Introduction to Social Network Analysis with **R**

Part 1: Introduction to **R**

Michał Bojanowski
m.bojanowski@uw.edu.pl
www.bojanorama.pl/snar:start

ICM, University of Warsaw

July 1, 2014
EUSN 2014, Barcelona

Outline of the workshop

1. Introduction to **R**
2. Basics of SNA in **R**

Goals and work flow

Goal

Give enough information to be able to start using **R** on your own.

General approach: **R** for non-programmers.

- Task-oriented rather than formally describing the language.
- Presentation and **R** examples well interspersed

All slides and scripts are available on-line:

1. Navigate to www.bojanorama.pl
2. EUSN link in the menu on the right.
3. Alternatively, via "Teaching" link on top

Data in this part of the workshop

Two example datasets. Subsets of Polish General Social Survey (PGSS) retrieved from www.ads.org.pl:

`sex_data.txt` tab-delimited plain text file with opinions on gender roles.

`earnings_data.sav` SPSS system file with data on salaries

Plan for part 1 (intro to R)

1. What is R?
2. Basics of R and RStudio
3. Vectors
4. Data
5. Descriptive statistics
6. Creating functions
7. Tables and matrices
8. Group-wise descriptives
9. Linear Regression

Outline

1. What is R?
2. Basics of R and RStudio
3. Vectors
4. Data
5. Descriptive statistics
6. Creating functions
7. Tables and matrices
8. Group-wise descriptives
9. Linear Regression

What is R?

Program for data analysis For many people R is a program of choice for statistical analysis, visualization and predictive analytics.

Programming language Data analysis with functions and scripting. Interactive programming language.

Environment for statistical analysis Access to standard models and cutting edge methods.

Open Source project Open source code. Free. Over 15 years of peer review. Integration with other tools/systems.

Community 20 people in R Core. Approximately 2 mln users worldwide: forums, mailing lists, blogs.

CRAN and addon packages

- R's functionality is contained in packages.
  - Base distribution contains 32 packages.
  - Developers and users created the next ~~4528~~ **5163** (as on 2014-02-03) packages available on the Internet on the CRAN (Comprehensive R Archive Network) servers.
- Other things on CRANs
  - Official documentation
  - User-contributed documentation
  - R Journal

CRAN master server in Vienna, Austria

http://cran.r-project.org

Outline

1. What is R?
2. Basics of R and RStudio
3. Vectors
4. Data
5. Descriptive statistics
6. Creating functions
7. Tables and matrices
8. Group-wise descriptives
9. Linear Regression

Interacting with R

Bare R:

- Native R interface is a bare console.
- Windows and Mac versions come with a Graphical User Interface (GUI), but they are still rather basic.

RStudio (www.rstudio.com):

- Script editor with syntax highlighting.
- Workspace explorer.
- Access to help system.
- User-friendly access to many useful operations.

▸ RStudio: panes etc.

R console and R syntax

R syntax is

- Case-sensitive.
- # is a comment character.
- Spaces and line breaks can be used generously.

```
# This is a comment describing that below we are computing
# a base 2 logarithm of 8
log( 8,
    base=2 )
```

R syntax: typographical convention

In R script:

```
# This is a comment describing that below we are computing
# a base 2 logarithm of 8
log( 8,
    base=2 )
```

In R console:

```
> log(8,
+     base=2)
[1] 3
```

On these slides:

```
# This is a comment describing that below we are computing
# a base 2 logarithm of 8
log( 8,
    base=2 )

## [1] 3
```

R as an advanced calculator

R can be used as an advanced calculator.

- Type-in a mathematical formula and press Enter

Examples:

```
2 + 2             # simple computations
## [1] 4
3 + 2 * 3         # correct operator precedence
## [1] 9
sin(pi/2)^2       # mathematical functions and constants
## [1] 1
(3 * 2^2 - 4 * 3^2 ) / 9    # complex formulas
## [1] -2.667
```

Assignment

Computing would be useless if you cannot store a result for later use.

- Use '<-' (left arrow) to **assign** a result to a named **object**
- Typing name of the object will print its contents.
- Object name can consist of alphanumeric characters, '_', and '.'. Cannot start with a number though.

```
x <- 3 + 2 * 3                     # storing result as 'x'
x

## [1] 9

some_long.objectName <- x + 11  # reusing 'x'
some_long.objectName

## [1] 20
```

Using functions

Functions are the workhorses of **R**.

- Computations.
- Data manipulation and transformation.
- Visualization.
- Estimation of statistical models.
- etc. . .

A function is also an object. It has a **name** and **arguments** (inputs):

```
log
## function (x, base = exp(1))  .Primitive("log")
```

Calling functions

Function arguments can be specified in different ways:

```
log(1)           # using default value of 'base' argument
## [1] 0
log(2, base=2)   # specifying value of 'base' by name
## [1] 1
log(2, 10)       # specifying value of 'base' by order
## [1] 0.301
```

Values returned by one function can become an argument to different function:

```
log( cos(0), log(1) )
## [1] 0
```

▸ rintro.R: Basic computing

Help system

In the help system:

- Manuals.
- Package help pages.
- Help pages of individual functions.

Accessing help

- Using dedicated pane in RStudio
- Using functions `help` and `help.search`

Workspace

R **workspace** is where all the created objects reside.

- Dedicated pane in RStudio
- Using functions `ls` and `rm`

Operations:

- Listing objects
- Removing objects

▸ RStudio: Help and workspace

Working directory

R **working directory** is a default place where R looks for files.

- Where files are loaded from
- Where files are saved to

What's the current working directory?

- In RStudio look at the top of console pane
- Use `getwd` function

Set working directory

- Use RStudio menu "Session" » "Set working directory".
- Use `setwd` function.

Saving and loading

R objects can be saved and loaded to files with functions `save` and `load`. Using R's native file format.

- Saving objects `x` and `y` to file `filename.rda`.

```r
save(x, y, file='filename.rda')
```

- Loading objects from file

```r
load('filename.rda')
```

▸ rintro.R: workspace, save/load

Outline

1 What is R?

2 Basics of R and RStudio

3 Vectors

4 Data

5 Descriptive statistics

6 Creating functions

7 Tables and matrices

8 Group-wise descriptives

9 Linear Regression

Vectors

Vector is a basic type of **R** object.

- Ordered collection of elements of the same type.
- Elements: numbers, strings, logical values (TRUE/FALSE).
- Computations on vectors are performed **element-wise**
- Vector elements can have **names**.
- Special element for missing value: `NA`. More on that later.

Creating vectors

Vectors can be created using c function

```
v <- c(1, 2, 3, 4, 5, 4, 2, 1) # numeric vector
v
```

```
## [1] 1 2 3 4 5 4 2 1
```

```
vnamed <- c(a=1, b=20, c=30) # named numeric vector
vnamed
```

```
##  a  b  c
##  1 20 30
```

```
ch <- c('R', 'is', 'great', 'for', 'SNA') # character vector
ch
```

```
## [1] "R"     "is"    "great" "for"   "SNA"
```

```
lg <- c(TRUE, FALSE, TRUE) # logical vector
lg
```

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

Computing with vectors

Computations on vectors are performed **element-wise** (element by element).

- Usual math: `+`, `-`, `*`, `/`, `^` (powers)
- Logical operators: `==`, `<=`, `>=`, `<`, `>`, `!` (not), `!=` (not equal).
- Common functions for numeric vectors: `length`, `sum`, `mean`.
- Common functions for logical vectors: `any`, `all`, `which`, `%in%` operator.
- `names` returns character vector of element names.
- Convert between vector types with: `as.numeric`, `as.character`, `as.logical`.
- Creating regular vectors with `seq`, `:` operator, `rep`.

Subscripting with [ ]

Subscripting

Referring to individual elements or subsets of elements of a larger object.

Syntax for subscripting some vector `v`:

```
v[ i ]
```

where `i` can be a vector (also of length 1) which is one of:

numerical Selecting elements on positions given by `i`. Negative `i` means “drop `i`th element”.

character Selecting elements with names given by `i`

logical Selecting elements, for which `i == TRUE`.

▸ rintro.R: Vectors

Outline

1. What is R?
2. Basics of R and RStudio
3. Vectors
4. Data
5. Descriptive statistics
6. Creating functions
7. Tables and matrices
8. Group-wise descriptives
9. Linear Regression

Importing data

R can import data from variety of formats:

- Native format: .rda with `load` and `save`.
- Reading plain text files (including tab-delimited, CSV, etc.) with `read.table`, `read.csv`, `read.csv2`.
- Reading SPSS files with `read.spss` from package `foreign`.
- Reading Stata files with `read.dta` from package `foreign`.
- Interacting with relational database systems, e.g. with package `RMySQL`
- Web-scraping with functions from `XML` package.

Most data-importing functions return a **data frame**: an object storing a rectangular data set (variables measured for cases).

Data frames

Data frame is a collection[1] of vectors of the same length.

- RStudio: click on object name to show the data matrix.
- Use `str` to compactly show the content.
- Get variable names with `names`.
- Extract vectors corresponding to columns with `$` operator (i.e. `datframe$varname`).
- Use function `with` to simplify expressions involving variables from the same data frame.
- Other useful functions: `dim`, `nrow`, `ncol`, `summary`, `head`, `tail`

▸ `rintro.R`: Importing and data frames

[1] Technically, a `list`. More on that later

Subscripting data frames

Subscripting data frames is also performed using `[ ]`, but now we have two dimensions to work with.
The syntax is:

```
datfr[ i, j ]
```

where `i` and `j` can be numeric/character/logical vectors as before.

- If either `i` or `j` is omitted, then we take the whole dimension (all rows or all columns).

Missing values

There is a single symbol representing missing value: `NA` (not available).

- Most of the basic computations involving `NA`s will result in an `NA`.

  ```
  v <- c(1, 2, NA, 3)
  mean(v)
  ## [1] NA
  ```

- Functions to deal with missing values: `is.na`, `na.omit`.

```
is.na(v)
## [1] FALSE FALSE  TRUE FALSE
```

Recoding

Two primary ways to recode variables (vectors):

- Using subscripting by assigning new values to subscripted elements:

```
# Replace all 3s and 4s with NA
x[ x %in% c(3,4) ] <- NA
```

- Using `replace` function. Syntax

```
replace( vec, index, newvalues )
```

Argument `index` can be anything what we'd use inside `[ ]`.

```
# Create a new vector 'y' from 'x' by replacing 3s with NAs
y <- replace(x, x==3 , NA)
```

▸ rintro.R: Subscripting, missing values, and recoding

Outline

1 What is R?

2 Basics of R and RStudio

3 Vectors

4 Data

5 Descriptive statistics

6 Creating functions

7 Tables and matrices

8 Group-wise descriptives

9 Linear Regression

Numerical and visual descriptives, formulas

There are many functions for numerical and visual statistical description.

- Functions: `mean`, `median`, `sd`, `var`, `quantile`.
- Built-in visualization functions, e.g.: `plot`, `barplot`, `pie`, `hist`, `boxplot`.

Some of them use **R** formulas: symbolic, equation-like, representation of relationships between variables, for example:

```
y ~ x + z
```

- Symbol `~` (tilde) separates “dependent” from “independent” variables.
- “`y` as a function of `x` and `z`”, or “`y` modeled with `x` and `z`”

▸ `rintro.R`: Descriptive stats

Outline

1 What is R?

2 Basics of R and RStudio

3 Vectors

4 Data

5 Descriptive statistics

6 Creating functions

7 Tables and matrices

8 Group-wise descriptives

9 Linear Regression

Creating functions

R function is also an object, which is created using a function `function`.

```
f <- function(x, y)
{
  pr <- x * y
  mean(pr)
}
```

- Function `f` expects two arguments, which are called `x` and `y` within function definition.
- Curly braces `{ }` are used to block several R expressions together.
- Last operation within `{ }` produces the value returned by `f`.

▸ rintro.R: Functions

Outline

1 What is R?

2 Basics of R and RStudio

3 Vectors

4 Data

5 Descriptive statistics

6 Creating functions

7 Tables and matrices

8 Group-wise descriptives

9 Linear Regression

Frequency tables, matrices, and arrays

We used `table` to compute frequency distributions. It can be used to create cross-tabulations

```
table(x, y, z, ...)
```

The result is a matrix or array (if more than two dimensions).

- Row and column names accessed with `rownames` and `colnames`.
- Subscripting with `[ ]` works like for data frames: more dimensions require more indices, e.g. `mat[ i, j, k ]`.
- Useful functions: `rowSums`, `rowMeans`, `colSums`, `colMeans`, `prop.table`,
- Constructing matrices from vectors or other matrices: `matrix`, `cbind`, `rbind`.

Function apply

Function apply can be used to compute **any function** on rows, columns or layers of an array/matrix. Syntax:

```
apply( tab, margin, fun, ... )
```

tab Array/matrix/table

margin id of a dimension, 1=rows, 2=columns, 3=layers, etc.

fun function to be applied

... other arguments passed to function fun

Compute row means of matrix mat disregarding NAs:

```
apply(mat, 1, mean, na.rm=TRUE)
```

▸ rintro.R: Tables and matrices

Outline

1. What is R?
2. Basics of R and RStudio
3. Vectors
4. Data
5. Descriptive statistics
6. Creating functions
7. Tables and matrices
8. Group-wise descriptives
9. Linear Regression

Functions `tapply` and `aggregate`

Often we need summarize variable(s) in groups defined by other variables.

- `tapply`: Use any function to summarize a single variable in subgroups defined by specified other variables, returns a matrix/array.

```
tapply( vec, groupingvecs, fun, ... )
```

- `aggregate`: Summarize a data frame of variables, returns a data frame.

```
aggregate( formula, data, fun, ...)
```

Example formula: `cbind(x,y) ~ g1 + g2`

Compute means of variables `x` and `y` in subgroups defined by `g1` and `g2`.

▸ rintro.R: Group-wise

Outline

1 What is R?

2 Basics of R and RStudio

3 Vectors

4 Data

5 Descriptive statistics

6 Creating functions

7 Tables and matrices

8 Group-wise descriptives

9 Linear Regression

Fitting regression models

Linear regression models are fitted using `lm` function, e.g.:

```
lm( y ~ x1 + x2 + ..., data=datframe )
```

- Model summary with `summary`, `coef`.
- Comparing nested models with `anova`.
- Model-predicted values with `predict`.
- Protect in-formula variable transformations with `I()`.

▸ rintro.R: Regression models