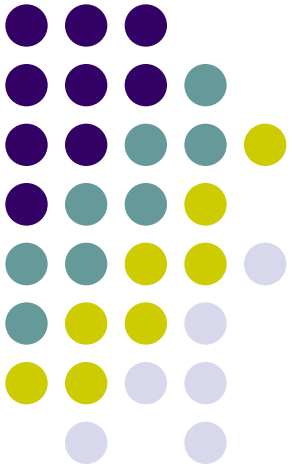
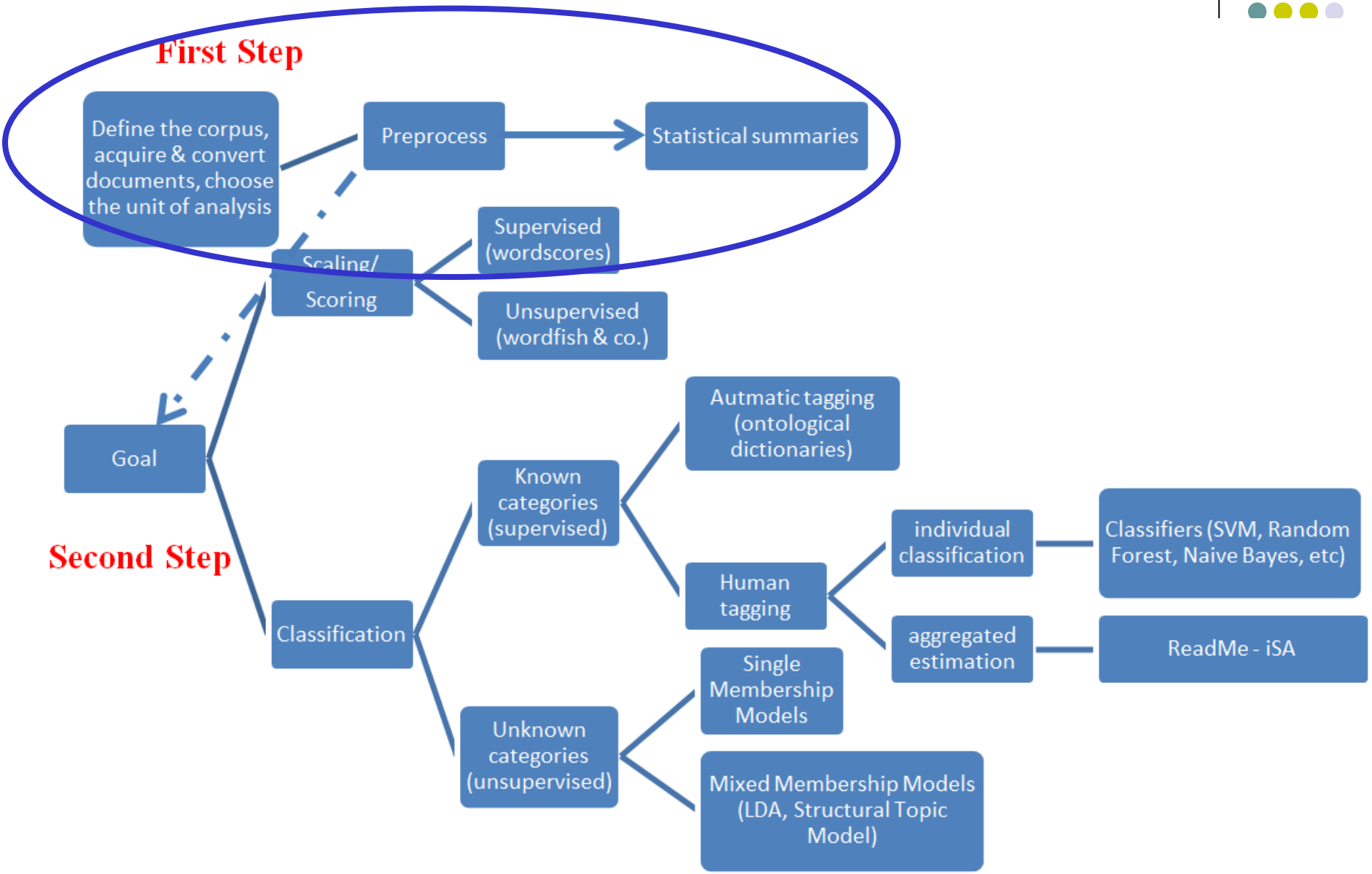
Applied Scaling & Classification Techniques in Political Science

Lecture 1 (second part)

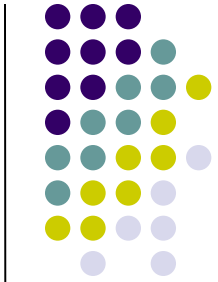
How to prepare a text for analysis



Our Course Map



The First Step: the preparation



Three stages:

1. **Defining the corpus, acquiring & converting the texts, choosing the unit of analysis**
2. **Preprocessing stage**: defining and refining features as well as converting of textual features into a quantative matrix
3. **Statistical summaries**

Define the corpus

Jargon: we refer to *text* or *document* as the unit of analysis (it could apply to any unit of text: a tweet, Facebook status, spoken utterance, press briefing, sentence, or paragraph)

We refer to the population of texts to be analyzed as the *corpus* and a collection of these as *corpora*

Define the corpus

A year of articles about the economy from The New York Times, for instance, could form a sample for analysis, where the unit (text or document) is an article

A set of debates during (one of the many) votes on Brexit in the UK House of Commons could form another corpus, where the unit is a speech act (one intervention by a speaker on the floor of parliament)

German-language party election manifestos from 1949 to 2017 could form a corpus, where a unit is a manifesto

In each example, distinguishing external attributes, chosen by the researcher for the purpose of analysing a specific research question, are used to **define the document** distinguishing one unit of textual data from another

Define the corpus

The key is to be aware of the **mechanisms governing the generation of text**, with an aim to making sure that the observable text provides **representative coverage** of the phenomenon that it will be used to investigate

That is, using automated text analysis procedures **does not absolve the researcher of responsibility** for ensuring that the texts under examination are related to the research question and have theoretical consistency

Acquire the texts

The burst of interest in automated content methods is mainly due to the proliferation of easy-to-obtain digital texts

Some of these texts are already available (for example, legislative speeches), other should be recollected by you (for example, from social media via API or from audio that you should transform in texts)

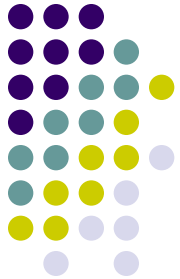
Later on we will discuss how to retrieve data from social media (i.e., Twitter)

Unfortunately we won't have time to discuss how to scrape data using R or how to transform audio in texts

Convert the texts

The step of converting the texts into a common electronic format is a purely technical one, involving no research design decisions, but it can nonetheless poses one of the stickiest problems in text analysis (pdf as image…)

Choose the unit of analysis



This step **differs from the selection of the corpus** in that prior to move on with the analysis, the **unit of analysis** may need further definition, through selection or sampling or by aggregating documents into larger units or splitting them into smaller ones

The attributes that differentiate source texts, in other words, may not form the ideal units for analysing the text as data

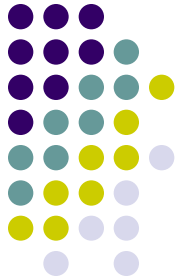
Choose the unit of analysis

For example, while we might have a corpus of social media posts, these might be **better aggregated** over some time period, such as a day, or by user

This not only ameliorates a possible problem with overly short documents, but also focuses attention on the **unit of interest**

Whether this is time or a user (or speaker or other unit of authorship) will depend on the research problem. For other problems, segmenting a document into smaller units might be the answer

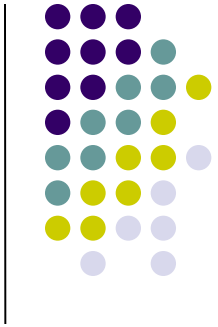
Preprocessing stage



But then…how to move from words to number? That is:

- *how a text can be transformed into digital data so that an algorithm can then treat it?*

Preprocessing stage



Introducing some terms…

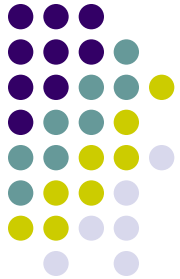
Words as they occur in a text are commonly known as **tokens**, so that the text "*one two one two*" contains four tokens

Tokenization is the process of **splitting a text** into its constituent tokens

Tokenization usually happens by recognizing the delimiters between words, which in most languages takes the form of a space. In more technical language, inter-word delimiters are known as **whitespace**, and include additional machine characters such as newlines, tabs, and space variants

Most languages separate words by whitespace, but some major ones such as Chinese, Japanese, and Korean do not

Preprocessing stage

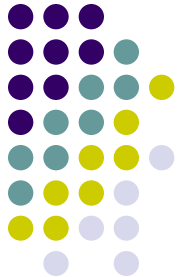


For example, Japanese sentence is only distinguished by commas and periods, and words are put in sequence without spaces in between. And so?

Tokenizing these languages requires a set of rules to recognise word boundaries, usually from a listing of common word endings

Smart tokenizers will also separate punctuation characters that occur immediately following a word, such as the comma after word in this sentence

Preprocessing stage

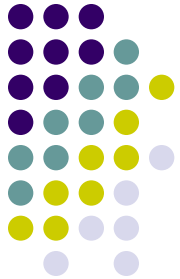


私は、日本社会党を代表して、当面する内外の諸問題につき、佐藤総理大臣にその所見をたださんとするものであります。

↓ after tokenization

私_は_、_日本_社会党_を_代表_し_て_、_当面_する 内外_の_諸問題_に_つき_、_佐藤_総理_大臣_に_その_所見_を_たださん_と_する_もの_で_あり_ます_。

Preprocessing stage



To introduce another term, **word types** refer to uniquely occurring words

So that the text "*one two one two*" contains four tokens, but only two word types, "one" and "two"

Preprocessing stage

For a **token** to become a **feature** of textual data (our basic unit of analysis), it typically undergoes transformation in a step often called “pre-processing”

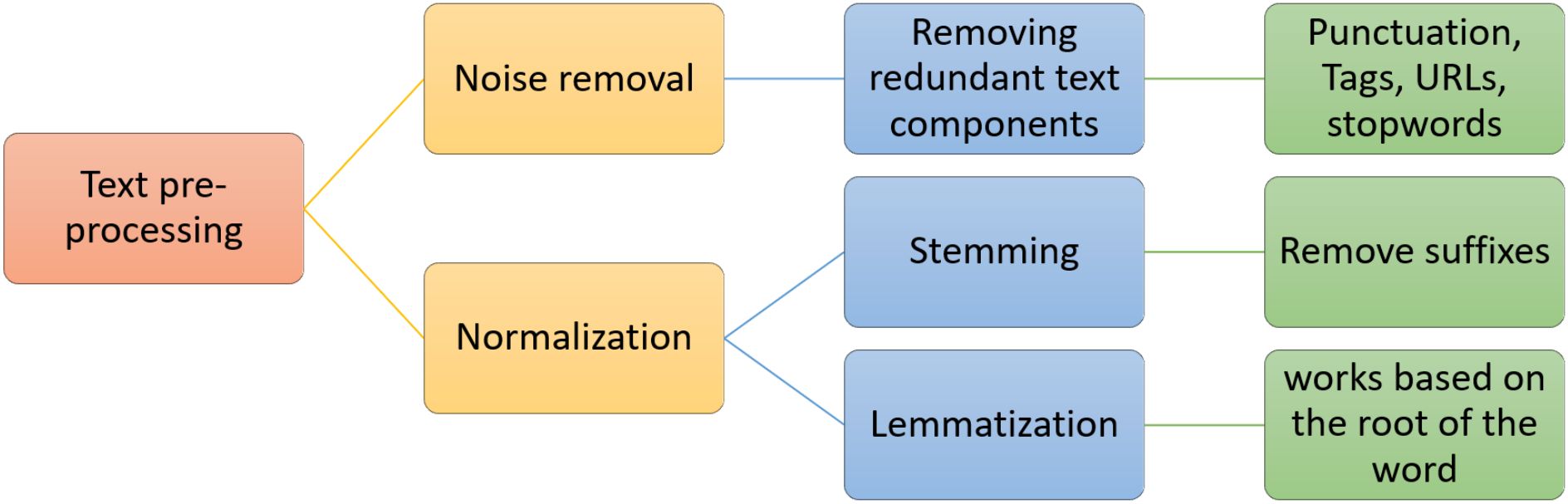
Why such transformation is needed? Cause language is **complex**. But not all of language’s complexity is necessary to effectively analyze texts

We should **retain information** that will be used by the automated methods, **while discarding information** that will likely be unhelpful, ancillary, or too complex for use in a statistical model

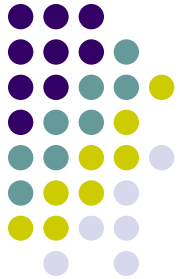
In other words: there are many forms of “words”, and these typically undergo a process of selection and transformation before they become **features** of our textual dataset

Preprocessing stage

Text pre-processing can be divided into two broad categories—**noise removal & normalization**



Preprocessing stage



1. **Noise removal**: Data components that are redundant to the core text analytics can be considered as **noise**

   Such as?!?

The First Step: the preparation

Stopwords! They include the large number of prepositions, pronouns, conjunctions etc. in sentences such as *the, is, at, which,* and *on* in English that occur in the greatest frequency in natural language texts

These words can be considered **unlikely** to contribute useful information for analysis, adding little specific political meaning to the text

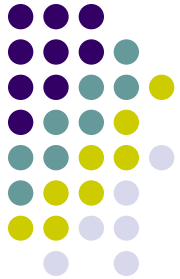
However…

The First Step: the preparation

…the pronoun “**her”**, as Monroe, Quinn and Colaresi (2008) found, has a decidedly partisan orientation in debates on abortion in the U.S. Senate

For these reasons, when preparing textual data for analysis, always check the impact on your final results of eliminating or not stopwords…

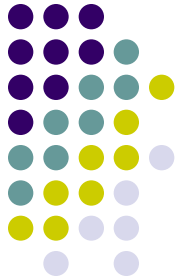
The First Step: the preparation



We also typically discard:

- **Punctuation**
- **Capitalization**: we apply lower-casing, which treats words as equivalent regardless of how they were capitalised
- We can also decide to eliminate words through the use of **predefined lists of words to be ignored** (for example: tags, Urls, etc.) or based on **their relative infrequency** (words that appear only once or twice in the corpus are unlikely to be discriminating)

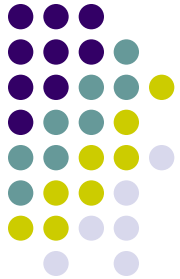
The First Step: the preparation



2. **Normalization**: Handling multiple occurrences / representations of the same word is called normalization

There are two types of normalization: **stemming** and **lemmatization**

The First Step: the preparation



Stemming normalizes text by reducing words to their stems, which is a cruder algorithmic means of equating a word with its canonical (dictionary) form, i.e., stemming treats **words as equivalent** when they differ only in their inflected forms

For example, the different words *taxes, tax, taxation, taxing, taxed, and taxable* are all converted to their word stem "**tax**"

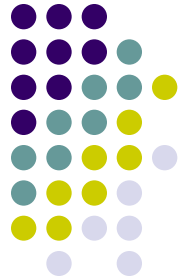
The First Step: the preparation

Form	Suffix	Stem
studies	-es	studi
studying	-ing	study
niñas	-as	niñ
niñez	-ez	niñ

Stemming

By doing that, stemming reduce the total number of unique words in the data set

The First Step: the preparation



Lemmatization is a more advanced technique which works based on the **root of the word** taking into consideration the **morphological analysis of the words**

To do so, it is necessary to have detailed dictionaries which the algorithm can look through to link the form back to its lemma

The First Step: the preparation

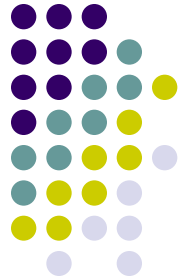
Form	Suffix	Stem
studies	-es	studi
studying	-ing	study
niñas	-as	niñ
niñez	-ez	niñ

→ **Stemming**

Lemmatization ←

Form	Morphological information	Lemma
studies	Third person, singular number, present tense of the verb study	study
studying	Gerund of the verb study	study
niñas	Feminine gender, plural number of the noun niño	niño
niñez	Singular number of the noun niñez	niñez

The First Step: the preparation



In our analysis, **we also discard the order** in which words occur in documents, i.e., we assume that documents are a **bag of words**, where order does not inform our analyses

Is it a problem?

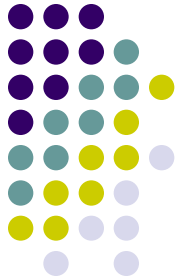
For instance, the expressions '*We are against lowering taxes, and for tax increases*' and '*We are for lowering taxes, and against tax increases*' use the exact same words, even though the meaning is reversed

The First Step: the preparation

While it is easy to construct sample sentences where word order fundamentally changes the nature of the sentence, empirically these sentences are rare

As a result, a simple list of words, which we call **unigrams**, is often sufficient to convey the general meaning of a text

The First Step: the preparation

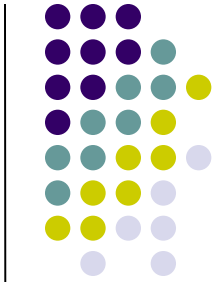


We can also *retain some word-order* by including **bigrams** (word pairs, for example to distinguish the “White House” from the color and the domicile) or any other (defined as sequences of n consecutive tokens to form not words but phrases)

This is a brute force method of recovering politically meaningful multi-word expressions that might contain identical unigrams but as phrases, mean exact opposites, such as economy in the multi-word expressions “*command economy*” and “*market economy*”

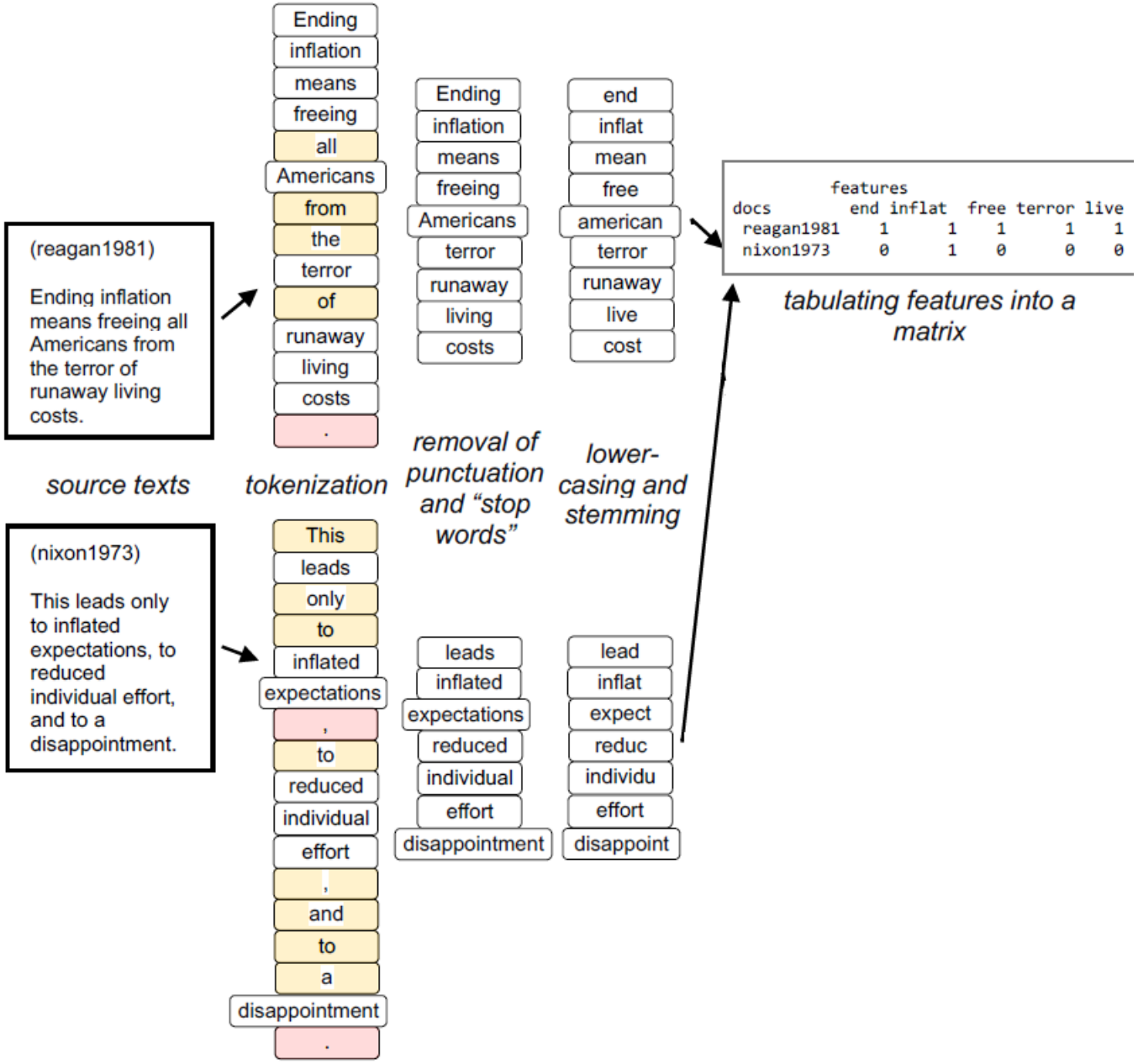
In practice, for common tasks, n-grams do little to improve the performance of text analysis

The First Step: the preparation

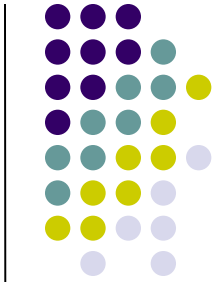


The result of the *preprocessing steps* is that each document can be represented as a **vector that counts** the number of times each of the unique words occur in each document

Multiple document vectors are then put together in a **document-term matrix (or document-feature matrix)**, where each **row** represents a document and each **column** represents a unique word, or term


(reagan1981)
Ending inflation means freeing all Americans from the terror of runaway living costs.
(nixon1973)
This leads only to inflated expectations, to reduced individual effort, and to a disappointment.
source texts
Ending
inflation
means
freeing
all
Americans
from
the
terror
of
runaway
living
costs
.
This
leads
only
to
inflated
expectations
,
to
reduced
individual
effort
,
and
to
a
disappointment
.
tokenization
Ending
inflation
means
freeing
Americans
terror
runaway
living
costs
leads
inflated
expectations
reduced
individual
effort
disappointment
removal of punctuation and "stop words"
end
inflat
mean
free
american
terror
runaway
live
cost
lead
inflat
expect
reduc
individu
effort
disappoint
lower-casing and stemming
features
docs end inflat free terror live
reagan1981 1 1 1 1 1
nixon1973 0 1 0 0 0
tabulating features into a matrix

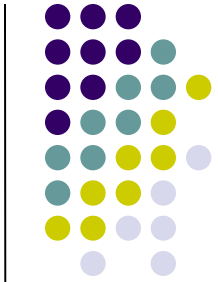
The First Step: the preparation



This matrix form of textual data can then be used as input into a variety of **analytical methods** for describing the texts

Quantitative text analysis thus moves textual data into the same domain as other types of quantitative data analysis, making it possible to bring to bear well-tested statistical and machine learning tools of analysis and prediction

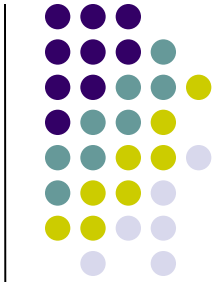
The First Step: the preparation



Ironically, generating insight from text as data is only possible **once we have destroyed** our ability to make sense of the texts directly

To make it useful as data, we had to **obliterate the structure of the original text and turn its stylised and oversimplified features** into a glorified spreadsheet that no reader can interpret directly, no matter how expert in linear algebra

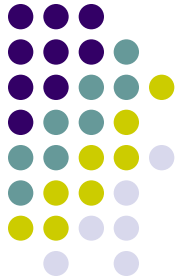
The First Step: the preparation



We should not lose any sleep over it, because the point in analysing text as data was **never to interpret the data but rather to mine it for patterns**

(text) Mining is a destructive process - just ask any mountain! - and some destruction is inevitable in order to extract its valuable resources

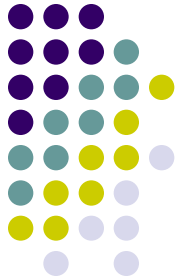
The First Step: the preparation



A bag-of-words approach seems to result in a shocking reduction of information, leaving many to conclude that there will be too little information to extract anything meaningful from the texts

But consistently across applications, scholars have shown that this simple representation of text is sufficient to infer **substantively interesting properties of texts**!

The First Step: the preparation



Of course, in some contexts bringing back the context in which a word appears, can be very important…

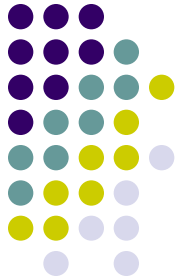
A bag-of-words assumption is that each word is independently generated from some underlying distribution

The problem is that speeches and other texts typically have a large number of words that are not actually independently generated

This could yields problematic likelihood-based uncertainty estimates in text analysis models more generally

How to deal with that?

The First Step: the preparation



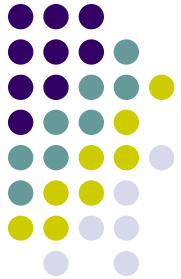
"You shall know a word by the company it keeps" (John Firth, 1957, 11)

Word embeddings!

It's a means of building a low-dimensional vector representation from corpus of text, which preserves the contextual similarity of words

If we have time, we will discuss about word embedding later on

The First Step: the preparation

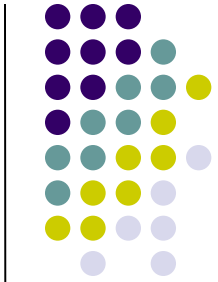


Thinking about the number of words every language is made of, one might think that the document-term matrix might possess a **huge number of columns** in any given analysis

For instance, the *Oxford English Dictionary* classifies more than 650,000 words

What turns out to be true is that, after stemming, the typical length of the stem vector (i.e., the number of columns) is no more than 300 or 500 and often much less

The First Step: the preparation

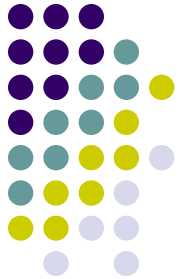


The main dimension that increases the computational challenge is, on the contrary, quite often the number of rows of the matrix, that is the number of texts to be analyzed

This number is usually in the **order of millions** in social media analysis for example

Still, regardless of the number of columuns, you could still have a **problem of sparsity**

The First Step: the preparation



In text mining, huge matrices are created based on word frequencies with many cells having zero values

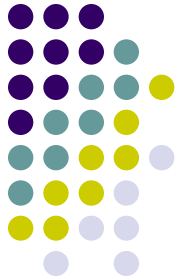
This problem is called **sparsity**

Document-term matrices are affected by what is known in machine learning as the **curse of dimensionality**: new observations tend to grow the feature set, and each new term found in even a document adds a new column to the matrix

Several of the pre-processing techniques just discussed allows to minimize such problems

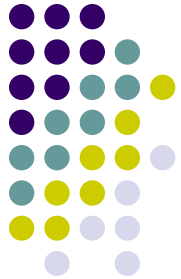
But several others are still available

The First Step: the preparation



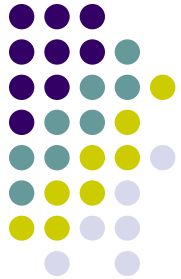
One further strategy for mitigating the problem of exponentially increasing dimensionality is to **trim** or to **weight** the document-feature(term) matrix

The First Step: the preparation



Trimming can be done on various criteria, but usually takes the form of a filter based on some form of feature frequency (i.e., keeping only features that appear just in 10% of documents for example)

The First Step: the preparation

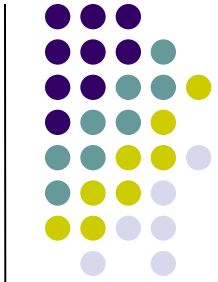


Weighting schemes convert a matrix of counts into a matrix of weights

The most common of these is **relative term frequency**, a weighting process also known as document normalisation because it homogenises the sum of the counts for each document

Since documents in a typical corpus vary in length, this provides a method for comparing frequencies more directly than counts, which are inflated in longer documents

The First Step: the preparation

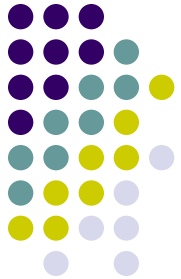


Words may also be weighted according to how rare or frequent they are in the corpus via a ***tf-idf*** **(term frequency-inverse document frequency)** matrix

tf-idf is a method in information retrieval for down-weighting the terms that are common to documents

tf-idf adds a weight that approaches zero as the number of documents in which a term appears (in any frequency) approaches the number of documents in the collection. When we have selected our texts because they pertain to a specific topic - as we usually will - then inverse document frequency weighting means zeroing out most of our topical words, since these will appear in most or all documents

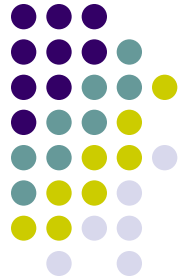
The First Step: the preparation



In texts of debates over health care, for instance, tf-idf weighting is likely to eliminate all words related to health care, even when they might occur at very different rates across different documents

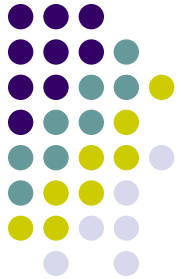
If we think that it is not the occurrence, but rather the **relative frequencies** of words that are informative, then using tf-idf weighting is the opposite of what we want!!!

The First Step: the preparation



Note that many models commonly used in political science - such as the Wordfish model or Latent Dirichlet allocation (topic) models that we will see later on - only work with counts as inputs, so that tf-idf or other weighting schemes are inapplicable

The First Step: the preparation



Never underestimate the *power* of the preprocessing stage!

Preprocessing has tremendous consequences for the quality of automated text analysis

The First Step: the preparation

In one of the few systematic studies of feature processing choices and their consequences, Denny and Spirling (2018) replicated several published text analyses from political science using a variety of alternative feature processing steps

Their results shows that “under relatively small perturbations of of preprocessing decisions...very different substantive interpretations would emerge”

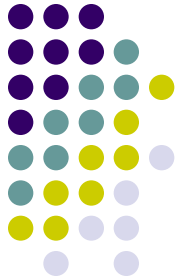
Researchers in practice should be aware of these decisions, critically examine the assumptions of their methods and how these relate to feature selection, and test the robustness of these results

Statistical summaries

Once you have your DtM, you can start by running some statistical summaries

Statistical summary methods are essentially quantitative summaries of texts to describe their characteristics on some indicator, and may use (or not) statistical methods based on sampling theory for comparison

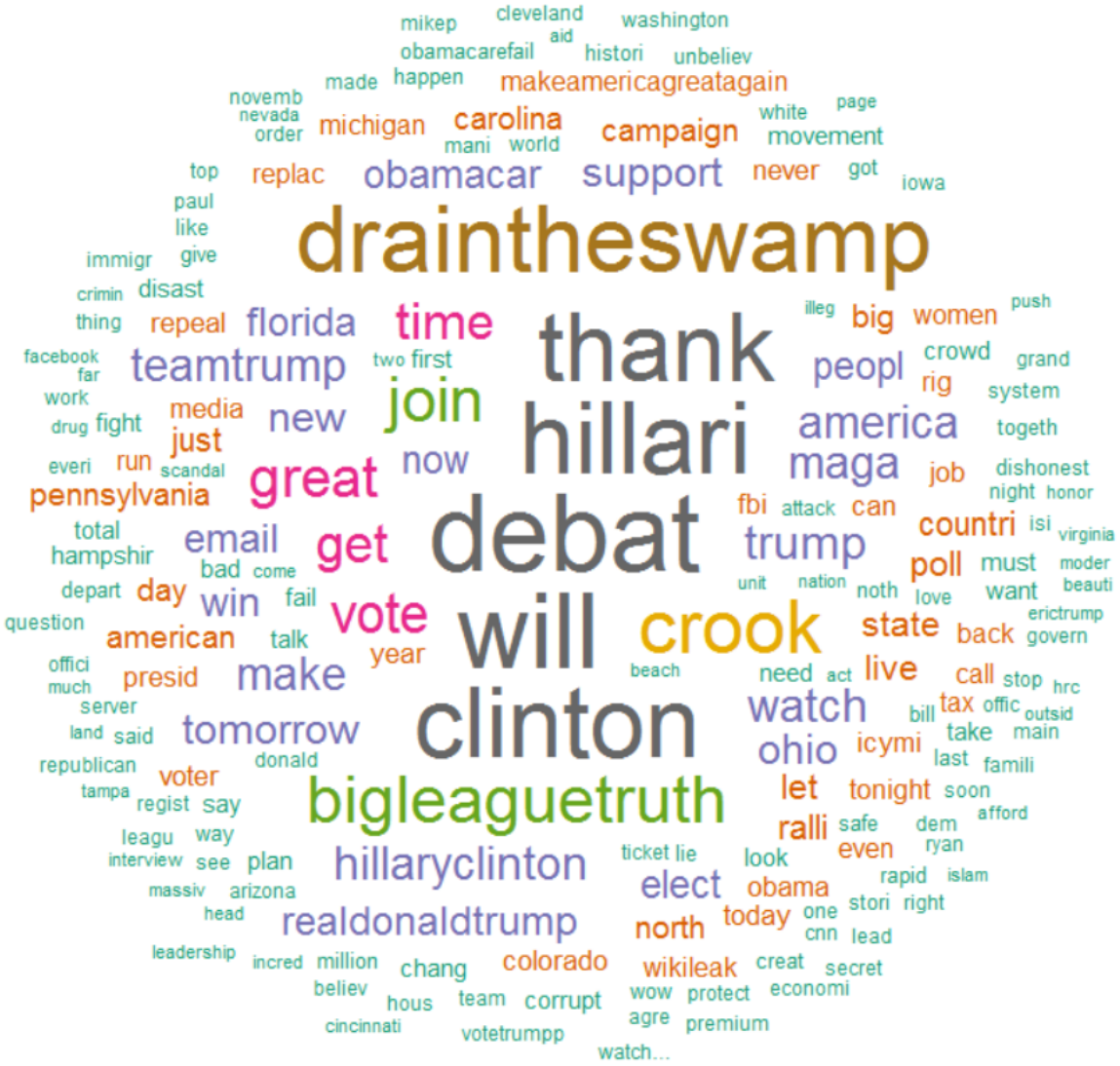
Statistical summaries (1)



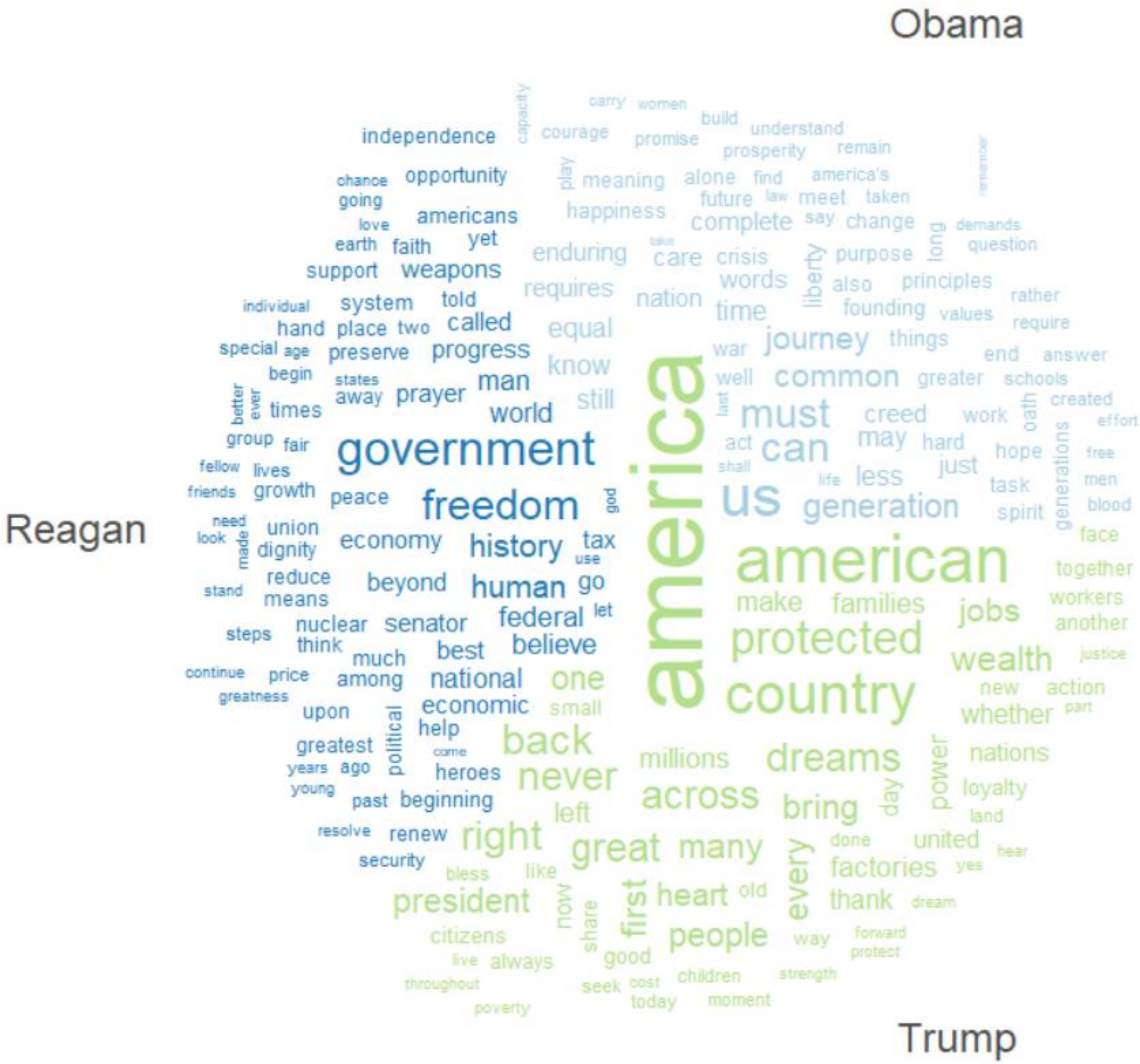
The simplest such measures identify the most commonly occurring words, and summarize these as frequency distributions

For example: **tag clouds**! A tag cloud is a visual representation of text data, in which tags are single words whose frequency is shown with different font size (and/or color)

Tag-cloud of the tweets posted in @realDonaldTrump during the last 2 months of the electoral campaign



Comparing tag-clouds! The US Presidential inaugural speeches example



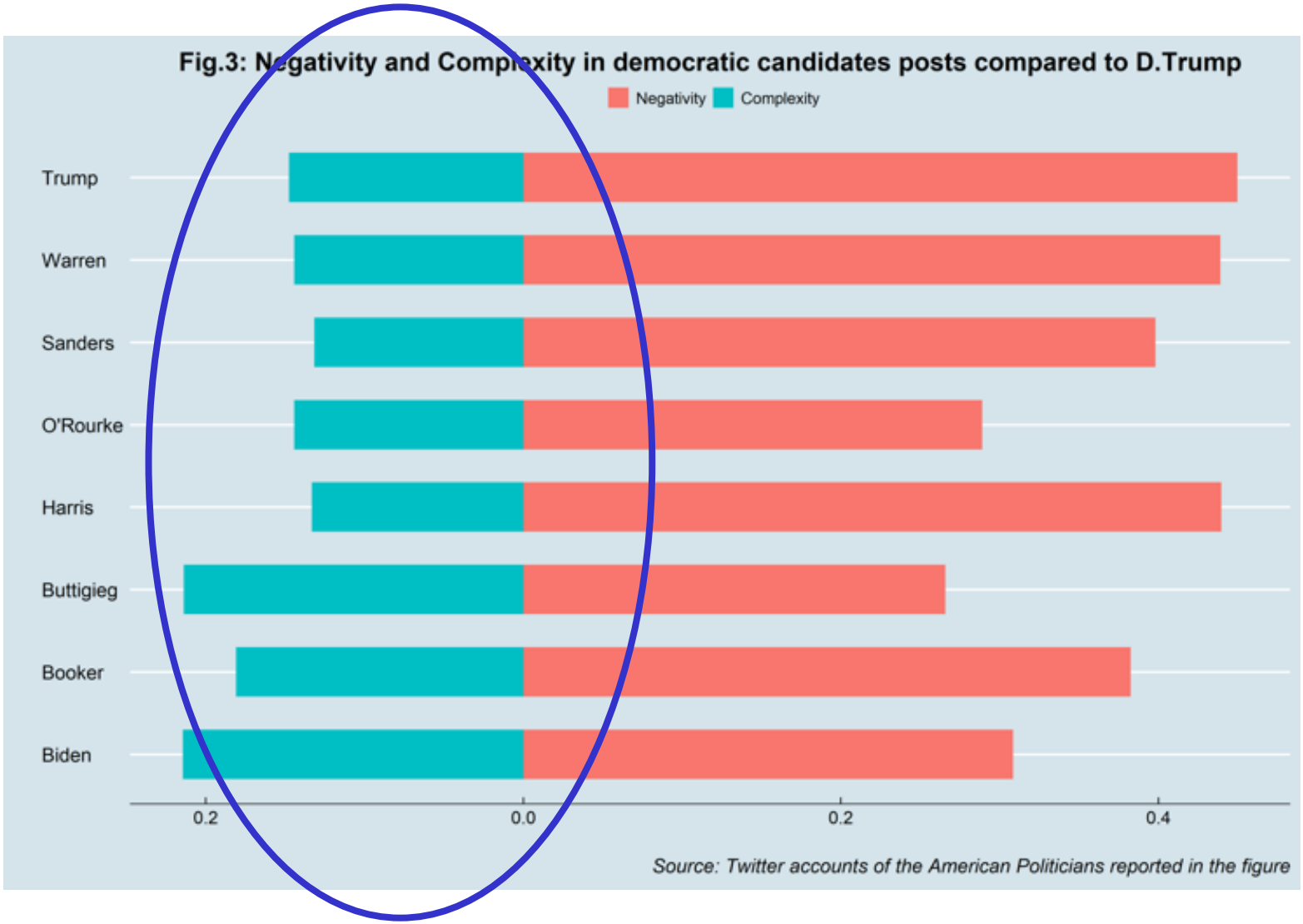
Statistical summaries (2)

Other quantitative summary measures of documents are designed to characterize specific qualities of texts

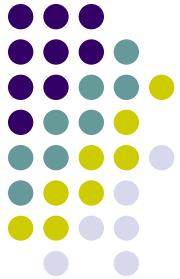
Comparing the **rates of types and tokens** forms the foundation for measures of **lexical diversity** (the rate of vocabulary usage), with most common such measure comparing the number of types to the number of tokens (the “type-token ratio”)

For example, it is argued that populist communication means simplified political discourse (lower diversity), in an attempt to reach the public more easily

So different, yet so alike (to Donald Trump?)

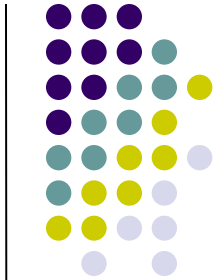


Statistical summaries (3)



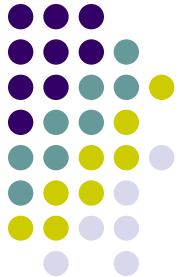
More sophisticated methods **compare the differential occurrences of words across texts** or partitions of a corpus, using statistical association measures, to identify the **words that belong primarily to sub-groups** such as those predominantly associated with male- versus female - authored documents, or Democratic versus Republican speeches

Statistical summaries (4)



By treating each document as a vector of term occurrences - or conversely, each feature as a vector of document occurrences - sim**ilarity and distance measures** allow **two documents (or features) to be compared** using bivariate measures such as the widely cosine similarity measure or Pearson's correlation coefficient, or one of the many distance measures such as the Euclidean distance

Statistical summaries (5)



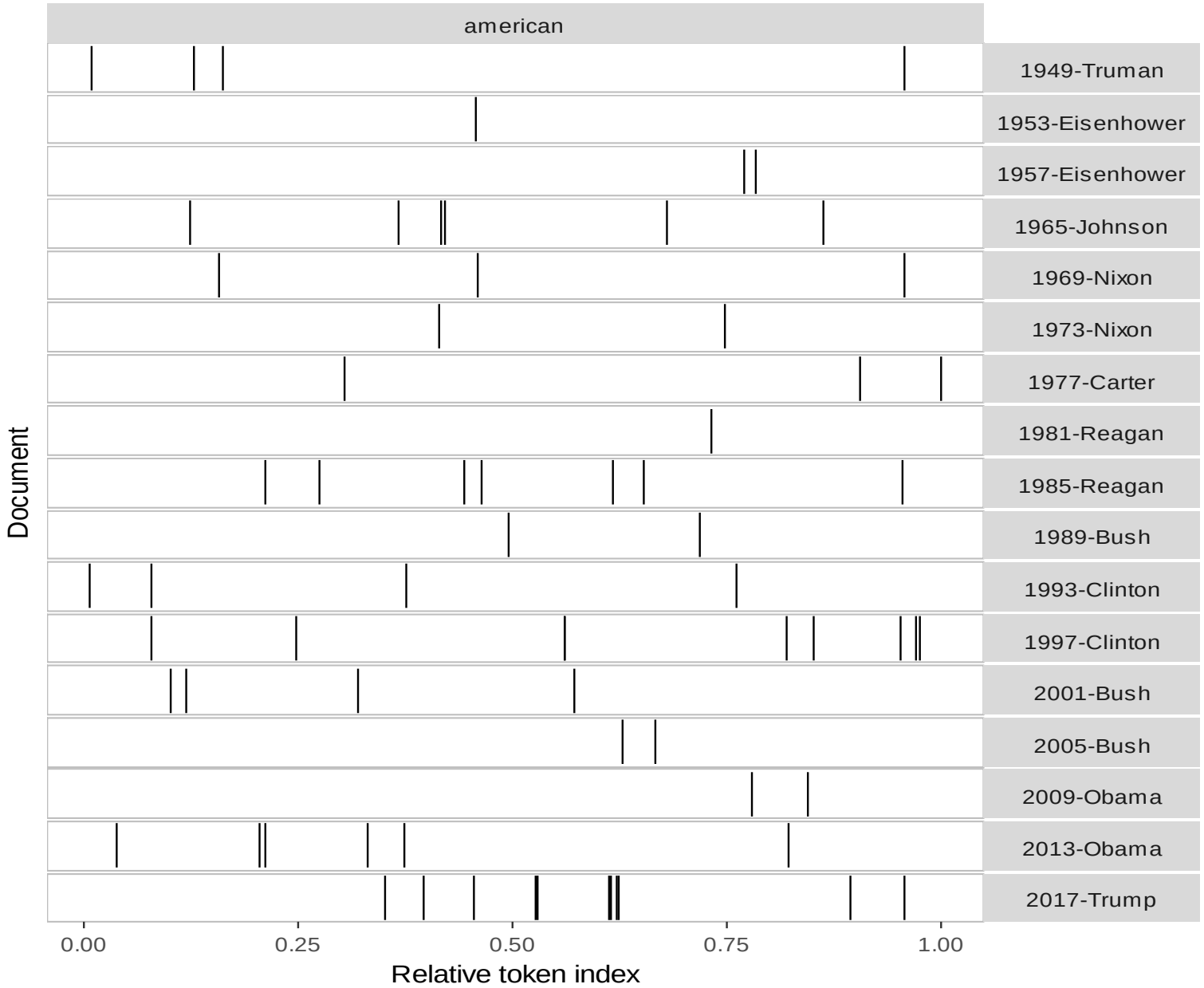
You can also group words by dictionary or equivalence class and check their presence in different documents

Statistical summaries (6)

Interesting descriptive statistics can also be produced directly by working with the corpus, rather than with the DtM

This allows us to retain the original text sequence, and therefore, for example, to detect both **the relative frequency of an employed word across documents as well as the "timing" of that word** via a *Lexical dispersion plot*

Inaugural Speeches by US Presidents



Before our first Lab class

If you **have a laptop** with you (you need around 20 mins):

1) Install the latest version of R
2) Install latest version of Rtools from here (https://cran.r-project.org/bin/windows/Rtools/) if you have **Windows platforms**

For **OS X**, do the following:

a) First try to install Quanteda directly
b) If you fail in doing that, install XCode from the App Store

Before our first Lab class

c) To install XCode, follow these simple rules:

1 Access to “Apple Developer”

https://developer.apple.com/download/more/

(You need Apple ID and password)

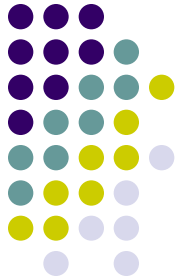
2 Insert “Xcode” in “Search Downloads” located on the left side of the page.

3 Choose "Xcode 11" and download.

4 After finishing download, click "Finder" and then "download." Double click "Xcode 11“. It may take a while to open this file

d) If you have problems to install the **latest version** of Xcode, **uses an earlier one**, such as Xcode 9!

Before our first Lab class



3) Install the following packages by running these lines:

install.packages('devtools', repos='http://cran.us.r-project.org')

install.packages('quanteda', repos='http://cran.us.r-project.org')

install.packages('readtext', repos='http://cran.us.r-project.org')

install.packages('ggplot2', repos='http://cran.us.r-project.org')

install.packages('stopwords', repos='http://cran.us.r-project.org')

Before our first Lab class

REMEMBER (if you do not have a laptop):

Once you have installed and set up everything, you should not log out of that computer or shut it down - if you do this, you'll have to start again from zero