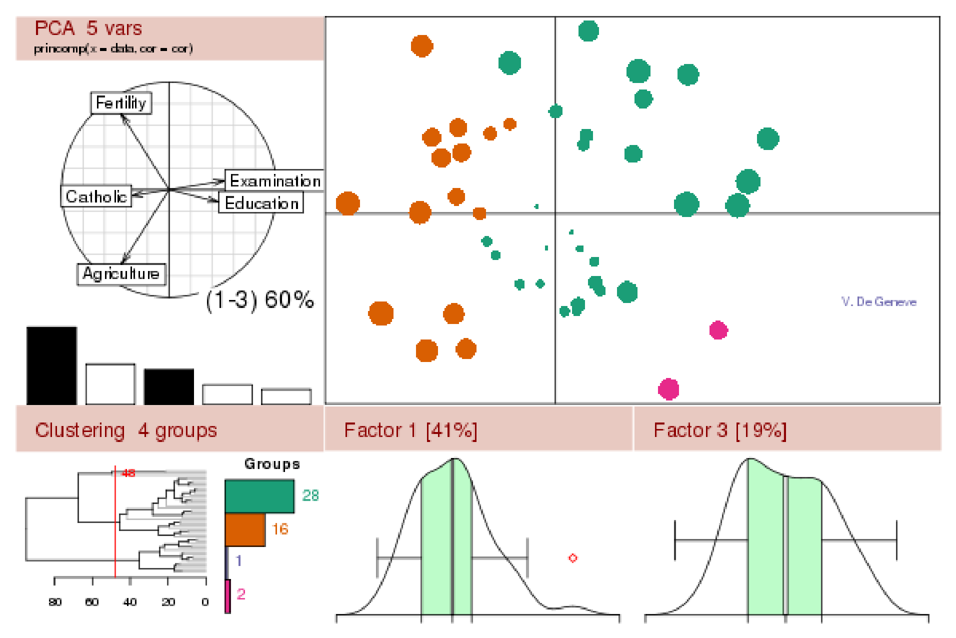
R Language

Compiled by Muhammad Faizan

R Introduction

- GNU Project Developed by John Chambers @ Bell Lab
- Free software environment for statistical computing and graphics
- Functional programming language written primarily in C, Fortran



R Technical Introduction

- R is functional programming language
- R is an interpreted language
- R is object oriented-language
- R works in an environment level

R Downloadable links

- Two sources to get R-environment
  - R-Project
  - R-Studio (Preferred)

Why we R?

I. For Statistical Analysis

II. For Data Visualization

III. For Mathematical Functions and modeling

Getting Started

- We will work as we go

- Declaring a variable
- > x = 11
- > print(x)
- > y <- 11
- > X // Error
- You can use ‘=‘ , ‘<-’ or ‘->’ to assign a variable

Check your variables

- You can see it in workspace section
- Or use the following command
  > ls()
- To remove a variable from Workspace memory
- > rm(x)

Variable name rule

- Object name can use characters, numbers or period
- But number may not occur first, you can use period as first character, but then you can expect it to be skipped when you call ‘>ls’ command
- a
- .a
- a.1
- 1a

String

- > string = “notice double quote”
- > string <- ‘it works with single quote too’

Numeric Operations

- > a + a
- > x – y
- > m / n
- > x^2
- > log(2)
- > sqrt(y)
- > exp(z)
- > log2(1024)
- > abs(-10)

Vectors and operation

- > v_number = c(1,2,3,4,5)
- > v_gender = c(‘male’ , ‘female’)
- > seq (from = 1, to = 10, by = 1)
- > rep (1, times =10)
- > rep (1:3, times = 2)

Vectors Operation & Extraction

- > x = seq(from =1 , to=10, by=2)
- > y = seq (from = 2, to 10, by=2)
- > x + y
- > x[1] # To extract first element
- > x[-1] # To extract all except first element
- > x[1:3] # To extract 1st three elements
- > x[c(1,3)] # To extract 1st and 3rd element
- > y[y < 6] # To extract element less than 6

Matrix

- > m_seq1 = matrix(1:9 , nrow=3, byrow =TRUE)
- > m_seq2 = matrix(1:9 , nrow=3, byrow =FALSE)

Reading Data

- > read.csv (file=“~/Dataset/titanic.csv” , header=TRUE, sep=‘,’)
- > read.csv2 (file=“~/Dataset/titanic.csv” , header=T)
- > read.csv2 (file.choose() , header=TRUE)
- > read.table (file.choose() , header=TRUE, sep=‘,’)

Testing Data

- > dim(data1)   # To check dimension of data
- > head(data1)   # To check first 6 entries of data
- > tail(data1)   # To check last 6 entries of data
- > data1[c(23,5,7,55),]   # To extract specific data
- > name(data1)   # To check field names
- > attach(cap)   # To make properties recognizable
- > mean(Age)   # To find Mean
- > median (Age)   # To find Median
- > detach (cap)   # To find Median

Getting Summary

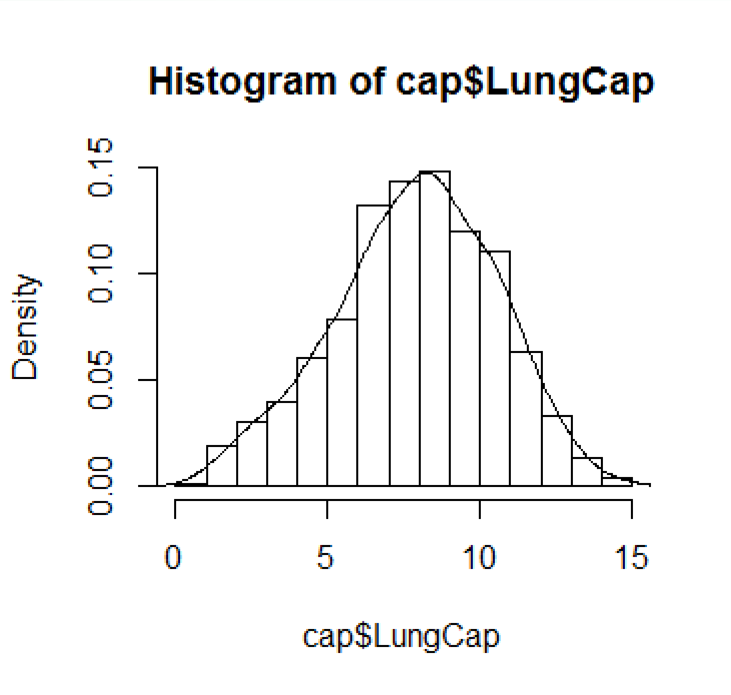
- You can ask for data Summary using
- > summary(data)
- You can also ask what values are available using levels.
- > class(data$value)
- > levels(data$value)
- > x = c(1,0,1,0,0,0,1,1,0)
- > x<- as.factor(x)
- > class(x)
- > summary(x)

If & blocks

- > attach(titanic_data)
- > summary(age[sex==“women”])
- > # Sub setting data
- > childData <- titanic_data[age==“child” & sex==“male”,]
- > detach(titanic_data)
- > areWomanandchild <- titanic_data$sex == "women" & titanic_data$age == "child"
- > titanic_data_with_classification <- cbind(titanic_data, areWomanandchild) #you can also use ‘rbind’ for row wise binding

Histogram

- > hist(cap$lungCap, prob =T)
- > lines(density(cap$lungCap))

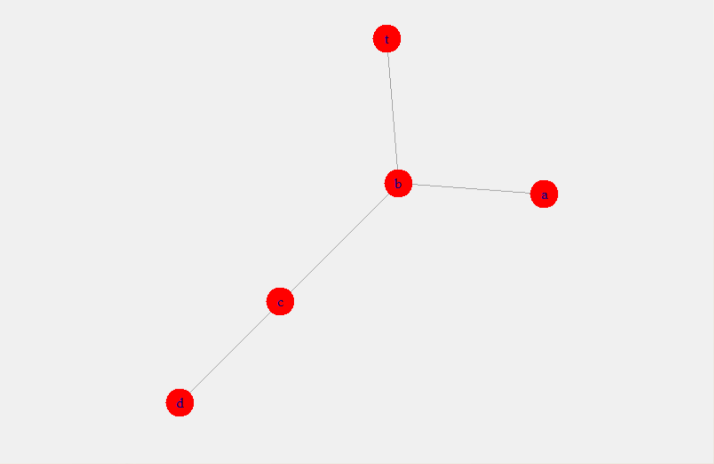


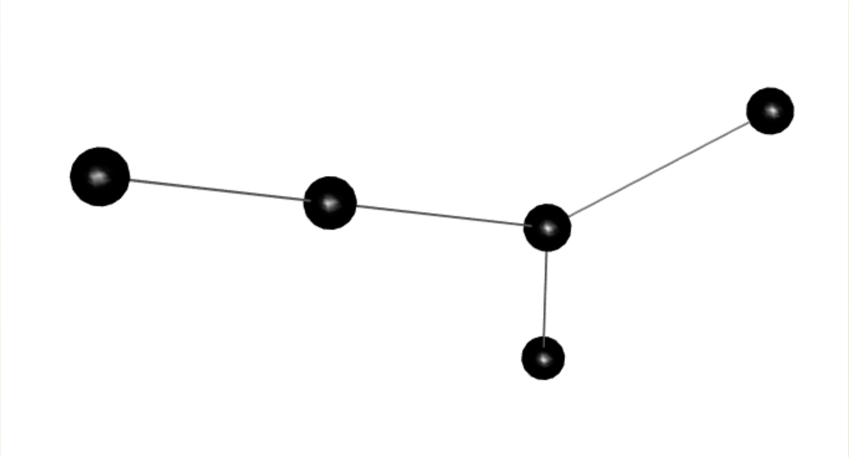
Installing and loading packages

- Packages are simple modules that provides common functionalities.
- Packages are open sourced as well so you can create your own and contribute
- > install.packages(‘igraph’)
- > library(igraph)
- > help(package = igraph)

Creating A simple graph

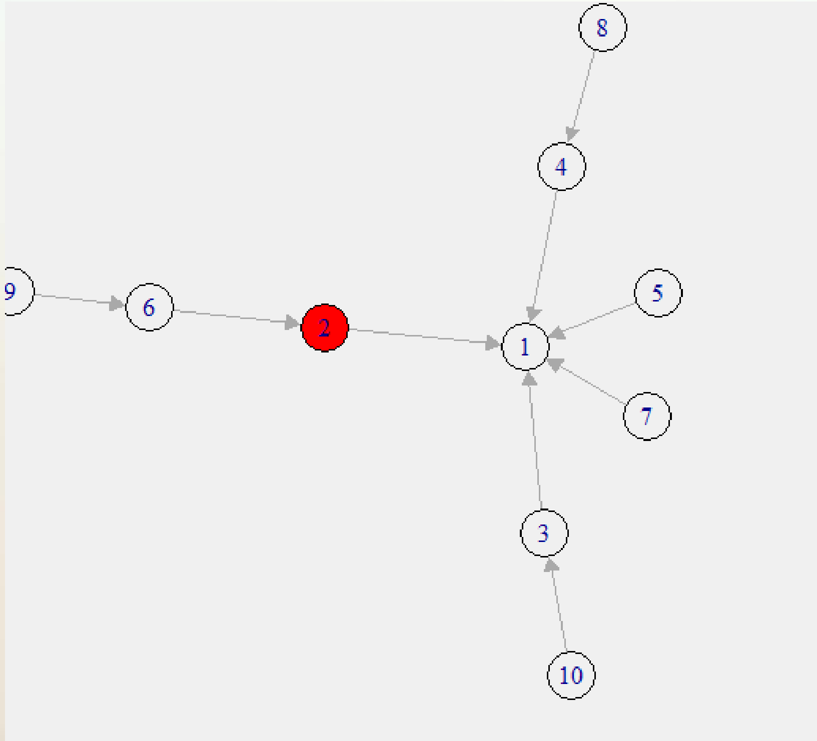
- > g.manual <- graph.formula(a—b, b—c, c—d, t—b)
- > tkplot(g.manual, vertex.color=“white”, vertex.size=15)
- > rglplot(g.manual)
- > plot(g.manual)





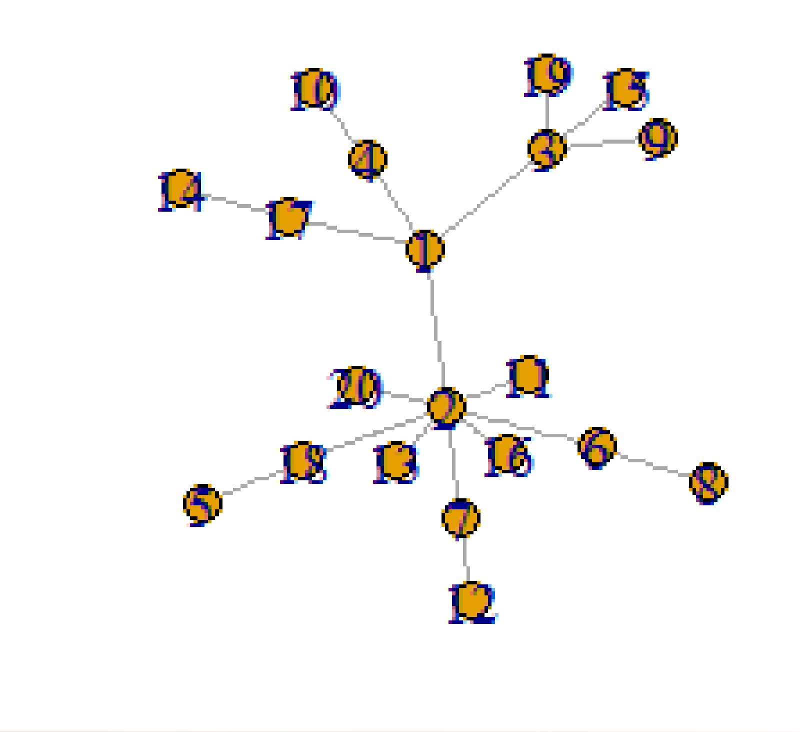
Taking out a node

- > g.barabasi = barabasi.game(n=10, p = 0.5)
- > V(g.barabasi)[which.max(betweenness(g.barabasi, v = V(g.barabasi)))]$color="red“
- > tkplot(v.barabasi)



Playing more

- > g.random = erdos.renyi.game(10, 0.3)
- > g.mst = minimum.spanning.tree(g.random)



You can find

- > closeness(g)
- > betweenness(g)
- > degree(g)
- Other important commands are
- > max(v)
- > min(v)
- > which.max(v)
- > which.min(v)

You can Also try

Function	Description
aging.prefatt.game	Evolving graph, based on preferential attachment and aging
bipartite.random.game	Generate Bi-partite graph using random model
degree.sequence.game	Generate random graph with given degree sequence
forest.fire.game	Evolve a graph based on fire spreading phenomena
graph.adjacency	Create a graph from adjacency matrix
graph.bipartite	Creates a bi-partite graph
graph.complementer	Creates a complementary graph for a given graphs
graph.empty	Creates an empty graph