Topic 4: Clustering and Topic Modelling

LSE MY459: Computational Text Analysis and Large Language Models

https://lse-my459.github.io/

Ryan Hübert
Associate Professor of Methodology

Slides last updated: 8 February 2026

Discovering patterns across documents

Last week: we formalised comparisons across documents using the vector space and probabilistic approaches

This week: techniques for *discovering* **latent structure** in collections of documents

- ➜ Let the structure emerge from data (**unsupervised learning**)
- ➜ Interpret and evaluate the resulting structure

What do we mean by "latent structure"?

- ➜ Core goal: identify how documents organise into groups
- ➜ This organisation is a form of *structure*
- ➜ It is *latent* because it is unobserved, but inferable from data

Basic framework

We begin with a DFM, $\mathbf{W}$, where:

- Document features are indexed by $j \in \{1, ..., J\}$
  - We'll assume: preprocessed unigrams ("tokens")
  - The set of features is the "vocabulary" of the corpus, which has J unique "types"
- Documents are indexed by $i \in \{1, ..., N\}$
  - Each document represented by a vector of token counts, $\mathbf{W}_i$
- Assume a set of K unobserved groups, each indexed by k
  - Each document's membership in a group is represented by $\boldsymbol{\pi}_i$
  - Within each group k, $\boldsymbol{\mu}_k$ indicates how word (token) use varies in the group's documents

Ways to discover latent organisation

These methods can differ in two main ways:

1. Assignment structure:
   - ➜ Hard: each document belongs to one group, e.g. $\boldsymbol{\pi}_i = (0, 1, 0)$
   - ➜ Soft: documents are distributed across groups in proportions, e.g. $\boldsymbol{\pi}_i = (0.2, 0.3, 0.5)$
2. Level of organisation
   - ➜ Document-level structure: documents are grouped into **clusters**
   - ➜ Token-level structure: tokens *within* documents are grouped into **topics**
   - ➜ Affects what level of latent structure $\boldsymbol{\mu}_k$ summarizes (details to come)

Standard recipe

We use data to discover latent organisation using this recipe:

1. *Research design*: think about what you want to know
   - ➜ Commit to discovering K meaningful groups
   - ➜ Decide how groups should assigned: hard versus soft, and/or documents versus features
   - ➜ Write a model or choose an algorithm
2. *Analysis*: determine documents' membership across K groups
   - ➜ Use estimation techniques to infer latent structure
3. *Interpretation*: given the estimated membership of documents in K groups, what do they mean substantively? (yikes!)

Two flavours: **clustering** and **topic modelling**

Clustering methods

Clustering methods seek to discover how documents are organised across K groups, typically referred to as **clusters**

This is a huge area!

- ➜ Our priority is conceptual foundation
- ➜ The goal is substantive application to text analysis

We'll look at one vector space approach and one model-based approach

- ➜ k-means clustering
- ➜ mixture modelling

Hard clustering

k-means clustering

Very common method for clustering: **k-means clustering**

- Simple, efficient and common
- Each document is assigned to one of K clusters using an iterative algorithm
- Algorithm tries to group documents so that items in the same cluster are close to their cluster's "average" document
- Although: sensitive to algorithm's initialisation

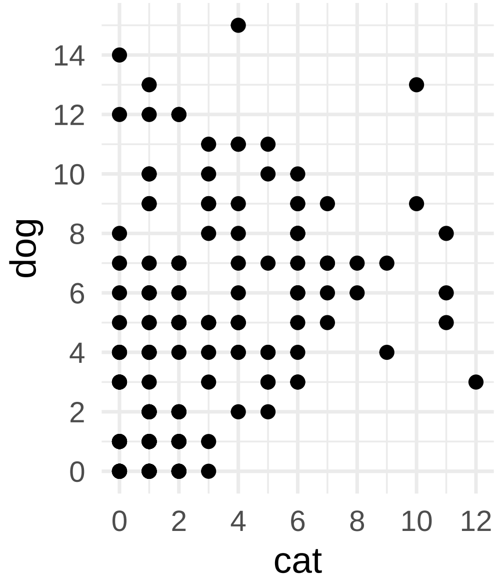
There are other clustering algorithms

- See extra slides for another clustering approach

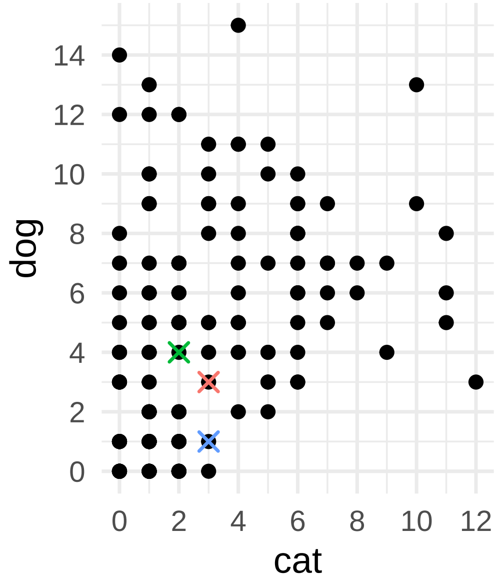
Algorithm details

0. For DFMs: perform TF-IDF weighting with a normalisation
1. Choose starting values
   - ➜ Assign random positions to K starting values that will serve as the “cluster centres”, known as “centroids”
2. Assign each item to the class of the centroid that is “closest”
   - ➜ Euclidean distance is most common, but others can be used
3. Update: recompute the cluster centroids as the mean value of the points assigned to that cluster
4. Repeat reassignment of points and updating centroids
5. Repeat 2–4 until some stopping condition is satisfied
   - ➜ e.g. when no items are reclassified following update of centroids

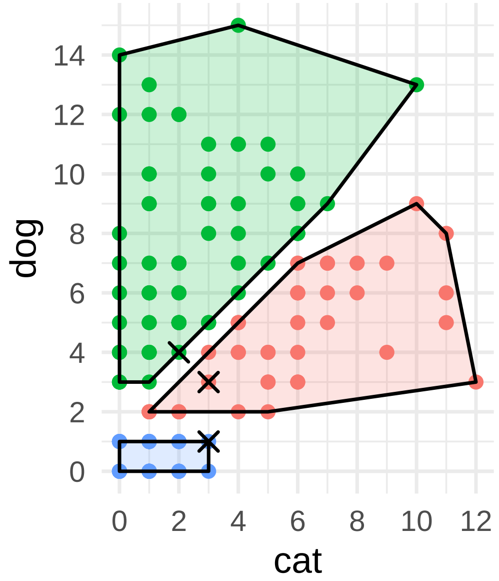
k-means clustering illustrated



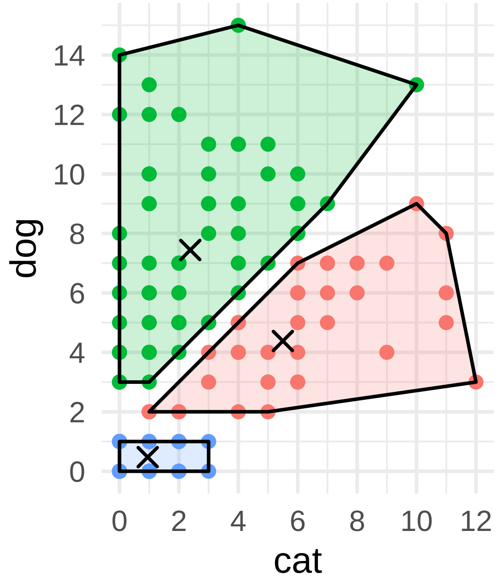
k-means clustering illustrated



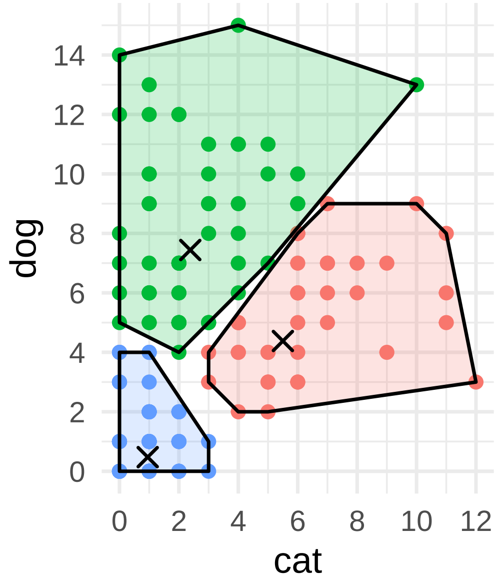
k-means clustering illustrated



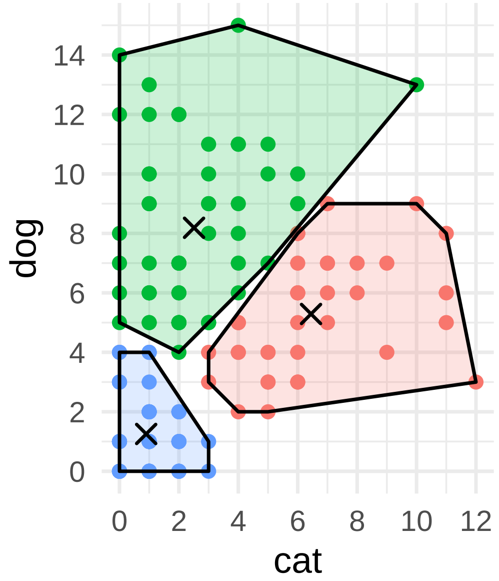
k-means clustering illustrated



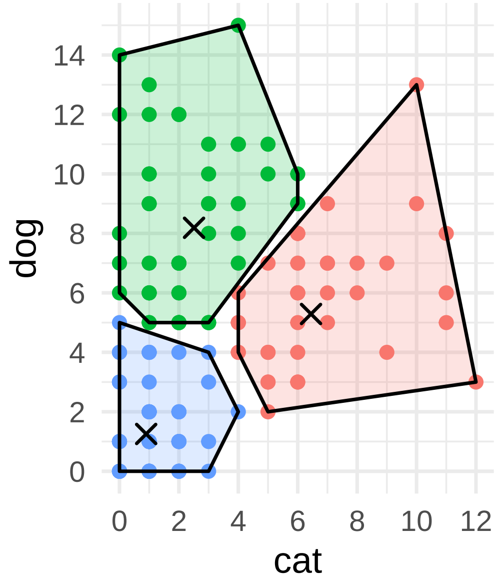
k-means clustering illustrated



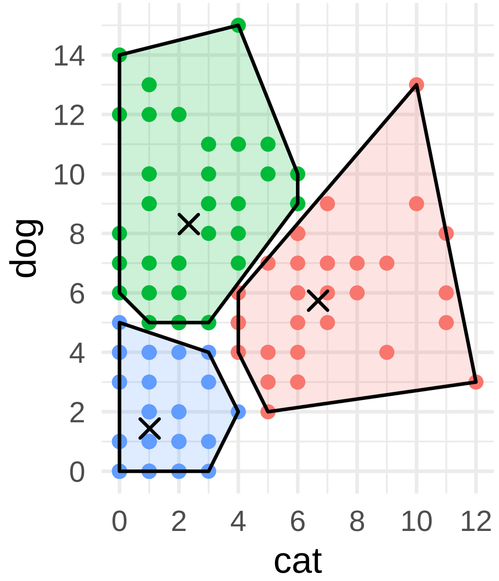
k-means clustering illustrated



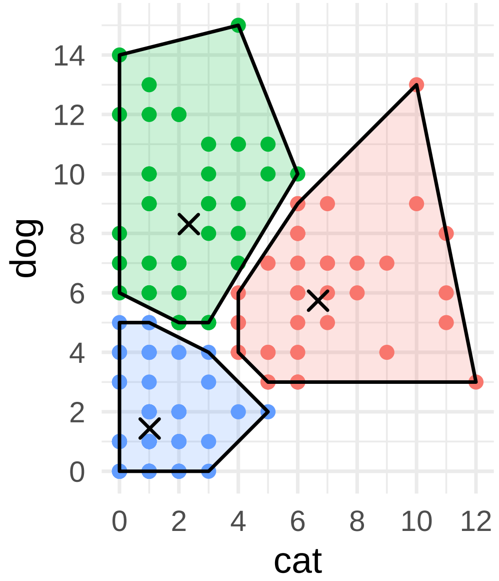
k-means clustering illustrated



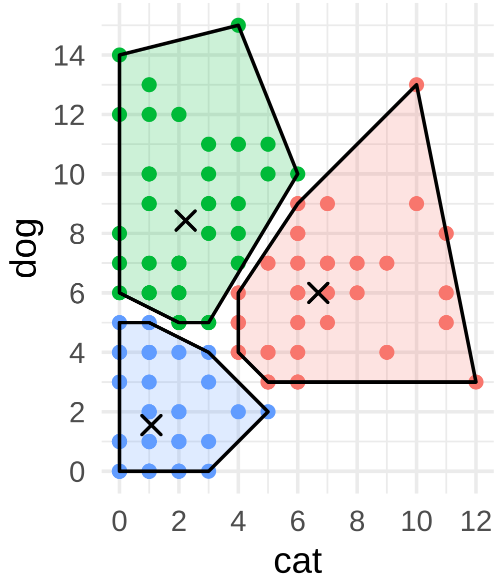
k-means clustering illustrated



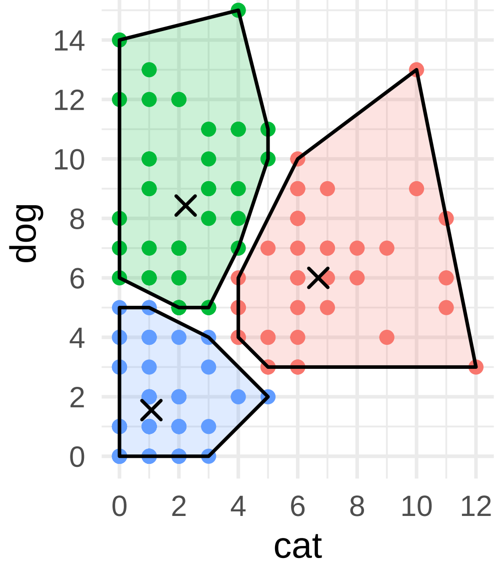
k-means clustering illustrated



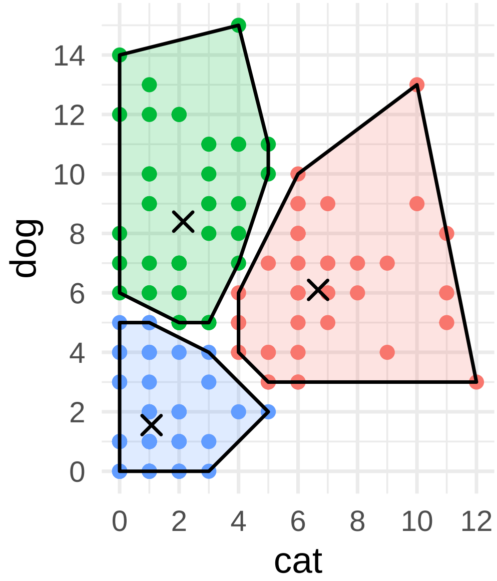
k-means clustering illustrated



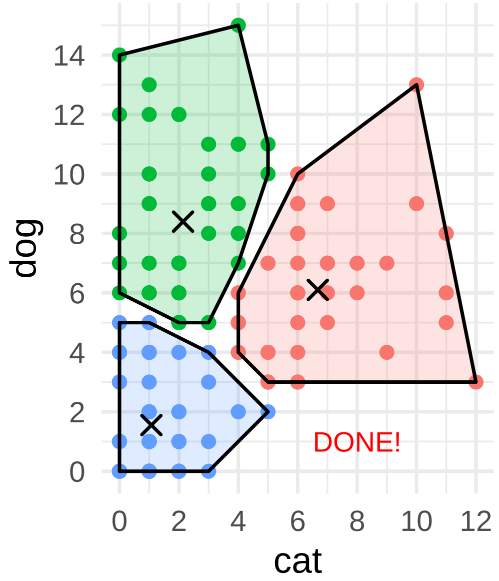
k-means clustering illustrated



k-means clustering illustrated



k-means clustering illustrated



Calculating centroids

The **centroid** of a cluster k is a J-length vector $\boldsymbol{\mu}_k = (\mu_{1k}, ..., \mu_{Jk})$ giving the "mean" of that cluster

- ➜ When we represent documents by token frequencies, a centroid is a within-cluster mean calculated over counts for each feature j

With real data: compute an estimated centroid $\widehat{\boldsymbol{\mu}}_k$ for each cluster k—the sample mean of the documents assigned k

$$\hat{\mu}_{jk} = \frac{\overbrace{\sum_{i=1}^{N} \pi_{ik} W_{ij}}^{\substack{\text{feature } j \text{ term frequency} \\ \text{across docs in cluster } k}}}{\underbrace{\sum_{i=1}^{N} \pi_{ik}}_{\text{docs in cluster}}}$$

Roughly: it captures the "average" word usage in a cluster

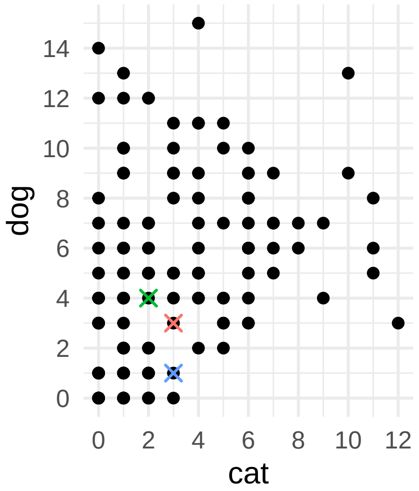
Calculating centroids

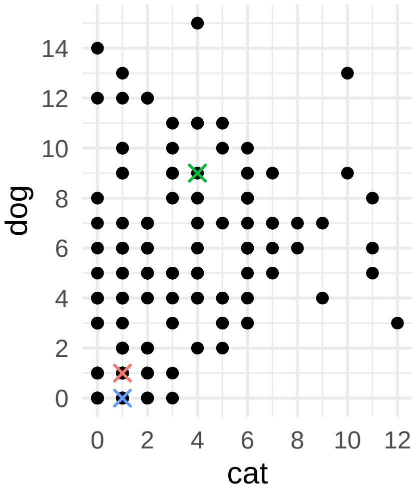
k	i	cat	dog	rat
1	1	2	1	0
1	2	3	1	1
1	3	0	2	4
2	4	1	0	3
2	5	2	2	1

$$\widehat{\mu}_{\text{cat},1} = \frac{2+3+0}{3} = \frac{5}{3} \quad \widehat{\mu}_{\text{dog},1} = \frac{1+1+2}{3} = 1 \quad \widehat{\mu}_{\text{rat},1} = \frac{0+1+4}{3} = \frac{5}{3}$$

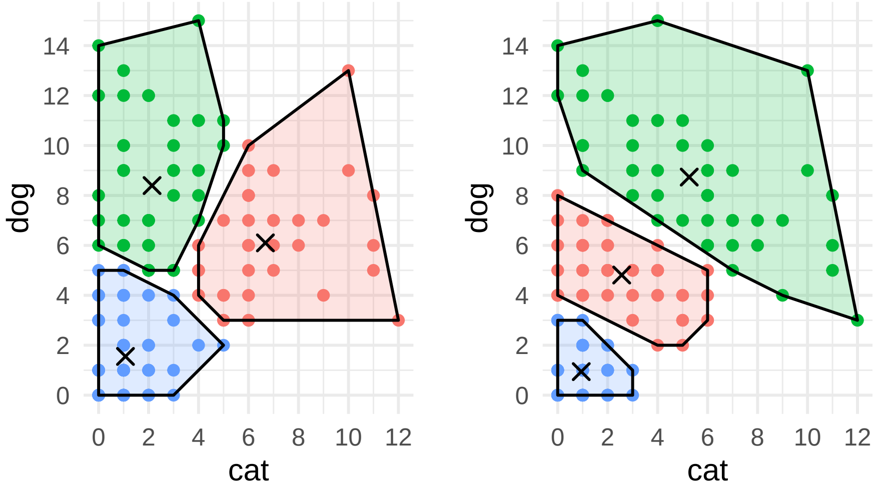
$$\widehat{\mu}_{\text{cat},2} = \frac{1+2}{2} = \frac{3}{2} \qquad \widehat{\mu}_{\text{dog},2} = \frac{0+2}{2} = 1 \qquad \widehat{\mu}_{\text{rat},2} = \frac{3+1}{2} = 2$$

Disadvantage 1: sensitivity to starting point





Disadvantage 1: sensitivity to starting point



Disadvantage 1: sensitivity to starting point

Common approach:

- ➜ do not initialise starting centroids using uniform sampling
- ➜ sample starting centroids in sequence, where subsequent centroids are sampled from a distribution placing more weight to further points
- ➜ Roughly: push starting centroids far apart

Known as **kmeans++**, and is the default used in `sklearn`

Disadvantage 2: choosing K

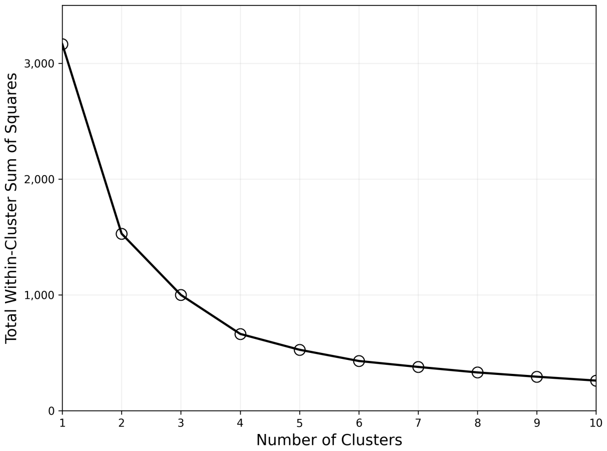
Major point of analyst discretion: how many clusters?

- ➜ Fundamental problem for *all* methods this week
- ➜ Very often based on prior information about the number of categories sought

For clustering, there are some ways to approach:

- ➜ A (rough!) guideline: set $K = \sqrt{N/2}$ where N is the number of documents to be clustered
    - ➜ usually too big: setting K too large values will improve within-cluster similarity, but risks *overfitting*
- ➜ **Elbow plots**: try different K values, and choose K beyond which are diminishing gains (not always useful)

Disadvantage 2: choosing K



Looking at the clusters

Hard clustering methods (like k-means clustering) assign each document to a cluster

How do you interpret what each cluster means, substantively?

- ➜ Short answer: read lots of the documents in each cluster

But can also get a better sense by exploiting geometry of clusters:

- ➜ Look at the top features from each centroid
    - ➜ Formally, which features correspond to highest values in $\widehat{\boldsymbol{\mu}}_k$?
- ➜ Sample documents near the centroids to read
- ➜ Sample documents near the boundaries to read
- ➜ Look at discriminating words for each cluster

Looking at the clusters

Recall our Trump tweets data

➜ Pre-processed, trimmed, weighted and normalised

If we perform k-means with $K = 5$, we get:

	cluster_size	top_features
0	0.131402	korea, trade, north, countri, border, deal
1	0.674340	trump, news, fake, media, america, thank
2	0.051992	honor, @whitehouse, welcom, today, host, thank
3	0.101914	tax, cut, republican, vote, senat, healthcar
4	0.040352	watch, us, busi, confid, small, level

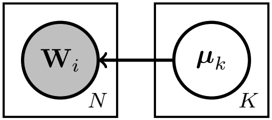
Mixture modelling for soft clustering

Back to probabilistic modelling

Recall the simple multinomial model we saw last week

- ➜ Each k has its own parameter $\boldsymbol{\mu}_k$, which we interpreted as word usage patterns by author k

We drew the plate diagram as follows

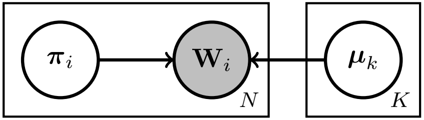


This is a very "reduced form" model

We will expand it to be more useful and powerful

Expanding the model

First, we want to explicitly incorporate clusters in the model:



The DGP for our data is now: $\mathbf{W}_i \sim \text{Multinomial}(M_i, \boldsymbol{\mu}\boldsymbol{\pi}_i)$

Where $\boldsymbol{\mu}$ is an $J \times K$ matrix: $\boldsymbol{\mu} = [\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_K]$

This is just "notation"

- ➜ No real change to the model
- ➜ Each cluster k still has its own **parameter** $\boldsymbol{\mu}_k$ (a column of $\boldsymbol{\mu}$)
- ➜ But now we *explicitly* incorporate **latent structure**

Expanding the model

This change allows us to make further additions:

- By asking: where do $\boldsymbol{\mu}_k$ and $\boldsymbol{\pi}_i$ come from?
- Formally: specify upstream generative distributions: for $\boldsymbol{\mu}_k$ and $\boldsymbol{\pi}_i$
  - Add a **prior** on the parameter $\boldsymbol{\mu}_k$
  - Specify how each document's cluster $\boldsymbol{\pi}_i$ is drawn
- **Hierarchical Bayesian model**: parameters are generated from higher-level distributions

Why do this?

- Gives structured way to combine elements into one model
- Useful for regularisation and stabilising parameter estimates

Expanding the model

What distributions will we add for $\boldsymbol{\mu}_k$ and $\boldsymbol{\pi}_i$?

First, we will assume $\boldsymbol{\pi}_i$ is sampled from a multinomial distribution:

$$\boldsymbol{\pi}_i \sim \mathrm{Multinomial}(1, \boldsymbol{\alpha})$$

where:

- $\boldsymbol{\alpha}$ is a K-length vector of cluster assignment probabilities
- This returns a one-hot vector (indicating cluster assignment)
- Equivalent to drawing a cluster from a categorical distribution
- On p. 133, GRS write $\boldsymbol{\pi}_i \sim \mathrm{Multinomial}(\boldsymbol{\eta})$
  - There is a typo (should be $\boldsymbol{\alpha}$, see figure 12.2)
  - They drop the “1”, but it is implied

Expanding the model

Second, we will assume $\boldsymbol{\mu}_k$ is sampled from a **Dirichlet distribution** with a J-length parameter vector $\boldsymbol{\eta}$:

$$\boldsymbol{\mu}_k \sim \text{Dirichlet}(\boldsymbol{\eta})$$

where:

- ➜ $\boldsymbol{\eta}$ controls how concentrated word use is within a cluster

Why the Dirichlet distribution?

- ➜ Sampling yields a vector of probability weights
- ➜ Nice mathematical properties, including conjugacy with the multinomial distribution (will not go into detail)

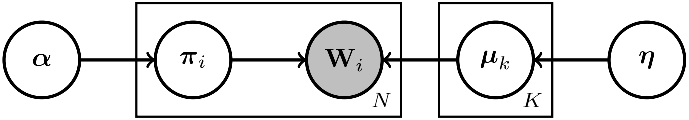
Full model of the DGP

More complex model of the DGP:

1. Sample each document i's cluster, $\boldsymbol{\pi}_i \sim \text{Multinomial}(1, \boldsymbol{\alpha})$
2. Sample each cluster k's "word usage", $\boldsymbol{\mu}_k \sim \text{Dirichlet}(\boldsymbol{\eta})$
3. Create a matrix $\boldsymbol{\mu}$ by staking each $\boldsymbol{\mu}_k$ as columns of $\boldsymbol{\mu}$
4. Using $\boldsymbol{\mu}$ and $\boldsymbol{\pi}_i$, draw each document i:

$$\mathbf{W}_i | \boldsymbol{\mu}, \boldsymbol{\pi}_i \sim \text{Multinomial}(M_i, \boldsymbol{\mu}\boldsymbol{\pi}_i)$$

Graphically:



Simulating the DGP in Python

```python
import numpy as np
rng = np.random.default_rng(2026) # set seed
```

```python
# Draw word usage
eta = [1, 1, 1]                     # DFM with 3 features
mu = rng.dirichlet(eta, size=2)     # make mu_k's for 2 clusters
mu
```

```
array([[0.08072903, 0.68662324, 0.23264773],
       [0.25118333, 0.35784279, 0.39097388]])
```

```python
## Draw cluster for a document
alpha = [0.25, 0.75]                # probs. for 2 clusters
pi = rng.multinomial(1, alpha)      # assign doc to cluster
pi                                  # doc assigned to cluster 1
```

```
array([1, 0])
```

```python
# Generate the document given pi and mu
rng.multinomial(10, pi @ mu)        # assume doc has 10 tokens
```

```
array([0, 7, 3])
```

Mixture modelling

This is a **mixture model** (specifically a **mixture of multinomials**)

In mixture models, the probability of the observed data (e.g., $\mathbf{W}_i$) is a *mixture* of conditional probabilities

Our model is a mixture model because it's a mixture of cluster-specific probabilities:

$$p(\mathbf{W}_i) = \sum_{k=1}^{K} \alpha_k \underbrace{p(\mathbf{W}_i | \boldsymbol{\mu}_k)}_{\text{multinomial}}$$

- ➜ Notice: $p(\mathbf{W}_i)$ is a mixture of multinomials since it is a convex combination of multinomials (each $p(\mathbf{W}_i | \boldsymbol{\mu}_k)$)

There are other mixture models used for clustering

Estimating a mixture model

Mixture models cannot be solved in closed form

Instead, we need to use iterative algorithms

- ➜ Won't cover details in MY459
- ➜ Keep in mind: no pen-and-paper way to calculate

Estimating a mixture model

When you estimate, what do you get?

1. $\widehat{\boldsymbol{\pi}}$: a $K \times N$ **cluster-assignment matrix**
   - Each column $\widehat{\boldsymbol{\pi}}_i$ gives the cluster assignment probabilities for each document i
   - This is soft clustering since $\widehat{\boldsymbol{\pi}}_i$ is not one-hot
   - Usually: hard clusters are obtained by picking the top probability: $\arg\max_k \hat{\pi}_{ik}$
2. $\widehat{\boldsymbol{\mu}}$: a $J \times K$ **cluster-feature matrix**
   - Each column $\widehat{\boldsymbol{\mu}}_k$ gives the distribution over features in cluster k
   - This is the equivalent of the "centroid" above

Note: GRS uses the indexing above, but `sklearn` usually reverses this, e.g. reporting a $\widehat{\boldsymbol{\mu}}$ that is $K \times J$ — **just pay attention**!

Topic modelling

Clustering versus topic modelling

Clustering: assign documents to clusters

- ➜ Even mixture modelling is conceptualised this way (but allows for uncertainty)

But what if a document *genuinely* belongs to multiple clusters?

- ➜ For example: a document covers a mix of topics

This is what **topic modelling** explores

Latent Dirichlet Allocation (LDA)

The standard workhorse topic model is **Latent Dirichlet Allocation (LDA)** from Blei, Ng and Jordan (2003)

We will get into the formal details next, but the basic idea behind the model of the DGP:

- There are K topics covered in a corpus (where K is assumed)
- Word usage varies by topic—different topics use words differently
- Each token inside a document relates to a specific topic
- Each document is "about" a mix of topics, corresponding to the proportion of token topics in the document

LDA is a (more complicated) mixture model

The LDA model

Quick reminder about indexing:

- Documents: $i \in \{1, ..., N\}$
- Features of a vocabulary: $j \in \{1, ..., J\}$
- Topics: $k \in \{1, ..., K\}$
- Within-document token "slot": $m \in \{1, ..., M_i\}$

Recall: we used m last week for language models, e.g.:

- String: `I like to eat apples.`
- Preprocessed tokens: `["like", "eat", "apples"]`:
  - "like" is in slot $m = 1$
  - "eat" is in slot $m = 2$
  - "apple" is in slot $m = 3$
- Note: *not* using zero-based indexing here (unlike Python)

The LDA model

Let's first look at the generative model in words:

1. **What topics are discussed in documents?**
   - ➜ For each document i, draw a K-length vector of topic shares, $\boldsymbol{\pi}_i$, from a Dirichlet distribution with (K-length) parameter $\boldsymbol{\alpha}$
2. **How are words used in topics?**
   - ➜ For each topic k, draw a J-length vector of token probabilities, $\boldsymbol{\mu}_k$, from a Dirichlet distribution with (J-length) parameter $\boldsymbol{\eta}$
     - ➜ Let $\boldsymbol{\mu} = [\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_K]$ be a $J \times K$ matrix of the $\boldsymbol{\mu}_k$ vectors

The LDA model

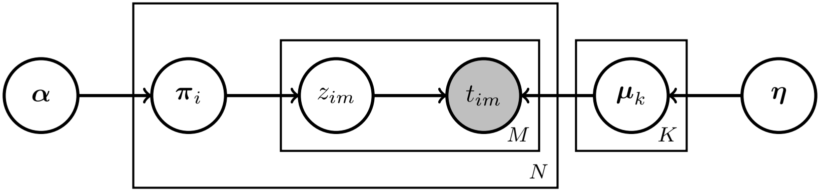
3. Fill each document with words

- For each document i, and each token slot m, draw a topic indicator $z_{im} \sim \text{Categorical}(\boldsymbol{\pi}_i)$, where $z_{im} \in \{1, \dots, K\}$ indicates the token's assigned topic

- For each document i, and each token slot m, draw a token $t_{im} \sim \text{Categorical}(\boldsymbol{\mu}_{z_{im}})$, where $t_{im} \in \{1, \dots, J\}$ indicates the chosen token from the vocabulary

- For each document i and each feature j, calculate that document's term frequency $W_{ij} = \sum_{m=1}^{M_i} \mathbf{1}\{t_{im} = j\}$, where $\mathbf{1}\{\cdot\}$ is the indicator function

The LDA model



Note: unlike before, we now model individual token selection within a document

➜ But still using aggressive (unigram) Markovian assumption, since each token is drawn independently ignoring prior tokens

The LDA model

You can write the LDA model mathematically as:

$$
\begin{aligned}
\boldsymbol{\pi}_i &\sim \mathrm{Dirichlet}(\boldsymbol{\alpha}) \\
\boldsymbol{\mu}_k &\sim \mathrm{Dirichlet}(\boldsymbol{\eta}) \\
z_{im}|\boldsymbol{\pi}_i &\sim \mathrm{Categorical}(\boldsymbol{\pi}_i) \\
t_{im}|z_{im}, \boldsymbol{\mu} &\sim \mathrm{Categorical}(\boldsymbol{\mu}_{z_{im}})
\end{aligned}
$$

Then $\mathbf{W}_i$ can be calculated directly from the sampled t_{im}s

Note: the original paper uses different symbols

- E.g., it uses β instead of μ
- We'll try to stick to common notation that emphasises commonalities among methods
- Mostly aligned with GRS, but not completely

Estimation from a DFM

You just saw how the model generates data *in theory*

➜ Useful for understanding model assumptions about the DGP

But of course: we already have data and want to discover the topics of each token in each document

➜ We *assume* this model of the DGP, and then use the data to estimate

Since it's a mixture model, exact estimation of LDA is intractable

➜ We will use tools in `sklearn` to do this
➜ Again, keep in mind: no pen-and-paper way to calculate!

Estimation from a DFM

When we estimate LDA, what do we get?

Somewhat imprecisely:

1. Estimated topic-feature distributions $\widehat{\boldsymbol{\mu}} = (\widehat{\boldsymbol{\mu}}_1, ..., \widehat{\boldsymbol{\mu}}_K)$
2. Estimated document-topic distributions $\widehat{\boldsymbol{\pi}}_i$ for all i

Similar to what we get with mixture clustering

Of particular interest: do the estimated $\widehat{\boldsymbol{\mu}}_k$s seem intelligible to the human reader?

Estimation from a DFM

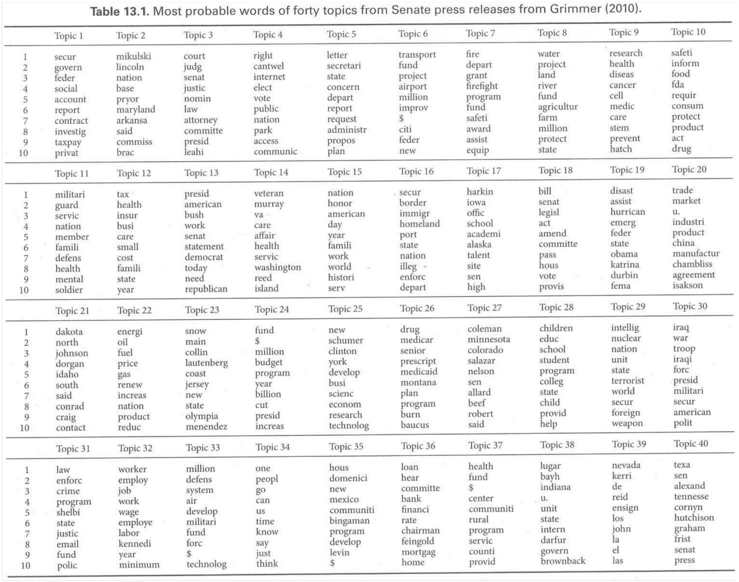


Table 13.1. Most probable words of forty topics from Senate press releases from Grimmer (2010).

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9	Topic 10
1	secur	mikulski	court	right	letter	transport	fire	water	research	safeti
2	govern	lincoln	judg	cantwel	secretari	fund	depart	project	health	inform
3	feder	nation	senat	internet	state	project	grant	land	diseas	food
4	social	base	justic	elect	concern	airport	firefight	river	cancer	fda
5	account	pryor	nomin	vote	depart	million	program	fund	cell	requir
6	report	maryland	law	public	report	improv	fund	agricultur	medic	consum
7	contract	arkansa	attorney	nation	request	$	safeti	farm	care	protect
8	investig	said	committe	park	administr	citi	award	million	stem	product
9	taxpay	commiss	presid	access	propos	feder	assist	protect	prevent	act
10	privat	brac	leahi	communic	plan	new	equip	state	hatch	drug

	Topic 11	Topic 12	Topic 13	Topic 14	Topic 15	Topic 16	Topic 17	Topic 18	Topic 19	Topic 20
1	militari	tax	presid	veteran	nation	secur	harkin	bill	disast	trade
2	guard	health	american	murray	honor	border	iowa	senat	assist	market
3	servic	insur	bush	va	american	immigr	offic	legisl	hurrican	u.
4	nation	busi	work	care	day	homeland	school	act	emerg	industri
5	member	care	senat	affair	year	port	academi	amend	feder	product
6	famili	small	statement	health	famili	state	alaska	committe	state	china
7	defens	cost	democrat	servic	work	nation	talent	pass	obama	manufactur
8	health	famili	today	washington	world	illeg	site	hous	katrina	chambliss
9	mental	state	need	reed	histori	enforc	sen	vote	durbin	agreement
10	soldier	year	republican	island	serv	depart	high	provis	fema	isakson

	Topic 21	Topic 22	Topic 23	Topic 24	Topic 25	Topic 26	Topic 27	Topic 28	Topic 29	Topic 30
1	dakota	energi	snow	fund	new	drug	coleman	children	intellig	iraq
2	north	oil	main	$	schumer	medicar	minnesota	educ	nuclear	war
3	johnson	fuel	collin	million	clinton	senior	colorado	school	nation	troop
4	dorgan	price	lautenberg	budget	york	prescript	salazar	student	unit	iraqi
5	idaho	gas	coast	program	develop	medicaid	nelson	program	state	forc
6	south	renew	jersey	year	busi	montana	sen	colleg	terrorist	presid
7	said	increas	new	billion	scienc	plan	allard	state	world	militari
8	conrad	nation	state	cut	econom	program	beef	child	secur	secur
9	craig	product	olympia	presid	research	burn	robert	provid	foreign	american
10	contact	reduc	menendez	increas	technolog	baucus	said	help	weapon	polit

	Topic 31	Topic 32	Topic 33	Topic 34	Topic 35	Topic 36	Topic 37	Topic 38	Topic 39	Topic 40
1	law	worker	million	one	hous	loan	health	lugar	nevada	texa
2	enforc	employ	defens	peopl	domenici	hear	fund	bayh	kerri	sen
3	crime	job	system	go	new	committe	$	indiana	de	alexand
4	program	work	air	can	mexico	bank	center	u.	reid	tennesse
5	shelbi	wage	develop	us	communiti	financi	communiti	unit	ensign	cornyn
6	state	employe	militari	time	bingaman	rate	rural	state	los	hutchison
7	justic	labor	fund	know	program	chairman	program	intern	john	graham
8	email	kennedi	forc	say	develop	feingold	servic	darfur	la	frist
9	fund	year	$	just	levin	mortgag	counti	govern	el	senat
10	polic	minimum	technolog	think	$	home	provid	brownback	las	press

Extensions

There are many other approaches to topic modelling

Many common approaches add additional structure to LDA

- ➜ More complex models of the DGP
- ➜ Draw information from other data sources (“metadata”), e.g. **structural topic models**
- ➜ Read about this in GSR, especially if you want to use topic models

Can also go in a *non-model* based direction

- ➜ Some recent approaches use neural embeddings to learn richer document representations, then apply clustering
- ➜ Can also have LLMs read texts directly and articulate topics
- ➜ More on this at end of course

Learning about latent structure

Learning about latent structure

Point of methods today: to *discover* new latent structures of documents in a corpus

This means:

1. Clusters/topics are *corpus-dependent* — interpretation of clusters/topics depends on document (and feature) selection
2. The way you interpret clusters/topics is crucial (and potentially misleading)

Evaluation is part art, part science

- ➜ Researchers typically consult a combination of quantitative metrics and human judgements
- ➜ You have to read a lot!

Learning about latent structure

Three separate issues:

1. Choosing K (hard): elbow plots, rules of thumb, post-estimation validation
2. Evaluating your estimates
3. Interpreting your estimates

Quantitative metrics

You can evaluate the quality your clusters/topics using a wide range of quantitative metrics

Two metrics proposed by Mimno et al. (2011) and Roberts et al. (2014)

1. **Exclusivity**: "several clusters [or topics] don't represent the same basic content" (GRS, p. 138)
2. **Cohesion**: "the content of documents in the same group are similar" (GRS, p. 138)

Quantitative metrics

Let $\mathcal{M}_k$ be the set of indices corresponding to features with the top M_k weights for cluster/topic k

- E.g., if $\widehat{\boldsymbol{\mu}}_k = (0.10, 0.56, 0.20, 0.14)$, then the set of top two weights ($M_k = 2$) would be: $\mathcal{M}_k = \{2, 3\}$
- Have to choose M_k—rule of thumb, $M_k = M$ for all k

Using the top weights from $\boldsymbol{\mu}_k$, calculate group k's **exclusivity**

$$\text{Exclusivity}_k = \sum_{m \in \mathcal{M}_k} \frac{\mu_{mk}}{\sum_{l=1}^{K} \mu_{lk}}$$

And **average exclusivity** across all groups is

$$\text{Average Exclusivity} = \frac{1}{K} \sum_{k=1}^{K} \text{Exclusivity}_k$$

Quantitative metrics

Consider a simple example:

$$\widehat{\boldsymbol{\mu}}_1 = (0.18, 0.09, 0.51, 0.22) \qquad \widehat{\boldsymbol{\mu}}_2 = (0.35, 0.33, 0.12, 0.20)$$

If we focus on the top 3 features in each group:

$$\text{Exclusivity}_1 = \frac{0.51}{0.51 + 0.12} + \frac{0.22}{0.22 + 0.20} + \frac{0.18}{0.18 + 0.35} \approx 1.67$$
$$\text{Exclusivity}_2 = \frac{0.35}{0.35 + 0.18} + \frac{0.33}{0.33 + 0.09} + \frac{0.20}{0.20 + 0.22} \approx 1.92$$

And the average exclusivity is:

$$\text{Average Exclusivity} = \frac{1}{2}(1.67 + 1.92) \approx 1.80$$

Quantitative metrics

For each k, create an $M_k \times M_k$ co-occurrence matrix $\mathbf{D}_k$ counting the number of documents where the top M_k words co-occur in k

➜ Note: diagonal gives each term's document frequency

$$\mathbf{D}_k = \begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1M_k} \\ d_{21} & d_{22} & \cdots & d_{2M_k} \\ \vdots & \vdots & \ddots & \vdots \\ d_{M_k1} & d_{M_k2} & \cdots & d_{M_kM_k} \end{bmatrix}$$

Calculate with a small $\varepsilon > 0$ (to avoid $\log(0)$):

$$\text{Cohesion}_k = \sum_{m=1}^{M_k} \sum_{l=1}^{m-1} \ln \left(\frac{d_{lm} + \varepsilon}{d_{ll}} \right)$$

Quantitative metrics

Back to simple example, assume these co-occurrence matrices for the top three features in k:

$$\mathbf{D}_1 = \begin{bmatrix} 10 & 3 & 2 \\ 3 & 8 & 4 \\ 2 & 4 & 6 \end{bmatrix} \qquad \mathbf{D}_2 = \begin{bmatrix} 7 & 2 & 1 \\ 2 & 9 & 3 \\ 1 & 3 & 5 \end{bmatrix}$$

We can calculate cohesion for each group ($\varepsilon = 0.1$):

$$\text{Cohesion}_1 = \ln\left(\frac{3+0.1}{10}\right) + \ln\left(\frac{2+0.1}{10}\right) + \ln\left(\frac{4+0.1}{8}\right) \approx -3.40$$
$$\text{Cohesion}_2 = \ln\left(\frac{2+0.1}{7}\right) + \ln\left(\frac{1+0.1}{7}\right) + \ln\left(\frac{3+0.1}{9}\right) \approx -4.12$$

Average cohesion: $\frac{1}{2}(-3.40 - 4.12) = -3.76$

Quantitative metrics

Note: you cannot use these to *choose* K!

Increasing K tends to inflate exclusivity and cohesion, even if the clustering is not substantively better

- Why? More clusters $\rightarrow$ smaller, more specialised groups
- Similar to how adding predictors inflates R^2

But still useful for knowing:

- whether different clustering methods are better or worse (fixing K)
- whether different initialisations (e.g., in k-means) yield better or worse clusters

Labelling clusters and topics

One of the most controversial parts of clustering and topic models is **labelling**

Analyst takes a statistically-created contraption and attempts to make it meaningful with concise labels

This is necessary: the consumer/reader needs an interpretation

Can also be misleading — people disagree on how to interpret texts

Unfortunately, that's life

➜ But we can try to be more principled...

Guiding principles adapted from GRS

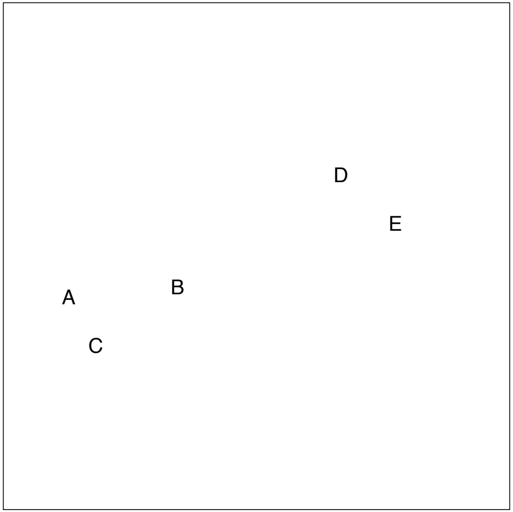
1. Choose number of clusters *carefully*, *thoughtfully* and *transparently* as this will affect how you interpret the resulting clusters
2. Carefully read a random selection of documents in each cluster you estimate (between 10 and 30); take notes and synthesise notes into a corresponding label that matches content of documents
3. Have multiple people provide labels and compare; run experiments to see if subjects broadly agree with your interpretations
4. Provide list of most "distinctive" words of each cluster (if possible)
5. Check if your cluster labels/interpretations hold up after using different clustering approaches
6. Always provide all the details of your process in **replication materials**

Extra slides

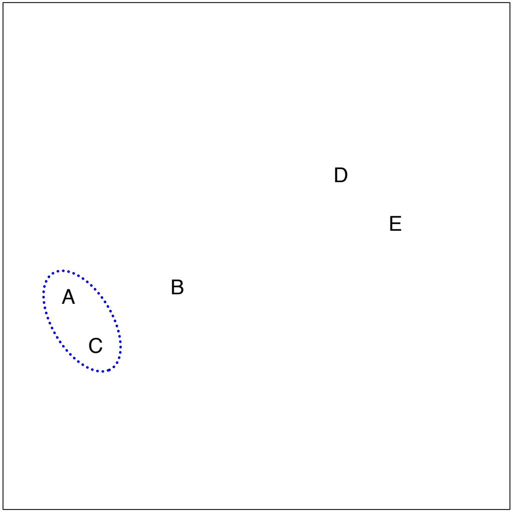
Algorithm details

1. Start with each document as its own "cluster" (N clusters)
2. Calculate pairwise distances between each cluster
3. Find smallest distance between two different clusters, and merge them into a new cluster
4. Recalculate pairwise distances between $N - 1$ clusters
   - ➜ Several options for determining the location of a cluster, e.g. using centroids from above
5. Repeat 3 and 4 until all items are in a one single cluster

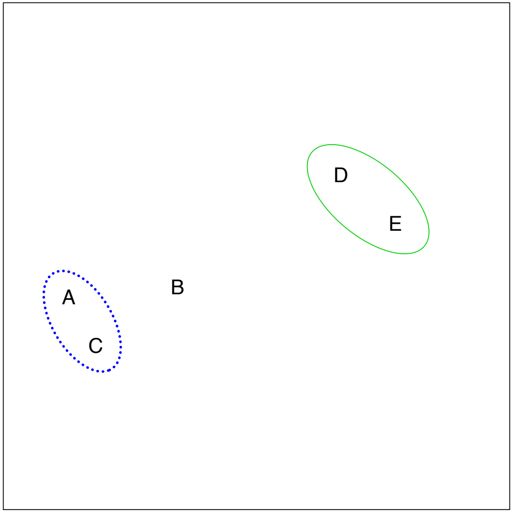
Hierarchical clustering illustrated



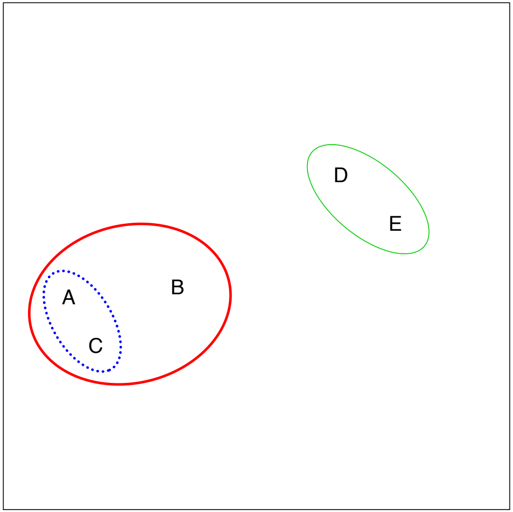
Hierarchical clustering illustrated



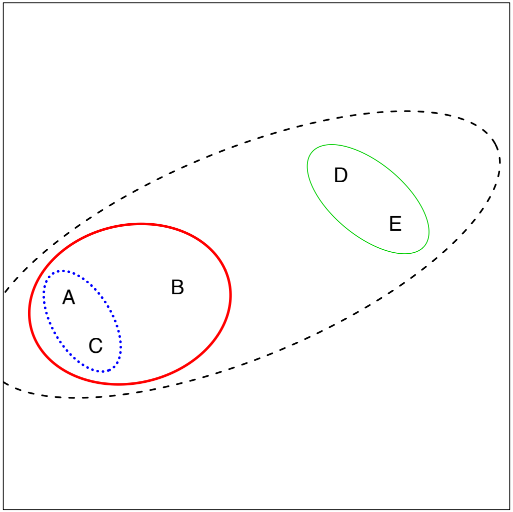
Hierarchical clustering illustrated



Hierarchical clustering illustrated

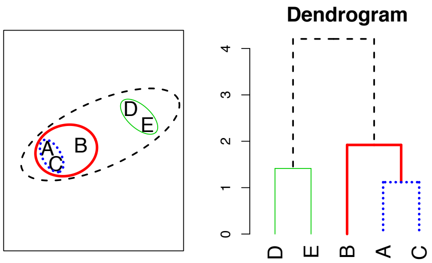


Hierarchical clustering illustrated



Hierarchical clustering output

After you have finished this, you can create a tree-like visualisation called a **dendrogram**



You determine K by "cutting" the tree where you want

Note: there isn't a unique *ordering* of the leaves of a dendrogram

➜ Making it look nice and interpretable can be tricky

Looking at the clusters

Recall our Trump tweets data

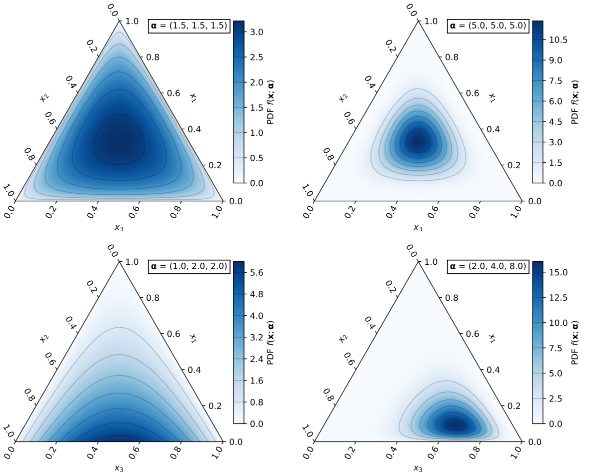
➜ Pre-processed, trimmed, weighted and normalised

If we perform hierarchical clustering with $K = 5$, we get:

	cluster_size	top_features
0	0.851267	job, countri, year, tax, peopl, border
1	0.102690	fake, news, media, russia, fbi, witch
2	0.031298	@whitehouse, honor, welcom, prime, minist, today
3	0.003880	thank, amend, pompeo, resist, merit, substanti
4	0.010864	america, make, togeth, safe, govern, continu

Dirichlet distribution with three categories

Example where $K = 3$ and thus, $\boldsymbol{\alpha}$ has three elements



: Source: https://en.wikipedia.org/wiki/Dirichlet_distribution

Simulating the LDA DGP in Python

Let's set up a simple simulation

```
# Set up steps
import numpy as np                          # load numpy
rng = np.random.default_rng(239)            # create a seed
```

```
# What does our data look like?
vocabulary = ["word1", "word2", "word3"]
J = len(vocabulary)                         # how many features?
N = 100                                     # how many documents?
M = [5] * N                                 # very small documents
```

```
# What do we assume about # of topics?
K = 2                                       # how many topics?
```

Simulating the LDA DGP in Python

First, we draw $\boldsymbol{\pi}_i$ for all i and $\boldsymbol{\mu}_k$ for all k

```
## Draw topic mix
alpha = [0.5] * K     # encourages dominant topics per document
pi = rng.dirichlet(alpha, size=N)
pi.shape
```

```
(100, 2)
```

```
# Draw word usage for (all) topics
eta   = [0.1] * J     # encourages sparse word distributions
mu = rng.dirichlet(eta, size=K)
mu.shape
```

```
(2, 3)
```

Simulating the LDA DGP in Python

Next, we fill tokens for a specific document

➜ Note: we could do this for all documents

```python
# Define an M_i x J matrix to collect tokens
t_1 = np.zeros((M[0], J), dtype=int)
# Iterate over token slots and randomly generate topics and tokens
for m in range(M[0]):
    ## 1. Assign topics to tokens
    z_1m = rng.multinomial(1, pi[0], 1)
    ## 2. Draw tokens using topics
    t_1m = rng.multinomial(1, z_1m @ mu, 1)
    ## 3. Stack into t_1
    t_1[m] = t_1m
# Now, construct row of DFM by summing across t_1 rows
W_1 = t_1.sum(axis=0)
W_1
```

```
array([1, 4, 0])
```