Short Guide to the Most Essential R Functions

In the table below: LOGICAL – logical test, such as is.na(data) or data==1; ACTION – an executable expression, such as data<-3 or lm(y~x) or 2+4; BODY – set of expressions; path – access path to a file; [requires NAME] – installation of the NAME package is reuired; FORMULA – formula object; MODEL – model object; NAME – any custom NAME. Anu numbers indexing lists of commands in the first column are only for reference and should not be used with commands provided.

Function and arguments	Description and details
Operators and basic operations	
!x, x|y, x&y, xor(x,y)	NOT x, x OR y, x AND y, logical exclusive OR on x, y
#	Comment line – not executed
+, -, *, /, %%, %/%, %*%, ^	add, subtract, multiply, divide, modulo, integer division, matrix product, power
==, >, <, >=, <=, !=	Equal, smaller than, larger than, smaller or equal, larger or equal, not equal
A -> B	Assignemnt – B gets the value of A
abs(NAME)	Absilute value
cor(NAME1,NAME2)	Correlation of elemenets of two objects
cov(NAME1,NAME2)	Covariance of elements of two objects
exp(NAME)	Exponent (e^{NAME})
Inf, NA, NaN	Infinity, missing value, not-a-number variable
install.packages("NAME")	Install a package "NAME"
is.na(NAME)	Logical test if NAME is a missing value
library(NAME)	Load a library NAME
list.files()	List all files in the current working directory
log(NAME)	Logarithm of NAME
ls()	Display all object in the workspace
mean(NAME)	Mean of elements of name
median(NAME)	Median of elements of name
prod(NAME)	Product of elements of NAME
quantile(NAME)	Quantiles (median, minimum, maximum, 25% and 75% quantile)
round(x, digits=n)	Round x to n digits
save(file="NAME")	Save workspace to file
savehistory(file="NAME")	Save history of commands to file

sd(NAME)	Standard deviation of elements of NAME
search()	Display the namespace and all loaded packages and attached objects
setwd(path)	Set working directory to path
sqrt(NAME)	Square-root of NAME
sum(NAME)	Sum of elements of NAME
T or TRUE, F or FALSE	Logical variable – true or false
var(NAME)	Variance of elements of NAME
Vector and matrix functions; data-type functions	
as.vector(X), as.list(X), as.matrix(X), as.data.frame(X), as.array(X), as.numeric(X), as.character(X), as.logical(X) as.factor(X)	Treat X as the type specified without changing its type
c(a,b,c,d,...)	Concatenate obejcts to a vector
class(), attributes()	Check class and attributes of an object
cumprod(VECTOR)	Cumulatiove product of elements of VECTOR
cumsum(VECTOR)	Cumulative sum of elements of VECTOR
det(MATRIX)	Determinant of MATRIX
dim(ARRAY)	Returns lengths of dimensions of ARRAY (may also be matrix and data-frame)
eigen(MATRIX)	Eigenvalue of MATRIX
fix(NAME)	Opens window for manual edition of the table NAME
is.vector(), is.list, etc.	Logical test if object is of type specified
length(VECTOR)	Number of elements in a VECTOR
max(NAME)	Maximum value of NAME
min(NAME)	Minimum value in NAME
names(NAME)	Names of the elements of the vector or variables of the data-frame – you can assign new values
order(VECTOR)	Returns permutation of elements that – when applied as and index – sorts elements of VECTOR ascending
paste(VECTOR, sep=”.”)	Paste elements of VECTOR as a text string using sep as separators (may also be “”)
range(VECTOR)	The range of values
rank(VECTOR)	Ranks of values
rev()	Reverses a function, eg. Rev(sort(x)) sorts x descending
rownames(NAME), colnames(NAME)	Returns names of columnsand rows of thematrix or data-frame; may also be used for assigning names

`sort(VECTOR)`	Sorts elements ascending
`summary(NAME)`	Generic function, returns type-specific summary
`t(MATRIX)`	Transpose a matrix
`which(VECTOR, LOGICAL)`	Indexes of elements satysying the condition LOGICAL
Reading data; manipulating tables	
`$ e.g. data$name`	Accesses the variable using its name (in data-frames)
`[] e.g. data[2,3]`	Accesses column, row or element; in >2D objects dimensions are specified in the order: rows, columns, …; omitting one dimension but retaining commas means that we want the whole dimension extracted
`apply(matrix, 1 or 2, FUNCTION)`	Applies FUNCTION to rows (1) or columns (2) of matrix

`1. attach(NAME), detach(NAME)`
`2. detach(package:NAME)`
	1. Attaches or detaches an object
	2. Detaches package NAME

`boxcox(NAME)`	[requires MASS] Box-Cox transformation of the data
`cbind(x,y)`	Column-wise bind of two objects (numbers of rowns must be the same)
`na.omit(NAME)`	Returns object with NAs removed; in data-frame whole rows in at least one NA are removed
`rbind(x,y)`	Row-wise bind two objects; numbers of coulmns are the same
`read.csv(file=path)`	Read CSV (comma-separated) file
`read.delim2()`	Read file with commans as decimal separators; arguments as in read.table()
`read.table(path, header=T, sep=”\t”, skip=N)`	Read file in path, header=T sets the first line as names of variables, sep sets the character separating columns, skip skips N first columns
`subset(NAME, LOGICAL)`	Extract from data-frame NAME cases satisfying LOGICAL condition, eg. subset(data, sex==”M”)
`table(group1, group2)`	Create contingency table counting cases in grouping variables (one or two)
`tapply(data, group, FUNCTION)`	Apply function to data group-wise
`with(NAME, procedures)`	Alternative for attach; procedures use data from NAME without the need of specifying variable names by $
`write.table(data, file=path, sep=”\t”)`	Save data to disc using filename path and sep as column separator
Writing new functions	
`break`	Break lood and go outside to the next operation

F <- function(ARGUMENTS) {BODY}	Define function F, taking several ARGUMENTS (names, comma separated), executing some expressions using these arguments in BODY

for (i in X) {ACTIONS}
for (i in X) ACTION
	Loop – iterate through elements of X (may be vector or range), for each execute ACTIONS or single ACTION

1. if (LOGICAL) {ACTIONS} 2. if (LOGICAL) {ACTIONS} else {ACTIONS} 3. ifelse (LOGICAL, ACTIONS1, ACTIONS2)	1. Execute ACTIONS if LOGICAL is TRUE 2. See above, if FALSE execute else 3. Execute ACTIONS1 if LOGICAL is TRUE, execute ACTIONS2 otherwise
next	Stop iteration and go to the next one (does not break the entire loop)
repeat {ACTION if (LOGICAL) break}	Execute ACTION as long as LOGICAL remains false
while (LOGICAL) {ACTIONS}	Execute ACTIONS as long as LOGICAL remains TRUE
Generating random data	
rep(A, length.out=B, times=C, each=D)	Repeat A C times, or as many times as necessary to fill length.out; if each defined – each element of A (if it's a vector) will be repeated D times; e.g. rep(c(1,2),times=2,each=4) yields 1111222211112222
rnorm(N, mean, sd), pnorm(X, mean, sd), qnorm(P, mean, sd), dnorm(X, mean, sd)	Use normal distribution with parameters mean and sd to: generate N random samples (r); get probability x<=X (p); get quantile X for P(x<=X) (q); get the density function for X (d); see help for more arguments, e.g. log=T yields log transformed values
OTHER DISTRIBUTIONS [add r, q, p or d; first argument may be P, X or N]: t(., df), f(., df1, df2), binom(., size, probab), pois(., lambda), gamma(., shape, scale), chisq(., df), nbinom(., size, probab, mu), lnorm(., meanlog, sdlog), hyper(., m, n, k), geom(., probab), multinom(., size, prob), logis(., location, scale), exp(., rate), cauchy(., location, scale), unif(., a, b)	*t* distribution, *F*, binomial, Poisson, gamma, Chi-squared, negative binomial, lognormal, hypergeometric, geometric, multinomial, logistic, exponential, Cauchy, uniform. See respective help files for more details and arguments.
rTraitCont(tree, model, sigma, alpha)	Simulate evolution along the tree phylogeny, using selected evolution model, sigma as standard deviation for random process at each branching and alpha as slelective force acting along the tree

rtree()	Generate random tree; see help for more details
sample(A, B, replace=T or F)	Choose random sample of size B from vector A, if replace TRUE each element will may be sampled more tha once; executing with replace=F and B>length(A) yields error
1. seq(A, B, by=C) 2. seq(A, B, length.out=C)	1. Generate numbers between A and B with increment of by 2. Generate sequence between A and B of the final length of length.out If A>B the sequence is generated in descending order
unique(A)	Extract all unique values from A
Hypothesis testing Most testing functions accept the following arguments: alternative="two-sided" or "less" or "greater" (one or two-tailed test); conf.level=0.95 specifying significance threshold.	
binom.test(n_succ, n_trials, P)	Binomial test for population with P successes
chisq.test(x,y) or chisq.test(A)	Accepts two vectors or a matrix (contingency table)
cor.test(x,y,method)	Correlation test; available methods: spearman, kendall, pearson
fischer.test()	Exact Fisher test, takes two vectors or one matrix
kruskal.test()	Kruskal-Wallis test; takes one list with groups as subvectors, two vectors – one with data nad one with group ids or formula object
ks.test()	Takes two vectors with data or one vector and the name of distribution to test (e.g. ks.test(x,pnorm)
prop.test()	Propotion test
qqnorm(), qqline()	Give quantile-quantile plot testing for normality and adds a line to it
shapiro.test()	Shapiro-Wilk test for normality, takes one vector of data
t.test(A,B,var.equal=T or F, paired=T or F)	t-test, takes two vectors of formula object
TukeyHSD()	Tukey Honest Significant Difference; takes anova or lm model object
var.test()	Takes two vectors and compares variances using F-test
wilcox.test(A,B,paired=T or F)	Wilcoxon signed-rank test – takes two vectors
power.t.test(delta=A, sd=B, power=C, n=D, sig.level=E, alternative=F)	Power calculation. Specify all parameters but one and it will be estimated based on the remaning ones. See help for detailed description of arguments.
Bootstrapping	

a <- numeric(N)
for (i in 1:N) {
a[i] <- STATISTIC using
sample(data,replace=T) }
hist(a)
quantile(a, c(0.025, 0.975))
	Sample bootstrapping with N randomizations using sample function; STATISTIC is the expression calculating the value of test statistic; hist generates histogram of bootstrapped samples; quantile allow for hypothesis testing

FUNCTION <- function(A,i)
STAT(A[i])
BOOT <- boot(data, FUNCTION, N)
	[requires boot] First, the FUNCTION is defined – it calculates the test statistic. Then it is bootstrapped.

boot.ci(BOOT)	Confidence intervals from bootstrapping.
Linear models	
FORMULA 1. y ~ x 2. x + y 3. x:y 4. x*y 5. x – y 6. x/y 7. 1 8. (x + y + z)^2 9. poly(x, 2, raw=T) or x+ I(x^2) 10. s(x) 11. lo(x)	1. Simple formula, with independent (x) and dependent (y) variable 2. + defines additional variables 3. colon forms interaction 4. * fits interaction and all main effects 5. – removes a term 6. Slash defines nesting, from higher to lower level 7. One represents intercept 8. Fits all two factor interactions of x, y, z and main effects 9. Fits quadratic term of x 10. Uses smoother to fit x (in GAM) 11. Uses LOESS (local regression) to fit x (in GAM)
lm(FORMULA, data=NAME, weights=A)	Linear model for data, weights optional
predict(MODEL, newdata)	Prediction from model; if newdata specified (as additional data-frame) prdictions for new values are made
resid(MODEL)	Residuals from model
update(MODEL, ~. -A)	Update model's formula
summary(MODEL)	Summary of model
plot(MODEL)	Diagnostic plots
anova(MODEL)	ANOVA table for model (if supported)
anova(MODEL1, MODEL2)	Compare two models using ANOVA
gam(FORMULA, data)	[requires mgcv] Additive linear models
tree(FORMULA, data)	[requires tree] Tree regression models
plot(TREEMODEL), text(TREEMODEL)(	Plots tree regression and adds text labels
step(MODEL)	Stepwise simplification of MODEL based on AIC
contrasts(DATA$FACTOR)	Displays contrasts for factor variable
contrasts(DATA$FACTOR) <- metrix of contrasts	Sets contrasts for factor variable

summary.lm(MODEL)	Regression-like summary of a model
summary.aov(MODEL)	ANOVA-like summary of a model
glm(FORMULA, data=NAME, family=distribution name)	Generalized linear model with distribution defined by family; possible values: gaussian, poisson, binomial, exponential, gamma, quasibinomial, quasipoisson.
MCMCglmm(	Fits generalized linear mixed models using Markov Chain Monte Carlo method
y ~ fixed effects OR cbind(y, z) ~ trait + fixed effects,	Fixed effects formula; cbind() used if more than two response variables; trait is a restricted name indexing response variables in multivariate models

random=~a + b OR
random=~idh(fixed):a + us(fixed):b OR
random=~idh(trait):a + us(trait):b,
	Random effects formula; idh used for covariance structures with covariances set to zero; us used for (co)variance structures with covariances not fixed; in random effects – animal used for additive genetic/phylogenetic effect in animal models; be sure to create proper structure in multivariate models (hence the 'trait' effect)

rcov=~idh(fixed):units,	Optional, defines residual (co)variance structure
data=NAME,	Name of the data object
pedigree=NAME,	Optional, name of the pedigree datafile/phylogenetic tree from ape()
mev=NAME,	Optional, in meta-analysis defines vector of measurements error

family=NAMES OR
family=c(NAME,NAME),
	Defines the type of distribution; c() used when more than one response; not necessary if gaussian

prior=NAME,	Defines the name of the prior
saveX=T or F, saveZ=T or F,	Saves (if T) design matrices for fixed and random effects
pr =T or F, pl=T or F)	Saves (if T) random effects (BLUPs) and latent variables (fitted values on link scale)
my_prior <- list(R=list(V=1,nu=0.002), B=list(mu=0, V=1e+06), G=list(G1=list(V=1,nu=0.002), G2=list(V=1,fix1), G3=list(V=1,nu=0.002,alpha.mu=0, alpha.V=1000)))	Prior for MCMCglmm; R – priors for residual variance; G – priors for random effects (as many as there are random terms); B – priors for fixed effects (if more than one: mu=c(0,0,0), V=diag(3)*1e+06); B is optional and required only in difficult models (such as binary data with large separation; see relevant chapters)
fitted(MODEL)	Returns values fitted by model (equal to predict() with no newdata argument)
lmer(y ~ x + y + (1|a) + (fixed|b), family=distribution name, data=NAME)	Fits (generalized) linear mixed models using REML; random effects formed by (X|...)
mcmcsamp(MODEL from lmer)	[requires arm] Uses lmer object to create MCMC samples for estimated parameters
Graphics and plots	

plot(x,y OR y~x OR object,	Generic function for creating plots; takes two vectors (x and y variables), a formula object or a (model) object.
main,	Graph title
xlab,	x axis label
ylab,	y axis label
xlim,	Limits for x axis in the form of c(A,B)
ylim,	Limits for y axis
cex.axis,	Font size for axes' ticks in points
cex.lab,	Font size for axes' labels in points
cex.main,	Font size for graph's label
cex,	Size of graph's points
pch,	Type of points (see points() function)
lty,	Line type for line plots (see lines())
lwd)	Line width in pixels

abline(a=X,b=Y)
abline(h=A)
abline(v=B)
abline(lm model)
	Adds line to a plot, by defining slope and intercept (a,b), horizontal line for Y=A, vertical line for X=B or line from a lm object

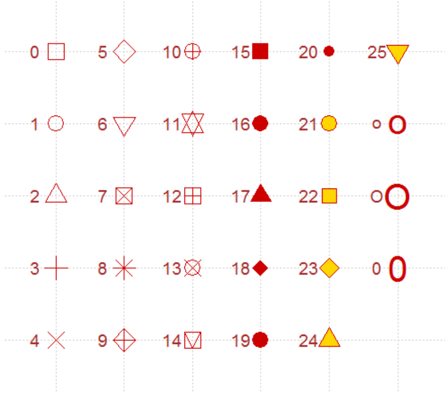
boxplot(Y~X)	Creates boxplot fro data given group(X)-wise
hist(X, freq=T or F, breaks=N)	Histogram (with frequencies if freq=T), with custom number of bars (breaks)
identify(x,y)	Identifies points on the graph
legend(x,y,legend)	Adds a legend to the graph
library(lattice) and library(gplot)	Two libraries for high-level specialized graphs (see manuals and help files)

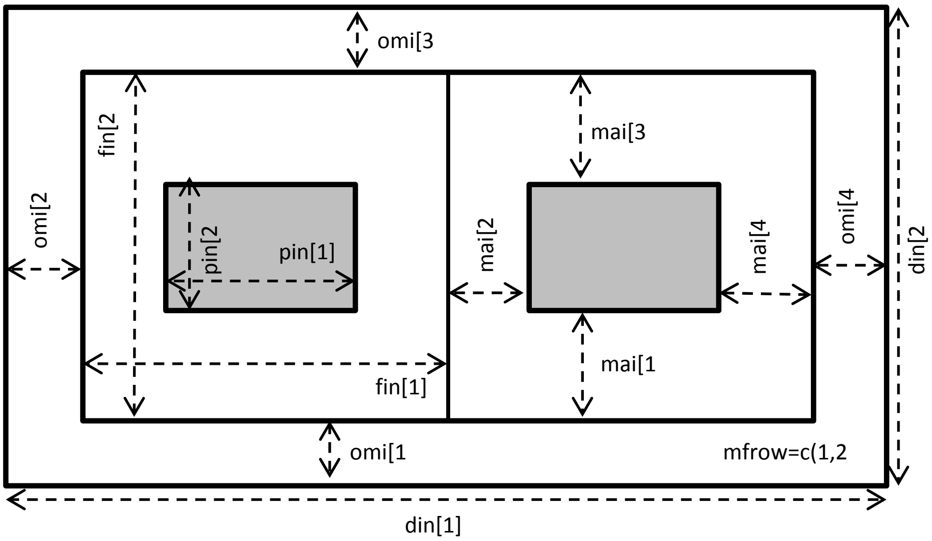
lines(x,y,lty=N)	Adds lines to a plot. Types of lines (lty):
	lty=1 solid line
	lty=2 dashed line
	lty=3 dotted line
	lty=4 dash-and-dot line
	lty=5 broken line
	lty=6 broken-and-dot line

locator(x)	Identifies points on the graph
par(	Sets graphical parameters (see figure below)
font,	1-standard, 2-*italic*, 3-**bold**, 4-***bold italic***, 5-σψμβολ
mar, mai,	Width of margins in Inches or lines, as four-element vectors
mfrow,	Sets number of columns and rows on the plot
oma, omi,	Widths of outer margins in Inches or lines, as four-element vectors
din, fin, pin)	Length and width of the image (in Inches or lines) as two-element vectors

mfg	Position of active figure in device with multiple figures
persp(x,y,z)	3D plot, with x and y independent variables and one dependent variable z

png(file=path) PLOTTING dev.off()
jpeg(file=path) PLOTTING dev.off()
pdf(file=path) PLOTTING dev.off()
	Using devices for saving graphs to graphic files; can also be done using Save As menu in the R Console (Windows/Mac OS)

points(x, pch=N)	Adds points to the graph. Type of pints (pch): 
rainbow(N), heat.colors(N), terrain.colors(N), cm.colors(N)	Generates color vectors of size = N


omi[3
fin[2
omi[2
pin[2
pin[1]
mai[3
mai[2
mai[4
omi[4
din[2
fin[1]
mai[1
omi[1
mfrow=c(1,2
din[1]