Module 2
Introduction to R

MULTIMEDIA UNIVERSITY

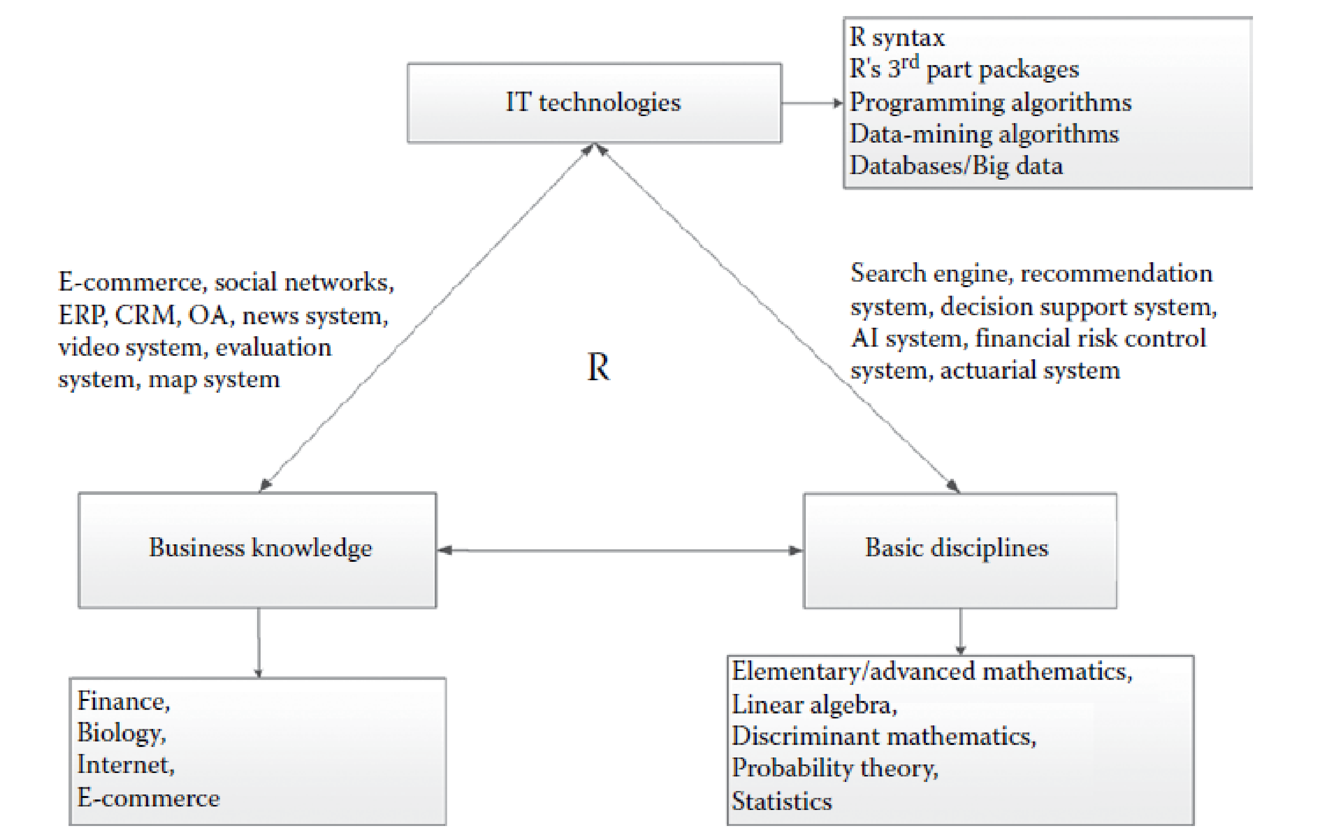
Microsoft® IT Academy Program

Module 2 Introduction to R

No.	Topics
1	Getting Started with R
2	Why R
3	Installing R and RStudio
4	Introduction to R interface
5	Getting Help
6	The Workspace & The Packages
7	Getting Familiar with R Command
8	The Basic Data types (string/character, numeric, logical, factor) Variables
9	Little Tricks on R
10	More About R Objects Vectors Matrices & Array Lists Data Frames Functions & Basic Operators

1. Getting Started with R

- To master R, you need knowledge from multiple disciplines



IT technologies
R syntax
R's 3rd part packages
Programming algorithms
Data-mining algorithms
Databases/Big data
E-commerce, social networks, ERP, CRM, OA, news system, video system, evaluation system, map system
R
Search engine, recommendation system, decision support system, AI system, financial risk control system, actuarial system
Business knowledge
Basic disciplines
Finance,
Biology,
Internet,
E-commerce
Elementary/advanced mathematics,
Linear algebra,
Discriminant mathematics,
Probability theory,
Statistics

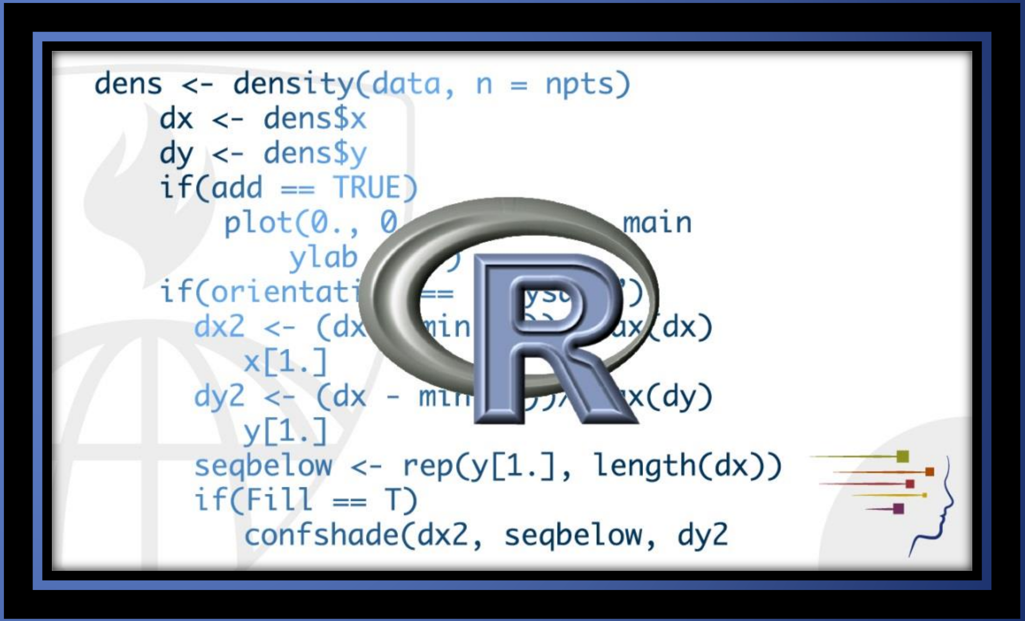
Figure 1: The Knowledge System of R (Zhang, 2017)

Getting Started with R cont…

- Origins: S programming (created 1970s)
- R was created as alternative (1990s)
  - Initially used by statisticians and scientist
  - Now by data scientist due to great interactive exploratory data analysis
- Best to
  - Manipulate moderately sized datasets
  - Do statistical analysis
  - Produce data-centric documents
  - presentations
- Popular due to
  - Interactive nature
    - A lot of choices in graphic packages
  - Expressiveness
    - Single line of code with minimal number of characters to perform calculations
      - E.g.: mean(c(1,2,3,4,5,6))
  - Extensive collection of third-party libraries created for R
    - reshape2
    - ggplot2
    - stringr
    - etc...

Getting Started with R cont…

- It has 7 core packages
  - Mathematical calculation
  - Statistical
  - Date
  - Package loading
  - Data processing
  - Function operating
  - Graphics devices
  - +others more such as data-accessing (e.g.: RMySQL)



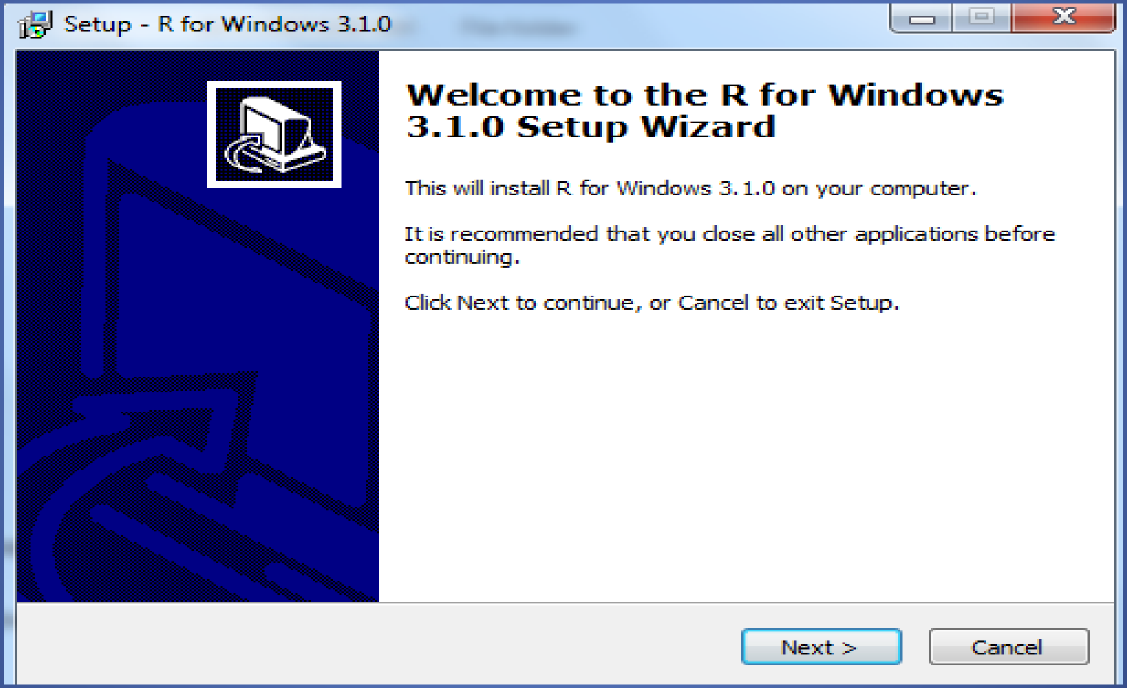
```
dens <- density(data, n = npts)
    dx <- dens$x
    dy <- dens$y
    if(add == TRUE)
        plot(0., 0                  main
            ylab         )
    if(orientati    ==     ysc      ")
      dx2 <- (dx    min            ax(dx)
        x[1.]
      dy2 <- (dx - min          ))/  x(dy)
        y[1.]
      seqbelow <- rep(y[1.], length(dx))
      if(Fill == T)
        confshade(dx2, seqbelow, dy2
```

2. Why R?

- Free and inexpensive
- Available for Windows, Linux and Mac
- Open source software
- Comparable with those commercial products
- Lots of advice and guidance available online
- Perform complex calculations using only few commands or short scripts
- To use R with GUI:
  - Rstudio: http://www.rstudio.org
  - R Commander: *https://www.rcommander.com/*
  - StatET: http://www.walware.de/goto/statet/
  - JGR (Java GUI for R): http://cran.r-project.org/web/packages/JGR/index.html

3. Install R & RStudio

- Get R from CRAN (Comprehensive R Archive Network): https://cran.rstudio.com/
- Current R is 3.5.1.
- Select previous releases as Azure Machine Learning is using CRAN R 3.1.0.
- Just install R according to the wizard:



Welcome to the R for Windows 3.1.0 Setup Wizard

This will install R for Windows 3.1.0 on your computer.

It is recommended that you close all other applications before continuing.

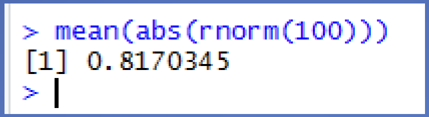
Click Next to continue, or Cancel to exit Setup.

- Download RStudio at:

https://www.rstudio.com/products/rstudio/download/#download

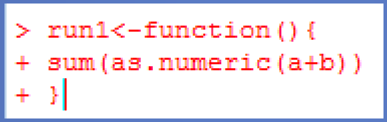
4. Introduction to R Interface

- R issues a prompt (“>”) when it expects input commands
  - Press enter, the command is sent to R for processing



```
> mean(abs(rnorm(100)))
[1] 0.8170345
> 
```

- If a command is not complete at the end of a line, R will give a different prompt (“+”), i.e. a continuation state



```
> run1<-function(){
+ sum(as.numeric(a+b))
+ }
```

- Can go back to previous history by pressing up arrow key
- Command are separated with semicolon (;) or by a newline
- Comments can be put almost anywhere, starting with a hash mark (“#”): # first line
- To quit R: q(save=“no”)

5. Getting Help

- Type of Help:
  - General search:
    - help.start()   #be patient while waiting
  - Specific search of a topic:
    - help.search(“linear regression”)
  - Search for a function:
    - help(“mean”)
  - Search a string:
    - ??lubridate
- Shortcut to help is a question mark (?):
  - ?mean
- Can obtain examples of command:
  - example(mean)

6. The Workspace & The Packages

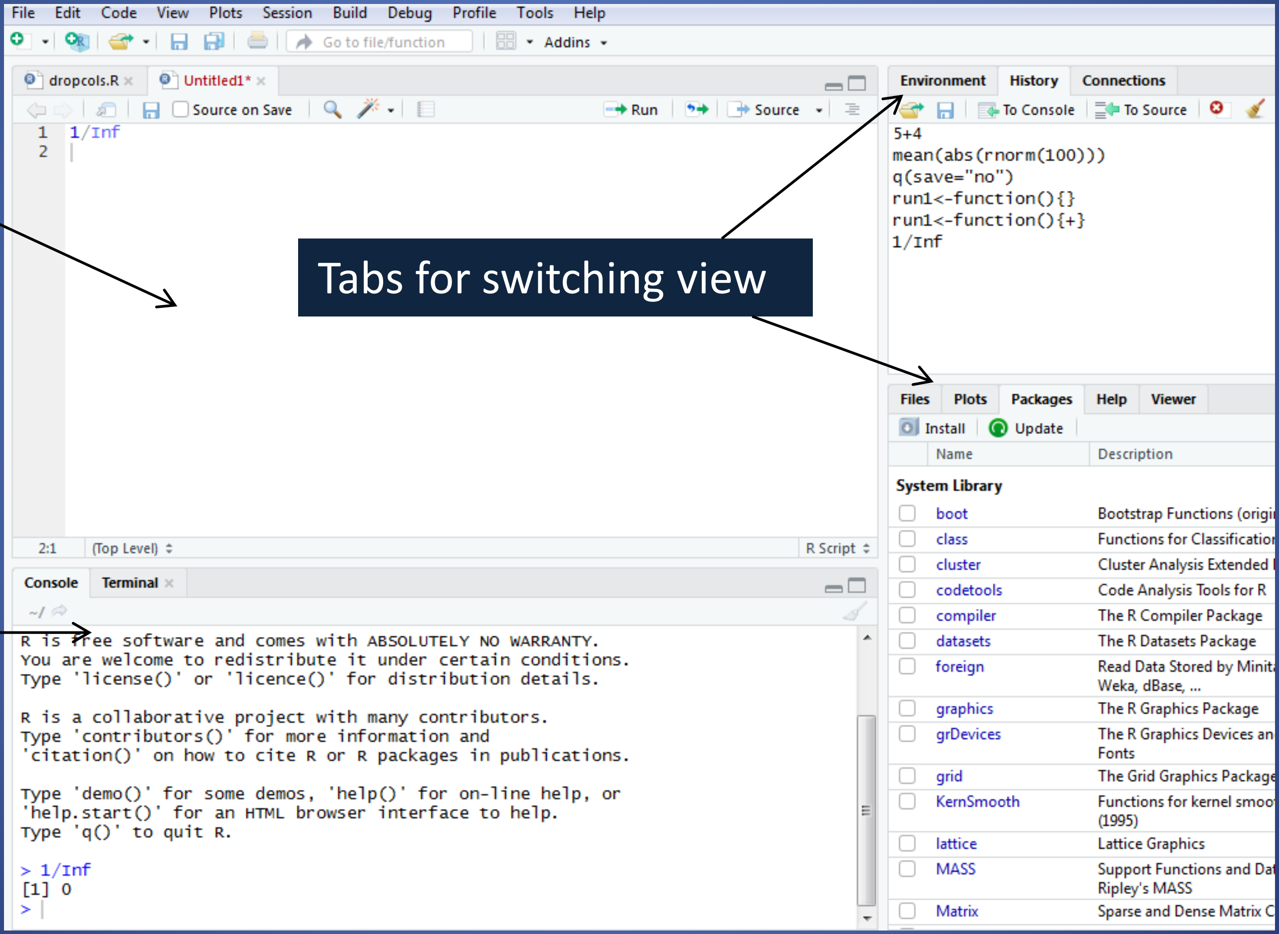
- Workspace: a collection of objects that you currently have in R session
- Set my desire directory through:
  - Session->Set Working Directory->Choose Directory
- Objects are lost when you quit R without save the objects
- To save the current working directory:
  - Session->Save Workspace
- To load the current workspace
  - Session->Load workspace

The Workspace & The Packages cont…

Editor windows: R scripts

Tabs for switching view

R console



```
1/Inf
```

```
5+4
mean(abs(rnorm(100)))
q(save="no")
run1<-function(){}
run1<-function(){+}
1/Inf
```

	Name	Description
	System Library	
□	boot	Bootstrap Functions (origi
□	class	Functions for Classificatio
□	cluster	Cluster Analysis Extended
□	codetools	Code Analysis Tools for R
□	compiler	The R Compiler Package
□	datasets	The R Datasets Package
□	foreign	Read Data Stored by Minit Weka, dBase, ...
□	graphics	The R Graphics Package
□	grDevices	The R Graphics Devices an Fonts
□	grid	The Grid Graphics Package
□	KernSmooth	Functions for kernel smoo (1995)
□	lattice	Lattice Graphics
□	MASS	Support Functions and Dat Ripley's MASS
□	Matrix	Sparse and Dense Matrix C

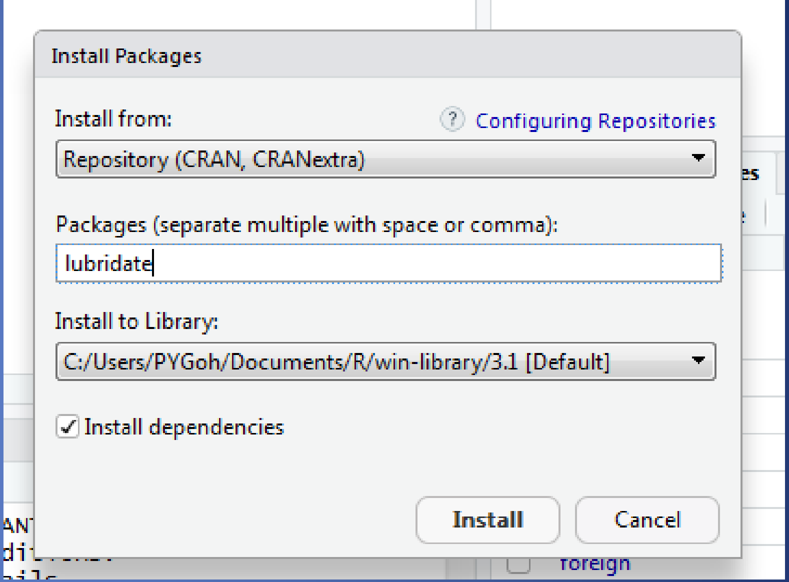
```
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

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.

> 1/Inf
[1] 0
>
```

The Workspace & The Packages cont…

- Packages are installed to your machine but it is not loaded to your current R session
  - Try this command in R console:
    - library(lubridate)
- Let's try to load a package called Lubridate
  - Tools->Install package
  - Under packages, type lubridate->Install
  - After done, type the following command in R script:

    ```
    library(lubridate)
    mdy(c("6/12/16", "2/9/16"))
    ```

    Run
  - Use command to install:
    - install.packages('lubridate')
- To know the available packages in R, try the following command
  - data.set<-data.frame(installed.packages())
  - data.set



Install Packages

Install from: Repository (CRAN, CRANextra)

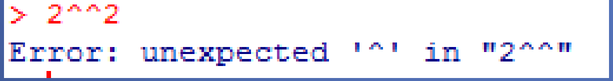
Packages (separate multiple with space or comma): lubridate

Install to Library: C:/Users/PYGoh/Documents/R/win-library/3.1 [Default]

☑ Install dependencies

7. Getting Familiar with R Command

- The command line is known as console
- Go to start->R->Ri386 3.1.0
- Type each of the following command and press enter:
  - 2 + 2   Answer: [1] 4
  - (1+3+2+12+8)/5   Answer: [1] 5.2
  - ((2-1)^2+(1-3)^2)^(1/2)   Answer: [1] 2.236068
- The return output [1] is the prefix
- Standard mathematical operators in R: +, -, *, /, ^
- Can combine commands using ;
- Error message is generated when the syntax is incorrect:



```
> 2^^2
Error: unexpected '^' in "2^^"
```

- To check the history of your command, you need to use Rstudio's history pane

8. The Basic – Data Type

- The most basic type of R is vector
- Empty vector is created through vector()
- Vector with single value is called mode (which describe how data is stored)
- We can check the type of a variable through mode()
  - E.g.:
    - x<-c(1,2,3)
    - mode(x)
- String / text: single-element vector with type character
  - Type the command:
    - str1 <- "Hello world!“
    - str1
  - Convert number to strings
    - str2 <- as.character(99)
    - str2
  - Concatenate/combine the string
    - paste("Hello", "World", "!")
    - paste("Hello", "World", sep="") ## no separator
  - To get the length of string
    - length(str1)
  - From upper to lower and vice versa
    - toupper(str1)/tolower(str1)
  - NULL is to indicate *nothing is present*
    - Type the following command:
      - NULL
      - c()
      - Is.null(c()) # to check whether it is null

The Basic – Data Type cont…

- Integer/numeric: decimal values are considered as numeric type in R
  - Type the command:
    - x = 10.5
    - x   # to print out x
    - class(x) #to know the type of x
  - To check is numeric:
    - is.integer(x)
  - To assign an integer, we can use is.integer
    - y=as.integer(5)
    - y   # to print out y
    - class(y)   #to know the type of y
    - y = as.integer(5.12)
    - y
- Other operations: ceiling, floor, round, exp, log, tan...
- Special Inf represent infinity
  - 1/Inf #it gives you zero

The Basic – Data Type cont…

- Logical – TRUE/FALSE
  - Type the command:
    - u = TRUE; v = FALSE
    - u & v   # u AND v
    - u | v   # u OR v
    - !u   # negation of u

The Basic – Data Type cont…

- Factor
  - to represent categorical data and can be unordered or ordered
  - Similar to integer vector but with levels
  - Type the command:
    - x<- factor(c(" yes ", " yes ", " no ", " yes ", " no "))
    - x
- From numbers to categorical
  - Type the command:
    - marital_status<-c(0,3,2,2,1)
    - fstatus<-factor(marital_status, levels = 0:3)
    - levels(fstatus)<-c("single", "married" , "divorced", "widowed")
    - fstatus
    - as.numeric(fstatus) #extract the numerical coding as numbers
    - levels(fstatus) #extract the name of levels
-

The Basic – Variables

- Assignment
  - To assign a value to a variable:
    - x <- 5
    - x ##(observe the output)
    - print(x)
  - Type the following commands and observe the output:
    - x <- c(1,2,4) #this is a simple data set consists of 1, 2, 4
    - q<-c(x,x,8)
    - x #auto printing
    - print(x)   #explicit printing
    - x[2]   #print the 2nd element of x
    - mean(x)   #print the average based on values of x
    - sd(x) # standard deviation
    - y <- mean(x)
    - y
    - z<-10:30   #assign numbers from 10 to 30 to z
    - z

The Basic – Variables cont…

- Naming – useful in writing readable code
  - To add names to object
    - named_try1 <- c(a = 1.02, b = 2, 3)
    - #two elements are named as a and b
    - names(named_try1)
    - x = c(one=1,two=2,three=3)
    - x
    - x = c(1,2,3)
    - names(x) = c('one', 'two', 'three')
    - x
    - names(x)[1:2] = c('uno', 'dos') # to modify the name
    - x
  - We can access through the vector through name instead of indices
    - named_try1["b"]
    - named_try1[c("a", "b")]

Exercises 2.1 – 15 minutes

- Solve the following problems by using R command. For e.g.: 1+2, is written as 1+2 in R. Print screen your answer after done.
  - 2^8
  - $\text{Log}_{10}800$
  - √560
  - $\frac{0.25-0.2}{\sqrt{0.2(1-0.2)/100}}$
  - $\frac{2}{1}, \frac{2^2}{2}, \frac{2^3}{3}, \dots \frac{2^{25}}{25}$

1. **2^8**
2. **log10(800)**
3. **sqrt(560)**
4. **(0.25-0.2)/(sqrt(0.2*(1-0.2)/100))**
5. **2^(1:25) / (1:25)**

9. Little Tricks on R

- R is case sensitive. R and r are treated as different object.
- Names are unlimited in length.
- Alphanumeric symbols are allowed, together with "." and "_"
- Commands are separated by semi-colon (";") or by newline
- To return to previous command, just press the up arrow and down arrow key
- Ctrl+L: clear console
- To record your work: write it in the script and save as R script
  - File/New File/R script
  - Or use notepad
  - Work will be saved as .R extension

10. More About R - Objects

- Objects
  - All function, data structures, exist as objects
  - Could be variables, arrays of numbers, character strings, functions
  - Objects are stored in the workspace
  - Can be stored permanently for use in future
  - Objects have a mode
  - Type the command:
    - ls()   #to see what objects are in your workspace
    - objects()   #another alternative to ls()
    - save.image(file= " archive.RData ") # save content of workspace
    - rm()   #to remove objects
    - exists()   #to check whether the existence of an object
  - Objects are written to a file called '.RData' and the command lines used are saved to a file called '.Rhistory'
  - To count the future value of a deposit RM100 today after 5 years with interest rate of 3%
    - interest<-0.03   #3%
    - deposit<-100
    - future.value<-deposit*(1+interest/12)^(5*12)
    - future.value   #print the result
    - rm(future.value) #to remove objects

More About R – Objects cont…

- Date and Time
  - To create a date object:
    - Date only object, as.Date is the best choice
    - x<-as.Date("2018-09-28")
    - as.Date('1/15/2001',format='%m/%d/%Y')
    - as.Date('April 26, 2001',format='%B %d, %Y')

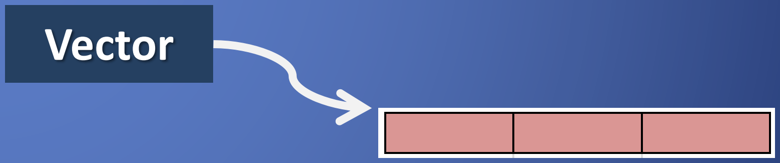
Code	Value
%d	Day of the month (decimal number)
%m	Month (decimal number)
%b	Month (abbreviated)
%B	Month (full name)
%y	Year (2 digit)
%Y	Year (4 digit)

More About R – Objects cont…

- Date and Time
  - To create a date and time object with time zone:
    - x<-Sys.time()
    - x
    - class(x) #it is categorised as a *'POSIXct'* object
      - Classes which allow for dates and times with control for times zones
    - POSIX represents a portable operating system interface, primarily for UNIX systems, but available for other operating system as well
      - Provide more accurate representation of times
  - To create date and time without time zone:
    - chron library is the best choice and it is often used to split date and time apart.
    - install.packages("chron")
    - dtimes = c("2002-06-09 12:45:40","2003-01-29 09:30:40", "2002-09-04 16:45:40","2002-11-13 20:00:40", "2002-07-07 17:30:40")
    - dtparts = t(as.data.frame(strsplit(dtimes,' ')))
    - row.names(dtparts) = NULL
    - library(chron)
    - thetimes = chron(dates=dtparts[,1],times=dtparts[,2], format=c('y-m-d','h:m:s'))
    - thetimes
-

More About R - Vectors

- The fundamental data type in R is vector
- Dataset is usually in the shape of rows and columns
  - Statisticians: observation and variables
  - Database analysts: records and fields
  - Data miner: examples and attributes
- Thus, data structure of R is based on rows and columns
- A vector can only contain objects of similar data types
  - String: a vector
  - Integer/numeric: a vector
  - Vector: a sequence of number
    - c(3,4,5)
  - Scalar: a single number
    - x=5
  - In R, all are vectors, no scalar
  - Vectors type: characters/string; integer/numeric; logical (the basic data type of R)



Vector

More About R – Vectors cont…

- Declaration – refer The Basic - Variables
  - This will not work:
    - y[1] <- 5
    - y[2] <- 12
  - This will work:
    - y<-vector(length=2)
    - y[1] <- 5
    - y[2] <- 12
    - y
- Adding & deleting vector elements
  - Add an element to the middle of a 4-element vector:
    - x<-c(88,5,12,13)
    - x
    - x<-c(x[1:3], 168,x[4])
    - x
- Get the length of a vector:
  - length(x)

More About R – Vectors cont…

- Common vector operations
  - Vector Arithmetic and Logical Operations
    - 2+3
    - "+" (2,3)
    - x<- c(2,4,6)
    - x*c(5,0,-1) #multiplication is done element by element
    - Try to change the above * to other operators such as /, %% and the operator is applied element by element
  - Vector indexing: all indexes in R begin at 1, not 0
    - x[1] #to obtain the value of first element in x
    - x[c(1,3)] #to extract element 1 and element 3 of x
    - x[1:2] #to extract continuously element 1 and 2
    - y<-1:2
    - x[y]
    - x[-1] #negative means that we want to exclude element 1
    - x[length(x)] #length(x) is 3, so it prints the 3rd element
    - x[-length(x)] #-length(x) is equal to 1:(3-1), i.e. 1:2
    - x[x>2]
      - Elements with values >2 are displayed

More About R – Vectors cont…

- Generating vector
  - 5:8
  - 5:1
  - Commands:
    - i<-2
    - 1:i-1 # this is 1:i, not 1: (i-1)
  - Vector sequences
    - seq(from=20, to=50, by=3) #for integer
    - seq(from=2.1, to = 3, length=10) # for non integer
    - seq(from=2.1, to = 3, by=0.2)
    - For for loop, we can write as:
      - for(i in 1:length(x)) or for(i in seq(x))
      - Try the following command to see why it works for for loop
        - x
        - seq(x)
        - x<-NULL
        - seq(x)

More About R – Vectors cont…

- Repeating vector constants
  - Type the following commands:
    - x<-rep(8,4)
    - x
    - rep(c(5,6,7),3)
    - rep(1:3, 2) #repeat 2 times
    - rep(c(1,2,3), each=2)
    - rep(1:4,times=c(2,3,2,3))
- Using all() and any(): report whether any or all of their arguments are TRUE
  - Type the following commands:
    - x<-1:30
    - any(x>8)
    - all(x>8)
    - all(x>0)
- Count the elements (combine length and which)
  - length(which(x>2))
  - (1:length(x))[x>2] # retrieve index of x with values >2

Exercise 2.2a – 15 minutes

1. Define the variable v1 as the vector (3.7, -4.0) and view the vector
2. Define the variable v2 as the vector (1,2,3,…,48,49,50)
3. Define the variable v3 as the vector (3.7, -4.0, 1,2,3,…,48,49,50) by combining v1 and v2
4. Sum over all elements of v1 and v2

1. v1 <- c(3.7, -4.0)
2. v2 <- c(1:50)
3. v3 <- c(v1, v2)
4. "+"(v1,v2)
   sum(v1,v2)

Exercise 2.2b – 20 minutes

- Create a vector with value (1,2,3) and assign it to temp.
  - Let(1,2,3…..1,2,3) occurs up to 5 times. Ans : rep(temp, 5)
  - Let(1,2,3) with 10 occurrences of 1, 20 occurrences of 2 and 30 occurrences of 3. Ans : rep(temp,times=c(10,20,30))
- Let say x<-c(7,8,12,11,9,7)
  - Extract the first element of x. Ans : x[1]
  - Extract the 2nd and 5th element of x. Ans : x[c(2,5)]
  - Extract all values in x except fourth element. Ans : x[-4]
- Write out the output of u <- c(TRUE, FALSE, TRUE, TRUE) without running R
  - !u Ans : FALSE TRUE FALSE FALSE
  - u | !u Ans : TRUE TRUE TRUE TRUE
  - any(u) Ans : TRUE

MULTIMEDIA UNIVERSITY

Microsoft IT Academy Program

Exercise 2.2c – 20 minutes

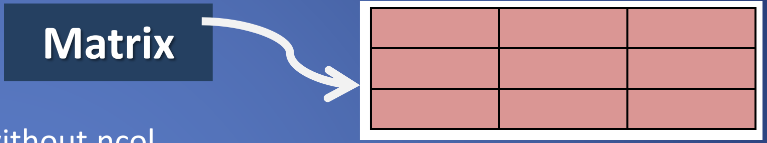
- Create vectors: xVec and yVec with the following commands:
  - set.seed(50)
  - xVec <- sample(0:999, 250, replace=T)
    - Choose random integers from 0-999 with replacement and length is 250
  - yVec <- sample(0:999, 250, replace=T)
- Answer the following question:
  - Display the values in yVec which is >600. Ans : yVec [yVec > 600]
  - What are the index position of elements in yVec which is >600?. Ans : (1:length(yVec))[yVec>600] or which (yVec>600)
  - How many numbers in xVec are divisible by 2? (Modulus operator is %%) Ans : length(which(xVec %% 2 == 0)) or sum(xVec %% 2 == 0)
  - Pick out the elements in yVec at index positions 1*;* 4*;* 7*;* 10*;* 13*;* …… [Hints: you need to apply indexing with logical value T/F] Ans : yVec[c(T,F,F)]

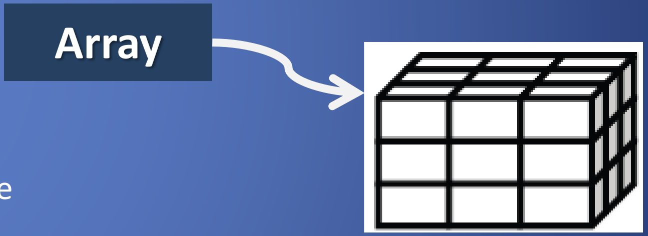
MULTIMEDIA UNIVERSITY

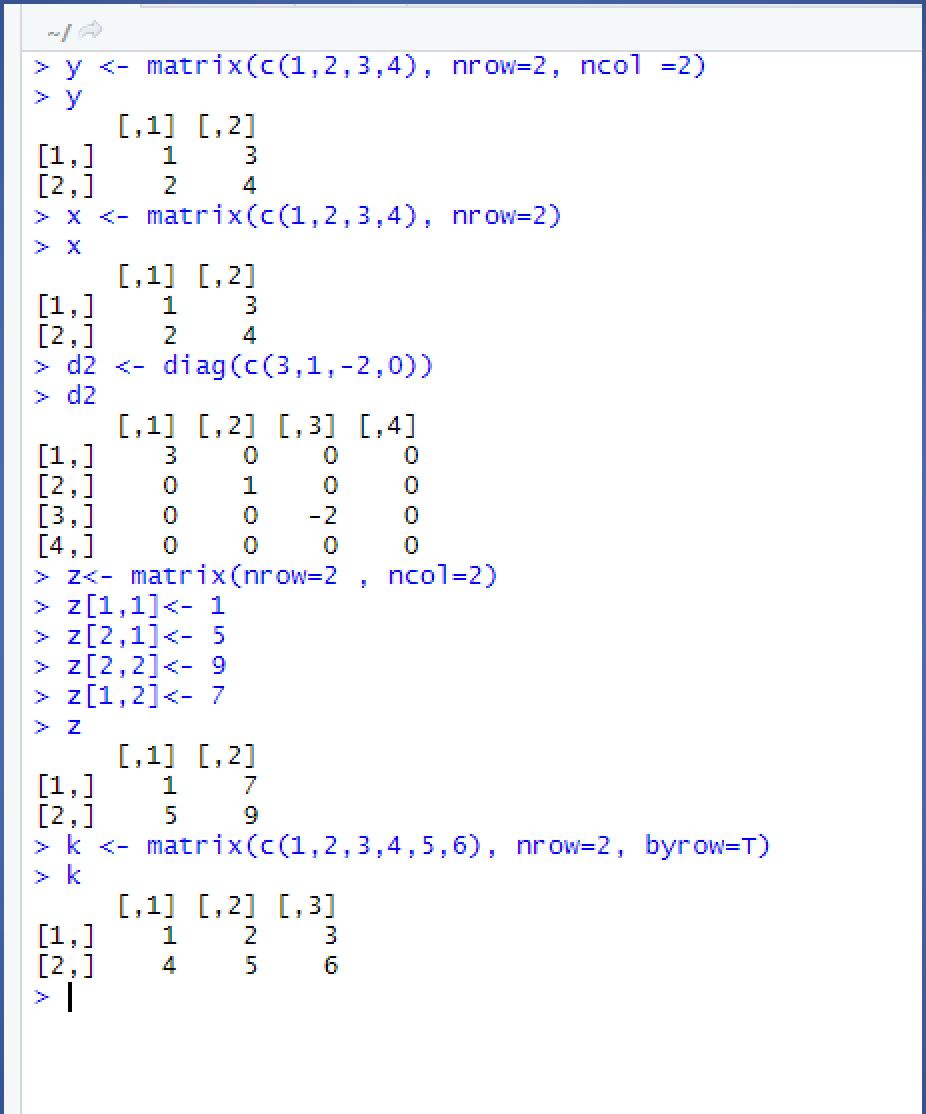
Microsoft® IT Academy Program

More About R - Matrices & Array

- Matrix: a vector with 2 additional attributes, no. of rows and no. of columns
- Creating matrix
  - Use matrix function
    - y<-matrix(c(1,2,3,4),nrow=2,ncol=2)
    - y
    - x<-matrix(c(1,2,3,4),nrow=2) # can declare without ncol
    - d2 <- diag(c(3, 1, -2, 0)) #use diagonal
  - Specify each element individually
    - z[1,1]<-1
    - z[2,1]<-2
    - z[1,2]<-3
    - z[2,2]<-4
  - To indicate row-major
    - k<-matrix(c(1,2,3,4,5,6),nrow=2,byrow=T) #byrow = True
    - k
  - To create a matrix with rows and columns name
    - cells <- c(1,26,24,68)
    - rnames <- c("R1", "R2")
    - cnames <- c("C1", "C2")
    - mymatrix <- matrix(cells, nrow=2, ncol=2, byrow=TRUE,dimnames=list(rnames, cnames))
    - mymatrix







```
> y <- matrix(c(1,2,3,4), nrow=2, ncol =2)
> y
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> x <- matrix(c(1,2,3,4), nrow=2)
> x
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> d2 <- diag(c(3,1,-2,0))
> d2
     [,1] [,2] [,3] [,4]
[1,]    3    0    0    0
[2,]    0    1    0    0
[3,]    0    0   -2    0
[4,]    0    0    0    0
> z<- matrix(nrow=2 , ncol=2)
> z[1,1]<- 1
> z[2,1]<- 5
> z[2,2]<- 9
> z[1,2]<- 7
> z
     [,1] [,2]
[1,]    1    7
[2,]    5    9
> k <- matrix(c(1,2,3,4,5,6), nrow=2, byrow=T)
> k
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
> 
```

More About R - Matrices & Array cont…

- Applying functions to matrix row and columns
  - General form of apply for matrices:
    - apply(m,dimcode,f,fargs)
    - m is the matrix
    - dimcode is the dimension, rows= 1, cols = 2
    - f is the function
    - fargs is an optional set of arguments to b supplied to f
    - Type the command:

```
apply(y,2,mean)
f<- function(g)   g/c(2,8)
apply(y,1,f)  #return a 3 by 2 result due to the nature of apply
t(apply(y,1,f) )  #apply transpose to get back the size of original matrix
```

MULTIMEDIA UNIVERSITY

Microsoft® IT Academy Program

More About R - Matrices & Array cont…

- Matrix operations
  - Inverse of matrix
    - solve(y) #must be square by square
  - Algebra operations
    - 3*y #multiply matrix with scalar
    - y+y #matrix addition
    - y%*%y #matrix multiplication
    - y^2 #power of
  - Changing the size of matrixes
    - rbind(x,y) #no of column of matrices must match
    - cbind(x,k) #no of rows of matrices must match
  - Name the matrix
    - rownames(x)<-c("a", "b")
    - colnames(x)<-c("c", "d")
    - x
  - Transpose a matrix: change row to column and column to row
    - t(k) # t() is the transpose function

More About R - Matrices & Array cont…

- Array – store in the form of matrices, rows and columns
- Can have more than 2 dimensions
- Create array:
  - vector1 <- c(2,9,3)
    vector2 <- c(10,16,17,13,11,15)
    result <- array(c(vector1,vector2),dim = c(3,3,2))

    result1<-array(c(vector1,vector2))
  - result creates two arrays, each with 3x3 matrix
  - result1 creates an array, 1x9 matrix
  - dim1 <- c("A1", "A2")
  - dim2 <- c("B1", "B2", "B3")
  - dim3 <- c("C1", "C2", "C3", "C4")
  - z <- array(1:24, c(2, 3, 4), dimnames=list(dim1, dim2, dim3))
  - z
- Print
  - result[3,,2] # print the 3rd row of the 2nd matrix
  - result[1,3,2] # print the 1st row 3rd col of the 2nd matrix

Exercises 2.3a - 20 minutes

- Assign variable Mynumber1 as 1,2,3,4
- Assign variable Mynumber2 as 11,12,13,14
- Build a column major matrix by combining both Mynumber1 and Mynumber2 and assign it as *a*
- Build a row major matrix by combining Mynumber1 and Mynumber2 and assign it as *b*
- Multiply both matrixes together. What is the result?

1. Mynumber1 <- 1:4
2. Mynumber2 <- 11:14
3. a <- cbind(Mynumber1, Mynumber2)
4. b <- rbind(Mynumber1, Mynumber2)
5. Mynumber1%*%Mynumber2

Exercise 2.3b – 15 minutes

- Based on the figure, create a 3 x 3 matrix and assign it o M1:

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 3 \\ 5 & 2 & 6 \\ -2 & -1 & -3 \end{bmatrix}$$

Ans : matrix(c(1,5,-2,1,2,-1,3,6,-3), nrow=3, ncol=3)

- F nd it to become a new matrix M3 as in the following figure:

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 3 \\ 5 & 2 & 6 \\ -2 & -1 & -3 \\ 4 & 3 & -1 \end{bmatrix}$$

- Name the columns Col3 and rows as A, B, C, D.
- Obtain the transpose of M3 and assign it to M1.
- By using apply and function(x) {sum(x>4)}, get the number of rows in M1 that is greater than 4.

```
M3 <- matrix(c(1,5,-2,4,1,2,-1,3,3,6,-3,-1), nrow=4, ncol=3)
M3 <- rbind(M1, c(4,3,-1))
rownames(M3) <- c("A", "B", "C", "D")
colnames(M3) <- c("Col1", "Co12", "Col3")
M1 <- t(M3)
apply (M1, 1, function(x) {sum(x>4)}
```

Exercise 2.3c – 10 minutes

- Generate an array with dimensions 4 x 3 x 2 with 1 till 24. Ans : a <- array(1:24, dim = c(4, 3, 2))
- Extract the last slice (a[, , 2]) of the array and save it as a matrix m. Ans: m<- a[,,2]

More About R – Lists

- List: a recursive vector that can contain elements of different types
- Can include vector, matrix, array, data frame, list and function as its elements
- Form the basis of data frame and object oriented programming
- Similar to Python dictionary, C struct of C programming
- Create list through list function
  - A<-list(name= "George", salary = 55000, union=T)
  - A
- Create list through vector
  - n = c(2, 3, 5)
  - s = c("aa", "bb", "cc", "dd", "ee")
  - b = c(TRUE, FALSE, TRUE, FALSE, FALSE)
  - x = list(n, s, b, 3) # x contains copies of n, s, b
- Print with different method
  - A$name #by using the name of the element
  - A[["name"]] #similar to A["name"]
  - A[[1]] #similar to A[1], where A[i], i is the index of c

More About R – Lists cont…

- Adding/deleting list elements
  - Add after a list is created:
    - A$ID<-"123456" #add a new component: ID
  - Add via vector index
    - A[[5]]<-"male"
  - Delete by setting it to NULL
    - A[[5]]<-NULL
- Getting the size
  - length(A)
- Let us create a random number matrix which the name of the column is created through Lists
  - rmat = matrix(rnorm(15),5,3, dimnames=list(NULL,c('A','B','C')))

Exercise 2.4 – 15 minutes

- Three students record the time spent on homework per class, Their data is:

– Marsha	25	0	45	90	0
– Bill	30	30	30	30	
– Holly	15	0	90	0	

  - Use a list to store these values.

```
Record1 <- list(name="Marsha", score=c(25,0,45,90,0))
Record2 <- list(name="Billy", score=c(30,30,30,30))
Record3 <- list(name="Holly", score=c(15,0,90,0))
MainRecord <- list(Record1, Record2, Record3)
```

More About R – Data Frames

- Data frames is like a matrix, with 2-dimensional rows and columns where each column may have different mode. However, within a column, all data will have similar mode.
- Can make changes to the data without changing the original data
- Characteristics of data frame:
  - Column name should be non-empty
  - Row names should be unique
  - Data can be of numeric, factor or character
  - Each column should contain same number of data items
- Example:

Weight	Head circumstances	Height
3.22	34	54
4.21	38	59
2.87	NA	58
2.33	24	49

  - eight<-c(3.22,4.21,2.87,2.33)
  - height<-c(54,59,58,49)
  - head.circum<-c(34,38,NA,24)
  - d<-data.frame(weight,head.circum,height,stringsAsFactors=FALSE)
  - #the false is to avoid character strings are treated as categorical data
  - d # to print out d
  - dim(d)
  - nrow(d)
  - ncol(d)

More About R – Data Frames cont…

- Accessing data frames
  - d[,1] #accessing 1st column
  - d$head.circum #accessing through name
- Get the structure of data frame:
  - str(d) #it tells the number of observations
- Extracting sub-data frames
  - d[2:3,] #extract row 2 to 3 only
  - d[d$weight>=3,] #extract row with weight>=3
  - d[d$weight,d$]
  - result<-data.frame(d$eight,d$height) #retrieve specific columns
  - result #print the value of result
  - d[c(2,4),c(1,3)] #retrieve 2nd and 4th row with 1st and 3rd column
- To add new row/column
  - rbind(d,list(3.51,35,56)) #add a new row
  - gender<-c("F","M","M","F") #declare a new column
  - cbind(d,gender) #add a new column

More About R – Data Frames cont…

- Transpose a data frame:
  - df<- data.frame(JOB_ID= c('AD_ASST','ST_MAN','PU_MAN', 'SH_CLERK'), DEPARTMENT_NAME = c('Administration','Shipping','Purchasing','Shipping'), SALARY=c(4400, 36400, 11000, 64300))
  - t(df)

Exercise 2.5a – 20 minutes

- Suppose you keep track of your mileage each time you fill up. At your last 6 fill-ups the mileage was 65311, 65624, 65908, 66219, 66499, 66821, 67145, and 67447. Enter these numbers into R. Use the function diff on the data. What does it give? Find the maximum number of miles between fill-ups. Find the mean and the minimum too.
- Suppose that 67145 is recorded wrongly. The correct value is 67130. How to fix the data?

```
R <- c(65311, 65624, 65908, 66219, 66499, 66821, 67145, 67447)
max (df)
mean (df)
min (df)
R[R=7] <- 67130

A<- list(c(65311, 65624, 65908, 66219, 66499, 66821, 67145, 67447))
Max(A[[1]])
A[[1

R<- data.frame(mileage=c(65311, 65624, 65908, 66219, 66499, 66821, 67145, 67447))
max(R$mileage)
R$mileage[7]
```

Exercise 2.5b – 20 minutes

- Your cell phone bill varies from month to month. Suppose your year has the following monthly amounts:
  - 44,33,39,37,46,30,48,32,49,35,30,48

Enter the data into a variable called bill. Find the total amount that you spend in this year. How many months was the amount greater than $40? What percentage was this?

```
bill <- c(44,33,39,37,46,30,48,32,49,35,30,48)
sum (bill)
a1 <- which (bill >40)
length(a1)
percent <- length(a1)/length(bill)*100
```

Exercise 2.5c

- Create a data frame with the following data:

x	y
Yes	Yes
Yes	No
No	No
Not sure	Yes
No	No

```
x <- c("Yes", "Yes", "No", "Not sure", "No")
y <- c("Yes", "No", "No", "Yes", "No")
d<-data.frame(x,y)
```

More About R – Functions & Basic Operators

- Functions: a group of instructions that
  - store as R object
  - takes inputs
  - to compute other values
  - returns a result
  - can be passed
  - can be nested
- Internal function such as c function c() is used to create vectors of objects by concatenating things together. c is similar to concatenate in R.
  - Simplest and quick way to enter data into R
- Function is an object too
- Function is developed through R script/text editor
- Handy function: lapply() and sapply(), functions to the columns of an array or data frame
- A simple function
  - Type the following command (do not type +, R will automatically include "+" when you press enter after "{") in script and click run (make sure your cursor is at the beginning of the script before Run):

```
f <-function() {
  cat("Hello, World \n")}  ##do not type +
f()   ##call the function in console
```

More About R – Functions & Basic Operators cont…

- A function with arguments:

  ```
  f <- function(num) {
    for(i in seq_len(num)){
     cat("Hello, world!\n")
    }
  }
  ```

  - Key in f(3) and observe the result

- Let us have a function which receive argument and return argument. Type the following command to R script and call the function with evenCount(c(1,2,3)), it should return number of even numbers in the vector.

  - evenCount<-function(x){

    ```
    k<-0   #assign zero to k
    for(n in x){
    if(n %% 2 == 0)  k<-k+1
    }
    ```

More About R – Functions & Basic Operators cont…

- Try the following command:

```
mean.and.sd<-function(x=1:10){
        av<-mean(x)
        sd<-sqrt(var(x))
        c(mean=av, SD=sd)
}
mean.and.sd() #call the function in console
```

- Assign a function to a new object:
  - Type the command:
    - letsCount<-evenCount
    - letsCount

More About R – Functions & Basic Operators cont…

- Try the following command with missing input argument

  -
    ```
    f<-function(a,b)
       {
       a^2
       }
    ```

  -
    ```
    f(2)    #any error?
    ```

  -
    ```
    f<-function(a,b)
       {
       print(a)
       print(b)
       }
    ```

  -
    ```
    f(2)   #any error? Can you explain why?
    ```

More About R – Functions & Basic Operators cont…

Operation	Description	Operation	Description
x + y	Addition	x <= y	Test for less than or equal to
x - y	Subtraction	x >= y	Test for greater than or equal to
x * y	Multiplication	x && y	Boolean AND for scalars
x / y	Division	x || y	Boolean OR for scalars
x ^ y	Exponentiation	x & y	Boolean AND for vectors (vector x, y, result)
x %% y	Modular arithmetic	x | y	Boolean OR for vectors (vector x, y, result)
x %/% y	Integer division	!x	Boolean negation
x == y	Test for equality		

Exercise 2.6 – 20 minutes

- Write a function which receive two arguments and return the summation of these arguments
  - Intermediate Level: create a main function that pre-store two arguments. These two arguments are then pass to the summation function and return the result.
- Write a function which convert Celsius to Fahrenheit. Here is the formula:
  - Fahrenheit = Celsius * (9/5) + 32