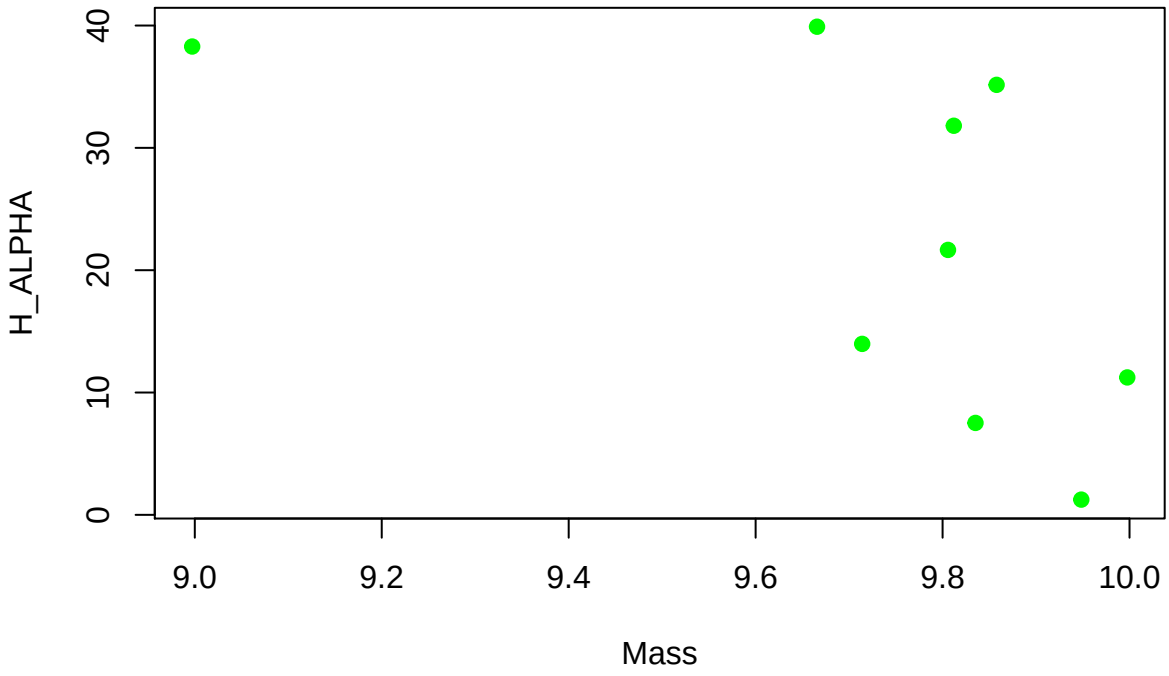
Question 7

(5 points)

Notes 5D (9-10)

Make a bivariate scatter plot of `H_ALPHA` vs. `mass` (y vs. x, i.e., `H_ALPHA` is on the y-axis). Change the axis labels, change the point type to 19, and add some color.

```
plot(emline$mass,emline$H_ALPHA,xlab = "Mass",ylab = "H_ALPHA",pch = 19,col = "green")
```



Question 8

(5 points)

Notes 5A (2-4)

Redo Q6, but here use a combination of the functions `sum()` and `complete.cases()` to determine how many rows have no `NA` values. Also: if you use a `readr` function here, try to suppress the message about parsing specification, by wrapping your call to `read_csv()` with a call to `suppressMessages()`. Note: `complete.cases()` returns `TRUE` if a row has no `NA` values and `FALSE` otherwise. Remember that applying `sum()` to a vector of logicals returns the total number of `TRUE` values.

```
suppressMessages(sum(complete.cases(read_csv("emline.csv",na = "-9999"))))
```

```
## [1] 5
```

Question 9

(5 points)

Notes 5A (2-4)

Download `students.csv` from the Canvas site. It is in the `DATA` 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`. Apply appropriate arguments or other function calls after the fact to ensure that the first three columns of the final data frame (or tibble) are character vectors, and the fourth column is a factor variable. (Simply print out the data frame or tibble; the variable type will be shown.) (Hint: if you use a `readr` function, then the argument `col_types` can be a single string with one letter for each column: "c" for a character vector, "d" for double, "i" for integer, "f" for factor, etc.). Hint: if using base `R`, it is simplest to start with `stringsAsFactors=FALSE` and then convert one column to a factor variable.

```
students.df <- read.csv("students.csv",stringsAsFactors=FALSE,skip = 7)
students.df$Vegetarian <- as.factor(students.df$Vegetarian)
students.df
```

```
##   Last.Name First.Name Andrew.ID Vegetarian
## 1   Baggins      Bilbo  bilbobag          N
## 2       Bar        Foo     fbar1          Y
## 3    Umlaut     Dieter   dumlaut          N
```

Question 10

(5 points)

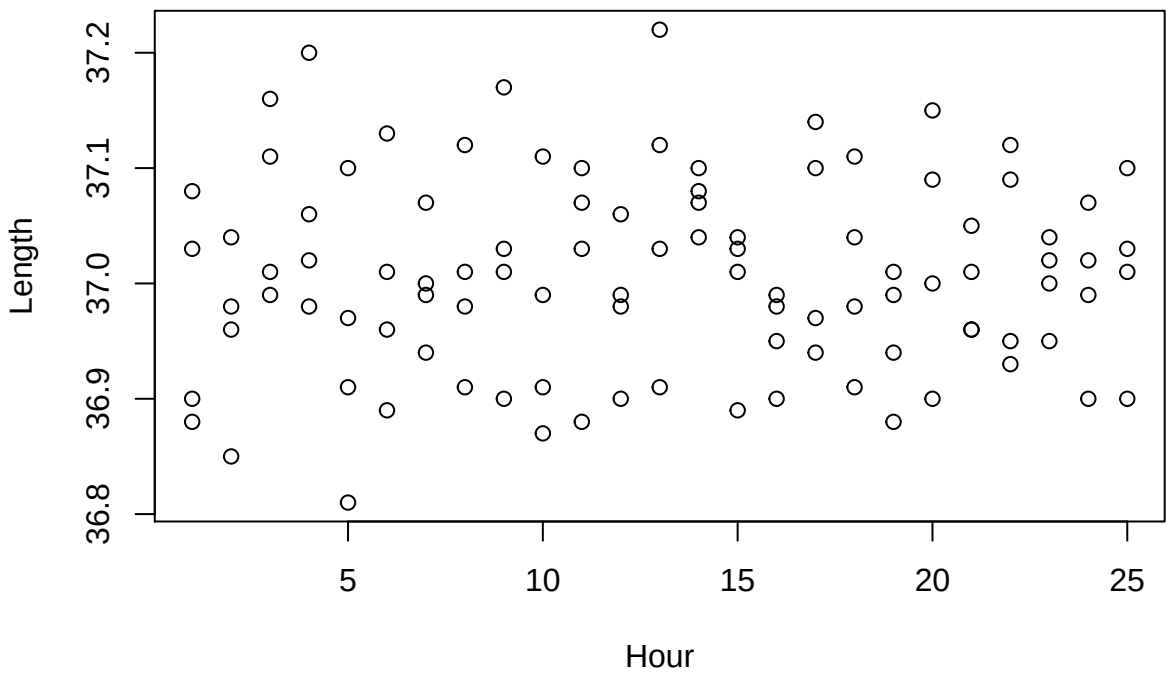
Notes 5A (5)

Download `bolts.xls` from the Canvas site. It is in the `DATA` directory. If you cannot view the file with EXCEL, there are 100 rows and two columns: `Hour` and `Length`. Read the file into `R`, compute for average `Length` for each value of `Hour`, then plot that average versus `Hour`. (Remember: the variable before the "versus" is the y variable.) Note that if you pass a two-column data frame to `plot()`, it will know what to do. Hint: perhaps `group_by()` and `summarize()` here.

```
require(readxl)
```

```
## Loading required package: readxl
```

```
tbl <- read_excel("bolts.xls")
tbl %>%
  group_by(Hour) %>%
  suppressMessages(summarize(avg = mean(Length))) %>%
  plot()
```



Question 11

(5 points)

Notes 5A (5)

Download `data.xlsx` from the Canvas site. It is in the `DATA` directory. Also in the `DATA` directory is `data.png`; download and open it. It is a screen shot of a portion of the data that indicates possibly problematic data. Read the file into `R` while properly dealing with these problematic data. Note that data actually exists in the field(s) marked "#########"; you need not actually do anything about that marker. Display just the four columns in which problematic data existed in the screen shot (use `select()`); those data should be replaced with `NA`.

```
tbl <- read_excel("data.xlsx",na = "999")
tbl %>% select(Sex,RaceEthnicity,ADHD,Presentation)
```

```
## # A tibble: 3 x 4
##     Sex RaceEthnicity  ADHD Presentation
##   <dbl>         <dbl> <dbl> <chr>
## 1     2             4     1 3
## 2    NA            NA    NA <NA>
## 3     2             1     0 4
```

Question 12

(5 points)

Notes 5B (3-7)

Extract the first two bullet-pointed text items on the wikipedia page for "Timeline of the Universe", using `rvest` functions. (Hint: review the notes to see how bullet-pointed items are delimited in HTML.)

```
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
```

```
page = read_html("https://en.wikipedia.org/wiki/Timeline_of_the_Universe")
page %>% html_nodes("li") %>% .[c(1,2)] %>% html_text %>% paste(.,collapse="\n\n") %>% cat()
```

```
## For a timeline of the universe from formation to the present day, see:
Timeline of cosmological epochs
##
## For a timeline of the universe from the present to its presumed conclusion,
see: Timeline of the far future
```

Question 13

(10 points)

Notes 5B (3-7)

Fascinating. What's in the "Timeline of the far future"? How far out can we extrapolate? Apply `rvest` functions to extract the tables on this page, and show the last row of the longest table. If you do things correctly, you'll see that we can extrapolate out to 10 to the 10 to the 10 to the 56 years. That's a mighty long time. (Hint: to extract a particular element from a list in a piping enviroment, you can do `[[`(n), where you would replace `n` with the number of the list element you want.)

```
page = read_html("https://en.wikipedia.org/wiki/Timeline_of_the_far_future")
page %>% html_nodes("table") %>% html_table(fill = TRUE) %>% `[[`(which.max(lapply(.,nrow))) %>% .[nrow
```

```
## Years from now
## 112 NA 10101056{\\displaystyle 10^{10^{10^{56}}}}[note 2][note 7]
## Event
## 112 Time for quantum effects to generate a new Big Bang, resulting in a new
universe. Around this vast timeframe, quantum tunnelling in any isolated patch
of the now-empty universe could generate new inflationary events, resulting in
new Big Bangs giving birth to new universes.[135]Because the total number of
ways in which all the subatomic particles in the observable universe can be
combined is 1010115{\\displaystyle 10^{10^{115}}},[136][137] a number which,
when multiplied by 10101056{\\displaystyle 10^{10^{10^{56}}}}, disappears into
the rounding error, this is also the time required for a quantum-tunnelled and
quantum fluctuation-generated Big Bang to produce a new universe identical to
our own, assuming that every new universe contained at least the same number of
subatomic particles and obeyed laws of physics within the landscape predicted
by string theory.[138]
```

Question 14

(10 points)

Notes 5B (3-7)

For Q10 of HW 3, you extracted the number of submitted astrophysics articles per month in 2019. My solution utilized `readLines()`, `regexpr()`, and `regmatches()`. Repeat the exercise here using `rvest` functions. The web page is here. Your final output should be a vector of numbers whose names are given by the built-in constant `month.abb`. Hint: utilize `html_nodes()` such that you are look for bolded content.

```
arxiv = read_html("https://arxiv.org/year/astro-ph/19")
x <- arxiv %>% html_nodes("b") %>% .[c(2:13)] %>% html_text() %>% as.numeric(.)
names(x) <- month.abb
x
```

```
##  Jan  Feb  Mar  Apr  May  Jun  Jul  Aug  Sep  Oct  Nov  Dec
## 1125 1038 1475 1258 1159 1023 1277 1225 1344 1295 1100 1100
```

Question 15

(5 points)

Notes 5B (3-7)

When was the statistician William Sealy Gosset born? (He is the "Student" of "Student's t distribution.") Yes, you can look this up, but I'd rather you apply functions from the `rvest` library, in particular, `html_table()`, as applied to Gosset's wikipedia page. Once you have the list of tables, determine which one is the correct one (hint: it's the one that has "Born" in one of the rows), keep that, then show the row that has the birthdate. You need not do more. You could imagine expanding upon this code to, e.g., search all tables associated with a given person to find the one with the word "Born" and then using string manipulation tools to pull the birthdate out...and thus you could create a table of people and birthdates. But one step at a time.

```
gosset = read_html("https://en.wikipedia.org/wiki/William_Sealy_Gosset")
gosset %>% html_nodes("table") %>% .[2] %>% html_text()
```

```
## [1] "William Sealy GossetWilliam Sealy Gosset (aka Student) in 1908 (age
32)Born(1876-06-13)13 June 1876Canterbury, Kent, EnglandDied16 October
1937(1937-10-16) (aged 61)Beaconsfield, Buckinghamshire,
EnglandOther namesStudentAlma materNew College, Oxford, Winchester
CollegeKnown forStudent's t-distribution, statistical significance, design of
experiments, Monte Carlo method, quality control, Modern synthesis,
agricultural economics, econometricsScientific careerInstitutionsGuinness
BreweryInfluencesWilliam Archibald Spooner, Karl PearsonInfluencedRonald A.
Fisher, Harold Jeffreys, Egon Pearson, Jerzy Neyman, John Wishart, W. Edwards
Deming, Edwin S. Beaven, Herbert Hunter\n"
```

In Notes 5C, we discuss `httr` and provide a very simple example that did not require an authentification key.

Before completing the questions below, go to this web site and sign up to receive an authentification key that will allow you to access `data.gov`. (Be sure to check your spam folder after signing up, if you don't see the email with your key in your inbox.) Note that the key should be treated as a string.

Question 16

(5 points)

Notes 5C (4-5)

Use your data.gov key to query the URL below. In your call to the `GET()` function, you would include an argument, `query`, that is a list with two entries: `api_key=[YOUR KEY]`, and `school.name="Carnegie Mellon University"`. Save the output from `GET()`, and confirm that the type of the output is JSON.

```
URL = "https://api.data.gov/ed/collegescorecard/v1/schools?"

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

```
## Loading required package: httr
```

```
response = GET(url = URL, query = list(api_key = "x0fZIwthRR362YAaIkkK2kbAdTwsPCsh2p7vzV6M", school.nam
http_type(response)
```

```
## [1] "application/json"
```

Question 17

(10 points)

Notes 5C (4-5)

What was CMU's tuition in 2015? Use `fromJSON` (with the default setting: `simplifyVector=TRUE`) to convert your output from `GET()` in Q16 to an `R` object. (Feel free to surround your call to `fromJSON` with `suppressMessages()` so you don't have to see that irritating message about default encoding.) Then use `names()` on the output of `fromJSON` to start honing in on the answer. You'll need to access list/data frame elements, and you may find at some point you have to use `[["2015"]]` to access the year 2015 data frame rather than the more usual `$2015`, which my parser at least does not like. Once you find the tuition, you can extract the number using the usual `[row,col]` format. (Note: to access a list within a list, you can chain things together: to get element `z` of list `y`, which is an element of list `x`, do xyz or x[[“y”]]$z or x[[“y”]][[“z”]].) Your final answer should be $50,665. Oh, the good old days.

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

```
## Loading required package: jsonlite
```

```
##
## Attaching package: 'jsonlite'
```

```
## The following object is masked from 'package:purrr':
##
##     flatten
```

```
x <- suppressMessages(fromJSON(content(response,"text")))
res = x["results"]
prices = res[["results"]][["2015"]][["cost"]][["tuition"]][1,c(1,2)]
prices
```

```
##   out_of_state in_state
## 1        50665    50665
```