

# One Page R Reference Card

Alexey Shipunov, April 5, 2018

`c(a, b)`: concatenate, vectorize  
`#`: comment char  
`install.packages("package1")`: install package `package1`  
`is.na(x1)`: TRUE if `x1 == NA`  
`library(package1)`: load package `package1`  
`NA`: missing value  
`options(scipen=100)`: get rid of logarithmic notation  
`<-` or `assign()`: assign  
`q("no")`: quit R, do not save environment  
`history(Inf)`; `savehistory(file1)`: look on command history;  
save history (not on macOS GUI)  
; used to separate commands    `::` used in `package::command()`  
Tab; Up; Ctrl+U: completion; repeat command; delete command  
(not on macOS)

## Help

`example(com1)`: run examples for command `com1`  
`help(com1)` or `?com1`: help about command `com1`  
`help(package=rpart)`: help for the package, e.g. `rpart`  
`function1`; `methods(function2)`; `:::` or  
`getAnywhere(method2)`: look on the `function1` and 2 codes  
`??"topic1"`: finds `topic1` in all help files (slow!)

## Entering and saving data

`dir(...)` and `setwd()`: list files in directory, go to another  
`read.table("file1", h=T, sep=";", as.is=T)`: read data into  
data frame from `file1` which has header and semicolon for  
separator; do not convert variables to factors  
`scan("file1", what="char")`: read one series of character codes  
from disk into variable  
`sink("file1")`: output to `file1` until `sink()`  
`source("file1.r")`: run commands from file `file1.r`  
`write.table(x1, "file1")`: write object `x1` to the file `file1`

## Manage variables and objects

`1:3` or `c(1, 2, 3)`: concatenate 1, 2, 3 into vector  
`as.data.frame(x1)`, `as.matrix(x1)`: conversion  
`cbind(a1, b1, c1)` or `rbind(a1, b1, c1)`: join columns or  
rows into matrix  
`cut(v1, c(0, 100, 200), labels=c("small", "big"))`: split  
vector `v1` in two intervals  
`data.frame(v1, v2)`: list from same-length vectors `v1` and `v2`  
`df1$a1`: variable (column) `a1` in data frame `df1`  
`dimnames(mat1)`, or `names(df1)` and `row.names(df1)`: names of  
rows and columns of `mat1` or data frame `df1`  
`droplevels(factor1)`: drop unused levels  
`grep("str1", x1)`: search `str1` in `x1`  
`gsub("str1", "str2", x1)`: replace `str1` to the `str2` in `x1`  
`head(df1)`: first rows of data frame  
`length(v1)`: length of vector `v1`  
`list1[[-5]]`: all elements of list except fifth  
`ls()`: list all active objects  
`mat1[, 2:5]` or `mat1[, c(2, 3, 4, 5)]`: columns 2–5  
`matrix(vector1, r1, c1)`: `vector1` to matrix with `r1` rows and  
`c1` columns  
`merge(df1, df2)`: merge two data frames  
`nrow(mat1)`: how many rows?  
`paste("cow", "boy", sep="")` outputs "cowboy"  
`rep(x1, n1)`: repeat vector `x1` `n1` times  
`sample(x1, n1)`: sample `n1` elements from `x1`  
`seq(n1, n2, n3)`: sequence from `n1` to `n2` by `n3` steps  
`stack()` and `unstack()`: convert from short to long format and  
back again  
`str(obj1)`: structure of object `obj1`  
`t(mat1)`: transpose (rotate 90°) matrix or data frame  
`with(x1, ...)`: do something within environment of `x1`

## Cycles, conditions and functions

`plot(..., pch=..)`: 1 ○ 2 △ 3 + 4 × 5 ◇ 6 ▽ 7 ☒ 8 \* 9 ⊕ 10 ⊕ 11 ⊗ 12 ⊞ 13 ⊗ 14 ⊞ 15 ■  
16 ● 17 ▲ 18 ◆ 19 ● 20 ● 21 ○ 22 □ 23 ◇ 24 △ 25 ▽ \* \* . . a a ? ? 0 □

`for (i1 in sequence1) command1`: cycle  
`fun1 <- function(args1) something1`: define function  
`if (condition1) ...else ...`: condition  
`ifelse(condition1, yes, no)`: vectorized condition

## Logic and math

`is.factor(obj1)`, `is.matrix(obj1)`, `is.vector(obj1)`: check  
the type of object `obj1`  
`mat1[mat1 > 0]`: elements of `mat1` which are positive  
`!<`, `&`, `|`, `==`: "not less", "and", "or", "equal"  
`round(x1)`: round  
`unique(x1)`: list unique elements of `x1`  
`*`, `^`, `sqrt(pi)`, `abs(-3)`, `log(1)`: multiplication, degree,  $\sqrt{\pi}$ ,  
3, natural logarithm  
`x1 %in% x2`, `match(x1, x2)`: if elements of `x1` are in `x2`

## Descriptive statistics

`aggregate(...)`: pivot table  
`apply(x1, n1, f1)`: apply function `f1` (e.g., `mean`) to all rows (if  
`n1 = 1`) or columns (`n1 = 2`) of `x`  
`colSums(mat1)`: calculate sum of every column  
`rev(x1)`, `order(x1)`, `scale(x1)`, `sort(x1)`: reverse etc.  
`sapply(...)`; `lapply(...)`; `do.call(...)`; `replicate(...)`:  
useful vectorizing functions  
`summary(x1)`; `IQR(x1)`; `cumsum(x1)`; `diff(x1)`;  
`fivenum(x1)`; `mad(x1)`; `max(x1)`; `mean(x1)`;  
`median(x1)`; `min(x1)`; `prod(x1)`; `sd(x1)`; `sum(x1)`;  
`var(x1)`: statistics  
`table(x1, x2)`: cross-tabulation  
`tapply(x1, list1, f1)`: apply function `f1` to `x1` grouping by  
`list1`

## Inferential statistics

`chisq.test(tab1)`:  $\chi^2$ -test for table `tab1`  
`cor(df1)`: correlations between all columns of the data frame  
`cor.test(x1, x2)`: Pearson correlation test  
`ks.test(...)`; `prop.test(...)`; `t.test(...)`,  
`wilcox.test(...)`: other tests  
`lm(...)`; `glm(...)`; `aov(...)`; `anova(...)`: linear and non-  
linear models, analyses of variation (ANOVA)  
`predict(model1)`: predict from model  
`lm(y ~ x + z, data=...)`: formula interface to the additive lin-  
ear model, `y` responses on two variables, `x` and `z`

## Multivariate statistics

`cmdscale(...)`: metric multidimensional scaling (MDA)  
`hclust(...)`: hierarchical cluster analysis  
`princomp(...)`; `prcomp(...)`: principal component analyses  
(PCA)

## Plots

`boxplot(...)`, `dotchart(...)`, `hist(...)`: useful plots  
`identify(...)`: reveal information from points using mouse  
`legend("topleft", legend="...")`: add legend to the top left  
corner  
`lines(...)`; `points(...)`; `text(...)`: add lines, then points,  
then text  
`pdf("file1.pdf")`: write subsequent plot into `file1.pdf` until  
`dev.off()`  
`oldpar <- par(mfrow=c(2,1))`: plots will be stacked until  
`par(oldpar)`  
`oldpar <- par(mar=c(0,0,0,0))`: all plot margins set to zero  
until `par(oldpar)`  
`plot(..., cex=1|2)`: normal dot size, double dot size  
`plot(..., col=0|1|2|3)`: white, black, red, green color  
`plot(..., lty=0|1|2)`: no lines, straight line, dashed line  
`plot(..., type="p|l|s|n")`: points, lines, stairs and no plot  
`qqnorm(vec1)`; `qqline(vec1)`: check normality