**Module 1 R Code – Introduction to R Programing**

**Installing R**

* Install R Software on Your Laptop
* Go to R main website <https://cran.r-project.org/> and download R based on your operating system.
* Install of R on Windows Operating System.
  + **Installation on Windows OS** Step by Step installation Video <https://www.youtube.com/watch?v=mfGFv-iB724>
  + **Installation on MacOS** Step by Step installation Video <https://www.youtube.com/watch?v=uxuuWXU-7UQ>

To be able to work much easier, we recommand to use a graphical user interface — Integrated Development Environment (IDE) for R Programming.

**RStudio** (IDE Recommended )

RStudio <https://www.rstudio.com/> is a free and open-source Integrated Development Environment (IDE) for R Programming.

How to install RStudio. Step by step video <https://www.youtube.com/watch?v=cX532N_XLIs>

**Introduction to R**

In this mini tutorial we will review some basic concepts of R programing.

Run R and check the following commands:

# R includes numeric types: interger, double

> 348

# Characters

> "my string"

# logical

> TRUE

> FALSE

# Arithmetic operators as you'd expect

> 42 + 1 \* 2^4

# We have logical operators/comparison

TRUE | FALSE

> 1 + 7 != 7

# Other logical operators:

# &, |, !

# <,>,<=,>=, ==, !=

**Variable Assignment**

# Variables assignment is done with the <- operator

> mynumber <- 483

# typeof() tells use type

> typeof(mynumber)

[1] "double"

# we can convert between types

myint <- as.integer(mynumber)

> typeof(myint)

[1] "integer"

**Vector of Data**

# The vector is the most important data structure

# create it with c() - named combiner function

my.vec <- c(1, 2, 67, -8)

# get some properties

str(my.vec)

##

num [1:4] 1 2 67 -8

> length(my.vec)

[1] 4

# access elements with []

> my.vec[3]

[1] 67

# We can use another vector to index , like following command.

> my.vec[c(3,4)]

# can do assignment too

my.vec[5] <- 41.2

# Working directory - Setting/Getting}

# It is the default location of all input and output files

# List all the objects in the current workspace

> getwd()

# Set working directory

> setwd("/YOUR-HOME-FOLDER/YOURFOLDER")

Remember to use double backslashes or use a single forward slash "/"

# List all the objects in the current workspace

# For example like following settings in Windows.

> setwd("C:/Users/xyz/Documents/work/R")

\greenb{You can use the RStudio menus to set your working directory.}

Reading Data into R

Read a Comma-Separated Values (CSV) data file from a text file

> read.csv("filename")

First Line is the header, default value for header is True

> read.csv("filename", header=True)

It reads a \blueb{Dataframe} into R. \red{Datatrame is an important data type in R.}

**Dataframes in R**

> read.csv("filename", header=True)

# It data type similar to an Excel Sheet or a Database Table like:

# Like following data

"age","job","marital","education","balance","housing","loan","contact"

30,"unemployed","married","primary",1787,"no","no","cellular"

33,"services","married","secondary",4789,"yes","yes","cellular"

35,"management","single","tertiary",1350,"yes","no","cellular"

30,"management","married","tertiary",1476,"yes","yes","unknown"

**Installing Library Packages in R**

# Install a package (only need to do it once)

> install.packages("package name")

# It will recognize dependencies between packages and install required sub packages

# Access the package and loading it into memory

> library("package name")

# View a list of installed packages

> library()

**R Session Commands**

Session Commands

> q() # end R session

Save workshpace image? [y/n/c]:

# y - yes

# n – no

# c – cancel

# Save content of the current workspace into .Rdata file

> save.image()

> save.image(file = "abc.Rdata")

# Save some objects of the current workspace into the file

> save.image(a, b, file = "abc.Rdata")

**Load Stored Objects**

# You can load a set of objects in R from a Rdata binary file.

> load("abc.Rdata")

# List all the objects in the current workspace

> ls()

OR

> objects()

# Remove objects from the current workspace

> rm(a, b)

# delete a file

> unlink("myFile.Rdata")

**Learning R in R: swirl**

You can learn R in R. Step by Step Tutorial: the package swirl <http://swirlstats.com/students.html> can be used to learn R.

> install.packages("swirl")

> library("swirl")

> swirl()

**Module 1 R Code – R Functions for Normal Distribution**

**pnorm()** – computes the probability} that a normally distributed random number will be less than the given number.

This function is also called the "Cumulative Distribution Function" (CDF).

Calculate the area to the left of z: P(Z <= z)

pnorm(z)

Non-standardized normal distribution

pnorm(x, mean=a, sd=b) # calculate the area to the left of x

Find the area under the standard curve  
z = 1.53

pnorm(1.53)

0.9369916

**Standardized Normal Distribution – An Example**

Let’s assume that the birth weights of newborns are normally distributed with a mean of 3500g with a standard deviation of 500g.

What proportion of infants weigh less than 2800g?

The variable x, the birth weight, has the N(3500,500) distribution.

> pnorm(2800, mean=3500, sd=500)

0.08075666

What proportion of infants weigh between 3250g and 3750g?

> pnorm(3750, mean=3500, sd=500)-pnorm(3250, mean=3500, sd=500)

0.3829249

**R Functions on Normal Distribution – Normal Distribution Probability Density Function (PDF)**

**dnorm()** – Given a set of values it returns the height of the probability distribution at each point. If you only give the points it assumes you want to use a mean of 0 and standard deviation of 1.

density\_standard\_norm <- function(x){1/sqrt(2\*pi)\*exp(-0.5\*x^2)}

> dnorm(0)

0.3989423

> dnorm(0, mean=4, sd=10)

0.03682701

> pnorm(0)

0.5

In statistics, quantiles are cut points dividing the range of a probability distribution into contiguous intervals with equal probabilities.

Given a probability p and a distribution, we want to calculate the corresponding quantile for p: the value x such that P(X <= x) = p

# For non-standardized normal distribution

> qnorm(x)

# For any normal distribution with mean and sd

> qnorm(x, mean=a, sd=b)

**R Function on Normal Distribution**

Density, distribution function, quantile function and random generation for the normal distribution with mean equal to mean and standard deviation equal to sd.

Usage:

dnorm(x, mean = 0, sd = 1, log = FALSE)

pnorm(q, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)

qnorm(p, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)

rnorm(n, mean = 0, sd = 1)

**Summary**

**pnorm** – calculates the probabilty value given a critical value

**qnorm** – is invese computation and calculate a critical value given a probabilty value

**dnorm** – calculates density value for a given critical value (you can understand this as y value for a given z or x critical point)

**rnorm** – generates random samples from a normal distribution

For more information read the manuals.

To read the manual, use one of the following commands:

?pnorm

or

help(pnorm)

**Data Summarization**

In R you can simply pass your data object to summary function.

* summary() function is a very general function and we use it in R a lot to pass data and model object to it.
* summary() is a intelligent function and can return well-formatted prints depending on object types.

> earning <- c(35, 40, 145, 33, 30, 42, 32, 32, 25)

> summary(earning)

Min. 1st Qu. Median Mean 3rd Qu. Max.

25 32 33 46 40 145

You can calculate the results from the above summary() using the following separate functions:

> mean(earning)

> median(earning)

> min(earning)

> max(earning)

> quantile(earning)

You can calculate the Variance and standard deviation using following functions:

> var(earning)

> sd(earning)

**Data Summarization and Visualiztion – R Functions – Graphical data summaries**

**Histograms**

> hist(data$variable)

> hist(data$variable, bins) # specify the number of bins

> hist(data$variable, breaks=c(x,y,z..)) # specify cutpoints

> hist(data$variable, breaks=seq(a,b,by=c)) # specify cutpoints

data$variable is a vector of data that we read for example from a dataframe "data" and access specific variable of it using a dollar sign.

**Boxplot**

We can create a boxplot using the **boxplot()** function.

boxplot(earning)

boxplot() function in R can detect outliers and visualize it.

Read the documentations of boxplot function to find out how you can turn this automated process ON or OFF.

**Formatting**

You can make your graphs look much better and exactly what you want to have:

* **Labeling:** Each graph in R can have a title a X and Y-Axis label. You can use the following lables, main, xlab and ylab.
* Title: main="Histogram of xyz"
* X-axis label: xlab="Nile flow"

Y-axis lable: ylab = "Frequency"

* **Colors:** You can fund [a list of collor numbers](http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf).
* **Controlling the window:** you can use the following attributes xlim and ylim to have a limitation for x and y axis.
* X-axis: xlim=c(min, max)

Y-axis: ylim=c(min, max)

**Combine multiple plots into one overall graph**

Creating graphs side-by-side. Before you create your plot you need to set an enviroment variable that controls the positioning of plots.

You can use the **par()** function and use the **mfrow** attribute

par(mfrow=c(2,2)) # 2 by 2 panels

par(mfrow=c(1,1)) # Go back to single graph mode

**R Functions – Qualitative Data Summaries**

* Numerical summary:
  + Class Frequencies

> table(data$variable)

Or

> summary(data$variable)

* + Relative Class Frequencies: Divide class frequencies by number of rows in the dataset using nrow(data).
* Graphical summary: You can create a Pie chart or a barplot.
* > Pie(table(data$variable))

> Barplot(table(data$variable))

**Module 1 R Code – Examples and Test Yourself Questions in R**

In this section you can find the above Examples implemented in R code.

**Module 1 Example 1.3 in R**

x <- c(42,20,32,47,31, 66,25,64,25,46, 76,56,32,20,50,60,58,31,83,51,22,32,64,49,75,

40, 43, 54 ,44, 62,

46, 27 ,32 ,49 ,37,

38 ,59 ,33, 59, 73,

26, 26 ,83 ,71 ,39,

35 ,33, 35, 28, 35)

result.mean <- mean(x)

print(result.mean)

**Module 1 Test Yourself 1.2 in R**

x <- c(35,40,145,33,30,42,32,32,25)

result.mean <- mean(x)

print(result.mean)

**Module 1 Test Yourself 1.3 in R**

x <- c(35,40,145,33,30,42,32,32,25)

median.result <- median(x)

print(median.result)

**Module 1 Example 1.7 in R**

x <- c(35,40,145,33,30,42,32,32,25)

quantile(x)

**Module 1 Example 1.8 in R**

x <- c(42 , 20 , 32 , 47 , 31 ,

66 , 25 , 64 , 25 , 46 ,

76 , 56 , 32 , 20 , 50 ,

60 , 58 , 31 , 83 , 51 ,

22 , 32 , 64 , 49 , 75 ,

40 , 43 , 54 , 44 , 62 ,

46 , 27 , 32 , 49 , 37 ,

38 , 59 , 33 , 59 , 73 ,

26 , 26 , 83 , 71 , 39 ,

35 , 33 , 35 , 28 , 35 )

# Print lenght of vector x

print(length(x))

# Print mean of vector x

print(mean(x))

# Print standard deviation of vector x

print(sd(x))

# Print variance of vector x

print(var(x))

**Module 1 Example 1.12 in R**

mean <- 3500

sd <- 500

pnorm(4000, mean, sd) - pnorm(3000, mean,sd)

pnorm(4500, mean, sd) - pnorm(2500, mean,sd)

**Module 1 Example 1.13 in R**

pnorm(1.53) \* 100

**Module 1 Example 1.14 in R**

(1 - pnorm(−0.58) ) \* 100

**Module 1 Example 1.15 in R**

pnorm(0.77) - pnorm(-1.25)

**Module 1 Example 1.26 in R**

sample <- c(20, 44, 46, 20, 44)

mean(sample)

median(sample)

**Module 1 Example 1.28 in R**

((1-pnorm(-1)) - (1-(pnorm(1))))\*100

**Module 1 Test Yourself 1.10 in R**

((1-pnorm(-1.67)) - (1-(pnorm(1.67))))\*100

**References for Module Materials**

McClave, J. T., & Sincich, T. (2012). *Statistics* (12th ed.). Pearson.

Moore, D. S. (2003). *The basic practice of*