Evolutionary Genetics

LV 25600-01 | Lecture with exercises | 4KP

HS2020

Bioinformatics - R

Robert Gentleman Ross Ihaka

1990 Start: Ross Ihaka and Robert Gentleman at the University of Auckland

↪ hence “Kiwi Creation”

1992 S language syntax, name “R” was chosen

1994 Initial version is complete

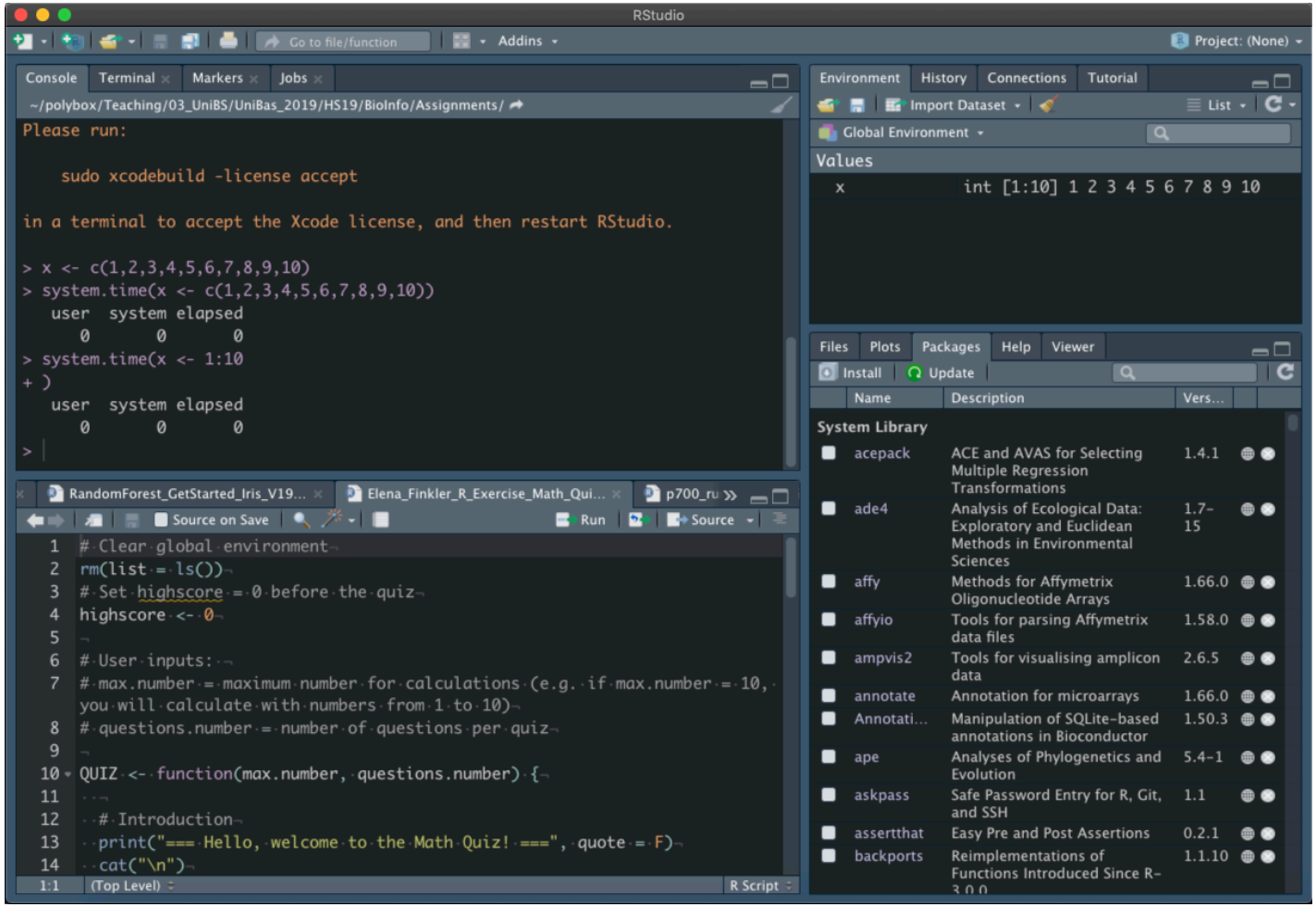
1997 CRAN archive is established by Kurt Hornik and Fritz Leisch

Open development process -> core developer

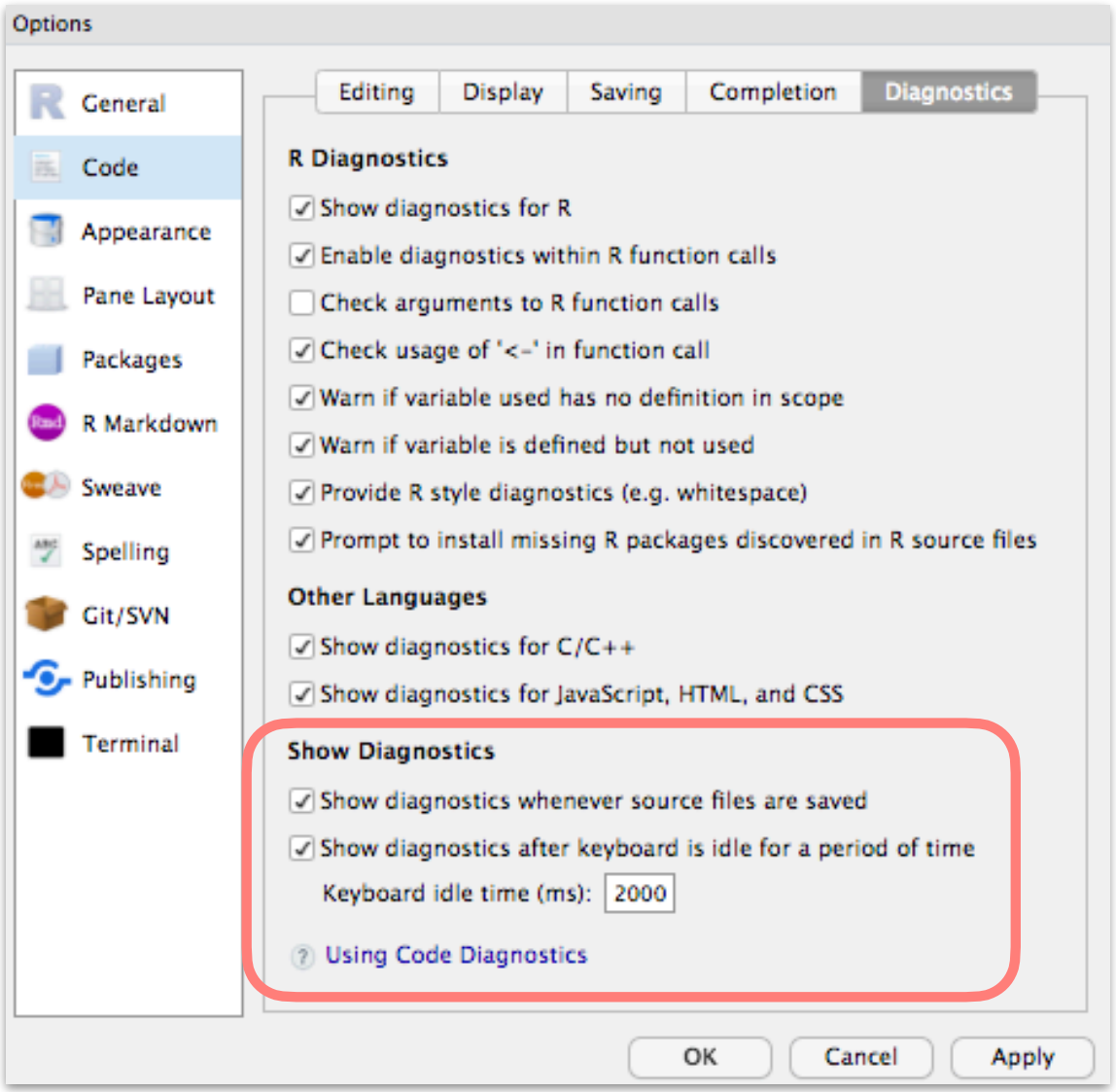
2000 First stable release (Version 1.0.0)

2019 R version 3.6

Bioinformatics - R



Bioinformatics - R



Example:
RStudio Settings

Bioinformatics - R

https://rstudio.com

https://jupyter.org/

https://rkward.kde.org

https://www.getarchitect.io

https://atom.io/packages/**rbox**

HELP

Bioinformatics - R

R-Project: http://www.r-project.org/

R-FAQ: http://www.ci.tuwien.ac.at/~hornik/R/R-FAQ.html

R-help: https://stat.ethz.ch/mailman/listinfo/r-help

Quick-R: http://www.statmethods.net/interface/help.html

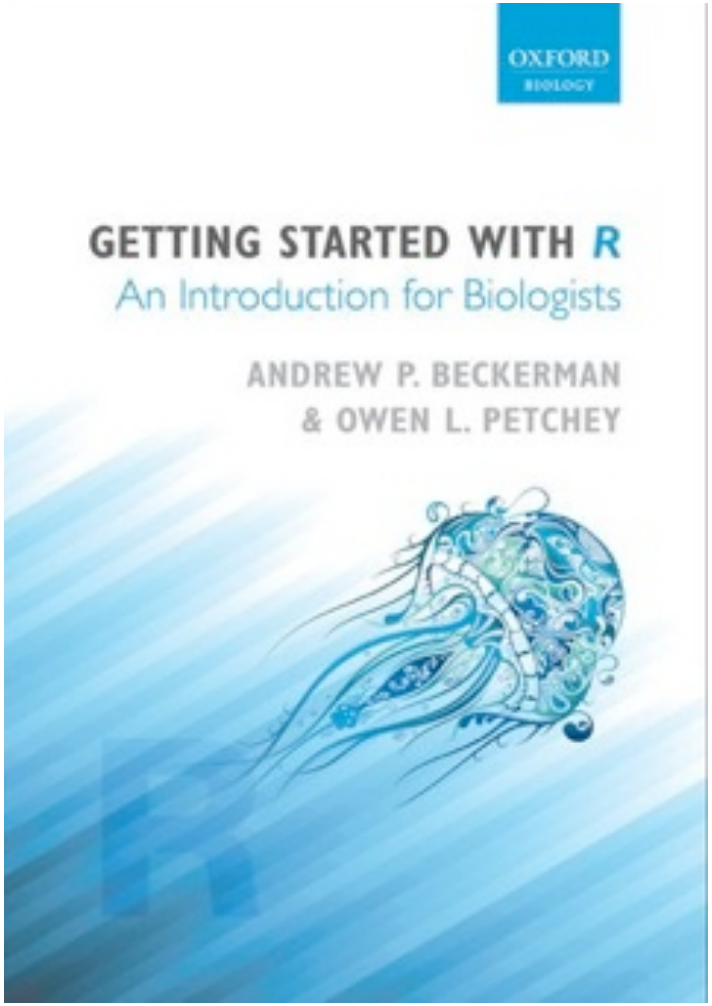
RSEEK: http://rseek.org/

WIKIBOOKS: http://en.wikibooks.org

Bioinformatics - R

Command in R	Result
`help(t.test)` or `?t.test`	These functions provide access to documentation. In the example R will provide documentation for the function t.test
`help.search("anova")`	Searches the help system for documentation matching a given character string. Names and titles of the matched help entries are displayed. In this example, a list of functions that are related to anova will be returned.
`apropos("test")`	Provides a list of command names that contain the pattern in quotes. This example lists commands that contain the word "test".
`example(t.test)`	This initiates running an example, if available, of the use of the function specified by the argument function.
`help.start()`	Start the hypertext (currently HTML) version of R's online documentation.
`RSiteSearch("")`	Search for key words or phrases in the R-help mailing list archives, or R manuals and help pages, using the search engine at http://search.r-project.org and view them in a web browser.

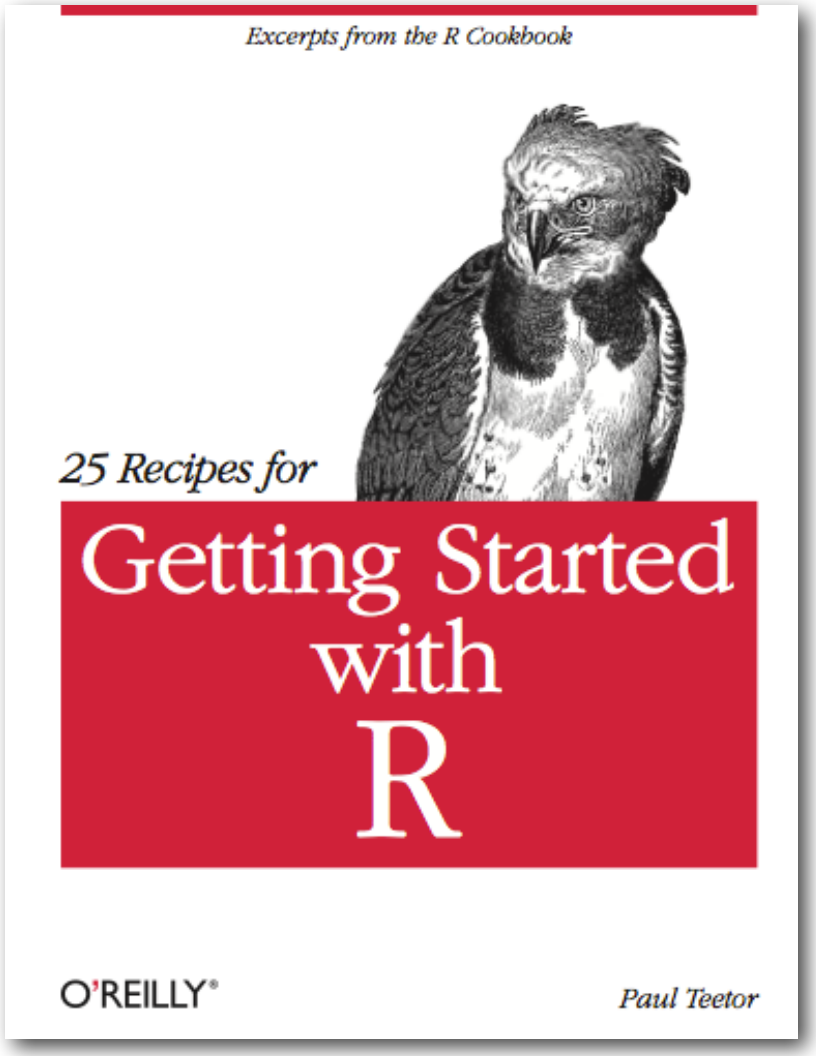
Bioinformatics - R



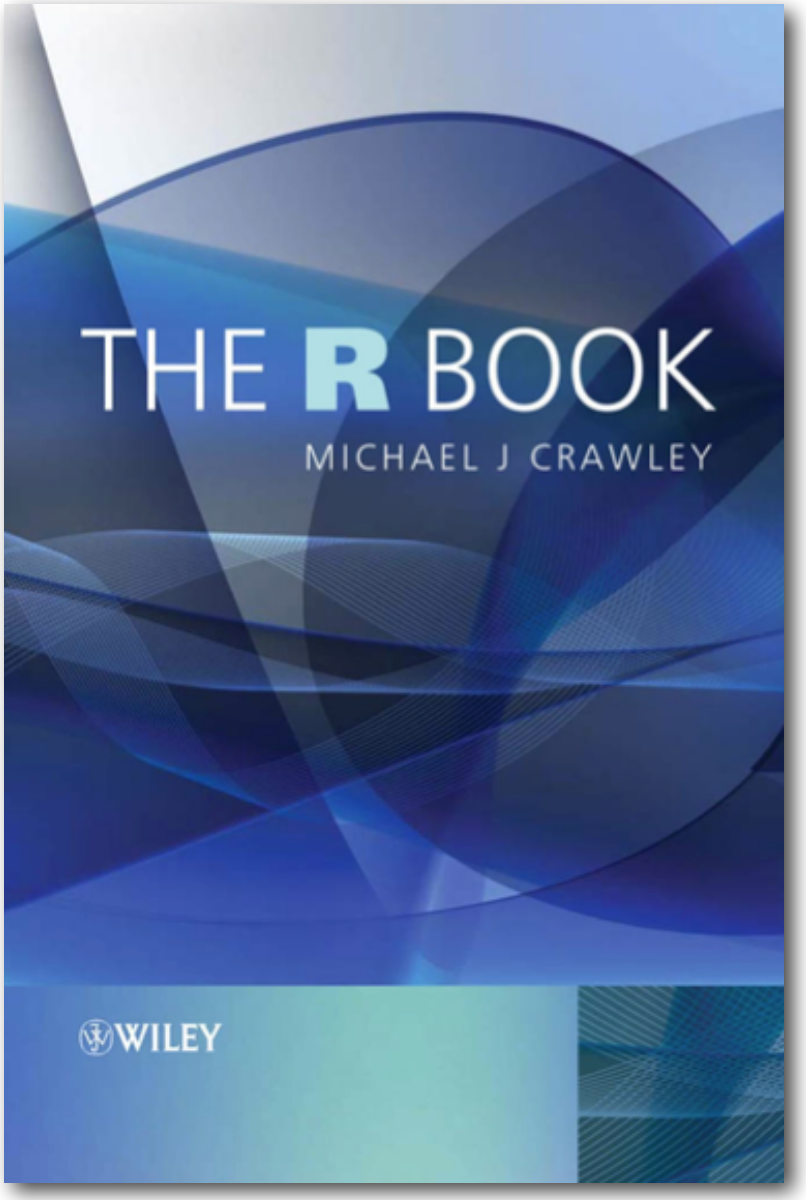
- A book containing a simple, easy, and engaging introduction to R for biologists
- Walks readers through the fundamentals of using R to import, manage and explore data, produce figures, and do basic statistics
- Designed for biologists new to R, who want an easy boost up the initially steep learning curve
- Focuses on producing tables and graphs that can be used in poster and oral presentations, and to put in manuscripts and theses
- Presents an efficient, accurate, reliable, and reproducible workflow for using R
- Based on a very successful three-day course
- Incorporates a range of biological examples

Publisher: Oxford University Press, USA (July 22, 2012)
ISBN-10: 0199601623
ISBN-13: 978-0199601622

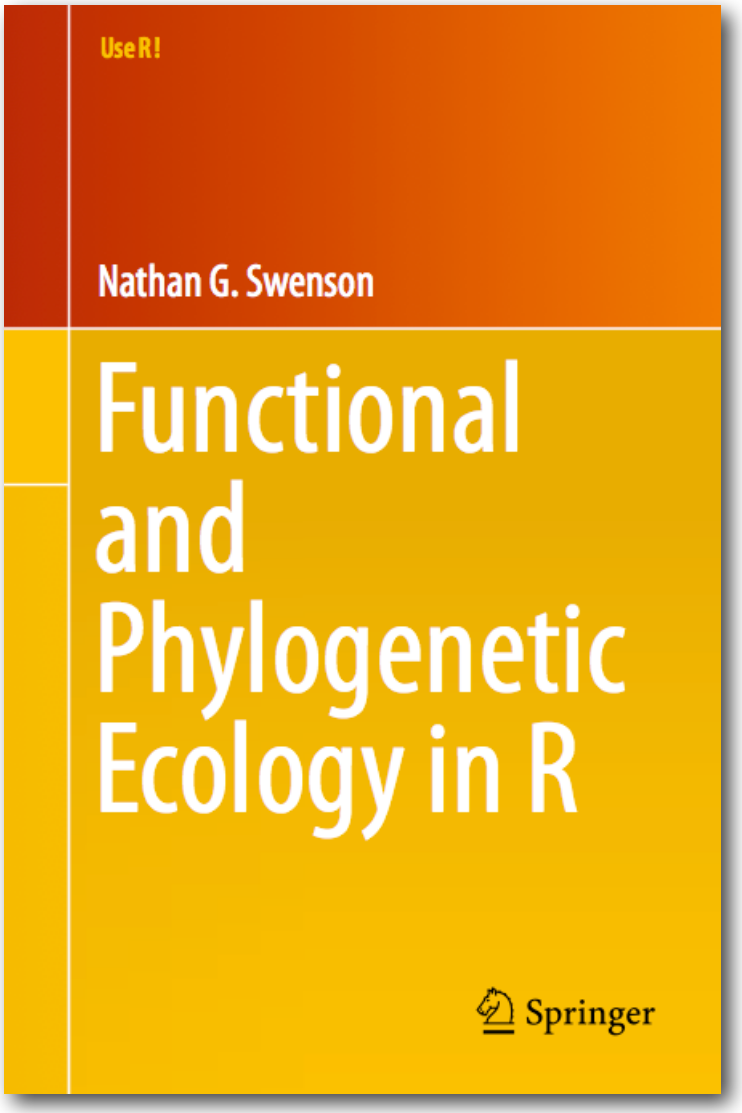
Bioinformatics - R



Bioinformatics - R



Bioinformatics - R



1. Introduction
2. Phylogenetic Data in R
3. PhylogeneticDiversity
4. FunctionalDiversity
5. Phylogenetic and Functional Beta Diversity
6. NullModels
7. Comparative Methods and Phylogenetic Signal
8. Partitioning the Phylogenetic, Functional, Environmental, and Spatial Components of Community Diversity
9. Integrating R with Other Phylogenetic and Functional
10. Trait Analytical Software

Bioinformatics - R

Learn R, in R.

swirl teaches you R programming and data science
interactively, at your own pace, and right in the R console!

Bioinformatics - R


```
> 3+4
[1] 7
> 2-3
[1] -1
> 2*3
[1] 6
> 3/2
[1] 1.5
> 2*3-2
[1] 4
> 2*(3-2)
[1] 2
> (2+5)/2
[1] 3.5
```

```
> sin(3)
[1] 0.14112
> 3^2
[1] 9
> sqrt(9)
[1] 3
> abs(-3)
[1] 3
> factorial(3)
[1] 6
> log10(10)
[1] 1
> pi
[1] 3.141593
```

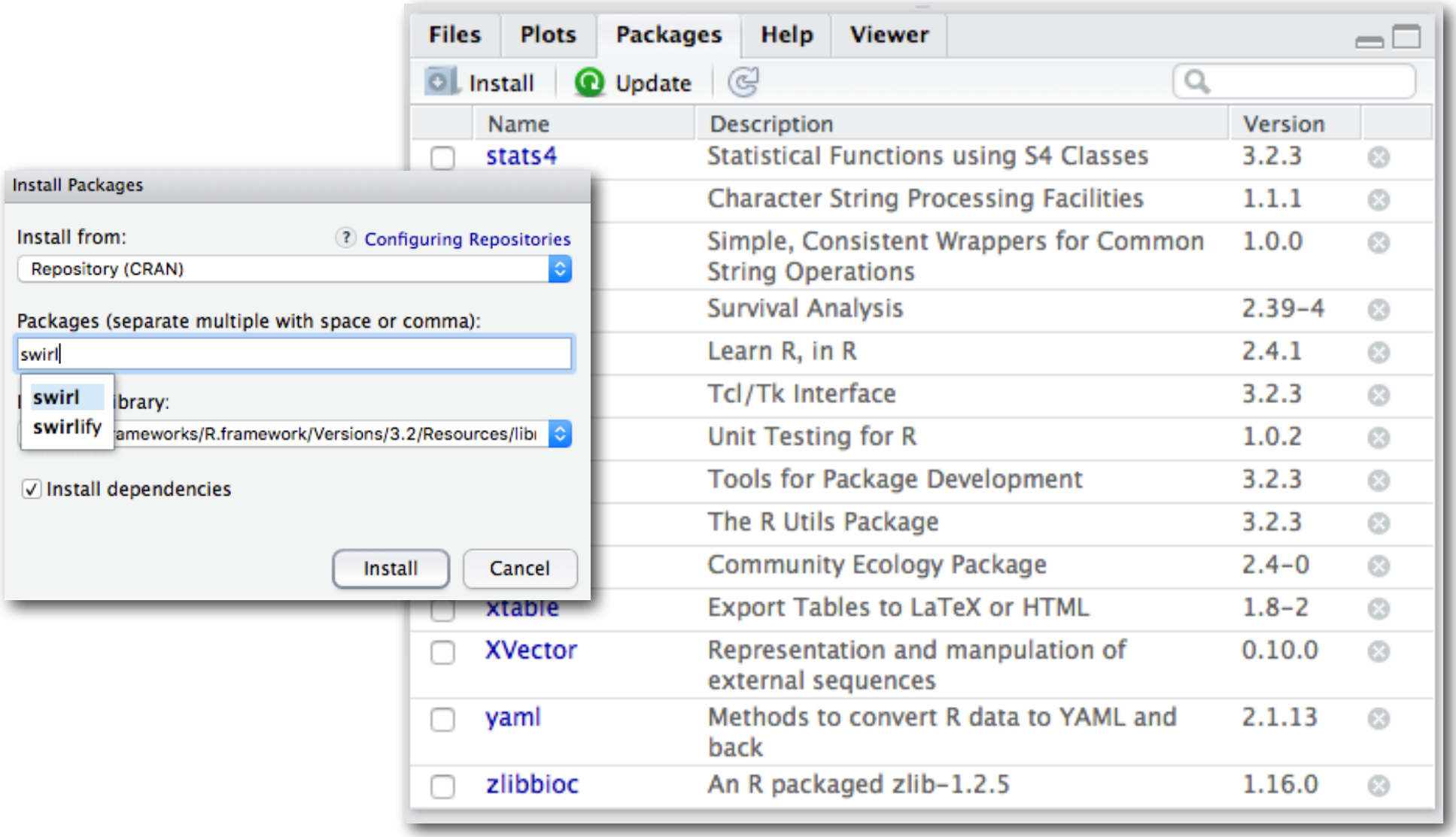
Bioinformatics - R

```
> install.packages("hangman"
> library("hangman")
> hangman()
[1] _ _ _ _ _ _ _ _ _

 Choose a letter:
1: A
Error in guess() : A is not a letter

 Choose a letter:
1: a
The word does not contain a

[1] _ _ _ _ _ _ _ _ _

 Choose a letter:
1: s
Correct!
```

Libraries / Packages

An R installation contains one or more libraries of packages. Some of these packages are part of the base installation. Others can be downloaded from CRAN, which hosts over 1000 packages for various purposes.

Bioinformatics - R



```
> install.packages("packagename", lib= "~/Rpack/")
> library(packagename, lib.loc="~/Rpack/")
```

Note: You might need to designate a directory where you will store the downloaded packages (e.g. /Rpack/).

```
> install.packages("seqinr")
> library("seqinr")
```

seqinr-package {seqinr} - Biological Sequences Retrieval and Analysis

Exploratory data analysis and data visualisation for biological sequence (DNA and protein) data. Include also utilities for sequence data management under the ACNUC system.

Bioinformatics - R

Bioconductor
OPEN SOURCE SOFTWARE FOR BIOINFORMATICS

Search:

Home | Install | Help | Developers | About

About Bioconductor

Bioconductor provides tools for the analysis and comprehension of high-throughput genomic data. Bioconductor uses the R statistical programming language, and is open source and open development. It has two releases each year, more than 400 packages, and an active user community.

Use Bioconductor for...

Microarrays
Import Affymetrix, Illumina, Nimblegen, Agilent, and other platforms. Peform quality assesement, normalization, differential expression, clustering, classification, gene set enrichment, genetical genomics and other workflows for expression, exon, copy number, SNP, methylation and other assays. Access GEO, ArrayExpress, Biomart, UCSC, and other community resources.

High Throughput Assays
Import, transform, edit, analyze and visualize flow cytometric, mass spec, HTqPCR, cell-based, and other assays.

Sequence Data
Import fasta, fastq, ELAND, MAQ, BWA, Bowtie, BAM, gff, bed, wig, and other sequence formats. Trim, transform, align, and manipulate sequences. Perform quality assessment, ChIP-seq, differential expression, RNA-seq, and other workflows. Access the Sequence Read Archive.

Annotation
Use microarray probe, gene, pathway, gene ontology, homology and othe annotations. Access GO, KEGG, N Biomart, UCSC, vendor, and othe sources.

Bioinformatics - R

```
## Note: try http:// if secure (https://) URLs are not supported
source("https://bioconductor.org/biocLite.R")
biocLite()
biocLite("Package")
```

Install Bioconductor

```
source("http://bioconductor.org/biocLite.R")
biocLite("BiocInstaller")
```

Install a Bioconductor package (e.g. muscle)

```
source("http://bioconductor.org/biocLite.R")
biocLite("muscle")
library("muscle")
```

More Sources for R packages

```
> # install.packages("devtools")
```

```
> library(devtools)
> install_github("hangman", "trinker")
```

Variable

```
v <- 2 # (alternatively a=2)
v <- "AAA"
```

Scalar (variable with one numeric element)

```
s <- 13
pi
```

Vector (homogeneous)

```
v <- c(1,2,3) # c for concatenate
```

List (heterogeneous)

```
l <- c(1,"A") #or list(a="1",name="A")
```

Array / Matrix (vector with dimensions)

```
D <- array(c(1,2,3,4,5,6), dim=c(2,3))
dim(D)
```

Variable names - You can use simple variable names like x, y, A, and a (note that A and a are different variable names). You can also use longer more **descriptive names** (good idea!). A variable name may contain digits, but it cannot begin with a digit! It may contain underscores (_) but not operators (e.g. <, =) punctuation (e.g. !, []) or the comment character (i.e. #). Be careful about "clobbering" built-in symbols with your own variable names (e.g. c for concatenate).

```
# Variable
v <- 2 # (alternatively a=2)
v <- "AAA"
```

What is the problem with the following R code snippet:

```
a <- 12
b <- 23
c <- 45
z <- (a * b) / c
```

What is the problem with the following R code snippet:

```
a <- 12
b <- 23
c <- 45
z <- (a * b) / c
Y <- c(a,b)
```

c {base} - combine values into a vector or list

What is the problem with the following R code snippet:

```
# A simple test to check
# for build-in functions
?c
?pi
?mean
```

Code Style for R

The tidyverse style guide

https://style.tidyverse.org

2.2.2 Parentheses

Do not put spaces inside or outside parentheses for regular function calls.

```
# Good
mean(x, na.rm = TRUE)

# Bad
mean (x, na.rm = TRUE)
mean( x, na.rm = TRUE )
```

Naming

File names and identifiers should be meaningful but short (<80).

```
avg.length <- mean(data$length)
x <- mean(data$length)
```

Comments

Comment your code!
Entire commented lines should begin with # and one space.
Short comments can be placed after code preceded by two spaces, #, and then one space.

Spacing

```
tab.prior <- table(df[df$days.from.opt < 0, "campaign.id"])
total <- sum(x[, 1])

tab.prior <- table(df[df$days.from.opt<0,"campaign.id"])
total <- sum(x[,1])
```

Global Options(e.g. digits):

```
> pi
[1] 3.141593
> getOption("digits")
[1] 7
> print(pi, digits=15)
[1] 3.14159265358979
> options(digits=3)
> pi
[1] 3.14
> options(digits=7)
```

{options} allows user to set and examine a variety of global *options* which affect the way in which R computes and displays its results.

Similar but not the same

```
> x <- c(1,2,3,4,5,6,7,8,9,10)
> x <- 1:10
> x <- seq(1, 10, by=1)
> x <- seq(length=10, from=1, by=1)
> assign("x", 1:10)
```

Profiling in R

```
x <- matrix( rnorm( 50000 * 900),
        nrow = 50000,
        ncol =   900)

system.time(t(x) %*% x)

system.time(crossprod(x))
```

```
> x <- c(1,3,5)
> y <- x * 2
> y
[1]  2  6 10
> x + y
[1]  3  9 15
> z <- c(1,3)
> x * z
[1] 1 9 5
Warning message:
In x * z : longer object length is not a multiple
of shorter object length
```

```
> x <- c(1,3,5)
> y <- x * 2
> x * y
```

$$\begin{matrix}1\\3\\5\end{matrix} \times 2 = \begin{matrix}2\\6\\10\end{matrix}$$

```
> x <- c(1,3,5)
> z <- c(1,3)
> x * z
```

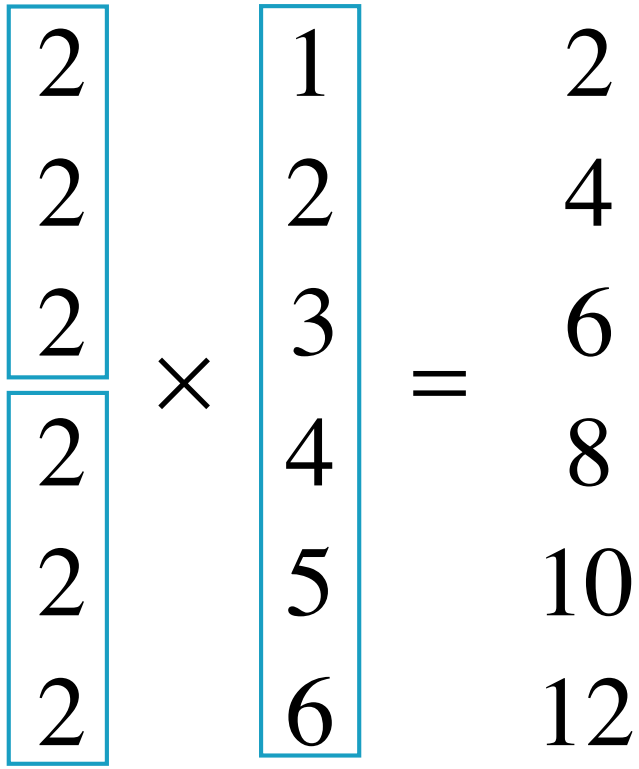
$$\begin{matrix}1\\3\\5\end{matrix} \times \begin{matrix}1\\3\\1\end{matrix} = \begin{matrix}1\\9\\5\end{matrix}$$

```
> x <- rep(2,3)
> z <- c(1:6)
> x * z
```

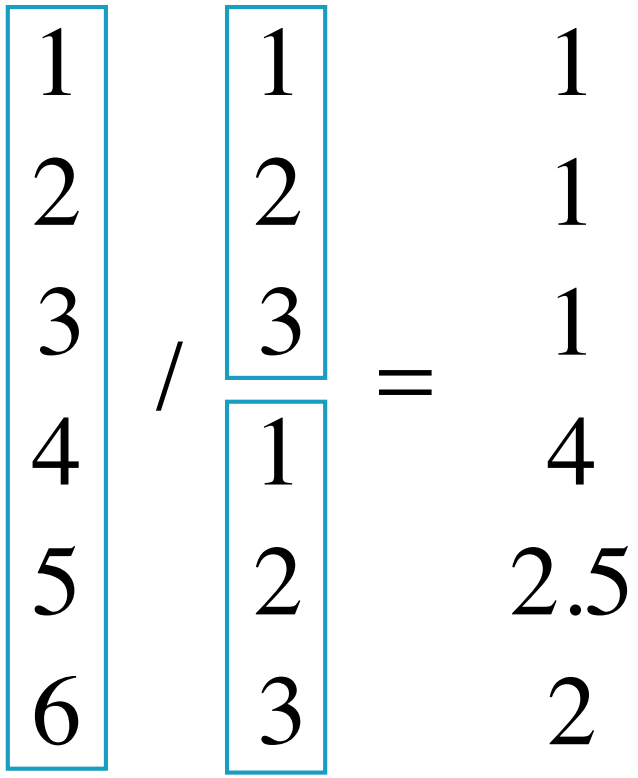
```
> x <- seq(1,3,1)
> z <- c(1:6)
> z / x
```

$$\begin{bmatrix}2\\2\\2\\2\\2\\2\end{bmatrix} \times \begin{bmatrix}1\\2\\3\\4\\5\\6\end{bmatrix} = \begin{matrix}2\\4\\6\\8\\10\\12\end{matrix}$$

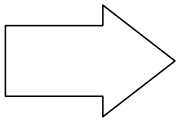
```
> x <- rep(2,3)
> z <- c(1:6)
> x * z
```




```
> x <- seq(1,3,1)
> z <- c(1:6)
> z / x
```

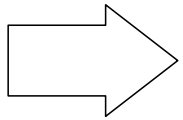


Bioinformatics - R

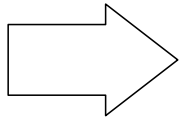


```
read.table("file.txt")
read.csv("file.csv")
read.delim("file.tab")
```

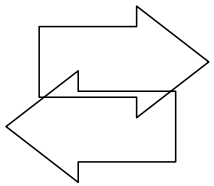
```
write(x,"file.txt")
write.csv(x,"file.csv")
write.table(x,"file.tab")
```



```
pdf()
dev.off()
```



```
save.image("Name.Rdata")
load("Name.Rdata")
```



Note: comma-separated values (CSV) file

Bioinformatics - R

```R
## Set working directory
setwd("my/working/directory") # change accordingly
list.files(getwd())

### Get data
url <- "https://gdc-web.ethz.ch/UniBas/data/egg_production_R.csv"
download.file(url, destfile = basename(url))

### Import data
egg <- read.table("egg_production_R.csv", sep = ",", header = TRUE)

### Remove NAs
egg.noNA <- egg[complete.cases(egg), ]

### Regular Boxplot
r.boxplot <- boxplot(egg.noNA$egg_production ~ egg.noNA$pop)

### Boxplot with ggplot
gg.boxplot <- ggplot(data = egg.noNA, aes(x = pop, y = egg_production))
gg.boxplot <- gg.boxplot + geom_boxplot(notch = FALSE)
gg.boxplot <- gg.boxplot + ylab("Number of Eggs")
gg.boxplot <- gg.boxplot + xlab("Origin")
gg.boxplot

### Save Session
save.image("tmp.Rdata")

## clean/reset environment
rm(list = ls())

## Load Session
load("tmp.Rdata")

## Primt Boxplot
r.boxplot
p.boxplot
```

Example:
Save and Restore
R Session

Sample	Replicate	Treatment	Run	S-level	N-level
A1	A	T1	r160815	3.4	1.1
A2	A	T1	r160815	3.5	0.9
B1	B	T1	r160815	6.4	0.5
B2	B	T1	r160815	6.2	0.9
C1	C	T1	r170815	6.4	0.4
C2	C	T1	r170815	6.2	0.8
D1	D	T1	r170815	7.7	1.5
D2	D	T1	r170815	9.0	0.9

```
A1 A T1 r160815 3.4 1.1
A2 A T1 r160815 3.5 0.9
B1 B T1 r160815 6.4 0.5
B2 B T1 r160815 6.2 0.9
C1 C T1 r170815 6.4 0.4
C2 C T1 r170815 6.2 0.8
D1 D T1 r170815 7.7 1.5
D2 D T1 r170815 9.0 0.9
```

text file with space deliminator and without header (title)

```
read.table("input_A.txt")

  V1 V2 V3       V4  V5  V6
1 A1  A T1 r160815 3.4 1.1
2 A2  A T1 r160815 3.5 0.9
3 B1  B T1 r160815 6.4 0.5
4 B2  B T1 r160815 6.2 0.9
5 C1  C T1 r170815 6.4 0.4
6 C2  C T1 r170815 6.2 0.8
7 D1  D T1 r170815 7.7 1.5
8 D2  D T1 r170815 9.0 0.9
```

By default, read.table is assuming white spaces between fields.

e.g. A B C D

```
R> read.table("file.txt")
```

What can you do if your input file uses a separator other than white space (e.g. A:B:C:D or A-B-C-D)

```
R> read.table("file.txt" ???)
```

Hint: You might find the answer with: ?read.table

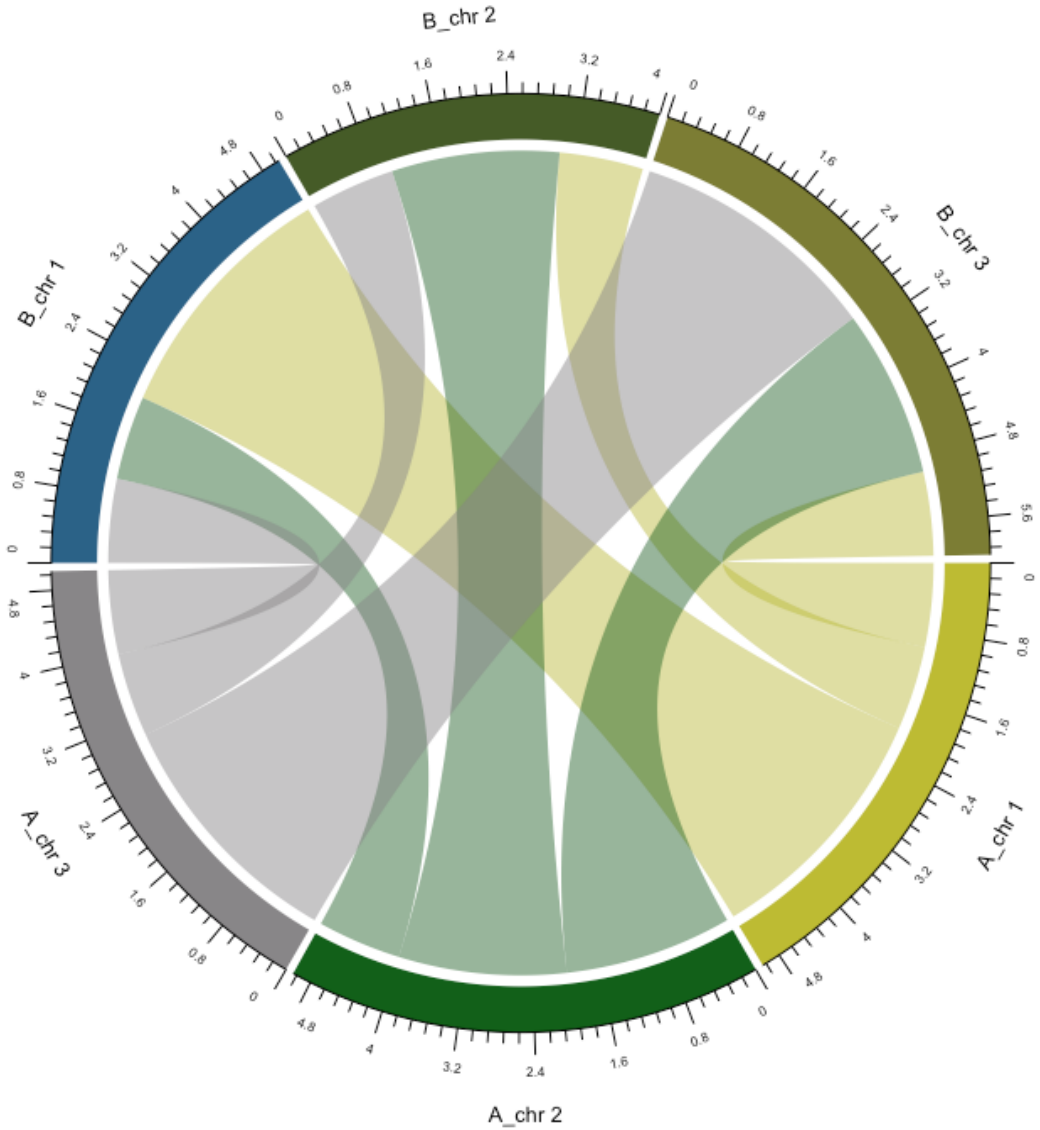
Use the **sep** parameter. For example, if our file used a colon (:) as the field separator, we would read your file this way:

```
> your.data <- read.table("file.txt", sep=":")
```

How can you tell read.table that it should use the first line (e.g. column names) for the table header?

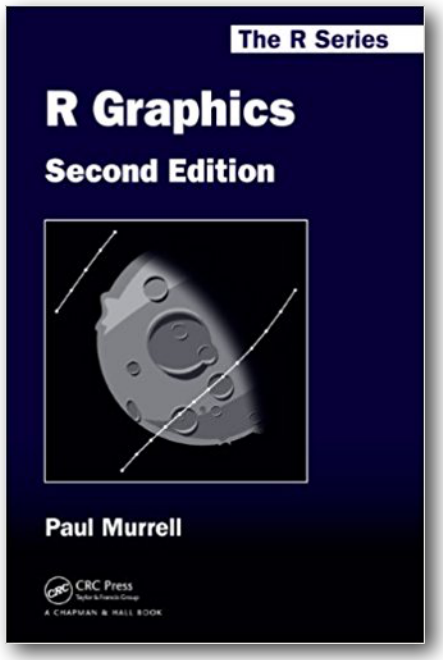
```
# Create data set without header
data <- read.table("file.txt", header=TRUE)
```

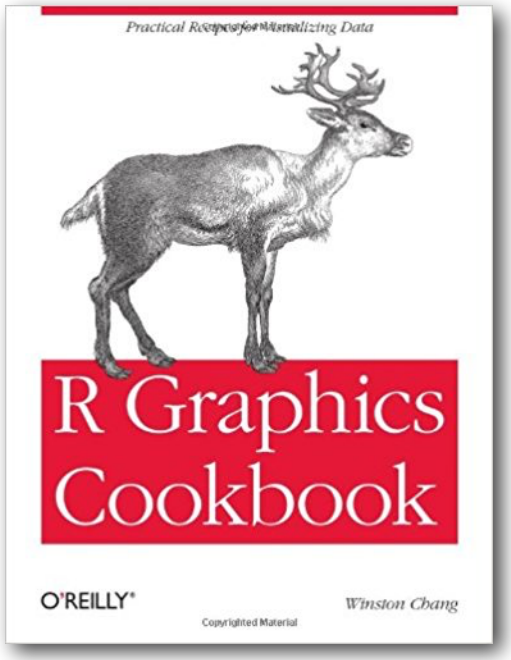
Bioinformatics - R

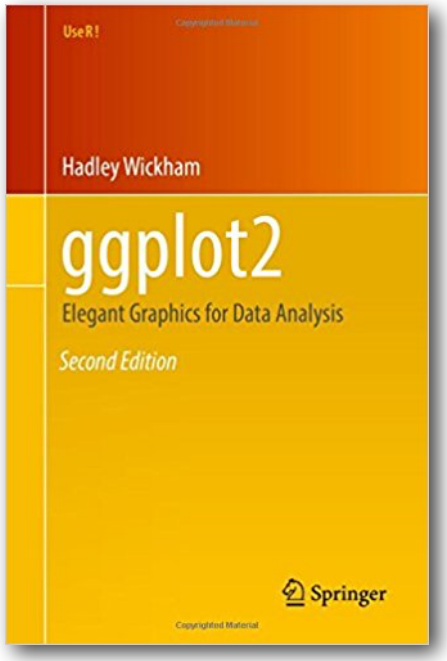


http://www.r-graph-gallery.com

Bioinformatics - R





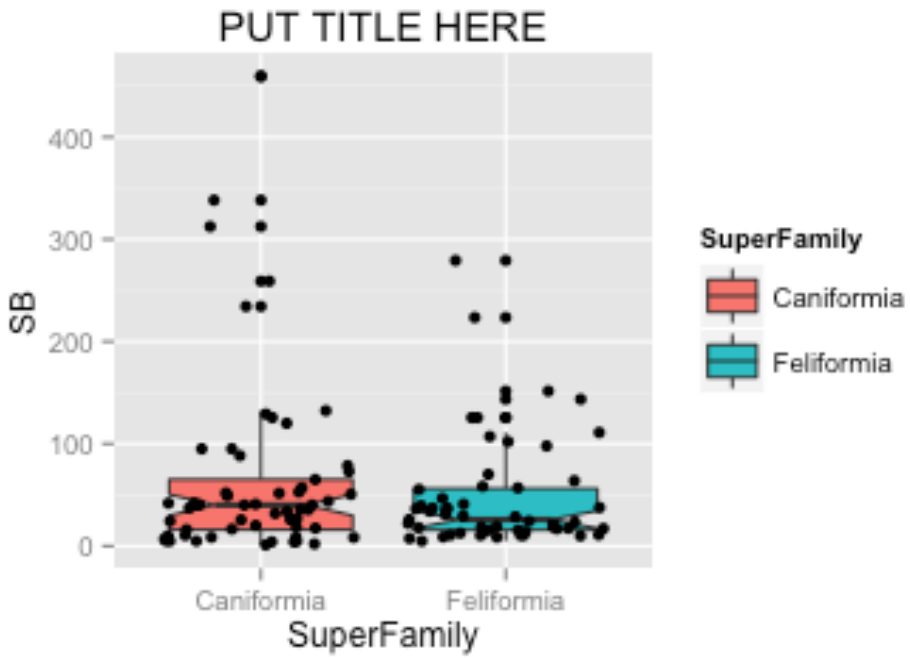


ggplot2 is a **plotting system for R**, based on the grammar of graphics, which tries to take the good parts of base and lattice graphics and none of the bad parts. It takes care of many of the fiddly details that make plotting a hassle (like drawing legends) as well as providing a powerful model of graphics that makes it easy to produce complex multi-layered graphics.

```
install.packages("ggplot2")
library(ggplot2)
p <- ggplot(data)
p <- p + geom_?
p
```

Documentation: http://docs.ggplot2.org/current/

Bioinformatics - R




```
install.packages("ggplot2")
library(ggplot2)
p <- ggplot(carnivora, aes(SuperFamily, SB))
p <- p + geom_boxplot(notch = TRUE,aes(fill = SuperFamily))
p <- p + geom_jitter()
p <- p + labs(title = "PUT TITLE HERE")
p
```

User Functions

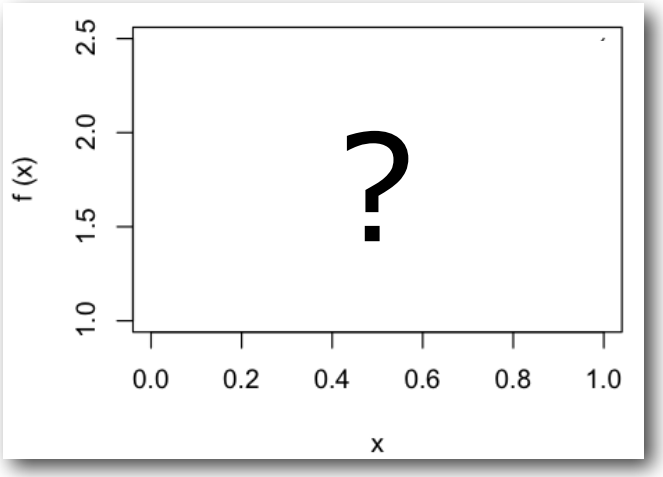
```
myfunction <- function(arg1, arg2, ... ){
   statements
   return(object)
}
```

Functions

$$y = x^3 + \frac{x}{2} + 1$$

$x = 2;\ y = ?$

$x = 10;\ y = ?$



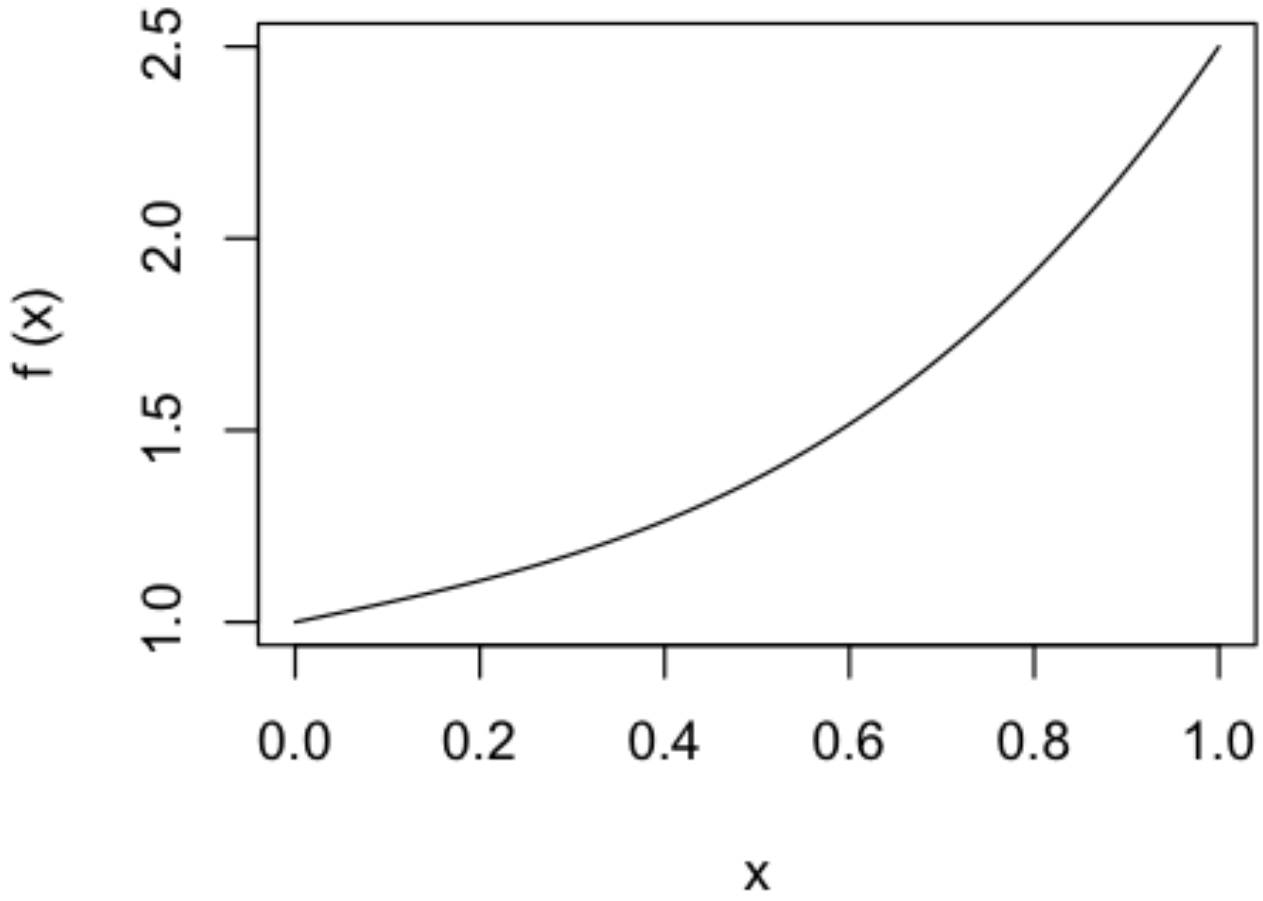
Bioinformatics - R

```
> x <- 2
> x^3 + x/2 + 1
[1] 10
> x <- 10
> x^3 + x/2 + 1
[1] 1006
```

or

```
> x <- c(2,10)
> x^3 + x/2 + 1
[1] 10 1006
```

```
> x <- seq(0,1,0.01)
> y <- (x^3 + x/2 + 1)
> plot(x, y, type="l")
```



Bioinformatics - R

```
> f <- function(x) {
 return( x^3 + x/2 + 1 )
 }
> f(2)
[1] 10
> plot(f)
```

Write a function to convert fahrenheit into celsius.

$$F = \frac{9}{5}C + 32$$

Function to convert fahrenheit into celsius.

```
> C2F <- function(celsius) {
 return( 9/5*celsius + 32 )
 }
> C2F(25)
[1] 77
```

Function to convert celsius into fahrenheit.

```
> F2C <- function(fahrenheit) {
 return( (fahrenheit-32)*5/9 )
 }
> F2C(77)
[1] 25
```

A function with options.

```
> require(stats)
> avme <- function(x, type) {
  switch(type,
         average = mean(x),
         median = median(x))
         }
> x <- 1:9
> avme(x, "average")
> avme(x, "median")
```

Functions

```
F2C <- function(fahrenheit) {
  return( (fahrenheit-32)*5/9 )
}
```

```
C2F <- function(celsius) {
  return( 9/5*celsius + 32 )
}
```

Idea

```
avme <- function(x, type) {
  switch(type,
         average = mean(x),
         median = median(x))
         }
```

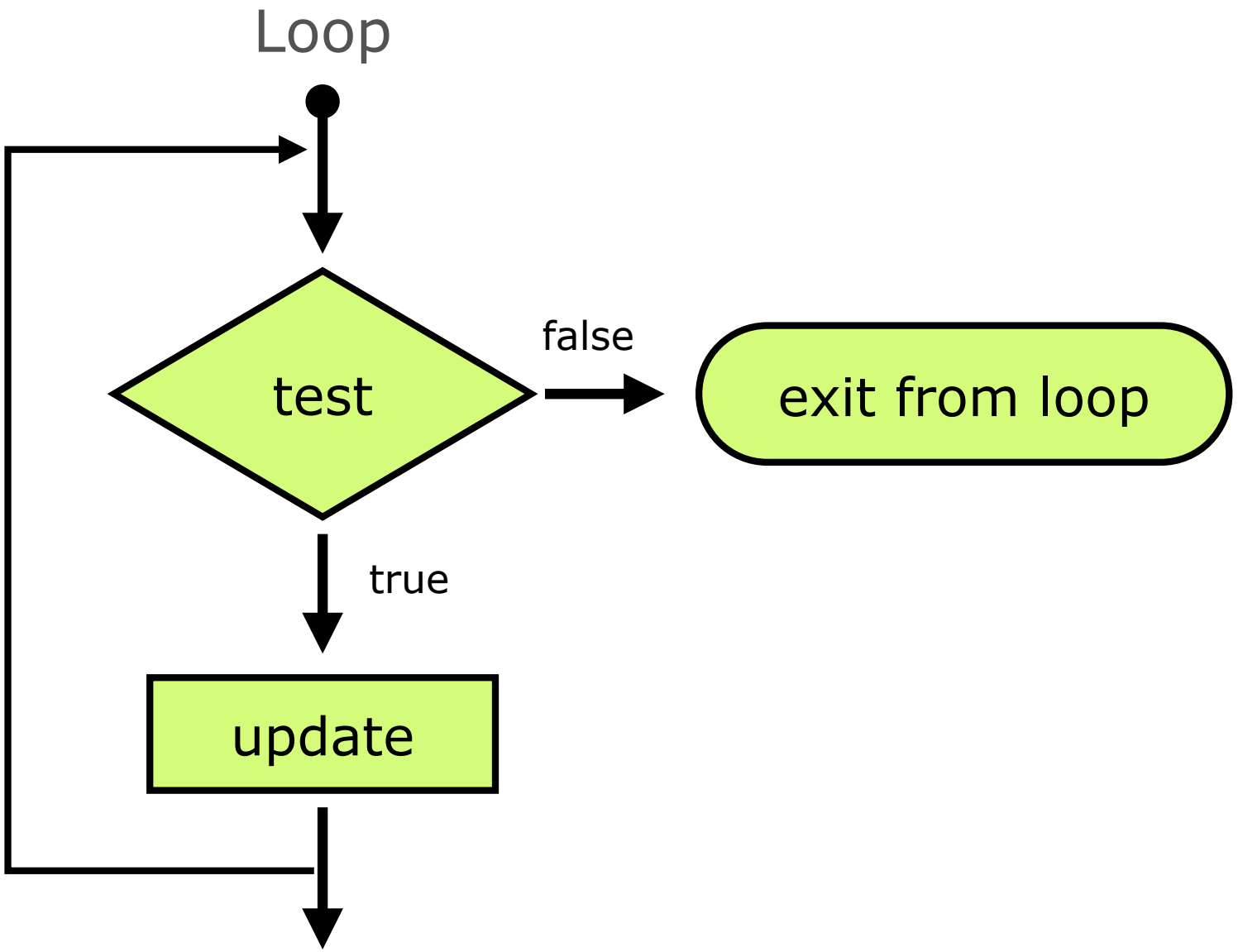
Combined Solution

Bioinformatics - R

```
> require(stats)
> hot <- function(x, type) {
  switch(type,
         f2c = ((x-32)*5/9),
         c2f = (9/5*x+32))
         }
> x <- 25
> hot(x, "f2c")
> hot(x, "c2f")
```

Default

```
> require(stats)
> hot <- function(x, type = "f2c") {
  switch(type,
         f2c = ((x-32)*5/9),
         c2f = (9/5*x+32))
         }
> x <- 25
> hot(x) # f2c default usage
> hot(x, "c2f")
```


Loop
test
false
exit from loop
true
update

Control Flow (looping)

```
for(var in seq) expr
while(cond) expr
```

```
> for(i in 1:3) print(i)
[1] 1
[2] 2
[3] 3
```

Control Flow (looping)

```
for(var in seq) expr
while(cond) expr
```

```
> y <- 0
> while(y < 3){ print( y <- y + 1 ) }
[1] 1
[2] 2
[3] 3
```

```
> for(i in 1:3) print(i)
[1] 1
[2] 2
[3] 3
```

```
> for(i in 1:3) write(i, "")
1
2
3
```

```
> for(i in 1:3) cat(i,"\n")
1
2
3
```

```
> for(i in 1:3) print(i)
1
2
3
```

```
> for(i in 1:3) print(i+1)
[1] 2
[2] 3
[3] 4
```

```
> WL <- c("Output","Some","Text")
> for(i in 1:3) print(WL[i])
[1] "Output"
[2] "Some"
[3] "Text"
```

```
> for(i in 1:3) write(i, "")
1
2
3
```

```
> for(i in 1:3) write(i+1, "")
[1] 2
[2] 3
[3] 4
```

```
> WL <- c("Output","Some","Text")
> for(i in 1:3) write(WL[i], "")
Output
Some
Text
```

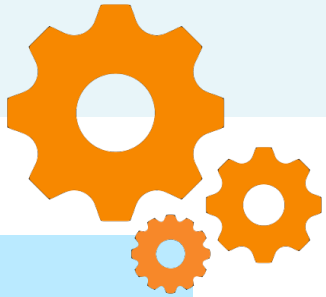
```
> for(i in 1:3) cat(i, "\n")
1
2
3
```

```
> for(i in 1:3) cat(i+1, " ")
2 3 4
```

```
> for(i in 1:3) cat("N=", i, ";", sep = "")
N=1;N=2;N=3
```

```
> WL <- c("Output","Some","Text")
> for(i in 1:3) cat(WL[i], "")
Output Some Text
```

```
> A <- 0
> for (j in c(10,20,30)) {A <- j+A}
> A
[1] ?
```



```
> A <- 0
> for (j in c(10,20,30)) {A <- j+A}
> A
[1] 60
```

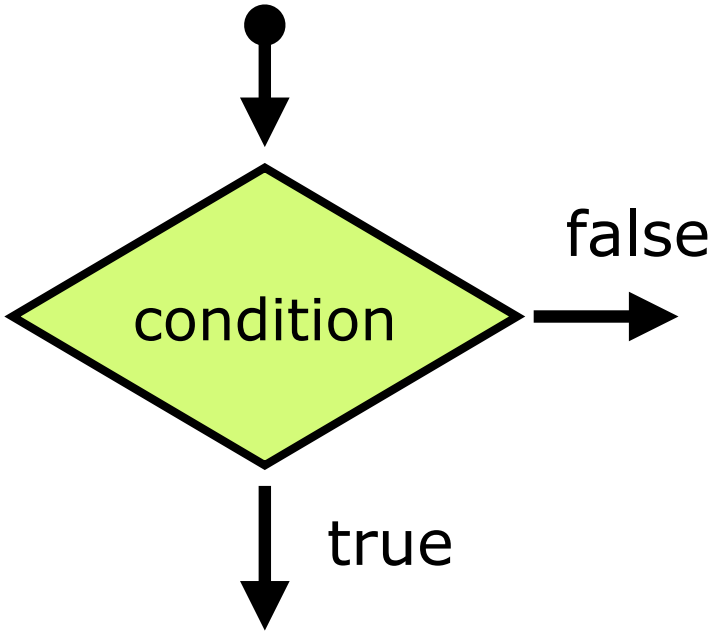
The calculation iteratively builds up the object A, using the successive values of j listed in the vector (10,20,30).

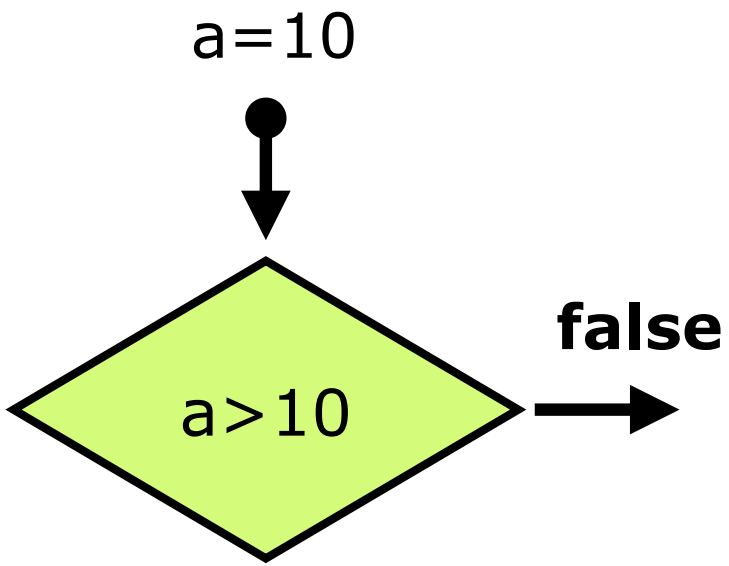
Initially, j=10, and **A** is assigned the value 10+0=10.

Then j=20, and **A** is assigned the value 20 + 10 = 30.

Finally, j=30, and **A** is assigned the value 30 + 30 = 60.

Control structure





```
> x <- 10 # let x be 10
> x <  10 # operation
FALSE
> x <- c(8:12)
> x < 10
TRUE TRUE FALSE FALSE FALSE
```

```
> a <- 17
> b <- 20
> a > b
[1] FALSE
> a == b
[1] FALSE
> a != b
[1] TRUE
> a <= b
[1] TRUE
```

Operators

equal	==
not equal	!=
bigger	>
less	<
bigger or equal	>=
less or equal	<=

Control structure (if/else)

```
if(cond) expr
if(cond) true.expr  else  false.expr
```

```
> a <- 2
> if(a < 3) a+1 else a-1
> a
[1] 3
> if(a < 3) a=a+1 else a=a-1
> a
[1] 2
```

```
> a <- c(2,4)
> if(a > 3) 5*pi else 2*pi
[1] 6.283185
Warning message:
In if (a > 3) 5 * pi else 2 * pi :
  the condition has length > 1 and only the first
element will be used
```

Bioinformatics - R

```
> x <- seq(-1,5)
> x
[1] -1 0 1 2 3 4 5
> x < 0
[1]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
> if(any(x < 0)) (x*2) else (x^2)
[1] -2  0  2  4  6  8 10
```

```
a <- c(2,4)
for(var in a) cat(var,"")
[1] 2 4
```

```
a <- c(2,4)
for(var in a)
    if(var > 3) print(5*pi) else print(2*pi)

[1] 6.283185
[1] 15.70796
```

```
> for(var in x)
  if(var > 3) cat(var,"> 3 TRUE\n") else
  cat(var,"< 3 FALSE\n")

-1 < 3 FALSE
0 < 3 FALSE
1 < 3 FALSE
2 < 3 FALSE
3 < 3 FALSE
4 > 3 TRUE
5 > 3 TRUE
```

```
> x <- seq(-1,5,1)
> if(any(x < 0)) cat("vector x contains negative
values\n") else cat("vector x contains NO negative
values\n")
```

vector x contains negative values

```
> y <- c(1:5,"Hello")
> if(any(y == "Hello")) cat("list contains Hello\n") else
cat("list contains NO \Hello\n")
```

list contains Hello

```
x <- seq(-1,5)
if(all(x != 6)) cat(“vector x does not contain 6”)
[1] vector x does not contain 6


if(all(x != 5)) cat(“vector x does not contain 5”)
? (there is no alternative output)

> if(all(x != 5)) cat("vector x does not contain 5") else
cat("vector x does contain 5”)
[1] vector x does contain 5
```

```
plus  <- "Input is > 0"
zero  <- "Input is = 0"
minus <- "Input is < 0"

cv <- function(x){
            if(x > 0) return(plus)
            if(x < 0) return(minus)
            else return(zero)
            }
```

```
> cv(5)
"Input is > 0"
```

Can you re-write the following function using the control structure if-else instead of the switch option?

```
hot <- function(x, type) {
  switch(type,
         f2c = ((x-32)*5/9),
         c2f = (9/5*x+32))
         }
```

```
hot2 <- function(x,y){
  if(y == "t2c") return((x-32)*5/9)
  if(y == "c2t") return(9/5*x+32)
  else return("sorry that does not work")
}
```

Control structure (**ifelse**)

```
> a = 5
> ifelse(a > 3, “Yes", “No")
[1] “Yes”

> ifelse(a > 3, a*5, a*10)
[1] 25
```

Control structure (**ifelse**)

```
> a = 5
> ifelse(a > 3, “Yes", “No")
[1] “Yes”

> ifelse(a > 3, a*5, a*10)
[1] 25

> a <- c(2,4)
> ifelse(a > 3, “Yes”, “No”)
[1] “No” “Yes”
```

interesting use case for ifelse

```
> x <- c(-2,2)
> sqrt(x)
NaN 1.414214
Warning message:
In sqrt(x) : NaNs produced

> sqrt(ifelse(x >= 0, x, NA))
NA 1.414214
```

Add up all the numbers from 1 to 100 **in a loop**.

Exercise: Add up all the numbers from 1 to 100 **in a loop**.

```
> A <- 0
> for (j in c(1:100)) {A <- j+A}
> A
[1] 5050
> sum(1:100) # double check the results
[1] 5050
```

Multiply all the numbers from 1 to 50 in a **loop.**

Exercise: Multiply all the numbers from 1 to 50 in a **loop.**

```
> A <- 1
> for (j in c(1:50)) {A <- j*A}
> [1] 3.041409e+64
> prod(1:50) # double check if it worked
[1] 3.041409e+64
```

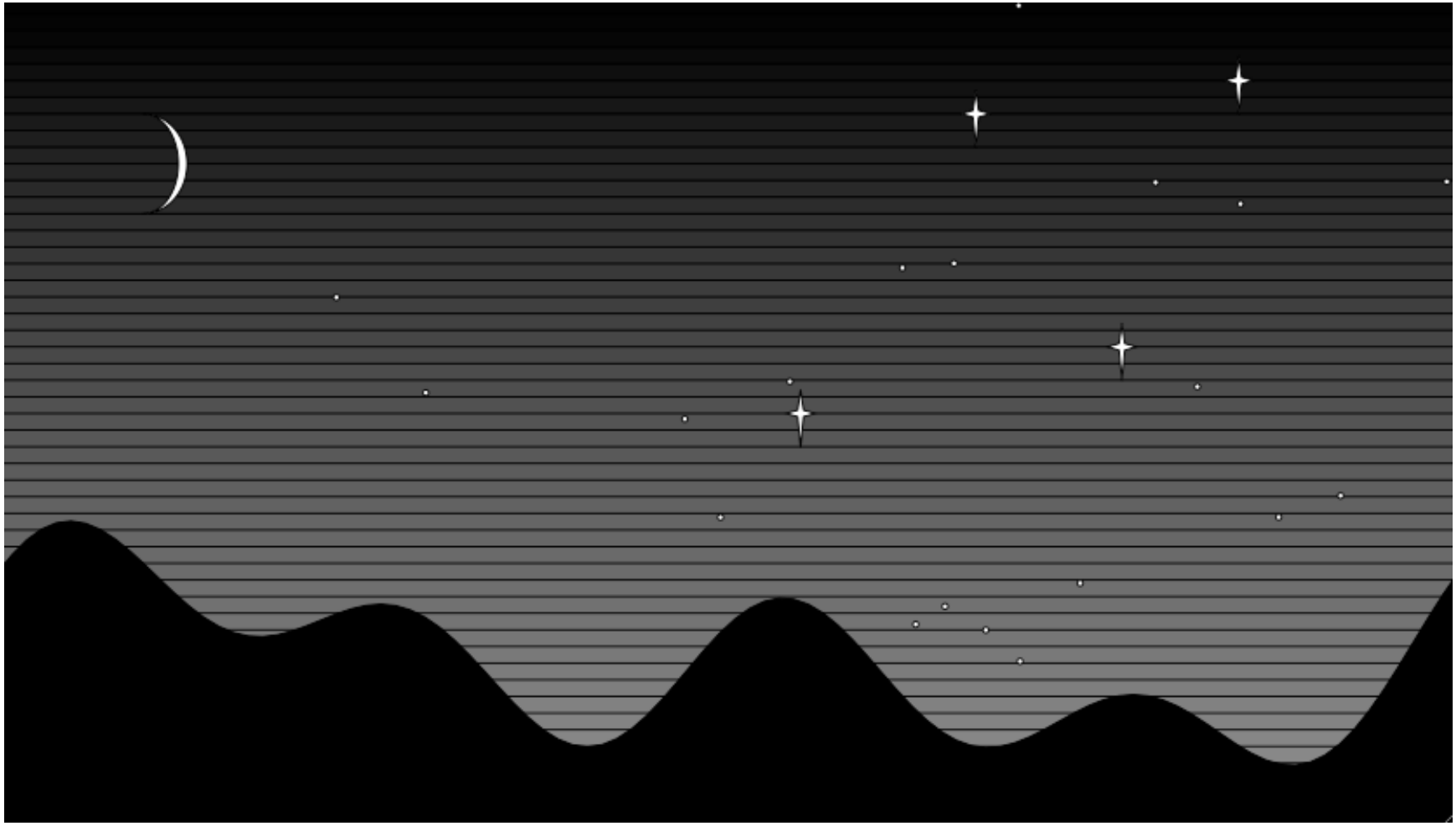
Create a loop that would convert a range of celsius values into fahrenheit.

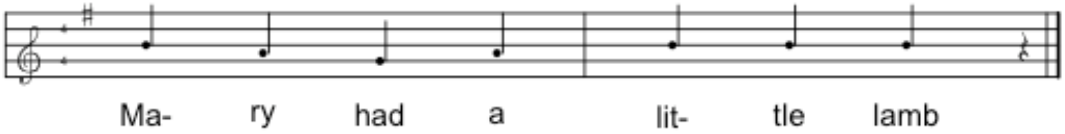
```
fahrenheit = 9/5*celsius + 32
```

Bioinformatics - R

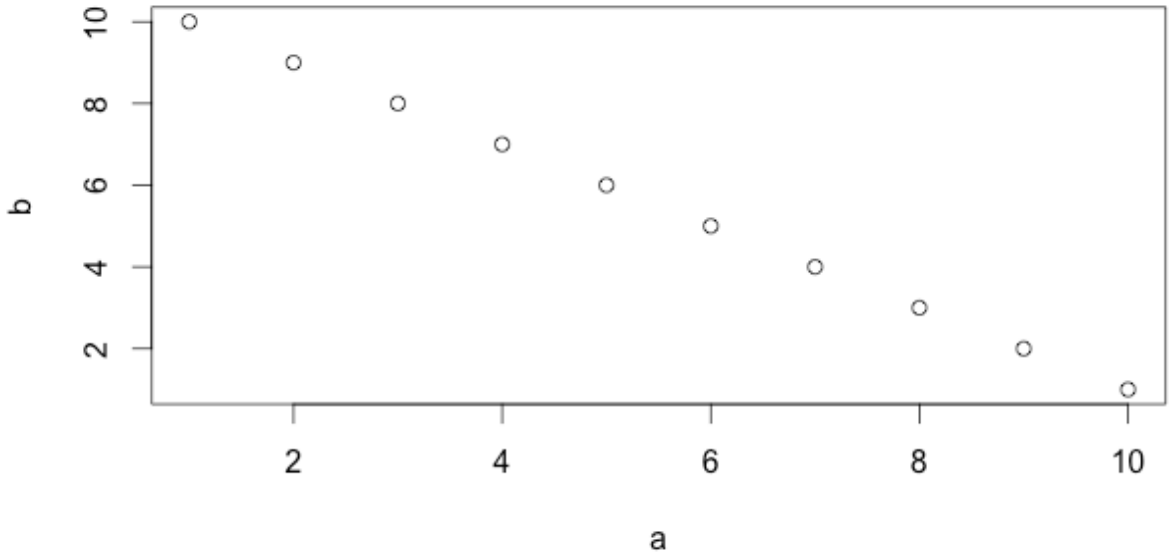
```
# Celsius to Fahrenheit
for (celsius in 25:30)
  print(c(celsius, ">" ,9/5*celsius + 32), digits=2, quote=F)

[1] 25 > 77
[1] 26 > 79
[1] 27 > 81
[1] 28 > 82
[1] 29 > 84
[1] 30 > 86
```


MORE
THINGS
CONSIDERED

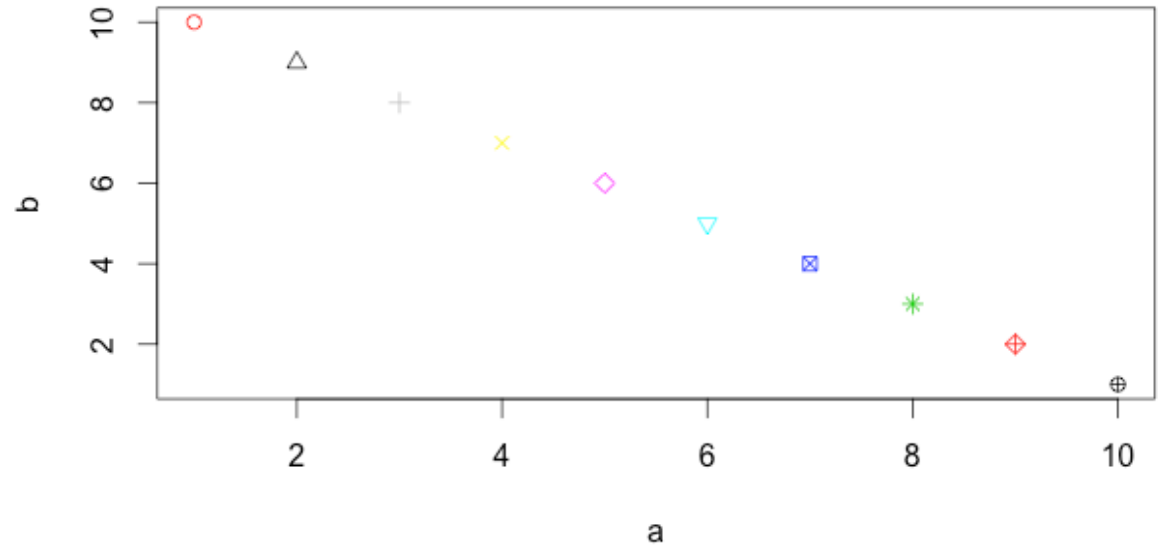



Ma- ry had a lit- tle lamb

```
> a <- 1:10
> b <- 10:1
> plot(a,b)
```



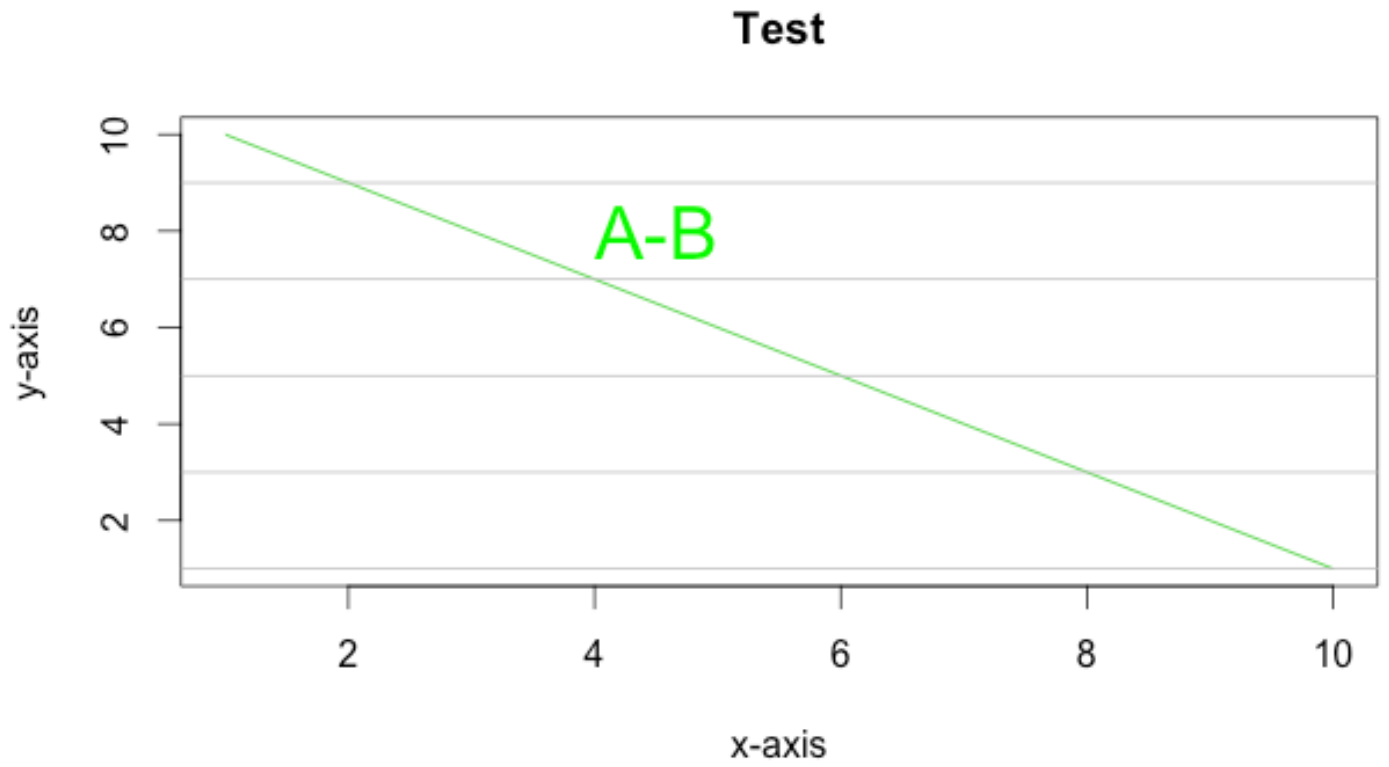

```
> a <- 1:10
> b <- 10:1
> plot(a,b,pch=a,col=b)
```

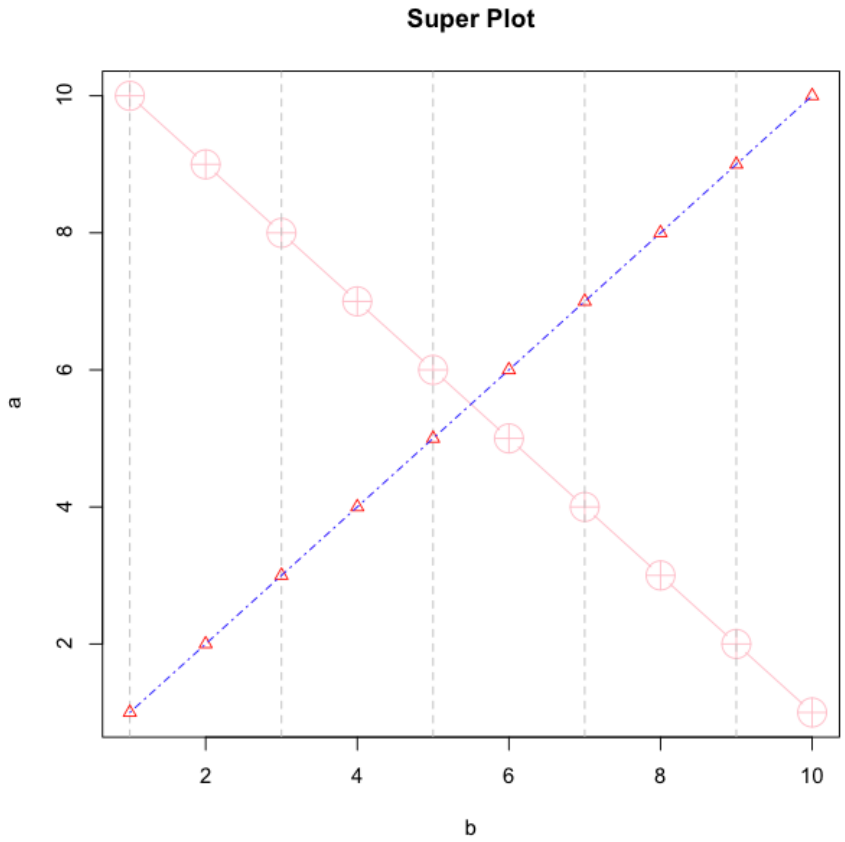


col=

pch=

```
> a <- 1:10
> b <- 10:1
> plot(a,b,type="l",col=3,main="Test",xlab="x-axis",ylab="y-axis")
> abline(h=(seq(1,10,2)),col="grey")
> text(4,8,"A-B",adj=c(0,NA),col="green",cex=2)
```

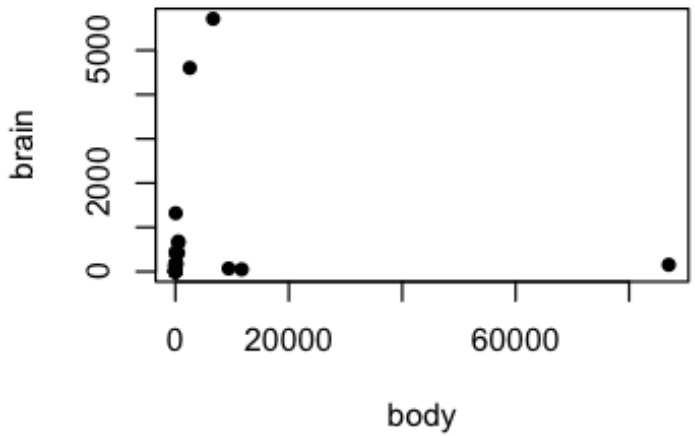


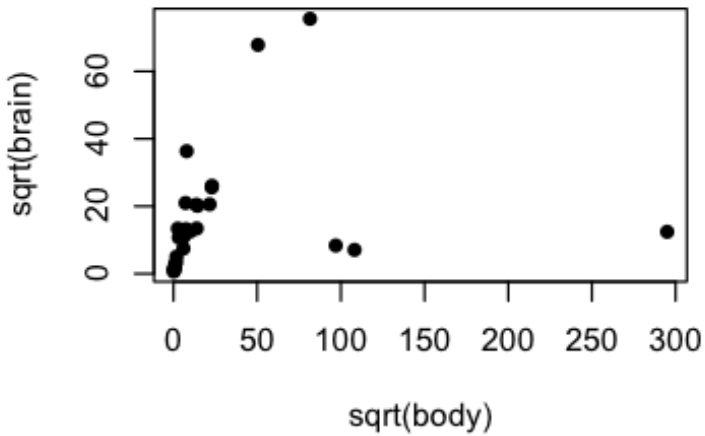


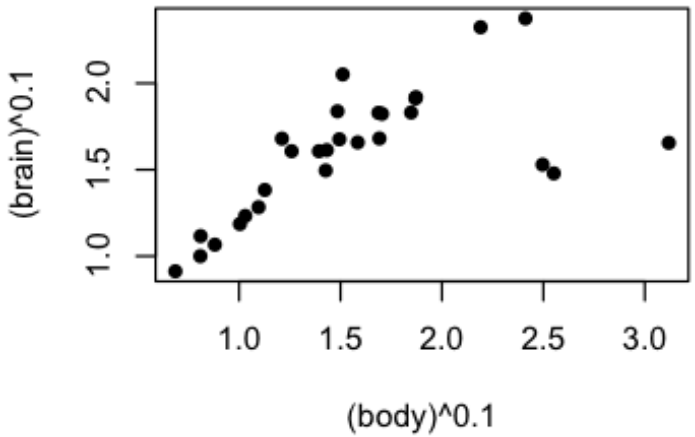

```
> a <- 1:10
> b <- 10:1
> plot(b,a,type="b",col="pink",cex=3,pch=10,main="Super Plot")
> abline(v=(seq(1,10,2)),col="grey")
> points(a,rev(b),pch=24,col="red")
```

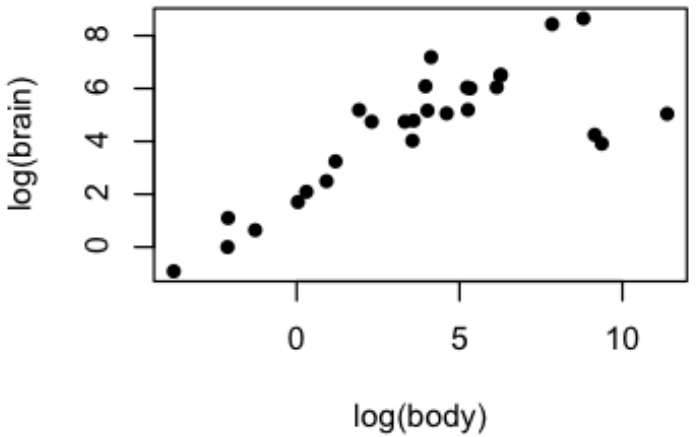
Animals	MASS	Brain and Body Weights for 28 Species

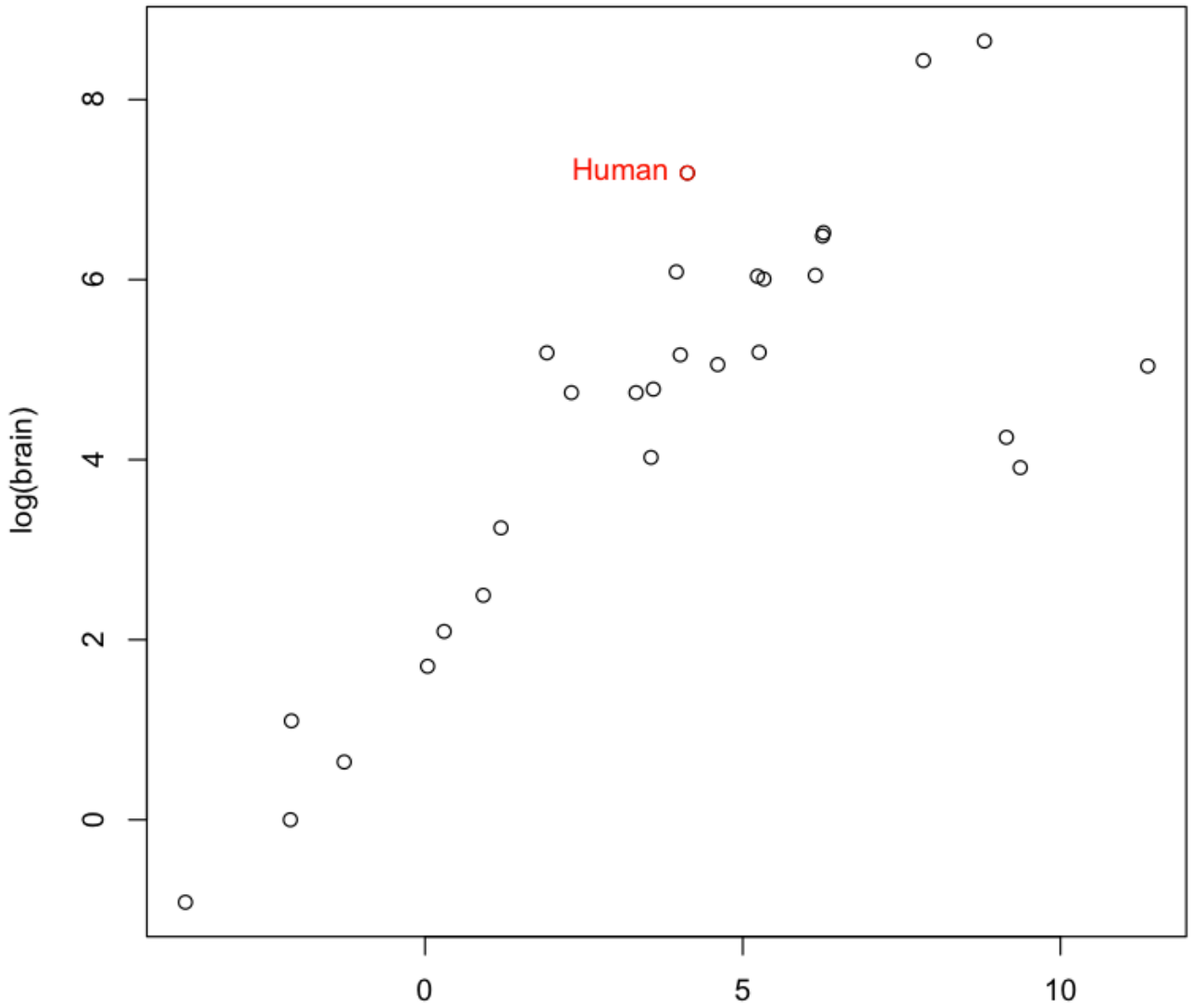
```
> par(mfrow=c(2,2), pch=16) # 4 figures per page
 > library(MASS)
> attach(Animals) # dataset from the MASS package
> plot(body, brain)
> plot(sqrt(body), sqrt(brain))
> plot((body)^0.1, (brain)^0.1)
> plot(log(body),log(brain))
> detach(Animals)
> par(mfrow=c(1,1), pch=1) # restore
```


brain
0
2000
5000
0
20000
60000
body

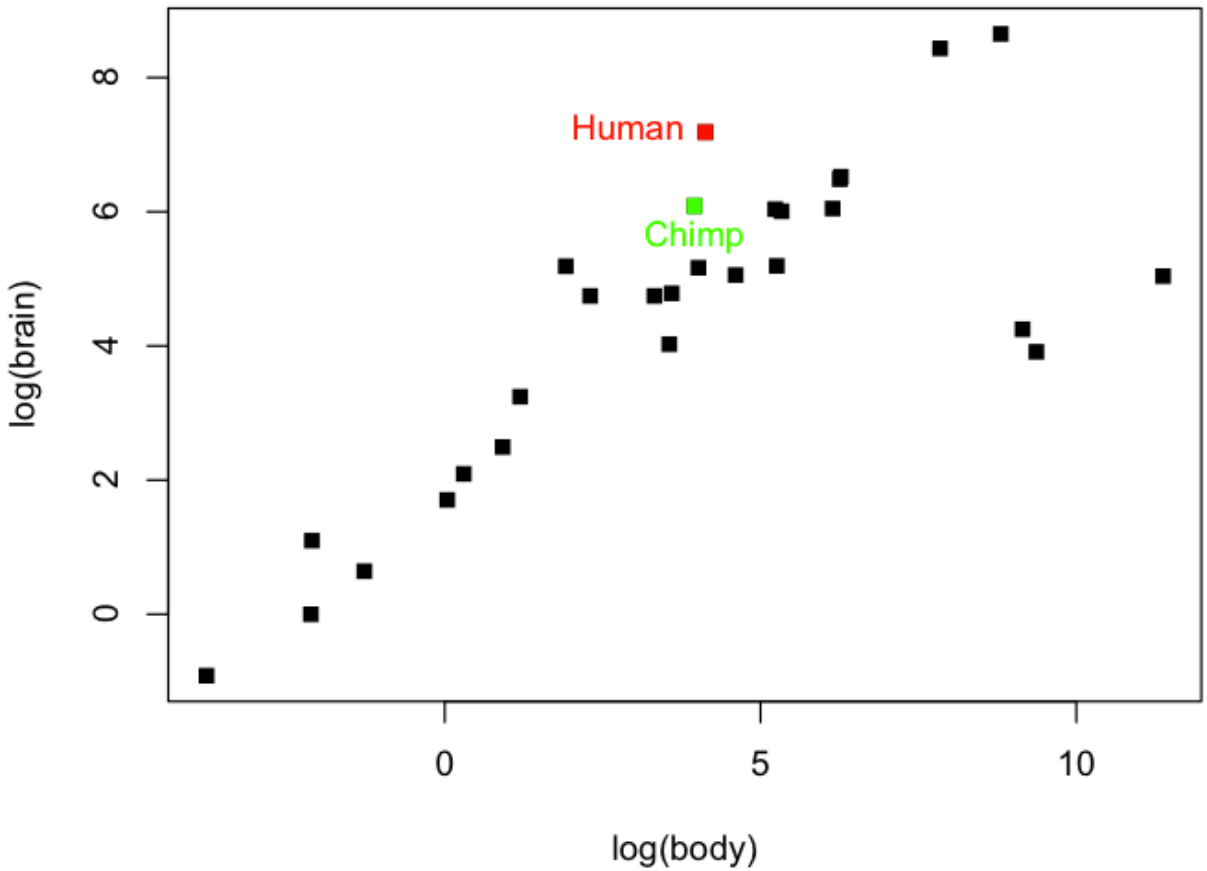

sqrt(brain)
0
20
40
60
0
50
100
150
200
250
300
sqrt(body)


(brain)^0.1
1.0
1.5
2.0
1.0
1.5
2.0
2.5
3.0
(body)^0.1


log(brain)
0
2
4
6
8
0
5
10
log(body)


Human
log(brain)
0
2
4
6
8
0
5
10

```
> library(MASS)
> attach(Animals) # dataset from the MASS package
> plot(log(body),log(brain))
> which(rownames(Animals)=="Human") # get row for Human
[1] 14
> points(log(Animals[14,]), col="red")
> text(x=log(Animals[14,1]), y=log(Animals[14,2]),
labels="Human", pos=2, col="red")
> detach(Animals)
```



Ex1: Highlight the chimpanzee datapoint in green.

Ex2: Add the chimpanzee label below datapoint in green.

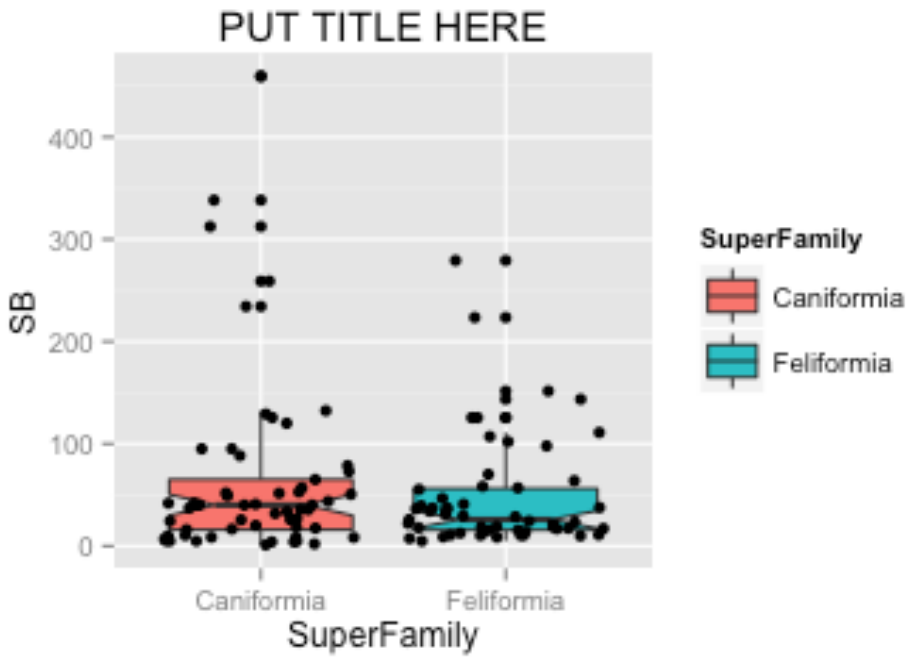
Ex3: Change the plotting characters (pch=15) from open circles to closed squares.

```
> attach(Animals) # dataset from the MASS package
> plot(log(body),log(brain),pch=15)
> points(log(Animals[14,]), col="red", pch=15)
> points(log(Animals[24,]), col="green", pch=15)
> text(x=log(Animals[14,1]), y=log(Animals[14,2]),
labels="Human", pos=2, col="red")
> text(x=log(Animals[24,1]), y=log(Animals[24,2]),
labels="Chimp", pos=1, col="green")
> detach(Animals)
```

ggplot2 is a **plotting system for R**, based on the grammar of graphics, which tries to take the good parts of base and lattice graphics and none of the bad parts. It takes care of many of the fiddly details that make plotting a hassle (like drawing legends) as well as providing a powerful model of graphics that makes it easy to produce complex multi-layered graphics.

```
install.packages("ggplot2")
library(ggplot2)
p <- ggplot(data)
p <- p + geom_?
p
```

http://docs.ggplot2.org/current/



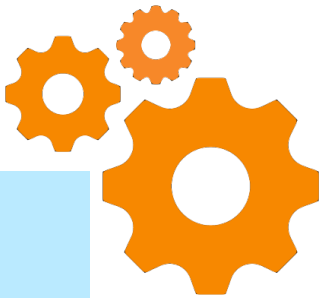

```
install.packages("ggplot2")
library(ggplot2)
p <- ggplot(carnivora, aes(SuperFamily, SB))
p <- p + geom_boxplot(notch = TRUE,aes(fill = SuperFamily))
p <- p + geom_jitter()
p <- p + labs(title = "PUT TITLE HERE")
p
```

Exploratory data analysis and data visualization for biological sequence (DNA and protein) data using the **R package seqinr**.

```
> library(seqinr)
# Help for seqinr
> library(help = seqinr)
> help(package = seqinr)
```

```
# Create and save nt sequences in fasta format:
cat(">SEQ1","ATGGAATGA",file = "SEQ1.fasta", sep = "\n")

# Locate the created fasta file

list.files(path=".")

# Load sequence:
dna1 = read.fasta("SEQ1.fasta", seqtype = "DNA")

# display sequence 1
dna1

# Calculate sequence Length
getLength(dna1)

# Calculates some codon usage
uco(dna1$SEQ1)

# Translate DNA into AA
translate(dna1$SEQ1,frame=0,sens="F")
```

Bioinformatics - R



```
# Calculate GC content:
> GC(dna1$SEQ1)
[1] 0.3333333
```

```
# Split a sequence into sub-sequences of 3
> splitseq(dna1$SEQ1, frame = 0, word = 3)
[1] "atg" "gaa" "tga"
```

R package ape - Analyses of Phylogenetics and Evolution

```
> library(ape)
# Help for ape
> library(help = ape)
> help(package = ape)
```

ape-package {ape} - Analyses of Phylogenetics and Evolution

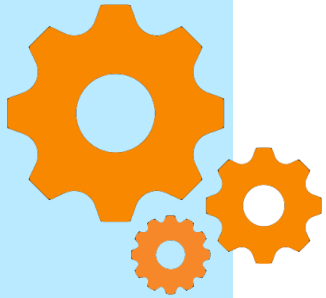
ape provides functions for reading and manipulating phylogenetic trees and DNA sequences, computing DNA distances, estimating trees with distance-based methods, and a range of methods for comparative analyses and analysis of diversification. Functionalities are also provided for programming new phylogenetic methods.

Instead of going to the NCBI website to retrieve sequence data from the NCBI database, you can retrieve sequence data from NCBI directly from R, by using for example the **R package ape**.

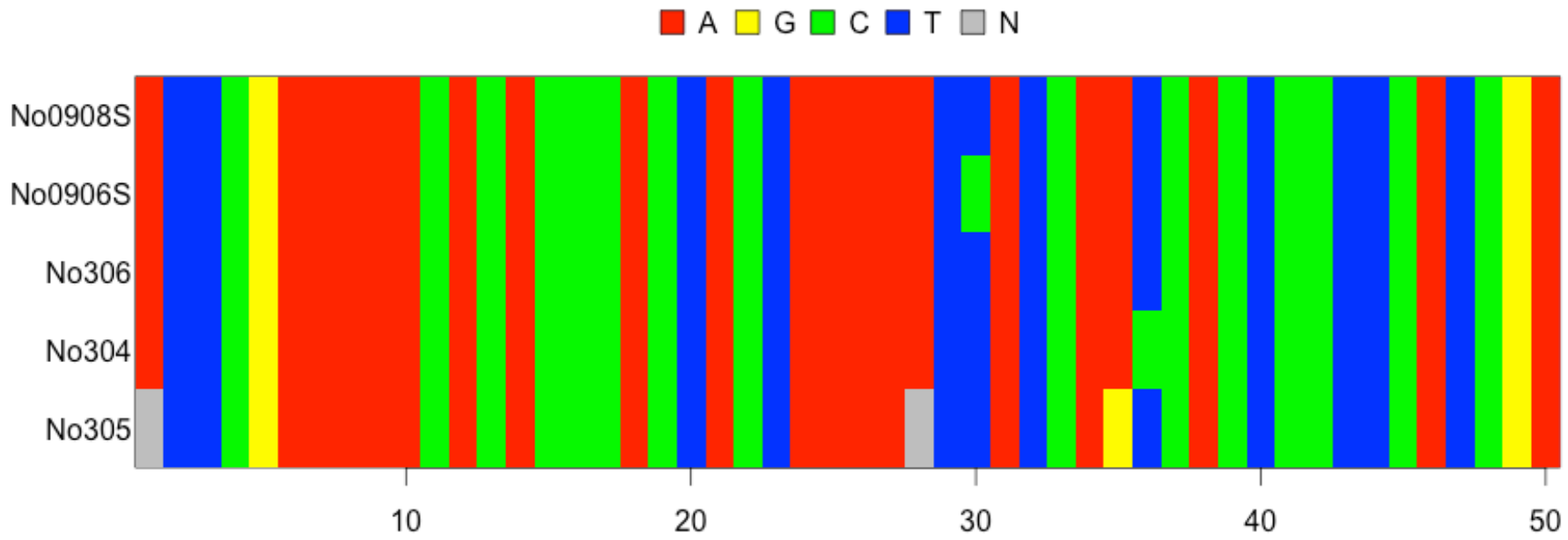
```
> library(ape)
# Get sequences from Genebank
> ref <- ("AF502402")
> Deug_adh <- read.GenBank(ref)
```

```
### get a summary of the sequences
> Deug_adh
1 DNA sequence in binary format stored in a list.

Sequence length: 411

Label: AF502402

Base composition:
    a     c     g     t
0.231 0.319 0.238 0.212

> base.freq(Deug_adh)
        a         c         g         t
0.2311436 0.3187348 0.2384428 0.2116788
```

```
> ref1 <- (c("NM_169441", "NM_080059")
> Dmel_Hsp70 <- read.GenBank(ref1)
> Dmel_Hsp70
2 DNA sequences in binary format stored in a list.

Mean sequence length: 2378.5
   Shortest sequence: 2375
    Longest sequence: 2382

Labels: NM_169441 NM_080059

Base composition:
    a     c     g     t
0.286 0.265 0.263 0.186

> Dmel_Hsp70$NM_169441
```

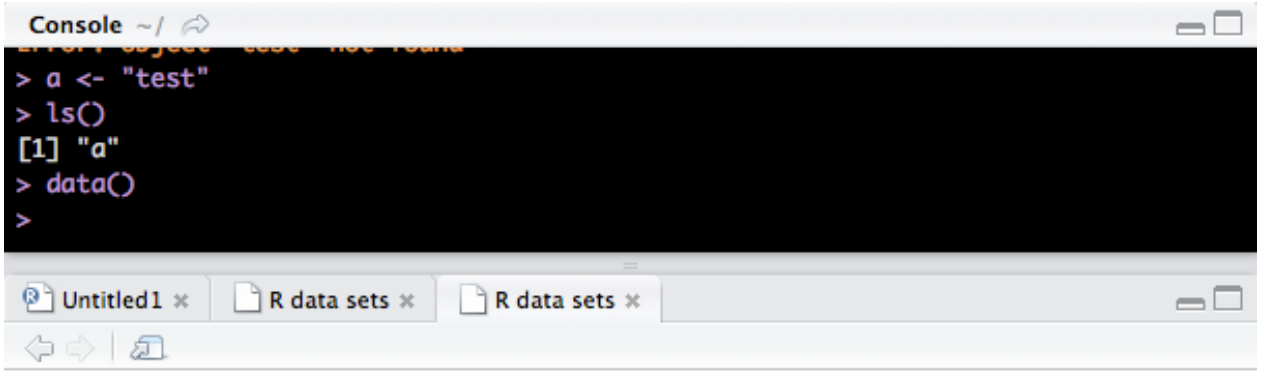


```
> data(woodmouse)
> image(woodmouse[1:5,1:50])
```



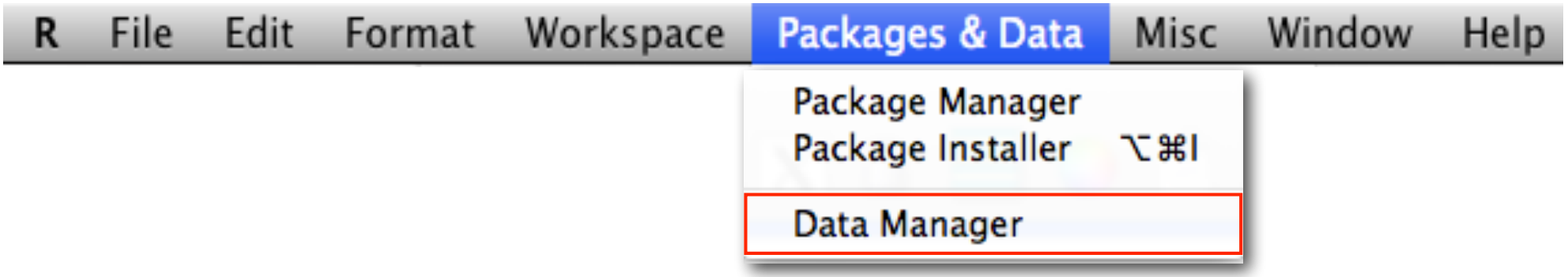
Data

Bioinformatics - R




```
Data sets in package 'datasets':

AirPassengers            Monthly Airline Passenger Numbers 1949-1960
BJsales                  Sales Data with Leading Indicator
BJsales.lead (BJsales)
                         Sales Data with Leading Indicator
BOD                      Biochemical Oxygen Demand
CO2                      Carbon Dioxide Uptake in Grass Plants
ChickWeight              Weight versus age of chicks on different diets
DNase                    Elisa assay of DNase
EuStockMarkets           Daily Closing Prices of Major European Stock
                         Indices, 1991-1998
Formaldehyde             Determination of Formaldehyde
HairEyeColor             Hair and Eye Color of Statistics Students
Harman23.cor             Harman Example 2.3
Harman74.cor             Harman Example 7.4
Indometh                 Pharmacokinetics of Indomethacin
InsectSprays             Effectiveness of Insect Sprays
JohnsonJohnson           Quarterly Earnings per Johnson & Johnson Share
LakeHuron                Level of Lake Huron 1875-1972
LifeCycleSavings         Intercountry Life-Cycle Savings Data
Loblolly                 Growth of Loblolly pine trees
Nile                     Flow of the River Nile
Orange                   Growth of Orange Trees
OrchardSprays            Potency of Orchard Sprays
PlantGrowth              Results from an Experiment on Plant Growth
Puromycin                Reaction Velocity of an Enzymatic Reaction
Seatbelts                Road Casualties in Great Britain 1969-84
Theoph                   Pharmacokinetics of Theophylline
```

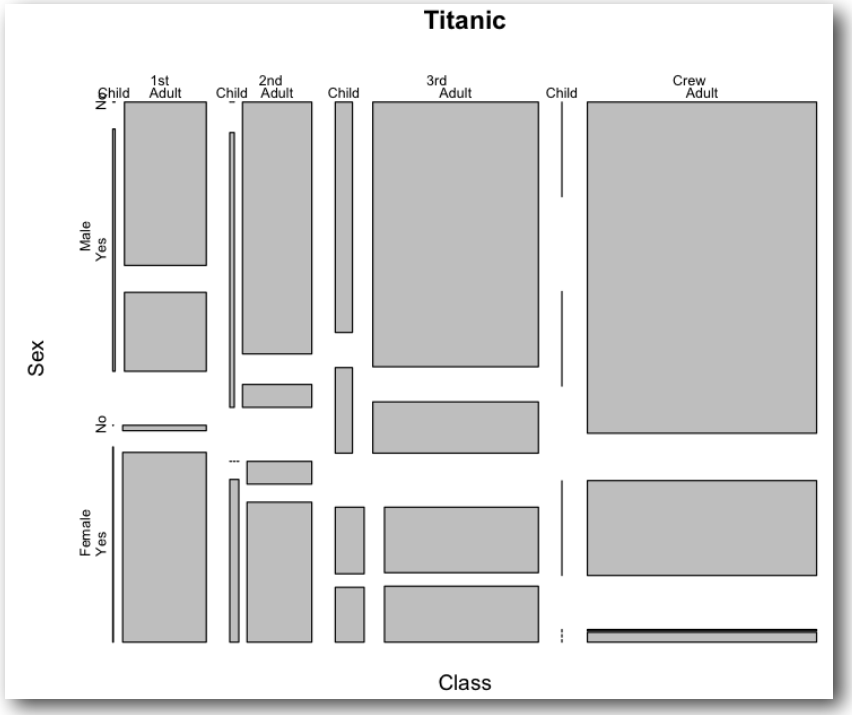
Bioinformatics - R



Data	Package	Description
AirPassengers	datasets	Monthly Airline Passenger Numbers 19
BJsales	datasets	Sales Data with Leading Indicator
BJsales.lead (BJs	datasets	Sales Data with Leading Indicator
BOD	datasets	Biochemical Oxygen Demand
CO2	datasets	Carbon Dioxide Uptake in Grass Plants
ChickWeight	datasets	Weight versus age of chicks on differen
DNase	datasets	Elisa assay of DNase
EuStockMarkets	datasets	Daily Closing Prices of Major European
Formaldehyde	datasets	Determination of Formaldehyde
HairEyeColor	datasets	Hair and Eye Color of Statistics Student
Harman23.cor	datasets	Harman Example 2.3
Harman74.cor	datasets	Harman Example 7.4
Indometh	datasets	Pharmacokinetics of Indomethicin
InsectSprays	datasets	Effectiveness of Insect Sprays
JohnsonJohnson	datasets	Quarterly Earnings per Johnson & Johns
LakeHuron	datasets	Level of Lake Huron 1875-1972

Bioinformatics - R

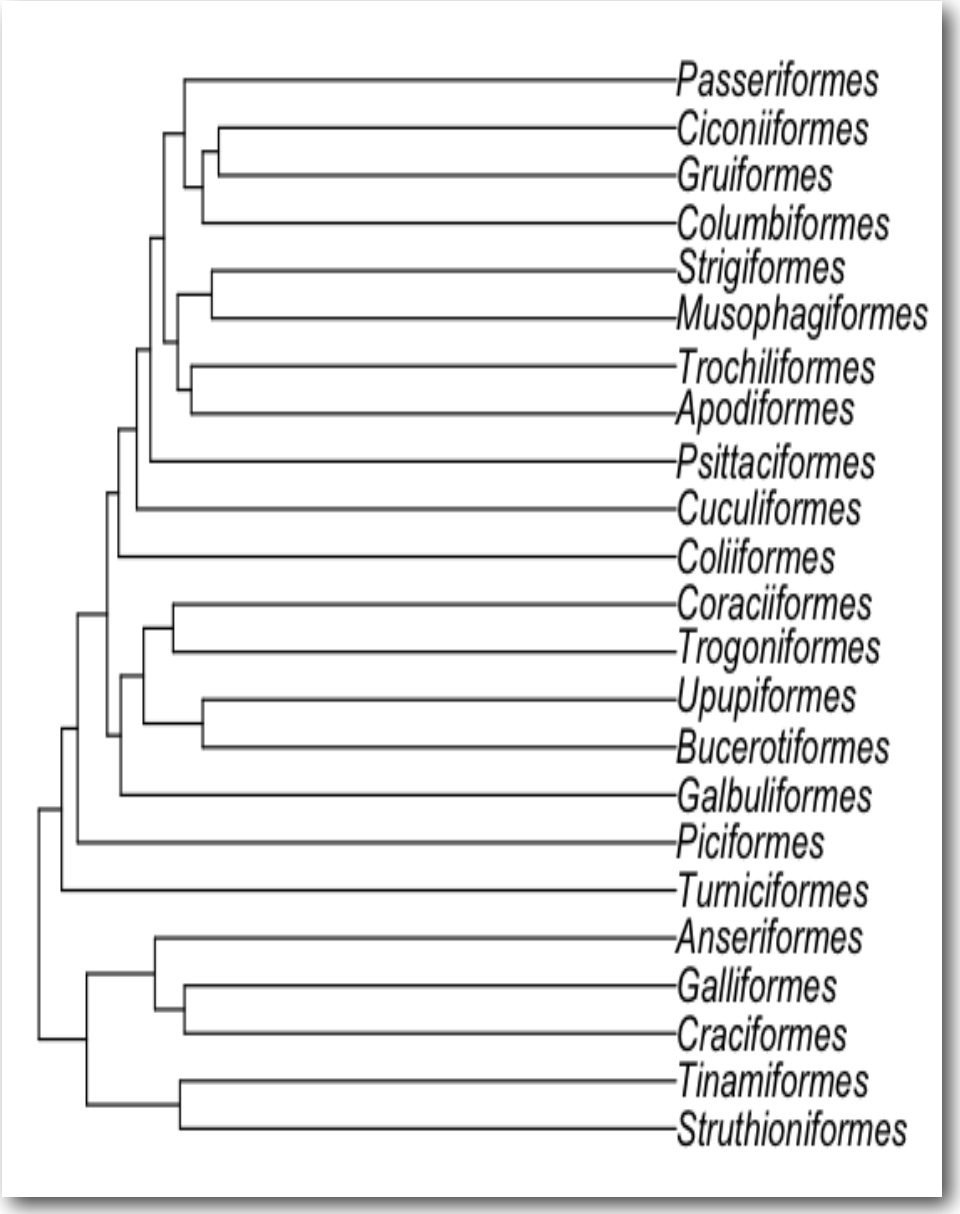
```
> data()
> plot(Titanic)
```




```
Data sets in package 'datasets':

AirPassengers            Monthly Airline Passenger Numbers 1949-1960
BJsales                  Sales Data with Leading Indicator
BJsales.lead (BJsales)   Sales Data with Leading Indicator
BOD                      Biochemical Oxygen Demand
CO2                      Carbon Dioxide Uptake in Grass Plants
ChickWeight              Weight versus age of chicks on different diets
DNase                    Elisa assay of DNase
EuStockMarkets           Daily Closing Prices of Major European Stock Indices,
                         1991-1998
Formaldehyde             Determination of Formaldehyde
HairEyeColor             Hair and Eye Color of Statistics Students
Harman23.cor             Harman Example 2.3
Harman74.cor             Harman Example 7.4
Indometh                 Pharmacokinetics of Indomethicin
InsectSprays             Effectiveness of Insect Sprays
JohnsonJohnson           Quarterly Earnings per Johnson & Johnson Share
LakeHuron                Level of Lake Huron 1875-1972
LifeCycleSavings         Intercountry Life-Cycle Savings Data
Loblolly                 Growth of Loblolly pine trees
Nile                     Flow of the River Nile
Orange                   Growth of Orange Trees
OrchardSprays            Potency of Orchard Sprays
PlantGrowth              Results from an Experiment on Plant Growth
Puromycin                Reaction Velocity of an Enzymatic Reaction
Seatbelts                Road Casualties in Great Britain 1969-84
Theoph                   Pharmacokinetics of Theophylline
Titanic                  Survival of passengers on the Titanic
ToothGrowth              The Effect of Vitamin C on Tooth Growth in Guinea Pigs
UCBAdmissions            Student Admissions at UC Berkeley
UKDriverDeaths           Road Casualties in Great Britain 1969-84
UKgas                    UK Quarterly Gas Consumption
USAccDeaths              Accidental Deaths in the US 1973-1978
USArrests                Violent Crime Rates by US State
USJudgeRatings           Lawyers' Ratings of State Judges in the US Superior
                         Court
```

Bioinformatics - R

```
> library(help = ape)
> library("ape")
> example(bird.orders)
brd.rd> data(bird.orders)
brd.rd> plot(bird.orders)
Hit <Return> to see next plot:
```



Matrix Access

```
> let <- letters
> let
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l"
"m" "n" "o" "p" "q" "r" "s" "t" "u" "v" "w" "x" "y"
"z"
> let[3]
[1] "c"
> let[3:5]
[1] "c" "d" "e"
```

```
> let <- letters
> let
[1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l"
"m" "n" "o" "p" "q" "r" "s" "t" "u" "v" "w" "x" "y"
"z"
> which(let == “g”)
[1] 7
```

```
> x["A2"]
A2
 3
> x[2]
A2
 3
> x[2:3]
A2 A3
 3  5
> x[-2]
A1 A3
 1  5
> x[-(2:3)]
A1
 1
```

```
> x[c(3,2,1)]
A3 A2 A1
 5  3  1
> order(x)
[1] 1 2 3
> order(x, decreasing=TRUE)
[1] 3 2 1
> x[c(3,2,1,4)]
  A3   A2   A1 <NA>
   5    3    1   NA
> x[4] <- 7
> x[c(3,2,1,4)]
A3 A2 A1
 5  3  1  7
```

```
> x <- c(A=1, B=3, C=5)
> x
A B C
1 3 5
> names(x)
[1] "A" "B" "C"
> names(x) <- c(“A1","A2","A3") # change names
> names(x)
[1] "A1" "A2" “A3"
> names(y) <- c("A","B","C")
> y
 A  B  C
 2  6 10
```

```
> set.seed(161004)
> x <- runif(5) # uniform distribution
> x
0.500 0.201 0.197 0.268 0.406
> x < 0.5
FALSE  TRUE  TRUE  TRUE  TRUE
> x
> x[ x < 0.5]
0.201 0.197 0.268 0.406
> x[x >= 0.5]
0.500
```

Bioinformatics - R

```
> A <- matrix(c(1,2,3, 11,12,13), nrow = 2, ncol = 3, byrow = TRUE,
+         dimnames = list(c("row1", "row2"),
+                         c("C.1", "C.2", "C.3")))
> A
     C.1 C.2 C.3
row1   1   2   3
row2  11  12  13

> A[2,3] # line 2,column 3
[1] 13
> A[1,] # line 1
C.1 C.2 C.3
  1   2   3
> A[,1] # column 1
row1 row2
   1   11
```

```
> a <- 17
> b <- 20
> a > b
[1] FALSE
> a == b
[1] FALSE
> a != b
[1] TRUE
> a <= b
[1] TRUE
```

Operators

equal	==
not equal	!=
bigger	>
less	<
bigger or equal	>=
less or equal	<=