EC999: Introduction

Thiemo Fetzer

University of Chicago & University of Warwick

March 30, 2017

Some Organizatorial Issues

1. Housekeeping: Visiting Prof at UChicago, usually based at University of Warwick in the UK. You can reach me on `tfetzer@uchicago.edu`.
2. About this course: this is still a relatively new course and you guys are the second cohort of students:
   - Computational Linguistics
   - Machine Learning
   - Data science
   - Statistics
   - Programming in R
3. Your input is needed: too slow, too fast, ...? just email me.
4. Assessment: I be based on 5 course assignments and a class project presented in week 10.
5. Course Material: `http://www.trfetzer.com/ec999/`

Plan for the Course - Part 1

First part will introduce simple NLP concepts, introduce the R language and illustrate how we can work with getting text into a useful format for subsequent analysis. The unit of analysis in the first part of the course will mainly be (short) sequences of words.

1. Introduction: Motivation, setting up, examples, basic introduction to "R".
2. Sourcing data: String manipulation, regular expressions, loading text data, basic web scraping, (social media) API's.
3. Text Normalization: Zipf's law, Herd's law, tokenization methods and routines, stemming, Levenshtein distance.
4. Describing Text: readability measures
5. Identifying and extracting collocations: heuristic and statistical methods.
6. Part-of-Speech Tagging.
7. N-Gram generative language model.
8. ...

Plan for the Course - Part 2

In the second part, we will turn (larger) documents into a vector-space representation and perform (supervised) machine learning for classification type purposes.

1. Vector Space model: vector space representation, bag of words model, measuring distance/ similarity between texts.
2. Ideological scaling: introduction to supervised learning concepts; naive LBK ideology scaling, Bayesscore.
3. Classification introduction: classification error types, measuring accuracy and bias-variance trade-off
4. kNN classification
5. Naive Bayes: Bernoulli versus Multi-nomial language models
6. Logistic regression: best subset selection, feature selection through regularized logistic regression
7. Application example: sentiment analysis

Plan for the Course - Part 3

In the third part, we will turn to unsupervised learning methods for textual data.

1. Unsupervised learning
2. K-Means clustering: k-medoids (PAM), importance of distance measures
3. Hierarchical clustering: different linkage
4. Topic Modelling: static and dynamic

Organization

- You will be asked to try to reproduce what is done on the lectures. Most slides are written using Markdown, which means the R-Code is executed on the fly behind the scenes.
- Lectures, including sample source files, are made available on `http://www.trfetzer.com/EC999`
- Questions can be asked on `https://piazza.com/uchicago/spring2017/ph30570/`
- Homework assignments will ask you to perform some analysis and coding work. You are asked to produce an Markdown document (more later) that solves the assignment you were asked to do.
- Course project will involve own data analysis in groups
- **Teaching Assistant**: Alvaro Valdes Mena, avaldes@uchicago.edu - will be available for appointments.

Plan

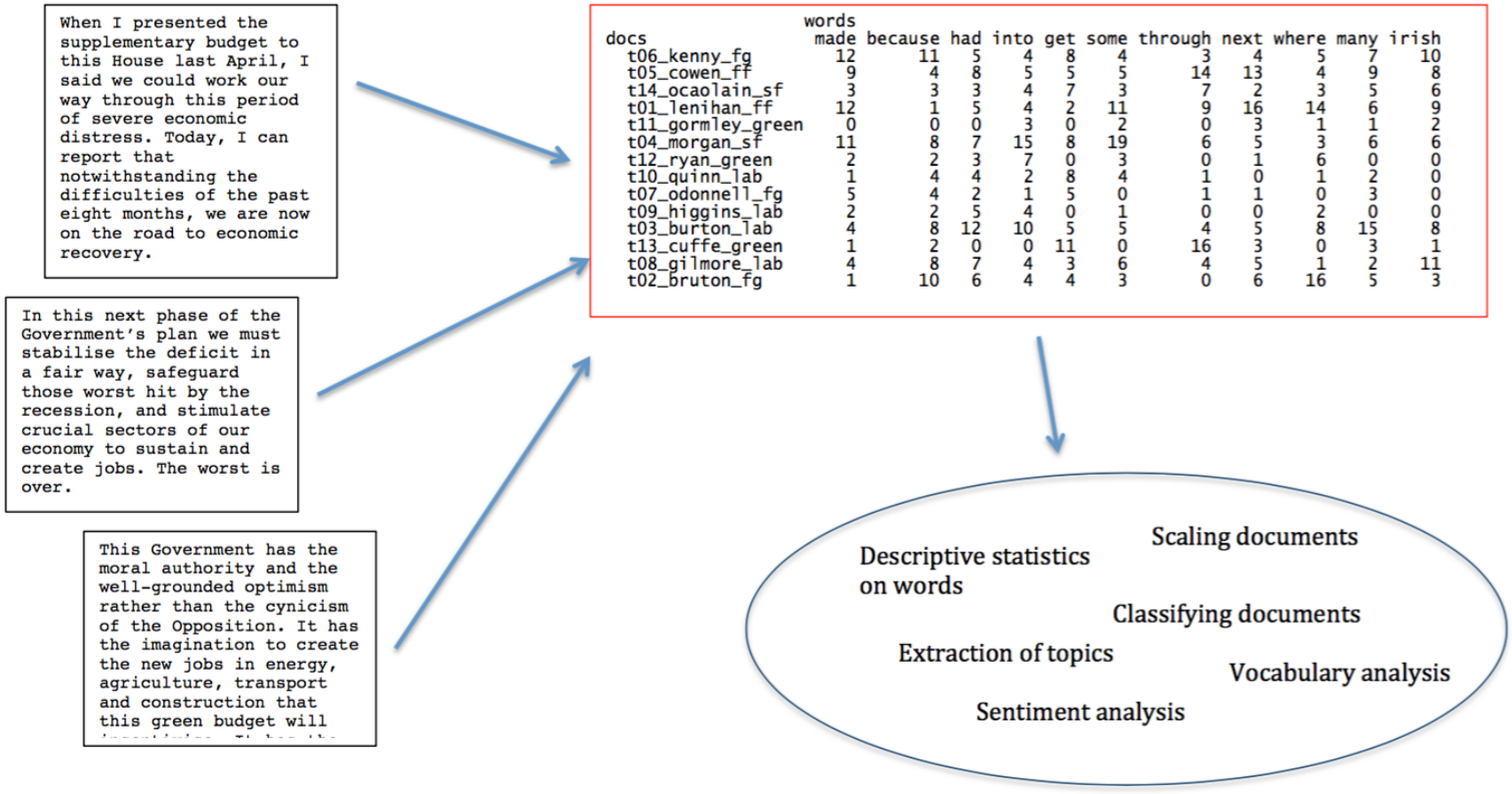
Introduction and Terminology

Examples from Research

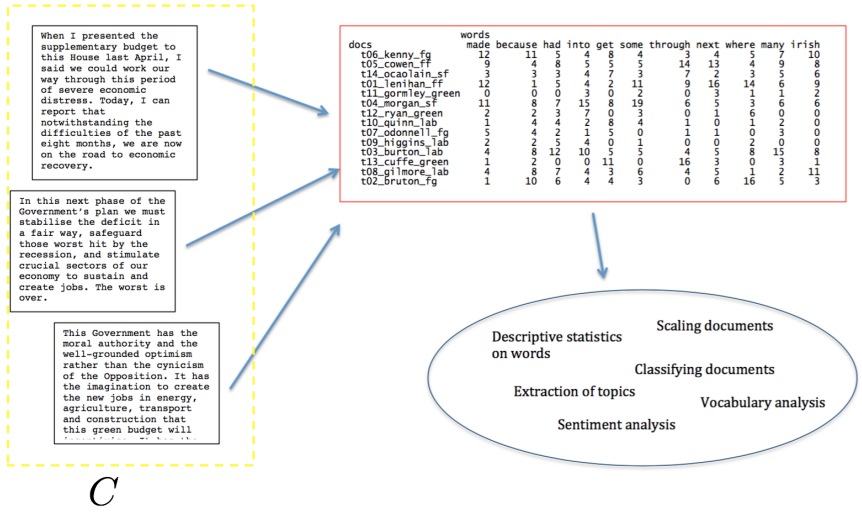
An Introduction to R

Quantitative Text Analysis as process

Above all, there needs to be a formulatedresearch question or **goal** to be achieved.



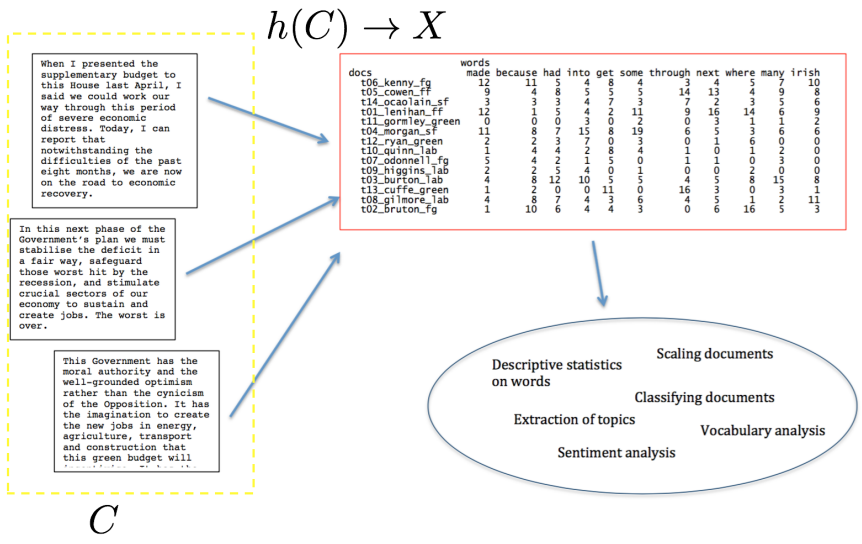
Step 1: Corpus Definition



First step consists of definition of *corpus*

1. Selecting texts: definition of corpus of texts and being aware of underlying selection criteria.
2. Conversion of texts into a common electronic format
3. Defining documents: what will be the unit of analysis
   - Sentence, Paragraph, Section, Chapter,...

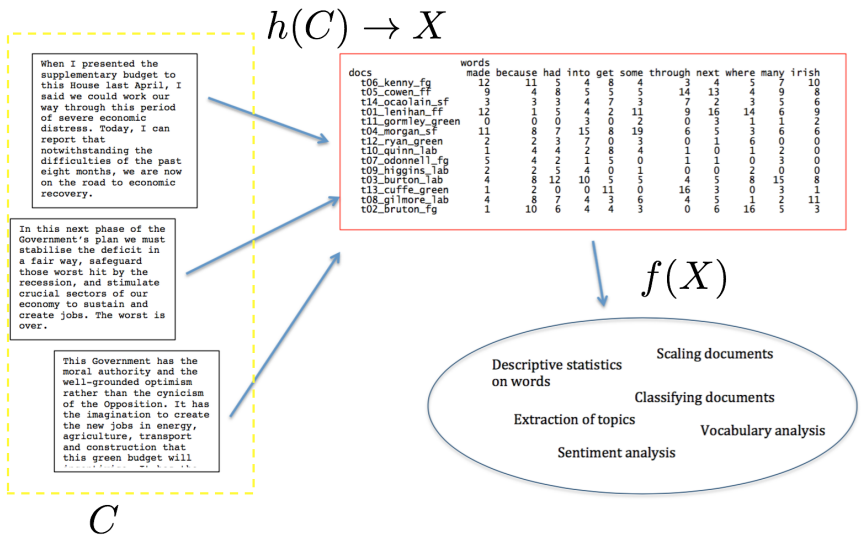
Step 2: Feature Definition and Selection



Second step typically consists of definition of features

1. This can be word tokens, *n-grams*, keywords, indicators, patterns: challenge is to identify *meaningful* features
2. For example: classification tasks you want to identify words that are characteristic and informative about underlying classes.
3. *Curse of dimensionality*: Feature selection often required/desirable.
4. Convert corpus-features into a numerically accessible data matrix X

Step 3: Vector space representation & analysis



Third step involves analyzing data repesentation to achieve a give task.

1. Classification (=Prediction): estimating a function $\hat{f}$ that maps individual documents x_i into specific classes (e.g. republican versus democrat, sentiment label,...)
2. Clustering: identify groups of documents x_i that are similar because they belong to similar latent class C, without knowing the set of classes *ex ante*.

Some Terminology

(text) corpus usually a large set of documents that have limited structure.

tokens any word so token count is total words

stems word stems with suffixes removed

lemmas cannonical word forms: the base form of a word that has the same meaning even when different suffixes (or prefixes) are attached.

"key" words words selected because of special attributes, meanings, or rates of occurrence

stopwords words that are excluded from text analysis for a range of reasons (typically lack of informational content)

complexity word complexity usually measures the number of syllables

diversity (lexical diversity) A measure of how many words occur per fixed word rate (a normalized vocabulary measure)

Plan

Introduction and Terminology

Examples from Research

An Introduction to R

Measuring Political Slant

Reseach Question Are newspapers maximizing profits by pandering to the ideology of their readership?

Corpus Republican and democratic speeches, newspaper front page content

Features Distincively Republican and Democratic n-grams

Analysis *Slant measure* of front page newspaper coverage picking up democratic vs. republican features.

See: Gentzkow, M., & Shapiro, J. M. (2010). What Drives Media Slant? Evidence From U.S. Daily Newspapers. Econometrica, 78(1), 3571.

Measuring Political Slant

Panel A: Phrases used more often by Democrats

Two-word phrases

private accounts	rosa parks	workers rights
trade agreement	president budget	poor people
american people	republican party	republican leader
tax breaks	change the rules	arctic refuge
trade deficit	minimum wage	cut funding
oil companies	budget deficit	american workers
credit card	republican senators	living in poverty
nuclear option	privatization plan	senate republicans
war in iraq	wildlife refuge	fuel efficiency
middle class	card companies	national wildlife

Identify n-grams (*collocations*) and extract those that are distinctively more likely to appear in the corpus of republican versus democratic speaches congressional speeches

Measuring Political Slant

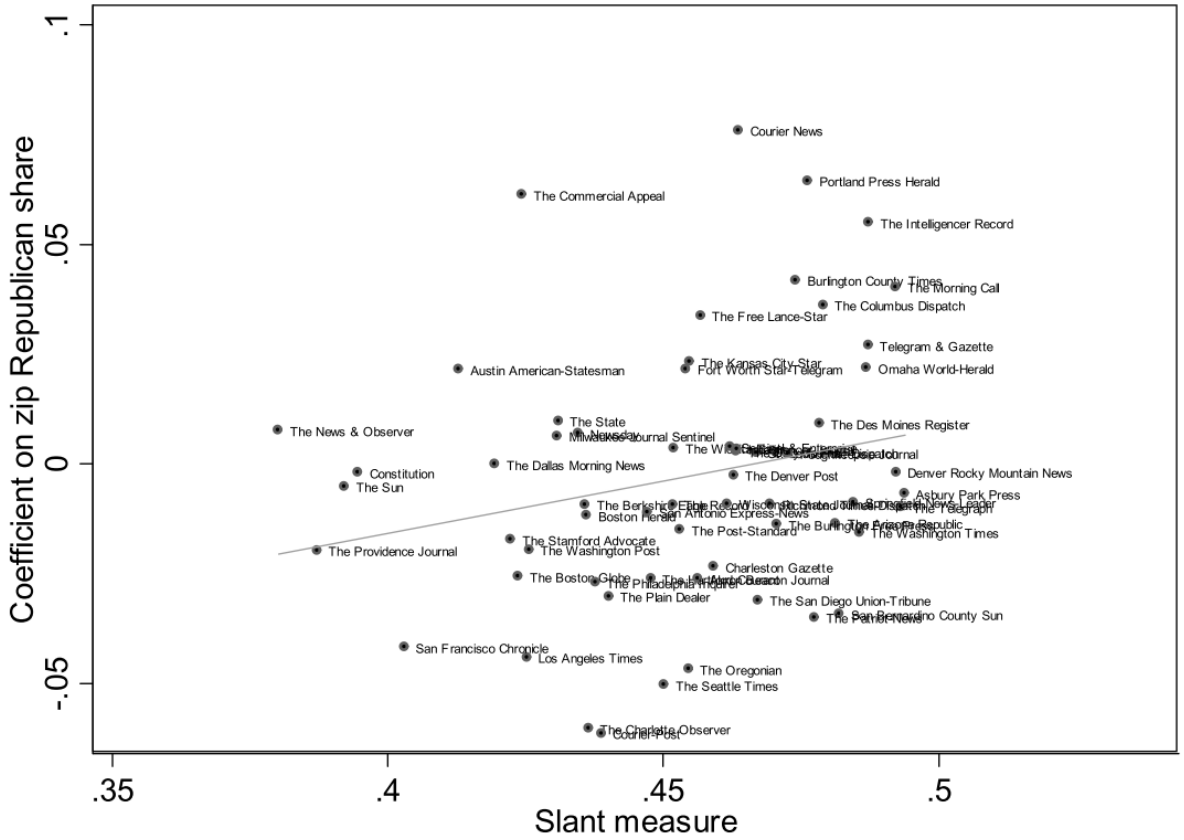
Panel B: Phrases used more often by Republicans

Two-word phrases

stem cell	personal accounts	retirement accounts
natural gas	saddam hussein	government spending
death tax	pass the bill	national forest
illegal aliens	private property	minority leader
class action	border security	urge support
war on terror	president announces	cell lines
embryonic stem	human life	cord blood
tax relief	chief justice	action lawsuits
illegal immigration	human embryos	economic growth

Identify n-grams (*collocations*) and extract those that are distinctively more likely to appear in the corpus of republican versus democratic speaches congressional speeches

Measuring Political Slant



Identify n-grams (*collocations*) and extract those that are distinctively more likely to appear in the corpus of republican versus democratic speaches congressional speeches

Predicting National Identity

Reseach Question What are the costs of social ostracism? Evidence from two World Wars

Corpus Names in census records

Features Distinctly german letter combinations, suffixes in names and surnames

Analysis Predict national identity and study evolution of distinctively German (+Asian) names.

Use machine learning model to predict national identity and study evolution over time. Are there non-linearities in underlying assimmilation pressures? What is the case if (national) identity is non-malleable. This is ongoing research.

Predicting National Identity

Build a predictive model based on common names identified using a Lasso and include dummy coded features extracted using *regular expressions*.

Predicting National Identity

German	English
Koch	Cook
Bauer	Bower
Meier/Meyer/Mayer	Myer
Schmidt/Schmitt	Smith
Muller/Mueller	Miller
Braun	Brown
Weiss	White
Schneider	Taylor
Fischer	Fisher
Weber	Weaver

Surnames with same meaning in German	prefixes and suffixes	
Schmitt/Schmitz/Schmidt/	Neu-	New-
Mueller/Muller	-mann	-man
Becker/Beck	-stein	
Meyer/Mayer/Meier/	-berg	
	-burg	
	-lich	
	-wig	

-sen	-son
-tz	
Sch-	
Pf-	
-ei-	
-oe-	
-ae-	
-ue-	

Build a predictive model based on common names identified using a Lasso and include dummy coded features extracted using *regular expressions*.

Measuring Policy Uncertainty

Reseach Question Construct a measure of policy uncertainty (second moment shocks) to study economic effects.

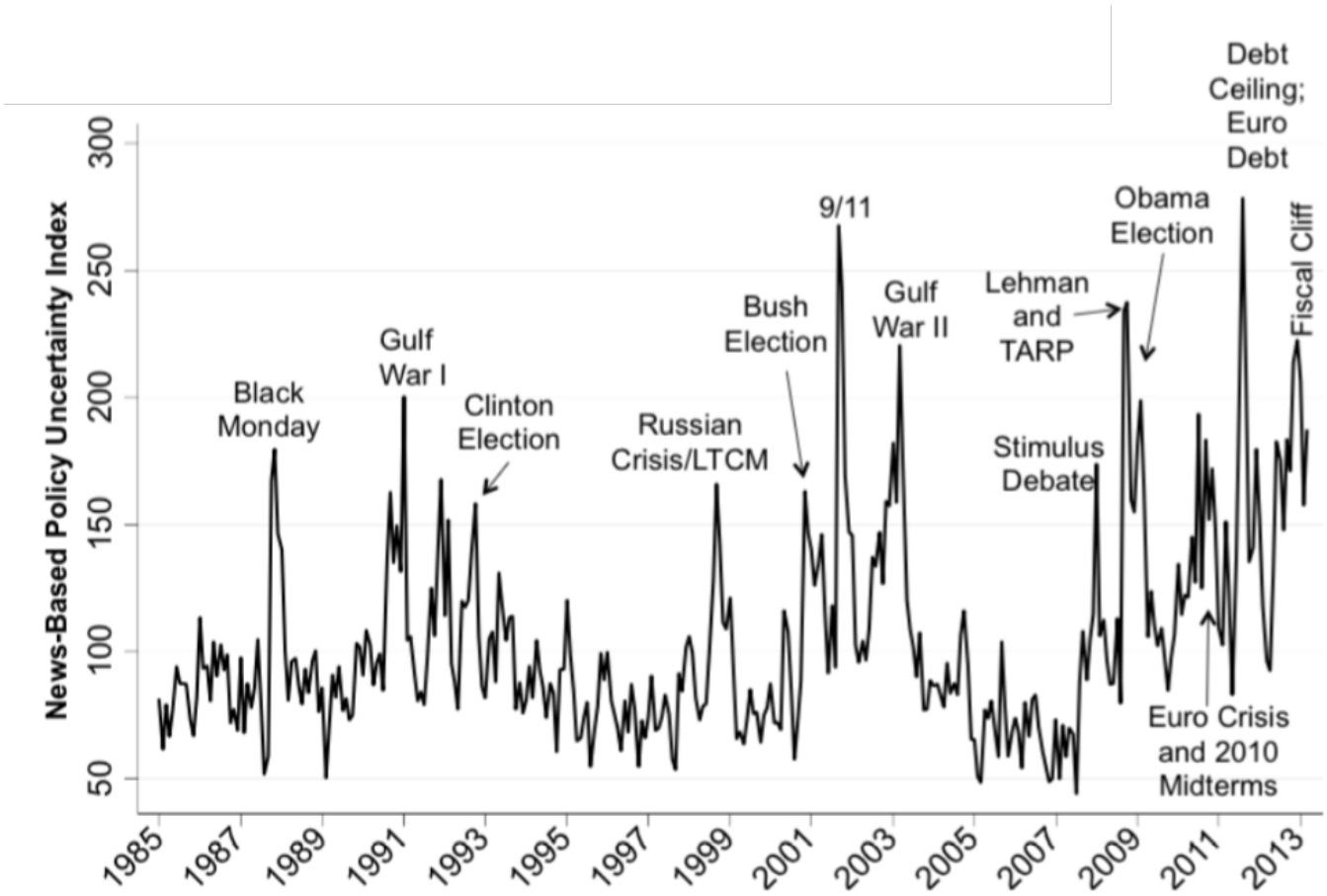
Corpus Digital archives of newspaper

Features Counting articles ("uncertain" OR "uncertainty") AND ("economic" OR "economy") AND ("congress" OR "deficit" OR "federal reserve" OR "legislation" OR "regulation" OR "white house") normalize article counts by total newspaper articles that month.

Analysis Perform econometric analysis on resulting index.

Baker, Bloom, and Davis measure economic policy uncertainty using Boolean search of newspaper articles:
`http://www.policyuncertainty.com/`.

Measuring Policy Uncertainty



Baker, Bloom, and Davis measure economic policy uncertainty using Boolean search of newspaper articles:
`http://www.policyuncertainty.com/`.

Measuring Consumer Confidence

Reseach Question Can Twitter feeds be used to construct consumer high frequency consumer confidence time series?

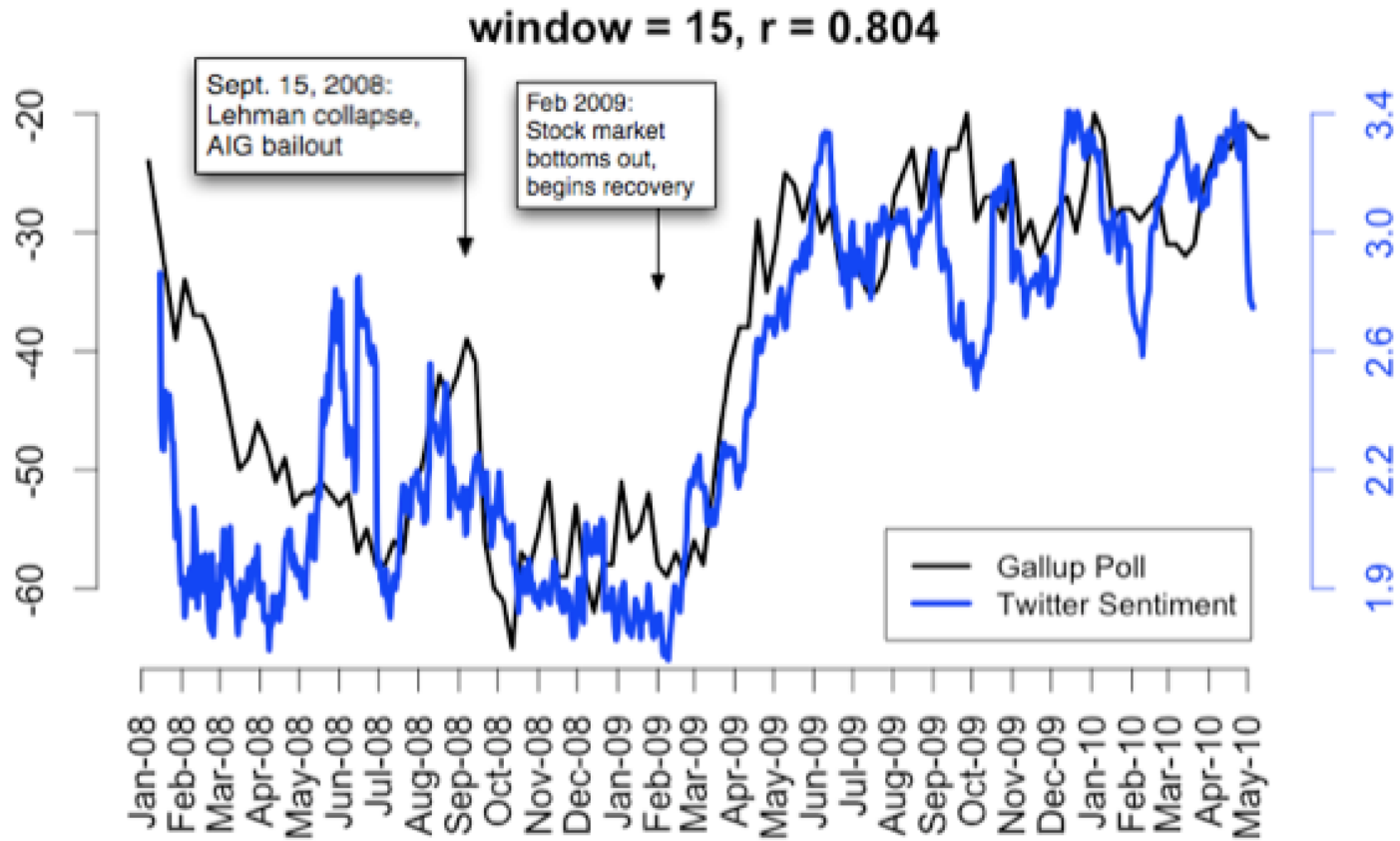
Corpus A large collection of twitter feeds

Features Fragments of text around topic keywords: *economy*, *job*.

Analysis (Heuristic) Approach measuring ratio of positive versus negative coded words around topic keywords.

O'Connor, B., Balasubramanyan, R., Routledge, B. R., & Smith, N. a. (2010). From tweets to polls: Linking text sentiment to public opinion time series.

Measuring Consumer Confidence



O'Connor, B., Balasubramanyan, R., Routledge, B. R., & Smith, N. a. (2010). From tweets to polls: Linking text sentiment to public opinion time series.

Retrieving (Conflict) Event Information

Reseach Question Automatically extract (conflict) event information from textual data.

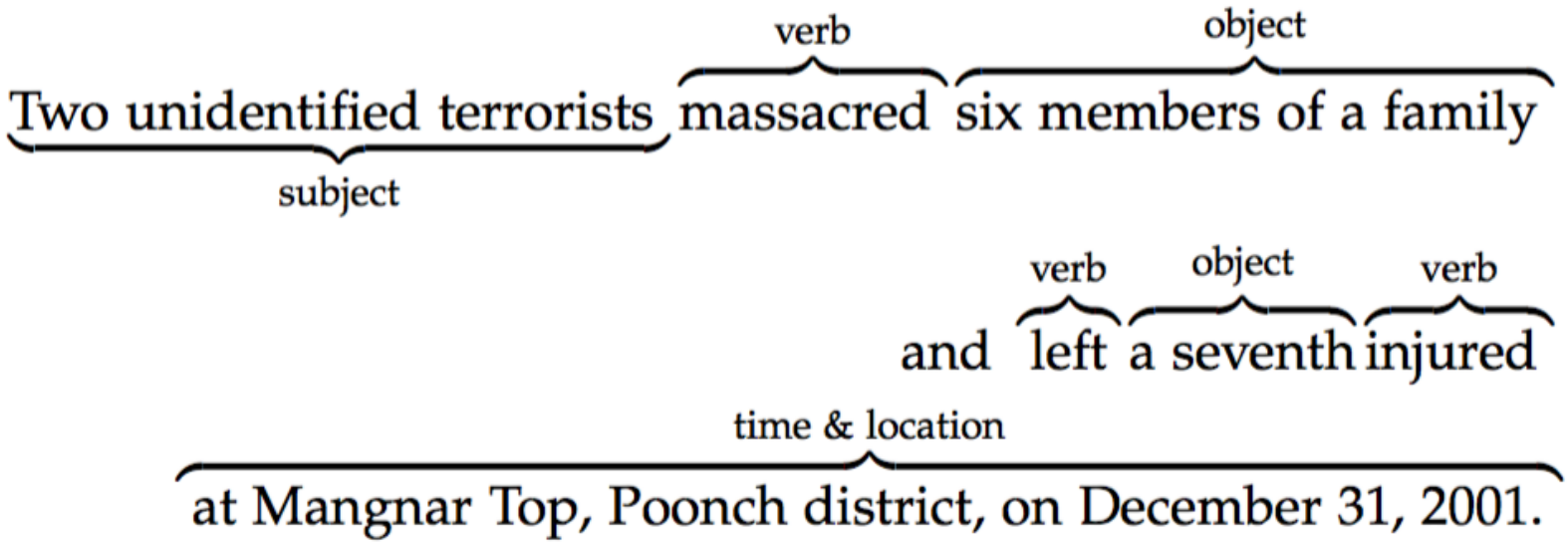
Corpus Digital archive of newspaper clippings about conflict in South Asia.

Features Individual sentences in newspaper article.

Analysis Extract linguistic features to fill event tuple: Subject, Verb, Object pairs and certain Named Entities (Location, Date)

Fetzer, T. (2016) Social Insurance and Conflict: Evidence from India, mimeo.

Retrieving (Conflict) Event Information



Fetzer, T. (2016) Social Insurance and Conflict: Evidence from India, mimeo.

Retrieving (Conflict) Event Information

$$E_1 = \{\text{'Mangar Top Poonch'}, \text{' December 31 2001'}, \text{'massacre'}, \text{' two unidentified terrorists'}, \text{'six members of a family at Mangnar Top, Poonch district'}\}$$

Fetzer, T. (2016) Social Insurance and Conflict: Evidence from India, mimeo.

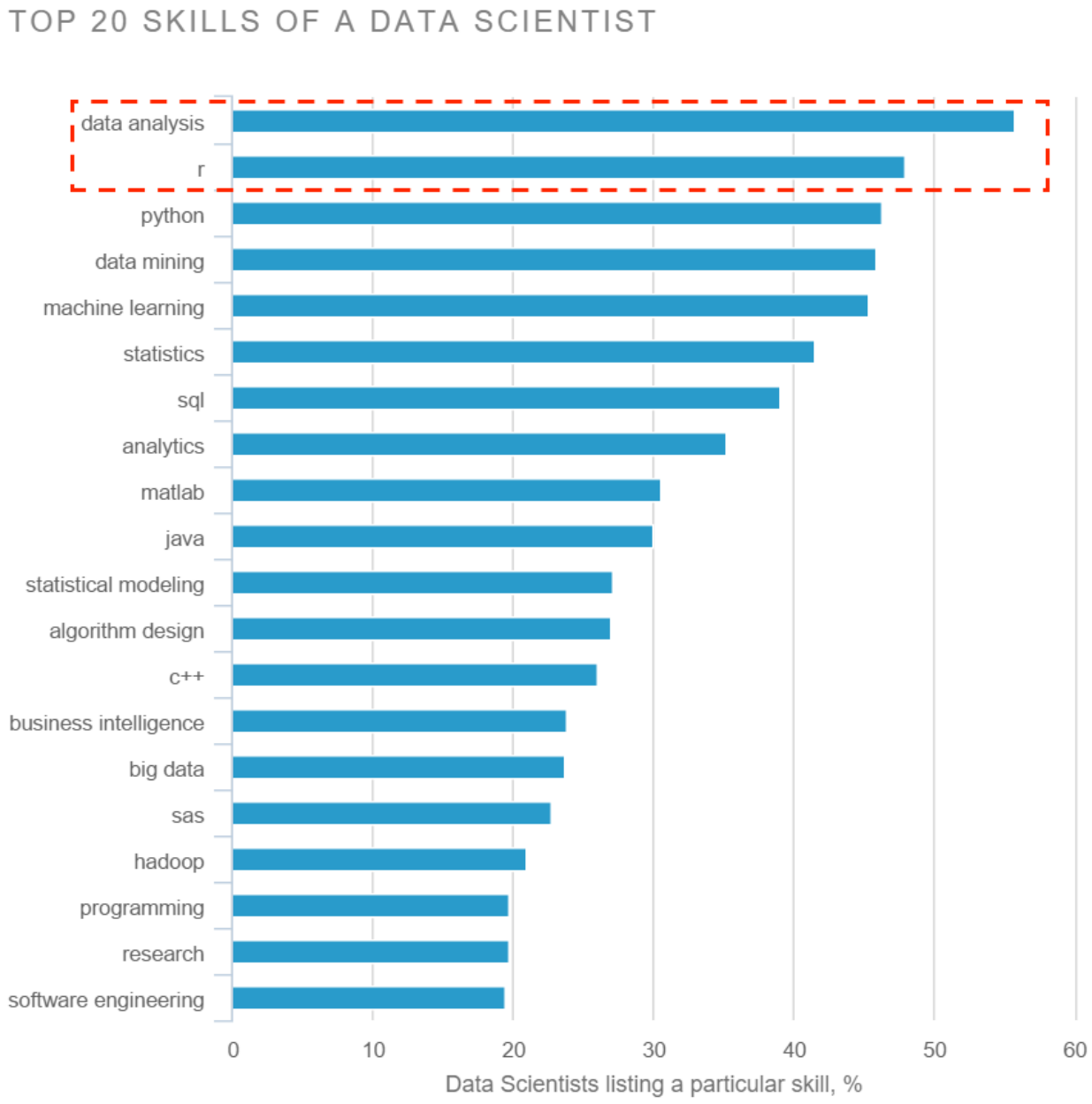
Plan

Introduction and Terminology

Examples from Research

An Introduction to R

Learning R is an investment

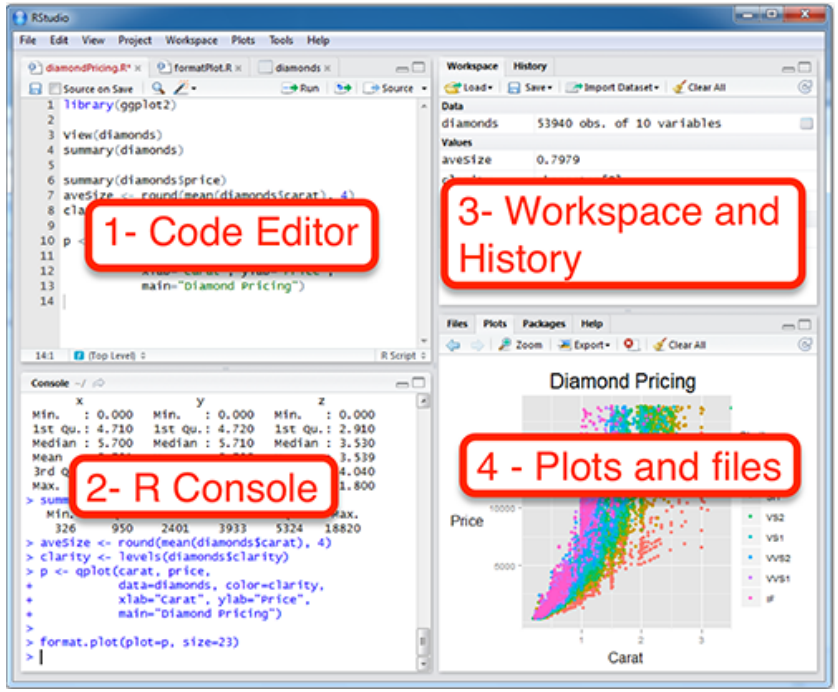


Advantages of R

- Many of you will have some form of prior experience with various programming languages.
- Separation of tasks is quite common: Python often used for *web scraping*, *Stata* used by some for data cleaning and analysis, *ArcGIS* or *QGIS* (and indirectly python) for spatial analysis,...
- Beautiful feature of *R* is its versatility due to the many extensions.

Getting started with R: Rstudio

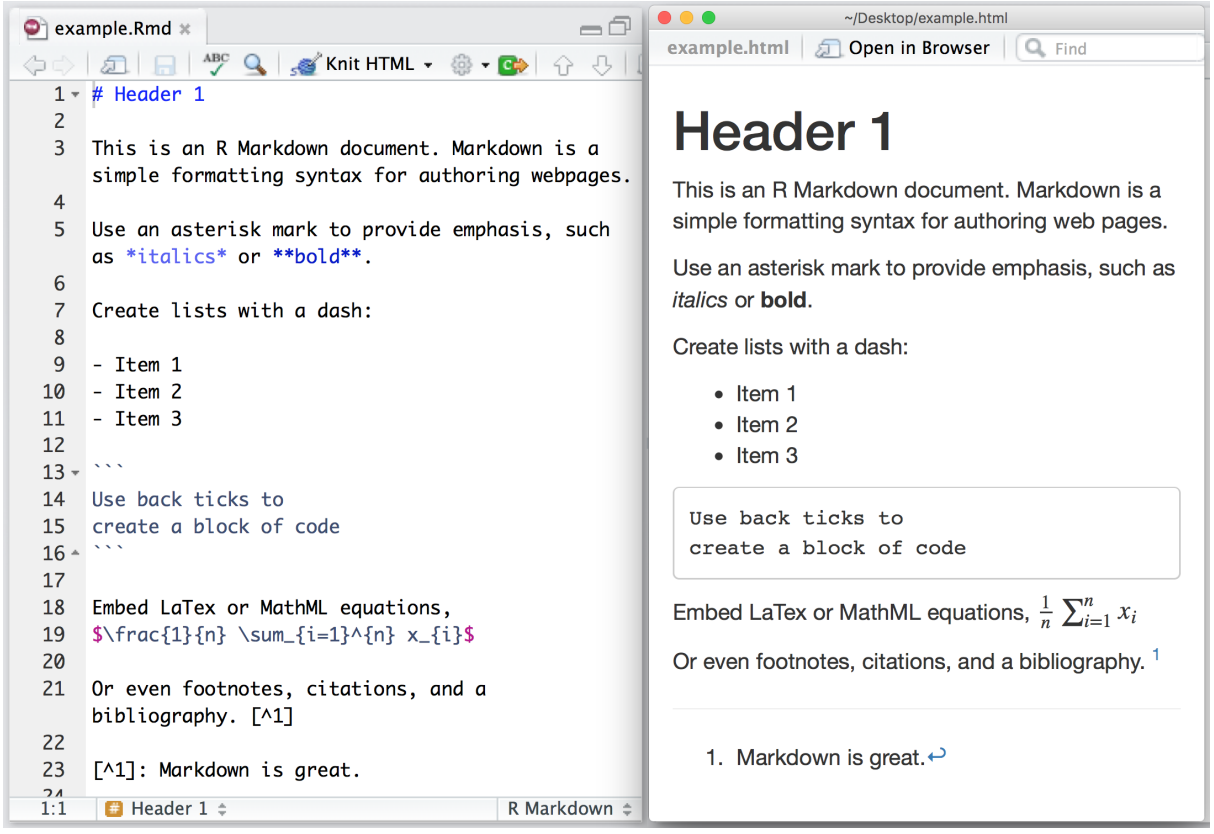
RStudio is a free and open-source integrated development environment (IDE) for R, a programming language for statistical computing and graphics.



`https://www.rstudio.com/products/rstudio/download/`
Alternatively: work in Terminal (command line) or using the R-gui that is shipped with base-R.

Markdown for homeworks

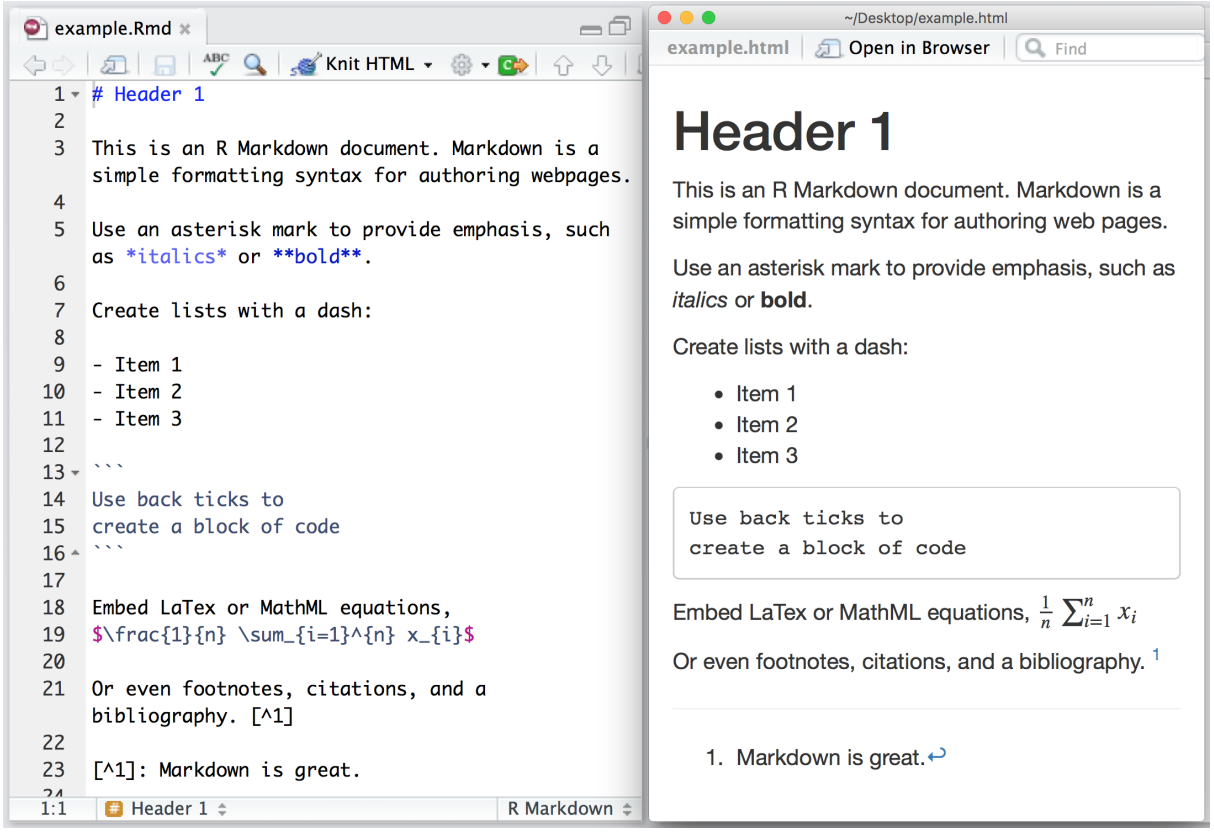
R Markdown makes it easy to turn analyses in R into simple documents or reports.



You will be asked to provide homeworks in R-Markdown format. You will You can describe in the Markdown document what, why and how you proceed with your analysis proiding the code an the results in a simple document.

Markdown for homeworks

R Markdown makes it easy to turn analyses in R into simple documents or reports.



You will be asked to provide homeworks in R-Markdown format. You will You can describe in the Markdown document what, why and how you proceed with your analysis proiding the code an the results in a simple document.

Data Types

These are the basic data types

numeric 8-byte numeric representations

integer non-floating point numbers

character text

logical `TRUE` or `FALSE`

Recursive types also exist, such as lists and vectors; there are also special classifications for `NA`.

Data Types: Integer

```
x <- 10
typeof(x)
```

```
## [1] "double"
```

```
is.integer(x)
```

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

```
x <- 7L
typeof(x)
```

```
## [1] "integer"
```

```
object.size(x)
```

```
## 48 bytes
```

```
as.integer(3.14)
```

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

Data Types: Character

```
typeof("test string")
## [1] "character"
object.size("a")
## 96 bytes
s <- ""  # Unicode cat(s)

as.character("3.14")  # coerce numerics to character
## [1] "3.14"
```

Data Types: numeric

```
x <- 10.5

typeof(x)
## [1] "double"
object.size(x)
## 48 bytes
```

Data Types: factor

```
# create random sample of letters abcd drawn with replacement
x <- sample(letters[1:4], 50, replace = TRUE)
head(x)
```

```
## [1] "c" "a" "d" "b" "d" "c"
```

```
x <- factor(x)
x
```

```
##  [1] c a d b d c c d b b b d a d a b c c c a c b b d c b c a a c a a c d d c d a a b d a d
## [44] c a d b a a a
## Levels: a b c d
```

```
levels(x)
```

```
## [1] "a" "b" "c" "d"
```

Factors are more efficiently stored compared to strings as they are coded as numeric values. It can be dangerous. Setting:
`options(stringsAsFactors=FALSE)`
ensures that when creating data frames, strings are not automatically replaced as factors.

Data Types: `is.*()` and `as.*()`

```
is.numeric(x)
## [1] FALSE
is.numeric(7.1)
## [1] TRUE
is.numeric("7.1")
## [1] FALSE
is.numeric(as.numeric("7.1"))
## [1] TRUE
```

Data Structures

R operates on named *data structures*. The simplest such structure is a numeric vector, which is a single entity consisting of a collection of numbers.

```
x <- c(10.4, 5.6, 3.1, 6.4, 21.7)
x
```

```
## [1] 10.4  5.6  3.1  6.4 21.7
```

```
class(x)
```

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

```
length(x)
```

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

```
b <- c(x, "test")
class(b)
```

```
## [1] "character"
```

Operations on vectors are element by element.

```
z <- sqrt(x)
z * sqrt(x)
```

```
## [1] 10.4  5.6  3.1  6.4 21.7
```

```
# vector - dot product
z %*% sqrt(x)
```

```
##      [,1]
## [1,] 47.2
```

Selecting and modifying subsets of a data set

You can select data or subset data using logical conditions in `[]`.

```
x <- c(x, 1/0)
x
## [1] 10.4  5.6  3.1  6.4 21.7  Inf
x[is.infinite(x)]
## [1] Inf
is.infinite(x)
## [1] FALSE FALSE FALSE FALSE FALSE  TRUE
x[which.max(x)]
## [1] Inf
x[!is.infinite(x)]
## [1] 10.4  5.6  3.1  6.4 21.7
x[1:4]
## [1] 10.4  5.6  3.1  6.4
x[-6]
## [1] 10.4  5.6  3.1  6.4 21.7
```

Functions

```
plusOne <- function(x) {
    return(x + 1)
}

plusOne2 <- function(num) {
    return(num + 1)
}

plusOne(8)
## [1] 9
plusOne2(10)
## [1] 11
plusOne2(num = 5)
## [1] 6
# plusOne2(wrongVar=2)
```

Loops

```
for (number in 1:5) {
    print(number)
}
```

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

```
# Looping over functions
a <- c(1, 2, 3, 4, 5)
for (value in a) {
    print(plusOne(value))
}
```

```
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
```

```
listOfNumbers <- c(1, 2, 3, 4, 5)
for (number in listOfNumbers) {
    print(number + 1)
}
```

```
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
```

```
a <- c(1, 2, 3, 4, 5)

for (i in 1:length(a)) {
    print(plusOne(a[i]))
}
```

Complex Data Structures: Matrices

Matrices combine data vectors coercing a common type: all vectors that make up a matrix have same data type (e.g. numeric, character).

```
x <- rnorm(n = 26)
y <- runif(n = 26)
MAT <- cbind(x, y)
class(MAT)
```

```
## [1] "matrix"
```

```
rownames(MAT) <- 1:nrow(MAT)
colnames(MAT) <- c("x", "y")

# subsetting
MAT[12, "y"]
```

```
## [1] 0.601
```

```
MAT[12, ]
```

```
##      x      y
## -1.285  0.601
```

```
head(letters)
```

```
## [1] "a" "b" "c" "d" "e" "f"
```

```
MAT <- cbind(MAT, letters)
# adding a character column changes type of all elements
head(MAT)
```

```
##   x                    y                    letters
## 1 "0.614396521078847"  "0.830132150556892" "a"
## 2 "-0.575673837968414" "0.919079559622332" "b"
## 3 "1.0992417048328"    "0.982811939436942" "c"
## 4 "0.183811909875599"  "0.430371002294123" "d"
## 5 "1.06152098103392"   "0.824281533248723" "e"
## 6 "-1.18143403051857"  "0.97188643598929"  "f"
```

Complex Data Structures: data.frame

data.frames allow the combination of multiple object types without their underlying types being changed. This is the central data object that is typically worked with.

```
x <- rnorm(n = 26)
y <- runif(n = 26)
DF <- data.frame(x, y)
class(DF)
```

```
## [1] "data.frame"
```

```
DF[1:2, c("y", "x")]
```

```
##       y    x
## 1 0.399 1.45
## 2 0.150 2.07
```

```
head(DF$x)
```

```
## [1]  1.454  2.070  1.141 -0.887  0.581 -0.933
```

```
class(DF$x)
```

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

```
DF <- cbind(DF, letters)
class(DF)
```

```
## [1] "data.frame"
```

```
class(DF$x)
```

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

```
class(DF$letters)
```

```
## [1] "factor"
```

data.frames with functionality: data.table

A data.table is a list of vectors, just like a data.frame. However :

- it never has or uses rownames. Rownames based indexing can be done by setting a key of one or more columns or done ad-hoc using the on argument (now preferred).
- it has enhanced functionality in [.data.table for fast joins of keyed tables, fast aggregation, fast last observation carried forward (LOCF) and fast add/modify/delete of columns by reference with no copy at all.
- There are several other methods that are available for operating on data.tables efficiently.

data.table builds on base R functionality to reduce 2 types of time:

1. programming time (easier to write, read, debug and maintain), and
2. compute time (fast and memory efficient).

The general form of data.table syntax is:

```
DT[ i,   j,   by ] # + extra arguments
    |    |    |
    |    |     -------> grouped by what?
    |    -------> what to do?
    ---> on which rows?
```

data.frames with functionality: data.table

```
library(data.table)
# every data.table is also a data.frame
DT <- data.table(DF)
## aggregation
DT[, list(mean(x), sum(y))]
```

```
##        V1   V2
## 1: -0.219 13.6
```

Complex Data Structures: Lists

Lists are the most versatile data structure, allowing the collection of different lists.

```
LIST <- list(DT, DF, letters)

class(LIST)

## [1] "list"

class(LIST[[3]])

## [1] "character"

head(LIST[[2]])

##        x     y letters
## 1  1.454 0.399       a
## 2  2.070 0.150       b
## 3  1.141 0.939       c
## 4 -0.887 0.866       d
## 5  0.581 0.290       e
## 6 -0.933 0.959       f

# can append items
LIST[[4]] <- y

length(LIST)

## [1] 4
```

Apply functionality: vectorization

Loops in R are inherently inefficient; most often you need to apply a function to every observation and this is where the `apply()` function family comes into play.

base::apply Apply Functions Over Array Margins

base::eapply Apply a Function Over Values in an Environment

base::lapply Apply a Function over a List or Vector

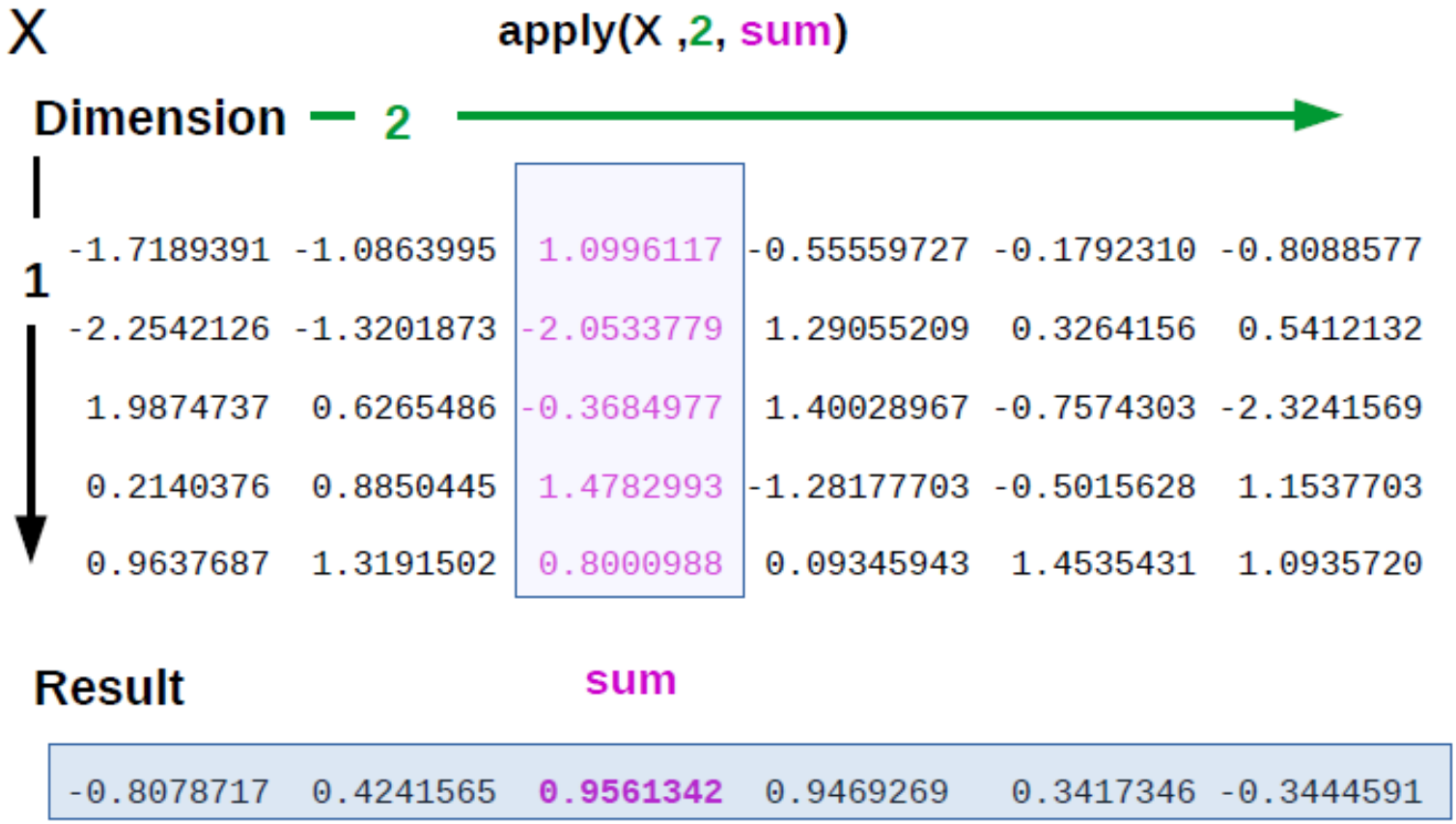
base::mapply Apply a Function to Multiple List or Vector Arguments

base::rapply Recursively Apply a Function to a List

base::tapply Apply a Function Over a Ragged Array

Apply functionality: `apply`

Returns a vector or array or list of values obtained by applying a function to margins of a matrix.



Apply functionality: apply

Returns a vector or array or list of values obtained by applying a function to margins of a matrix.

```
# create a matrix with 100 rows and 100 colums with random values drawn from uniform
# distribution
DF <- do.call("cbind", lapply(1:100, function(x) runif(n = 100)))

class(DF)
```

```
## [1] "matrix"
```

```
# apply function to each row: computing row sums
res <- apply(DF, 1, sum)

res
```

```
##   [1] 47.2 47.5 56.4 43.0 50.7 51.3 48.4 50.6 53.4 52.4 46.8 54.1 50.7 47.5 49.9 49.1 46.9
##  [18] 51.9 48.4 54.9 49.3 52.8 53.9 51.2 45.1 42.6 53.5 49.9 51.8 44.8 53.6 42.8 46.9 54.8
##  [35] 49.0 50.7 47.1 53.6 52.5 53.3 49.2 46.1 56.8 50.0 48.0 53.2 49.1 52.3 49.3 55.0 46.5
##  [52] 47.6 49.1 55.4 49.2 54.6 52.6 51.0 49.2 46.1 47.8 54.4 50.5 49.3 54.2 48.2 45.6 48.6
##  [69] 50.4 50.8 49.7 50.0 49.4 45.9 52.8 48.1 52.5 50.0 49.9 49.9 48.0 51.5 52.0 55.2 50.4
##  [86] 49.8 52.5 44.4 52.0 53.2 47.6 48.8 51.5 49.1 49.9 53.8 53.2 51.1 55.0 53.2
```

```
rowSums(DF)
```

```
##   [1] 47.2 47.5 56.4 43.0 50.7 51.3 48.4 50.6 53.4 52.4 46.8 54.1 50.7 47.5 49.9 49.1 46.9
##  [18] 51.9 48.4 54.9 49.3 52.8 53.9 51.2 45.1 42.6 53.5 49.9 51.8 44.8 53.6 42.8 46.9 54.8
##  [35] 49.0 50.7 47.1 53.6 52.5 53.3 49.2 46.1 56.8 50.0 48.0 53.2 49.1 52.3 49.3 55.0 46.5
##  [52] 47.6 49.1 55.4 49.2 54.6 52.6 51.0 49.2 46.1 47.8 54.4 50.5 49.3 54.2 48.2 45.6 48.6
##  [69] 50.4 50.8 49.7 50.0 49.4 45.9 52.8 48.1 52.5 50.0 49.9 49.9 48.0 51.5 52.0 55.2 50.4
##  [86] 49.8 52.5 44.4 52.0 53.2 47.6 48.8 51.5 49.1 49.9 53.8 53.2 51.1 55.0 53.2
```

```
sum(res != rowSums(DF))
```

```
## [1] 0
```

```
# apply function to each column: computing column sums
res <- apply(DF, 2, sum)
```

Apply functionality: `lapply`

`lapply` returns a list of the same length as X, each element of which is the result of applying FUN to the corresponding element of X.

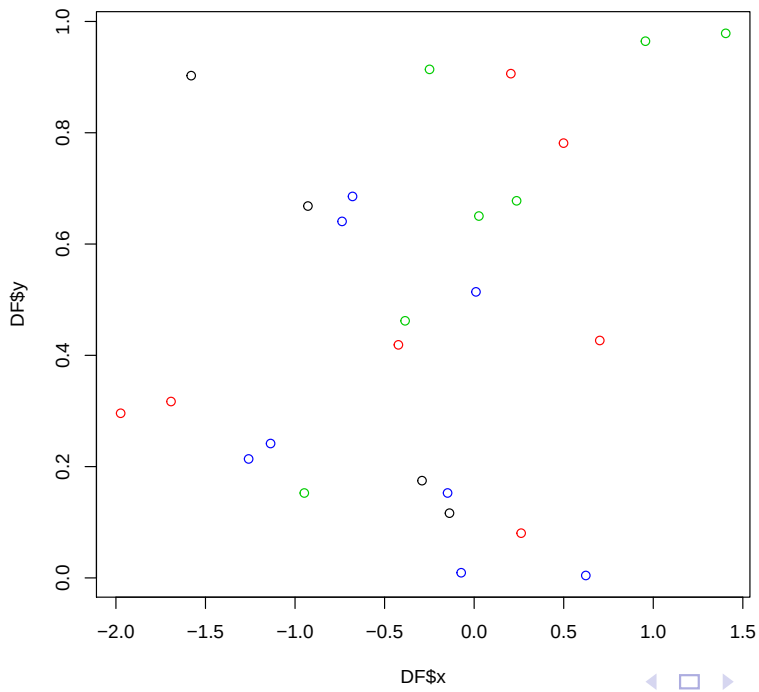
```
# apply any function on a vector, returns a list
res <- lapply(1:1000, function(x) x^0.5)

class(res)

## [1] "list"

head(unlist(res))

## [1] 1.00 1.41 1.73 2.00 2.24 2.45
```

"sapply is a user-friendly version of lapply by default returning a vector or matrix if appropriate."

Basic Plots

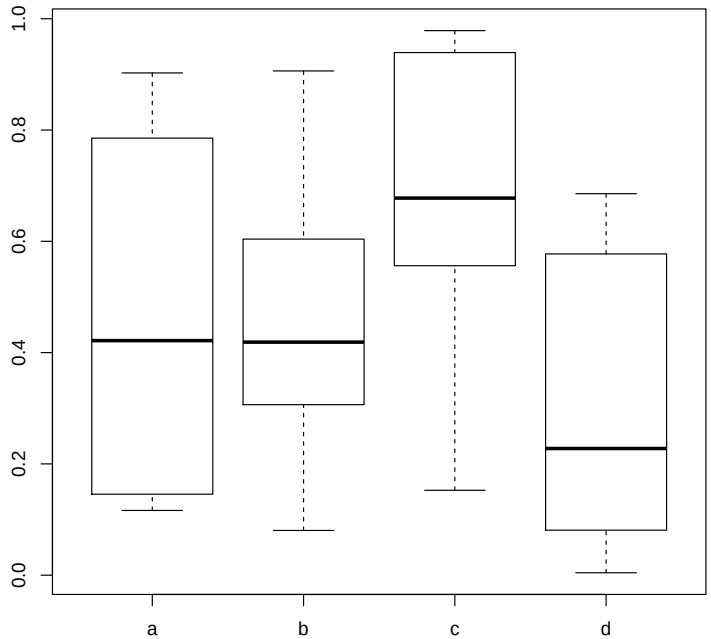
R has amazing plotting functionality, beyond the scope of this course to go in too much detail. Base-R provides simple plotting functionality, advanced and fancy plots are possible with the ggplot package.
Base-R tries to guess the appropriate plotting function dependent on the data type.

```
x <- rnorm(n = 26)
y <- runif(n = 26)
DF <- data.frame(x, y, letters = sample(letters[1:4], 26, replace = TRUE))
# simple scatter plot
plot(DF$x, DF$y, col = DF$letters)
```



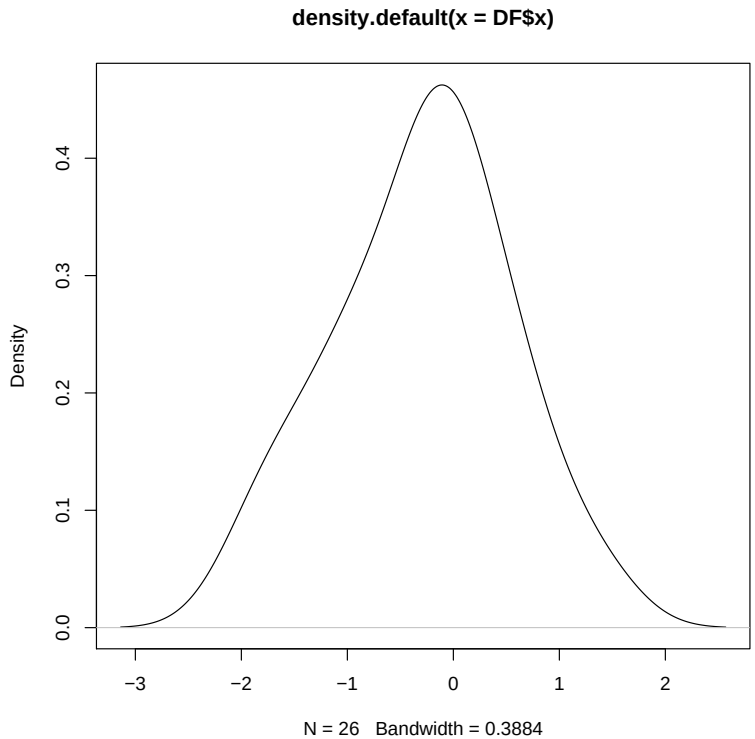
Basic Plots

```
# letters is a factor (an encoded string), so appropriate is to draw a boxplot
plot(DF$letters, DF$y)
```



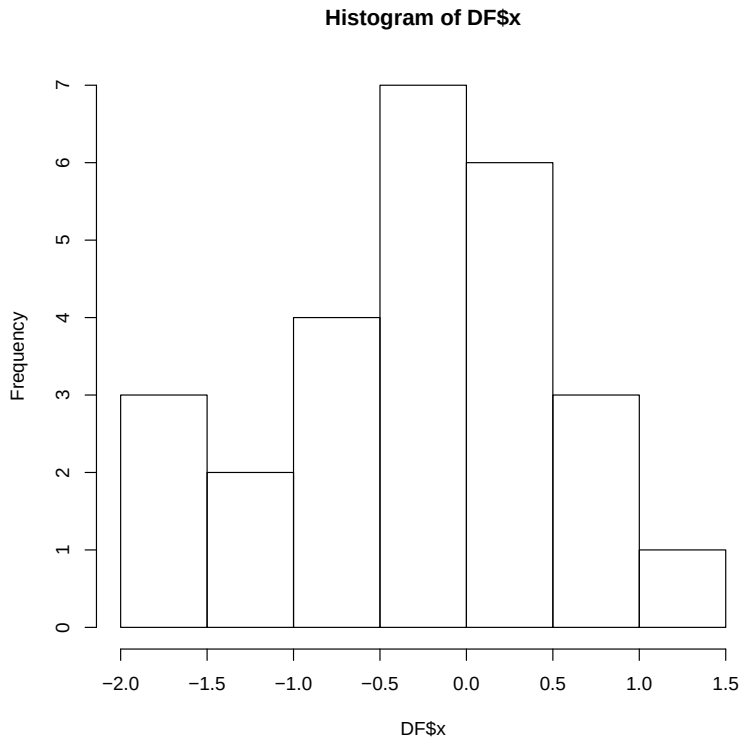
Basic Plots

```
# kernel density
plot(density(DF$x))
```



Basic Plots

```
# histogram
hist(DF$x)
```

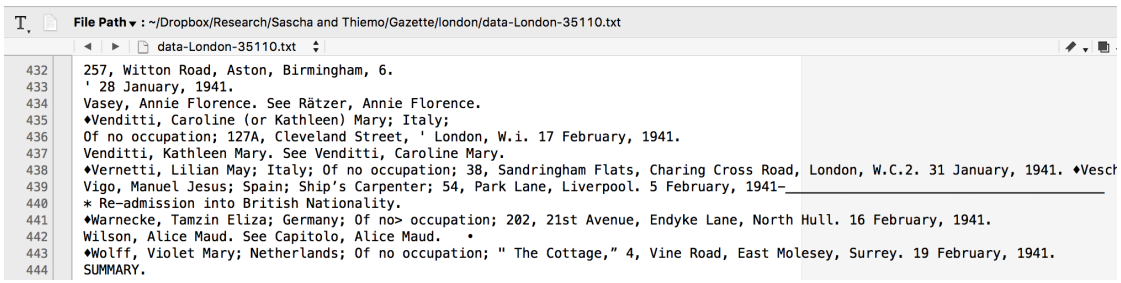


A Texteditor is very very useful...

- Textwranger (Mac), Notepad++ (Windows), or system text editor (Windows Editor / Mac Text.app)
- Very good to look at raw text in editor before loading into R to see how messy it is.
- Especially machine read data (PDFs processed with Optical Character recognition - OCR)

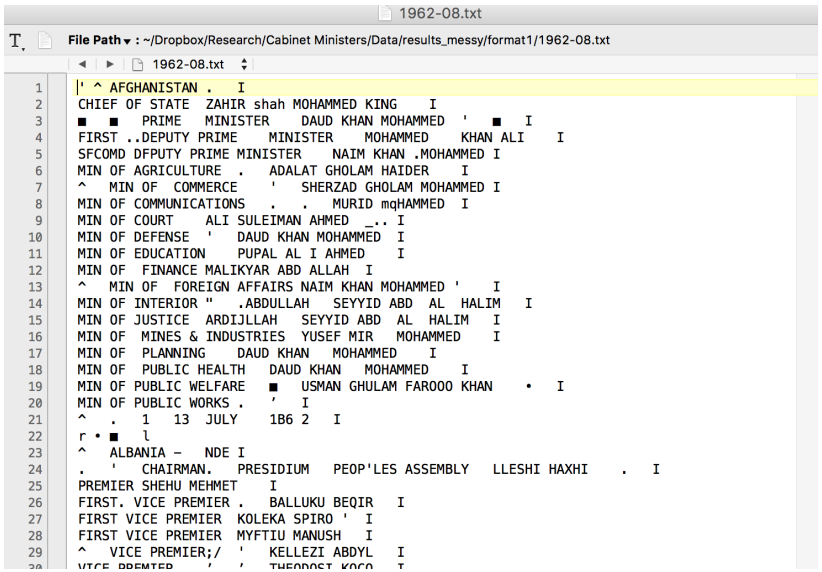
A Texteditor is very very useful...

- Textwranger (Mac), Notepad++ (Windows), or system text editor (Windows Editor / Mac Text.app)
- Very good to look at raw text in editor before loading into R to see how messy it is.
- Especially machine read data (PDFs processed with Optical Character recognition - OCR)



A Texteditor is very very useful...

- Textwranger (Mac), Notepad++ (Windows), or system text editor (Windows Editor / Mac Text.app)
- Very good to look at raw text in editor before loading into R to see how messy it is.
- Especially machine read data (PDFs processed with Optical Character recognition - OCR)



Text Encoding: A cautionary note

- Especially working with data from different computer systems (Mac, Unix, Windows) and different languages, character encoding issues can occur.
- The Latin character set (Encoding: `latin-1`) is used by English and most European languages, though the Greek character set is used only by the Greek language.
- Loading data without knowing character encoding can sometimes cause errors
- Text editors like Textwranger attempt to recognize character encoding
- Most common encoding is `UTF-8`

```
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

Error in nchar(homeDir) : invalid multibyte string 1
>
Error in nchar(homeDir) : invalid multibyte string 1
Error in nchar(homeDir) : invalid multibyte string 1
Error in nchar(homeDir) : invalid multibyte string 1
>
```