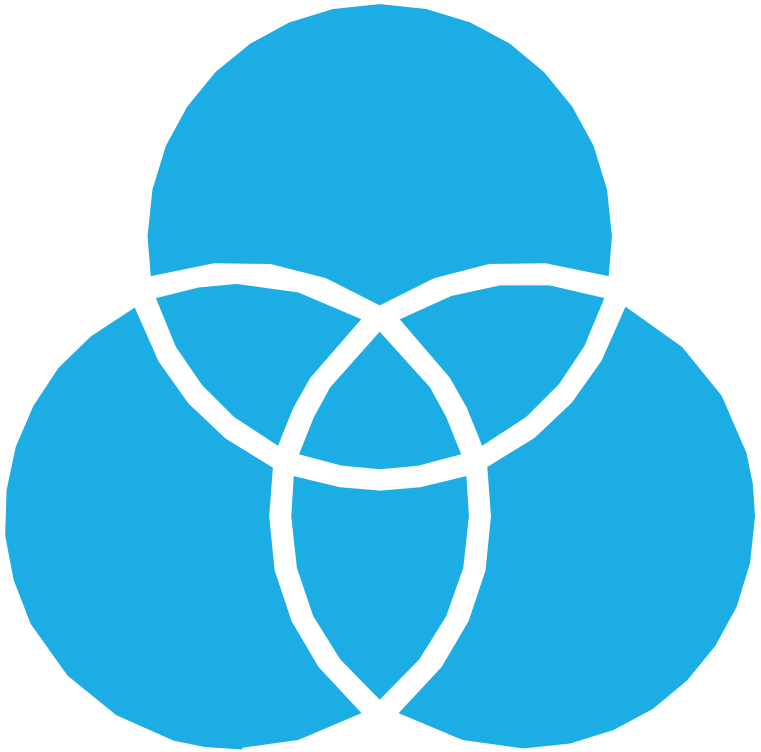


Introduction to R for argument mining

MARTÍN PEREIRA-FARIÑA

UNIVERSITY OF SANTIAGO DE COMPOSTELA

Objectives

- Overview of computational approach to argument mining
- Basic concepts of R
- Some of basic ideas to start with argument mining using R
- Basic skills for a further research

Outline 1st tutorial

- Introduction to argument mining
  - Characterization
  - NLP tools for argument mining
- Introduction to R
  - Data types
  - Programming structures
  - Graphics (Shiny)

Outline 2nd tutorial

- Manipulating texts
- Plotting results (graphs)

1st Tutorial

INTRODUCTION

Outline 1st tutorial

- Introduction to argument mining
  - Characterization
  - NLP tools for argument mining
- Introduction to R
  - Data types
  - Basic commands
  - Graphics (Shiny)

Argument Mining

OVERVIEW

Argument mining

Argument mining aims at automatically extracting structured arguments from unstructured textual documents . Its goal is to provide structured data for computational models of argument and reasoning models (Lippi 2016)

Argument mining

CLAIM 1 While those on the far-right think that immigration threatens national identity, as well as cheapening labor and increasing dependence on welfare.

[...]

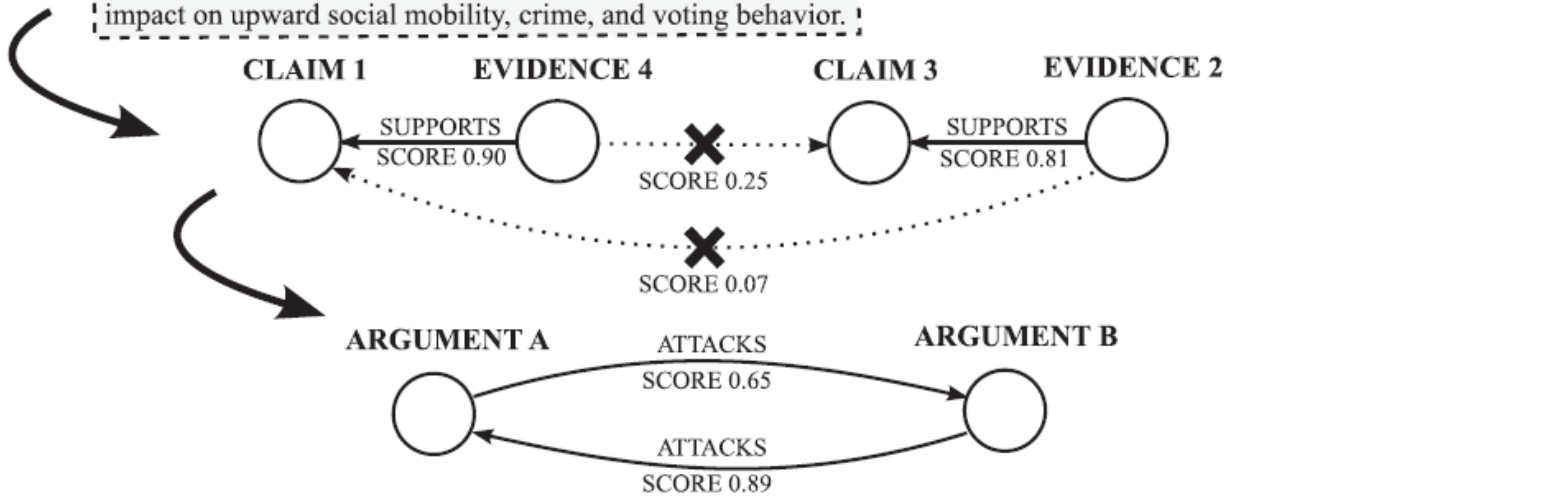
Proponents of immigration maintain that, according to Article 13 of the Universal Declaration of Human Rights, everyone has the right to leave or enter a country, along with movement within it [...] **EVIDENCE 2**

[...]

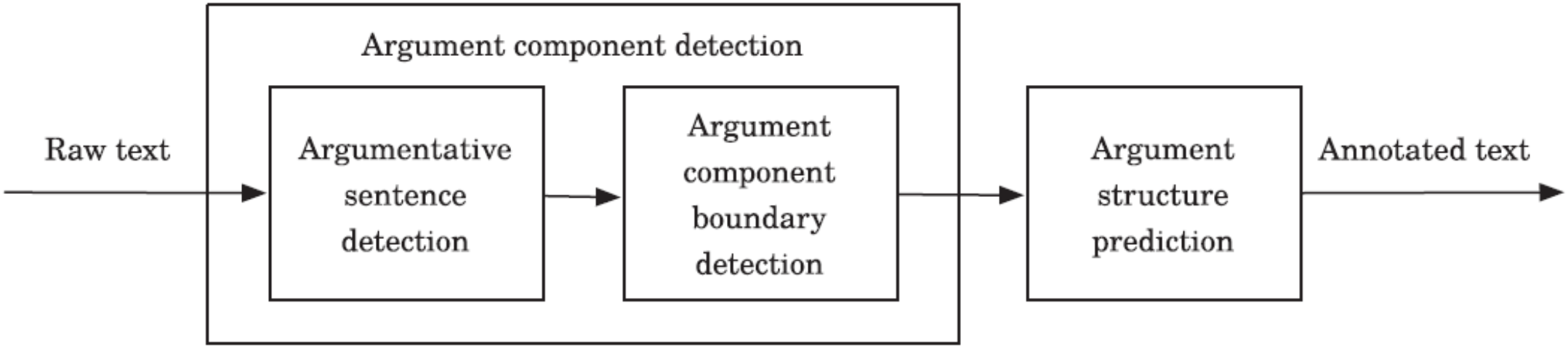
CLAIM 3 Some argue that the freedom of movement both within and between countries is a basic human right, and that the restrictive immigration policies, typical of nation-states, violate this human right of freedom of movement.

[...]

Immigration has been a major source of population growth and cultural change throughout much of the history of Sweden. The economic, social, and political aspects of immigration have caused controversy regarding ethnicity, economic benefits, jobs for non-immigrants, settlement patterns, impact on upward social mobility, crime, and voting behavior. **EVIDENCE 4**



Argument mining



Pipeline architecture of an AM system (Lippi2016)

Computational tools

- Most of NLP tools, libraries, etc. are useful in AM
- Argument components recognition (sentence classification, sentiment analysis, question classification, etc.)
- Argument components boundaries (text segmentation, named entity recognition, etc.)
- Argument structure (link prediction, semantic textual similarity, etc.)

Applications

WHAT

- Social networks (Twitter, Facebook, etc.)
- Comments in newspapers, magazines, etc.
- Acceptance of arguments
- Anomalous behaviour

WHO

- Policy-makers
- Researchers in social and political sciences
- Marketing
- Business

Challenges

- General purpose AM system (specific genre, better results)
- ML techniques require reliable corpora (time-comsuming task)
- Positive and negative examples

Argument mining and R

- Is there an *ArgumentMining* package for R?

Sorry...

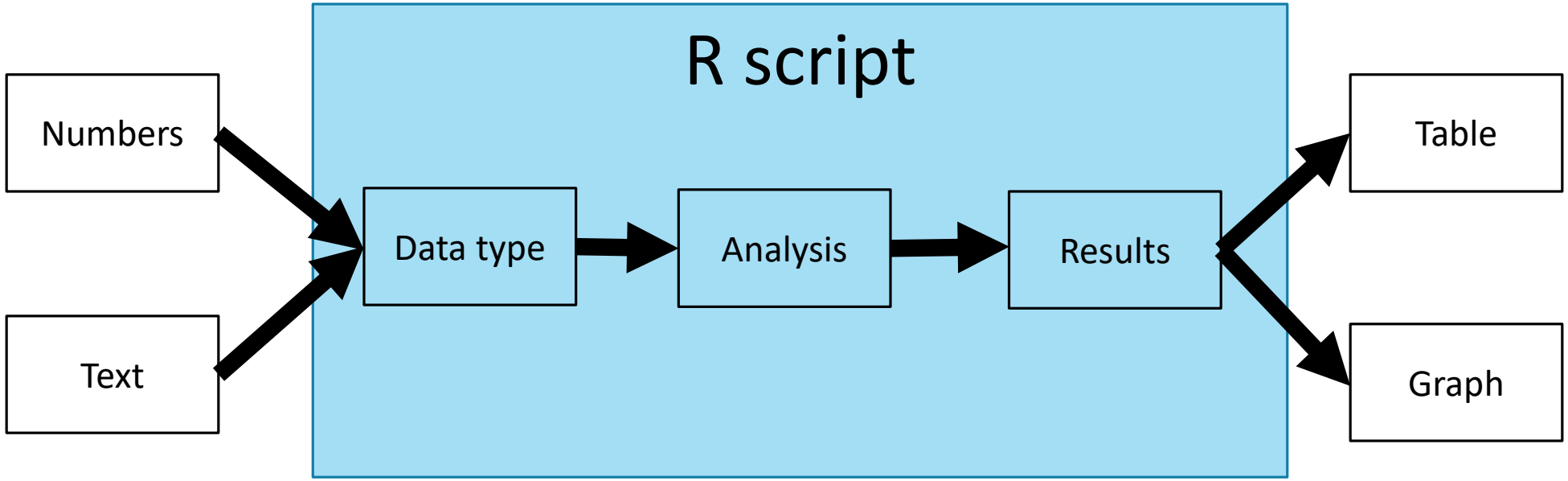
R is not useful for AM

AM has never been explored in R

R

BASIC CONCEPTS IN R

Pipeline in R



Workspace

- Set of variables, files and scripts that are currently loaded in memory
- Basic operator: <-
- How to examine your workspace:
  - `getwp()`: current working directory
  - `ls()`: local worksapce (variables, functions, etc. loaded)
  - `list.files(); dir()`: files in your working directory
  - `dir.create("var_directory")`: create a new directory in the current one
  - `setwd("directory_name")`: change your working directory
  - `rm(list=ls()):` clean your working space

Data types

- Vectors
- Lists
- Matrices
- Arrays
- Data frames

- Factors and tables

Vectors

Nº dimensions	1
Elements mode	1
Creating	v <- c(“a”,”b”) | v <- c(1:10), etc.
Deleting	rm(v)
Index	v[pos]| which(v==val)
Adding elements	v[pos] <- element
Size	length(v)
NA and NULL	v[pos] <- NULL | etc.
Arithmetic op.	Yes
Logical op.	Yes
Sort/Order	Yes
Components	-
Values	unique
Functions	vectorization, filtering, recycling,

Lists

Nº dimensions	1
Elements mode	>1
Creating	lst <- list(“a”,1); etc.
Deleting	rm(lst)
Index	lst$tag | l[[“tag”]] | lst[[i]]
Adding elements	lst$tag <- element | lst[[i]] <- element
Size	length(lst)
NA and NULL	lst$tag <- NULL | etc.
Arithmetic op.	No
Logical op.	No
Sort/Order	No
Components	names(lst)
Values	unlist(lst)
Functions	lapply(), sapply()

Matrices

Nº dimensions	2
Elements mode	1
Creating	mx <- matrix(c(1,2,3,4),nrow=2,ncol=2)
Deleting	rm(mx)
Index	mx[r,c]
Adding elements	mx[r,c] <- value
Size	length(mx)
NA and NULL	mx[r,c] <- NA (not NULL)
Arithmetic op.	Matrix multiplication, scalar multiplication, etc.
Logical op.	Numerical
Sort/Order	Yes
Components	Unique
Values	index, submatrix[i:I,j:j]
Functions	Filtering, apply(),tapply(),lapply(),rbind()

Data Frames

Nº dimensions	2
Elements mode	>1
Creating	df <- data.frame(cmpnt1, cmpnt2, …., stringsAsFactors=FALSE)
Deleting	rm(df)
Index	df$cmptname | df[[i]] | df[r,c]
Adding elements	df[r,c] <- value | df[r,c]
Size	length(d)
NA and NULL	df[r,c] <- NA || df[,c] <- NULL
Arithmetic op.	Per columns
Logical op.	complete.cases(df)
Sort/Order	Per columns or per rows
Components	Yes
Values	comptname | df[i:I,j:j]
Functions	subset(df,comptname condition) | rbind(), cbind() | apply() |merge()

Data types in R

	Vectors	Lists	Matrices	Data frames
Nº dimensions	1	1	2	2
Elements mode	1	>1	1	>1
Creating	x <- c(1,2)	l <- list(a,1)	m <- matrix()	d <- data.frame()
Deleting	rm(x)	rm(l)	rm(m)	rm(d)
Index	x[i]	lst$tag	m[i,j]	d$cn | df[[i]] | df[r,c]
Adding elements	Yes	Yes	Yes	df[r,c] <- value | df[r,c]
Size	Yes	Yes	Cells	Columns
NA and NULL	Yes	Yes	NA	df[r,c] <- NA || df[,c] <- NULL
Arithmetic op.	Yes (N)	Comp	R or C	C
Logical op.	Yes (N)	Comp	R or C	C
Sort/Order	Yes	Comp	R or C	C
Components	No	Yes	Yes	Yes
Values	Yes	Yes	Yes	comptname | df[i:I,j:j]
Functions	Vectoring Filtering Recylcing	sapply() lapply()	Reassigned *apply Filtering, etc.	subset(d,comptname condition) | rbind(), cbind() | apply() | merge ()

Basic commands

Input

- Hardware reading (keyboard, monitor)
  - `scan("", args)`: vector
  - `readline()`: vector (optional prompt "")
- File reading
  - `scan("filename")`: vector
  - `scan("url")`: vector
  - `matrix(scan("filename"), nrow or ncol, byrow=TRUE)`
    - `as.matrix`: Convert a data frame into a matrix.
  - `read.table("filename")`: data frame

Output

- Hardware utterance
  - `print()`: monitor
  - `cat() (+ “\n”)`: monitor
- File utterance
  - `write.table()`: data frame
  - `write.table(mx,”filename”,row.names=FALSE,col.names=FALSE)`: matrix
  - `cat(“strings\n”,file=“filename”)`: “strings”
  - `writelines(“strings”,filename)`: “strings”
  - `Filename <- file(“filename”,”w”)`: file

Function

```
namefunction <- function(args) {

  Body function

  }

return(var)
```

Function

```
> cadrado <- function(v){
+   y <- v^2
+   return(y)
+ }
> cadrado(1:10)
 [1]   1   4   9  16  25  36  49  64  81 100
>
```

```
> cadrado(y)
  [1] 1.400211e-01 7.999722e-01 9.639619e-01 1.101700e-01 1.551626e-02 3.825452e-02
  [7] 6.854039e-02 5.196404e-01 1.547064e+00 2.385762e-01 2.403668e+00 2.894005e-01
 [13] 2.129110e+00 2.090521e+00 2.717104e-01 1.122782e+00 1.984661e-01 6.087751e-03
 [19] 1.052829e-01 1.095891e-02 1.138771e+00 2.206064e+00 3.198791e-01 3.379136e-02
 [25] 3.022086e-01 3.388271e-01 6.904811e-02 2.249201e+00 9.700885e-02 2.548893e-01
 [31] 8.499868e-01 2.316932e-02 2.535164e-03 9.268705e-01 4.214751e-02 4.776246e-01
 [37] 1.182810e-01 1.995207e-01 7.987232e-01 5.304012e-01 1.913144e-02 2.887825e+00
 [43] 1.538577e+00 1.494791e-01 9.159588e-01 2.532318e-03 2.824475e+00 5.389741e-01
 [49] 6.517890e-03 5.030677e-01 8.928415e-01 1.252612e+00 1.900589e-01 2.902465e+00
 [55] 4.228062e+00 2.511350e-01 6.597182e-01 3.969188e-01 2.466357e-01 6.797182e-01
 [61] 2.184299e-01 2.647901e-01 2.677018e-02 1.245127e-01 4.094858e-01 2.643801e-01
 [67] 2.018846e+00 1.240744e+00 4.295306e+00 2.031951e+00 2.499433e-01 4.048556e+00
 [73] 7.641879e-02 1.300841e-01 1.806969e+00 2.325225e+00 1.236218e-01 3.712992e-01
 [79] 5.320495e-01 7.274536e-02 9.509901e-01 4.096071e+00 4.691383e-02 3.587440e-01
 [85] 1.455035e+00 3.356415e-01 1.658455e+00 3.310555e-01 3.684572e-01 4.673195e+00
 [91] 2.304575e+00 1.740344e-02 5.943591e-05 1.075280e-02 3.577305e+00 6.171586e-01
 [97] 5.763971e-03 2.200073e-02 6.147735e-02 1.097276e-02
>
```

Basic programming structures: For

```
for (n in x) {

    body

    }
```

```
> for (i in 1:length(x)) {
+     print(paste("Value ", i, ": ", x[i], sep=""))}
[1] "Value 1: 0.20804881984974"
[1] "Value 2: 0.648721242633039"
[1] "Value 3: 0.702074483594297"
[1] "Value 4: -0.752672749925293"
[1] "Value 5: 1.1414878797976"
[1] "Value 6: 1.88162273174325"
[1] "Value 7: 1.63655394287066"
[1] "Value 8: 0.130516140744042"
[1] "Value 9: -0.516950142529137"
[1] "Value 10: -1.68226604753219"
>
```

Basic programming structures: While

```
while (i <= 10) i <-
i+4
while(TRUE) {
        expressions
        break
        }
Repeat {
        expressions
        break
        }
```

```
> countdown <- function(from)
+ {
+     print(from)
+     while(from!=0)
+     {
+         Sys.sleep(1)
+         from <- from - 1
+         print(from)
+     }
+ }
>
> countdown(5)
[1] 5
[1] 4
[1] 3
[1] 2
[1] 1
[1] 0
```

Basic programming structures: If-else

```
If (condition) {

  Expression1

} else {

  Expression2

}
```

```
> x <- 5
> if(x > 0){
+   print("Positive number")
+ } else {
+   print("Negative number")
+ }
[1] "Positive number"
>
```

Regular expressions

Regular Expressions

- Regular expression (regex) is a pattern that describes a set of strings.
- They can be used for retrieving and editing set of strings.

- How to use regular expressions:
  - Predefined in R
  - Functionally defined

Regular Expressions (retrieving)

`grep`	Performing regular expressions pattern matching
`pearl = {TRUE, FALSE}`	FALSE: extended regular expressions (default) TRUE: Perl-like regular expressions
`value = {TRUE, FALSE}`	FALSE: it returns the index position. TRUE: it returns the value of the position.
`grepl`	Similar to grep, but value arg is not supported.
`regexpr`	Similar to grep, but returns an integer vector with the same length as the input vector (value arg is not supported)

Regular Expressions (retrieving)

`gregexpr`	It is the same as regexpr, except that it finds all matches in each string. It returns a list
`regmatches`	We get the actual substrings matched by the regular expression. It returns a vector or list depending on the function of the input vector.

Regular Expressions (editing)

sub sub (regex, replacement text, vector)	It returns a new vector with the same length as the input vector.
\1 - \9	Combination with backreferences (numbered regex).
gsub	It is very similar to sub.

Character sequences

anchor	interpretation
`.`	any character except a newline
`\\w`	any word character
`\\W`	anything but a word character
`\\d`	any digit character
`\\D`	anything but a digit character
`\\b`	a word boundary
`\\B`	anything but a word boundary
`\\s`	any space character
`\\S`	anything but a space character

(Desagulier 2017)

Splitting in single words and erasing punctuation

“Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris iaculis mauris ac euismod elementum. Duis mollis, sem eu gravida rutrum, est ligula suscipit elit, vehicula finibus arcu enim non justo. Nullam pretium lacus sed sapien viverra, ac venenatis urna ultrices. Phasellus lacus ipsum, ultricies quis nulla sit amet, aliquet mollis dolor. Phasellus accumsan gravida feugiat. Quisque convallis tellus at erat faucibus aliquet. Proin a maximus nibh, posuere suscipit ex. Integer erat dui, tristique et elit eu, consequat mollis purus. Nullam scelerisque semper lorem nec vehicula. Duis non ex vel ex consectetur ullamcorper. Integer vehicula felis sed dignissim placerat.”

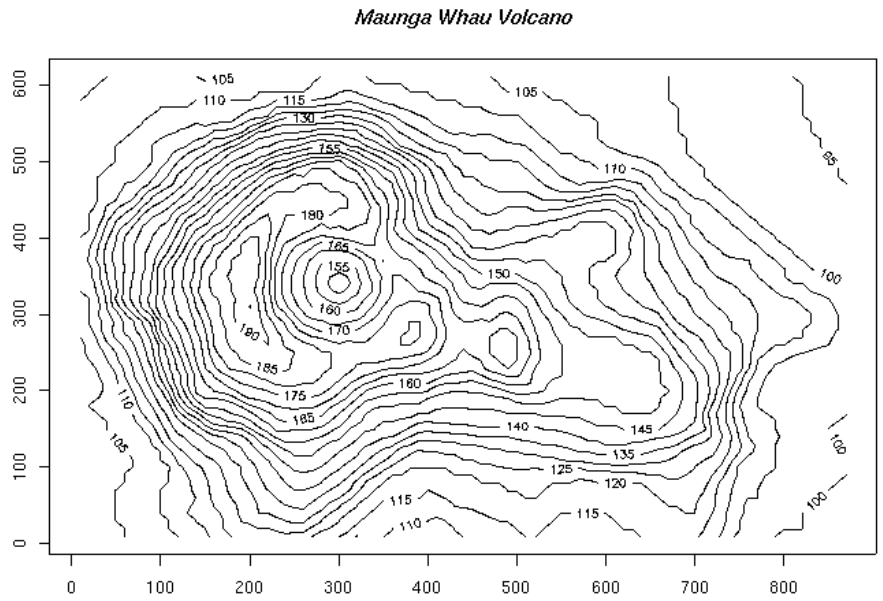
```
 [1] "lorem"       "ipsum"       "dolor"       "sit"         "amet"        "consectetur" "adipiscing" "elit"
 [9] "mauris"      "iaculis"     "mauris"      "ac"          "euismod"     "elementum"   "duis"       "mollis"
[17] "sem"         "eu"          "gravida"     "rutrum"      "est"         "ligula"      "suscipit"   "elit"
[25] "vehicula"    "finibus"     "arcu"        "enim"        "non"         "justo"       "nullam"     "pretium"
[33] "lacus"       "sed"         "sapien"      "viverra"     "ac"          "venenatis"   "urna"       "ultrices"
[41] "phasellus"   "lacus"       "ipsum"       "ultricies"   "quis"        "nulla"       "sit"        "amet"
[49] "aliquet"     "mollis"      "dolor"       "phasellus"   "accumsan"    "gravida"     "feugiat"    "quisque"
[57] "convallis"   "tellus"      "at"          "erat"        "faucibus"    "aliquet"     "proin"      "a"
[65] "maximus"     "nibh"        "posuere"     "suscipit"    "ex"          "integer"     "erat"       "dui"
[73] "tristique"   "et"          "elit"        "eu"          "consequat"   "mollis"      "purus"      "nullam"
[81] "scelerisque" "semper"      "lorem"       "nec"         "vehicula"    "duis"        "non"        "ex"
[89] "vel"         "ex"          "consectetur" "ullamcorper" "integer"     "vehicula"    "felis"      "sed"
[97] "dignissim"   "placerat"
```

```
lorem.v <- scan("lorem.txt", what="character", sep="\n")

lorem.v <- tolower(lorem.v)

lorem.words.l <-strsplit(lorem.v, "\\W")
lorem.words.v <- unlist(lorem.words.l)
blank.v <- which(lorem.words.v=="")
lorem.words.v <- lorem.words.v[-blank.v]
```

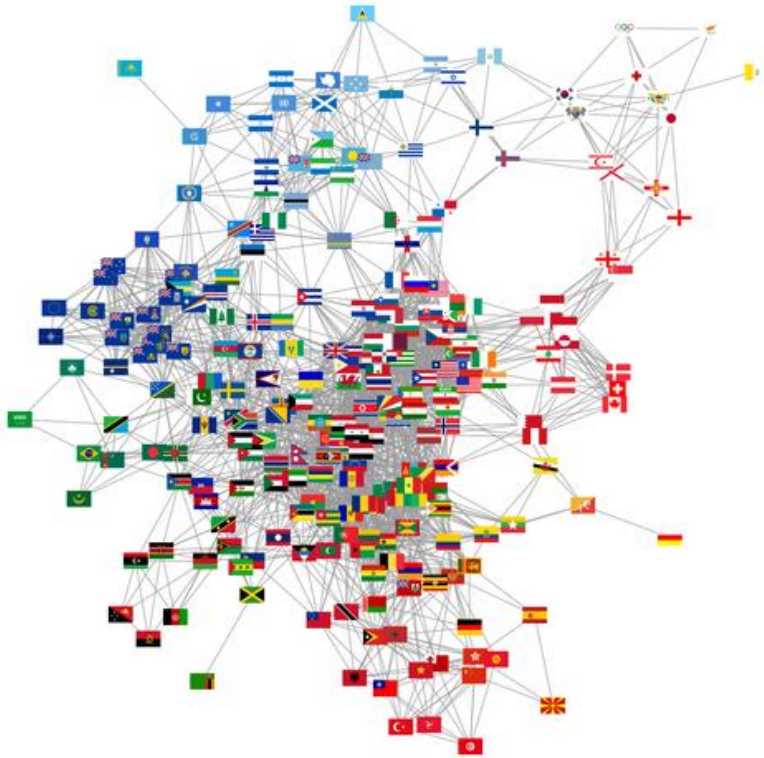
Graphics

Graphics

This is one of the strengths of R.



https://www.stat.auckland.ac.nz/~paul/Talks/gridSVG/slide5.html



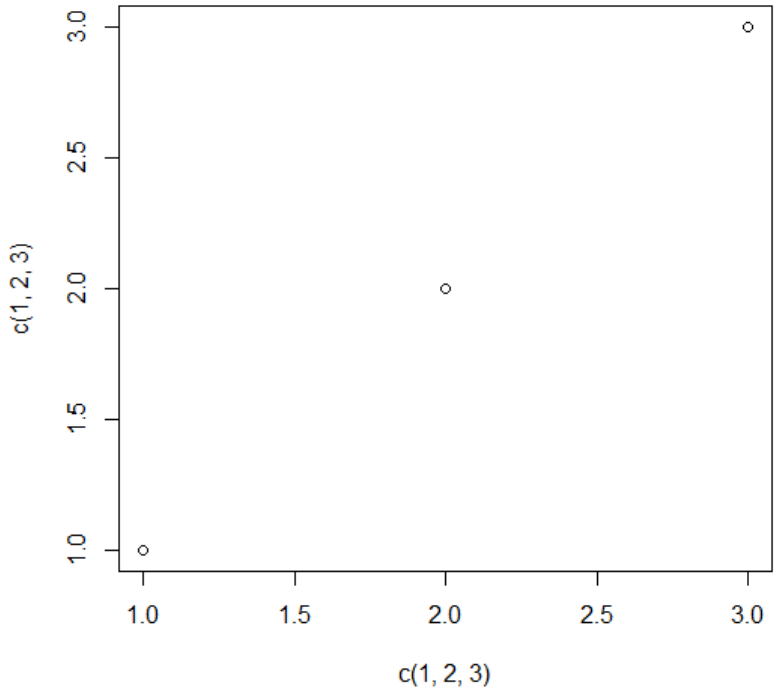
http://revolution-computing.typepad.com/.a/6a010534b1db25970b017c34dd6f91970b-pi

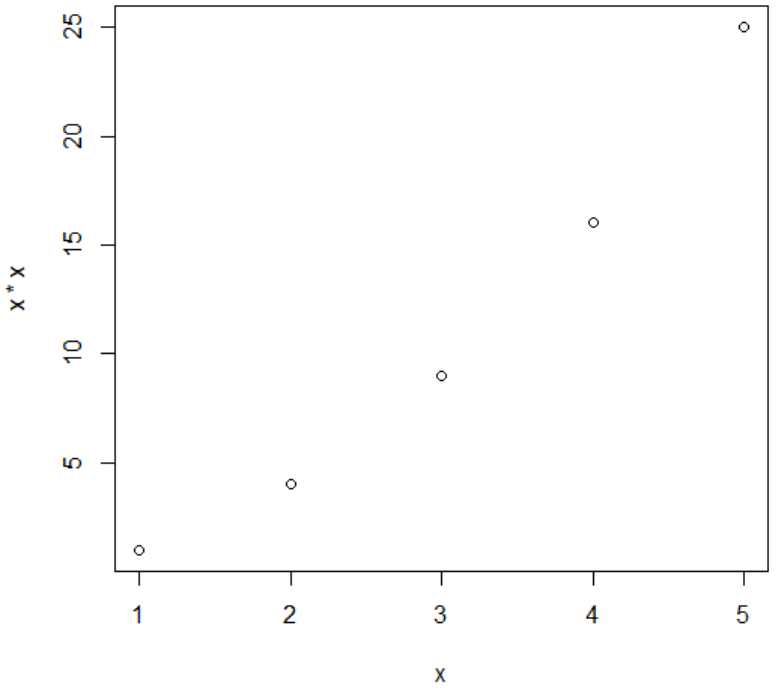
Graphics

Function	Type of plot
stripchat(args)	It plots the data in order along a line with each data point represented as a box.
plot(args) barplot(args)	It provides a graphical view of the relationship between two sets of numbers.
boxplot(args)	It provides a graphical view of the median, quartiles, maximum, and minimum of a data set
hist(args)	It plots the frequencies that data appears within certain ranges

plot()

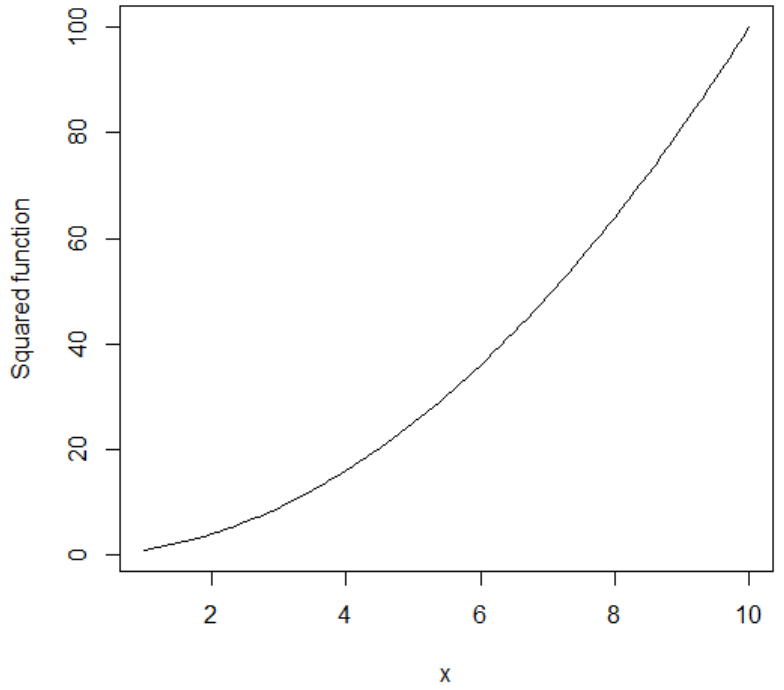
Basic function: plot(x_vector,y_vector)



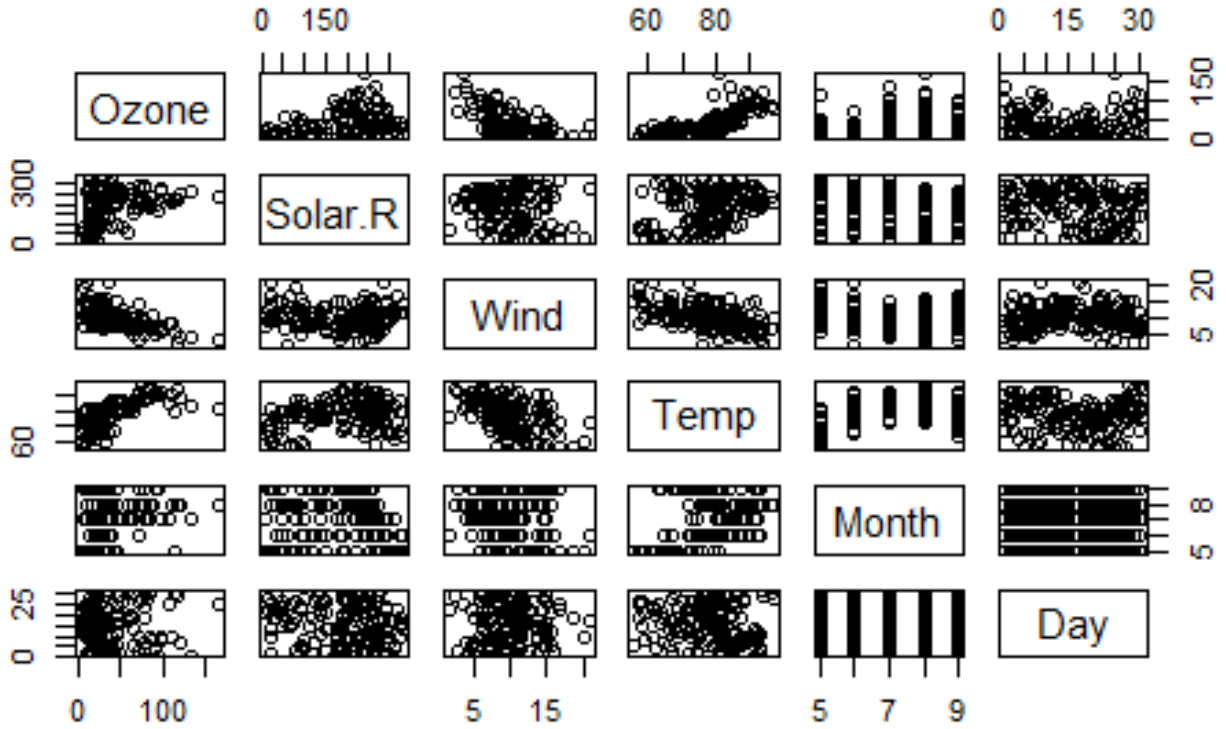


plot()

```
> x <- seq(1,10, by=0.1)
> plot(x,x*x, type="l", ylab="Squared function")
>
```



plot()

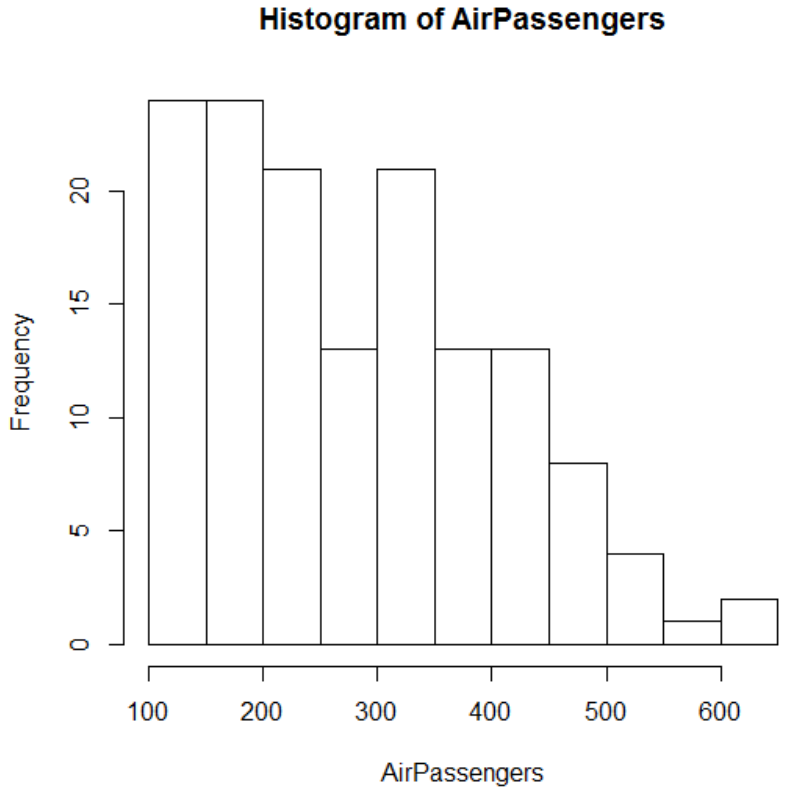

Ozone
Solar.R
Wind
Temp
Month
Day
0 150
60 80
0 15 30
0 300
0 150
5 20
60
5 8
0 25
0 100
5 15
5 7 9

Customizing

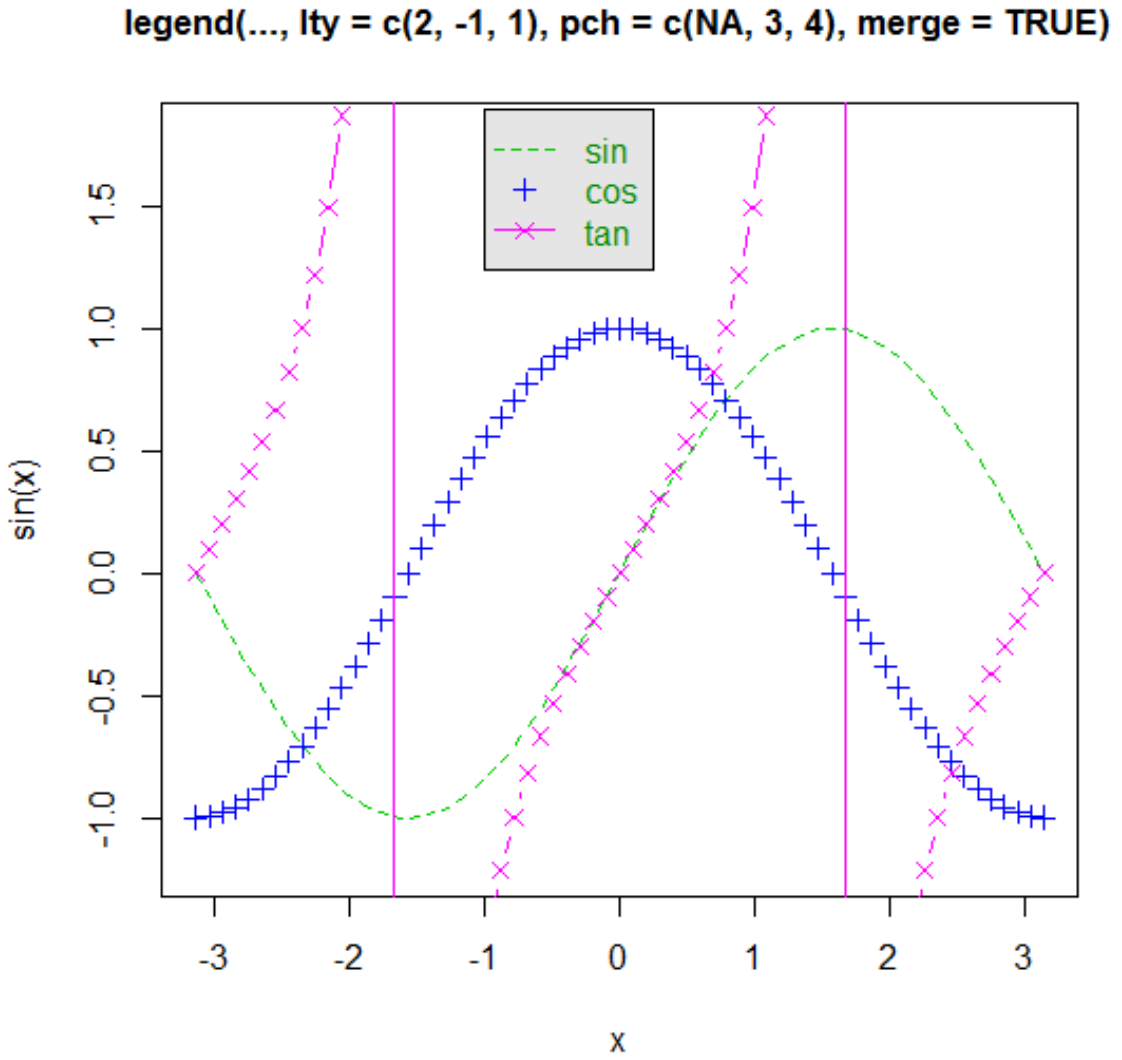
- Most of them are parameters (arguments).
- Font attributes: cex = n times normal size [text() argument].
- Colour
- Labeling axes, points, etc.
- Type of line: `lty`
- Range of Axes: `xlim=c(lb,ub); ylim=c(lb,ub) [plot() argument]`
- Adding polygons: `polygon(c(xbl,xbr,xtr,xtl), c(ybl,ybr,ytl,ytr), args.)`

hist()

```
> AirPassengers
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1949 112 118 132 129 121 135 148 148 136 119 104 118
1950 115 126 141 135 125 149 170 170 158 133 114 140
1951 145 150 178 163 172 178 199 199 184 162 146 166
1952 171 180 193 181 183 218 230 242 209 191 172 194
1953 196 196 236 235 229 243 264 272 237 211 180 201
1954 204 188 235 227 234 264 302 293 259 229 203 229
1955 242 233 267 269 270 315 364 347 312 274 237 278
1956 284 277 317 313 318 374 413 405 355 306 271 306
1957 315 301 356 348 355 422 465 467 404 347 305 336
1958 340 318 362 348 363 435 491 505 404 359 310 337
1959 360 342 406 396 420 472 548 559 463 407 362 405
1960 417 391 419 461 472 535 622 606 508 461 390 432
```



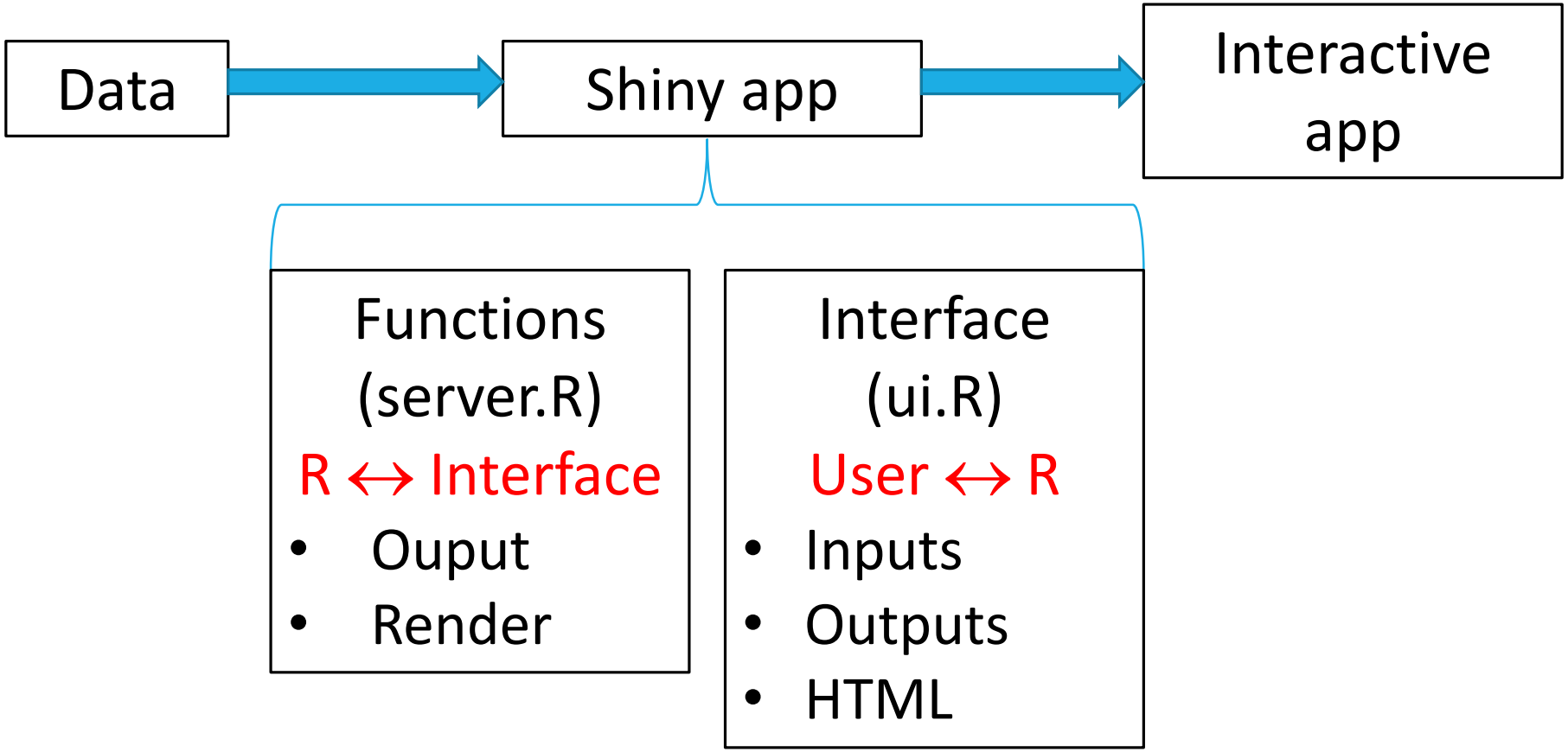
legend()


legend(..., lty = c(2, -1, 1), pch = c(NA, 3, 4), merge = TRUE)
sin
cos
tan
sin(x)
1.5
1.0
0.5
0.0
-0.5
-1.0
-3
-2
-1
0
1
2
3
x

Shiny

- “Web application framework for R”.
- Rstudio.
- Making your data analysis interactive: the user can manage the data through an interactive way.
- Different input values → output automatically updated.
- https://shiny.rstudio.com/gallery/faithful.html

Shiny architecture



From 0.10.2: Single-file applications (app.R)

Tutorials of Shiny

https://shiny.rstudio.com/

References

- Lippi, Marco, y Paolo Torroni. «Argumentation Mining: State of the Art and Emerging Trends». ACM Transactions on Internet Technology 16, n.º 2 (30 de marzo de 2016): 1-25.
- Arnold, Taylor. Humanities Data in R. New York, NY: Springer Science+Business Media, 2015.
- Desagulier, Guillaume. Corpus Linguistics and Statistics with R. Quantitative Methods in the Humanities and Social Sciences. Cham: Springer International Publishing, 2017.
- Jockers, Matthew L. Text Analysis with R for Students of Literature. New York: Springer, 2014.
- https://www.tidytextmining.com

2nd Tutorial

EXERCISES

Outline

- Manipulationg strings in R
- Reading a text file
- Cleaning the data
- Tokenization
- Visualization
- Network
- Drawing a graph

Manipulating strings in R

- Reading a text file
- Cleaning your data (removing metadata, punctuation, capital letters, stop-words, etc.)
- How to create a corpus of words

Reading a text file

- Read the file “melville.text” into a vector and use line breaks to separate the components of the vector.

- Function: Scan
- New line indicador: “\n”
- Be aware about the codification (latin1, UTF-8; etc.)

Cleaning your text file

- Removing metadata
- Table of contents, title of the chapter, etc.

Tokenization

- Necessary step for POS-Tagging
- Several different libraries
- Very good results (>96%)

- Sentences
- Words

- Be aware about the ouput of each function

Graph

- A graph consists of vertices (nodes) and edges (links).
- Circle -> node
- Line -> Edge
- Package: igraph
- Example undirect graph:

```
g <- graph.formula(1--2, 1--3, 1--4, 2--5, 2--6, 2--7, 3--8, 3--9, 4--10)
```

- Example direct graph:

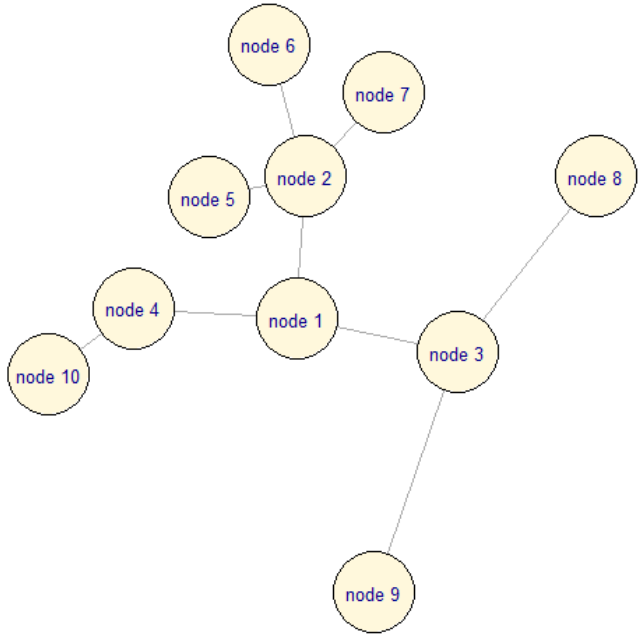
```
g.d <- graph.formula(1-+2, 1-+3, 1-+4, 2-+5, 2-+6, 2-+7, 3-+8, 3-+9, 4-+10)
```

Graph

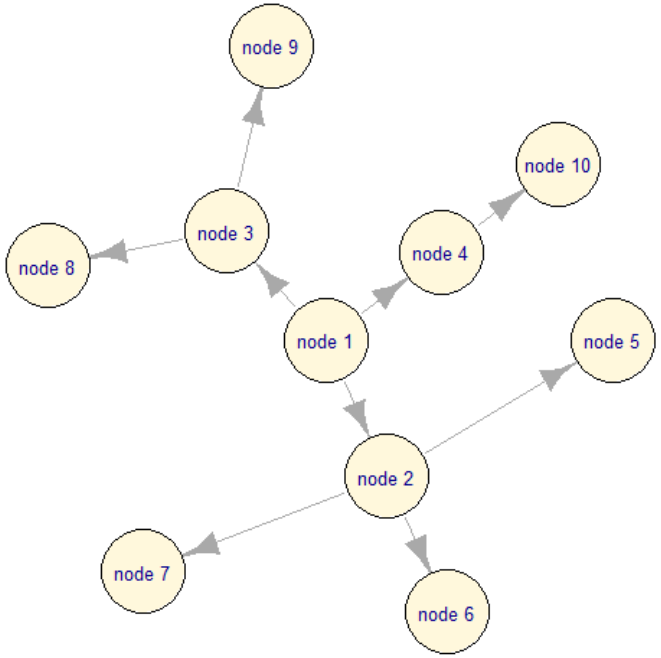
- Generating the visualization of a graph
- Specifying the plotting parameters
  - `igraph.options(vertex.size=30, vertex.color="cornsilk", vertex.label=labels, vertex.label.cex=0.8, vertex.label.family="Helvetica", edge.width=0.5)`
- Plotting the graph
  - `set.seed(42)`
  - `plot(g, layout=layout.reingold.tilford(g, circular=T))`

Graph

UNDIRECT GRAPH



DIRECT GRAPH



Graph

- Argument representation
- `graph.data.frame`: it creates an igraph graph from one or two data frames containing the (symbolic) edge list and edge/vertex attributes.
- Frequency
- Graph metrics: centrality, eigenvector centrality, etc.