R Statistics Essential Training

Trainer: Ravi

Website:www.tertiarycourses.com.sg
Email: enquiry@tertiaryinfotech.com

Agenda

Module 1. Getting Started

- What is R
- Install R and RStudio
- Explore RStudio Interface
- Variables

Module 2. Data Types

- Numbers
- Text
- Vector
- Matrix
- Array
- Data Frame
- Factor
- List

Agenda

Module 3. Packages & Data Sets

- Packages
- Data Sets

Module 4. File Input/Output

- Read data from file
- Write data to file

Module 5. Charts

- Scatter Plot
- Line Plot
- Bar chart
- Boxplot
- Pie chart
- Histogram

Agenda

Module 6. Control Structures

- Conditional
- Loop
- Break & Next
- Operators

Module 7: Function

- Function syntax
- Function Example
- Function With Default Arguments
- Advanced Functions (Optional)

Agenda

Module 8. Statistical Application of R

- Basic Statistics
- Correlation
- Linear Regression
- Distributions
- Hypothesis Testing
- T - Test

Module 9. Intro to Advanced Statistics (Optional)

- ANOVA
- Chi-squared test
- Clustering

Module 1
Getting Started

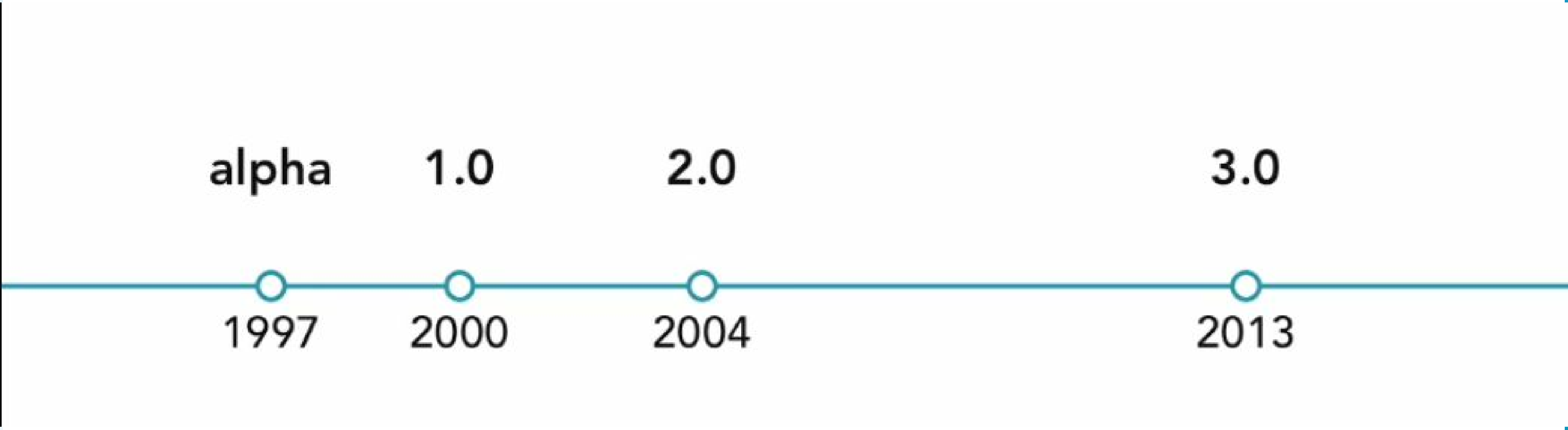
What is R?

- R was developed by Ross Ihaka and Robert Gentleman
- R is a language and environment for statistical computing and graphics
- R was developed based on S

Why R?

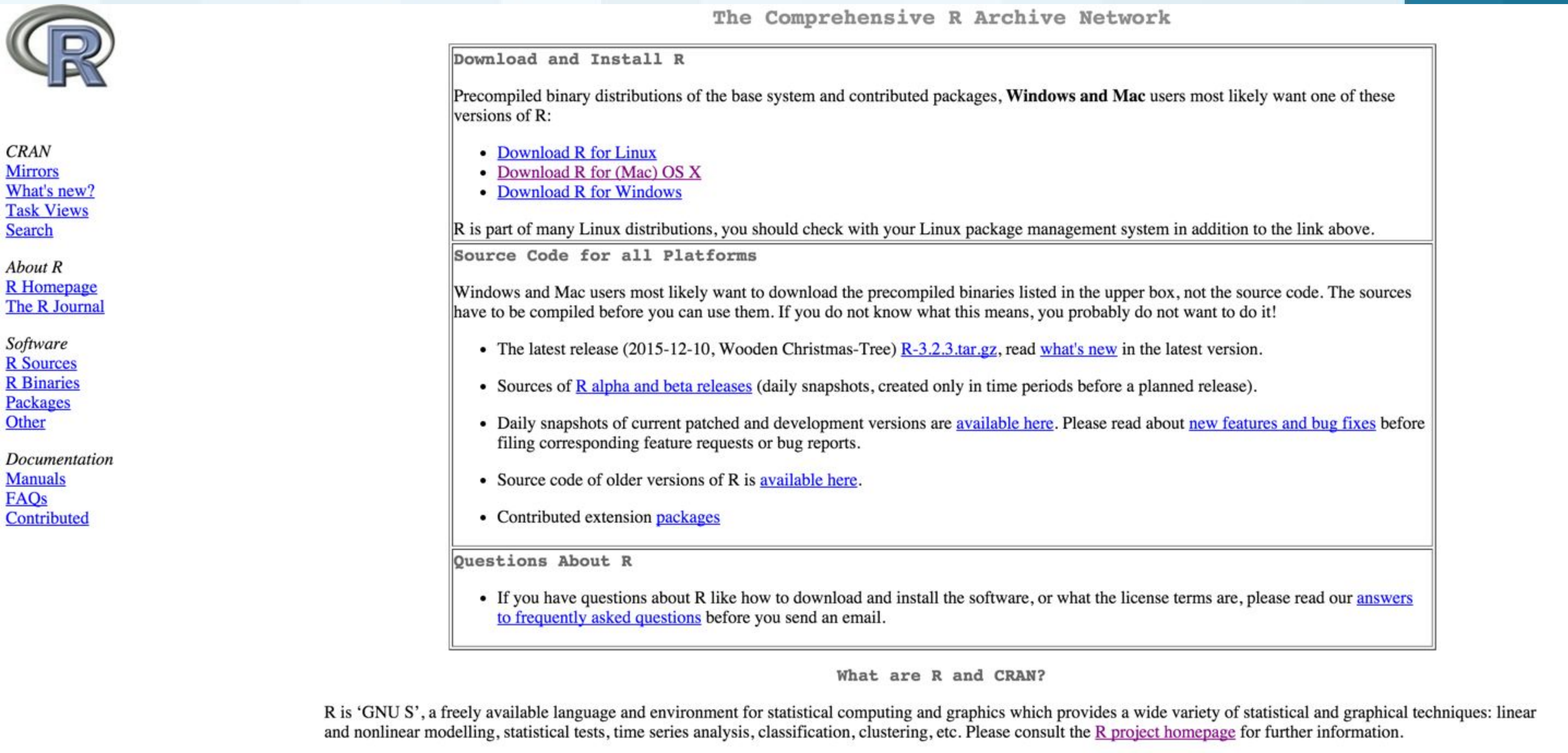
- R is open source and free!
- R is vector-based
- R is a programming language meant for statistics
- R can be integrated with other software such as SAS, SPSS, Excel..

R Development History



Install R

http://cran.stat.nus.edu.sg/



CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed

The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux
- Download R for (Mac) OS X
- Download R for Windows

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2015-12-10, Wooden Christmas-Tree) R-3.2.3.tar.gz, read what's new in the latest version.
- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
- Source code of older versions of R is available here.
- Contributed extension packages

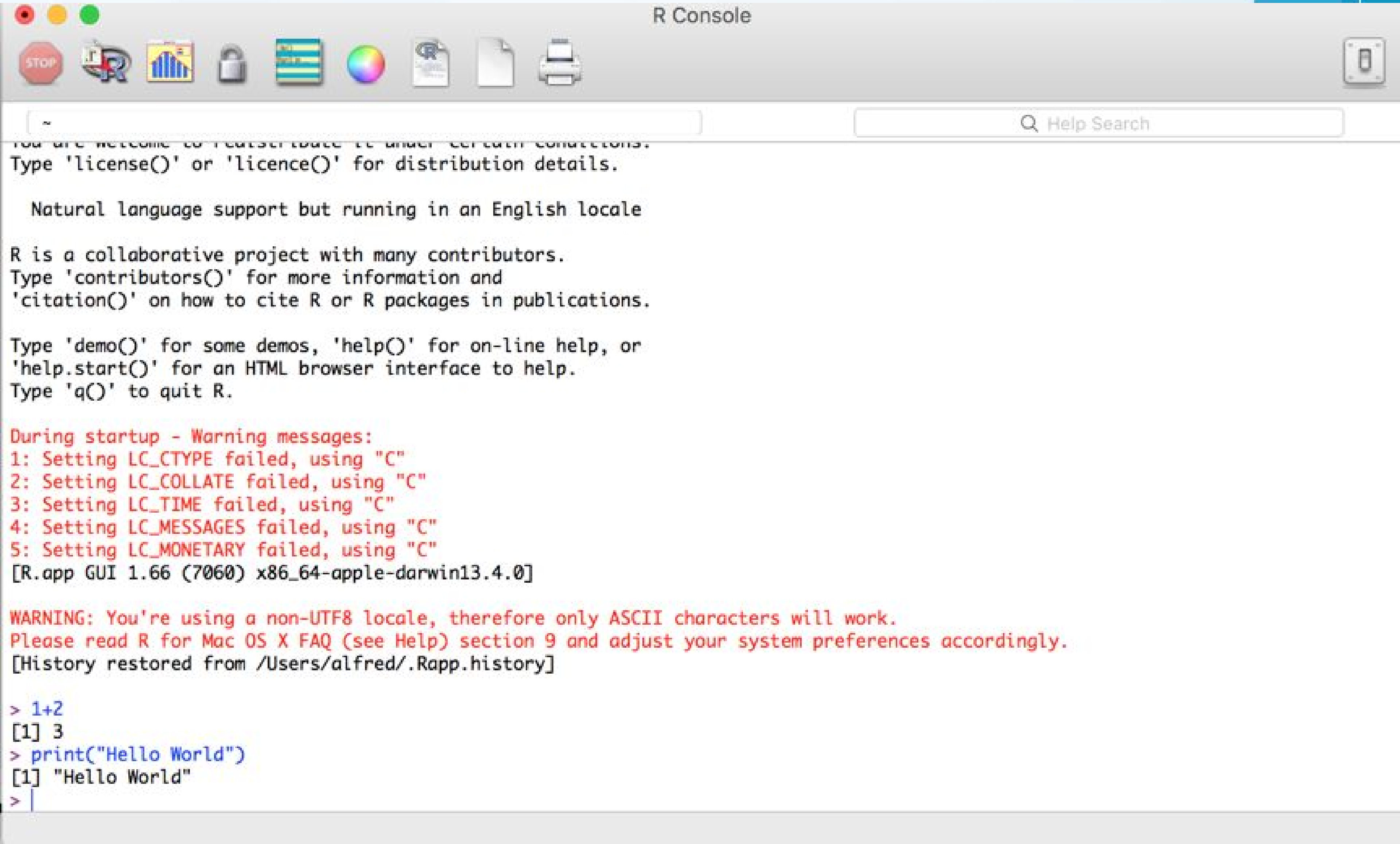
Questions About R

- If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

What are R and CRAN?

R is ‘GNU S’, a freely available language and environment for statistical computing and graphics which provides a wide variety of statistical and graphical techniques: linear and nonlinear modelling, statistical tests, time series analysis, classification, clustering, etc. Please consult the R project homepage for further information.

Run R Console



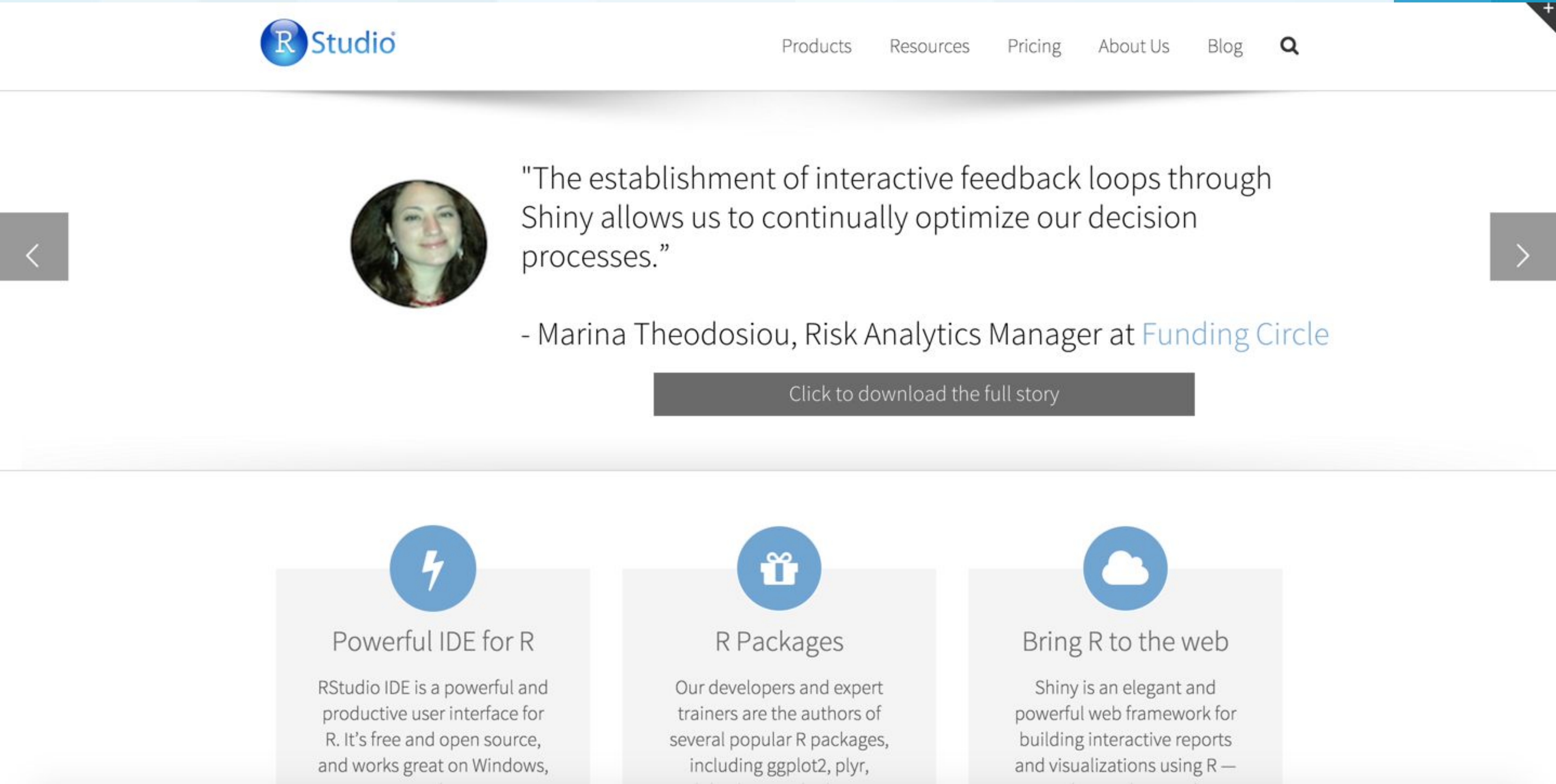
```
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

During startup - Warning messages:
1: Setting LC_CTYPE failed, using "C"
2: Setting LC_COLLATE failed, using "C"
3: Setting LC_TIME failed, using "C"
4: Setting LC_MESSAGES failed, using "C"
5: Setting LC_MONETARY failed, using "C"
[R.app GUI 1.66 (7060) x86_64-apple-darwin13.4.0]

WARNING: You're using a non-UTF8 locale, therefore only ASCII characters will work.
Please read R for Mac OS X FAQ (see Help) section 9 and adjust your system preferences accordingly.
[History restored from /Users/alfred/.Rapp.history]

> 1+2
[1] 3
> print("Hello World")
[1] "Hello World"
> 
```

Run R on Terminal

Type 'r' on the terminal/cmd to start r

Install R Studio IDE

https://www.rstudio.com/



"The establishment of interactive feedback loops through Shiny allows us to continually optimize our decision processes."

- Marina Theodosiou, Risk Analytics Manager at Funding Circle

Powerful IDE for R

RStudio IDE is a powerful and productive user interface for R. It's free and open source, and works great on Windows,

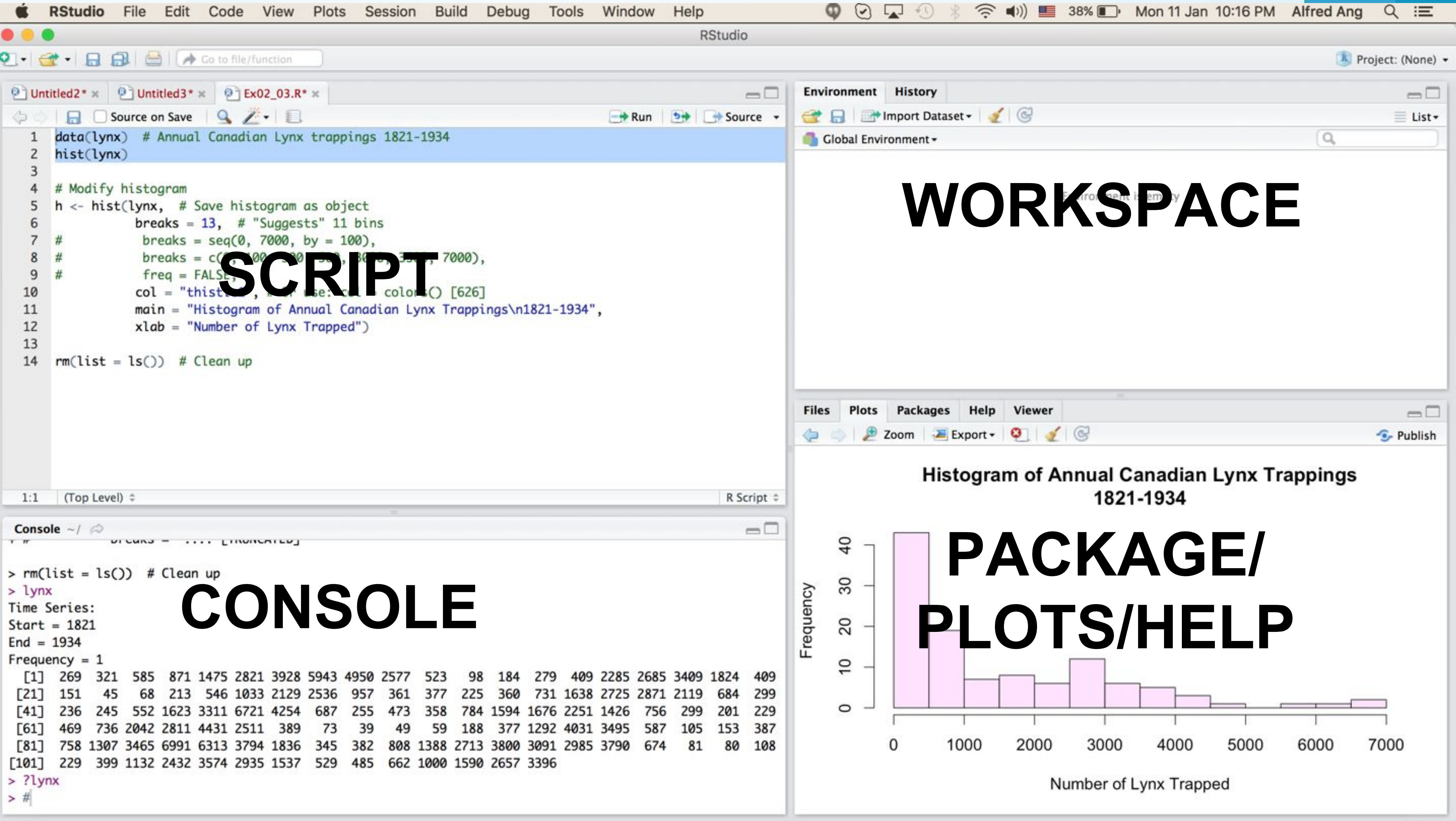
R Packages

Our developers and expert trainers are the authors of several popular R packages, including ggplot2, plyr,

Bring R to the web

Shiny is an elegant and powerful web framework for building interactive reports and visualizations using R —

Explore RStudio Interface



```
data(lynx)  # Annual Canadian Lynx trappings 1821-1934
hist(lynx)

# Modify histogram
h <- hist(lynx,  # Save histogram as object
          breaks = 13,  # "Suggests" 11 bins
#          breaks = seq(0, 7000, by = 100),
#          breaks = c(0, 100, 300, 500, 3000, 3500, 7000),
#          freq = FALSE,
          col = "thistle", # Or use: col = colors() [626]
          main = "Histogram of Annual Canadian Lynx Trappings\n1821-1934",
          xlab = "Number of Lynx Trapped")

rm(list = ls())  # Clean up
```

```
+ #          breaks = .... [TRUNCATED]

> rm(list = ls())  # Clean up
> lynx
Time Series:
Start = 1821
End = 1934
Frequency = 1
  [1]  269  321  585  871 1475 2821 3928 5943 4950 2577  523   98  184  279  409 2285 2685 3409 1824  409
 [21]  151   45   68  213  546 1033 2129 2536  957  361  377  225  360  731 1638 2725 2871 2119  684  299
 [41]  236  245  552 1623 3311 6721 4254  687  255  473  358  784 1594 1676 2251 1426  756  299  201  229
 [61]  469  736 2042 2811 4431 2511  389   73   39   49   59  188  377 1292 4031 3495  587  105  153  387
 [81]  758 1307 3465 6991 6313 3794 1836  345  382  808 1388 2713 3800 3091 2985 3790  674   81   80  108
[101]  229  399 1132 2432 3574 2935 1537  529  485  662 1000 1590 2657 3396
> ?lynx
> #
```

Comments and Help

Single line comment

#

Help

?....

help(...)

Variables

```
a <- 1        (Most common)
a = 1
1 -> a
```

List all variables

```
ls()
```

Clean Up the Variables

```
rm(x)
rm(a, b)
rm(list = ls())
```

Module 2
Data Types

Data Types

- Numbers
- Text
- Vector
- Matrix
- Array
- Data Frame
- Factor
- List

Numbers

Working With Numbers

```
2^4
abs(-3.2)
round(3.4), round(5.24, digits = 1), round(5.24, 1)
sqrt(4)
cos(pi/2), factorial(3)

3/0
Inf/Inf
```

NaN : Not a Number

Text

Splitting Text

```
a <- "Today is a good day"
```

Splitting & Joining Text

```
a <- "Today is a good day"
strsplit(a," ")
strsplit(a,"is ")

a<-"angch"
b<-"tertiaryinfotech.com"
paste(a,b,sep="@")
```

Sorting Text

```
v <- c("Red","Blue","yellow","violet")
sort(v)
sort(v, decreasing=TRUE)
```

Vector

Create Vector Using : Operator

```
a <- 0:10
a <- 5:13
a <- 10:-4
```

class(a) - check the type of a

str(a) - check the structure of a

Create Vector Using c Operator

```
c(1,2,4)
c(1,7:9)
c('red','green',"yellow")
c(1:5, 10.5, "red")
```

Create Vector Using seq

```
seq(10)
seq(3,20)
seq(3,20,3)
seq(1, 9, by = 2)
seq(from=4.5, to=2.5, by=-0.2)
seq(0, 1, length.out = 11)
```

Application of seq

```
x<-seq(0,4*pi,length.out = 200);
y <- sin(x);
plot(x,y)
```

Creating Vectors Using rep -1

```
rep(3,10)
rep(1:3, 3)
rep(c(1,2,3),3)
rep(seq(3),3)
rep(seq(3),length.out=5)
rep(seq(3),len=5)
rep(1:4, each = 2)
rep(1:4, c(2,1,2,1))
```

Creating Vectors Using rep -2

```
rep(1:4, each = 2, len = 4)
rep(1:4, each = 2, len = 10)
rep(1:4, each = 2, times = 3)
```

Accessing Vector Elements

```
a[5]
a[5:8]
a[5:3]
a[-2]  - take away the 2nd element
a[-1:-4]
a[ c(5, 6, 7, 8) ]
```

Tail and Head

```
tail(a)
tail(a,n=3)
tail(a,3)

head(a)
head(a,n=3)
head(a,3)
```

Logical Indexing

```
c(1,2,3) = c(3,2,1)

1 %in% c(3,4,5)
c(1,2) %in% c(1,2,3,4)
```

Challenge: Logical Indexing

```
a <-c(2,3,-1,3,5,2,-3,1)
```

find the sum of the positive elements of a

Hint: a>0

Dropping NA values

```
a<-c(3,-2,4,NA,-1,8,-4,9,NA, 11,3,8,NA);
a[!is.na(a)]
```

NA: Missing data

Manipulating Vectors

```
x <- c(1,2,1)
y <- c(3,2,4)
v <- x + y
v <- x - y
```

Vector Arithmetic

```
a <-c(2,3,4,2,5,6)
mean(a)
median(a)
sum(a), prod(a)
min(a). max(a)
cummin(a), cummax(a)
cumsum(a), cumprod(a)
diff(a)
```

Sorting Vector Elements

```
v <- c(3,8,4,5,0,11, -9, 304)
sort(v)
sort(v, decreasing=TRUE)
```

Sum Vector Elements

```
v <- c(3,6,2,NA,1)
sum(v)
sum(v, na.rm=TRUE)
sum(a[!is.na(a)])
```

Structure of Vectors

```
v <- c(3,8,4,5,0,11, -9, 304)
str(v)
v <- c("Red","Blue","yellow","violet")
str(v)
```

Matrix

What is Matrix

Matrices are the R objects in which the elements are arranged in a two-dimensional rectangular layout.

Syntax:

matrix(data, nrow, ncol, byrow = FALSE, dimnames)

Creating Matrix

Usage: matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)

```
matrix(1:12, ncol=4)
matrix(1:12, nrow=4)
matrix(c(3, 9, -1, 4, 2, 6), nrow=2)
matrix(1:12, ncol=4,byrow=TRUE)
matrix(1:12, nrow=4,byrow=TRUE)
```

Convert Vector to Matrix

```
a <- c(1,1,1,1)

dim(a)<-c(2,2)
```

Combining Vectors to Matrix

```
a1 <- c(1,1,1,1)
a2 <- c(2,2,2,2)
rbind(a1,a2)
cbind(a1,a2)
```

Accessing Matrix Elements

M[1,3]	- row 1, col 3
M[2,]	- row 2
M[,3]	- col 3
M[1:2,3:4]	- row 1 to 2, col 3 to 4

Challenge: Matrix Elements

a <- matrix(1:20,ncol=4). Extract the elements in red

```
     [,1] [,2] [,3] [,4]
[1,]    1    6   11   16
[2,]    2    7   12   17
[3,]    3    8   13   18
[4,]    4    9   14   19
[5,]    5   10   15   20
```

Time: 5 mins

Manipulating Matrix Elements

```
M1 <- matrix(c(3, 9, -1, 4, 2, 6), nrow=2)
M2 <- matrix(c(5, 2, 0, 9, 3, 4), nrow=2)
M3 <- M1 + M2
M4 <- M1 + M2
```

Row and Column Names

```
matrix(c(3, 9, -1, 4, 2, 6), nrow=2)
rownames(M)<-c("row1","row1")
colnames(M)<-c("col1","col2","col3")


M["row2",]
M[,"col3"]
```

Matrix Arithmetic

rowSums(M)	- sum of each row
colSums(M)	- sum of each col
colMeans(M)	- mean of each col
t(M1)	- transpose
solve(M1)	- invert
*	- element-wise multiple
%*%	- matrix multiplication

Array

What is Array?

Arrays are the R data objects which can store data in more than two dimensions

Creating Array

```
a <- array(c(11:14,21:24,31:34),dim=c(2,2,3))
```

Accessing Array Elements

a[,,1]	- 1st matrix
a[1,,1]	- 1st row, 1st matrix

Data Frame

What is Data Frame?

A data frame is a table or a two-dimensional array-like structure in which each column contains values of one variable and each row contains one set of values from each column.

- The column names should be non-empty.
- The row names should be unique.
- The data stored in a data frame can be of numeric, factor or character type.
- Each column should contain same number of data items.

Creating Data Frame

Usage: data.frame(..., row.names = NULL,..)

```
gender  <- c('Female','Female','Male')
height  <- c(162,169,170)
weight  <- c(40,50,60)
age     <- c(21,22,23)
name    <- c('Ally','Belinda','Alfred')
a <- data.frame(gender, height, weight, age,
row.names= name)
```

Creating Data Frame - Compact

```
a <- data.frame(
        gender = c("Male", "Male","Female"),
        height = c(152, 171.5, 165),
        weight = c(81,93, 78),
        age =c(42,38,26),
        row.names=c('Ally','Belinda','Alfred')
        )
```

Note: there is only 1 variable generated

Create Data Frame from Matrix

from matrix to data frame

```
a <- matrix(1:20, nrow=4)
b <- as.data.frame(a)
```

from data frame to matrix

```
c <- as.matrix(b)
```

Create DF from Vectors

```
gender = c("Female", "Male","Female")
height = c(155, 171.5, 155)
weight = c(71,93, 68)
x4 = cbind(gender,height,weight)
```

Catch: Everything is converted to text

Columns & Rows Names

rownames(x)

colnames(x) or names(x)

Rows & Column Numbers

nrow(x) : Number of Rows

ncol(x) : Number of Columns

Subset Operators

$: Select a single component from the data

[[: Select single component by name or position

[: Select multiple components

Accessing Single Component

Using $ operator

```
x$gender
x$height
```

Using [[operators]]

```
x[["gender"]]
x[[1]]
```

Accessing Multiple Components

Using [operator

```
a[1]
a["gender"]
a[-2]
a[1:2,]
a[c(1,4)]
a[,2]
a[c(2,3),c(1,2)]
```

Filter Data using subset

```
subset(a, select = c("gender", "age"))
subset(a, subset = height > 163, select = c
("gender", "height", "age"))
```

Add Column to Data Frame

```
a$name =c('Ally','Belinda','Jane')
```

Add Rows to Data Frame

```
b <- data.frame(
        gender = c("Female", "Male","Female"),
        height = c(155, 171.5, 155),
        weight = c(71,93, 68),
        age =c(42,38,26),
        row.names =c('SC','Alfred','TC')
        )

c <- rbind(a,b)
```

Summary of Data Frame

```
summary(a)
```

Factor

What is Factor?

Factors are the data objects which are used to categorize the data and store it as levels. They can store both strings and integers

```
data <- factor(c("male","female","female","male"))
is.factor(data)
is.factor(sleep$group)
is.factor(sleep$extra)
```

List

What is List?

Lists are the R objects which contain elements of different types like − numbers, strings, vectors and another list inside it.

Unnamed List

```
a  <-list(1:10, matrix(1:4,ncol = 2),"r" )
```

Named List

List groups different objects

```
a  <-list('A'=1:10, 'B'=matrix(1:4,ncol = 2),'C'="r"
)
```

Accessing List Element

Using [[ ]] operator

a[[1]]

a[['A']]

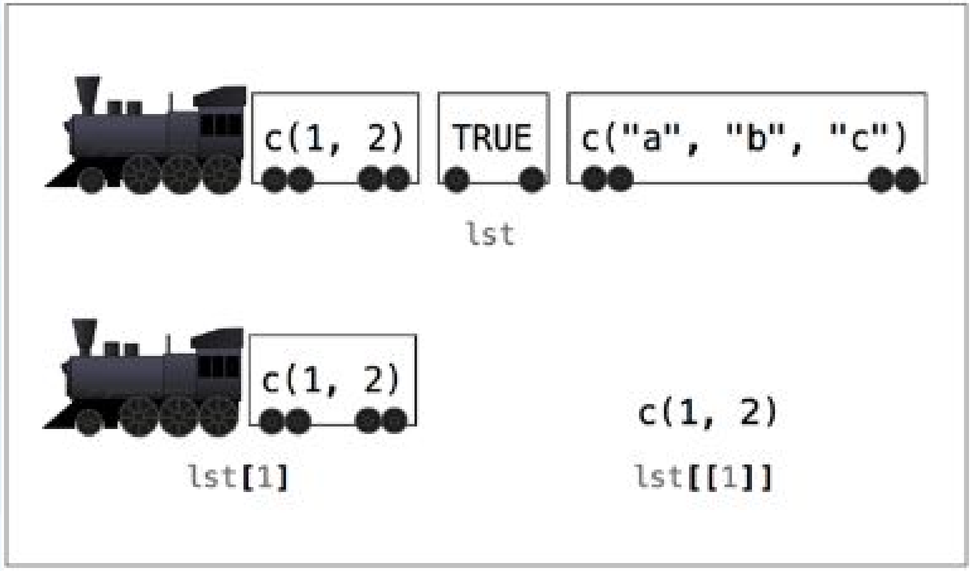
Using [ ] operator

a[1]

a['A']

Using $ operator

a$A



Modifying List Element

```
list1[[1]] = c("d","e","f")
list1$'letters' - c("aa","bb","cc")
```

Merging Lists

```
list1 <- list(1,2,3)
list2 <- list("Sun","Mon","Tue")


merged.list <- c(list1,list2)
```

Converting List to Vector

```
v1 <- unlist(list1)
```

Date

Create a Date

```
xd <-as.Date("2016-03-13")
xd <-as.Date("5 Aug 2016",format="%d %b %Y")
```

%Y	: Year with century
%y	: Year without century
%m	: Month in decimal
%B	: Full month name
%b	: Abbreviated month name
%d	: Day in decimal

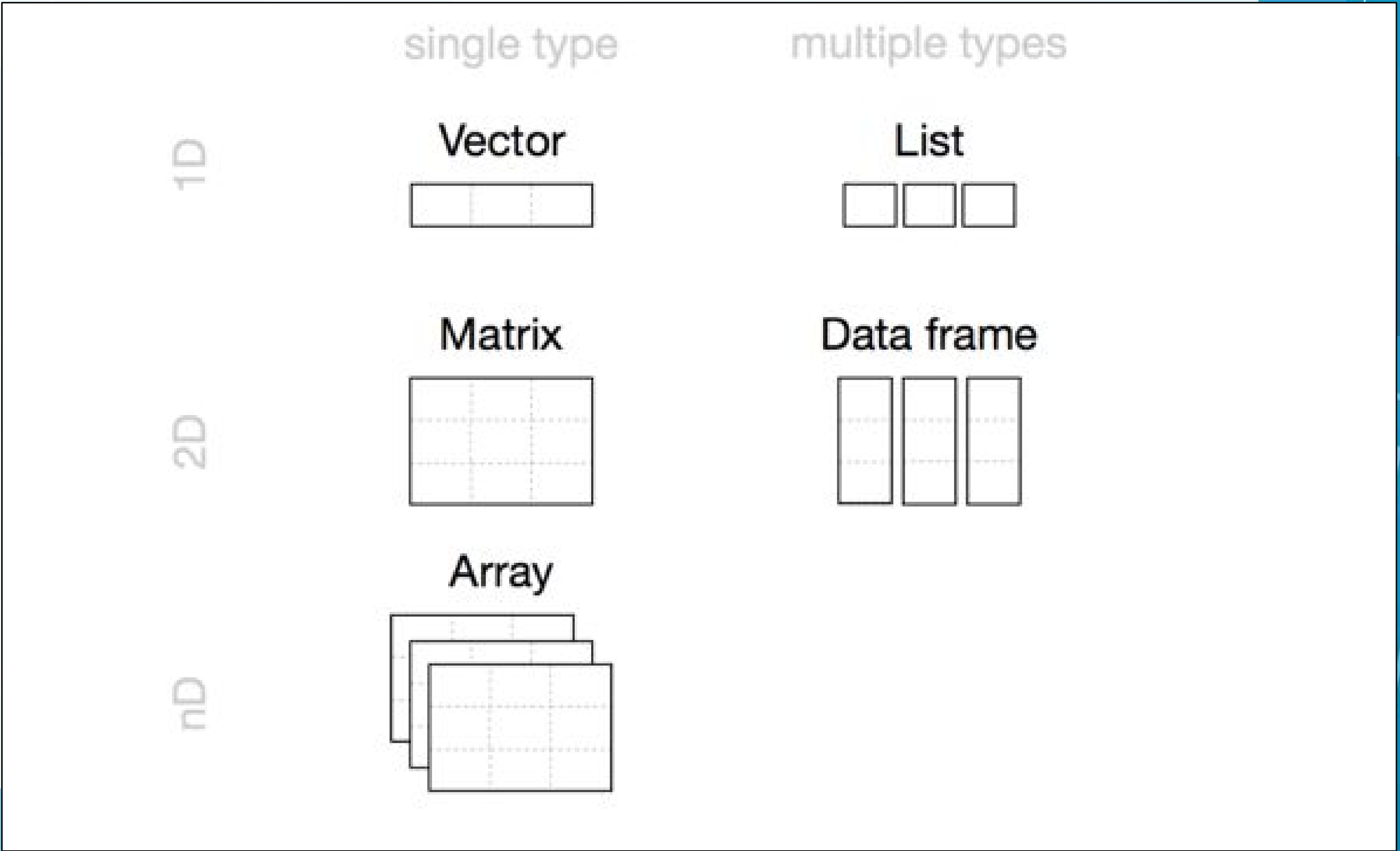
Working with Date

weekdays(xd)

months(xd)

xd+7

Data Types Summary

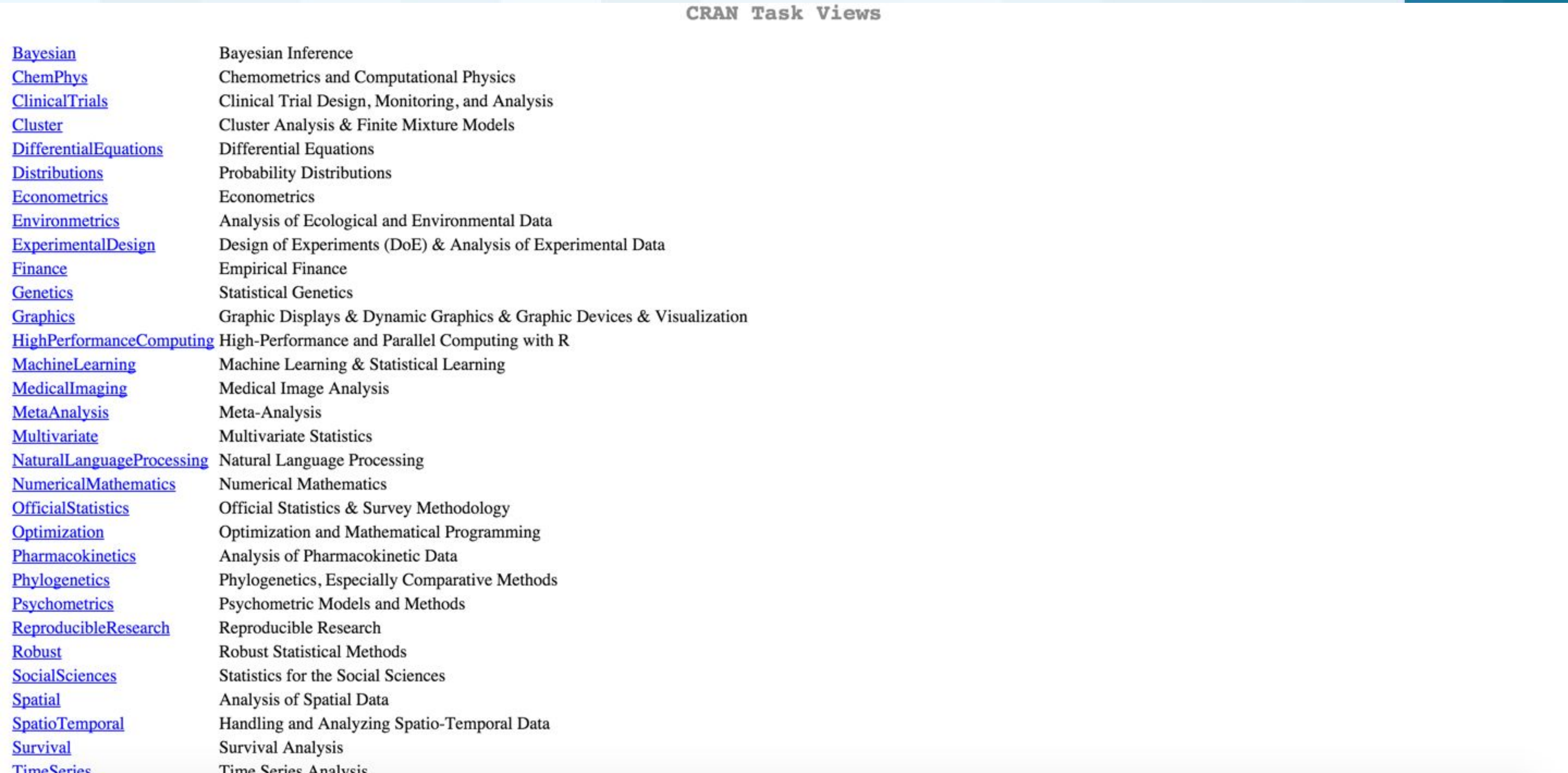


Module 3
Packages & Data Sets

Packages

R Packages

https://cran.r-project.org/web/views/



CRAN Task Views

Bayesian	Bayesian Inference
ChemPhys	Chemometrics and Computational Physics
ClinicalTrials	Clinical Trial Design, Monitoring, and Analysis
Cluster	Cluster Analysis & Finite Mixture Models
DifferentialEquations	Differential Equations
Distributions	Probability Distributions
Econometrics	Econometrics
Environmetrics	Analysis of Ecological and Environmental Data
ExperimentalDesign	Design of Experiments (DoE) & Analysis of Experimental Data
Finance	Empirical Finance
Genetics	Statistical Genetics
Graphics	Graphic Displays & Dynamic Graphics & Graphic Devices & Visualization
HighPerformanceComputing	High-Performance and Parallel Computing with R
MachineLearning	Machine Learning & Statistical Learning
MedicalImaging	Medical Image Analysis
MetaAnalysis	Meta-Analysis
Multivariate	Multivariate Statistics
NaturalLanguageProcessing	Natural Language Processing
NumericalMathematics	Numerical Mathematics
OfficialStatistics	Official Statistics & Survey Methodology
Optimization	Optimization and Mathematical Programming
Pharmacokinetics	Analysis of Pharmacokinetic Data
Phylogenetics	Phylogenetics, Especially Comparative Methods
Psychometrics	Psychometric Models and Methods
ReproducibleResearch	Reproducible Research
Robust	Robust Statistical Methods
SocialSciences	Statistics for the Social Sciences
Spatial	Analysis of Spatial Data
SpatioTemporal	Handling and Analyzing Spatio-Temporal Data
Survival	Survival Analysis
TimeSeries	Time Series Analysis

Install and Load a Package

1. Install package

```
install.packages("ggplot2")
```

2. Load a package

```
library("ggplot2")
```

Update Packages

```
update.packages()
```

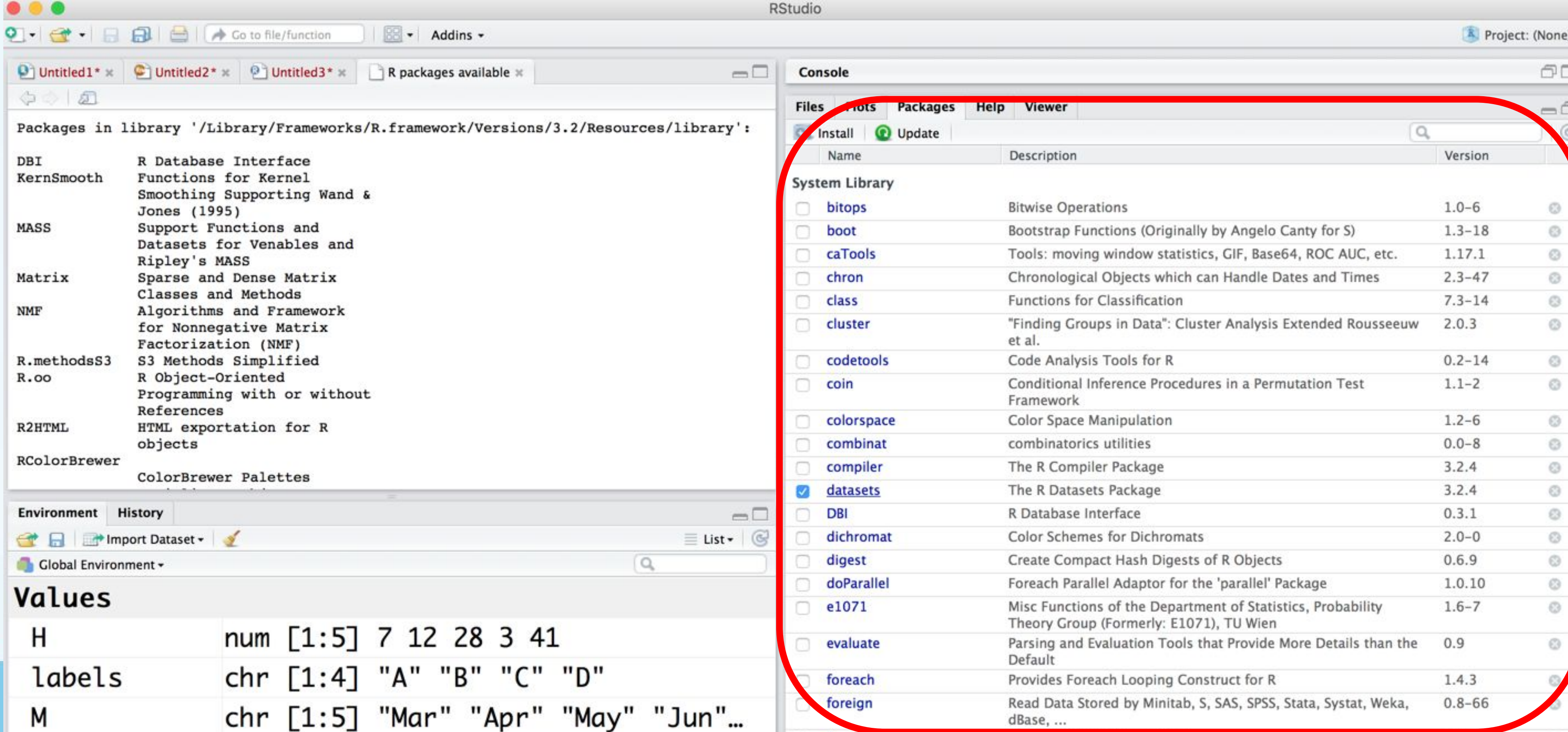
Unload and Remove Package

```
detach("package:ggplot2", unload = TRUE)

remove.packages("ggplot2")
```

RStudio Package Management

install and load package in RStudio

unload and remove a package in RStudio



```
Packages in library '/Library/Frameworks/R.framework/Versions/3.2/Resources/library':

DBI            R Database Interface
KernSmooth     Functions for Kernel
               Smoothing Supporting Wand &
               Jones (1995)
MASS           Support Functions and
               Datasets for Venables and
               Ripley's MASS
Matrix         Sparse and Dense Matrix
               Classes and Methods
NMF            Algorithms and Framework
               for Nonnegative Matrix
               Factorization (NMF)
R.methodsS3    S3 Methods Simplified
R.oo           R Object-Oriented
               Programming with or without
               References
R2HTML         HTML exportation for R
               objects
RColorBrewer
               ColorBrewer Palettes
```

	Name	Description	Version
System Library			
☐	bitops	Bitwise Operations	1.0-6
☐	boot	Bootstrap Functions (Originally by Angelo Canty for S)	1.3-18
☐	caTools	Tools: moving window statistics, GIF, Base64, ROC AUC, etc.	1.17.1
☐	chron	Chronological Objects which can Handle Dates and Times	2.3-47
☐	class	Functions for Classification	7.3-14
☐	cluster	"Finding Groups in Data": Cluster Analysis Extended Rousseeuw et al.	2.0.3
☐	codetools	Code Analysis Tools for R	0.2-14
☐	coin	Conditional Inference Procedures in a Permutation Test Framework	1.1-2
☐	colorspace	Color Space Manipulation	1.2-6
☐	combinat	combinatorics utilities	0.0-8
☐	compiler	The R Compiler Package	3.2.4
☑	datasets	The R Datasets Package	3.2.4
☐	DBI	R Database Interface	0.3.1
☐	dichromat	Color Schemes for Dichromats	2.0-0
☐	digest	Create Compact Hash Digests of R Objects	0.6.9
☐	doParallel	Foreach Parallel Adaptor for the 'parallel' Package	1.0.10
☐	e1071	Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien	1.6-7
☐	evaluate	Parsing and Evaluation Tools that Provide More Details than the Default	0.9
☐	foreach	Provides Foreach Looping Construct for R	1.4.3
☐	foreign	Read Data Stored by Minitab, S, SAS, SPSS, Stata, Systat, Weka, dBase, ...	0.8-66

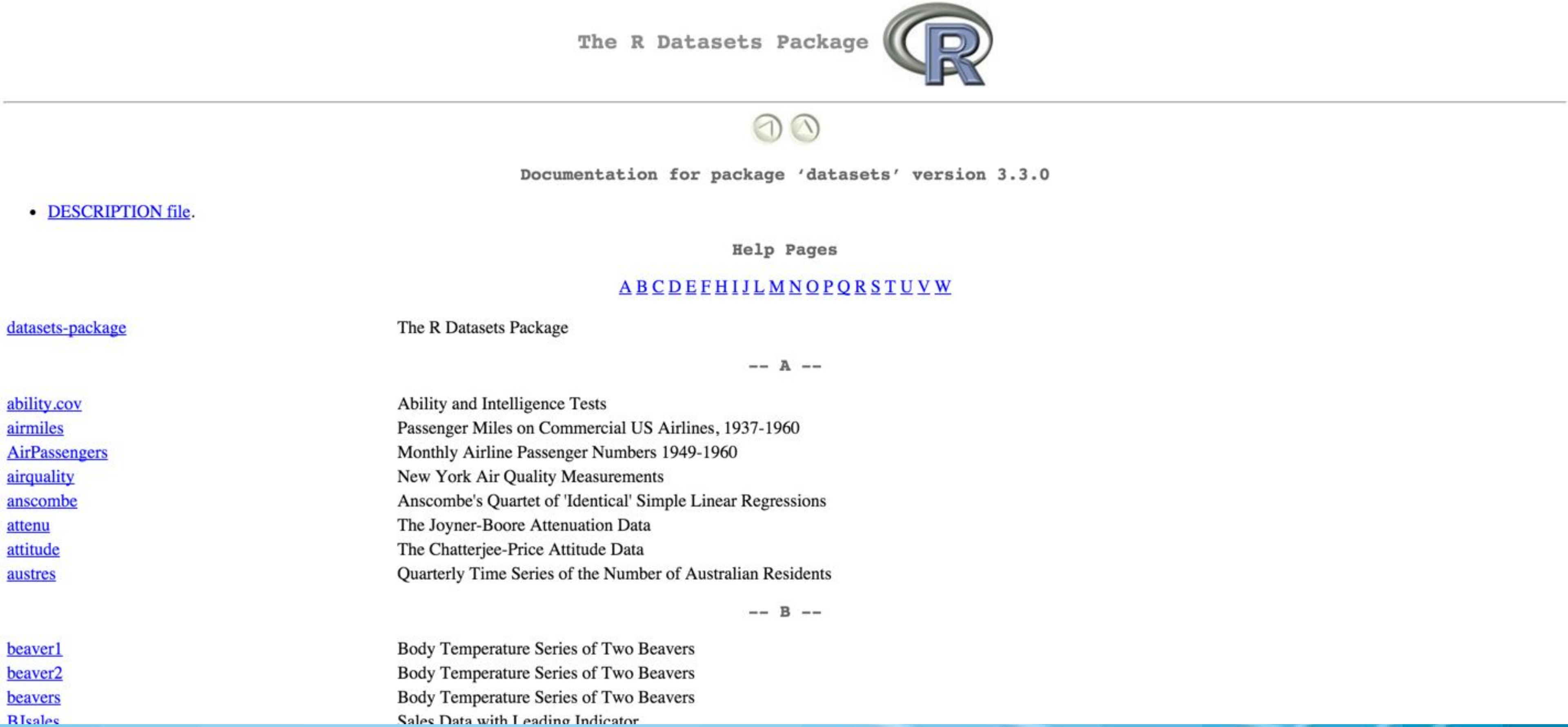
Values

```
H          num [1:5] 7 12 28 3 41
labels     chr [1:4] "A" "B" "C" "D"
M          chr [1:5] "Mar" "Apr" "May" "Jun"…
```

Data Sets

R Dataset Packages

http://stat.ethz.ch/R-manual/R-devel/library/datasets/html/00Index.html



The R Datasets Package

Documentation for package ‘datasets’ version 3.3.0

- DESCRIPTION file.

Help Pages

A B C D E F H I J L M N O P Q R S T U V W

datasets-package	The R Datasets Package
	-- A --
ability.cov	Ability and Intelligence Tests
airmiles	Passenger Miles on Commercial US Airlines, 1937-1960
AirPassengers	Monthly Airline Passenger Numbers 1949-1960
airquality	New York Air Quality Measurements
anscombe	Anscombe's Quartet of 'Identical' Simple Linear Regressions
attenu	The Joyner-Boore Attenuation Data
attitude	The Chatterjee-Price Attitude Data
austres	Quarterly Time Series of the Number of Australian Residents
	-- B --
beaver1	Body Temperature Series of Two Beavers
beaver2	Body Temperature Series of Two Beavers
beavers	Body Temperature Series of Two Beavers
BJsales	Sales Data with Leading Indicator

Load Data Set

```
data(sleep)



# Check the structure of dataase
str(sleep)
```

RStudio Import Dataset

Upload from local file

Upload from Web URL

Module 4
File Input/Output

Working Directory

1. Get current working directory

```
getwd()
```

2. Set current working directory

```
setwd("...")
```

3. Add function

```
source("...")
```

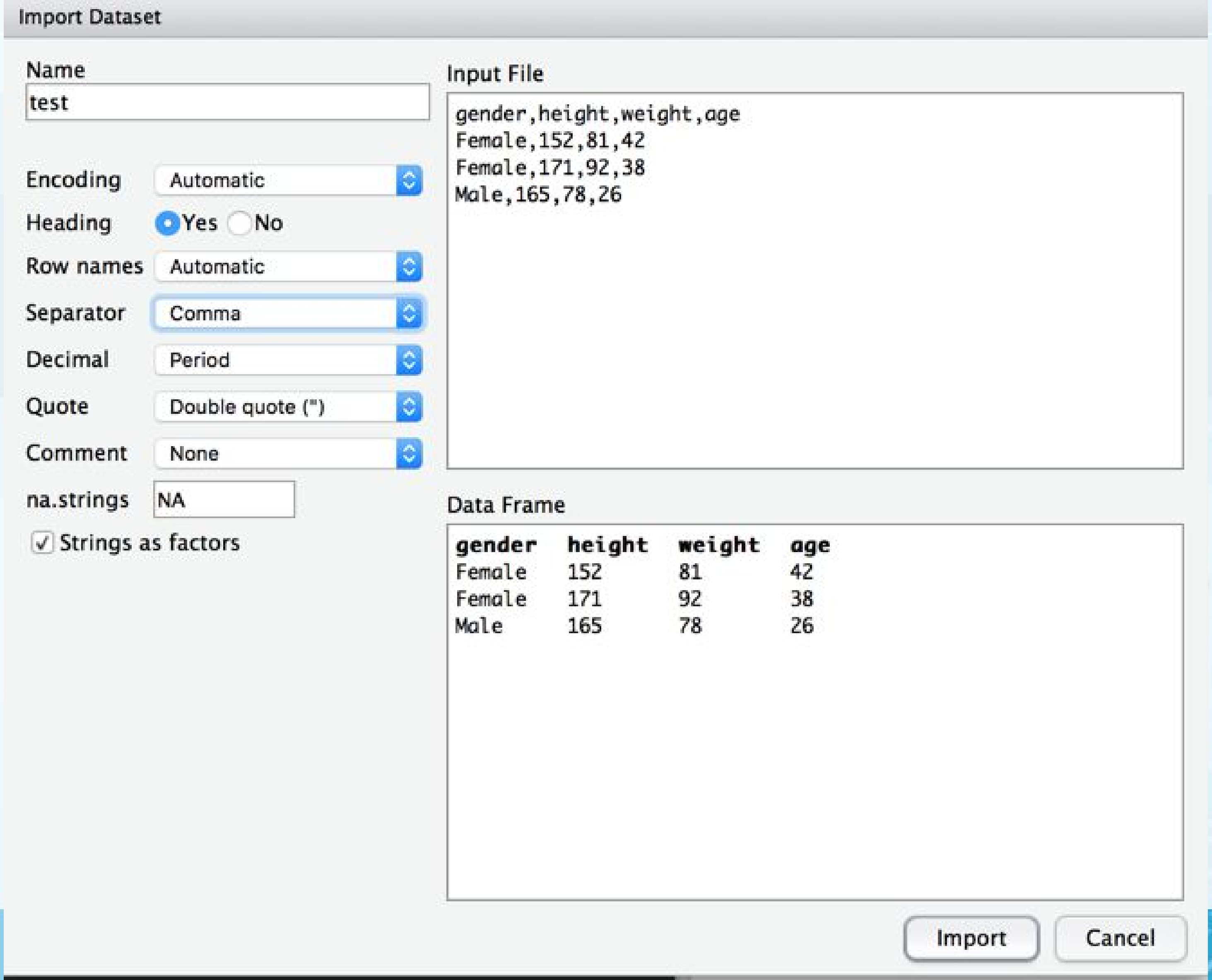
CSV File

Read CSV File

```
a <- read.csv("input.csv", header = TRUE)

View(a)
```

Ex:csvimport.R

Import Dataset in RStudio



Name: test

Encoding: Automatic

Heading: ● Yes ○ No

Row names: Automatic

Separator: Comma

Decimal: Period

Quote: Double quote (")

Comment: None

na.strings: NA

☑ Strings as factors

Input File

```
gender,height,weight,age
Female,152,81,42
Female,171,92,38
Male,165,78,26
```

Data Frame

```
gender  height  weight  age
Female  152     81      42
Female  171     92      38
Male    165     78      26
```

Write CSV File

```
write.csv(a,"output.csv")
```

Ex: CSV Read/Write

1. Read a data set from a csv file
2. Add one more row of data
3. Write the data back to a csv file

Time: 5 min

Text File

Read Text File

```
a <- read.table("file.txt", header = TRUE, sep = "\t")
```

Ex:readwritetext.R

Write Text File

```
write.table(a1, "file.txt", sep="\t")
```

Ex:readwritetext.R

Module 5
Charts

Bar Chart

Bar Chart Syntax

```
barplot(H,xlab,ylab,main, names.arg,col)
```

H - a vector or matrix

xlab - label for x axis.

ylab - label for y axis.

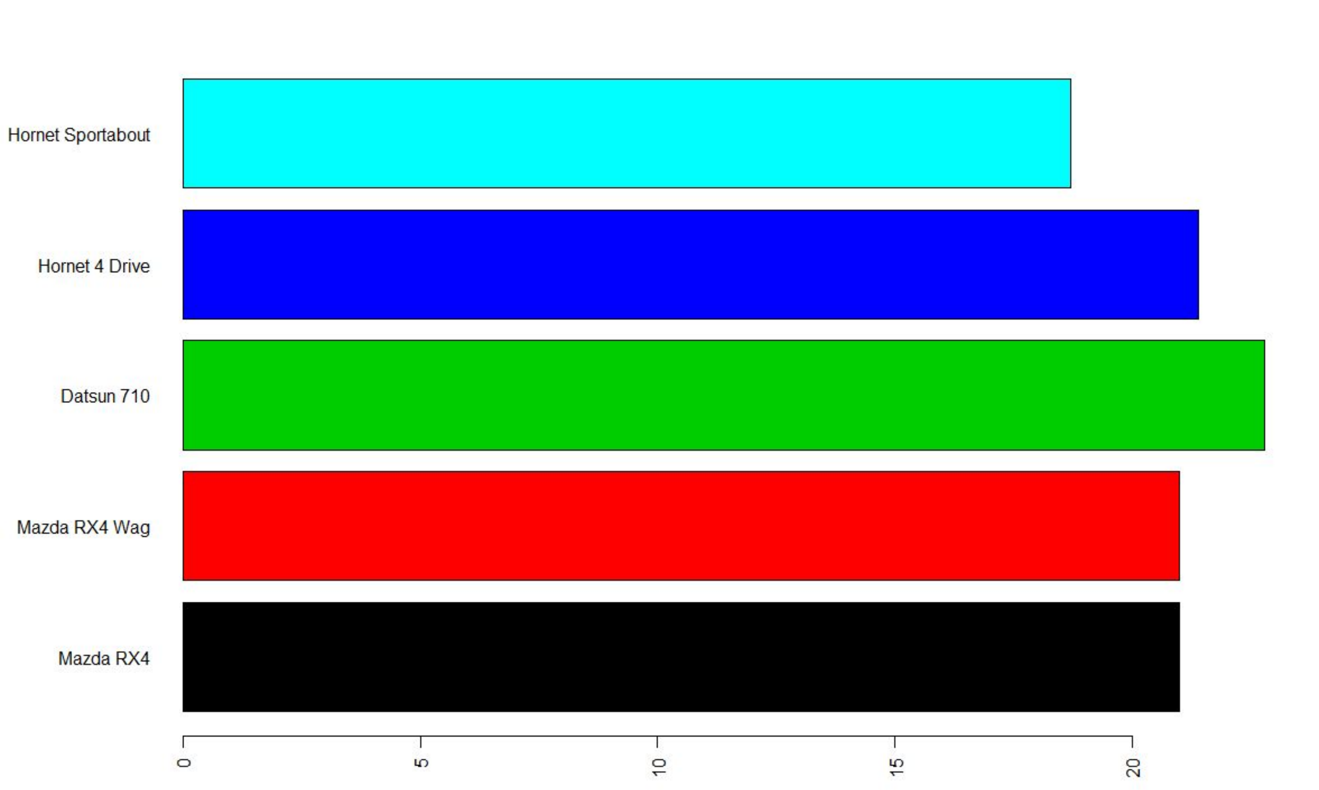
main - title of the bar chart.

names.arg - vector of names appearing under each bar.

col - colors to the bars in the graph.

Bar Chart Example

```
par(mar = c(2.5, 8, 2.5, 2.5))
barplot(mtcars$mpg[1:5], names.arg =   row.names(mtcars)[1:5],
col = 1:5, las = 2, horiz = FALSE)
```



Color List

```
colors()
```

Box Plot

Boxplot Syntax

```
boxplot(x,data,notch,varwidth,names,main)
```

x - formula.

data - data frame.

notch - Set as TRUE to draw a notch.

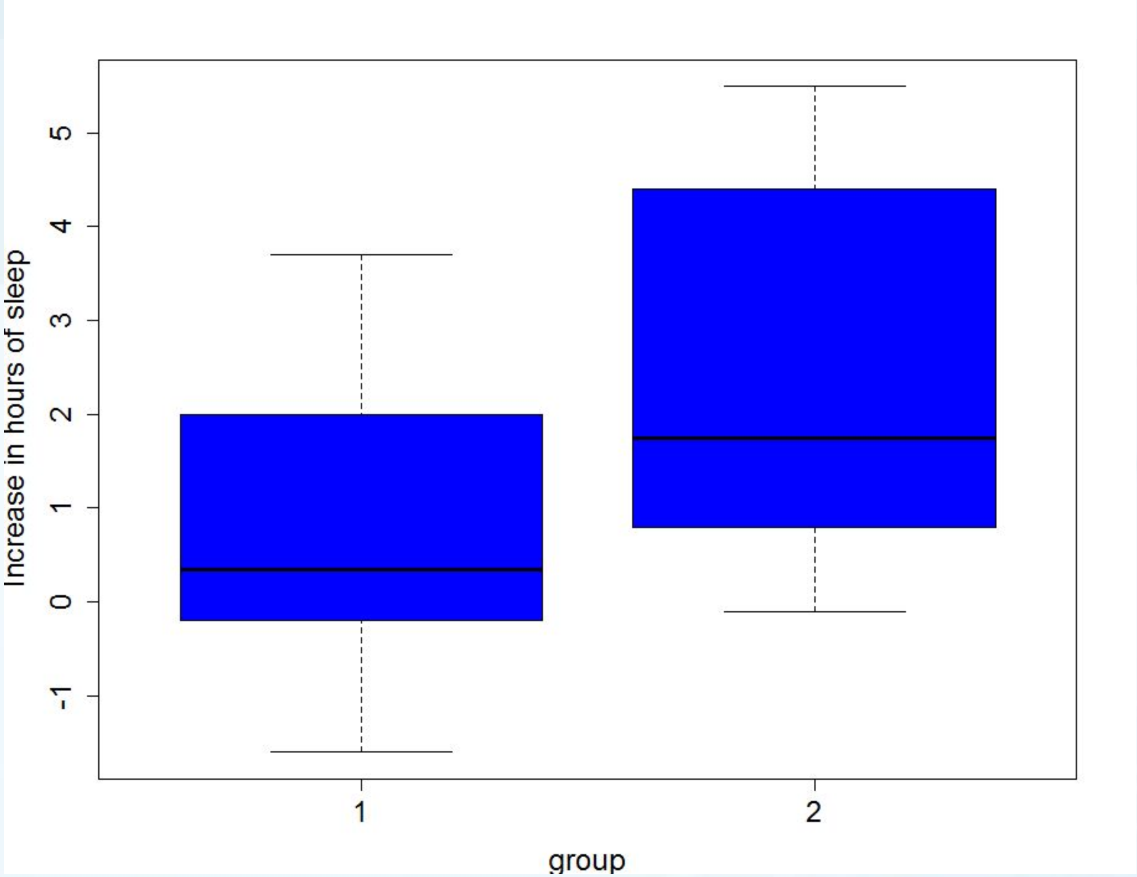
varwidth - Set as true to draw width of the box proportionate to the sample size.

names - group labels

main - title to the graph.

Boxplot Example

```
boxplot(extra ~ group, data = sleep, col = “blue”,
ylab = “Increase in hours of sleep”, xlab = “group”, cex.lab = 1.5,
cex.axis = 1.5)
```



Pie Chart

Pie Chart Syntax

```
pie(x, labels, radius, main, col, clockwise)
```

x - vector

labels - description to the slices.

radius - radius (value between -1 and +1).

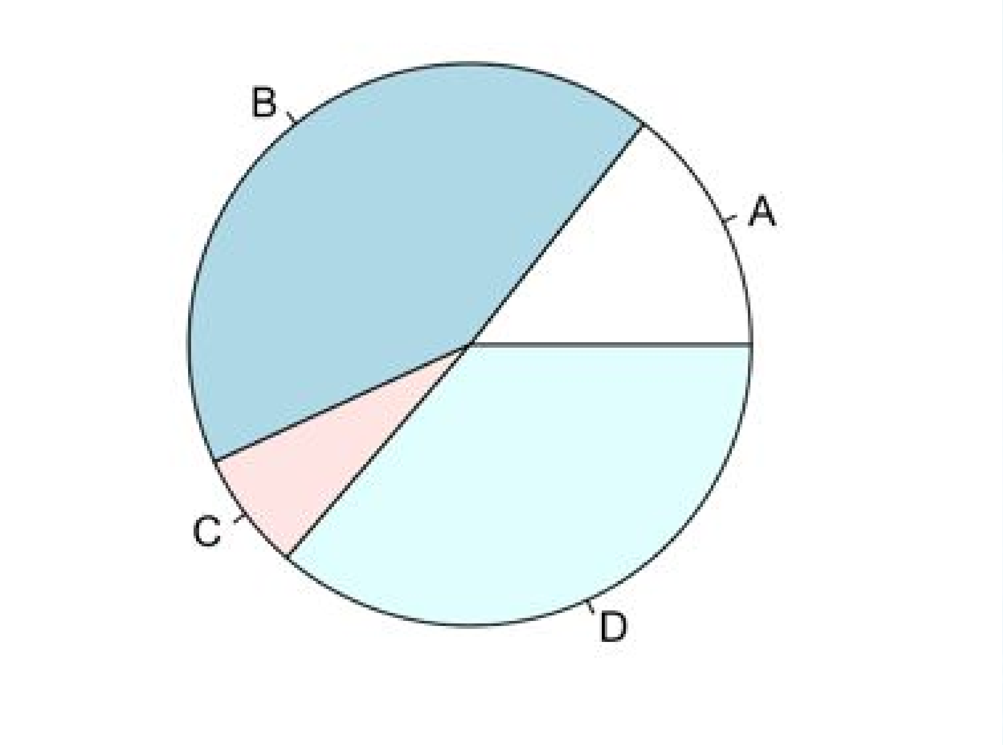
main - title of the chart.

col - color palette.

clockwise - drawn clockwise or anti clockwise.

Pie Chart

```
x <- c(21, 62, 10, 53)
labels <- c("A", "B", "C", "D")
pie(x,labels)
```



Histogram

Histogram Syntax

hist(v,main,xlab,xlim,ylim,breaks,col,border)

v - a vector

main - title of the chart.

col - set color of the bars.

border - set border color of each bar.

xlab - description of x-axis.

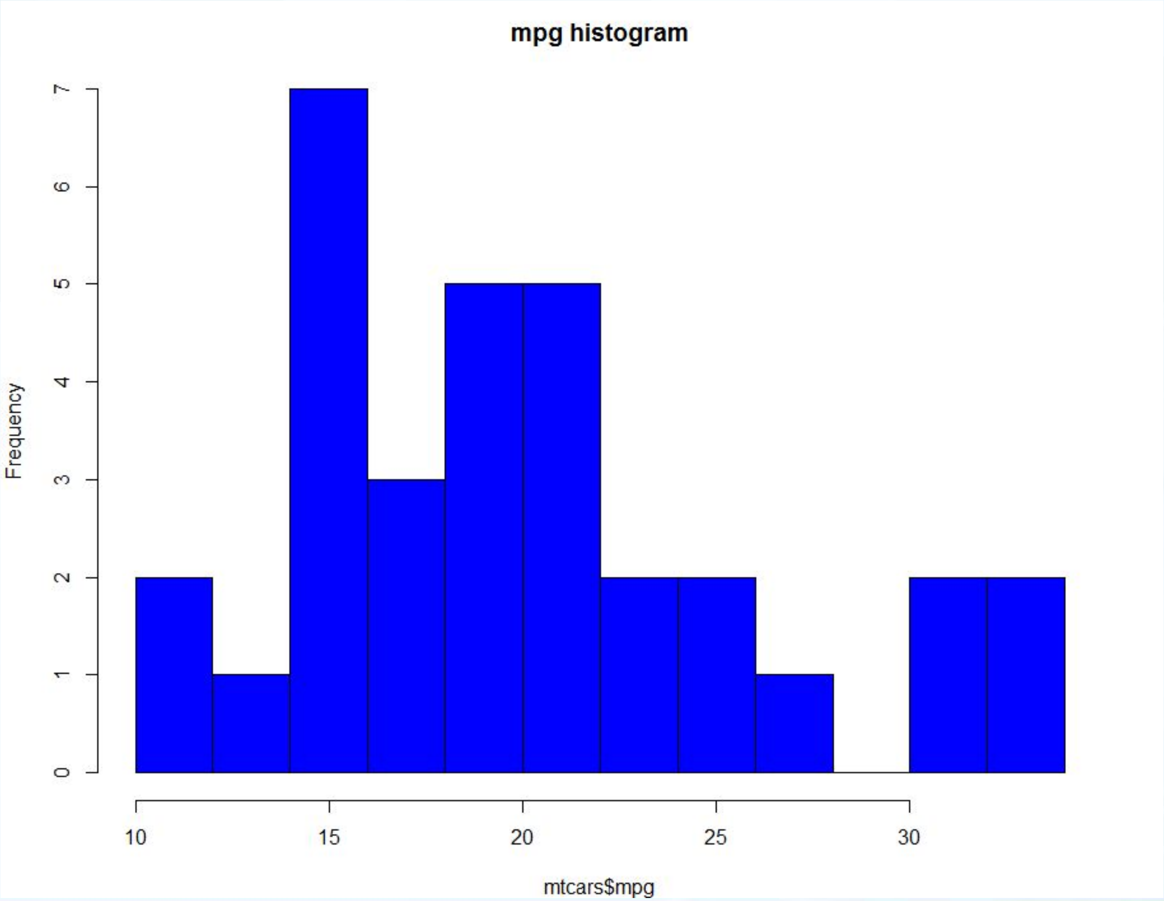
xlim - range of values on the x-axis.

ylim - range of values on the y-axis.

breaks - the width of each bar.

Histogram

```
hist(mtcars$mpg, breaks = 10, col="blue", main = "mpg
histogram")
```



Line Plot

Line Plot Syntax

```
plot(v,type,col,xlab,ylab)
```

v - a vector

type - "p" to draw only the points, "i" to draw only the lines and "o" to draw both points and lines.

xlab - label for x axis.

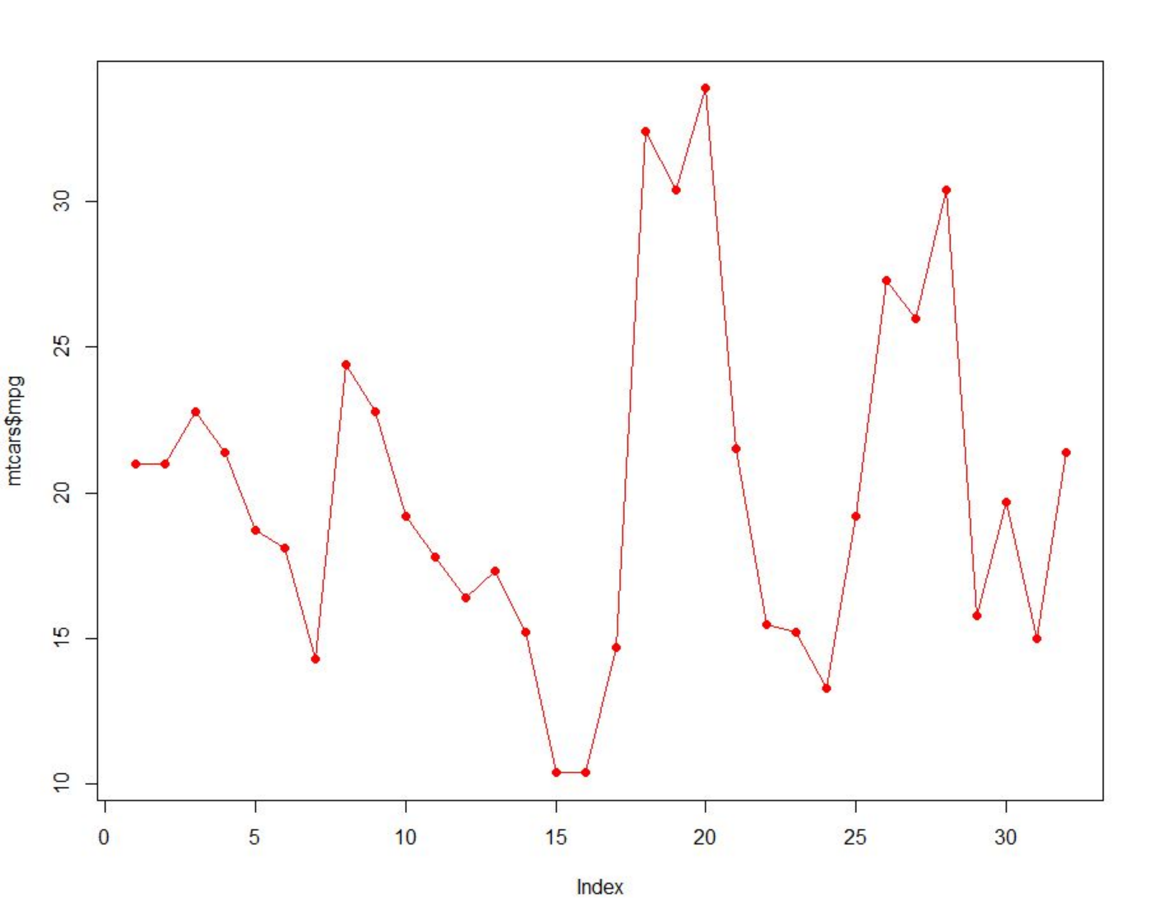
ylab - label for y axis.

main - Title of the chart.

col - colors to both the points and lines.

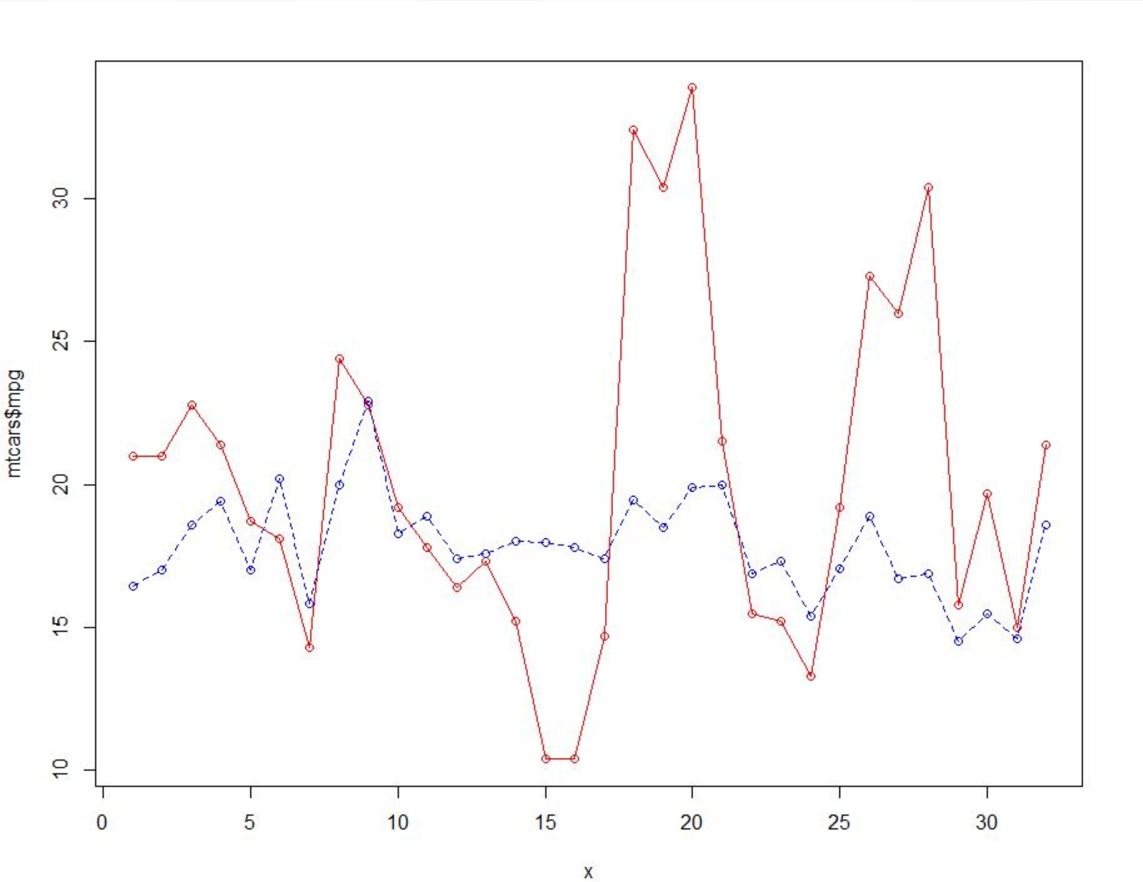
Plot

```
plot(mtcars$mpg, type = "p", col = "red", xlab = "Index", pch = 19)
```



Multiple Lines in a Line Chart

```
plot(mtcars$mpg, type = "o", col = "red", xlab = "x")
lines(mtcars$qsec, type = "o", col = "blue", lty = 2)
```



Scatter Plot

X-Y Scatter Plot Syntax

plot(x, y, main, xlab, ylab, xlim, ylim, axes)

x - x data set

y - y data set

main - Title of the chart.

type - type of points

xlab - label for x axis.

ylab - label for y axis

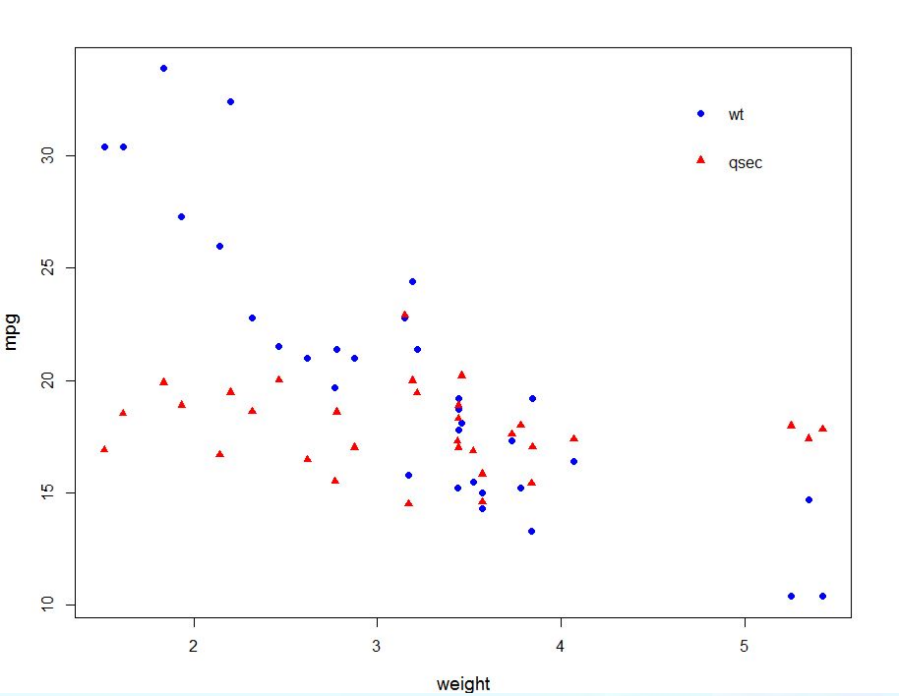
xlim - x limits

ylim - ylimes

axes - axes should be drawn on the plot.

X-Y Scatter Plot Example

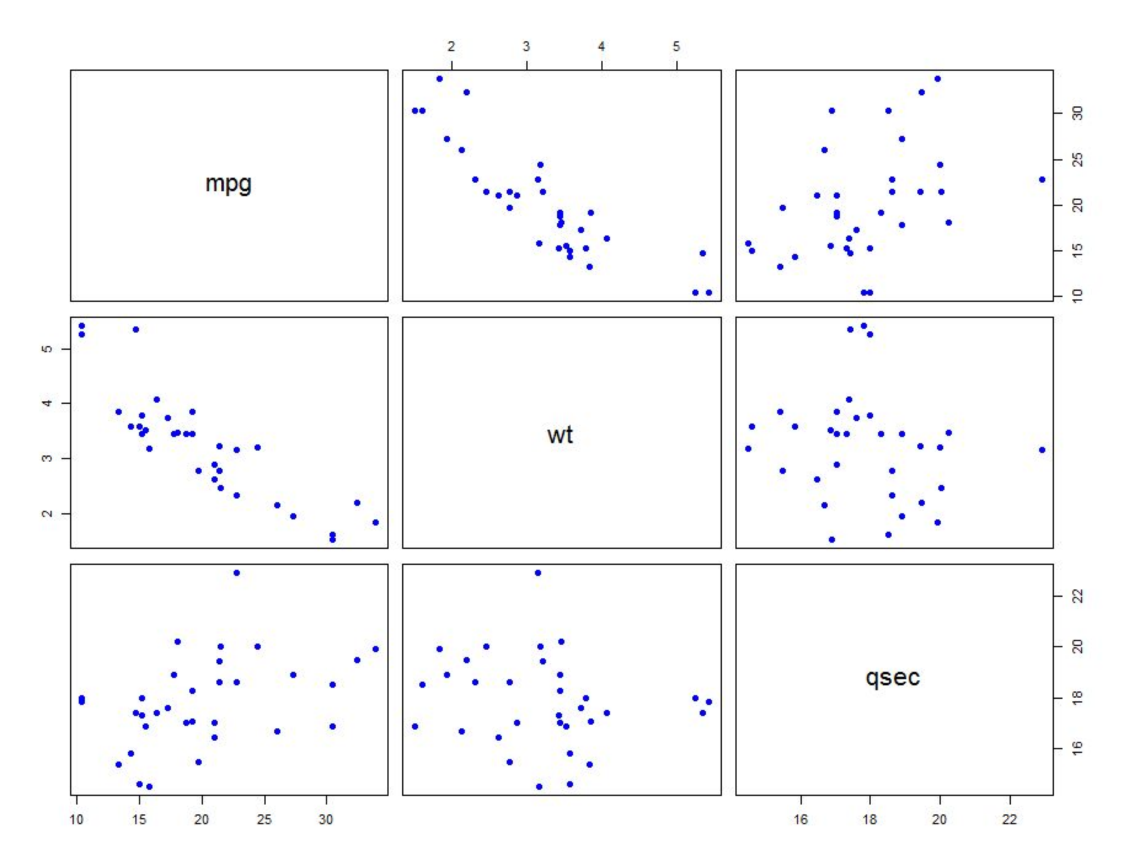
```
plot(mtcars$wt, mtcars$mpg, col = "blue", pch=19, xlab="weight", ylab="mpg",
    cex.lab=1.2)
points(mtcars$wt, mtcars$qsec, col = "red", pch = 17)
legend(x=4.6,y=34, legend=c("wt", "qsec"), pch = c(19,17), col = c('blue', 'red'),
    bty="n")
```



scatterplot.R

Multiple Scatter Plot

```
plot(mtcars[,c(1,6,7)], col = "blue", pch=19)
```

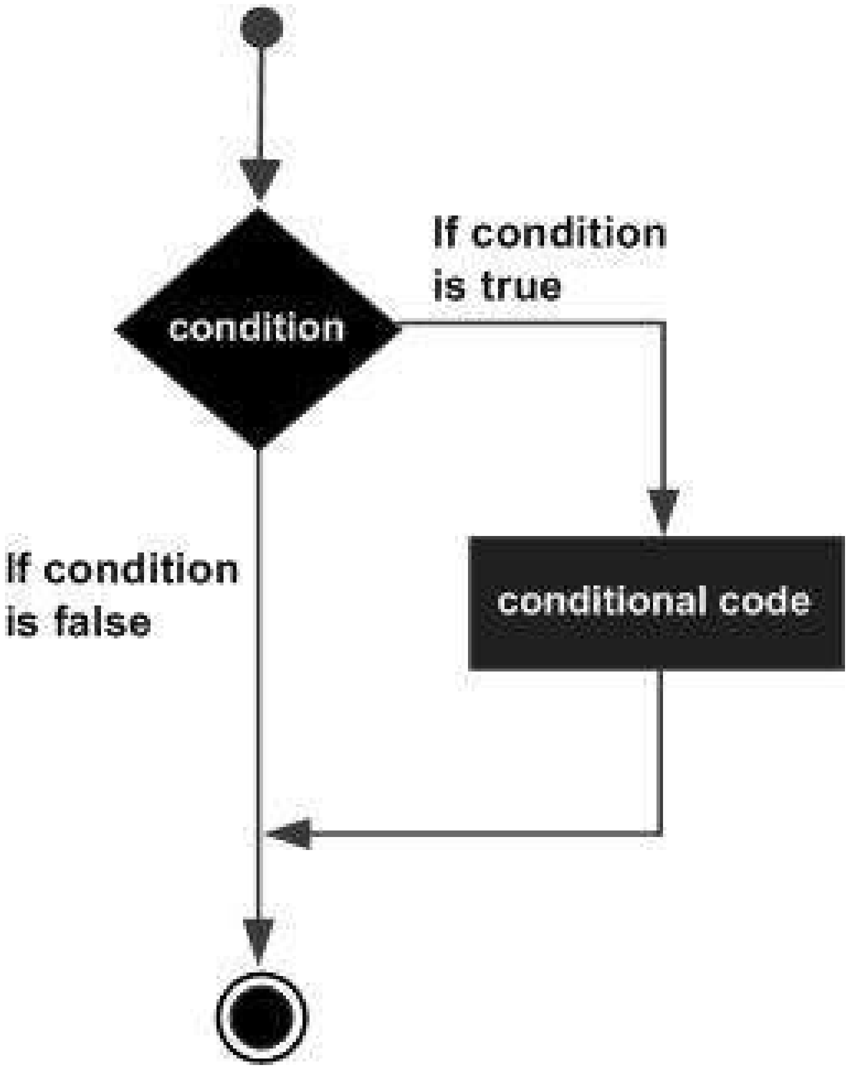


erplot.R

Module 6
Control Structures

Conditional

if Statement



if Syntax

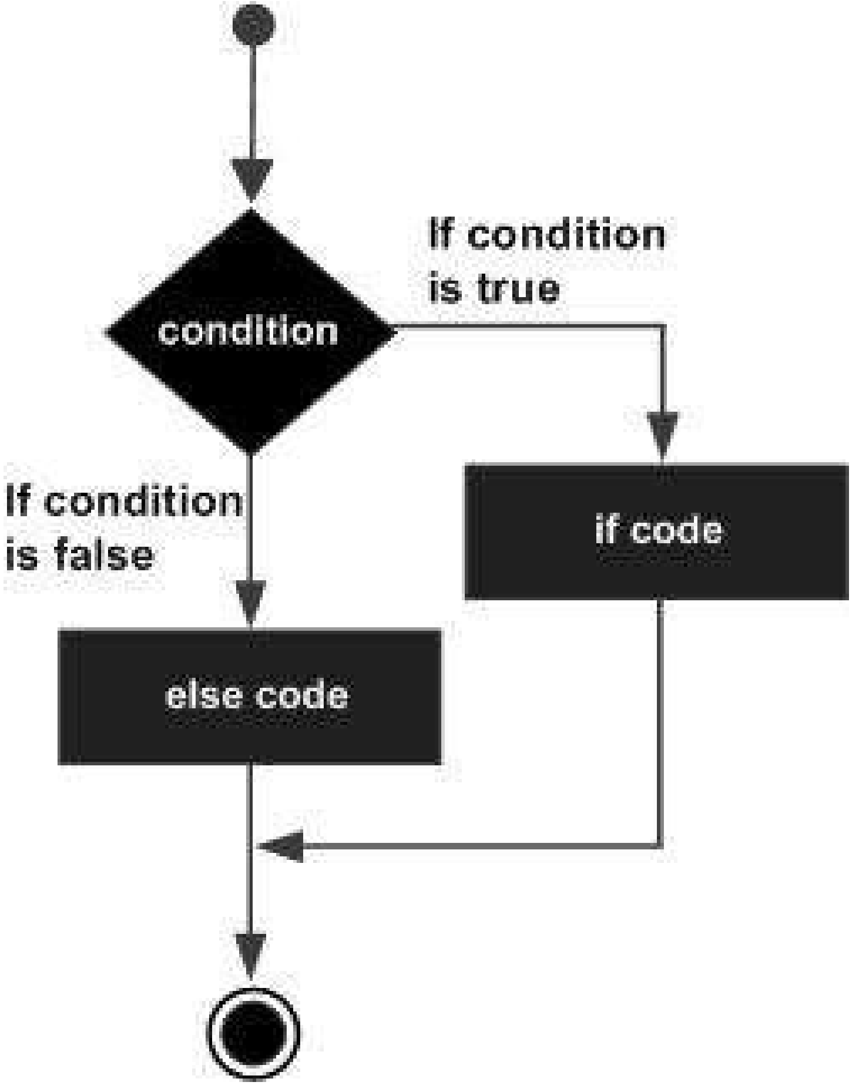
```
if (condition) {
    do Something
    }
```

if Statement Example

```
x <- 3
y <- 4

if (x<y) {
    print("x is smaller than y")
}
```

Ex file: ifelse.R

if-else Statement



if-else Syntax

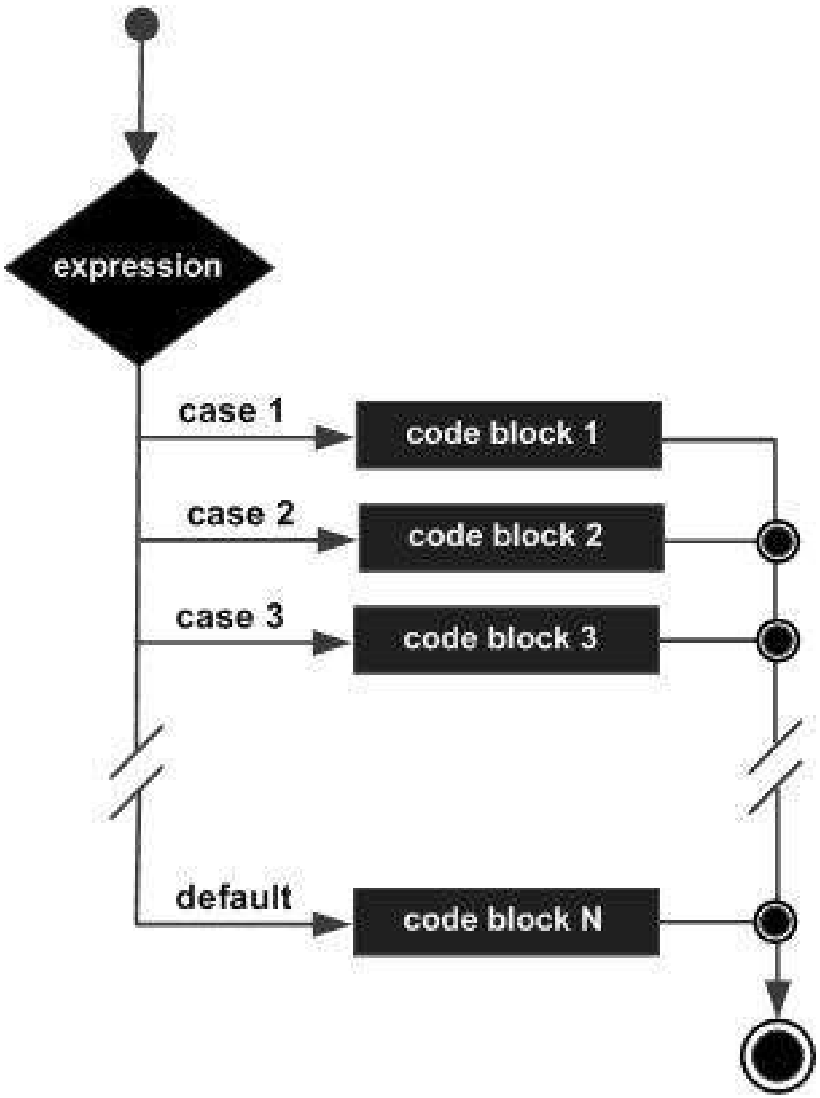
```
if (condition) {
    do Something}
else {
    do Something Else}
```

If-else Statements

```
x <- 5
y <- 4

if (x<y) {
    print("x is smaller than y")
} else {
    print("x is larger than y")
}
```

Ex file: ifelse.R

Switch Statement



Switch Syntax

switch(expression, case1, case2, case3....)

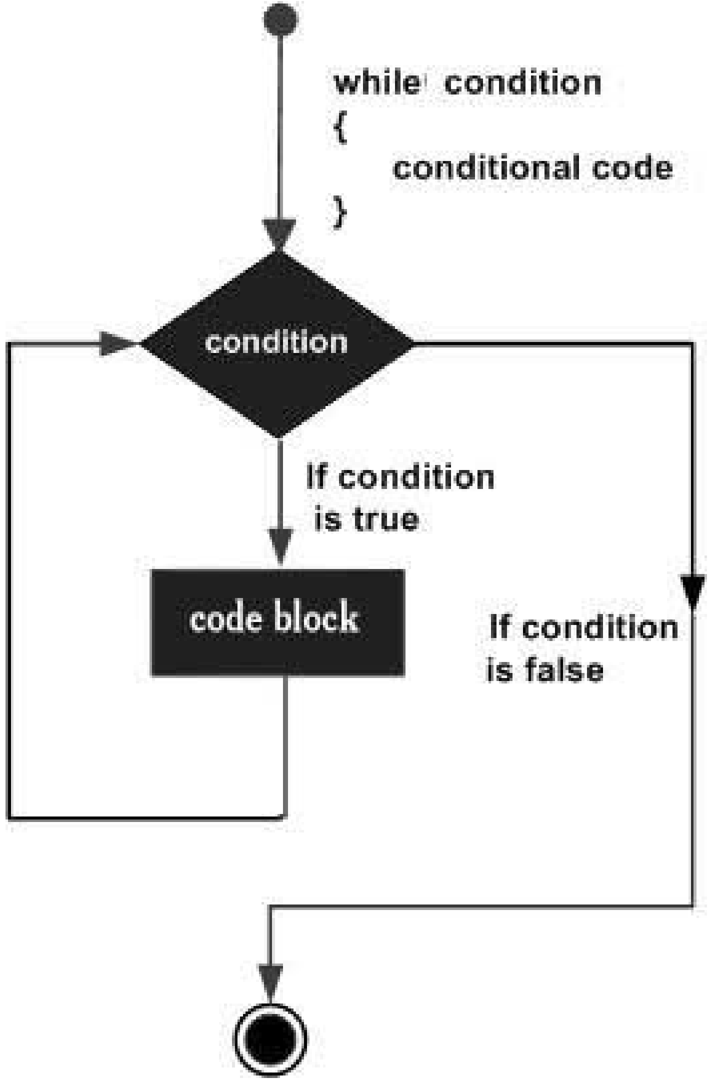
Switch Statements

```
x <- "three"
switch(x,
       zero = print(0),
       one =   print(1),
       two =   print(2),
       three = print(3),
       print("i understand only upto three :("))
```

Ex file: ifelse.R

Loop

While Loop Statement



```
while  condition
{
      conditional code
}
```

While Loops Syntax

```
while (condition) {
    do Something
}
```

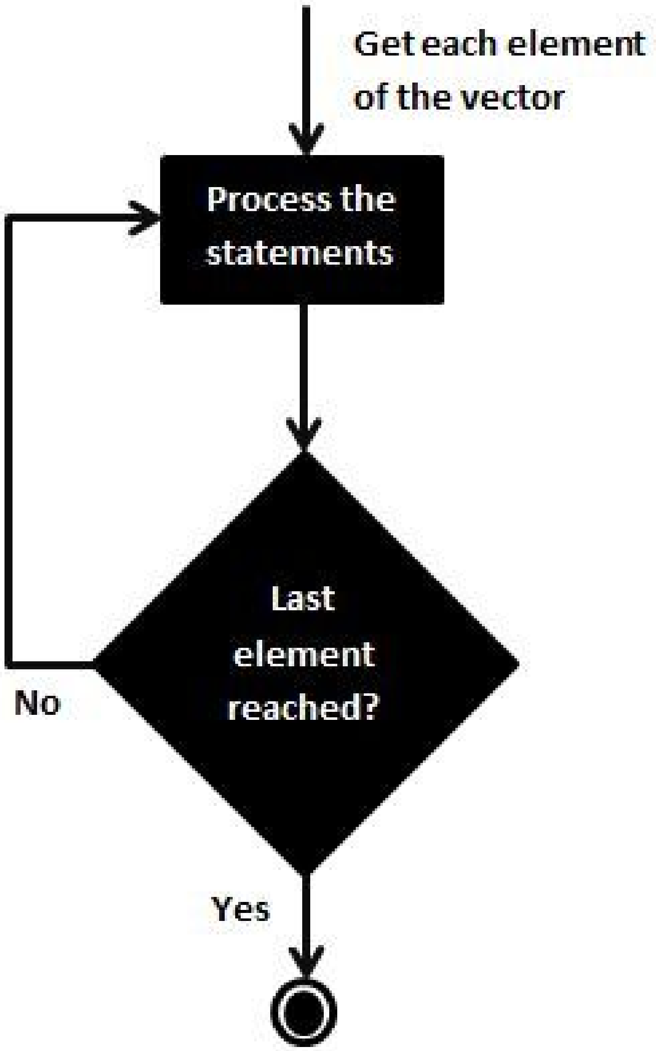
While Loops Examples

```
x <- 0
while (x<10) {
    print(x)
    x = x+1
 }
```

While loop can potentially be infinite loop, be careful!

Ex file: whileloop.R

For Loop Statement



For Loops Syntax

```
for (value in vector) {
    statements
}
```

For Loops Example

```
v <- c(1,2,3,4,5)
for ( i in v) {
   print(i)
}
```

Ex file: forloop.R

Next

```
for ( i in 1:10) {
    if (i == 7) {next}
    print(i)
}
```

Ex file: next.R

Break

```
for ( i in 1:10) {
    if (i == 7) {break}
    print(i)
}
```

Ex file: Break.R

Challenge

1. Read data1.csv
2. Do a subset of month of May and June
3. Count the number of days where temp is more than 65

Time: 10 mins

Hint to Challenge

```
data <- read.csv('data1.csv',header=TRUE)
data.month <- subset(data, Month<7)
data.month.temp <- data.month$Temp
a = 0
for (i in data.month.temp) {
  if (i >65) { a = a +1}
}
print(a)
```

Operators

Arithmetic Operators

Addition	+
Substration	-
Multiplication	*
Division	/
Modulus	%%

Logical Operators

and	&&
or	||
not	!

elementwise

and	&
or	|

Module 7
Function

Function syntax

```
variable_name  <- function(arg_1, arg_2, ...) {
   Function body
}
```

The last expression is the return value

Built In Functions in R

```
factorial(3)
mean(1:6)
```

Function Examples

```
f <- function(x,y) {
  x*x+y*y*y
}

filter <- function(x) {
  x[x>0]
}
```

Ex file: function.R

Function with Default Args

```
above10 <- function(x,n=10) {
  x[x>n]
}

f <- function(a, b = 1, c = 2, d = NULL) {
}
```

Ex file: function.R

Named Args

```
f <- function(x,y) {
  x*x+y*y*y
}
f(x=3,y=2)
f(x=2,y=3)
```

Argument Matching

The order of argument matching is

1. Check for exact match for a named argument
2. Check for a partial match
3. Check for a positional match

... Argument

The ... argument indicate a variable number of arguments that are usually passed on to other functions.

```
f <- function(x,y,...) {
  plot(x,y,...)
}
```

Eg

```
f(x,y,col="red",main="sine")
```

Nested Function

```
make.power <- function(n) {
  pow <- function(x) {
    x^n
  }
  pow
}

cube = make.power(3)
cube(4)
```

Challenge: Function

Write a function to roll 2 dices, return the sum of the 2 dices

1 dice : 1, 2,3,4,5,6

Advanced Functions

Apply

Applies a function to sections of an array and returns the results in an array.

Syntax

```
apply(array, margin, function, ...)
```

margin: 1 (row), 2 (column)

Eg

```
apply(mtcars, 2, max)
apply(mtcars, 2, mean)
```

Lapply

Applies a function to elements in a list or a vector and returns the results in a list.

Syntax

```
lapply(list, function, ...)
```

Eg

```
lapply(mtcars, max)
lapply(mtcars, mean)
```

Module 8
Statistical Application of R

Basic Statistics

Mean Syntax

```
mean(x, trim = 0, na.rm = FALSE, ...)
```

trim is used to drop some observations from both end of the sorted vector

na.rm is used to remove the missing values from the input vector.

Eg:

```
mean(x)
mean(x,trim=0.3)
```

Ex:selectingdata.R

Median Syntax

```
median(x, na.rm = FALSE)
```

x is the input vector.

na.rm is used to remove the missing values from the input vector.

Ex:selectingdata.R

Summary

```
summary(x)
```

Min, 1st Q, Median, Mean, 3rd Q, Max

Ex:selectingdata.R

Sample

```
sample(1:6,size=2)
sample(1:6,size=2, replace=TRUE)
sample(c('head','tail'),size=1,prob=c(0.2,0.8))
```

Challenge : Sample

Create a function to return the sum of two dices rolling

Hint to Challenge

```
roll <- function() {
    …
    dice <-sample(1:6,size=2,replace=TRUE)
    ….
}
```

Correlation

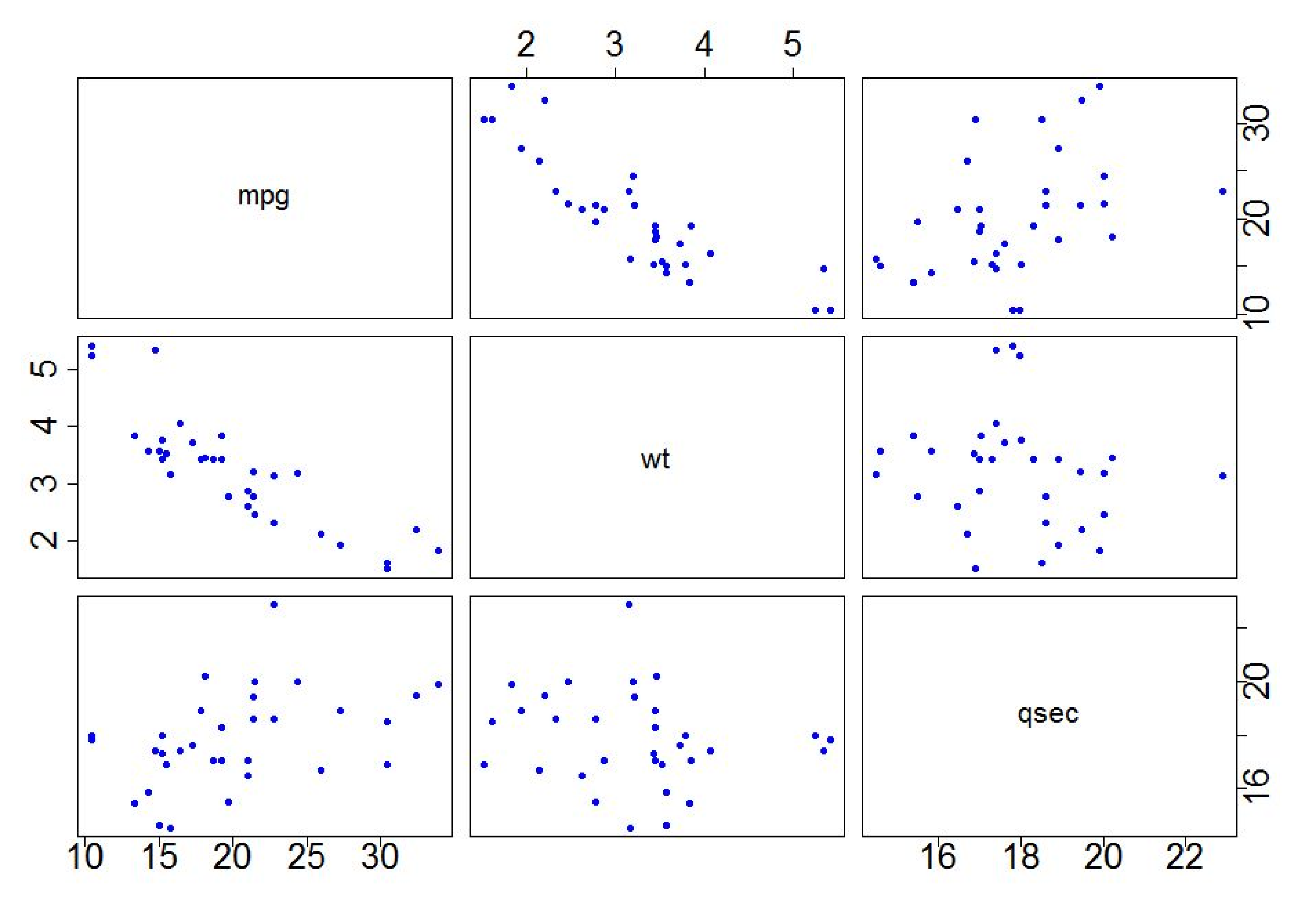
Correlation

It provides a measure of strength and direction of linear relationship between two variables

```
> corMat <- cor(mtcars[,c(1,3,6)])
> round(corMat,2)
         mpg    disp     wt
mpg     1.00   -0.85   -0.87
disp   -0.85    1.00    0.89
wt     -0.87    0.89    1.00
```

Ex:correlation.R

Relationship among selected variables of mtcars dataset

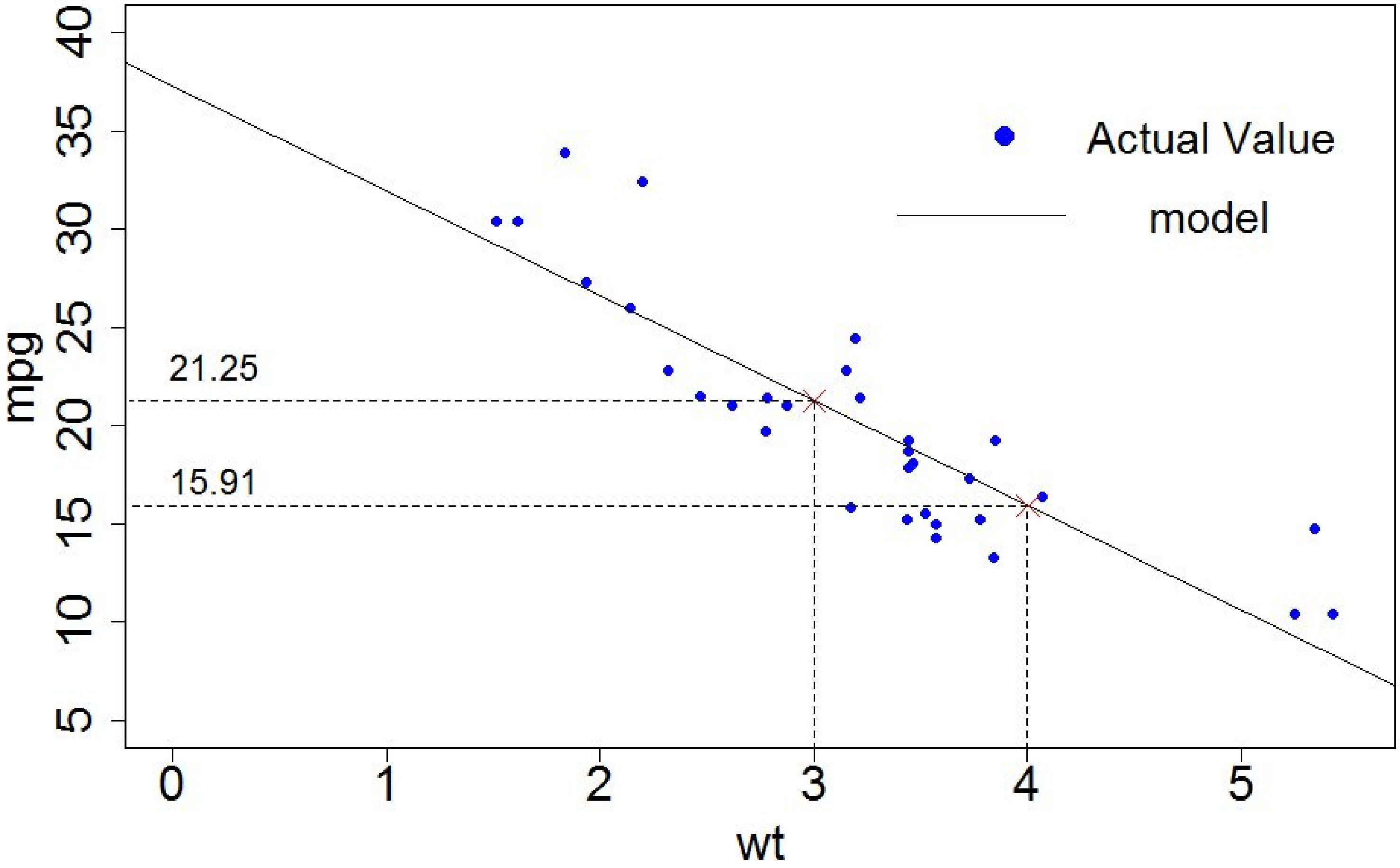


Linear Regression

Linear Regression

- Used for modeling quantitative response
- Involves finding a straight line that best describes the relationship between the response variable and the predictor variables
- The best fit line is then used for making prediction

Linear Regression Example: mtcars dataset



Linear Regression Code

```
> lmModel <- lm(mpg ~ wt, data = mtcars)
> predValue <- predict(lmModel, data.frame
(wt = 3))

> coef(lmModel)
> sumModel <- summary(lmModel)
> sumModel$r.squared

> lmModel <- lm(mpg ~ wt + qsec, data =
mtcars)
```

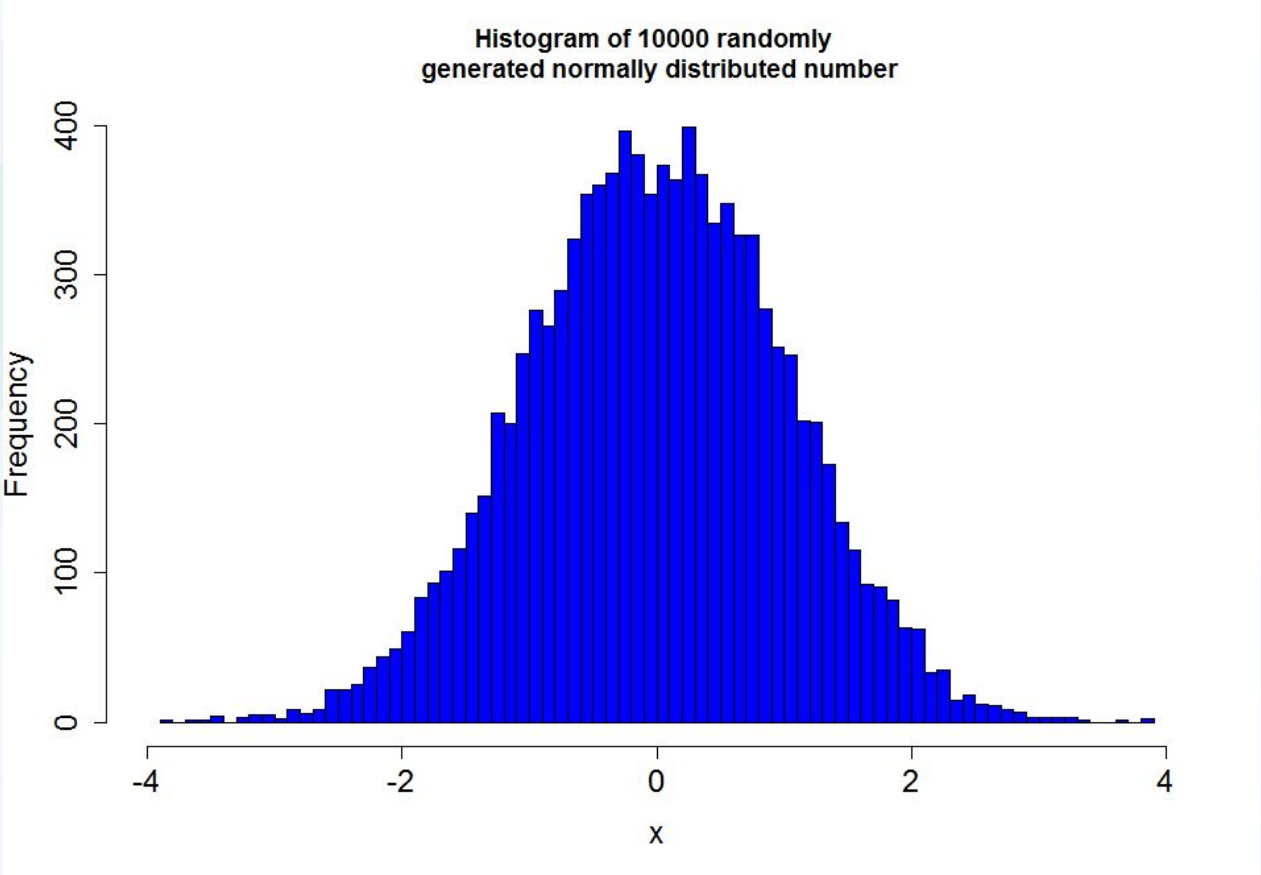
Distributions

Probability Distribution

It is a function that gives the theoretical probability of observing a random variable to have a particular value when the variable is discrete or to fall within a certain range.

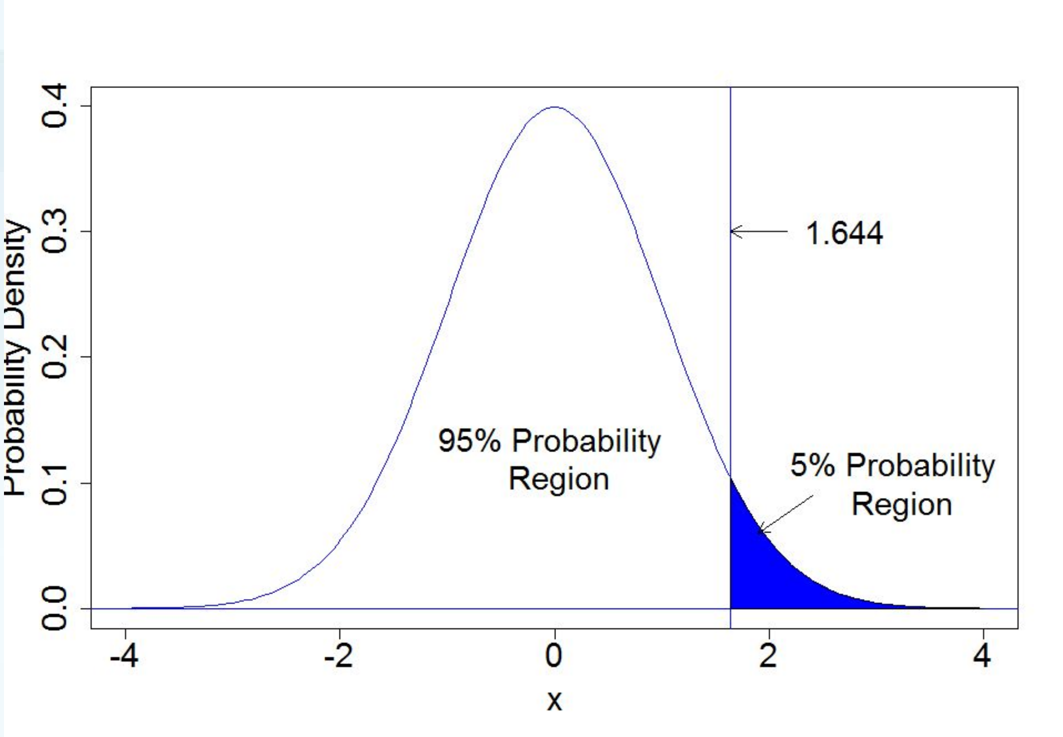
The most famous probability distribution is the **normal (Gaussian) distribution**

Histogram of 10000 Normally Distributed numbers



```
> rnorm(n, mean = 0, sd = 1)  # generates n normally distributed numbers
```

Normal Distribution: probability density and related quantities



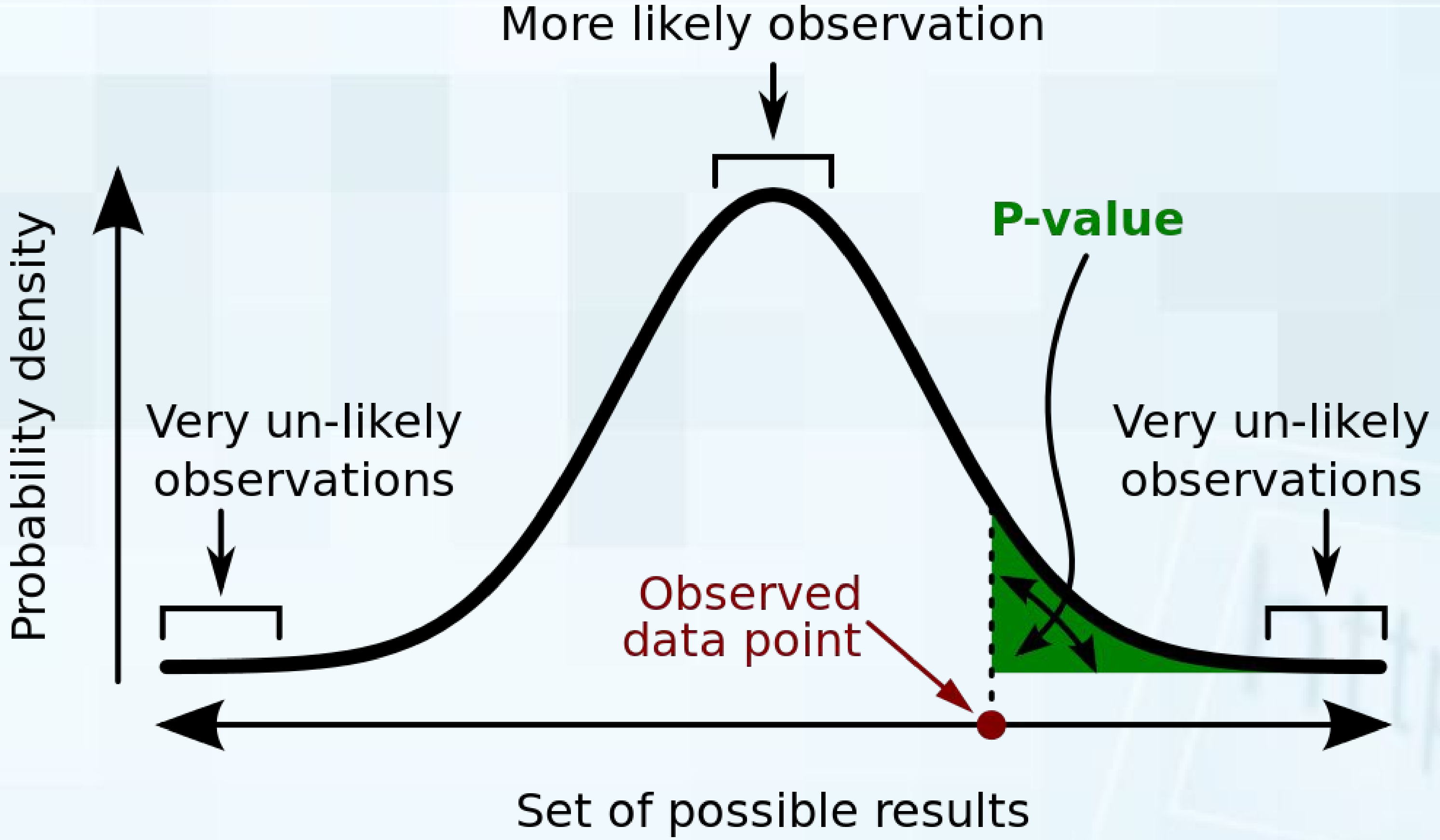
```
> x <- seq(from = -4, to = 4, length.out = 200)
> dnorm(x, mean = 0, sd = 1)  # generates probability density
> qnorm(p=0.95, mean = 0, sd = 1) # gives the quantile associated with p
> pnorm(q=1.644, mean = 0, sd = 1) # gives the probability associated with q
```

Hypothesis Testing

Hypothesis Testing

- Used to determine if different sets of observations come from the same underlying distribution
- Under NULL HYPOTHESIS assumption it is assumed that they come from the same distribution
- If the p-value assuming that the NULL HYPOTHESIS is true is less than 0.05 then NULL HYPOTHESIS is rejected

P-Value



A **p-value** (shaded green area) is the probability of an observed (or more extreme) result assuming that the null hypothesis is true.

Guideline for p-value

If p value > .10 → “not significant”

If p value ≤ .10 → “marginally significant”

If p value ≤ .05 → “significant”

If p value ≤ .01 → “highly significant.”

Module 9.
Intro to Advanced Data Analysis (optional)

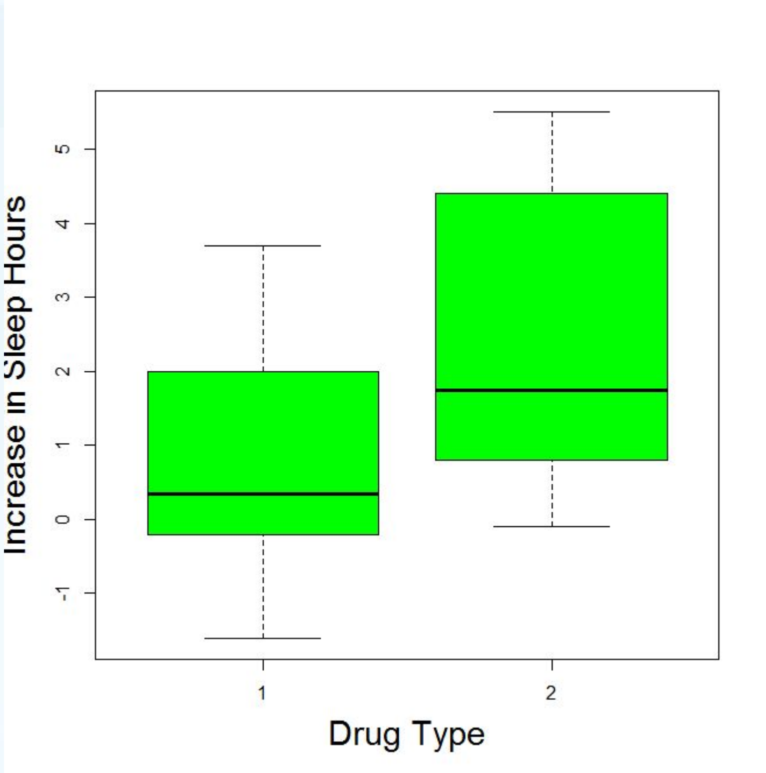
T-Test

T-Test Application

To test if the mean of two different groups of observations differ, when the observations were carried out

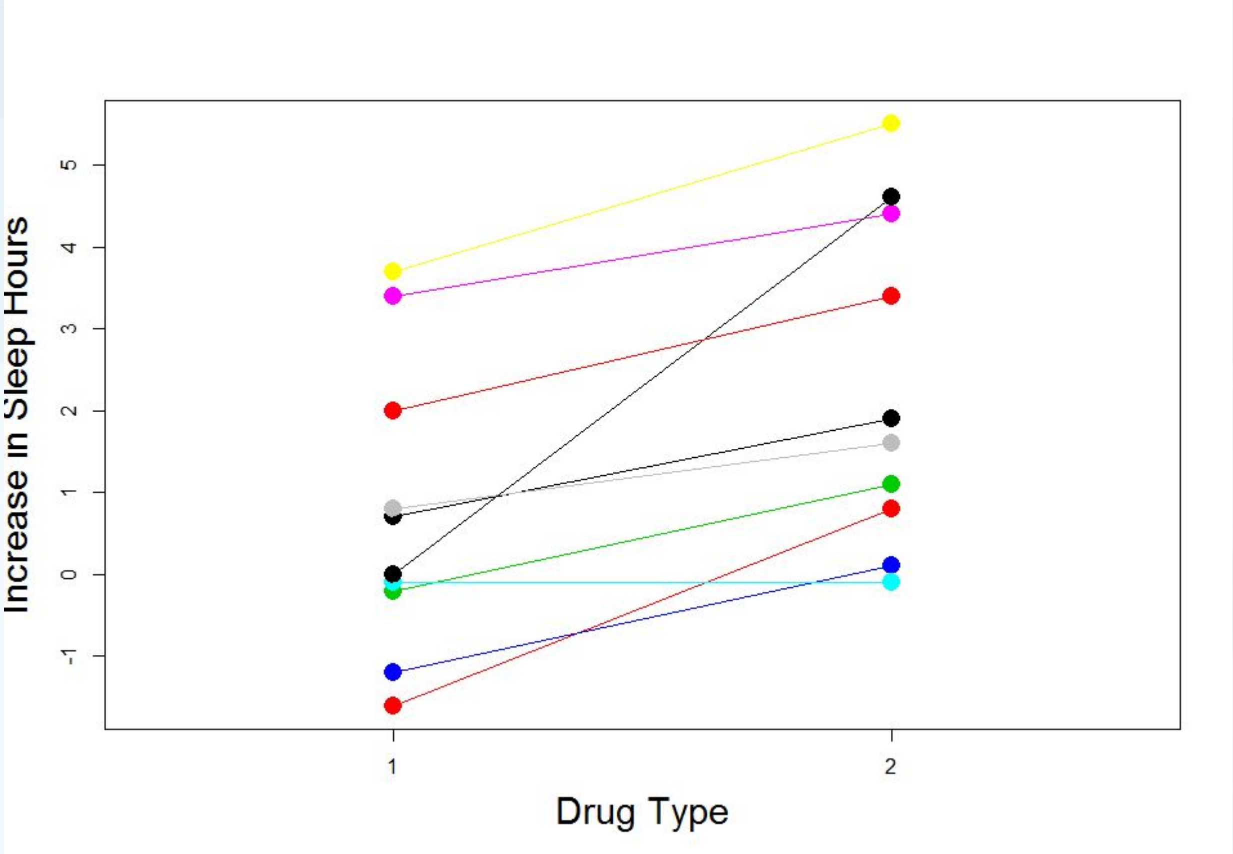
1) on two different sets of participants (independent group)
2) the same set of participants at different time (paired)

Are the mean increase in sleep hours different for the two drug types: 1



```
> t.test(extra~group,data = sleep)
```

Are the mean increase in sleep hours different for the two drug type: 2



```
> t.test(extra ~ group, sleep, paired=TRUE)
```

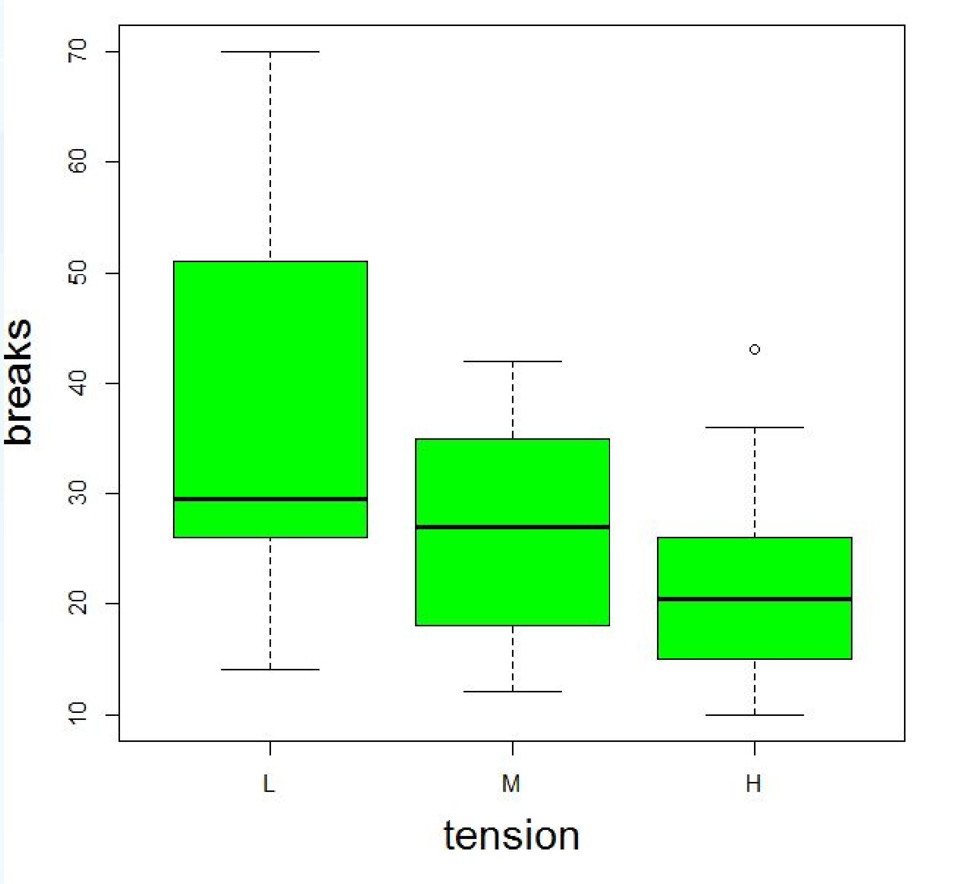
ANOVA

When there are more than 2 groups then we use ANOVA to test if there are statistically significant difference between the means.

Anova does this by analysing the variance in the data set.

One Way ANOVA

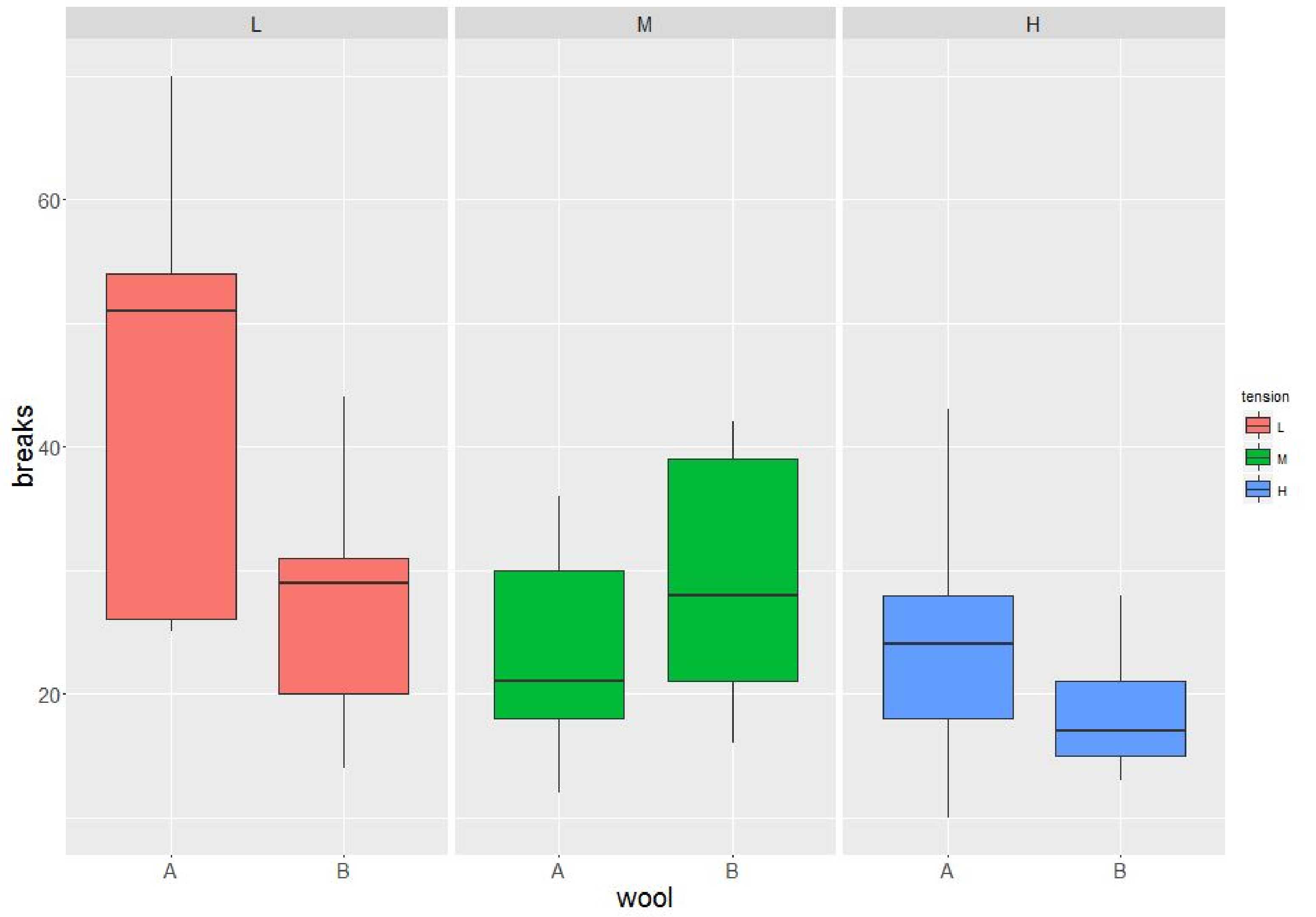
Are there differences in the mean number of breaks when different tensions are applied



```
> aov1 <- aov(breaks ~ tension, data = warpbreaks)
> summary(aov1)
> TukeyHSD(aov1))
```

Two-Way ANOVA

Does the type of wool affects the number of breaks when different tension is applied?



Two -way ANOVA testing on warpbreaks data

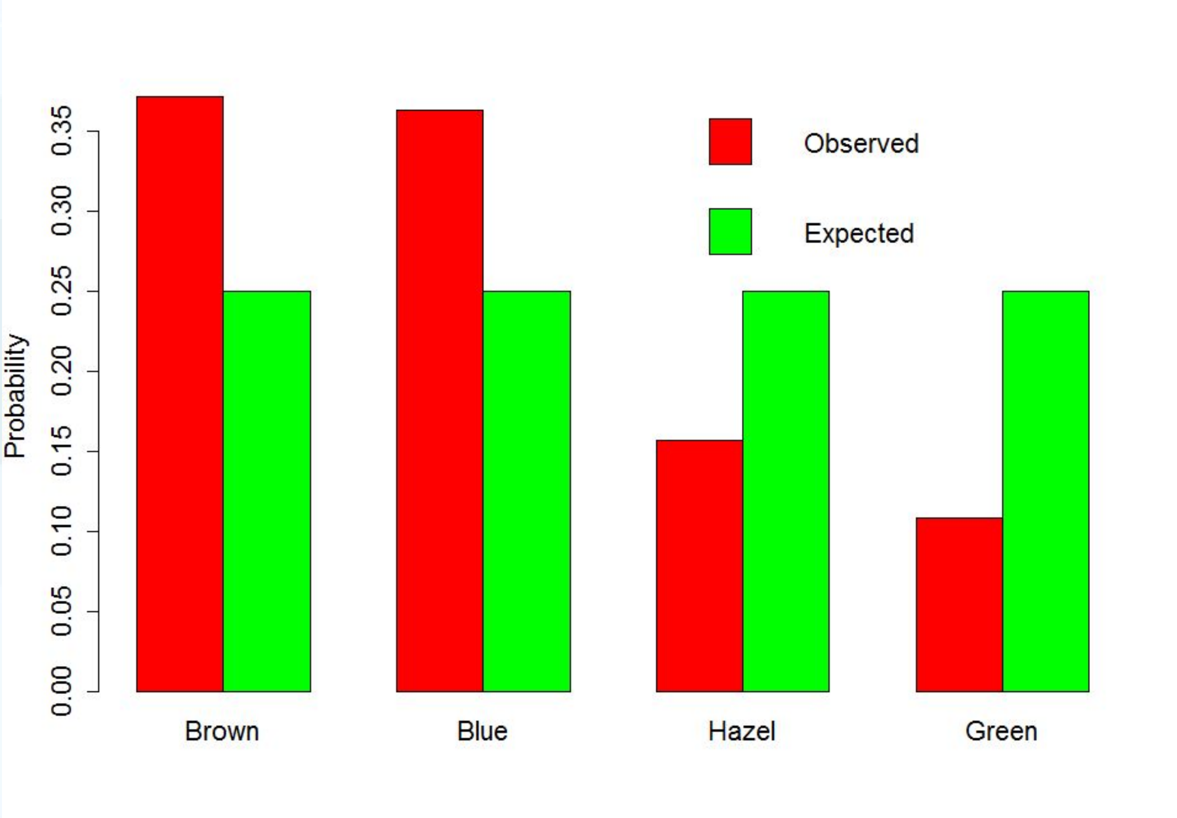
```
> aov2 <- aov(breaks ~ wool + tension + wool:tension, data =
warpbreaks)

> summary(aov2)

               Df   Sum Sq       Mean Sq      F value   Pr(>F)

wool           1    451          450.7        3.765     0.058213 .

tension        2    2034         1017.1       8.498     0.000693 ***

wool:tension  2    1003         501.4        4.189     0.021044 *

Residuals      48   5745   119.7



>TukeyHSD(aov2)

>tapply(warpbreaks$breaks, list(warpbreaks$tension, warpbreaks$wool),

mean)
```

Chi Square Test

Chi Square Test Application

A chi-squared test, also referred to as χ^2 test (or chi-square test), is used to compare observed frequencies with the frequencies expected under some null hypothesis

Chi Square Test



```
> eyeCol <- apply(HairEyeColor, 2, sum)
> chisq.test(eyeCol, p = c(0.25, 0.25, 0.25, 0.25))
```

Cluster Analysis

Cluster Analysis

- Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar to each other than to those in other groups (clusters)
- Cluster analysis is used in data mining, machine learning, pattern recognition, image analysis, information retrieval, and bioinformatics.

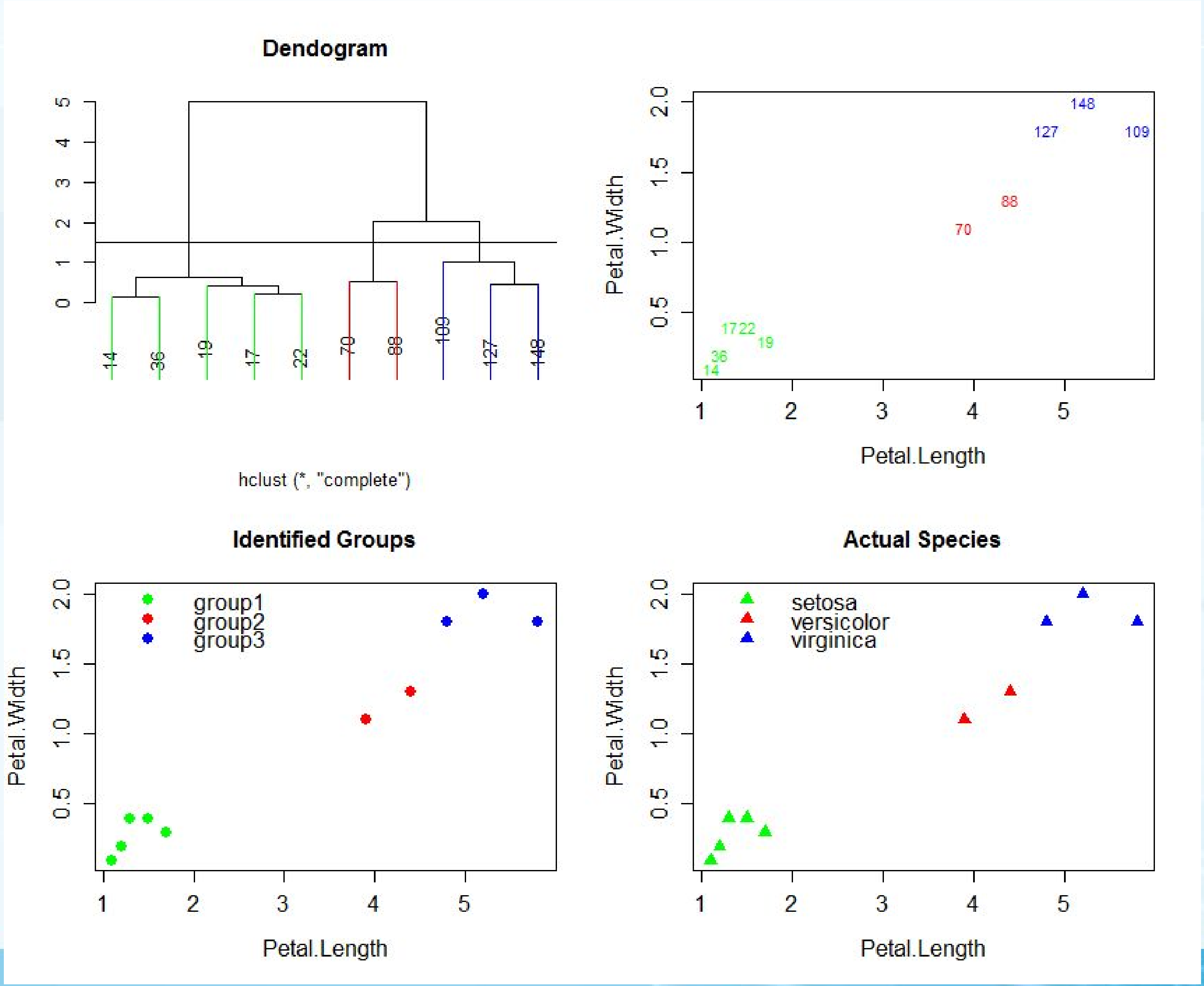
Clustering Algorithm

- Hierarchical clustering (connectivity model)
- K Means clustering (centroid model)

Hierarchical Clustering

- Hierarchical clustering connects "objects" to form "clusters" based on their distance.
- At different distances, different clusters will form, which can be represented using a dendrogram

Hierarchical Clustering illustration on a subset of Iris data

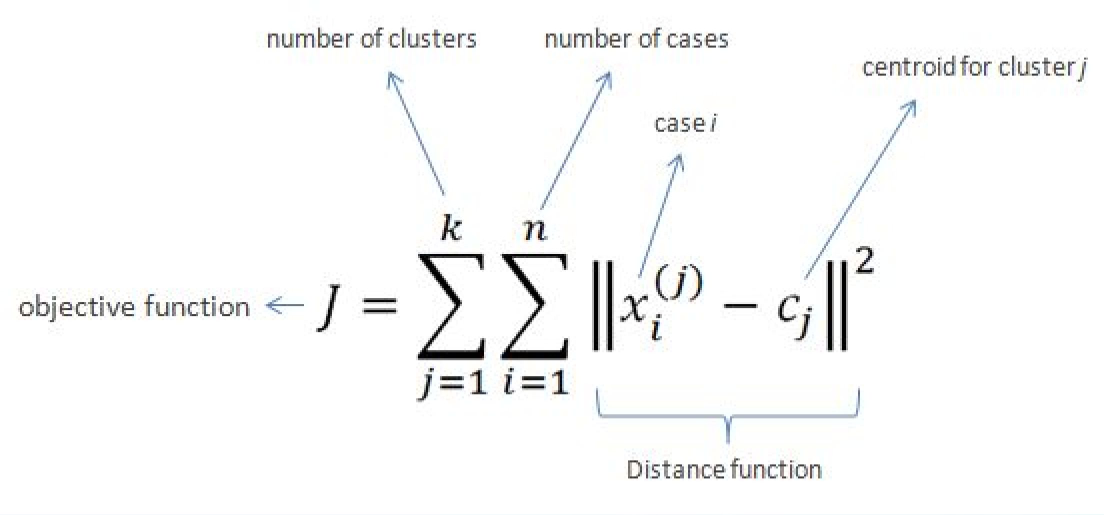


Hierarchical Clustering on a subset of iris dataset

```
### data preprocessing for hierarchichal clustering.
> set.seed(4)  # For reproducibility in subsetting
> index <- sample(c(TRUE, FALSE), nrow(iris), p = c(0.05, 0.95),
replace = TRUE)
> myIris <- iris[index,3:4]  # Choose only 5% of observations

### Clustering done on a subset of iris data which is  named myIris
> disM <- dist(myIris)   # calculate distance Matrix
> irisClust <- hclust(disM)  # do hierarchical clustering
> clusters <- cutree(irisClust, k = 3)  # make three clusters
> clusters   # output the cluster  associated with each observation
```

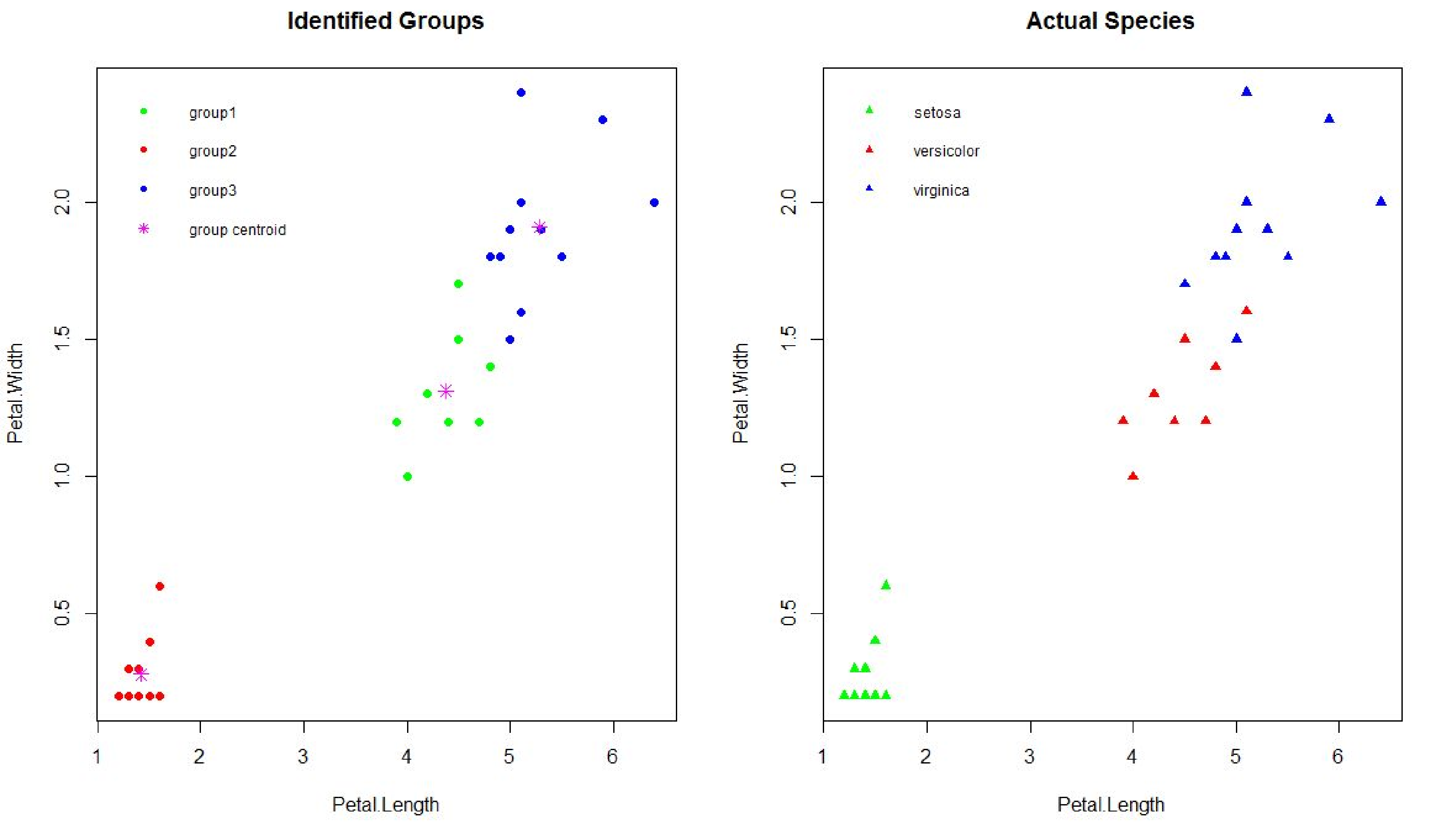
K- Means Clustering

- Place K points into the space represented by the objects that are being clustered. These points represent initial group centroids.
- Assign each object to the group that has the closest centroid.
- When all objects have been assigned, recalculate the positions of the K centroids.
- Repeat Steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized can be calculated.



$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \underbrace{\left\| x_i^{(j)} - c_j \right\|^2}_{\text{Distance function}}$$

K- Means Clustering illustration on subset of iris data set



K- Means Clustering on a subset of iris dataset

```
#Data Preprocessing
> set.seed(100)
> index <- sample(c(TRUE, FALSE), nrow(iris), p = c(0.2, 0.8), replace = TRUE)
> myIris <- iris[index,3:4]
> group <- iris$Species[index]

# kmeans clustering on a subset of iris data set called myIris
> set.seed(100)
> predGroup <- kmeans(myIris, centers = 3, nstart = 10)
> predGroupC <- ifelse(predGroup$cluster==1, "setosa", ifelse (predGroup$cluster==2,  "versicolor", "virginnica"))
> predGroupC <- factor(predGroupC)
> table(predGroupC, group)
```

Resources

- R Packages : https://cran.r-project.org/web/views/
- R Packages http://cran.stat.ucla.edu/web/packages/available_packages_by_name.html
- R Packages http://crantastic.org/

Alfred Ang
96983731
angch@tertiaryinfotech.com