# R Programming for Computational Linguists and Similar Creatures

Marco Baroni[1] and Stefan Evert[2]

[1]Center for Mind/Brain Sciences
University of Trento

[2]Cognitive Science Institute
University of Onsabrück

Potsdam, 3-14 September 2007

# Outline

# Goals of the course

- Learn R basics and basic R programming
- Learn R implementations of various statistical/data analysis techniques useful in various domains of (computational) linguistics
- A little bit of background in statistics along the way
- Practice R skills on real-life data-sets

# What this course is *not* about

- Statistical theory
- Specific statistical methods
- Cookbook recipes for specific 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 issues

# A tentative syllabus 1

## Topics we will probably cover

- Introduction to R: set-up, data manipulation and exploration, plotting, basic statistics, input/output
- Using an R extension package:
  frequency distribution modeling with zipfR
- Co-occurrence statistics and frequency comparisons:
  contingency tables, association measures, evaluation
- Unsupervised multivariate data exploration:
  principal component analysis and clustering

# A tentative syllabus 2

## Further topics, to be selected depending on time and interests

- Supervised machine learning
- Matrix operations and linear algebra: application to the word space model
- More R programming: functions, list processing, non-interactive use
- Advanced 2D and 3D plots
- Generalized linear models, mixed effect models

# Some useful R references for linguists

## Available on the net, cover the theoretical and cookbook stuff we'll skip

- Shravan Vasishth, *The foundations of statistics: A simulation-based approach*
  `http://www.ling.uni-potsdam.de/~vasishth/SFLS.html`

- Harald Baayen, *Analyzing Linguistic Data: A practical introduction to statistics*
  `http://www.mpi.nl/world/persons/private/baayen/publications/baayenCUPstats.pdf`
  - (If you print this, you should commit yourself to buying the final published version.)

# Some textbooks on statistics & R programming

- Peter Dalgaard, *Introductory Statistics with R*. New York: Springer, 2002.
- Morris H. DeGroot and Mark J. Schervish, *Probability and Statistics*, 3rd edition. Boston: Addison Wesley, 2002.
  - (Stefan's favourite statistics textbook.)
- Christopher Butler, *Statistics in Linguistics*. Oxford: Blackwell, 1985.
  `http://www.uwe.ac.uk/hlss/llas/statistics-in-linguistics/bkindex.shtml`
  - (Out of print and available online for free download.)

# Course materials

- Handouts, example scripts, data sets available online:

  **http://www.ling.uni-potsdam.de/fallschool/r/**

- Homework assignments
  - mainly to encourage you to get practice with R :-)
  - required to get credit for the fall school
  - hand in solutions as plain text files by e-mail to **fallschool.R@gmail.com**

# Outline

# R

- `http://www.r-project.org/`
- Free, open source development of the S language of Venables and Ripley
- Available for Linux, Mac and Windows
- Command-line interface and GUI (for Mac and Windows)
  - for Windows, we recommend `www.sciviews.org` GUI
- Non-interactive use possible via scripting
- Less user-friendly than other statistical software, but immensely more powerful
- A wealth of packages implementing impressive range of classic and cutting edge statistical and data analysis techniques available

# Outline

# 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
```

# 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 value in scripts: use `print(sd(a))` instead of `sd(a)`

# Outline

# Input from an external file

- We like to keep our data in space/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 other formats into this one using other 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)

# Reading in 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")

# R can also read files in CSV format
> brown <- read.csv("brown.stats.csv")
```

# 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$to < 2200, ]   # index with Boolean vector
> length(mydata$ty[mydata$to >= 2200])
> sum(mydata$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

# 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 significant difference in the token counts (since document length in tokens was controlled by corpus builders)
- If difference is significant (we will see that it is), then types are not truly comparable on doc-by-doc basis, and sentence lengths should be normalized (dividing 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 significant difference in the token counts (since document length in tokens was controlled by corpus builders)
- If difference is significant (we will see that it is), then types are not truly comparable on doc-by-doc basis, and sentence lengths should be normalized (dividing by token count)
- Is word length correlated to document length? (in which case, corpus comparison would also not be appropriate)
- Please read in the LOB data-set in a LOB data-frame and look at basic statistics
- Also, plot the data-frame for a quick look at relations between 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?

```
# token and type wl are 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)
```