Inverting Stochastic Models of Language To Detect Structure

Thiemo Fetzer

University of Warwick University of Bonn CEPR LSE AEAI

April 24, 2025

Uncovering structure in unstructured data

- We have looked at generative models of language in particular a classic n-gram language model
- The task here is language or word prediction to generate language
- In many instances problems involving natural language or any other unstructured data involves *dimensionality reduction*
- This is vital to understand concepts such as *attention*
- Often times few *relevant* keywords together are indicative of a concept
- This provides a brief introduction into topic modeling which can be interpreted as a technique of constructing *embeddings*

Plan

Topic modelling

Topic models as embeddings ?

Topic Modelling

- Topic models are based upon the idea that documents are *mixtures of topics*, where a topic is a *probability distribution over words*.
- A topic model is a *generative model* for documents: it specifies a simple probabilistic procedure by which documents can be generated.
- How to generate a new document?
  - To make a new document, one chooses a distribution over topics.
  - Then, for each word in the document that you want to generate, one chooses a topic at random according to this distribution, and draws a word from that topic.

$\Rightarrow$ topic modelling, inverts this process, inferring the set of topics that were responsible for generating a collection of documents.

An Example from Research: FOMC communications

We have noticed a change in the relationship between the core CPI and the chained core CPI, which suggested to us that maybe something is going on relating to substitution bias at the upper level of the index. You focused on the nonmarket component of the PCE, and I wondered if something unusual might be happening with the core CPI relative to other measures.

Figure: FOMC Committee Minutes

An Example from Research: FOMC communications

noticed change relationship between core CPI
chained core CPI suggested maybe something going
relating substitution bias upper level index focused
nonmarket component PCE wondered something
unusual happening core CPI relative measures

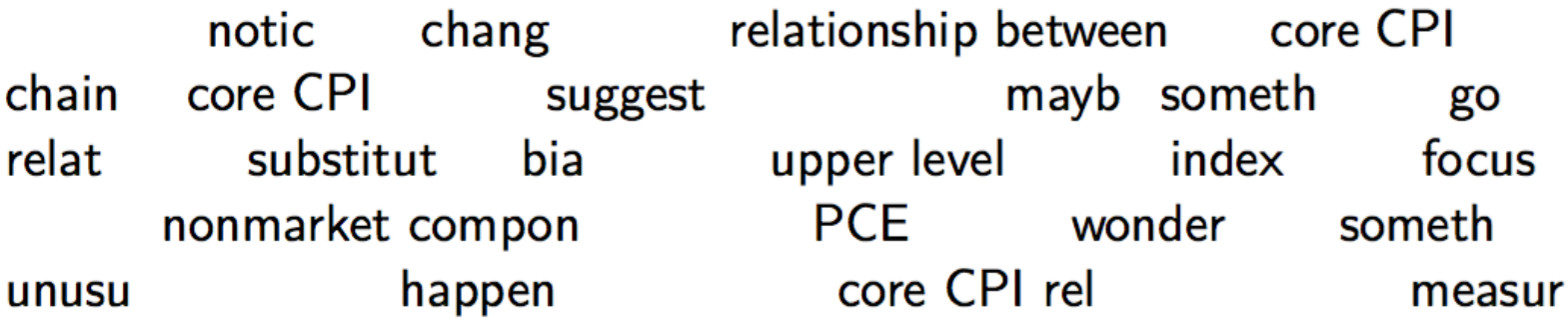
Figure: FOMC Committee Minutes

An Example from Research: FOMC communications

notic chang relationship between core CPI
chain core CPI suggest mayb someth go
relat substitut bia upper level index focus
nonmarket compon PCE wonder someth
unusu happen core CPI rel measur

Figure: FOMC Committee Minutes

An Example from Research: FOMC communications



Federal Funds Rate $\rightarrow$ fed fund rate $\rightarrow$ ffr

monetary policy $\rightarrow$ monetari polici $\rightarrow$ monpol

Figure: FOMC Committee Minutes

An Example from Research: FOMC communications

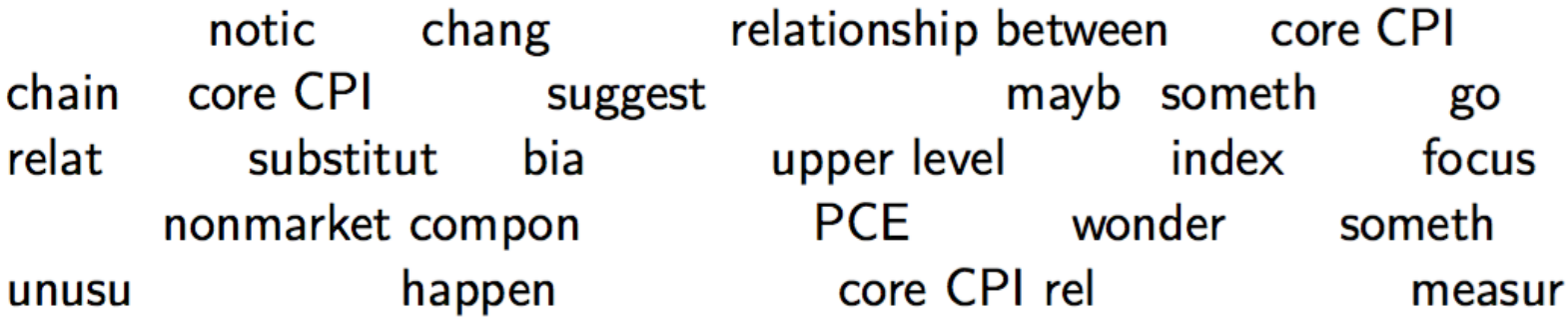


Figure: FOMC Committee Minutes

An Example from Research: FOMC communications

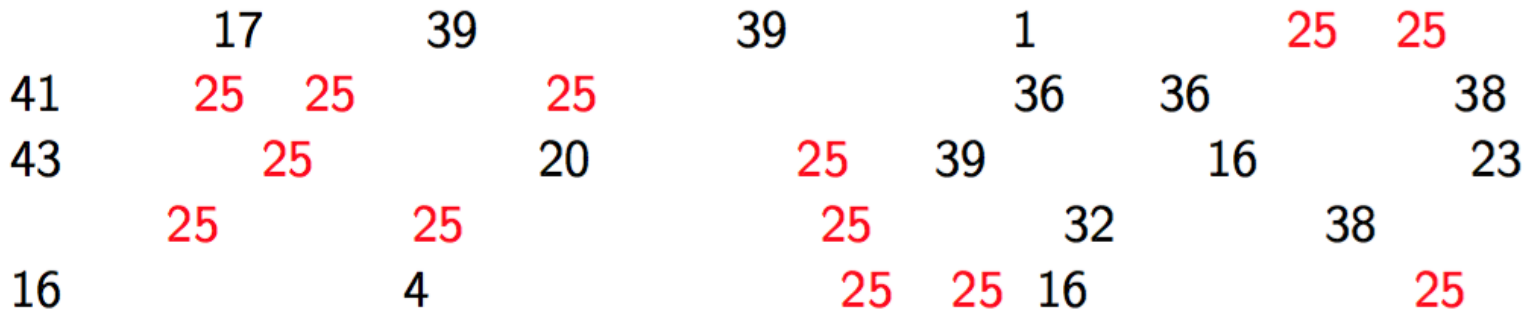


Figure: FOMC Committee Minutes

An Example from Research: FOMC communications

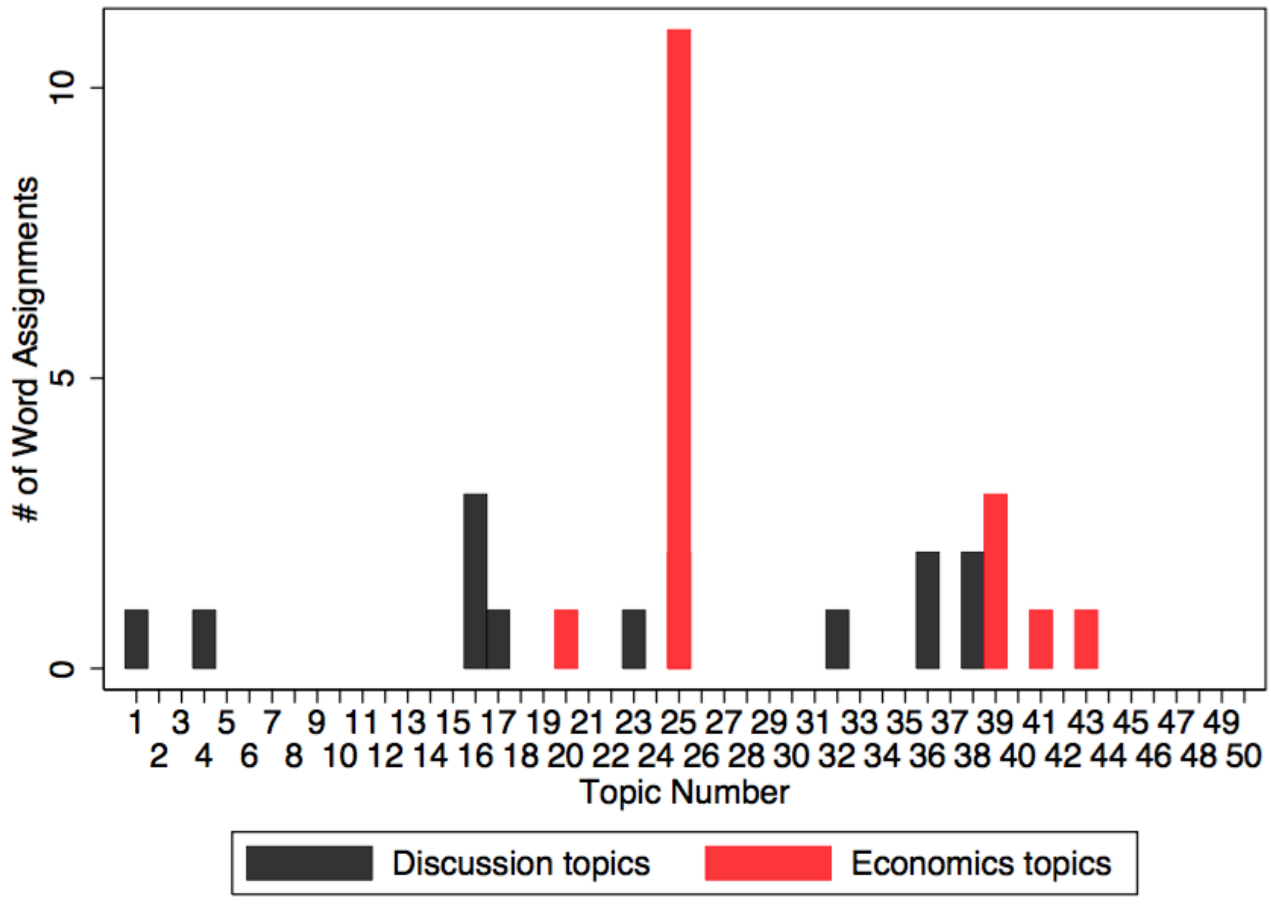


Figure: FOMC Committee Minutes

An Example from Research: FOMC communications

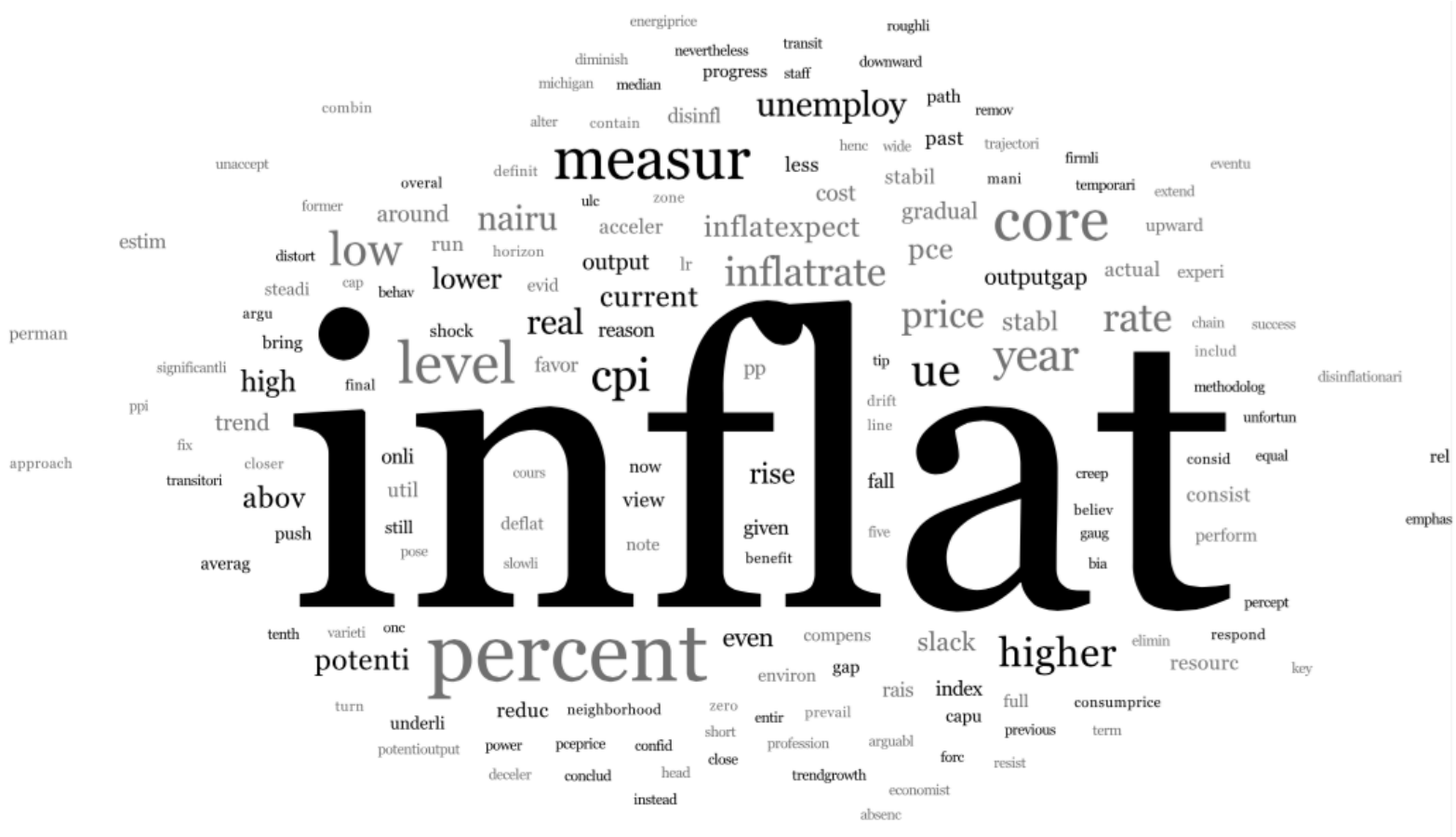


Figure: FOMC Committee Minutes

An Example from Research: FOMC communications

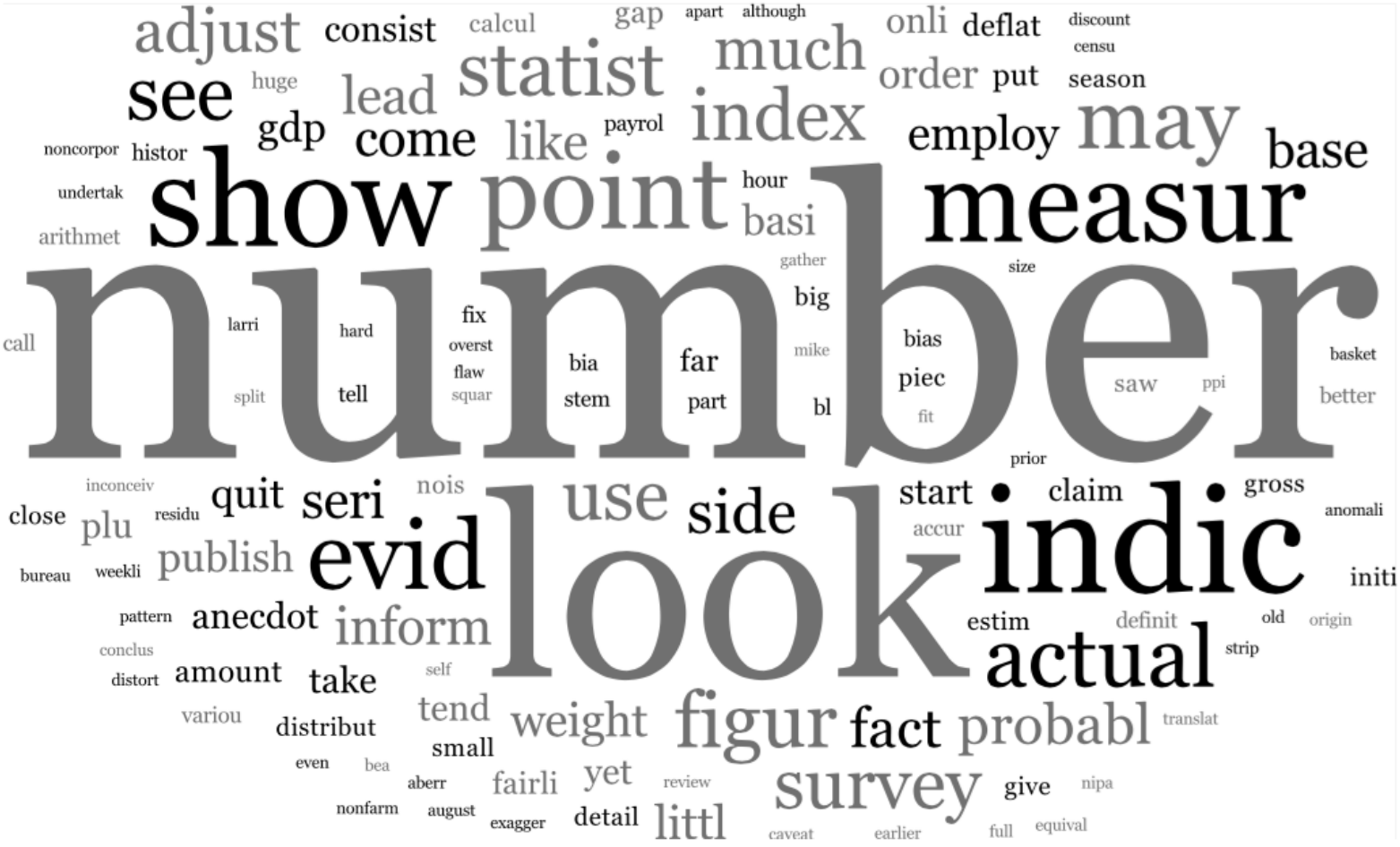
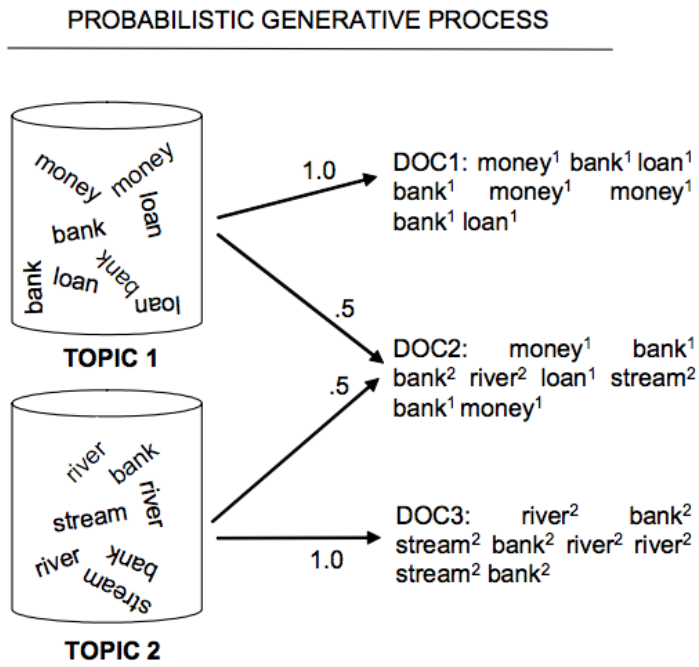


Figure: FOMC Committee Minutes

A Generative Model

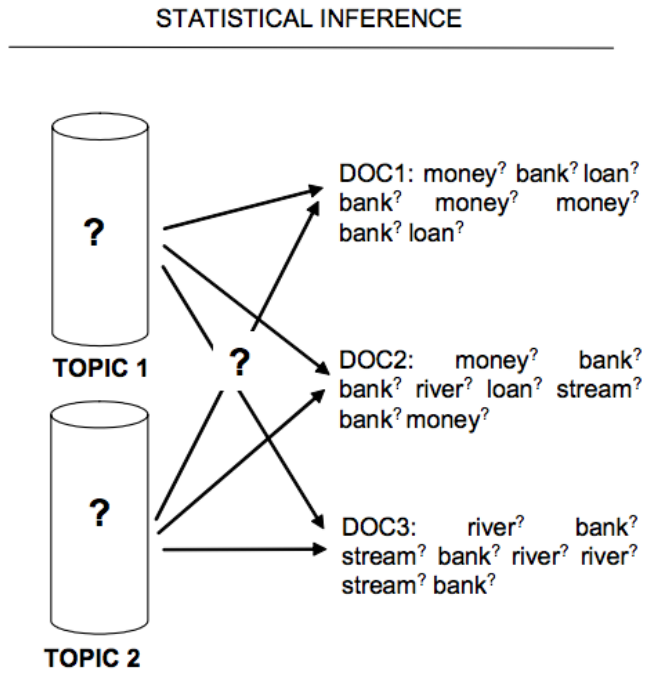
- Generative models for documents are based on simple probabilistic sampling rules that describe how words in documents might be generated on the basis of latent unobservable (random) variables
- When fitting a generative model, the goal is to find the best set of latent variables that can explain the observed data (i.e., observed words in documents), assuming that the model actually generated the data

A Generative Model



Documents 1 and 3 were generated by sampling only from topic 1 and 2 respectively while document 2 was generated by an equal mixture of the two topics.

Statistical Inference about a Generative Model



Given the observed words in a set of documents, we would like to know what topic model is most likely to have generated the data. This involves *inferring* the probability distribution over words associated with each topic, the distribution over topics for each document, and, often, the topic responsible for generating each word.

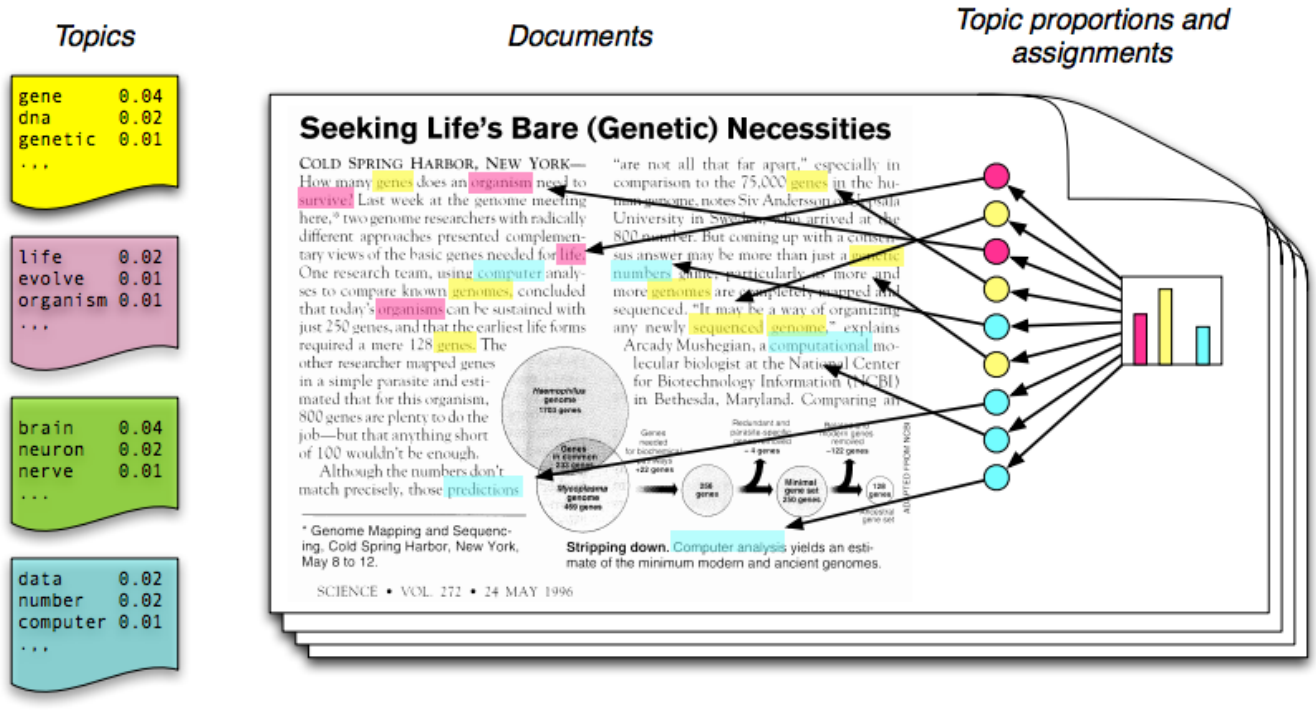
Key Assumptions of LDA Topic Model

- Elements of LDA generative process: documents D, corpus C, words w and topics Z.
- Documents exhibit multiple topics (but typically not many)
- We assume that documents are constructed out of some finite set of available topics Z.
- LDA is a probabilistic model with a corresponding generative process

  $\Rightarrow$ each document is assumed to be generated by this process
- A topic is a distribution over a fixed vocabulary

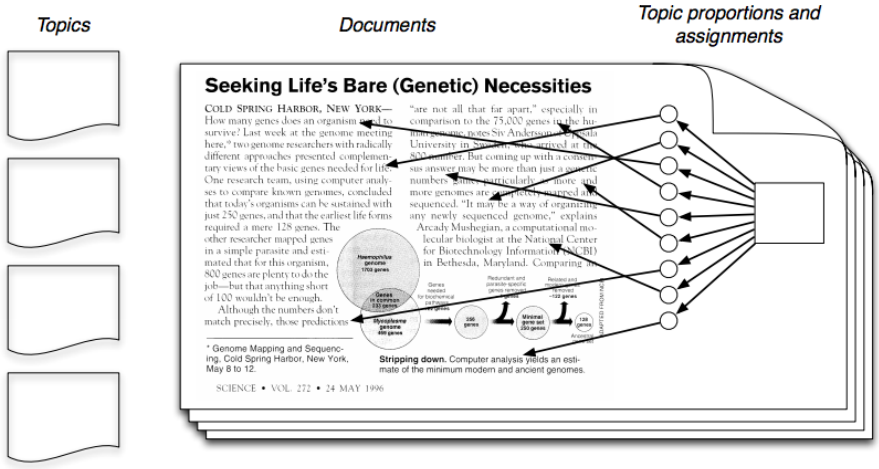
  these topics are assumed to be generated first, before the documents
- Only the number of topics is specified in advance

Generative model



- Each **topic** is a distribution over words
- Each **document** is a mixture of corpus-wide topics
- Each **word** is drawn from one of those topics specific word distributions.

Generative model: Unobservable parameters



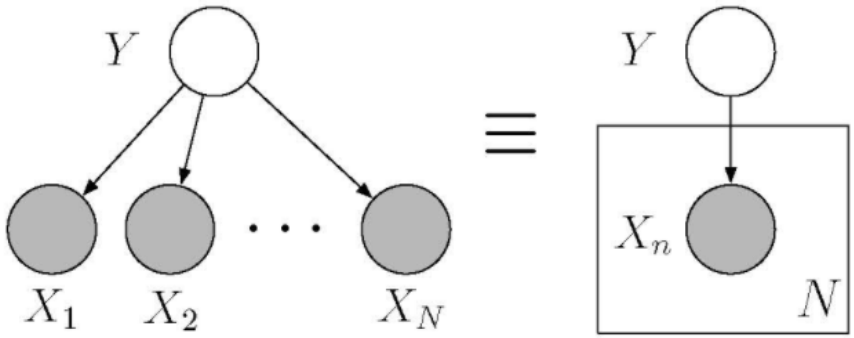
- We dont observe the parameters of the model: we only observe documents and associated word count vectors, the remaining structure are hidden variables
- Conditional on the observed documents/ word counts, we want to infer the parameters of the multinomials
- e.g. the multinomial parameter vector of p's that give us the probability of observing a word conditional on a topic k or the parameter vector that gives us the distribution of topics in documents.

How do we model the text generation process?

We want to generate a document d.

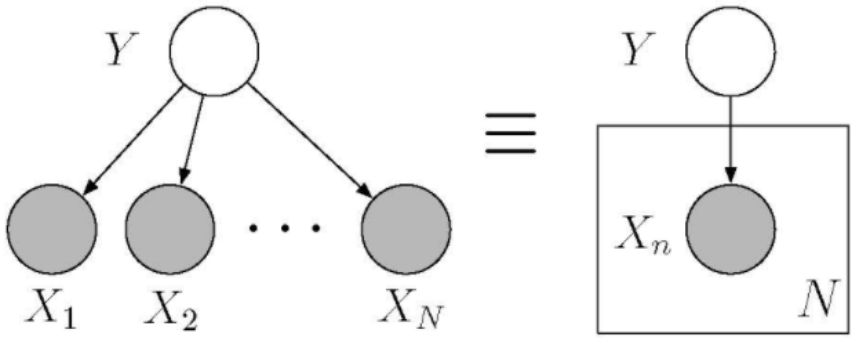
1. Randomly choose a distribution over topics
2. For each word in the document
   - 2.1 randomly choose a topic from the distribution over topics
   - 2.2 randomly choose a word from the corresponding topic specific word distribution
3. So we need a distribution over a distribution (for step 1)
4. Note that words are generated independently of other words (unigram bag-of-words model)

Plate Notation



- Nodes are random variables
- Edges denote possible dependency
- Observed variables are shaded, so here Y is unobserved.
- Plates denote replicated structure
- So here there are N observed X_n's, that are possibly all dependent on Y (but not on each other)

Plate Notation (2)

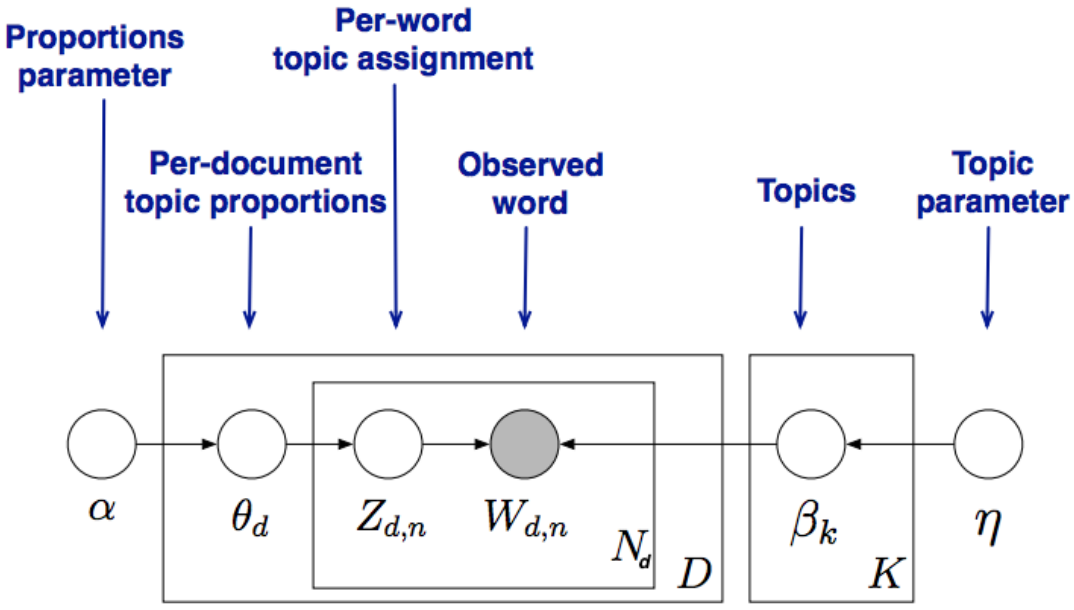


- Structure of graph gives pattern of conditional dependence
- This figure would represent

$$p(x_1, ..., x_n, y) = p(y) \prod_{n=1}^{N} p(x_n|y)$$

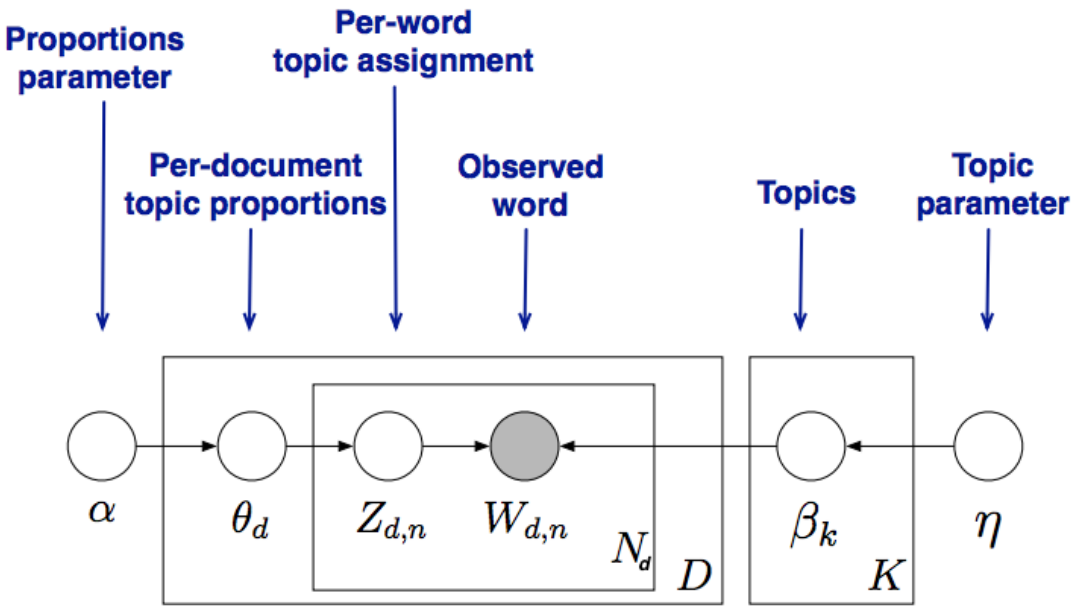
- The X_i are conditionally independent from one another.

A generative model



- Each β_k is a distribution over our vocabulary, there are K of them.
- There is one K dimensional θ_d for every of D documents.
- For each word in N_d plate, we have a variable $Z_{d,n}$ which gives the topic assignment for the n-th word drawn from θ_d distribution.

A generative model - joint distribution



Joint distribution of observed and hidden random variables:

$$p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D}|\alpha, \eta) = \prod_{k=1}^{K} p(\beta_k|\eta) \prod_{d=1}^{D} p(\theta_d|\alpha) \prod_{n=1}^{N} p(z_{d,n}|\theta_d) p(w_{d,n}|\beta_{1:K}, z_{d,n})$$

Lookup tables

What are $p(w_{d,n}|z_{d,n}, \beta_{1:K})$ and $p(z_{d,n}|\theta_d)$?

		K				
		β_1	β_2	β_3	.	β_K
	w_1	β_{11}	.	.	.	β_{1K}
	w_2	β_{21}	.	.	.	.
V	.	.	.	.	.	.
	.	.	.	.	.	.
	.	.	.	.	.	.
	w_V	.	.	.	.	β_{VK}

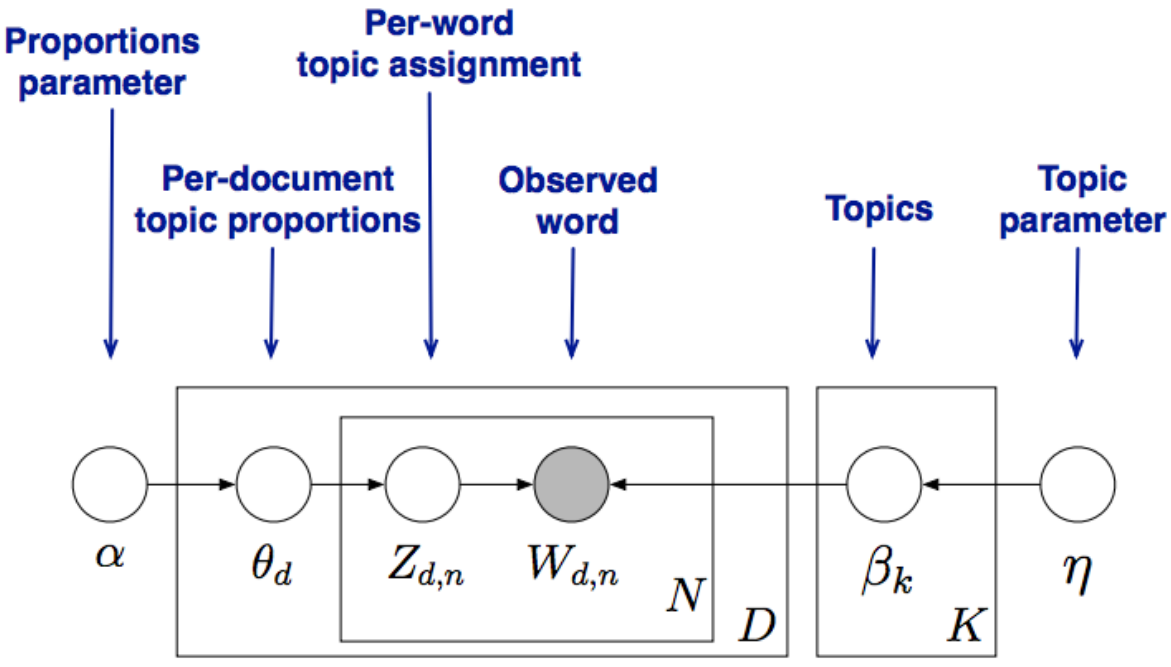
We can think of these being defined in some lookup tables (matrices).

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

- look up probability of the word conditional on assigned topic $Z_{d,n}$ in respective correct β vector (i.e. there is V x K) matrix.

Similarly, $p(z_{d,n}|\theta_d) = \theta_{d,z_{d,n}}$ looks up $z_{d,n}$-th entry in θ_d vector, i.e. there is a D x K matrix.

LDA in plate notation



This joint defines a posterior distribution of the parameter space. From a collection of documents, we then infer

- Per-word topic assignment $z_{d,n}$
- Per-document topic proportions θ_d
- Per-corpus topic distributions β_k

We can use these posteriors to assign new virgin data to topics.

Revisiting the multinomial distribution

We introduced the multinoial distribution to represent a document x as bag of words counts:

$$p(x; p_1, ..., p_k) = \frac{n!}{x_1! \cdots x_k!} p_1^{x_1} \cdots p_k^{x_k}$$

Each time word j appears in the document it contributes an amount p_j to the total probability, hence the term $p_j^{x_j}$.
The components of $\mathbf{p}$ are non-negative and have unit sum: $\sum_j p_j = 1$.
Each vector x of word counts can be generated by a whole range of different sequences of words: the number of possible combinations to produce a given vector of word counts is the first factor - called the *multinomial coefficient*.
The second factor is the probability of any individual sequence that yields identical word count vector x.

Dirichlet Distribution

The Dirichlet distribution is an exponential family distribution over the simplex of positive numbers that sum to one.

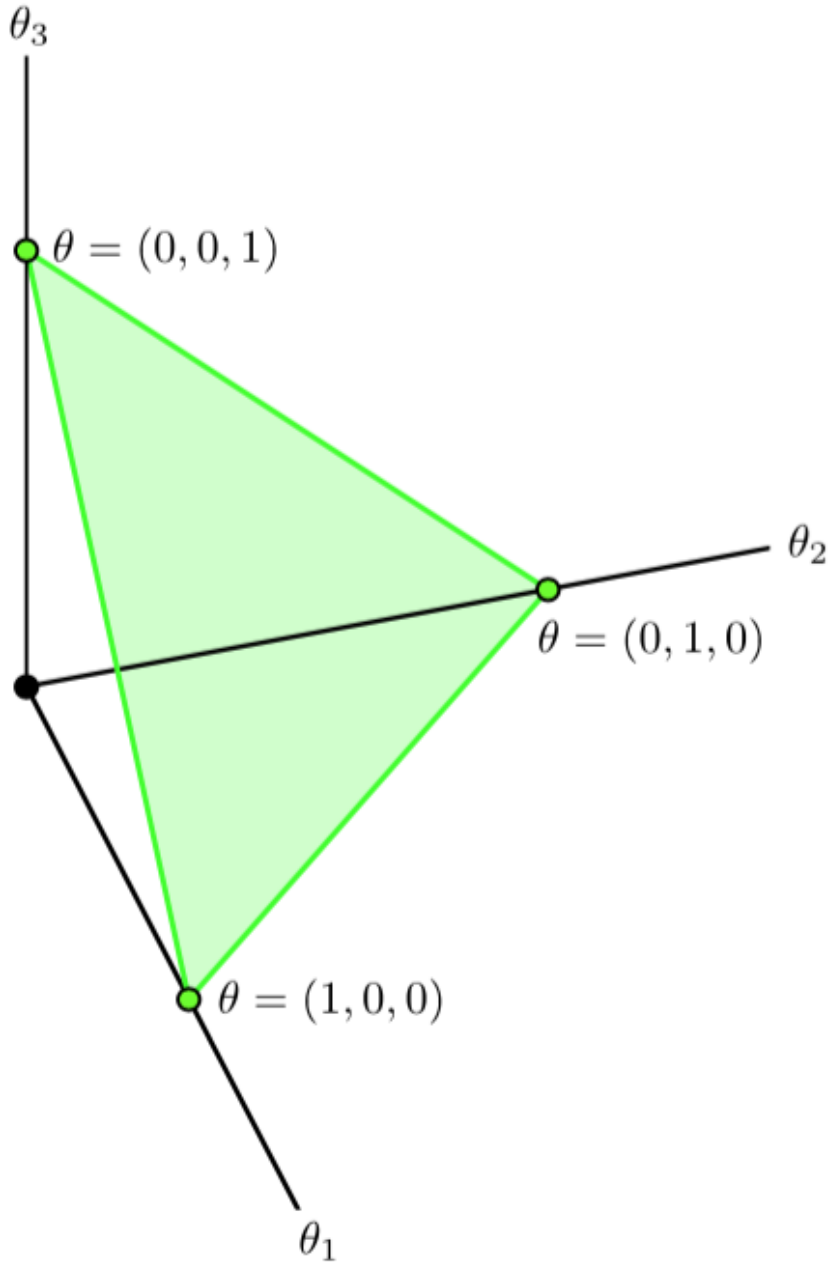
What does that mean? A Dirichlet distribution is a probability density

function over the set of all multinomial parameter vectors.
$\Rightarrow$ a Dirichlet distribution is a distribution over distributions.

$$p(\mathbf{p}|\alpha) = \frac{\Gamma(\sum_{i=1}^{k} \alpha_i)}{\prod_{i=1}^{k} \Gamma(\alpha_i)} \prod_{i=1}^{k} p_i^{\alpha_i - 1}$$

where $\sum p_i = 1$ and $p_i \geq 0$.

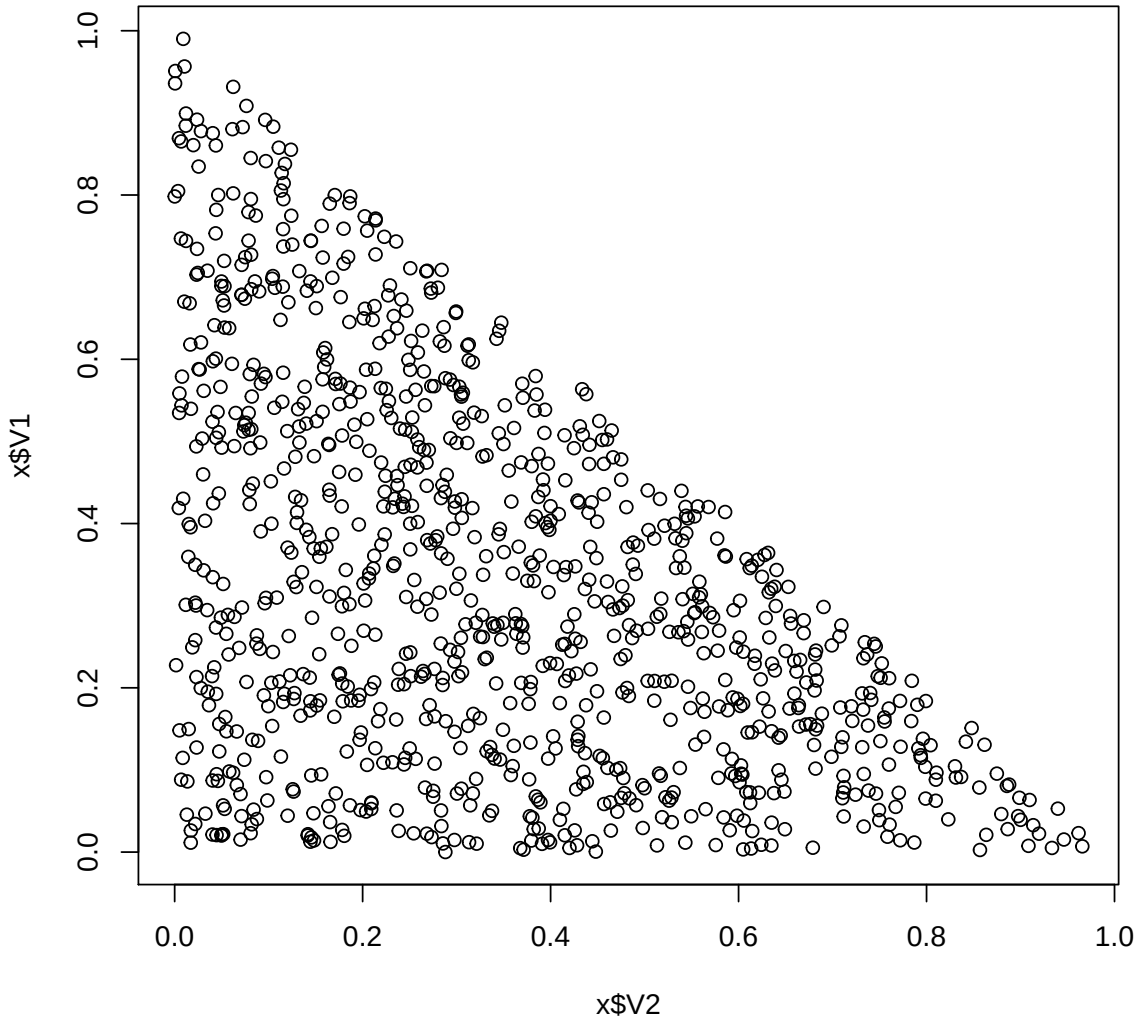
The case of $k = 3$



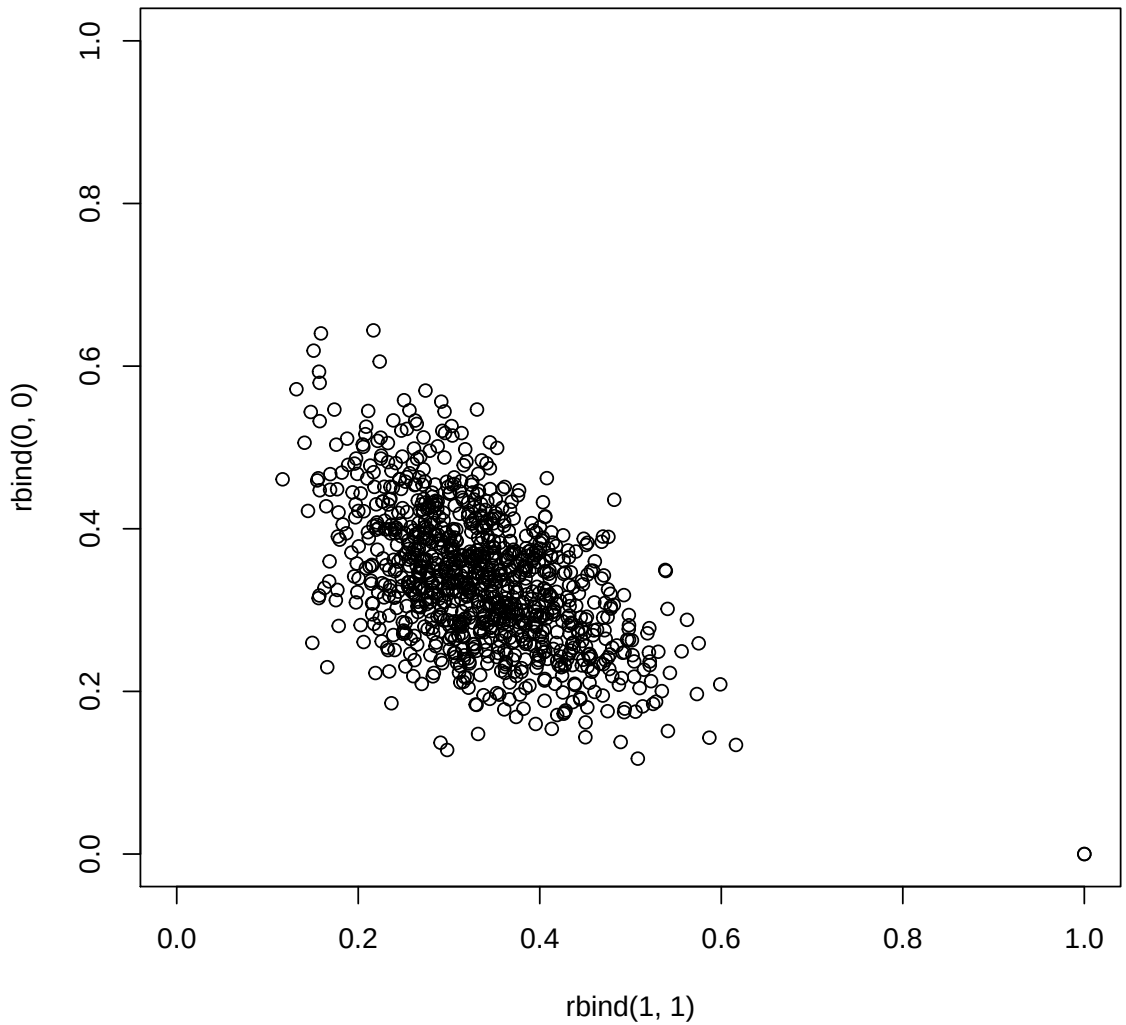
An example dirichlet distribution with $k = 3$ and $\alpha_j = 1$

```
require(MCMCpack)

alpha <- 1

draws <- 1000

dimen <- 3
x <- rdirichlet(draws, rep(alpha, dimen))

x<-data.table(x)

head(x)

##         V1    V2      V3
## 1: 0.0244 0.697 0.27828
## 2: 0.7365 0.106 0.15794
## 3: 0.1632 0.507 0.32940
## 4: 0.4229 0.520 0.05679
## 5: 0.5515 0.259 0.18942
## 6: 0.5884 0.402 0.00953
```

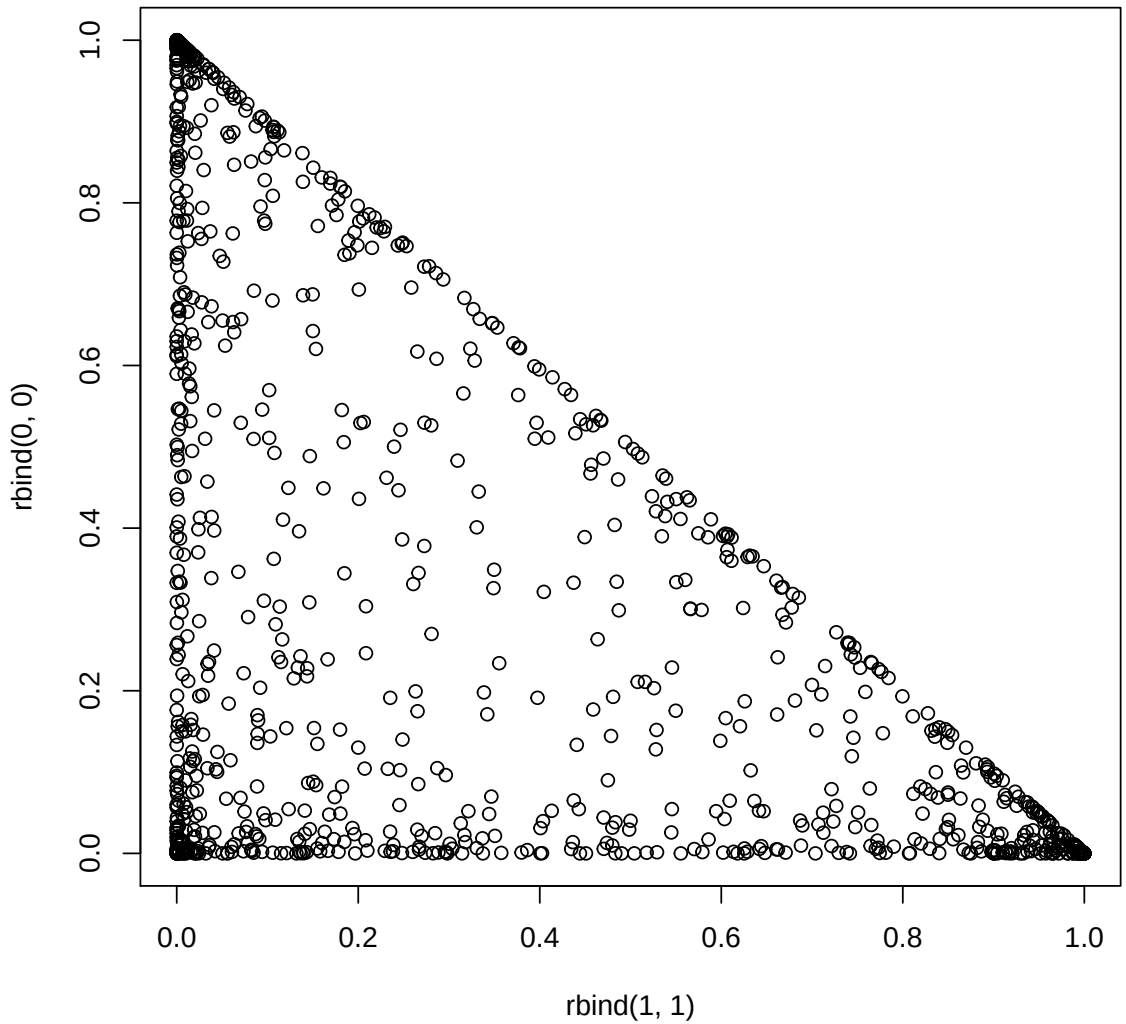
An example dirichlet distribution with $k = 3$ and $\alpha_j = 1$



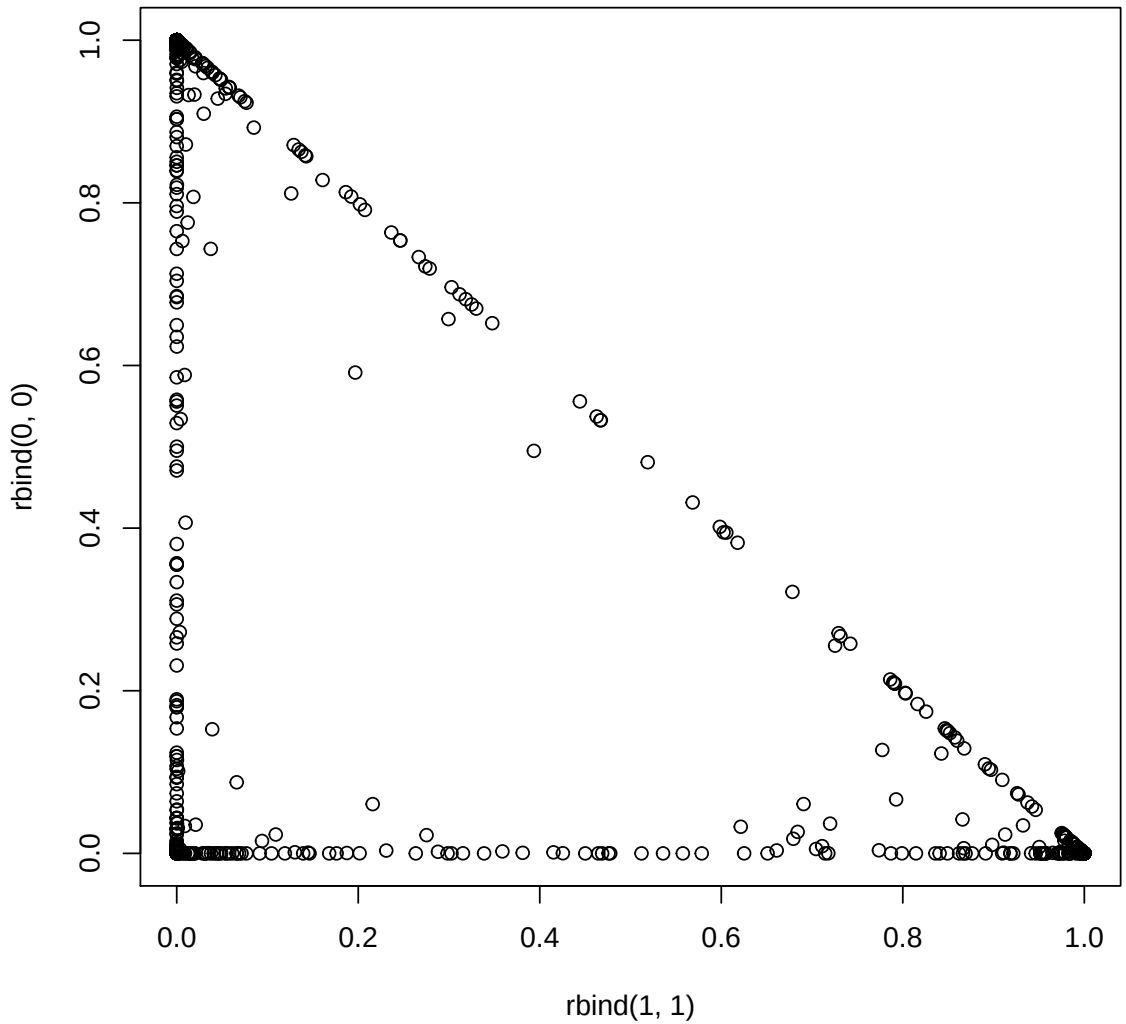
An example dirichlet distribution with $k = 3$ and $\alpha_j = 10$



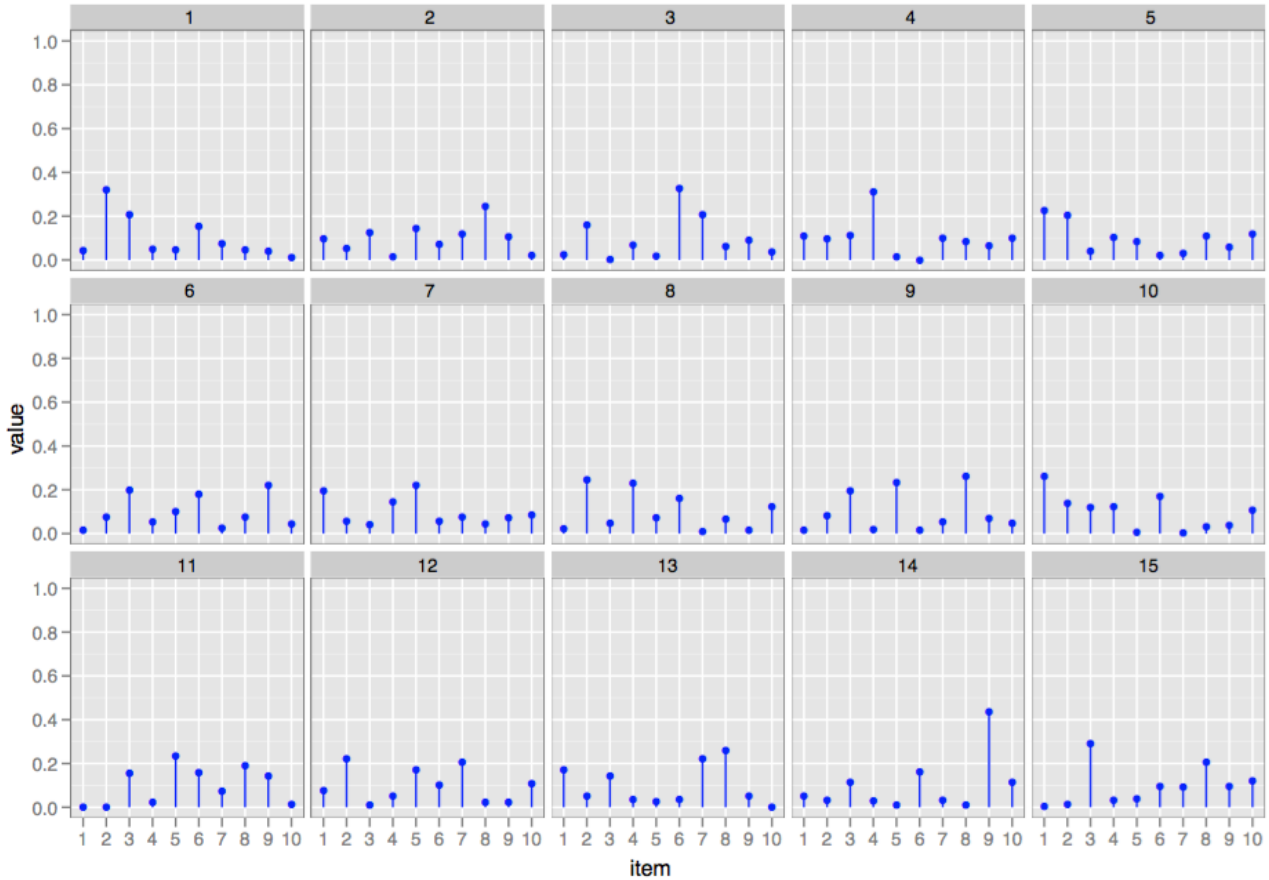
An example of a sparse dirichlet distribution with $k = 3$ and $\alpha_j = 0.25$



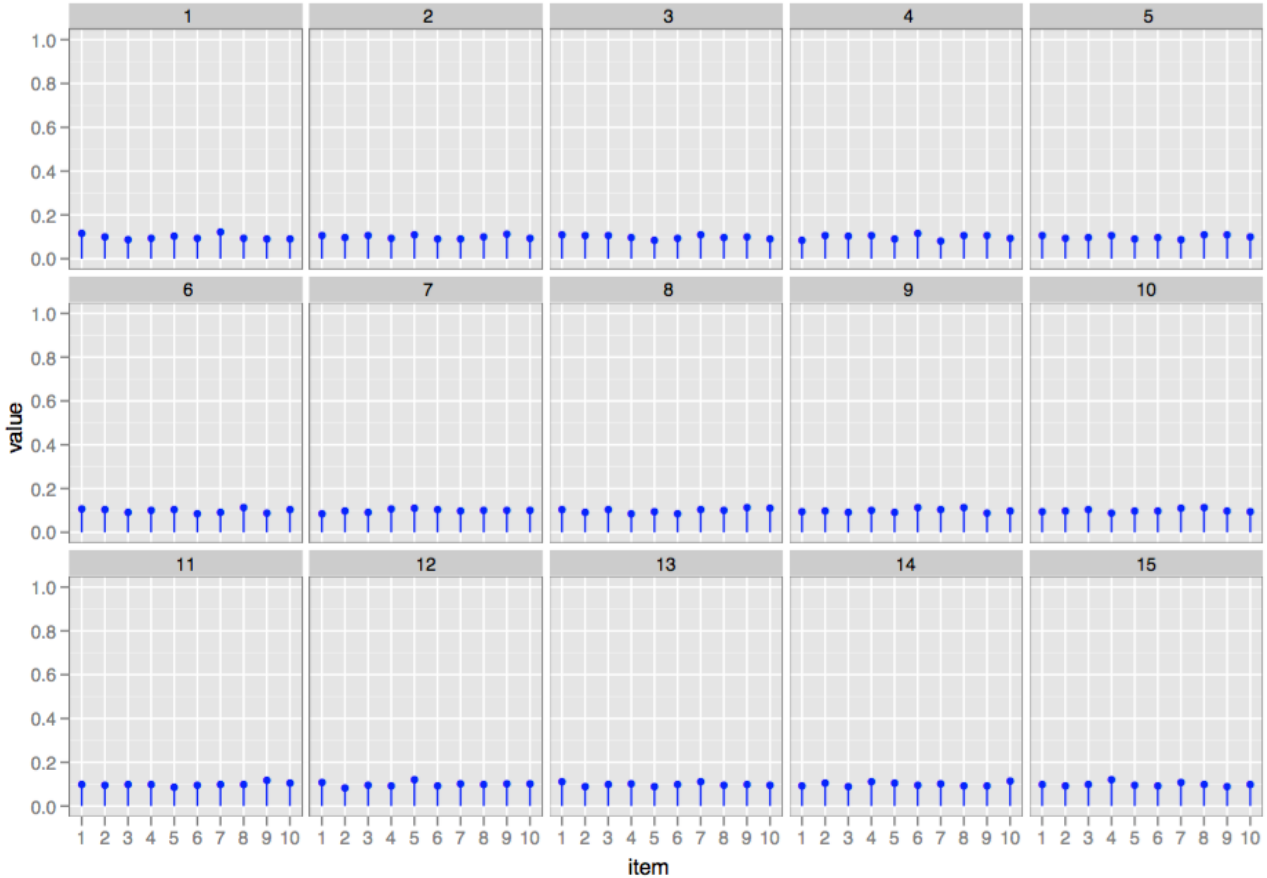
An example of a sparse dirichlet distribution with $k = 3$ and $\alpha_j = 0.05$



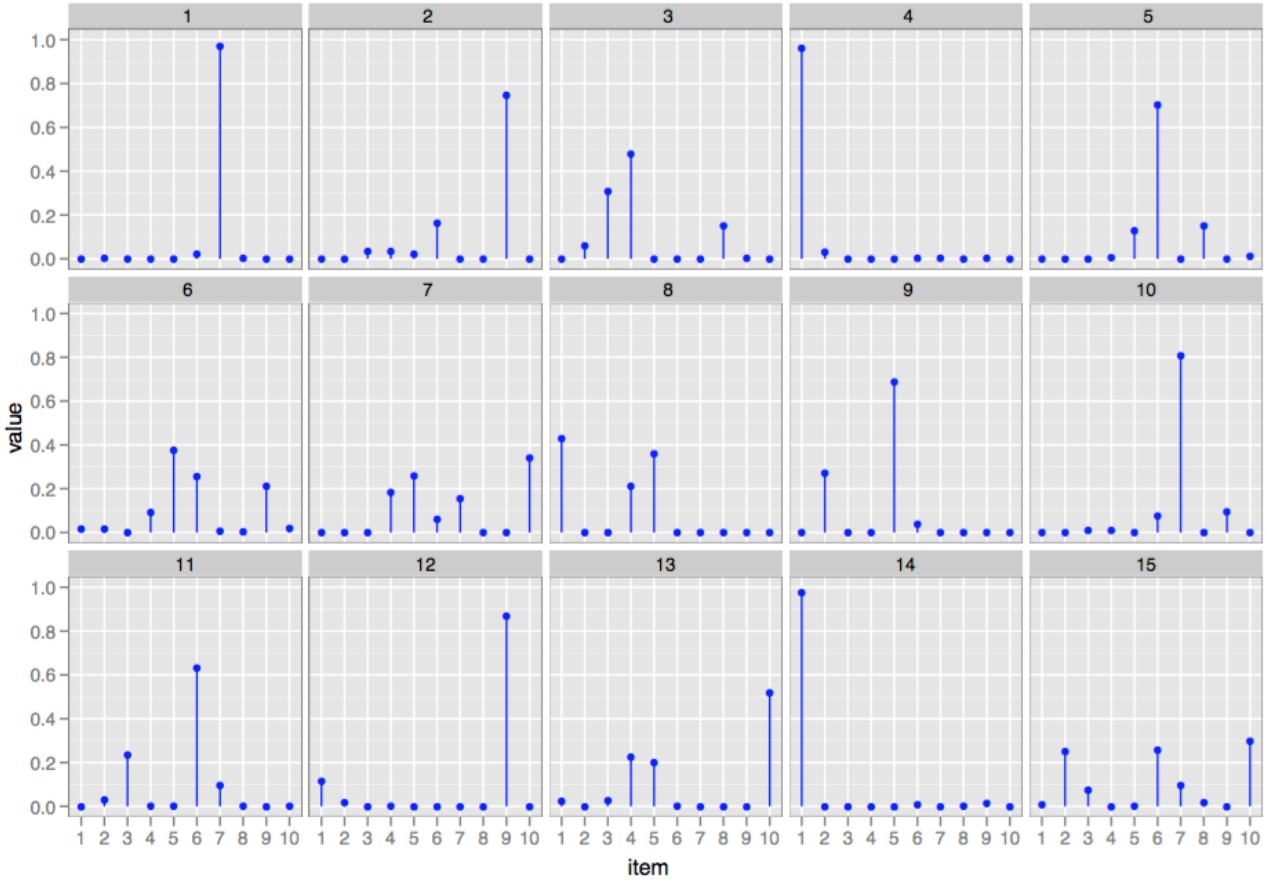
15 dirichlet with $\alpha = 1$, $k = 10$



15 dirichlet with $\alpha = 100$, $k = 10$

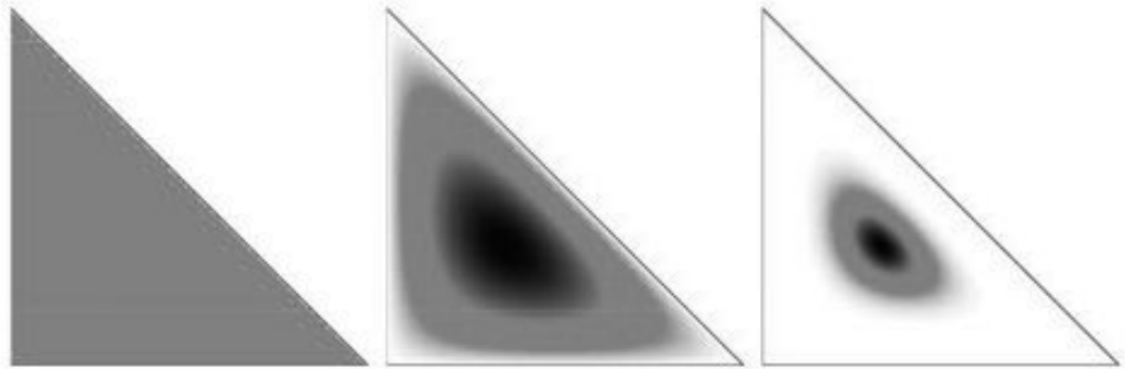


15 dirichlet with $\alpha = 0.01$, $k = 10$

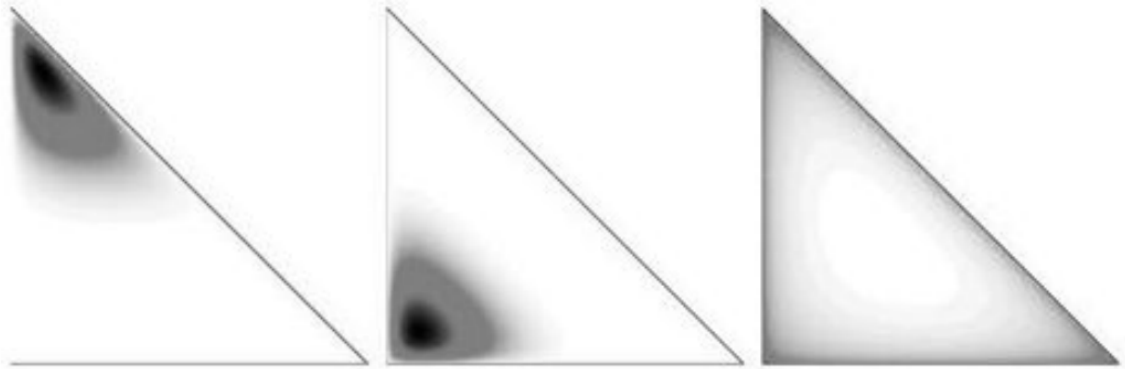


An alternative parameterization

$$P(\boldsymbol{p} \mid \alpha \boldsymbol{m}) = \frac{\Gamma(\sum_k \alpha m_k)}{\prod_k \Gamma(\alpha m_k)} \prod_k p_k^{\alpha m_k - 1}$$



$\alpha = 3, \boldsymbol{m} = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ $\alpha = 6, \boldsymbol{m} = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ $\alpha = 30, \boldsymbol{m} = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$



$\alpha = 14, \boldsymbol{m} = (\frac{1}{7}, \frac{5}{7}, \frac{1}{7})$ $\alpha = 14, \boldsymbol{m} = (\frac{1}{7}, \frac{1}{7}, \frac{5}{7})$ $\alpha = 2.7, \boldsymbol{m} = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$

Why the shape? $\alpha m = (1, 1, 1)$

With $\alpha = (1, 1, 1)$...note $\Gamma(k) = (k-1)!$ for integer k.

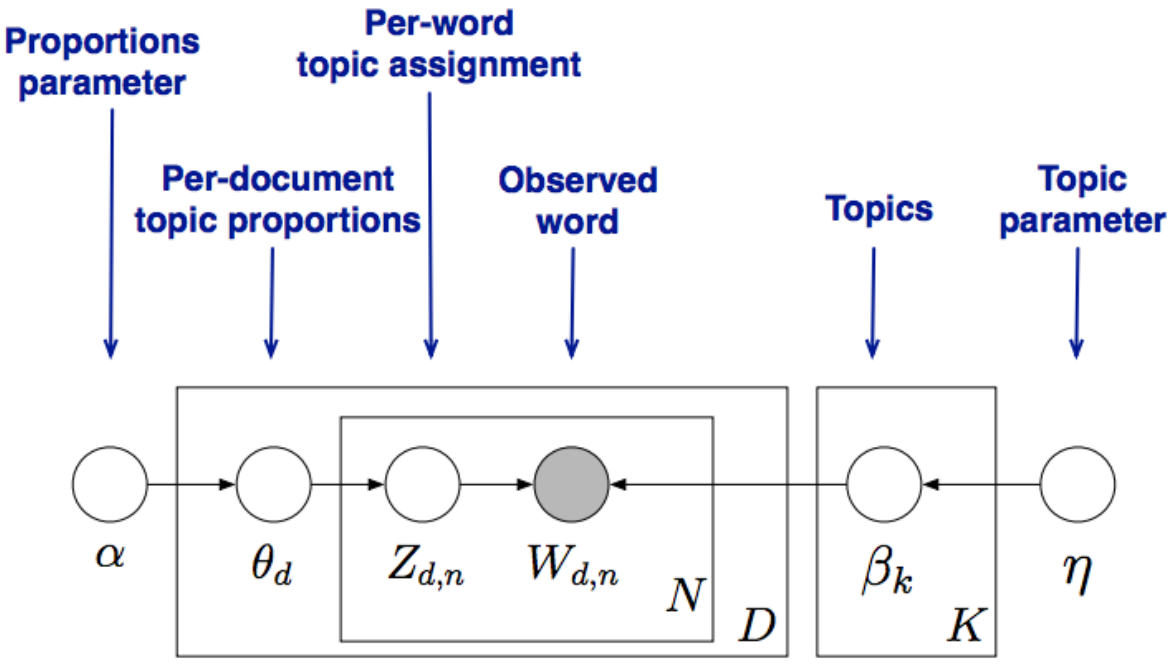
$$p(\mathbf{p}|\alpha) = \frac{\Gamma(3)}{1} \prod_{i=1}^{k} = \textit{constant}$$

This is the PDF of a uniform distribution.

Dirichlet distribution has some nice properties

- as scaling parameter gets smaller and smaller, the distributions get more sparse.
- sparsity is a feature of text data
- as α gets larger, the mass concentrates.
- If $\theta \sim Dir(\alpha)$ and $Z_n \sim Mult(\theta)$, the $p(\theta|Z_{1:N}) \sim Dir(\alpha + n(Z_{1:N}))$ - this is known as conjugacy.
- the more data we see, the peakier our dirichlet.

LDA in plate notation



It is impossible to "fill" the missing values (i.e. the latent variables), but it is possible to use the information contained in the word counts and the structure of our model to perform *posterior inference* to approximate the posterior distribution.

Gibbs Sampling Intuition

Initialize: Go through each document, and randomly assign each word in the document to one of the K topics. This assignment already gives you both topic representations of all the documents – that is $p(\text{topic } t|\text{document } d)$ – and word distributions across all topics $p(\text{word } w|\text{topic } t)$.

1. For each document d
2. For each word w in d
3. For each topic t, compute
   - 3.1 $p(\text{topic } t|\text{document } d)$ = the proportion of words in document d that are currently assigned to topic t,
   - 3.2 $p(\text{word } w|\text{topic } t)$ = the proportion of words w among all words that are currently assigned to topic t.
   - 3.3 Reassign w to a *different topic*. We choose topic t with probability $p(\text{topic } t|\text{document } d) * p(\text{word } w|\text{topic } t)$ - this is the probability that topic t generated word w.

Gibbs Sampling Intuition

- In other words, in the resampling step, we're assuming that all word to topic assignments except for the current word w are correct. We update the assignment of the word w using our model of how documents are generated.
- Repeat these steps a large number of times, the process will eventually converge to a local optimum where topic assignments dont change anymore.

  $\Rightarrow$ Use these assignments to estimate the topic mixtures of each document (by counting the proportion of words assigned to each topic within that document) and the words associated to each topic (by counting the proportion of words assigned to each topic overall).

Why does LDA work?

LDA trades off two goals.

1. For each document, allocate its words to as few topics as possible.
2. For each topic, assign high probability to as few terms as possible.

These goals are at odds with each other

- Putting a document in a single topic makes (2) hard:

  All of its words must have probability under that topic.
- Putting very few words in each topic makes (1) hard:

  To cover a document's words, it must assign many topics to it.

  $\Rightarrow$ Trading off these goals finds groups of tightly co-occurring words.

Topic model implementation in R

```
library(topicmodels)

data("AssociatedPress")

ap_lda <- LDA(AssociatedPress, k = 3, control = list(seed = 24022017))

class(ap_lda)

## [1] "LDA_VEM"
## attr(,"package")
## [1] "topicmodels"

slotNames(ap_lda)

##  [1] "alpha"           "call"            "Dim"             "control"
##  [5] "k"               "terms"           "documents"       "beta"
##  [9] "gamma"           "wordassignments" "loglikelihood"   "iter"
## [13] "logLiks"         "n"
```

Easily extracting the vector β

```
library(tidytext)

ap_lda_td <- data.table(tidy(ap_lda))

nrow(ap_lda_td[topic==1])

## [1] 10473

ap_lda_td[topic==1][order(beta, decreasing=TRUE)][1:20]

##     topic     term    beta
##  1:     1  percent 0.01563
##  2:     1  million 0.01052
##  3:     1     year 0.00851
##  4:     1      new 0.00787
##  5:     1  billion 0.00735
##  6:     1   market 0.00544
##  7:     1  company 0.00498
##  8:     1     last 0.00476
##  9:     1   prices 0.00429
## 10:     1    stock 0.00409
## 11:     1  federal 0.00386
## 12:     1     york 0.00320
## 13:     1 business 0.00303
## 14:     1    first 0.00301
## 15:     1      oil 0.00300
## 16:     1    sales 0.00299
## 17:     1   dollar 0.00286
## 18:     1   report 0.00280
## 19:     1     rose 0.00280
## 20:     1   higher 0.00276
```

Easily extracting the vector β

```
ap_lda_td[topic==2][order(beta, decreasing=TRUE)][1:20]
```

```
##     topic      term    beta
##  1:     2         i 0.00648
##  2:     2    police 0.00599
##  3:     2    people 0.00580
##  4:     2       two 0.00521
##  5:     2     years 0.00381
##  6:     2     court 0.00318
##  7:     2 officials 0.00304
##  8:     2     state 0.00303
##  9:     2      city 0.00297
## 10:     2       new 0.00296
## 11:     2     three 0.00276
## 12:     2      last 0.00258
## 13:     2      time 0.00237
## 14:     2       day 0.00221
## 15:     2    killed 0.00208
## 16:     2     found 0.00206
## 17:     2      case 0.00204
## 18:     2     first 0.00197
## 19:     2     miles 0.00195
## 20:     2      home 0.00194
```

Easily extracting the vector β

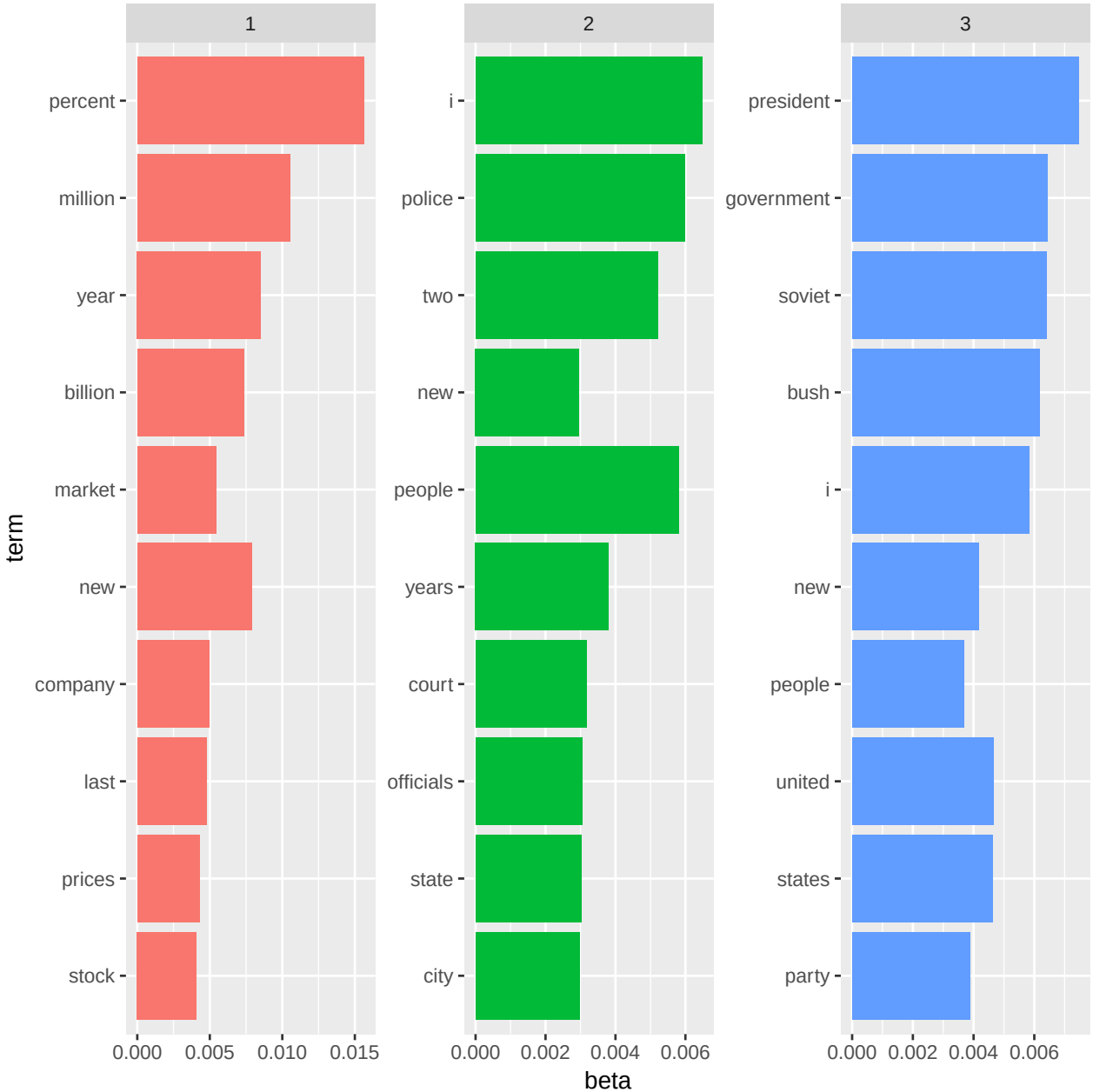
```
ap_lda_td[topic==3][order(beta, decreasing=TRUE)][1:20]
```

```
##     topic       term    beta
##  1:     3  president 0.00747
##  2:     3 government 0.00644
##  3:     3     soviet 0.00639
##  4:     3       bush 0.00617
##  5:     3          i 0.00583
##  6:     3     united 0.00466
##  7:     3     states 0.00463
##  8:     3        new 0.00416
##  9:     3      party 0.00389
## 10:     3     people 0.00367
## 11:     3  political 0.00350
## 12:     3      house 0.00327
## 13:     3      state 0.00309
## 14:     3        two 0.00308
## 15:     3   military 0.00306
## 16:     3       last 0.00301
## 17:     3    dukakis 0.00296
## 18:     3       told 0.00286
## 19:     3   campaign 0.00284
## 20:     3      first 0.00275
```

Plotting out topic terms β_k

```
library(ggplot2)
library(dplyr)
ap_top_terms <- ap_lda_td %>%
  group_by(topic) %>%
  top_n(10, beta) %>%
  ungroup() %>%
  arrange(topic, -beta)

ap_top_terms %>%
  mutate(term = reorder(term, beta)) %>%
  ggplot(aes(term, beta, fill = factor(topic))) +
  geom_bar(stat = "identity", show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free") +
  coord_flip()
```

Plotting out topic terms β_k



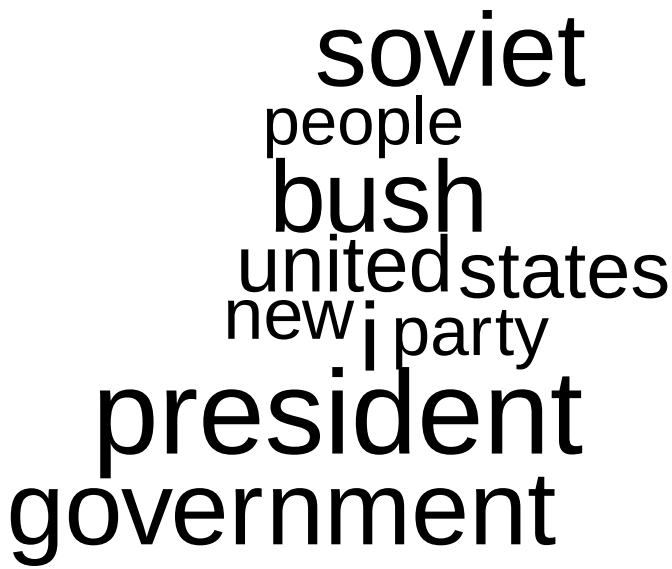
Alternative: Use word clouds to plot out β_k

```
library(wordcloud)
ap_top_terms<-data.table(ap_top_terms)
wordcloud(ap_top_terms[topic==1]$term, ap_top_terms[topic==1]$beta)
wordcloud(ap_top_terms[topic==2]$term, ap_top_terms[topic==2]$beta)
wordcloud(ap_top_terms[topic==3]$term, ap_top_terms[topic==3]$beta)
```

Word clouds for β_k

Word clouds for β_k

Word clouds for β_k



Plotting out document topic associations (i.e. our θ_d's)

```
ap_gamma <- data.table(tidy(ap_lda, matrix = "gamma"))

ap_gamma[document==1]
```

```
##    document topic    gamma
## 1:        1     1 0.000462
## 2:        1     2 0.999076
## 3:        1     3 0.000462
```

Asset Declarations of Brasilian Politicians

Asset declarations available from TSE
`http://divulgacandcontas.tse.jus.br`.

Asset Declarations of Brasilian Politicians

	DS_TIPO_BEM_CANDIDATO	N	TYPE
1	ne	883060	NOTHING
2	apartamento	33877	REAL ESTATE
3	casa	184618	REAL ESTATE
4	terra nua	33058	REAL ESTATE
5	outros bens imÃ³veis	51954	REAL ESTATE
6	veÃ-culo automotor terrestre c	331267	CAR
9	depÃ³sito bancÃ¡rio em conta c	44363	FINANCIAL ASSETS
11	terreno	133371	REAL ESTATE
12	outras participaÃ§Ãµes societÃ	11870	FINANCIAL ASSETS
13	quotas ou quinhÃµes de capital	44972	FINANCIAL ASSETS
14	outros fundos	9124	FINANCIAL ASSETS
16	aplicaÃ§Ã£o de renda fixa cdb r	14504	FINANCIAL ASSETS
24	prÃ©dio comercial	10405	REAL ESTATE
25	crÃ©dito decorrente de alienaÃ	464	FINANCIAL ASSETS
26	ouro ativo financeiro	660	FINANCIAL ASSETS

Asset declarations available from TSE
`http://divulgacandcontas.tse.jus.br`.

Asset declarations without assuming sparsity

```
library(quanteda)
load("R/BEM.rdata")
BEM[, `:=`(rowid, 1:nrow(BEM))]
BEM.corpus <- corpus(BEM[1000:nrow(BEM)]$BEMDETAIL)
docnames(BEM.corpus) <- BEM[1000:nrow(BEM)]$rowid
BEM.dfm <- dfm(BEM.corpus, ngrams = 1, stem = TRUE)
BEM.dfm <- dfm_trim(BEM.dfm, min_count = 5, min_docfreq = 5)
BEM.dfm <- dfm_weight(BEM.dfm, type = "tfidf")
BEM.lda <- LDA(convert(BEM.dfm, to = "topicmodels"), k = 5, control = list(alpha = 100, seed = 20022017))

get_terms(BEM.lda, 8, threshold = 0.005)

##      Topic 1     Topic 2     Topic 3     Topic 4     Topic 5
## [1,] "de"        "do"        "de"        "com"       "de"
## [2,] "do"        "no"        "com"       "no"        "com"
## [3,] "com"       "terreno"   "no"        "terreno"   "no"
## [4,] "residenci" "residenci" "terreno"   "residenci" "terreno"
## [5,] "lote"      "da"        "residenci" "da"        "lote"
## [6,] "veiculo"   "lote"      "da"        "placa"     "veiculo"
## [7,] "capit"     "placa"     "moto"      "terra"     "rural"
## [8,] "ltda"      "rural"     "m"         "bairro"    "brasil"
```

Asset declarations without assuming sparsity

We see the vocabulary is very similar, this is because we forced the hyperparameter α for topic proportions assignment to be large, resulting in peaked central distribution.

```
ASSIGNMENTS <- data.table(rowid = names(get_topics(BEM.lda)), topic = as.character(get_topics(BEM.lda)))
BEM <- join(BEM, ASSIGNMENTS)

BEM[topic == 1][1:10]$BEMDETAIL


##  [1] "participação capitalda empresa antonio fonseca silva"
##  [2] "semoventes equinos cavalos burros"
##  [3] "casa própria"
##  [4] "ações on banco do brasil"
##  [5] "um ponto comercial av rui lino"
##  [6] "caminhao ford cargo e ano cor branca placa hfd"
##  [7] "veiculo ford fiesta portas"
##  [8] "camionete"
##  [9] "conta poupanca vinculada conta corrente"
## [10] "um veiculo honda civic ano"
```

This is not great...

Asset declarations without assuming sparsity

```
posterior(BEM.lda)$topics[1:10, ]

##          1     2     3     4     5
## 1001 0.199 0.200 0.202 0.200 0.199
## 1002 0.200 0.199 0.199 0.202 0.200
## 1003 0.200 0.199 0.200 0.200 0.201
## 1004 0.199 0.202 0.200 0.199 0.200
## 1005 0.201 0.199 0.198 0.201 0.200
## 1006 0.201 0.200 0.199 0.201 0.199
## 1007 0.201 0.201 0.199 0.199 0.200
## 1008 0.200 0.199 0.201 0.201 0.200
## 1009 0.199 0.201 0.200 0.199 0.201
## 1010 0.200 0.200 0.200 0.200 0.200
```

These are all super close to 20%, due to our choice of high α, the prior distribution is very peaked at the centroid of that simplex and the evidence we observe has a hard time moving us away from these priors.

Asset declarations with sparsity

```
library(quanteda)
load("R/BEM.rdata")
BEM[, `:=`(rowid, 1:nrow(BEM))]
BEM.corpus <- corpus(BEM[1000:nrow(BEM)]$BEMDETAIL)
docnames(BEM.corpus) <- BEM[1000:nrow(BEM)]$rowid
BEM.dfm <- dfm(BEM.corpus, ngrams = 1, stem = TRUE)
BEM.dfm <- dfm_trim(BEM.dfm, min_count = 5, min_docfreq = 5)
BEM.dfm <- dfm_weight(BEM.dfm, type = "tfidf")
BEM.lda <- LDA(convert(BEM.dfm, to = "topicmodels"), k = 5, control = list(alpha = 0.005, seed = 20022017))

get_terms(BEM.lda, 8, threshold = 0.005)

##      Topic 1   Topic 2     Topic 3   Topic 4   Topic 5
## [1,] "do"      "de"        "placa"   "da"      "placa"
## [2,] "banco"   "do"        "veiculo" "capit"   "veiculo"
## [3,] "conta"   "com"       "moto"    "empresa" "automovel"
## [4,] "brasil"  "no"        "fiat"    "ltda"    "gol"
## [5,] "saldo"   "terreno"   "honda"   "jose"    "vw"
## [6,] "corrent" "residenci" "modelo"  "titan"   "dinheiro"
## [7,] "cabeça"  "lote"      "ford"    "firma"   "palio"
## [8,] "gado"    "m"         "mod"     "qd"      "trator"
```

Asset declarations with sparsity

```
ASSIGNMENTS <- data.table(rowid = names(get_topics(BEM.lda)), topic = as.character(get_topics(BEM.lda)))
BEM <- join(BEM, ASSIGNMENTS)

BEM[topic == 1][1:10]$BEMDETAIL


##  [1] "semoventes equinos cavalos burros"
##  [2] "casa própria"
##  [3] "ações on banco do brasil"
##  [4] "ações bradesco empresa de capital aberto"
##  [5] "bradesco fundo de investimento em ações petrobras"
##  [6] "conta poupanca vinculada conta corrente"
##  [7] "situado na rua alexandre gomes da fonseca varzea"
##  [8] "ford ecosport"
##  [9] "bovinos"
## [10] "caixa econômica federal agência contas pertence aos filhos thiago marcela"
```

Comparing with labelled data

```
table(BEM[DS_TIPO_BEM_CANDIDATO == "veículo automotor terrestre caminhão automóvel moto etc"]$DS_TIPO_BEM_CAN
    BEM[DS_TIPO_BEM_CANDIDATO == "veículo automotor terrestre caminhão automóvel moto etc"]$topic)

##
##                                                          1   2   3   4   5
##   veículo automotor terrestre caminhão automóvel moto etc 430 928 730 389 535
```

Compare with

```
get_terms(BEM.lda, 8, threshold = 0.005)

##      Topic 1   Topic 2     Topic 3   Topic 4   Topic 5
## [1,] "do"      "de"        "placa"   "da"      "placa"
## [2,] "banco"   "do"        "veiculo" "capit"   "veiculo"
## [3,] "conta"   "com"       "moto"    "empresa" "automovel"
## [4,] "brasil"  "no"        "fiat"    "ltda"    "gol"
## [5,] "saldo"   "terreno"   "honda"   "jose"    "vw"
## [6,] "corrent" "residenci" "modelo"  "titan"   "dinheiro"
## [7,] "cabeça"  "lote"      "ford"    "firma"   "palio"
## [8,] "gado"    "m"         "mod"     "qd"      "trator"
```

Scoring virgin data

```
BEM[1]$DETALHE_BEM

## [1] "UMA CASA RESIDENCIAL RUA MARIA LINA 823"

# coding virgin documents - ignores non overlapping vocabulary
posterior(BEM.lda, convert(dfm(BEM[1]$DETALHE_BEM), to = "topicmodels"))$topics

##            1     2      3     4      5
## text1 0.0114 0.467 0.0114 0.499 0.0114

BEM[topic == 3][1:10]$BEMDETAIL

##  [1] "carro modelo gol placa jtw financiado"
##  [2] "vaga de garagen na rua alagoas"
##  [3] "caminhao ford cargo e ano cor branca placa hfd"
##  [4] "veiculo ford fiesta portas"
##  [5] "camionete"
##  [6] "moto placa gvy ano modelo"
##  [7] "scania ano placa joa"
##  [8] "uma moto bros"
##  [9] "um veiculo honda civic ano"
## [10] "veículo pick up marca ford"
```

Another Example - Trump Tweets

```
load(file = "../../Data/trumpstweets.rdata")
# remove retweet entities
tw.user.df$text <- gsub("(RT|via)((?:\\b\\W*@\\w+)+)", "", tw.user.df$text)
# remove at people
tw.user.df$text <- gsub("@\\w+", "", tw.user.df$text)
# remove punctuation
tw.user.df$text <- gsub("[[:punct:]]", "", tw.user.df$text)
# remove numbers
tw.user.df$text <- gsub("[[:digit:]]", "", tw.user.df$text)
# remove html links
tw.user.df$text <- gsub("http\\w+", "", tw.user.df$text)
# remove unnecessary spaces
tw.user.df$text <- gsub("[ \t]{2,}", "", tw.user.df$text)
tw.user.df$text <- gsub("^\\s+|\\s+$", "", tw.user.df$text)
tw.user.df <- tw.user.df[text != ""]
```

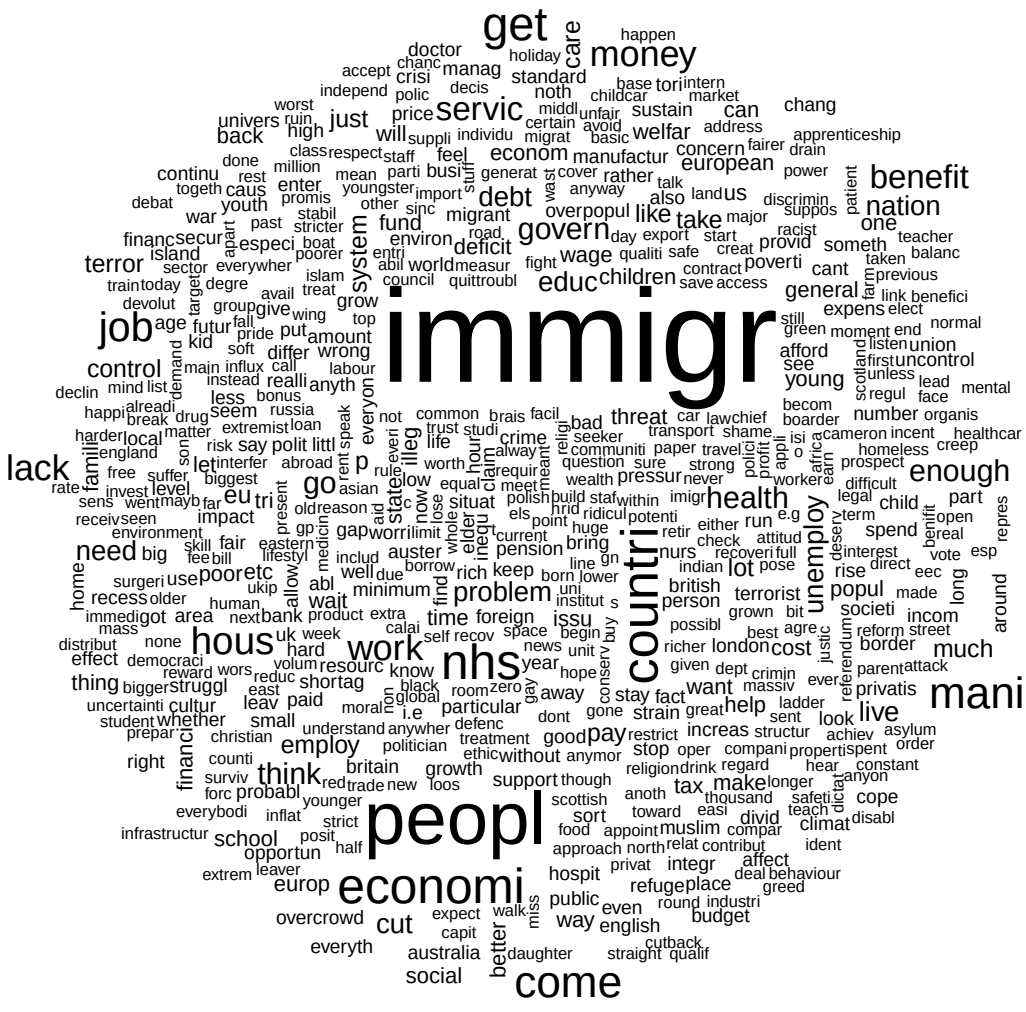
Topic model over Trump Tweets

```
## create DTM
set.seed(24022017)
library(quanteda)
## builds a corpus object using the field named text as containing the document
tw.user.df[, `:=`(docid, 1:nrow(tw.user.df))]
trump.corpus <- corpus(tw.user.df)
docnames(trump.corpus) <- tw.user.df$docid
trump.dfm <- dfm(trump.corpus, remove = c(stopwords("english"), "can", "say", "one", "way",
    "use", "also", "howev", "tell", "will", "much", "need", "take", "tend", "even", "like",
    "particular", "rather", "said", "get", "well", "make", "ask", "come", "end", "first", "two",
    "help", "often", "may", "might", "see", "someth", "thing", "point", "post", "look", "right",
    "now", "think", "'ve ", "'re ", "anoth", "put", "set", "new", "good", "want", "sure",
    "kind", "larg", "yes,", "day", "etc", "quit", "sinc", "attempt", "lack", "seen", "awar",
    "littl", "ever", "moreov", "though", "found", "abl", "enough", "far", "earli", "away",
    "achiev", "draw", "last", "never", "brief", "bit", "entir", "brief", "great", "lot"), ngrams = 1,
    stem = TRUE)
trump.dfm <- dfm_trim(trump.dfm, min_count = 5, min_docfreq = 2)
trump.dfm <- dfm_weight(trump.dfm, type = "tfidf")
trump.lda <- LDA(convert(trump.dfm, to = "topicmodels"), k = 5, control = list(alpha = 0.01,
    seed = 20022017))
get_terms(trump.lda, 5)
```

```
##      Topic 1          Topic 2  Topic 3 Topic 4   Topic 5
## [1,] "join"           "just"   "amp"   ""       "makeamericagreatagain"
## [2,] "pm"             "rate"   "job"   "clinton" "america"
## [3,] "colorado"       "immigr" "peopl" "crook"   "interview"
## [4,] "tomorrow"       "goofi"  "vote"  "bad"     "support"
## [5,] "crookedhillari" "peopl"  "bring" "berni"   "enjoy"
```

```
assignment <- data.frame(docid = names(topics(trump.lda)), topic = topics(trump.lda))

tw.user.df <- join(tw.user.df, assignment)[!is.na(topic)]
```

A third example - Brexit



Topic model over Brexit single most important issue

```
## create DTM
library(quanteda)
library(tidytext)
library(topicmodels)
## builds a corpus object using the field named text as containing the document
C <- corpus(DTA$A1)
## defining features that are informative about leaveeu decision using e.g. document scaling
## method
docnames(C) <- as.numeric(DTA$finalserialno)
C.dfm <- dfm(C, tolower = TRUE, stem = TRUE, removeNumbers = FALSE, removePunct = TRUE, remove = stopwords("en

C.lda <- LDA(convert(C.dfm, to = "topicmodels"), k = 5, control = list(alpha = 0.2, seed = 10052017))

get_terms(C.lda, 10)

##       Topic 1   Topic 2   Topic 3    Topic 4    Topic 5
##  [1,] "eu"      "immigr"  "terror"   "nhs"      "peopl"
##  [2,] "get"     "peopl"   "money"    "hous"     "economi"
##  [3,] "nhs"     "countri" "go"       "job"      "get"
##  [4,] "wait"    "come"    "cut"      "unemploy" "health"
##  [5,] "need"    "mani"    "govern"   "lack"     "work"
##  [6,] "nation"  "benefit" "servic"   "enough"   "job"
##  [7,] "popul"   "take"    "countri"  "money"    "debt"
##  [8,] "threat"  "let"     "european" "peopl"    "servic"
##  [9,] "foreign" "control" "back"     "employ"   "nation"
## [10,] "stay"    "work"    "financ"   "deficit"  "educ"

assignment <- data.frame(finalserialno = as.numeric(names(topics(C.lda))), topic = topics(C.lda))

DTA <- join(DTA, assignment)[!is.na(topic)]
```

Topic model over Brexit single most important issue

```
## create DTM

# Immigration topic
DTA[topic == 2, list(topic, substr(A1, 1, 60))][1:10]
```

```
##     topic                                                          V2
##  1:     2                                                  Immigration
##  2:     2                                           too many immigrants
##  3:     2                                                  Immigration
##  4:     2                                                    criminals
##  5:     2                                                  IMMIGRATION
##  6:     2                                                  immigration
##  7:     2 Influx of people who shouldn't be here British people should
##  8:     2                                                   immigrants
##  9:     2 Immigration. we are letting too many people in we are sinkin
## 10:     2 Immigration the amount of people we having coming in with no
```

```
# Austerity/ public goods/ employment topic
DTA[topic == 4, list(topic, substr(A1, 1, 60))][1:10]
```

```
##     topic                                                       V2
##  1:     4                                             unemployment
##  2:     4                                             unemployment
##  3:     4                                LACK OF STAFF IN THE NHS
##  4:     4                               WELFARE REFORM -AUSTERITY
##  5:     4                             Lack of Money too much debt
##  6:     4                                                 Housing
##  7:     4                                                 The NHS
##  8:     4 small businesses as family business, funding's available
##  9:     4                      economy. cuts, and way doing welfare
## 10:     4                             Hospitals lack of resources
```

Topic Distribution

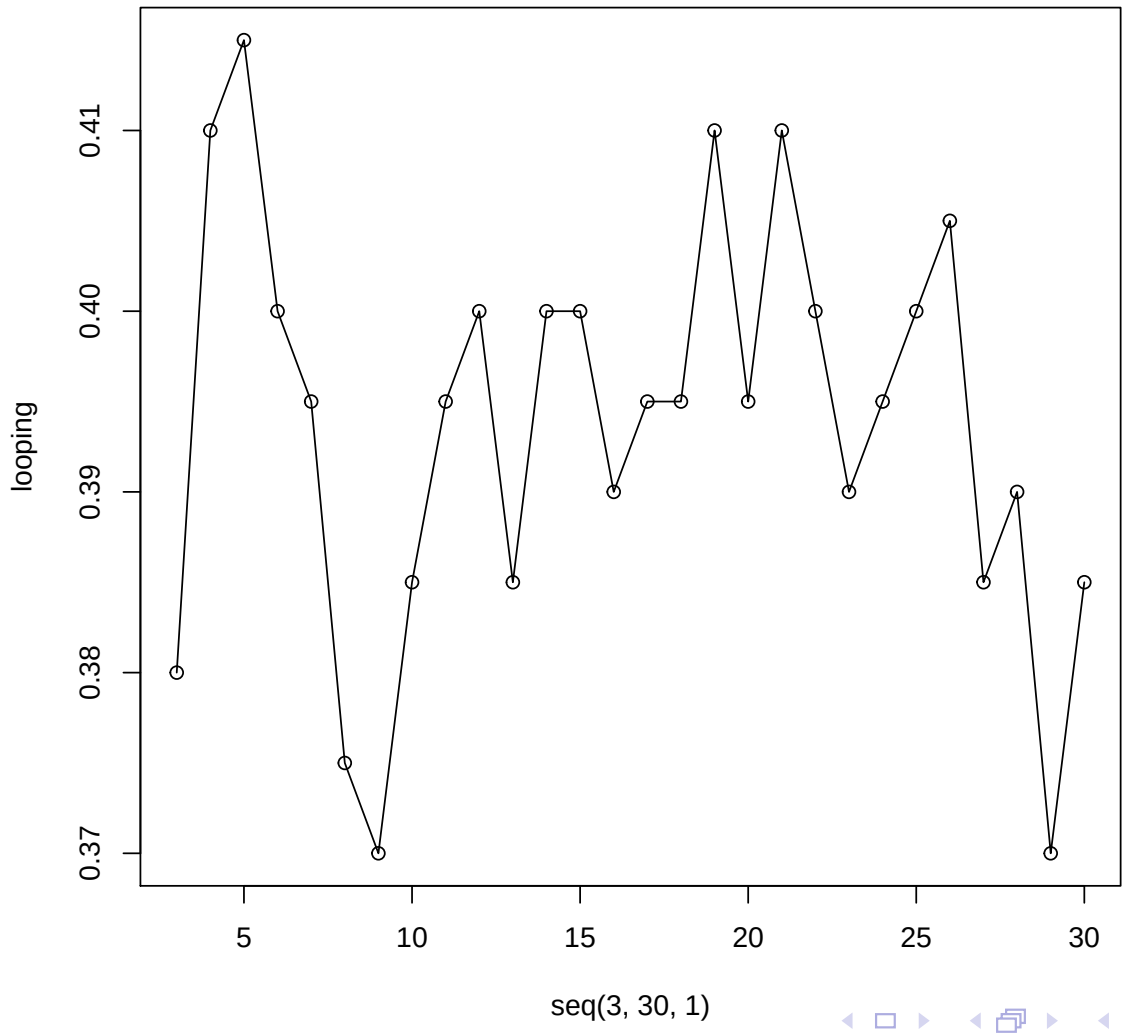
```
## create DTM

hist(DTA$topic)
```

Searching for a topic model to predict Brexit vote

```
## create DTM
leaveout <- sample(1:nrow(DTA), 200)  ## sample 500 random indices
looping <- NULL
for (kk in seq(3, 30, 1)) {
    C.lda <- LDA(convert(C.dfm, to = "topicmodels"), k = kk, control = list(alpha = 0.075,
        seed = 10052017))
    assignment <- data.frame(finalserialno = as.numeric(names(topics(C.lda))), topic = topics(C.lda))

    MAT <- data.frame(leaveeu = DTA$leaveeu, C.lda@gamma)
    # train the model WITHOUT these observations (-index removes those obs)
    trained <- glm(leaveeu ~ ., data = MAT[-leaveout, ], binomial(link = logit))
    # get the predicted probability of spam on the left out data
    glm.probs <- predict(trained, newdata = MAT[leaveout, ], type = "response")
    # plot the OOS fit

    glm.pred <- rep("remain", length(glm.probs))
    glm.pred[glm.probs > 0.5] <- "leave"

    looping <- c(looping, sum(diag(2) * table(glm.pred, MAT[leaveout, ]$leaveeu))/200)
}
```

Trade-off between topic depth and prediction performance?

```
## create DTM
plot(seq(3, 30, 1), looping)
lines(seq(3, 30, 1), looping)
```



Predicting Conflict using NYTimes corpus and topic modelling

- Construct NY Times corpus for the past 50 years, digital versions.
- Estimate a topic model and identify the topics that are associated with conflict content (essentially browsing through the word clouds)
- Use topic shares to predict political instability/conflict using *within country* variation.
- Benchmark model is one with country and year fixed effects.

"Reading between the Lines: Prediction of Political Violence", Hannes Mueller and Christopher Rauh, 2017.

Predicting Conflict using NYTimes corpus and topic modelling

"Reading between the Lines: Prediction of Political Violence", Hannes Mueller and Christopher Rauh, 2017.

Predicting Conflict using NYTimes corpus and topic modelling

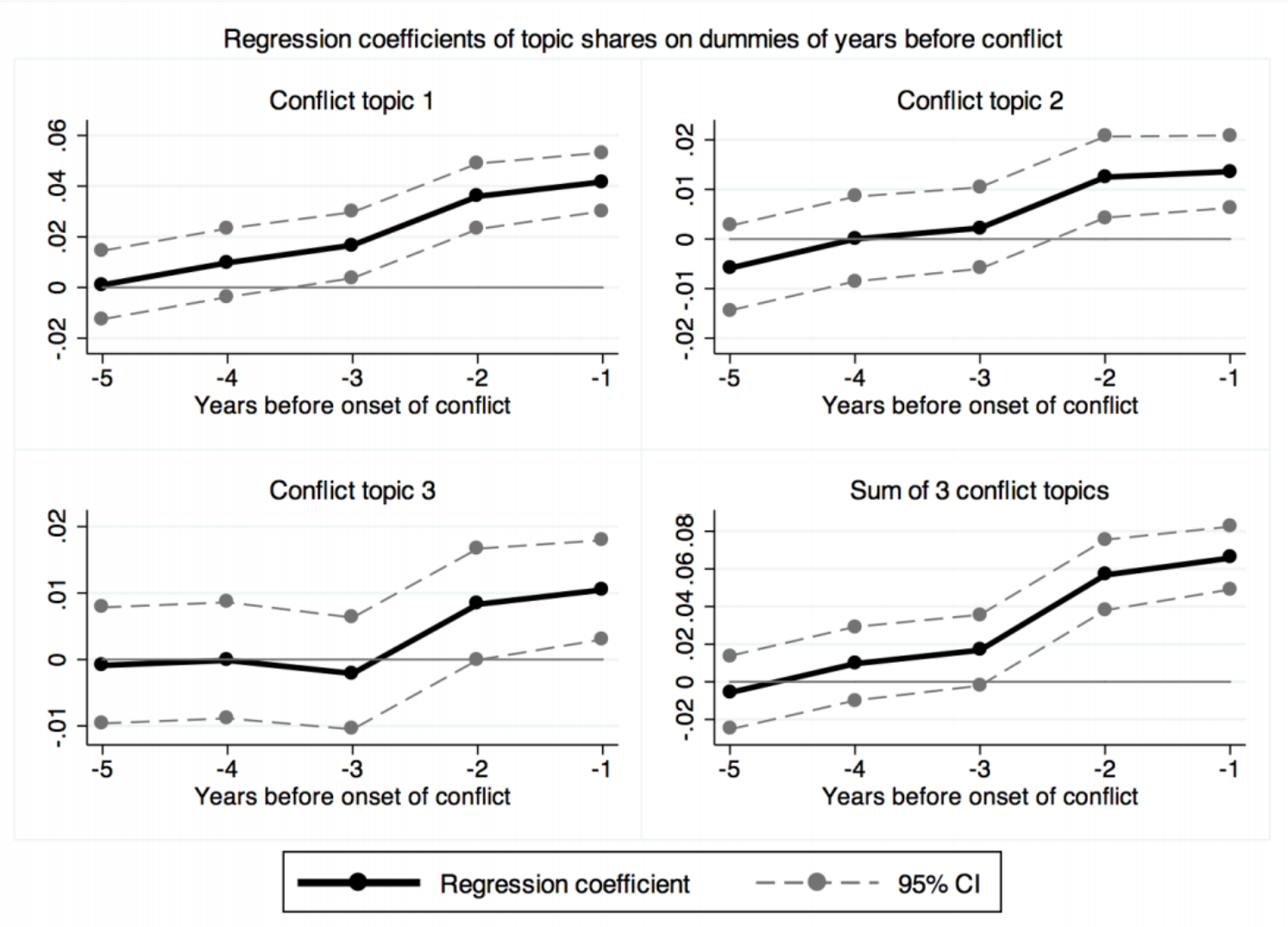
(a) Conflict Topic 2

(b) Conflict Topic 3

"Reading between the Lines: Prediction of Political Violence", Hannes Mueller and Christopher Rauh, 2017.

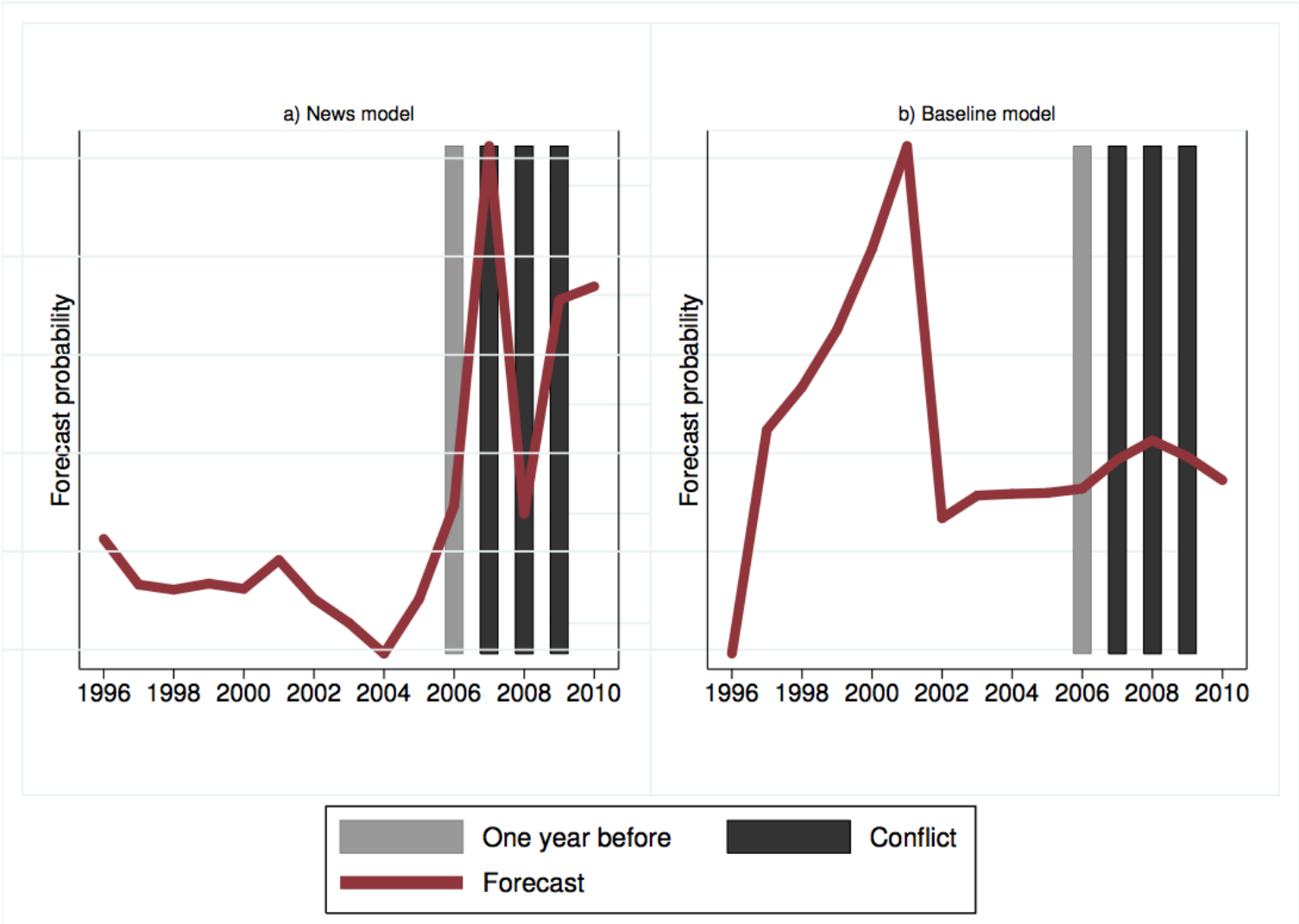
Predicting Conflict using NYTimes corpus and topic modelling

Figure 6: Conflict Topics and Years Before Conflict



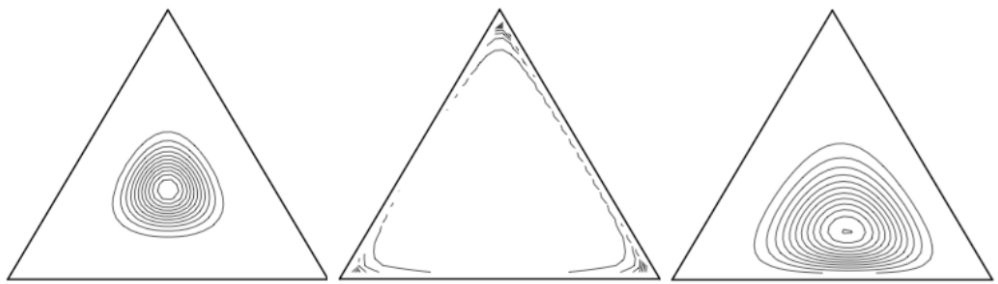
Predicting Conflict using NYTimes corpus and topic modelling

Figure 16: Prediction of Touareg Rebellion in Mali



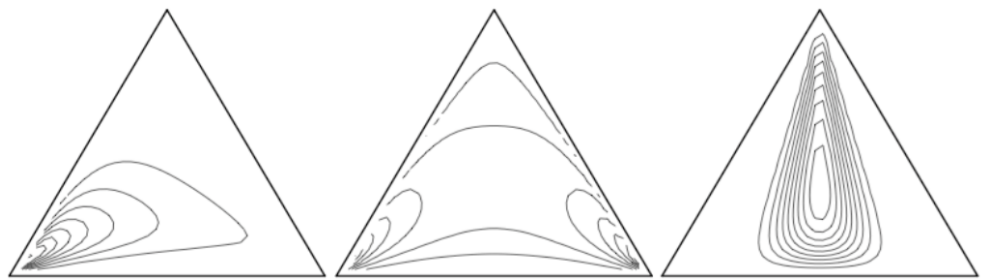
Correlated Topic models

- Dirichlet distribution over simplex only allows for distributions that are basically independent of one another.
- Alternative distributions over simplex allow for correlations.



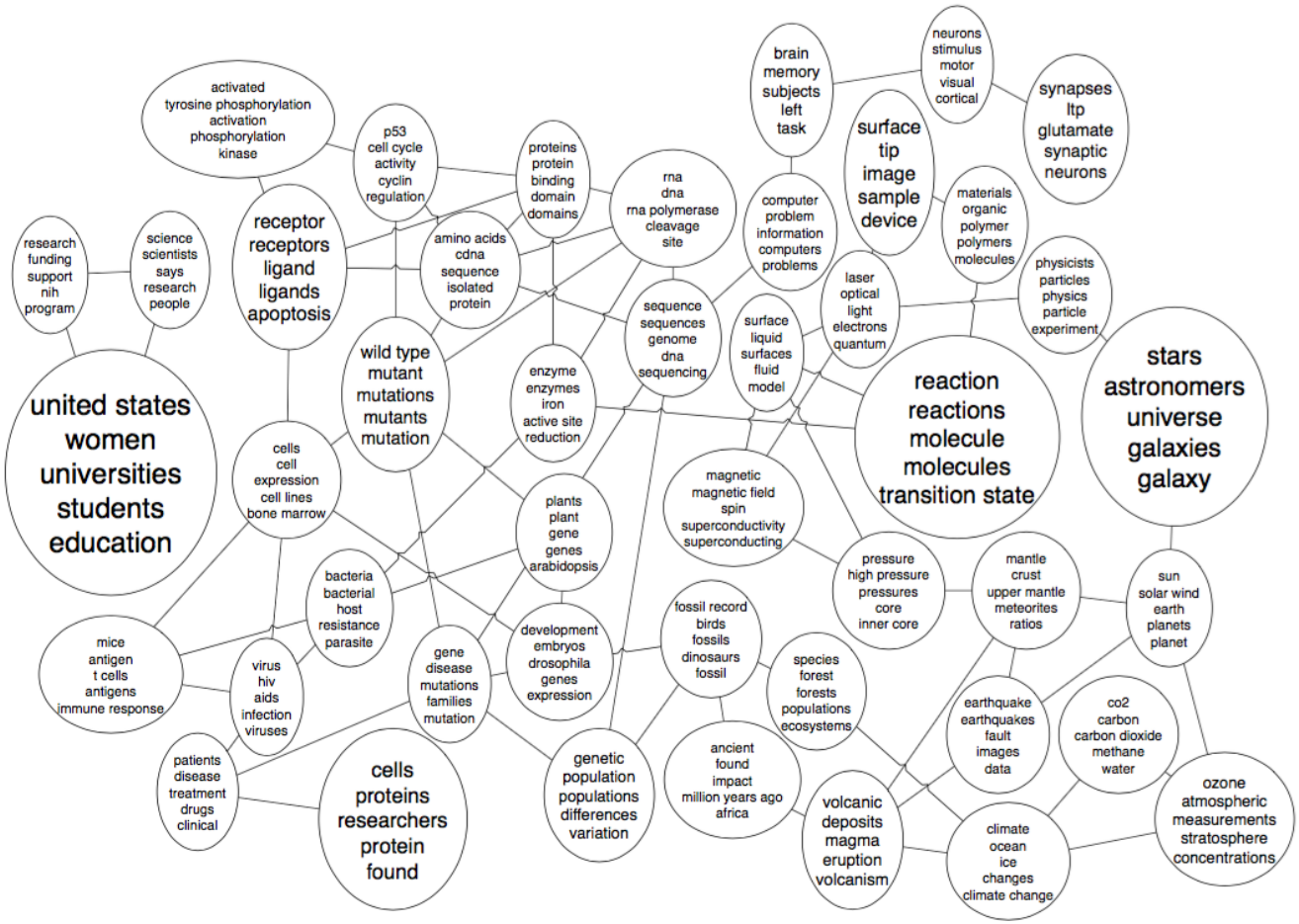
Correlated Topic models

- Dirichlet distribution over simplex only allows for distributions that are basically independent of one another.
- Alternative distributions over simplex allow for correlations.



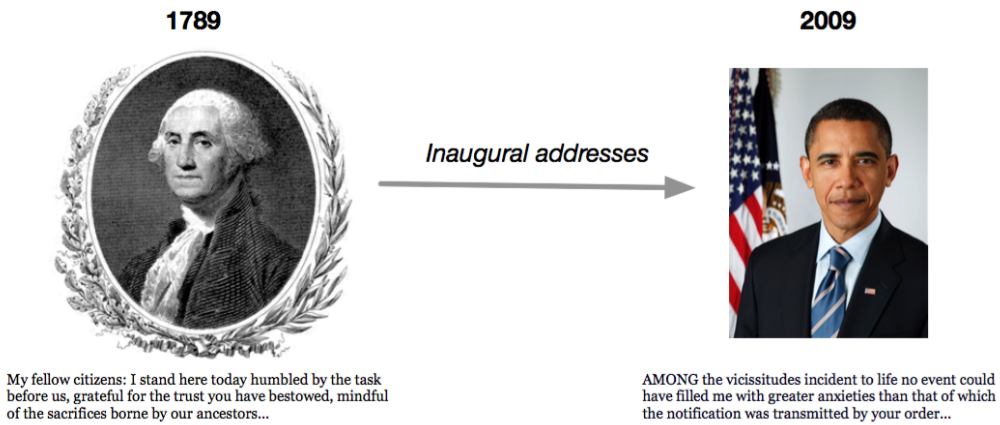
Correlated Topic models

- Dirichlet distribution over simplex only allows for distributions that are basically independent of one another.
- Alternative distributions over simplex allow for correlations.



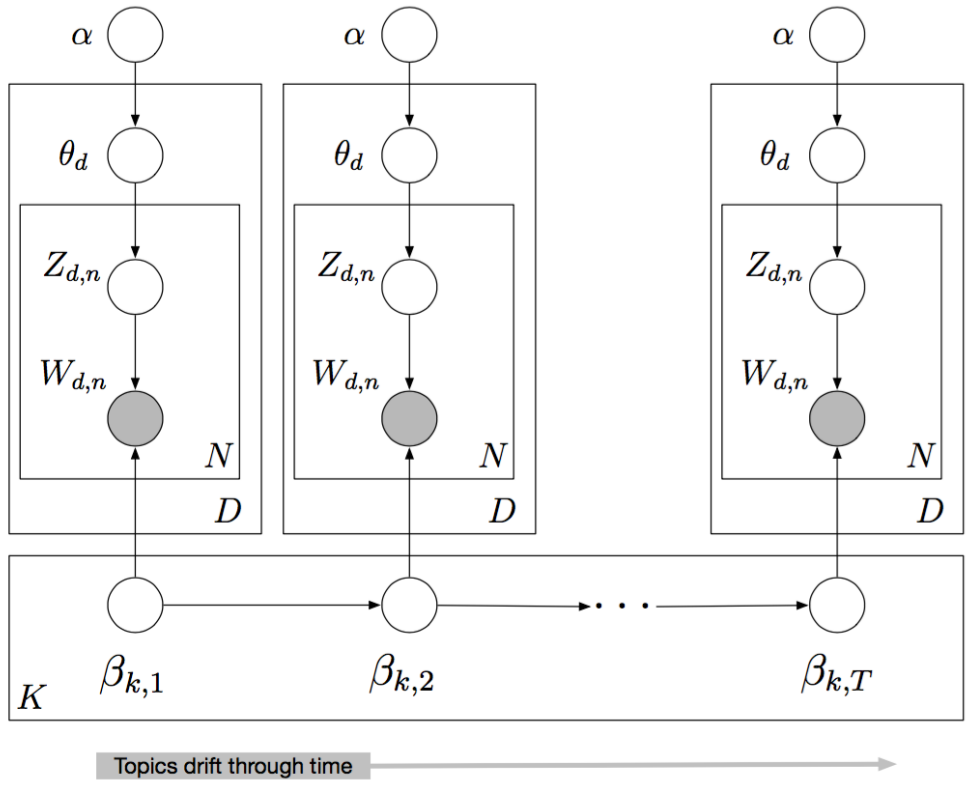
Dynamic Topic models

- Basic topic model ignores the order of documents.
- Dynamic topic models lets topics drift over time.



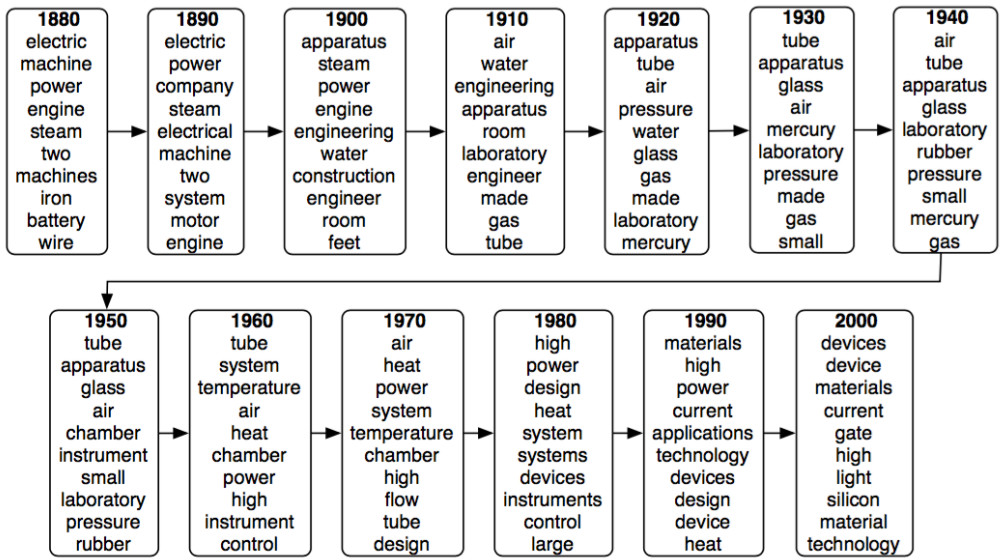
Dynamic Topic models

- Basic topic model ignores the order of documents.
- Dynamic topic models lets topics drift over time.



Dynamic Topic models

- Basic topic model ignores the order of documents.
- Dynamic topic models lets topics drift over time.



Plan

Topic modelling

Topic models as embeddings ?

Overview

- Topic models produce interpretable vector representations.
- These can be treated as a form of embedding.
- We compare topic model outputs to standard embeddings.

Topic Modeling Basics

Latent Dirichlet Allocation (LDA):

- D: Number of documents
- V: Vocabulary size
- K: Number of topics

Each document d has:

$$\boldsymbol{\theta}_d \in \Delta^{K-1} \quad \text{(document-topic distribution)}$$

Each topic k has:

$$\boldsymbol{\phi}_k \in \Delta^{V-1} \quad \text{(topic-word distribution)}$$

LDA Generative Process

For each word in a document:

1. Sample topic $z \sim \text{Multinomial}(\boldsymbol{\theta}_d)$
2. Sample word $w \sim \text{Multinomial}(\boldsymbol{\phi}_z)$

What is an Embedding?

An embedding is a mapping:

$$f : \mathcal{X} \to \mathbb{R}^k$$

where $\mathcal{X}$ is a set of objects (e.g., documents or words).
Similarity is often measured using cosine similarity:

$$\text{sim}(x, y) = \frac{f(x) \cdot f(y)}{\|f(x)\| \|f(y)\|}$$

LDA as Embedding Function

Define:

$$f_{\text{LDA}}(d) = \boldsymbol{\theta}_d \in \mathbb{R}^K$$

Similarity between documents:

$$\text{sim}(d_1, d_2) = \cos(\boldsymbol{\theta}_{d_1}, \boldsymbol{\theta}_{d_2})$$

$\rightarrow$ LDA gives a valid embedding in topic space.

As we will see embeddings provide dense representation in semantic space, but can also account for grammar, and other linguistic features quite well.

Word Embeddings in Topic Space

Estimate word-topic association:

$$f(w)_k = \mathbb{P}(z = k \mid w) \propto \mathbb{P}(w \mid z = k) \cdot \mathbb{P}(z = k)$$

You can also embed words using topic distributions.

Feature	Topic Models	Embeddings
Interpretability	High	Low
Dimensionality	Low ($K \approx 10$–100)	High (e.g., 300, 768)
Training Objective	Generative	Predictive
Sparsity	Often Sparse	Dense