A brief introduction to R

Pre-tutorial handout for "Beyond example extraction: Quantitative analysis of the JANES corpus"

Maja Miličević
University of Belgrade
m.milicevic@fil.bg.ac.rs

1 What is needed for the tutorial

The focus of the tutorial is on the basics of statistical analysis of corpus data (using data samples from JANES). To be able to follow the tutorial, you will need **R** (`https://www.r-project.org`), which is a powerful environment for statistical computation and data visualisation. The main advantages of R compared to other statistical software are its flexibility and availability free of charge. It has a reputation of being (initially) difficult to learn, but it is well worth the effort to persist.

To save time during the actual tutorial, you are kindly asked to **install R on your computer beforehand** (see section 2 for instructions). In addition, as R relies heavily on add-ons for some advanced options and functionalities not provided in the main suite, please familiarise yourself with how to install and load **packages** (see section 3).

We will use the graphical interface that comes with the default R installation, so there is no need to install any additional programs. The datasets we will work on will be made available during the tutorial itself; two simple files you can use to test some basic operations (see section 4) are included with this handout (`testdata1.txt` and `testdata2.csv`).

2 Setting up R

To download R, go to `https://cran.r-project.org`[1] and select the appropriate link for your operating system, as shown in Figure 1.[2] Assuming that you use either Windows or Mac OS X:

- For Windows, select the "base" distribution and download the installation file linked under "Download R 3.2.2 for Windows".
- For Mac OS X, you will most likely want the "R.3.2.2.pkg" file (or "R-3.2.1-snowleopard.pkg", if you have an older version of OS X)

Install proceeds in the usual double-click-and-follow-the-instructions way; default settings will be fine for most users, so you can go ahead with those (see Figure 2).[3] On Windows, an R shortcut will be created on your desktop and you will see the program folder in your Start Menu; on Mac, a program icon will be placed in your Applications folder.

[1] R code and documentation are stored in a server network called CRAN – Comprehensive R Archive Network. This link points to a mirror site in Austria; other mirror sites are listed at `https://cran.r-project.org/mirrors.html`.

[2] R is regularly updated; the current release is R 3.2.2 Fire Safety. If you already have an older version installed, there is no pressing need to upgrade – new versions fix old bugs, but commands are mostly kept stable across releases.

[3] See `https://cran.r-project.org/doc/manuals/r-release/R-admin.html#Installing-R-under-OS-X` and `https://cran.r-project.org/doc/manuals/r-release/R-admin.html#Installing-R-under-Windows` for specifics.

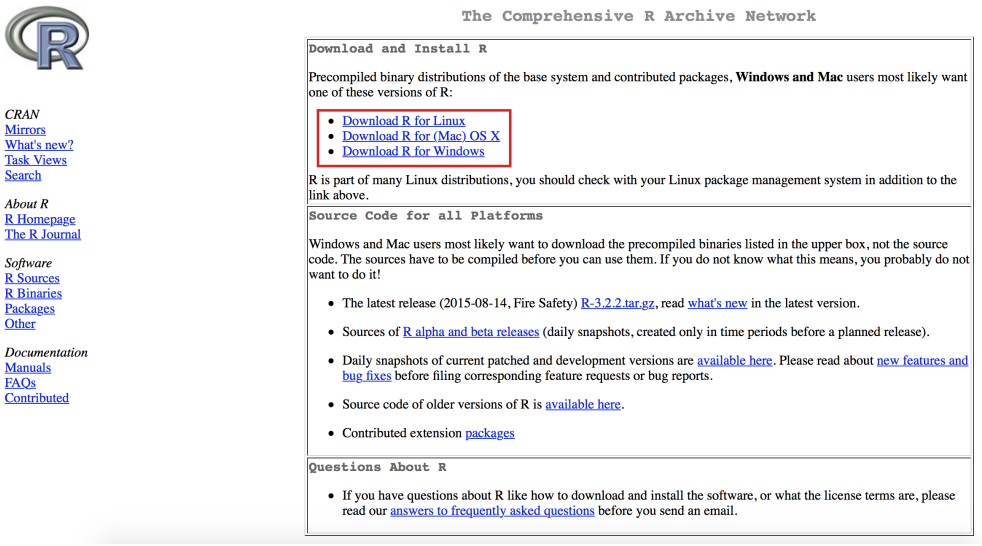

Figure 1: Selecting the operating system

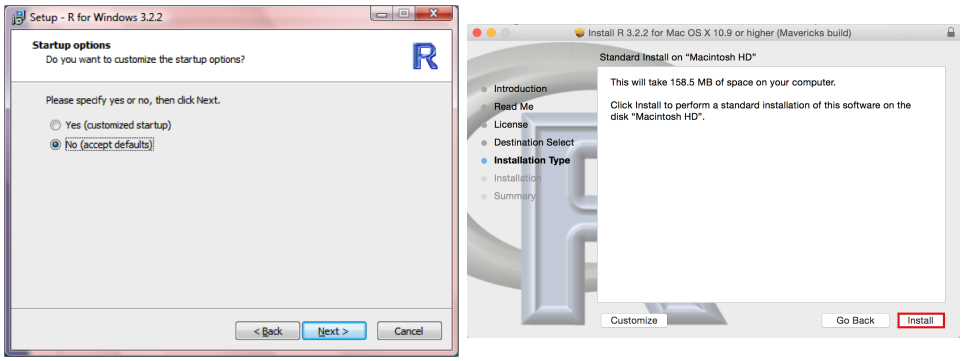

Figure 2: Accepting the default options on Windows and Mac OS X

When you open R, you will see a graphical user interface screen similar to the one shown in Figure 3 (Windows), or Figure 4 (Mac). The main part of the screen is the **R Console**, a command line window through which most of your interaction with R will happen – you can use the menus for various kinds of editing and for setting preferences, but statistical functions are accessed via the command line.

The greater than symbol (>) is a command prompt; if you want R to do something, you need to type a command you want it to execute, and press Enter. Of course, commands can also be copied from elsewhere and pasted into R. It is also possible to add notes or comments, by typing a hash symbol (#); anything following this symbol on the same line will be ignored by R.

If you wish to customise the appearance of the console, use Edit > GUI preferences (Windows), and R > Preferences or Format (Mac) .

3 Installing R packages

Packages – collections of functions and data – are central to R's functioning; R comes with a standard set of preinstalled packages, and many others are available for download and install. Information about the current state of packages present on the system can be obtained via the Package manager (Packages > Package manager on both Windows and Mac); the manager can, for instance, be used to load packages that are installed but are not loaded. You can also see the list of installed packages via the `library()` command; all currently loaded packages can be displayed with `search()`.

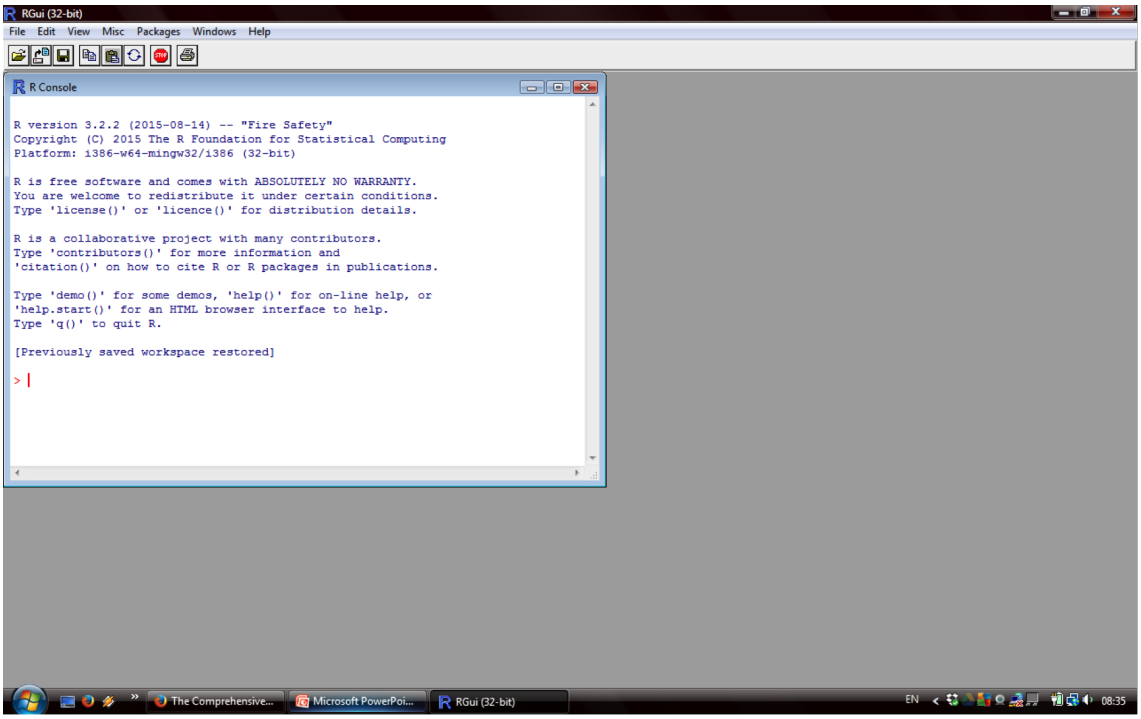

Figure 3: RGui interface on Windows

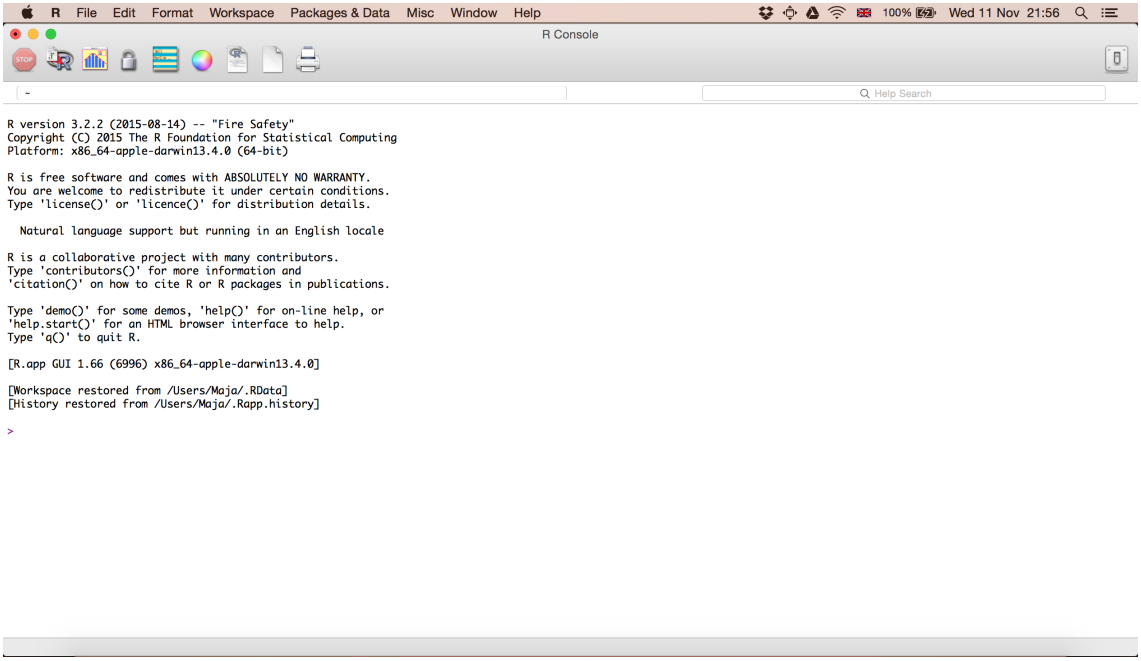

Figure 4: R.app interface on Mac OS X

To install new packages, you can either type `install.packages("package.name")` in the console, or you can use the GUI menus. On Windows, if you go to Packages > Install package(s) you will see a pop-up list of available packages (see Figure 5), from which you can select the package you need and simply click "OK". You might be asked if you want to create a personal library to install packages into; if this happens, select "Yes". On Mac, you need Packages > Package installer (Figure 6); leave "CRAN (binaries)" selected, search for a specific package or obtain a list of available packages ("Get list"), then select the package you want and click "Install Selected".

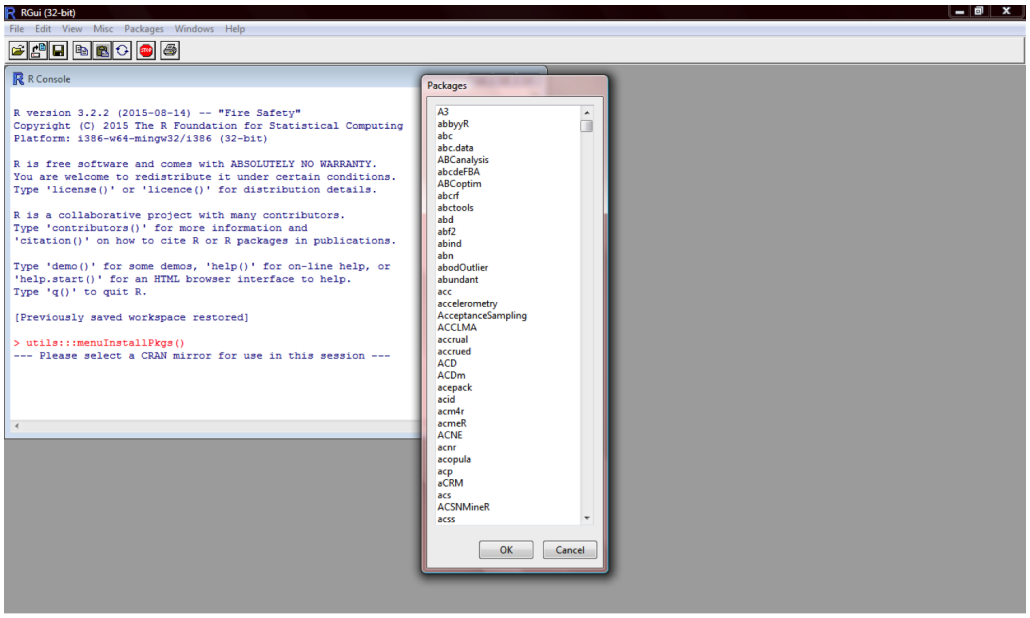

Figure 5: Package installer on Windows

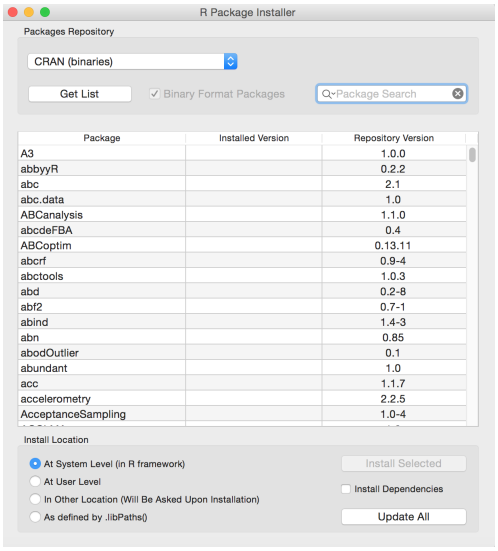

Figure 6: Package installer on Mac OS X

Regardless of the installation method, you will also need to select a CRAN mirror site. You can do that on your own via the `chooseCRANmirror()` command, which opens a pop-up window that displays a list of mirror sites, or when prompted by the program. On Windows you can also go to

Packages > Set CRAN mirror, while on Mac you can select the default mirror under R > Preferences > Startup. You are advised to choose a mirror in Austria.

You will most likely decide which packages you need based on the descriptions of what they do, or based on having heard/read that someone else used a specific package to do the same thing you want to do. A list of available packages distributed through CRAN is available at `https://cran.r-project.org/web/packages/available_packages_by_name.html`. CRAN has very strict rules about package creation, so you can be sure to find detailed documentation on each package. The documentation can also be accessed from within R by typing `help(package="package.name")`, e.g. `help(package="datasets")`.

Once installed, packages have to be loaded in order to be used – to load a package you need to type `library(package.name)`. Note that when you quit and reopen R you will need to reload any packages you need (except for the standard preinstalled packages).

To test package installation and use, go ahead and install "corpora", a package by Stefan Evert that provides data and functions for statistical analyses of corpus data. After you install the package (you should get a confirmation message about that in the console), type the following commands to load it and see a bit of data:

```
> library(corpora)
> data(BNCcomparison)
> BNCcomparison
```

You should now see a table showing 60 English nouns and their frequencies in written and spoken portions of the British National Corpus (the top part of the table is shown in Figure 7). If that is so, congratulations, you have installed your first R package! In case you are keen to try other things, have a look at the package reference manual, available at `https://cran.r-project.org/web/packages/corpora/corpora.pdf`.

	noun	written	spoken
1	time	156300	21720
2	year	146308	15593
3	people	94568	20939
4	way	93325	14037
5	man	88370	6460
6	day	80062	11011
7	thing	51670	23171
8	child	64256	5146
9	government	63306	3221
10	system	59022	2327
11	woman	56417	3824
12	part	55809	4288
13	case	56764	3237
14	number	53194	6551
15	group	55914	3422
16	work	53636	5077
17	world	55900	2566
18	area	52957	5156
19	company	55148	2398
20	house	50634	5587

Figure 7: Top 20 nouns in BNC

4 Getting to know R

4.1 Functions and arguments

A command passed to R typically consists of a **function** and its arguments, provided in parentheses. A function instructs R to do something, and the arguments indicate what that instruction is to be applied to, and how. In a very simple example, `sum(1,2)` is an addition function that is applied to arguments 1 and 2.

Many functions have default settings, which are used if an argument is not explicitly defined by the user; this also applies to cases where no arguments or options are provided and a function is followed by `()`; note that parntheses must always be provided, whether filled or empty.

A useful list of frequently used functions can be found at `http://ww2.coastal.edu/kingw/statistics/R-tutorials/text/function_ref.txt`. A reference manual page for a function can be displayed by typing `help(function.name)` or ?function.name, e.g. `help(sum)` or `?sum`. Note, however, that manual pages tend to be quite technical and can be difficult to understand for a beginner; an alternative is to simple search the Internet – R has a very lively comunity of users and developers and it is quite unlikely that you will encounter an issue that is not already explained somewhere.

4.2 Objects

In R you will work with different types of data **objects**; arguments of functions, for example, typically include objects to which functions are applied. Objects are created by **assignment**, using =, <- or -> (the choice is entirely up to you). You will see no printed output when you create an object, but if you type the object name you will get its value. After creating an object, you can include it in further operations by referring to its name. Some simple examples are:

```
> x = 1
> x
[1] 1

> y <- 1+2
> y
[1] 3

> y*4+x -> z
> z
[1] 13
```

Several things you need to be careful about:

- R will let you overwrite things you previously created without warning; if you assign something new to an object name that already exists, the old object content will be gone
- Object names (or any other names you will use in R) cannot contain spaces or dashes – use underscores and dots instead; you can freely use alphanumeric characters, just keep in mind that R is case sensitive
- R is flexible about spaces, you can include them or not (`x=1` and `x = 1` are the same thing, as is `x  =  1`), but you cannot have spaces between components of a single operator (`- >` is bad)

4.2.1 Vectors

The basic data object type in R is a **vector**. Vectors are sequences of numbers or character strings, typically created using the `c()` (concatenate) function:

```
> vector1 = c(2, 4, 6, 8, 10)
> vector1
[1] 2 4 6 8 10

> vector2 = c("Ljubljana","Maribor","Celje","Kranj")
> vector2
[1] "Ljubljana" "Maribor" "Celje" "Kranj"
```

Note that character strings have to be enclosed in quotation marks. These can be either double or single quotes, but you need to be consistent.

4.2.2 Data frames

Another central object type are **data frames**; you are most likely to work with them, in the tutorial and in the future, as they are the kinds of tables typically used by researchers to organise data. Data frames can be quite complex and while they can be created in R, it will often make more sense to load them from external files.

Data frames are tables similar to those typically created in Excel or other spreadsheet software; in R they are composed of a list of column vectors. The structure of data frames in R must be such that each variable is represented in its own column, while each case has its own row. An example is the above BNC table, in which each noun represents a case that is assigned a value for each of two variables (frequency in the written and frequency in the spoken part of BNC). We can recreate this table from scratch in the following way (the first five rows will suffice to illustrate the principle):

```
> BNC5frame = data.frame(
+ noun=c("time","year","people","way","man"),
+ written=c(156300,146308,94568,93325,88370),
+ spoken=c(21720,15593,20939,14037,6460))
> BNC5frame
      noun  written  spoken
 1    time   156300   21720
 2    year   146308   15593
 3  people    94568   20939
 4     way    93325   14037
 5     man    88370    6460
```

On a side note, this example also shows that you can type R commands on several lines; if a command on a single row is not complete, R will show the plus sign (+) as a continuation prompt. In case you see an unexpected plus sign check your command and see what is missing; you are most likely to forget to close a bracket.

In order to refer to specific columns in a data frame, you use the dollar ($) symbol. For instance, if for some reason you would like to add up the frequencies of all 60 nouns in the BNCcomparison dataset, but only for the written part of the corpus, you can do:

```
> sum(BNCcomparison$written)
[1] 19277930
```

To create data frames by importing data from external files, it is best to save your data in tab-delimited or comma-separated format, in .txt or .csv files. If you are used to working with spreadsheet software such as Microsoft Excel or OpenOffice Calc, the easiest thing to do is to save your data as comma-separated values by selecting .csv under "Save As" options (see Figure 8). In Excel you can proceed in the same way when you want to save a file as tab-delimited; in Calc you again select the CSV option, but also tick "Edit filter settings", which in the next step will allow you to select the tab field delimiter (Figure 9).

Figure 8: Saving comma-delimited tables in Excel (left) and Calc (right)

Once your data file is ready, you can use one of the following functions to get it into R:

- `read.table`: a generic function that can be used for different table formats
- `read.csv`: a specialised function for tables with comma-separated fields
- `read.delim`: a specialised function for tables with tab-delimited fields

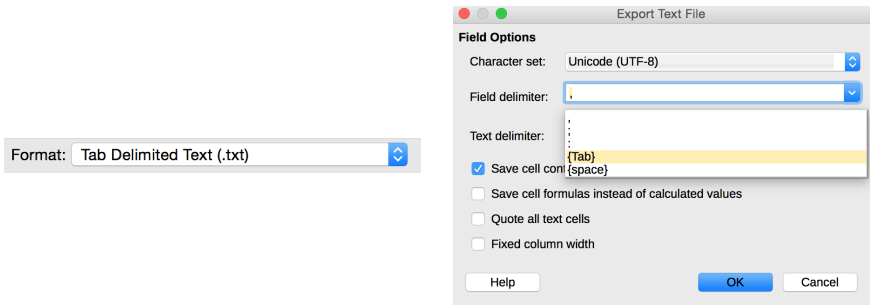

Figure 9: Saving tab-delimited tables in Excel (left) and Calc (right)

In other words, `read.csv` and `read.delim` can be thought of as shortcuts for the two most frequent table formats. For table import options, in particular for the `read.table` function, see `https://stat.ethz.ch/R-manual/R-devel/library/utils/html/read.table.html`. One thing that is important to mention about all three functions is that you need to pay attention to whether your tables have headers; use `header=TRUE` for tables with a header, and `header=FALSE` for tables without one.

You can now try loading the test data you received with this handout (or alternatively some files you create on your own). Of course, you will need to change the file location to where you saved the files on your computer. Don't forget the quotes!

```
> testdata1 = read.delim("/Users/Maja/Documents/testdata1.txt", header=TRUE)
> testdata2 = read.csv("/Users/Maja/Documents/testdata2.csv", header=TRUE)
```

The first file contains the 50 most frequent words from three JANES corpora, Tweet, Forum and Comment, jointly with their frequencies and frequency ranks. The second file contains the top 50 collocates of the adverb *lahko* in tweets published by corporate vs. private accounts; for each collocate there is information about its frequency of co-occurrence with *lahko*, and about two collocativity measures. The purpose of the test files is "technical" practice with R, so don't worry if you don't know anything about these measures, or can't see much sense in the choice of the data.

It is good practice to always check if your data has been loaded correctly. An easy way to do that is to use the `head()` command, which will show the first six rows of the data frame, or the `summary()` command, which lists some basic info about the data. The former command should give you the following results for the two test files:

```
> head(testdata1)
   Word  Corpus     Freq  FreqRank
1    je   Tweet  1456489         1
2     v   Tweet  1096518         2
3    in   Tweet   887432         3
4    na   Tweet   839581         4
5    pa   Tweet   786536         5
6    se   Tweet   784677         6

> head(testdata2)
   Word  Subcorpus  Freq  T.score     MI
1    se  corporate  2685   44.368  2.798
2    si  corporate  2342   46.350  4.565
3    bi  corporate  1591   35.606  3.220
4    je  corporate  1389   18.046  0.955
5    pa  corporate  1114   21.785  1.526
6  tudi  corporate  1007   28.550  3.318
```

If at some point you get strange-looking outputs, e.g. all data merged in a single column, try switching to a different import command. Note in particular that the file formats (.csv vs. .txt) do not strictly correspond to field separator formats; it is perfectly possible to have a tab-delimited .csv file and a comma-delimited .txt file, and R commands need to be selected based on the separator.

As above, you can refer to columns within data frames by using `$`. You can also use square brackets (`[]`) to single out subsets of data within the same column. For example, in the `testdata1` object it is possible to add up all available word frequencies for each corpus separately:

```
> sum(testdata1$Freq[testdata1$Corpus=="Tweet"])
[1] 14826068

> sum(testdata1$Freq[testdata1$Corpus=="Forum"])
[1] 12534849

> sum(testdata1$Freq[testdata1$Corpus=="Comment"])
[1] 4016299
```

Note also that the equal to symbol needs to be written as `==` rather than just `=`.

If you have time, you can play with adding up other values in the datasets you loaded into R, or you can try some other basic functions, such as `min()`, `max()`, `sqrt()` or `mean()`. Again, do not pay too much attention to the meaningfulness of applying these operations to the given data, it is only important that you are beginning to understand how R works.

4.3 Other fundamentals

Another key notion for R is that of **workspace**, the current working environment, which includes objects created or loaded by the user (functions, data objects, etc.). At the end of each session, you will be prompted to save the changes you made to the workspace during that session. If you do keep the changes, a workspace image will be saved in an .RData file. A related file is .Rhistory, which saves the commands used in a given session.

The .RData file will by default be saved in R's global default **working directory**; this file will automatically be loaded at the next startup. The default working directory is normally the user's home directory, or the Documents folder under it. You can find your working directory by typing the `getwd()` command in the console:

```
> getwd()
[1] "/Users/Maja"
```

It is a good idea to check where your default working directory is, because – following the usual command line principles – this is the place from which you can load data by file name alone, rather than by giving a full path (as was the case in the example in the previous section).[4] For instance, if you place the file `testdata2.csv` in your default working directory you can just type:

```
> testdata2 = read.csv("testdata2.csv", header=TRUE)
```

The defaults will suffice for the tutorial; you can look at the ways of changing the working directory and saving different R projects separately later on.

To check whether everything you need is loaded, you can list the current contents of the workspace using the `ls()` or `objects()` functions. This can be very useful when you reopen a workspace, as R does not automatically show a list of previously created objects.

And a final note on decimal separators: Slovenia, unlike English-speaking countries, uses comma as the decimal mark (e.g. 0,5 rather than 0.5). Given that in R commas are used to separate arguments of functions, even though it is possible to let the program know about number formats, the simplest approach is not to interfere with the overall number representation and to set only the output to use commas. This is easily done via `options(OutDec= ",")`; this command will ensure that the output you see in the console, as well as the graphs you create, will have commas as decimal marks.

To exit R, type `quit()` or `q()`.

[4] Note that on Mac OS X you will have to set hidden files to visible in order to see .RData and .Rhistory.

5 Some useful resources

R manuals from CRAN:

- https://cran.r-project.org/manuals.html
- https://cran.r-project.org/doc/manuals/r-release/R-admin.html
- https://cran.r-project.org/other-docs.html

Additional R user interfaces and web applications:

- R Commander (installed as an R package, Rcmdr)
- RStudio (https://www.rstudio.com)
- Tinn-R (http://nbcgib.uesc.br/lec/software/editores/tinn-r/en)
- iNZight (https://www.stat.auckland.ac.nz/~wild/iNZight/index.php)
- Langtest (http://langtest.jp)

Some R courses/tutorials:

- Statistical Inference - a Gentle Introduction for Linguists (http://www.stefan-evert.de/SIGIL/; contains material from several specialised courses for corpus linguists)
- A hands-on tutorial on using R for (mostly) linguistics research (http://coltekin.net/cagri/R/r-exercises.html)

Useful books:

- Baayen, R. H. (2008) *Analyzing Linguistic Data. A Practical Introduction to Statistics Using R*. Cambridge: Cambridge University Press.
- Field, A., J. Miles and Z. Field (2012) *Discovering Statistics Using R*. London: SAGE Publications. (psychology rather than linguistics, but very clear and accessible)
- Gries, S. (2013) *Statistics for Linguistics with R*. 2nd ed. Berlin and New York: De Gruyter.
- Gries, S. (2009) *Quantitative Corpus Linguistics with R: A Practical Introduction*. London and New York: Routledge.
- Larson-Hall, J. (2009). *A Guide to Doing statistics in Second Language Research using SPSS*. New York and London: Routledge. (comes with a supplement about R: http://cw.routledge.com/textbooks/9780805861853/R/full-version.pdf)

A much more comprehensive list of R-related links relevant for linguists is available at
https://experimentalfieldlinguistics.wordpress.com/statistics-and-r-blogs/

6 Acknowledgements

This tutorial is a result of collaboration between several projects, JANES (http://nl.ijs.si/janes/) - funded by the Slovenian Research Agency, ReLDI (http://reldi.spur.uzh.ch/) - funded by the Swiss National Science Foundation, and the bilateral Serbian-Slovenian project "The Construction of Corpora and Lexica of Nonstandard Serbian and Slovenian" - funded by the Slovenian Research Agency and the Serbian Ministry of Education, Science and Technological Development. For direct financial support, thanks are due to all organisers of the conference "Slovenščina na spletu in v novih medijih" – Faculty of Arts of the University of Ljubljana (http://www.ff.uni-lj.si), Slovenian Language Technologies Society (http://www.sdjt.si), and especially the CLARIN.SI consortium (http://www.clarin.si).