Quantitative Content Analysis: Lecture 2

Olga Kononykhina

15 Februar 2017

Today's Outline

Intro to R (Part I)

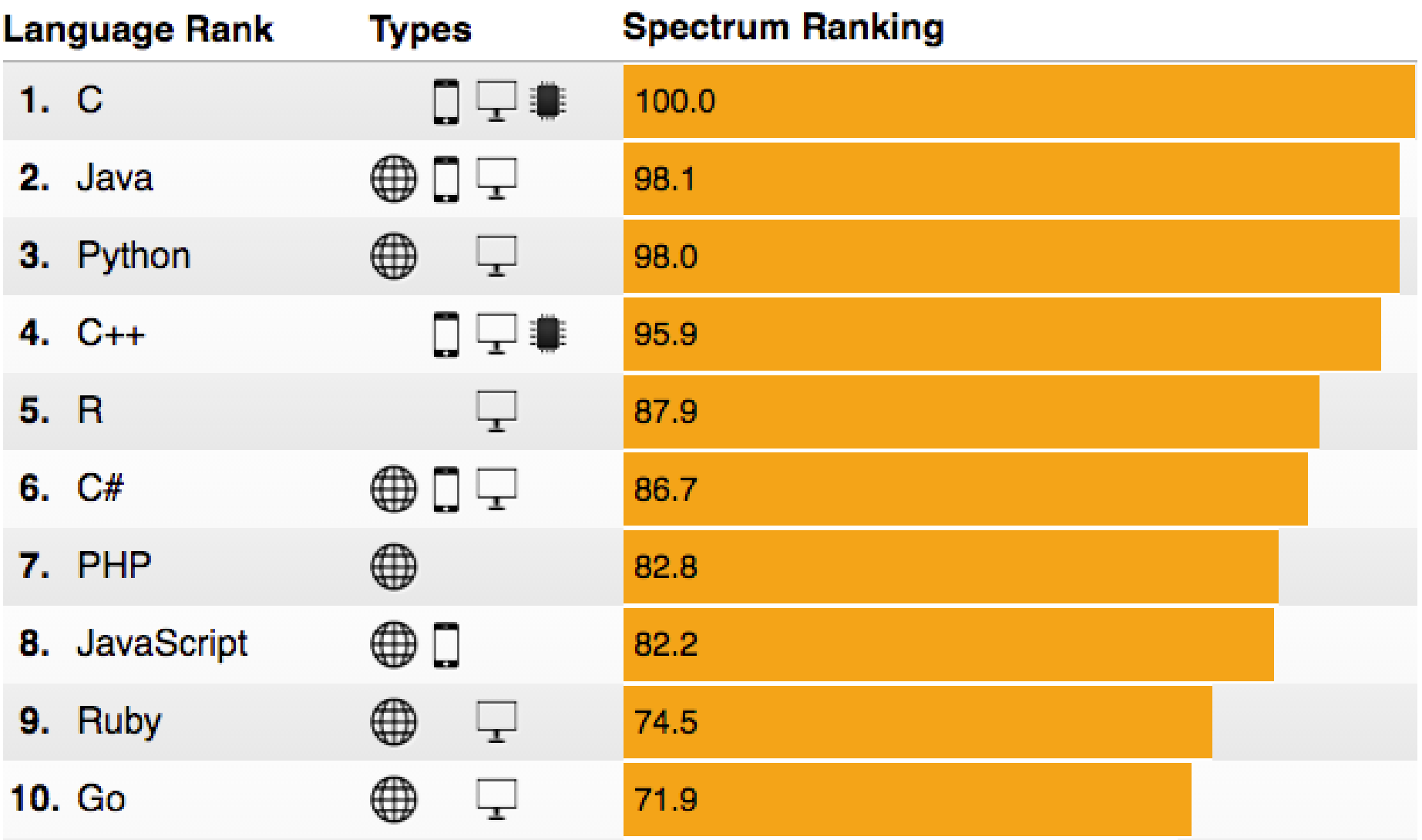
- R and RStudio
- Basic syntax
- Assigning objects
- Scalars, vectors, matrices
- Control flow: Loops and conditions

What is R?

R is an open source programming language with a particular focus on statistical programming

- Originally developed as 'S' by Bell Labs in 1993

R in comparison



(source: IEEE Spectrum)

R interface (console-only)

- Command line, unlike SPSS/Stata
- An interpreted programming language
- Purist's interface: Text-editor & copy paste

R with RStudio

RStudio is an Integrated Developer Environment (IDE) and serves as:

- Code editor
  - Code highlighting/completion, indentation, ...
  - Feed code from editor to R-console
- Project manager
- Workspace viewer
- Data browser
- Enhanced output viewer
- Help browser

Using RStudio

RStudio windows at startup

- Source editor
- Console
- Workspace
- Multi-purpose-panel

Projects

- A working directory for each project
- Code: .r files
- Dataset/Workspace: .Rdata files

Using RStudio (II)

Basic Workflow

- Edit in code editor (.r-file)
- Paste to console
- Save Workspace/Datasets (.Rdata-file)
- Save code routinely (no auto-save!)

Shortcuts

- Run code from editor: Select line and ctrl+Enter
- Switch between source and console: ctrl+1, ctrl+2
- Clear console: ctrl+L
- 'Arrow up' gives you the last line of code in the console
- Press Alt+Shift+K to see all keyboard shortcuts

Fundamentals of the R language

- Use # to comment code (will not be run)
- Case-sensitivity: `data` vs `Data`
- Assigning objects: <- and =

```
# Assign the number 5 to an object called number
number <- 5
number
```

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

```
# Assign the character string Hello World
string <- "Hello World"
string
```

```
## [1] "Hello World"
```

Functions

Functions perform operations on the input given and end in ()

```
# e.g. find the class of number
class(number)
```

```
## [1] "numeric"
```

Operations on scalars

You can use R as a calculator:

```
2 + 3
2 - 3
2 * 3
2 / 3
```

Functions on scalars:

```
a <- 5
factorial(a)
```

```
## [1] 120
```

Special values in R

- `NA`: not available, missing
- `NULL`: does not exist, is undefined
- `TRUE`, `T`: logical true
- `FALSE`, `F`: logical false

Finding special values

Function	Meaning
`is.na`	Is the value `NA`
`is.null`	Is the value `NULL`
`isTRUE`	Is the value `TRUE`
`!isTRUE`	Is the value `FALSE`

```
absent <- NA
is.na(absent)
```

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

Operations

Operator	Meaning
<	less than
>	greater than
==	equal to
<=	less than or equal to
>=	greater than or equal to
!=	not equal to
a | b	a or b
a & b	a and b

Naming objects

- Object names cannot have spaces
  - Use `CamelCase`, `name_underscore`, or `name.period`
- Avoid creating an object with the same name as a function (e.g. `c` and `t`) or special value (`NA`, `NULL`, `TRUE`, `FALSE`).
- Use descriptive object names
- Each object name must be unique in an environment.
  - Assigning something to an object name that is already in use will overwrite the object's previous contents.

R is object-oriented

Objects are R's nouns and include (not exhaustive):

- character strings
- numbers
- vectors of numbers or character strings
- matrices
- data frames
- lists

Vectors

A vector is a container of objects put together in an order.

```
# Define a vector
a <- c(1,4,5)
b <- c(3,6,7)
```

```
# Join multiple vectors
ab <- c(a,b)
```

```
# Find vector length (number of its elements)
length(a)
```

```
# Reference vector components
ab[4] # Index one element in vector
ab[4:6] # Index several elements in a vector
ab[ab==6] # Index with condition
```

Operations on vectors

Operation	Meaning
`sort(x)`	sort a vector
`sum(x)`	sum of vector elements
`mean(x)`	arithmetic mean
`median(x)`	median value
`var(x)`	variance
`sd(x)`	standard deviation
`factorial(x)`	factorial of a number

Task 1

Calculate the mean of the vector 1,99,3,4,5,6,8. What's the mean if you out the 'outlier'?

Task 1 (solution)

Calculate the mean of the vector 1,99,3,4,5,6,8. What's the mean if you out the 'outlier'?

```
# Defining vector using sequence between 3 and 6
a <- c(1,99,3:6,8)
mean(a)
```

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

```
# Calculate the mean of a but exclude the highest number
mean(a[a!=max(a)])
```

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

Matrices

A Matrix is a square 2 dimensional container, i.e. vectors combined by row or column

- Must specify number or rows and columns
  `matrix(x,nrow,ncol,byrow)`
  - x: vector of length nrow*ncol
  - nrow: number of rows
  - ncol: number of columns
  - byrow: TRUE or FALSE, specifies direction of input

Task 2

Assign a 6 x 10 matrix with 1,2,3,. . . ,60 as the data.

Task 2 (solution)

Assign a 6 x 10 matrix with 1,2,3,. . . ,60 as the data.

```
m <- matrix(1:60, nrow=6, ncol=10)
m
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## [1,]    1    7   13   19   25   31   37   43   49    55
## [2,]    2    8   14   20   26   32   38   44   50    56
## [3,]    3    9   15   21   27   33   39   45   51    57
## [4,]    4   10   16   22   28   34   40   46   52    58
## [5,]    5   11   17   23   29   35   41   47   53    59
## [6,]    6   12   18   24   30   36   42   48   54    60
```

Referencing matrices

- Like vectors, you can reference matrices by elements
  - `a[i,j]` refers to the ith row, jth column element of object `a`
- Can also reference rows/columns, these are vectors
  - `a[i,]` is ith row, `a[,j]` is jth column

Task 3

Extract the 9th column of the matrix from the previous problem. How can you find the 4th element in the 9th column?

Task 3 (solution)

Extract the 9th column of the matrix from the previous problem. How can you find the 4th element in the 9th column?

```
m[,9]
```

```
## [1] 49 50 51 52 53 54
```

```
m[4,9]
```

```
## [1] 52
```

```
m[,9][4]
```

```
## [1] 52
```

For() loops

For() loops are used to loop around a vector/matrix to do something.

```
m <- matrix(1:5, nrow=1, ncol=5)
m
```

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

```
for (j in 1:3){
     m[,j]=0
}
m
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    0    0    0    4    5
```

For() loops (II)

You can also 'nest' a `for()` loop in another `for()` loop

```
m <- matrix(1:15, nrow=3, ncol=5)
for (i in 1:2){
  for (j in 1:4){
    m[i,j]=0
  }
}
m
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    0    0    0    0   13
## [2,]    0    0    0    0   14
## [3,]    3    6    9   12   15
```

Task 4

Create a matrix with `matrix(1:20,nrow=4,ncol=5)` and another with `matrix(NA,nrow=4,ncol=5)`. Adapt the second to the first matrix using `for()`

- hint: define a 'counter' variable that increments by 1 each time you move to the next cell

Task 4 (solution)

Create a matrix with `matrix(1:20,nrow=4,ncol=5)` and another with `matrix(NA,nrow=4,ncol=5)`. Adapt the second to the first matrix using `for()`

```
counter=1
m1 <- matrix(1:20,nrow=4,ncol=5)
m2 <- matrix(NA,nrow=4,ncol=5)
for (j in 1:5){
  for (i in 1:4){
    m2[i,j]=counter
    counter=counter+1
  }
}
```

Task 4 (alternative solution)

Create a matrix with `matrix(1:20,nrow=4,ncol=5)` and another with `matrix(NA,nrow=4,ncol=5)`. Adapt the second to the first matrix using `for()`

```
m1 <- matrix(1:20,nrow=4,ncol=5)
m2 <- matrix(NA,nrow=4,ncol=5)
for (j in 1:5){
  for (i in 1:4){
    m2[i,j]=m1[i,j]
  }
}
```

If() statements

If() statements are used to make conditions on executing some code. If condition is true, action is done.

```
a <- 3
b <- 4
number <- 0
if(a<b){
  number=number+1
}
number
```

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

Tests for conditions: ==; >; <; >=; <=; !=

Task 6

Create the two objects `a <- sample(c(4,0),20,replace=TRUE)` and `m <- matrix(a,nrow=4,ncol=5)`. Now recode all the 4s into 1s using if() and for() statements.

Task 6 (solution)

Create the objects `a <- sample(c(4,0),20,replace=TRUE)` and `b <- matrix(a,nrow=4,ncol=5)`. Now recode all the 4s into 1s in `b` using if() and for() statements.

```
a <- sample(c(4,0),20,replace=TRUE)
b <- matrix(a,nrow=4,ncol=5)

for (j in 1:5){
  for (i in 1:4){
    if (b[i,j]==4){
      b[i,j]=1
    }
  }
}
```

Help

- Type help() (`or ?`) to see documentation
- Read Wickham & Grolemund's *R For Data Science* Handbook
- Check out the help function for a couple of functions used in today's course to see 'what is what' in the documentation

Next Session

- More sophisticated objects (dataframes, lists etc.)
- Basic plotting
- Reading and saving data
- Basic text-manipulation with R