Reading in data

Leo Schalkwyk

MRC
Social, Genetic and Developmental Psychiatry Centre
Institute of Psychiatry
London

June 12, 2014

Where's the spreadsheet?

- statistical packages normally have a spreadsheet
- R has a minimal-but-usable spreadsheet
- more emphasis on data generated/curated externally
- very powerful data import tools
- can talk to databases

review of data structures -vectors

- everything is based on vectors
- a vector is a series of values of the same type
- we've seen 'character', 'numeric' and 'logical' vectors
- a 'factor' is a special type of vector for categorical data

review of data structures -attributes

- attributes are information specific to an object
- can be anything including your own notes
  - `attr(mydata,'comment')<-'deeply dubious data'`
- attributes not called 'comment' are shown when you print

especially important attributes

- dimensions, names

```
a<-1:100
dim(a)<-c(10,10)
a[50]
a[10,5]
colnames(a)<-letters[1:10]
a[,'j']
```

review of data structures - lists, data frames

- a table of data that's all one vector must be all one type
- your typical excel file isn't like that
- a list is a collection of any assorted objects
- a dataframe is a list of assorted vectors all the same length
  - the vectors (columns) must also have unique names
  - `b<-data.frame(a)`
  - `b$j`
  - `b[,10]`
- dataframes can be treated like matrices for many purposes
- there are a few stumbling blocks
  - compare `a[1]` and `b[1]`

editing or entering data in R

- you can make a data frame with `data.frame()`
- almost always it's best to do `options(stringsAsFactors=FALSE)`

```
dogs<-data.frame() # empty data frame
edit(dogs)         # doesn't save anything!
dogs<-edit(data.frame())
```

the read.table() function

- the commonest way to input tabular data, eg Excel
  - many other possibilities exist
- save data as tab-or comma separated text
- one sheet per file
- use a version without too much extraneous junk
- simple row and column labels, one line header

read.table(), continued

- the defaults work in simple cases
- safer to be explicit for general use

```
(fpe <-read.table("http://data.princeton.edu/wws509/datasets/effort.dat"))
str(fpe)
readLines(a <-'http://www.cs.tut.fi/~jkorpela/tsv.tsv'))
read.table(a, sep='\t'))
```

gdata package

- `read.xls()`
- works for xls and xlsx files
- understands multi-sheet workbooks
- based on read.table

the foreign package

- add- on package, need to install and load
  - `install.packages('foreign')`
  - `library(foreign)`
- has methods for importing data files from most stats packages
- SAS,SPSS,STATA, Systat not Statistica
- also some database like files
- .DBF (xbase, foxpro, clipper, etc)
- corresponding export methods as well

Microsoft Excel and Access

- RODBC R Open Database Connectivity
- easy to read in whole Excel or Access tables

```
library(RODBC)
ch<-odbcConnectExcel()
sqlTables(ch)
sqlFetch(ch,'table')
```

data may need cleanup

- missing value codes often differ
- people often use -1 or 999
- statistics software varies as well
- type coersion is sometimes all you need
- graphing and summaries are important sanity checks
- `image(dat)` is a favourite

keeping your data under control

- both workspace and command history can be saved
- you can restart where you left off
- this gives a false sense of security
- a much better strategy is to keep a log
  - paste commands that have done what you want into an editor
  - (optional) name it something.R
  - put comments starting with #
  - such a log can be used as a script to generate your analysis again