**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`?

`apropos("topic")` search the help system `a`propo`"s` the names of all objects in the search list matching the regular expression "topic"

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

`str()` display the internal `str`tructure of an R object

`summary()` gives a "summary" of `su`mmary a, usually a statistical summary but it is `gene`ric 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()` str() for each variable in the search path

`dir()` show files in the current directory

`methods()` shows S3 methods of a

`methods(class=class(a))`

`dir() show files in the current directory

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

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

`methods()` 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()` 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 all lines before reading data; use 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="\t")`, `as.is=FALSE` read a table of `f`ixed `w`idth 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 ("a") and `sep` 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, zipped 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="",col.names=NA)`

For database interaction, see packages `RODBC`, `DBI`, `MySQL`, `RPGSQL`, and `ROracle`. See packages `XML`, `hdfs5`, `netCDF` for reading other file formats.

### 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 by `spec`ifies increment; length=`spec`ifies desired length

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

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

`data.frame(...)` create a data frame of the named or unnamed arguments: `data.frame(v=1:4,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=3i)`

`array(x,...)` array with `data` of `x` and `dim` lengths `...`; use `each=TRUE` to create a 3D array

`matrix(x,nrow=,ncol=)` create a matrix with `data` of `x` and `dim` lengths; `matrix(x,nrow=3)` creates a row matrix

`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

`indexing ...` id. by columns

### Slicing and extracting data

Indexing vectors

```
x[n]        n-th element
x[-n]       all but the n-th element
x[-(1:n)]   first n elements
x[c(1,2,4)] elements from n=1 to the end
x["name"]  elements name "name"
```

Indexing matrices

```
x[i,j]      element at row i, column j
x[i,]       row i
x[,j]       column j
x[,c(1,3)]  columns 1 and 3
```

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), ...
```

### Variable information

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

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; NROW(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 as 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 l 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 = n!/(k!(n-k)!)!
```

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

```
aa.fail(x) returns an error message if x contains at least one NA
```
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–Fryd 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) if x is a matrix 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 contour lines)
filled.contour(x, y, z) draws lines from points (x0,y0) to points (x1,y1)
segments(x0, y0, x1, y1) draws lines from points (x0,y0) to points (x1,y1)
lines(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=3; angle controls the angle from the shaft of the arrow to the edge of the arrow head
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 abscissa x
abline(lm.obj) draws the regression line given by lm.obj
rect(xl, yl, x2, y2) draws a rectangle which left, right, bottom, and top limits are xl,x2,yl,y2, respectively
polygon(x, y) polygons linking 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
points(x, y, ...) 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)
levels of the segments
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
text(x, y, type="n") adds text given by labels at coordinates (x,y); a typical use is: plot(x,y, type="n"); text(x, y, names)

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", "s", "h", "b", "p", "u" on "n" (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 effect 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 "#rrggbb"; 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="*4" 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(5.1, 4.1, 4.1, 2.1)
mfcolf the vector of form c(nc, 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 "" which sizes, colours . . . are specified by supplementary arguments
ps an integer which controls 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)
xxaxt if xaxt="n" the x-axis is but not drawn (useful in conjunction with axis(side=1, ...))
yaxt if yaxt="n" the y-axis is but not drawn (useful in conjunction with axis(side=2, ...))

Lattice (Trellis) graphics
xypolygon(y) bivariate plots (with many functionalities)
barchart(y) histogram of the values of y with respect to those of x
dotplot(y) Cleveland dot plot (stacked plots line-by-line and column-by-column)
densityplot(x) density functions plot
dotplot(x) Cleveland dot plot (stacked plots line-by-line and column-by-column)
densityplot(x) density functions plot
densityplot(x) density functions plot
densityplot(x) density functions plot
bwplot(x) "box-and-whiskers" plot
ggmath(x) quantiles of x with respect to the values expected under a theoretical distribution
stripchart(x) 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'
splothem(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, \( y \times |g1*g2 \) 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 \( \text{data=} \) the data frame for the formula variables and \( \text{subset=} \) for subsetting. Use \text{panel=} to define a custom panel function (see \text{apropos("panel")} and \text{?llines}). Lattice functions return an object of class trellis and have to be \text{print-ed} to produce the graph. Use \text{print(xyplot(...))} inside functions where automatic printing doesn’t work. Use \text{lattice.theme} and \text{lset} to change Lattice defaults.

Optimization and model fitting

\text{optim(par, fn, method = c("Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN") general-purpose optimization; \( \text{par=} \) is initial values, \( \text{fn=} \) is function to optimize (normally minimize) \( \text{nlm(f,p)} \) minimize function \( f \) using a Newton-type algorithm with starting values \( p \).

\text{lm(formula)} \) fit linear models; \( \text{formula=} \) is typically of the form \( \text{response} + \text{termA} + \text{termB} + ... \); use \( \text{I(x*y)} + \text{I(x^2)} \) for terms made of nonlinear components.

\text{glm(formula,family=)} \) fit generalized linear models, specified by giving a symbolic description of the linear predictor and a description of the error distribution; \( \text{family=} \) is a description of the error distribution and link function to be used in the model; see \text{?family}.

\text{nls(formula)} \) nonlinear least-squares estimates of the nonlinear model parameters.

\text{approx(x,y=)} \) linearly interpolate given data points; \( x \) can be an \text{xy} plotting structure.

\text{spline(x,y=)} \) cubic spline interpolation.

\text{loess(formula)} \) fit a polynomial surface using local fitting.

Many of the formula-based modeling functions have several common arguments: \( \text{data=} \) the data frame for the formula variables, \( \text{subset=} \) a subset of variables used in the fit, \( \text{na.action=} \) action for missing values: \( \text{"na.fail"}, \text{"na.omit"} \), or a function. The following generics often apply to model fitting functions:

\text{predict(fit,...)} \) predictions from \( \text{fit=} \) based on input data.

\text{df.residual(fit)} \) returns the number of residual degrees of freedom.

\text{coef(fit)} \) returns the estimated coefficients (sometimes with their standard-errors).

\text{residuals(fit)} \) returns the residuals.

\text{deviance(fit)} \) returns the deviance.

\text{fitted(fit)} \) returns the fitted values.

\text{logLik(fit)} \) computes the logarithm of the likelihood and the number of parameters.

\text{AIC(fit)} \) computes the Akaike information criterion or AIC.

Statistics

\text{aov(formula)} \) analysis of variance model.

\text{anova(fit,...)} \) analysis of variance (or deviance) tables for one or more fitted model objects.

\text{density(x)} \) kernel density estimates of \( x \).

\text{binom.test()}, \text{pairwise.t.test()}, \text{power.t.test()}, \text{prop.test()}, \text{t.test()}, ... \) use \text{help.search("test")}.

Distributions

\text{rnorm(n, mean=0, sd=1)} \) Gaussian (normal).

\text{rexp(n, rate=1)} \) exponential.

\text{rgamma(n, shape, scale=1)} \) gamma.

\text{rpois(n, lambda)} \) Poisson.

\text{rweibull(n, shape, scale=1)} \) Weibull.

\text{rcauchy(n, location=0, scale=1)} \) Cauchy.

\text{rbeta(n, shape1, shape2)} \) beta.

\text{rt(n, df)} \) ‘Student’ \( t \).

\text{rf(n, df1, df2)} \) Fisher–Snedecor \( (F) \).

\text{rchisq(n, df)} \) Pearson.

\text{rbinom(n, size, prob)} \) binomial.

\text{rgeom(n, prob)} \) geometric.

\text{rhyper(nn, m, n, k)} \) hypergeometric.

\text{rlogis(n, location=0, scale=1)} \) logistic.

\text{rlnorm(n, meanlog=0, sdlog=1)} \) lognormal.

\text{rnbinom(n, size, prob)} \) negative binomial.

\text{runif(n, min=0, max=1)} \) uniform.

\text{rwilcox(nn, m, n)} \) and \text{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 \( (dfunc(x, \ldots)) \), the cumulative probability density \( (pfunc(x, \ldots)) \), and the value of quantile \( (qfunc(p, \ldots)) \), with \( 0 < p < 1 \).

Programming

\text{function( arglist ) expr} \) function definition.

\text{return(value)} \) if \( (\text{cond}) \) expr.

\text{if(\text{cond}) cons.expr else alt.expr} \) for \( \text{var in seq} \) expr.

\text{while(\text{cond}) expr} \) repeat expr.

\text{break} \) next Use braces \{\} around statements.

\text{ifelse(test, yes, no)} \) a value with the same shape as \( \text{test} \) filled with elements from either \( \text{yes} \) or \( \text{no} \).

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

Distributions

\text{rnorm(n, mean=0, sd=1)} \) Gaussian (normal).

\text{rexp(n, rate=1)} \) exponential.

\text{rgamma(n, shape, scale=1)} \) gamma.