HW 10

SDS348 Fall 2019

Olga Ostrovskaya, oo3662

This homework is due on Nov 26, 2019 at 11:59pm. Please submit as a pdf file on Canvas.

For all questions, include the python commands/functions that you used to find your answer. Answers without supporting code will not receive credit.

> ### How to submit this assignment
>
> All homework assignments will be completed using R Markdown. These `.Rmd` files consist of >text/syntax (formatted using Markdown) alongside embedded R code. When you have completed the assignment (by adding R code inside codeblocks and supporting text outside of the codeblocks), create your document as follows:

> - Click the "Knit" button (above) to create an .html file
> - Open the html file in your internet browser to view
> - Go to `File > Print` and print your .html file to a .pdf
> - Upload the .pdf file to Canvas

WARNING: DUE TO AN OLD VERSION OF RSTUDIO SERVER, PYTHON AND R CODE CHUNKS DO NOT INTERACT ON THE EDUCCOMP SERVERS. HOWEVER, IT SHOULD STILL KNIT CORRECTLY (i.e., YOU SHOULD STILL BE ABLE TO USE `r.obj_from_r` and `py$obj_from_python`). YOU COULD JUST KNIT IT EVERY TIME YOU ANSWER A QUESTION, BUT IT WOULD BE A VERY GOOD IDEA TO DO THIS ASSIGNMENT ON YOUR OWN RSTUDIO. UPDATING RSTUDIO ONLY TAKES A SECOND AND DOESN'T DELETE ANY PACKAGES, ETC. JUST GO TO https://rstudio.com/products/rstudio/download/#download (https://rstudio.com/products/rstudio/download/#download) AND DOWNLOAD/RUN/RESTART AND YOU SHOULD BE ALL SET. REGARDLESS, I HAVE ADDED ALTERNATIVE WAYS TO COMPLETE THE ASSIGNMENT (e.g., BY

READING IN THE DATA FROM ONLINE OR FROM A CSV): STILL, I DO ENCOURAGE YOU TO PLAY WITH RETICULATE! THE REMAINING HOMEWORKS WILL BE DONE USING JUPYTER NOTEBOOKS.

Question 1: (2 pts)

The dataset `faithful` contains information about eruptions of the Old Faithful geyser in Yellowstone National Park. The first few observations are listed below.

```
library(reticulate)
#use_python("/usr/bin/python3") uncomment if on the servers
faithful<-faithful
head(faithful)
```

```
##   eruptions waiting
## 1     3.600      79
## 2     1.800      54
## 3     3.333      74
## 4     2.283      62
## 5     4.533      85
## 6     2.883      55
```

Bring the dataset `faithful` over from R and look at it in Python. THE REST OF THIS ASSIGNMENT MUST BE COMPLETED IN PYTHON UNLESS OTHERWISE NOTED! What type of object is it? What are the minimum and maximum values of the variables eruptions and waiting? Put a dot after the object and hit tab: Look at all those functions!

```


### 1.
import numpy as np
faithful=r.faithful
#again, this may not work by running the chunk on the servers, but it should still knit.
#to read this into python directly, sidestepping the issue, just uncomment and run the following two l
ines
import pandas as pd
#faithful=pd.read_csv("https://vincentarelbundock.github.io/Rdatasets/csv/datasets/faithful.csv")

#faithful
min(faithful.eruptions); min(faithful.waiting)
```

```
## 1.6
## 43.0
```

```
max(faithful.eruptions); max(faithful.waiting)
```

```
## 5.1
## 96.0
```

```
faithful.describe()
```

```
##          eruptions     waiting
## count  272.000000  272.000000
## mean     3.487783   70.897059
## std      1.141371   13.594974
## min      1.600000   43.000000
## 25%      2.162750   58.000000
## 50%      4.000000   76.000000
## 75%      4.454250   82.000000
## max      5.100000   96.000000
```

```
type(faithful)
```

```
## <class 'pandas.core.frame.DataFrame'>
```

The min values for eruptions and waiting: 1.6min and 43min; max: 5.1 and 96 mins. Faithful is a Data Frame.

Question 2: (4 pts)

Now, import `numpy` and use functions from it to compute the mean, median, and variance for each variable. Finally, compute the correlation between the two variables using corrcoef (it will return a matrix). You can access individual variables in a dataframe using the `.` operator (e.g., `faithful.eruptions`).

```
### 2.
# the easier way is described in Q1 already.
np.mean(faithful.eruptions)
```

```
## 3.4877830882352936
```

```
np.var(faithful.eruptions)
```

```
## 1.2979388904492855
```

```
np.median(faithful.eruptions)
```

```
## 4.0
```

```
np.mean(faithful.waiting)
```

```
## 70.8970588235294
```

```
np.var(faithful.waiting)
```

```
## 184.14381487889264
```

```
np.median(faithful.waiting)
```

```
## 76.0
```

```
np.corrcoef(faithful, rowvar = False)
```

```
## array([[1.        , 0.90081117],
##        [0.90081117, 1.        ]])
```

The mean and variance for eruptions is 3.488 min and 1.298 min. For waiting - 70.9 and 184.1 min. The correlation between eruptions and waiting is very good: 0.901.

Question 3: (6 pts)

Recall how logical indexing of a dataframe works in python. To refresh your memory, in the example code below I ask python for the number of rows in the dataset where the variable `waiting` takes on values greater than 60. Then I ask for the average of the variable `eruptions` when the variable `waiting` is above 60.

```
faithful[faithful.waiting>60].shape
```

```
## (189, 2)
```

```
faithful[faithful.waiting>60].eruptions.mean()
# or np.mean(faithful[faithful.waiting>60].eruptions)
```

```
## 4.138587301587303
```

3.2 (1 pt) What is the standard deviation of the variable `eruptions` ?

```
### Q3.2.
faithful.eruptions.std()
```

```
## 1.1413712511052077
```

```
np.std(faithful.eruptions)
```

```
## 1.1392712102257678
```

Q: Why 2 numbers are different? The SD for the ‘eruptions’ is 1.14 min.

3.3 (2 pts) What is the standard deviation of the variable `eruptions` when `waiting` is *less than* the median?

```
# Q - I thought it's python, not R. Is it ok?

### 3.3.
from statistics import median

#np.median(faithful.waiting)
faithful[faithful.waiting<median(faithful.waiting)].eruptions.std()
```

```
## 0.9583443399236853
```

```
np.std(faithful[faithful.waiting<np.median(faithful.waiting)].eruptions)
```

```
## 0.9547617317201806
```

Depending on the package, the SD of this subset is 0.95 or 0.96 min.

3.4 (2 pts) What is the standard deviation of the variable `eruptions` when `waiting` is *greater than* the median?

```
### 3.4
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
faithful %>% filter(waiting > median(waiting)) %>% summarize(m1 = mean(eruptions), sd1 = sd(eruption
s))
```

```
##         m1       sd1
## 1 4.358364 0.3730518
```

SD of eruptions in this subset is 0.373 min.

Question 4: (4 pts)

Both variables are measured in minutes. Create two new variables named `eruptions_h` and `waiting_h` that give each variable **in hours rather than minutes** and add them to the dataset `faithful`. To help get you started, I have given you code that creates a new variable called eruptions_minus_one. Instead, computes the requested transformation. Print out the first few rows of the updated dataset using `head()`.

```
### 4.

# update the code below
#faithful['eruptions_minus_one']=(faithful['eruptions']-1)
#faithful
#del faithful['eruptions_minus_one']

faithful['eruptions_h']=(faithful['eruptions']/60)
faithful['waiting_h']=(faithful['waiting']/60)
faithful.head()
```

```
##    eruptions  waiting  eruptions_h  waiting_h
## 0      3.600     79.0      0.06000   1.316667
## 1      1.800     54.0      0.03000   0.900000
## 2      3.333     74.0      0.05555   1.233333
## 3      2.283     62.0      0.03805   1.033333
## 4      4.533     85.0      0.07555   1.416667
```

Question 5: (4 pts)

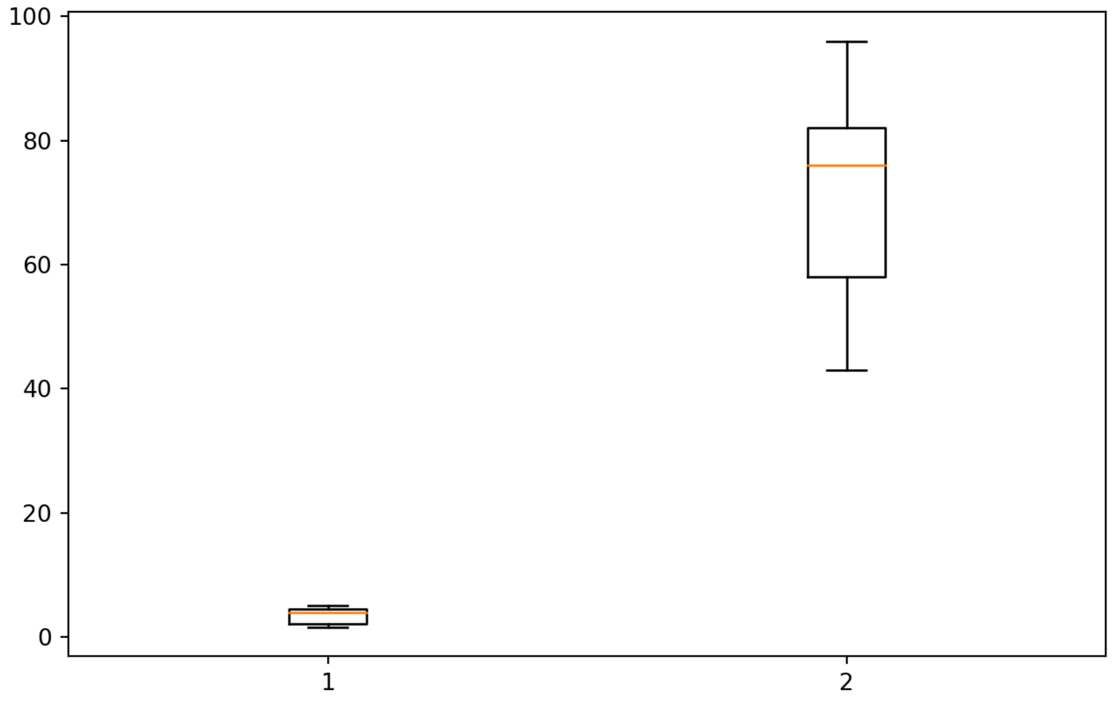
Let's make some plots in python!

5.1 (1 pt) Create a boxplot of each original variable (eruptions and waiting) using the `boxplot()` function from matplotlib

```
### 5.

import matplotlib.pyplot as plt
#plt.boxplot(faithful['eruptions'])
#plt.boxplot(faithful['waiting'])
plt.boxplot([faithful['eruptions'], faithful['waiting']])
```

```
## {'whiskers': [<matplotlib.lines.Line2D object at 0x115d663d0>, <matplotlib.lines.Line2D object at 0
x115d66910>, <matplotlib.lines.Line2D object at 0x115d7d950>, <matplotlib.lines.Line2D object at 0x115
d74e10>], 'caps': [<matplotlib.lines.Line2D object at 0x115d66e10>, <matplotlib.lines.Line2D object at
0x115d56690>, <matplotlib.lines.Line2D object at 0x115d74850>, <matplotlib.lines.Line2D object at 0x11
5d84850>], 'boxes': [<matplotlib.lines.Line2D object at 0x115d56650>, <matplotlib.lines.Line2D object
at 0x115d66e90>], 'medians': [<matplotlib.lines.Line2D object at 0x115d74890>, <matplotlib.lines.Line2
D object at 0x115d84d50>], 'fliers': [<matplotlib.lines.Line2D object at 0x115d74d90>, <matplotlib.lin
es.Line2D object at 0x115d7de90>], 'means': []}
```

```
plt.show()
#replace df, variable with your stuff
```



5.2 (1 pt) Create a histogram of each original variable using the `hist()` function.

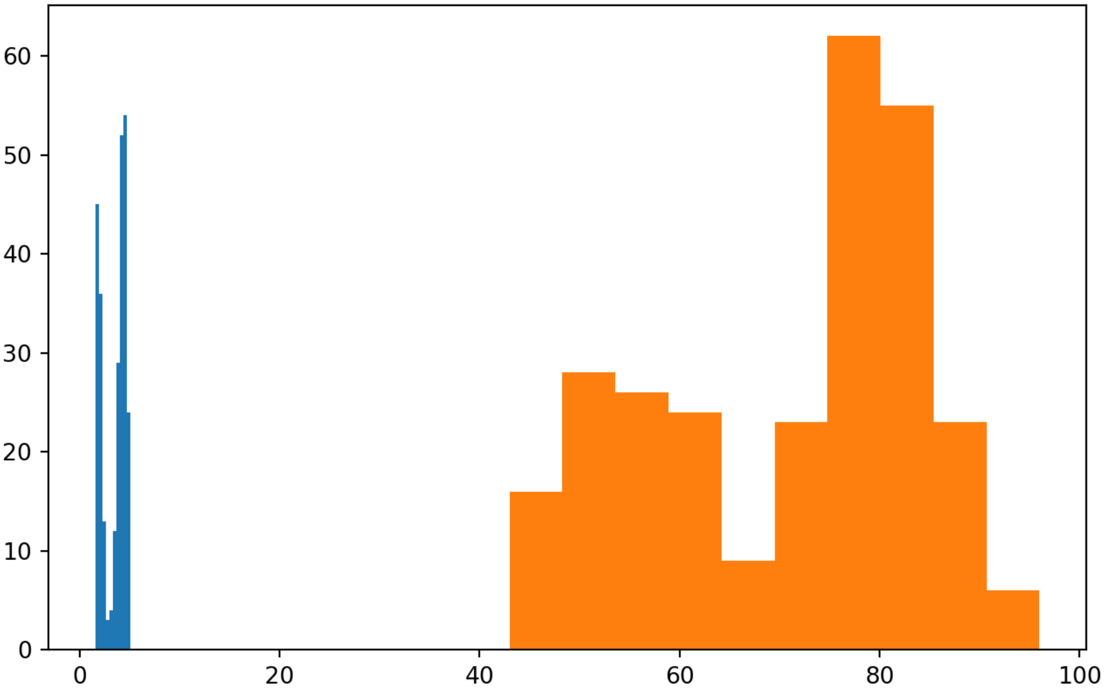
```
### 5.2.

plt.hist(faithful['eruptions'])
```

```
## (array([45., 36., 13.,  3.,  4., 12., 29., 52., 54., 24.]), array([1.6 , 1.95, 2.3 , 2.65, 3.  , 3.
35, 3.7 , 4.05, 4.4 , 4.75, 5.1 ]), <a list of 10 Patch objects>)
```

```
plt.hist(faithful['waiting'])
```

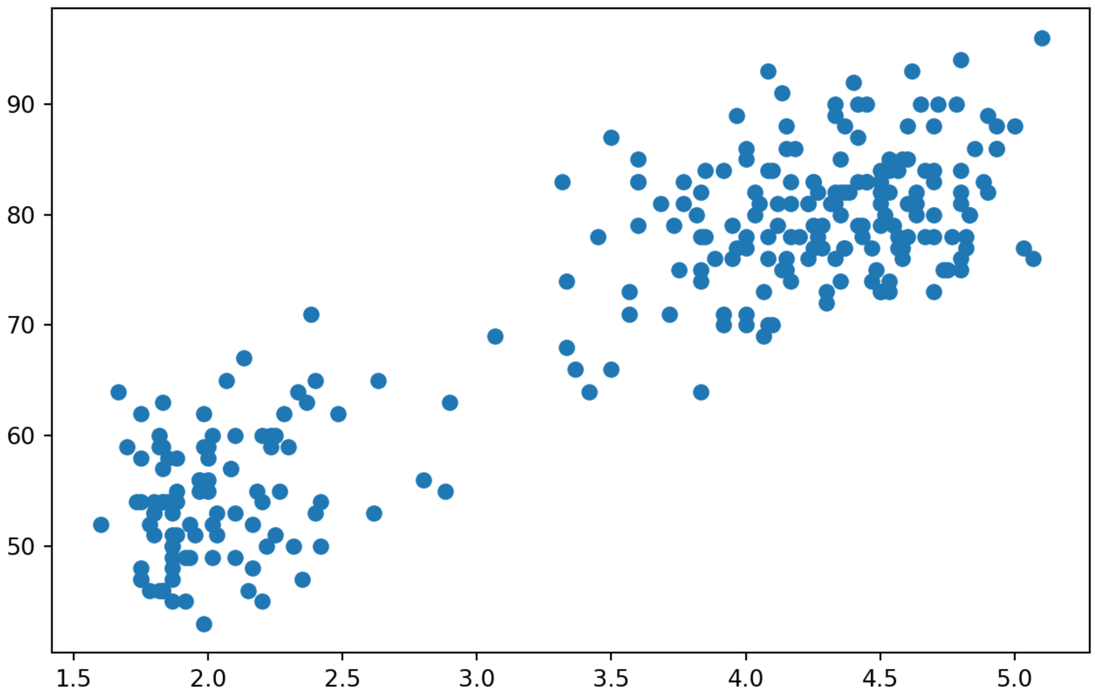
```
## (array([16., 28., 26., 24.,  9., 23., 62., 55., 23.,  6.]), array([43. , 48.3, 53.6, 58.9, 64.2, 6
9.5, 74.8, 80.1, 85.4, 90.7, 96. ]), <a list of 10 Patch objects>)
```

```
plt.show()
```



5.3 (1 pt) Create a scatterplot by plotting both variables against each other using the `plot()` function. Note that plot takes two arguments: the x-axis variable and the y-axis variable.

```
### 5.3.
plt.scatter(faithful["eruptions"],faithful["waiting"])
plt.show()
```



6.1 (1 pt) Load the tips dataset from `seaborn` (a plotting package we will use later). Have a look at it below. What type of object is it?

```
#import pandas as pd
import seaborn as sns
#pd.read_csv("http://www.justFYIyouCanReadDataFromURLs.com/myfile.csv")


tips = sns.load_dataset('tips')
iris = sns.load_dataset('iris') #yep, that iris

type(tips)
```

```
## <class 'pandas.core.frame.DataFrame'>
```

```
tips.head()
```

```
##    total_bill   tip     sex smoker  day    time  size
## 0       16.99  1.01  Female     No  Sun  Dinner     2
## 1       10.34  1.66    Male     No  Sun  Dinner     3
## 2       21.01  3.50    Male     No  Sun  Dinner     3
## 3       23.68  3.31    Male     No  Sun  Dinner     2
## 4       24.59  3.61  Female     No  Sun  Dinner     4
```

```
tips.describe()

#using pandas groupby functionality
```

```
##        total_bill         tip        size
## count  244.000000  244.000000  244.000000
## mean    19.785943    2.998279    2.569672
## std      8.902412    1.383638    0.951100
## min      3.070000    1.000000    1.000000
## 25%     13.347500    2.000000    2.000000
## 50%     17.795000    2.900000    2.000000
## 75%     24.127500    3.562500    3.000000
## max     50.810000   10.000000    6.000000
```

```
groups = tips.groupby('sex')
groups['tip'].mean()

#tips.to_csv("tips.csv") #create tips.csv file in case reticulate isn't working so you can read it int
o R
```

```
## sex
## Male      3.089618
## Female    2.833448
## Name: tip, dtype: float64
```

Tips is a data frame. The dropped vars are categorical - sex, smoker, date, time.

6.2 (1 pt) Access tips using `py$df` in R and make a plot with ggplot illustrating the relationship between total_bill, tip, time, and sex (find an effective way to include all four variables in the same plot)!

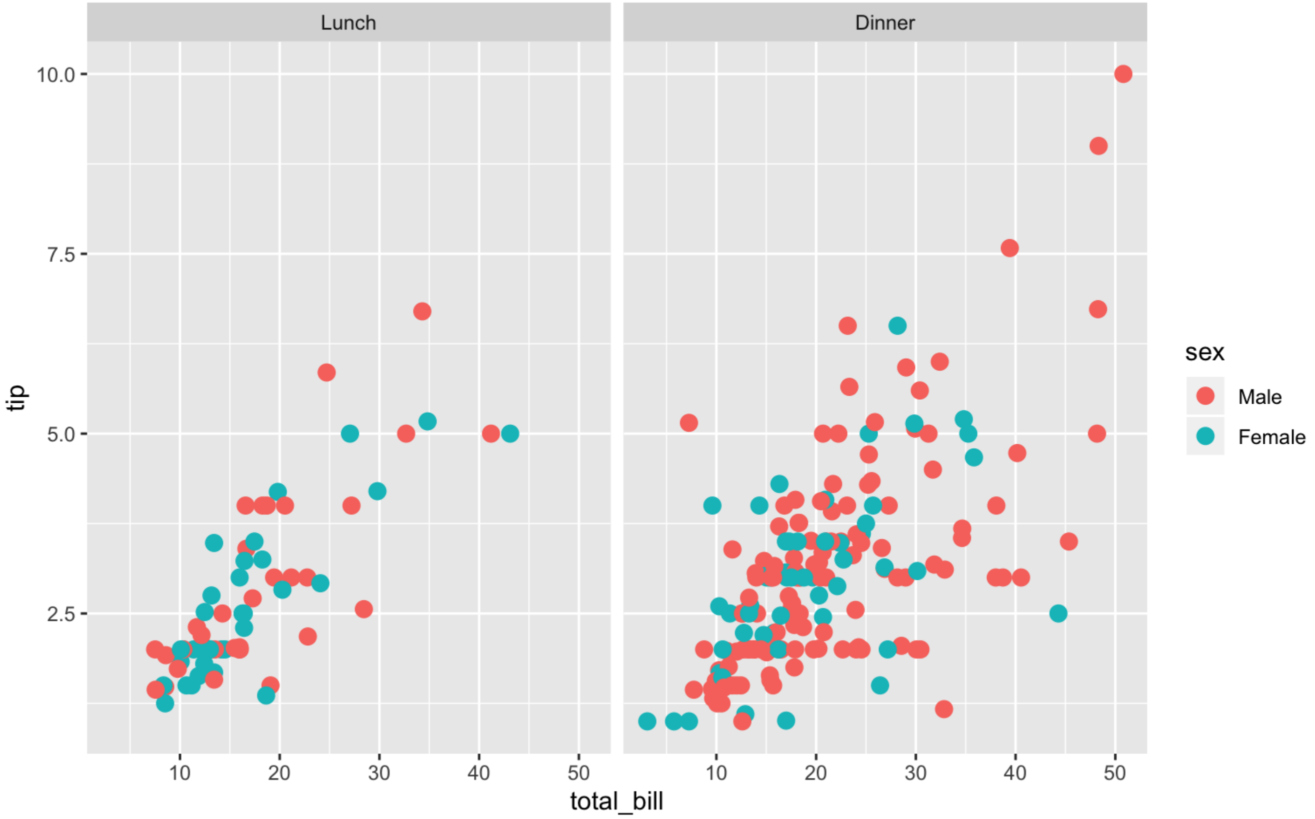
```
library(tidyverse)
```

```
## ── Attaching packages ──────────────────────────────────────────────────── tidyverse 1.
2.1 ──
```

```
## ✔ ggplot2 3.2.1     ✔ readr   1.3.1
## ✔ tibble  2.1.3     ✔ purrr   0.3.3
## ✔ tidyr   1.0.0     ✔ stringr 1.4.0
## ✔ ggplot2 3.2.1     ✔ forcats 0.4.0
```

```
## ── Conflicts ──────────────────────────────────────────────────── tidyverse_conflict
s() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
#tips<-read.csv("tips.csv")

tips<- py$tips
tips %>% ggplot(aes(total_bill, tip, color = sex))+
  geom_point(size = 3)+
  facet_wrap(~time)
```



```
## R version 3.6.1 (2019-07-05)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS Mojave 10.14.4
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] forcats_0.4.0    stringr_1.4.0    purrr_0.3.3      readr_1.3.1
##  [5] tidyr_1.0.0      tibble_2.1.3     ggplot2_3.2.1    tidyverse_1.2.1
##  [9] dplyr_0.8.3      reticulate_1.13 knitr_1.25
##
## loaded via a namespace (and not attached):
##  [1] tidyselect_0.2.5 xfun_0.10        haven_2.1.1      lattice_0.20-38
##  [5] colorspace_1.4-1 vctrs_0.2.0      generics_0.0.2   htmltools_0.4.0
##  [9] yaml_2.2.0       rlang_0.4.1      pillar_1.4.2     withr_2.1.2
## [13] glue_1.3.1       modelr_0.1.5     readxl_1.3.1     lifecycle_0.1.0
## [17] munsell_0.5.0    gtable_0.3.0     cellranger_1.1.0 rvest_0.3.4
## [21] evaluate_0.14    labeling_0.3     broom_0.5.2      Rcpp_1.0.2
## [25] backports_1.1.5  scales_1.0.0     jsonlite_1.6     hms_0.5.1
## [29] png_0.1-7        digest_0.6.22    stringi_1.4.3    grid_3.6.1
## [33] cli_1.1.0        tools_3.6.1      magrittr_1.5     lazyeval_0.2.2
## [37] crayon_1.3.4     pkgconfig_2.0.3  zeallot_0.1.0    Matrix_1.2-17
## [41] xml2_1.2.2       lubridate_1.7.4  assertthat_0.2.1 rmarkdown_1.16
## [45] httr_1.4.1       rstudioapi_0.10  R6_2.4.0         nlme_3.1-141
## [49] compiler_3.6.1
## [1] "2019-11-22 13:27:06 CST"
```

```
##                                                                                             sysname 
##                                                                                            "Darwin" 
##                                                                                             release 
##                                                                                            "18.5.0" 
##                                                                                             version 
## "Darwin Kernel Version 18.5.0: Mon Mar 11 20:40:32 PDT 2019; root:xnu-4903.251.3~3/RELEASE_X86_64" 
##                                                                                            nodename 
##                                                                                   "cns-f-gdca18425" 
##                                                                                             machine 
##                                                                                            "x86_64" 
##                                                                                               login 
##                                                                                            "ndr432" 
##                                                                                                user 
##                                                                                            "ndr432" 
##                                                                                      effective_user 
##                                                                                            "ndr432"
```