DATA SCIENCE LABORATORY

LAB MANUAL

Year	**:**	**2016 - 2017**
Course Code	**:**	BCS101
Regulations	**:**	**IARE - R16**
Semester	**:**	**I**
Branch	**:**	**CSE**

Prepared By

Dr. Madhu Bala Myneni, Professor
Mr.Y SubbaRayudu, Assistant Professor

INSTITUTE OF AERONAUTICAL ENGINEERING
(Autonomous)
Dundigal, Hyderabad - 500 043

VISION AND MISSION OF THE DEPARTMENT

VISION

The Vision of the department is to produce competent graduates suitable for industries and organizations at global level including research and development with Social responsibility.

MISSION

To provide an open environment to foster professional and personal growth with a strong theoretical and practical background having an emphasis on hardware and software development making the graduates industry ready with social ethics.

Further the Department is to provide training and to partner with Global entities in education and research.

INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

DUNDIGAL – 500 043, HYDERABAD

COMPUTER SCIENCE AND ENGINEERING

Program Outcomes	
PO1	**Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO2	**Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO3	**Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO4	**Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5	**Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO6	**The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO7	**Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO8	**Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9	**Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO10	**Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO11	**Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO12	**Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.
Program Specific Outcomes	

PSO1	**Professional Skills:** The ability to research, understand and implement computer programs in the areas related to algorithms, system software, multimedia, web design, big data analytics, and networking for efficient analysis and design of computer-based systems of varying complexity.
PSO2	**Problem-Solving Skills:** The ability to apply standard practices and strategies in software project development using open-ended programming environments to deliver a quality product for business success.
PSO3	**Successful Career and Entrepreneurship:** The ability to employ modern computer languages, environments, and platforms in creating innovative career paths, to be an entrepreneur, and a zest for higher studies.

Attainment of Program Outcomes and Program Specific Outcomes

S.No	Experiment	Program Outcomes Attained	Program Specific Outcomes Attained
1	R AS CALCULATOR APPLICATION	PO1,PO4	PSO1
2	DESCRIPTIVE STATISTICS IN R	PO1,PO4	PSO1
3	READING AND WRITING DIFFERENT TYPES OF DATASETS	PO2,PO3	PSO1
4	VISUALIZATIONS	PO6,PO:11	PSO1
5	CORRELATION AND COVARIANCE	PO4,PO6	PSO1
6	REGRESSION MODEL	PO6,PO11	PSO1, PSO2
7	MULTIPLE REGRESSION MODEL	PO6,PO11	PSO1, PSO2
8	REGRESSION MODEL FOR PREDICTION	PO6,PO11	PSO1, PSO2
9	CLASSIFICATION MODEL	PO6,PO11	PSO1, PSO2
10	CLUSTERING MODEL	PO6,PO11	PSO1, PSO2

Mapping Course Objectives Leading To the Achievement of Program Outcomes

Course Objectives	Program Outcomes												Program Specific Outcomes		
	PO1	PO2	PO3	PO4	PO5	PO6	PO 7	PO8	PO9	PO10	PO1 1	PO1 2	PSO1	PSO2	PSO3
I	√		√	√											
II	√	√		√											
III				√		√									
IV						√					√				

Data Science Laboratory

SYLLABUS

SOFTWARE AND HARDWARE REQUIREMENTS FOR A BATCH OF 18 STUDENTS:

HARDWARE:

Desktop systems: 30 nos Printers: 02

SOFTWARE:

System Software : Windows 7. Application Software: MS Office.
Programming Languages : R Software
IDE : R Studio Software

DATA SCIENCE LABORATORY

I Semester: CSE								
Course Code	**Category**	**Hours / Week**			**Credits**	**Maximum Marks**		
BCS101	**Core**	**L**	**T**	**P**	**C**	**CIA**	**SEE**	**Total**
		-	-	3	2	30	70	100
Contact Classes: Nil	**Total Tutorials: Nil**	**Total Practical Classes: 36**			**Total Classes: 36**			

OBJECTIVES:

The course should enable the students to:

I. Illustrate R objects.
II. Make use of different types of datasets for analysis in R.
III. Define relations among variables using correlation and covariance analysis.
IV. Analyze and differentiate the data models for predictions using R.

LIST OF EXPERIMENTS

Week-1	**R AS CALCULATOR APPLICATION**

a. Using with and without R objects on console
b. Using mathematical functions on console
c. Write an R script, to create R objects for calculator application and save in a specified location in disk

Week-2	**DESCRIPTIVE STATISTICS IN R**

a. Write an R script to find basic descriptive statistics using summary, str, quartile function on mtcars & cars datasets.
b. Write an R script to find subset of dataset by using subset (), aggregate () functions on iris dataset.

Week-3	**READING AND WRITING DIFFERENT TYPES OF DATASETS**
a. Reading different types of data sets (.txt, .csv) from web and disk and writing in file in specific disk location. b. Reading Excel data sheet in R. c. Reading XML dataset in R.	
Week-4	**VISUALIZATIONS**
a. Find the data distributions using box and scatter plot. b. Find the outliers using plot. c. Plot the histogram, bar chart and pie chart on sample data.	
Week-5	**CORRELATION AND COVARIANCE**
a. Find the correlation matrix. b. Plot the correlation plot on dataset and visualize giving an overview of relationships among data on iris data. c. Analysis of covariance: variance (ANOVA), if data have categorical variables on iris data.	
Week-6	**REGRESSION MODEL**
Import a data from web storage. Name the dataset and now do Logistic Regression to find out relation between variables that are affecting the admission of a student in a institute based on his or her GRE score, GPA obtained and rank of the student. Also check the model is fit or not. require (foreign), require(MASS).	
Week-7	**MULTIPLE REGRESSION MODEL**
Apply multiple regressions, if data have a continuous independent variable. Apply on above dataset.	
Week-8	**REGRESSION MODEL FOR PREDICTION**
Apply regression Model techniques to predict the data on above dataset.	
Week-9	**CLASSIFICATION MODEL**
a. Install relevant package for classification. b. Choose classifier for classification problem. c. Evaluate the performance of classifier.	
Week-10	**CLUSTERING MODEL**
a. Clustering algorithms for unsupervised classification. b. Plot the cluster data using R visualizations.	
Reference Books:	
Yanchang Zhao, "R and Data Mining: Examples and Case Studies", Elsevier, 1st Edition, 2012.	
Web References:	
1. http://www.r-bloggers.com/how-to-perform-a-logistic-regression-in-r/ 2. http://www.ats.ucla.edu/stat/r/dae/rreg.htm 3. http://www.coastal.edu/kingw/statistics/R-tutorials/logistic.html 4. http://www.ats.ucla.edu/stat/r/data/binary.csv	

INDEX

EXPERIMENTS PROGRAMS

Week-1: R AS CALCULATOR APPLICATION

Objective:

a. **Using with and without R objects on console**
b. **Using Mathematical Functions on console**
c. **Write an R Script, to create R objects for calculator application and save in a specified location in disc.**

To perform the calculator application the list of arithmetic and logical operators available in R are shown in table 1&2.

Table1. Arithmetic Operators in R

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
^ or **	Exponentiation
x %%y	modulus (x mod y) 5%%2 is 1
x %/%y	integer division 5%/%2 is 2

Logical Operators in R

Operator	Description
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	exactly equal to
!=	not equal to
!x	Not x
x | y	x OR y
x & y	x AND y
isTRUE(x)	test if X is TRUE

a. Using with and without R objects on console

Arithmetic operations without R objects

```
>2+5          #Addition
[1] 7
>34-4         #Subtraction
[1] 30
>3*6          #Multiplication
[1] 18
>25/5         #Division
[1] 5
>25%%2        #Modulus
[1] 1
```

Arithmetic operations with R objects

```
>a=5
>b=2
>a+b          #Addition
[1] 7
>a-b          #Subtraction
[1] 3
>a*b          #Multiplication
[1] 10
>c=25
>c/a     #Division
[1] 5
>c%%2         #Modulus operation
[1] 1
```

b. Using Mathematical Functions on console

Built-in Functions:

```
pi                  #The value of pi (π), which is approximately 3.142.
x^y                 # The value of x is raised to the power of y, that is, xy.
sqrt(x)             # Square root of x.
abs(x)              # Absolute value of x.
factorial(x)        # Factorial of x.
log(x, base = n)    # Logarithm of x using base = n
log10(x)            #Logarithms of x to the base of 10
log2(x)             #Logarithms of x to the base of  2.
exp(x)             #Exponent of x.
cos(x)
sin(x)
tan(x)
acos(x)
asin(x)
```

atan(x)

Example:

```
> pi
[1] 3.14
> exp(3)          ## provides the cube of e
> log(1.4)        ## provides the natural logarithm of the number 1.4
> log10(1.4)      ## provides the log to the base of 10
> sqrt(16)        ## provides the square root of 16
```

c. Write an R Script, to create R objects for calculator application and save in a specified location in disc.

#Open a new R script and save with Calculator.r

R object creation:

Use variable name and use the symbol <- (which is formed by the "less than" symbol followed immediately by a hyphen) to assign a value to object.

 # #To create a comment line in R script.

#Create a R object of integer type and assign a value of 2.5

```
> x <- 5
# To know what is in a variable x
> x
[1] 5
# To see in the console
# To store a computed value in another variable y
> y <- 3*exp(x)
> x <- 3*exp(x)
```

 # Declare variables of different types:

```
my_numeric <- 42
my_character <- "forty-two"
my_logical <- FALSE
```

 # Check which type these variables have:

```
class(my_numeric)
class(my_character)
class(my_logical)
```

output:

 # Declare variables of different types:

```
> my_numeric <- 42
> my_character <- "forty-two"
> my_logical <- FALSE
```

> **# Check which type these variables have:**
> class(my_numeric)
[1] "numeric"
> class(my_character)
[1] "character"
> class(my_logical)
[1] "logical"

Week-2

DESCRIPTIVE STATISTICS IN R

Objective:

a. Write basic descriptive using str, summary, quartile functions on mtcars and cars dataset.
b. Write an R script to find subset of dataset by using subset (), aggregate () functions on iris dataset.

a) Write basic descriptive using str, summary, quartile functions on mtcars and cars dataset

str():

library(utils)

Display the internal structure of an R object. Ideally, only one line for each 'basic' structure is displayed. It is especially well suited to compactly display the (abbreviated) contents of (possibly nested) lists. The idea is to give reasonable output for any R object. It calls args for (non-primitive) function objects.

str(object, ...)

Description:

```
str(object, max.level = NA,
vec.len  = strO$vec.len, digits.d = strO$digits.d,
nchar.max = 128, give.attr = TRUE,
give.head = TRUE, give.length = give.head,
width = getOption("width"), nest.lev = 0,
indent.str = paste(rep.int(" ", max(0, nest.lev + 1)), collapse = ".."),
comp.str = "$ ", no.list = FALSE, envir = baseenv(),
strict.width = strO$strict.width,
formatNum = strO$formatNum, list.len = 99, ...)
```

Arguments

object	any **R** object about which you want to have some information.
max.level	maximal level of nesting which is applied for displaying nested structures, e.g., a list containing sub lists. Default NA: Display all nesting levels.
vec.len	numeric (>= 0) indicating how many 'first few' elements are displayed of each vector. The number is multiplied by different factors (from .5 to 3) depending on the kind of vector. Defaults to the vec.len component of option "str" (see options) which defaults to 4.
digits.d	number of digits for numerical components (as for print). Defaults to the digits.d component of option "str" which defaults to 3.

nchar.max	maximal number of characters to show for character strings. Longer strings are truncated, see longch example below.
give.attr	logical; if TRUE (default), show attributes as sub structures.
give.length	logical; if TRUE (default), indicate length (as [1:...]).
give.head	logical; if TRUE (default), give (possibly abbreviated) mode/class and length (as <type>[1:...]).
width	the page width to be used. The default is the currently activeoptions("width"); note that this has only a weak effect, unless strict.width is not "no".
nest.lev	current nesting level in the recursive calls to str.
indent.str	the indentation string to use.
comp.str	string to be used for separating list components.
no.list	logical; if true, no 'list of ...' nor the class are printed.
envir	the environment to be used for *promise* (see delayedAssign) objects only.
strict.width	string indicating if the width argument's specification should be followed strictly, one of the values c("no", "cut", "wrap"), which can be abbreviated. Defaults to the strict.widthcomponent of option "str" (see options) which defaults to"no" for back compatibility reasons; "wrap" usesstrwrap(*, width = width) whereas "cut" cuts directly to width. Note that a small vec.length may be better than setting strict.width = "wrap".
formatNum	a function such as format for formatting numeric vectors. It defaults to the formatNum component of option "str", see "Usage" of strOptions() above, which is almost back compatible to **R** <= 2.7.x, however, using formatC may be slightly better.
list.len	numeric; maximum number of list elements to display within a level.
...	potential further arguments (required for Method/Generic reasons).

Example: Structure of mtcars dataset using str()

```
> str(mtcars)
'data.frame':   32 obs. of  11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
 $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
 $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
 $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

Summary():

library(base)

Description:

summary is a generic function used to produce result summaries of the results of various model fitting functions. The function invokes particular methods which depend on the class of the first argument.

Usage:

summary(object, ...)
Default S3 method:
summary(object, ..., digits = max(3, getOption("digits")-3))
S3 method for class 'data.frame'
summary(object, maxsum = 7,digits = max(3, getOption("digits")-3), ...)

S3 method for class 'factor'
summary(object, maxsum = 100, ...)

S3 method for class 'matrix'
summary(object, ...)

Arguments:

object	an object for which a summary is desired.
maxsum	integer, indicating how many levels should be shown for factors.
digits	integer, used for number formatting with signif() (forsummary.default) or format() (for summary.data.frame).
...	additional arguments affecting the summary produced.

Example: Summary of mtcars dataset:

```
> summary(mtcars)
      mpg             cyl             disp             hp             drat             wt
 Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0   Min.   :2.760   Min.   :1.513
 1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5   1st Qu.:3.080   1st Qu.:2.581
 Median :19.20   Median :6.000   Median :196.3   Median :123.0   Median :3.695   Median :3.325
 Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7   Mean   :3.597   Mean   :3.217
 3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0   3rd Qu.:3.920   3rd Qu.:3.610
 Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0   Max.   :4.930   Max.   :5.424
      qsec             vs              am              gear            carb
 Min.   :14.50   Min.   :0.0000   Min.   :0.0000   Min.   :3.000   Min.   :1.000
 1st Qu.:16.89   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000
 Median :17.71   Median :0.0000   Median :0.0000   Median :4.000   Median :2.000
 Mean   :17.85   Mean   :0.4375   Mean   :0.4062   Mean   :3.688   Mean   :2.812
 3rd Qu.:18.90   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000
 Max.   :22.90   Max.   :1.0000   Max.   :1.0000   Max.   :5.000   Max.   :8.000
```

Quantile():

library(stats)

Description:

The generic function quantile produces sample quantiles corresponding to the given probabilities. The smallest observation corresponds to a probability of 0 and the largest to a probability of 1.

Usage:

quantile(x, ...)

Default S3 method:
quantile(x, probs = seq(0, 1, 0.25), na.rm = FALSE,names = TRUE, type = 7, ...)

Arguments:

x	numeric vector whose sample quantiles are wanted, or an object of a class for which a method has been defined (see also 'details'). NA and NaNvalues are not allowed in numeric vectors unless na.rm is TRUE.
probs	numeric vector of probabilities with values in *[0,1]*. (Values up to 2e-14outside that range are accepted and moved to the nearby endpoint.)
na.rm	logical; if true, any NA and NaN's are removed from x before the quantiles are computed.
names	logical; if true, the result has a names attribute. Set to FALSE for speedup with many probs.
type	an integer between 1 and 9 selecting one of the nine quantile algorithms detailed below to be used.
...	further arguments passed to or from other methods.

Example:

```
> x=1:10
> quantile(x)
   0%   25%   50%   75%  100%
 1.00  3.25  5.50  7.75 10.00

> quantile(mtcars$mpg)
    0%    25%    50%    75%   100%
10.400 15.425 19.200 22.800 33.900
```

b) Write basic descriptive using subset, aggregate on iris

library(base)
Subsetting Vectors, Matrices and Data Frames

Description:
Return subsets of vectors, matrices or data frames which meet conditions.

Usage:
subset(x, ...)

Default S3 method:
subset(x, subset, ...)

S3 method for class 'matrix'
subset(x, subset, select, drop = FALSE, ...)

S3 method for class 'data.frame'
subset(x, subset, select, drop = FALSE, ...)

Arguments:

x	object to be subsetted.
subset	logical expression indicating elements or rows to keep: missing values are taken as false.
select	expression, indicating columns to select from a data frame.
drop	passed on to [indexing operator.
...	further arguments to be passed to or from other methods.

Examples:

subset(airquality, Temp > 80, select = c(Ozone, Temp))

subset(airquality, Day == 1, select = -Temp)

subset(airquality, select = Ozone:Wind)

with(airquality, subset(Ozone, Temp > 80))

```
> subset(iris,iris$Sepal.Length>7)
    Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
103          7.1         3.0          5.9         2.1 virginica
106          7.6         3.0          6.6         2.1 virginica
108          7.3         2.9          6.3         1.8 virginica
110          7.2         3.6          6.1         2.5 virginica
118          7.7         3.8          6.7         2.2 virginica
119          7.7         2.6          6.9         2.3 virginica
123          7.7         2.8          6.7         2.0 virginica
126          7.2         3.2          6.0         1.8 virginica
130          7.2         3.0          5.8         1.6 virginica
131          7.4         2.8          6.1         1.9 virginica
132          7.9         3.8          6.4         2.0 virginica
136          7.7         3.0          6.1         2.3 virginica
```

aggregate():

library(stats)

Compute Summary Statistics of Data Subsets

Description

Splits the data into subsets, computes summary statistics for each, and returns the result in a convenient form.

Usage

aggregate(x, ...)

Default S3 method:
aggregate(x, ...)

S3 method for class 'data.frame'
aggregate(x, by, FUN, ..., simplify = TRUE)

S3 method for class 'formula'
aggregate(formula, data, FUN, ...,subset, na.action = na.omit)

S3 method for class 'ts'
aggregate(x, nfrequency = 1, FUN = sum, ndeltat = 1,ts.eps = getOption("ts.eps"), ...)

Arguments:

x	an R object.
by	a list of grouping elements, each as long as the variables in the data frame x. The elements are coerced to factors before use.
FUN	a function to compute the summary statistics which can be applied to all data subsets.
simplify	a logical indicating whether results should be simplified to a vector or matrix if possible.
formula	a formula, such as y ~ x or cbind(y1, y2) ~ x1 + x2, where the y variables are numeric data to be split into groups according to the grouping x variables (usually factors).
data	a data frame (or list) from which the variables in formula should be taken.
subset	an optional vector specifying a subset of observations to be used.
na.action	a function which indicates what should happen when the data contain NA values. The default is to ignore missing values in the given variables.
nfrequency	new number of observations per unit of time; must be a divisor of the frequency of x.
ndeltat	new fraction of the sampling period between successive observations; must be a divisor of the sampling interval of x.
ts.eps	tolerance used to decide if nfrequency is a sub-multiple of the original frequency.
...	further arguments passed to or used by methods.

Examples:

Formulas, one ~ one, one ~ many, many ~ one, and many ~ many:

aggregate(weight ~ feed, data = chickwts, mean)
aggregate(breaks ~ wool + tension, data = warpbreaks, mean)
aggregate(cbind(Ozone, Temp) ~ Month, data = airquality, mean)
aggregate(cbind(ncases, ncontrols) ~ alcgp + tobgp, data = esoph, sum)
Dot notation:
aggregate(. ~ Species, data = iris, mean)

```
> aggregate(. ~ Species, data = iris, mean)
     Species Sepal.Length Sepal.Width Petal.Length Petal.Width
1     setosa        5.006       3.428        1.462       0.246
2 versicolor        5.936       2.770        4.260       1.326
3  virginica        6.588       2.974        5.552       2.026
```

```
> aggregate(Sepal.Length ~ Species, data = iris, mean)
     Species Sepal.Length
1     setosa        5.006
2 versicolor        5.936
3  virginica        6.588
```

Week-3

READING AND WRITING DIFFERENT TYPES OF DATASETS

Objective:

a. Reading different types of data sets (.txt, .csv) from web and disk and writing in file in specific disk location.
b. Reading Excel data sheet in R.
c. Reading XML dataset in R.

a). Reading different types of data sets(.txt, .csv) from web and disk and writing in file in specific disk location

read.table()

library(utils)
Data Input

Description:
Reads a file in table format and creates a data frame from it, with cases corresponding to lines and variables to fields in the file.

Usage:

```
read.table(file, header = FALSE, sep = "", quote = "\"",
           dec = ".", numerals = c("allow.loss", "warn.loss", "no.loss"),
           row.names, col.names, as.is = !stringsAsFactors,
           na.strings = "NA", colClasses = NA, nrows = -1,
           skip = 0, check.names = TRUE, fill = !blank.lines.skip,
           strip.white = FALSE, blank.lines.skip = TRUE,
           comment.char = "#",
           allowEscapes = FALSE, flush = FALSE,
           stringsAsFactors = default.stringsAsFactors(),
           fileEncoding = "", encoding = "unknown", text, skipNul = FALSE)

read.csv(file, header = TRUE, sep = ",", quote = "\"",
         dec = ".", fill = TRUE, comment.char = "", ...)

read.csv2(file, header = TRUE, sep = ";", quote = "\"",
          dec = ",", fill = TRUE, comment.char = "", ...)

read.delim(file, header = TRUE, sep = "\t", quote = "\"",
           dec = ".", fill = TRUE, comment.char = "", ...)

read.delim2(file, header = TRUE, sep = "\t", quote = "\"",
            dec = ",", fill = TRUE, comment.char = "", ...)
```

Arguments:

file	the name of the file which the data are to be read from. Each row of the table appears as one line of the file. If it does not contain an *absolute* path, the file name is *relative*to the current working directory, getwd(). Tilde-expansion is performed where supported. This can be a compressed file (see file). Alternatively, file can be a readable text-mode connection(which will be opened for reading if necessary, and if soclosed (and hence destroyed) at the end of the function call). (If stdin() is used, the prompts for lines may be somewhat confusing. Terminate input with a blank line or an EOF signal, Ctrl-D on Unix and Ctrl-Z on Windows. Any pushback on stdin() will be cleared before return.) file can also be a complete URL. (For the supported URL schemes, see the 'URLs' section of the help for url.)
header	a logical value indicating whether the file contains the names of the variables as its first line. If missing, the value is determined from the file format: header is set to TRUE if and only if the first row contains one fewer field than the number of columns.
sep	the field separator character. Values on each line of the file are separated by this character. If sep = "" (the default forread.table) the separator is 'white space', that is one or more spaces, tabs, newlines or carriage returns.
quote	the set of quoting characters. To disable quoting altogether, use quote = "". See scan for the behaviour on quotes embedded in quotes. Quoting is only considered for columns read as character, which is all of them unlesscolClasses is specified.
dec	the character used in the file for decimal points.
numerals	string indicating how to convert numbers whose conversion to double precision would lose accuracy, seetype.convert. Can be abbreviated.
row.names	a vector of row names. This can be a vector giving the actual row names, or a single number giving the column of the table which contains the row names, or character string giving the name of the table column containing the row names. If there is a header and the first row contains one fewer field than the number of columns, the first column in the input is used for the row names. Otherwise if row.names is missing, the rows are numbered. Using row.names = NULL forces row numbering. Missing or NULL row.names generate row names that are considered to be 'automatic' (and not preserved byas.matrix).
col.names	a vector of optional names for the variables. The default is to use "V" followed by the column number.
as.is	the default behavior of read.table is to convert character variables (which are

	not converted to logical, numeric or complex) to factors. The variable as.is controls the conversion of columns not otherwise specified bycolClasses. Its value is either a vector of logicals (values are recycled if necessary), or a vector of numeric or character indices which specify which columns should not be converted to factors. Note: to suppress all conversions including those of numeric columns, set colClasses = "character". Note that as.is is specified per column (not per variable) and so includes the column of row names (if any) and any columns to be skipped.
na.strings	a character vector of strings which are to be interpreted asNA values. Blank fields are also considered to be missing values in logical, integer, numeric and complex fields.
colClasses	character. A vector of classes to be assumed for the columns. Recycled as necessary. If named and shorter than required, names are matched to the column names with unspecified values are taken to be NA. Possible values are NA (the default, when type.convertis used), "NULL" (when the column is skipped), one of the atomic vector classes (logical, integer, numeric, complex, character, raw), or "factor", "Date" or "POSIXct". Otherwise there needs to be an as method (from package**methods**) for conversion from "character" to the specified formal class. Note that colClasses is specified per column (not per variable) and so includes the column of row names (if any).
nrows	integer: the maximum number of rows to read in. Negative and other invalid values are ignored.
skip	integer: the number of lines of the data file to skip before beginning to read data.
check.names	logical. If TRUE then the names of the variables in the data frame are checked to ensure that they are syntactically valid variable names. If necessary they are adjusted (bymake.names) so that they are, and also to ensure that there are no duplicates.
fill	logical. If TRUE then in case the rows have unequal length, blank fields are implicitly added. See 'Details'.
strip.white	logical. Used only when sep has been specified, and allows the stripping of leading and trailing white space from unquoted character fields (numeric fields are always stripped). See scan for further details (including the exact meaning of 'white space'), remembering that the columns may include the row names.
blank.lines.skip	logical: if TRUE blank lines in the input are ignored.
comment.char	character: a character vector of length one containing a single character or an empty string. Use "" to turn off the interpretation of comments altogether.

allowEscapes	logical. Should C-style escapes such as \n be processed or read verbatim (the default)? Note that if not within quotes these could be interpreted as a delimiter (but not as a comment character). For more details see scan.
flush	logical: if TRUE, scan will flush to the end of the line after reading the last of the fields requested. This allows putting comments after the last field.
stringsAsFactors	logical: should character vectors be converted to factors? Note that this is overridden by as.is and colClasses, both of which allow finer control.
fileEncoding	character string: if non-empty declares the encoding used on a file (not a connection) so the character data can be re-encoded. See the 'Encoding' section of the help for file, the 'R Data Import/Export Manual' and 'Note'.
encoding	encoding to be assumed for input strings. It is used to mark character strings as known to be in Latin-1 or UTF-8 (seeEncoding): it is not used to re-encode the input, but allows**R** to handle encoded strings in their native encoding (if one of those two). See 'Value' and 'Note'.
text	character string: if file is not supplied and this is, then data are read from the value of text via a text connection. Notice that a literal string can be used to include (small) data sets within R code.
skipNul	logical: should nuls be skipped?
...	Further arguments to be passed to read.table.

Example: Reading dataset from disk by choosing location dynamically

```
> read.csv(file.choose())
                X Garden Hedgerow Parkland Pasture Woodland
1       Blackbird     47       10       40       2        2
2       Chaffinch     19        3        5       0        2
3        GreatTit     50        0       10       7        0
4 HouseSparrow       46       16        8       4        0
5          Robin      9        3        0       0        2
6     SongThrush      4        0        6       0        0
```

Reading dataset from disk by specifying file path

>chennaifloodtweets=read.csv("C:/Users/rohit/Desktop/chennaifloods.csv")

b). Reading Excel data sheet in R

XLConnect: It might be slow for large dataset but very powerful otherwise.

require (XLConnect)

wb <- loadWorkbook("myfile.xlsx")

myDf <- readWorksheet(wb, sheet = "Sheet1", header = TRUE)

xlsx: Prefer the read.xlsx2() over read.xlsx(), it's significantly faster for large dataset.

require(xlsx)

read.xlsx2("myfile.xlsx", sheetName = "Sheet1")

c). Reading XML dataset in R

```
> install.packages("XML")
>install.packages("plyr")
>library("plyr", lib.loc="~/R/win-library/3.2")
> library("XML", lib.loc="~/R/win-library/3.2")
> fileurl<-"http://www.w3schools.com/xml/simple.xml"
> doc<-xmlParse(fileurl,useInternalNodes=TRUE)
> class(doc)
[1] "XMLInternalDocument" "XMLAbstractDocument"
> doc
<?xml version="1.0" encoding="UTF-8"?>
<breakfast_menu>
    <food>
        <name>Belgian Waffles</name>
        <price>$5.95</price>
        <description>Two of our famous Belgian Waffles with plenty of real maple
      syrup</description>
        <calories>650</calories>
    </food>
    <food>
        <name>Strawberry Belgian Waffles</name>
        <price>$7.95</price>
        <description>Light Belgian waffles covered with strawberries and whipped
            cream</description>
        <calories>900</calories>
    </food>
    <food>
        <name>Berry-Berry Belgian Waffles</name>
        <price>$8.95</price>
        <description>Light Belgian waffles covered with an assortment of fresh berries and
        whipped cream</description>
        <calories>900</calories>
    </food>
   <food>
        <name>French Toast</name>
        <price>$4.50</price>
        <description>Thick slices made from our homemade sourdough bread</description>
        <calories>600</calories>
   </food>
  <food>
```

```
        <name>Homestyle Breakfast</name>
         <price>$6.95</price>
         <description>Two eggs, bacon or sausage, toast, and our ever-popular hash
        browns</description>
          <calories>950</calories>
    </food>
</breakfast_menu>
```

XML to List

```
> xl<-xmlToList(doc)
> class(xl)
[1] "list"
> xl
$food
$food$name
[1] "Belgian Waffles"

$food$price
[1] "$5.95"

$food$description
[1] "Two of our famous Belgian Waffles with plenty of real maple syrup"

$food$calories
[1] "650"

$food
$food$name
[1] "Strawberry Belgian Waffles"
$food$price
[1] "$7.95"

$food$description
[1] "Light Belgian waffles covered with strawberries and whipped cream"

$food$calories
[1] "900"

$food
$food$name
[1] "Berry-Berry Belgian Waffles"

$food$price
[1] "$8.95"

$food$description
[1] "Light Belgian waffles covered with an assortment of fresh berries and whipped cr
```

$food$calories
[1] "900"

$food
$food$name
[1] "French Toast"

$food$price
[1] "$4.50"

$food$description
[1] "Thick slices made from our homemade sourdough bread"

$food$calories
[1] "600"

$food
$food$name
[1] "Homestyle Breakfast"

$food$price
[1] "$6.95"

$food$description
[1] "Two eggs, bacon or sausage, toast, and our ever-popular hash browns"

$food$calories
[1] "950"

> data<-ldply(xl,data.frame)

> head(data)

	id	Name Price
1	food	Belgian Waffles $5.95
2	food	Strawberry Belgian Waffles $7.95
3	food Berry-	Berry Belgian Waffles $8.95
4	food	French Toast $4.50
5	food	Homestyle Breakfast $6.95

	Description
1	Two of our famous Belgian Waffles with plenty of real maple syrup
2	Light Belgian waffles covered with strawberries and whipped cream
3	Light Belgian waffles covered with an assortment of fresh berries and whipped cream
4	Thick slices made from our homemade sourdough bread
5	Two eggs, bacon or sausage, toast, and our ever-popular hash browns

Calories

1	650
2	900
3	900
4	600
5	950

Week-4

VISUALIZATIONS

Objective:

a. Find the data distributions using box and scatter plot.
b. Find the outliers using plot.
c. Plot the histogram, bar chart and pie chart on sample data.

a) Find the data distributions using box and scatter plot

boxplot
library(graphics)

Description:

Produce box-and-whisker plot(s) of the given (grouped) values.

Usage:

```
boxplot(x, ...)

## S3 method for class 'formula'
boxplot(formula, data = NULL, ..., subset, na.action = NULL)

## Default S3 method:
boxplot(x, ..., range = 1.5, width = NULL, varwidth = FALSE,
        notch = FALSE, outline = TRUE, names, plot = TRUE,
        border = par("fg"), col = NULL, log = "",
        pars = list(boxwex = 0.8, staplewex = 0.5, outwex = 0.5),
        horizontal = FALSE, add = FALSE, at = NULL)
```

Arguments:

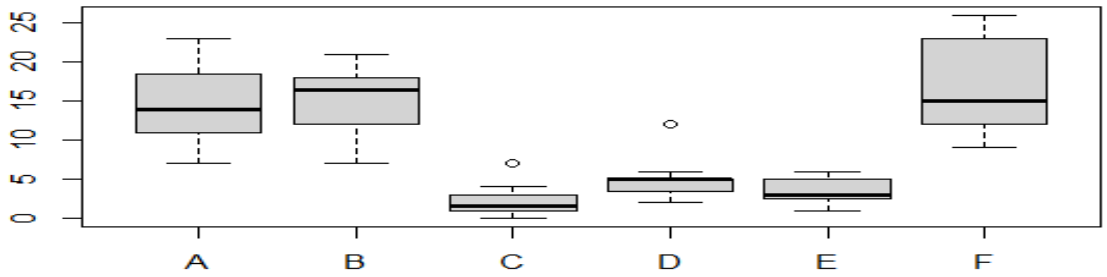
formula	a formula, such as y ~ grp, where y is a numeric vector of data values to be split into groups according to the grouping variable grp (usually a factor).
data	a data.frame (or list) from which the variables in formula should be taken.
subset	an optional vector specifying a subset of observations to be used for plotting.
na.action	a function which indicates what should happen when the data contain NAs. The default is to ignore missing values in either the response or the group.
x	for specifying data from which the boxplots are to be produced. Either a numeric vector, or a single list containing such vectors. Additional unnamed arguments specify further data as separate vectors (each corresponding to a

	component boxplot). NAs are allowed in the data.
...	For the formula method, named arguments to be passed to the default method. For the default method, unnamed arguments are additional data vectors (unless x is a list when they are ignored), and named arguments are arguments and graphical parameters to be passed to bxp in addition to the ones given by argument pars(and override those in pars). Note that bxp may or may not make use of graphical parameters it is passed: see its documentation.
range	this determines how far the plot whiskers extend out from the box. If range is positive, the whiskers extend to the most extreme data point which is no more thanrange times the interquartile range from the box. A value of zero causes the whiskers to extend to the data extremes.
width	a vector giving the relative widths of the boxes making up the plot.
varwidth	if varwidth is TRUE, the boxes are drawn with widths proportional to the square-roots of the number of observations in the groups.
notch	if notch is TRUE, a notch is drawn in each side of the boxes. If the notches of two plots do not overlap this is 'strong evidence' that the two medians differ (Chambers*et al*, 1983, p. 62). See boxplot.stats for the calculations used.
outline	if outline is not true, the outliers are not drawn (as points whereas S+ uses lines).

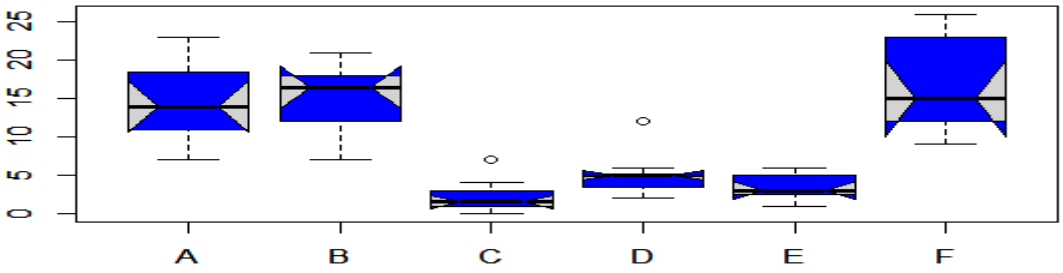
Examples:

boxplot on a formula:

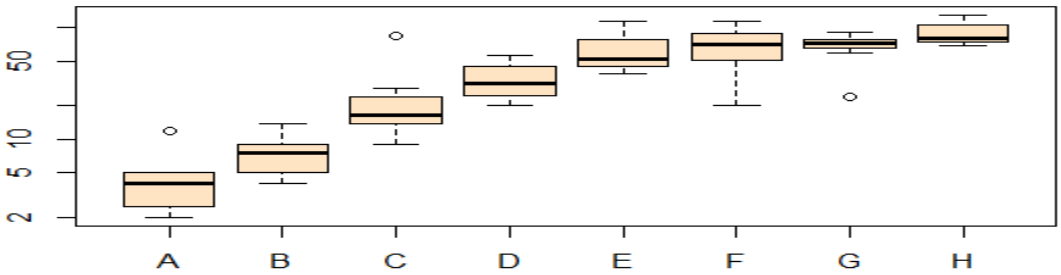
```
boxplot(count ~ spray, data = InsectSprays, col = "lightgray")
```



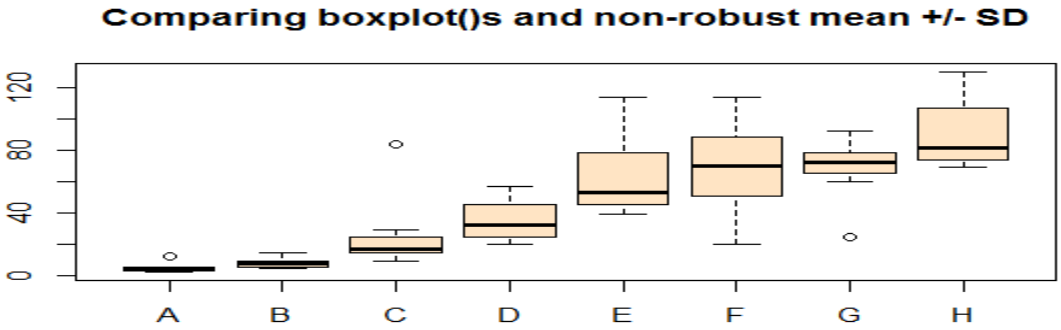

```
# *add* notches (somewhat funny here):
boxplot(count ~ spray, data = InsectSprays, notch = TRUE, add = TRUE, col = "blue")
```



boxplot(decrease ~ treatment, data = OrchardSprays,log = "y", col = "bisque")



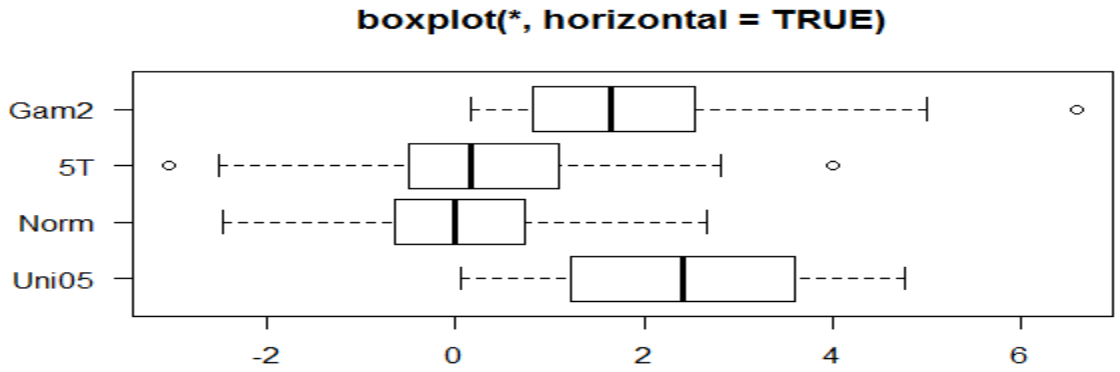
rb <- boxplot(decrease ~ treatment, data = OrchardSprays, col = "bisque")
title("Comparing boxplot()s and non-robust mean +/- SD")



mn.t <- tapply(OrchardSprays$decrease, OrchardSprays$treatment, mean)
sd.t <- tapply(OrchardSprays$decrease, OrchardSprays$treatment, sd)
xi <- 0.3 + seq(rb$n)
points(xi, mn.t, col = "orange", pch = 18)
arrows(xi, mn.t - sd.t, xi, mn.t + sd.t,
code = 3, col = "pink", angle = 75, length = .1)

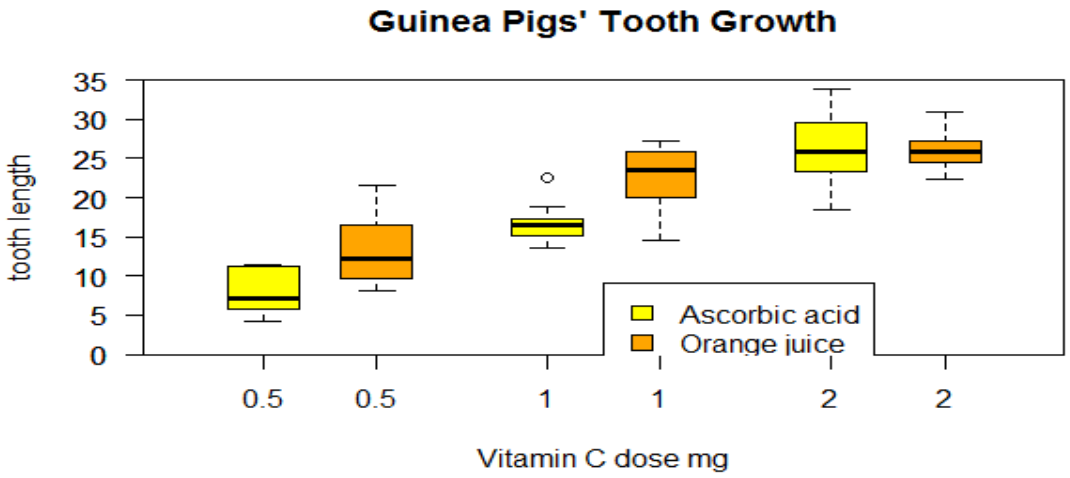
boxplot on a matrix:

```
mat <- cbind(Uni05 = (1:100)/21, Norm = rnorm(100),
`5T` = rt(100, df = 5), Gam2 = rgamma(100, shape = 2))
boxplot(as.data.frame(mat),
main = "boxplot(as.data.frame(mat), main = ...)")
par(las = 1) # all axis labels horizontal
boxplot(as.data.frame(mat), main = "boxplot(*, horizontal = TRUE)",
horizontal = TRUE)
```



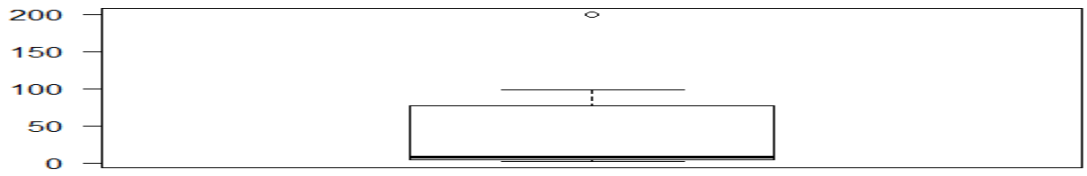
Using 'at = ' and adding boxplots -- example idea by Roger Bivand :

```
boxplot(len ~ dose, data = ToothGrowth,
      boxwex = 0.25, at = 1:3 - 0.2,
      subset = supp == "VC", col = "yellow",
      main = "Guinea Pigs' Tooth Growth",
      xlab = "Vitamin C dose mg",
      ylab = "tooth length",
      xlim = c(0.5, 3.5), ylim = c(0, 35), yaxs = "i")
      boxplot(len ~ dose, data = ToothGrowth, add = TRUE,
      boxwex = 0.25, at = 1:3 + 0.2,
      subset = supp == "OJ", col = "orange")
      legend(2, 9, c("Ascorbic acid", "Orange juice"),
      fill = c("yellow", "orange"))
```



c) Find the outliers using plot

```
####outliers
v=c(2,5,3,7,10,8,4,77,99,200)
boxplot(v)
```



d) Plot the histogram, bar chart and pie chart on sample data

hist()

library(graphics)

Description:

The generic function hist computes a histogram of the given data values. If plot = TRUE, the resulting object of class "histogram" is plotted by plot.histogram, before it is returned.

Usage:

```
hist(x, ...)
## Default S3 method:
hist(x, breaks = "Sturges",
    freq = NULL, probability = !freq,
```

include.lowest = TRUE, right = TRUE,
density = NULL, angle = 45, col = NULL, border = NULL,
main = paste("Histogram of" , xname),
xlim = range(breaks), ylim = NULL,
xlab = xname, ylab,
axes = TRUE, plot = TRUE, labels = FALSE,
nclass = NULL, warn.unused = TRUE, ...)

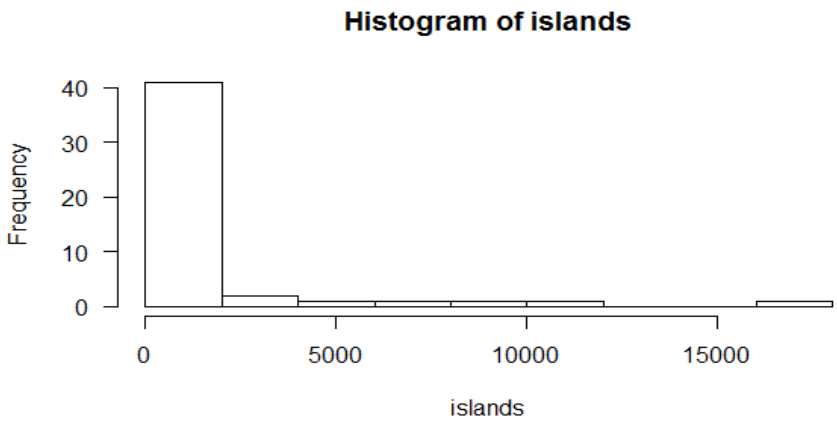
Arguments:

x	a vector of values for which the histogram is desired.
breaks	one of: • a vector giving the breakpoints between histogram cells, • a function to compute the vector of breakpoints, • a single number giving the number of cells for the histogram, • a character string naming an algorithm to compute the number of cells (see 'Details'), • a function to compute the number of cells. In the last three cases the number is a suggestion only; the breakpoints will be set to pretty values. If breaks is a function, the x vector is supplied to it as the only argument.
freq	logical; if TRUE, the histogram graphic is a representation of frequencies, thecounts component of the result; if FALSE, probability densities, componentdensity, are plotted (so that the histogram has a total area of one). Defaults to TRUE *if and only if* breaks are equidistant (and probabilityis not specified).
probability	an *alias* for !freq, for S compatibility.
include.lowest	logical; if TRUE, an x[i] equal to the breaks value will be included in the first (or last, for right = FALSE) bar. This will be ignored (with a warning) unless breaks is a vector.
right	logical; if TRUE, the histogram cells are right-closed (left open) intervals.
density	the density of shading lines, in lines per inch. The default value of NULLmeans that no shading lines are drawn. Non-positive values of density also inhibit the drawing of shading lines.
angle	the slope of shading lines, given as an angle in degrees (counter-clockwise).
col	a colour to be used to fill the bars. The default of NULL yields unfilled bars.
border	the color of the border around the bars. The default is to use the standard foreground color.

main, xlab, ylab	these arguments to title have useful defaults here.
xlim, ylim	the range of x and y values with sensible defaults. Note that xlim is *not*used to define the histogram (breaks), but only for plotting (when plot = TRUE).
axes	logical. If TRUE (default), axes are draw if the plot is drawn.
plot	logical. If TRUE (default), a histogram is plotted, otherwise a list of breaks and counts is returned. In the latter case, a warning is used if (typically graphical) arguments are specified that only apply to the plot = TRUEcase.
labels	logical or character string. Additionally draw labels on top of bars, if notFALSE; see plot.histogram.
nclass	numeric (integer). For S(-PLUS) compatibility only, nclass is equivalent tobreaks for a scalar or character argument.
warn.unused	logical. If plot = FALSE and warn.unused = TRUE, a warning will be issued when graphical parameters are passed to hist.default().
...	further arguments and graphical parameters passed to plot.histogramand thence to title and axis (if plot = TRUE).

Examples:

```
op <- par(mfrow = c(2, 2))
hist(islands)
```

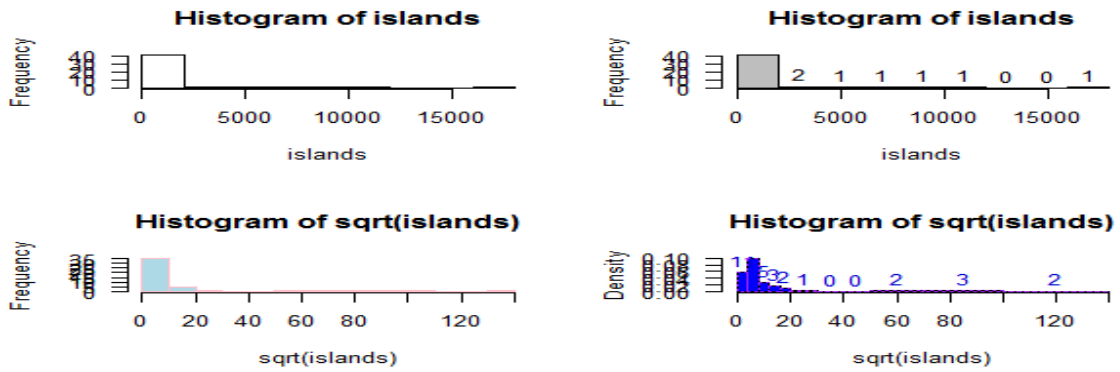



```
utils::str(hist(islands, col = "gray", labels = TRUE))

hist(sqrt(islands), breaks = 12, col = "lightblue", border = "pink")
##-- For non-equidistant breaks, counts should NOT be graphed unscaled:
r <- hist(sqrt(islands), breaks = c(4*0:5, 10*3:5, 70, 100, 140),
        col = "blue1")
text(r$mids, r$density, r$counts, adj = c(.5, -.5), col = "blue3")
sapply(r[2:3], sum)
sum(r$density * diff(r$breaks)) # == 1
```

```
lines(r, lty = 3, border = "purple") # -> lines.histogram(*)
par(op)
require(utils) # for str
str(hist(islands, breaks = 12, plot =  FALSE)) #-> 10 (~= 12) breaks
str(hist(islands, breaks = c(12,20,36,80,200,1000,17000), plot = FALSE))
hist(islands, breaks = c(12,20,36,80,200,1000,17000), freq = TRUE,

main = "WRONG histogram") # and warning
```



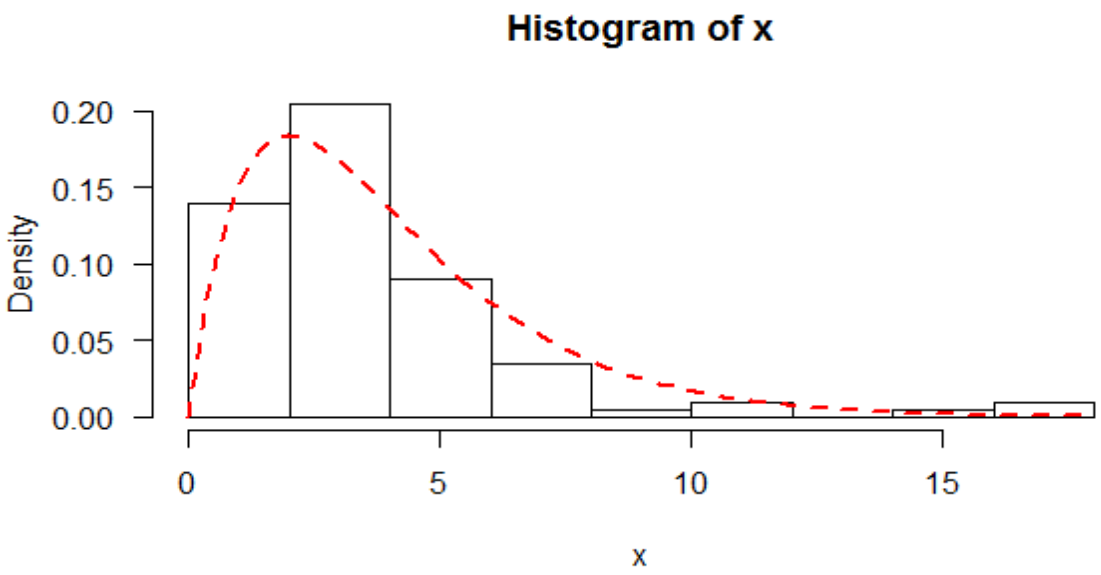

```
require(stats)
set.seed(14)
x <- rchisq(100, df = 4)
```

Comparing data with a model distribution should be done with qqplot()!

```
qqplot(x, qchisq(ppoints(x), df = 4)); abline(0, 1, col = 2, lty = 2)
```

if you really insist on using hist() ... :

```
hist(x, freq = FALSE, ylim = c(0, 0.2))
curve(dchisq(x, df = 4), col = 2, lty = 2, lwd = 2, add = TRUE)
```


Histogram of x
Density
0.20
0.15
0.10
0.05
0.00
0
5
10
15
x

Week-5

CORRELATION AND COVARIANCE

Objective:

a. Find the correlation matrix.
b. Plot the correlation plot on dataset and visualize giving an overview of relationships among data on iris data.
c. Analysis of covariance: variance (ANOVA), if data have categorical variables on iris data.

a. Find the correlation matrix

cor()

library(stats)

Correlation, Variance and Covariance (Matrices)

Description:

var, cov and cor compute the variance of x and the covariance or correlation of x and y if these are vectors. If x and y are matrices then the covariances (or correlations) between the columns of xand the columns of y are computed.

cov2cor scales a covariance matrix into the corresponding correlation matrix *efficiently*.

Usage:

```
var(x, y = NULL, na.rm = FALSE, use)

cov(x, y = NULL, use = "everything",
    method = c("pearson", "kendall", "spearman"))

cor(x, y = NULL, use = "everything",
    method = c("pearson", "kendall", "spearman"))

cov2cor(V)
```

Arguments:

x	a numeric vector, matrix or data frame.
y	NULL (default) or a vector, matrix or data frame with compatible dimensions to x. The default is equivalent to y = x (but more efficient).
na.rm	logical. Should missing values be removed?
use	an optional character string giving a method for computing covariances in the presence of missing values. This must be (an abbreviation of) one of the

	strings "everything","all.obs", "complete.obs", "na.or.complete", or"pairwise.complete.obs".
method	a character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman": can be abbreviated.
V	symmetric numeric matrix, usually positive definite such as a covariance matrix.

Examples:

var(1:10) # 9.166667

var(1:5, 1:5) # 2.5

Two simple vectors
cor(1:10, 2:11) # == 1

```
> var(1:10)  # 9.166667
[1] 9.166667
>
> var(1:5, 1:5) # 2.5
[1] 2.5
>
> ## Two simple vectors
> cor(1:10, 2:11) # == 1
[1] 1
```

Correlation Matrix of Multivariate sample:
Cl <- cor(longley)

Graphical Correlation Matrix:
symnum(Cl) # highly correlated

```
> ## Two simple vectors
> cor(1:10, 2:11) # == 1
[1] 1
> ## Correlation Matrix of Multivariate sample:
> (Cl <- cor(longley))
             GNP.deflator       GNP Unemployed Armed.Forces Population      Year
GNP.deflator    1.0000000 0.9915892  0.6206334    0.4647442  0.9791634 0.9911492
GNP             0.9915892 1.0000000  0.6042609    0.4464368  0.9910901 0.9952735
Unemployed      0.6206334 0.6042609  1.0000000   -0.1774206  0.6865515 0.6682566
Armed.Forces    0.4647442 0.4464368 -0.1774206    1.0000000  0.3644163 0.4172451
Population      0.9791634 0.9910901  0.6865515    0.3644163  1.0000000 0.9939528
Year            0.9911492 0.9952735  0.6682566    0.4172451  0.9939528 1.0000000
Employed        0.9708985 0.9835516  0.5024981    0.4573074  0.9603906 0.9713295
             Employed
GNP.deflator 0.9708985
GNP          0.9835516
Unemployed   0.5024981
Armed.Forces 0.4573074
Population   0.9603906
Year         0.9713295
Employed     1.0000000
> ## Graphical Correlation Matrix:
> symnum(Cl) # highly correlated
             GNP. GNP U A P Y E
GNP.deflator 1
GNP          B    1
Unemployed   ,    ,   1
Armed.Forces .    .     1
Population   B    B   , . 1
Year         B    B   , . B 1
Employed     B    B   . . B B 1
attr(,"legend")
[1] 0 ‘ ’ 0.3 ‘.’ 0.6 ‘,’ 0.8 ‘+’ 0.9 ‘*’ 0.95 ‘B’ 1
```

Spearman's rho and Kendall's tau

symnum(clS <- cor(longley, method = "spearman"))
symnum(clK <- cor(longley, method = "kendall"))

How much do they differ?

i <- lower.tri(Cl)
cor(cbind(P = Cl[i], S = clS[i], K = clK[i]))

```
> ## Spearman's rho  and  Kendall's tau
> symnum(clS <- cor(longley, method = "spearman"))
             GNP. GNP U A P Y E
GNP.deflator 1
GNP          B    1
Unemployed   ,    ,   1
Armed.Forces          . 1
Population   B    B   ,   1
Year         B    B   ,   1 1
Employed     B    B   .   B B 1
attr(,"legend")
[1] 0 ' ' 0.3 '.' 0.6 ',' 0.8 '+' 0.9 '*' 0.95 'B' 1
> symnum(clK <- cor(longley, method = "kendall"))
             GNP. GNP U A P Y E
GNP.deflator 1
GNP          B    1
Unemployed   .    .   1
Armed.Forces           1
Population   B    B   .   1
Year         B    B   .   1 1
Employed     *    *   .   + + 1
attr(,"legend")
[1] 0 ' ' 0.3 '.' 0.6 ',' 0.8 '+' 0.9 '*' 0.95 'B' 1
> ## How much do they differ?
> i <- lower.tri(cl)
> cor(cbind(P = cl[i], S = clS[i], K = clK[i]))
          P         S         K
P 1.0000000 0.9802390 0.9572562
S 0.9802390 1.0000000 0.9742171
K 0.9572562 0.9742171 1.0000000
```

```
## cov2cor() scales a covariance matrix by its diagonal
##          to become the correlation matrix.
cov2cor # see the function definition {and learn ..}
stopifnot(all.equal(Cl, cov2cor(cov(longley))),
          all.equal(cor(longley, method = "kendall"),
            cov2cor(cov(longley, method = "kendall"))))
```

##--- Missing value treatment:

```
C1 <- cov(swiss)
range(eigen(C1, only.values = TRUE)$values) # 6.19       1921

## swM := "swiss" with  3 "missing"s :
swM <- swiss
colnames(swM) <- abbreviate(colnames(swiss), min=6)
swM[1,2] <- swM[7,3] <- swM[25,5] <- NA # create 3 "missing"
```

Consider all 5 "use" cases :

```
(C. <- cov(swM)) # use="everything"  quite a few NA's in cov.matrix
try(cov(swM, use = "all")) # Error: missing obs...
C2 <- cov(swM, use = "complete")
stopifnot(identical(C2, cov(swM, use = "na.or.complete")))
range(eigen(C2, only.values = TRUE)$values) # 6.46   1930
C3 <- cov(swM, use = "pairwise")
range(eigen(C3, only.values = TRUE)$values) # 6.19   1938
```

```
## Kendall's tau doesn't change much:
symnum(Rc <- cor(swM, method = "kendall", use = "complete"))
symnum(Rp <- cor(swM, method = "kendall", use = "pairwise"))
symnum(R. <- cor(swiss, method = "kendall"))

## "pairwise" is closer componentwise,
summary(abs(c(1 - Rp/R.)))
summary(abs(c(1 - Rc/R.)))

## but "complete" is closer in Eigen space:
EV <- function(m) eigen(m, only.values=TRUE)$values
summary(abs(1 - EV(Rp)/EV(R.)) / abs(1 - EV(Rc)/EV(R.)))
```

a) Plot the correlation plot on dataset and visualize giving an overview of relationships among data on iris data.

corrplot()

library(corrplot)

A visualization of a correlation matrix.

Description:

A graphical display of a correlation matrix, confidence interval. The details are paid great attention to. It can also visualize a general matrix by setting is.corr = FALSE.

Usage:

```
corrplot ( corr,
        method = c("circle", "square", "ellipse", "number", "shade",   "color", "pie"),
        type = c("full", "lower", "upper"),
        add = FALSE,
        col = NULL,
        bg = "white",
        title = "",
        is.corr = TRUE,
        diag = TRUE,
        outline = FALSE,
        mar = c(0, 0, 0, 0),
        addgrid.col = NULL,
        addCoef.col = NULL,
        addCoefasPercent = FALSE,
        order = c("original",   "AOE", "FPC", "hclust",  "alphabet"),
        hclust.method  =  c("complete",  "ward",     "ward.D",  "ward.D2",  "single",  "average",
        "mcquitty", "median", "centroid"),
```

addrect = NULL,
rect.col = "black",
rect.lwd = 2,
tl.pos = NULL,
tl.cex = 1,
tl.col = "red",
tl.offset = 0.4,
tl.srt = 90,
cl.pos = NULL,
cl.lim = NULL,
cl.length = NULL,
cl.cex = 0.8,
cl.ratio = 0.15,
cl.align.text = "c",
cl.offset = 0.5,
number.cex = 1,
number.font = 2,
number.digits = NULL,
addshade = c("negative", "positive", "all"),
shade.lwd = 1,
shade.col = "white",
p.mat = NULL,
sig.level = 0.05,
insig = c("pch", "p-value", "blank", "n"),
pch = 4,
pch.col = "black",
pch.cex = 3,
plotCI = c("n", "square", "circle", "rect"),
lowCI.mat = NULL,
uppCI.mat = NULL,
na.label = "?",
na.label.col = "black", ...)

Arguments:

corr	The correlation matrix to visualize, must be square if order is not"original". For general matrix, please using is.corr = FALSE to convert.
method	Character, the visualization method of correlation matrix to be used. Currently, it supports seven methods, named "circle" (default),"square", "ellipse", "number", "pie", "shade" and "color ". See examples for details. The areas of circles or squares show the absolute value of corresponding correlation coefficients. Method "pie" and "shade" came from Michael Friendly's job (with some adjustment about the shade added on), and"ellipse" came from D.J. Murdoch and E.D. Chow's job, see in section

	References.
type	Character, "full" (default), "upper" or "lower", display full matrix, lower triangular or upper triangular matrix.
add	Logical, if TRUE, the graph is added to an existing plot, otherwise a new plot is created.
col	Vector, the color of glyphs. It is distributed uniformly in cl.lim. If NULL,col will be colorRampPalette(col2)(200), see example about col2.
bg	The background color.
title	Character, title of the graph.
is.corr	Logical, whether the input matrix is a correlation matrix or not. We can visualize the non-correlation matrix by setting is.corr = FALSE.
diag	Logical, whether display the correlation coefficients on the principal diagonal.
outline	Logical or character, whether plot outline of circles, square and ellipse, or the color of these glyphs. If outline is TRUE, the default value is"black".
mar	See par.
addgrid.col	The color of the grid. If NA, don't add grid. If NULL the default value is chosen. The default value depends on method, if method is color orshade, the color of the grid is NA, that is, not draw grid; otherwise"grey".
addCoef.col	Color of coefficients added on the graph. If NULL (default), add no coefficients.
addCoefasPercent	Logic, whether translate coefficients into percentage style for spacesaving.
order	Character, the ordering method of the correlation matrix. • "original" for original order (default). • "AOE" for the angular order of the eigenvectors. • "FPC" for the first principal component order. • "hclust" for the hierarchical clustering order. • "alphabet" for alphabetical order. See function corrMatOrder for details.
hclust.method	Character, the agglomeration method to be used when order is hclust. This should be one of "ward", "ward.D", "ward.D2", "single","complete", "average", "mcquitty", " median" or "centroid".
addrect	Integer, the number of rectangles draws on the graph according to the hierarchical cluster, only valid when order is hclust. If NULL (default), then add no rectangles.
rect.col	Color for rectangle border(s), only valid when addrect is equal or greater than 1.

rect.lwd	Numeric, line width for borders for rectangle border(s), only valid whenaddrect is equal or greater than 1.
tl.pos	Character or logical, position of text labels. If character, it must be one of"lt", "ld", "td", "d" or "n". "lt"(default if type=="full") means left and top, "ld"(default if type=="lower") means left and diagonal,"td"(default if type=="upper") means top and diagonal(near), "d"means diagonal, "n" means don't add textlabel.
tl.cex	Numeric, for the size of text label (variable names).
tl.col	The color of text label.
tl.offset	Numeric, for text label, see text.
tl.srt	Numeric, for text label string rotation in degrees, see text.
cl.pos	Character or logical, position of color labels; If character, it must be one of"r" (default if type=="upper" or "full"), "b" (default iftype=="lower") or "n", "n" means don't draw colorlabel.
cl.lim	The limits (x1, x2) in the colorlabel.
cl.length	Integer, the number of number-text in colorlabel, passed tocolorlegend. If NULL, cl.length is length(col) + 1 whenlength(col) <=20; cl.length is 11 when length(col) > 20
cl.cex	Numeric, cex of number-label in colorlabel, passed to colorlegend.
cl.ratio	Numeric, to justify the width of colorlabel, 0.1~0.2 is suggested.
cl.align.text	Character, "l", "c" (default) or "r", for number-label in colorlabel, "l"means left, "c" means center, and "r" means right.
cl.offset	Numeric, for number-label in colorlabel, see text.
number.cex	The cex parameter to send to the call to text when writing the correlation coefficients into the plot.
number.font	the font parameter to send to the call to text when writing the correlation coefficients into the plot.
number.digits	indicating the number of decimal digits to be added into the plot. Non-negative integer or NULL, default NULL.
addshade	Character for shade style, "negative", "positive" or "all", only valid when method is "shade". If "all", all correlation coefficients' glyph will be shaded; if "positive", only the positive will be shaded; if"negative", only the negative will be shaded. Note: the angle of shade line is different, 45 degrees for positive and 135 degrees for negative.
shade.lwd	Numeric, the line width of shade.
shade.col	The color of shade line.

p.mat	Matrix of p-value, if NULL, arguments sig.level, insig, pch,pch.col, pch.cex is invalid.
sig.level	Significant level, if the p-value in p-mat is bigger than sig.level, then the corresponding correlation coefficient is regarded as insignificant.
insig	Character, specialized insignificant correlation coefficients, "pch"(default), "p-value", "blank" or "n". If "blank", wipe away the corresponding glyphs; if "p-value", add p-values the corresponding glyphs; if "pch", add characters (see pch for details) on corresponding glyphs; if "n", don't take any measures.
pch	Add character on the glyphs of insignificant correlation coefficients(only valid when insig is "pch"). See par.
pch.col	The color of pch (only valid when insig is "pch").
pch.cex	The cex of pch (only valid when insig is "pch").
plotCI	Character, method of ploting confidence interval. If "n", don't plot confidence interval. If "rect", plot rectangles whose upper side means upper bound and lower side means lower bound, respectively, and meanwhile correlation coefficients are also added on the rectangles. If "circle", first plot a circle with the bigger absolute bound, and then plot the smaller. Warning: if the two bounds are the same sign, the smaller circle will be wiped away, thus forming a ring. Method "square" is similar to "circle".
lowCI.mat	Matrix of the lower bound of confidence interval.
uppCI.mat	Matrix of the upper bound of confidence interval.
na.label	Label to be used for rendering NA cells. Default is "?". If "square", then the cell is rendered as a square with the na.label.col color.
na.label.col	Color used for rendering NA cells. Default is "black".
...	Additional arguments passing to function text for drawing text lable.

##corrplot Examples

```
data(mtcars)
M <- cor(mtcars)

## different color series
col1 <- colorRampPalette(c("#7F0000","red","#FF7F00","yellow","white",
                           "cyan", "#007FFF", "blue","#00007F"))
col2 <- colorRampPalette(c("#67001F", "#B2182B", "#D6604D", "#F4A582", "#FDDBC7",
                           "#FFFFFF", "#D1E5F0", "#92C5DE", "#4393C3", "#2166AC", "#053061"))
col3 <- colorRampPalette(c("red", "white", "blue"))
col4 <- colorRampPalette(c("#7F0000","red","#FF7F00","yellow","#7FFF7F",
                           "cyan", "#007FFF", "blue","#00007F"))
wb <- c("white","black")
```

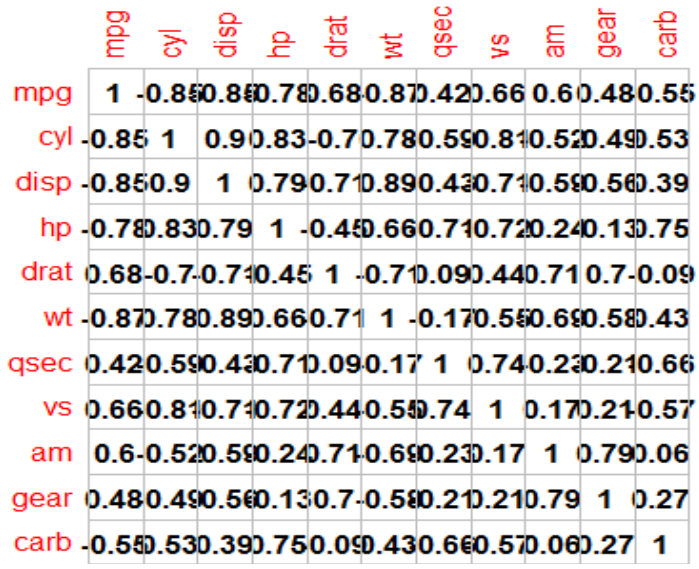
par(ask = TRUE)

different color scale and methods to display corr-matrix

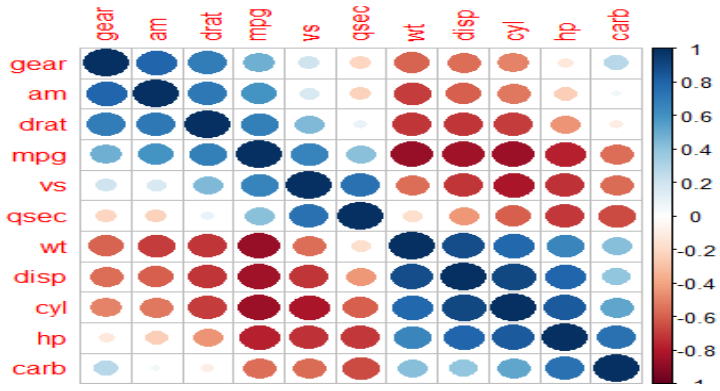
corrplot(M, method = "number", col = "black", cl.pos = "n")

corrplot(M, method = "number")

corrplot(M)

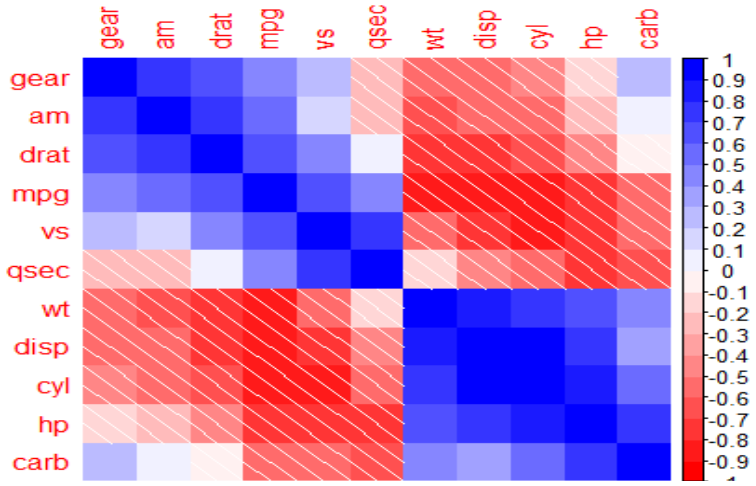


corrplot(M, order = "AOE")
corrplot(M, order = "AOE", addCoef.col = "grey")



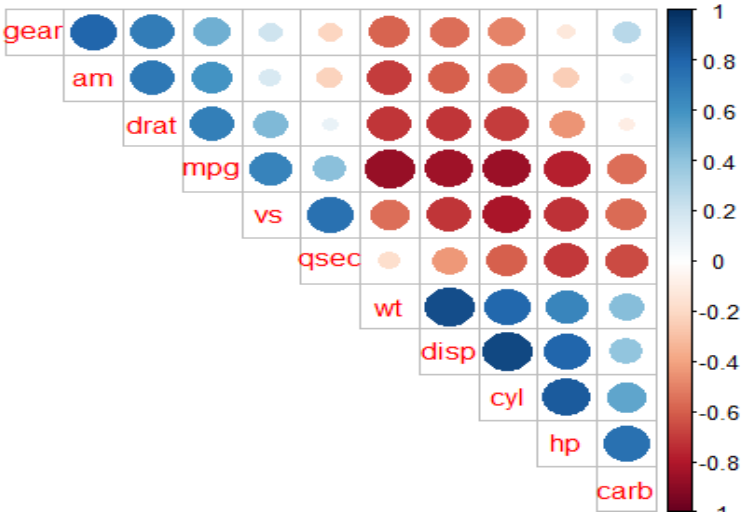
corrplot(M, method="color", col=col1(20), cl.length=21,order = "AOE", addCoef.col="grey")
corrplot(M, method="square", col=col2(200),order = "AOE")

corrplot(M, method="ellipse", col=col1(200),order = "AOE")
corrplot(M, method="shade", col=col3(20),order = "AOE")
corrplot(M, method="pie", order = "AOE")

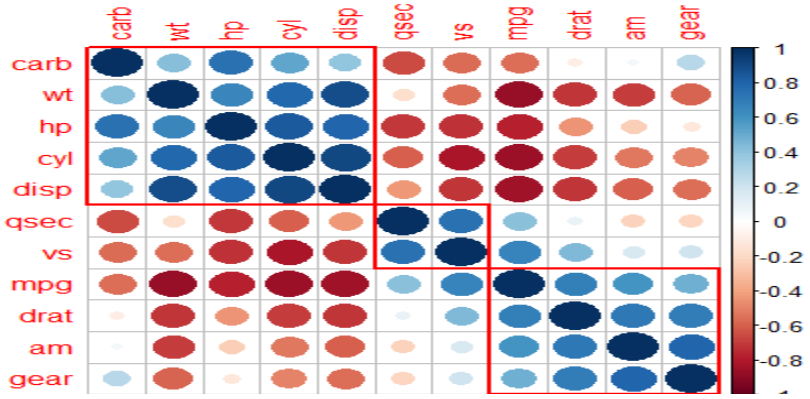


mixed methods: It's more efficient if using function "corrplot.mixed"

circle + ellipse
corrplot(M,order="AOE",type="upper",tl.pos="d")
corrplot(M,add=TRUE,type="lower",method="ell",order="AOE",diag=FALSE,tl.pos="n", cl.pos="n")



order is hclust and draw rectangles
corrplot(M, order="hclust")
corrplot(M, order="hclust", addrect = 2)
corrplot(M, order="hclust", addrect = 3, rect.col = "red")
corrplot(M, order="hclust", addrect = 4, rect.col = "blue")
corrplot(M, order="hclust", hclust.method="ward", addrect = 4)



b) Analysis of covariance: variance (ANOVA), if data have categorical variables on iris data.

anova():

library(stats)

Description:

Compute analysis of variance (or deviance) tables for one or more fitted model objects.

Usage:

anova(object, ...)

Arguments:

object	an object containing the results returned by a model fitting function (e.g., lm orglm).
...	additional objects of the same type.

```
> fit=lm(iris$Sepal.Length ~ iris$Petal.Length)
> anova(fit)
```

Analysis of Variance Table

```
Response: iris$Sepal.Length
                   Df Sum Sq Mean Sq F value    Pr(>F)
iris$Petal.Length   1 77.643  77.643  468.55 < 2.2e-16 ***
Residuals         148 24.525   0.166
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Week-6

REGRESSION MODEL

Objective:

Import a data from web storage. Name the dataset and now do Logistic Regression to find out relation between variables that are affecting the admission of a student in a institute based on his or her GRE score, GPA obtained and rank of the student. Also check the model is fit or not. require (foreign), require(MASS).

Regression analysis mainly focuses on :

- Finding a relationship between a dependent variable and one or more independent variables.
- Predict the value of a dependent variable
- Finding the impact of changes in an independent variable on the dependent variable.

Y = f(X, β)

where Y is the dependent variable ,X is the independent variable ,β is the unknown coefficient

- Simple Linear Regression
-

Y= α + βX + ε

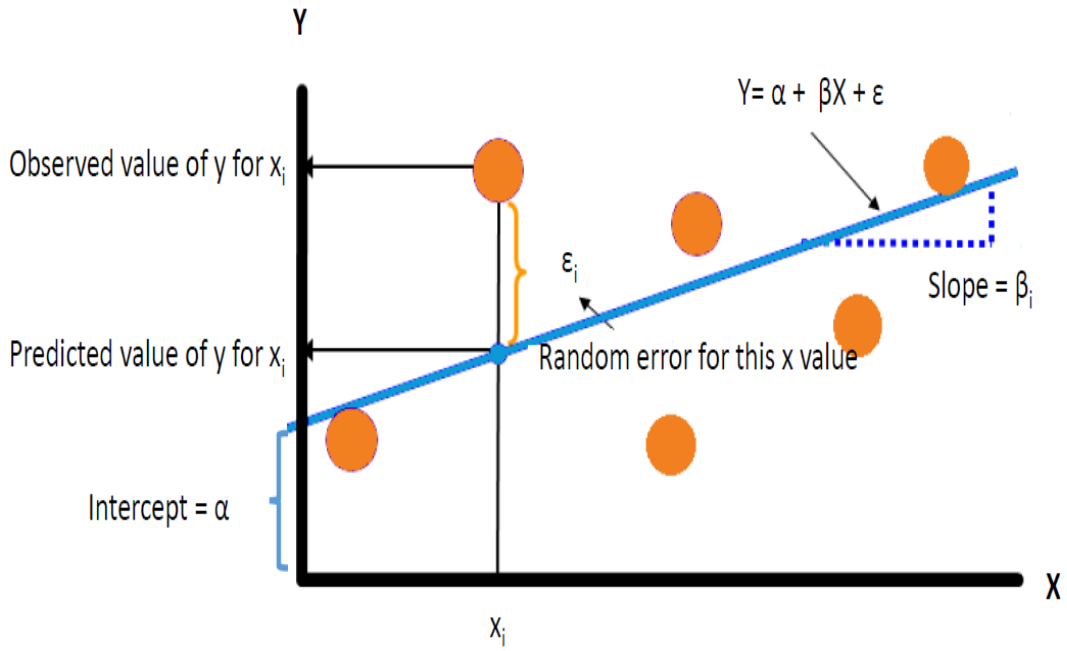
α = intercept coefficients

β = slope coefficients

ε = residuals

- Multiple Linear Regression

Y= α + β1X1+ β2X2+ β3X3 + …..+βkXk + ε



- A researcher is interested in how variables, such as GRE (Graduate Record Exam scores), GPA (grade point average) and prestige of the undergraduate institution, affect admission into graduate school.
- This data set has a binary response (outcome, dependent) variable called admit, which is equal to 1 if the individual was admitted to graduate school, and 0 otherwise.
- There are three predictor variables: gre, gpa, and rank.
- Treat the variables gre and gpa as continuous. The variable rank takes on the values 1 through 4.
- Institutions with a rank of 1 have the highest prestige, while those with a rank of 4 have the lowest.
- The outcome variable, admit/don't admit, is binary.

```
>
> mydata <- read.csv("http://www.ats.ucla.edu/stat/data/binary.csv")
> head(mydata)
  admit gre  gpa rank
1     0 380 3.61    3
2     1 660 3.67    3
3     1 800 4.00    1
4     1 640 3.19    4
5     0 520 2.93    4
6     1 760 3.00    2
```

Week-7
MULTIPLE REGRESSION MODEL

Objective:
Apply multiple regressions, if data have a continuous independent variable. Apply on above dataset.

```
> mydata$rank <- factor(mydata$rank)
> mylogit <- glm(admit ~ gre + gpa + rank, data = mydata, family = "binomial")
> summary(mylogit)

Call:
glm(formula = admit ~ gre + gpa + rank, family = "binomial",
    data = mydata)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.6268  -0.8662  -0.6388   1.1490   2.0790

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.989979   1.139951  -3.500 0.000465 ***
gre          0.002264   0.001094   2.070 0.038465 *
gpa          0.804038   0.331819   2.423 0.015388 *
rank2       -0.675443   0.316490  -2.134 0.032829 *
rank3       -1.340204   0.345306  -3.881 0.000104 ***
rank4       -1.551464   0.417832  -3.713 0.000205 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 499.98  on 399  degrees of freedom
Residual deviance: 458.52  on 394  degrees of freedom
AIC: 470.52

Number of Fisher Scoring iterations: 4
```

In the output above:

- What the model we ran was, what options we specified, etc.
- Residuals, which are a measure of model fit.
- This part of output shows the distribution of the deviance residuals for individual cases used in the model.
- The next part of the output shows the coefficients, their standard errors, the z-statistic (sometimes called a Wald z-statistic), and the associated p-values.

** Both gre and gpa are statistically significant, as are the three terms for rank. The logistic regression coefficients give the change in the log odds of the outcome for a one unit increase in the predictor variable.

**For every one unit change in gre, the log odds of admission (versus non-admission) increases by 0.002.

**For a one unit increase in gpa, the log odds of being admitted to graduate school increases by 0.804.

**The indicator variables for rank have a slightly different interpretation. For example, having attended an undergraduate institution with rank of 2, versus an institution with a rank of 1, changes the log odds of admission by -0.675.

Week-8
REGRESSION MODEL FOR PREDICTION

Objective:
Apply regression Model techniques to predict the data on above dataset

Source code:

```
install.packages("DMwR")
library("DMwR")
train <- centralImputation(raw_data)
raw_data$Attendance <- as.numeric(raw_data$Attendance)
sum(raw_data$UG.CGPA)
mean_Attendance <- mean(train$Attendance)
#
# train <- subset(train, train$Attendance > 90)
# train2 <- subset(train, train$UG.CGPA > 3)

# descriptive visualizations

hist(train$Attendance)

train$Birth.Year <- as.integer(train$Birth.Year)
plot(train$PG.CGPA, train$Attendance)
cor(train$PG.CGPA, train$Attendance)

cor(train, method ="pearson")

max(train$Work.Exp.)

# linear regression

model <- lm(PG.CGPA ~ ., train)

summary(model )
plot(model)
hist(model$residuals)

test <- read.csv("C:/Users/sarang.venukala/Desktop/data/sample_data3.csv")
colnames(test)

predictions <- predict(model, newdata = test)
predictions <- as.data.frame(predict(model, newdata = test))

class(test)
class(predictions)

test2 <- cbind(test, predictions)

# library("car")
# sqrt(vif(model)) > 2
```

```
model3 <- lm(PG.CGPA ~ Attendance, train)
plot(model2)
```

summary(model3)

```
model2$residuals
predictions <- predict(model3, newdata = test)
predictions <- as.data.frame(predict(model3, newdata = test))

test2 <- cbind(test2, predictions)

##########################
model4 <- lm(PG.CGPA ~ UG.CGPA, train)
summary(model4)
model2$residuals
predictions <- predict(model4, newdata = test)
predictions <- as.data.frame(predict(model4, newdata = test))

test2 <- cbind(test2, predictions)

##########################
model5 <- step(model)

summary(model5)
model2$residuals
predictions <- predict(model5, newdata = test)
predictions <- as.data.frame(predict(model5, newdata = test))

test2 <- cbind(test2, predictions)

test2 <- cbind(test2, (as.data.frame(train$PG.CGPA)))

###################################

test_new_sem <- read.csv("C:/Users/sarang.venukala/Desktop/data/test_data.csv")

predictions_new_sem <- as.data.frame(predict(model5, newdata = test_new_sem))
test_new_sem_2 <- cbind(test_new_sem, predictions_new_sem)
students_target <- subset(test_new_sem_2, test_new_sem_2$`predict(model5,
newdata = test_new_sem)` < 2.6)

hist(train$PG.CGPA)
```

Week-9

CLASSIFICATION MODEL

Objective:

a. Install relevant package for classification.
b. Choose classifier for classification problem.
c. Evaluate the performance of classifier.

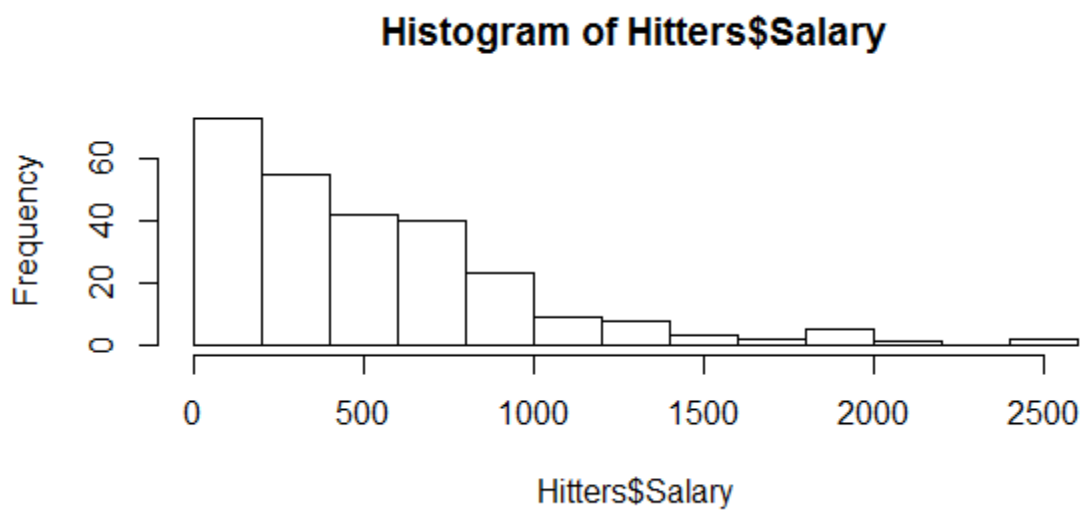
```
install.packages("rpart.plot")
install.packages("tree")
install.packages("ISLR")
install.packages("rattle")

library(tree)
library(ISLR)
library(rpart.plot)
library(rattle)

attach(Hitters)
View(Hitters)
# Remove NA data
Hitters<-na.omit(Hitters)

# log transform Salary to make it a bit more normally distributed
hist(Hitters$Salary)

Hitters$Salary <- log(Hitters$Salary)
hist(Hitters$Salary)
```

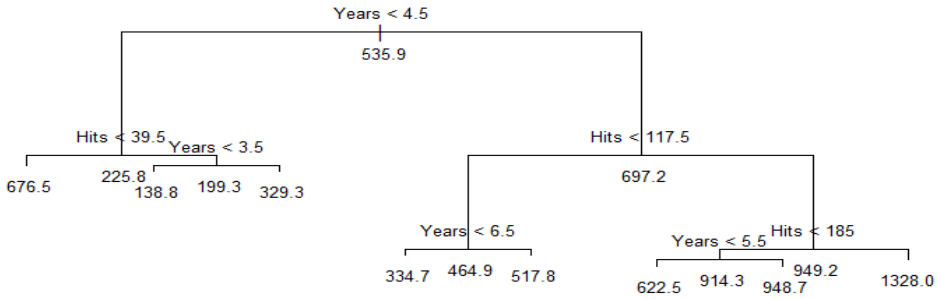



```
> tree.fit <- tree(Salary~Hits+Years, data=Hitters)
```

> summary(tree.fit)

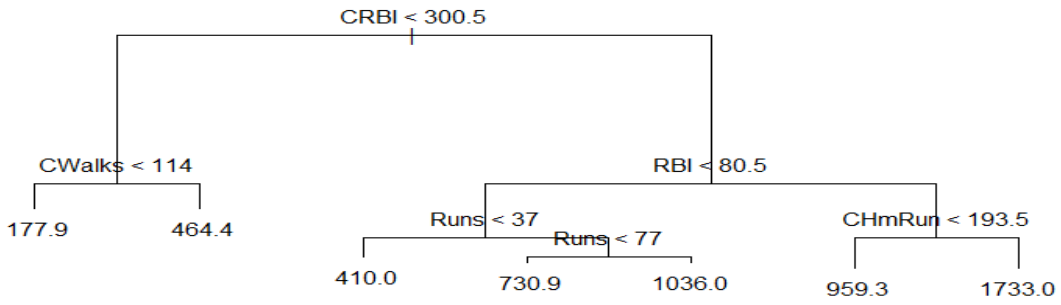
Regression tree:
tree(formula = Salary ~ Hits + Years, data = Hitters)
Number of terminal nodes: 8
Residual mean deviance: 101200 = 25820000 / 255
Distribution of residuals:
Min. 1st Qu. Median Mean 3rd Qu. Max.
-1238.00 -157.50 -38.84 0.00 76.83 1511.00

plot(tree.fit, uniform=TRUE,margin=0.2)
text(tree.fit, use.n=TRUE, all=TRUE, cex=.8)
#plot(tree.fit)

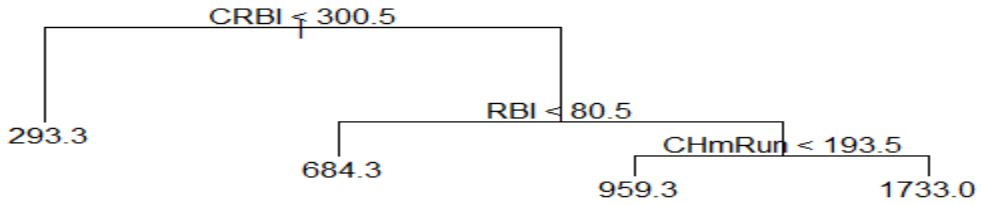


> split <- createDataPartition(y=Hitters$Salary, p=0.5, list=FALSE)
> train <- Hitters[split,]
> test <- Hitters[-split,]
#Create tree model
> trees <- tree(Salary~., train)
> plot(trees)
> text(trees, pretty=0)

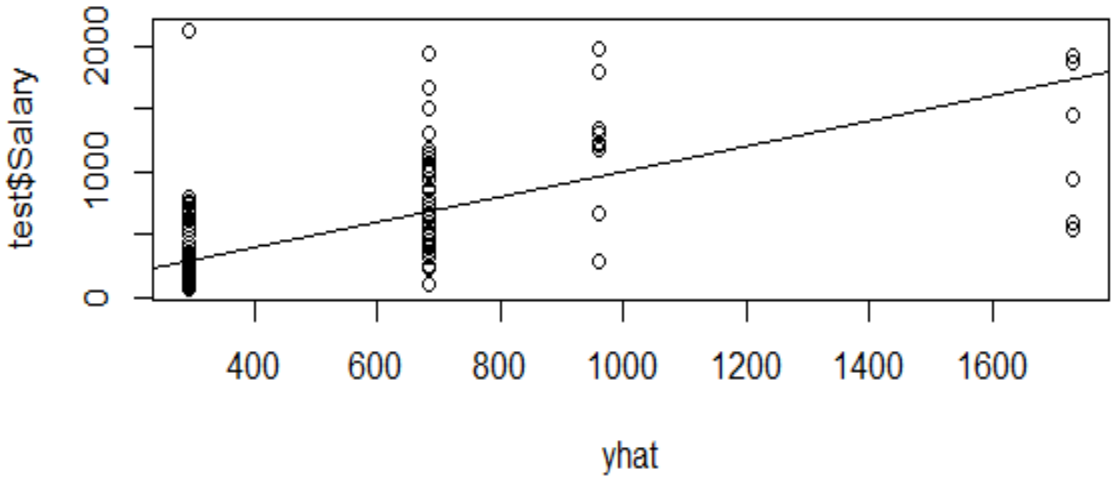
Cross validate to see whether pruning the tree will improve
performance



#Cross validate to see whether pruning the tree will improve performance
> cv.trees <- cv.tree(trees)
> plot(cv.trees)
> prune.trees <- prune.tree(trees, best=4)
> plot(prune.trees)
> text(prune.trees, pretty=0)



> yhat <- predict(prune.trees, test)
> plot(yhat, test$Salary)
> abline(0,1
[1] 150179.7
> mean((yhat - test$Salary)^2)
[1] 150179.7




```
> mean((yhat - test$Salary)^2)
[1] 150179.7
```

Week-10

CLUSTERING MODEL

Objective:

a. Clustering algorithms for unsupervised classification.
b. Plot the cluster data using R visualizations.

1. Clustering algorithms for unsupervised classification.

Partitioning Around Medoids(PAM)

library(cluster)

Description:

Partitioning (clustering) of the data into k clusters "around medoids", a more robust version of K-means.

Usage:

```
pam(x, k, diss = inherits(x, "dist"), metric = "euclidean",
    medoids = NULL, stand = FALSE, cluster.only = FALSE,
    do.swap = TRUE,
    keep.diss = !diss && !cluster.only && n < 100,
    keep.data = !diss && !cluster.only,
    pamonce = FALSE, trace.lev = 0)
```

Arguments:

x	data matrix or data frame, or dissimilarity matrix or object, depending on the value of the diss argument. In case of a matrix or data frame, each row corresponds to an observation, and each column corresponds to a variable. All variables must be numeric. Missing values (NAs) *are* allowed—as long as every pair of observations has at least one case not missing. In case of a dissimilarity matrix, x is typically the output of daisy or dist. Also a vector of length n*(n-1)/2 is allowed (where n is the number of observations), and will be interpreted in the same way as the output of the above-mentioned functions. Missing values (NAs) are *not* allowed.
k	positive integer specifying the number of clusters, less than the number of observations.
diss	logical flag: if TRUE (default for dist or dissimilarity objects), then xwill be considered as a dissimilarity matrix. If FALSE, then x will be

	considered as a matrix of observations by variables.
metric	character string specifying the metric to be used for calculating dissimilarities between observations. The currently available options are "euclidean" and "manhattan". Euclidean distances are root sum-of-squares of differences, and manhattan distances are the sum of absolute differences. If x is already a dissimilarity matrix, then this argument will be ignored.
medoids	NULL (default) or length-k vector of integer indices (in 1:n) specifying initial medoids instead of using the '*build*' algorithm.
stand	logical; if true, the measurements in x are standardized before calculating the dissimilarities. Measurements are standardized for each variable (column), by subtracting the variable's mean value and dividing by the variable's mean absolute deviation. If x is already a dissimilarity matrix, then this argument will be ignored.
cluster.only	logical; if true, only the clustering will be computed and returned, see details.
do.swap	logical indicating if the **swap** phase should happen. The default, TRUE, correspond to the original algorithm. On the other hand, the **swap** phase is much more computer intensive than the **build** one for large *n*, so can be skipped by do.swap = FALSE.
keep.diss, keep.data	logicals indicating if the dissimilarities and/or input data x should be kept in the result. Setting these to FALSE can give much smaller results and hence even save memory allocation *time*.
pamonce	logical or integer in 0:2 specifying algorithmic short cuts as proposed by Reynolds et al. (2006), see below.
trace.lev	integer specifying a trace level for printing diagnostics during the build and swap phase of the algorithm. Default 0 does not print anything; higher values print increasingly more.

Details:

The basic pam algorithm is fully described in chapter 2 of Kaufman and Rousseeuw(1990). Compared to the k-means approach in kmeans, the function pam has the following features: (a) it also accepts a dissimilarity matrix; (b) it is more robust because it minimizes a sum of dissimilarities instead of a sum of squared euclidean distances; (c) it provides a novel graphical display, the silhouette plot (see plot.partition) (d) it allows to select the number of clusters using mean(silhouette(pr)[, "sil_width"]) on the result pr <- pam(..), or directly its component pr$silinfo$avg.width, see also pam.object.

When cluster.only is true, the result is simply a (possibly named) integer vector specifying the clustering, i.e.,

pam(x,k, cluster.only=TRUE) is the same as
pam(x,k)$clustering but computed more efficiently.

The pam-algorithm is based on the search for k representative objects or medoids among the observations of the dataset. These observations should represent the structure of the data. After finding a set of k medoids, k clusters are constructed by assigning each observation to the nearest medoid. The goal is to find k representative objects which minimize the sum of the dissimilarities of the observations to their closest representative object.
By default, when medoids are not specified, the algorithm first looks for a good initial set of medoids (this is called the **build** phase). Then it finds a local minimum for the objective function, that is, a solution such that there is no single switch of an observation with a medoid that will decrease the objective (this is called the **swap** phase).

When the medoids are specified, their order does *not* matter; in general, the algorithms have been designed to not depend on the order of the observations.

The pamonce option, new in cluster 1.14.2 (Jan. 2012), has been proposed by Matthias Studer, University of Geneva, based on the findings by Reynolds et al. (2006).

The default FALSE (or integer 0) corresponds to the original "swap" algorithm, whereas pamonce = 1 (or TRUE), corresponds to the first proposal .... and pamonce = 2 additionally implements the second proposal as well.

Value

an object of class "pam" representing the clustering. See ?pam.object for details.

Note

For large datasets, pam may need too much memory or too much computation time since both are *O(n^2)*. Then, clara() is preferable, see its documentation.

There is hard limit currently, *n <= 65536*, at *2^{16}* because for larger *n*, *n(n-1)/2* is larger than the maximal integer (.Machine$integer.max = *2^{31} - 1*).

Author(s)

Kaufman and Rousseeuw's orginal Fortran code was translated to C and augmented in several ways, e.g. to allow cluster.only=TRUE or do.swap=FALSE, by Martin Maechler.
Matthias Studer, Univ.Geneva provided the pamonce implementation.

References

Reynolds, A., Richards, G., de la Iglesia, B. and Rayward-Smith, V. (1992) Clustering rules: A comparison of partitioning and hierarchical clustering algorithms; *Journal of Mathematical Modelling and Algorithms* **5**, 475–504 (http://dx.doi.org/10.1007/s10852-005-9022-1).

See Also

agnes for background and references; pam.object, clara, daisy, partition.object,plot.partition, dist.

Examples

```
> set.seed(20)
> irisCluster <- kmeans(iris[, 3:4], 3, nstart = 20)
# nstart = 20. This means that R will try 20 different random starting assignments and then select the one with the lowest within cluster variation.
> irisCluster
```

K-means clustering with 3 clusters of sizes 50, 52, 48

Cluster means:

```
  Petal.Length Petal.Width
1     1.462000    0.246000
2     4.269231    1.342308
3     5.595833    2.037500
```

Clustering vector:

```
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [42] 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2
 [83] 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3
[124] 3 3 3 2 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3
```

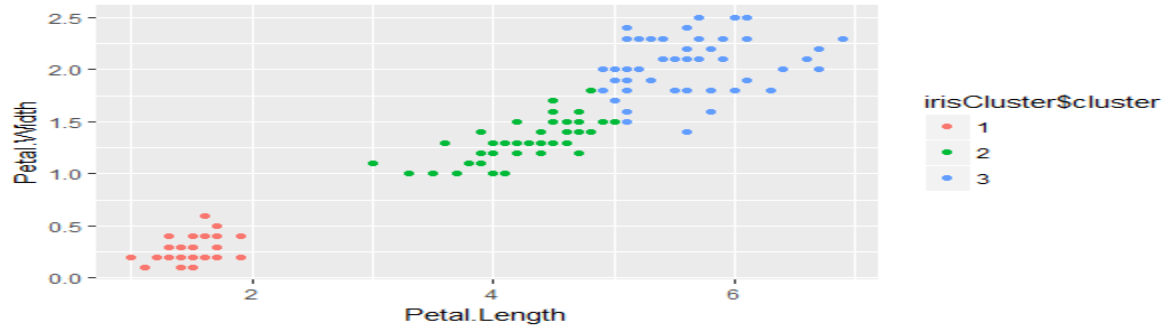
Within cluster sum of squares by cluster:

```
[1]  2.02200 13.05769 16.29167
 (between_SS / total_SS =  94.3 %)
```

Available components:

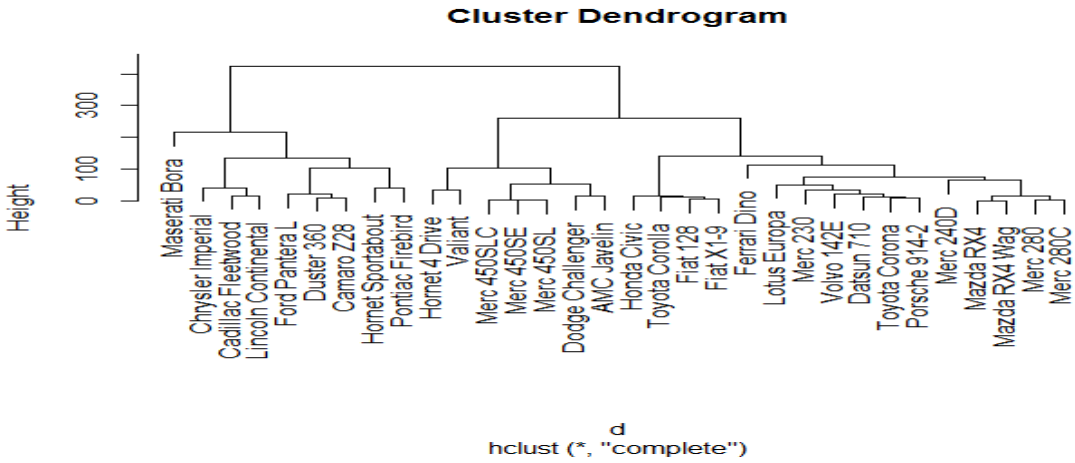
```
[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"

[6] "betweenss"    "size"         "iter"         "ifault"
> irisCluster$cluster <- as.factor(irisCluster$cluster)
> ggplot(iris, aes(Petal.Length, Petal.Width, color = irisCluster$cluster)) + geom_point()
```



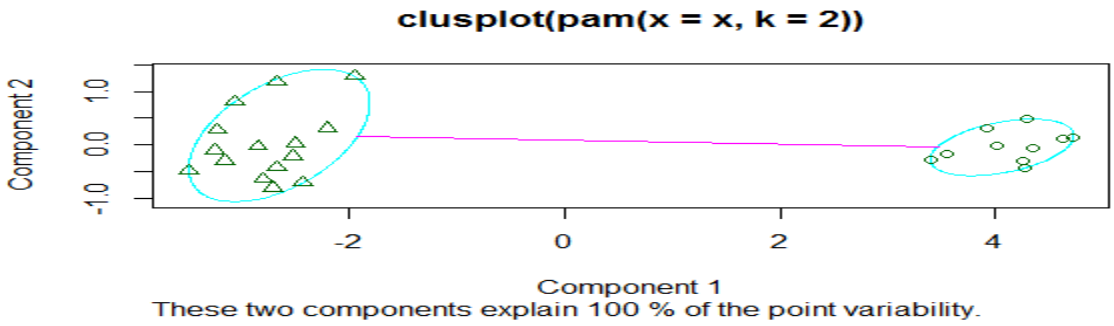
Hierarchical clustering

```
> d <- dist(as.matrix(mtcars))   # find distance matrix
> hc <- hclust(d)                # apply hirarchical clustering
> plot(hc)                       # plot the dendrogram
```



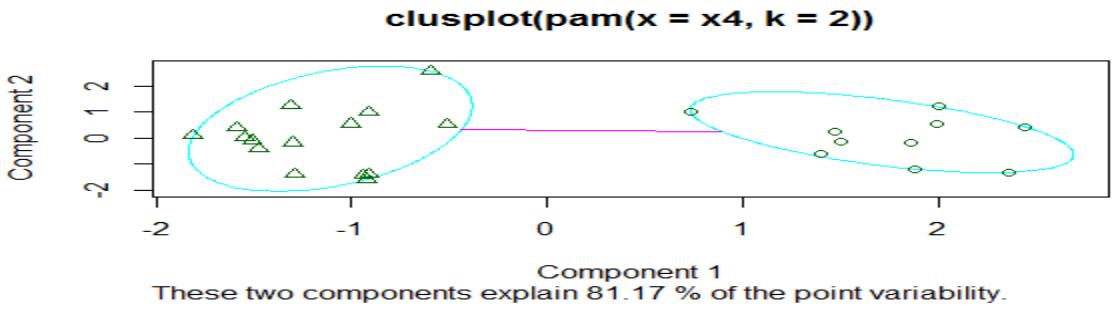
2. Plot the cluster data using R visualizations.

```
## generate 25 objects, divided into 2 clusters.
x <- rbind(cbind(rnorm(10,0,0.5), rnorm(10,0,0.5)),
           cbind(rnorm(15,5,0.5), rnorm(15,5,0.5)))
clusplot(pam(x, 2))
```



add noise, and try again :

```
x4 <- cbind(x, rnorm(25), rnorm(25))
clusplot(pam(x4, 2))
```



Viva-Voce Questions

Week 1:

1.Specify various features of R.
2.Explain various data types, Numeric, Character, date with suitable examples.
3.Explain how the following will be defined in R script:
a)data frame b) array c) matrix
4.Write R script to Read Datasets: Choose any data set as you like.
5.Explain how can you work with .txt and .csv files.
6.Write R script to handle .txt and .csv file through proper examples.
7.What do you understand about Combining Data sets?. Explain it.
8.Write R script to combine two data sets.
9.Explain R functions and R loops with examples.
10.Write Rscript to find out the factorial of any given number.
11.What is the goal of the R?
12.What is Data set?
13.What does the word Unified in UMRL mean?
14.What is R Script?
15.What is a package in R?
16.What is the difference between the array and matrix?.Explain with examples.
17.Explain the procedure to write R script to combine two data sets.
18.What is RStudio?. Explain its features.
19.Explain Rfunctions with suitable examples?

Week-2:

1. Distinguish between SQL and NoSQL.
2. Expain how to execute R code from an Excel spreadsheet
3. Why NoSQL is preferred than SQL?. Explain.
4. What is NoSQL?
5. What is SQL?
6. How no SQL is faster than SQL?
7. What are the applications o NoSQL ?
8. What is data storage model?
9. What did you understand about schemas?
10. What do you mean by data manipu

Week 3:

1. What is a Regression Analysis?. Explain how will you practically do it.
2. What is OLS Regression? Explain.
3. What is Regression Modelling?Explain how will you do it.
4. What did you understand about Regression residuals?Explain.
5. What is Correlation? How can you find out correlated items in a dataset.Explain.
6. Explain Correlation and Covariance.
7. Explain Autocorrelation in detail.
8. What did you understand about Multiple Regression? Explain.
9. What is Residual Plots? Explain.
10. Explain a)Correlation b)Auto correlation
11. What is Correlation Coefficients?

Week: 4:

1. Explain basic regression analysis.
2. Explain OLS regression.

Week 5:

1. Explain Regression Modelling in detail.
2. What is Heteroscedasticity ?. Explain.
3. What Autocorrelation and Multicollinearity ?Explain.

Week 6:

1. What is Machine Learning algorithm?
2. What is Hypothesis Testing ?
3. Explain supervised learning and unsupervised learning.
4. Listout Machine learning tasks and explain.
5. Explain about support vector machine

Week 7:

1. Distinguish between machine learning and data mining.
2. Explain the KDD task in detail
3. Explain Train model using machine learning algorithms, Test model.
4. What are steps followed in Machine Learning Algorithm? Explain.

Week 8:

1. What are practical uses of Machine Learning?
2. Why is Machine Learning joined with Data Analytics?
3. What is Train Model and Test Model? Explain

Week 9:

1. What is Supervised learning ?
2. What is reinforcement learning ?

Week 10:

1. What is unsupervised learning ?
2. List different clustering algorithms