GR5206 Midterm Exam

Name and UNI

10/19/2018

The STAT GR5206 Fall 2018 Midterm is open notes, open book(s), open computer and online resources are allowed. Students are **not** allowed to communicate with any other people regarding the exam. This includes emailing fellow students, using WeChat and other similar forms of communication. Before the exam, the students should **turn off** their cellphones and pass them to the left side of each row. At the same time, please **close** the mailbox and **log out** WeChat and all the other apps for messaging and chatting. If there is any suspicion of one or more students cheating, further investigation will take place. If students do not follow the guidelines, they will receive a zero on the exam and potentially face more severe consequences. The exam will be posted on Canvas at 2:50PM. Students are required to submit both the .pdf and .Rmd files on Canvas (or .html if you must) by 4:30PM. Late exams will not be accepted.

Part 1 (Google Play Store Apps Data - Split/Apply/Combine and R plot, 11 + 2 pts)

We work on the `apps` dataset which contains approximately $7,700$ Google Play Store apps. There are 13 features that describe a given app. They are:

- **App** – Application name.
- **Category** – Category the app belongs to.
- **Rating** – Overall user rating of the app (betweem 0 and 5).
- **Reviews** – Number of user reviews for the app.
- **Size** – Size of the app.
- **Installs** – Number of user downloads/installs for the app.
- **Type** – Paid of Free
- **Price** – Price of the app
- **Content Rating** Age group the app is targeted at
- **Genres** An app can belong to multiple genres (apart from its main category). For eg, a musical family game will belong to Music, Game, Family genres.
- **Last Updated** Date when the app was last updated on Play Store
- **Current Ver** Current version of the app available on Play Store
- **Android Ver** Mininum required Android version

Read in the dataset using the following code:

```
apps<-read.csv("apps.csv", header = T)
apps$Reviews<-as.numeric(apps$Reviews)
apps$Installs<-factor(apps$Installs, level= c("1+", "5+", "10+", "50+", "100+", "500+", "1,000+", "5,000
head(apps)
```

```
##                                                 App       Category Rating
## 1     Photo Editor & Candy Camera & Grid & ScrapBook ART_AND_DESIGN    4.1
## 2                               Coloring book moana ART_AND_DESIGN    3.9
```

```
## 3 U Launcher Lite - FREE Live Cool Themes, Hide Apps ART_AND_DESIGN    4.7
## 4                                Sketch - Draw & Paint ART_AND_DESIGN    4.5
## 5               Pixel Draw - Number Art Coloring Book ART_AND_DESIGN    4.3
## 6                          Paper flowers instructions ART_AND_DESIGN    4.4
##   Reviews Size    Installs Type Price Content.Rating
## 1     159 19.0     10,000+ Free     0       Everyone
## 2     967 14.0    500,000+ Free     0       Everyone
## 3   87510  8.7  5,000,000+ Free     0       Everyone
## 4  215644 25.0 50,000,000+ Free     0           Teen
## 5     967  2.8    100,000+ Free     0       Everyone
## 6     167  5.6     50,000+ Free     0       Everyone
##                      Genres Last.Updated        Current.Ver  Android.Ver
## 1              Art & Design     7-Jan-18              1.0.0 4.0.3 and up
## 2 Art & Design;Pretend Play    15-Jan-18              2.0.0 4.0.3 and up
## 3              Art & Design     1-Aug-18              1.2.4 4.0.3 and up
## 4              Art & Design     8-Jun-18 Varies with device   4.2 and up
## 5   Art & Design;Creativity    20-Jun-18                1.1   4.4 and up
## 6              Art & Design    26-Mar-17                  1   2.3 and up
```

Problem 1.0

Check the dimension of `apps`, make sure that there are $7,726$ lines and 13 variables (features). [1 pt]

```
# code goes here
dim(apps)
```

```
## [1] 7726   13
```

Prblem 1.1

In order to get an overview of the dataset, we want to check some summary statistics of each variable, and this can be done by calling the `R` function `summary()`. Compuate the summary statistics of all 13 variables and display the result in a **list**. To receive full credit, you must use a vectorized function from the `apply` family or `plyr` family. [2 pts]

```
# code goes here
library(plyr)
alply(apps, 2, summary)
```

```
## $`1`
##                 App
##  ROBLOX           :   9
##  8 Ball Pool      :   7
##  Candy Crush Saga:   7
##  Bubble Shooter   :   6
##  Helix Jump       :   6
##  Nick             :   6
##  (Other)          :7685
##
## $`2`
##             Category
##  FAMILY         :1617
##  GAME           : 974
##  TOOLS          : 633
```

```
##  MEDICAL        : 324
##  LIFESTYLE      : 280
##  PERSONALIZATION: 280
##  (Other)        :3618
##
## $`3`
##      Rating
##  Min.   :1.000
##  1st Qu.:4.000
##  Median :4.300
##  Mean   :4.174
##  3rd Qu.:4.500
##  Max.   :5.000
##
## $`4`
##     Reviews
##  Min.   :       1
##  1st Qu.:     107
##  Median :    2324
##  Mean   :  294777
##  3rd Qu.:   38959
##  Max.   :44893888
##
## $`5`
##       Size
##  Min.   :  0.008
##  1st Qu.:  5.300
##  Median : 14.000
##  Mean   : 22.959
##  3rd Qu.: 33.000
##  Max.   :100.000
##
## $`6`
##         Installs
##  1,000,000+ :1301
##  100,000+   :1037
##  10,000+    : 969
##  10,000,000+: 825
##  1,000+     : 690
##  5,000,000+ : 535
##  (Other)    :2369
##
## $`7`
##    Type
##  Free:7147
##  Paid: 579
##
## $`8`
##      Price
##  Min.   :  0.000
##  1st Qu.:  0.000
##  Median :  0.000
##  Mean   :  1.128
##  3rd Qu.:  0.000
```

```
##  Max.   :400.000
##
## $`9`
##       Content.Rating
##  Everyone    :6172
##  Everyone 10+: 318
##  Mature 17+  : 368
##  Teen        : 868
##
## $`10`
##              Genres
##  Tools          : 633
##  Entertainment  : 448
##  Education      : 417
##  Medical        : 324
##  Action         : 322
##  Personalization: 280
##  (Other)        :5302
##
## $`11`
##     Last.Updated
##  3-Aug-18 : 205
##  31-Jul-18: 189
##  1-Aug-18 : 178
##  2-Aug-18 : 173
##  30-Jul-18: 130
##  25-Jul-18: 124
##  (Other)  :6727
##
## $`12`
##   Current.Ver
##  1      : 476
##  1.1    : 206
##  1.2    : 133
##  2      : 128
##  1.3    : 119
##  1.4    :  82
##  (Other):6582
##
## $`13`
##        Android.Ver
##  4.1 and up  :1929
##  4.0.3 and up:1194
##  4.0 and up  :1109
##  4.4 and up  : 805
##  2.3 and up  : 566
##  5.0 and up  : 490
##  (Other)     :1633
##
## attr(,"split_type")
## [1] "array"
## attr(,"split_labels")
##                X1
## 1             App
```

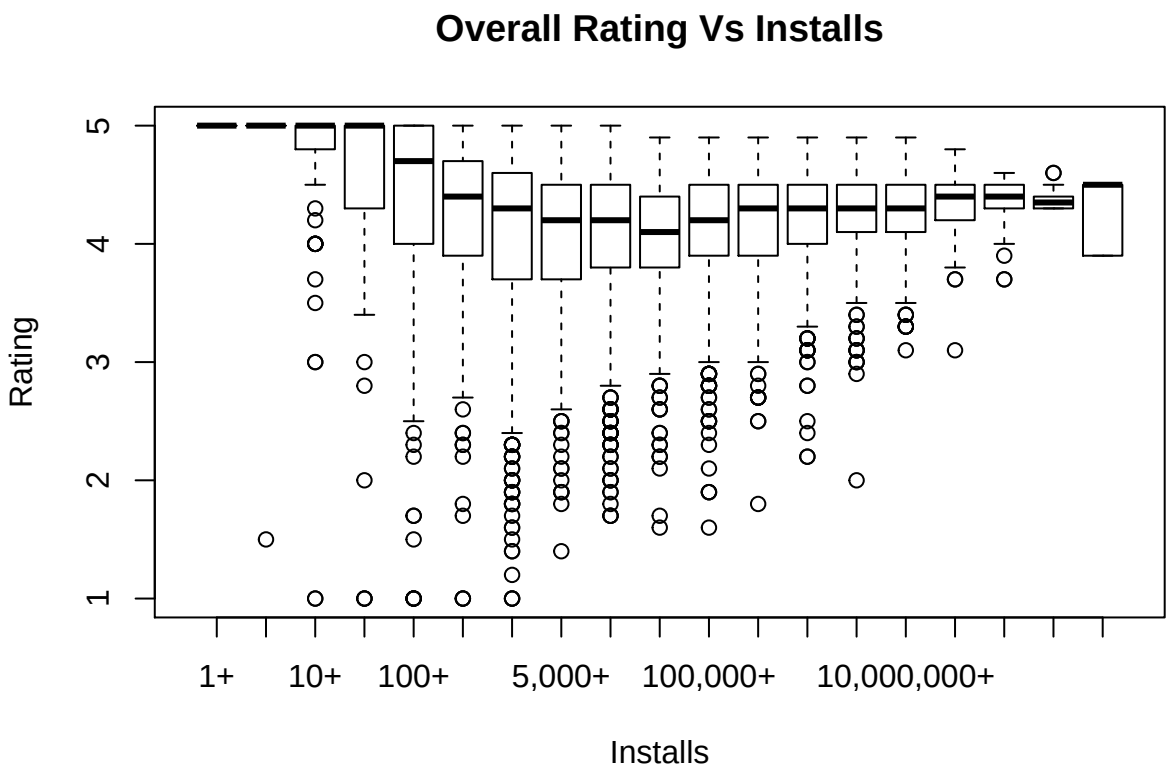
```
## 2        Category
## 3          Rating
## 4         Reviews
## 5            Size
## 6        Installs
## 7            Type
## 8           Price
## 9  Content.Rating
## 10         Genres
## 11   Last.Updated
## 12    Current.Ver
## 13    Android.Ver
```

Problem 1.2

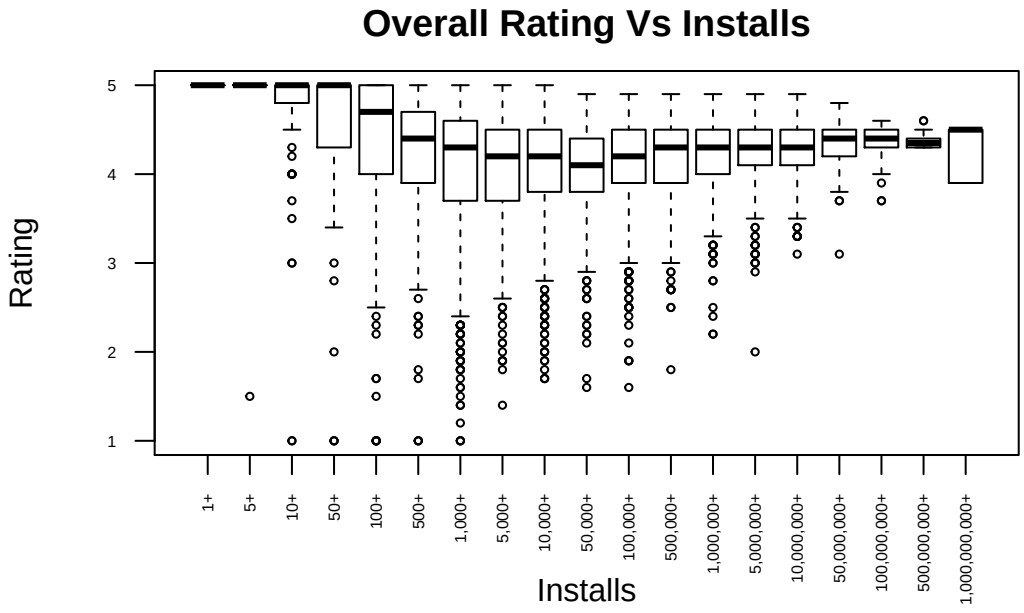
We want to investigate the association between the user's overall rating (`Rating`) of an app and the total number of installs (`Installs`). Use function `plot()` to construct a multiple boxplot of the overall rating of an app split by number of installs (`Installs`). Can you see any relationship in the plot? [3 pts]

Adjust the labels of x axis. Make sure that all levels of the variable `Installs` show in the plot. [2 extra pts]

```
# code goes here
plot(apps$Installs, apps$Rating, xlab="Installs", ylab="Rating", main="Overall Rating Vs Installs")
```



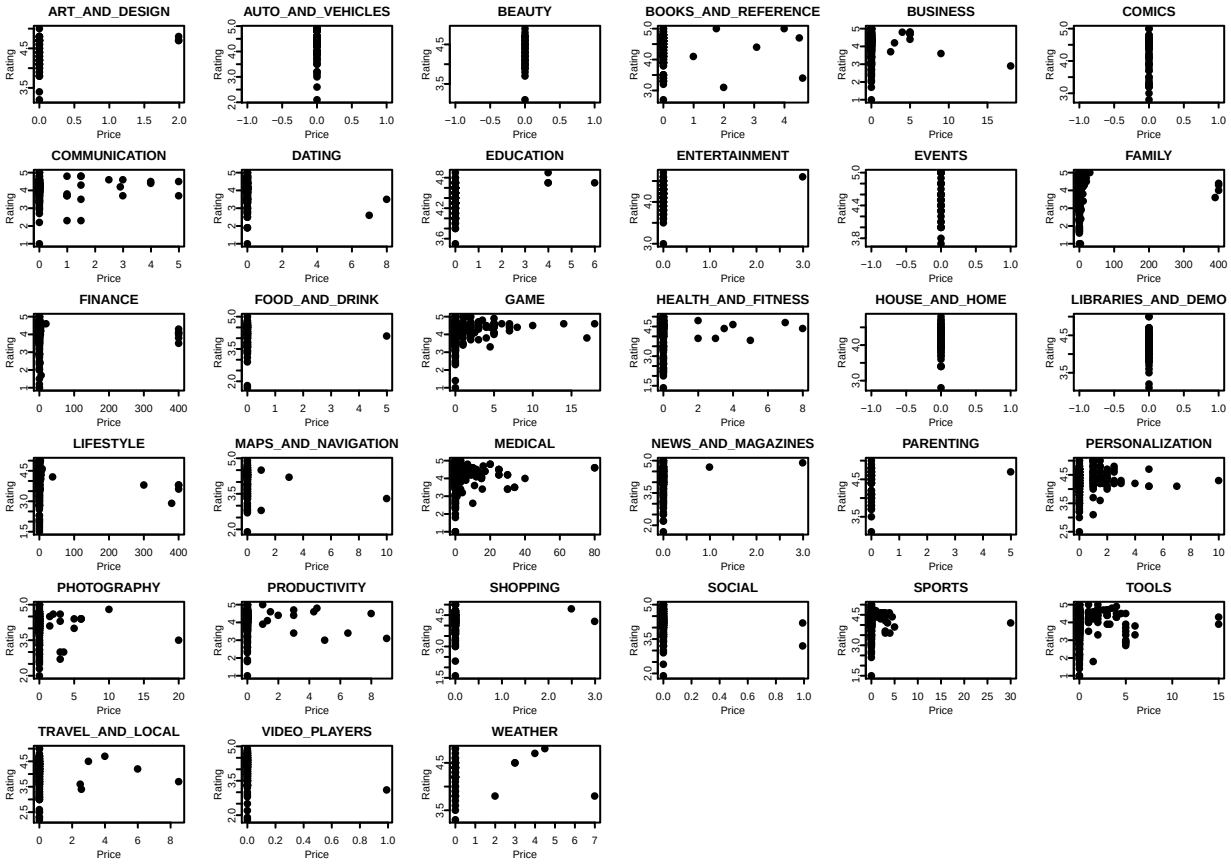

```
par(mfrow=c(1,1), mai=c(2, 1,0.5,1))
plot(apps$Installs, apps$Rating,xlab="Installs", ylab="Rating", main="Overall Rating Vs Installs", las=
```



Problem 1.3

We now investigate how the overall rating (`Rating`) is associated with the category of the app (`Category`) and its price (`Price`). Use **Split/Apply/Combine** strategy to split the dataset by `Category`. For each category, generate a plot of user's rating (`Rating`) against the app's price (`Price`). Display all plots in one figure. To receive full credit, you must use a vectorized function from the **plyr** family. Make sure your figure contains **33** subplots, with each of them corresponding to one category. [5 pts]

```
#code goes here
myplot<-function(apps.subset) {
  plot(apps.subset$Price, apps.subset$Rating, xlab="Price", ylab="Rating", pch=16,lwd=0.8,
       main=unique(apps.subset$Category), cex=0.8, cex.axis=0.5, cex.main=0.6, cex.lab=0.5,
       tcl=-0.2, mgp = c(0.6, 0.05, 0.0))
}
par(mfrow=c(6,6), mai=c(0.2, 0.15, 0.15, 0.15))
d_ply(apps, .(Category), myplot)
```



Part 2 (Basic Web Scraping, 15 + 2 pts)

In this part, we look at the voting record of the 2018 US Congress for roll call 274. The votes were compiled from http://clerk.house.gov. The raw data have been saved in the file `Roll_Call_274.xml`. We want to extract the voting results for 427 members of the House of Representatives. First, we read in data.

```
rollCall274<-readLines("Roll_Call_274.xml")
```

Problem 2.0

Check the number of lines contained in the `Roll_call_274.xml` file. There should be 485 lines. [1 pt]

```
# code goes here
length(rollCall274)
```

```
## [1] 485
```

Problem 2.1

Use the `grep()` function to find the lines in the file that correspond to the votes. Make sure `grep()` finds 427 lines. Hint: such a line starts with `<recorded-vote>`. [2 pts]

```
# code goes here
voteline_exp<-"^(<recorded-vote>)"
```

```
vote_lines<-grep(voteline_exp, rollCall274)
all_votes<-rollCall274[vote_lines]
```

Problem 2.2

Write a regular expression that will capture the ID of a member. Using it extract the ID of each member. Hint: you can use the fact that `name-id=` appears before the ID. The ID is inside a pair of quotes, and it consists of one capital latter and six digits. [2 pts]

```
# code goes here
id_exp<-"name-id=\"[A-Z][0-9]{6}\""
id.locations <- gregexpr(id_exp, all_votes)
id<- regmatches(all_votes, id.locations)
id<-substring(id, first = 10, last = nchar(id)-1)
```

Problem 2.3

Using a regular expresssion extract the name of each member. Make sure that you can extract all names for 427 members. [2 extra pts]

```
# code goes here
name_exp<-"unaccented-name=\"[a-zA-z]+.+\"\\sparty"
name.locations<-gregexpr(name_exp, all_votes)
name<-regmatches(all_votes, name.locations)
name<-substring(name, first = 18, last = nchar(name)-7)
```

Problem 2.4

Extract the party of each member by using a regular expression. There should be 193 Democrats and 234 Republicans. [2 pts]

```
# code goes here
party_exp<-"party=\"[RD]\""
party.locations<-gregexpr(party_exp, all_votes)
party<-regmatches(all_votes, party.locations)
party<-substring(party, first = 8, last = 8)
```

Problem 2.5

Extract the state for each member by using a regular expression. [2 pts]

```
# code goes here
state_exp<-"state=\"[A-Z]{2}\""
state.locations<-gregexpr(state_exp, all_votes)
state<-regmatches(all_votes, state.locations)
state<-substring(state, first = 8, last = nchar(state)-1)
```

Problem 2.6

Last, use a regular expression to extract the vote. [2 pts]

```
# code goes here
vote_exp<-"<vote>[a-zA-z]+(\\s[a-zA-z]+)*</vote>"
vote.locations<-gregexpr(vote_exp, all_votes)
vote<-regmatches(all_votes, vote.locations)
vote<-substring(vote, first = 7, last = nchar(vote)-7)
```

Problem 2.7

Make the extracted vote as a factor, and check its levels. Make a new variable called `numeric.vote`, which takes value 1 if the member voted "Yes (Aye)", 0 if the vote is "No", and -1 for "Not Voting". [2 pts]

```
# code goes here
vote<-as.factor(vote)
table(vote)
```

```
## vote
##        Aye         No Not Voting
##        225        180         22
```

```
numeric.vote<-rep(-1, length(vote))
numeric.vote[vote=="Aye"]=1
numeric.vote[vote=="No"]=0
```

Problem 2.8

Create a dataframe `rollCall274`, which contains the following five variables: `name`, `state`, `party`, `vote`, `numeric.vote`. Use `id` to name the rows of this dataframe. [2 pts]

```
# code goes here
rollCall274<-data.frame(name, state, party, vote, numeric.vote)
rownames(rollCall274)<-id
head(rollCall274, n=10)
```

```
##              name state party vote numeric.vote
## A000374   Abraham    LA     R  Aye            1
## A000370     Adams    NC     D   No            0
## A000055  Aderholt    AL     R  Aye            1
## A000371   Aguilar    CA     D   No            0
## A000372     Allen    GA     R  Aye            1
## A000367     Amash    MI     R   No            0
## A000369    Amodei    NV     R  Aye            1
## A000375 Arrington    TX     R  Aye            1
## B001291     Babin    TX     R  Aye            1
## B001298     Bacon    NE     R  Aye            1
```

Part 3 (Bootstrap, 4 + 2 pts)

We consider the `strikes` data which appears in Lecture 6. The data set is about strikes in 18 countries over 35 years (compiled by Bruce Western, in the Sociology Department at Harvard University). The measured variables are:

- **country, year** – country and year of data collection

- **strike.volume** – days on strike per 1000 workers
- **unemployment** – unemployment rate
- **inflation** – inflation rate
- **left.parliament** – leftwing share of the goverment
- **centralization** – centralization of unions
- **density** – density of unions

In this problem, we *only* look at the strikes in Italy. On this subset, we run a simple linear regression using `strike.volume` as the response (Y) and `left.parliament` as the predictor (X). Our model is

$$Y = \beta_0 + \beta_1 X + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2).$$

```
strikes<-read.csv("strikes.csv", header = T)
italy.strikes<-strikes[strikes$country == "Italy", ]
dim(italy.strikes)
```

```
## [1] 35  8
```

```
lm.fit<- lm(strike.volume ~ left.parliament, data = italy.strikes)
round(lm.fit$coefficients,3)
```

```
##     (Intercept) left.parliament
##        -738.745          40.291
```

Problem 3.1

Denote our estimate of β_1 as $\hat{\beta}_1$, which is 40.291 according to the above analysis. Use Bootstrap to estimate the variance of $\hat{\beta}_1$. Here, you may draw 100 Bootstrap samples. [4 pts]

```
B=100
b.beta<-rep(0, B)
for(b in 1:B) {
  index<-sample.int(35, 35, replace = T)
  temp.sample<-italy.strikes[index,]
  temp.fit<-lm(strike.volume ~ left.parliament, data =temp.sample)
  b.beta[b]<-temp.fit$coefficients[2]
}
b.sd<-sd(b.beta)
```

Problem 3.2

Construct a 95% confidence interval of β_1 based on the Bootstrap result. [2 extra pts]

```
# The lower bound of the confidence interval
2*lm.fit$coefficients[2]- quantile(b.beta, probs= 0.975)
```

```
## left.parliament
##       -1.760943
```

```
# The upper bound of the confidence interval
2*lm.fit$coefficients[2]- quantile(b.beta, probs= 0.025)
```

```
## left.parliament
##        79.27543
```