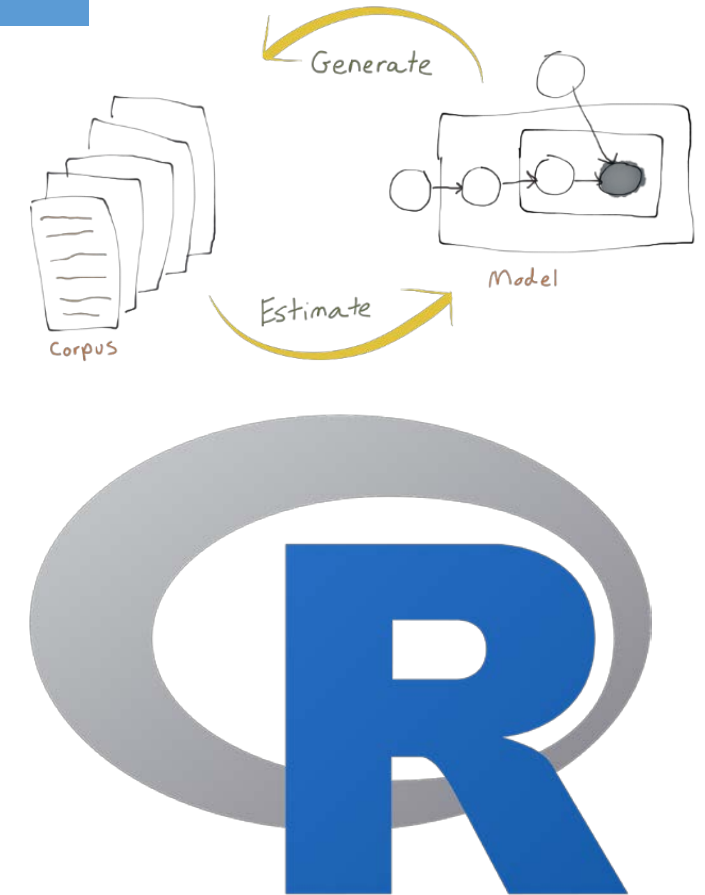


# Textual Analysis with Topic Modelling Using R

**Diego Ravenda, Associate Professor of Accounting, TBS Business School,  
Campus Barcelona. Email: [d.ravenda@tbs-education.es](mailto:d.ravenda@tbs-education.es)**

## February 2021



# Quick Text Mining Introduction



With internet and social media we have entered the so-called **Information Age**.

As more information becomes available, it becomes more difficult to find and discover what we need.

We need tools to help us organize, search and understand this vast amount of information.

**Topic modelling** provides methods for automatically organizing, understanding, searching, and **summarizing** large electronic archives:

1. Discover the **hidden themes** in the collection.
2. Annotate the documents according to these themes.
3. Use annotations to organize, summarize, search, and form predictions.

Today, the large collection of data calls for **unsupervised** probabilistic models.

## Example Applications:

### 1. Summarizing Collections of Images



SKY WATER TREE  
MOUNTAIN PEOPLE



SCOTLAND WATER  
FLOWER HILLS TREE

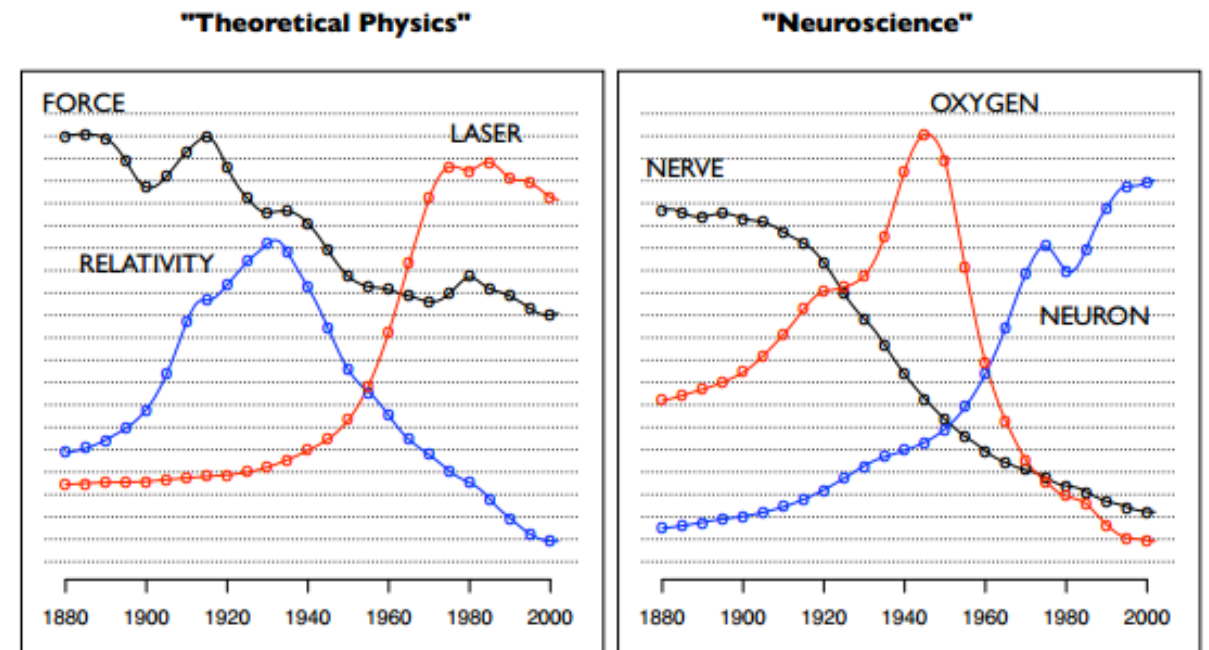


FISH WATER OCEAN  
TREE CORAL



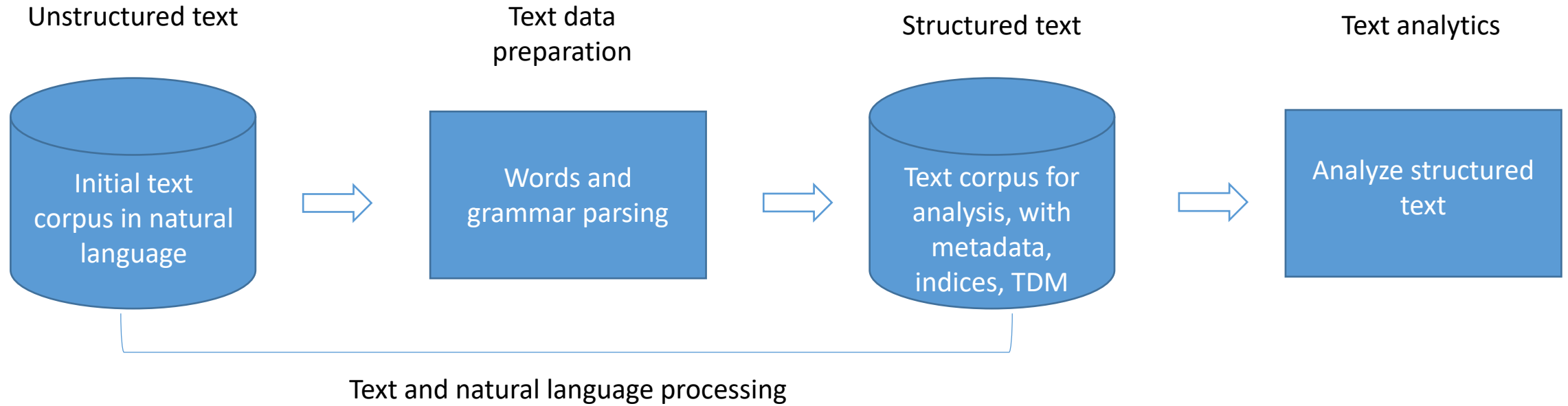
PEOPLE MARKET PATTERN  
TEXTILE DISPLAY

### 2. Evolution of Pervasive Topics by Time



# Text Mining Introduction

## Intro Text Mining & Pre-Processing



Source: Adaptive from Miller (2005)

# Text Mining Introduction

## Text mining and other terms



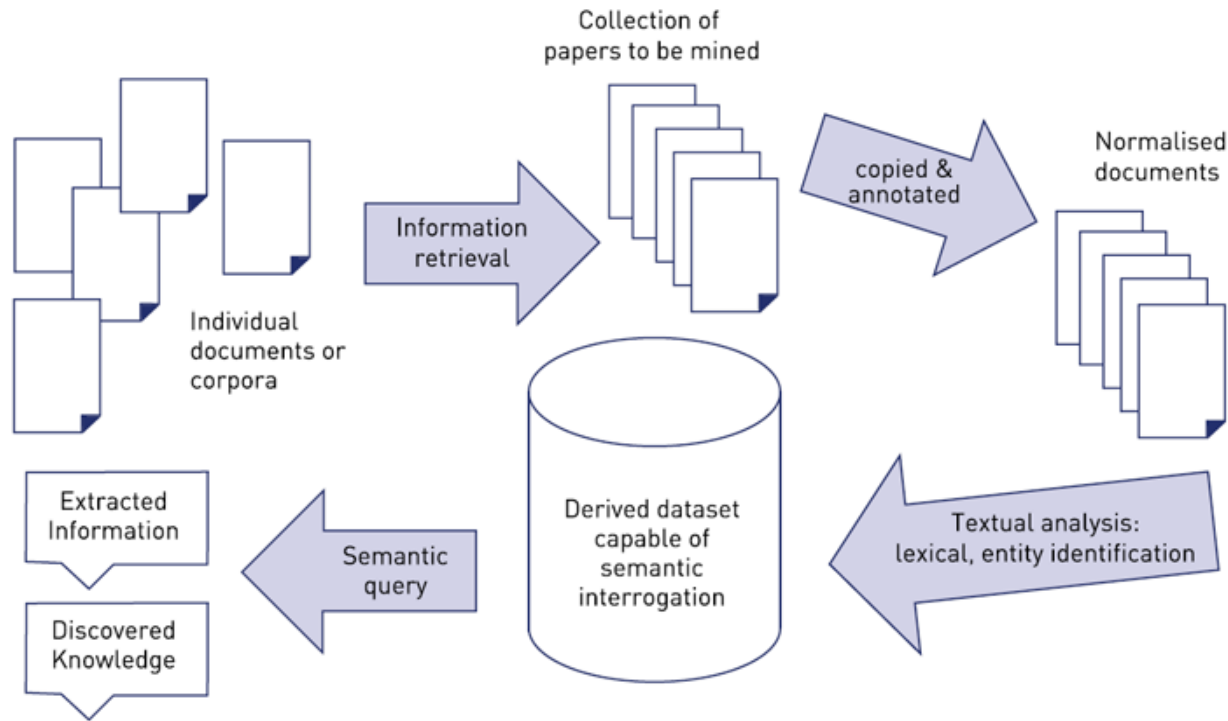
Source: Wikipedia

- **Text mining**: the process of deriving high-quality information from text.
- **Corpus**: is a large and structured set of texts.
- **Stop words**: words which are filtered out before or after processing of natural language data (text).
- **Unstructured text**: information that either does not have a pre-defined data model or is not organized in a pre-defined manner.
- **Tokenizing**: process of breaking a stream of text up into words, phrases, symbols, or other meaningful elements called tokens (see also lexical analysis).
- **Natural language processing**: field of computer science, artificial intelligence, and linguistics concerned with the interactions between computers and human (natural) languages.
- **Term document (or document term) matrix**: a mathematical matrix that describes the frequency of terms that occur in a collection of documents.
- **Supervised learning**: the machine learning task of inferring a function from labeled training data.
- **Unsupervised learning**: find hidden structure in unlabeled data.
- **Stemming**: the process for reducing inflected (or sometimes derived) words to their root form. (e.g., "close" will be the root for "closed", "closing", "close", "closer" etc.).



# Text Mining Introduction

## Document & information retrieval



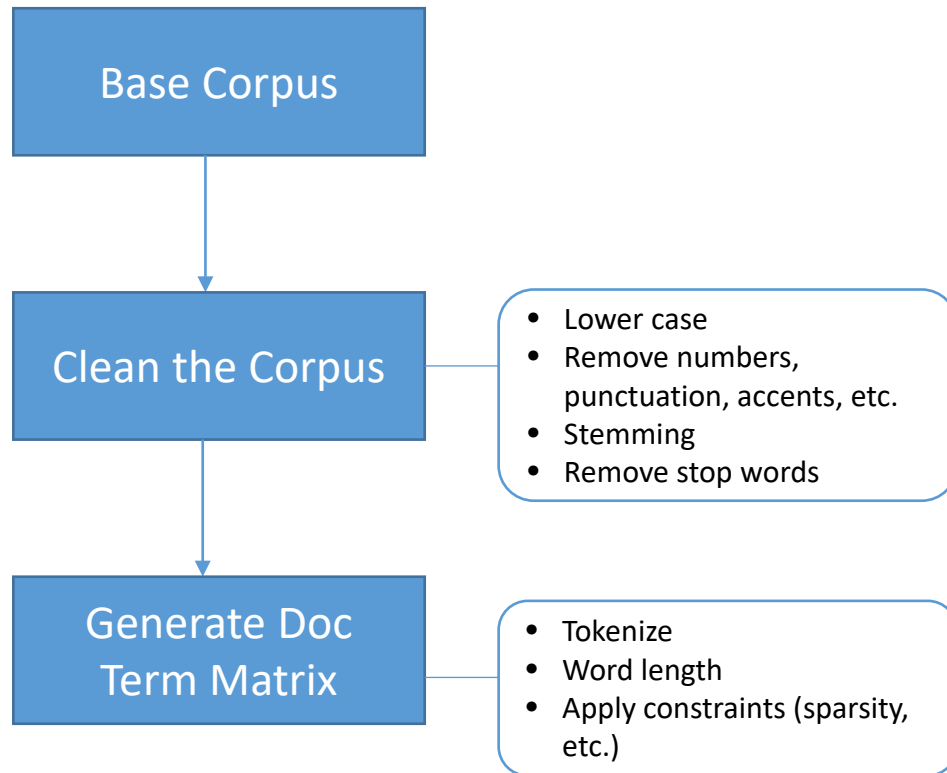
Source: <http://www.jisc.ac.uk/reports/value-and-benefits-of-text-mining>

The idea is how do we take this unstructured text, index it in such a way that allows us to integrate the structure analytics back to the core information to move, sort, search, process, categorize, etc. by document.

Common IR goals:

- Ad-hoc retrieval
- Filtering/Sorting
- Browsing

## Pre-Processing for Topic Modeling



```
packages <- c('quanteda', 'tm', 'NLP', 'SnowballC', 'openNLP',  
'openNLPmodels.en', 'RWeka')
```

### Pre-processing

- The input data for topic models is a document-term matrix. The rows in this matrix correspond to the documents and the columns to the terms.
- The number of rows is equal to the size of the corpus and the number of columns to the size of the vocabulary.
- Mapping from the document to the term frequency vector involves tokenizing the document and then processing the tokens for example by converting them to lower-case, removing punctuation characters, removing numbers, stemming, removing stop words and omitting terms with a length below a certain minimum.
- Each term in a collection's vocabulary, the index maps in which document the term was posted (inverted indices or lists).



# Topic Modeling

- We want to find themes (or topics) in documents
  - ✓ useful for e.g., search or browsing.
- We do not want to do supervised topic classification
  - ✓ rather not fix topics in advance nor do manual annotation.
- Need an approach which automatically teases out the topics based on co-occurring words.
- This is essentially a *clustering* problem - can think of both words and documents as being clustered.



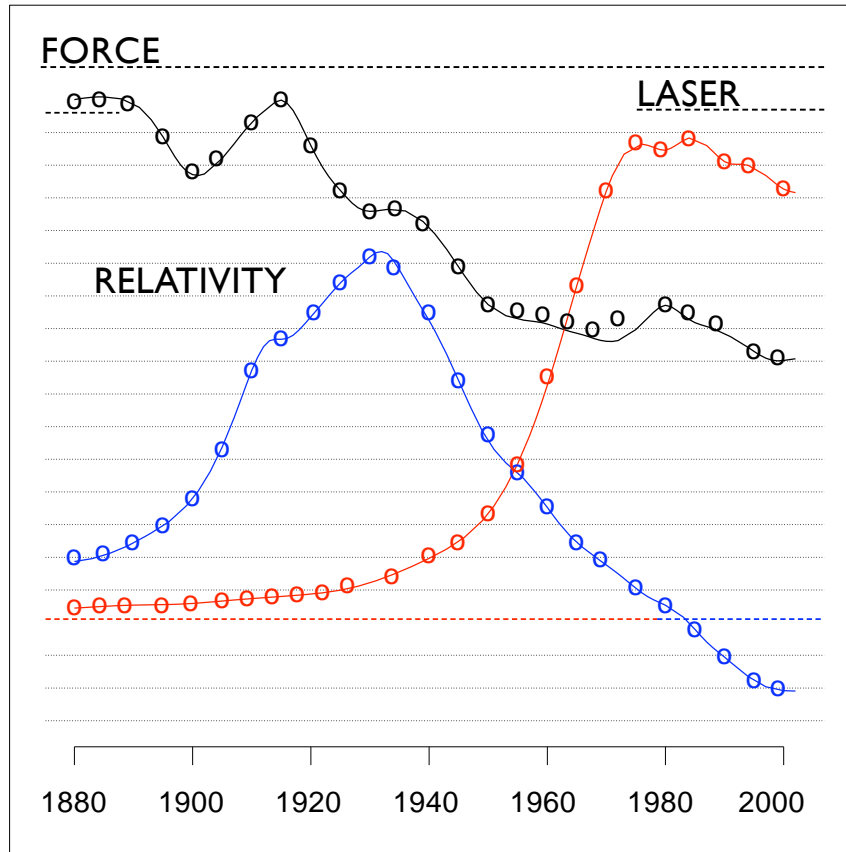
Topic modeling provides methods for automatically organizing, understanding, searching, and summarizing large electronic archives:

1. Discover the hidden themes that pervade the collection.
2. Annotate the documents according to those themes.
3. Use annotations to organize, summarize, and search the texts.
4. **Track topic prevalence, compute similarity between topics and use regression tools.**

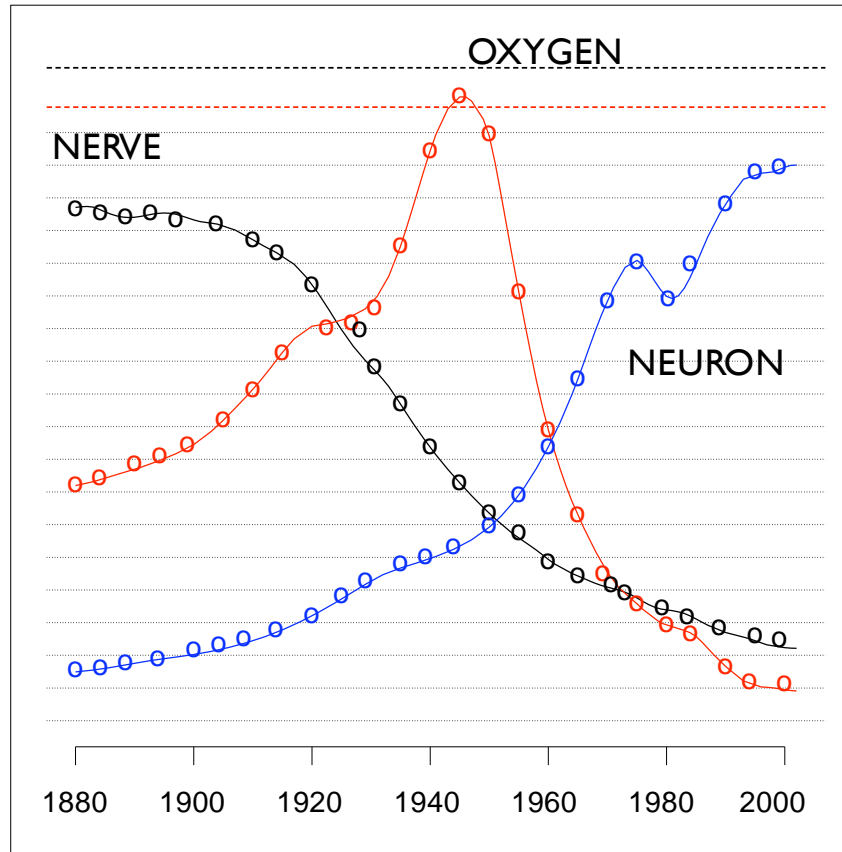
## Probabilistic Topic Models

<b>Topic 1</b>	<b>Topic 2</b>	<b>Topic 3</b>	<b>Topic 4</b>
human	evolution	disease	computer
genome	evolutionary	host	models
dna	species	bacteria	information
genetic	organisms	diseases	data
genes	life	resistance	computers
sequence	origin	bacterial	system
gene	biology	new	network
molecular	groups	strains	systems
sequencing	phylogenetic	control	model
map	living	infectious	parallel
information	diversity	malaria	methods
genetics	group	parasite	networks
mapping	new	parasites	software
project	two	united	new
sequences	common	tuberculosis	simulations

"Theoretical Physics"



"Neuroscience"



# Probabilistic Topic Models



SKY WATER TREE  
MOUNTAIN PEOPLE



SCOTLAND WATER  
FLOWER HILLS TREE



SKY WATER BUILDING  
PEOPLE WATER



FISH WATER OCEAN  
TREE CORAL



PEOPLE MARKET PATTERN  
TEXTILE DISPLAY



BIRDS NEST TREE  
BRANCH LEAVES



# Latent Dirichlet allocation

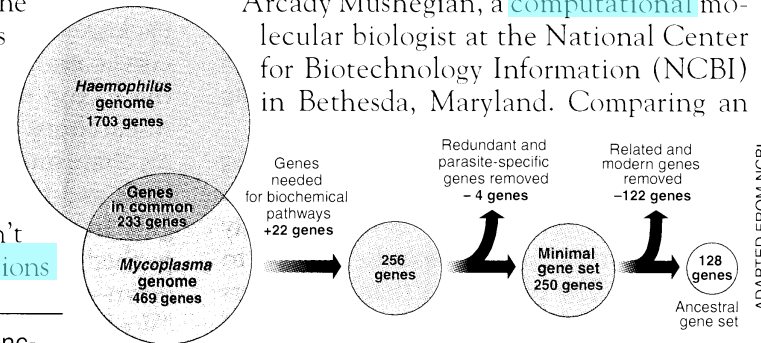
## Seeking Life's Bare (Genetic) Necessities

COLD SPRING HARBOR, NEW YORK—How many **genes** does an **organism** need to **survive**? Last week at the genome meeting here,\* two genome researchers with radically different approaches presented complementary views of the basic genes needed for **life**. One research team, using **computer** analyses to compare known **genomes**, concluded that today's **organisms** can be sustained with just 250 genes, and that the earliest life forms required a mere 128 **genes**. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.

Although the numbers don't match precisely, those **predictions**

\* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.

“are not all that far apart,” especially in comparison to the 75,000 **genes** in the human genome, notes Siv Andersson of Uppsala University in Sweden, who arrived at the 800 number. But coming up with a consensus answer may be more than just a **genetic numbers** game, particularly as more and more **genomes** are completely mapped and sequenced. “It may be a way of organizing any newly **sequenced genome**,” explains Arcady Mushegian, a **computational** molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an



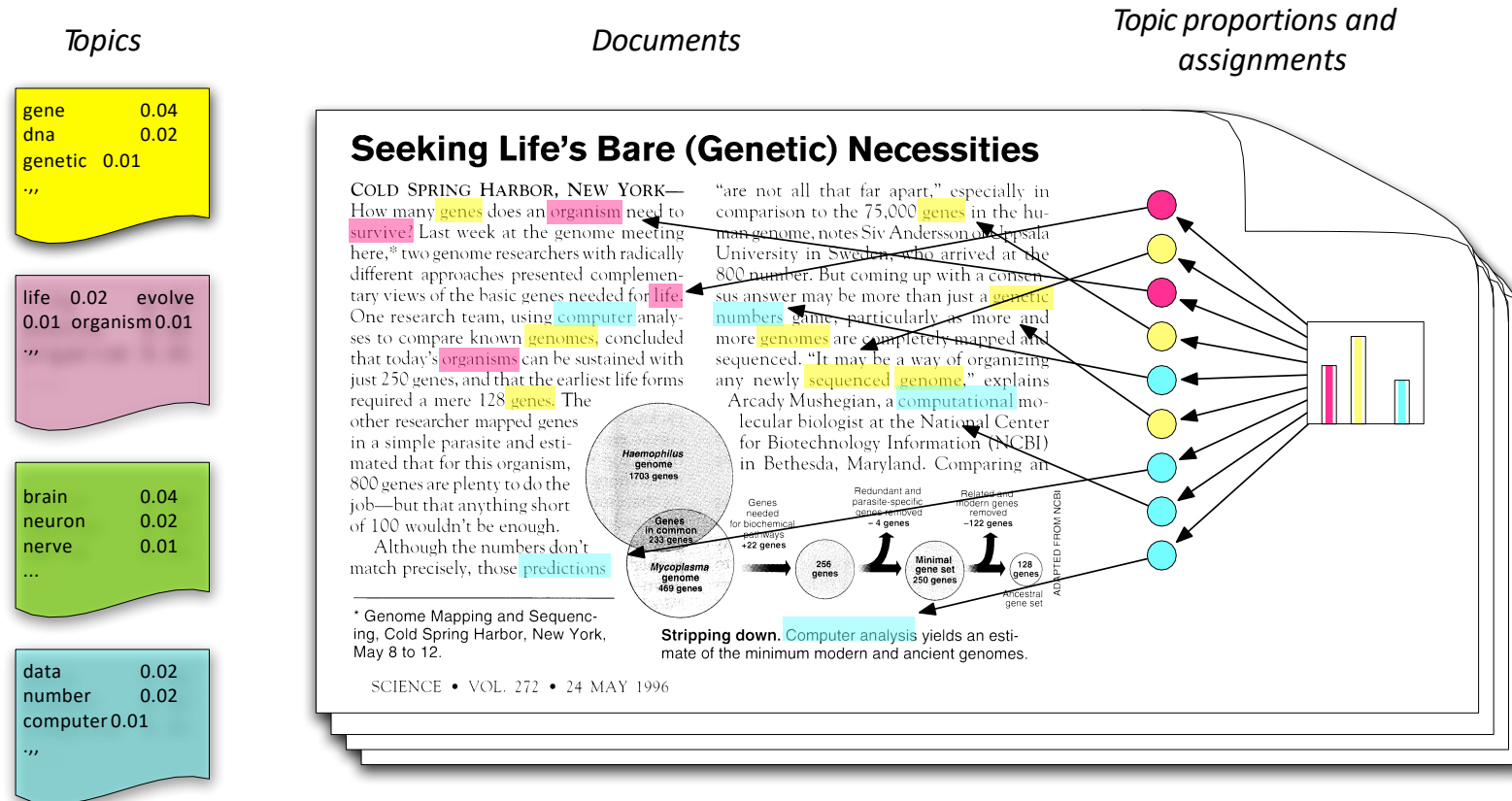
**Stripping down.** **Computer analysis** yields an estimate of the minimum modern and ancient genomes.

ADAPTED FROM NCBI

SCIENCE • VOL. 272 • 24 MAY 1996

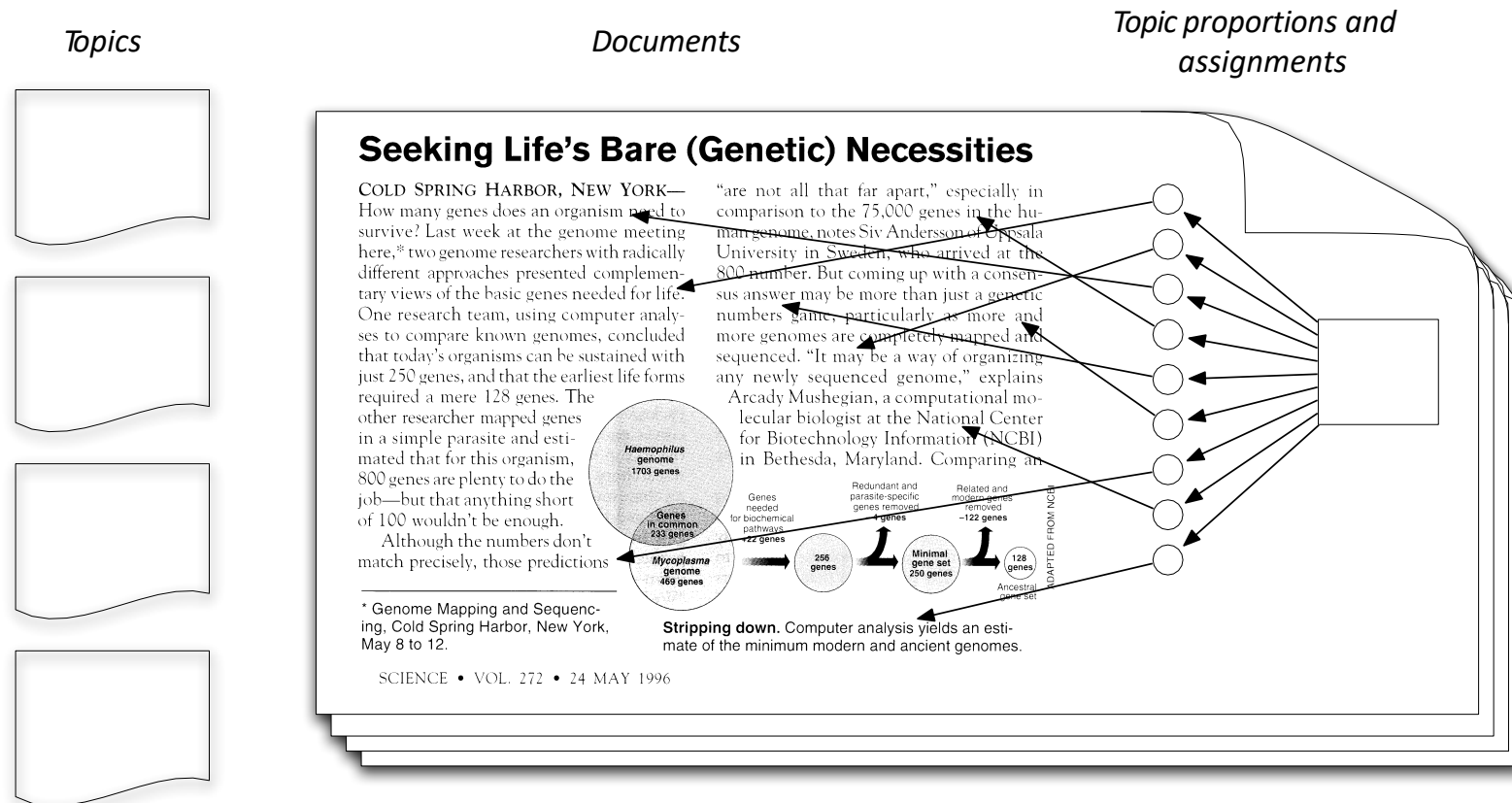
**Simple intuition:** Documents exhibit multiple topics.

# Introduction into Latent Dirichlet Allocation (LDA)



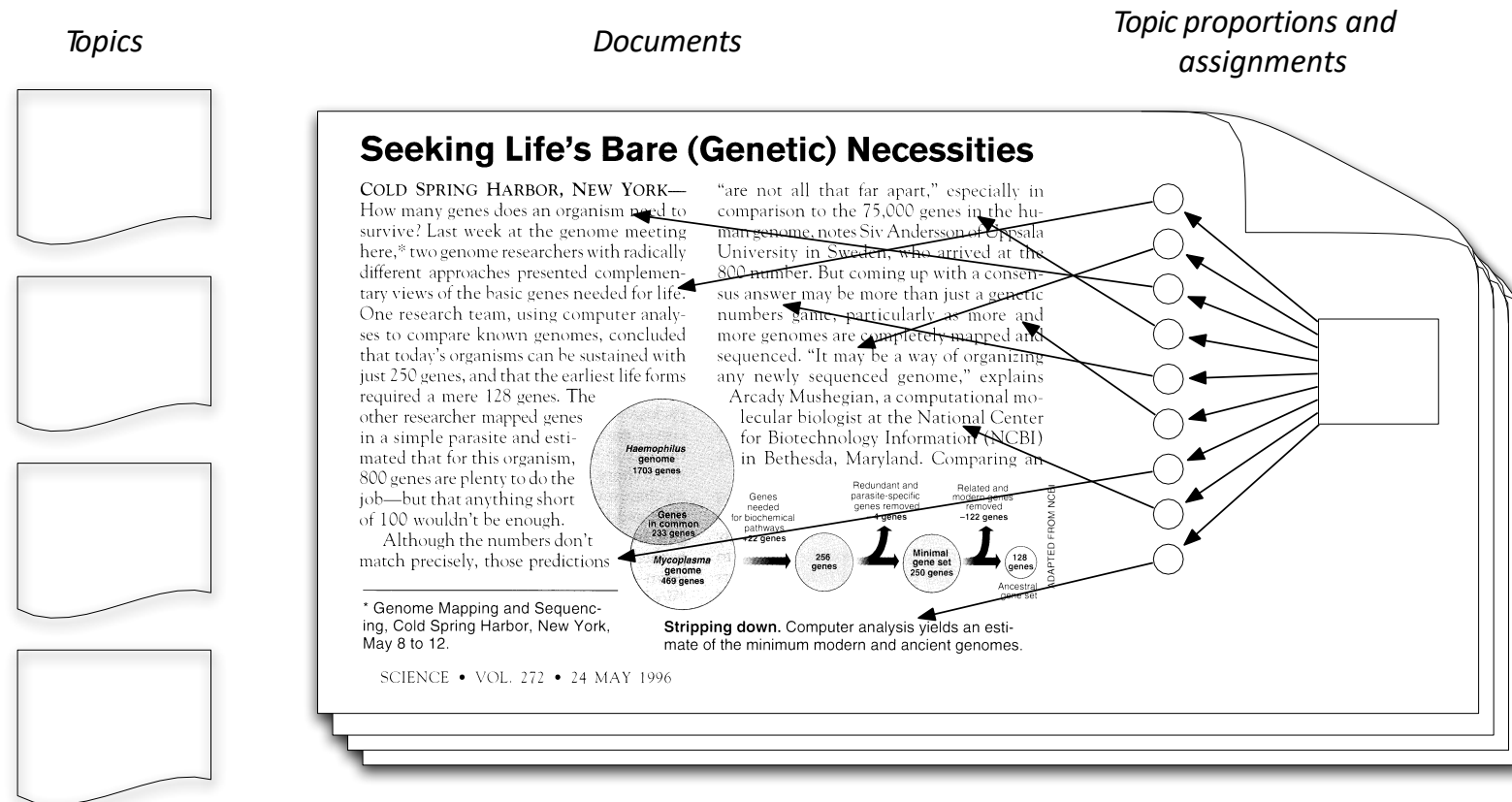
- Each **topic** is a distribution over a fixed vocabulary
- Each **document** is a mixture of corpus-wide topics
- Each **word** is drawn from one of those topics
- Only the **number of topics** is specified in advance

# Introduction into Latent Dirichlet Allocation (LDA)



- In reality, we only observe the documents
- The other structure are **hidden variables**

# Introduction into Latent Dirichlet Allocation (LDA)



- Our goal is to **infer** the hidden variables
- I.e., compute their distribution conditioned on the documents

$$p(\text{topics, proportions, assignments} \mid \text{documents})$$

## Introduction into Latent Dirichlet Allocation (LDA)

The Latent Dirichlet Allocation (LDA) model is a generative probabilistic model applied to text introduced by Blei et al. (2002) and Blei et al. (2003).

Some Assumptions:

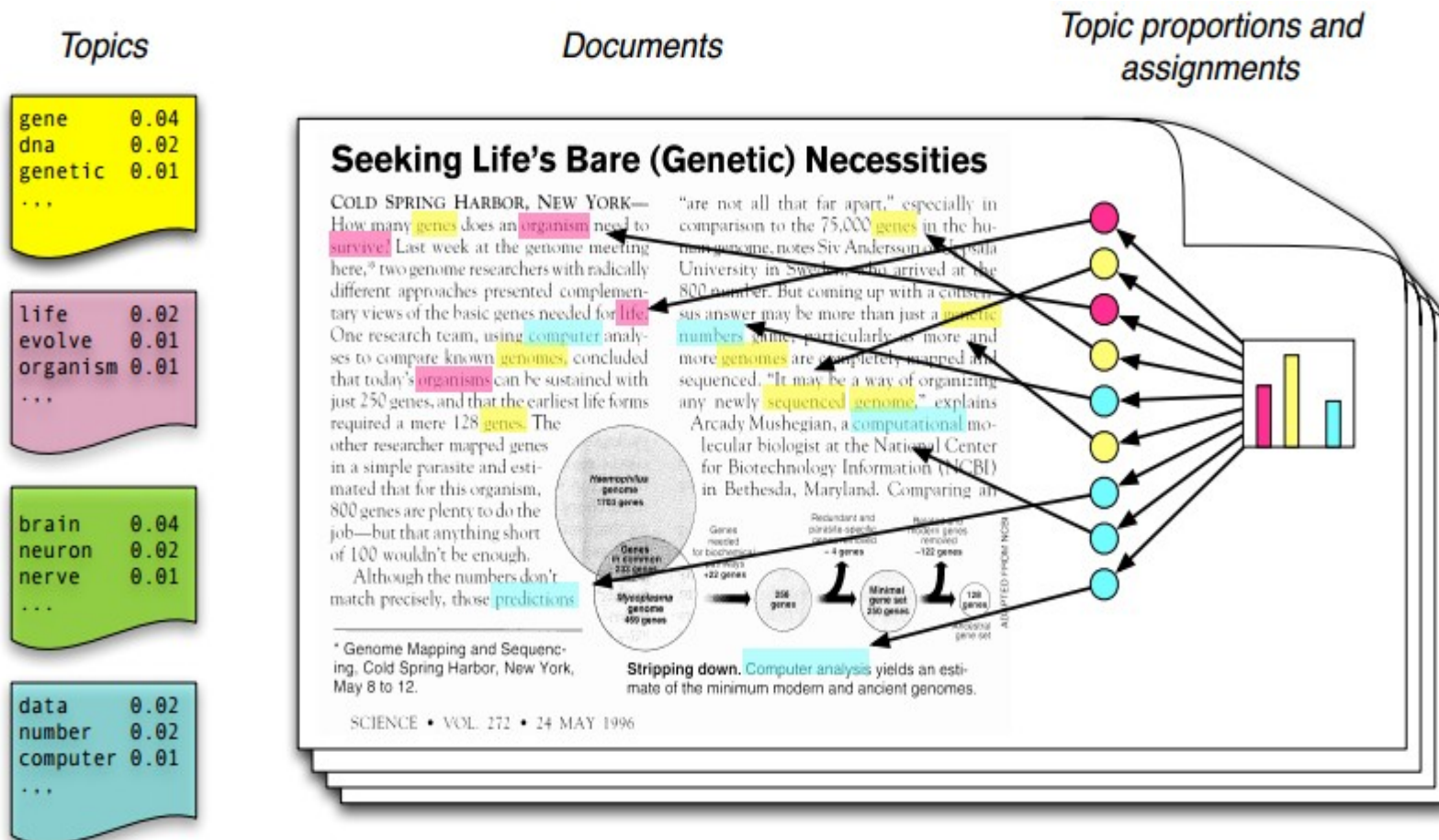
- We have a set of documents  $D_1, D_2, \dots, D_D$ .
- Each document is just a collection of words or a “**bag of words**”. Thus, the **order** of the words and the **grammatical role** of the words (subject, object, verbs, ...) are **not** considered in the model.
- Words like am/is/are/of/a/the/but/... can be eliminated from the documents as a preprocessing step since they don't carry any information about the “topics”.
- In fact, we can eliminate words that occur in at least %80 ~ %90 of the documents!
- Each document is composed of  $N$  “important” or “effective” words, and we want  $K$  topics.



## Probabilistic modeling

- 1 Treat data as observations that arise from a generative probabilistic process that includes hidden variables:
  - For documents, the hidden variables reflect the thematic structure of the collection.
- 2 Infer the hidden structure using posterior inference:
  - What are the topics that describe this collection?
- 3 Situate new data into the estimated model:
  - How does this query or new document fit into the estimated topic structure?

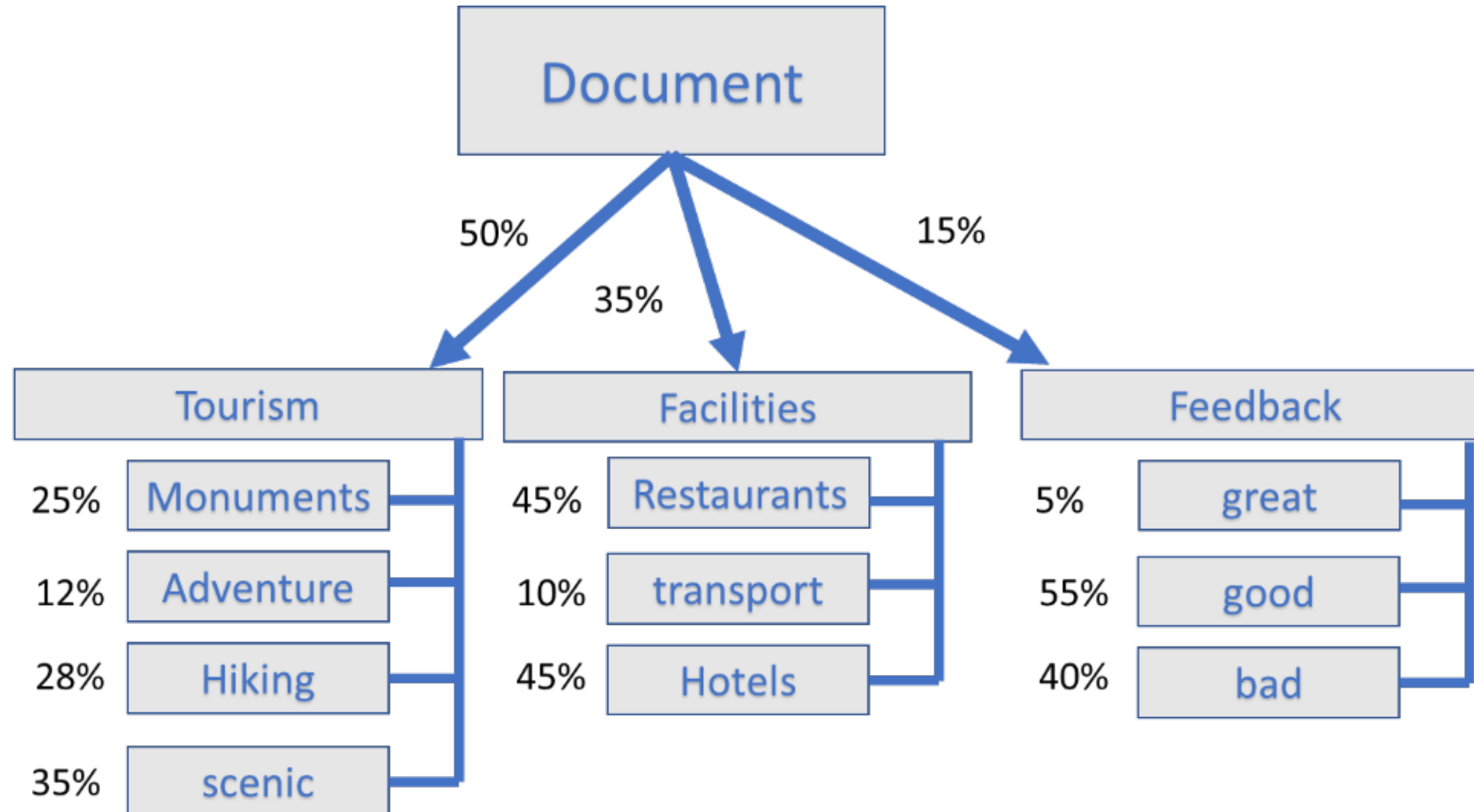
# LDA Model Definition



- Each **topic** is a distribution over words (topic-term matrix).
- Each **document** is a mixture of corpus-wide topics (document-topic matrix).
- Each **word** is drawn from one of these topics.
- We only observe the words within the documents and the other structure are **hidden variables**.

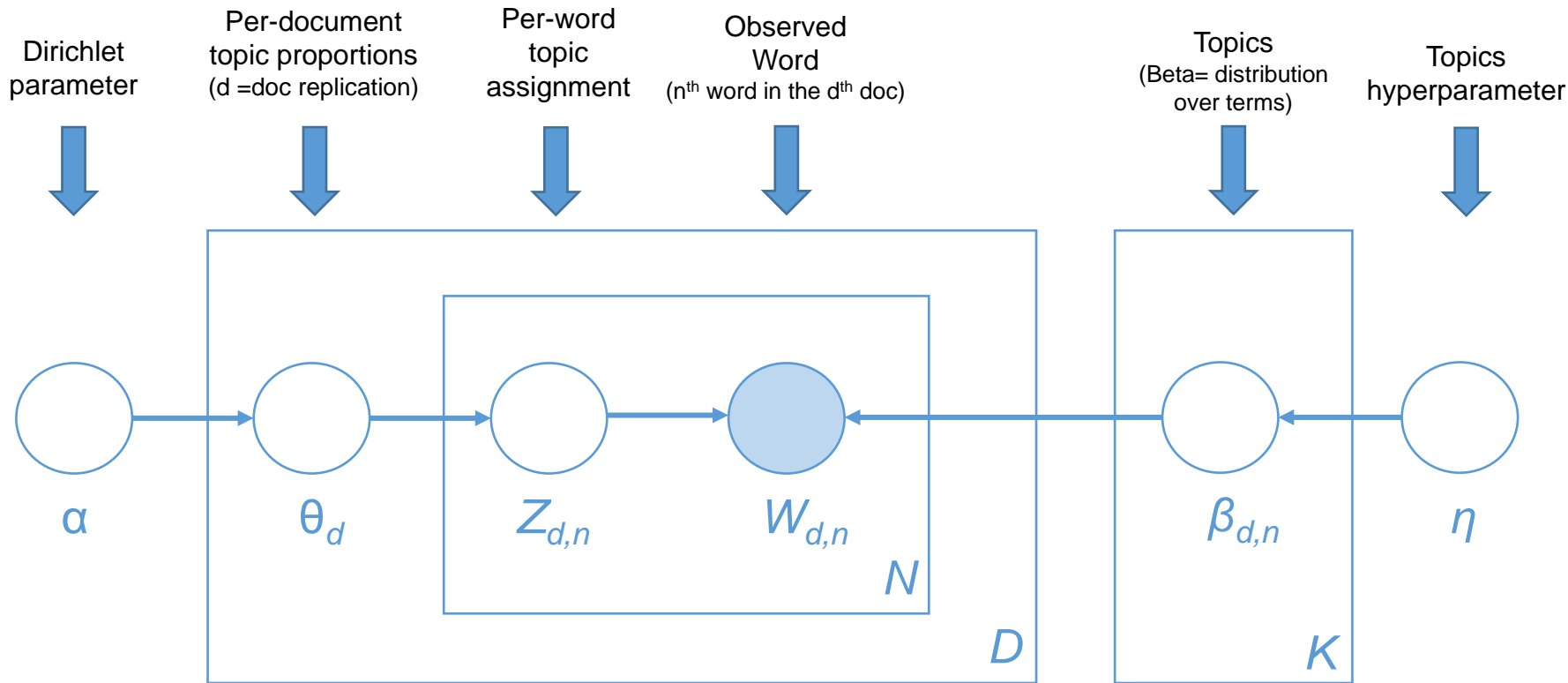
## LDA Model Definition

- In the document generation, first, a topic is selected from the **document-topic distribution** and later, from the selected topic, a word is selected from the **multinomial topic-word distributions**.



# LDA Model Definition

## LDA Graphical Model (Plate Notation)



Graphical model representation of LDA. The boxes are “plates” representing replicates.

The outer plate represents documents, while the inner plate represents the repeated choice of topics and words within a document.

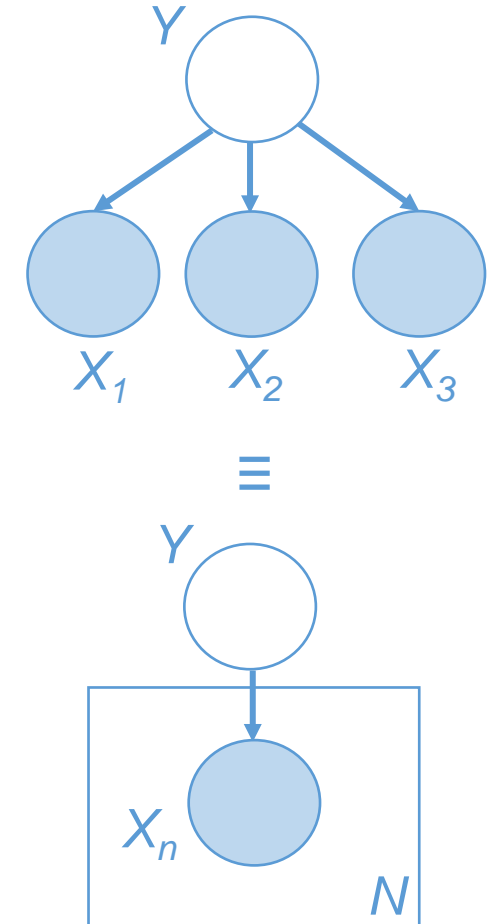
- Nodes are random variables
- Edges denote possible dependence
- **Observed variables are shaded**
- Plates denote replicated structure

$K$	specified number of topics	$i$	auxiliary index over words in a document
$k$	auxiliary index over topics	$\alpha$	positive $K$ -vector
$V$	number of words in vocabulary	$\beta$	positive $V$ -vector
$v$	auxiliary index over topics	$Dir(\alpha)$	a $K$ -dimensional Dirichlet
$d$	auxiliary index over documents	$Dir(\beta)$	a $V$ -dimensional Dirichlet
$N_d$	document length (number of words)	$z$	Topic indices: $z_{d,i} = k$ means that the $i$ -th word in the $d$ -th document is assigned to topic $k$

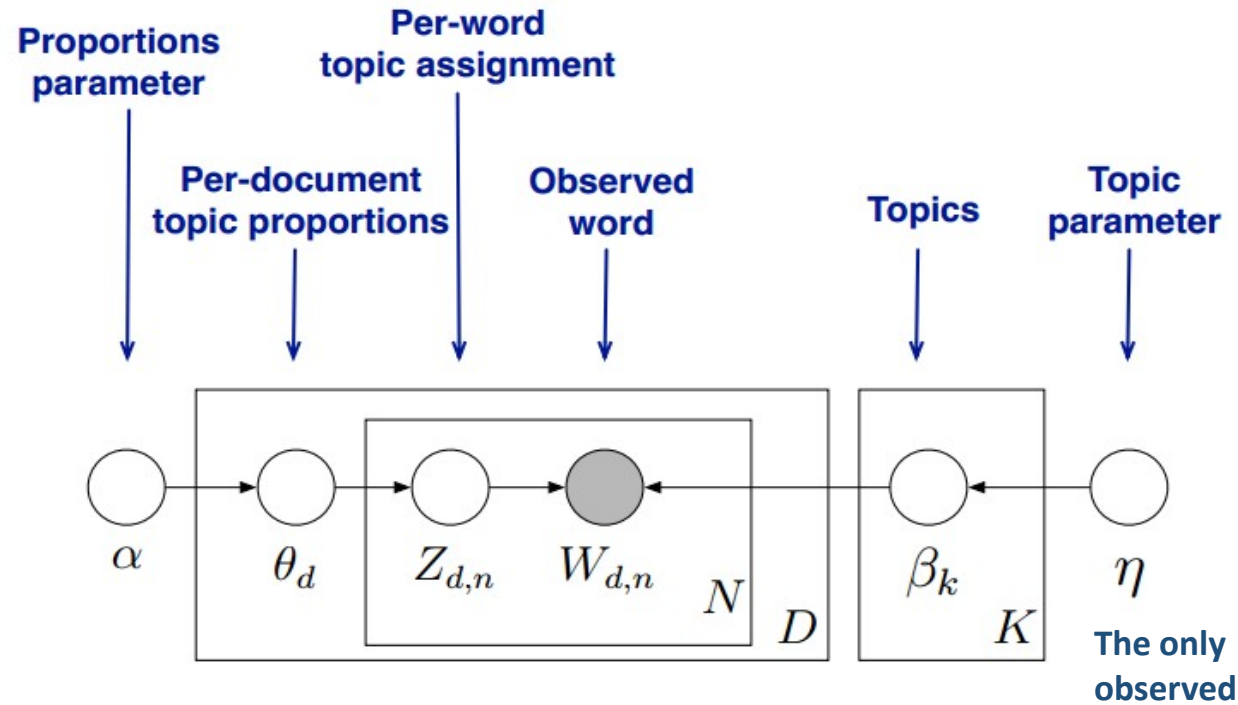
## Plates

$D$  = docs  
 $N$  = words  
 $K$  = topics

## Graphical models



# LDA Model Definition



$$p(\beta, \theta, \mathbf{z}, \mathbf{w}) = \left( \prod_{i=1}^K p(\beta_i | \eta) \right) \left( \prod_{d=1}^D p(\theta_d | \alpha) \prod_{n=1}^N p(z_{d,n} | \theta_d) p(w_{d,n} | \beta_{1:K}, z_{d,n}) \right)$$

The only observed

1) Draw each topic  $\beta_i \sim \text{Dir}(\eta)$  for  $i = 1, \dots, K$

2) For each document:

First, Draw topic proportions  $\theta_d \sim \text{Dir}(\alpha)$

For each word within the document:

a) Draw  $Z_{d,n} \sim \text{Multi}(\theta_d)$

b) Draw  $W_{d,n} \sim \text{Multi}(\beta_{z_{d,n}})$

This joint distribution defines a posterior  $p(\theta, z, \beta \mid w)$ .

From a collection of documents **we have to infer**:

1. Per-word topic assignment  $z_{d,n}$
2. Per-document topic proportions  $\theta_d$
3. Per-corpus topic distributions over words  $\beta_k$

Then use posterior expectations ( $E\{\beta_k \mid w\}$  for the corpus,  $E\{\theta_d \mid w\}$  for each document) to perform the task at hand: information retrieval, document similarity, exploration, and others.



## LDA Model Definition

Formal definition of the model:

$$p(\beta, \theta, z, w) = \left( \prod_{i=1}^K p(\beta_i | \eta) \right) \left( \prod_{d=1}^D p(\theta_d | \alpha) \prod_{n=1}^N p(z_{d,n} | \theta_d) p(w_{d,n} | \beta_{1:K}, z_{d,n}) \right)$$

$$(\beta_d | \eta) \sim \text{Dir}(\beta)$$

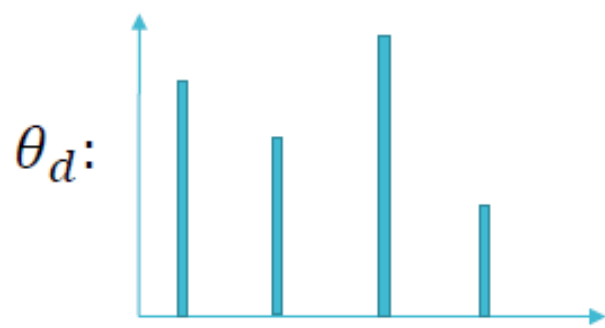
$$(\theta_d | \alpha) \sim \text{Dir}(\alpha)$$

$$Z_{d,n} \sim \text{Multi}(\theta_d)$$

$$W_{d,n} \sim \text{Multi}(\beta_{z_{d,n}})$$

$$p(z_{d,n} | \theta_d) = \theta_{d,z_{d,n}}$$

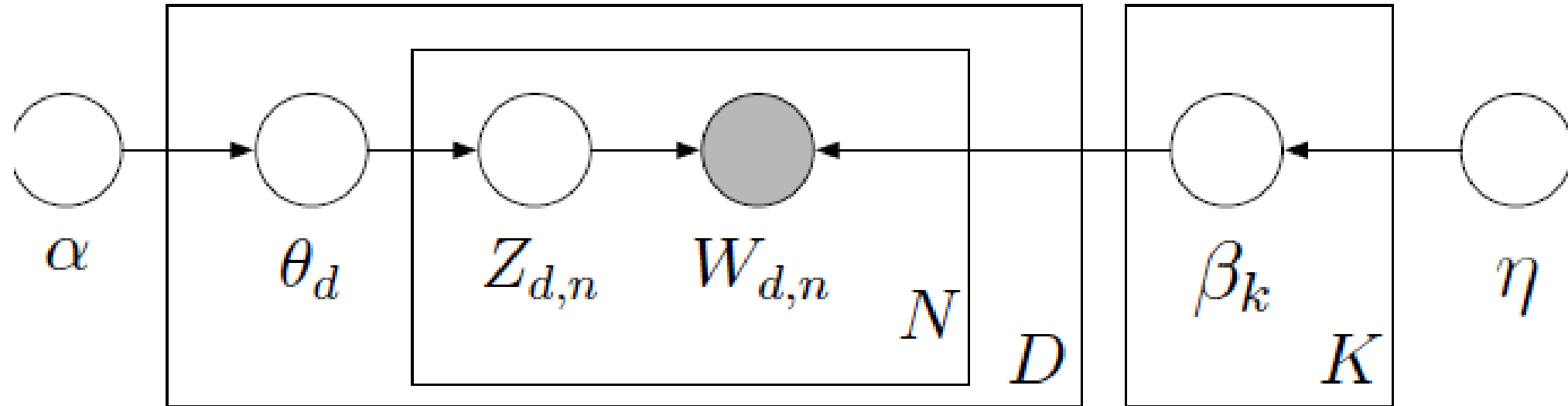
$$p(w_{d,n} | z_{d,n}, \beta_{1:K}) = \beta_{z_{d,n}, w_{d,n}}$$



$\beta$ :

Word probabilities for each topic			
Topics			

## LDA Model Definition



The probabilistic generative model estimation is intractable, so one of the following approximate posterior inference algorithms is used in modeling:

- Mean field variational methods (Blei et al., 2001, 2003).
- Expectation propagation (Minka and Lafferty, 2002).
- Collapsed Gibbs sampling (Griffiths and Steyvers, 2002).
- Collapsed variational inference (Teh et al., 2006).
- Online variational inference (Hoffman et al., 2010).

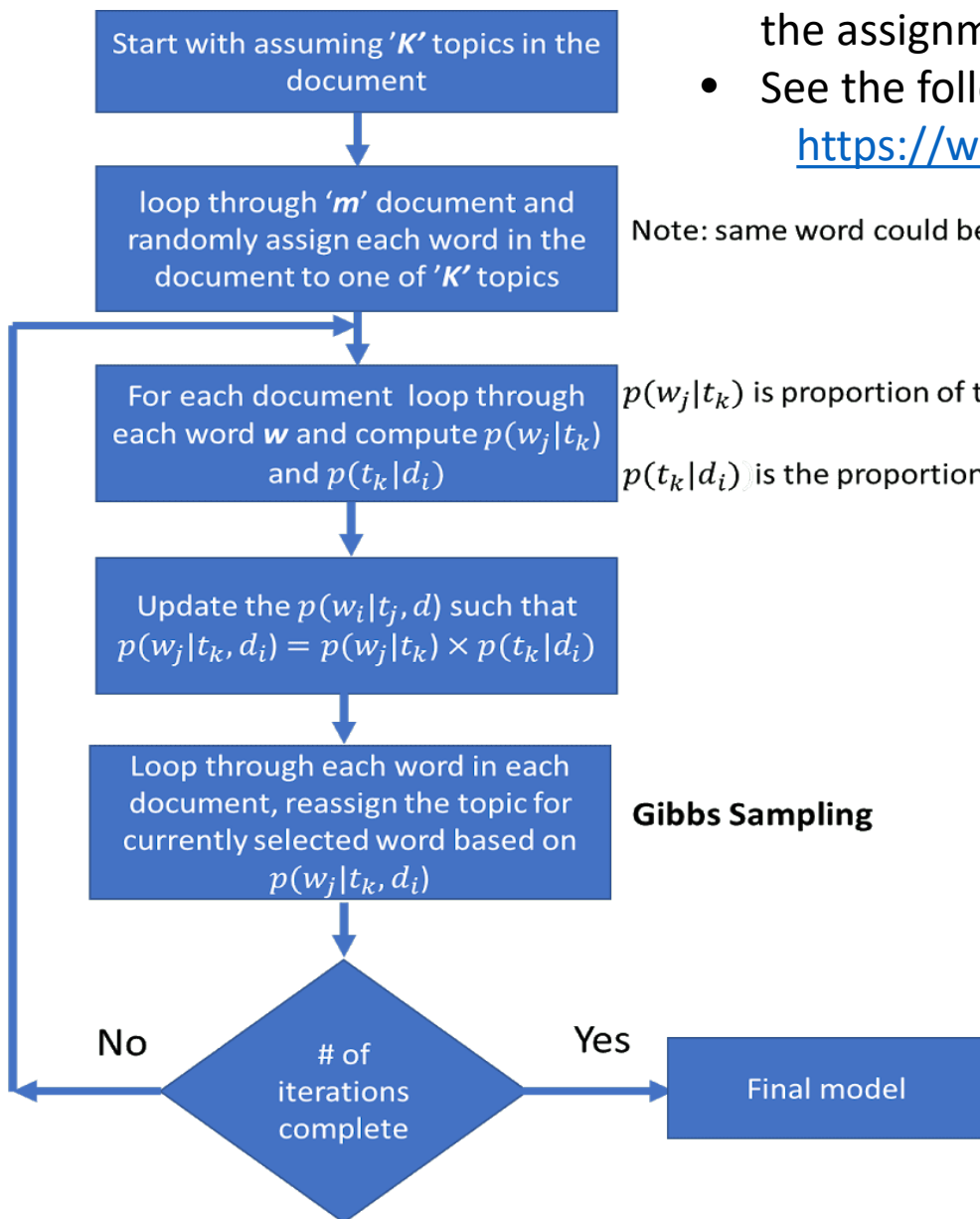
Also see Mukherjee and Blei (2009) and Asuncion et al. (2009).

# LDA Model Definition

- LDA begins with random assignment of topics to each word and iteratively improves the assignment of topics to words through **Gibbs sampling**.
- See the following link for more details on Gibbs sampling algorithm:

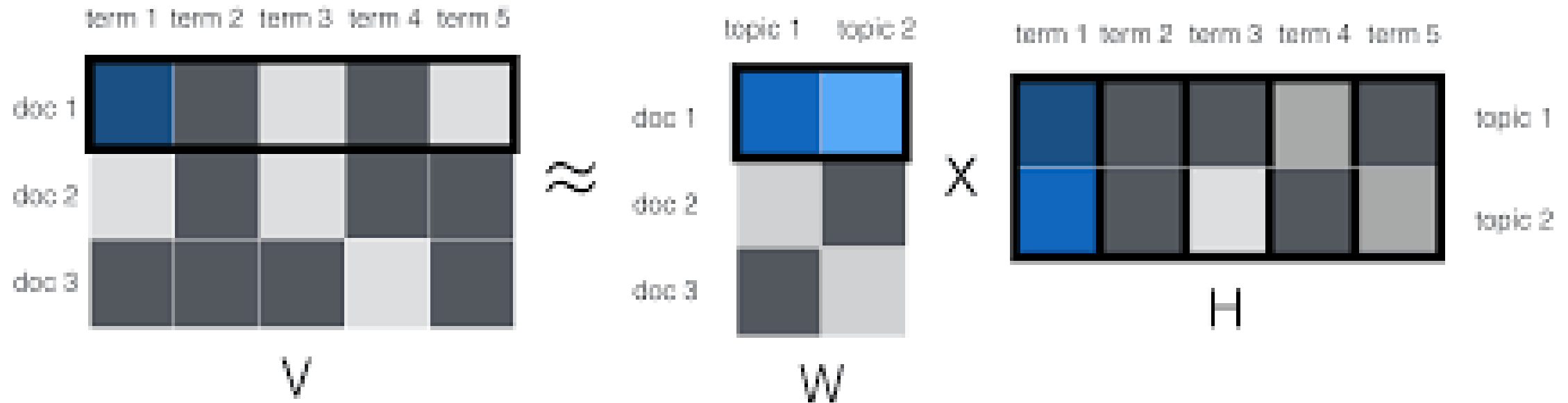
<https://www.mygreatlearning.com/blog/understanding-latent-dirichlet-allocation/>

Note: same word could be assigned different topics at different instances initially



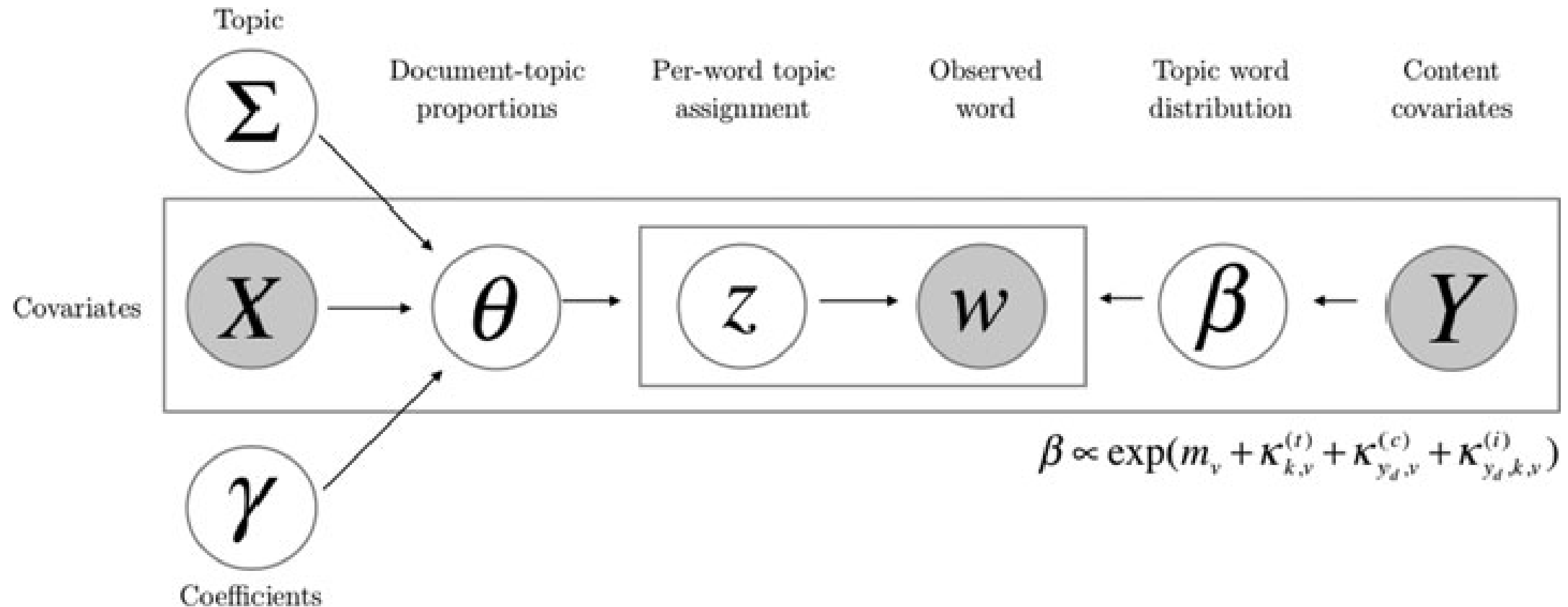
## LDA Model Definition

Input and output matrixes:



## Structural Topic Modeling (STM)

- STM allows for the inclusion of covariates of interest into the prior distributions for document-topic proportions and topic-word distributions.
- The effects of specific determinants on topic prevalence and topic content can be estimated with the model through a regression framework.



**Using R & Demo**



## LDA Example in R

- Text Mining package in R  $\longrightarrow$  `library("quanteda")`.
- Topic Models package in R  $\longrightarrow$  `library("topicmodels")`.
- Structural Topic Models package in R  $\longrightarrow$  `library("stm")`.
- We can perform preprocessing steps using functions in the “quanteda” package such as removing unimportant words (stopwords, ...).
- LDA function in the “topicmodels” package can fit the LDA model for a specific number of topics  $K$ .
- `R > LDA(x, k, method = "Gibbs", control = NULL, model = NULL, ...)`.

Data: Collection of Science from 1990-2000

17K documents

11M words

20K unique terms (redundant words removed)

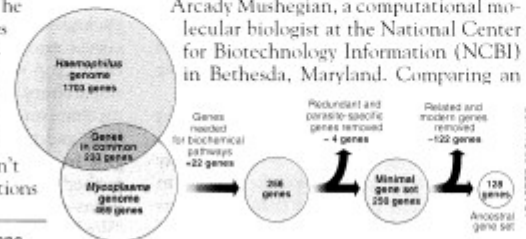
Model: 100-topic LDA model using variational inference

### Seeking Life's Bare (Genetic) Necessities

COLD SPRING HARBOR, NEW YORK—How many genes does an organism need to survive? Last week at the genome meeting here,\* two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today's organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.

Although the numbers don't match precisely, those predictions

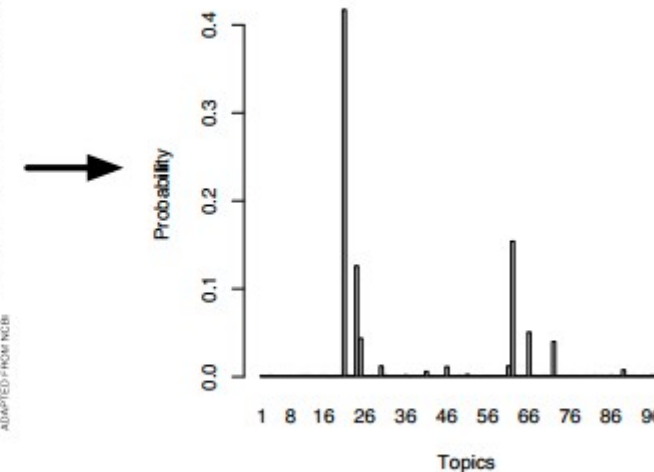
"are not all that far apart," especially in comparison to the 75,000 genes in the human genome, notes Siv Andersson of Uppsala University in Sweden, who arrived at the 800 number. But coming up with a consensus answer may be more than just a genetic numbers game, particularly as more and more genomes are completely mapped and sequenced. "It may be a way of organizing any newly sequenced genome," explains Arcady Mushegian, a computational molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an




\* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.

**Stripping down.** Computer analysis yields an estimate of the minimum modern and ancient genomes.

SCIENCE • VOL. 272 • 24 MAY 1996



Most probable words in four of the topics:



Topic 1	Topic 2	Topic 3	Topic 4
human	evolution	disease	computer
genome	evolutionary	host	models
dna	species	bacteria	information
genetic	organisms	diseases	data
genes	life	resistance	computers
sequence	origin	bacterial	system
gene	biology	new	network
molecular	groups	strains	systems
sequencing	phylogenetic	control	model
map	living	infectious	parallel
information	diversity	malaria	methods
genetics	group	parasite	networks
mapping	new	parasites	software
project	two	united	new
sequences	common	tuberculosis	simulations

## References

- Benoit, K., Watanabe, K., Wang, H., Nulty, P., Obeng, A., Müller, S., & Matsuo, A. (2018). quanteda: An R package for the quantitative analysis of textual data. *Journal of Open Source Software*, 3(30), 774.
- Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3(4–5), 993–1022.
- Grün, B., & Hornik, K. (2011). Topicmodels: An r package for fitting topic models. *Journal of Statistical Software*, 40(13), 1–30.
- Jelodar, H., Wang, Y., Yuan, C., Feng, X., Jiang, X., Li, Y., & Zhao, L. (2019). Latent Dirichlet allocation (LDA) and topic modeling: models, applications, a survey. *Multimedia Tools and Applications*, 78(11), 15169–15211.
- Roberts, M. E., Stewart, B. M., & Airolidi, E. M. (2016). A Model of Text for Experimentation in the Social Sciences. *Journal of the American Statistical Association*, 111(515), 988–1003.
- Roberts, M. E., Stewart, B. M., & Tingley, D. (2019). Stm: An R package for structural topic models. *Journal of Statistical Software*, 91(2).
- Roberts, M. E., Stewart, B. M., Tingley, D., Lucas, C., Leder-Luis, J., Gadarian, S. K., ... Rand, D. G. (2014). Structural topic models for open-ended survey responses. *American Journal of Political Science*, 58(4), 1064–1082.

**Why R?**

## What is R?

- R is a programming language and environment commonly used in statistical computing, data analytics and scientific research.
- It was developed in the early 1990s by Ross Ihaka and Robert Gentleman, and released in 2000.
- R is mostly used to retrieve, clean, analyze, visualize and present data.
- R can be regarded as an implementation of the S programming language.
- An environment for **statistical techniques** built into the **base R**, but many are supplied as **packages** (<https://cran.r-project.org/>).
- R is world's most widely used statistics programming language.
- Some companies using R: **Facebook, Google, Twitter, Uber, Airbnb**, etc.
- SAS/STATA/SPSS give copious statistical output, whereas R stores results in a fit **object** for subsequent interrogation by further R functions.
- R is an **object-oriented programming language**:

IN R, EVERYTHING THAT EXISTS IS AN OBJECT AND EVERYTHING THAT HAPPENS IS A FUNCTION CALL

- **The main reasons for using R are:**

- R and RStudio are open-source and free!
- R is a powerful data-analysis package with lots of user-contributed functions and packages.
- R is Interpreted (code can be executed as soon as it is written).
- Analyses done using R are reproducible.
- Using R makes collaboration easier.
- Finding answers to questions is much simpler (through Google).
- Struggling through programming helps you learn.



- Free resources available on the web, knowledge should be free!
  - **Textbooks:**
    - An Introduction to R (2020): <https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf>
    - Hands-On Programming with R: <https://rstudio-education.github.io/hopr/>
    - R for Data Science: <https://r4ds.had.co.nz/>
    - R for Data Science: Exercise Solutions: <https://jrnold.github.io/r4ds-exercise-solutions/>
  - **Free web tutorials:**
    - <https://www.datamentor.io/r-programming/>
    - <https://www.tutorialspoint.com/r/index.htm>
    - <https://www.learnbyexample.org/r-introduction/>
    - <https://www.statmethods.net/>
    - <https://www.r-exercises.com/2016/07/22/start-here-to-learn-r/>
  - **Documentation and R communities:**
    - Documentation for packages and functions: <https://www.rdocumentation.org/>
    - Documentation for packages and functions: <https://rdr.io/>
    - R-BLOGGERS (news and tutorials): <https://www.r-bloggers.com/>
    - StackOverflow (Q&A): <https://stackoverflow.com/questions/tagged/r>

- Google is your best friend.
- Learning by doing, possibly with others!
- Copy, paste and tweak strategy.
- Last but not least passion, curiosity and discipline.
- It is never too late.
- You do not need a degree in computer science to learn coding.

- See the example of **Garrett Grolemund**:
  - Garrett Grolemund is the **co-author of R for Data Science** and **author of Hands-On Programming with R**. He wrote the **lubridate R package** and works for **RStudio** as an **advocate who trains engineers to do data science with R and the Tidyverse**. If you use R yourself, you may recognize Garrett from his video courses on **Datacamp.com** and **O'Reilly media**, or for his series of popular R cheatsheets distributed by **RStudio**. Garrett earned his PhD in Statistics from **Rice University** in 2012 under the guidance of **Hadley Wickham**. Before that, **he earned a Bachelor's degree in Psychology from Harvard University and briefly attended law school**. Garrett has been one of the foremost promoters of **Shiny**, **R Markdown**, and the **Tidyverse**, documenting and explaining each in detail.

## Comparison R vs. Python

- R and Python are the most popular data analysis software.
- R is more **functional**; Python is more **object-oriented**.
- R has more data analysis functionality built-in; Python relies on packages.
- Python has “main” packages for data analysis tasks, R has a larger ecosystem of small packages.
- **R has more statistical support in general.**
- **It's usually more straightforward to do non-statistical tasks in Python (e.g., web scraping).**
- There are many parallels between the data analysis workflow in both.
- Use R to analyze a dataset and present the findings in a research paper.
- Use Python to write a data analysis program that runs in a distributed system and interacts with lots of other components.
- Consider your network in your choice.
- **Personal recommendation:** start with R for research and learn Python in your free time.

- Coding is political ([https://ehmatthes.github.io/pcc\\_2e/](https://ehmatthes.github.io/pcc_2e/)):
  - “I've often been asked why I care so much about helping people learn to code. I've always responded by saying that **code is power and learning to code gives you power. The more you understand code, the more you understand how much impact tech platforms and products have on people's lives** - whether through the implementation of specific features, or the lack of implementation of certain features. It matters little whether these impacts are intentional or not. **When you've built a platform that a significant part of society uses for communication, for example, your code has direct impact on society itself.** When you write code that helps determine who gets a loan, you impact who can afford to buy a home. **It is abundantly clear in the US, and in many places around the world, that those who have power are desperate to hold on to that power and are willing to use force - "any means necessary" is an all-too-common phrase - to hold on to that power. And that force is disproportionately, to a mind-boggling degree, used against Black people, Indigenous people, and anyone who can be classified as a minority.** As you learn to code, please be aware of the power you are gaining. As you work on projects, whether your own projects or those that are controlled by someone else, please focus on projects that share power. **Please refuse to work on projects that consolidate power, especially for those who will use it against others**” (Python Crash Course - Eric Matthes)