### 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”

#### Character or factor columns are surrounded by quotes

```
x[5]  # x[5] is the field separator; col.names=NA to add a blank column header to
given arguments
```

<table>
<thead>
<tr>
<th>Character or factor columns are surrounded by quotes</th>
<th>( x[5] ) is the field separator; col.names=NA to add a blank column header to given arguments</th>
</tr>
</thead>
</table>

### 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: “:” has operator priority; 1:4+1 is “2,3,4,5”
```

<table>
<thead>
<tr>
<th>from:to generates a sequence</th>
<th>“:” has operator priority; 1:4+1 is “2,3,4,5”</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>seq(from,to) generates a sequence</th>
<th><em>by</em> specifies increment; length specifies desired length</th>
</tr>
</thead>
</table>

| 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 each 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(v=c(1,4),c=c("a","b","c","d"),n=10); shorter vectors are recycled to the length of the longest |
|---------------------|----------------------------------------------------------------|

### Input and output

<table>
<thead>
<tr>
<th>load()</th>
<th>load the datasets written with save</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>data(x) loads specified data sets</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>library() load add-on packages</th>
</tr>
</thead>
</table>

| 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= to skip n lines before reading data; see the help for options on row naming, NA treatment, and others |
|-----------------|----------------------------------|

<table>
<thead>
<tr>
<th>read.csv(“filename”,header=TRUE)</th>
<th>id. but with defaults set for reading comma-delimited files</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>read.delim(“filename”,header=TRUE)</th>
<th>id. but with defaults set for reading tab-delimited files</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>read.table(file,header=FALSE,sep=&quot;&quot;,as.is=FALSE)</th>
<th>read a table of fixed width formatted data into a “data.frame”; width is an integer vector, giving the widths of the fixed-width fields</th>
</tr>
</thead>
</table>

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

| save.image(file) saves all objects |
|-----------------------------------|--------------------------------------------------------------------------------|

<table>
<thead>
<tr>
<th>cat(..., file=&quot;&quot;, sep=&quot; “&quot;)</th>
<th>prints the arguments after coercing to character; sep is the character separator between arguments</th>
</tr>
</thead>
</table>

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

<table>
<thead>
<tr>
<th>format(x,...)</th>
<th>format an R object for pretty printing</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>write.table(x,file=&quot;&quot;,row.names=TRUE,col.names=TRUE,sep=&quot; “&quot;)</th>
<th>prints x after converting to a data frame; if quote is TRUE,</th>
</tr>
</thead>
</table>

### Slicing and extracting data

#### Indexing vectors

<table>
<thead>
<tr>
<th>x[n]</th>
<th>list elements</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>x[n]</th>
<th>list elements</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>x[-n]</th>
<th>all but the n(^{th}) element</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>x[1:n]</th>
<th>first n elements</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>x[-(1:n)]</th>
<th>elements from n+1 to the end specific elements</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>x[c(1,4,2)]</th>
<th>element named “name”</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>x[“name”]</th>
<th>all elements greater than 3</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>x[x &gt; 3]</th>
<th>all elements between 3 and 5</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>x[x &gt; 3 &amp; x &lt; 5]</th>
<th>elements in the given set</th>
</tr>
</thead>
</table>

### 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.complex(x)**, **is.character(x)**, **...** test for type; for a complete list, use methods(is)

### 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))

<table>
<thead>
<tr>
<th>cut(x,breaks)</th>
<th>divides x into intervals (factors); breaks is the number of cut intervals or a vector of cut points</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>match(x,y)</th>
<th>returns a vector of the same length than x with the elements of x which are in y (NA otherwise)</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>which(x==a)</th>
<th>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)</th>
</tr>
</thead>
</table>

| choose(n,k) | computes the combinations of k elements 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 |
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 > 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) id. 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

quantile(x,probs=) sample quantiles corresponding to the given probabilities (defaults to 0.25, 0.5, 0.75, 1)

weighted.mean(x, w) mean of x with weights w

rank(x) ranks of the elements of x

var(x) or cov(x) 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.

sd(x) standard deviation of x

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

var(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], ... .

pmx(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(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 \%\% b for x
	solve(a) matrix inverse of a

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

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

rowMeans(x) 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

lapply(X, 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

by(data, INDEX, FUN=) apply FUN to data frame data subsetted by INDEX merge(s,a,b) merge two data frames by common columns or row names

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

aggregate(x, by, FUN=) 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 repeated measurements in separate columns of the same record and 'long' format with the repeated measurements in separate records; use (direction="wide") or (direction="long")

Strings:
paste(...) concatenate vectors after converting to character; sep= 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 assign, as substr(x, start, stop) <- value

split(x, split) split x according to the string split

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

gsub(pattern, replacement, x) replacement of matches determined 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() are useful. Date also accepts and −. DateTimes classes give more information. See also package chron.

as.Date(s) and as.POSIXct(s) convert to the respective class; format(dt) converts to a string representation. The default string format is "2001-02-21". These accept a second argument to specify a format for conversion. Some common formats are:

%Y Abbreviated full weekday name.
%A Abbreviated and full month name.
%j Day of the month (01–31).
%W Hours (00–23).
%w Hours (01–12).
%y Day of year (001–366).
%m Month (01–12).
%t Minute (00–59).
%p AM/PM indicator.
%z Second as decimal number (00–61).
%U Week (00–53); the first Sunday as day 1 of week 1.
%U Weekday (0–6, Sunday is 0).
%Y Year without century (00–99). Don’t use.
%Y Year with century.
%z (output only:) Offset from Greenwich: -0800 is 8 hours west of.
%z (output only:) Time zone as a character string (empty if not available).

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

Plotting:

plot(x) plot the values of x (on the y-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(x) 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)) (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 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:

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, "h": points connected by lines, "o": id. but the lines are over the points, "n": 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

xlab=, ylab= specifies the lower and upper limits of the axes, for example with xlim=c(1, 13) or xlim=range(x)

xlab=, ylab= 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 vs. 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, y1, x2, and y2, respectively

dotsplot(x, y) draws a polygon 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

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 abscissa (or ordinates) where tick-marks are drawn

rug(x) draws the data on the x-axis as small vertical lines

locator(n, type="n", ...) 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", ...)

las specifies the lower and upper limits of the axes, for example with xlim=c(1, 13) or xlim=range(x)

xlab=, ylab= 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 vs. 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, y1, x2, and y2, respectively

dotsplot(x, y) draws a polygon 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

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 abscissa (or ordinates) where tick-marks are drawn

rug(x) draws the data on the x-axis as small vertical lines

locator(n, type="n", ...) 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")

Lattice (Trellis) graphics

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

dotchart(y ~ x) histogram of the values of y with respect to those of x
dotplot(x ~ y) Cleveland dot plot (stacked plots line-by-line and column-by-column)

densityplot(x) density functions plot

bwplot(y ~ x) box-and-whiskers plot

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

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

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

dotplot(x ~ y) matrix of bivariate plots

parallel(y ~ 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, \( y \mid x \mid g_1 \mid g_2 \) has combinations of optional conditional 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 \( \texttt{?panel.par} \)). Lattice functions return an object of class trellis and have to be print-ed to produce the graph. Use \( \text{print}(\text{xyplot}(\ldots)) \) inside functions where automatic printing doesn’t work. Use \( \text{lattice.theme} \) and \( \text{set} \) to change Lattice defaults.

**Optimization and model fitting**

- \( \text{optim} \) (par, fn, method = c("Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SAHN") general-purpose optimization; \( \text{par} \) is initial values, \( \text{fn} \) is function to optimize (normally minimized)
- \( \text{nlm} \) (f, p) minimize function \( f \) using a Newton-type algorithm with starting values \( p \)
- \( \text{lm(formula)} \) fit linear models; formula is typically of the form \( \text{response} \sim \text{term1} + \text{term2} + \ldots \); use \( \text{I(x%*%y)} \) or \( \text{I(x^2)} \) 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 \( x-y \) 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: 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:

- \( \text{predict} \) (fit, \ldots) predictions from 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, \\ldots) analysis of variance (or deviance) tables for one or more fitted model objects
- \( \text{density}(x) \) kernel density estimates of \( x \)
- \( \text{binom.test} \). \( \text{pairwise.t.test} \). \( \text{power.t.test} \). \( \text{prop.test} \). \( \text{t.test} \), \ldots use help.search("\text{test}\"")

**Distributions**

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

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

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

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

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

\( \text{rt}(n, \text{df}) \) 'Student' \( t \)

\( \text{rf}(n, \text{df1}, \text{df2}) \) Fisher–Snedecor (F) \( \chi^2 \)

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

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

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

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

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

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

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

\( \text{rwilcox}(n, m, n) \). \( \text{rsignrank}(n, n) \) Wilcoxon’s statistics

All these functions can be used by replacing the letter \( v \) with \( a \), \( p \), \( q \) to get, respectively, the probability density (\( \text{dfunc}(x, \ldots) \)), the cumulative probability density (\( qfunc(x, \ldots) \)), and the value of quantile (\( qfunc(p, \ldots) \)), with \( 0 < p < 1 \).