Basic R commands for data analysis

Version 1.4, January 2017

David Lorenz
English Department, Albert-Ludwigs-Universität Freiburg

PART I – Objects, work space, working directory

Basic maths

```
> 1+1             # addition
[1] 2
> 6/2             # division
[1] 3
> 2*10+5          # multiplication and division before addition and subtraction
[1] 25
> 2*(10+5)        # bracketing
[1] 30
> 3^2             # exponentiation
[1] 9
> sqrt(9)         # square root
[1] 3
> round(10/3, digits=2)      # rounding
[1] 3.333
> abs(-5)         # absolute value
[1] 5
```

Creating 'objects'

```
> x <- 43                    # assigning a value; x is now an object (also
                               works the other way around: 43 -> x).
> venus <- "evening star"
> x
[1] 43
> venus
[1] "evening star"
> y <- c(1:5)
> y
[1] 1 2 3 4 5                 # 'c' is for "concatenate" and is used to create
                                sets (here a sequence from 1 to 5).
> elements <- c("fire", "earth", "air", "water")
> elements
[1] "fire"  "earth" "air"   "water"
```

Work space and working directory

The work space is the set of objects currently defined.

```
> ls()      # shows the work space
> rm(x)     # deletes the object x
```

The working directory is the directory on your computer that R currently 'works' in. (For how to save items there, see at the end of 'Data frames'.)

```
> getwd()   # shows the current working directory
> setwd(dir="~/Desktop")     # change the working directory (here to Desktop;
                               the path given has to be an existing folder).
```

PART II – Data frames, data types

The following examples assume a data set with the variables: variant ('short', 'long'), duration (numeric), gender, age, sentence_type – *replace these with the variables of your own data set.*

Data frames

Reading files works best with .csv format ('comma-separated values'). Excel tables can be saved as .csv (choose UTF-8 encoding; separator can also be ";" or tab stop). Alternatively, read.table() reads tables in .txt format.
=> RStudio lets you read in data with "Import Dataset" in the "Environment" panel – always note the filename in your R script!

```
> read.csv(file.choose())    # reads a .csv file (which you have to select);

> read.csv(file="~/Desktop/mydata.csv")
      # reads the file in the given URL (this must be an existing file on your
        computer);

> read.csv(file="~/Desktop/mydata.csv", sep="\t", header=F)
      # reads the .csv file as separated by tab stop, and with no header row
        (i.e. column headings); for data separated by ";" use sep=";" or the
        function read.csv2();

> mydata <- read.csv(file="~/Desktop/mydata.csv")
           # reads the .csv file as above, and assigns a variable to it. Now
> mydata    # shows the data set. This is a 'data frame':

> is.data.frame(mydata)
[1] TRUE

> nrow(mydata)    # shows number of rows;
> ncol(mydata)    # shows number of columns;
> colnames(mydata)# shows the names of the columns;
> colnames(mydata)[3] # shows the name of the third column;
> colnames(mydata)[3] <- "variant" # changes the name of the third column to
                                   "variant";
> head(mydata)    # shows the first 6 rows of the data frame;
> str(mydata)     # shows the structure of the data frame;
> mydata[23,]     # shows the 23rd row; the format for selection is:
                    dataframe[row(s) , columns]
> mydata[,3]      # shows the 3rd column;
> mydata[23,3]    # shows the value in the 23rd row of the 3rd column;
> mydata[c(1:10),3]     # rows 1–10, 3rd column;
> mydata$variant  # shows the column with the heading "variant";
> mydata[mydata$variant == "short" ,]    # all rows in which the value for
                                           'variant' is "short".
> mydata[mydata$duration >= 10 ,]        # all rows in which the value for
                                           'duration' is greater or equal 10.
> mydata[mydata$duration <= 10 ,]        # all rows in which the value for
                                           'duration' is smaller or equal 10.
> mydata$newcolumn <- "x"     # adds a column 'newcolumn' to the data frame,
                                with the value "x" in every row;
> mydata[mydata$variant=="short",]$newcolumn <- "y"
                        # sets the value for 'newcolumn' to "y" in rows in which
                          the value for 'variant' is "short".
```

```
> write.csv(file="~/Desktop/mydata.csv")
                        # writes the data frame to a .csv file in the specified
                          file path. (!Note: if this is an existing file, it
                          will overwrite it!)
> save(mydata, file="~/Desktop/myworkspace.RData")
                        # saves the data to a workspace file (.RData) (!Note:
                          if this is an existing file, it will overwrite it!)
> save("mydata", "elements", file="~/Desktop/myworkspace.RData")
                        # saves the listed items to the specified file;
> save.image(file="~/Desktop/myworkspace.RData")
                        # saves the entire work space to the specified file.
> load("~/Desktop/myworkspace.RData")
                        # loads the data in the specified file. (You can also
                          just double-click the file.)
```

Data types

Columns in a data frame are called 'vectors'; a vector must contain data of only one type. The main types are 'factor', 'numeric' and 'character'; factors may be ordered.

```
> is.numeric(57)              # numbers are numeric
[1] TRUE
> is.character("blablabla")   # 'words' are character strings
[1] TRUE
> class(mydata$duration)      # gives out the data type of this vector

> mydata$variant <- as.factor(mydata$variant)
                              # turns the vector (column) into a 'factor';
                                likewise: as.numeric(), as.character().
> levels(mydata$variant)      # only factors have levels.
[1] "long" "medium" "short"
> mean(mydata$duration)
> sd(mydata$duration)
> sum(mydata$duration)        # only numeric vectors have a mean, standard
                                deviation, sum, etc.
> summary(mydata$duration)    # shows the central tendency measures all at once.
                                (Or token counts if it is a factor.)
```

PART III – Data inspection and analysis

See the textbooks at 'Further reading' (PART IV) for detail on how to apply and validate statistical tests and models!

Categorical data:

```
> xtabs(~ variant + gender, mydata)
                  # shows a contingency table of 'variant' by the variable
                    'gender' in the data frame 'mydata' (the variables must be
                    columns in the data frame).
> prop.table(xtabs(~ variant + gender, mydata), 1)
                  # shows a table of proportions (rather than raw frequencies),
                    such that the rows add up to 1;
```

```
> prop.table(xtabs(~ variant + gender, mydata), 2)
                # a proportions table – columns add up to 1.

> chisq.test(mydata$variant, mydata$gender)
                # runs a Chi-Squared test for the specified variables (which
                  must be categorical). 'chisq.test()' can also be run
                  directly on a table of numbers, e.g. chisq.test(xtabs(~
                  variant + gender, mydata)).

> chisq.test(mydata$variant, mydata$gender)$observed   # observed frequencies;
> chisq.test(mydata$variant, mydata$gender)$expected   # expected frequencies;
> chisq.test(mydata$variant, mydata$gender)$residuals
                # residuals (chi value for each cell in the table).


> fisher.test(mydata$variant, mydata$gender)
                # runs a Fisher's exact test for the specified variables
                  (which must be categorical).
```

A numeric and a categorical variable:

```
> tapply(mydata$duration, mydata$gender, mean)
                # with a numeric and a categorical variable: shows the mean of
                  the numeric variable 'duration' for each level of the
                  categorical variable 'gender' – "mean" can be replaced by
                  "sum", "sd", etc. (If the data contains NAs, add na.rm=T )


> boxplot(mydata$duration)         # creates a boxplot of 'duration' (numeric);
> boxplot(duration ~ gender, mydata)
                # creates boxplots of 'duration' (numeric) for each level of
                  'gender' (categorical). Many additional parameters can be
                  set, see the help function:
> ?boxplot


> shapiro.test(mydata$duration)    # Shapiro-Wilk test of normal distribution
                                     (if p<0.05, the data is NOT normally
                                     distributed).
> plot(density(mydata$duration))
                # A density plot of the distribution of the 'duration' values;
                  normal distribution yields a symmetric bell curve;

> abline(v=mean(mydata$duration), lty="dashed")
                # adds a (dashed) line at the mean of the 'duration' values.
> t.test(duration ~ gender, mydata, paired=F)
                # An unpaired t-test of 'duration' by 'gender' (set paired=T
                  for paired data).

> wilcox.test(duration ~ gender, mydata, paired=F)
                # A Wilcoxon rank-sum (Mann-Whitney U) test of 'duration' by
                  'gender'; set paired=T for the Wilcoxon signed-rank test.

> lm(duration ~ variant, mydata)
                # A linear regression model (for more than two sets:
                  mydata$variant has three levels)

> summary(lm(duration ~ variant, mydata))
                # Coefficients of the linear regression.
```

```
> aov(duration ~ variant, mydata))
> summary(aov(duration ~ variant, mydata))
                # Analysis of Variance of 'duration' by 'variant'.
```

Correlation (numeric data):

```
> plot(mydata$age, mydata$duration)# creates a graph of the numeric variables
                                    'duration' and 'age';
> plot(mydata$duration ~ mydata$age)     # the same plot.
> abline(lm (mydata$duration ~ mydata$age))
                # adds the regression line to the plot.
> lm(mydata$duration ~ mydata$age) # a linear model: the intercept and slope of
                                    the regression line.


> cor(mydata$age, mydata$duration) # The (Pearson's product-moment) correlation
                                    coefficient;
> cor.test(mydata$age, mydata$duration)
                # A Pearson correlation test with coefficient, p-value and
                  confidence interval for the coefficient;
> cor.test(mydata$age, mydata$duration, method="spearman")
                # A Spearman's rho rank correlation test (set method="kendall"
                  for Kendall's tau test).
```

Multifactorial:

```
# Logistic regression (binary dependent variable):
> ftable(mydata$gender, mydata$sentence_type, mydata$variant)
                # a 'flat' table of all combinations of several factors.
> mydata$variant <- relevel(mydata$variant, "short")
                # defines the reference level of a factor.
> mymodel <- glm(variant ~ gender + age + sentence_type, data=mydata,
family="binomial")
                # a logistic regression model; here 'variant' is the dependent
                  variable, 'gender', 'age' and 'sentence_type' are
                  independent variables. Use...
> summary(mymodel)      # ...to see the coefficients and p-values.
> anova(mymodel)        # Analysis of Variance of the model.


# Linear regression (numeric dependent variable):
> mylm <- lm(duration ~ sentence_type + variant, mydata)
                # a linear regression model: 'duration' is the dependent
                  variable, 'sentence_type' and 'variant' are independent
                  variables. Use...
> summary(mymodel)      # ...to see the coefficients and p-values.
```

PART IV – Packages, online help, further readings

Packages

Packages are specialized 'add-ons' that contain useful functions for data analysis, plotting, etc.

```
> install.packages("effects")      # installs the package 'effects';
> library(effects)           # loads the package into the library. To see all
                               the functions in a package, use help():
> help(package=effects)
```

Some useful packages:

effects

```
## contains the allEffects() function that calculates estimated probabilities
from the coefficients of a logistic/linear regression model:
> allEffects(mymodel)
> plot(allEffects(mymodel)   # a graph showing the effects.
```

ggplot2

```
## contains the functions qplot() and ggplot() for creating sophisticated
graphs. See http://ggplot2.org
```

plyr

```
## Tools for splitting, applying and combining data, e.g. ddply()
> ddply(mydata, .(variant, gender), summarize, mean=mean(duration))
                  # a data frame showing the mean 'duration' for each
                    combination of 'variant' and 'gender' in 'mydata' (other
                    measures such as sd() can be added).
```

rms

```
## "regression modeling strategies" – contains the function lrm() for logistic
regression with a few more features than glm()
```

swirl

```
## A tutorial for working with R, for beginners and advanced learners – see
http://swirlstats.com
```

Further readings

R and statistics textbooks for linguists:

Baayen, Harald. 2008. *Analyzing Linguistics Data*. Cambridge: Cambridge University Press.

Gries, Stefan Th. 2013. *Statistics for Linguistics with R.* 2nd edition. Berlin, New York: De Gruyter.

Johnson, Keith. 2008. *Quantitative Methods in Linguistics.* Malden: Blackwell.

Levshina, Natalia. 2015. *How to do Linguistics with R: Data exploration and Statistical Analysis.* Amsterdam: John Benjamins.

General introductions to R available online:

Burns, Patrick. *Impatient R*. http://www.burns-stat.com/documents/tutorials/impatient-r/

Chang, Winston. *Cookbook for R.* http://www.cookbook-r.com

Kohl, Matthias. 2015. *Introduction to statistical data analysis with R.* http://bookboon.com/en/introduction-to-statistical-data-analysis-with-r-ebook [also available in German.]

Navarro, Daniel. 2015. *Learning Statistics with R.* http://health.adelaide.edu.au/psychology/ccs/teaching/lsr/

Owen, W. J. 2010. *The R Guide*. Version 2.5. http://www.mathcs.richmond.edu/~wowen/TheRGuide.pdf

Introductory R books:

Adler, Joseph. 2010. R in a Nutshell: A Desktop Quick Reference. Sebastopol, CA: O'Reilly Media.

Crawley, Michael J. 2012. The R Book, 2nd Edition. Hoboken, NJ: Wiley.

Kabacoff, Robert I. 2011. R in Action: Data Analysis and Graphics with R. Greenwich, CT: Manning Publications.

Teetor, Paul. 2011. R Cookbook. Sebastopol, CA: O'Reilly Media.

Verzani, John. 2014. *Using R for Introductory Statistics*. 2nd edition. Boca Raton: Chapman & Hall / CRC Press.

Check the official R bibliography for more...

Online help

```
http://www.statmethods.net   # a very good overview of the possibilities for
                               statistics and graphics in R

http://www.ats.ucla.edu/stat/r/      # lots of useful R and statistics stuff, with
                                       helpful examples!

https://groups.google.com/forum/#!forum/corpling-with-r
                                     # Stefan Gries' Corpus linguistics with R forum

http://www.inside-r.org      # "A community site for R" – with overviews of functions
                               and packages

http://stackoverflow.com     # a programming/statistics forum (the site itself may
                               not look very helpful, but online searches for stats
                               questions often lead here...)

http://www.r-bloggers.com    # an R forum where you will often find answers to your
                               "How do I do this in R" questions
```

Citing R and R packages

R Core Team. 2016. *R: A Language and Environment for Statistical Computing*. Vienna: The R Foundation for Statistical Computing. URL: http://www.r-project.org

```
> citation()      # shows the reference for R;
> citation("effects")  # shows the reference for a package (here: "effects").
```