Assignment Three

Jiale Zha

11/16/2022

Question One

Part (1)

I'm using a Mac, so could complete the work from my terminal. My final link for github repository is https://github.com/JialeZha/Stats506.

Git Initialization

First of all, in the terminal, we change the current directory to the *hw3* folder. Then, use the command `git init` to initialize this folder as a Git project.

Path to Git Configuration File

Basically, we use the command `git config --list --show-origin` to list all the Git configuration files and their directories. The result is shown in the Figure 1 below.

```
zzl@ZZLdeMacBook-Pro hw3 % git config --list --show-origin
file:/Library/Developer/CommandLineTools/usr/share/git-core/gitconfig   credential.helper=osxkeychain
file:/Users/JialeZha/.gitconfig user.name=JialeZha
file:/Users/JialeZha/.gitconfig user.email=jiale.zha@gmail.com
file:.git/config        core.repositoryformatversion=0
file:.git/config        core.filemode=true
file:.git/config        core.bare=false
file:.git/config        core.logallrefupdates=true
file:.git/config        core.ignorecase=true
file:.git/config        core.precomposeunicode=true
zzl@ZZLdeMacBook-Pro hw3 %
```

Figure 1: Git Configuration File List

Global Git Configuration

```
zzl@ZZLdeMacBook-Pro hw3 % cat /Users/JialeZha/.gitconfig
[user]
        name = JialeZha
        email = jiale.zha@gmail.com
zzl@ZZLdeMacBook-Pro hw3 %
```

Figure 2: Global Git Configuration File

We could see that the **global level configuration file** is located at '`/Users/JialeZha/.gitconfig`', while the **project level one** is located at a subfolder of current directory '`.git/config`', whose absolute path

should be '`/Users/JialeZha/Daily/Umich/STAT506/hw/hw3/.git/config`'.

The including of user name and email to the global configuration could be completed through commands '`git config --global user.name JialeZha`' and '`git config --global user.email jiale.zha@gmail.com`'. Since we have set that before, we do not execute them again.

To dsiplay the global configuration file, we could use the `cat` command, the result is shown in the Figure 2 above.

Gitignore

We use the command '`touch .gitignore`' and `echo` to create the .gitignore file and write the path to .Rhistory files into it. The result is shown in the following Figure 3

```
zzl@ZZLdeMacBook-Pro hw3 % echo $'~/.Rhistory\nhw3.pdf\n506hw3.pdf' > .gitignore
zzl@ZZLdeMacBook-Pro hw3 % cat .gitignore
~/.Rhistory
hw3.pdf
506hw3.pdf
zzl@ZZLdeMacBook-Pro hw3 %
```

Figure 3: Gitignore

Part (2)

The function for reading data and creating dataframe is in the following chunk. Basically, we just need two parameters `skip` and `nrows` of `read.table()` function to control the rows we want to read. We also paste the FIPS code and census tract code together for each census tract to get their unique tract code.

```
read_business_academic <- function(n1, n2) {
  # Read Column Name
  col_name <- read.table('data/2020_Business_Academic_QCQ.txt',
                         sep=',', quote="\"", nrows=1, header=TRUE) %>%
    colnames()

  # Read Data
  academic_table <- read.table('data/2020_Business_Academic_QCQ.txt',
                         fileEncoding="latin1", sep=',', quote="\"",
                         skip=n1, nrows=n2 - n1, header=FALSE,
                         colClasses = rep('character', length(col_name)))

  # Polish Field Name
  colnames(academic_table) <- gsub('\\W+', '_', col_name) %>% tolower()

  # Fields of Interest
  cols <- c('state', 'county_code', 'employee_size_5_location',
            'sales_volume_9_location', 'census_tract')

  # Paste FIPS Code and Census Tract Code
  academic_table$census_tract <- paste(academic_table$fips_code,
                                       academic_table$census_tract, sep="")

  # Numeric Fields
  numeric_field <- c('employee_size_5_location',
            'sales_volume_9_location')
  academic_table[, numeric_field] <- sapply(academic_table[, numeric_field],
```

```
                                          as.numeric)

  # Result
  return(academic_table[, cols] %>% na.omit())
}
```

Part (3)

In the following chunk, we use `for` loop to read records in small batches and combine them together.

```
business_academic_table <- data.frame()
for (i in 0:14) {
  business_academic_table <- business_academic_table %>%
    rbind(read_business_academic(20000*i + 1, 20000*(i + 1) + 1))
}
business_academic_table %>% head()
```

```
##   state county_code employee_size_5_location sales_volume_9_location
## 1    AL         073                        3                      98
## 2    AL         073                        3                     165
## 3    AL         073                        6                    1793
## 4    AL         073                        3                     586
## 5    AL         073                       15                     738
## 6    AL         073                        2                     297
##   census_tract
## 1  01073010706
## 2  01073012401
## 3  01073000800
## 4  01073004500
## 5  01073010804
## 6  01073014404
```

The `df1` table could be generated through the `group_by` operation, as shown in the chunk below.

```
df1 <- business_academic_table %>%
  group_by(census_tract) %>%
  summarise(employee_size=sum(employee_size_5_location),
            sales_volume=sum(sales_volume_9_location))
df1 %>% head()
```

```
## # A tibble: 6 x 3
##   census_tract employee_size sales_volume
##   <chr>                <dbl>        <dbl>
## 1 01001020100            214        34928
## 2 01001020200           1494        75970
## 3 01001020300            877        86037
## 4 01001020400            585        76889
## 5 01001020500           3740       404804
## 6 01001020600            909       459088
```

Part (4)

Connect to MySQL

We use the library `RMySQL` to connect our R studio to MySQL, this could be done by the following three lines of code. Note that, we hide the password variable for security reason.

```
mysqlconnection = dbConnect(RMySQL::MySQL(), host='localhost', port=3306,
                            user='root', password=db_password)
db_connect = dbConnect(mysqlconnection)
```

Create Database

The creation of database `Hw3db` could be realized through the `dbSendQuery()` function.

```
create_db_query <- "CREATE DATABASE Hw3db;"
db_result <- dbSendQuery(db_connect, create_db_query)
dbClearResult(db_result)
```

df1 Table

Have created the database, we reconnect to the MySQL server and specify the database we want to use. Then, write a query to create the `df1` table to hold our dataframe `df1`.

```
mysqlconnection = dbConnect(RMySQL::MySQL(), host='localhost', port=3306,
                            user='root', password=db_password, dbname='Hw3db')
db_connect = dbConnect(mysqlconnection)
```

```
create_table_query <- "CREATE TABLE df1 (
                         census_tract char,
                         employee_size int,
                         sales_volume int
                       );"
table_result <- dbSendQuery(db_connect, create_table_query)
dbClearResult(table_result)
```

Function `dbWriteTable()` could help us write the R dataframe to MySQL.

```
dbWriteTable(db_connect, 'df1', df1, overwrite=TRUE)
```

```
## [1] TRUE
```

Part (5)

We display the result of query through `dbGetQuery()` function, as shown below.

```
top_sales_query <- "SELECT census_tract, sales_volume
                      FROM df1
                     ORDER BY sales_volume DESC
                     LIMIT 10;"
top_sales <- dbGetQuery(db_connect, top_sales_query)
top_sales
```

```
##    census_tract sales_volume
## 1   02016000200      6860101
## 2   02020001900      5973563
## 3   01103005101      4777101
## 4   05131000100      4646421
## 5   01073004500      4459034
## 6   02020002502      4351429
## 7   01089000201      3854197
## 8   02020001000      3273042
## 9   05119004400      2949919
## 10  05119004800      2896424
```

Part (6)

The commit could be made by command `git add` and `git commit`, we use the following commands to make it along with a message. Notice that, given we execute the sql query in R, the only thing we want to track by git is this Rmd file and its figures, so we only add them to git.

```
git add hw3.Rmd fig/
```

```
git commit -m 'HW3 until Q1 Part 5'
```

The result of commit is shown in the following Figure 4

```
zzl@ZZLdeMacBook-Pro hw3 % git add hw3.Rmd fig/
zzl@ZZLdeMacBook-Pro hw3 % git commit -m 'HW3 until Q1 Part 5'
[master (root-commit) 5f070a3] HW3 until Q1 Part 5
 5 files changed, 346 insertions(+)
 create mode 100644 fig/.DS_Store
 create mode 100644 fig/git_config_origin.png
 create mode 100644 fig/global_config.png
 create mode 100644 fig/ignore.png
 create mode 100644 hw3.Rmd
zzl@ZZLdeMacBook-Pro hw3 %
```

Figure 4: First Commit

To create and switch to a new branch, the command `git checkout` is useful, as shown in the next line.

```
git checkout -b HW3_2
```

```
zzl@ZZLdeMacBook-Pro hw3 % git checkout -b HW3_2
Switched to a new branch 'HW3_2'
```

Figure 5: New Branch

Part (7)

Field Name

We first read the associated data to find reasonable name of those fields.

```
# Read Field Name
field_name_table <- read_xlsx('data/historicalconsumerlayout.residential.xlsx', skip=1)

# Fields of Interest
field_list <- c("FIELD19", "FIELD20", "FIELD22", 'FIELD45', 'FIELD64', 'FIELD65')

# Meaning of the above Fields
field_name_table[field_name_table$`Header Field Name` %in% field_list, ]
```

```
## # A tibble: 6 x 7
##   Order `Header Field Name` InfoGroup Field Defini~1 Outpu~2 Outpu~3 Type  Notes
##   <dbl> <chr>               <chr>                  <chr>   <chr>   <chr> <lgl>
## 1    19 FIELD19             WEALTH_FINDER_SCORE    Modele~ 0-9999~ Char  NA
## 2    20 FIELD20             FIND_DIV_1000          FIND i~ 5-500 ~ Char  NA
## 3    22 FIELD22             ESTMTD_HOME_VAL_DIV_1000 Estima~ 5-9999~ Char  NA
## 4    45 FIELD45             STATE                  Standa~ AL, AK~ Char  NA
```

```
## 5    64 FIELD64             CENSUS2010COUNTYCODE     County~ 001, 0~ Char  NA
## 6    65 FIELD65             CENSUS2010TRACT          A numb~ 000100~ Char  NA
## # ... with abbreviated variable names 1: `InfoGroup Field Definition`,
## #   2: `Output Field Description`, 3: `Output Value Set`
```

Script to Extract Data

The following chunk of code displays the function we use to extract house data. We source it to a script called `load_house_data.R`, so won't execute the following chunk.

```
# Fields of Interest
field_list <- c("FIELD19", "FIELD20", "FIELD22", 'FIELD45', 'FIELD64',
                'FIELD65')

# Meaning of the above Fields
field_name <- c('household_wealth', 'household_income', 'home_valuation',
                'state', 'county_code', 'census_tract')

load_house_data <- function(path) {
  # Read House Data
  house_table <- read.csv(path, colClasses = rep('character', 66))[, field_list]

  # Rename Fields of Interest
  colnames(house_table) <- field_name

  # Get Unique Census Tract Code
  house_table$census_tract <- paste('01', house_table$county_code,
                                    house_table$census_tract, sep='')

  # Numeric Fields
  numeric_field <- c('household_wealth', 'household_income', 'home_valuation')
  house_table[, numeric_field] <- sapply(house_table[, numeric_field],
                                         as.numeric)

  # Result
  return(house_table[house_table$home_valuation != 0, ])
}
```

Extract Data

Then, we use the function in the script and load the house data.

```
source('load_house_data.R')
house_table <- load_house_data('data/AL.csv')
```

Summarize Statistics

The `df2` table could also be gotten by the `group_by()` function as shown below.

```
df2 <- house_table %>%
  group_by(census_tract) %>%
  summarise(household_income=sum(household_income),
            household_wealth=sum(household_wealth),
            home_valuation=sum(home_valuation))
df2 %>% head()
```

```
## # A tibble: 6 x 4
##   census_tract household_income household_wealth home_valuation
##   <chr>                  <dbl>            <dbl>          <dbl>
## 1 01001020100            55065          1502990         161241
## 2 01001020200            36580          1207547         133304
## 3 01001020300            92678          2746420         257839
## 4 01001020400           142272          3894904         351842
## 5 01001020500           407105          8988586        1085285
## 6 01001020600           110231          3677971         337405
```

Part (8)

The import of `df2` to MySQL could be done by `dbWriteTable()` function as well.

```
dbWriteTable(db_connect, 'df2', df2, overwrite=TRUE)
```

Part (9)

Until now, we only changed this Rmd file, so just add it and then commit. The commit command is shown in the next two lines.

```
git add hw3.Rmd
```

```
git commit -m 'HW3 until Q1 Part 8'
```

```
zzl@ZZLdeMacBook-Pro hw3 % git add hw3.Rmd
zzl@ZZLdeMacBook-Pro hw3 % git commit -m 'HW3 until Q1 Part 8'
[HW3_2 27f6571] HW3 until Q1 Part 8
 1 file changed, 5 insertions(+), 4 deletions(-)
zzl@ZZLdeMacBook-Pro hw3 %
```

Figure 6: Second Commit

The `git log` command will display the information of history commit, and the `HEAD` parameter means we track the history information from the very beginning of each branch.

```
git log HEAD
```

```
zzl@ZZLdeMacBook-Pro hw3 % git log HEAD
commit 27f6571a1b45847681010cd64848e4876eea706a (HEAD -> HW3_2)
Author: JialeZha <jiale.zha@gmail.com>
Date:   Fri Nov 18 18:20:50 2022 -0500

    HW3 until Q1 Part 8

commit 5f070a3f3ae709565a4d3c55c1c3d688f0b30020 (master)
Author: JialeZha <jiale.zha@gmail.com>
Date:   Fri Nov 18 18:18:07 2022 -0500

    HW3 until Q1 Part 5
zzl@ZZLdeMacBook-Pro hw3 %
```

Figure 7: Git Log (1)

Part (10)

Census API Key

To get the census data, we just do the same thing as we did in HW2. First, set up the `census_api_key` and find the state code for Alabama.

```
census_api_key('4b219ac6667bd109a2b29199c753dc2cdd268da7', install = TRUE,
               overwrite=TRUE)
```

```
## Your original .Renviron will be backed up and stored in your R HOME directory if needed.
```

```
## Your API key has been stored in your .Renviron and can be accessed by Sys.getenv("CENSUS_API_KEY").
## To use now, restart R or run `readRenviron("~/.Renviron")`
```

```
## [1] "4b219ac6667bd109a2b29199c753dc2cdd268da7"
```

```
readRenviron("~/.Renviron")
print(lookup_code(state = 'Alabama'))
```

```
## [1] "The code for Alabama is '01'."
```

Field Name

Those variables for different race groups could be found by the following chunk of code.

```
# Variable list for data of 2010
census_var = load_variables(year = 2010, dataset = 'sf1', cache=T)

# Variables for population of differnet race groups
race_field <- census_var[census_var$concept == 'RACE OF HOUSEHOLDER', c('name', 'label')]
race_field
```

```
## # A tibble: 8 x 2
##   name    label
##   <chr>   <chr>
## 1 H006001 Total
## 2 H006002 Total!!Householder who is White alone
## 3 H006003 Total!!Householder who is Black or African American alone
## 4 H006004 Total!!Householder who is American Indian and Alaska Native alone
## 5 H006005 Total!!Householder who is Asian alone
## 6 H006006 Total!!Householder who is Native Hawaiian and Other Pacific Islander ~
## 7 H006007 Total!!Householder who is Some Other Race alone
## 8 H006008 Total!!Householder who is Two or More Races
```

Census Data

The demographics data at tract census level could be downloaded by the following code.

```
# Download Original Data
tract_population = get_decennial(geography = 'tract',
                                 variables = race_field$name,
                                 year=2010, state='01', geometry = FALSE)
```

```
## Getting data from the 2010 decennial Census
```

```
## Using Census Summary File 1
```

```
# Spread the Dataset
tract_population = tract_population[c('GEOID', 'variable', 'value')]
tract_population = tract_population %>% spread(variable, value)
```

```
# Rename Fields
colnames(tract_population) <- c('census_tract', 'Total', 'White', 'Black',
                                'Indian', 'Asian', 'Hawaiian', 'other',
                                'two_races')

# Group by Census Tract
df3 <- tract_population %>%
  group_by(census_tract) %>%
  summarise(Total=sum(Total), White=sum(White), Black=sum(Black),
            Indian=sum(Indian), Asian=sum(Asian), Hawaiian=sum(Hawaiian),
            other=sum(other), two_races=sum(two_races))
df3 %>% head()
```

```
## # A tibble: 6 x 9
##   census_tract Total White Black Indian Asian Hawaiian other two_races
##   <chr>        <dbl> <dbl> <dbl>  <dbl> <dbl>    <dbl> <dbl>     <dbl>
## 1 01001020100    693   599    73      9     3        0     2         7
## 2 01001020200    743   311   417      2     1        0     7         5
## 3 01001020300   1256   996   222      4     8        2    12        12
## 4 01001020400   1722  1627    63      6     6        2     6        12
## 5 01001020500   4082  3349   553     19    79        5    24        53
## 6 01001020600   1311  1028   231      4     4        1    25        18
```

Write to MySQL

Again, we use `dbWriteTable()` function to import the above data into MySQL database.

```
dbWriteTable(db_connect, 'df3', df3, overwrite=TRUE)
```

Part (11)

We use the following query to combine those three tables, `df1`,`df2` and `df3`, together. We'll use the average home value per sale as the target variable. For the key effect, we use the cut-off of the ratio of White people in each census tract as the random effect, while the ratio of Black, Asian and Indian people as the fixed effect. The query to get them is shown below.

```
combine_table_query <- "SELECT d1.census_tract,
                               1000*d2.home_valuation/d1.sales_volume as avg_home_value,
                               ROUND(d3.White/d3.Total, 1) as white_ratio,
                               d3.Black/d3.Total as black_ratio,
                               d3.Asian/d3.Total as asian_ratio,
                               d3.Indian/d3.Total as indian_ratio
                          FROM df1 d1
                          JOIN df2 d2
                            ON d1.census_tract = d2.census_tract
                          JOIN df3 d3
                            ON d1.census_tract = d3.census_tract;"
combine_table <- dbGetQuery(db_connect, combine_table_query)
combine_table %>% head()
```

```
##   census_tract avg_home_value white_ratio black_ratio asian_ratio indian_ratio
## 1  01001020100      4616.3823         0.9  0.10533911 0.004329004  0.012987013
## 2  01001020200      1754.6926         0.4  0.56123822 0.001345895  0.002691790
## 3  01001020300      2996.8386         0.8  0.17675159 0.006369427  0.003184713
## 4  01001020400      4575.9732         0.9  0.03658537 0.003484321  0.003484321
```

```
## 5  01001020500      2681.0135         0.8  0.13547281 0.019353258  0.004654581
## 6  01001020600       734.9462         0.8  0.17620137 0.003051106  0.003051106
```

Part(12)

Commit

Similar with previous part, we use `git add` and `git commit` to make this commit.

```
git add hw3.Rmd fig/
```

```
git commit -m 'HW3 until Q1 Part 11'
```

```
zzl@ZZLdeMacBook-Pro hw3 % git add hw3.Rmd fig/
zzl@ZZLdeMacBook-Pro hw3 % git commit -m 'HW3 until Q1 Part 11'
[HW3_2 06c6cbd] HW3 until Q1 Part 11
 5 files changed, 7 insertions(+), 6 deletions(-)
 create mode 100644 fig/first_commit.png
 create mode 100644 fig/git_log.png
 create mode 100644 fig/new_branch.png
 create mode 100644 fig/second_commit.png
zzl@ZZLdeMacBook-Pro hw3 %
```

Figure 8: Third Commit

The log is shown in the following Figure 8.

```
commit 06c6cbd7410f525217a2893f565a7639cc38f985 (HEAD -> HW3_2)
Author: JialeZha <jiale.zha@gmail.com>
Date:   Fri Nov 18 19:24:08 2022 -0500

    HW3 until Q1 Part 11

commit 27f6571a1b45847681010cd64848e4876eea706a
Author: JialeZha <jiale.zha@gmail.com>
Date:   Fri Nov 18 18:20:50 2022 -0500

    HW3 until Q1 Part 8

commit 5f070a3f3ae709565a4d3c55c1c3d688f0b30020 (master)
Author: JialeZha <jiale.zha@gmail.com>
Date:   Fri Nov 18 18:18:07 2022 -0500

    HW3 until Q1 Part 5
zzl@ZZLdeMacBook-Pro hw3 %
```

Figure 9: Git Log (2)

Merge

To merge the new branch to the main, we first need to switch back to the main branch and then merge the newer one. We use the following two commands and the result is shown below.

```
git checkout master
```

```
git merge HW3_2
```

```
zzl@ZZLdeMacBook-Pro hw3 % git checkout master
Switched to branch 'master'
zzl@ZZLdeMacBook-Pro hw3 % git merge HW3_2
Updating 5f070a3..06c6cbd
Fast-forward
 fig/first_commit.png  | Bin 0 -> 481357 bytes
 fig/git_log.png       | Bin 0 -> 496884 bytes
 fig/new_branch.png    | Bin 0 -> 81933 bytes
 fig/second_commit.png | Bin 0 -> 241910 bytes
 hw3.Rmd               |  22 ++++++++++++----------
 5 files changed, 12 insertions(+), 10 deletions(-)
 create mode 100644 fig/first_commit.png
 create mode 100644 fig/git_log.png
 create mode 100644 fig/new_branch.png
 create mode 100644 fig/second_commit.png
zzl@ZZLdeMacBook-Pro hw3 %
```

Figure 10: Git Merge

To reset the repository back to the older version, the command `git reset` will be helpful.

Part (13)

Basically, we fit a mixed effect model from the table we get in part (11) and for different ratios of White people, we use different slope.

```
mixed_effect <- lmer(avg_home_value ~ 1 +
                       (black_ratio +  asian_ratio + indian_ratio | white_ratio),
                     data=combine_table)
```

```
## boundary (singular) fit: see ?isSingular
```

The following result illustrates the coefficients of our model. Basically, we could not see any uniform trend in those coefficients, thus, there should not exist any racial bias in the home evaluation.

```
ranef(mixed_effect)
```

```
## $white_ratio
##     (Intercept) black_ratio asian_ratio indian_ratio
## 0    -346.83288  -2791.9078  -6373.7887   12502.0532
## 0.1    26.27483    211.5050    482.8555    -947.1121
## 0.2    48.20748    388.0568    885.9128   -1737.7015
## 0.3  -212.93622  -1714.0770  -3913.1565    7675.5708
## 0.4   465.35226   3745.9554   8551.8419  -16774.2833
```

```
## 0.5  1714.72823  13803.0826  31511.7764  -61809.6689
## 0.6   -13.35059   -107.4685   -245.3455     481.2063
## 0.7  -412.87050  -3323.4920  -7587.3769   14882.4655
## 0.8  -358.47360  -2885.6120  -6587.7224   12921.6834
## 0.9   -30.23249   -243.3629   -555.5936    1089.8017
## 1     -62.88026   -506.1685  -1155.5588    2266.6004
##
## with conditional variances for "white_ratio"
```

Question Two

Part (1)

For HPC, it is a combination or network of many computer systems or input/output devices, each of which is called a node. A job assigned to one or more nodes means it will be executed on different devices in parallel or sequentially.

For a core, typically, it is a microprocessor, which is a processing part of a system-on-chip. Basically, a computer could be a node, and it could contain several cores. So we could say that a core is a part of a node.

The log-in nodes are shared servers and are not assumed to be used for calculation purpose. Typically, those operations on files, such as moving, deleting and so on, are executed on log-in node. On the contrast, the compute node is used for those high-load processing program, such as model fitting or some computation.

Part (2)

Generally, the script for terminal job is a `.sh` file, we will submit such a file, and its content is shown below.

```
salloc --account=stats506s001f22_class --nodes=1 --ntasks-per-node=4 \
--cpus-per-task=1 --mem-per-cpu=8GB --time=3:00:00
```

Part (3)

The general format of scratch directory for a great lake user is, '`/scratch/$account$_root/$account$/user_name`', so the absolute path for my scratch directory is

`scratch/stats506s001f22_class_root/stats506s001f22_class/jialezha`

To create a symbolic link, we could use the command `ln -s`, thus for my account, I could use the following command

```
ln -s scratch/stats506s001f22_class_root/stats506s001f22_class/jialezha ~/scratch
```

Given that the symbolic link is just a pointer, the removal of it won't have any influence on the original folder.