Evolutionary Genetics

LV 25600-01 | Lecture with exercises | 4KP

Jean-Claude Walser

jean-claude.walser [at] usys.ethz.ch

HS2024

Please forget the over-hyped clowns and make way for the real heroes....

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)

2022 R version 4.2.1

Integrated Development Environment

RStudio https://rstudio.com

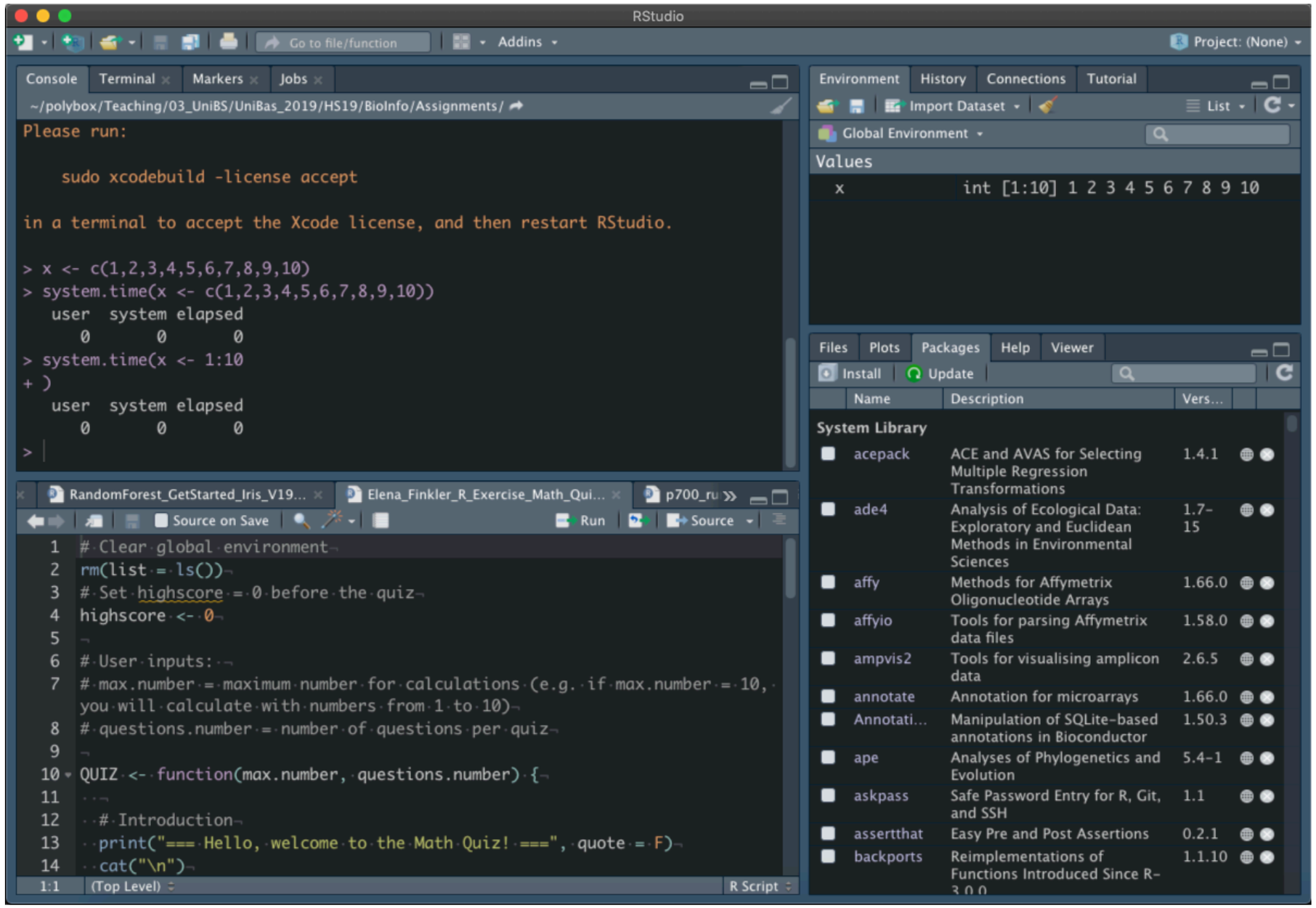
jupyter https://jupyter.org/

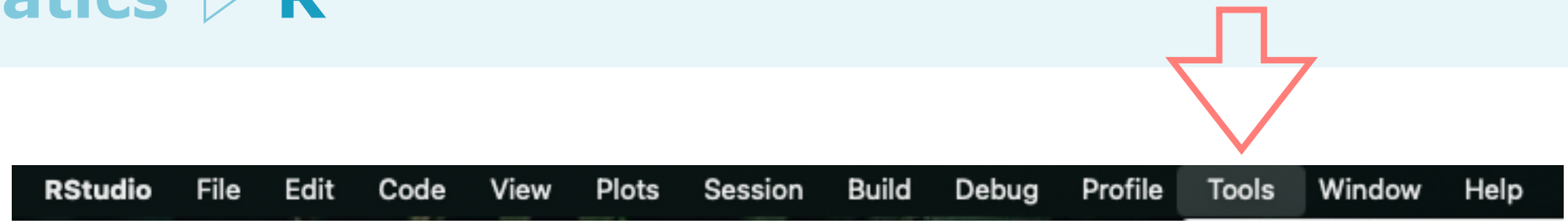
https://rkward.kde.org

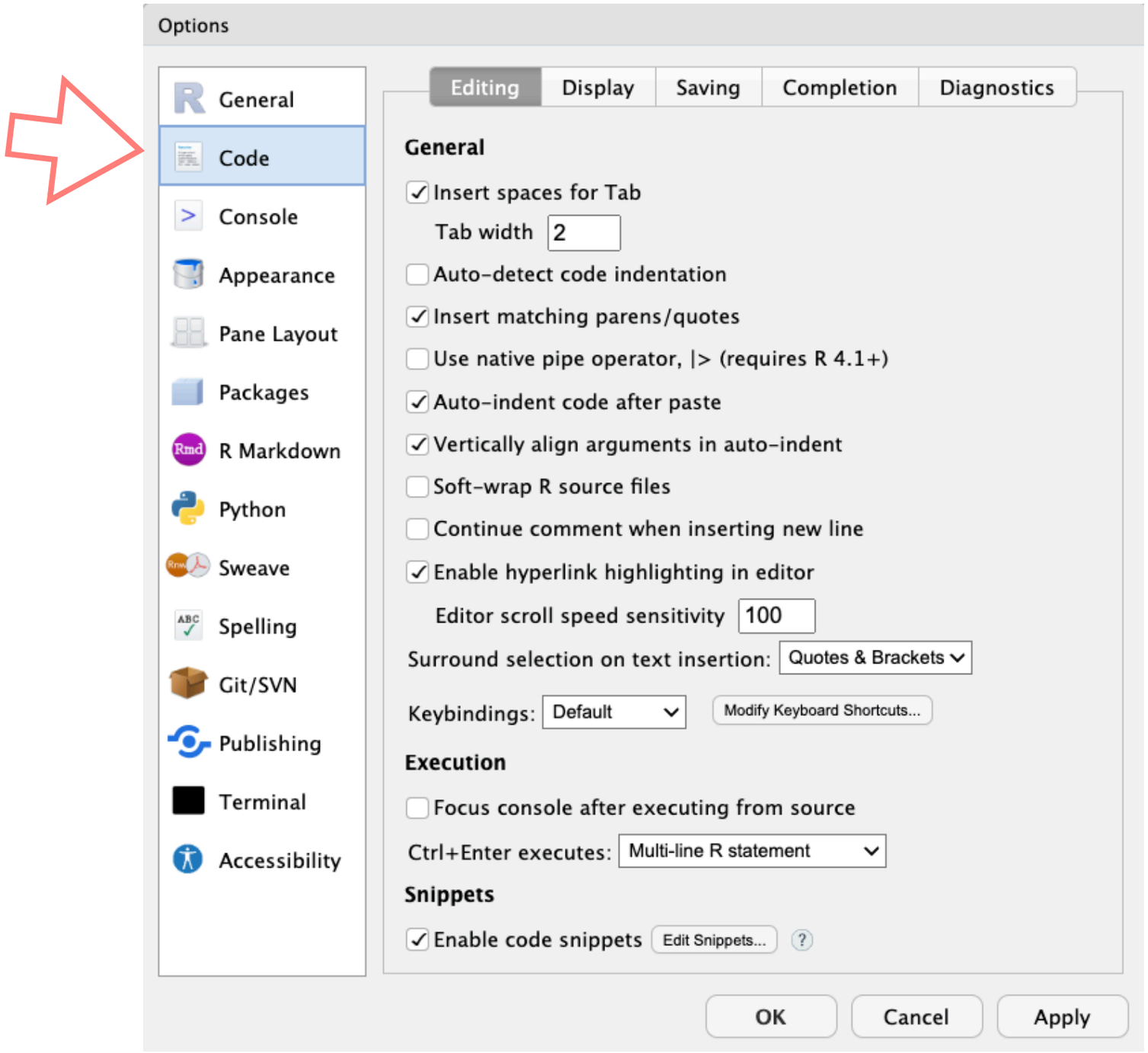
https://www.getarchitect.io

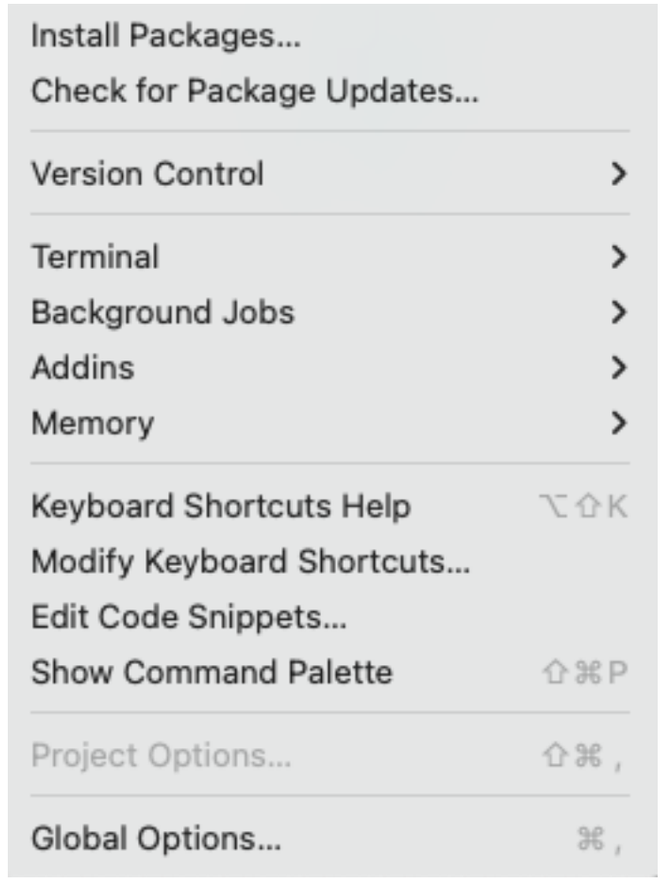
https://atom.io/packages/**rbox**

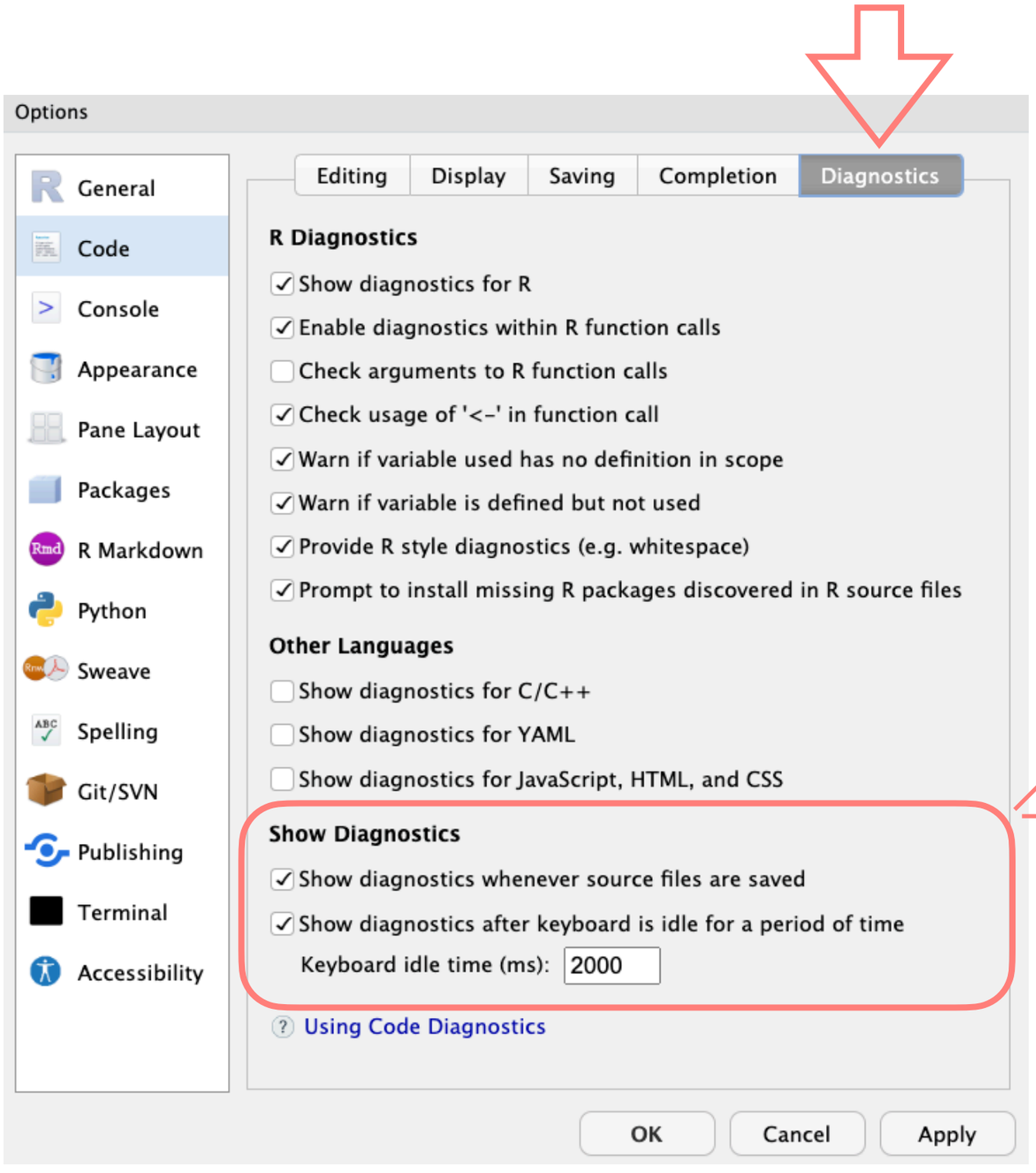
RStudio


RStudio
Go to file/function
Addins
Project: (None)
Console
Terminal
Markers
Jobs
~/polybox/Teaching/03_UniBS/UniBas_2019/HS19/BioInfo/Assignments/
Please run:
sudo xcodebuild -license accept
in a terminal to accept the Xcode license, and then restart RStudio.
> x <- c(1,2,3,4,5,6,7,8,9,10)
> system.time(x <- c(1,2,3,4,5,6,7,8,9,10))
user system elapsed
0 0 0
> system.time(x <- 1:10
+)
user system elapsed
0 0 0
>
Environment
History
Connections
Tutorial
Import Dataset
List
Global Environment
Values
x int [1:10] 1 2 3 4 5 6 7 8 9 10
Files
Plots
Packages
Help
Viewer
Install
Update
Name
Description
Vers...
System Library
acepack ACE and AVAS for Selecting Multiple Regression Transformations 1.4.1
ade4 Analysis of Ecological Data: Exploratory and Euclidean Methods in Environmental Sciences 1.7-15
affy Methods for Affymetrix Oligonucleotide Arrays 1.66.0
affyio Tools for parsing Affymetrix data files 1.58.0
ampvis2 Tools for visualising amplicon data 2.6.5
annotate Annotation for microarrays 1.66.0
Annotati... Manipulation of SQLite-based annotations in Bioconductor 1.50.3
ape Analyses of Phylogenetics and Evolution 5.4-1
askpass Safe Password Entry for R, Git, and SSH 1.1
assertthat Easy Pre and Post Assertions 0.2.1
backports Reimplementations of Functions Introduced Since R-3.0.0 1.1.10
RandomForest_GetStarted_Iris_V19...
Elena_Finkler_R_Exercise_Math_Qui...
p700_ru
Source on Save
Run
Source
1 # Clear global environment
2 rm(list = ls())
3 # Set highscore = 0 before the quiz
4 highscore <- 0
5
6 # User inputs:
7 # max.number = maximum number for calculations (e.g. if max.number = 10, you will calculate with numbers from 1 to 10)
8 # questions.number = number of questions per quiz
9
10 QUIZ <- function(max.number, questions.number) {
11
12 # Introduction
13 print("=== Hello, welcome to the Math Quiz! ===", quote = F)
14 cat("\n")
1:1
(Top Level)
R Script

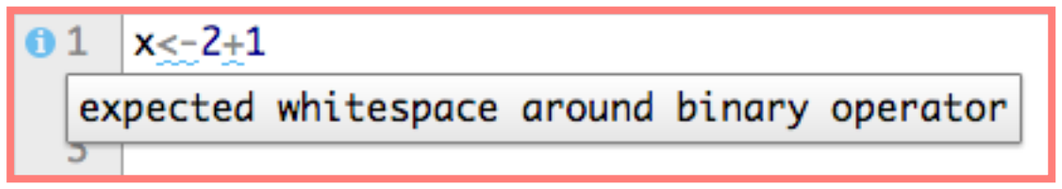

RStudio
File
Edit
Code
View
Plots
Session
Build
Debug
Profile
Tools
Window
Help

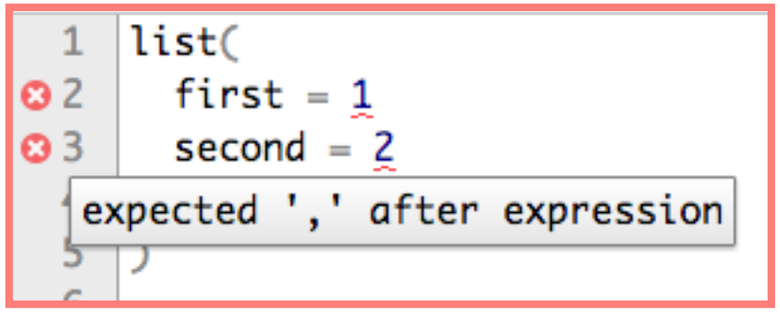

Options
General
Code
Console
Appearance
Pane Layout
Packages
R Markdown
Python
Sweave
Spelling
Git/SVN
Publishing
Terminal
Accessibility
Editing
Display
Saving
Completion
Diagnostics
General
Insert spaces for Tab
Tab width 2
Auto-detect code indentation
Insert matching parens/quotes
Use native pipe operator, |> (requires R 4.1+)
Auto-indent code after paste
Vertically align arguments in auto-indent
Soft-wrap R source files
Continue comment when inserting new line
Enable hyperlink highlighting in editor
Editor scroll speed sensitivity 100
Surround selection on text insertion: Quotes & Brackets
Keybindings: Default
Modify Keyboard Shortcuts...
Execution
Focus console after executing from source
Ctrl+Enter executes: Multi-line R statement
Snippets
Enable code snippets
Edit Snippets...
OK
Cancel
Apply


Install Packages...
Check for Package Updates...
Version Control
Terminal
Background Jobs
Addins
Memory
Keyboard Shortcuts Help
Modify Keyboard Shortcuts...
Edit Code Snippets...
Show Command Palette
Project Options...
Global Options...



Enable **diagnostics** within R function calls
Controls whether diagnostics are performed within function calls. Toggle this if your code makes heavy use of non-standard evaluation, and RStudio is unable to produce correct diagnostics for you.





HELP

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

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

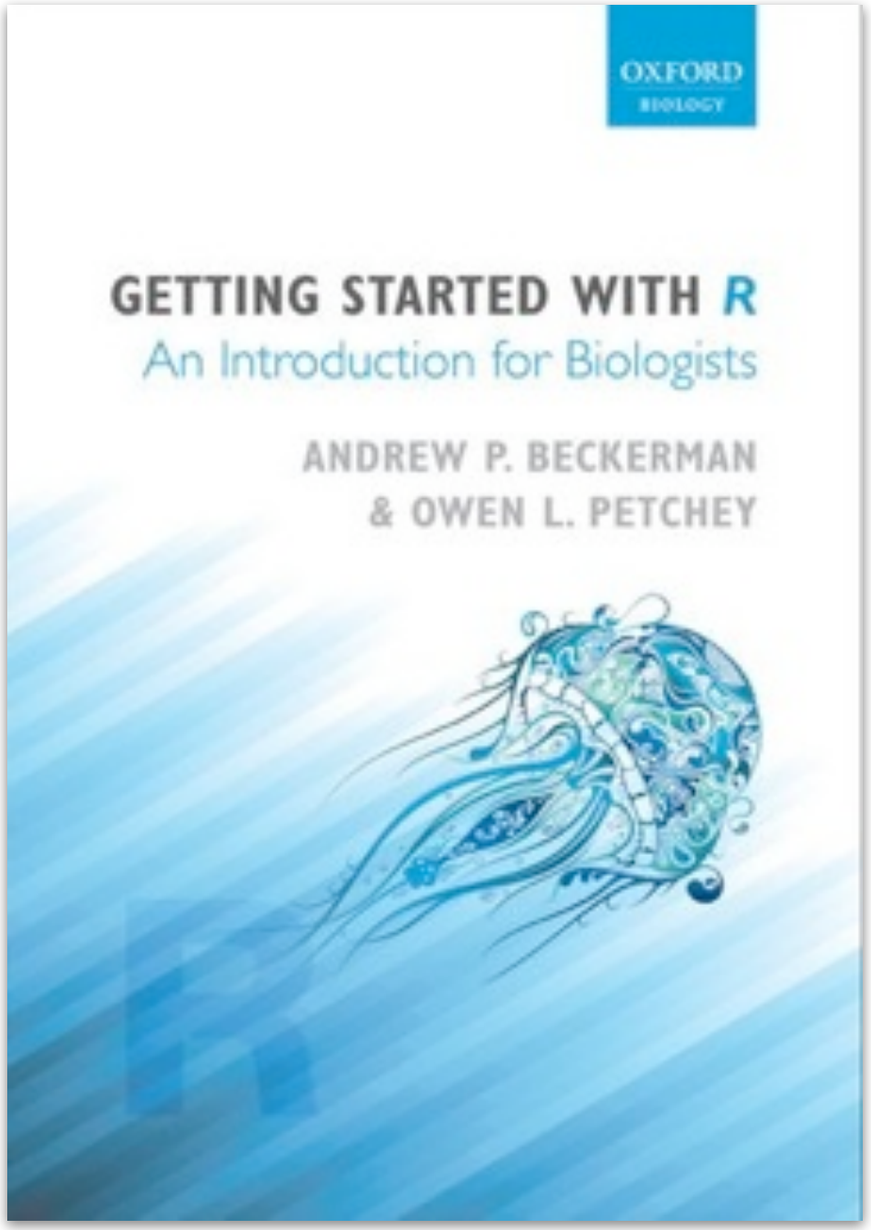
R-help: https://stat.ethz.ch/mailman/listinfo/r-help

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

RSEEK: http://rseek.org/

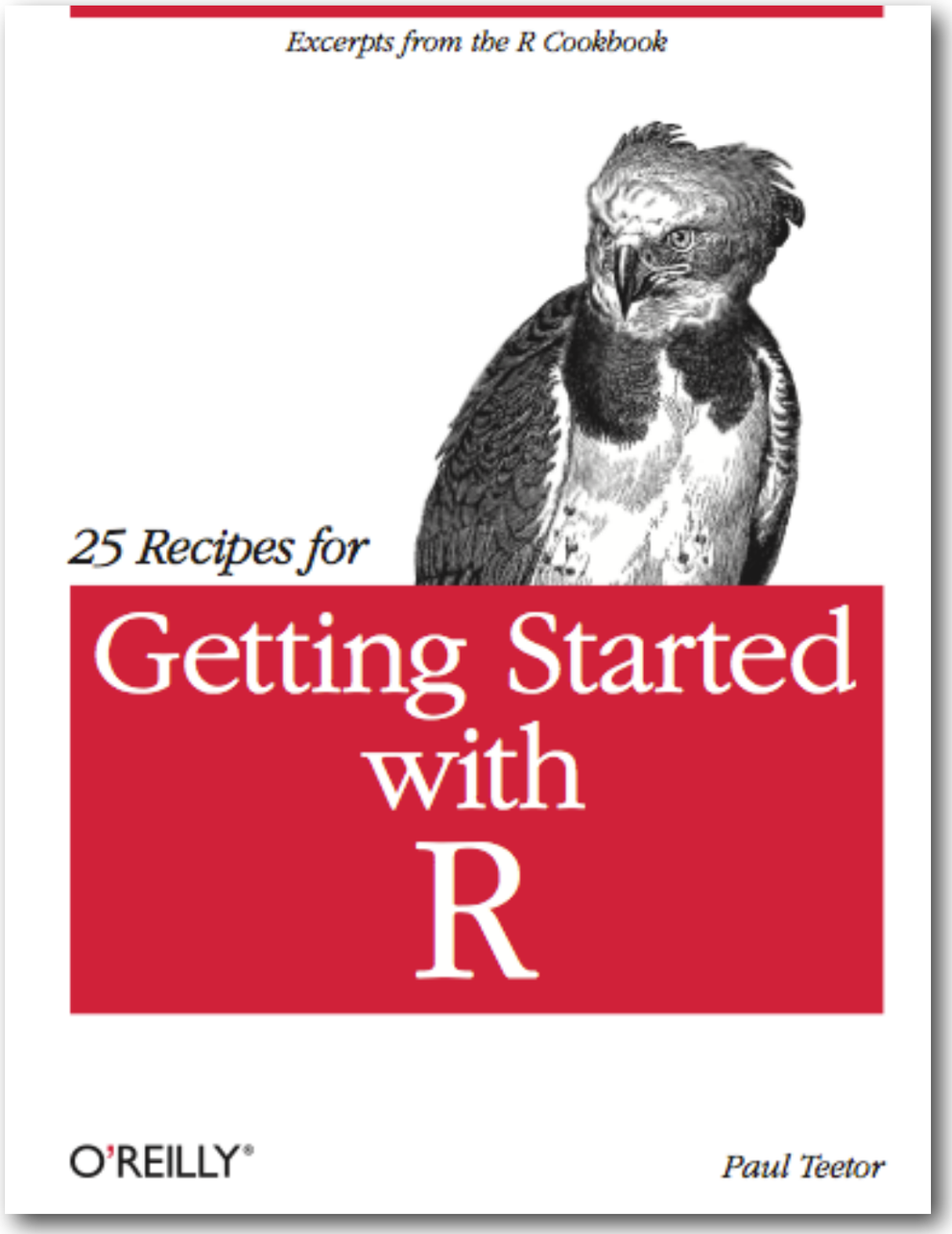
WIKIBOOKS: http://en.wikibooks.org

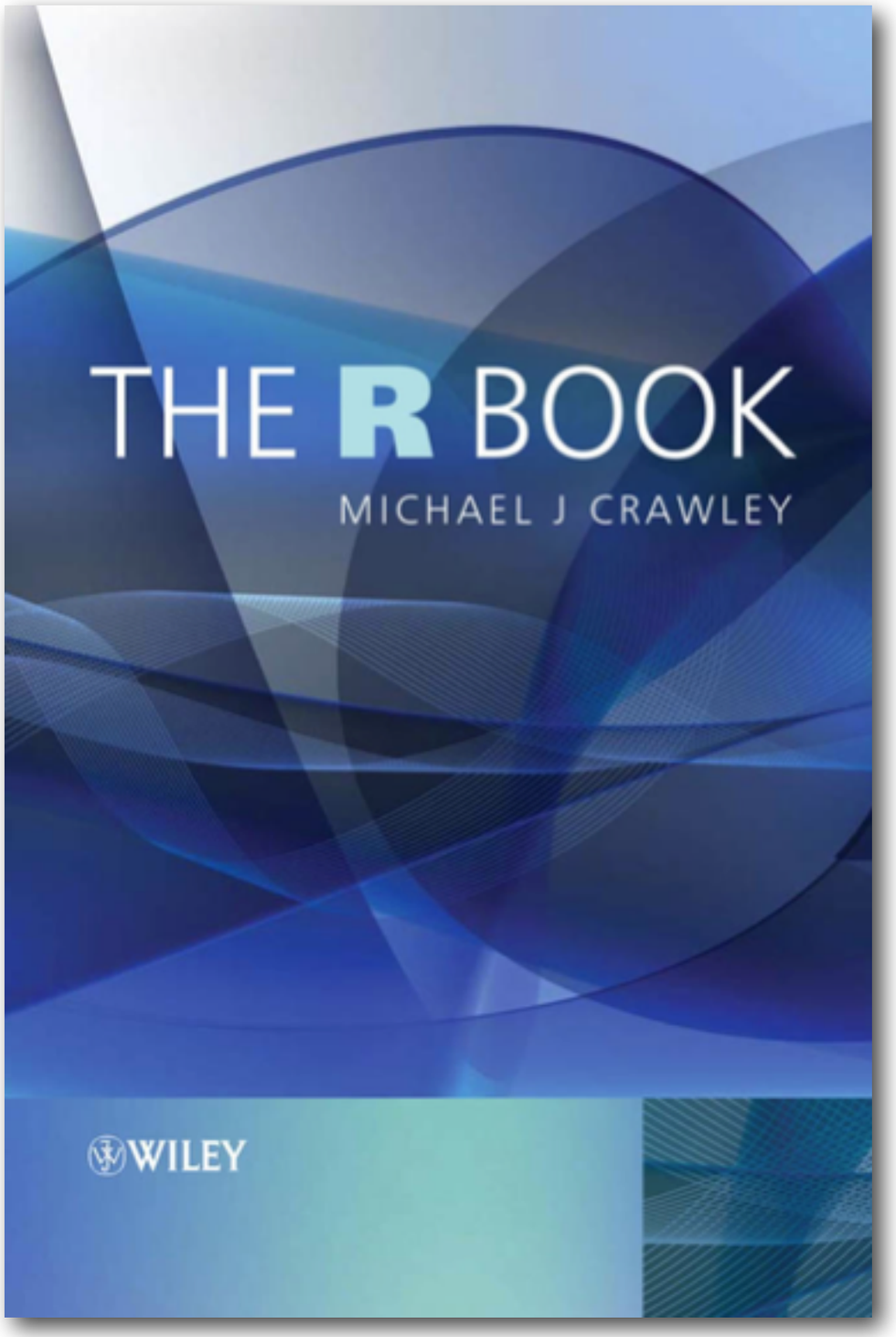
Command in R	Result
`help(t.test)` or `?t.test`	These functions provide access to documentation. In the example R will provide documentation for the function t.test
`help.search("anova")`	Searches the help system for documentation matching a given character string. Names and titles of the matched help 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.

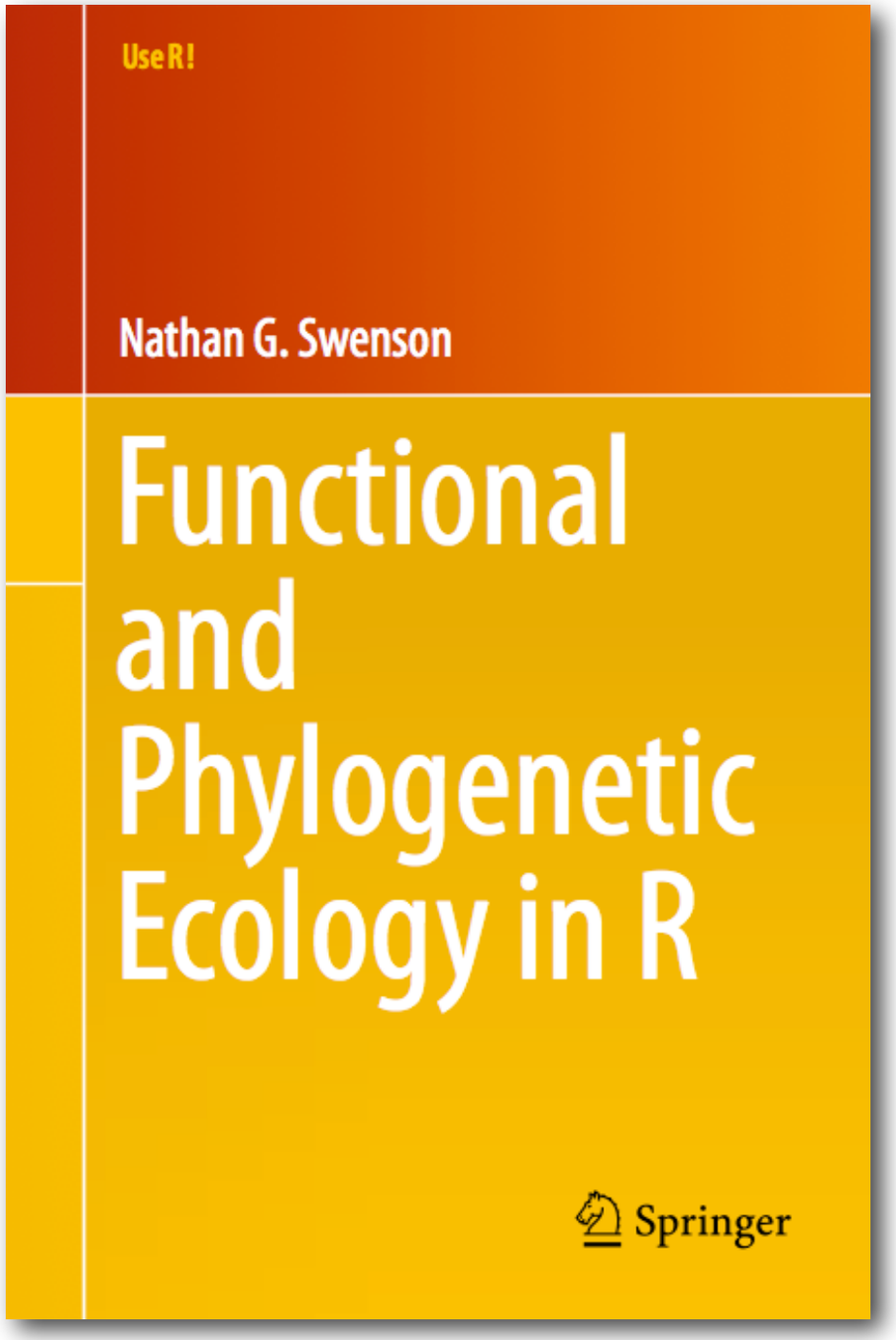


- A book containing a simple, easy, and engaging introduction to R for biologists
- Walks readers through the fundamentals of using R to import, manage and explore data, produce figures, and do basic statistics
- Designed for biologists new to R, who want an easy boost up the initially steep learning 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







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

{swirl}

Learn R, in R.

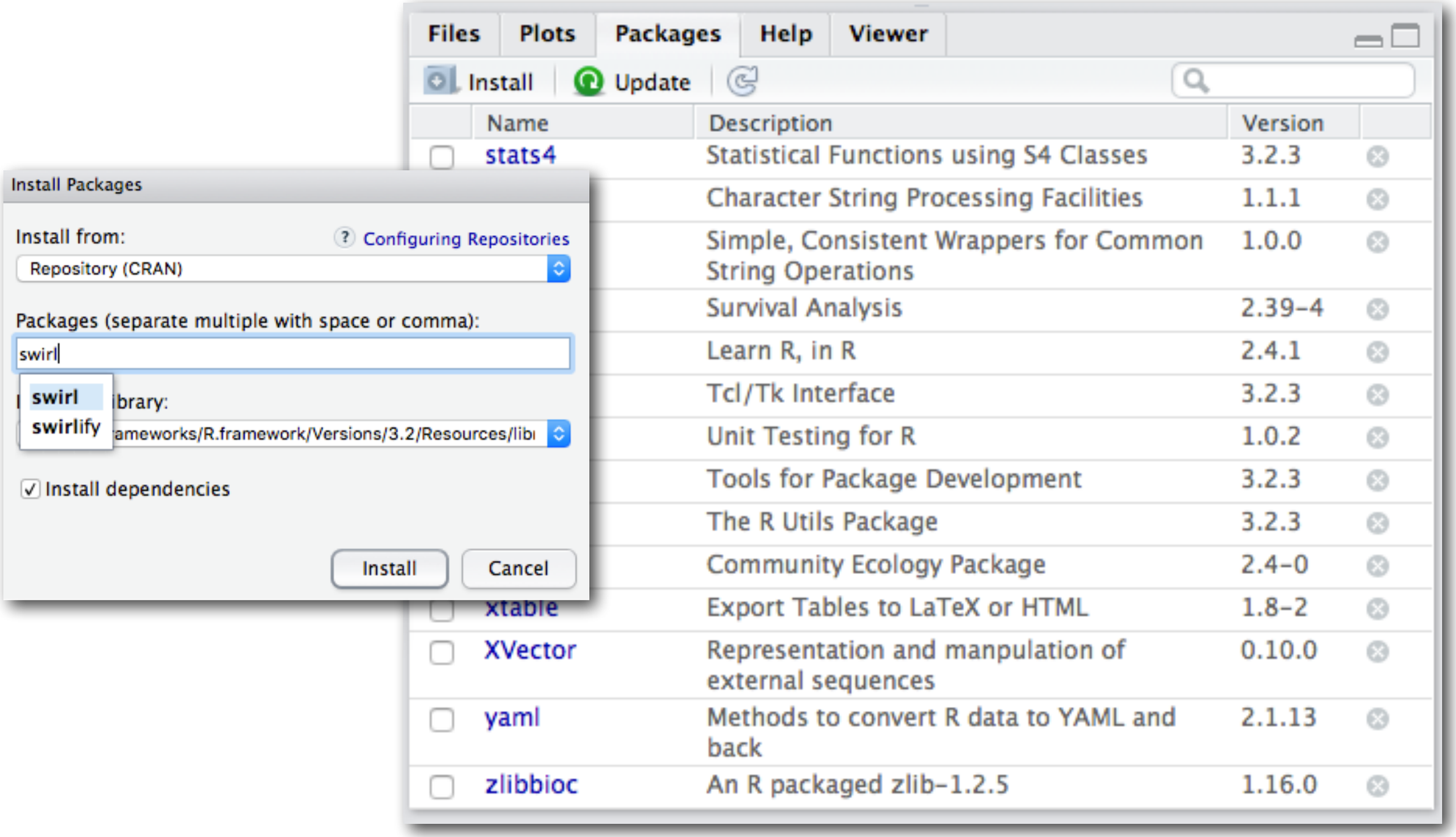
swirl teaches you R programming and data science
interactively, at your own pace, and right in the R console!


```
> 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
```

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 repositories like CRAN, which hosts over 1000 packages for various purposes.


Files
Plots
Packages
Help
Viewer
Install
Update
Name
Description
Version
stats4
Statistical Functions using S4 Classes
3.2.3
Character String Processing Facilities
1.1.1
Simple, Consistent Wrappers for Common String Operations
1.0.0
Survival Analysis
2.39-4
Learn R, in R
2.4.1
Tcl/Tk Interface
3.2.3
Unit Testing for R
1.0.2
Tools for Package Development
3.2.3
The R Utils Package
3.2.3
Community Ecology Package
2.4-0
xtable
Export Tables to LaTeX or HTML
1.8-2
XVector
Representation and manpulation of external sequences
0.10.0
yaml
Methods to convert R data to YAML and back
2.1.13
zlibbioc
An R packaged zlib-1.2.5
1.16.0
Install Packages
Install from:
Configuring Repositories
Repository (CRAN)
Packages (separate multiple with space or comma):
swirl
swirl
swirlify
ibrary:
ameworks/R.framework/Versions/3.2/Resources/libı
Install dependencies
Install
Cancel

```
> 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.

Bioconductor
OPEN SOURCE SOFTWARE FOR BIOINFORMATICS

Home | Install | Help | Developers | About

Search:

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.

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.

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

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

```
if (!require("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
```

Bioconductor Installer

```
BiocManager::install("muscle")
```

The MUSCLE algorithm employs a progressive alignment approach to optimise pairwise alignment scores, and achieves both high accuracy and reduced computational time even when handling thousands of sequences (Edgar, 2004,a). The R package muscle integrates the MUSCLE algorithm into the Bioconductor project by utilizing existing Biostrings classes for representing sequence objects and multiple alignments.

More Sources for R packages

```
> install.packages("devtools")
> devtools::install_github("<author>/<package>"
```

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

pak - A Fresh Approach to R Package Installation

```
install.packages("pak")
```

Install a package from CRAN or Bioconductor

```
> pak::pkg_install("tibble")
```

Install a package from GitHub

```
> pak::pkg_install("tidyverse/tibble")
```

Update all dependencies of a package

```
> pak::pkg_install("tibble", upgrade = TRUE)
```

Reinstall a package

```
> pak::pkg_install("tibble?reinstall")
```

Dependencies of a CRAN or Bioconductor package

```
> pak::pkg_deps("tibble")
```

Dependency **tree** of a CRAN / Bioconductor package

```
> pak::pkg_deps_tree("tibble")
```

```
✔ Updated metadata database: 5.26 MB in 15 files.
✔ Updating metadata database ... done
tibble 3.2.1 ✨ ⬇ (684.10 kB)
├─fansi 1.0.4 ✨ ⬇ (380.94 kB)
├─lifecycle 1.0.3 ✨ ⬇ (123.64 kB)
│ ├─cli 3.6.1 ✨ ⬇ (1.38 MB)
│ ├─glue 1.6.2 ✨ ⬇ (155.72 kB)
│ └─rlang 1.1.1 ✨ ⬇ (1.88 MB)
├─magrittr 2.0.3 ✨ ⬇ (232.43 kB)
├─pillar 1.9.0 ✨ ⬇ (648.86 kB)
│ ├─cli
│ ├─fansi
│ ├─glue
│ ├─lifecycle
│ ├─rlang
│ ├─utf8 1.2.3 ✨ ⬇ (209.41 kB)
│ └─vctrs 0.6.3 ✨ ⬇ (1.88 MB)
│   ├─cli
│   ├─glue
│   ├─lifecycle
│   └─rlang
├─pkgconfig 2.0.3 ✨ ⬇ (18.21 kB)
├─rlang
└─vctrs

Key:  ✨ new | ⬇ download
```

Understanding basic data types (classes)

Objects have a class attribute, which is important because R "decides" how to treat objects based on their class.

- Everything in R is an **object**.
- R has 5 basic atomic classes.

Classes of Objects:

- **num**eric (real numbers)
- **log**ical (True/False)
- **int**eger
- **cha**racter
- **com**plex

```
> str()
> typeof()
> class()
> mode()
```

R has various data structures

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 <- list(a = "1", name = "A")
```

Array / Matrix (vector with dimensions)

```
D1 <- array(c(1,2,3,4,5,6), dim=c(2,3))
D2 <- matrix(nrow = 2, ncol = 3)
dim(D1); dim(D2)
```

R has various data structures

Factors (special vector with categorical data)

```
x <- factor(c("y", "n", "y"))
```

Data frame (structure for tabular data)

```
df <- data.frame(a = 1:3, b = rnorm(3))
class(df)   # class attribute
dim(df)     # dimension
names(df)   # column names
str(df)     # class of columns
head(df, 5) # first 5 rows
tail(df, 5) # last 5 rows
```

Missing values

In R, missing values are represented by the symbol **NA** (**not available**). Impossible values (e.g., dividing by zero) are represented by the symbol **NaN** (**not a number**).

```
> is.na()
> is.nan()
```

```
> x <- c(2, 3, NA, NaN)
> is.na(x)
[1] FALSE FALSE  TRUE  TRUE
> is.nan(x)
[1] FALSE FALSE  FALSE TRUE
```

```
> x <- c(2, "b", NA, NaN, "na")
> is.na(x)
[1] FALSE FALSE  TRUE FALSE FALSE
> is.nan(x)
[1] FALSE FALSE FALSE FALSE FALSE
```

Variable names - You can use simple variable names such as 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 can contain numbers, but it cannot start with a number!** It can contain underscores (_), but not operators (e.g. <, =), punctuation (e.g. !, [) or the comment character (e.g. #). Be careful not to "clobber" 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

```
# 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

```
# A Big Matrix
x <- matrix( rnorm( 50000 * 900),
        nrow = 50000,
        ncol =   900)

dim(x)

# Version A
system.time(t(x) %*% x)

# Version B
system.time(crossprod(x))
```

Profiling in R

```
mean(replicate(10, system.time(t(x) %*% x))[1,])
mean(replicate(10, system.time(crossprod(x)))[1,])
```

Vector Recycling

Vector recycling is one of the most useful concepts in R. When we try to perform an operation on two vectors of different lengths, R automatically recycles the required number of additional vectors. In the example below, the shorter vector is recycled and added to the number of operators.

```
> 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
> suppressWarnings(x*z)
[1] 1 9 5
```

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

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

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

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

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

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

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

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

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

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

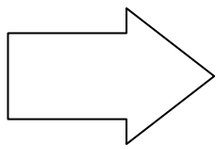
$$\begin{bmatrix}1\\2\\3\\4\\5\\6\end{bmatrix} / \begin{bmatrix}1\\2\\3\\1\\2\\3\end{bmatrix} = \begin{bmatrix}1\\1\\1\\4\\2.5\\2\end{bmatrix}$$

To **permanently suppress warnings** in the **global settings**, you can use the warn option in options function as follows:

```
> options(warn=-1)
```

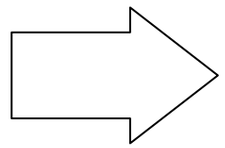
To **temporarily suppress warnings** in **global settings** and turn them back on, use the following code:

```
> Default_Warnings <- getOption("warn")
> options(warn = -1)
  [YOUR CODE]
> options(warn = Default_Warnings)
```

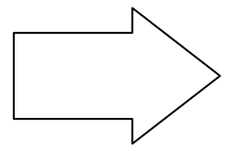


```
read.table("file.txt")
read.csv("file.csv")
read.delim("file.tab")
```

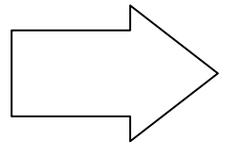
```
write(x,"file.txt")
write.csv(x,"file.csv")
write.table(x,"file.tab")
```



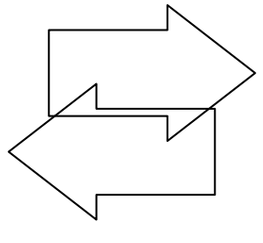
```
sink("file.txt")
   …
sink()
```



```
pdf()
   …
dev.off()
```



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



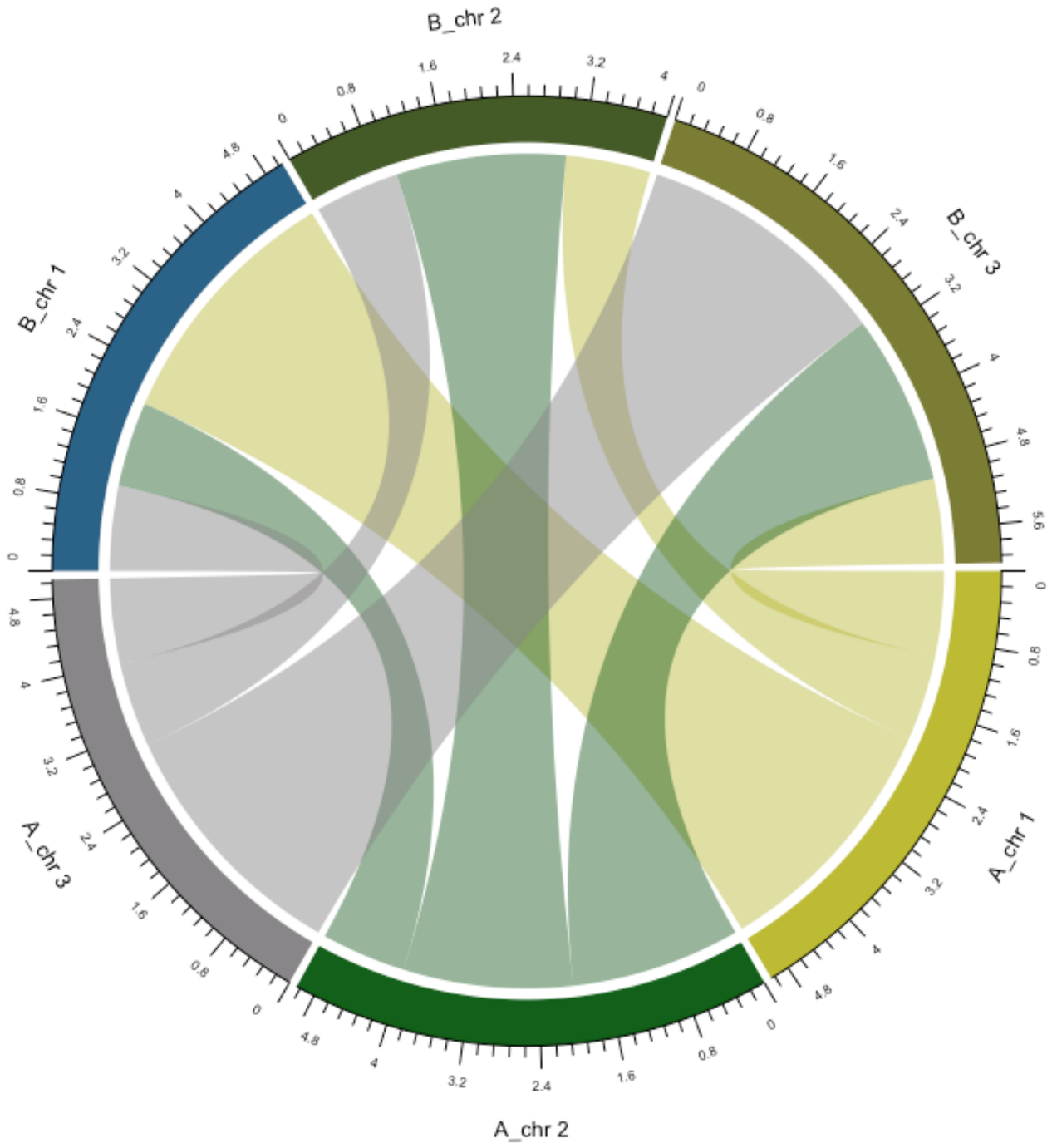
Note: comma-separated values (CSV) file

```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
```

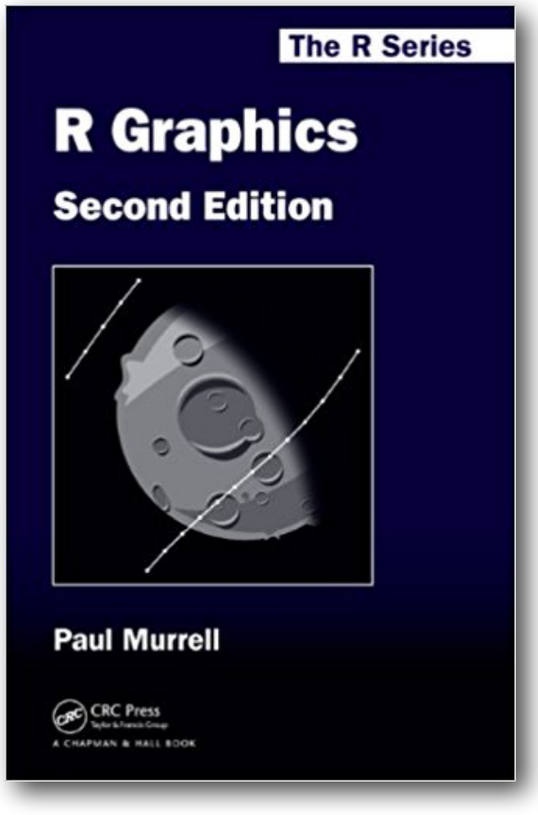
Example: Save and Restore R Session

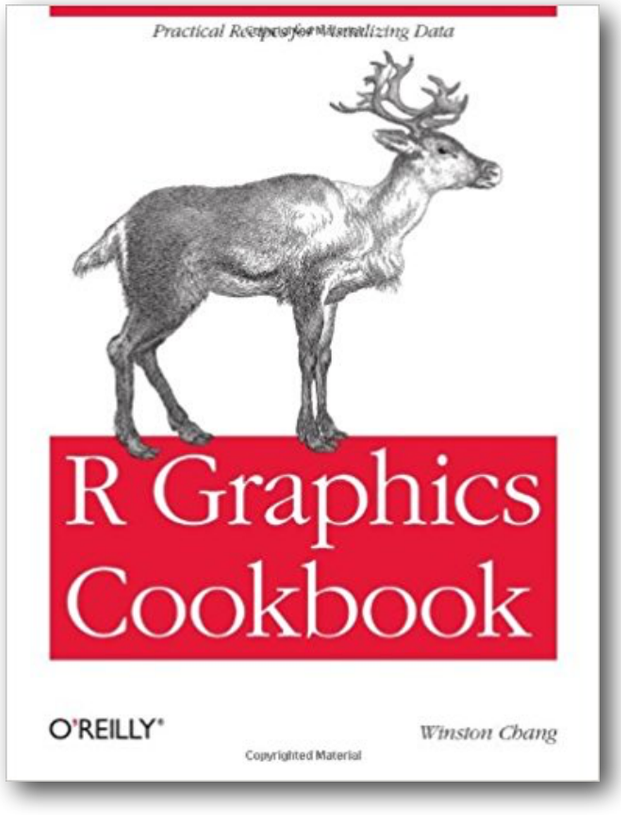
The **readr** package is a faster alternative to the base R analogues (e.g. *read.table* and *read.csv*). **readr** supports seven file formats with seven read_ functions:

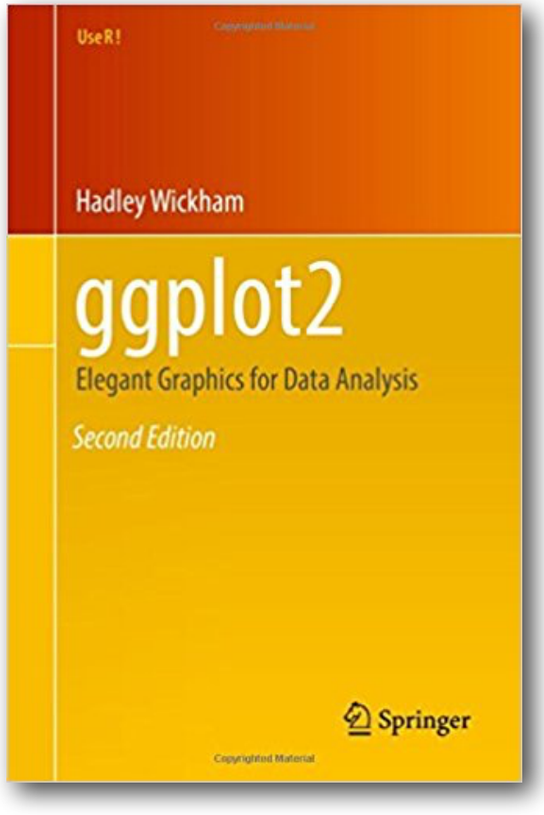

```
read_csv()    comma separated files
read_tsv()    tab separated files
read_delim()  delimited files
read_fwf()    fixed width files
read_table()  tab files separated by (white)space
read_log()    web log files
```



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


The R Series
R Graphics
Second Edition
Paul Murrell
CRC Press
A CHAPMAN & HALL BOOK

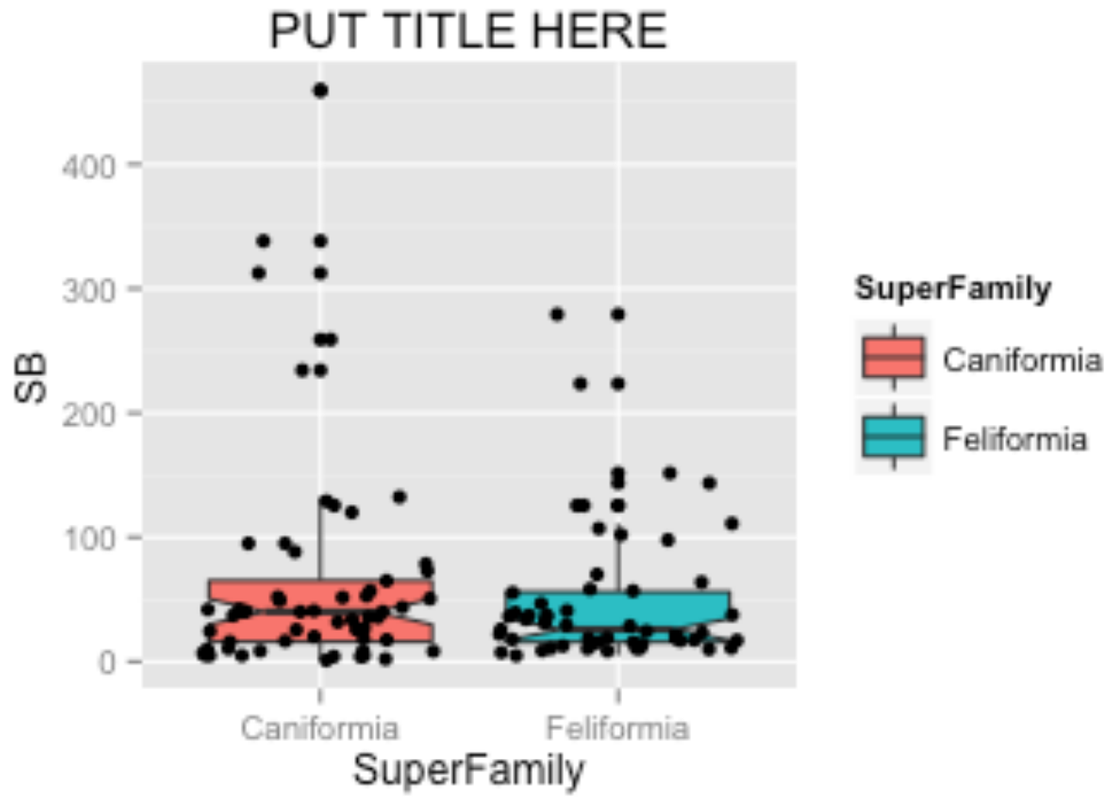

Practical Recipes for Visualizing Data
R Graphics Cookbook
O'REILLY
Winston Chang
Copyrighted Material


UseR!
Hadley Wickham
ggplot2
Elegant Graphics for Data Analysis
Second Edition
Springer
Copyrighted Material

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.

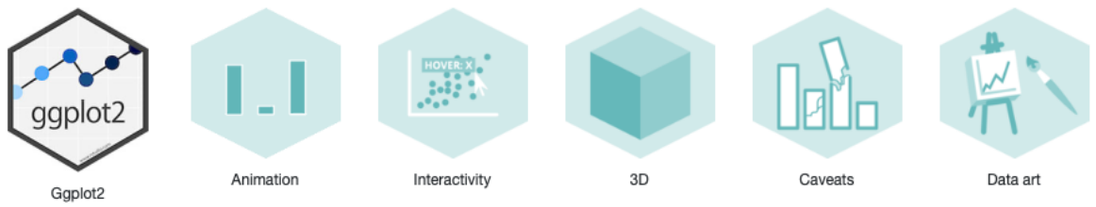
```
install.packages("ggplot2")
library(ggplot2)
p <- ggplot(data)
p <- p + geom_?
p
```

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



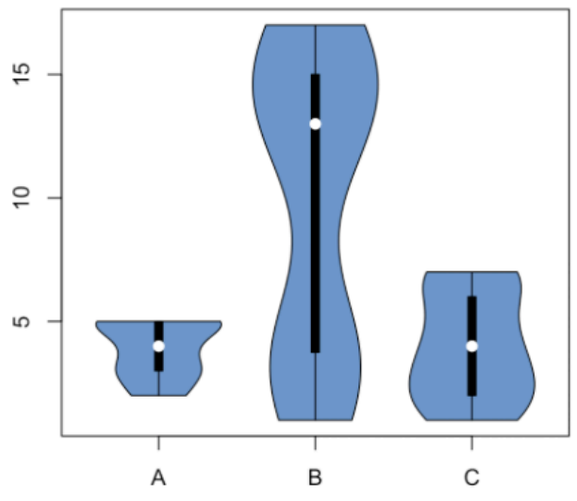

```
install.packages("ggplot2")
library(ggplot2)
p <- ggplot(carnivora, aes(SuperFamily, SB))
p <- p + geom_boxplot(notch = TRUE,aes(fill = SuperFamily))
p <- p + geom_jitter()
p <- p + labs(title = "PUT TITLE HERE")
p
```

The R Graph Gallery



The `Vioplot` library builds the violin plot as a boxplot with a rotated kernel density plot on each side. If you want to represent several groups, the trick is to use the `with` function as demonstrated below.

Note: consider using the ggplot2 package as shown in graph #95.




```r
# Load the vioplot library
library(vioplot)

# Create data
treatment <- c(rep("A", 40) , rep("B", 40) , rep("C", 40) )
value <- c( sample(2:5, 40 , replace=T) , sample(c(1:5,12:17), 40 , replace=T), sample(1:7, 40 , repl
data <- data.frame(treatment,value)

# Draw the plot
with(data , vioplot(
  value[treatment=="A"] , value[treatment=="B"], value[treatment=="C"],
  col=rgb(0.1,0.4,0.7,0.7) , names=c("A","B","C")
))
```

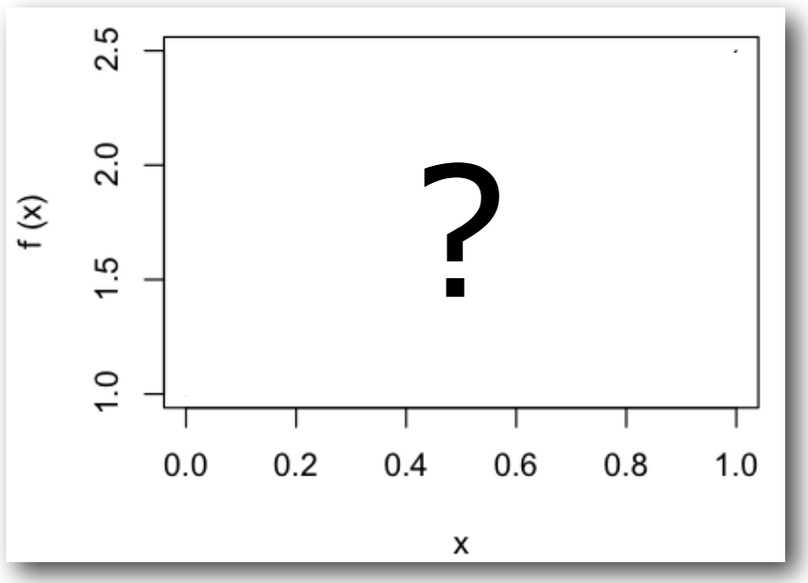
User Functions

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

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

$x = 2;\ y = ?$

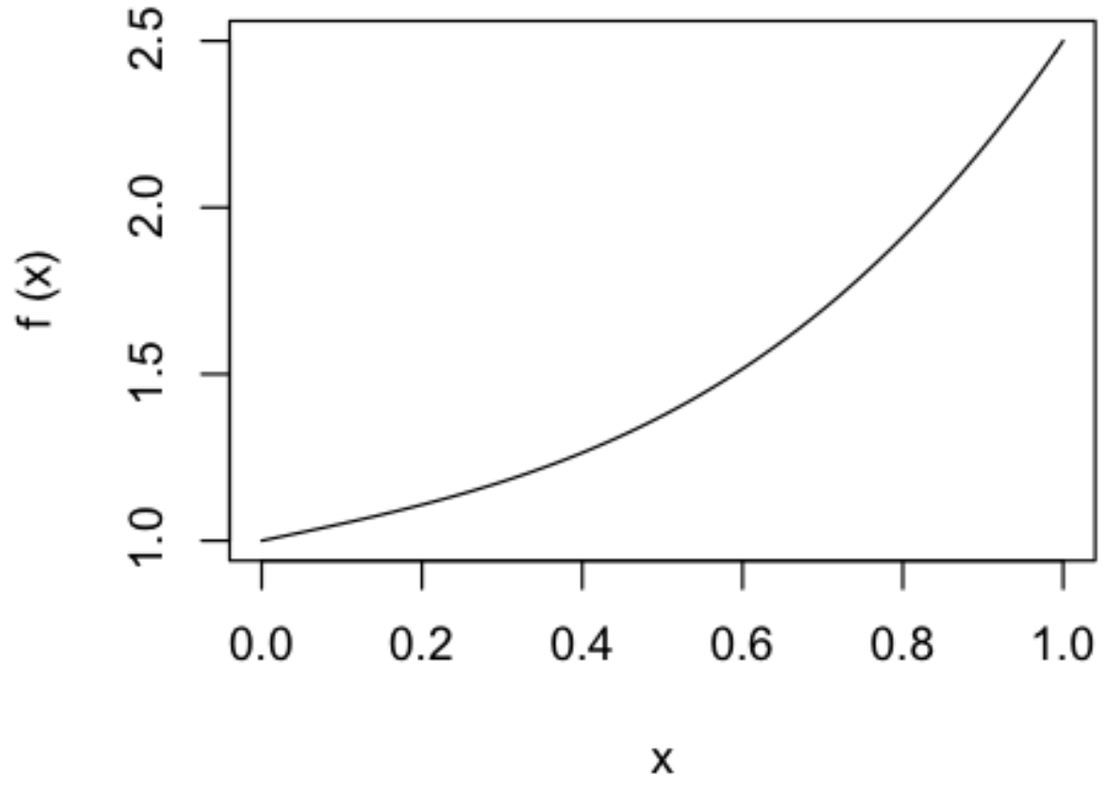
$x = 10;\ y = ?$



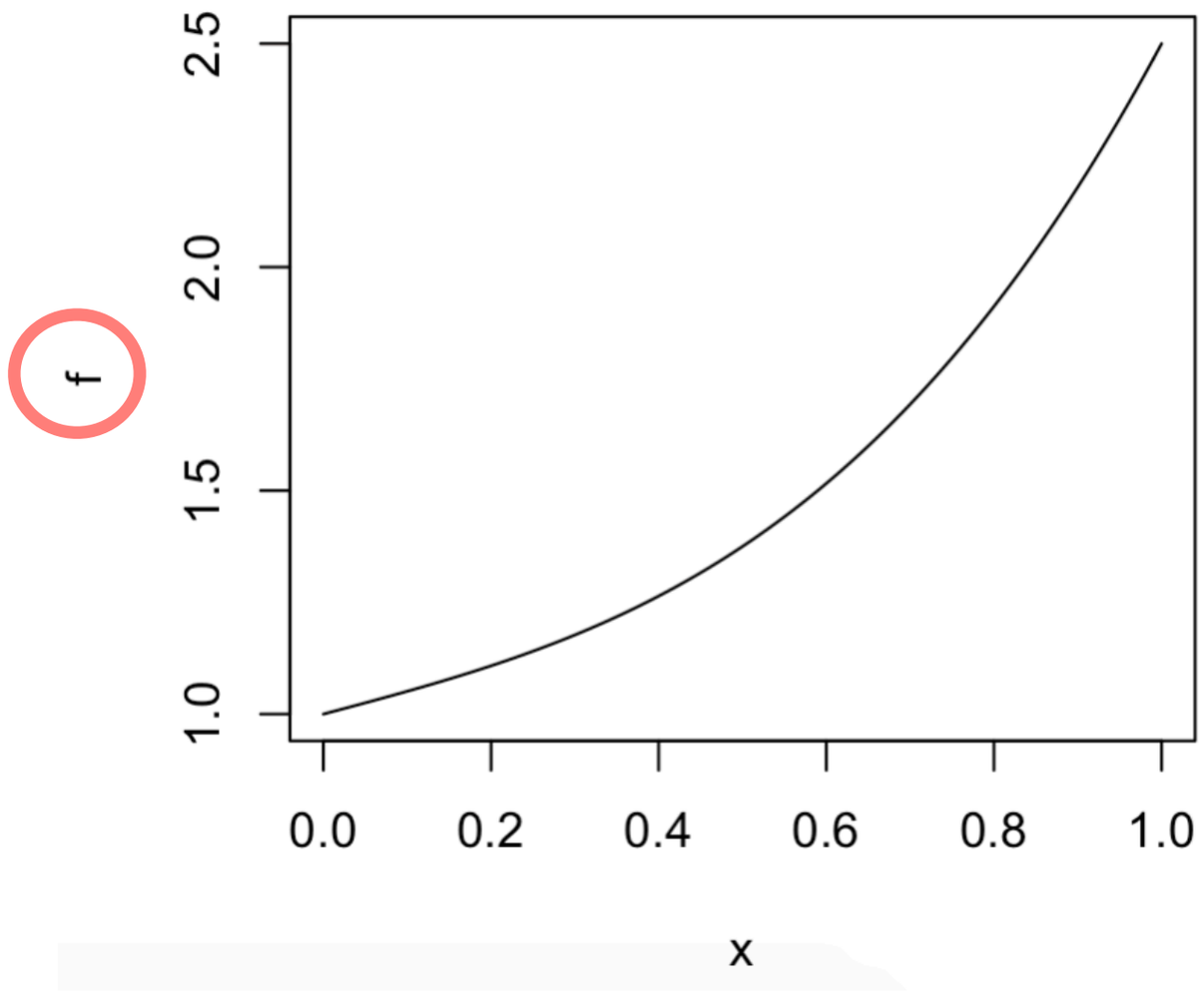
```
> x <- 2
> x^3 + x/2 + 1
[1] 10
> x <- 10
> x^3 + x/2 + 1
[1] 1006
```

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

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



```
> f <- function(x) {
  return( x^3 + x/2 + 1 )
  }
> f(c(2, 10))
[1] 10 1006
> plot(f)
```



```
> f <- function(){
        cat("Hello, world!\n")
}
> f()
Hello, world!
```

Function

```
> f <- function(name){
        cat("Hello,", name,"!\n")
}
> args(f)
function (name)
NULL
> f("Me")
Hello, Me!
> f()
Error in cat("Hello,", name, "!\n") :
  argument "name" is missing, with no default
```

Function with (user) input (argument)

```
> f <- function(name = "World"){
        cat("Hello,", name,"!\n")
}
> f()
  Hello, World!
> f("Globi")
  Hello, Globi!
> f(Globi)
Error in cat("Hello,", name, "!\n") : object
'Globi' not found f(Globi)
```

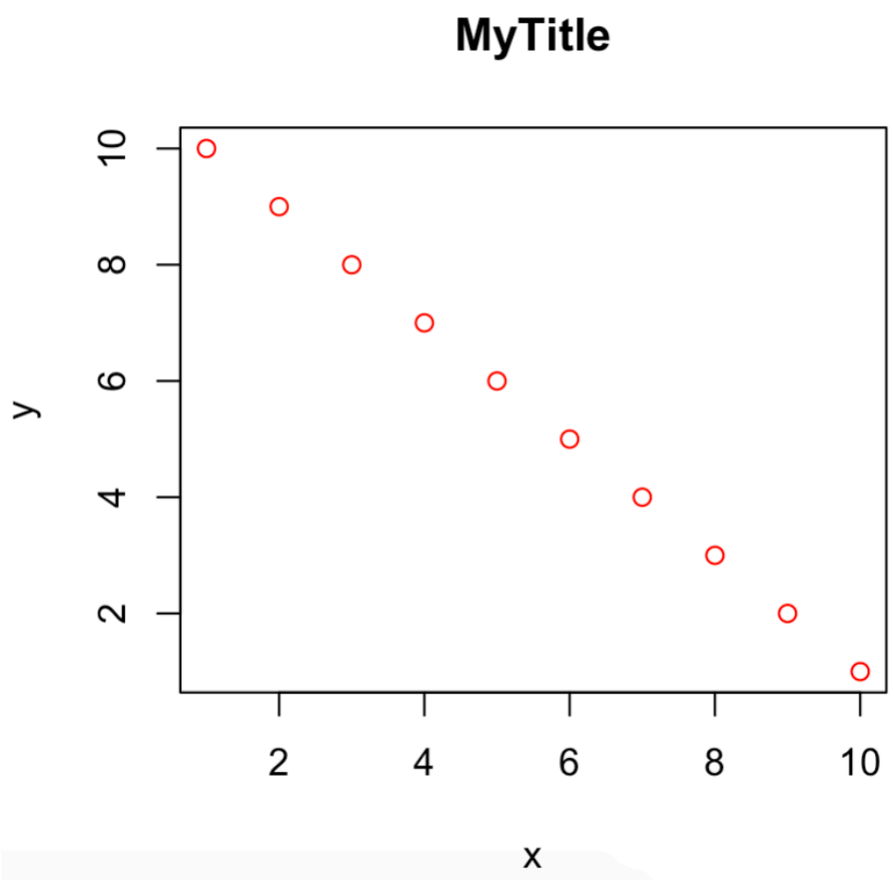
Function with default arguments

```
> f <- function(name = NULL){
        cat("Hello,", name,"!\n")
}
> f()
  Hello, !
```

Function with argument NULL

```
> x <- 1:10
> y <- 10:1
> f <- function(x, y, ...){
 plot(x, y, ...)
}
> f(x, y)
> f(x, y, main = "MyTitle", col = "red")
```

Nested arguments



```
> f <- function(x,y){
        x^2
  }
> f()
> f(2)
> f(2,4)
```

?

Unused arguments
(lazy evaluation)

```
> f <- function(x,y){
        x^2
  }
> f()
Error in f() : argument "x" is
missing, with no default
> f(2)
[1] 4
> f(2,4)
[1] 4
```

Unused arguments
(lazy evaluation)

```
> f <- function(x, y) {
          x + y + z
  }
> f(1,2)
> f(1,2,3)
```

Unused arguments
(lazy evaluation)

```
> f <- function(x, y) {x + y + z}
> f(1, 2)
Error in f(1, 2) : object 'z' not
found
> f(1, 2, 3)
Error in f(1, 2, 3) : unused
argument (3)
> z <- 3
> f(1, 2)
[1] 6
```

Environmental arguments

Write a function that converts Fahrenheit to Celsius or Celsius to Fahrenheit.

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

$$C = \frac{5}{9}(F - 32)$$

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,
           av = mean(x),
           me = median(x))
           }
> x <- 1:9
> avme(x, "av")
> avme(x, "me")
```

Two Functions

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

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

Idea:

Combined Solution

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

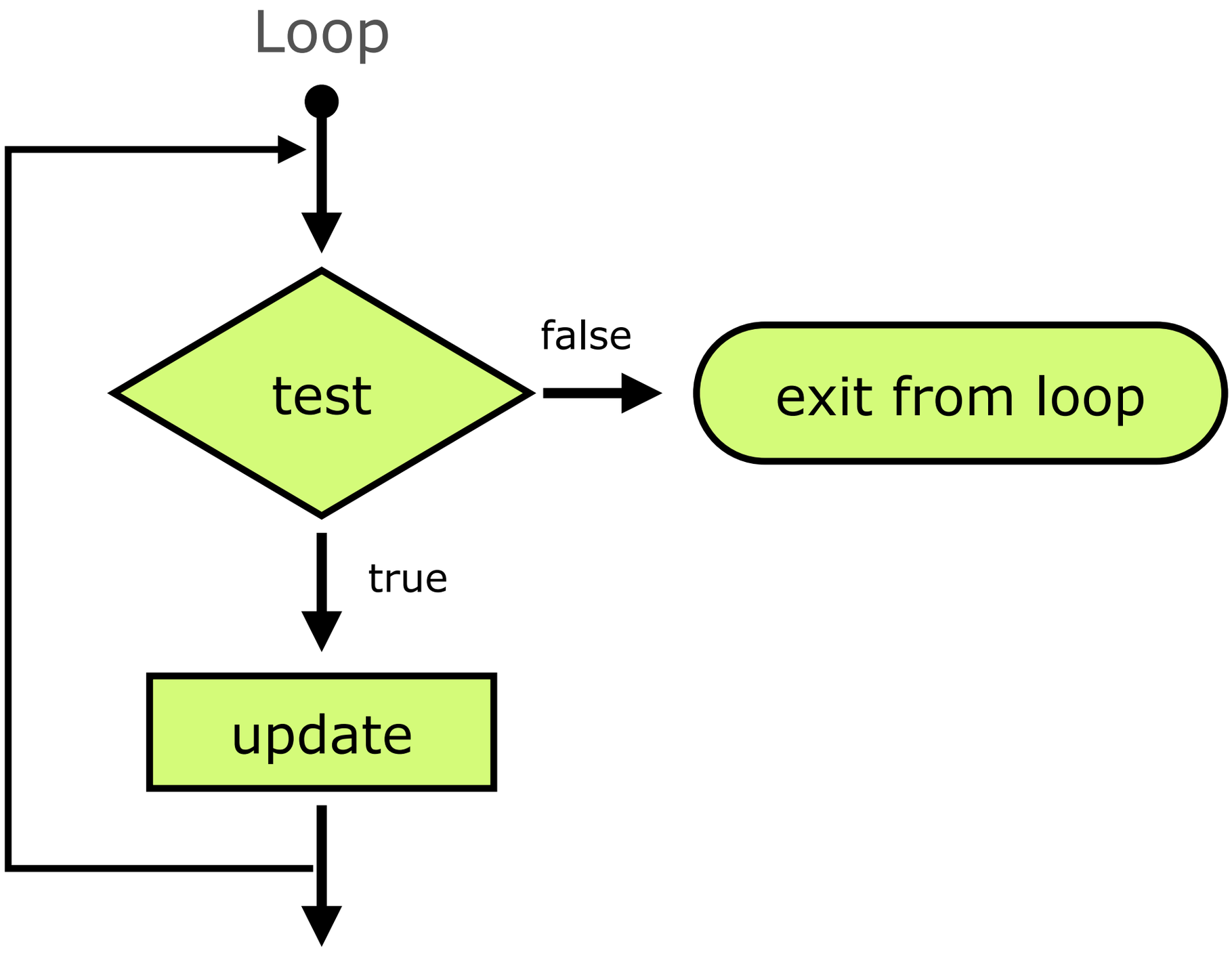
```
> require(stats)
> hot <- function(x, type) {
  switch(type,
         f2c = ((x-32)*5/9),
         c2f = (9/5*x+32))
         }
> x <- 25
> hot(x, "f2c")
> hot(x, "c2f")
```

```
> 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")
```

Default

```
> test <- function(V1, V2){
   x <- V1 + V2
   x
}

> test(3,4)
[1] 7
> x
Error: object 'x' not found
```

```
> test <- function(V1, V2){
  x <<- V1 + V2 # Super-Assigned

  x
}

> test(3,4)
[1] 7
> x
[1] 7
```


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
[1] 2
[1] 3
```

```
> for(i in letters) print(i)
[1] a
[1] b
[1] c
[1] …
[1] z
```

Control Flow (looping)

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

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

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

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

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

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

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

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

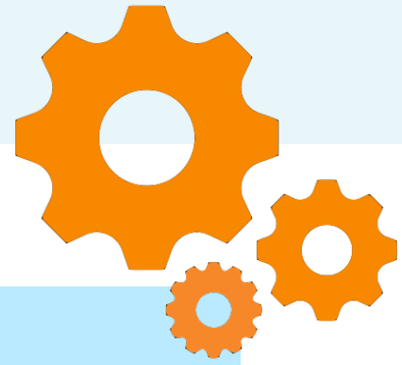
```
> for(i in 1:3) cat(i, "\n")
1
2
3
```

```
> 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
> for(i in 1:3) cat(WL[4-i], "")
Text Some Output
```

```
> 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).

A is reset to zero. We loop three times …

#1: j=10 - **A** is assigned the value 10 + 0 = 10.

#2: j=20 - **A** is assigned the value 20 + 10 = 30.

#3: j=30 - **A** is assigned the value 30 + 30 = 60.

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

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.**

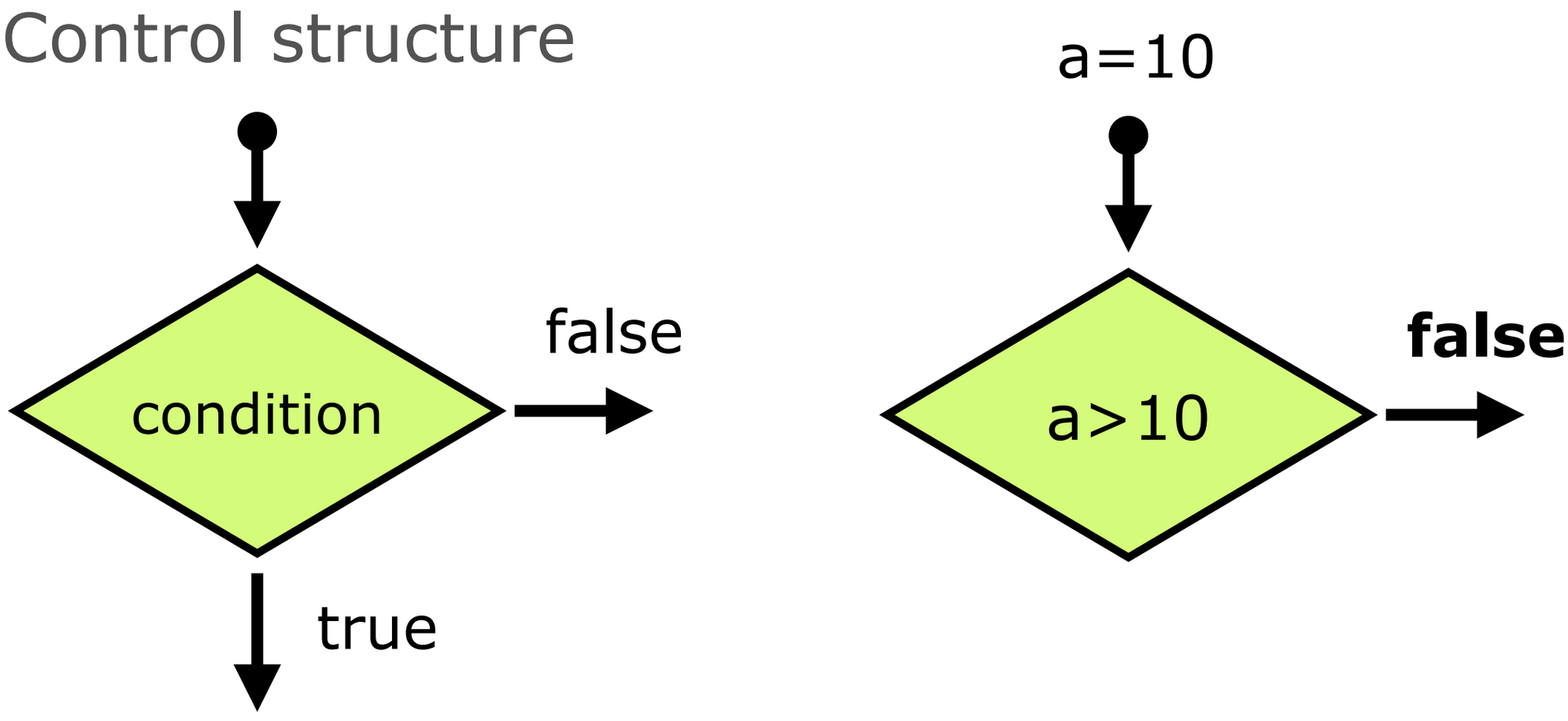
Multiply all the numbers from 1 to 50 in a **loop.**

```
> 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
```

```
# Celsius to Fahrenheit
for (celsius in 25:30)
  print(c(celsius, ">" ,
          9 / 5 * celsius + 32),
        digits = 1,
         quote = FALSE)
[1] 25 > 77
[1] 26 > 79
[1] 27 > 81
[1] 28 > 82
[1] 29 > 84
[1] 30 > 86
```


Control structure
condition
false
true
a=10
a>10
false

```
> x <- 10 # let x be 10
> x <  10 # operation for a variable
FALSE
> x <- c(8:12)
> x < 10  # operations for a vector
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 = a+1 else a = a-1
> a
[1] 3
> if(a < 3) a+1 else a-1
> a
[1] 2
```

```
> x <- seq(-1,5)

> x

# [1] -1 0 1 2 3 4 5

> x < 0

# [1]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
> if(x < 0) (x*2) else (x^2)

# [1] -2  0  2  4  6  8 10

Warning message:

In if (x < 0) (x * 2) else (x^2) :

  the condition has length > 1 and only the first element will be used

> if(any(x < 0)) (x*2) else (x^2)

# [1] -2  0  2  4  6  8 10

> if(all(x < 0)) (x*2) else (x^2)

# [1]  1  0  1  4  9 16 25
```

a loop

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

if-else statment in a loop

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

[1] 6.283185
[1] 15.70796
```

if-else statment in a loop with long output for e.g. troubleshooting

```
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
```

if-else "search" with text output

```
> 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
```

if-else "search" with text output

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

```
list contains Hello
```

```
> if((sum(y == "Hello") > 0))
  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
```

messages

```
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
      # Error
      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

> 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
```

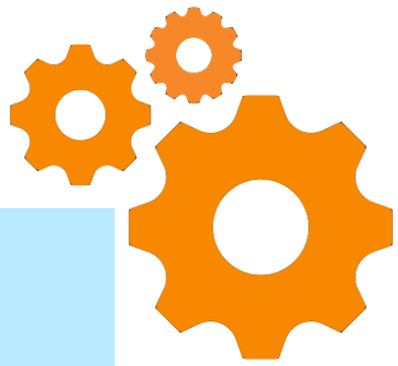
```
> sqrt(ifelse(x >= 0, x, NA))
NA 1.414214
```


MORE
THINGS
CONSIDERED

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")
```



```
# 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

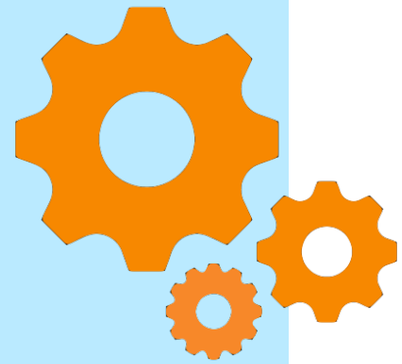
ape provides functions for reading and manipulating phylogenetic trees and DNA sequences, computing DNA distances, estimating trees with distance-based methods, and a range of methods for comparative analyses and analysis of diversification. Functionalities are also provided for programming new phylogenetic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

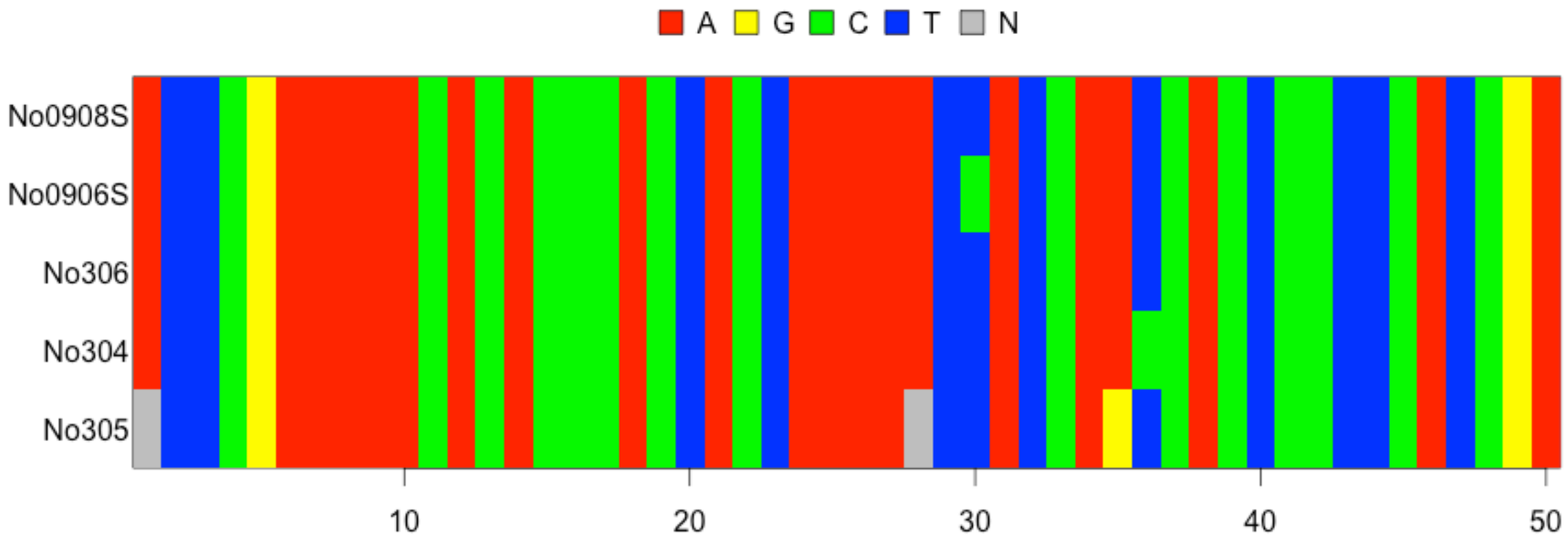
```
> library(ape)
# Get sequences from Genebank
> ref <- ("AF502402")
> Deug_adh <- read.GenBank(ref)
```

```
### 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
```

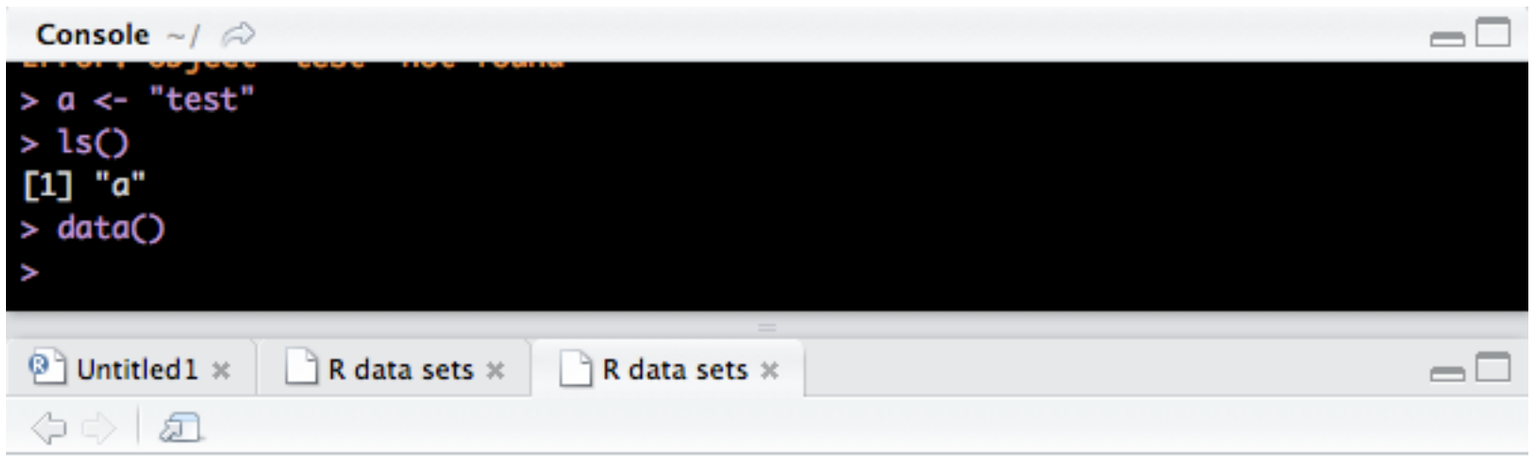
```
> ref1 <- (c("NM_169441", "NM_080059")
> Dmel_Hsp70 <- read.GenBank(ref1)
> Dmel_Hsp70
2 DNA sequences in binary format stored in a list.

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
```



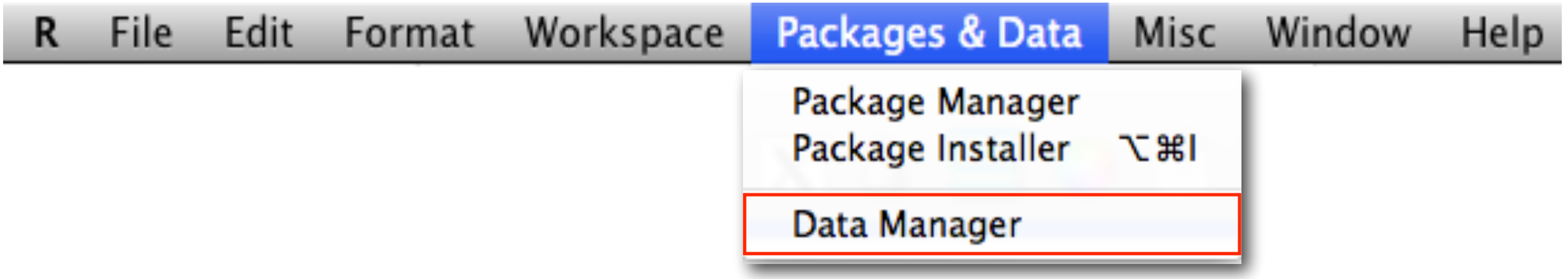
```
> data(woodmouse)
> image(woodmouse[1:5,1:50])
```




Data

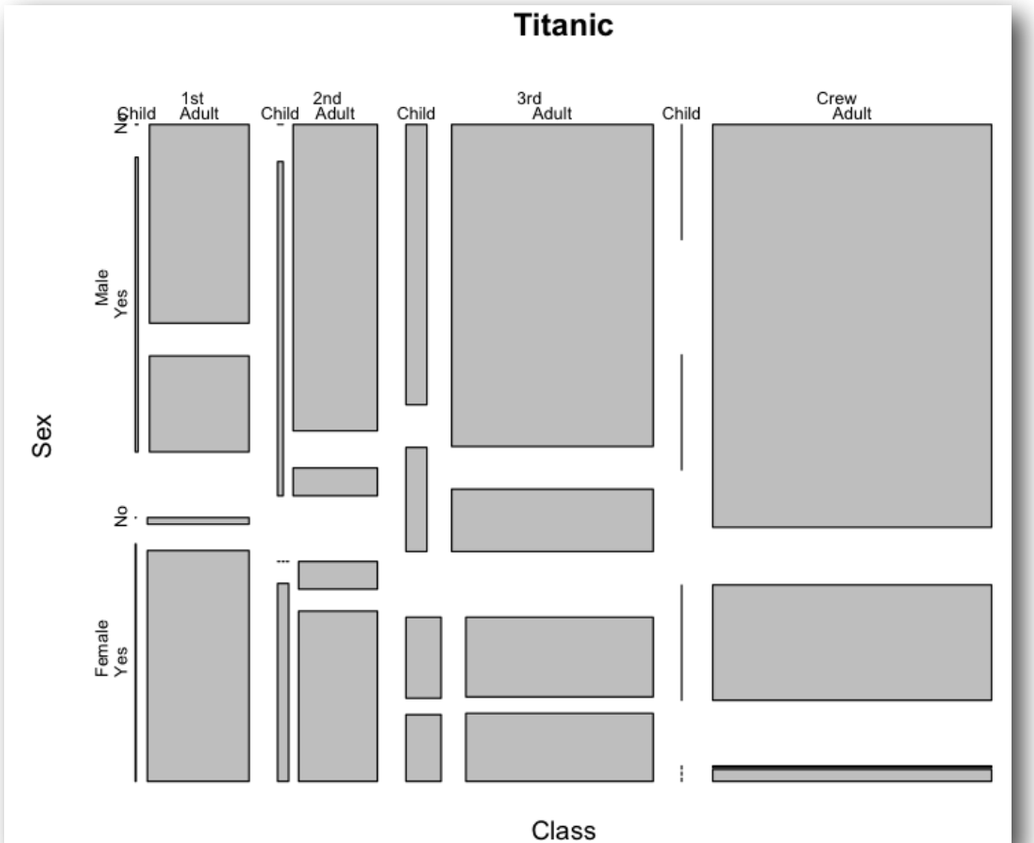



```
Data sets in package 'datasets':

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
```



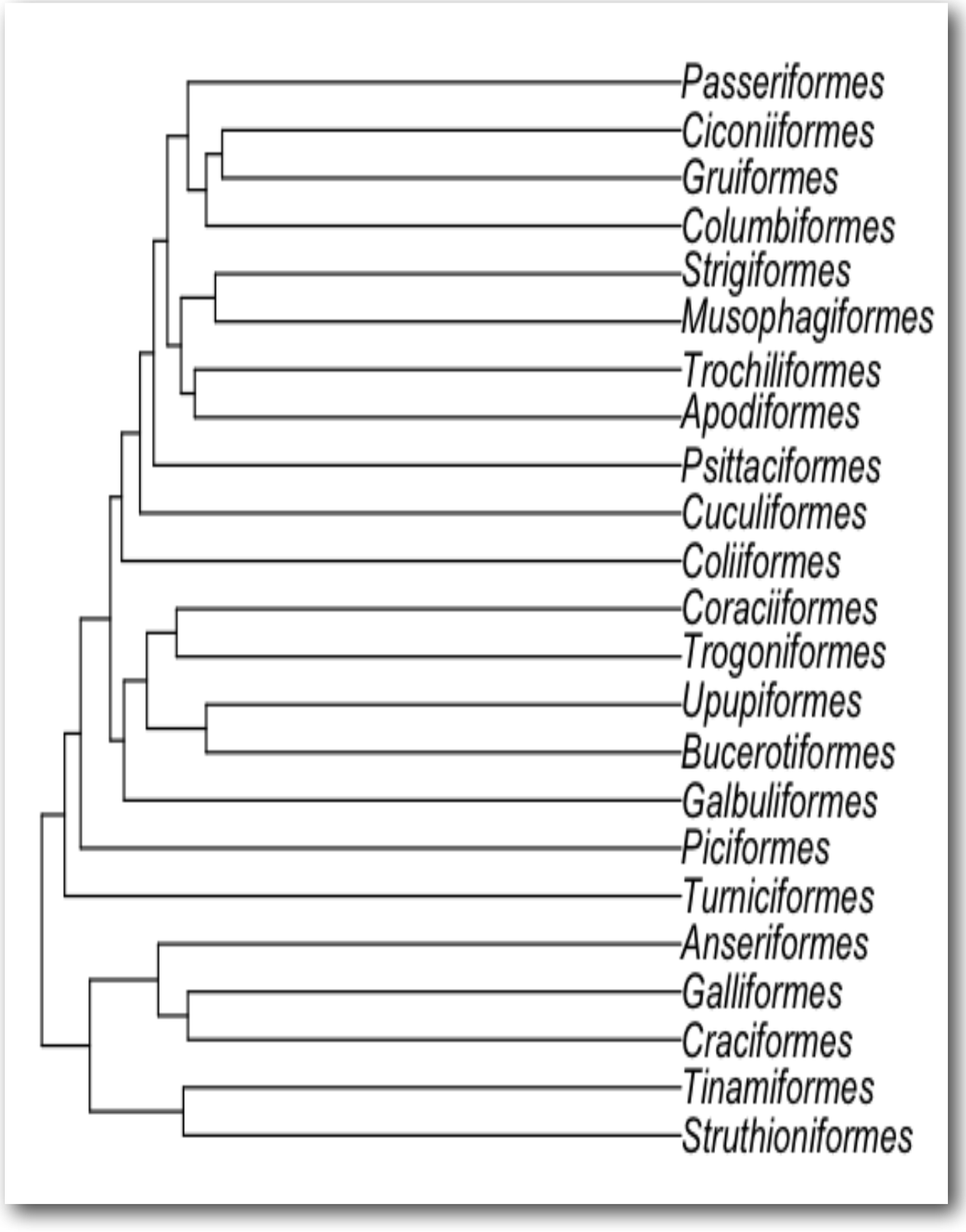
Data	Package	Description
AirPassengers	datasets	Monthly Airline Passenger Numbers 19
BJsales	datasets	Sales Data with Leading Indicator
BJsales.lead (BJs	datasets	Sales Data with Leading Indicator
BOD	datasets	Biochemical Oxygen Demand
CO2	datasets	Carbon Dioxide Uptake in Grass Plants
ChickWeight	datasets	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

```
> 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
```

```
> 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
```

```
> 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	<=