

Statistical Analysis of Corpus Data with R

A Gentle Introduction

for Computational Linguists and Similar Creatures

Designed by Marco Baroni¹ and Stefan Evert²

¹Center for Mind/Brain Sciences (CIMEC)
University of Trento

²Institute of Cognitive Science (IKW)
University of Osnabrück

Outline

General Information

- What is R?

- About this course

R Basics

- Basic functionalities

- External files and data-frames

- A simple case study: comparing Brown and LOB documents

Why do we need statistics?

Why do we need statistics?

- ▶ **Significance** (control for sampling variation)
 - ▶ all linguistic data are samples (of language, speakers, ...)
 - ▶ observed effects may be coincidence of particular sample
 - ➔ **inferential statistics**

Why do we need statistics?

- ▶ **Significance** (control for sampling variation)
 - ▶ all linguistic data are samples (of language, speakers, ...)
 - ▶ observed effects may be coincidence of particular sample
 - ➔ **inferential statistics**

- ▶ Managing **large data sets**
 - ▶ statistical summaries, data analysis, visualisation
 - ▶ e.g. collocations as compact summary of word usage
 - ➔ **descriptive statistics**

Why do we need statistics?

- ▶ **Significance** (control for sampling variation)
 - ▶ all linguistic data are samples (of language, speakers, ...)
 - ▶ observed effects may be coincidence of particular sample
 - ➔ **inferential statistics**

- ▶ Managing **large data sets**
 - ▶ statistical summaries, data analysis, visualisation
 - ▶ e.g. collocations as compact summary of word usage
 - ➔ **descriptive statistics**

- ▶ Discovering **latent** (hidden) **properties**
 - ▶ clustering, multivariate analysis, distributional semantics
 - ▶ advanced statistical modelling (e.g. mixed-effects models)
 - ➔ **exploratory data analysis**

R – An environment for statistical programming

- ▶ “Traditional” statistical software packages offer specialised procedures (e.g. SAS) or interactive GUI (e.g. SPSS)

R – An environment for statistical programming

- ▶ “Traditional” statistical software packages offer specialised procedures (e.g. SAS) or interactive GUI (e.g. SPSS)
- ▶ New approach: statistical programming language **S** with interactive environment (Bell Labs, since 1976)
 - ▶ *White Book* (version 3, 1992); *Green Book* (version 4, 1998)
 - ▶ commercial: S-Plus (Insightful Corporation, since 1987)

R – An environment for statistical programming

- ▶ “Traditional” statistical software packages offer specialised procedures (e.g. SAS) or interactive GUI (e.g. SPSS)
- ▶ New approach: statistical programming language **S** with interactive environment (Bell Labs, since 1976)
 - ▶ *White Book* (version 3, 1992); *Green Book* (version 4, 1998)
 - ▶ commercial: S-Plus (Insightful Corporation, since 1987)
- ▶ **R** is an open-source implementation of the S language
 - ▶ originally by Ross Ihaka and Robert Gentleman (Auckland)
 - ▶ open-source development since mid-1997

R – An environment for statistical programming



- ▶ binary packages available for Linux, Mac OS X and Windows
- ▶ 64-bit versions on Linux and OS X
- ▶ extensive documentation & tutorials
- ▶ hundreds of add-on packages ready to install from CRAN

<http://www.R-project.org/>

Recommended Windows GUI:

Tinn-R from <http://www.sciviews.org/>

More about R

- ▶ Advantages of R
 - ▶ free & open source
 - ▶ many add-on packages with state-of-the-art algorithms
 - ▶ large, enthusiastic and helpful user community
 - ▶ easy to automate and extend (every analysis is a program)
 - ▶ no point & click interface

More about R

▶ Advantages of R

- ▶ free & open source
- ▶ many add-on packages with state-of-the-art algorithms
- ▶ large, enthusiastic and helpful user community
- ▶ easy to automate and extend (every analysis is a program)
- ▶ no point & click interface

▶ Disadvantages

- ▶ learning curve sometimes rather steep
- ▶ not good at manipulating non-English text (yet)
- ▶ no built-in data editor (spreadsheet)
- ▶ no point & click interface

Goals of the course

- ▶ Learn R basics and elementary R programming
- ▶ Get to know R implementations of statistical techniques, data analysis and visualisation that are useful in various areas of (computational) linguistics
- ▶ A little bit of background in the statistical analysis of corpus frequency data along the way
- ▶ Practice your R skills on real-life data-sets

What this course is *not* about

- ▶ Theoretical foundations of statistics
- ▶ Specific statistical methods
- ▶ Cookbook recipes for particular analyses with R

What you should know

- ▶ Very basic math and statistics
(vectors, logarithms, correlation, t -tests, ...)
- ▶ Some familiarity with programming/scripting
and/or with a command-line environment
- ▶ Interest in (computational) linguistics

Course syllabus

- ▶ Introduction to R: set-up, data manipulation and exploration, plotting, basic statistics, input/output
- ▶ Hypothesis tests for corpus frequency data
- ▶ Using an R extension package: modelling word frequency distributions with zipfR
- ▶ Unsupervised multivariate data exploration: principal component analysis and clustering
- ▶ Co-occurrence statistics and frequency comparisons: contingency tables, association measures, evaluation
- ▶ Efficient data processing using vector operations
- ▶ The limitations of random sampling models for corpus data

Introductions



Who are you?

R textbooks for (computational) linguists

Much more comprehensive theoretical background and cookbook examples

- ▶ Stefan Th. Gries (to appear). ***Statistics for Linguistics with R: A practical introduction***. Mouton de Gruyter.
 - ▶ German original is already available
- ▶ Shravan Vasishth (2006–2009). ***The foundations of statistics: A simulation-based approach***.
 - ▶ <http://www.ling.uni-potsdam.de/~vasishth/SFLS.html>
- ▶ R. Harald Baayen (2008). ***Analyzing Linguistic Data: A practical introduction to statistics***. CUP.
 - ▶ <http://www.ualberta.ca/~baayen/publications.html>
 - ▶ if you download the PDF, you should also buy the book

Other recommended textbooks on statistics and R

- ▶ Peter Dalgaard (2008). ***Introductory Statistics with R***, 2nd ed. New York: Springer.
- ▶ Morris H. DeGroot and Mark J. Schervish (2002). ***Probability and Statistics***, 3rd ed. Addison Wesley.
 - ▶ Stefan's favourite statistics textbook
- ▶ John M. Chambers (2008). ***Software for Data Analysis: Programming with R***. New York: Springer.
- ▶ Christopher Butler (1985), ***Statistics in Linguistics***. Oxford: Blackwell.
 - ▶ out of print and available online for free download
 - ▶ <http://www.uwe.ac.uk/hlss/llas/statistics-in-linguistics/bkindex.shtml>

Course materials

- ▶ Handouts, example scripts and data sets are available on our homepage for this course:

<http://purl.org/stefan.evert/SIGIL/>

- ▶ You will also find additional material, software and links to background reading there

Outline

General Information

What is R?

About this course

R Basics

Basic functionalities

External files and data-frames

A simple case study: comparing Brown and LOB documents

Outline

General Information

What is R?

About this course

R Basics

Basic functionalities

External files and data-frames

A simple case study: comparing Brown and LOB documents

R as an oversized calculator

```
> 1+1
```

```
[1] 2
```

```
> a <- 2      # assignment does not print anything by default
```

```
> a * 2
```

```
[1] 4
```

```
> log(a)      # natural, i.e. base-e logarithm
```

```
[1] 0.6931472
```

```
> log(a, 2)   # base-2 logarithm
```

```
[1] 1
```

Basic session management

Some of it is not necessary if you only use the GUI

to start R on command line, simply type **R**

```
setwd("path/to/data") # or use GUI menus
```

```
ls() # probably empty for now
```

```
ls # notice difference with previous line
```

```
quit() # or use GUI menus
```

```
quit(save="yes")
```

```
quit(save="no")
```

NB: at least some interfaces support history recall, tab completion

Vectorial math

```
> a <- c(1, 2, 3) # c (for combine) creates vectors
```

```
> a * 2 # operators are applied to each element of a vector
```

```
[1] 2 4 6
```

```
> log(a) # also works for most standard functions
```

```
[1] 0.0000000 0.6931472 1.0986123
```

```
> sum(a) # basic vector operations: sum, length, product, ...
```

```
[1] 6
```

```
> length(a)
```

```
[1] 3
```

```
> sum(a)/length(a)
```

```
[1] 2
```

Initializing vectors

```
> a <- 1:100 # integer sequence
> a

> a <- 10^(1:100)

> a <- seq(from=0, to=10, by=0.1) # general sequence

> a <- rnorm(100) # 100 random numbers

> a <- runif(100, 0, 5) # what you're used to from Java etc.
```

Summary statistics

```
> length(a)
```

```
> summary(a)      # statistical summary of numeric vector
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.02717 0.51770 1.05200 1.74300 2.32600 9.11100
```

```
> mean(a)
```

```
> median(a)
```

```
> sd(a)           # standard deviation is not included in summary
```

```
> quantile(a)
  0%    25%   50%   75%  100%
0.0272 0.5177 1.0518 2.3261 9.1107
```

```
> quantile(a, .75)
```

Basic plotting

```
> a<-2^(1:100) # don't forget the parentheses!
```

```
> plot(a)
```

```
> x<-1:100 # most often: plot x against y
```

```
> plot(x,a)
```

```
> plot(x,a,log="y") # various logarithmic plots
```

```
> plot(x,a,log="x")
```

```
> plot(x,a,log="xy")
```

```
> plot(log(x),log(a))
```

```
> hist(rnorm(100)) # histogram and density estimation
```

```
> hist(rnorm(1000))
```

```
> plot(density(rnorm(100000)))
```

(Slightly less) basic plotting

```
> a <- rbinom(10000,100,.5)
> hist(a)

> hist(a, probability=TRUE)
> lines(density(a))

> hist(a, probability=TRUE)
> lines(density(a), col="red", lwd=3)

> hist(a, probability=TRUE,
      main="Some Distribution", xlab="value",
      ylab="probability")
# better to type command on a single line!
> lines(density(a), col="red", lwd=3)
```

Help!

```
> help("hist")    # R has excellent online documentation
> ?hist          # short, convenient form of the help command

> help.search("histogram")

> ?help.search

> help.start()   # searchable HTML documentation

# or use GUI menus to access & search documentation
```

Installing add-on packages

- ▶ Much of R's power comes from its add-on packages
- ▶ Can be downloaded from CRAN with GUI installer
 - ▶ automatically installs other required packages
 - ▶ Mac OS X: check “install dependencies”
 - ▶ Windows: only most essential dependencies installed

Installing add-on packages

- ▶ Much of R's power comes from its add-on packages
- ▶ Can be downloaded from CRAN with GUI installer
 - ▶ automatically installs other required packages
 - ▶ Mac OS X: check “install dependencies”
 - ▶ Windows: only most essential dependencies installed
- ▶ The “sumo” package for linguists: **languageR**
 - ▶ data sets & utilities for Baayen (2008)
 - ▶ also installs most other packages that you'll need
- ▶ Magic command: `install.packages("languageR",
libPaths()[1], dependencies=TRUE)`

Installing add-on packages

- ▶ Much of R's power comes from its add-on packages
- ▶ Can be downloaded from CRAN with GUI installer
 - ▶ automatically installs other required packages
 - ▶ Mac OS X: check “install dependencies”
 - ▶ Windows: only most essential dependencies installed
- ▶ The “sumo” package for linguists: **languageR**
 - ▶ data sets & utilities for Baayen (2008)
 - ▶ also installs most other packages that you'll need
- ▶ Magic command: `install.packages("languageR", .libPaths()[1], dependencies=TRUE)`
- ▶ Other highly recommended packages:
 - ▶ `corpora` for a few data sets used in this course
 - ▶ `rgl` and `misc3d` for interactive 3D graphics
 - ▶ `plyr` and `gsubfn` for convenience
 - ▶ advanced: `rggobi` for high-dimensional visualisation

Your first R script

- ▶ Simply type R commands into a text file & save it
- ▶ Use built-in GUI functionality or external text editor
 - ▶ Microsoft Word is *not* a text editor!
 - ▶ nor is Apple's TextEdit application ...
- ▶ Execute R script from GUI editor or by typing

```
> source("my_script.R") # more about files later
> source(file.choose()) # select with file dialog box
```
- ▶ Just typing a variable name will not automatically print its value in a script: use `print(sd(a))` instead of `sd(a)`

Outline

General Information

What is R?

About this course

R Basics

Basic functionalities

External files and data-frames

A simple case study: comparing Brown and LOB documents

Input from an external file

- ▶ We like to keep our data in space- or TAB-delimited text files with a first row (“header”) labeling the fields, like so:

```
word    frequency  cat
dog     15         noun
bark    10         verb
```

- ▶ This is an easy format to import into R, and it is easy to convert from/to other tabular formats using standard tools
- ▶ We assume that external input is always in this format (or can easily be converted to it)
 - ▶ spreadsheet applications prefer CSV format (comma-separated values)
 - ▶ Microsoft Excel is a nice table editor, but beware of localised number formats

Reading a TAB-delimited file with header

```
> brown <- read.table("brown.stats.txt",  
  header=TRUE)  
# if file is not in working directory, you must specify the full path  
# (or use setwd() function we introduced before)  
  
# exact behaviour of file.choose() depends on operating system  
> brown <- read.table(file.choose(), header=TRUE)  
  
# more robust if you are sure file is in tab-delimited format  
> brown <- read.delim("brown.stats.txt")
```

Reading and writing CSV files

R can also read and write files in CSV format

```
> write.csv(brown, "brown.stats.csv",  
  row.names=FALSE)
```

this is convenient for exchanging data with database and
spreadsheet software (or using Excel as a data editor)

NB: comma-separated values are not always separated by commas
(e.g. in German; use `write.csv2` if Excel doesn't recognise columns)

```
> write.csv2(brown, "brown.stats.csv",  
  row.names=FALSE)
```

TASK: load `brown.stats.csv` into Excel or OpenOffice.org

check generated CSV file (use `read.csv2` with `write.csv2` above)

```
> brown.csv <- read.csv("brown.stats.csv")  
> all.equal(brown.csv, brown)
```

Data-frames

- ▶ The commands above create a **data frame**
- ▶ This is the basic data structure (object) used to represent statistical tables in R
 - ▶ rows = objects or “observations”
 - ▶ columns = variables, i.e. measured quantities
- ▶ Different types of variables
 - ▶ numerical variables (what we've used so far)
 - ▶ Boolean variables
 - ▶ factor variables (nominal or ordinal classification)
 - ▶ string variables
- ▶ Technically, data frames are collections of column vectors (of the same length), and we will think of them as such

Data-frames

```
> summary(brown)
```

```
> colnames(brown)
```

```
> dim(brown)           # number of rows and columns
```

```
> head(brown)
```

```
> plot(brown)
```

Access vectors inside a data frame

```
> brown$to
```

```
> head(brown$to)
```

**# TASK: compute summary statistics (length, mean, max, etc.)
for vectors in the Brown data frame**

what does the following do?

```
> summary(brown$ty / brown$to)
```

```
> attach(brown)    # attach data frame for convenient access
```

```
> summary(ty/to)
```

```
> detach()    # better to detach before you attach another frame
```

More data access

```
> brown$ty[1]      # vector indexing starts with 1
> brown[1,2]      # row, column

> brown$ty[1:10]  # use arbitrary vectors as indices
> brown[1:10,2]

> brown[1,]
> brown[,2]
```

Conditional selection

```
> brown[brown$to < 2200, ] # index with Boolean vector
> length(brown$ty[brown$to >= 2200])
> sum(brown$to >= 2200) # standard way to count matches

> subset(brown, to < 2200) # no need to attach here
> lessdata <- subset(brown, to < 2200)

> a <- brown$ty[brown$to >= 2200]

# equality: == (also works for strings)
# inequality: !=
# complex constraints: and &, or |, not !
# NB: always use single characters, not && or ||
```

Outline

General Information

What is R?

About this course

R Basics

Basic functionalities

External files and data-frames

A simple case study: comparing Brown and LOB documents

Type, token and word length counts in the Brown and LOB documents

Variables:

- to** Token count
- ty** Type count (*distinct* words)
- se** Sentence count
- towl** Average word length
(averaged across tokens in document)
- tywl** Average word length
(averaged across distinct types in document)

Procedure

- ▶ Collect basic summary statistics for the two corpora
- ▶ Check if there is a significant difference in the token counts (since document length was controlled by corpus builders)
- ▶ If difference is significant (we will see that it is), then type counts are not directly comparable, and sentence counts should be normalized (divide by token count)
- ▶ Is word length correlated to document length? (in which case, corpus comparison would also not be appropriate)

Procedure

- ▶ Collect basic summary statistics for the two corpora
- ▶ Check if there is a significant difference in the token counts (since document length was controlled by corpus builders)
- ▶ If difference is significant (we will see that it is), then type counts are not directly comparable, and sentence counts should be normalized (divide by token count)
- ▶ Is word length correlated to document length? (in which case, corpus comparison would also not be appropriate)
- ▶ Please read the LOB data set into a data frame named `lob` now, and take a look at its basic statistics
- ▶ Also, plot the data frame for a first impression of correlations between the variables

Comparing token counts

```
> boxplot(brown$to, lob$to)
> boxplot(brown$to, lob$to, names=c("brown", "lob"))
> boxplot(brown$to, lob$to, names=c("brown", "lob"),
  ylim=c(1500, 3000))
> ?boxplot

> t.test(brown$to, lob$to)
> wilcox.test(brown$to, lob$to)

> brown.to.center <- brown$to[brown$to > 2200
  & brown$to < 2400]
> lob.to.center <- lob$to[lob$to > 2200
  & lob$to < 2400]

> t.test(brown.to.center, lob.to.center)
```

how about sentence length?

Is word length correlated with token count?

average word length by tokens and types almost identical:

```
> plot(brown$towl, brown$tywl)
> cor.test(brown$towl, brown$tywl)
> cor.test(brown$towl, brown$tywl,
  method="spearman")
```

correlation with token count

```
> plot(brown$to, brown$towl)
> cor.test(brown$to, brown$towl)
```