STAT 385 Statistics Programming Methods - Spring 2021

Final Exam

Due: Tuesday May 11, 2021 11:59 pm US Central Time

Created by Course staff Grading Rubric (per question):
2 points if complete and correct
1 point if incomplete or incorrect
0 points if no attempt made

Retrieving your work

This .Rmd file is written in RMarkdown. The .Rmd file is rendered as an .html file for easier viewing. This is located in the **exams** directory within the **stat385-sp21-course-content** repo, i.e. **stat385-sp21-course-content/exams** in GitHub. It is always recommended that you **pull** (or refresh) the **stat385-sp21-course-content** repo to ensure that you have the most updated version of all course content. After pulling (or refreshing) the **stat385-sp21-course-content** repo, the exam file will be in the exam directory.

Submitting your work

In your individual student repo (named as your netID), you are to submit ***two*** files:

a. Your reproducible document file (.Rmd) which should be saved as final-netID.Rmd. For example, my final file would be saved as final-kinson2.Rmd.

b. Your rendered reproducible document file (.html) which should be saved as final-netID.html. For example, my final file would be saved as final-kinson2.html.

You have an unlimited number of submissions, but only the latest proper submission (commit and push) will be viewed and graded. **Remember the .Rmd file needs to render properly to .html before submitting.**

Students are expected to complete all problems.

The following problems should be completed by you as an individual. If any problem asks you a particular question, be sure to answer it completely (with code, written sentences, or both). Written sentences should not appear in code chunks or code cells. Written sentences should appear in Markdown syntax unless specifically stated otherwise.

Do not change anything in this file above the double line.

Use only R and RMarkdown for this entire assignment. Use the URLs to access the data (if any). No local files allowed. The `subset()` function is never allowed in this course. Only use `set.seed()` when instructed.

If you have not installed the "tidyverse" package on your computer, then you need to install that package in order to complete the exam successfully. Do not include `install.packages()` as executable code in your submitted exam files. Doing so will result in a grade of 0 on the exam.

#1 Using Markdown syntax, make a numbered list with your first name in normal text as the first item and your last name in italic text as the second item.

1. Yeonchul
2. *Choi*

#2 Merge conflicts are the erroneous result of files being changed in the local remote repo without first pulling the repo in GitHub.

If the text in bold is the term that makes the statement true, then write TRUE below in all caps. If the text in bold is the term that makes the statement false, then write FALSE below in all caps and write the correct term in bold text in a new line under FALSE.

TRUE

#3 Beginning with the `chickwts` dataset, answer the following question in Markdown syntax and complete sentence(s):

- How many chickens weigh over 0.50 pounds?

You must answer in Markdown syntax in a complete sentence and show code as evidence of your answer. The help documentation, `?chickwts`, may be useful in completing this problem.

```
data(chickwts)
weight = chickwts$weight
weight = weight*0.00220462
length(weight[weight>0.50])
```

```
## [1] 47
```

-47 chickens weight over 0.50 pounds.

#4 Reproduce the result using vectors, sequences, and no more than one additional base R function. Using `c()` and `seq()` are allowed in addition to the one base R function. This one base R function could be used multiple times. *The shape, height, and width of the result do not matter, but you cannot simply use brute force.*

```
"goldeneye+0010"  "goldeneye+0020"  "goldeneye+0030"  "goldeneye+0040"  "goldeneye+0050" "goldeneye+006
```

```
x=seq(10,70,length.out=7)
q4=paste0("goldeneye+00",x)
q4
```

```
## [1] "goldeneye+0010" "goldeneye+0020" "goldeneye+0030" "goldeneye+0040"
## [5] "goldeneye+0050" "goldeneye+0060" "goldeneye+0070"
```

#5 Answer the following question in Markdown syntax and complete sentence(s):

- Based on the formula below, what is the value of someone's down payment in US Dollars, d?

$$n = \frac{\ln\left(\frac{r}{\frac{m}{p}-r}+1\right)}{\ln(1+r)}$$

- $n = 360$: number of mortgage payment terms.
- $m = 1250$: the monthly payment amount (US Dollars).
- $p =?$: the loan amount (US Dollars).
- $r = 0.00475$: the monthly interest rate.
- $d = 0.2 \cdot p$: down payment (US Dollars).

You must answer in Markdown syntax in a complete sentence and show code as evidence of your answer.

```
n=360
m=1250
r=0.00475
p=(m*exp(n*log(1+r))-m)/(r*exp(n*log(1+r)))
d=0.2*p
d
```

```
## [1] 43073.71
```

-43073.71 is the value of the someone's down payment in US Dollar.

#6 Using one programming language, create one custom user-defined tool that converts measurements of units of length including: inches, centimeters, and feet. Any value should be allowed to be a decimal value. In particular, your function should be able to take three inputs: the number, original length unit, converted length unit. Your function should return one output as a single phrase containing the converted value. **For example, if the three inputs are 2.5, "inches", "feet" then your function should return "2.5 inches equals X feet", where you have computed the number X.** Convert the following:

```
length = function(x, input, output){
  if (input == "inches"){
    if (output == "centimeters")
      y = x * 2.54
    else if (output == "feet")
      y = x /12
  }
  else if (input == "centimeters"){
    if (output == "feet")
      y = x/30.48
    else if (output == "inches")
      y = x/2.54
  }
```

```
  else if (input == "feet"){
    if (output == "inches")
      y = x*12
    else if (output == "centimeters")
      y = x*30.48
  }
  if (input == output){
    y = x
  }
  paste(x, input, "equals", y, output)
}
```

i. 112 inches to feet

```
length(112,'inches','feet')
```

```
## [1] "112 inches equals 9.33333333333333 feet"
```

ii. 6.48 feet to centimeters

```
length(6.48,'feet','centimeters')
```

```
## [1] "6.48 feet equals 197.5104 centimeters"
```

#7 Beginning with R's internal `mtcars` dataset and using tidyverse functionality, use regex and string manipulation to print the very first word of each car's name, e.g., the manufacturer name of a Ford Focus SE is Ford. **You must use tidyverse functionality to complete this problem. There should be 32 names.**

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0

## Warning: package 'tibble' was built under R version 4.0.3

## Warning: package 'readr' was built under R version 4.0.3

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

```
data(mtcars)
car= rownames(mtcars)
car = word(car, 1)
car
```

```
##  [1] "Mazda"    "Mazda"    "Datsun"   "Hornet"   "Hornet"   "Valiant"
##  [7] "Duster"   "Merc"     "Merc"     "Merc"     "Merc"     "Merc"
## [13] "Merc"     "Merc"     "Cadillac" "Lincoln"  "Chrysler" "Fiat"
## [19] "Honda"    "Toyota"   "Toyota"   "Dodge"    "AMC"      "Camaro"
## [25] "Pontiac"  "Fiat"     "Porsche"  "Lotus"    "Ford"     "Ferrari"
## [31] "Maserati" "Volvo"
```

#8 Beginning with the `mtcars` dataset and using tidyverse functionality, use regex and string manipulation to print the very first word of each car's model name, e.g., the model name of a Ford Focus SE is Focus SE. **You must use tidyverse functionality to complete this problem. There should be 32 names.**

```
model = rownames(mtcars)
model = word(model, 2, -1)
model
```

```
##  [1] "RX4"         "RX4 Wag"     "710"         "4 Drive"     "Sportabout"
##  [6] NA            "360"         "240D"        "230"         "280"
## [11] "280C"        "450SE"       "450SL"       "450SLC"      "Fleetwood"
## [16] "Continental" "Imperial"    "128"         "Civic"       "Corolla"
## [21] "Corona"      "Challenger"  "Javelin"     "Z28"         "Firebird"
## [26] "X1-9"        "914-2"       "Europa"      "Pantera L"   "Dino"
## [31] "Bora"        "142E"
```

#9 Create a tibble (named as `df01`) with three columns such that column 1 contains the last 10 capitalized letters of the alphabet in repeating sequence (i.e. Q, Q, R, R, ..., Z, Z), column 2 contains 20 positive random continuous standard normal distribution deviates rounded to the hundredths place, and column 3 contains a randomized ordering of the concatenated values of columns 1 and 2. **The concatenation should have an underscore'_' sign between the values of columns 1 and 2. For example, the concatenated values might appear as "Q_1.5". There should be only 20 rows in `df01`.**

```
library(tibble)
col1 = LETTERS[-c(1:16)]
col1 = c(col1,col1)
col1 = sort(col1)

col2 = rnorm(20)
col2 = round(col2, 2)

col3 = paste0(col1,"_",col2)
col3 = sample(col3, 20, replace = FALSE)

df01 = cbind(col1, col2, col3)
df01 = as_tibble(df01)
df01
```

```
## # A tibble: 20 x 3
##    col1  col2  col3
##    <chr> <chr> <chr>
##  1 Q     -0.52 R_0.9
##  2 Q     -0.8  Z_-1.24
##  3 R     0.9   V_-0.14
##  4 R     0.68  U_-2.21
##  5 S     -0.64 Z_-0.85
```

```
##  6 S     1.01  X_0.33
##  7 T     0.39  U_-0.93
##  8 T     -1.12 W_0.01
##  9 U     -2.21 T_-1.12
## 10 U     -0.93 W_-0.57
## 11 V     -0.14 R_0.68
## 12 V     -1.59 X_0.04
## 13 W     0.01  S_-0.64
## 14 W     -0.57 Q_-0.52
## 15 X     0.33  T_0.39
## 16 X     0.04  S_1.01
## 17 Y     -0.54 Q_-0.8
## 18 Y     -0.56 V_-1.59
## 19 Z     -1.24 Y_-0.54
## 20 Z     -0.85 Y_-0.56
```

#10 Let `df02` be equal to the `df01` that was in **Problem 9**. Now, create a new `df02` that has its entire rows reordered such that the column 2 numeric values are in decreasing order. *Column 2 is determining the row order for the entire object; not just switching the order of column 2 alone.* Print the resulting `df02`. *There should be only 20 rows in `df02`.*

```
df02 = df01
df02 = df02[order(df02[,2], decreasing = TRUE), ]
df02
```

```
## # A tibble: 20 x 3
##    col1  col2  col3
##    <chr> <chr> <chr>
##  1 S     1.01  X_0.33
##  2 R     0.9   V_-0.14
##  3 R     0.68  U_-2.21
##  4 T     0.39  U_-0.93
##  5 X     0.33  T_0.39
##  6 X     0.04  S_1.01
##  7 W     0.01  S_-0.64
##  8 U     -2.21 T_-1.12
##  9 V     -1.59 X_0.04
## 10 Z     -1.24 Y_-0.54
## 11 T     -1.12 W_0.01
## 12 U     -0.93 W_-0.57
## 13 Z     -0.85 Y_-0.56
## 14 Q     -0.8  Z_-1.24
## 15 S     -0.64 Z_-0.85
## 16 W     -0.57 Q_-0.52
## 17 Y     -0.56 V_-1.59
## 18 Y     -0.54 Q_-0.8
## 19 Q     -0.52 R_0.9
## 20 V     -0.14 R_0.68
```

#11 Create two new vectors `x01` and `x02` such that `x01` coerces the second column of the tibble `df01` in **Problem 9** to be a logical vector such that any number greater than 1.30 should have a logical value of TRUE, while `x02` is a sequence of 20 elements beginning with 1.00 and ending with 1.01. Now, print both `x01` and `x02`.

```
x01=df01[,2]
x01 = x01 > 1.30
x01
```

```
##        col2
##  [1,] FALSE
##  [2,] FALSE
##  [3,] FALSE
##  [4,] FALSE
##  [5,] FALSE
##  [6,] FALSE
##  [7,] FALSE
##  [8,] FALSE
##  [9,] FALSE
## [10,] FALSE
## [11,] FALSE
## [12,] FALSE
## [13,] FALSE
## [14,] FALSE
## [15,] FALSE
## [16,] FALSE
## [17,] FALSE
## [18,] FALSE
## [19,] FALSE
## [20,] FALSE
```

```
x02 = seq(1.00,1.01, length.out=20)
x02
```

```
##  [1] 1.000000 1.000526 1.001053 1.001579 1.002105 1.002632 1.003158 1.003684
##  [9] 1.004211 1.004737 1.005263 1.005789 1.006316 1.006842 1.007368 1.007895
## [17] 1.008421 1.008947 1.009474 1.010000
```

#12 The **ui** within a Shiny app builds the output of the app such as the plot, table, and text by first recognizing the input or data values with the help of reactivity.

If the text in bold is the term that makes the statement true, then write TRUE below in all caps. If the text in bold is the term that makes the statement false, then write FALSE below in all caps and write the correct term in bold text in a new line under FALSE.

FALSE **server**

#13 Select the scenario(s) that are appropriate for using parallel programming. There may be more than one scenario that is appropriate. To select an answer, place an x within the brackets as `[x]`.

- ☐ computer has less than 1 core or less than 1 processor
- ☒ code can be made into independent repetitive chunks
- ☒ large-scale computational needs
- ☐ sequential code takes microseconds to run on a single computer

#14 Using tidyverse functionality, import the US State-level COVID-19 Historical Data from the New York Times and the Harvard's US Presidential Election Data using the data URLs covid and election - Box or election - GitHub. Now, print the structure for each dataset.

- Data information: The "covid" dataset (a .csv file) contains the number of COVID-19 cases and deaths per state as a time series. The "election" data (a .csv file) contains how each state voted in the each US Presidential Election and the number of votes each candidate received along with their political party affiliations. The candidate receiving the most votes can usually be considered as the winner of that state along with their political party. In other words, the state's political party affiliation is based on the candidate who who received the most votes for that state.

```
covid = read_delim('https://github.com/nytimes/covid-19-data/raw/master/us-states.csv', delim=',')
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   date = col_date(format = ""),
##   state = col_character(),
##   fips = col_character(),
##   cases = col_double(),
##   deaths = col_double()
## )
```

```
election = read_delim('https://uofi.box.com/shared/static/01k9up8gefdpo9t23sgu9gq5f36pylp9.csv', delim=
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   year = col_double(),
##   state = col_character(),
##   state_po = col_character(),
##   state_fips = col_double(),
##   state_cen = col_double(),
##   state_ic = col_double(),
##   office = col_character(),
##   candidate = col_character(),
##   party_detailed = col_character(),
##   writein = col_logical(),
##   candidatevotes = col_double(),
##   totalvotes = col_double(),
##   version = col_double(),
##   notes = col_logical(),
##   party_simplified = col_character()
## )
```

```
str(covid)
```

```
## tibble [23,884 x 5] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ date  : Date[1:23884], format: "2020-01-21" "2020-01-22" ...
##  $ state : chr [1:23884] "Washington" "Washington" "Washington" "Illinois" ...
##  $ fips  : chr [1:23884] "53" "53" "53" "17" ...
##  $ cases : num [1:23884] 1 1 1 1 1 1 1 1 2 ...
##  $ deaths: num [1:23884] 0 0 0 0 0 0 0 0 0 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   date = col_date(format = ""),
```

```
##   ..   state = col_character(),
##   ..   fips = col_character(),
##   ..   cases = col_double(),
##   ..   deaths = col_double()
##   .. )
```

```
str(election)
```

```
## tibble [4,287 x 15] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ year            : num [1:4287] 1976 1976 1976 1976 1976 ...
##  $ state           : chr [1:4287] "ALABAMA" "ALABAMA" "ALABAMA" "ALABAMA" ...
##  $ state_po        : chr [1:4287] "AL" "AL" "AL" "AL" ...
##  $ state_fips      : num [1:4287] 1 1 1 1 1 1 1 2 2 2 ...
##  $ state_cen       : num [1:4287] 63 63 63 63 63 63 63 94 94 94 ...
##  $ state_ic        : num [1:4287] 41 41 41 41 41 41 41 81 81 81 ...
##  $ office          : chr [1:4287] "US PRESIDENT" "US PRESIDENT" "US PRESIDENT" "US PRESIDENT" ...
##  $ candidate       : chr [1:4287] "CARTER, JIMMY" "FORD, GERALD" "MADDOX, LESTER" "BUBAR, BENJAMIN \
##  $ party_detailed  : chr [1:4287] "DEMOCRAT" "REPUBLICAN" "AMERICAN INDEPENDENT PARTY" "PROHIBITION"
##  $ writein         : logi [1:4287] FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ candidatevotes  : num [1:4287] 659170 504070 9198 6669 1954 ...
##  $ totalvotes      : num [1:4287] 1182850 1182850 1182850 1182850 1182850 ...
##  $ version         : num [1:4287] 20210113 20210113 20210113 20210113 20210113 ...
##  $ notes           : logi [1:4287] NA NA NA NA NA NA ...
##  $ party_simplified: chr [1:4287] "DEMOCRAT" "REPUBLICAN" "OTHER" "OTHER" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   year = col_double(),
##   ..   state = col_character(),
##   ..   state_po = col_character(),
##   ..   state_fips = col_double(),
##   ..   state_cen = col_double(),
##   ..   state_ic = col_double(),
##   ..   office = col_character(),
##   ..   candidate = col_character(),
##   ..   party_detailed = col_character(),
##   ..   writein = col_logical(),
##   ..   candidatevotes = col_double(),
##   ..   totalvotes = col_double(),
##   ..   version = col_double(),
##   ..   notes = col_logical(),
##   ..   party_simplified = col_character()
##   .. )
```

#15 Beginning with the imported "covid" data, answer the following question:

- Which states are in the top 10 for the largest number of positive cases of COVID-19 (at the date of May 07, 2021)?

Your question should be answered in complete sentence(s) in Markdown syntax, and the resulting tidyverse code and data print out should serve as evidence.

```
covid = covid[covid$date == "2020-05-07", ]
covid = covid[,c("state","cases")]
covid = covid %>%
  group_by(state) %>%
  summarise_all(funs(sum))
```

```
## Warning: `funs()` is deprecated as of dplyr 0.8.0.
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
covid = covid[order(covid$cases, decreasing = TRUE), ]
head(covid,10)
```

```
## # A tibble: 10 x 2
##    state          cases
##    <chr>          <dbl>
##  1 New York      332931
##  2 New Jersey    133635
##  3 Massachusetts  73721
##  4 Illinois       70802
##  5 California     62481
##  6 Pennsylvania   56149
##  7 Michigan       45643
##  8 Florida        38820
##  9 Texas          36682
## 10 Connecticut    31784
```

-New York, New Jersey, Massachusetts, Illinois, California, Pennsylvania, Michigan, Florida, Texas, Connecticut are the top 10 for the largest number of positive cases of COVID-19 at the date of May 07, 2021.

#16 Beginning with the imported "election" data, answer the following question:

- In the year 2000 election, who earned the most votes and how many votes did they have?

Your question should be answered in complete sentence(s) in Markdown syntax, and the resulting tidyverse code and data print out should serve as evidence.

```
election = election[election$year == "2000", ]
election = election[,c("candidate","candidatevotes")]
election = election %>%
  group_by(candidate) %>%
  summarise_all(funs(sum))
election = election[order(election$candidatevotes, decreasing = TRUE), ]
head(election,10)
```

```
## # A tibble: 10 x 2
##    candidate                      candidatevotes
##    <chr>                                   <dbl>
##  1 "GORE, AL"                           50996062
##  2 "BUSH, GEORGE W."                    50456169
##  3 "NADER, RALPH"                        2542163
##  4 "BROWNE, HARRY"                        374630
##  5 "BUCHANAN, PATRICK \"\"PAT\"\""        355412
##  6  <NA>                                  349547
##  7 "BLANK VOTE/SCATTERING"                169238
##  8 "NOT DESIGNATED"                       151723
##  9 "PHILLIPS, HOWARD"                      84457
## 10 "HAGELIN, JOHN"                         80274
```

-Gore, Al earned the most votes and he got 50996062 votes.

#17 In a **repeat** loop, the grouped expressions execute until the condition is TRUE.

If the text in bold is the term that makes the statement true, then write TRUE below in all caps. If the text in bold is the term that makes the statement false, then write FALSE below in all caps and write the correct term in bold text in a new line under FALSE.

TRUE

#18 Based on the code chunk below, answer the following question.

- What does the `do.call()` actually do in this code?

```
list0 <- list()
for (i in 1:10){
  size <- sample(100,1)
  list0[[i]] <- data.frame(a=sample(letters, size, replace=TRUE))
}
list0
mat0 <- do.call("rbind", list0)
mat0
```

Your question should be answered in complete sentence(s) in Markdown syntax. No coding (on your part) is required in this problem.

- The `do.call()` function stacked each component in `list0` on top of each other for each index `i`.