ST370 Lab 1 Instruction

Fall 2023

Written by TAs: Siqi Cao, Yang Xu, Shuvrarghya Ghosh, and Yi Liu

September 07, 2023

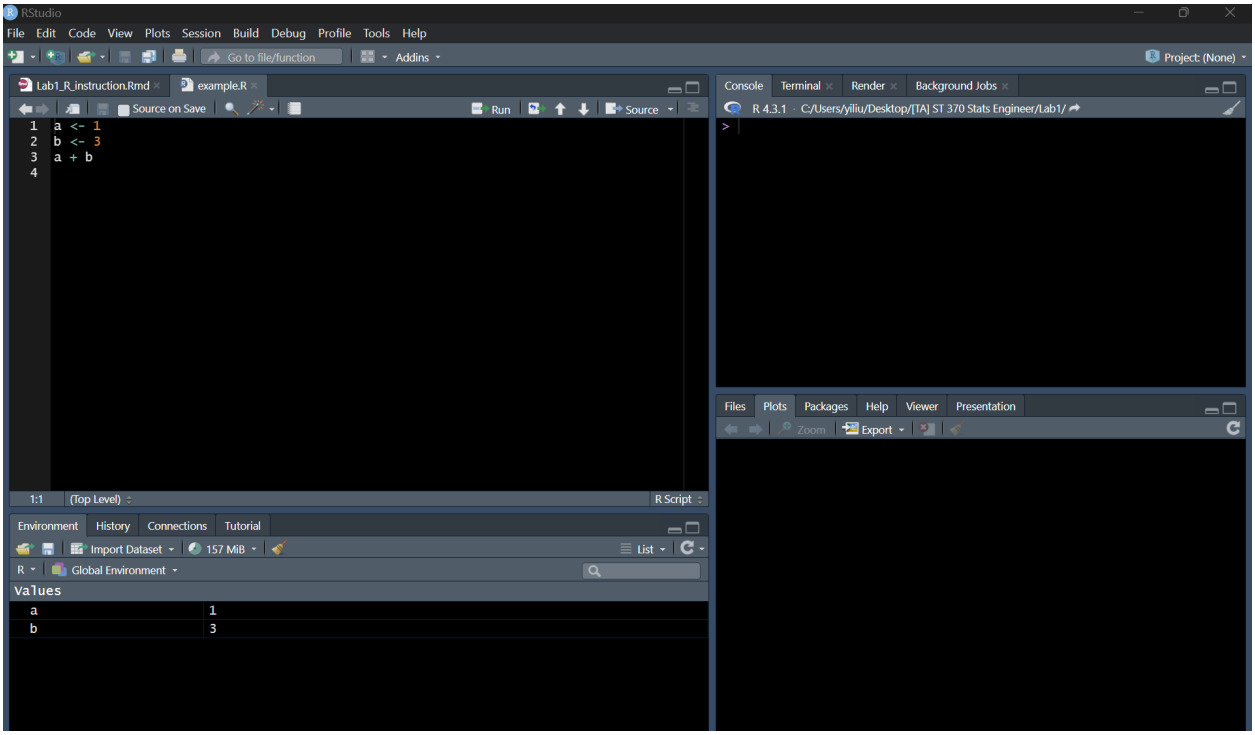
The objectives of this Lab

- Know how to read a data set into R
- Get to understand the very basic grammar of R and the use of basic R functions
- Conduct basic summarizing analysis for the data, e.g., mean, counting, etc.
- The most interesting part is to learn how to use the `sample()` function in R to draw a random sample (i.e., randomly select some data) from the entire data set

Preliminary

Overview of RStudio

Below is a typical Rstudio layout when you open it. There are usually 4 windows dividing the RStudio into 4 parts.



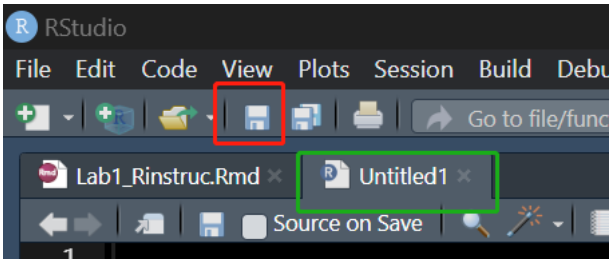
Note that the places of these panels in your RStudio might be different than what in this figure due to personal preference of the layout. For example, you may see the R Console window on the left lower corner when you first open the RStudio, where mine is on the right upper panel. For more information about the RStudio and how to personalize the layout, visit this website: Click Here.

Writing and implementing code in RStudio

When you open your RStudio for the first time, you want to know where is the place you can put your code and run them. The most common way to program in R is writing an R script.

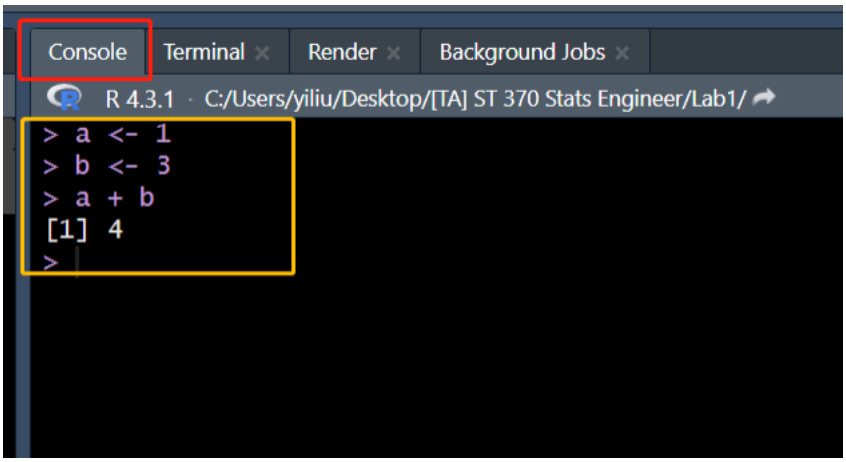
Click `File` on the upper (the menu bar) left corner of your RStudio, then select `New File`, and then select and click `R Script`. You will see a script `Untitled1.R` created in your RStudio.

You can modify the name of this `Untitled1.R` file via (1) click the save button (shown in the red box below), and choose the path on your computer you want to save the R file, and then (2) you can modify the name to what you want.



Then, you can type your code in the script file. Then select all the code you want to run (orange box below), and click the **Run** (red box below), your code can be executed and results will be printed in the Console window.

See Console window (below) for where your code being executed.



Read the data into the R environment

The first step of any data analysis in R is to read your data from external path into the R environment, and assign a name to your data you read in. Now, the following code implement the process of reading data `gdp.csv` you downloaded from the WebAssign page.

First of all, you need to put the data into the **current working dictionary**. To see this, first implement the following code

```
getwd()
```

```
## [1] "C:/Users/yiliu/Desktop/[TA] ST 370 Stats Engineer/Lab1"
```

Running the `getwd()` in your R Console window will return you the current working dictionary of R. For example, the above output tells me, the R is working under the `Lab1` subfolder of `[TA] ST 370 Stats Engineer` folder on my desktop. Therefore, I need to move the `gdp.csv` data to this folder first, in order to let R find it successfully.

If the data is not in the R working dictionary, you will see an error message showing that `In file(file, "rt") : cannot open file 'gdp.csv': No such file or directory`.

Then, the following code reads the `gdp.csv` data to the R environment

```
dat <- read.table(file = 'gdp.csv', header = T, sep = ',')
```

Here, `read.table` is a function to read a file in table format (e.g., `.csv` type file here), which will also automatically create a data frame with the same structure of the table into R. There are 3 inputs (arguments) in the `read.table` function:

- `file`: the name of the file you want to read in. It is always necessary to add single or double quotation marks to the name of the file.
- `header`: it specifies whether you want to make the first row of the data as the column names. It is a logical argument with only two possible values: `T` (or you can input `TRUE`) for true, and `F` (or you can input `FALSE`) for false. Here, we need the first row to be variable names, thus we assign `T` to `header`.
- `sep`: it specifies the

In addition, the `dat` and `<-` are necessary for the following reasons:

- `dat` is a user-specified name for the object on the right-hand side of `<-`
- and thus, `<-` links the name you assign to the object and the object
- In R, `<-` can be replaced by `=`. That is, you can also do

```
dat = read.table(file = 'gdp.csv', header = T, sep = ',')
```

You can even do

```
dat = read.table(file <- 'gdp.csv', header <- T, sep <- ',')
```

i.e., use `<-` inside the function, but this is not recommended in general. The often convention is to use `=` in the function and `<-` outside the function.

In addition, for `.csv` file, we can use another function `read.csv()` to read it with less coding

```
dat <- read.csv("gdp.csv")
```

After read the data into R, it is always a good practice to take a brief look at the dataset by printing it in R Console window. We can do the following using the `head()` function on the data.

```
head(dat)
```

```
##        state X2015Q2 X2015pop
## 1    Alabama  207303  4862058
## 2     Alaska   54256   736130
## 3    Arizona  295445  6816049
## 4   Arkansas  122492  2972524
## 5 California 2424033 39124308
## 6   Colorado  316535  5429368
```

If we want to know the exact numbers of rows and columns of the data, we can use the following functions

```
nrow(dat) # the number of rows
```

```
## [1] 50
```

```
ncol(dat) # the number of columns
```

```
## [1] 3
```

Thus, the data contains 50 rows (different states), and 3 columns: `state`, `X2015Q2`, and `X2015pop`.

Now, we can start analyzing the data!

Part 1 - Compute the "Per capita GDP", which can be obtained by dividing the GDP for each state by that state's population

In other words, it is to divide the second column of the data by the third column. So, we want to know how to extract a column from the data `dat` first. We use `$` operater as follows.

```
gdp <- dat$X2015Q2  # GDP
pop <- dat$X2015pop # population
```

In the above code, `$` extracts the `X2015Q2` and `X2015pop` columns from the `dat` object, and so `dat$X2015Q2` is a vector. The two vectors are then record to variables `gdp` and `pop`, respectively.

Now, to compute the "Per capita GDP" of each state, we do the following.

```
per_capita_GDP <- 1e6* (gdp/pop) # 1e6 is multiplied since the GDP is in millions of US dollars

# round the value to the nearest integer using the round() function
per_capita_GDP_rd <-  round(per_capita_GDP)

# create a data frame to view the results more clearly
data.frame(State = dat$state, perGDP = per_capita_GDP_rd)
```

```
##             State perGDP
## 1         Alabama  42637
## 2          Alaska  73704
## 3         Arizona  43345
## 4        Arkansas  41208
## 5      California  61957
## 6        Colorado  58301
## 7     Connecticut  72255
## 8        Delaware  70025
## 9         Florida  43857
## 10        Georgia  48711
## 11         Hawaii  55713
## 12          Idaho  39436
## 13       Illinois  59428
```

```
## 14        Indiana  49336
## 15           Iowa  53882
## 16         Kansas  50235
## 17       Kentucky  43589
## 18      Louisiana  54205
## 19          Maine  41423
## 20       Maryland  60076
## 21  Massachusetts  69892
## 22       Michigan  46606
## 23      Minnesota  60284
## 24    Mississippi  35683
## 25       Missouri  47247
## 26        Montana  44418
## 27       Nebraska  59300
## 28         Nevada  48805
## 29  New Hampshire  53849
## 30     New Jersey  64056
## 31     New Mexico  43573
## 32       New York  72997
## 33 North Carolina  50243
## 34   North Dakota  71266
## 35           Ohio  51061
## 36       Oklahoma  46441
## 37         Oregon  56319
## 38   Pennsylvania  52977
## 39   Rhode Island  53307
## 40 South Carolina  40302
## 41   South Dakota  52822
## 42      Tennessee  46576
## 43          Texas  60247
## 44           Utah  49261
## 45        Vermont  47510
## 46       Virginia  56952
## 47     Washington  62496
## 48  West Virginia  38504
## 49      Wisconsin  51474
## 50        Wyoming  68683
```

Part 2 - Compute the GDP (in millions of dollars) of all 50 states of the United States by taking the sum of the GDPs for all of the states

To get the sum, we can use the `sum()` function in R, which operates on a vector and sums all single elements in that vector.

```
sum(dat$X2015Q2)
```

```
## [1] 17674477
```

Therefore, the sum of GDPs of all states is 17,674,477 millions of US dollars.

Part 3 - Filter the list of states to a list of states that have more than 500,000 millions of US dollars

First, what we need is a list of states, so we can use `dat$state`, similar to Part 2 above, to extract the column of states from the data. The following code can further index (filter) the states satisify the criteria of GDP $>$ 500,000 millions of dollars from the `dat$state` vector.

```
dat$state[dat$X2015Q2>500000]
```

```
## [1] "California"     "Florida"        "Illinois"       "New Jersey"
## [5] "New York"       "North Carolina" "Ohio"           "Pennsylvania"
## [9] "Texas"
```

Whenever you need to filter a subset from a vector, you use `[]` after the vector, and specify the criteria inside the `[]`. Here, the `dat$X2015Q2>500000` is a logical vector, which means to compare each value in `dat$X2015Q2` to `500000`, which returns a vector of `TRUE`'s and `FALSE`'s, see below.

```
dat$X2015Q2>500000
```

```
##  [1] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
## [13]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [25] FALSE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE  TRUE FALSE  TRUE FALSE
## [37] FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE
## [49] FALSE FALSE
```

Then, `[]` filters the places of `TRUE` and apply to `dat$state`. Thus, we get the subset of states where their `X2015Q2` (GDP) values are $>$ 500,000 millions of dollars.

Part 4 - Use your software to compute the count of states with more than 500,000 millions of US dollars

To solve this question, the simplest way is to apply the `sum()` function to logical vector `dat$X2015Q2>500000`. This works because when doing numerical operations on `TRUE` and `FALSE`, R identifies `TRUE` to be number 1 and `FALSE` to be number 0. Therefore, it is easy to know, to get the number of states with more than 500,000 millions of dollars, the following code is sufficient.

```
sum(dat$X2015Q2>500000)
```

```
## [1] 9
```

Thus, in total, we have 9 states with more than 500,000 millions of US dollars. Another way (which might be more intuitive but not as direct as above) to get this is

```
rich_state <- dat$state[dat$X2015Q2>=500000]
length(rich_state)
```

```
## [1] 9
```

So, at first, we assign a variable `rich_state` for those states we GDP more than 500,000 millions of dollars, and then we apply the `length()` function on it, which returns the length of this vector. We know the length is just the number of elements of this vector, i.e., the number of "rich states" (with GDP $>$ 500,000 millions of dollars).

Remark. Another way to do this is to sort the data by GDPs from high to low (resp. low to high) and find the first (resp. last) 9 data. To do this, you can visit the `order()` or `sort()` function in R. We do not cover the details here and leave this as an exercise, though it is also pretty easy. You can type `?order` and `?sort` in R Console window to check how to use this function. In fact, you can also do `?` + the name of a function in R to get the details of the function.

Part 5 - Calculate the average GDP (in millions of dollars)

Since the data has 50 rows (as shown before), and we know how to the sum of GDPs in Part 2 above, so the first way you can do this is to use the sum divided by 50, shown below.

```
sum(dat$X2015Q2)/50
```

```
## [1] 353489.5
```

However, in R, it would be easier to use the `mean()` function to calculate the mean

```
mean(dat$X2015Q2)
```

```
## [1] 353489.5
```

Both results above are the same. The average GDP of the 50 states is then 353489.5 millions of dollars.

Part 6 - Find the difference between each state's GDP and the average GDP you calculated by subtracting the average GDP from each state's GDP. (Round your answers to the nearest integer.)

To do this, we can just use the column of the GDPs in the data (the vector `dat$X2015Q2`) subtracts the average GDP above in Part 5:

```
dat$X2015Q2 - mean(dat$X2015Q2)
```

```
##  [1] -146186.54 -299233.54  -58044.54 -230997.54 2070543.46  -36954.54
##  [7]  -93828.54 -287339.54  530245.46  142690.46 -273894.54 -288287.54
## [13]  411327.46  -26951.54 -185400.54 -207270.54 -160615.54 -100524.54
## [19] -298352.54    7479.46  120116.46  108762.46  -22707.54 -246609.54
## [25]  -66282.54 -307690.54 -241281.54 -212881.54 -281857.54  220457.46
## [31] -262679.54 1090916.46  150255.46 -299803.54  239409.46 -172400.54
## [37] -127828.54  324092.46 -297166.54 -156602.54 -308074.54  -46364.54
## [43] 1294517.46 -206792.54 -323739.54  123429.46   92606.46 -282366.54
## [49]  -56518.54 -313319.54
```

This is because when we subtract the number by a vector in R, what we are actually doing is subtract the number by every number in the vector at the same time. Thus, the code above works and returns what we need. However, we still need to (i) match these numbers to their states; (ii) round them to the nearest integer. To to these, see the following code.

```
gdp_diff <- dat$X2015Q2 - mean(dat$X2015Q2) # define the vector of difference of GDPs to their mean
gdp_diff_rd <- round(gdp_diff) # round to the nearest integer
data.frame(State = dat$state, GDP.Diff = gdp_diff_rd) # create a data frame
```

```
##             State GDP.Diff
## 1         Alabama  -146187
## 2          Alaska  -299234
## 3         Arizona   -58045
## 4        Arkansas  -230998
## 5      California  2070543
## 6        Colorado   -36955
## 7     Connecticut   -93829
## 8        Delaware  -287340
## 9         Florida   530245
## 10        Georgia   142690
## 11         Hawaii  -273895
```

```
## 12          Idaho  -288288
## 13       Illinois   411327
## 14        Indiana   -26952
## 15           Iowa  -185401
## 16         Kansas  -207271
## 17       Kentucky  -160616
## 18      Louisiana  -100525
## 19          Maine  -298353
## 20       Maryland     7479
## 21  Massachusetts   120116
## 22       Michigan   108762
## 23      Minnesota   -22708
## 24    Mississippi  -246610
## 25       Missouri   -66283
## 26        Montana  -307691
## 27       Nebraska  -241282
## 28         Nevada  -212882
## 29  New Hampshire  -281858
## 30     New Jersey   220457
## 31     New Mexico  -262680
## 32       New York  1090916
## 33 North Carolina   150255
## 34   North Dakota  -299804
## 35           Ohio   239409
## 36       Oklahoma  -172401
## 37         Oregon  -127829
## 38   Pennsylvania   324092
## 39   Rhode Island  -297167
## 40 South Carolina  -156603
## 41   South Dakota  -308075
## 42      Tennessee   -46365
## 43          Texas  1294517
## 44           Utah  -206793
## 45        Vermont  -323740
## 46       Virginia   123429
## 47     Washington    92606
## 48  West Virginia  -282367
## 49      Wisconsin   -56519
## 50        Wyoming  -313320
```

So now it is easy to copy these numbers to the WebAssign page.

Part 7 - How many states have GDP's that are above the average GDP?

This question is similar to Part 4 above. Recall that we can use the `sum()` function on a logical vector. Thus, the code below.

```
sum(dat$X2015Q2 - mean(dat$X2015Q2) > 0)
```

```
## [1] 15
```

Part 8 - Use your software to draw a random sample from the list of GDPs in the data file.

To generate a random sample in R, one of the most common ways is the `sample()` function. Use `?sample` to see more details. Now, what we want is to sample 10 different rows from the whole data set (having 50 rows), we use the code below.

```
n <- nrow(dat) # record the number of rows
samp <- sample(1:n, size=10, replace=F)
```

Let's take a look of `samp`:

```
samp
```

```
##  [1] 47  1 39 34 36  6 26 20  3 24
```

Indeed, it is a subset of 1 to 50.

Remark. In the `sample()` function, we assign `replace=F` (or `FALSE`), which means that we want to sample the vector without replacement. In contrast, if we let the `replace` argument be `T` (or `TRUE`), it might return us some elements of `1:50` several times in `samp`. You can try this to see the difference.

Now, we can use the randomly selected subset above to index the data, as follows.

```
dat.sub <- dat[samp,]
```

Here you use in `[]` a comma, which means, we want the rows with numbers in `samp` vector, and all columns.

Finally, we output the names of states, and output a `.csv` file of the subset, using code below.

```
dat.sub$state
```

```
##  [1] "Washington"   "Alabama"      "Rhode Island" "North Dakota" "Oklahoma"
##  [6] "Colorado"     "Montana"      "Maryland"     "Arizona"      "Mississippi"
```

```
write.csv(dat.sub, file="sample_gdp.csv")
```

Now, we have finished all the tasks in Lab 1. **It is then a good practice to save the R script you wrote for future references.** You can save your R file with any name you prefer in a specific location on your computer (with all other future lab R files), and you can revisit it any time later.