

EC999: Introduction

Thiemo Fetzer

University of Chicago & University of Warwick

January 5, 2017

Some Organizational 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. I need to change the slot on Tuesday morning, lets see whether we can find a slot <http://doodle.com/poll/ga2thc6k5w9xa2z3>.
3. About this course: this is a new course taught for the first time, and it draws from many different fields:
 - ▶ Computational Linguistics
 - ▶ Machine Learning
 - ▶ Data science
 - ▶ Statistics
 - ▶ Programming in R
4. Your input is needed: too slow, too fast, ...? just email me.
5. Assessment: I will do a mix of a short final exam and hand in assignments
6. 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 source files, are made available on <http://www.trfetzer.com/EC999> (and maybe Github)
- ▶ Three to four assignments where I will ask you to perform some analysis. You are asked to produce an Markdown document (more later) that solves the assignment you were asked to do.
- ▶ The course will become a bit more formal in places, formal aspects (not the programming part), will be assessed in a short final exam.

Plan

Introduction and Terminology

Examples from Research

An Introduction to R

Quantitative Text Analysis as process

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

When I presented the supplementary budget to this House last April, I said we could work our way through this period of severe economic distress. Today, I can report that notwithstanding the difficulties of the past eight months, we are now on the road to economic recovery.

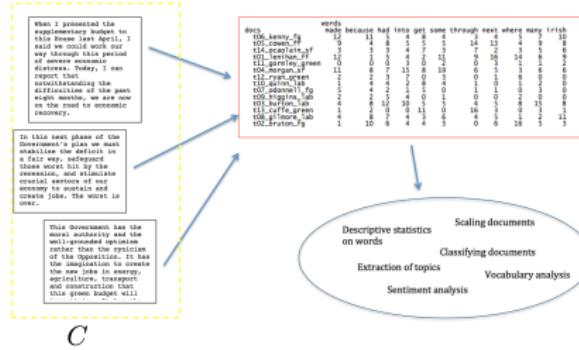
In this next phase of the Government's plan we must stabilise the deficit in a fair way, safeguard those worst hit by the recession, and stimulate crucial sectors of our economy to sustain and create jobs. The worst is over.

This Government has the moral authority and the well-grounded optimism rather than the cynicism of the Opposition. It has the imagination to create the new jobs in energy, agriculture, transport and construction that this green budget will

docs	made	because	had	into	get	some	through	next	where	many	irish
t06_kenny_fg	12	11	5	4	8	4	3	4	5	7	10
t05_cowen_fg	9	4	8	5	5	5	14	13	4	9	8
t14_aovalain_sf	3	3	3	4	7	3	7	2	3	5	6
t01_lenihan_ff	12	1	5	4	2	11	9	16	14	6	9
t11_gormley_green	0	0	0	3	0	2	0	3	1	1	2
t04_morgan_sf	11	8	7	15	8	19	6	5	3	6	6
t12_ryan_green	2	2	3	7	0	3	0	1	6	0	0
t10_quinn_lab	1	4	4	2	8	4	1	0	1	2	0
t07_odonnell_fg	5	4	2	1	5	0	1	1	0	3	0
t09_higgins_lab	2	2	5	4	0	1	0	0	2	0	0
t03_burton_lab	4	8	12	10	5	5	4	5	8	15	8
t13_cuffe_green	1	2	0	0	11	0	16	3	0	3	1
t08_gilmore_lab	4	8	7	4	3	6	4	5	1	2	11
t02_bruton_fg	1	10	6	4	4	3	0	6	16	5	3

-
- ```
graph TD; A[Text 1] --> B[Text 2]; A --> C[Table]; B --> C; C --> D[Scaling documents]; C --> E[Classifying documents]; C --> F[Extraction of topics]; C --> G[Vocabulary analysis]; C --> H[Sentiment analysis];
```
- Scaling documents
  - Classifying documents
  - Extraction of topics
  - Vocabulary analysis
  - Sentiment analysis

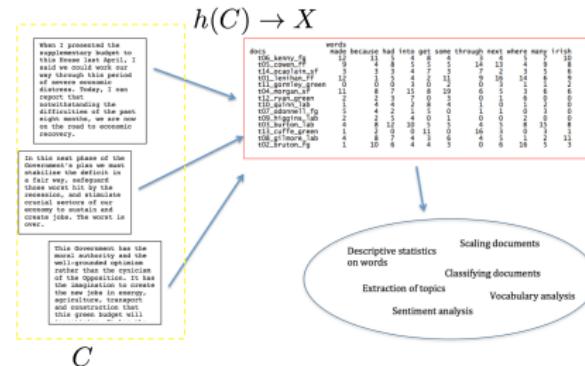
# Step 1: Corpus Definition



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,...

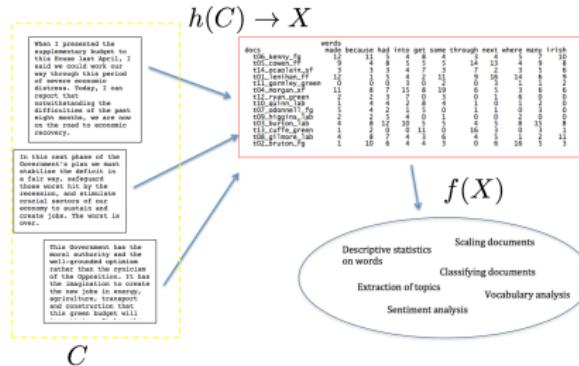
## Step 2: Feature Definition and Selection



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 representation 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** canonical 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

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

**Corpus** Republican and democratic speeches, newspaper front page content

**Features** Distinctively 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  
trade agreement  
american people  
tax breaks  
trade deficit  
oil companies  
credit card  
nuclear option  
war in iraq  
middle class

rosa parks  
president budget  
republican party  
change the rules  
minimum wage  
budget deficit  
republican senators  
privatization plan  
wildlife refuge  
card companies

workers rights  
poor people  
republican leader  
arctic refuge  
cut funding  
american workers  
living in poverty  
senate republicans  
fuel efficiency  
national wildlife

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

# Measuring Political Slant

Panel B: Phrases used more often by Republicans

## *Two-word phrases*

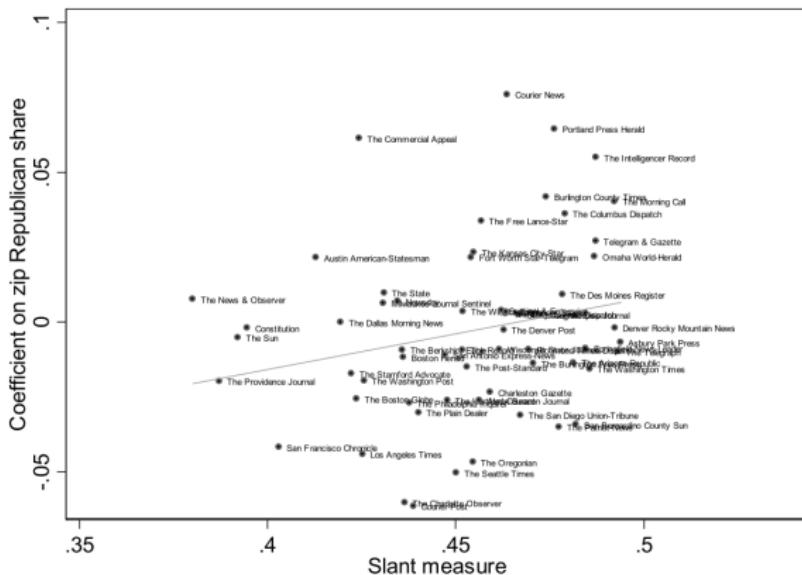
stem cell  
natural gas  
death tax  
illegal aliens  
class action  
war on terror  
embryonic stem  
tax relief  
illegal immigration

personal accounts  
saddam hussein  
pass the bill  
private property  
border security  
president announces  
human life  
chief justice  
human embryos

retirement accounts  
government spending  
national forest  
minority leader  
urge support  
cell lines  
cord blood  
action lawsuits  
economic growth

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

# Predicting National Identity

**Research 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 assimilation 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<br>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-                   |      |
|                                         | -ci-                  |      |
|                                         | -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

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

**Corpus** Digital archives of newspaper

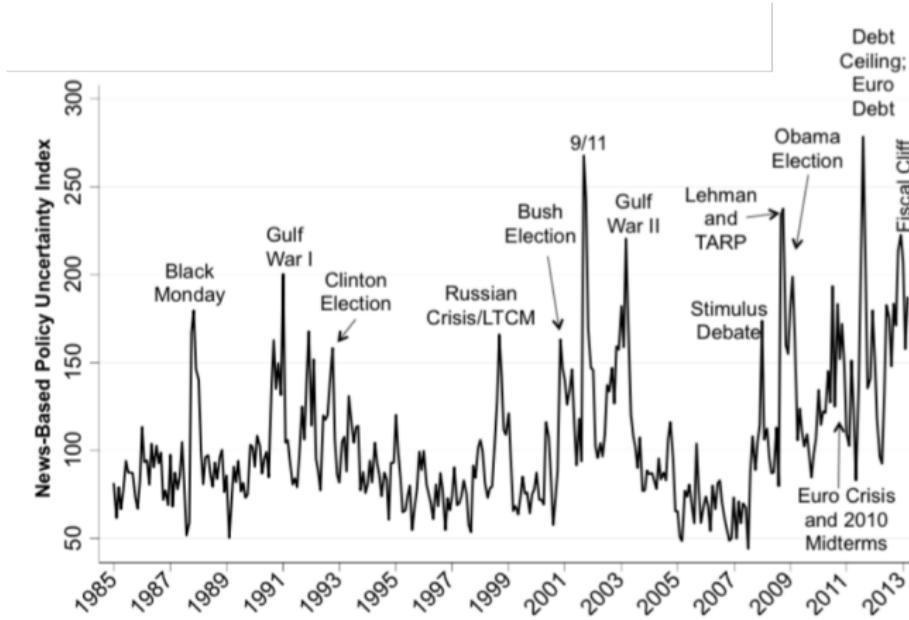
**Features** Counting articles (“uncertain” OR “uncertainty”) AND (“economic” OR “economy”) AND (“congress” OR “deficit” OR “federal reserve” OR “legislation” OR “regulation” OR “white house”) normalize article counts by total newspaper articles that month.

**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

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

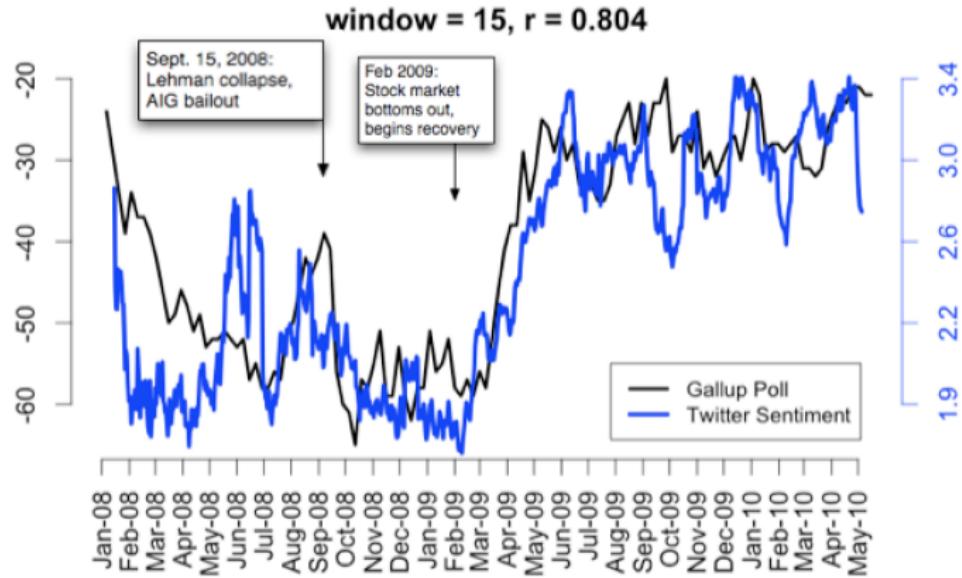
**Corpus** A large collection of twitter feeds

**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

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

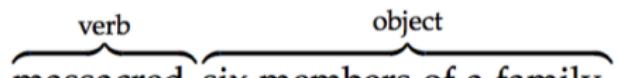
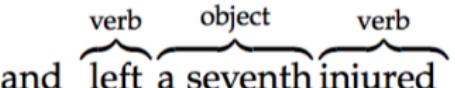
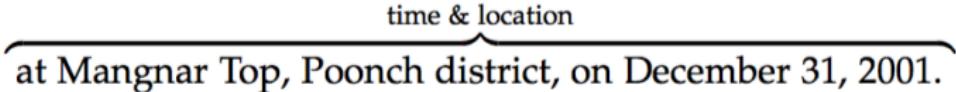
**Corpus** Digital archive of newspaper clippings about conflict in South Asia.

**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

Two unidentified terrorists   
and   
at Mangnar Top, Poonch district, on December 31, 2001. 

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

## Retrieving (Conflict) Event Information

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

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

# Plan

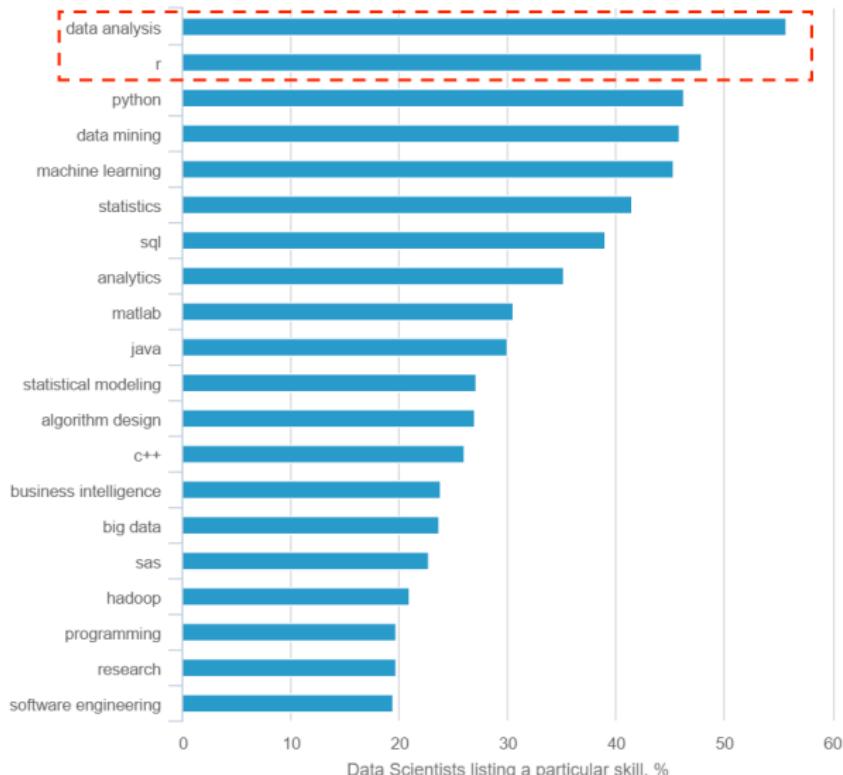
Introduction and Terminology

Examples from Research

An Introduction to R

# Learning R is an investment

TOP 20 SKILLS OF A DATA SCIENTIST

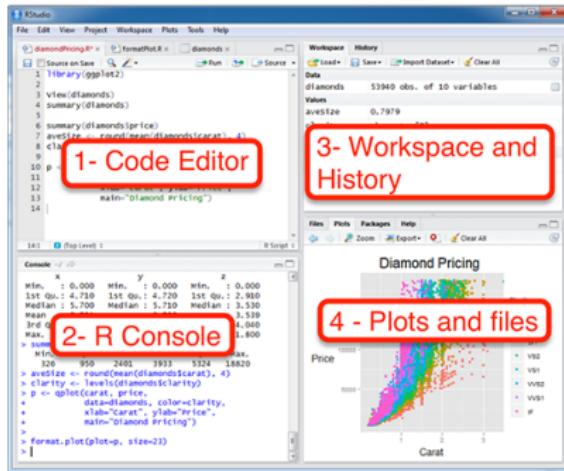


## Advantages of R

- ▶ 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.

The screenshot shows the RStudio interface. On the left, the code editor window titled "example.Rmd" contains R Markdown code. On the right, the browser window titled "example.html" shows the rendered document. The rendered document includes a header, text, lists, a code block, and a bibliography section.

```
example.Rmd
1 # Header 1
2
3 This is an R Markdown document. Markdown is a
4 simple formatting syntax for authoring webpages.
5 Use an asterisk mark to provide emphasis, such
6 as *italics* or **bold**.
7 Create lists with a dash:
8
9 - Item 1
10 - Item 2
11 - Item 3
12
13 ...
14 Use back ticks to
15 create a block of code
16 ```
17
18 Embed LaTeX or MathML equations,
19 $\frac{1}{n} \sum_{i=1}^n x_i
20
21 Or even footnotes, citations, and a
22 bibliography. [^1]
23
24 [^1]: Markdown is great.
```

Header 1

This is an R Markdown document. Markdown is a simple formatting syntax for authoring web pages. Use an asterisk mark to provide emphasis, such as *italics* or **bold**.

Create lists with a dash:

- Item 1
- Item 2
- Item 3

Use back ticks to create a block of code

Embed LaTeX or MathML equations,  $\frac{1}{n} \sum_{i=1}^n x_i$

Or even footnotes, citations, and a bibliography.<sup>[^1]</sup>

1. Markdown is great.<sup>[^1]</sup>

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

# Markdown for homeworks

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

The screenshot shows two windows side-by-side. On the left is a text editor window titled "example.Rmd" containing R Markdown code. On the right is a web browser window displaying the rendered HTML document.

**Code (example.Rmd):**

```
1 # Header 1
2
3 This is an R Markdown document. Markdown is a
4 simple formatting syntax for authoring webpages.
5 Use an asterisk mark to provide emphasis, such
6 as *italics* or **bold**.
7 Create lists with a dash:
8
9 - Item 1
10 - Item 2
11 - Item 3
12
13 ...
14 Use back ticks to
15 create a block of code
16 ...
17
18 Embed LaTeX or MathML equations,
19 $\frac{1}{n} \sum_{i=1}^n x_i$
20
21 Or even footnotes, citations, and a
22 bibliography. [^1]
23
24 [^1]: Markdown is great.
```

**HTML Output:**

# Header 1

This is an R Markdown document. Markdown is a simple formatting syntax for authoring web pages. Use an asterisk mark to provide emphasis, such as *italics* or **bold**.

Create lists with a dash:

- Item 1
- Item 2
- Item 3

Use back ticks to  
create a block of code

Embed LaTeX or MathML equations,  $\frac{1}{n} \sum_{i=1}^n x_i$

Or even footnotes, citations, and a bibliography.<sup>1</sup>

1. Markdown is great. [^](#)

You will be asked to provide homeworks in R-Markdown format. You will be able to describe in the Markdown document what, why and how you proceed with your analysis providing the code and 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] "b" "c" "d" "b" "b" "b"

x<-factor(x)
x

[1] b c d b b b c d d c a a d b a d a c b d b a c d c a a c a a d b c c d c d c d a b d d
[44] d c b a c c c
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)
```



# 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] 0.0373

MAT[12,]

x y
-1.9274 0.0373

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 "2.13631660354939" "0.364831202430651" "a"
2 "1.21815435633569" "0.222715873736888" "b"
3 "-1.62450225224365" "0.298199398675933" "c"
4 "-0.0354296365102945" "0.796236804686487" "d"
5 "-1.9211767346097" "0.688834731001407" "e"
6 "0.754742543238073" "0.903118321672082" "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 0.156 1.447
2 0.972 0.125

head(DF$x)
[1] 1.447 0.125 -0.387 0.264 0.434 -0.942

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: 0.216 14.9
```

# 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 1.447 0.1562 a
2 0.125 0.9717 b
3 -0.387 0.8357 c
4 0.264 0.7100 d
5 0.434 0.0975 e
6 -0.942 0.2296 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.

| X             | apply(X ,2, sum)                                                                   |
|---------------|------------------------------------------------------------------------------------|
| Dimension — 2 |  |
| 1             | -1.7189391 -1.0863995 1.0996117 -0.55559727 -0.1792310 -0.8088577                  |
|               | -2.2542126 -1.3201873 -2.0533779 1.29055209 0.3264156 0.5412132                    |
|               | 1.9874737 0.6265486 -0.3684977 1.40028967 -0.7574303 -2.3241569                    |
|               | 0.2140376 0.8850445 1.4782993 -1.28177703 -0.5015628 1.1537703                     |
|               | 0.9637687 1.3191502 0.8000988 0.09345943 1.4535431 1.0935720                       |
| Result        | sum                                                                                |
|               | -0.8078717 0.4241565 0.9561342 0.9469269 0.3417346 -0.3444591                      |

# 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 columns 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] 48.5 53.9 52.7 49.5 51.8 51.1 51.3 48.6 51.2 46.9 55.0 55.5 48.4 51.4 56.3 52.6 54.7
[18] 48.1 45.0 43.4 52.1 46.7 48.3 50.4 51.8 51.6 52.5 48.8 50.9 52.8 53.2 53.2 52.1 52.1
[35] 47.4 48.1 51.4 46.4 47.3 46.9 47.6 46.9 43.2 53.8 52.3 44.7 50.3 43.3 52.3 49.9 49.7
[52] 50.2 49.4 50.7 48.9 52.8 49.9 44.8 49.0 46.1 50.2 47.8 45.6 45.5 47.0 51.6 53.8 52.3
[69] 48.6 50.2 51.4 53.2 52.5 45.1 54.0 49.3 47.5 55.8 47.2 45.6 47.7 45.6 51.7 52.7 50.9
[86] 51.3 49.6 52.3 50.5 49.9 48.4 44.6 50.0 48.8 49.7 44.8 47.7 50.2 49.6 50.4

rowSums(DF)
[1] 48.5 53.9 52.7 49.5 51.8 51.1 51.3 48.6 51.2 46.9 55.0 55.5 48.4 51.4 56.3 52.6 54.7
[18] 48.1 45.0 43.4 52.1 46.7 48.3 50.4 51.8 51.6 52.5 48.8 50.9 52.8 53.2 53.2 52.1 52.1
[35] 47.4 48.1 51.4 46.4 47.3 46.9 47.6 46.9 43.2 53.8 52.3 44.7 50.3 43.3 52.3 49.9 49.7
[52] 50.2 49.4 50.7 48.9 52.8 49.9 44.8 49.0 46.1 50.2 47.8 45.6 45.5 47.0 51.6 53.8 52.3
[69] 48.6 50.2 51.4 53.2 52.5 45.1 54.0 49.3 47.5 55.8 47.2 45.6 47.7 45.6 51.7 52.7 50.9
[86] 51.3 49.6 52.3 50.5 49.9 48.4 44.6 50.0 48.8 49.7 44.8 47.7 50.2 49.6 50.4

sum(res!=rowSums(DF))
[1] 0

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

res
```

## 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)

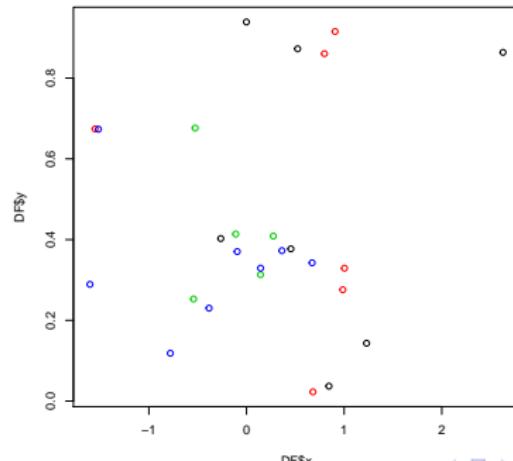
class(res)
[1] "list"
head(unlist(res))
[1] 1.00 1.41 1.73 2.00 2.24 2.45
```

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

# Basic Plots

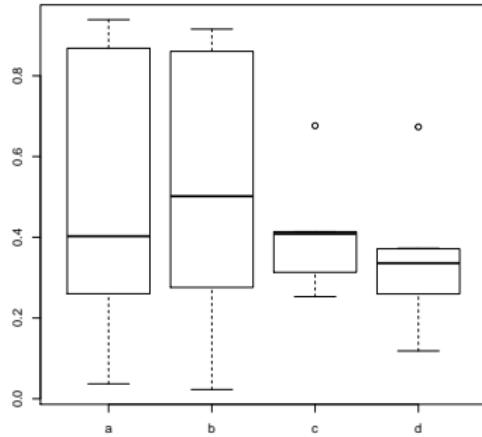
R has amazing plotting functionality, beyond the scope of this course to go in too much detail. Base-R provides simple plotting functionality, advanced and fancy plots are possible with the ggplot package. Base-R tries to guess the appropriate plotting function dependent on the data type.

```
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(DFx,DFy, col=DF$letters)
```



# Basic Plots

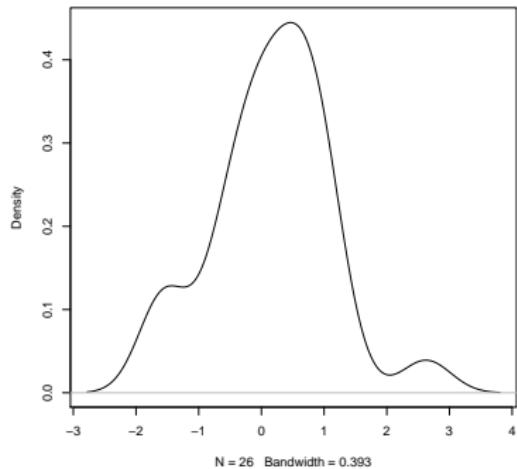
```
#letters is a factor (an encoded string), so appropriate is to draw a boxplot
plot(DF$letters,DF$y)
```



# Basic Plots

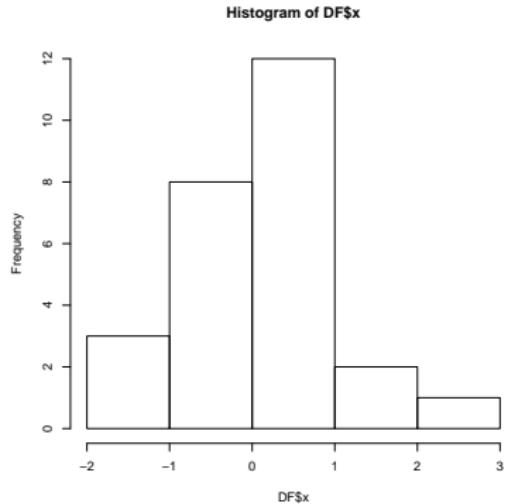
```
#kernel density
plot(density(DF$x))
```

density.default(x = DF\$x)



# Basic Plots

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



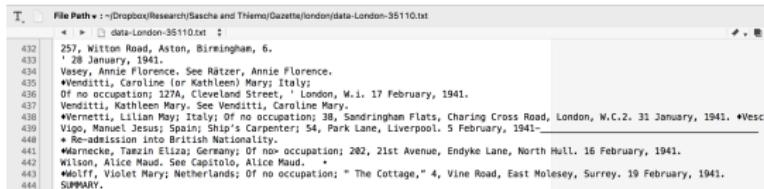
# A Texteditor is very very useful...

- ▶ Textranger (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)



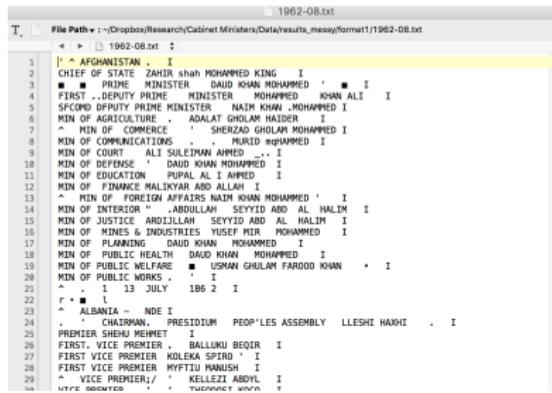
# A Texteditor is very very useful...

- ▶ 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...

- ▶ Textranger (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)



The screenshot shows a text editor window with the file path `File Path : ~/Dropbox/Research/Cabinet Ministers/Data/results_messy/format1/1962-08.txt`. The text content is a list of Afghan cabinet members and their titles from 1962, extracted from a PDF. The text is in a non-standard font and contains many errors, such as extra spaces and punctuation. The editor interface includes standard controls like back, forward, and search buttons at the bottom.

```
[1] ^ AFGHANISTAN . I
[2] CHIEF OF STATE ZAHIR shah MOHAMMED KING I
[3] ■ ■ PRIME MINISTER DAUD KHAN MOHAMMED ' ■ I
[4] FIRST ..DEPUTY PRIME MINISTER MOHAMMED KHAN ALI I
[5] SPCON OF PUBLIC TIME MINISTER NABIL KHAN MOHAMMED I
[6] MIN OF AGRICULTURE ADALAT GHOLAM HADIDI I
[7] ^ MIN OF COMMERCE SHERZAD GHOLAM MOHAMMED I
[8] MIN OF COMMUNICATIONS . MURID MOHAMMED I
[9] MIN OF COURT ALI SULEIMAN AHMED ... I
[10] MIN OF DEFENSE DAUD KHAN MOHAMMED I
[11] MIN OF EDUCATION PIR MOHAMMED I
[12] MIN OF FINANCE MALKIYAR ABD ALLAH I
[13] ^ MIN OF FOREIGN AFFAIRS NAIN KHAN MOHAMMED ' I
[14] MIN OF INTERIOR " ,ABDULLAH SEYYID ABD AL HALIM I
[15] MIN OF JUSTICE ARDIJILAN SEYYID ABD AL HALIM I
[16] MIN OF MINE & INDUSTRIES YUSEF HER MOHAMMED I
[17] MIN OF PLANTATION DAUD KHAN MOHAMMED I
[18] MIN OF PUBLIC HEALTH DAUD KHAN MOHAMMED I
[19] MIN OF PUBLIC WELFARE ■ USMAN GHULAM FAROOQ KHAN I
[20] MIN OF PUBLIC WORKS . I
[21] ^ . I 13 JULY 1962 2 I
[22] ^ .
[23] ^ ALBANIA - NDE I
[24] . ' CHAIRMAN PRESIDIUM PEOPLES ASSEMBLY LLESHE HAXHI . I
[25] PREMIER SHEHU MEHMET I
[26] FIRST VICE PREMIER . BALLUKU BEQIR I
[27] FIRST VICE PREMIER KOLEKA SPIRO ' I
[28] FIRST VICE PREMIER MYFTIJA BESI ' I
[29] ^ VICE PREMIER/ ' KELLEZI ABDYL I
[30] UTYPE DIRECTED TURKANNY VARNI T
```

## 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 Textwrangler 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
> |
```