

Introduction to R

Network Analysis: Practical 1

Sacha Epskamp

University of Amsterdam
Department of Psychological Methods

04-09-2014



What is R?

- ▶ R is a statistical programming language
 - ▶ Statistical analysis
 - ▶ Data visualization
 - ▶ Data mining
 - ▶ General programming
- ▶ It is *open-source*
 - ▶ Free as in “free beer” and “free speech”
 - ▶ Large active community around R
 - ▶ Many contributed packages



Why do you need to know R?

- ▶ R is very powerful, and allows us to manually compute everything we want if we choose to
- ▶ For many of the methods we use, R has either the simplest or the only implementation.
- ▶ The use of networks in psychology is very new and we are developing its methodology now. Since we don't own SPSS or any other commercial software, R is the only way for us to develop tools others can use



What do you need to know about R?

- ▶ In this course not much. We intend the methods we will describe to be usable by researchers with little experience in R
 - ▶ No programming
 - ▶ Cookbook method
- ▶ But it is strongly advised that you learn more about R if you want to do more in methodology or psychometrics.
- ▶ To install R and RStudio see links section on Blackboard



First use of R

- ▶ We will use the environment RStudio for our work in R
- ▶ RStudio has 4 panels:
 - Console This is the actual R window, you can enter commands here and execute them by pressing enter
 - Source This is where we can edit *scripts*. It is where you should always be working. Control-enter sends selected codes to the console
 - Plots/Help This is where plots and help pages will be shown
 - Workspace Shows which objects you currently have
- ▶ Anything following a # symbol is treated as a comment!



Use R as calculator

```
1 + 1
```

```
## [1] 2
```

```
(10 * 20) / 100
```

```
## [1] 2
```

```
exp(1.5 * (2.1 - 1.8)) / (1 + exp(1.5 * (2.1 - 1.8)))
```

```
## [1] 0.6106
```



Numbers

```
# Numbers:
```

```
1.5
```

```
## [1] 1.5
```

```
10
```

```
## [1] 10
```



Strings

```
# Strings (within single or double quotes):
'this is a string'

## [1] "this is a string"

"this is also a string"

## [1] "this is also a string"

"
This is a very long string
with multile lines
"

## [1] "\nThis is a very long string\n\nwith multile lines\n"
```



Logicals

```
# Logical (only TRUE or FALSE):
```

```
TRUE
```

```
## [1] TRUE
```

```
FALSE
```

```
## [1] FALSE
```



Assign and use objects

- ▶ The `<-` operator can be used to store values into *objects*
- ▶ an object can contain anything in R
- ▶ Objects can be overwritten
- ▶ R expressions that are not stored in an object are *printed*



Assign and use objects

```
a <- 1
```

```
a
```

```
## [1] 1
```

```
b <- 2
```

```
a + b
```

```
## [1] 3
```

```
a <- a + b
```

```
a
```

```
## [1] 3
```

```
b
```

```
## [1] 2
```



Vectors

- ▶ Everything in R is basically a vector
- ▶ Use `c(1, 2, 3)` to manually create a vector
- ▶ A colon can be used to create an integer sequence
- ▶ Vectors can be calculated with. Either calculate with vectors of same length or use a vector and a single number.
- ▶ A vector can be *indexed* using square brackets



Vectors

```
# Create vectors:
v1 <- c(5, 2, 10, 1)
v1

## [1] 5 2 10 1

v2 <- 1:4
v2

## [1] 1 2 3 4

# Add them:
v1 + v2

## [1] 6 4 13 5

# Add 1 to all elements of v1:
v1 + 1

## [1] 6 3 11 2
```



Indexing

```
v1 <- c(5, 2, 10, 1)
```

```
v1
```

```
## [1] 5 2 10 1
```

```
# Get 3rd element of v1:
```

```
v1[3]
```

```
## [1] 10
```

```
# Change this element:
```

```
v1[3] <- 0
```

```
v1
```

```
## [1] 5 2 0 1
```



Functions

- ▶ A *function* is a small program, it takes input, does something and gives you output
- ▶ It is always of the form
name(argument, argument, argument, ...)
- ▶ Its output needs to be stored in an object if you want to keep it
- ▶ A documentation can be found for every function using ?name
- ▶ See for an extensive list of many functions the reference card on blackboard



Functions

```
v1 <- c(5, 2, 10, 1)
# Compute mean of v1:
mean(v1)
```

```
## [1] 4.5
```

```
# Compute sum of v1:
sum(v1)
```

```
## [1] 18
```

```
# Help pages of these functions:
```

```
?mean
```

```
?sum
```



Matrices

- ▶ A very important function for this course is `matrix()`. It creates a matrix, which is basically a two dimensional table.
- ▶ Technically, a matrix is a vector with two dimension attributes
 - ▶ Rows indicate horizontal lines of cells
 - ▶ Columns indicate vertical lines of cells
- ▶ The first argument of `matrix` is a vector to fill the matrix with, the second argument the number of rows and the third argument the number of columns (see also `?matrix`)
- ▶ Again they can be indexed with square brackets, but now need both row and column information, separated by a comma



Matrices

```
set.seed(1)
# A matrix:
m1 <- matrix(sample(1:9),3,3)
m1

##      [,1] [,2] [,3]
## [1,]    3    6    8
## [2,]    9    2    7
## [3,]    5    4    1

# Another matrix:
m2 <- matrix(sample(1:9),3,3)
m2

##      [,1] [,2] [,3]
## [1,]    1    5    6
## [2,]    2    7    3
## [3,]    8    4    9
```



Indexing matrices

```
# Index first row second column:
```

```
m2[1,2]
```

```
## [1] 5
```

```
# Overwrite an element:
```

```
m2[3,3] <- 0
```

```
m2
```

```
##      [,1] [,2] [,3]
```

```
## [1,]    1    5    6
```

```
## [2,]    2    7    3
```

```
## [3,]    8    4    0
```



Lists

- ▶ The `list()` function can be used to create a *list*
- ▶ This is an object that can contain other objects
- ▶ To index a list use double square brackets, or a dollar sign (see example on next slides).



Lists

```
# A vector:
v1 <- c(5,10,1,3)
v1

## [1] 5 10 1 3

# A matrix:
m1 <- matrix(c(5, 2, 10, 1),2,2)
m1

##      [,1] [,2]
## [1,]    5  10
## [2,]    2    1

# Put them in a list:
l1 <- list(v1 = v1, m1 = m1)
```



Lists

```
str(l1)

## List of 2
## $ v1: num [1:4] 5 10 1 3
## $ m1: num [1:2, 1:2] 5 2 10 1
```



Indexing lists

```
# Index the vector v1:
l1$v1

## [1] 5 10 1 3

l1[['v1']]

## [1] 5 10 1 3

# Change an element in the matrix m1:
l1$m1[2,2] <- 0
l1$m1

##      [,1] [,2]
## [1,]    5  10
## [2,]    2    0
```



Data frames

- ▶ The `data.frame()` function can be used to create a *data frames*
- ▶ A data frame is a combination of a matrix and a list
 - ▶ Looks like a matrix and can be indexed as one
 - ▶ Contains *variables* as each row, which can be indexed as in lists
- ▶ This is the structure you will use when working with data
- ▶ Very similar to how data is stored in SPSS!



Data frames

A character vector:

```
sex <- c("male", "female", "male", "female")
```

A logical vector:

```
exp <- c(TRUE, TRUE, FALSE, FALSE)
```

2 numeric vectors:

```
A <- c(5, 10, 1, 3)
```

```
B <- 1:4
```

Put them in a data frame:

```
df1 <- data.frame(sex=sex, exp=exp, A=A, B=B)
```



Data frames

```
df1
```

```
##      sex  exp  A B
## 1  male TRUE  5 1
## 2 female TRUE 10 2
## 3  male FALSE  1 3
## 4 female FALSE  3 4
```



Indexing data frames

```
# Index the vector sex:
df1$sex

## [1] male   female male   female
## Levels: female male

df1[['sex']]

## [1] male   female male   female
## Levels: female male

# Subset of the data containing only A and B:
df1[,c('A', 'B')]

##      A B
## 1   5 1
## 2  10 2
## 3   1 3
## 4   3 4
```



Correlation matrix

Some functions take a data frame as input. Very important in this course is the function `cor`, which takes a data frame as input and computes the *correlation matrix*. The function `cov` computes the covariance (unstandardized correlations).

```
# Covariance matrix:  
cov(df1[,c('A', 'B')])
```

```
##           A           B  
## A  14.92  -2.500  
## B  -2.50   1.667
```

```
# Correlation matrix:  
cor(df1[,c('A', 'B')])
```

```
##           A           B  
## A  1.0000 -0.5014  
## B -0.5014  1.0000
```



Packages

- ▶ Packages are extensions contributed to R containing extra functions
- ▶ They can be installed using `install.packages()`
- ▶ Afterwards they can be loaded using `library()`



Packages

```
# Install package "qgraph"  
install.packages("qgraph")  
  
# Load package "qgraph":  
library("qgraph")
```



Load data into R

- ▶ There are many ways to load data into R
- ▶ The most common way is to read a plain text file (which can be exported from excel for instance) using `read.table` or `read.csv` (see their help files for how to do this)
- ▶ Because psychologists often use SPSS, it is useful to directly import data from SPSS
- ▶ This can be done using `read.spss()` in the `foreign` package.
- ▶ `file.choose()` can be used to select a file (it might be opened in the background of RStudio)



Load SPSS data into R

```
# Load package "foreign":  
library("foreign")  
  
# Select a SPSS file:  
file <- file.choose()  
  
# Read data:  
Data <- read.spss(file,to.data.frame=TRUE)  
# A warning about "Unrecognized record type" doesn't seem  
# to be a problem, so ignore it
```



Download `introR.pdf` from the course website and work through it (sections 7, 10.2 and 11 can be skipped, although might be interesting nonetheless).

If you had already done that, work through chapters 2 and 3 of http://cran.r-project.org/doc/contrib/Paradis-rdebuts_en.pdf

