COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

GU3105: R Tutorial 1

Applied Statistical Methods

Zhanhao Zhang
Department of Statistics, Columbia University

9/14/2021

What is R?

- An open-source programming language
- Statistical software & data analysis tool
- Can integrate with other languages (C, C++)

Install R

COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

- Go to https://cloud.r-project.org/.
- For Mac users:
    - Click “Download R for MacOS”
    - Download the package “R-4.1.1.pkg”
    - Complete the installation according to the prompts
- For Windows Users
    - Click “Download R for Windows”
    - Click on “Base”
    - Click on “Download R 4.1.1 for Windows”
    - Complete the installation according to the prompts
- Now you have the R environment in your computers

Install RStudio

COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

- Go to https://www.rstudio.com/products/rstudio/download/.
- Select “RStudio Desktop” (the free version!) and click on “Download”
- For Mac users, download the “RStudio-1.4.1717.dmg”. For Windows users, download the “RStudio-1.4.1717.exe”.
- Complete the installation according to the prompts
- Now you have the IDE for writing, debugging, and executing R code!

Basic R Operations

COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

- Arithmetic Operators: +, -, *, /, %%
- Power Operators: a**b, a^b (equivalent)
- Assignment Operators: name <- value, name = value (equivalent)
- Comparison Operators: <, >, <=, >=, != (not equals), == (equals)
- Logic Operators: & (AND), && (AND), | (OR), || (OR), ! (NOT)
- Other Operators:
    - factorial(n): n! = n * (n - 1) * … * 1
    - choose(n, k): $\binom{n}{k}$
    - sqrt(x): $\sqrt{x}$

R Vectors

- x1 <- c(1, 3, 4) # x1 contains 1, 3, 4
- x_reverse <- rev(1:5) # x_reverse contains 5, 4, 3, 2, 1
- x3 <- seq(1, 3, 0.5) # x3 contains 1.0, 1.5, 2.0, 2.5, 3.0
- y <- c(x1, x_reverse, 100) # y contains 1, 3, 4, 5, 4, 3, 2, 1, 100
- x_repeat <- rep(5, 3) # x_repeat contains 5, 5, 5
- x_repeat2 <- rep(c(1, 3), 2) # x_repeat2 contains 1, 3, 1, 3
- x_repeat_each <- rep(c(1, 3), each=2) # x_repeat_each contains 1, 1, 3, 3

R Vectors

- Character Vectors
  - x <- c('stat', "A", "A B", 'C') # x contains "stat" "A" "A B" "C"
  - The element of character vectors can be single character or a string of character, and it can be wrapped in either single quote or double quotes
  - letters[3:6] # "c" "d" "e" "f"
  - LETTERS[c(1, 14, 26)] # "A" "N" "Z"
- Logic Vectors
  - x <- c(T, F, TRUE, FALSE) # x contains TRUE, FALSE, TRUE, FALSE
  - y <- 1:5 < 3 # y contains TRUE TRUE FALSE FALSE FALSE
  - z <- y + 1 # z contains 2, 2, 1, 1, 1
  - TRUE ⇔ 1, FALSE ⇔ 0

R Vector Functions

- Check if an object is type xx:
    - is.numeric(), is.logical(), is.character(), is.integer(), is.factor(), is.function()
- Cast an object of type xx to yy:
    - as.numeric(), as.logical(), as.character(), as.integer(), as.factor(), as.function()
- Commonly used vector functions:
    - length(), sum(), prod(), min(), max(), cumsum(), cummin(), cummax(), diff(), sort(), order()
    - To get the documentation of a function, use "?". E.g. "?diff" gives the documentation that explains the usages of the function diff()

R Vector Slicing

x <- c(1, 3, 5, 7, 9)

- Extracting via a single value: x[3] # 5
- Extracting via a vector of single value: x[c(3)] # 5
- Extracting via a vector of multiple values: x[c(2, 4)] # 3, 7
- CANNOT extract via multiple values : x[2, 4] # error!
- Extracting via a vector of negative indices (excluding the values at the given indices): x[-c(2, 4)] # 1, 5, 9
- Extracting via a logic vector of same length: x[c(T, T, F, F, T)] # 1, 3, 9
- Extracting via a logic vector of shorter length (the shorter vector is recycled): x[c(T, F)] # 1, 5, 9.
    - Equivalent to x[c(T, F, T, F, T)]
- Extracting via a logic vector: x[x %% 3 == 0] # 3, 9
    - Equivalent to x[which(x %% 3 == 0)], where which() returns the indices where the statement holds true

R Vector Operations

Assume we have two vectors x and y

- Most arithmetic operations are vectorized:
    - x + y, x - y, x * y, x / y, x ^ y, x ** y, x %% y
- Many functions are vectorized:
    - cos(), sin(), cosh(), sinh(), sqrt(), log(), exp(), abs()

Try to vectorize your code as much as possible and avoid writing loops!!

R List

- List is an object which is a collection of other objects
- We can have list of character vectors, list of numbers, list of functions, list of lists, and etc.
- Lengths of different entries no longer have to be the same.
- Ex. L <- list(M=1:4, A=letters[1:6], F=function(x){x^2})

```
> L
    $M
    [1] 1 2 3 4

    $A
    [1] "a" "b" "c" "d" "e" "f"

    $F
    function(x){x^2}
```

R List -- Indexing


```
LL <- list(num=1:3, L=list(let_low=letters[3:1], LETTERS[1:2]))
> LL
$num
[1] 1 2 3

$L
$L$let_low
[1] "c" "b" "a"

$L[[2]]
[1] "A" "B"
```

R Matrix

- A rectangular n x m array of elements of SAME type
- A collection of equal length variable vectors
- Create a matrix:
    - matrix(1:12, nrow=3, ncol=4, byrow=F)
    - matrix(1:12, nrow=3, byrow=F)
    - matrix(1:12, nrow=3, byrow=T)
    - cbind(1:3, 4:6, 7:9, 10:12)
    - rbind(c(1, 4, 7, 10), c(2, 5, 8, 11), c(3, 6, 9, 12))
- Matrix arithmetic:
    - Elementwise: +, -, *, /, %%, ^, ** (matrices should have the same dimensions, otherwise will recycle the smaller one)
    - Matrix multiplication in linear algebra: %*%
    - Inversion: solve()

R Dataframe

- Can have different types of data into the dataframe object
- A collection of equal length variable vectors
- Create a dataframe: data.frame(name1=vector1, name2=vector2)
    - Ex. df <- data.frame(num=1:6, let=letters[1:6])
- Accessing the dataframe:
    - head(df) # displays the top 6 rows of the dataframe with headers
    - df$num # extract the column “num”: 1, 2, 3, 4, 5, 6
    - df[,1] # extract the first column of the dataframe
    - df[3,] # extract the third row of the dataframe: 3, ”c”
    - df$column3 <- 2:7 # create a new column named “column3” and assign the vector c(2, 3, 4, 5, 6, 7) to it

R Read/Write Files

- Read a file into a dataframe:
    - Use read.table(filename, # The path to the targeting file
    - header, # whether the file has a header
    - sep) # The deliminator (separator) of the file
    - Ex. read.table("data.txt", header=T, sep=",")
    - Also check the function read.csv()
- Write a dataframe into a file:
    - Use write.table(df, "filename", sep=",")

R Functions

- Functions can execute any number of commands within { and }

```
function_name <- function(param1, param2, …,
paramN){
        # commands
        # return statement
}
Ex. square <- function(x){
        squared_value <- x ^ 2
        return(squared_value)
}
square(3) # returns 9
```

R Loops

- Loops repeat a set of commands as long as some conditions are met
- for-loop

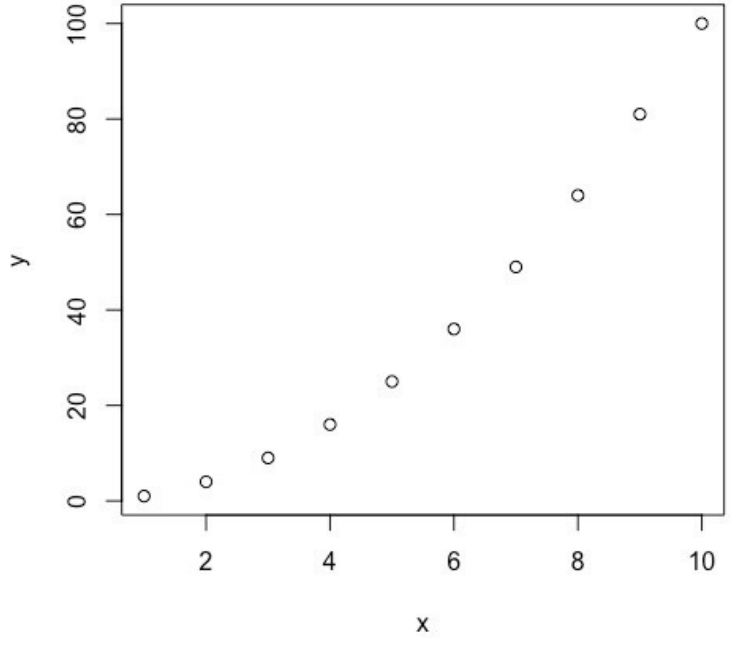
```
for(i in x){
        # do something that may or may not involve i
}
```

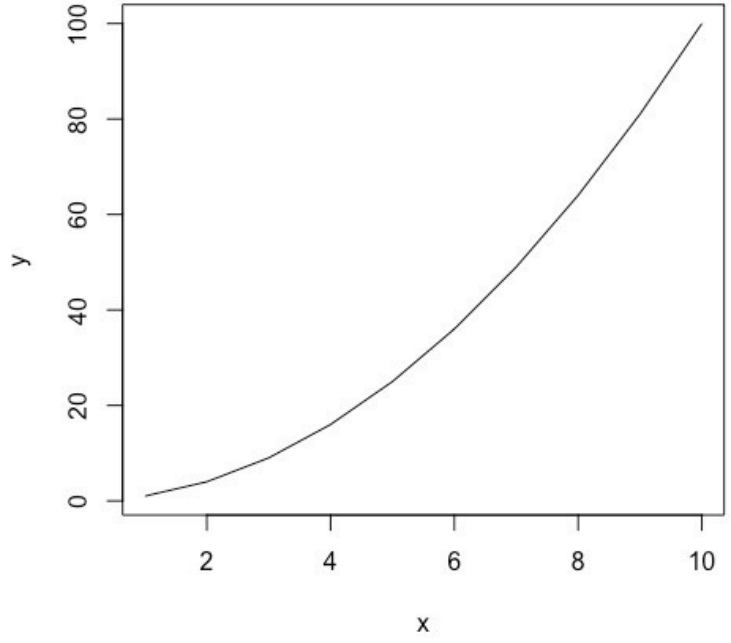
- while-loop:

```
while(logic evaluation){
        # repeat the commands inside the {} as long as the
        logic evaluation is TRUE
}
# Proceed further when the logic evaluation is FALSE
```

R Plots – plot()

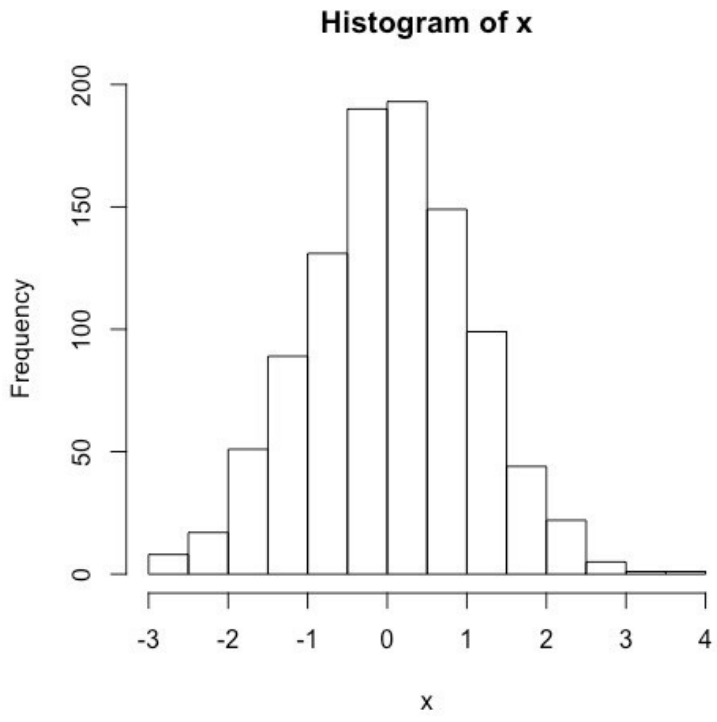

```
x <- 1:10
y <- x^2
plot(x, y) [Left]
plot(x, y, type="l") [Right]
```

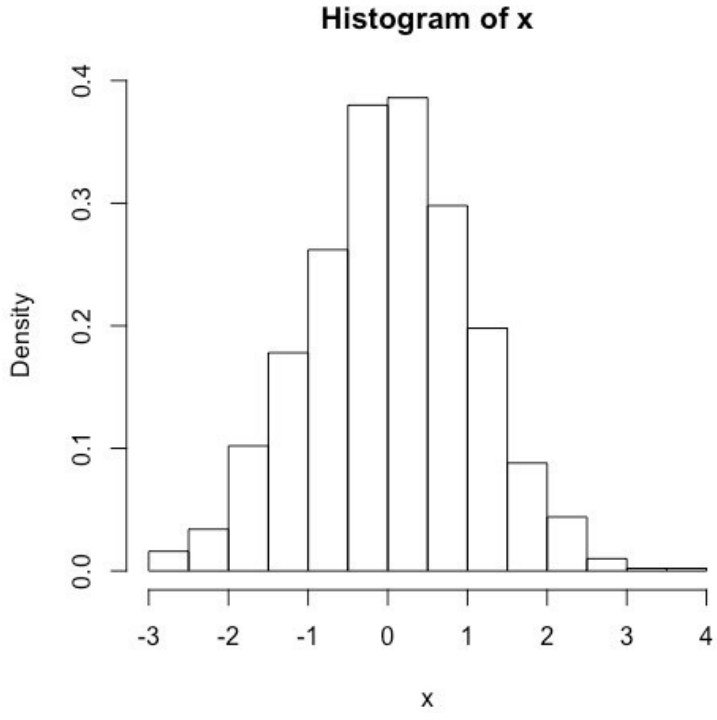




R Plots – histogram


```
x <- rnorm(1000)
hist(x) [Left]
hist(x, probability=T) [Right]
```



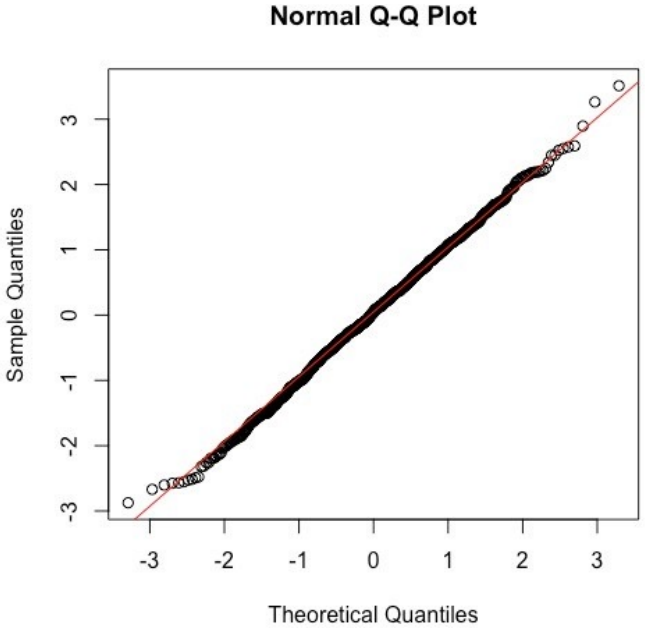


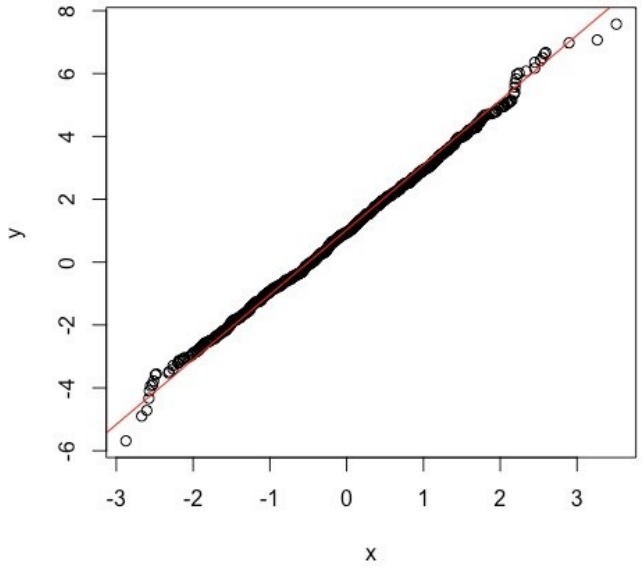
R Plots – QQ-plot

x <- rnorm(1000)

y <- rnorm(2000, mean = 1, sd = 2)

- QQ-plot using one variable [Left]:
  - qqnorm(x); qqline(x, col="red")
- QQ-plot using two variables [Right]:
  - qqplot(x, y); qqline(y, col="red")





More Questions?

COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

1. Consult the comprehensive R documentation: https://rdrr.io/r/
2. Search your questions on https://www.google.com/ or https://stackoverflow.com/
3. Post your questions on Piazza (https://piazza.com/class/kswf2c80ejx69v)