# R quick reference card

## Session management

> **q()** Quitting R (see page 6)
> **ls()** List the objects in the current environment (see page 5)
> **rm(...)** Remove objects from the current environment (see page 5)
> **setwd(dir)** Set the current working directory (see page 6)
> **getwd()** Get the current working directory (see page 6)

## Getting help

> **?*function*** Getting help on a function (see page 7)
> **help(*function*)** Getting help on a function (see page 7)
> **example(*function*)** Run the examples associated with the manual page for the function (see page **??**)
> **demo(*topic*)** Run an installed demonstration script (see page 7)
> **apropos("*topic*")** Return names of all objects in search list that match "topic" (see page 7)
> **help.search("*topic*")** Getting help about a concept (see page 7)
> **help.start()** Launch R HTML documentation (see page 7)

## Built in constants

> **LETTERS** the 26 upper-case letters of the English alphabet (see page 14)
> **letters** the 26 lower-case letters of the English alphabet (see page 14)
> **month.name** English names of the 12 months of the year
> **month.abb** Abbreviated English names of the 12 months of the year
> **pi** $\pi$ – the ratio of a circles circumference to diameter (see page 23)

## Packages

**R CMD INSTALL package** Install an add-on package (see page 34)
> **library(*package*)** Loading an add-on package (see page 35)
> **data(*name*)** Load an data set or structure inbuilt into R or a loaded package (see page **??**)

## Importing/Exporting

> **source("*file*")** Input, parse and sequentially evaluate the file (see page 35)
> **read.table("*file*", *header=T, sep=*)** Read data in table format and create a data frame, with variables in columns (see page 39)
> **read.table("*clipboard*", *header=T, sep=*)** Read data left on the clipboard in table format and create a data frame, with variables in columns (see page 40)
> **read.systat("*file.syd*", *to.data.frame=T*)** Read SYSTAT data file and create a data frame (see page 41)
> **read.spss("*file.sav*", *to.data.frame=T*)** Read SPSS data file and create a data frame (see page 41)
> **as.data.frame(read.mtp("*file.mtp*"))** Read Minitab Portable Worksheet data file and create a data frame (see page 41)
> **read.xport("*file*")** Read SAS XPORT data file and create a data frame (see page **??**)
> **write.table(*dataframe*, "*file*", *row.names=F, quote=F, sep=*)** Write the contents of a dataframe to file in table format (see page 41)
> **save(*object*, *file="file.RData"*)** Write the contents of the object to file (see page 42)
> **load(*file="file.RData"*)** Load the contents of a file (see page 42)
> **dump(*object*, *file="file"*)** Save the contents of an object to a file (see page **??**)

## Generating Vectors

> **c(...)** Concatenate objects (see page 5)
> **seq(*from*, *to*, *by=*)** Generate a sequence (see page 10)
> **rep(*x*, *times*, *each*)** Replicate each of the values of *x* (see page 10)

## Character vectors

> **paste(..., *sep=*)** Combine multiple vectors together after converting them into character vectors (see page 11)
> **substr(*x*, *start*, *stop*)** Extract substrings from a character vector (see page 12)

## Factors

> **factor(*x*)** Convert the vector (*x*) into a factor (see page 12)
> **gl(*levels*, *reps*, *length*, *labels=*)** Generate a factor vector by specifying the pattern of levels (see page 13)
> **levels(*factor*)** Lists the levels (in order) of a factor (see page 43)
> **levels(*factor*) <-** Sets the levels (and their order) of a factor (see page 43)

## Matrices

> **matrix(*x,nrow*, *ncol*, *byrow=F*)** Create a matrix with `nrow` and/or `ncol` dimensions out of a vector (*x*) (see page 13)
> **cbind(...)** Create a matrix (or data frame) by combining the sequence of vectors, matrices or data frames by columns (see page 14)
> **rbind(...)** Create a matrix (or data frame) by combining the sequence of vectors, matrices or data frames by rows (see page 14)
> **rownames(*x*)** Read (or set with **<-**) the row names of the matrix (x) (see page 14)
> **colnames(*x*)** Read (or set with **<-**) the column names of the matrix (x) (see page 14)

## Lists

> **list(...)** Generate a list of named (for arguments in the form `name=x`) and/or unnamed (for arguments in the form `(x)`) components from the sequence of objects (see page 14)

## Data frames

> **data.frame(...)** Convert a set of vectors into a data frame (see page 37)
> **row.names(*dataframe*)** Read (or set with **<-**) the row names of the data frame (see page 38)
> **fix(*dataframe*)** View and edit a dataframe in a spreadsheet (see page 38)

## Indexing

*Vectors*
> **x[i]** Select the $i^{th}$ element (see page **??**)
> **x[i:j]** Select the $i^{th}$ through $j^{th}$ elements inclusive (see page **??**)
> **x[c(1,5,6,9)]** Select specific elements (see page **??**)
> **x[-i]** Select all except the $i^{th}$ element (see page **??**)
> **x["name"]** Select the element called `"name"` (see page **??**)
> **x[x > 10]** Select all elements greater than 10 (see page **??**)
> **x[x > 10 & x < 20]** Select all elements between 10 and 20 (both conditions must be satisfied) (see page **??**)
> **x[y == "value"]** Select all elements of x according to which y elements are equal to "value" (see page **??**)
> **x[x > 10 | y == "value"]** Select all elements which satisfy either condition (see page **??**)

*Matricies*
> **x[i,j]** Select element in row *i*, column *j* (see page **??**)
> **x[i,]** Select all elements in row *i* (see page **??**)
> **x[,j]** Select all elements in column *j* (see page **??**)

> **x[-i,]** Select all elements in each row other than the $i^{th}$ row (see page **??**)
> **x["name",1:2]** Select columns 1 through to 2 for the row named "name" (see page **??**)
> **x[x,"Var1"]>4,]** Select all rows for which the value of the column named "Var1" is greater than 4 (see page **??**)
> **x[,x[,"Var1"]=="value"]** Select all columns for which the value of the column named "Var1" is equal to "value" (see page **??**)

*Lists*

> **x[[i]]** Select the $i^{th}$ object of the list (see page **??**)
> **x[["value"]]** Select the object named "value" from the list (see page **??**)
> **x[["value"]][1:3]** Select the first three elements of the object named "value" from the list (see page **??**)

*Data frames*

> **x[c(i,j),]** Select rows *i* and *j* for each column of the data frame (see page **??**)
> **x[,"name"]** Select each row of the column named "name" (see page **??**)
> **x[["name"]]** Select the column named "name" (see page **??**)
> **x$name** Refer to a vector named "name" within the data frame (x) (see page **??**)

## Object information

> **length(*x*)** number of elements in *x* (see page 28)
> **class(*x*)** get the class of object *x* (see page 15)
> **class(*x*)<-** set the class of object *x* (see page 15)
> **attributes(*x*)** get (or set) the attributes of object *x* (see page 16)
> **attr(*x*, *which*)** get (or set) the *which* attribute of object *x* (see page 16)
> **is.na(*x*), is.numeric(*x*), is.character(*x*), is.factor(*x*), ...** methods used to assess the type of object *x* (methods(is) provides full list) (see page 16)

## Object conversion

> **as.null(*x*), as.numeric(*x*), as.character(*x*), as.factor(*x*), ...** methods used to covert *x* to the specified type (methods(is) provides full list) (see page 17)

## Data manipulations

> **subset(*x*, *subset*=, *select*=)** Subset a vector or data frame according to a set of conditions (see page 44)
> **sample(*x*, *size*)** Randomly resample size number of elements from the x vector without replacement. Use the option replace=TRUE to sample with replacement. (see page **??**)
> **apply(*x*, *INDEX*, *FUN*)** Apply the function (*FUN*) to the margins (INDEX=1 is rows,INDEX=2 is columns, INDEX=c(1,2) is both) of a vector, array or list (*x*) (see page **??**)
> **tapply(*x*, *factorlist*, *FUN*)** Apply the function (*FUN*) to the vector (*x*) separately for each combination of the list of factors (see page 45)
> **lapply(*x*, *FUN*)** Apply the function (*FUN*) to each element of the list x (see page 45)
> **replicate(*n*, *EXP*)** Re-evaluate the expression (EXP) *n* times. Differs from rep function which repeats the result of a single evaluation (see page **??**)
> **aggregate(*x*, *by*, *FUN*)** Splits data according to a combination of factors and calculates summary statistics on each set (see page 45)
> **sort(*x*, *decreasing*=)** Sorts a vector in increasing or decreasing (default) order (see page 21)
> **order(*x*, *decreasing*=)** Returns a list of indices reflecting the vector sorted in ascending or descending order (see page 45)
> **rank(*x*, *ties.method*=)** Returns the ranks of the values in the vector, tied values averaged by default (see page 21)
> **which.min(*x*)** Index of minimum element in x (see page **??**)
> **which.max(*x*)** Index of maximum element in x (see page **??**)
> **rev(*x*)** Reverse the order of entries in the vector (x) (see page 21)
> **unique(*x*)** Removes duplicate values (see page **??**)
> **t(*x*)** Transpose the matrix or data frame (x) (see page **??**)
> **cut(*x*, *breaks*)** Creates a factor out of a vector by slicing the vector x up into chunks. The option breaks is either a number indicating the number of cuts or else a vector of cut values (see page **??**)
> **which(*x* == *a*)** Each of the elements of x is compared to the value of a and a vector of indices for which the logical comparison is true is returned (see page **??**)
> **match(*x*,*y*)** A vector of the same length as x with the indices of the first occurance of each element of x within y (see page **??**)
> **choose(*n*,*k*)** Computes the number of unique combinations in which k events can be arranged in a sequence of n (see page **??**)
> **combn(*x*,*k*)** List all the unique combinations in which the elements of x can be arranged when taken k elements at a time (see page **??**)

## Formating data

> **ceiling(*x*)** Rounds vector entries up to the nearest integer that is no smaller than the original vector entry (see page 22)
> **floor(*x*)** Rounds vector entries up to the nearest integer that is no smaller than the original vector entry (see page 22)
> **trunc(*x*)** Rounds vector entries to the nearest integer towards '0' (zero) (see page 22)
> **round(*x*, *digits*=)** rounds vector entries to the nearest numeric with the specified number of decimal places (digit=). Digits of 5 are rounded off to the nearest even digit (see page 22)
> **formatC(*x*, *format*=, *digits*=, ...)** Format vector entries according to a set of specifications (see page 23)

## Math functions

*Summary statistics*

> **mean(*x*)** Mean of elements of x (see page **??**)
> **var(*x*)** Variance of elements of x (see page **??**)
> **sd(*x*)** Standard deviation of elements of x (see page **??**)
> **length(*x*)** Number of elements of x (see page **??**)
> **sd(*x*)/sqrt(length(*x*))** Standard error of elements of x (see page **??**)
> **quantile(*x*, probs=)** Quantiles of x corresponding to probabilities (default: 0,0.25,0.5,0.75,1) (see page **??**)
> **median(*x*)** Median of elements of x (see page **??**)
> **min(*x*)** Minimum of elements of x (see page **??**)
> **max(*x*)** Maximum of elements of x (see page **??**)
> **range(*x*)** Same as c(min(x), max(x)) (see page **??**)
> **sum(*x*)** Sum of elements of x (see page **??**)
> **cumsum(*x*)** A vector the same length as *x* and whose $i^{th}$ element is the sum of all elements up to and including *i* (see page **??**)
> **prod(*x*)** Product of elements of x (see page **??**)
> **cumprod(*x*)** A vector the same length as *x* and whose $i^{th}$ element is the product of all elements up to and including *i* (see page **??**)
> **cummin(*x*)** A vector the same length as *x* and whose $i^{th}$ element is the minimum value of all elements up to and including *i* (see page **??**)
> **cummax(*x*)** A vector the same length as *x* and whose $i^{th}$ element is the maximum value of all elements up to and including *i* (see page **??**)
> **var(*x*,*y*)** variance between *x* and y (matrix if x and y are matrices of data frames) (see page **??**)
> **cov(*x*,*y*)** covariance between *x* and y (matrix if x and y are matrices of data frames) (see page **??**)
> **cor(*x*,*y*)** linear correlation between *x* and y (matrix if x and y are matrices of data frames) (see page **??**)

*Scale trasformations*

> **exp(*x*)** Transform values to exponentials (see page **??**)
> **log(*x*)** Transform values to $log_e$ (see page **??**)
> **log(*x*, 10)** Transform values to $log_1 0$ (see page **??**)
> **log10(*x*)** Transform values to $log_1 0$ (see page **??**)
> **sqrt(*x*)** Square root transform values of x (see page **??**)
> **asin(sqrt(*x*))** Arcsin transform values of x (which must be proportions) (see page **??**)
> **rank(*x*)** Transform values of x to ranks (see page **??**)
> **scale(*x*, center=, scale=)** Scales (mean of 0 and sd of 1) values of x to ranks. To only center data, use scale='FALSE', to only reduce data use center='FALSE' (see page **??**)

## Distributions

*The following are used for the following list of distribution functions*

**x=** a vector of quantiles
**q=** a vector of quantiles
**p=** a vector of probabilities
**n=** the number of observations

> **dnorm(x, mean, sd), pnorm(q, mean, sd), qnorm(p, mean, sd), rnorm(n, mean, sd)** Density,

distribution function, quantile function and random generation for the normal distribution with mean equal to `mean` and standard deviation equal to `sd` (see page **??**)

> **`dlnorm(x, meanlog, sdlog), pnorm(q, meanlog, sdlog), qnorm(p, meanlog, sdlog), rnorm(n, meanlog, sdlog)`** Density, distribution function, quantile function and random generation for the log normal distribution whose logarithm has a mean equal to `meanlog` and standard deviation equal to `sdlog` (see page **??**)

> **`dunif(x, min, max), punif(q, min, max), qunif(p, min, max), runif(n, min, max)`** Density, distribution function, quantile function and random generation for the uniform distribution with a minimum equal to `min` and maximum equal to `max` (see page **??**)

> **`dt(x, df), pt(q, df), qt(p, df), rt(n, df)`** Density, distribution function, quantile function and random generation for the t distribution with `df` degrees of freedom (see page **??**)

> **`df(x, df1, df2), pf(q, df1, df2), qf(p, df1, df2), rf(n, df1, df2)`** Density, distribution function, quantile function and random generation for the F distribution with `df1` and `df2` degrees of freedom (see page **??**)

> **`dchisq(x, df), pchisq(q, df), qchisq(p, df), rchisq(n, df)`** Density, distribution function, quantile function and random generation for the chi-squared distribution with `df` degrees of freedom (see page **??**)

> **`dbinom(x, size, prob), pbinom(q, size, prob), qbinom(p, size, prob), rbinom(n, size, prob)`** Density, distribution function, quantile function and random generation for the binomial distribution with parameters `size` and `prob` (see page **??**)

> **`dpois(x, lambda), ppois(q, lambda), qpois(p, lambda), rpois(n, lambda)`** Density, distribution function, quantile function and random generation for the Poisson distribution with parameter `lambda` (see page **??**)

## Spatial procedures

`sp` package

> **`Polygon(xy)`** Convert a 2-column numeric matrix (`xy`) with coordinates. Note the first point (row) must be equal to the last coordinates (row) (see page **??**)

> **`spsample(x, n, type=)`** Generate approximately `n` points on or within an spatial object (`x`). The option `type=` indicates the type of sampling (`"random"`, `"regular"`, `"stratisfied"` or `"non-aligned"`) (see page **??**)

## Plotting

> **`hist(x, breaks)`** Histogram of the frequencies of vector `x`. The option `breaks` specifies how the bins are constructed and is typically either a number (number of bins), a vector of breakpoints (see page **??**)

> **`plot(x)`** Plot the values of `x` (on *y*-axis) ordered on *x*-axis (see page **??**)

> **`plot(x, y)`** Scatterplot of `y` (on *y*-axis) against `x` (*x*-axis) (see page **??**)

> **`plot(formula)`** If all vectors numeric - Scatterplot of `lhs` (on *y*-axis) against `rhs` (*x*-axis), otherwise a "box-and-whisker" plot with a separate box for each combination of `rhs` categories (see page **??**)

> **`boxplot(x)`** "Box-and-whiskers" plot for vector or formula `x` (see page **??**)

> **`pairs(x)`** Scatterplot matrices for multiple numeric vectors or formula `x` (see page **??**)

> **`Mbargraph(dv, iv)`** Bargraph (*biology package*) of mean `dv` against categorical `iv` with error bars (see page **??**)

> **`interaction.plot(x.fact, trace.fact, response)`** Plots the mean (or other summary) of the response (`response`) for two-way combinations of factors (*x*-axis factor: `x.fact` and trace factor: `trace.fact`), thereby illustrating possible interactions (see page **??**)

> **`scatterplot(x)`** (`car package`) Fancy scatterplot for a pair of numeric vectors or formula `x`. Includes boxplots on margins and regression line (see page **??**)

> **`scatterplot.matrix(x)`** (`car package`) Fancy scatterplot matrices for multiple numeric vectors or formula `x`. Includes univariate displays in diagonals (see page **??**)

## Low-level plotting commands

> **`points(x, y)`** Adds points with coordinates `x, y`. Option `type=` can be used (see page **??**)

> **`lines(x, y)`** Adds lines with coordinates `x, y`. Option `type=` can be used (see page **??**)

> **`abline(fit)`** Adds a regression line from the linear model `fit` (see page **??**)

> **`abline(a, b)`** Adds a regression line with a *y*-intercept of `a` and a slope of `b` (see page **??**)

> **`axis(text, at, labels, ...)`** Adds an axis to the bottom (`side=1`), left (`side=2`), top (`side=3`) or right (`side=4`) plot margin. Options `at` and `labels` can be used to specify where to draw tick marks and what labels to put at each tick mark (see page **??**)

> **`box(which=, bty=, ...)`** Draws a box around the plot (`which="plot"`), figure (`which="figure"`), inner (`which="inner"`) or outer (`which="outer"`) region of the current plot. Option `bty` specifies the type of box to draw (`"o"`, `"l"`, `"7"`, `"c"`, `"u"` or `"]"` result in boxes that resembles the corresponding upper case letter) (see page **??**)

> **`mtext(text, side, line=0, ...)`** Adds text (`text`) to the plot margin specified by `side` (see `axis()` above). Option `line` specifies the distance (in lines) away from the axis to put the text (see page **??**)

## Graphical parameters

The following parameters can be set globally using the `par()` function. Many can also be passed as arguments to other plotting commands.

> **`adj`** Text justification (`0`: left-justified, `0.5`: centred, `1`:right=justified). Any other value between 0 and 1 allowed (see page**??**)

> **`bg`** Background color (e.g. `bg="red"`). `colors()` lists the 657 available colors (see page**??**)

> **`bty`** Type of box drawn around plots (`"o"`, `"l"`, `"7"`, `"c"`, `"u"` or `"]"` result in boxes that resembles the corresponding upper case letter) (see page**??**)

## Model fitting

> **`lm(formula)`** Fit linear model from `formula` of format `response ~ predictor1 + predictor2 + ...`. use `I(x*y) + I(x^2)` to include nonlinear terms (see page **??**)

> **`glm(formula, family)`** Fit generalized linear model from `formula`. Error distribution and link function are specified by `family` - see `family()` (see page **??**)

> **`aov(formula)`** Fit an anova model by making a call to `lm` for each stratum within `formula` (see page **??**)

> **`nls(formula, start)`** Determine the nonlinear least-squares estimates of the parameters of a nonlinear model `formula`. Starting estimates are provided as a named list or numeric vector (`start`) (see page **??**)

> **`plot(fit)`** Diagnostic plots for a fitted model `fit` (see page **??**)

> **`av.plots(fit)`** Added-variable (partial-regression) plots for a fitted model `fit` (see page **??**)

> **`residuals(fit)`** Residuals from a fitted model `fit` (see page **??**)

> **`deviance(fit)`** Deviance of a fitted model `fit` (see page **??**)

> **`influence.measures(fit)`** Regression diagnostics for a fitted model `fit` (see page **??**)

> **`vif(fit)`** Calculate variance-inflation factor for a fitted model `fit` (see page **??**)

> **`1/vif(fit)`** Calculate tolerance for each term in a fitted model `fit` (see page **??**)

## Statistics

> **`t.test(x, y), t.test(formula)`** One and two sample t-tests on vectors (`x, y`) or formula `formula`. Option `var.equal` indicates whether pooled or separate variance t-test and option `paired` indicates whether independent or paired t-test (see page **??**)

> **`wilcox.test(x, y), t.test(formula)`** One and two sample ("Mann-Whitney") Wilcoxon tests on vectors (`x, y`) or formula `formula`. Option indicates whether independent or paired Wilcoxon-test (see page **??**)

> **`hier.part(y, data, gof)`** (*hier.part package*) Hierarchical partitioning given a vector of dependent variables `y` and a data frame `data`. Option `gof=` used to specify assessment of

fit (root mean square prediction error: `"RMSPE"`, Log-Likelihood: `"logLik"` or R-squared: `"Rsqu"`) (see page **??**)

> **`simtest(formula, whichf=, type=)`** (*multcomp package*) Post-hoc, pairwise comparisons of factor (`whichf`). Option `type` specifies what type of post-hoc test to perform (`"Dunnett"`, `"Tukey"`, `"Sequen"`, `"AVE"`, `"Changepoint"`, `"Williams"`, `"Marcus"`, `"McDermott"`) **??**)

> **`anova(fit, ...)`** Compute analysis of variance table for a fitted model `fit` or models (see page **??**)

> **`summary(fit)`** Summarize parameter estimates for a fitted model `fit` (see page **??**)

## Power analysis

> **`power.t.test(n, delta, sd, power)`** Calculate one of; sample size (`n`), true difference in means (`delta`), standard deviation (`sd`) or power (`power`) of t-test. The option `type` indicates the type of t-test (`"two.sample"`, `"one.sample"`, `"paired"`) (see page **??**)

> **`power.r.test(n, r, power)`** (*pwr package*) Calculate one of; sample size (`n`), correlation coefficient (`r`) or power (`power`) of t-test. (see page **??**)