```
### META INFO
# Contacts:
# davide.eynard@gmail.com
# http://davide.eynard.it


### Installation
# Download binaries from http://cran.r-project.org/ or
https://cran.rstudio.com
# or, on linux, just get the base packages (MASS can also be installed
that way)

# MAC OS : if you have problems with locales, check this out:
# https://cran.r-project.org/bin/macosx/RMacOSX-
FAQ.html#Internationalization-of-the-R_002eapp
# from the terminal: defaults write org.R-project.R force.LANG
en_US.UTF-8
# from R itself: system("defaults write org.R-project.R force.LANG
en_US.UTF-8")

# TODO: also get RStudio: https://www.rstudio.com/, we are going to
use it
#       during our labs!

# NOTE: the system is case sensitive!
# NOTE: some objects only exist as nested, i.e. month.name exists but
month will
#       tell "object month not found"
#       => use tab completion as an aid to entering function names!!!


### Install new packages:
# try with ISLR and MASS
install.packages() # opens package installation window in R - does
not work in RStudio
install.packages('packagename')

# ... check if the package has been installed correctly (also used to
load the package)
library()
library('packagename')
search()
ls("package:packagename")
# also see http://r-pkgs.had.co.nz/namespace.html

# look for the paths where libraries are stored
.libPaths()

# Look for other datasets/code related to the book:
# http://www-bcf.usc.edu/~gareth/ISL/data.html
```

```
### Looking for help
?plot
help.search("regression")
help(package="packagename")
example(seq)

### Set up the working directory
setwd('~/Desktop')

### Options
pi
defaults = options()
options(digits=15)
pi
options(defaults)
pi

### VARIABLE ASSIGNMENTS:
a <- 'whatever'
a = 'whatever'
a = 5
a = c(1,2,3)
a = c(1:10)

# ==> the seq() function
# actually, 1:10 is a shorthand for seq(1,10)
# we can just write a = 1:10
x = seq(-pi,pi,length=100)
x = seq(-pi,pi,by=.1)
x = seq(-pi,pi,along=y)

# rep
x = rep(1:3, each=3)
x = rep(1:3, times=2)

# unique
unique(x)

### managing variables
ls()
ls(all.names=TRUE)
rm(name)
rm(list=ls()) # removes all


### operations on vectors
# note that operations are performed element wise

### CORRELATION example:
# rnorm initializes randomly, with mean=0 and sd=1 if not specified
differently
x = rnorm(50)
y = x + rnorm(50,mean=50,sd=.1)
cor(x,y)
```

```
# note that cor(x,y) = cov(x,y)/(sdx*sdy)
# cov(x,y) = mean((x-mean(x))*(y-mean(y)))
# sd(x) = sqrt(mean((x-mean(x))^2))

# why is sd calculated dividing by n-1 instead of n?
# see http://mathworld.wolfram.com/StandardDeviation.html
# and https://en.wikipedia.org/wiki/Bessel%27s_correction
# tell something about variance / sd - use the following as an example
# of operations on vectors:
set.seed(123)
a = rnorm(10)
sqrt(sum((a-mean(a))^2)/(length(a)-1))
sd(a)
# var(x) is defined as corrected_mean((x-mean(x))^2)


### MATRICES:
?matrix
x = matrix(c(1,2,3,4),2,2)
a = matrix(c(1:4),2,2,byrow=TRUE)
b = matrix(c(5:8),2,2,byrow=TRUE)
a * b
a%*%b

#      [,1] [,2]
# [1,]   19   22
# [2,]   43   50

v = c(1,2)
v %*% a

#      [,1] [,2]
# [1,]    7   10

# NOTE that if you don't treat v as a matrix you will be able to
# multiply it both from left and from right, with no errors (the
# interpreter will try to flip it the way it makes sense

a %*% v

#      [,1]
# [1,]    5
# [2,]   11

# NOTE that the vector might be changed into vertical/horizontal
# to make the two arguments conformable.

as.matrix(v) # vectors (horizontal) become vertical by default

#      [,1]
# [1,]    1
# [2,]    2
```

```
as.matrix(v) %*% a
# error

t(as.matrix(v)) %*% a
#      [,1] [,2]
# [1,]    7   10


### SAVING TO FILE:
x = rnorm(50)
y = x + rnorm(50,mean=50,sd=.5)
pdf('~/Desktop/test.pdf')
plot(x,y,main='main',xlab='xlab',ylab='ylab')
dev.off()

### 3D PLOTTING:
x = seq(-pi,pi,length=50)
y = x
f = outer(x,y,function(x,y)cos(y)/(1+x^2))
contour(x,y,f)
contour(x,y,f,nlevels=45)
fa = (f-t(f))/2
contour(x,y,fa,nlevels=45)
image(x,y,fa)
persp(x,y,fa)

### REFERENCE ELEMENTS OF A VECTOR/MATRIX
x = c(4, 7, 2, 10, 1, 0)

x[2]
x[1:3]
x[c(1,3,4)]
x[-3]
x[-c(2,3)]

# show these also work with matrices
a = matrix(1:100, 10 ,10)
a[1,5]

# [1] 41

a[c(1,3,4),c(2,4,6)]

#      [,1] [,2] [,3]
# [1,]   11   31   51
# [2,]   13   33   53
# [3,]   14   34   54

### functions that return indices
which
which.max
which.min
match
```

```
### check for wrong stuff
# show 0/0, Inf/Inf, whatever/0, NA, Nan, Inf
a = rnorm(10)
b = rnorm(10)
a[3] = 0
a[5] = Inf
b[3] = 0
b[5] = Inf
b[7] = 0
b[9] = NA
c = a/b

is.infinite(c)
is.nan(c)
is.na(c)
any(is.na(c))
which(is.na(c))
na.omit(c)

### load data
Auto = read.table('Auto.data')
fix(Auto)

Auto = read.table('Auto.data',header = T, na.strings="?")
Auto = na.omit(Auto)
dim(Auto)
names(Auto)


### additional graphical and numerical summaries
plot(cylinders, mpg)
plot(Auto$cylinders, Auto$mpg)
attach(Auto)
plot(cylinders, mpg)
cylinders=as.factor(cylinders)
plot(cylinders, mpg)
plot(cylinders, mpg, col="red")
plot(cylinders, mpg, col="red", varwidth=T)
plot(cylinders, mpg, col="red", varwidth=T,horizontal=T)
hist(mpg)
hist(mpg,col=2)
hist(mpg,col=2,breaks=15)
pairs(Auto)
pairs(~ mpg + displacement + horsepower + weight + acceleration, Auto)
plot(horsepower,mpg)
identify(horsepower,mpg,name)
summary(Auto)
summary(mpg)
```