**R Reference Card**

by Tom Short, EPRI PEAC, tshort@epri-peac.com 2004-11-07

Granted to the public domain. See www.Rpad.org for the source and latest version. Includes material from *R for Beginners* by Emmanuel Paradis (with permission).

**Getting help**
Most R functions have online documentation.

`help(topic)` documentation on topic `?topic` id.

`apropos("topic")` the names of all objects in the search list matching the regular expression "topic"

`help.start()` start the HTML version of help

`str(a)` display the internal structure of an R object

`summary(a)` gives a "summary" of a, usually a statistical summary but it is *generic* meaning it has different operations for different classes of a

`ls()` show objects in the search path; specify `pat="pat"` to search on a pattern

`ls.str()` show for each variable in the search path

`dir()` show files in the current directory

`methods(a)` shows S3 methods of a

`methods(class=class(a))` lists all the methods to handle objects of class a

**Input and output**

`load()` load the datasets written with `save`

`data(x)` loads specified data sets

`library(x)` load add-on packages

`read.table(file)` reads a file in table format and creates a data frame from it; the default separator `sep=""` is any whitespace; use `header=TRUE` to read the first line as a header of column names; use `as.is=TRUE` to prevent character vectors from being converted to factors; use `comment.char=""` to prevent `#` from being interpreted as a comment; use `skip=n` to skip n lines before reading data; see the help for options on row naming, NA treatment, and others

`read.csv("filename",header=TRUE)` id. but with defaults set for reading comma-delimited files

`read.delim("filename",header=TRUE)` id. but with defaults set for reading tab-delimited files

`read.fwf("filename",header=FALSE,sep="",as.is=FALSE)` read a table of fixed width formatted data into a data frame; `widths` is an integer vector, giving the widths of the fixed-width fields

`save(file,...)` saves the specified objects (... in the XDR platform-independent binary format

`save.image(file)` saves all objects

`cat(..., file="", sep="")` prints the arguments after coercing to character; `sep` is the character separator between arguments

`print(a,...)` prints its arguments; generic, meaning it can have different methods for different objects

`format(x,...)` format an R object for pretty printing

`write.table(x,file="",row.names=TRUE,col.names=TRUE,sep="")` prints x after converting to a data frame; if `quote=TRUE`, character or factor columns are surrounded by quotes (*x* $sep$ is the field separator; *col* is the end-of-line separator; *na* is the string for missing values; use `col.names=NA` to add a blank column header to get the column headers aligned correctly for spreadsheet input

`sink(file)` output to file until `sink()`

Most of the I/O functions have a file argument. This can often be a character string naming a file or a connection. `file=""` means the standard input or output. Connections can include files, pipes, ziped files, and R variables.

On windows, the file connection can also be used with description "clipboard". To read a table copied from Excel, use `x <- read.delim("clipboard")`

To write a table to the clipboard for Excel, use `write.table(x, "clipboard",sep="\t")`

For database interaction, see packages RODBC, DBI, RMySQL, RPostgreSQL, and ROracle. See packages XML, hdfs, netCDF for reading other file formats.

### Data creation

`c(...)` general function to combine arguments with the default forming a vector; with `recursive=TRUE` descends through lists combining all elements into one vector

`from:to` generates a sequence; `:` has operator priority; `1:4+1` is "2,3,4,5"

`seq(from,to)` generates a sequence by specifying increment; `length.specifies desired length`

`seq(along=x)` generates `1, 2, ..., length(along)` useful for loops

`rep(x,times)` replicate `x` `times`; use each- to repeat "each" element of `x` `times` each; `rep(c(1,2,3),2)` is `1 2 3 1 2 3`

`dim(x)` returns the number of rows and columns of `x`; but with defaults set to prevent character vectors from being converted to factors; use `comment.char=""` to prevent `#` from being interpreted as a comment; use `skip=n` to skip n lines before reading data; see the help for options on row naming, NA treatment, and others

`dim(x)` returns the number of rows and columns of `x`

`data.frame(...)` creates a data frame from the named or unnamed arguments; `data.frame(v1=1:4,cb=c("a","b","c","d"),n=10)`; shorter vectors are recycled to the length of the longest

`list(...)` create a list of the named or unnamed arguments; `list(a=c(1,2),b="hi",c=11)`

`array(x,dim=)` array with data `x`; specify dimensions like `dim(x)` for `dim(x)` is the same but treats a vector as a one-row matrix

`matrix(x,nrow=,ncol=)` creates a matrix of the named or unnamed arguments; `matrix(c(1,2,3),2)` is `1 2 3 1 2 3`

`data.frame(...)` create a data frame from the named or unnamed arguments; `data.frame(v1=1:4,cb=c("a","b","c","d"),n=10)`; shorter vectors are recycled to the length of the longest

`list(...)` create a list of the named or unnamed arguments; `list(a=c(1,2),b="hi",c=11)`

`array(x, dim=)` array with data `x`; specify dimensions like `dim(x)` for `dim(x)` is the same but treats a vector as a one-row matrix

`matrix(x, nrow=, ncol=)` creates a matrix of the named or unnamed arguments; `matrix(c(1,2,3),2)` is `1 2 3 1 2 3`

`data.frame(...)` create a data frame from the named or unnamed arguments; `data.frame(v1=1:4,cb=c("a","b","c","d"),n=10)`; shorter vectors are recycled to the length of the longest

`list(...)` create a list of the named or unnamed arguments; `list(a=c(1,2),b="hi",c=11)`

**Slicing and extracting data**

`x[n]` list with elements `n`

`x[[n]]` `n`th element of the list

`x["name"]]` element of the list named "name"

`x$name` id.

Indexing lists

`x[n]` list with elements `n`

`x[[n]]` `n`th element of the list

`x["name"]]` element of the list named "name"

`x$name` id.

Indexing matrices

`x[i,j]` element at row i, column j

`x[i,]$j` column j

`x[,c(1,3)]` columns 1 and 3

`x["name",]` row named "name"

Indexing data frames (matrix indexing plus the following)

`x["name"]$column named "name"`

`x$name` id.

### Variable conversion

`as.array(x, as.data.frame(x), as.numeric(x), as.logical(x), as.character(x), ...)` convert type; for a complete list, use `methods(as)`

### Variable information

`is.na(x), is.null(x), is.array(x), is.data.frame(x), is.numeric(x), is.character(x), ...` test for type; for a complete list, use `methods(is)`

`length(x)` number of elements in `x`

`dim(x)` Retrieve or set the dimension of an object; `dim(x) <- c(3,2)`

`dimnames(x)` Retrieve or set the dimension names of an object

`nrow(x)` number of rows; `NR(x)` is the same but treats a vector as a one-row matrix

`ncol(x)` and `NCOL(x)` id. for columns

`class(x)` get or set the class of `x`; `class(x) <- "myclass"`

`unclass(x)` remove the class attribute of `x`

`attr(x, which)` get or set the attribute which of `x`

`attributes(obj)` get or set the list of attributes of `obj`

**Data selection and manipulation**

`which.max(x)` returns the index of the greatest element of `x`

`which.min(x)` returns the index of the smallest element of `x`

`rev(x)` reverses the elements of `x`

`sort(x)` sorts the elements of `x` in increasing order; to sort in decreasing order: `rev(sort(x))`

`cut(x,breaks)` divides `x` into intervals (factors); `breaks` is the number of cut intervals or a vector of cut points

`match(x,y)` returns a vector of the same length than `x` with the elements of `x` which are in `y` (NA otherwise)

`which(x == a)` returns a vector of the indices of `x` if the comparison operation is true (`TRUE`), in this example the values of `i` for which `x[i]` == `a` (the argument of this function must be a variable of mode logical)

`choose(n, k)` computes the combinations of `k` events among `n` repetitions

`na.omit(x)` supresses the observations with missing data (NA) (supresses the corresponding line if `x` is a matrix or a data frame)

`na.fail(x)` returns an error message if `x` contains at least one NA
Matplotlib plots the first column of x vs. the first one of y, the second one of x vs. the second one of y, etc.

Fourfoldplot(x) visualizes, with quarters of circles, the association between two dichotomous variables for different populations (x must be an array with dim=c(2,2,k), or a matrix with dim=c(2,2) if k=1)

Asscplot(x) Cohen–Friendly graph showing the deviations from independence of rows and columns in a two dimensional contingency table.

Mosaicplot(x) 'mosaic' graph of the residuals from a log-linear regression of a contingency table.

Pairs(x) if x is a matrix or a data frame, draws all possible bivariate plots between the columns of x.

Plot.ts(x) if x is an object of class "ts", plot of x with respect to time, x may be multivariate but the series must have the same frequency and dates

Ts.plot(x) id. but if x is multivariate the series may have different dates and must have the same frequency

Qqnorm(x) quantiles of x with respect to the values expected under a normal law

Qqplot(x, y) quantiles of x with respect to the quantiles of y

Contour(x, y, z) contour plot (data are interpolated to draw the curves), x and y must be vectors and z must be a matrix so that dim(z)=c(length(x),length(y)) (x and y may be omitted)

Filled.contour(x, y, z) id. but the areas between the contours are coloured, and a legend of the colours is drawn as well

Image(x, y, z) id. but with colours (actual data are plotted)

Persp(x, y, z) id. but in perspective (actual data are plotted)

Stars(x) if x is a matrix or a data frame, draws a graph with segments or a star where each row of x is represented by a star and the columns are the segments.

Symbols(x, y, ...) draws, at the coordinates given by x and y, symbols (circles, squares, rectangles, stars, thermometers or "boxplots") which sizes, colours ... are specified by supplementary arguments

Termplot(mod.obj) plot of the partial (effects) of a regression model (mod.obj)

The following parameters are common to many plotting functions:

- Contour(x, y, z) draws the data x on the x-axis as small vertical lines
- Locator(n, type=",", ...) returns the coordinates (x,y) after the user has clicked n times on the plot with the mouse; also draws symbols (type=p) or lines (type=1) with respect to optional graphic parameters (...); by default nothing is drawn (type="")

Graphical parameters

These can be set globally with par(...); many can be passed as parameters to plotting commands.

- Adj controls text justification (0 left-justified, 0.5 centred, 1 right-justified)
- Bg specifies the colour of the background
- Cex controls the size of texts and symbols
- Cex.lab, the title, Cex.main, and the sub-title
- Col controls the color of symbols and lines
- Font an integer which controls the style of text (1: normal, 2: italics, 3: bold, 4: bold italics)
- Lys an integer which controls the orientation of the axis labels

Lattice (Trellis) graphics

Xyplot(y~x) bivariate plots (with many functionalities)

Barchart(y) histogram of the values of y with respect to those of x

Dotplot(y~x) Cleveland dot plot (stacked plots line-by-line and column-by-column)

Densityplot(y) density functions plot

Histogram(x) histogram of the frequencies of x

Boxplot(y) "box-and-whiskers" plot

Qqmath(x) quantiles of x with respect to the values expected under a theoretical distribution

Stripplot(y) single dimension plot, x must be numeric, y may be a factor

Qq(y) quantiles to compare two distributions, x must be numeric, y may be numeric, character, or factor but must have two 'levels'

Spline(x) matrix of bivariate plots

Parallel(x) parallel coordinates plot

Levelplot(z~x+y|g1+g2) coloured plot of the values of z at the coordinates given by x and y (x and y are all of the same length)

Wireframe(z~x+y|g1+g2) 3d surface plot

Cloud(z~x+y|g1+g2) 3d scatter plot
In the normal Lattice formula, \( y \mid x \mid g1 \) \( g2 \) has combinations of optional conditioning variables \( g1 \) and \( g2 \) plotted on separate panels. Lattice functions take many of the same arguments as base graphics plus also data- the data frame for the formula variables and subset- for subsetting. Use panel- to define a custom panel function (see apropos("panel") and ?llines). Lattice functions return an object of class trellis and have to be print-ed to produce the graph. Use print(xplot(...)) inside functions where automatic printing doesn’t work. Use lattice.theme and lset to change Lattice defaults.

**Optimization and model fitting**

```r
optim(par, fn, method = c("Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN") general-purpose optimization; par is initial values, fn is function to optimize (normally minimize) nlm(f, p) minimize function f using a Newton-type algorithm with starting values p
lm(formula) fit linear models; formula is typically of the form response ~ termA + termB + ...; use I(x^y) + I(x^2) for terms made of nonlinear components
glm(formula, family=) fit generalized linear models, specified by giving a symbolic description of the linear predictor and a description of the error distribution; family is a description of the error distribution and link function to be used in the model; see ?family
nls(formula) nonlinear least-squares estimates of the nonlinear model parameters
approx(x, y=) linearly interpolate given data points; x can be an xy plotting structure
spline(x, y=) cubic spline interpolation
loess(formula) fit a polynomial surface using local fitting
```

Many of the formula-based modeling functions have several common arguments:

- `data` the data frame for the formula variables,
- `subset` a subset of variables used in the fit,
- `na.action` action for missing values: `na.fail`, `na.omit`, or a function. The following generics often apply to model fitting functions:

- `predict(fit,...)` predictions from `fit` based on input data
- `df.residual(fit)` returns the number of residual degrees of freedom
- `coef(fit)` returns the estimated coefficients (sometimes with their standard-errors)
- `residuals(fit)` returns the residuals
- `deviance(fit)` returns the deviance
- `fitted(fit)` returns the fitted values
- `logLik(fit)` computes the logarithm of the likelihood and the number of parameters
- `AIC(fit)` computes the Akaike information criterion or AIC

**Statistics**

```r
aov(formula) analysis of variance model
anova(fit,...) analysis of variance (or deviance) tables for one or more fitted model objects
density(x) kernel density estimates of x
binom.test(), pairwise.t.test(), power.t.test(), prop.test(), t.test(), ... use help.search("*test")
```

**Distributions**

```r
rnorm(n, mean=0, sd=1) Gaussian (normal)
rexp(n, rate=1) exponential
rgamma(n, shape, scale=1) gamma
```

```r
rpois(n, lambda) Poisson
rweibull(n, shape, scale=1) Weibull
rcauchy(n, location=0, scale=1) Cauchy
rbeta(n, shape1, shape2) beta
rt(n, df) 'Student' (t)
rf(n, df1, df2) Fisher–Snedecor (F) (\( \chi^2 \))
rchisq(n, df) Pearson
rbinom(n, size, prob) binomial
rgeom(n, prob) geometric
rhyper(nn, m, n, k) hypergeometric
rlogis(n, location=0, scale=1) logistic
rlnorm(n, meanlog=0, sdlog=1) lognormal
rnbinom(n, size, prob) negative binomial
runif(n, min=0, max=1) uniform
rwilcox(nn, m, n), rsignrank pnq Wilcoxon’s statistics
```

All these functions can be used by replacing the letter \( r \) with \( d \), \( p \) or \( q \) to get, respectively, the probability density \((dfunc(x, \ldots))\), the cumulative probability density \((pfunc(x, \ldots))\), and the value of quantile \((qfunc(p, \ldots))\), with \( 0 < p < 1 \).

**Programming**

```r
function( arglist ) expr function definition
return(value)
if(cond) expr
if(cond) cons.expr else alt.expr
for(var in seq) expr
while(cond) expr
repeat expr
break
next
```

Use braces \{\} around statements

```r
ifelse(test, yes, no) a value with the same shape as test filled with elements from either yes or no
do.call(funname, args) executes a function call from the name of the function and a list of arguments to be passed to it
```