**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.

`help.search("topic")` search the help system

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

### 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 tabular 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(file,widths,header=FALSE,sep="",as.is=FALSE)` reads 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

### Data creation

`c(...)` generic 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; `1:4` 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(x)` useful for for loops

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

`data.frame(...)` create a data frame of the named or unnamed arguments: `data.frame(v1=c(1,2,3),v2=c(1,2),v3=c(1,2,3))` 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=31)`

`array(x,dim)` array with data `x`; specify dimensions like `dim=c(3,4,2)`; elements of `x` recycle if `x` is not long enough

`matrix(x, nrow=ncol) matrix; elements of `x` recycle

`factor(x, levels)` encodes a vector `x` as a factor

`gl(n,k,length=k,labels=1:n)` generate levels (factors) by specifying the pattern of their levels; `k` is the number of levels, and `n` is the number of repetitions

`expand.grid()` a data frame from all combinations of the supplied vectors or factors

`cbind(...)` combine arguments by rows for matrices, data frames, and others

### Variable information

`is.na(x), is.null(x), is.array(x), is.data.frame(x), is.numeric(x), is.complex(x), is.character(x), ...` convert to 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

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

`class(x)` or `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`

`sort(x)` reverses the elements of `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 `1` for which `x[1] == 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)` suppresses the observations with missing data (NA) (suppresses 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

### 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,]` row `i`

`x[,]` 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.complex(x), as.character(x), ...` convert type; for a complete list, use `methods(as)`

### Slicing and extracting data

Indexed vectors

`x[n]` `n`th element

`all but the `n`th element`

`first `n` elements`

`elements from `n` to the end`

`specific elements`

`element named "name"`

`all elements greater than 3`

`all elements between 3 and 5`

`elements in the given set`
matplot(x, y) bivariate plot of 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)
assocplot(x) Cohen–Fry 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 contingency 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) quantities of x with respect to the values expected under a normal law
qqplot(x, y) quantities of y with respect to the quantities of x
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 lengths of 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
terplot(mod.obj) plot of the (partial) effects of a regression model (mod.obj)

The following parameters are common to many plotting functions:
add=FALSE if TRUE superposes the plot on the previous one (if it exists)
axes=TRUE if FALSE does not draw the axes and the box
type="p" specifies the type of plot, "p": points, "l": lines, "b": points connected by lines, "o": id. but the lines are over the points, "h": vertical lines, "s": steps, the data are represented by the top of the vertical lines, "i": id. but the data are represented by the bottom of the vertical lines
xlim=, ylim= specifies the lower and upper limits of the axes, for example with xlim=c(0,1,10) or xlim=range(x)
labc=, ltyab= annotates the axes, must be variables of mode character
main= main title, must be a variable of mode character
sub= sub-title (written in a smaller font)

Low-level plotting commands
points(x, y) adds points (the option type= can be used)
lines(x, y) id. but with lines
text(x, y, labels, ...) adds text given by labels at coordinates (x,y); a typical use is: plot(x, y, type="n"); text(x, y, names)

mtext(text, side=3, line=0, ...) adds text given by text in the margin specified by side (see axis() below); line specifies the line from the plotting area
segments(x0, y0, x1, y1) draws lines from points (x0,y0) to points (x1,y1)
arrows(x0, y0, x1, y1, angle= 30, code=2) id. with arrows at points (x0,y0) if code=2, at points (x1,y1) if code=1, or both if code=2; angle controls the angle from the shaft of the arrow to the edge of the arrow
abline(a, b) draws a line of slope b and intercept a
abline(h=y) draws a horizontal line at ordinate y
abline(v=x) draws a vertical line at abcissa x
abline(lm.obj) draws the regression line given by lm.obj
rect(x1, y1, x2, y2) draws a rectangle which left, right, bottom, and top limits are x1,x2,y1, and y2, respectively
polygon(x, y) draws a polygon joining the points with coordinates given by x and y
legend(x, y, legend) adds the legend at the point (x,y) with the symbols given by legend

title() adds a title and optionally a sub-title
axis(side, vec) adds an axis at the bottom (side=1), on the left (2), at the top (3), or on the right (4); vec (optional) gives the abcissa (or ordinates) where tick-marks are drawn
rug(x) draws the data 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="l") with respect to optional graphic parameters (...,), by default nothing is drawn (type="n")

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 (ex.: bg="red", bg="blue", ...)
the list of the 657 available colours is displayed with colors()!
bty controls the type of box drawn around the plot, allowed values are: "o", "n", "o", "n", "c", "o", "n", "c", "o", "c" (the box looks like the corresponding character); if bty="n" the box is not drawn
cex a value controlling the size of texts and symbols with respect to the default; the following parameters have the same control for numbers on the axes, cex.axis, the axis labels, cex.lab, the title, cex.main, and the sub-title, cex.sub
col controls the color of symbols and lines; use color names: "red", "blue", see colors() or as "#FF0000", see rgb(), hsv(), gray(), and rainbow(); as for cex there are: col.axis, col.lab, col.main, col.sub
font an integer which controls the style of text (1: normal, 2: italics, 3: bold, 4: bold italics); as for cex there are: font.axis, font.lab, font.main, font.sub
las an integer which controls the orientation of the axis labels (0: parallel to the axes, 1: horizontal, 2: perpendicular to the axes, 3: vertical)
lty controls the type of lines, can be an integer or string (1: "solid", 2: "dashed", 3: "dotted", 4: "dotdash", 5: "longdash", 6: "twodash", or a string of up to eight characters between "0" and "9") which specifies alternatively the length, in points or pixels, of the drawn elements and the blanks, for example lty="44" will have the same effect than lty=2
lwd a numeric which controls the width of lines, default 1
mar a vector of 4 numeric values which control the space between the axes and the border of the graph of the form c(bottom, left, top, right), the default values are c(5,4,4,2,1)
mfcol a vector of the form c(nr,nc) which partitions the graphic window as a matrix of nr lines and nc columns, the plots are then drawn in columns
mfrow id. but the plots are drawn by row
pch controls the type of symbol, either an integer between 1 and 25, or any single character within ""10 2 3 4 5 6 7 8 9 0 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 partitions the size in points of texts and symbols
pty a character which specifies the type of the plotting region, "s": square, "n": maximal
tck a value which specifies the length of tick-marks on the axes as a fraction of the smallest of the width or height of the plot; if tck=1 a grid is drawn
tcl a value which specifies the length of tick-marks on the axes as a fraction of the height of a line of text (by default tcl=-0.5)
xaxt if xaxt="n" the x-axis is set but not drawn (useful in conjunction with axis(side=1,...))
yaxt if yaxt="n" the y-axis is set but not drawn (useful in conjunction with axis(side=2,...))

Lattice (Trellis) graphics
xyplot(y~x) bivariate plots (with many functionalities)
barchart(y~x) 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~x) density functions plot
bwplot(y~x) "box-and-whiskers" plot
gqmath(y~x) quantiles of x with respect to the values expected under a theoretical distribution
stripplot(y~x) single dimension plot, x must be numeric, y may be a factor
qq(y~x) quantiles to compare two distributions, x must be numeric, y may be numeric, character, or factor but must have two "levels"
spom(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, y and z 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, \( y \mid g_1 \cdot g_2 \) has combinations of optional conditioning variables \( g_1 \) and \( g_2 \) 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 ?panelfunctions). Lattice functions return an object of class trellis and have to be print-ed to produce the graph. Use print(xyplot(...)) inside functions where automatic printing doesn’t work. Use lattice.theme and lset to change Lattice defaults.

**Optimization and model fitting**

- `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 \( \text{response} + \text{termA} + \text{termB} + \ldots; \) use \( I(x^2) \) 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**

- `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**

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

- 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' \( \tau \)
- rf(n, df1, df2) Fisher–Snedecor \( F \)
- 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(nn, n) 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 (\( d \)func\( (x, \ldots) \)), the cumulative probability density (\( p \)func\( (x, \ldots) \)), and the value of quantile (\( q \)func\( (p, \ldots) \), with \( 0 < p < 1 \)).

**Programming**

- 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

- ifelse(test, yes, no) a value with the same shape as test filled with elements from either yes or no

**do.call(funnname, args)** executes a function call from the name of the function and a list of arguments to be passed to it