

# The Basics

H. Bjørn Nielsen  
R-introduction workshop

# Assignment

The “<-” operator equals “=”

```
> a <- 3 # Assigns “3” to the  
variable “a”
```

and evaluating the variable:

```
> a # returns the content of “a”:
```

```
[1] 3
```

# Operators

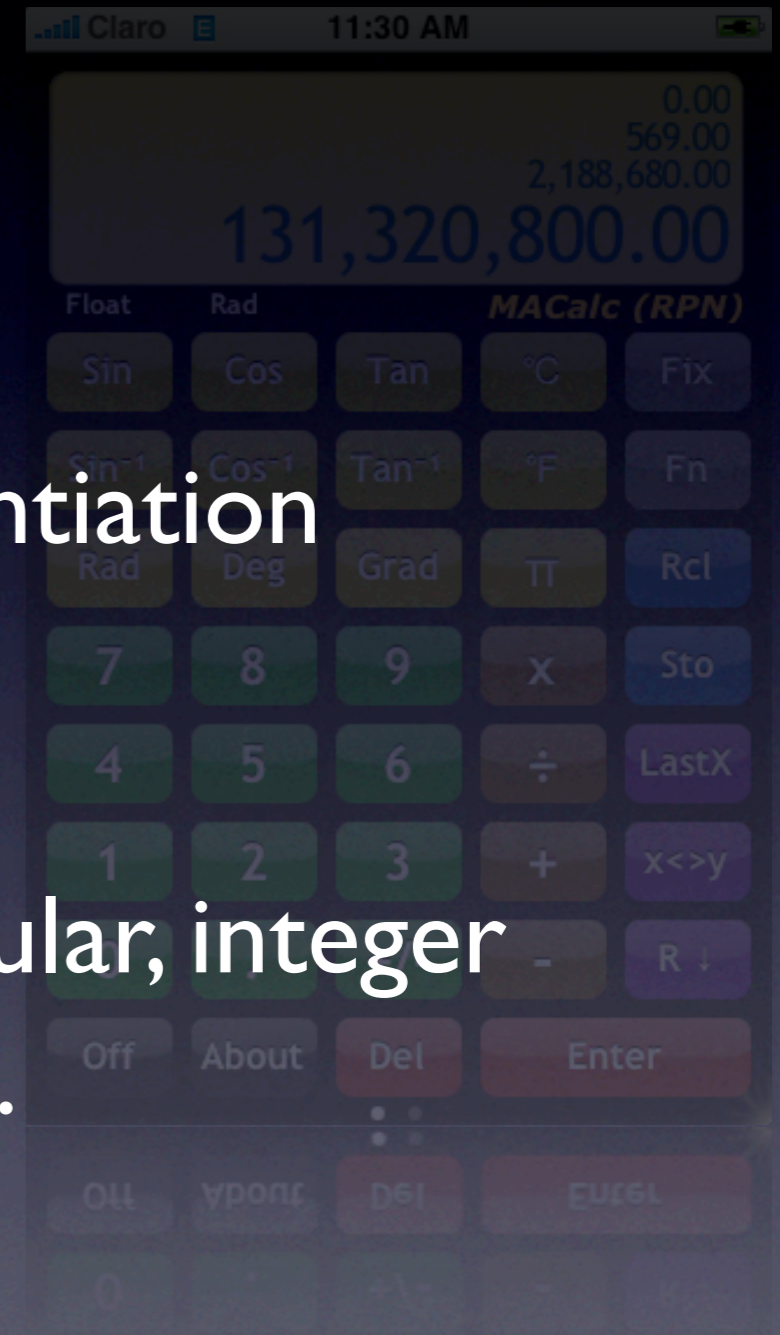
No surprises here

$+$   $-$   $*$   $/$   $^$  means:

$+$ ,  $-$ , multiply, division and exponentiation

But these may be surprising

$\% \%$ ,  $\% / \%$ ,  $\% * \%$  means: modular, integer  
division and matrix multiplication.



# Functions

Functions have parenthesis

```
function_name(arguments)
```

**Examples:**

```
mean(), cos(), sqrt(), rma() and  
myfunction()
```

# Functions II

## Functions take arguments

```
> sqrt(10)
[1] 3.162278
```

## Arguments may come ordered or named:

```
> rnorm(10, 2, 3) # equals
> rnorm(10, mean=2, sd=3)
```

## And some have defaults

```
rnorm(10) equals: rnorm(10, mean=0, sd=1)
```

# Lots of functions

## Examples

**Basics:** `mean()`, `sd()`, `seq()`, `cos()`,  
`sum()`, `c()`, `t()`, ...

**Graphics:** `plot()`, `hist()`, `barplot()`,  
`image()`, `pie()`, `contour()`, ...

**Statistics:** `t.test()`, `wilcox.test`,  
`chisq.test()`, `anova()`, ...

**Distributions:** `runif()`, `rnorm()`, `rbinom`,  
`density()`, `quantile()` ...

# Important data structures

**vector:** series of instances of same mode / class

**matrix:** table containing items of same mode

**data.frame:** a table that may contain columns of different class

**list:** series of objects (in fact a vector of mode list)

**function:** a function

**formula:** “describes relations between variables”

# Important data structures

**vector:** series of instances of same mode / class

**matrix:** table containing items of same mode

**data.frame:** a table that may contain columns of different class

**list:** series of objects (in fact a vector of mode list)

**function:** a function

**formula:** “describes relations betw

What is a mode?

Examples:

logical: TRUE, FALSE

numeric: 1, 2.4, 500

character: “GeneA”, “GeneB”, “GeneC”

integer: 1, 2, 3, 4

# Vector

**How to make one (examples):**

```
v1<-vector(length=10) # define one
```

```
v2<-c("a","b","c","d","e") # cat some items
```

```
v3<-1:10 # a sequence, short for seq(1,10)
```

```
v4<-rnorm(10) # some functions return vectors
```

```
v5<- v4 > 0 # evaluate another vector
```

```
v6<-my.matrix[1:10,1] # subset another object
```

# Functions that take a Vector

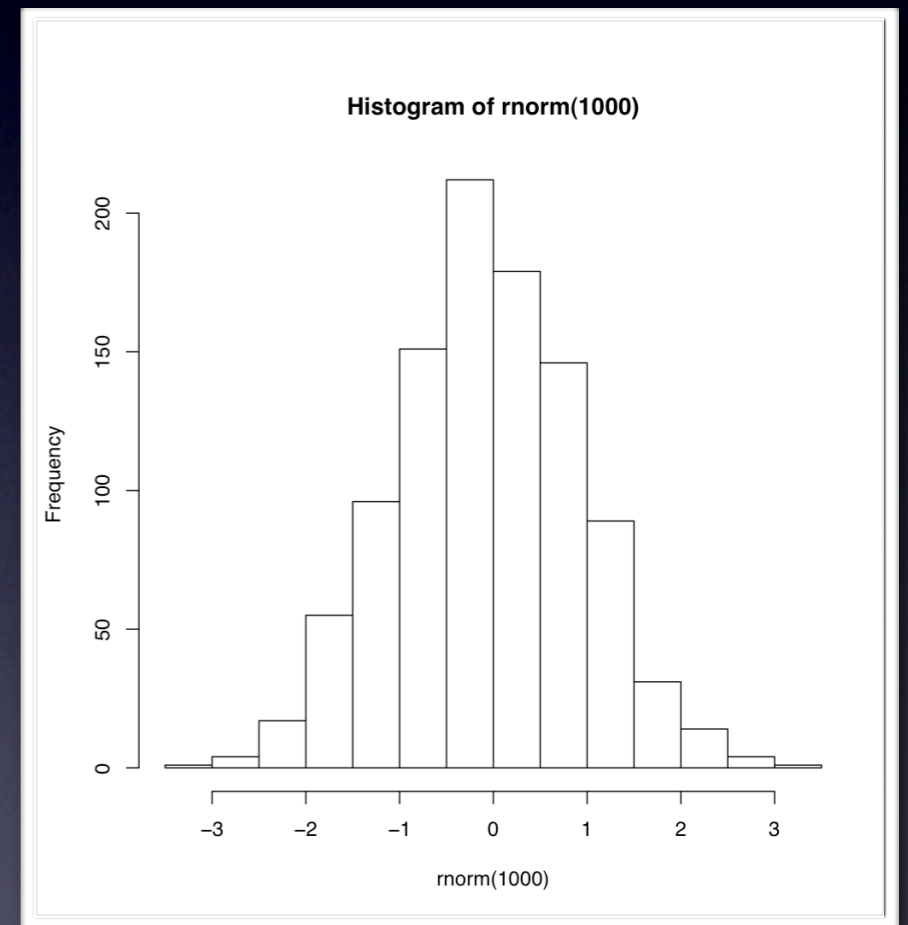
## Examples:

`str()`, `range()`, `max()`,  
`min()`, `summary()`,

`mean()`, `sum()`, `sd()`,  
`diff()`, `comsum()`, ...

`cbind()`, `matrix`,  
`factor()`, ...

`hist()`, `density()`,  
`plot()` ...



# Matrix

How to make one (examples):

```
m1<-matrix(1:20, nrow=5, ncol=4) # define one
```

```
m2<-cbind(v3, v4, v6) # cat some vectors (or rbind)
```

```
m3<-cor(m2) # some functions return a matrix
```

```
m4<-m1[1:4, 1:2] # subset another object
```

A Matrix is:

A table with items of identical mode, eg.  
*logical, numeric, character or factor*

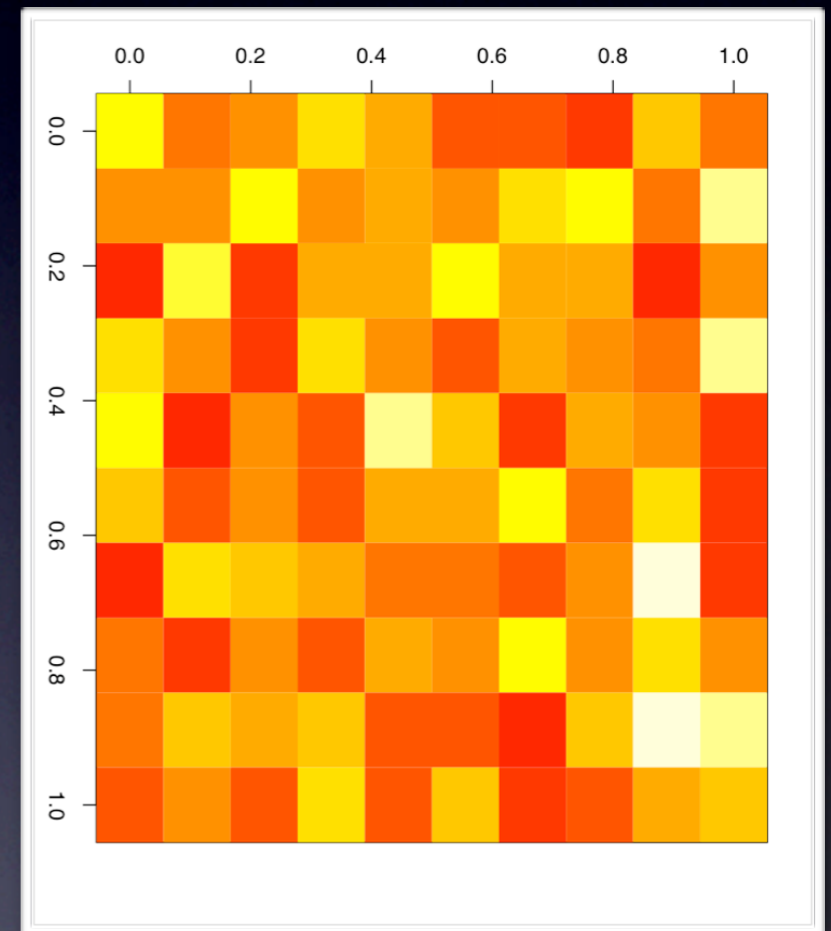
# Functions that take a Matrix

## Examples:

```
str(), summary(),  
rowMeans(), colMeans(),  
rowSums(),  
cor(), apply() ...
```

```
as.vector,  
as.data.frame, ...
```

```
image(), heatmap(),  
plot() ...
```



# data.frame

How to make one (examples):

```
df1<-data.frame(name=v1,mark=v2,course=v3)# define
```

```
df2<-read.table(file="myfile") # read from file
```

```
df3<-df1[,c("name","course")] # subset another object
```

A Data Frame is:

A table that may have columns of different modes

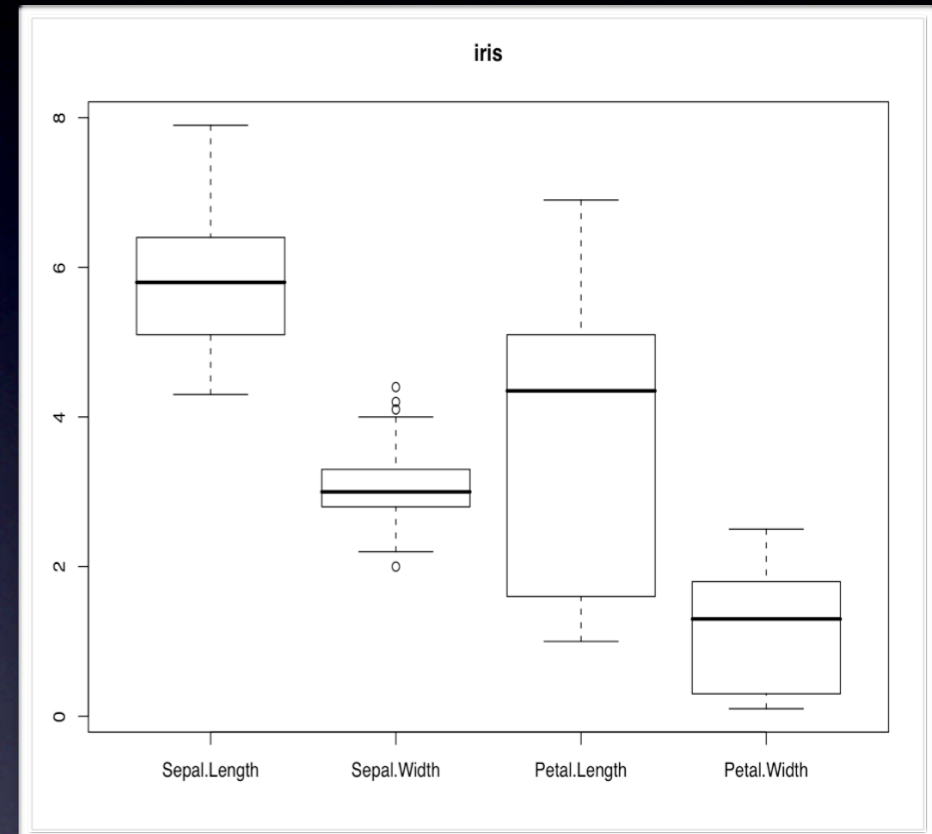
# Functions that take a data.frame

## Examples:

```
str(), summary(), anova(),  
lapply, ...
```

```
as.list, as.matrix,  
write.table, ...
```

```
boxplot, plot(), ...
```



# List

How to make one (examples):

```
L1<-list(marks=df1, letters=v2)# define one
```

```
L2<-t.test(v4) # many functions return a list
```

```
L3<-L2[c("statistic", "p.value")]# subset another list
```

A List is:  
a concatenated series of objects

# Functions that take a list

Examples:

`str()`, `summary()`, `lapply`, ...

`unlist()`,

# Function

How to make one (examples):

```
my.topnames.f<-function(v, top=100) {  
  return(names(rev(sort(v)) [1:top]))  
}
```

# str()

Get the “structure” of an object (vector)

```
str(euro)
```

```
Named num [1:11] 13.76 40.34 1.96 166.39 5.95 ...  
- attr(*, "names")= chr [1:11] "ATS" "BEF" "DEM"  
"ESP" ...
```

# str()

Get the “structure” of an object (matrix)

```
str(M)
```

```
  num [1:4, 1:4]  0.77210 -1.82619 -0.41742  
-0.00684 -1.66300 ...
```

```
- attr(*, "dimnames")=List of 2
```

```
 ..$ : chr [1:4] "a" "b" "c" "d"
```

```
 ..$ : chr [1:4] "a" "b" "c" "d"
```

# str()

Get the “structure” of an object (data.frame)

```
str(cars)
```

```
'data.frame': 50 obs. of 2 variables:
```

```
$ speed: num 4 4 7 7 8 9 10 10 10 11 ...
```

```
$ dist : num 2 10 4 22 16 10 18 26 34
```

```
17 ...
```

# str()

Get the “structure” of... now you try

write:

`data()` and find out the structure (`str()`) of some of these build in objects

Also try to plot some using e.g. `plot()`,  
`hist()`, `boxplot()`, `image()`,  
`barplot()`

# Index / subset / slice

R is very powerful in slicing

Examples:

“Get the values of the controls”

```
value[names(value)=="ctrl"]
```

“Get profiles of significant items”

```
profiles[, pval<0.05]
```

“order X by Y”

```
x[order(Y)]
```

# Recycling - be aware

**Example:**

```
a<-c(1,10,100,1000)
```

```
b<-c(1,2)
```

```
a+b
```

```
[1]      2     12    101   1002
```

```
m<-matrix(rep(1,16), ncol=4)
```

```
m+b
```

```
      [,1] [,2] [,3] [,4]  
[1,]     2     2     2     2  
[2,]     3     3     3     3  
[3,]     2     2     2     2  
[4,]     3     3     3     3
```

# Get help

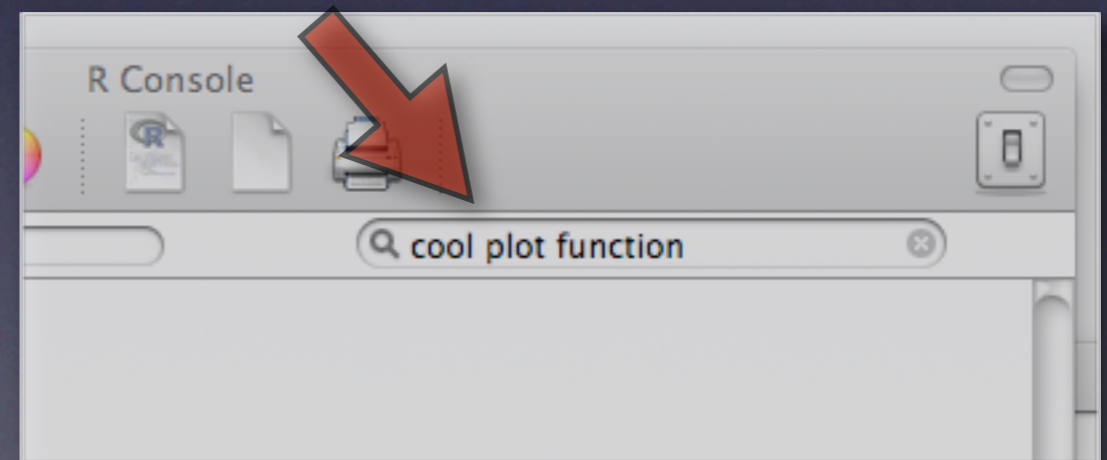
**Help on a specific function:**

```
help(rnorm) # get help on the "rnorm" function
```

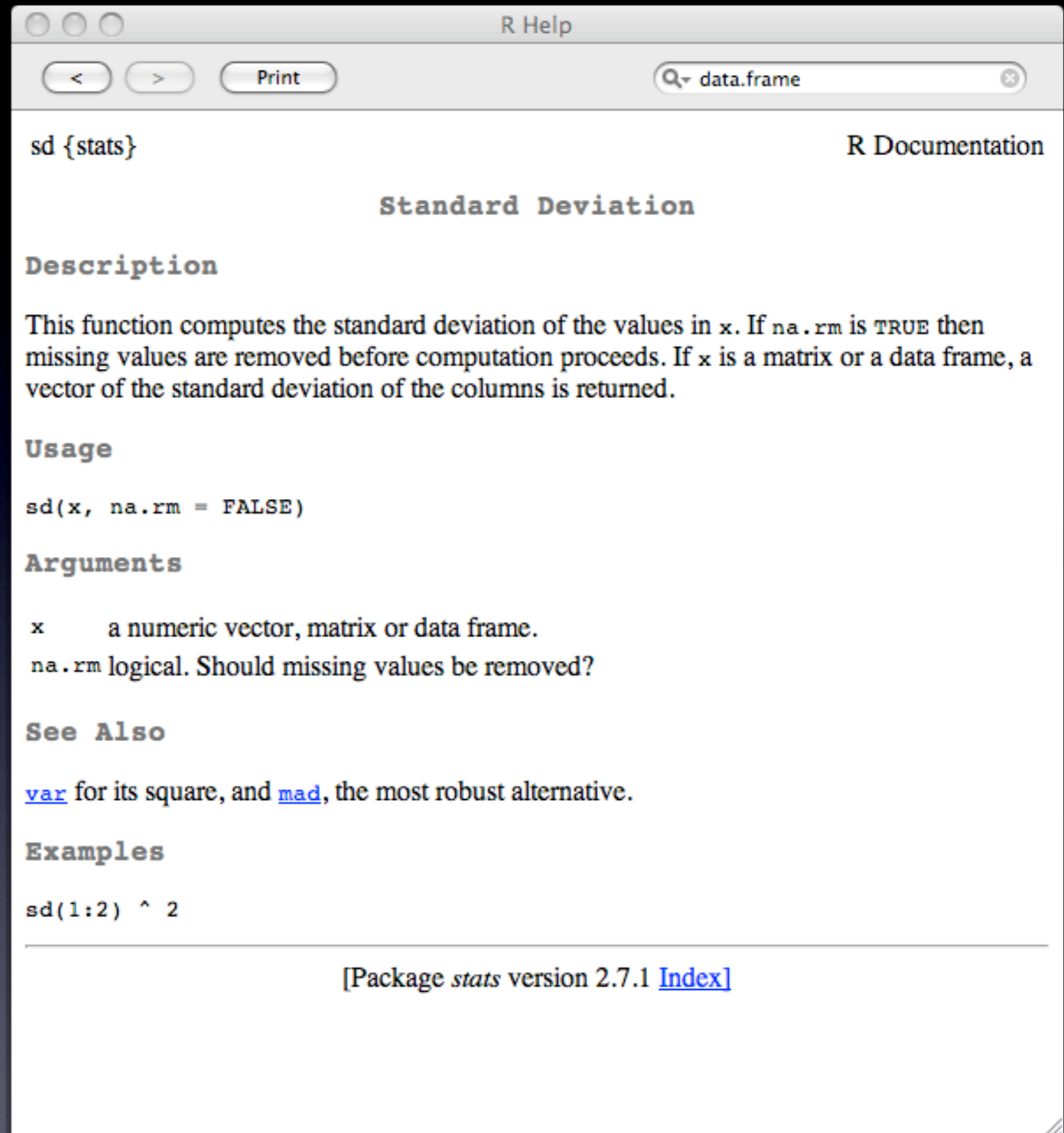
```
?rnorm # short for the same thing
```

**Fuzzy help search:**

```
help.search("pca")
```



# Help



The image shows a screenshot of an R Help window. The window title is "R Help". At the top, there are navigation buttons: a left arrow, a right arrow, and a "Print" button. On the right side, there is a search bar containing the text "data.frame". The main content area displays the documentation for the `sd` function. The title "sd {stats}" is on the left, and "R Documentation" is on the right. The function name "Standard Deviation" is centered. Below this, the "Description" section explains that the function computes the standard deviation of the values in `x`, and that missing values are removed if `na.rm` is `TRUE`. The "Usage" section shows the function signature `sd(x, na.rm = FALSE)`. The "Arguments" section lists `x` as a numeric vector, matrix, or data frame, and `na.rm` as a logical value. The "See Also" section points to `var` and `mad`. The "Examples" section shows the code `sd(1:2) ^ 2`. At the bottom, there is a footer indicating the package version and a link to the index.

sd {stats} R Documentation

**Standard Deviation**

**Description**

This function computes the standard deviation of the values in `x`. If `na.rm` is `TRUE` then missing values are removed before computation proceeds. If `x` is a matrix or a data frame, a vector of the standard deviation of the columns is returned.

**Usage**

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

**Arguments**

`x` a numeric vector, matrix or data frame.  
`na.rm` logical. Should missing values be removed?

**See Also**

[var](#) for its square, and [mad](#), the most robust alternative.

**Examples**

```
sd(1:2) ^ 2
```

---

[Package *stats* version 2.7.1 [Index](#)]

# Write reports

```
A_25-35_vs_salt_report.R - [Bjorns-Mac-2.local] /Users/hbjorn.o/Desktop/
File Edit Search Preferences Shell Macro Windows Help
/Users/hbjorn.o/Desktop/A_25-35_vs_salt_report.R 2325 bytes L: 12 C: 45

Design<-read.table("design.tab", header=TRUE, row.names=1)
Treatment<-factor(Design$Treatment, levels=c("A&1-42", "A&42-1", "A&25-35", "A&35-25", "ctrl"))

# Preprocessing
require(affy)
affyb<-read.affybatch(filenamees=row.names(Design)) # ignore warnings
ei<-exprs(rma(affyb))

# Statistics
require(genefilter)
subset.i<-Treatment %in% c("ctrl", "A&25-35")
pval<-rowttests(ei[,subset.i], factor(Treatment[subset.i]))$p.value

# fold change
fc<- apply(ei[,subset.i], 1, function(V) {ctrl.i<-Treatment[subset.i]=="ctrl"; mean(V[!ctrl.i])-mean(V[ctrl.i])})

# FDR
# Permutations
pf1<-factor(Treatment[subset.i][c(11,2,7,8,4,3, 9,5,12,6,10,1)])
pf2<-factor(Treatment[subset.i][c(12,1,8,3,7,2, 9,4,5,6,10,11)])
pf3<-factor(Treatment[subset.i][c(6,11,5,7,9,1, 4,2,8,10,3,12)])
pf4<-factor(Treatment[subset.i][c(6,5,11,1,7,9, 10,4,8,2,3,12)])
# factor check
cor(matrix(c(factor(Treatment[subset.i]),pf1,pf2,pf3,pf4), ncol=5))

pval.perm1<-rowttests(ei[,subset.i], pf1)$p.value
pval.perm2<-rowttests(ei[,subset.i], pf2)$p.value
pval.perm3<-rowttests(ei[,subset.i], pf3)$p.value
pval.perm4<-rowttests(ei[,subset.i], pf4)$p.value
pval.4xperm<-c(pval.perm1, pval.perm2,pval.perm3,pval.perm4)

ps<-sort(pval)
fdr_tmp<-vector(length=ps)
for(i in 1:length(ps)) {
  fdr_tmp[i]<-sum(pval.4xperm <= ps[i])/4/i
}
fdr_tmp[fdr_tmp>1]<-1
```

```
R Console
17 13 34
18 13 34
19 13 46
20 14 26
21 14 36
22 14 60
23 14 80
24 15 20
25 15 26
26 15 54
27 16 32
28 16 40
29 17 32
30 17 40
31 17 50
32 18 42
33 18 56
34 18 76
35 18 84
36 19 36
37 19 46
38 19 68
39 20 32
40 20 48
41 20 52
42 20 56
43 20 64
44 22 66
45 23 54
46 24 70
47 24 92
48 24 93
49 24 120
50 25 85
>
```

# Loops

for, while, repeat, break, next

## Examples:

```
for(i in 1:10) { }
```

```
while(i < 10) {i+1}
```

```
repeat {}
```

# if

No surprises here:

```
if (i > 10) {  
    message("i is more than 10")  
}  
else {  
    message("i is not more than 10")  
}
```

# if

No surprises here:

```
if (i>10) {  
    message("i is more than 10")  
}  
else {  
    message("i is not more than 10")  
}
```

See also “switch”