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

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)) 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(file,widths,header=FALSE,sep="",as.is=FALSE) read a table of fixed widthformatted 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 is TRUE, character or factor columns are surrounded by quotes (*x* sep is the field separator; eol 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. files="" 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, HTMLSQL, RFPGSQL, and ROracle. See packages XML, hdf5, 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; "c" 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 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,ch=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; 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=n,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 replications
expand.grid() a data frame from all combinations of the supplied vectors or factors
rbind(...) combine arguments by rows for matrices, data frames, and others
cbind(...) id. by columns

Slicing and extracting data
Indexing vectors
x[n]
x[-n]
x[1:n]
x[-(1:n)]
x[c(1,4,21)]
x["name"]
x[x > 3]
x[x > 3 & x < 5]
x[x == c("a","and","the")] all the nth element
all but the nth element
first n elements
elements from n+1 to end
specific elements
name element
elements greater than 3
elements between 3 and 5

Indicating lists
x[[1]] 1st element of the list
x[["name"]]] element of the list named "name"
x$name id.

Indexing matrices
x[1,] element at row 1, column j
x[1,] row 1
x[,1] 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)

Variable information
is.na(x), is.null(x), is.array(x), is.data.frame(x), is.numeric(x), is.complex(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; ncol(x) is the same but treats a vector as a one-row matrix

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 (breaks); 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[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!=0[k!(n-k)!]
na.omit(x) suppresses the observations with missing data (NA) (supersedes 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
unique(x) if x is a vector or a data frame, returns a similar object but with the duplicate elements suppressed

table(x) returns a table with the numbers of the different values of x (typically for integers or factors)

subset(x, ...) returns a selection of x with respect to criteria (...), typically comparisons: x[VI < 10] if x is a data frame, the option select gives the variables to be kept or dropped using a minus sign

sample(x, size) resample randomly and without replacement size elements in the vector x, the option replace = TRUE allows to resample with replacement

prop.table(x, margin=) table entries as fraction of marginal table

Math

sin, cos, tan, asin, acos, atan, atan2, log, log10, exp

max(x) maximum of the elements of x

min(x) minimum of the elements of x

range(x) then c(min(x), max(x))

sum(x) sum of the elements of x

diff(x) lagged and iterated differences of vector x

prod(x) product of the elements of x

mean(x) mean of the elements of x

median(x) median of the elements of x

var(x) or var(x, y) variance of the elements of x (calculated on n-1); if x is a matrix or a data frame, the variance-covariance matrix is calculated

cor(x) or cor(x, y) correlation matrix of x if it is a matrix or a data frame (1 if x is a vector)

corr(x, y) or cov(x, y) covariance between x and y, or between the columns of x and those of y if they are matrices or data frames

cor(x, y) linear correlation between x and y, or correlation matrix if they are matrices or data frames

round(x, n) rounds the elements of x to n decimals

log(x, base) computes the logarithm of x with base base

scale(x) if x is a matrix, centers and reduces the data; to center only use the option center=FALSE, to reduce only scale=FALSE (by default center=TRUE, scale=TRUE)

pmin(x, y, ..., .) a vector which ith element is the minimum of x[i], y[i], ...

pmax(x, y, ..., .) id. for the maximum

cumsum(x) a vector which ith element is the sum from x[1] to x[i]

cumprod(x) id. for the product

cummin(x) id. for the minimum

cummax(x) id. for the maximum

union(x, y, ...) intersect(x, y, ...), setdiff(x, y, ...), setequal(x, y), is.element(x, el, set) "set" functions

Re(x) real part of a complex number

Im(x) imaginary part

Mod(x) modulus; abs(x) is the same

Arg(x) angle in radians of the complex number

 Conj(x) complex conjugate

convolve(x, y) compute the several kinds of convolutions of two sequences

cummax(x) id. for the maximum

cummin(x) id. for the minimum

corr(x, y) or cov(x, y) correlation matrix of x if it is a matrix or a data frame (1 if x is a vector)

corr(x, y) linear correlation between x and y, or correlation matrix if they are matrices or data frames

round(x, n) rounds the elements of x to n decimals

log(x, base) computes the logarithm of x with base base

scale(x) if x is a matrix, centers and reduces the data; to center only use the option center=FALSE, to reduce only scale=FALSE (by default center=TRUE, scale=TRUE)

pmin(x, y, ..., .) a vector which ith element is the minimum of x[i], y[i], ...

pmax(x, y, ..., .) id. for the maximum

cumsum(x) a vector which ith element is the sum from x[1] to x[i]

cumprod(x) id. for the product

cummin(x) id. for the minimum

cummax(x) id. for the maximum

union(x, y, ...) intersect(x, y, ...), setdiff(x, y, ...), setequal(x, y), is.element(x, el, set) "set" functions

Re(x) real part of a complex number

Im(x) imaginary part

Mod(x) modulus; abs(x) is the same

Arg(x) angle in radians of the complex number

 Conj(x) complex conjugate

convolve(x, y) compute the several kinds of convolutions of two sequences

fft(x) Fast Fourier Transform of an array

mvfft(x) FFT of each column of a matrix

filter(x, filter) applies linear filtering to a univariate time series or to each series separately of a multivariate time series

Many math functions have a logical parameter na.rm=FALSE to specify missing data (NA) removal.

Matrices

t(x) transpose
diag(x) diagonal

%*% matrix multiplication

solve(a, b) solves a * x = b for x

solve(a) matrix inverse of a

crossum(x) sum of rows for a matrix-like object; rowSums(x) is a faster version

colsum(x) or colSums(x) id. for columns

rowMeans(x) or fast version of row means

colMeans(x) id. for columns

Advanced data processing

apply(X, INDEX, FUN=) a vector or array or list of values obtained by applying a function FUN to margins (INDEX) of X

laply(X, INDEX, FUN=) apply FUN to each element of the list X

tapply(X, INDEX, FUN=) apply FUN to each cell of a ragged array given by X with indexes INDEX

merge(a, b, data=x) merge two data frames by common columns or row names

xtabs(a, b, data=x) a contingency table from cross-classifying factors

aggregate(x, y, by=F, FUN) splits the data frame x into subsets, computes summary statistics for each, and returns the result in a convenient form; by is a list of grouping elements, each as long as the variables in x

stack(x, ...) transform data available as separate columns in a data frame or list into a single column

unstack(x, ...) inverse of stack()

reshape(x, ...) reshapes a data frame between 'wide' format with the repeated measurements in separate records; the option 'long' format with the repeated measurements in separate columns of the same record and 'short' format with the repeated measurements in separate records; use (direction="wide") or (direction="long")

Strings

paste(..., sep=) concatenate vectors after converting to character; sep= is the string to separate terms (a single space is the default); collapse= is an optional string to separate "collapsed" results

substr(x, start, stop) substrings in a character vector; can also as-sign, as such: substr(x, start, stop) <- value

strsplit(x, split) split x according to the substring split

grep(pattern, x) searches for matches to pattern within x; see ?regex

gsub(pattern, replacement, x) replaces matched formulas by regular expression matching sub () is the same but only replaces the first occurrence.

tolower(x) convert to lowercase
toupper(x) convert to uppercase

match(x, table) a vector of the positions of first matches for the elements of x among table

x %in% table id. but returns a logical vector

pmatch(x, table) partial matches for the elements of x among table

nchar(x) number of characters

Dates and Times

The class Date has dates without times. POSIXct has dates and times, including time zones. Comparisons (e.g. >), seq(), and diff(time) are useful. Date also allows + and -. Class Date and class POSIXct are the respective classes. The default string format is "%Y-%m-%d %. Time can be represented as a character string (empty if not available).

Where leading zeroes are shown they will be used on output but are optional on input. See ?strftime.

Plotting

plot(x) plot of the values of x (on the x-axis) ordered on the x-axis

plot(x, y) bivariate plot of x (on the x-axis) and y (on the y-axis)

hist(x) histogram of the frequencies of x

boxplot(x) histogram of the values of x; use horiz=FALSE for horizontal bars

dotchart(x) if x is a data frame, plots a Cleveland dot plot (stacked plots line-by-line and column-by-column)

pie(x) circular pie-chart

boxplot(x) "box-and-whiskers" plot

sunflowerplot(x, y) id. than plot() but the points with similar coordinates are drawn as flowers which petal number represents the number of points

stripchart() plot of the values of x on a line (an alternative to boxplot() for small sample sizes)

coplot(x=y | z) bivariate plot of x and y for each value or interval of values of z

interaction.plot (f1, f2, y) if f1 and f2 are factors, plots the means of y (on the y-axis) with respect to the values of f1 (on the x-axis) and of f2 (different curves); the option fun allows to choose the summary statistic of y (by default fun=mean)
`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–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 `y` with respect to the quantiles 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))` (and `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, thermometres or "boxplots") which sizes, colours ... are specified by supplementary arguments

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

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, "s": 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)` or `xlim=range(x)`
- `xlab=`, `ylab=` annotates the axes, may 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 on the x-axis as small vertical lines

`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(x1, y1, x2, y2)` draws a rectangle which left, right, bottom, and top limits are `x1, x2, y1, 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

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", ...`

- `cex` specifies 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)`
- `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 ""'
- `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`)
- `xaxt` if `xaxt="n"` the x-axis is set but not drawn (useful in conjonction with axis(side=1, ...))
- `yaxt` if `yaxt="n"` the y-axis is set but not drawn (useful in conjonction 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 columnn-by-column)
- `densityplot(y~x)` density functions plot
- `histogram(x~y)` histogram of the frequencies of `x`
- `bwplot(y~x)` "box-and-whiskers" plot
- `qqmath(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
- `qplot(y~x)` quantiles to compare two distributions, `x` must be numeric, `y` may be numeric, character, or factor but must have two 'levels'
- `splot(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` 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, \( y \mid x | g_1 \cdot g_2 \) 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 ?panel functions). 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 \( \sim \) termA + termB + ...; use I(x^2 y^2) + 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

dl(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:

```r
predict(fit,...) predictions from fit based on input data
df.residual(fit) returns the residual degrees of freedom
coeff(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) wilcox.test(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\( px \)), the cumulative probability density (\( p \)func\( px \)), and the value of quantile (\( q \)func\( ppt \)), 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
```

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