RstUdio ready?

Introduction to R

In this course, you will use R extensively for problem solving and calculations. R is a software package that is popular (and free!) among statisticians, economists, data scientists, and businesses for a broad variety of applications. A good command of R is essential to taking courses on Business Analytics in our UF Program. Putting "R" on your job resume also signal to your employer that you are digitally literate.

For students of AB1202, learning R will be a requirement rather than an option. Many assignments, exercises, and tests all depend on it. Don't fret if you're new to it, or have little experience about programming. We will cover basics bit by bit over the course of this semester.

The note provides initial guidance about R, including installation, introducing the R interface, and the basic commands and data structures. Commands related to the weekly topics will be explained during the course.

Installing R and R-studio

R is a free, open source statistical computing environment. In R, you must send commands to her so she will do what you want her to do.

You can download and install the base R program via

(Windows). https://cran.r-project.org/bin/windows/base/

(Mac). https://cran.r-project.org/bin/macosx/

Select the installation file based on your OS (Mac or Windows). Install the program just like what you would for any other software programs. I recommend you accept all default settings. After the installation, you will see links to open R.

NOTE: Students should install the latest version as given in the links above. If you have installed a previous version of R, please uninstall it first and then install the latest version.

The R package comes with a primitive code editor, which is the interface for you and R to communicate with each other (tell R what to do and get the analysis results back).

We won't use R directly tough. We will use a more user-friendly IDE (Integrated Development Environment). It is a platform built on top of R, thru which you can fully control R and get results &

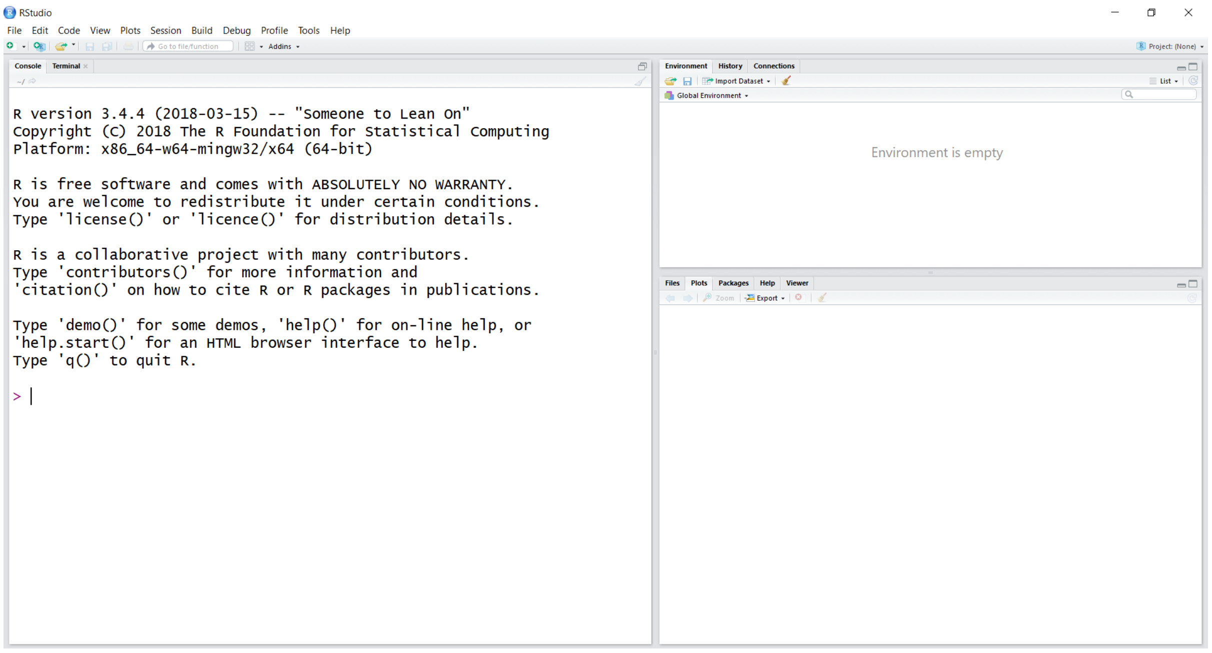
graphs in a more intuitive manner. It offers many features that make your data analytic projects and coding experience more efficient and pleasant.

Download and install RStudio https://www.rstudio.com/products/RStudio/ (choose the free “Open Source Edition”). Note that RStudio is just an IDE. So you should install R before RStudio.

Now you are all set. Click open RStudio to begin your analytic journey.

Using R-studio for the first time

You should see three panels in the RStudio environment.



```
R version 3.4.4 (2018-03-15) -- "Someone to Lean On"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> |
```

The left panel is the interactive R console. This is where you input and execute commands and get the results from R. When you create or open a R script (i.e., the text file which contains pre-stored R commends), this panel will split in two (Try “File”->”New file”->”R script”).

The top-right panel shows the existing R objects (basically, “data” of different types). You can also view the history of the previously execute commends here.

The bottom-right panel will be the place you see the plot that you generate. It is also the panel that shows the online help based on your query, files in the working directory, and the R packages that have been installed on your computer.

You can now try the following basic commands for arithmetic operations.

- You can use R for all usual arithmetic. If you type 100*7 after the “>” prompt and then hit Enter or Return key, R will return [1] 700 as the answer.

- If you executed a command without asking R to store the result, you would see the result displayed in the console, but the result will be lost after you execute another similar command. If you want to keep the result so you can refer to it later, you can ask R to assign some computer memory to keep it. This is done by the assignment operation, which in the R community is customary to use "<-" (or equivalently, "=" can also be used). For example, you can execute *weekly_salary* <-100*7. Then R will store the result of the operations on the right side of the command, under the name *weekly_salary*. You should see the object *weekly_salary* and its value in the top-right panel.
- When you execute the data object's name, the R console will show its content. For example, execute of *weekly_salary* will return 700
- When you execute *monthly_salary<- weekly_salary*4,* you build another variable *monthly_salary* thru invoking the stored value of *weekly_salary.*

General tips

- Note that functions and names in R are case sensitive and R's built-in functions are in lowercase.

  For example, the command sum(1:10) gives the sum of the numbers from 1 to 10. BUT if you execute Sum(1:10) or SUM(1:10) in R, it will give you an error message.

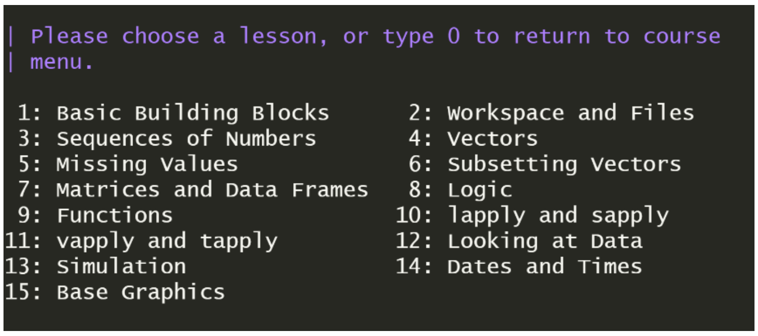
- R provides a detailed documentation about the functions that you have installed and loaded. Use help or "?" to open the document viewer when you need any help.

  For example, use help(sum) or ?sum to view the help file and examples for the function "sum()."

Learning R

In what follows, I will introduce some basic commands in R. You may have a look at them. To learn the basic stuff in a more interactive manner, I recommend you install the "swirl" package in R. This package is designed for self-learning of R in an interactive way. You can find the steps to install it here https://swirlstats.com/students.html.

For this course, it is (more than) enough to learn Sections 1, 2, 3, 4, 6, 7, & 8, under the course title "*1: R Programming: The basics of programming in R*." Try it. It is fun!



```
| Please choose a lesson, or type 0 to return to course
| menu.

 1: Basic Building Blocks      2: Workspace and Files
 3: Sequences of Numbers       4: Vectors
 5: Missing Values             6: Subsetting Vectors
 7: Matrices and Data Frames   8: Logic
 9: Functions                 10: lapply and sapply
11: vapply and tapply         12: Looking at Data
13: Simulation                14: Dates and Times
15: Base Graphics
```

1. Basic data structures in R

Basic Note:

Enter commands at the bottom console after **“>”** for immediate execution.

Entries will be shown in **blue**, and the results will in **black**.

To annotate your programming, add **“#”** in front of your scripts.

From the results, **[1]** indicates first element’s result, **[1,]**, **[2,]**, **[3,]**… will be shown when we have more than one elements for variables such as vector.

Arithmetic Operators

Symbol	Meaning
+	add
-	subtract
*	multiply
/	divide
^	exponentiate
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
<-	Assignment (equivalent to =)

Logical Operators

Symbol	Meaning
==	logical equals
!=	not equal

!	logical NOT
&	logical AND
|	logical OR
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
:	create a sequence

Addition

```
> 8+7
[1] 15
```

Division

```
> 100/50
[1] 2
```

Square root

```
> sqrt(81)
[1] 9
```

Natural log (base e)

```
> log(10)
[1] 2.302585
```

Log using base 10

```
> log10(10)
[1] 1
```

Exponentiation

```
> exp(2)
[1] 7.389056
```

Absolute Value

```
> abs(-4)
[1] 4
```

Sequential Operations

```
> (2+5)*2^2
[1] 28
```

Several Operations on the Same Line

```
> 2+5 ; 18/2 # Separate calculations are separated by semicolons
[1] 7
[1] 9
```

Assignment Function

To save the intermediate answer for later use, we can assign the value to a symbolic variable

```
> assign ("x", 2)
```

OR simplified by using **"<-"** or "**->** "

```
> x<-2
```

```
> 2->x
```

OR using "**=**" sign

```
> x=2
```

To assign same value to multiple variables

```
> x=

> x <- y <- 2 # Both x and y are assigned values of 2

> x # If I enter x, R returns 2

> y # If I enter y, R returns 2 again.

[1] 2
```

Exercise (A):

x=5; y=6;

i. x+y
ii. x-y
iii. x multiple by y
iv. y divided by x

Vectors, Concatenation & Vector Functions

A vector is a sequence of data elements of the same basic type. Members in a vector are officially called components. To create a vector named x, consisting of four numbers, naming 1.2, 2.3, 0.2 and 1.1, use "**c()**"

```
> x <- c(1.2, 2.3, 0.2, 1.1) #c stands for concatenation

> x

[1] 1.2 2.3 0.2 1.1
```

To find out how many elements the vector has, use "**length()**"

```
> length(x)

[1] 4
```

To select some elements in the vector, use indices. Take the last three elements as example,

```
> x [c(2,3,4)]

[1] 2.3 0.2 1.1
```

```
> x [x>=1]

[1] 1.2 2.3 1.1
```

TRUE, FALSE or NA can be used directly to create logical vectors. TRUE is equal to 1 and FALSE is equal to 0 in arithmetic calculations.

```
> 1>=3

[1] FALSE

> !(1>3)

[1] TRUE

> (3 != 1) & (2 >= 1.9)

[1] TRUE
```

```
> y <- c(TRUE, FALSE, 5 > 2)
> y
[1]  TRUE FALSE  TRUE
> sum(y)
[1] 2
```

Exercise (B):

i) Create a vector x numbers from 1 to 5 ; find the length of x
ii) Create a vector y numbers 2.2, 6.3, 9, 1, 3, 4.5 ; find the elements which is bigger than 4

Other Statistical Functions

- `max(x)`
- `min(x)`
- `sum(x)`
- `mean(x)`
- `median(x)`
- `range(x)`
- `var(x)`
- `sd(x)`

Matrices

A typical matrix is a rectangular array of numbers, with m rows and n columns. The following is an example of a matrix with 2 rows and 3 columns.

$$A = \begin{bmatrix} 2 & 4 & 3 \\ 1 & 5 & 7 \end{bmatrix}$$

Matrix can be created by using "**matrix (c(), ncol=)**", "**matrix (c(), nrow=)**", "**cbind()**" , "**rbind()**" or "**matrix(,ncol,nrow)**"

ncol stands for number of columns.

nrow stands for number of rows.

cbind stands for column binding/combine.

rbind stands for row binding/combine.

```
> x <- matrix(c(1,2,3,4,5,6), ncol=2)
> x
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
```

```
> p <- matrix(c(1,2,3,4,5,6), nrow=2)
> p
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

```
> x <- cbind (c(1,2,3), c(4:6))
> x
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
```

```
> p <- rbind(c(1,3,5),c(2,4,6))
> p
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

Types of Data

Use “str()” to find out the types of data :

```
> z<- matrix(1:16,4,4)

> z

    [,1] [,2] [,3] [,4]

[1,]    1    5    9   13

[2,]    2    6   10   14

[3,]    3    7   11   15

[4,]    4    8   12   16
```

- numeric
- vectors
- matrices
- factors (i.e., categorical variables)
- string (collection of characters, expressed in double quotation marks)
- data frames (Data frames are like matrices, but instead of numbers, the entries can be numbers or factor variables.)

```
> k <-3

> k

[1] 3

> str(k)

 num 3
```

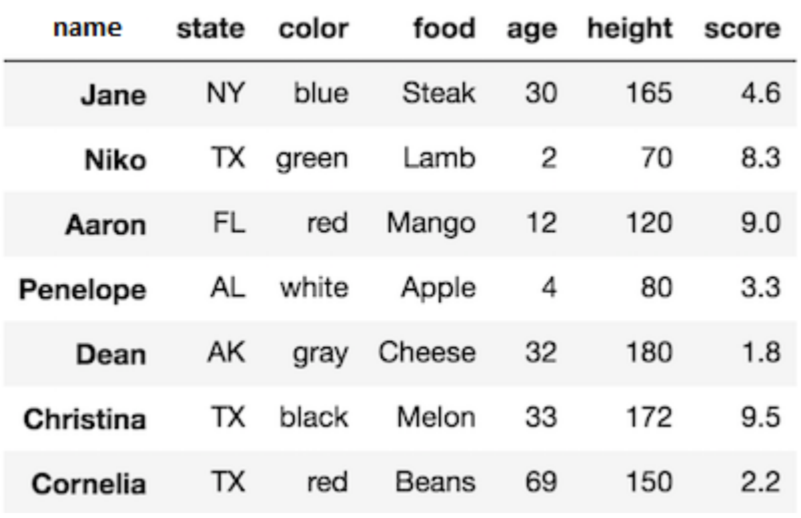
```
## String is a collection of characters and spaces

> B <-“my AB0402 got A+”

## the  “3” in A is stored as a string, not a numerical number. Thus, you cannot perform math operations on it.

> A <-“3”

> A+2

Error in A + 2 : non-numeric argument to binary operator
```

Be cautious when you copy the codes from PowerPoints or Word files to R-studio. The quotation marks in MS Office are in a different format than in R-Studio and will thus result in errors if you copy-and-paste the code to R. Retyping the quotation marks in R-studio is often necessary

Data Frames

A data frame is used for storing data tables. It is a list of vectors of equal length. In a datafrwame, each column contains values of one variable and each row contains one observation.

For example, the following data can be stored as a dataframe in R.



name	state	color	food	age	height	score
Jane	NY	blue	Steak	30	165	4.6
Niko	TX	green	Lamb	2	70	8.3
Aaron	FL	red	Mango	12	120	9.0
Penelope	AL	white	Apple	4	80	3.3
Dean	AK	gray	Cheese	32	180	1.8
Christina	TX	black	Melon	33	172	9.5
Cornelia	TX	red	Beans	69	150	2.2

```
> name=c("Jane","Niko","Aaron")
> state=c("NY","TX","FL")
> age=c(30,2,12)
> height=c(165,70,120)
> df<-data.frame(name,state,age,height)
> df
   name state age height
1  Jane    NY  30    165
2  Niko    TX   2     70
3 Aaron    FL  12    120
```

Note that, unlike a matrix, dataframe can include different types of variables. Name and state are both string variables, whereas age and hight are numeric.

You will not need to create dataframe in this course. The data files will be provided, and you only need to know how to load the data onto R.