A Primer in Text Mining with R

Paul Hofmarcher

November 20, 2024

What is Text Mining?

- **Definition:** Text mining is the process of extracting meaningful information, patterns, and insights from large volumes of textual data.
- **Purpose:**
  - Transform unstructured text data into structured formats.
  - Enable analysis, categorization, and actionable insights.
- **Fields:** Data science, business analytics, economics, political science, and more.

Outline I

- **Methods:**
  - Develop basic skills in Text Mining with R.
  - Convert natural language texts (e.g., legal and political documents) into data.
- **Idea:**
  - Relate text data to metadata to understand economic, political, and social forces.
- **Applications:**
  - Analyze the motivations and decisions of public officials through their writings and speeches.
  - Explore topics driving polarization in the US Congress.
  - Understand communication from central banks for managing expectations.

Schedule

- Two lectures of about 1 hour.
- Plenty of time for going over programming code.
- Provide helpful code snippets for practical use.

Key Techniques in Text Mining

- **Natural Language Processing (NLP):**
  - Regular Expressions: Patterns for matching text.
  - Tokenization: Breaking text into words or phrases.
  - Stemming: Reducing words to their root forms.
  - Part Of Speech tagging: nouns, adverbs...
- **Sentiment Analysis:** Determining emotional tone within text (dictionary based).
- **Scaling Model:** Ideology in text.
- **Topic Modeling:** Uncovering hidden themes or topics within text data.

Example: ELIZA Dialogue

User: I am unhappy.
ELIZA: DO YOU THINK COMING HERE WILL HELP YOU NOT TO BE UNHAPPY?

User: I need some help, that much seems certain.
ELIZA: WHAT WOULD IT MEAN TO YOU IF YOU GOT SOME HELP?

User: Perhaps I could learn to get along with my mother.
ELIZA: TELL ME MORE ABOUT YOUR FAMILY.

User: My mother takes care of me.
ELIZA: WHO ELSE IN YOUR FAMILY TAKES CARE OF YOU?

User: My father.
ELIZA: YOUR FATHER.

User: You are like my father in some ways.

ELIZA: Early NLP and Conversational AI

ELIZA is one of the earliest examples of natural language processing and conversational AI **(Weizenbaum, 1966)**.

1. **Pattern Matching:**
   - When a user types something, ELIZA searches the text for keywords or phrases that match predefined templates in its script.
2. **Scripts and Transformations:**
   - Each keyword is associated with a set of possible responses or transformations.
   - For example, if the user mentions "mother," ELIZA might respond with "Tell me more about your family."
3. **Pronoun Substitution:**
   - For input like "I am unhappy," ELIZA may replace "I" with "you" and "am" with "are" to generate a response.
4. **Fallback Responses:**
   - If ELIZA doesn't recognize any keywords, it uses generic responses like "I see," or "Can you elaborate on that?"

Why RegEx Are Core to Text Mining

Regular Expressions (RegEx) are fundamental tools in text mining because they:

- **Efficiently Process Text:** Enable searching, matching, and extracting patterns in large text datasets.
- **Transform Unstructured Data:** Convert raw text into structured formats, essential for analysis.
- **Versatility:** Work across programming languages (e.g., R, Python, Perl) and applications.
- **Customization:** Allow users to define precise patterns to extract meaningful information.
- **Applications:** Power tasks like:
  - Cleaning and preprocessing text data.
  - Extracting specific entities (e.g., dates, emails, or keywords).
  - Analyzing patterns in natural language.

Regular Expressions (RegEx)

- **Definition:** A sequence of characters that defines a search pattern.
- **Uses:**
  - Match, search, and manipulate text.
  - Data extraction and validation.
- Widely implemented in languages like Perl and Python.
- In R, RegEx is compatible with Perl syntax (`perl=TRUE`).
- We discuss two main functions for RegEx in R: `grep()` and `gsub()`.

Metacharacters in RegEx

- **Key Metacharacters:**
  - `*`: Matches 0 or more occurrences.
  - `+`: Matches 1 or more occurrences.
  - `?`: Matches 0 or 1 occurrence.
  - `.`: Matches any character.
  - `$`: Matches the end of a string.
  - `[...]`: Matches any character in brackets.
  - `^`: Matches the start of a string.

Metacharacters in RegEx (I)

In RegEx, there are the following (and more) metacharacters:

- `*` ... The preceding expression can appear any number of times (including zero times).
- `+` ... The preceding expression must appear at least once, but can appear multiple times (similar to `*`).
- `?` ... The preceding expression is optional; it can appear once, but it doesn't have to.
- `.` ... Represents any character in this position.
- `$` ..."Look at the end of the string." For example, `fox$` finds "silverfox" but not "fox jumped."
- `[...]` ... Square brackets are used for character selection.
- `^` ... Circumflex within `[...]` represents negation within a character set.

Metacharacters in RegEx (II)

Additional metacharacters include:

- `^` ... Circumflex outside `[...]` matches the starting position of a string.
- `{n}` ... Matches the preceding character exactly `n` times. For example, `[0-9]{3}-[0-9]{4}` matches all numbers in the format `123-1234`.
- `|` ... Logical "or." For example, `gr(a|e)y` matches both "gray" and "grey."
- `{m,n}` ... Matches the preceding character at least `m` times but no more than `n` times (`{m,}` matches `m` times or more).
- `(...)` ... Used for "grouping," e.g., `"H(a|ae|)ndel"`.

The `sub()` Function in R

Purpose: Replaces the **first occurrence** of a pattern in a string.

Syntax:

- `sub(pattern, replacement, x)`
- `pattern`: A regular expression to search for.
- `replacement`: The string to replace the matched pattern.
- `x`: A character vector to search and replace in.

The `gsub()` Function in R

Purpose: Replaces **all occurrences** of a pattern in a string.

Syntax:

- `gsub(pattern, replacement, x)`
- `pattern`: A regular expression to search for.
- `replacement`: The string to replace the matched pattern.
- `x`: A character vector to search and replace in.

Part-of-Speech

Part-of-Speech (POS) tagging is crucial in Natural Language Processing (NLP) because:

- **Grammatical Context:** Provides insights into the structure and meaning of sentences.
- **Text Analysis:** Helps in tasks such as:
  - Syntax analysis.
  - Named Entity Recognition (NER).
  - Sentiment analysis.
- **Linguistic Understanding:** Identifies parts of speech (e.g., nouns, verbs, adjectives), enabling deeper text comprehension.
- **Applications:**
  - Improving search engines with context-aware queries.
  - Text summarization.
  - Speech-to-text systems.

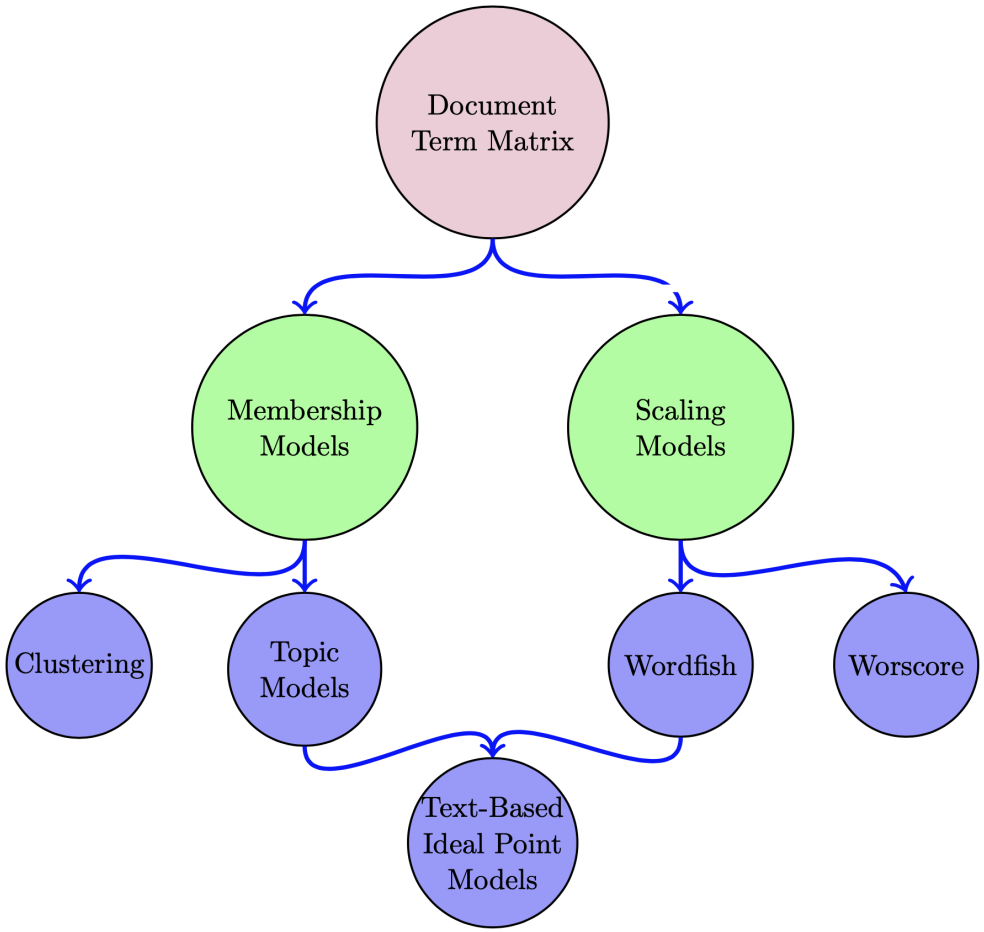
Some maybe useful packages

- `text`:
  - Link `R` with Transformers from Hugging Face to transform text variables to word embeddings;.
  - e.g., `textNER` EXPERIMENTAL
- `openNLP`
- `spacyr` provides a convenient `R` wrapper around the Python `spaCy` package.
- `udpipe` is a R package for:
  - Tokenization, POS-tagging, lemmatization, and dependency parsing.
  - Supports 64 languages.
  - Does not require external software.
  - Easy to train and tune custom models.

Part-of-Speech (POS) Tagging

- Part-of-Speech (POS) Tagging is the process of assigning a part of speech (e.g., noun, verb, adjective) to each word in a sentence.
- POS tags provide grammatical context, helping with understanding the structure and meaning of sentences.
- **Example:**
  - Sentence: "The quick brown fox jumps over the lazy dog."
  - POS Tags:
    - `The` - Determiner
    - `quick` - Adjective
    - `fox` - Noun
    - `jumps` - Verb
    - etc.
- POS-Tagging is crucial in NLP tasks such as syntax analysis, named entity recognition, and sentiment analysis.

Scaling and Content



Document-Term Matrix (DTM)

- A Document-Term Matrix (DTM) is a mathematical representation of a text corpus.
- It represents the frequency of terms (words) that appear in a collection of documents.
- The matrix has:
  - Rows: Documents in the corpus
  - Columns: Terms (words) from the entire corpus vocabulary
  - Values: Frequency or occurrence of a term in a document
- Example:

Document	**apple**	**banana**	**fruit**
Doc 1	1	0	1
Doc 2	0	2	1
Doc 3	1	1	2

- DTMs are fundamental in text analysis and information retrieval, enabling methods such as topic modeling, clustering, and classification.

Wordfish Model (Slapin & Proksch, 2008): Overview

- **Goal**: Estimate the ideological positions of political parties from text using word frequencies.
- **Latent Ideological Scale**:
  - Parties are positioned on a one-dimensional ideological scale.
  - The position is inferred from the frequency distribution of words across documents.
- **Advantages over Wordscore**:
  - Does not require predefined reference texts.
  - Flexible: Adaptive over time as more documents are analyzed.
- **Document Length Handling**: The model accounts for varying lengths of documents.

Wordfish Model

The Wordfish model estimates the latent ideological position of a party using the following probabilistic model for word frequencies:

$$y_{ijt} \sim \text{Poisson}(\lambda_{ijt})$$

Where:

- y_{ijt} is the count of word j in party's manifesto i at time t,
- λ_{it} is the expected frequency of word i in document t, modeled as:

$$\lambda_{ijt} = \exp(\mu_{it} + \omega_j + \beta_j \cdot x_{it})$$

- μ_{it} represents the document-level (party) time fixed effect. (length of the document)
- ω_j set of words fixed effects.
- β_j is an estimate of a word specific weight capturing the importance of word j in discriminating between party positions.
- x_{it} is the estimate of party i's position in election year t.

Textual Analysis and Topic Models

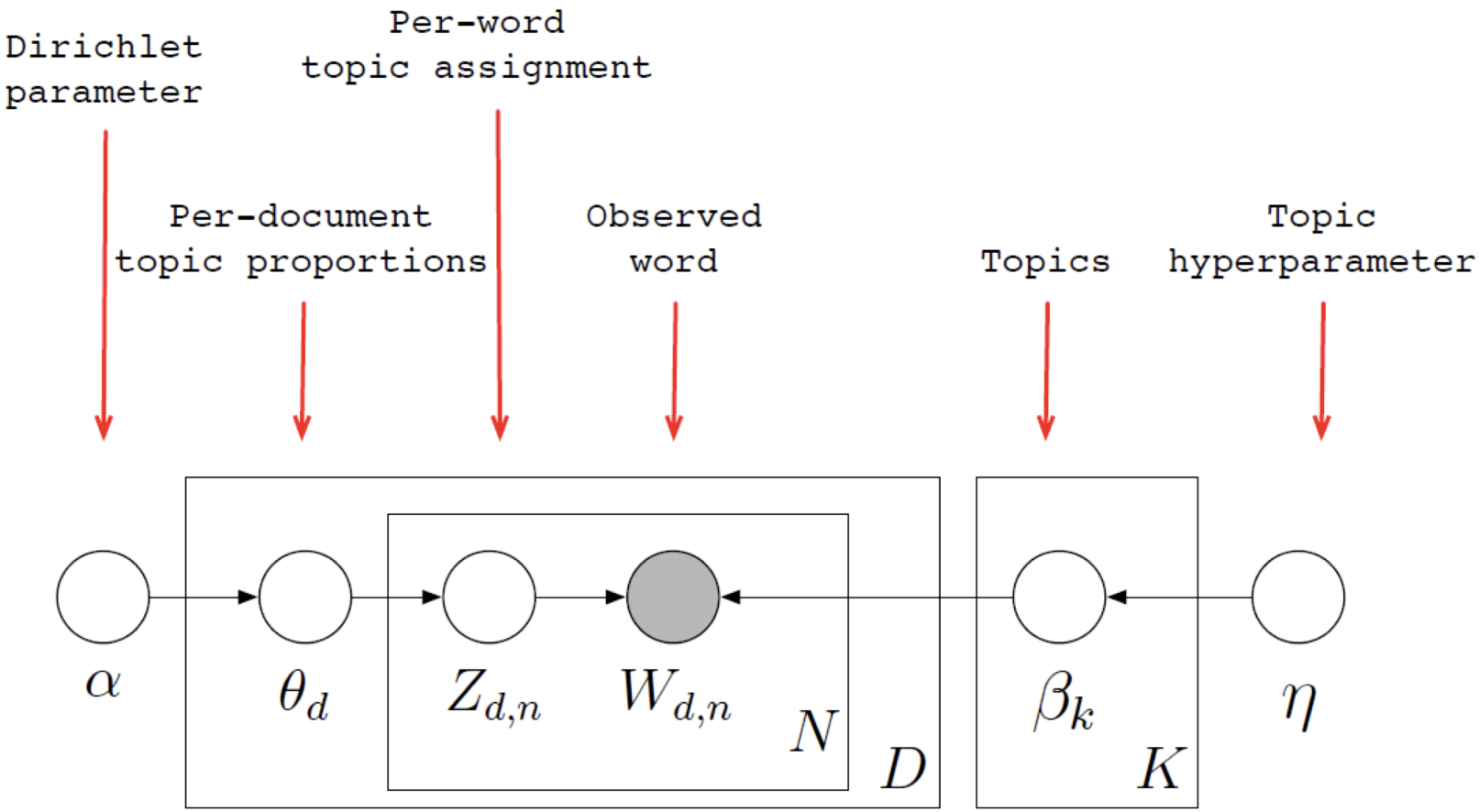
- Latent Semantic Indexing (LSI) (Deerwester1990) based on SVD (Factor Analysis).
- Probabilistic LSI (pLSI) (Hoffman,1999) improves LSI by introducing a probabilistic foundation.
- Latent Dirichlet Allocation (LDA) (Blei et.al 2003) clusters words into topics and represents documents as mixtures of topics.("Gold standard")
- Structural Topic Models (STMs) (Roberts et al, 2016) extend LDA by incorporating explanatory covariates.
- *Wordscore* project ideological positions to a real line.
- (Vafa et al 2020) propose Text-Based Ideal Point Models (TBIP), combining political science scoring and topic models:
  - Identifies ideological positions for each topic
  - Describes how wording changes with the author's ideological position

Methods of Unsupervised Text Analysis

- **Goal**: Describe main themes of a corpus (collection of texts/speeches).
- **Steps**:
  - Start with a **Document-Term Matrix (DTM)**.
  - Specify a **statistical model** for how the text was generated.
  - Find the most likely **topics** that generated the text.
- **Key Features**:
  - Similar to **clustering**.
  - Many variants of **topic models** exist:
    - Latent Dirichlet Allocation (LDA).
    - Correlated Topic Model (CTM).
    - Structural Topic Model (STM).

Latent Dirichlet Allocation (Blei et al. 2003)

- **Idea**:
  - Each document is a **mixture over topics**.
  - Each topic is a **mixture over words**.
- **Latent Dirichlet Allocation estimates**:
  - The **distribution over words** for each topic.
  - The **proportion of each document** in each topic.
- **Mixed Membership**:
  - Each document is assigned to **several topics**.
- **Maintained Assumptions**:
  - **Bag of words**: Assumes word order does not matter.
  - A **fixed number of topics** must be set ex ante.


Dirichlet parameter
Per-word topic assignment
Per-document topic proportions
Observed word
Topics
Topic hyperparameter
α
θ_d
$Z_{d,n}$
$W_{d,n}$
N
D
β_k
K
η

Latent Dirichlet Allocation: Generative Process

1. For each document d, draw a distribution θ_d over topics from a Dirichlet(α).
2. For each topic k, draw a distribution β_k over words from a Dirichlet(η).
3. For the n-th word in document d:
   - 3.1 Draw a topic z_{nd} from a Categorical(θ_d).
   - 3.2 Draw a word w_{nd} from a Categorical($\beta_{z_{nd}}$).

LDA

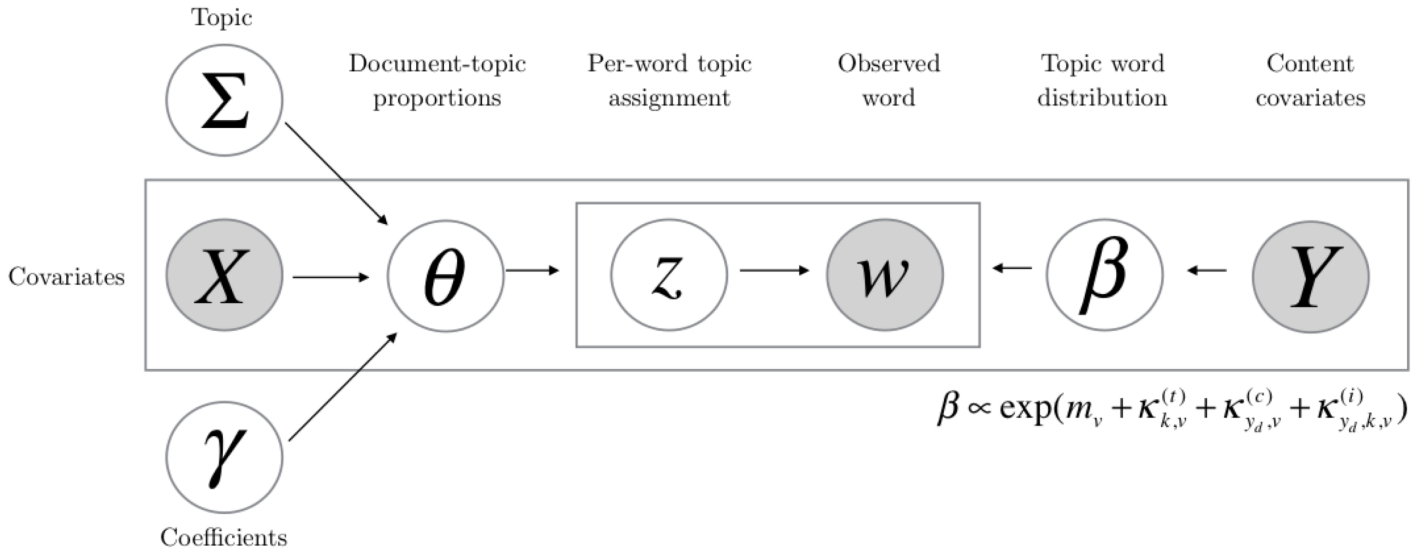
Two primary matrices of interest:

1. **Document-Topic Matrix**:
   - Each row corresponds to a document.
   - Each column corresponds to a topic.
   - Values represent the proportion of each topic in a document.
2. **Topic-Word Matrix**:
   - Each row corresponds to a topic.
   - Each column corresponds to a word.
   - Values represent the probability of each word in a topic.

$$\boldsymbol{\theta} = \left[\begin{array}{c|cccc} & \textit{Topic}1 & \textit{Topic}2 & \dots & \textit{TopicK} \\ \hline \textit{Doc}1 & .2 & .1 & \dots & 0.05 \\ \textit{Doc}2 & .2 & .1 & \dots & .3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \textit{DocD} & 0 & 0 & \dots & .5 \end{array}\right] \qquad \boldsymbol{\beta}^{\boldsymbol{T}} = \left[\begin{array}{c|cccc} & \textit{Topic}1 & \textit{Topic}2 & \dots & \textit{TopicK} \\ \hline \text{“}\textit{text}\text{”} & .02 & .001 & \dots & 0.001 \\ \text{“}\textit{data}\text{”} & .001 & .02 & \dots & 0.001 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \text{“}\textit{analysis}\text{”} & .01 & .01 & \dots & 0.0005 \end{array}\right]$$

Structural Topic Models (STM): Extending LDA

- **LDA Limitations**: LDA assumes that topic distributions are fixed for each document and do not vary with external covariates or document metadata.
- **STM Extension**: STM extends LDA by incorporating **covariates** (e.g., time, author, or other document-level features) to explain the variation in topic distributions.
- **Key Features of STM**:
  - **Topic modeling with covariates**: Allows external factors to influence topic proportions.
  - **Dynamic topics**: Can track how topics evolve over time or across different conditions.
  - **User-defined covariates**: Topic distributions are influenced by metadata (e.g., document-level features like political affiliation).
- **Result**: STM provides more nuanced insights, revealing how topics are influenced by and evolve with external factors.


Topic
Document-topic proportions
Per-word topic assignment
Observed word
Topic word distribution
Content covariates
Covariates
Coefficients
Σ
X
θ
z
w
β
Y
γ
$\beta \propto \exp(m_v + \kappa^{(t)}_{k,v} + \kappa^{(c)}_{y_d,v} + \kappa^{(i)}_{y_d,k,v})$

Combining Topic Models with Scaling Models

- Poisson factorization topic model, bigrams y_{dv}:

$$y_{dv} \sim \text{Pois}(\lambda_{dv}) \quad \text{where} \quad \lambda_{dv} = \boldsymbol{\theta}_d \boldsymbol{\beta}_v = \sum_{k=1}^{K} \theta_{dk} \beta_{kv}$$

- We follow Vavra et al. (2024), Structural Text-Based Scaling Model (`https://arxiv.org/abs/2410.11897`):

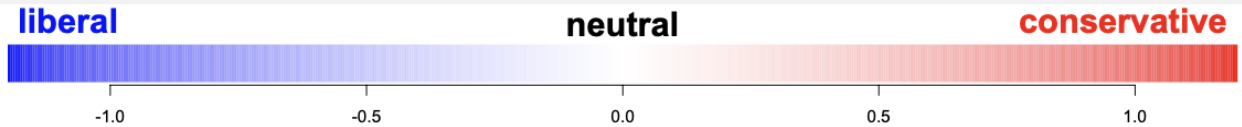
$$y_{dv} \sim \text{Pois}(\lambda_{dv}) \quad \text{where} \quad \lambda_{dv} = w_{a_d} \sum_{k=1}^{K} \lambda_{dkv} \quad \text{and} \quad \lambda_{dkv} = \theta_{dk} \beta_{kv}$$

a_d	author of document d
w_a	*verbosity* of author a
$\exp\{\eta_{kv} l_{a_d}\}$	*framing part*
l_{ka_d}	*ideological position* of author a for topic k
η_{kv}	*polarization* value of term v in topic k

Framing

Interpretation of $\beta_{kv} \exp\{\eta_{kv} \mathring{\imath}_{a_d}\}$

ideological position $\mathring{\imath}_a$:



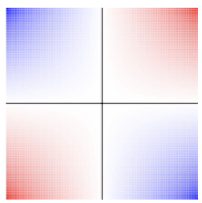
Let $k \approx$ „gun violence“

- If $\eta_{kv} < 0$
  - mass shooting

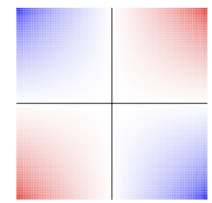
  $\beta_{kv} \exp\{-\eta_{kv}\} > \beta_{kv} > \beta_{kv} \exp\{+\eta_{kv}\}$
- If $\eta_{kv} = 0$
  - gun violence

  $\beta_{kv} = \beta_{kv} = \beta_{kv}$
- If $\eta_{kv} > 0$
  - terrorist attack

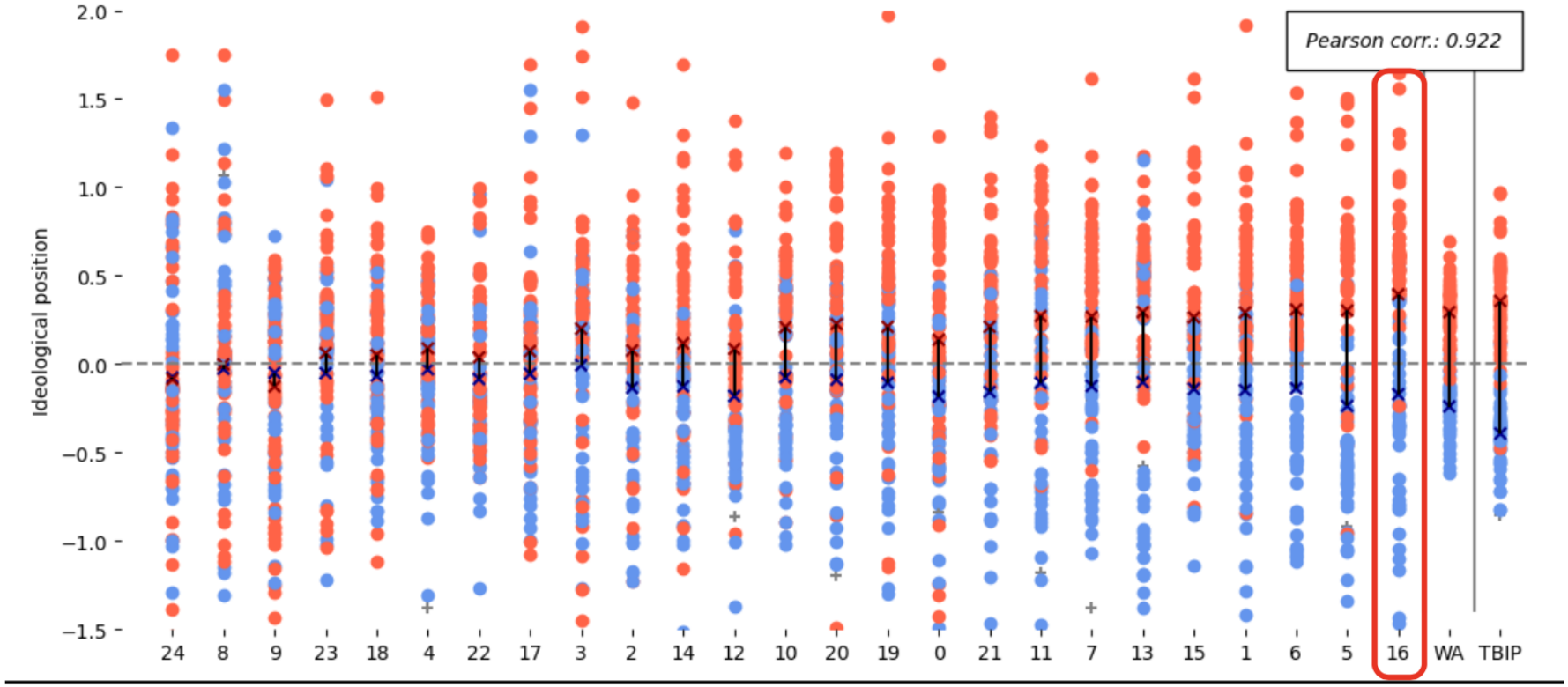
  $\beta_{kv} \exp\{-\eta_{kv}\} < \beta_{kv} < \beta_{kv} \exp\{+\eta_{kv}\}$



Beware of the ambiguity $\eta_{kv} \mathring{\imath}_{a_d} = \eta_{kv}(-1)(-1)\mathring{\imath}_{a_d} = (-\eta_{kv})(-\mathring{\imath}_{a_d})$!

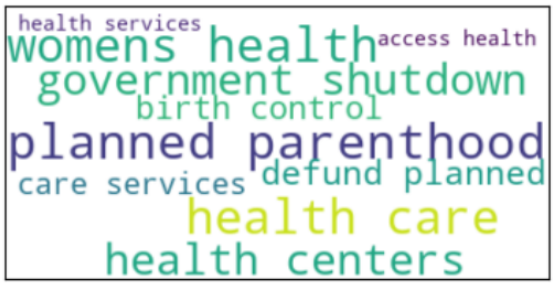


Scaling Positions



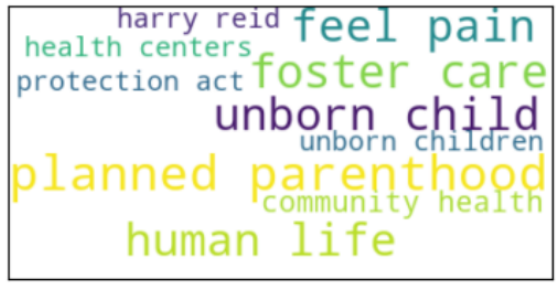
Abortion

E log(beta) - eta



E log(beta)

E log(beta) + eta

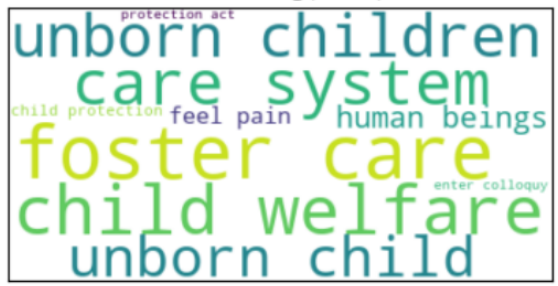


E -eta under log(beta) > -1.0

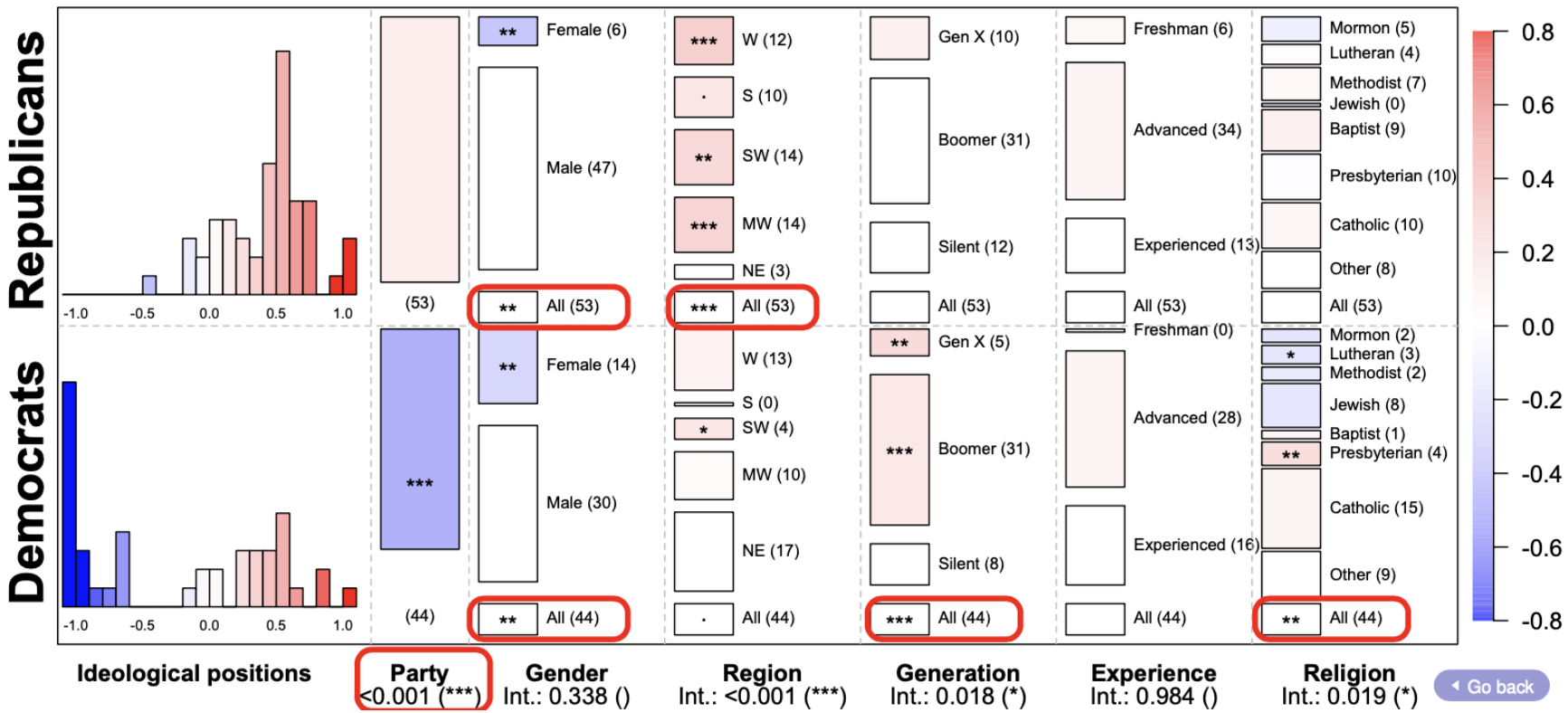
Topic 13

Go back

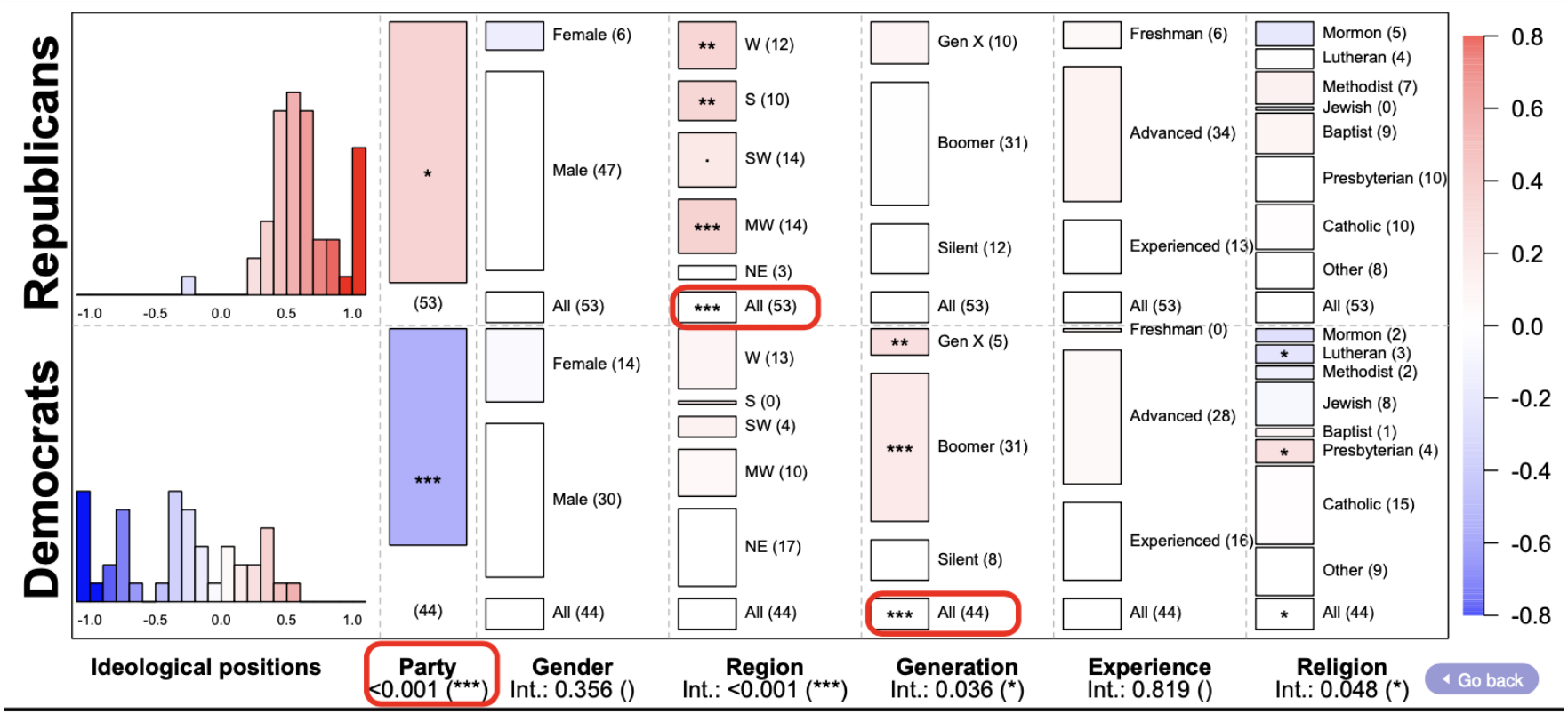
E +eta under log(beta) > -1.0



Abortion

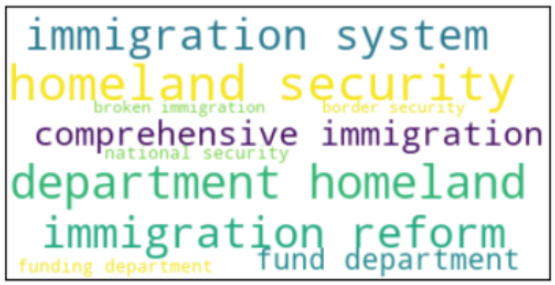

Republicans
Democrats
-1.0
-0.5
0.0
0.5
1.0
(53)
(44)
***
**
Female (6)
Male (47)
All (53)
Female (14)
Male (30)
All (44)
W (12)
S (10)
SW (14)
MW (14)
NE (3)
All (53)
W (13)
S (0)
SW (4)
MW (10)
NE (17)
All (44)
Gen X (10)
Boomer (31)
Silent (12)
All (53)
Gen X (5)
Boomer (31)
Silent (8)
All (44)
Freshman (6)
Advanced (34)
Experienced (13)
All (53)
Freshman (0)
Advanced (28)
Experienced (16)
All (44)
Mormon (5)
Lutheran (4)
Methodist (7)
Jewish (0)
Baptist (9)
Presbyterian (10)
Catholic (10)
Other (8)
All (53)
Mormon (2)
Lutheran (3)
Methodist (2)
Jewish (8)
Baptist (1)
Presbyterian (4)
Catholic (15)
Other (9)
All (44)
0.8
0.6
0.4
0.2
0.0
-0.2
-0.4
-0.6
-0.8
Ideological positions
Party
<0.001 (***)
Gender
Int.: 0.338 ()
Region
Int.: <0.001 (***)
Generation
Int.: 0.018 (*)
Experience
Int.: 0.984 ()
Religion
Int.: 0.019 (*)
Go back

Immigration

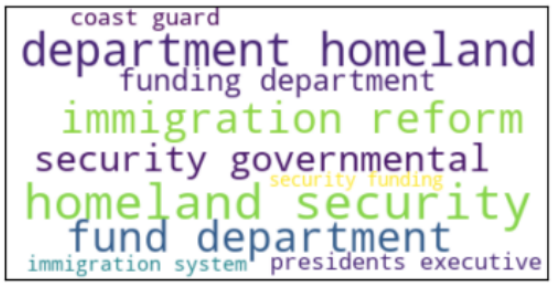


Immigration

E log(beta) - eta



E log(beta)



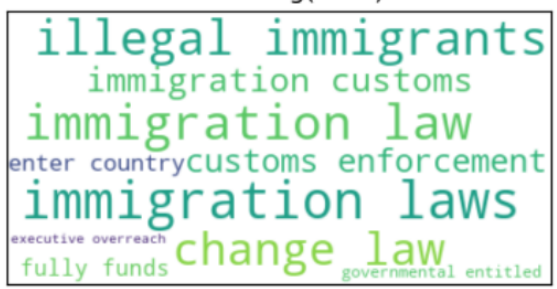
E log(beta) + eta

E -eta under log(beta) > -1.0

Topic 16

Go back

E +eta under log(beta) > -1.0



Latent Dirichlet Allocation (LDA)

- **Description**: A probabilistic model representing documents as mixtures of topics and topics as mixtures of words.
- **Key Paper**: Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). *Latent Dirichlet Allocation*. Journal of Machine Learning Research, 3(1), 993–1022.
- **R Package**: `topicmodels`
- **Use Case**: General-purpose topic modeling.

Correlated Topic Model (CTM)

- **Description**: Extends LDA by modeling correlations between topics using a logistic normal distribution.
- **Key Paper**: Blei, D. M., & Lafferty, J. D. (2007). *A correlated topic model of Science*. The Annals of Applied Statistics, 1(1), 17–35.
- **R Package**: `topicmodels`
- **Use Case**: Exploring related or dependent topics.

Structural Topic Model (STM)

- **Description**: Incorporates metadata (e.g., author, time) into the topic modeling process, enabling covariate-informed topic estimation.
- **Key Paper**: Roberts, M. E., Stewart, B. M., Airoldi, E. M. (2016). *A model of text for experimentation in the social sciences*. Journal of the American Statistical Association, 111.
- **R Package**: `stm`
  - Functions: `stm()`, `plot.STM()`
- **Use Case**: Analysis of topic variation with external metadata.

Embedded Topic Model (ETM)

- **Description**: Combines word embeddings with topic modeling. NO bag-of Words! Word Embeddings: Represent individual words as vectors (e.g., Word2Vec,). These focus on the context of a single word.
- **Key Paper**: Dieng, A. B., Ruiz, F. J. R., & Blei, D. M. (2020). *Topic modeling in embedding spaces*. Transactions of the Association for Computational Linguistics, 8, 439–453.
- **R Package**: No direct R package available; implementation in Python (`https://github.com/adjidieng/ETM`).
- **Use Case**: When semantic similarity of words is critical.

BERTopic

- **Description**: Uses transformer-based embeddings (e.g., BERT) combined with clustering for topic extraction.
- **Key Paper**: Grootendorst, M. (2020). *BERTopic: Neural topic modeling with a class-based TF-IDF procedure*. [ArXiv preprint](https://arxiv.org/abs/2203.05794).
- **R Package**: Not natively available in `R`; implemented in Python. Can be accessed via `reticulate` or `R` package `text`
- **Use Case**: Contextualized topic modeling for modern NLP tasks.

Text Based Ideal Points

- **Description**: Combines Topic Models with Scaling Models to estimate ideological positions.
- **Key Paper**: Text-Based Ideal Points, K Vafa, S Naidu, D.Blei (`https://arxiv.org/abs/2203.05794`).
  Hofmarcher et al.(2023) Gaining Insights on U.S. Senate Speeches Using a Time Varying Text Based Ideal Point Model (`https://arxiv.org/abs/2206.10877`)
- **R Package**: Not natively available in `R`; implemented in Python.`https://github.com/keyonvafa/tbip`

Structural Text Based Scaling

- **Description**: Combines Topic Models with Scaling Models to estimate ideological positions. Also allows for Covariates on ideological positions and topic specific idela points.
- **Key Paper**: A Structural Text-Based Scaling Model for Analyzing Political Discourse `https://arxiv.org/abs/2410.11897`
- **Software**: Not natively available in `R`; implemented in Python.`https://github.com/vavrajan/STBS`