

# One Page R Reference Card

Alexey Shipunov, January 7, 2020

`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 `3e3`:  $3 \times 10^3 = 3000$  (e-notation)  
`options(scipen=100)`: get rid of e-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 as `package::command()`  
Tab; Up; Ctrl+U: complete; repeat command; delete command  
(not on macOS GUI)

## 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)`; `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 as  
separator; do not convert variables into factors  
`scan("file1", what="char")`: read one series of character codes  
from disk into variable  
`sink("file1", split=TRUE)`: output to `file1` and to the termi-  
nal 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, 2, labels=c("small", "big"))`: split vector `v1` into  
two intervals  
`data.frame(v1, v2)`: list from same-length vectors `v1` and `v2`  
`df1$a1`: variable (column) named `a1` from data frame `df1`  
`dimnames(mat1)`, or `names(df1)` and `row.names(df1)`: names of  
rows and columns of `mat1` or `df1`  
`droplevels(factor1)`: drop unused factor 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)`, `nrow(mat1)`, `ncol(df1)`: sizes  
`list1[[-5]]`: all list elements except 5th  
`ls()`: list all active objects  
`mat1[, 2:5]` or `mat1[, c(2, 3, 4, 5)]`: columns from 2nd to  
5th  
`matrix(vector1, r1, c1)`: transform `vector1` into matrix with  
`r1` rows and `c1` columns, columnwise  
`merge(df1, df2)`: merge two data frames  
`paste("cow", "boy", sep="")`: outputs "cowboy"  
`rep(x1, n1)`: repeat vector `x1` `n1` times  
`sample(x1, n1)`: sample `n1` elements from `x1` without replace-  
ment  
`seq(n1, n2, n3)`: sequence from `n1` to `n2` by `n3` steps  
`stack()` and `unstack()`: convert from short to long form and  
back again  
`str(obj1)`: structure of object `obj1`  
`t(mat1)`: rotate  $90^\circ$  matrix or data frame  
`with(x1, ...)`: do something within `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) dosomething : cycle`  
`fun1 <- function(args1) dosomething : define function`  
`if(condition1) ...else ... : single condition`  
`ifelse(condition1, yes, no)`: vectorized condition

## Logic and math

`is.factor(obj1)`, `is.atomic(obj1)`, `is.data.frame(obj1)`:  
check the type of object `obj1`  
`mat1[mat1 > 0]`: elements of `mat1` which are positive  
`!<`, `&`, `|`, `==`: "not less", "and", "or", "equal"  
`cumsum(x1)`; `diff(x1)`; `prod(x1)`; `sum(x1)`: vector math  
`round(x1)`: round  
`unique(x1)`: list unique elements of `x1` (could be sparse)  
`*`, `^`, `sqrt(pi)`, `abs(-3)`, `log(1)`: multiplication, degree,  $\sqrt{\pi}$ ,  
`3`, natural logarithm  
`x1 %in% x2`: which elements of `x1` are in `x2`  
`which(logic1)`: indexes of all TRUE's

## Descriptive statistics

`aggregate(...)`: pivot table  
`apply(x1, n1, function)`: apply function to all rows (if `n1`  
`= 1`) or columns (`n1 = 2`), output matrix  
`colSums(mat1)`: calculate sums of every column  
`rev(x1)`, `order(x1)`, `scale(x1)`, `sort(x1)`: reverse, sorting  
indexes, scale and center, (ascending) sort  
`sapply()`; `lapply()`; `do.call()`; `replicate()`: vectorize  
`summary(x1)`; `IQR(x1)`; `fivenum(x1)`; `mad(x1)`; `max(x1)`;  
`mean(x1)`; `median(x1)`; `min(x1)`; `sd(x1)`; `var(x1)`: de-  
scriptive 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)`: (Pearson) correlations between all columns of the data  
frame  
`cor.test(x1, x2)`: (Pearson) correlation test  
`ks.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

`dist(...)`: distance calculation  
`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")`: draw 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