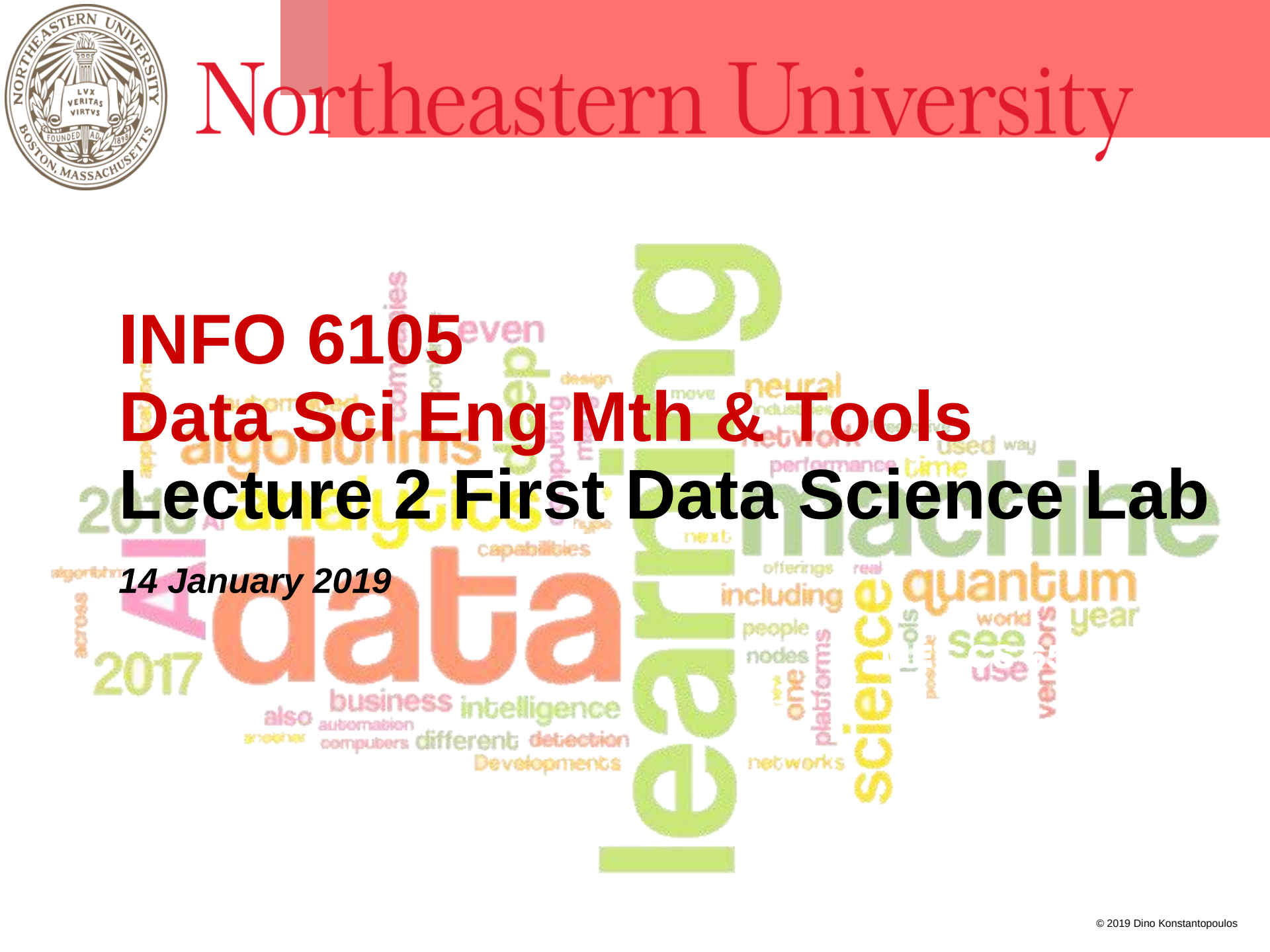


Northeastern University

INFO 6105
Data Sci Eng Mth & Tools
Lecture 2 First Data Science Lab

14 January 2019

Part 1

WHAT IS DATA SCIENCE?

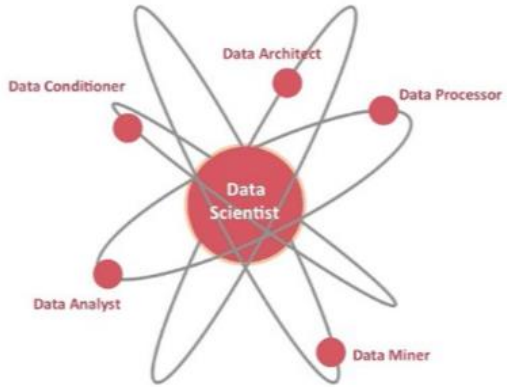
What is Data Science

- Using automated methods to analyze sometimes massive amounts of data for the purpose of extracting knowledge learned from derived results
- Extracting resulting insights used to understand and improve:
  - Business
  - Investment and finance
  - Research
  - Health and lifestyle
  - etc

3

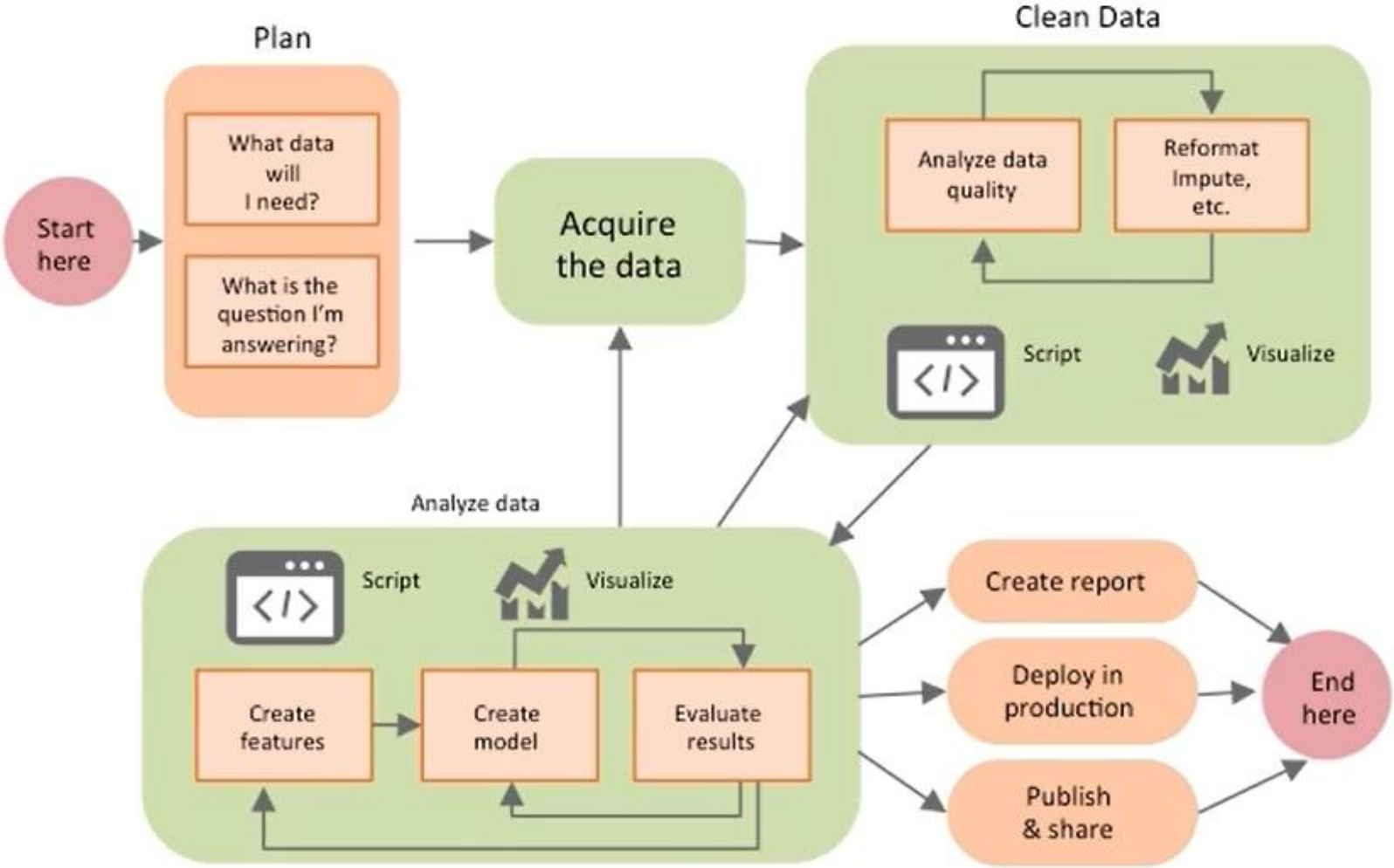
Data Science skillset

- ***Statistician***
  - **Individual who models and summarizes datasets**
- ***Domain expertise***
  - **Subject matter expert**
- ***Mathematician***
  - **Linear Algebra the lingua franca of Deep Learning**
- ***Computer scientist***
  - **Writes algorithms to summarize, visualize, and store data**



4

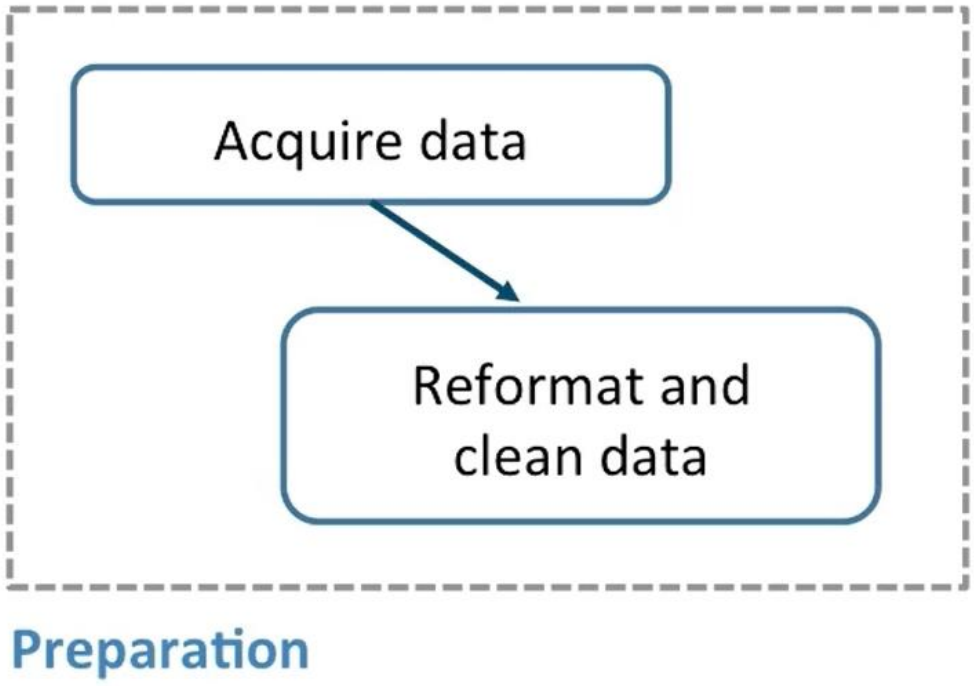
Data Science Process Flow



5

Stage 1 – Planning/Preparation

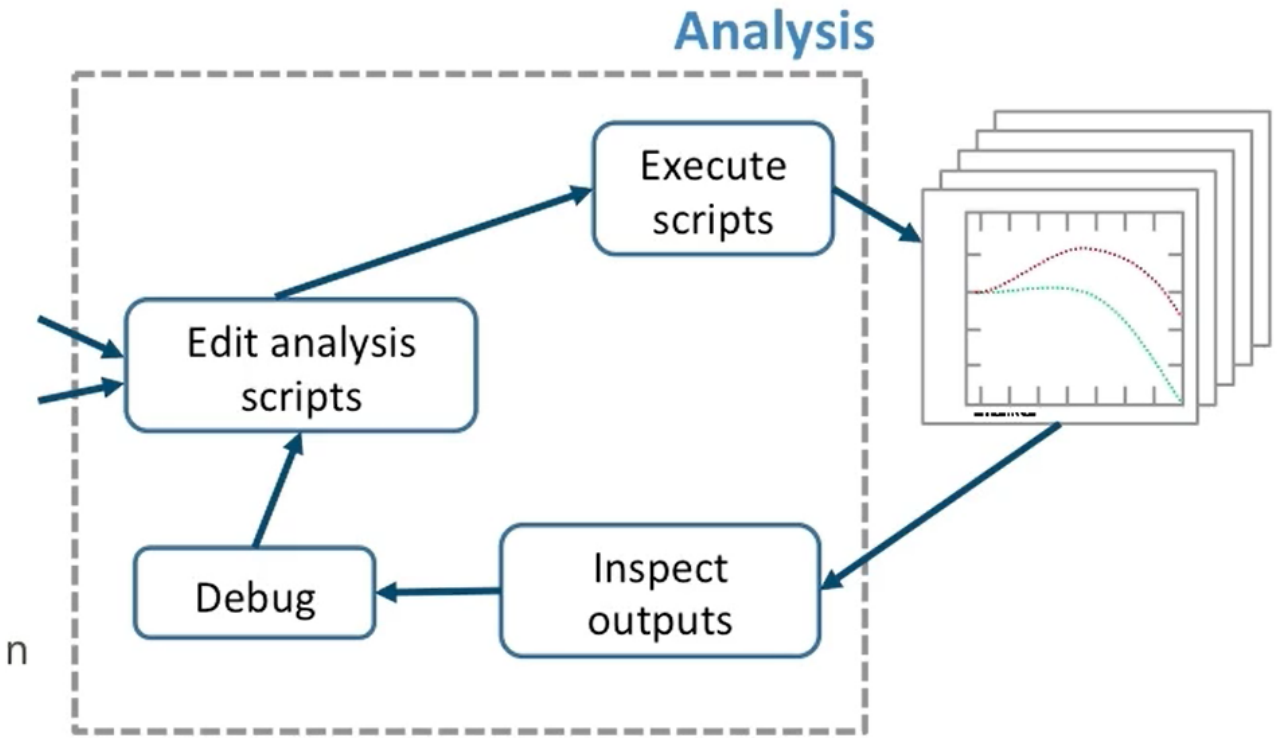
- **Obtain data**
- **Format and clean data**



6

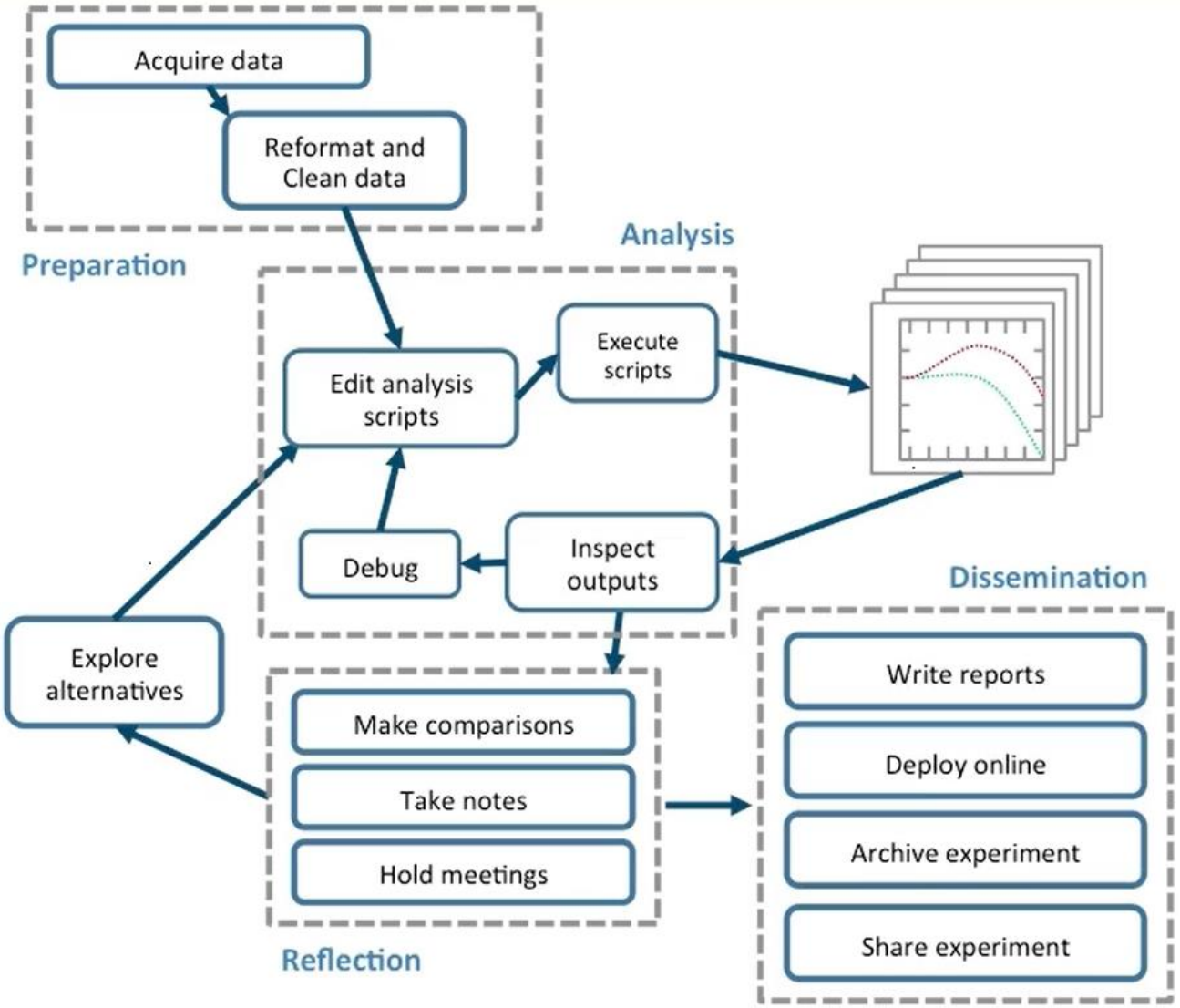
Stage 2 - Analysis

- Create scripts for analysis
- Run scripts
- Debug
- Inspect output
- Debug
- Repeat

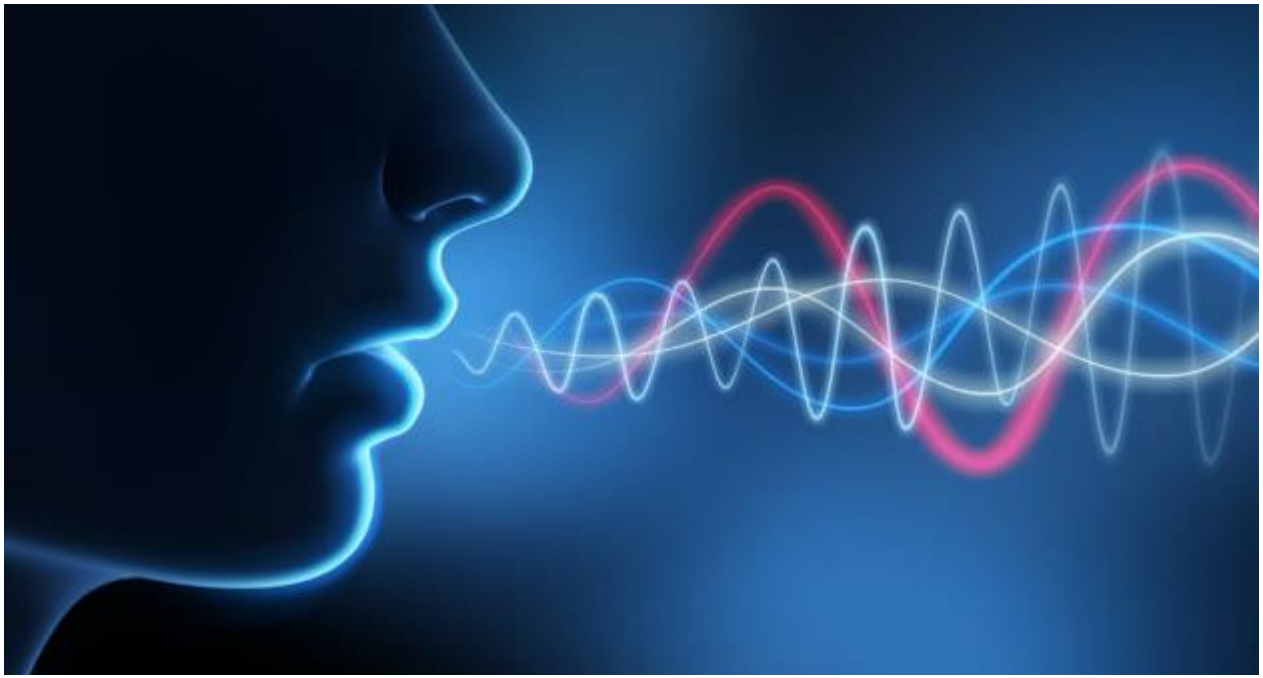


7

Stage 3 - Publish



8



Part 2

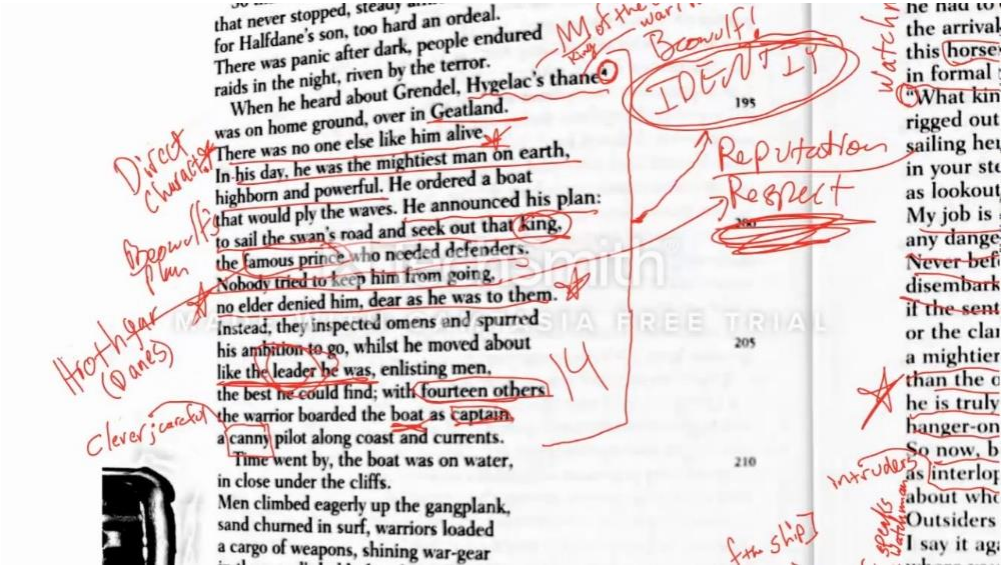
NATURAL LANGUAGE PROCESSING (NLP)

What is NLP?

- *Natural Language Processing* (NLP) is a field of computer science and engineering that has developed from the study of language and computational linguistics within the field of Artificial Intelligence
- The goals of NLP are to design and build applications that facilitate human interaction with machines through the use of natural language
  - Document classification, machine translation, and speech recognition are major areas of NLP
- 2018 was a milestone year for NLP and machines, because machines achieved human parity in translation (all languages) *and* speech to text
  - In other words, machine perform *as well as* the best human experts in the field

Annotations

- It is not enough to simply provide a computer with a large amount of data and expect it to learn to speak—the data has to be prepared in such a way that the computer can more easily find patterns and inferences
- This is usually done by adding relevant *metadata* to a dataset. Any metadata tag used to mark up elements of the dataset is called an ***annotation*** over the input



11

What kinds of Annotations?

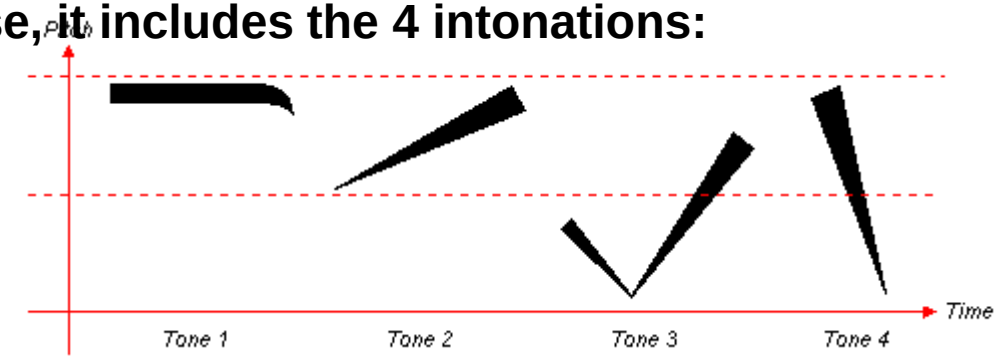
- ***Grammar*** is the name typically given to the mechanisms responsible for creating well-formed structures in language
- Most linguists view grammar as itself consisting of distinct modules or systems, either by cognitive design or for descriptive convenience These areas usually include:
- **Syntax**
  - **The study of how words are combined to form sentences. This includes examining parts of speech and how they combine to make larger constructions ("*I love you*", "*Wo ai ni*", etc..)**
- **Semantics**
  - **The study of meaning in language. Semantics examines the relations between words and what they are being used to represent.**
- **Morphology**
  - **The study of units of meaning in a language. A morpheme is the smallest unit of language that has meaning or function, a definition that includes words, prefixes, affixes, and other word structures that impart meaning. In Chinese, it's an *ideogram***

(fire)

12

Grammar areas (continued)

- **Phonology**
  - The study of the sound patterns of a particular language
  - Aspects of study include determining which patterns are significant and have meaning (i.e., the phonemes); how syllables are structured and combined; and what features are needed to describe the discrete units (segments) in the language, and how they are interpreted
  - In Chinese, it includes the 4 intonations:



- **Phonetics**
  - The study of the sounds of human speech, and how they are made and perceived. A phoneme is the term for an individual sound, and is essentially the *smallest* unit of human speech.

Grammar areas (continued)

- **Lexicon**
  - **The study of the words and phrases used in a language, that is, a language's vocabulary.**
- **Discourse analysis**
  - **The study of exchanges of information, usually in the form of conversations, and particularly the flow of information across sentence boundaries.**
- **Pragmatics**
  - **The study of how the context of text affects the meaning of an expression, and what information is necessary to infer a hidden or presupposed meaning.**
- **Text structure analysis**
  - **The study of how narratives and other textual styles are constructed to make larger textual compositions**

Reference: *adapted from* **Natural Language Annotation for Machine Learning**

https://www.safaribooksonline.com/library/view/natural-language-annotation/9781449332693/

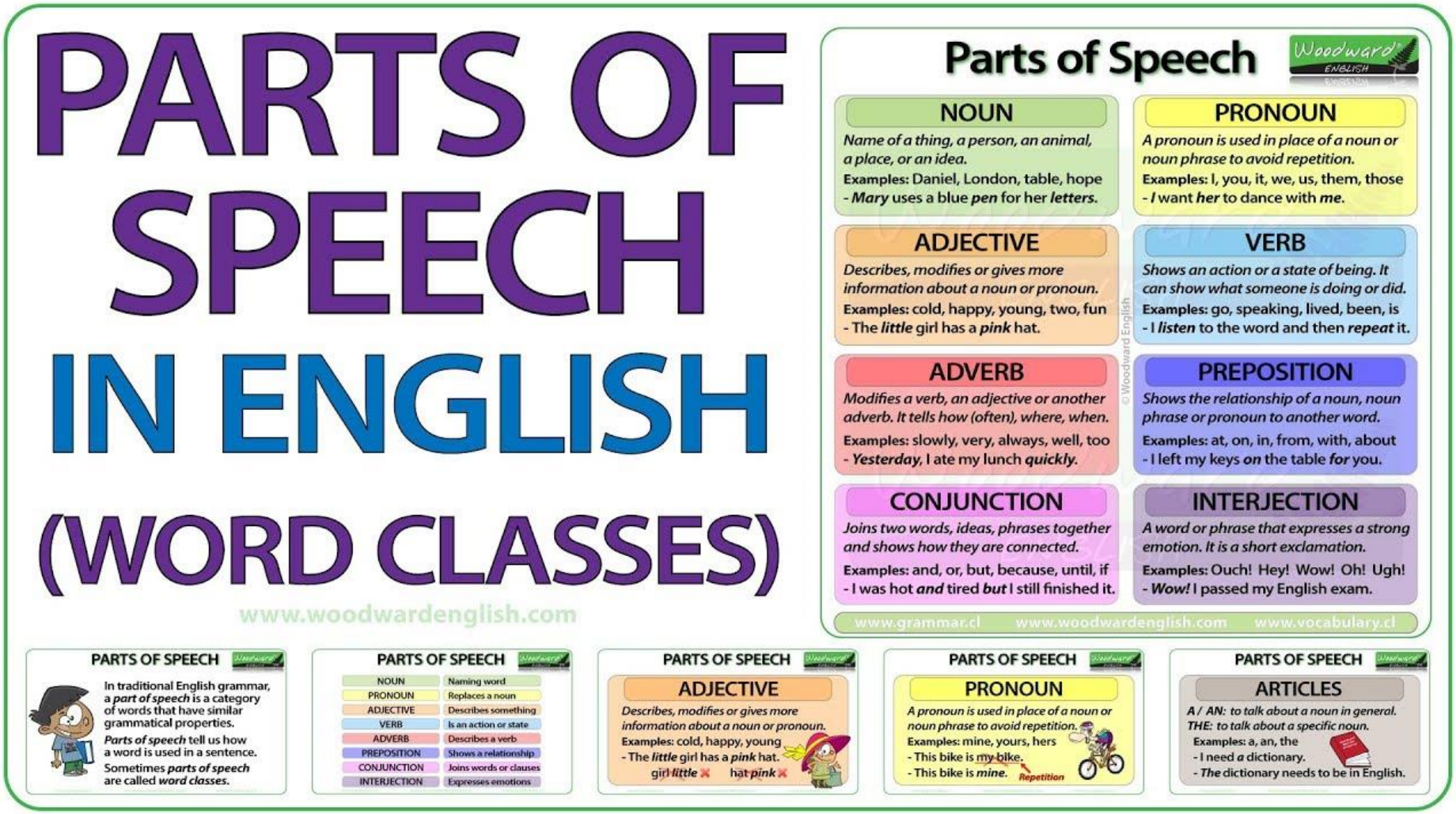
14

Parts of Speech

- In traditional grammar, a **part of speech** (abbreviated as **PoS** or **POS**) is a category of words that have similar grammatical properties
- Words that are assigned to the *same* POS generally display *similar* behavior in terms of syntax—they play similar roles within the grammatical structure of sentences—and sometimes in terms of morphology, in that they undergo inflection for similar properties
- Commonly listed *English* parts of speech are *noun*, *verb*, *adjective*, *adverb*, *pronoun*, *preposition*, *conjunction*, and *interjection*

15

Parts of Speech

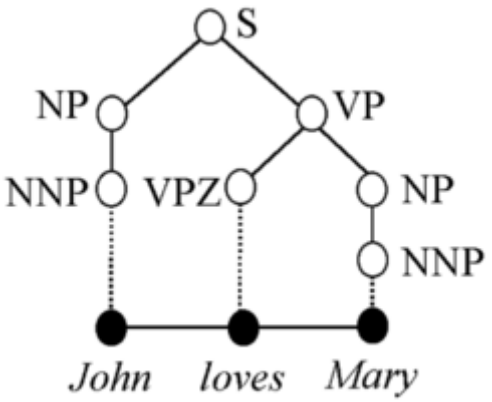


The Penn Treebank

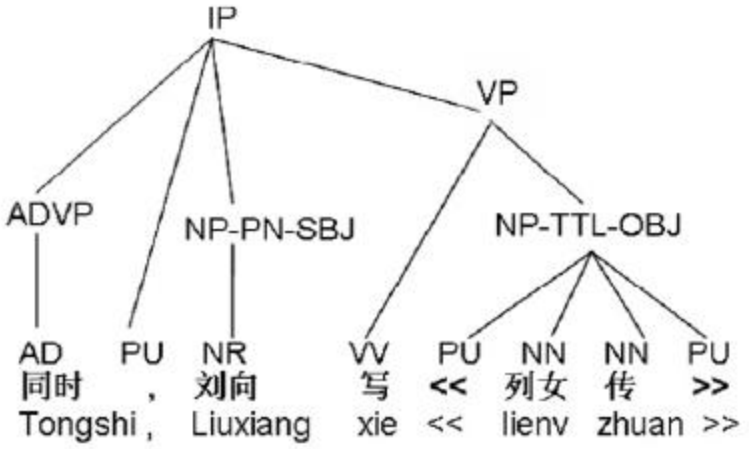
- **In linguistics, a treebank is a parsed text corpus that annotates syntactic or semantic sentence structure**
  - ***Syntax*** **(**from ancient greek: σύνταξις "coordination“) **is the set of rules, principles, and processes that govern the structure of sentences in a given language, usually including word order**
  - ***Semantics*** **(**from ancient greek: σημαντικός, "significant") **is the linguistic and philosophical study of meaning, in language. It is concerned with the relationship between signifiers—like words, phrases, signs, and symbols—and what they stand for, their denotation**
- **The most famous treebank for the English language is the Penn Treebank, which in its eight years of operation (1989-1996), produced approximately 7 million words of part-of-speech tagged text, 3 million words of skeletally parsed text, and over 2 million words of text parsed for structure**
  - **Material annotated includes such wide-ranging genres as IBM computer manuals, nursing notes, Wall Street Journal articles, and transcribed telephone conversations**
- **https://catalog.ldc.upenn.edu/ldc99t42**

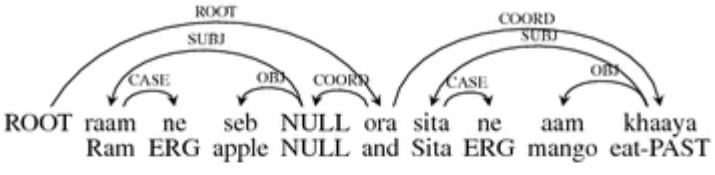
Languages have their own treebanks

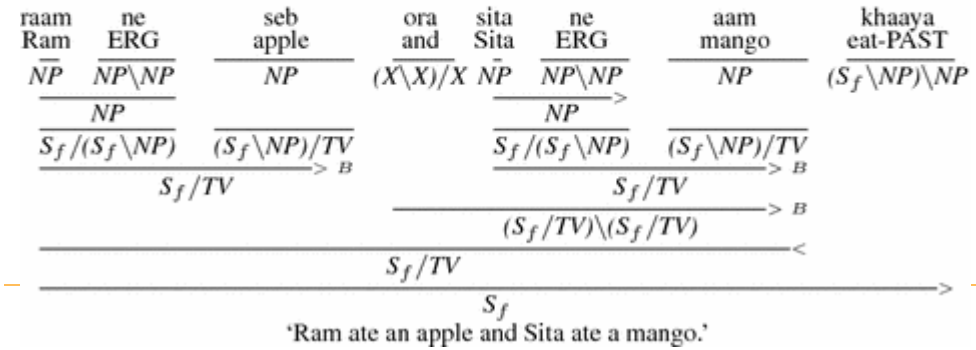
- **There are treebanks for each language of the world**
  - **What is *yours?***



bracketed sentence:
(IP (ADVP (AD 同时)) (PU ，) (NP-PN-SBJ (NR 刘向)) (VP (VV 写)
(NP-TTL-OBJ (PU 《) (NN 列女) (NN 传) (PU 》))))
English: At the same time, Liuxiang wrote <<The Biography of Strong-minded Women>>







'Ram ate an apple and Sita ate a mango.'

Part 3

DATA SCIENCE LAB WITH R - NLP WITH WORDCLOUDS *IN ENGLISH*

Document-Term and Term-Document Matrices

- A **document-term matrix** or **term-document matrix** is a mathematical matrix that describes the frequency of terms that occur in a collection of documents
- In a document-term matrix, rows correspond to documents in the collection and columns correspond to terms
- In a term-document matrix, rows correspond to terms, and columns to documents
- There are various schemes for determining the value that each cell in the matrix should take
  - In our lab and for now, we are just going to count occurrences

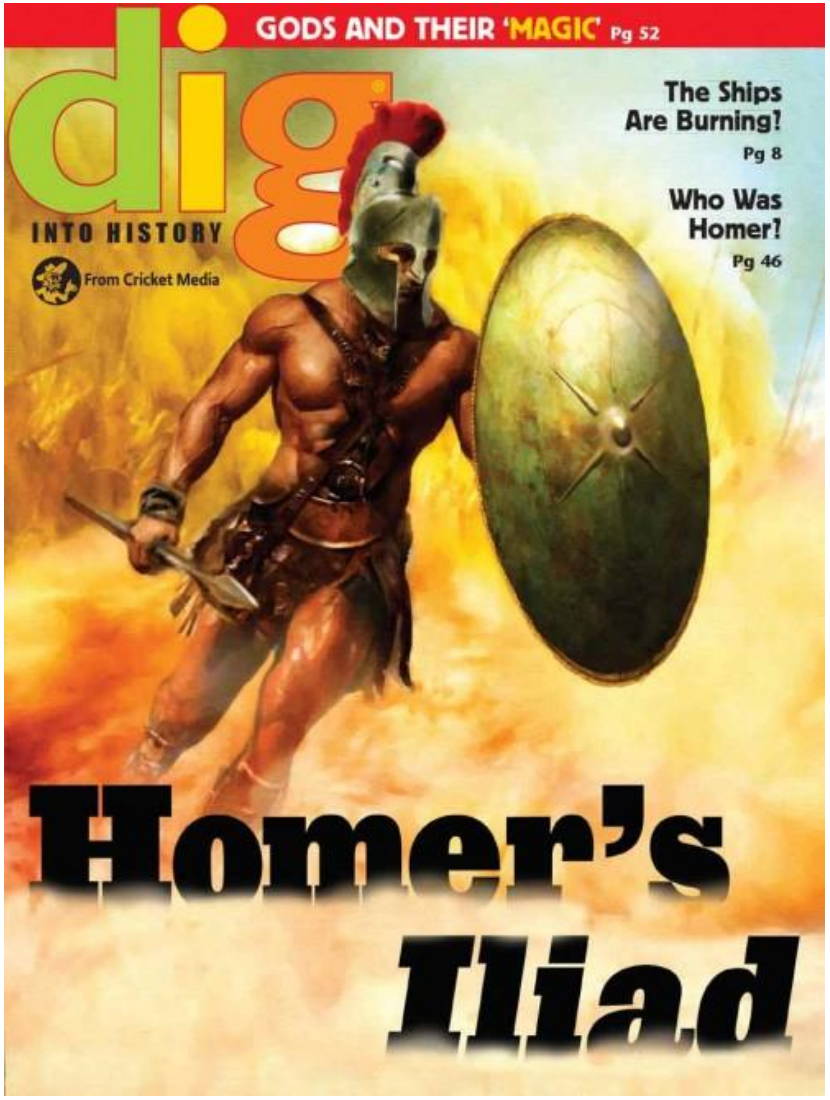
21

tf-idf

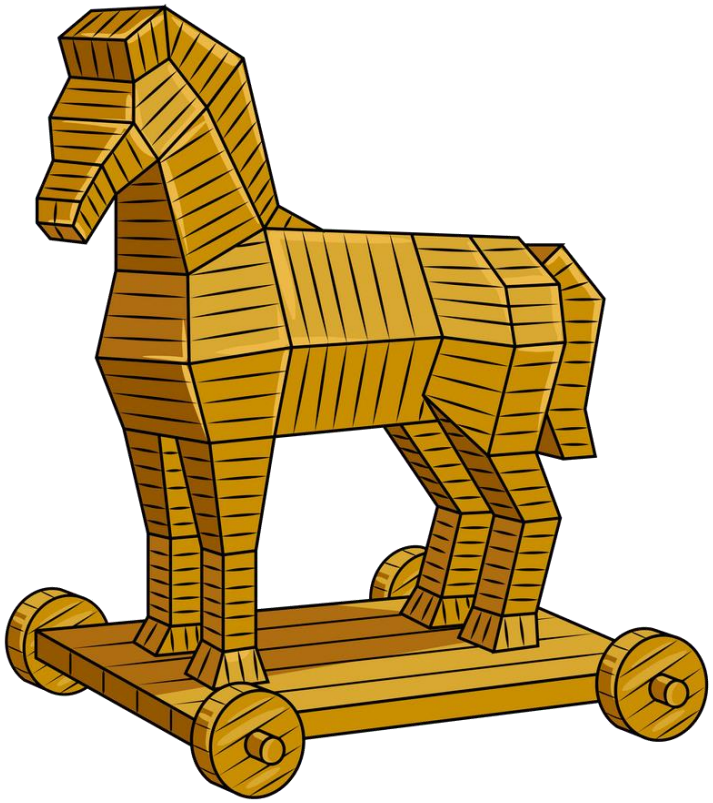
- **Stands for *Term Frequency – Inverse Document Frequency***
  - A numerical statistic that is intended to reflect how important a word is to a document in a collection of documents (also called a *corpus*)
- The tf-idf value:
  - Increases *proportionally* to the number of times a word appears in the document
  - *Offset* by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general
- So, if "**Troy**" appears very often in the document "Iliad", then "**Troy**" is a good descriptor for the document
- ..But so is the word "**the**"..
- However, the word "**the**" appears in all documents a lot
- And so its tf-idf should be low, while that of "**Troy**" high..
- Variations of the tf–idf weighting scheme are often used by search engines as a central tool in scoring and ranking a document's relevance given a user query

22

Homer's iliad



- A War in antiquity
  - Athens vs Troy
  - And lessons in humanity



23

Odyssey

- **Ulysses' return trip to Ithaca**

24

R packages for text mining

- **Our main text mining package:**
  - `tm` **(text mining)**
  - **https://cran.r-project.org/web/packages/tm/index.html**
- **Other packages we are going to use:**
  - `SnowballCC, RColorBrewer, ggplot2, wordcloud, biclust, cluster, igraph, fpc`
- **Package references on CRAN:**
  - **https://cran.r-project.org/web/views/NaturalLanguageProcessing.html**

25

Loading texts

- Start by saving your text files in a folder titled `texts` This will be the "corpus" (body) of texts you will be mining
  - You are going to work with *Homer*'s *Iliad* and the *Odyssey*, some of the oldest literature about Wars
  - You are going to also to work in your own language, so get started in English for now, but plan ahead..
- Download corpus from blackboard
- Mac:
  - `cname <- file.path("~", "Desktop", "texts")`
  - `cname`
- Windows:
  - `cname <- file.path("C:", "texts")`
  - `cname`
- Read your documents in the R terminal with `inspect(docs)`
- if you prefer to look at only one of the documents you loaded, then you can specify `inspect(docs[1])`

26

Preprocessing

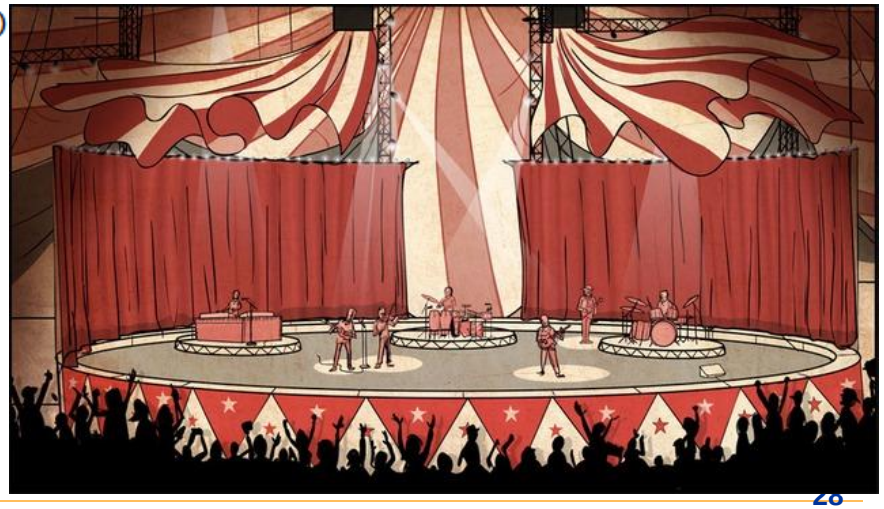
- **Remove numbers, capitalization, common words, punctuation, and prepare your texts for analysis**
  - **Remove punctuation, numbers, stopwords, common word endings, strip whitespace, convert to lower case, and combine words that should stay together**
- **Be sure to use the following script once you have completed preprocessing**
  - **This tells R to treat your preprocessed documents as text documents**
  - `library(tm)`
  - `docs <- Corpus(DirSource(cname))`
  - `docs <- tm_map(docs, PlainTextDocument)`

27

Data staging

- **Create a document term matrix and a term document matrix**

```
> dtm <- DocumentTermMatrix(docs)
> dtm
<<DocumentTermMatrix (documents: 2, terms: 11050)>>
Non-/sparse entries: 14535/7565
Sparsity           : 34%
Maximal term length: 35
Weighting          : term frequency (tf)
> tdm <- TermDocumentMatrix(docs)
> tdm
<<TermDocumentMatrix (terms: 11050, documents: 2)>>
Non-/sparse entries: 14535/7565
Sparsity           : 34%
Maximal term length: 35
Weighting          : term frequency (tf)
```



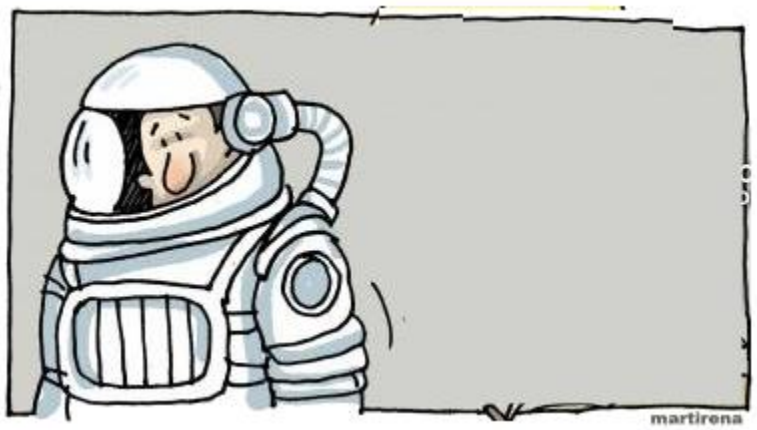
28

Data Exploration

- Organize terms by their frequency
  - `freq <- colSums(as.matrix(dtm))`
  - `length(freq)`
  - `ord <- order(freq)`
- Remove sparse terms
  - `dtms <- removeSparseTerms(dtm, 0.1)` **# This makes a matrix that is 10% empty space, maximum**
  - `inspect(dtms)`

```
                Terms
Docs             god now one said shall son thi thus ulyss will
  content        626 573 348  211   511 420 943  621   123  188
  meta           267 303 573  483   217 336   3  132   646  745
```

- *Least* frequently occurring words
  - `freq[head(ord)]`
- *Most* frequently occurring words
  - `freq[head(ord)]`



Frequency of frequencies

- Output is two rows of numbers, one per document
  - The top number is the frequency with which words appear
  - The bottom number reflects how many words appear that frequently
  - How many terms appear only once?
- Lowest word frequences:
  - `head(table(freq), 20)`
  - considering only the 20 *lowest* word frequencies, we can see that ??? terms appear only once
- Highest word frequencies:
  - `tail(table(freq), 20)`
  - Considering only the 20 greatest frequencies, there's usually a big disparity in how frequently some terms appear

30

Coarse-grained term frequencies

- For a less fine-grained look at term frequency, view a table of sparse terms:
  - `freq <- colSums(as.matrix(dtms))`
  - `freq`

```
> freq <- colSums(as.matrix(dtms))
> freq
     abandon         abat        abhor         abid          abl         abod       abound
           4            2            3            5           42           50            7
      abridg       abroad       absenc       absent       absolut       absurd         abus
           2            8           31           12            3           10            5
      accept       access     accident     accommod     accompani    accomplish       accord
          32           27            3            2           12           14           40
     account        accur      accustom          ach       achill      acquaint      acquiesc
```

- Alternatively:
  - `freq <- sort(colSums(as.matrix(dtm)), decreasing=TRUE)`
  - `head(freq, 40)`

```
head(freq, 40)
 thi    will     one     god     now   ulyss     son   thus   shall    said   great    hand     arm    jove
 946     933     921     893     876     769     756     753     728     694     686     653     625     617
 man  heaven     day     may     men     let     see   come  hector     can    ship    like     war  achill
 595     567     534     513     499     496     489     469     461     453     450     447     440     436
hous    troy     yet   chief   first  father   greek     eye    upon  trojan     oer    make
 429     425     420     417     414     408     395     391     391     368     364     363
```


31

Coarse-grained term frequencies

- **All terms that appear frequently (250 or more times):**
- **`findFreqTerms(dtm, lowfreq=250)`**

```
> findFreqTerms(dtm,lowfreq=250)
 [1] "achill"    "arm"       "back"      "came"      "can"       "care"      "chief"
 [8] "come"      "day"       "dead"      "death"     "even"      "everi"     "eye"
[15] "fall"      "fate"      "father"    "field"     "fight"     "fire"      "first"
[22] "fli"       "forc"      "friend"    "give"      "god"       "good"      "great"
[29] "greek"     "ground"    "hand"      "head"      "heart"     "heaven"    "hector"
[36] "hero"      "high"      "home"      "hous"      "jove"      "king"      "know"
[43] "let"       "lie"       "like"      "long"      "made"      "make"      "man"
[50] "may"       "men"       "much"      "must"      "now"       "oer"       "old"
[57] "one"       "place"     "plain"     "power"     "rage"      "round"     "said"
[64] "say"       "sea"       "see"       "shall"     "ship"      "son"       "stand"
[71] "still"     "suitor"    "take"      "telemachus" "tell"     "thi"       "thou"
[78] "though"    "thus"      "till"      "time"      "trojan"    "troy"      "two"
[85] "ulyss"     "upon"      "vain"      "war"       "way"       "went"      "whose"
[92] "will"      "word"      "work"      "yet"
```

Alternatively:

- **`wf <- data.frame(word=names(freq), freq=freq)`**
- **`head(wf)`**

```
> head(wf)
      word freq
thi    thi  946
will  will  933
one    one  921
god    god  893
now    now  876
ulyss ulyss  769
```


32

Plotting with ggplot2

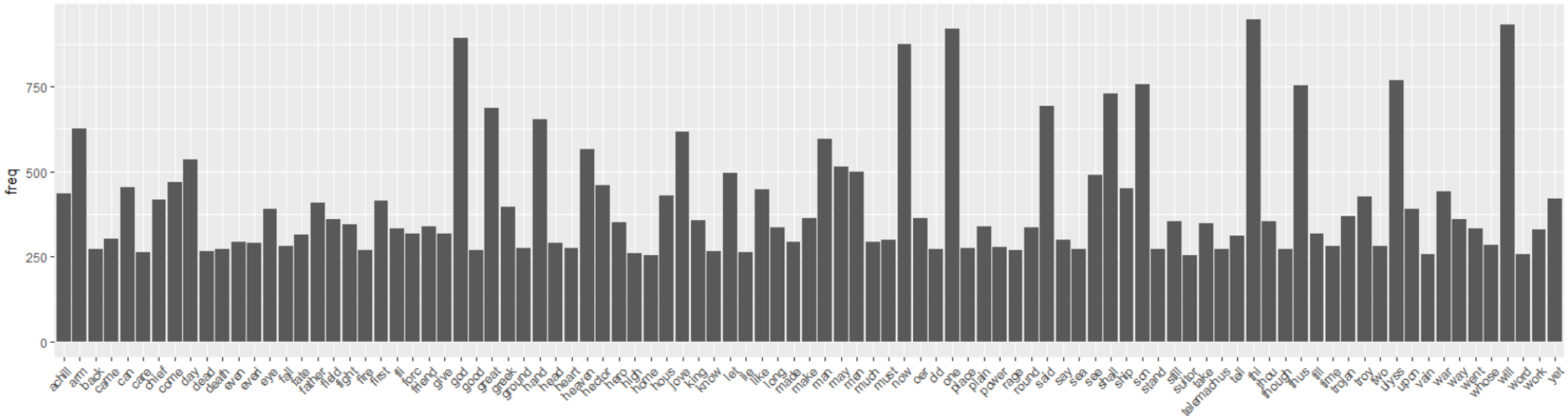
- `library(ggplot2)`
- ```
  p <- ggplot(subset(wf, freq>50), aes(word, freq))
  p <- p + geom_bar(stat="identity")
  p <- p + theme(axis.text.x=element_text(angle=45, hjust
  =1))
  p
  ```


33
  ```

Fixing potential ggplot2 error

- `library(ggplot2)`
  - **Error in loadNamespace(i, c(lib.loc, .libPaths()), versionCheck = vI[[i]]) : namespace ‘scales’ 0.4.0 is being loaded, but >= 0.4.1 is required**
    Error: package or namespace load failed for ‘ggplot2’
- **Then look at the complete error message. It lists a couple of packages which are missing or have errors. Uninstall and reinstall them. So for me:**
  - ```
    remove.packages("ggplot2")
    install.packages('ggplot2', dependencies = TRUE)
    remove.packages("scales")
    install.packages('scales', dependencies = TRUE)
    library(ggplot2)
    ```
    ```

Words with frequencies > 250 in iliad + odyssey



35

Relationships between terms

- If you have a term in mind that you have found to be particularly meaningful to your analysis, then you may find it helpful to identify the words that most highly correlate with that term
  - If words *always* appear together, then correlation=1
  - `library(tm)`
  - `findAssocs(dtm, "achilles", corlimit=0.98) # specifying a correlation limit of 0.98`
- Can also find associations for word collections:
  - `findAssocs(dtm, c("achilles", "hector"), corlimit=0.98)`

Word Clouds

- Humans are generally strong at visual analytics
  - We are going to construct word clouds on our document corpus
- Load the package that makes word clouds in R
  - `library(wordcloud)`
- Plot words that occur at least 250 times
  - `set.seed(142)` `#The "set.seed() function just makes the configuration of the layout of the clouds consistent each time you plot them. You can omit that part if you are not concerned with preserving a particular layout`
  - `wordcloud(names(freq), freq, min.freq=250)`

37

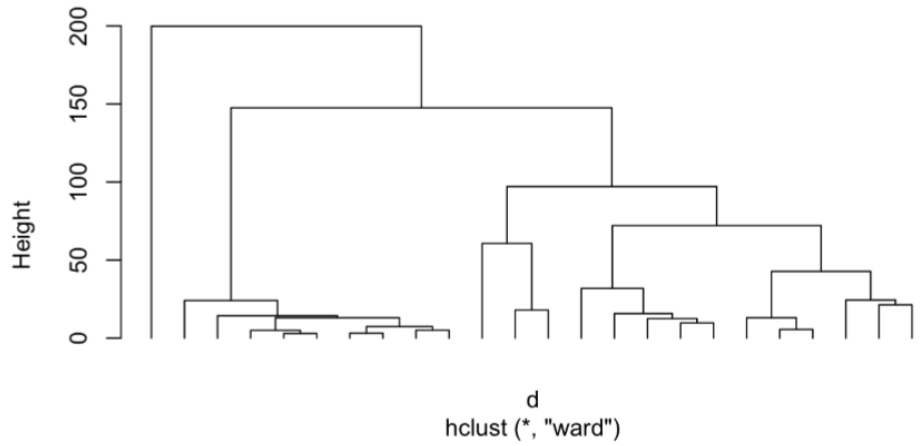
With Color

- **set.seed(142)**
  dark2 <- brewer.pal(6, "Dark2")
  wordcloud(names(freq), freq, max.words=250, rot.per=0.2, c
  olors=dark2)

38

Clustering by term similarity

- First calculate distance between words & then cluster them according to similarity:
  - `library(cluster)`
  - `dtmss <- removeSparseTerms(dtm, 0.15) # This makes a matrix that is only 15% empty space, maximum`
  - `inspect(dtmss)`
  - `d <- dist(t(dtmss),method="euclidian")`
  - `fit <- hclust(d=d, method="ward.D2")`
  - `plot(fit, hang=-1)`



39

Ask R to help identify the clusters

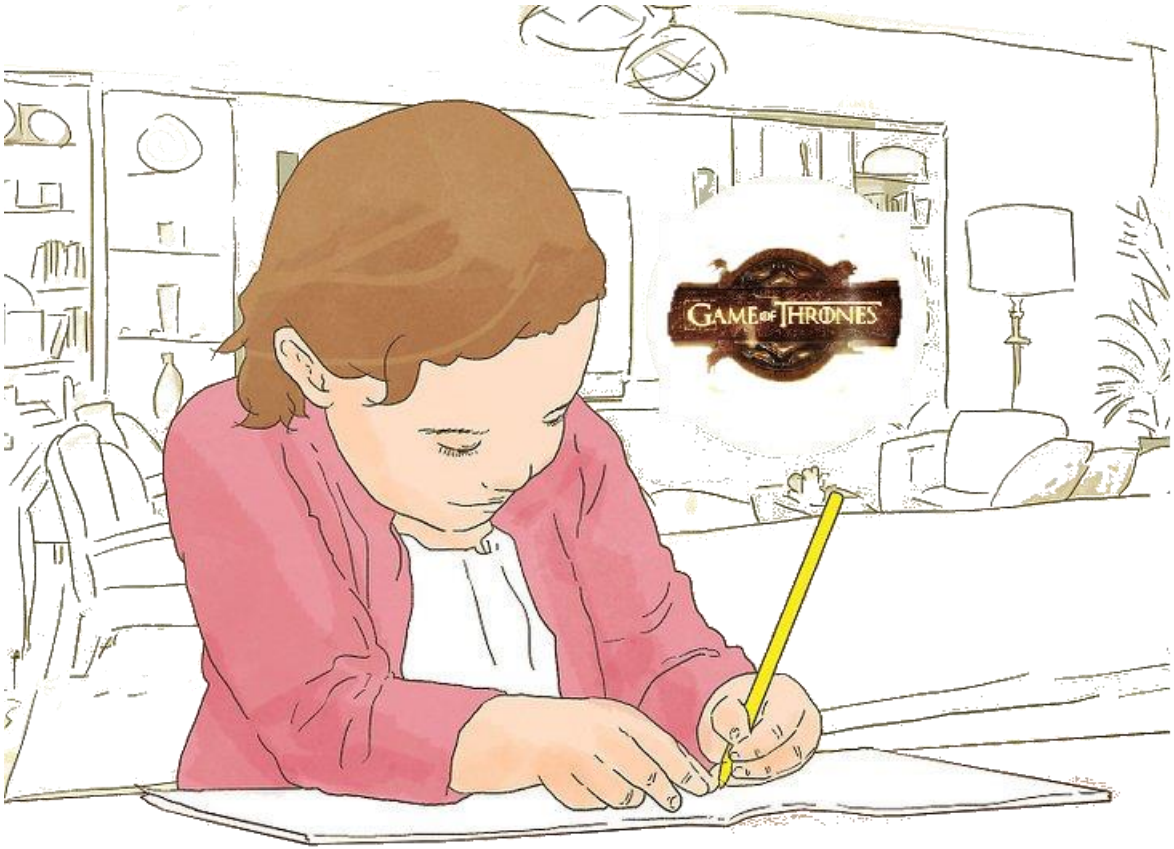
- `plot.new()`
- `plot(fit, hang=-1)`
- `groups <- cutree(fit, k=5)   # "k=" defines the number of clusters we are using`
- `rect.hclust(fit, k=5, border="red") # draw dendogram with red borders around the 5 clusters`

40

K-Means clustering

- **Cluster words into a specified number of groups (in this case 2), such that the sum of squared distances between individual words and one of the group centers is minimized**
  - `library(fpc)`
  - `d <- dist(t(dtmss), method="euclidian")`
  - `kfit <- kmeans(d, 2)`
  - `clusplot(as.matrix(d), kfit$cluster, color=T, shade=T, labels=2, lines=0)`

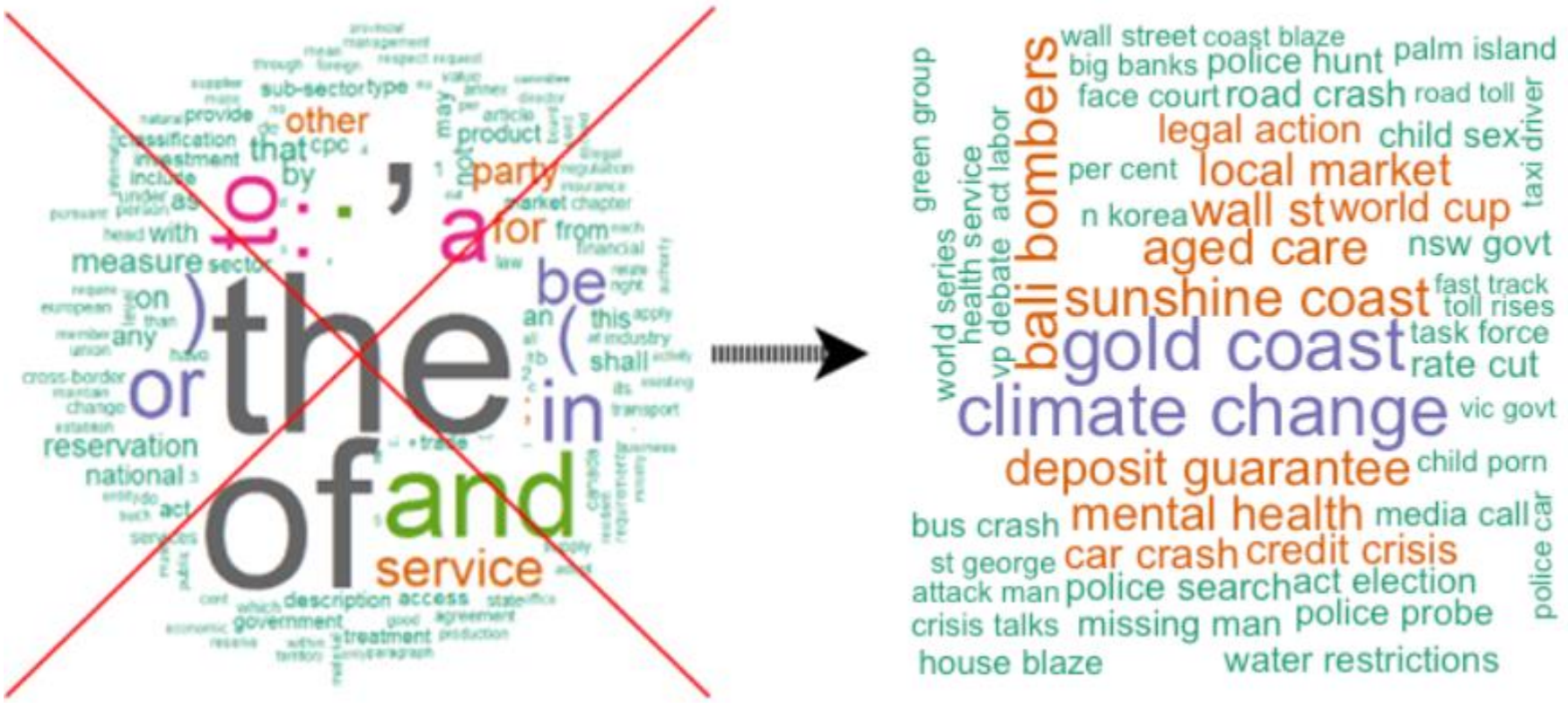
41



Part 4

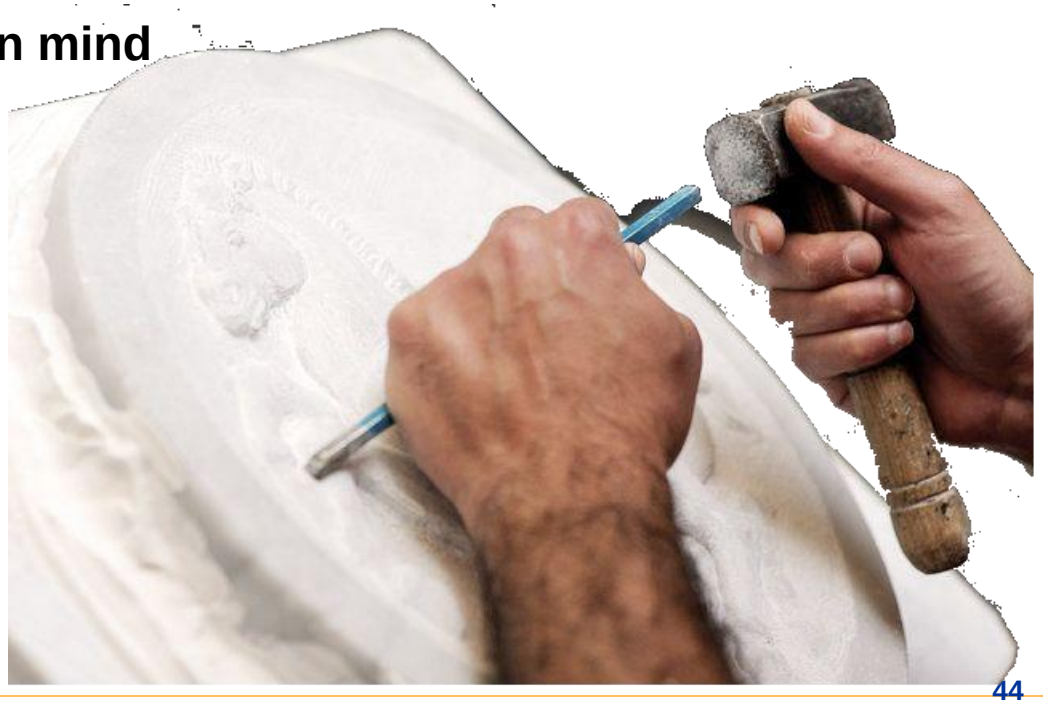
INTRODUCTION TO ADVANCED NLP: WORDCLOUDS IN *YOUR* LANGUAGE & FOR *YOUR* DATASET, WITH `UDPIPE`

Better word clouds with better data science



Versions

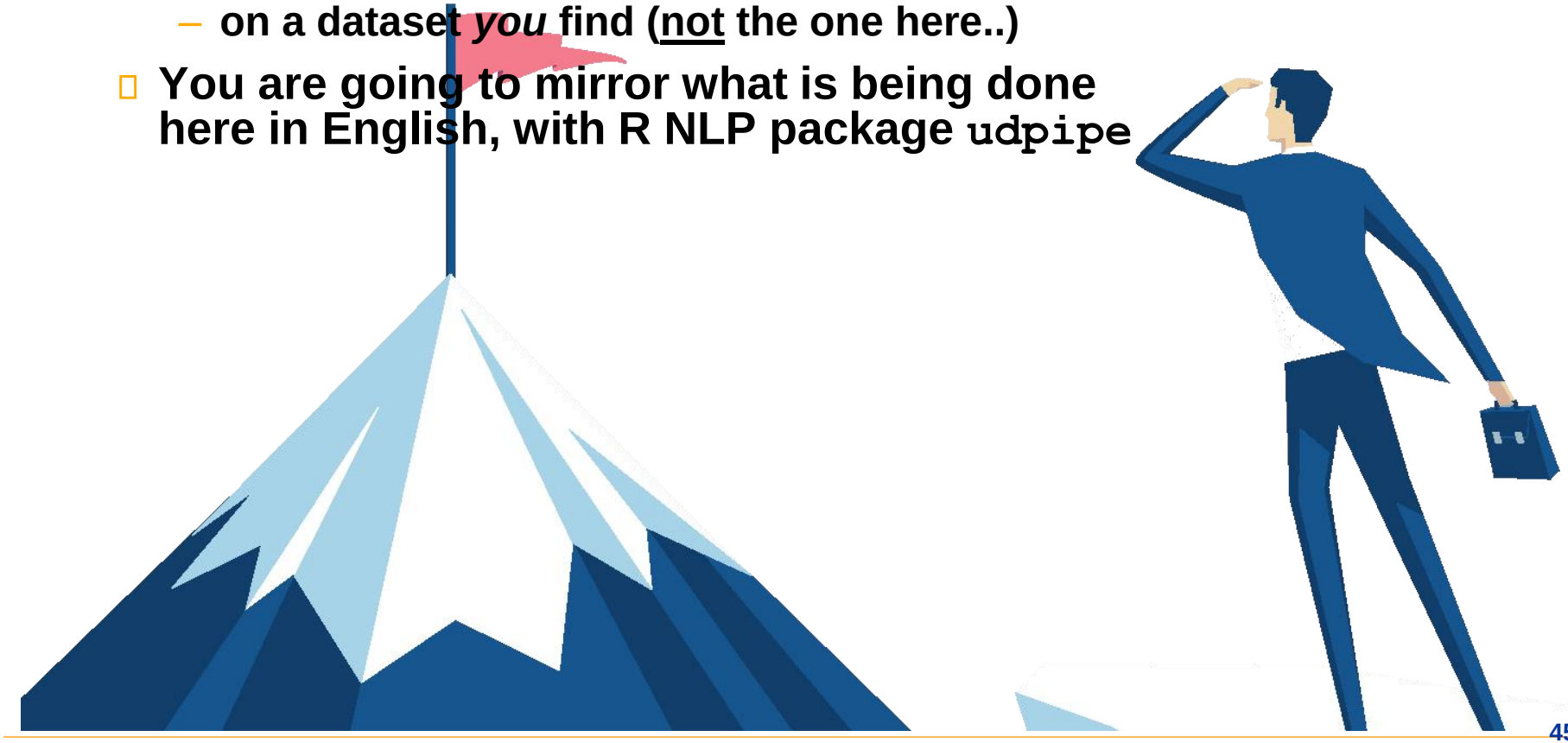
- **This lab may require the *latest* versions!**
  - **Version 3.5.0 of R runtime and 1.1.447 of RStudio**
- **This means you *may not be able to* run it with Anaconda Navigator's RStudio**
  - **You may have to download the latest R runtime and RStudio by hand..**
  - **Not sure, but keep that in mind**



44

Goal

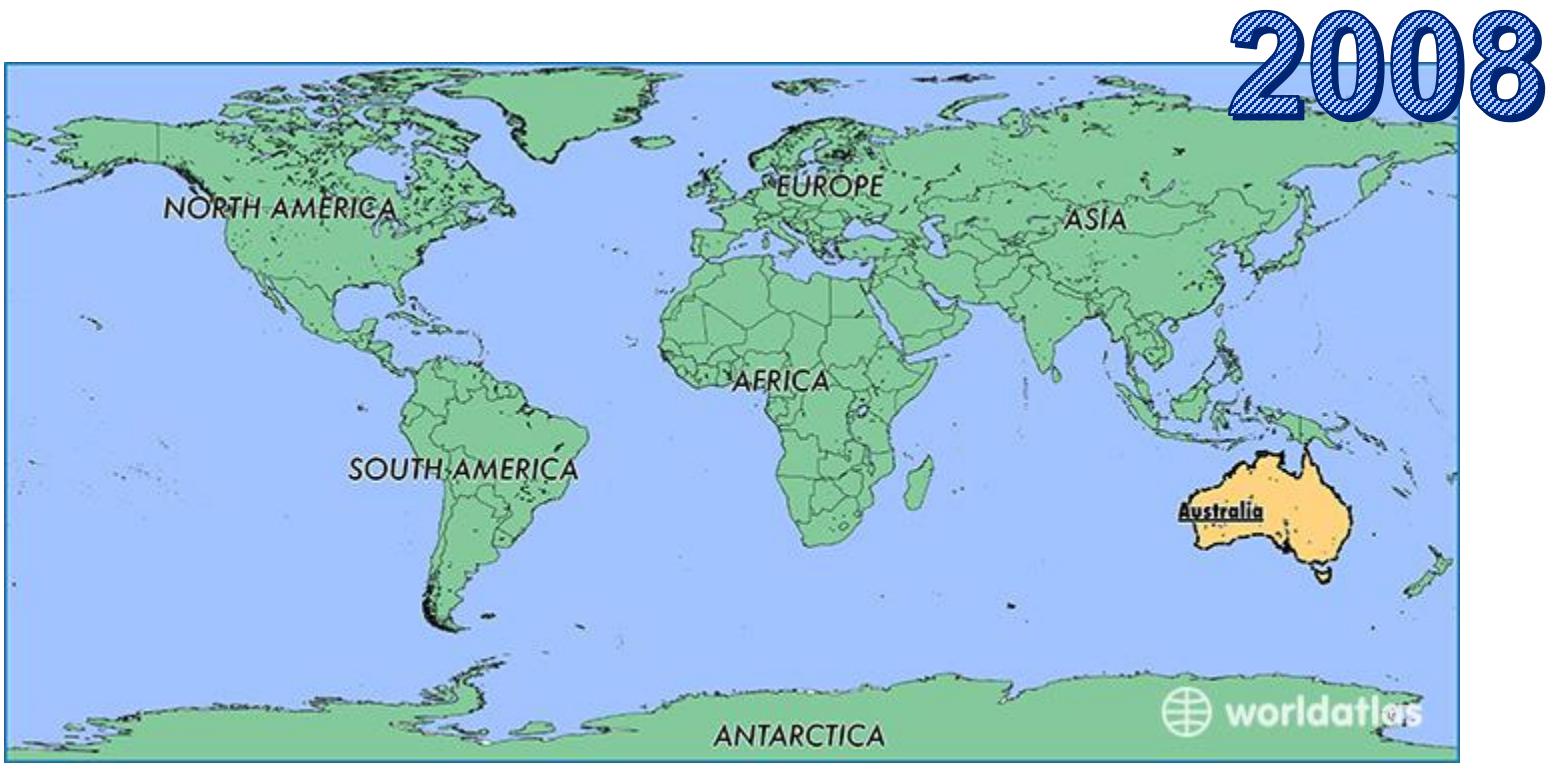
- You are going to do NLP..
  - in *your* language
  - on a topic of *your* choice
  - on a dataset *you* find (not the one here..)
- You are going to mirror what is being done here in English, with R NLP package `udpipe`



45

Ok, so what are we doing here?

- **A friend told you he dreamt about the years 2008 and Australia, and he knows you are a Data scientist, so he asks you to tell him what happened in 2008 that was noteworthy, in Australia..**



46

So you go look for datasets..

- You locate some news headlines published over a period of 15 years..
  - From Australian news source ABC (Australian Broadcasting Corp.)
  - At: http://www.abc.net.au/
- The Dataset is also available on Kaggle:
  - **https://www.kaggle.com/therohk/million-headlines**
- **Better use the version I put up on blackboard, instead (more data)**

47

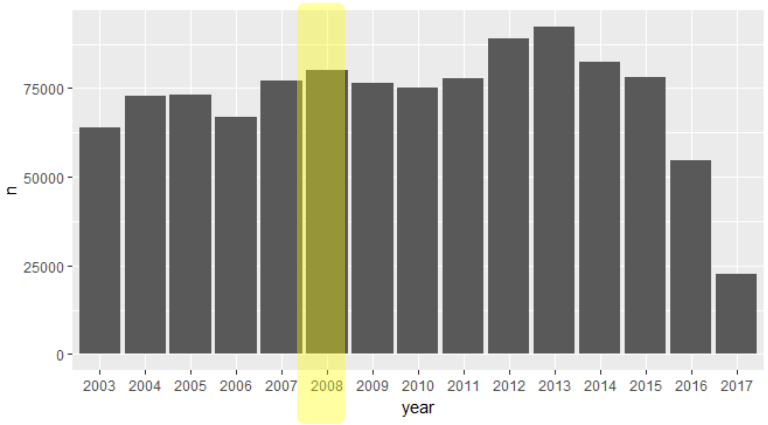
Package `udpipe`

- Then you go looking for data science libraries
  - You're still learning Python in class, so…
  - You go looking for R libraries on CRAN..
- You find.. `udpipe` provides *pretrained* language models for spoken languages (not programming languages) and you can download the required model using `udpipe_download_model()`

48

Cleaning dataset

- **Read in the data**
  - `news <- read.csv('abcnews-date-text.csv', header = T, stringsAsFactors = F)`
- **Do a quick exploration:**
  - `news %>% group_by(publish_date) %>% count() %>% arrange(desc(n))`
  - What is `%>%` ? Go to page 37 for an explanation
- **To subset data only for 2008:**
  - `news_more_2008 <- news_more %>% filter(year == 2008 & month == 10)`

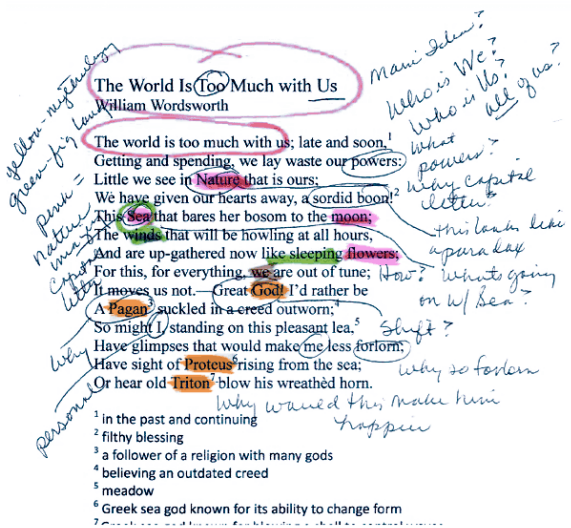


49

49

Staging with pretrained language models

```
library(udpipe)
model <- udpipe_download_model(language = "english")
udmodel_english <- udpipe_load_model(file = 'english-ud-2.0-170801.udpipe')
s <- udpipe_annotate(udmodel_english, news_more_2008$headline_text)
x <- data.frame(s)
```



The World Is Too Much with Us
William Wordsworth

The world is too much with us; late and soon,
Getting and spending, we lay waste our powers:
Little we see in Nature that is ours;
We have given our hearts away, a sordid boon![2]
This Sea that bares her bosom to the moon;
The winds that will be howling at all hours,
And are up-gathered now like sleeping flowers;
For this, for everything, we are out of tune;
It moves us not.—Great God! I'd rather be
A Pagan[3] suckled in a creed outworn;[4]
So might I, standing on this pleasant lea,[5]
Have glimpses that would make me less forlorn;
Have sight of Proteus[6] rising from the sea;
Or hear old Triton[7] blow his wreathèd horn.

[1] in the past and continuing
[2] filthy blessing
[3] a follower of a religion with many gods
[4] believing an outdated creed
[5] meadow
[6] Greek sea god known for its ability to change form

50

Package `lattice`

- Powerful and elegant high-level data visualization system inspired by Trellis graphics, with an emphasis on multivariate data
  - Sufficient for typical graphics needs, and flexible enough to handle most nonstandard requirements
  - Nice barcharts ☺
  - ```
    barchart(key ~ freq, data = head(stats, 20), col =
    "magenta", main = "Keywords-simple noun phrases",
    xlab = "Frequency")
    ```
  - https://cran.r-project.org/web/packages/lattice/index.html
    ```

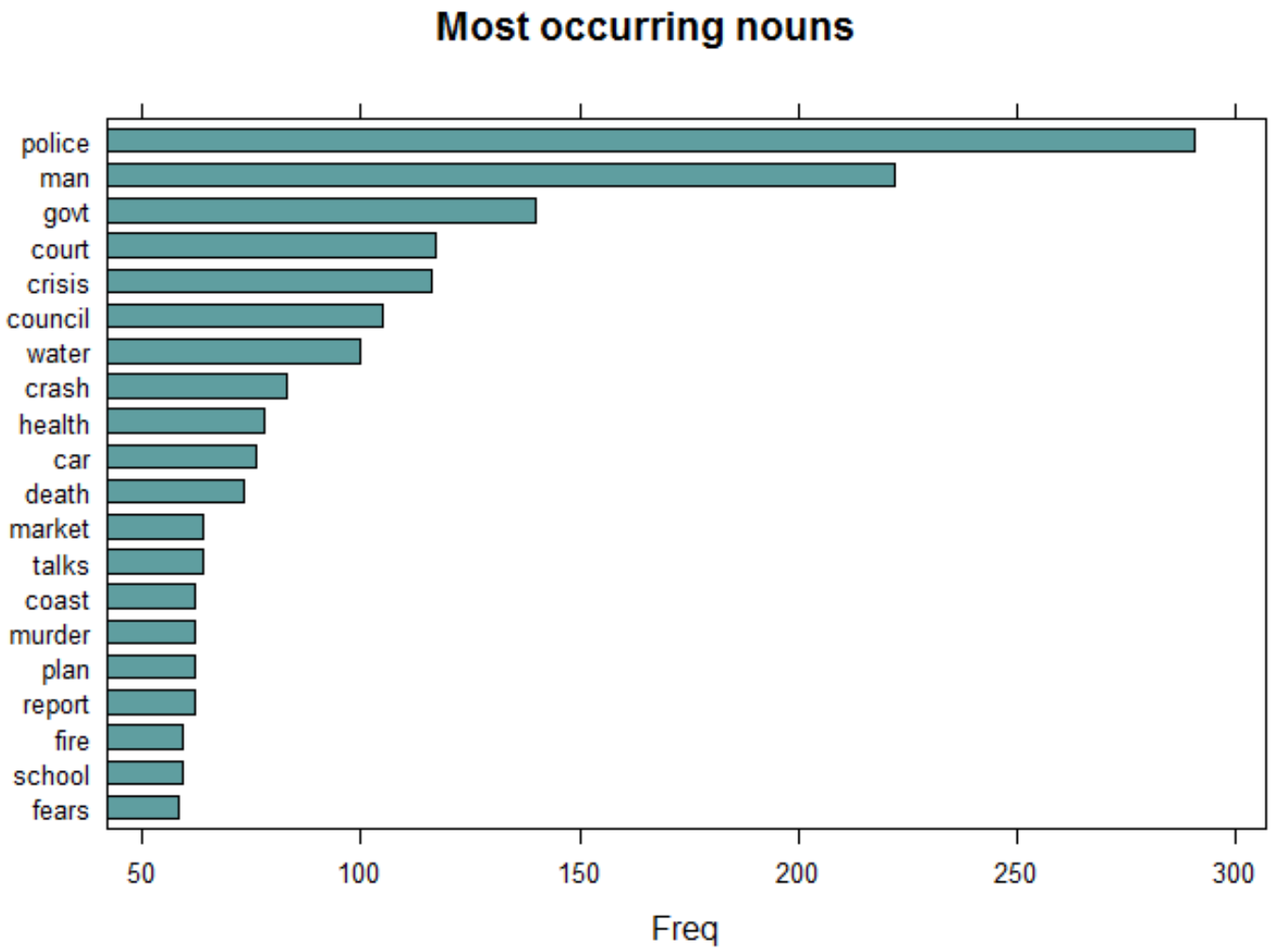
Factors in R

- Variables that take on a *limited* number of different values
  - Often referred to as *categorical* variables
- Very important in statistical modeling
  - Categorical variables enter into statistical models differently than continuous variables..
- Factors are also a very efficient way to store character values
  - Because each unique character value is stored only once, and the data itself is stored as a vector of integers
  - We'll use factors in artificial neural networks

```
data = c(1,2,2,3,1,2,3,3,1,2,3,3,1)
fdata = factor(data)
fdata
[1] 1 2 2 3 1 2 3 3 1 2 3 3 1
Levels: 1 2 3
```


52

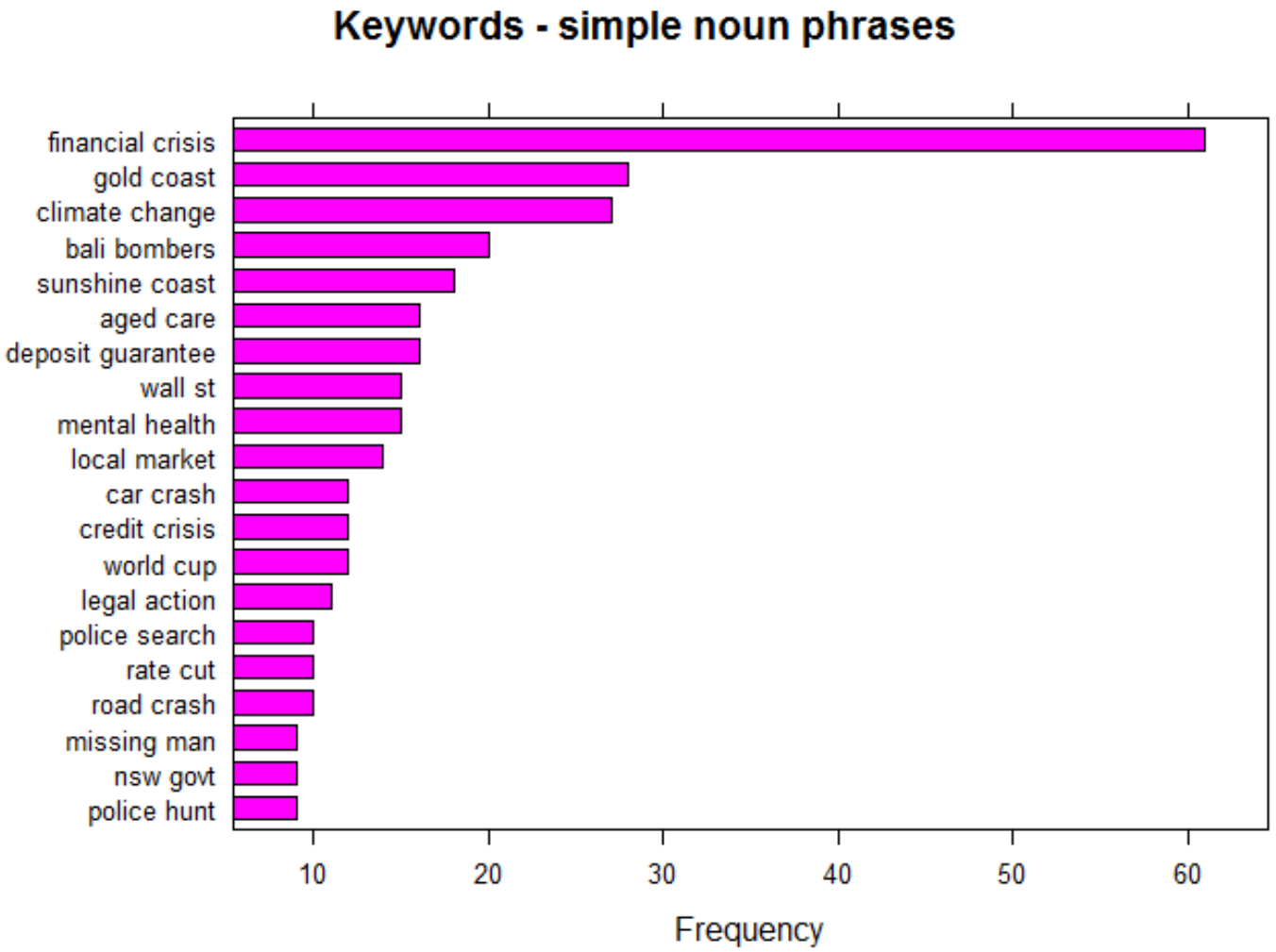
Most occurring nouns for 2008



RAKE/RKEA

- One you get good at this, you realize scientists wrote a package that already does what you're doing..
- Rapid Keyword Extraction Algorithm (RKEA) or Rapid Algorithm for Keyword Extraction (RAKE)
  - Never figured out which one it is..

Simple noun phrases



And you give your friend his word cloud..

```
library(wordcloud)
wordcloud(words = stats$key, freq = stats$freq,
min.freq = 3, max.words = 100, random.order =
FALSE, colors = brewer.pal(6, "Dark2"))
```

Labs due..

- ***First version: Next week Monday, teams of 2 students each***
- ***Second version: Next week Wednesday***
- **Code is up on Blackboard! At most *3 .R files are due***
  - **4-textmining-cloud.R**
    - **In English**
  - **5-nlp.R**
    - **In English**
    - **Adapted to *your* language (e.g. *Mandarin, Hindi, Greek*, etc.) and *your* dataset**
- **If you don't need to modify the .R file, that's great, just submit a picture of your wordclouds. If you need to modify the R file, then submit the R file as well. *For your language text, you will have to submit your R file, your dataset (in your language), and plots of your wordclouds***
- ***Your dataset does not have to be as big as the abcnews dataset, but it should not be too small, either***
- ***Best hindi & mandarin teams will present their work in class***


CLASS
DISMISSED
iStockphoto