Day 5: Unsupervised Learning

Kenneth Benoit

Data Mining and Statistical Learning

March 16, 2015

Unsupervised "learning": scaling distance

- Features are treated as a quantitative matrix of features
  - (standardized) variables
  - (normalized) word feature counts, in text
- Different possible definitions of *distance*
  - see for instance `summary(pr_DB)` from `proxy` library
- Works on any quantitative matrix of features

Distance measures

```
library(proxy, warn.conflicts = FALSE, quietly = TRUE)
summary(pr_DB)

## * Similarity measures:
## Braun-Blanquet, Chi-squared, correlation, cosine, Cramer, Dice,
## eJaccard, Fager, Faith, Gower, Hamman, Jaccard, Kulczynski1,
## Kulczynski2, Michael, Mountford, Mozley, Ochiai, Pearson, Phi,
## Phi-squared, Russel, simple matching, Simpson, Stiles, Tanimoto,
## Tschuprow, Yule, Yule2
##
## * Distance measures:
## Bhjattacharyya, Bray, Canberra, Chord, divergence, Euclidean,
## fJaccard, Geodesic, Hellinger, Kullback, Levenshtein, Mahalanobis,
## Manhattan, Minkowski, Podani, Soergel, supremum, Wave, Whittaker
```

Parametric v. non-parametric methods

- Parametric methods model feature occurrence according to some stochastic distribution, typically in the form of a measurement model
  - for instance, model words as a multi-level Bernoulli distribution, or a Poisson distribution
  - feature effects and "positional" effects are unobserved parameters to be estimated
- Non-parametric methods typically based on the Singular Value Decomposition of a matrix
  - principal components analysis
  - correspondence analysis
  - other (multi)dimensional scaling methods

Example: text, representing documents as vectors

- The idea is that (weighted) features form a vector for each document, and that these vectors can be judged using metrics of similarity
- A document's vector for us is simply (for us) the row of the document-feature matrix

Characteristics of similarity measures

Let A and B be any two documents in a set and $d(A, B)$ be the distance between A and B.

1. $d(x, y) \geq 0$ (the distance between any two points must be non-negative)
2. $d(A, B) = 0$ iff $A = B$ (the distance between two documents must be zero if and only if the two objects are identical)
3. $d(A, B) = d(B, A)$ (distance must be symmetric: A to B is the same distance as from B to A)
4. $d(A, C) \leq d(A, B) + d(B, C)$ (the measure must satisfy the triangle inequality)

Euclidean distance

Between document A and B where j indexes their features, where y_{ij} is the value for feature j of document i

- Euclidean distance is based on the Pythagorean theorem
- Formula

$$\sqrt{\sum_{j=1}^{j}(y_{Aj} - y_{Bj})^2} \tag{1}$$

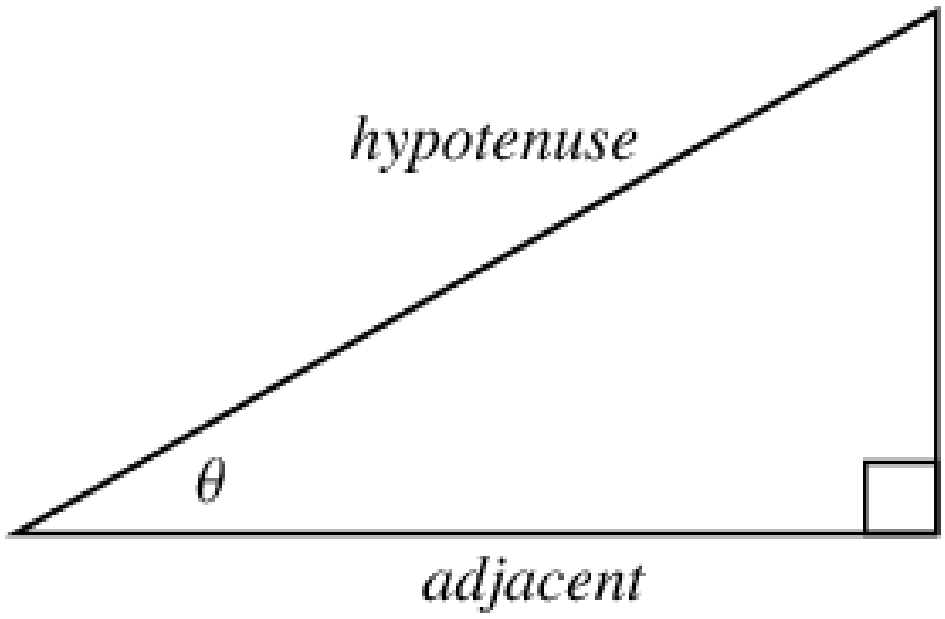
- In vector notation:

$$\|\mathbf{y}_A - \mathbf{y}_B\| \tag{2}$$

- Can be performed for any number of features J (or V as the vocabulary size is sometimes called – the number of columns in of the dfm, same as the number of feature types in the corpus)

A geometric interpretation of "distance"

In a right angled triangle, the cosine of an angle θ or $\cos(\theta)$ is the length of the adjacent side divided by the length of the hypotenuse



We can use the vectors to represent the text location in a V-dimensional vector space and compute the angles between them

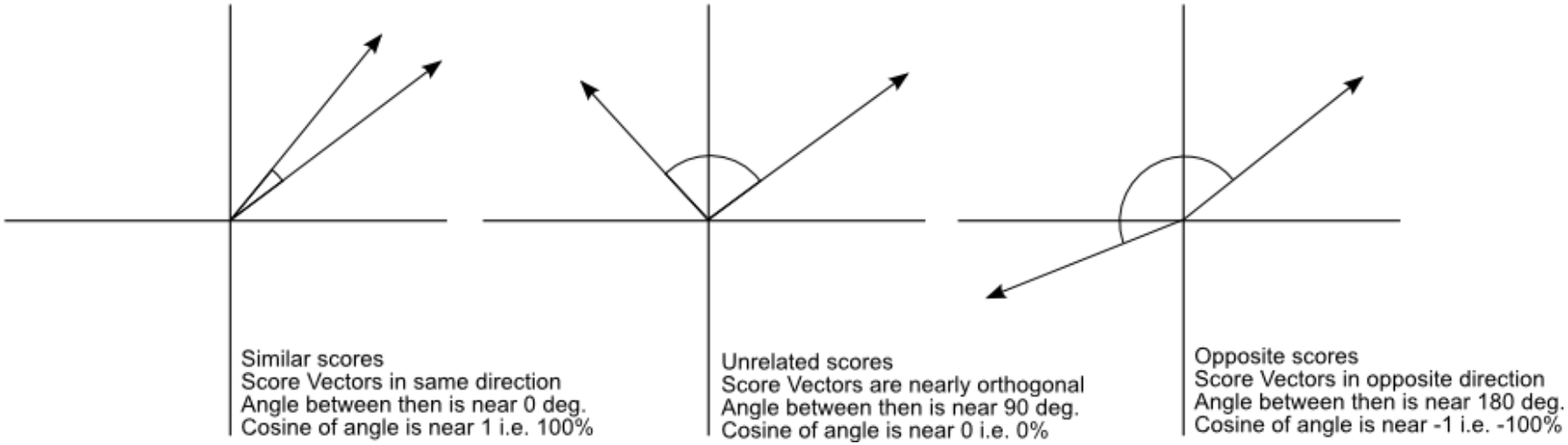
Cosine similarity

- Cosine distance is based on the size of the angle between the vectors
- Formula

$$\frac{\mathbf{y}_A \cdot \mathbf{y}_B}{\|\mathbf{y}_A\|\|\mathbf{y}_B\|} \tag{3}$$

- The $\cdot$ operator is the dot product, or $\sum_j y_{Aj} y_{Bj}$
- The $\|\mathbf{y}_A\|$ is the vector norm of the (vector of) features vector $\mathbf{y}$ for document A, such that $\|\mathbf{y}_A\| = \sqrt{\sum_j y_{Aj}^2}$
- Nice propertyfor text: cosine measure is independent of document length, because it deals only with the angle of the vectors
- Ranges from -1.0 to 1.0 for term frequencies, or 0 to 1.0 for normalized term frequencies (or tf-idf)

Cosine similarity illustrated



Example text

Hurricane Gilbert swept toward the Dominican Republic Sunday , and the Civil Defense alerted its heavily populated south coast to prepare for high **winds**, heavy **rains** and high seas.

The **storm** was approaching from the southeast with sustained **winds** of 75 mph gusting to 92 mph .

"There is no need for alarm," Civil Defense Director Eugenio Cabral said in a television alert shortly before midnight Saturday .

Cabral said residents of the province of Barahona should closely follow **Gilbert** 's movement .

An estimated 100,000 people live in the province, including 70,000 in the city of Barahona , about 125 miles west of Santo Domingo .

Tropical **Storm Gilbert** formed in the eastern Caribbean and strengthened into a **hurricane** Saturday night

The National **Hurricane** Center in Miami reported its position at 2a.m. Sunday at latitude 16.1 north , longitude 67.5 west, about 140 miles south of Ponce, Puerto Rico, and 200 miles southeast of Santo Domingo.

The National Weather Service in San Juan , Puerto Rico , said **Gilbert** was moving westward at 15 mph with a "broad area of cloudiness and heavy weather" rotating around the center of the **storm**.

The weather service issued a flash flood watch for Puerto Rico and the Virgin Islands until at least 6p.m. Sunday.

Strong **winds** associated with the **Gilbert** brought coastal flooding , strong southeast **winds** and up to 12 feet to Puerto Rico 's south coast.

Example text: selected terms

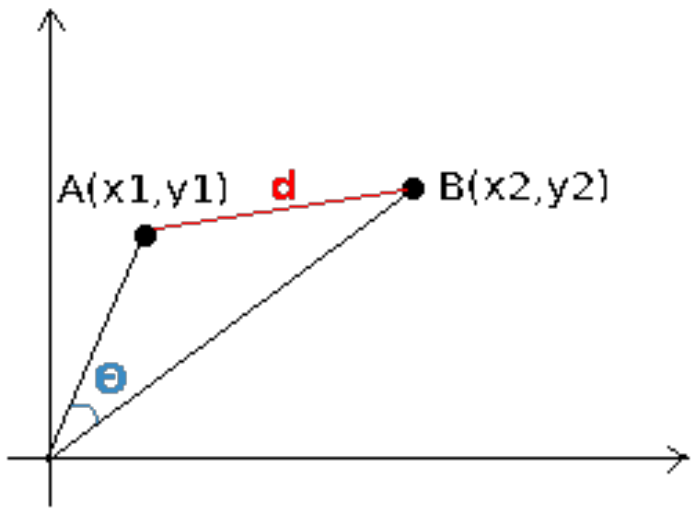
- Document 1
  Gilbert: 3, hurricane: 2, rains: 1, storm: 2, winds: 2
- Document 2
  Gilbert: 2, hurricane: 1, rains: 0, storm: 1, winds: 2

Example text: cosine similarity in R

```
require(quanteda)

## Loading required package:  quanteda

toyDfm <- matrix(c(3,2,1,2,2, 2,1,0,1,2), nrow=2, byrow=TRUE)
colnames(toyDfm) <- c("Gilbert", "hurricane", "rain", "storm", "winds")
rownames(toyDfm) <- c("doc1", "doc2")
toyDfm

##      Gilbert hurricane rain storm winds
## doc1       3         2    1     2     2
## doc2       2         1    0     1     2

simil(toyDfm, "cosine")

##           doc1
## doc2 0.9438798
```

Relationship to Euclidean distance

- Cosine similarity measures the similarity of vectors with respect to the origin
- Euclidean distance measures the distance between particular points of interest along the vector



Jacquard coefficient

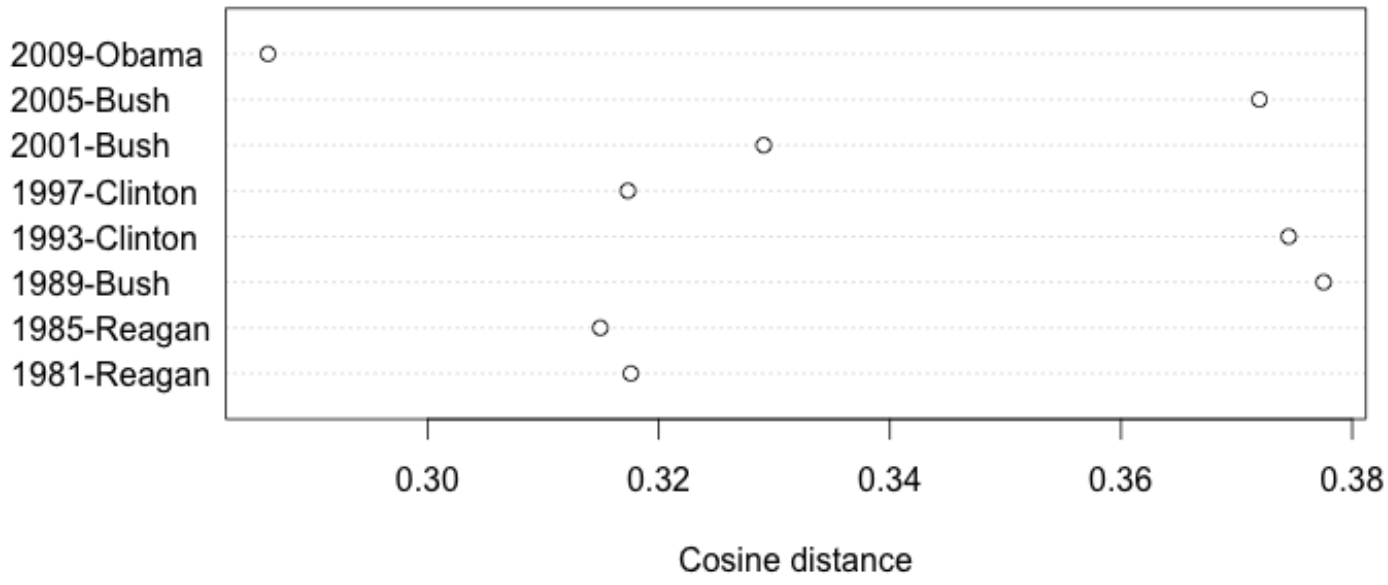
- Similar to the Cosine similarity
- Formula

$$\frac{\mathbf{y}_A \cdot \mathbf{y}_B}{\|\mathbf{y}_A\| + \|\mathbf{y}_B\| - \mathbf{y}_A \cdot \mathbf{y}y_B} \tag{4}$$

- Ranges from 0 to 1.0

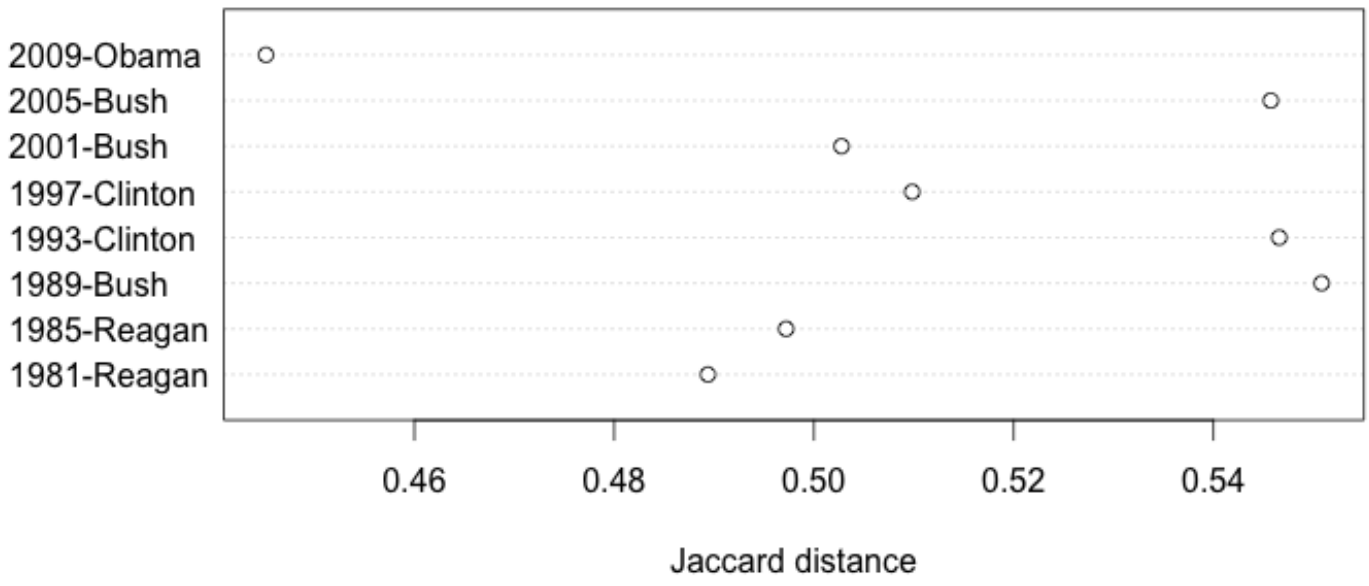
Example: Inaugural speeches, cosine distance to Obama 2014

```
presDfm <- dfm(subset(inaugCorpus, Year>1980),
               ignoredFeatures=stopwords("english", verbose=FALSE),
               stem=TRUE, verbose=FALSE)
obamaDistance <- as.matrix(dist(as.matrix(presDfm), "Cosine"))
dotchart(obamaDistance[1:8,9], xlab="Cosine distance")
```



Example: Jaccard distance to Obama

```
obamaDistance <- as.matrix(dist(presDfm, "eJaccard"))
 ## Error in as.matrix(dist(presDfm, "eJaccard")):  error in evaluating
the argument 'x' in selecting a method for function 'as.matrix':  Error
                    in dist(presDfm, "eJaccard") :
     ##  Can only handle data frames, vectors, matrices, and lists!

dotchart(obamaDistance[1:8,9], xlab="Jaccard distance")
```



Common uses

- Clustering (we will see this shortly)
- Used extensively in information retrieval
- Summmary measures of how far apart two texts are – but be careful exactly how you define "features"
- Some but not many applications in social sciences to measure substantive similarity — scaling models are generally preferred
- Can be used to generalize or represent features in machine learning, by combining features using kernel methods to compute similarities between textual (sub)sequences without extracting the features explicitly (as we have done here)

The idea of "clusters"

- Essentially: groups of items such that inside a cluster they are very similar to each other, but very different from those outside the cluster
- "unsupervised classification": cluster is not to relate features to classes or latent traits, but rather to estimate membership of distinct groups
- groups are given labels through post-estimation interpretation of their elements
- typically used when we do not and never will know the "true" class labels
- issues: how to weight distance is arbitrary
  - which dimensionality? (determined by which features are selected)
  - how to weight distance is arbitrary
  - different metrics for distance

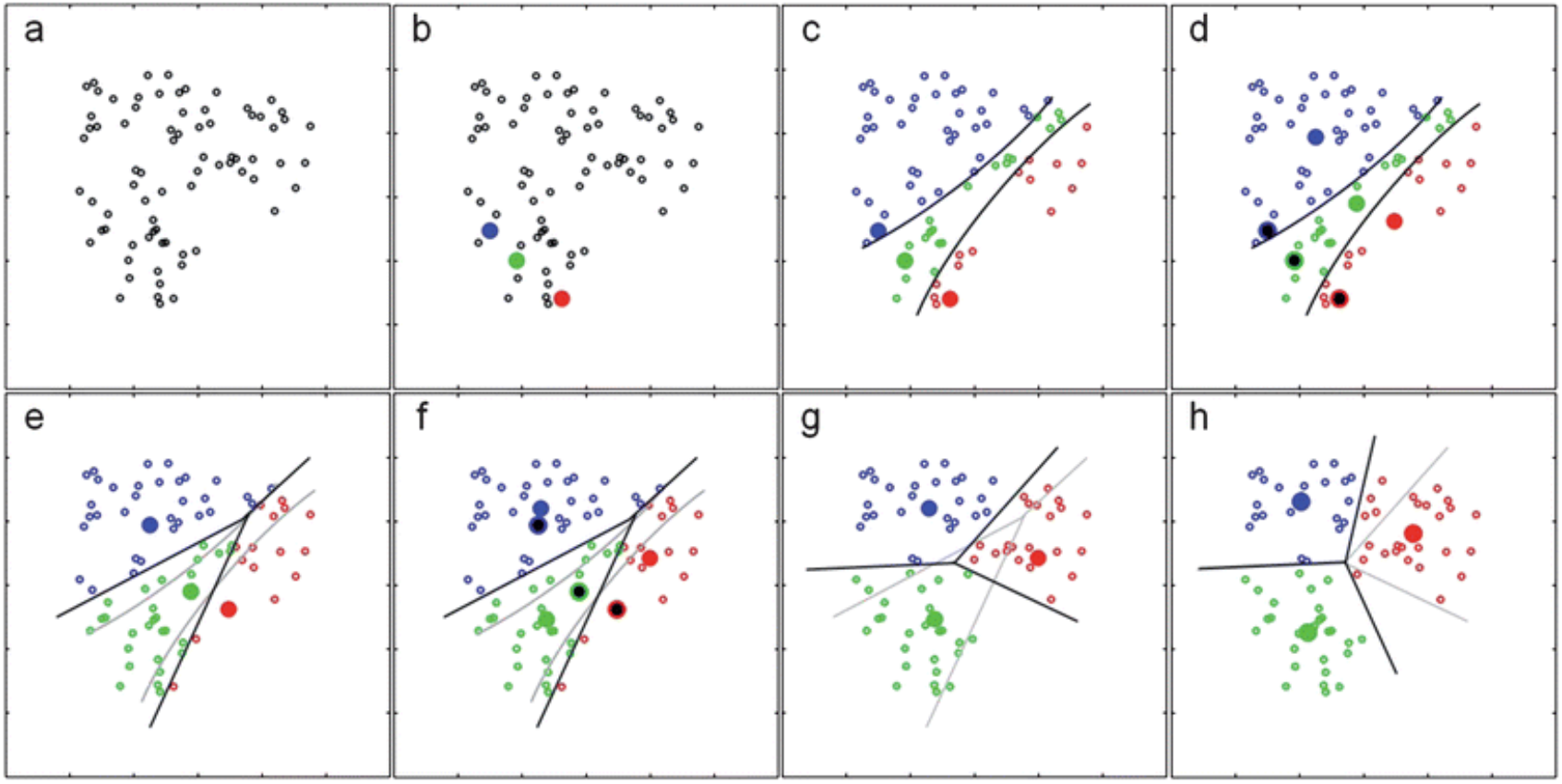
k-means clustering

- Essence: assign each item to one of k clusters, where the goal is to minimized within-cluster difference and maximize between-cluster differences
- Uses random starting positions and iterates until stable
- as with *kNN*, k-means clustering treats feature values as coordinates in a multi-dimensional space
- Advantages
  - simplicity
  - highly flexible
  - efficient
- Disadvantages
  - no fixed rules for determining k
  - uses an element of randomness for starting values

Algorithm details

1. Choose starting values
   - assign random positions to k starting values that will serve as the "cluster centres", known as "centroids" ; or,
   - assign each feature randomly to one of k classes
2. assign each item to the class of the centroid that is "closest"
   - Euclidean distance is most common
   - any others may also be used (Manhattan, Mikowski, Mahalanobis, etc.)
   - (assumes feature vectors have been normalized within item)
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

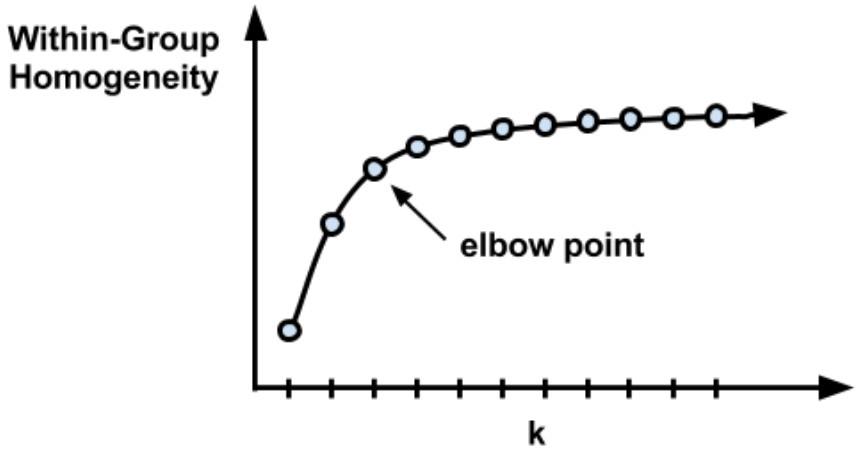


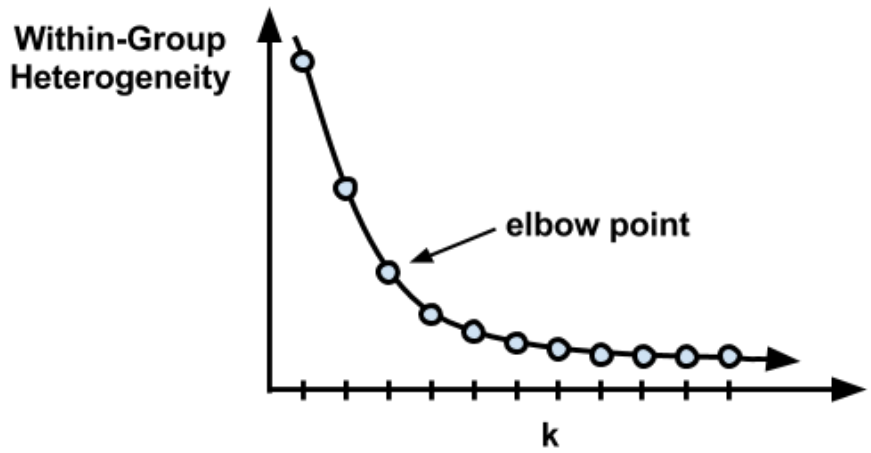
Choosing the appropriate number of clusters

- very often based on prior information about the number of categories sought
  - for example, you need to cluster people in a class into a fixed number of (like-minded) tutorial groups
- a (rough!) guideline: set $k = \sqrt{N/2}$ where N is the number of items to be classified
  - usually too big: setting k to large values will improve within-cluster similarity, but risks *overfitting*

Choosing the appropriate number of clusters

- "elbow plots": fit multiple clusters with different k values, and choose k beyond which are diminishing gains





Choosing the appropriate number of clusters

- "fit" statistics to measure homogeneity within clusters and heterogeneity in between
  - numerous examples exist
- "iterative heuristic fitting"* (IHF) (trying different values and looking at what seems most plausible)

* Warning: This is my (slightly facetious) term only!

Other clustering methods: hierarchical clustering

- *agglomerative*: works from the bottom up to create clusters
- like *k*-means, usually involves *projection*: reducing the features through either selection or projection to a lower-dimensional representation
  1. local projection: reducing features within document
  2. global projection: reducting features across all documents (Schütze and Silverstein, 1997)
  3. SVD methods, such PCA on a normalized feature matrix
  4. usually simple threshold-based truncation is used (keep all but 100 highest frequency or tf-idf terms)
- frequently/always involves weighting (normalizing term frequency, tf-idf)

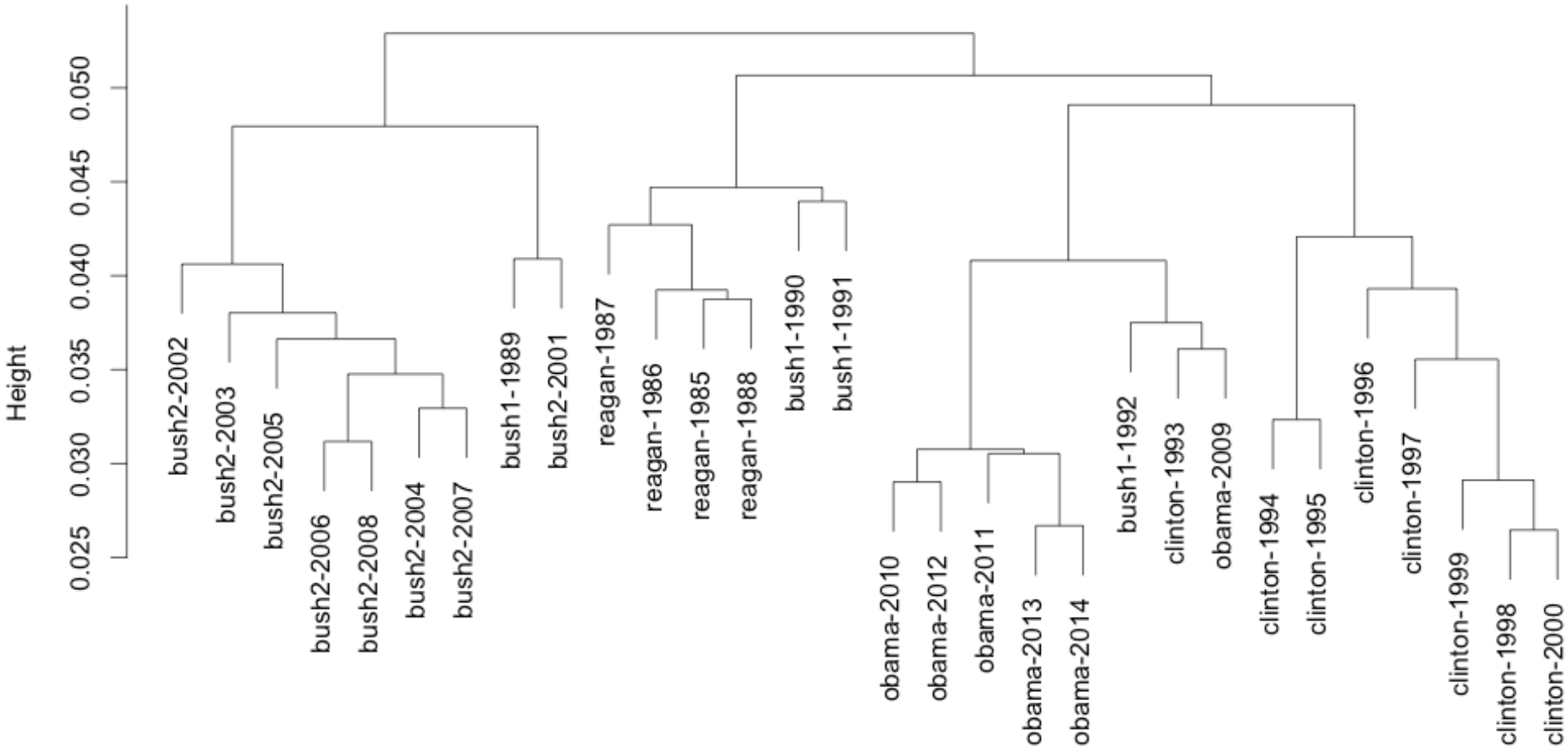
hierarchical clustering algorithm

1. start by considering each item as its own cluster, for n clusters
2. calculate the $N(N-1)/2$ pairwise distances between each of the n clusters, store in a matrix D_0
3. find smallest (off-diagonal) distance in D_0, and merge the items corresponding to the i, j indexes in D_0 into a new "cluster"
4. recalculate distance matrix D_1 with new cluster(s). options for determining the location of a cluster include:
   - centroids (mean)
   - most dissimilar objects
   - Ward's measure(s) based on minimizing variance
5. repeat 3–4 until a stopping condition is reached
   - e.g. all items have been merged into a single cluster
6. to plot the *dendrograms*, need decisions on ordering, since there are $2^{(N-1)}$ possible orderings

Dendrogram: Presidential State of the Union addresses

```
data(SOTUCorpus, package="quantedaData")
presDfm <- dfm(subset(SOTUCorpus, year>1960), verbose=FALSE, stem=TRUE,
               ignoredFeatures=stopwords("english", verbose=FALSE))
presDfm <- trim(presDfm, minCount=5, minDoc=3)

## Features occurring less than 5 times: 4049
## Features occurring in fewer than 3 documents: 3511

# hierarchical clustering - get distances on normalized dfm
presDistMat <- dist(as.matrix(weight(presDfm, "relFreq")))
# hiarchical clustering the distance object
presCluster <- hclust(presDistMat)
# label with document names
presCluster$labels <- docnames(presDfm)
# plot as a dendrogram
plot(presCluster)
```

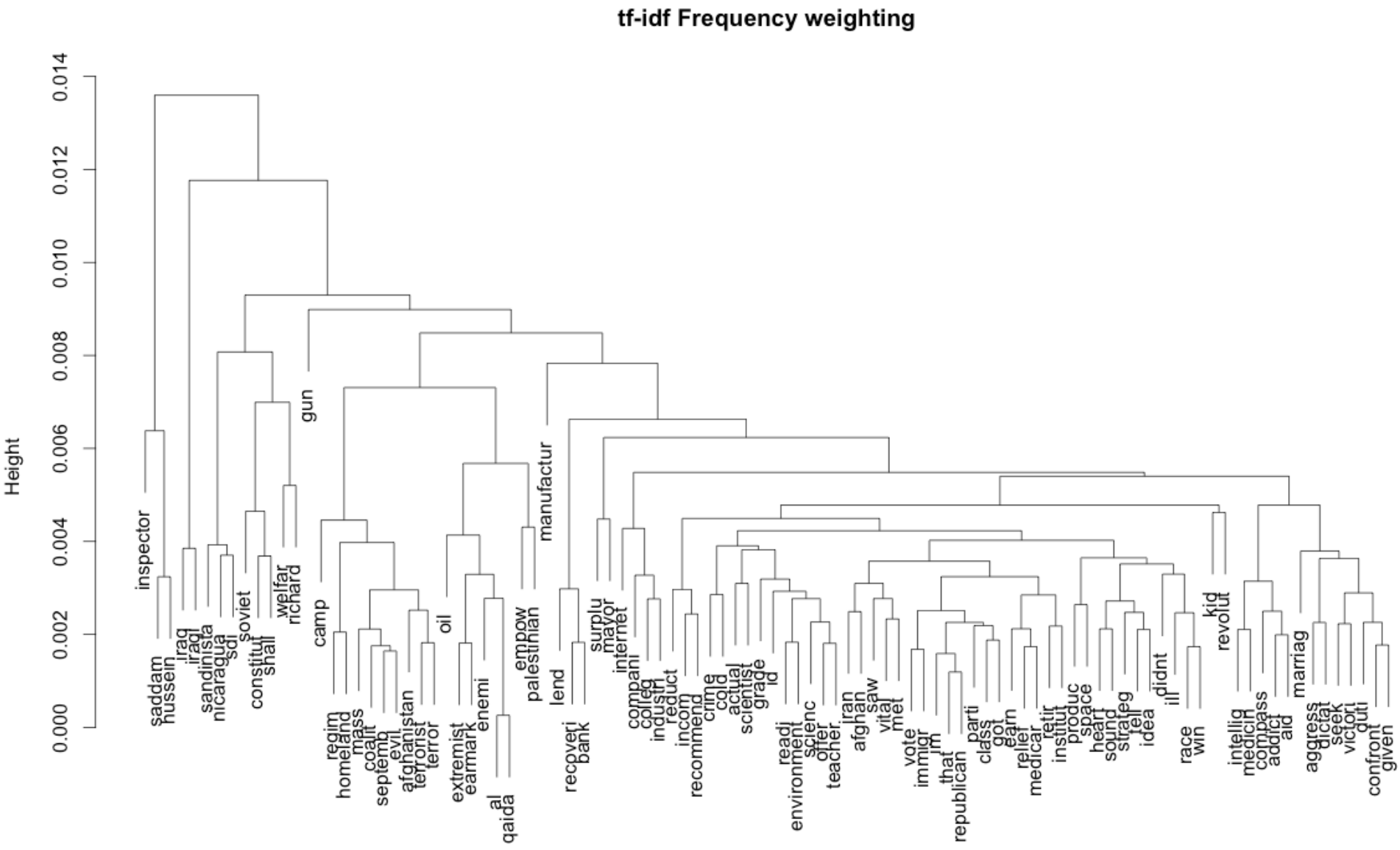
Dendrogram: Presidential State of the Union addresses



Dendrogram: Presidential State of the Union addresses

```
# word dendrogram with tf-idf weighting
wordDfm <- sort(tfidf(presDfm))  # sort in decreasing order of total word freq
wordDfm <- t(wordDfm)[1:100,]  # because transposed
wordDistMat <- dist(wordDfm)
wordCluster <- hclust(wordDistMat)
plot(wordCluster, xlab="", main="tf-idf Frequency weighting")
```

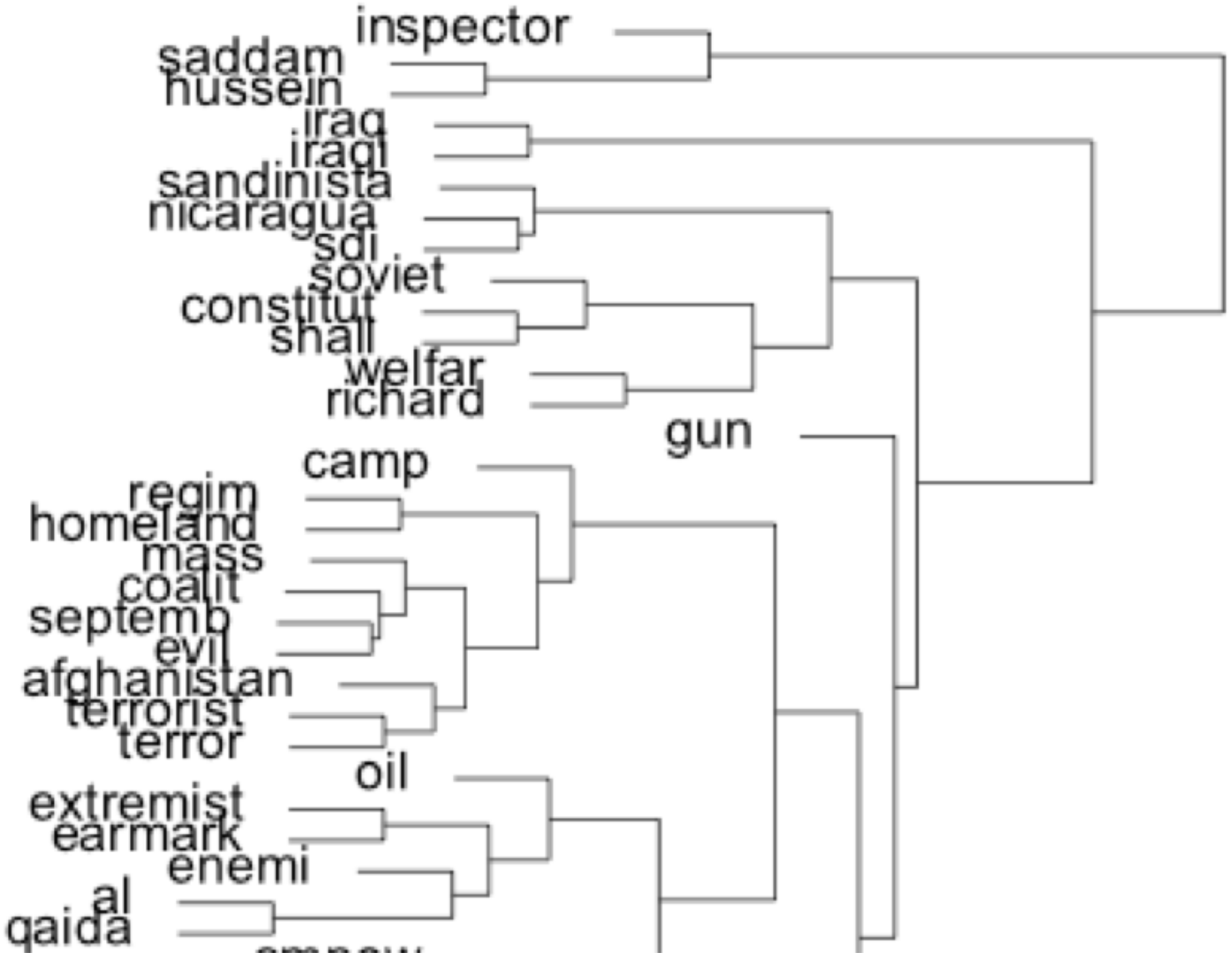
Dendrogram: Presidential State of the Union addresses



pros and cons of hierarchical clustering

- advantages
  - deterministic, unlike k-means
  - no need to decide on k in advance (although can specify as a stopping condition)
  - allows hierarchical relations to be examined (usually through *dendrograms*)
- disadvantages
  - more complex to compute: quadratic in complexity: $O(n^2)$ – whereas k-means has complexity that is $O(n)$
  - the decision about where to create branches and in what order can be somewhat arbitrary, determined by method of declaring the "distance" to already formed clusters
  - for words, tends to identify collocations as base-level clusters (e.g. "saddam" and "hussein")

Dendrogram: Presidential State of the Union addresses



Non-parametric dimensional reduction methods

- Non-parametric methods are algorithmic, involving no "parameters" in the procedure that are estimated
- Hence there is no uncertainty accounting given distributional theory
- Advantage: don't have to make assumptions
- Disadvantages:
  - cannot leverage probability conclusions given distribtional assumptions and statistical theory
  - results highly fit to the data
  - not really assumption-free (if we are honest)

Principal Components Analysis

- For a set of features $X_1, X_2, \ldots, X_p$, typically centred (to have mean 0)
- the first principal component is the normalized linear combination of the features

$$Z_1 = \phi_{11} X_1 + \phi_{21} X_2 + \ldots + \phi_{p1} X_p$$

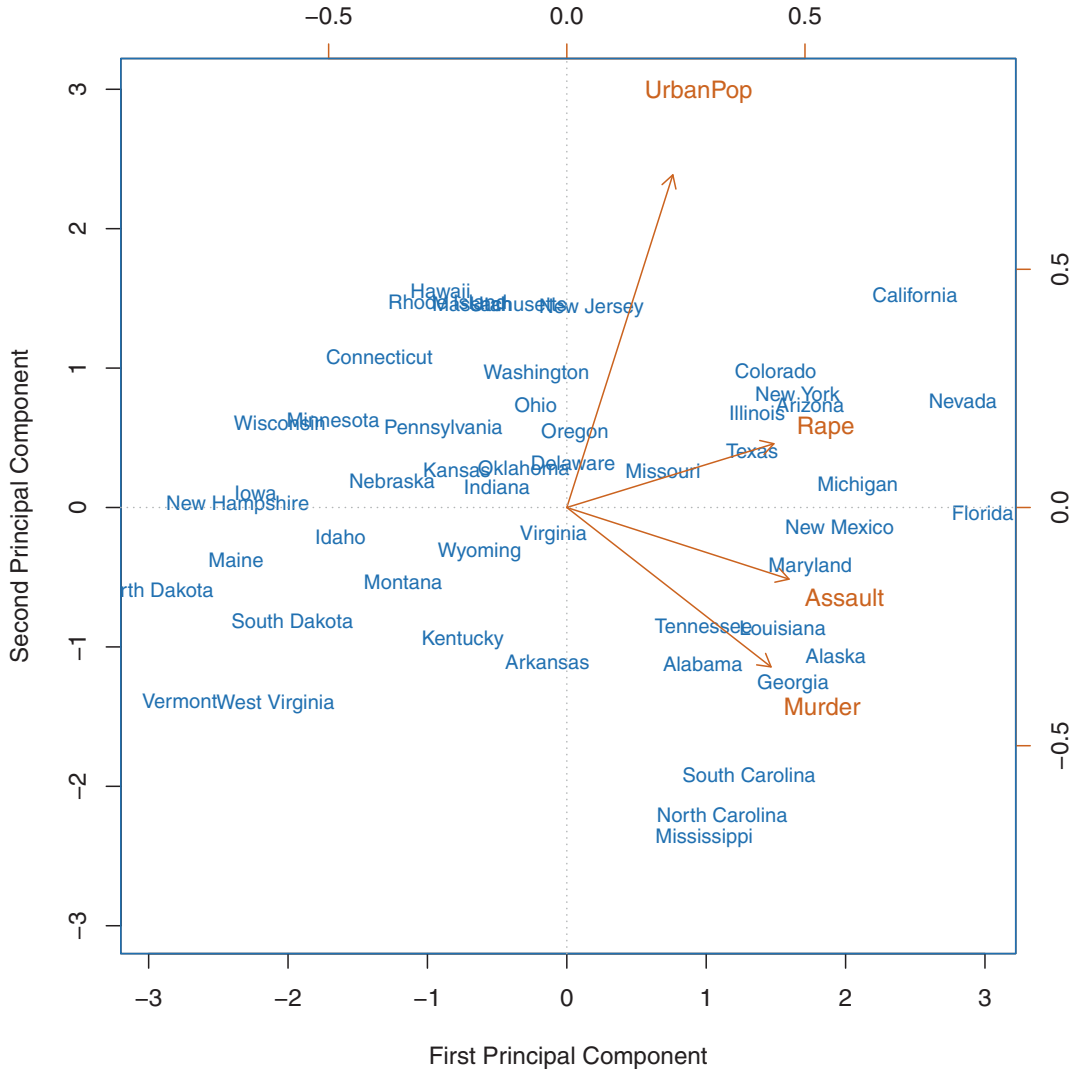
  that has the largest variance
- normalized means that $\sum_{j=1}^{p} \phi_{j1}^2 = 1$
- the elements $\phi_{11}, \ldots, \phi_{p1}$ are the loadings of the first principal component
- the second principal component is the linear combination Z_2 of $X_1, X_2, \ldots, X_p$ that has maximal variance out of all linear combinations that are *uncorrelated* with Z_1

PCA factor loadings example

	P
Murder	0.5358
Assault	0.5831
UrbanPop	0.2781
Rape	0.5434

TABLE 10.1. *The principal componer*
`USArrests` *data. These are also displayed*

PCA factor loadings biplot



PCA projection illustrated

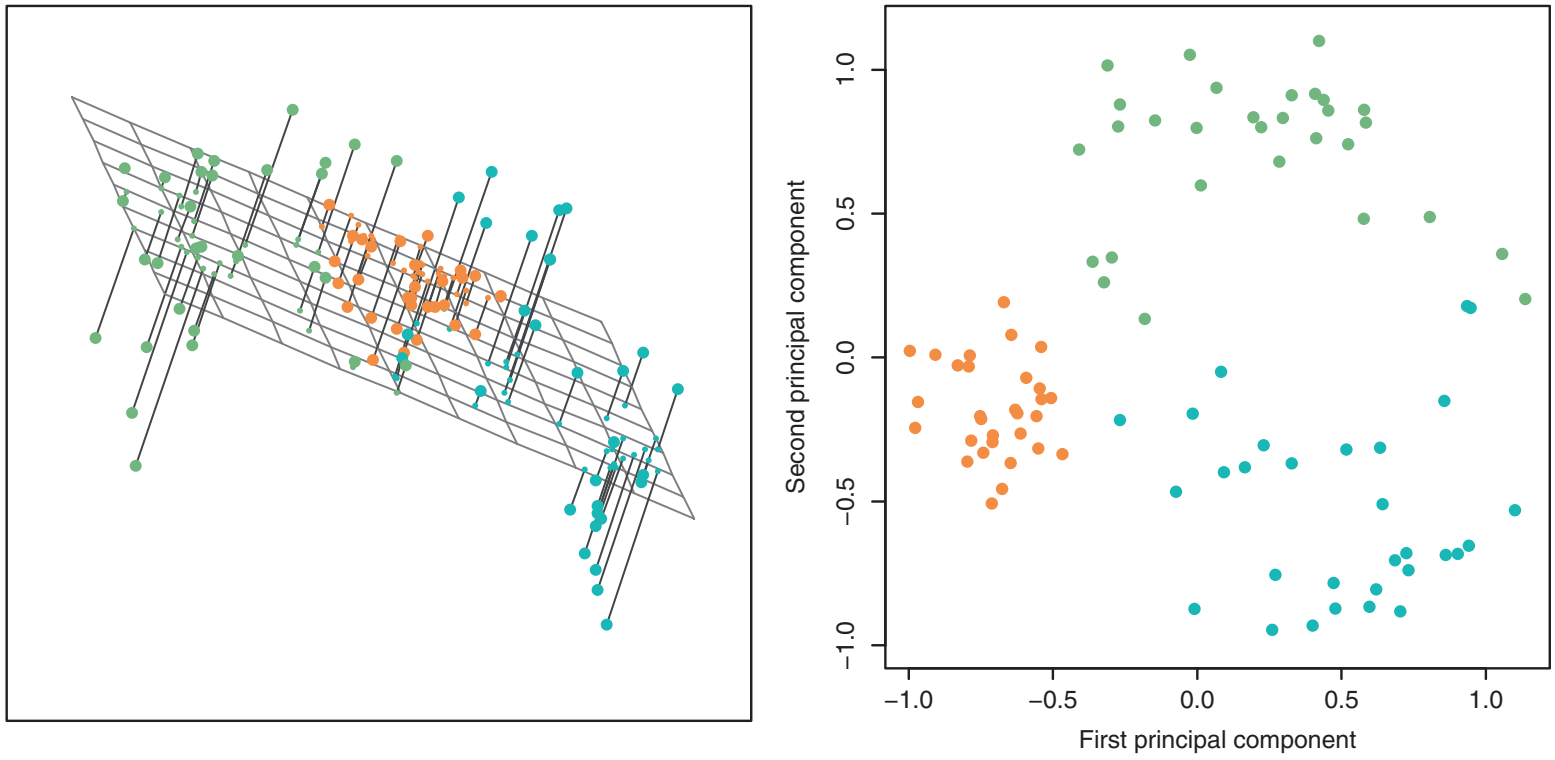


FIGURE 10.2. *Ninety observations simulated in three dimensions.* Left: *the first two principal component directions span the plane that best fits the data. It minimizes the sum of squared distances from each point to the plane.* Right: *the first two principal component score vectors give the coordinates of the projection of the 90 observations onto the plane. The variance in the plane is maximized.*

Correspondence Analysis

- CA is like factor analysis for categorical data
- Following normalization of the marginals, it uses Singular Value Decomposition to reduce the dimensionality of the word-by-text matrix
- This allows projection of the positioning of the words as well as the texts into multi-dimensional space
- The number of dimensions – as in factor analysis – can be decided based on the eigenvalues from the SVD

Singular Value Decomposition

- A matrix $\underset{i \times j}{\mathbf{X}}$ can be represented in a dimensionality equal to its rank k as:

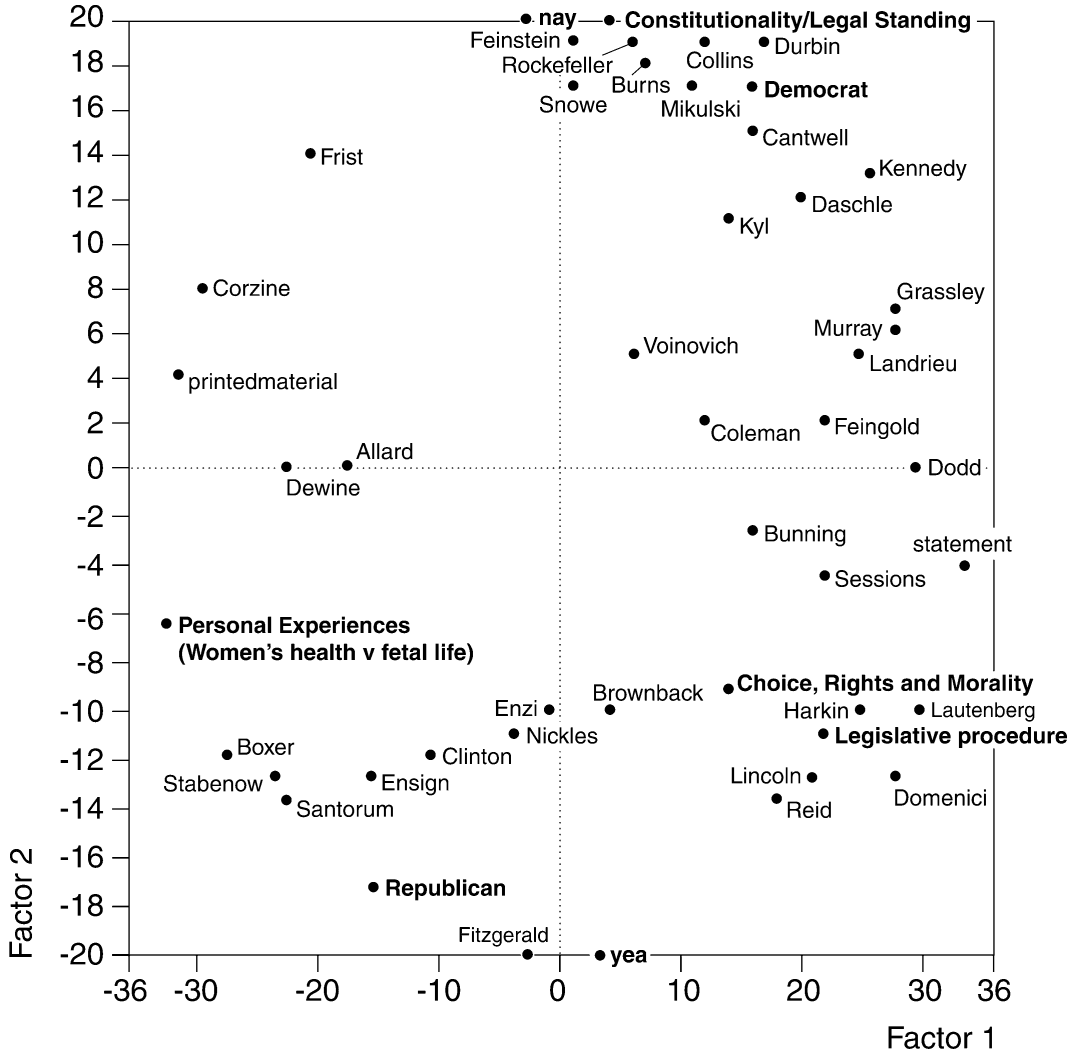
$$\underset{i \times j}{\mathbf{X}} = \underset{i \times k}{\mathbf{U}} \; \underset{k \times k}{\mathbf{d}} \; \underset{j \times k}{\mathbf{V}'} \tag{5}$$

- The **U**, **d**, and **V** matrixes "relocate" the elements of **X** onto new coordinate vectors in n-dimensional Euclidean space
- Row variables of **X** become points on the **U** column coordinates, and the column variables of **X** become points on the **V** column coordinates
- The coordinate vectors are perpendicular (*orthogonal*) to each other and are normalized to unit length

Correspondence Analysis and SVD

- Divide each value of **X** by the geometric mean of the corresponding marginal totals (square root of the product of row and column totals for each cell)
  - Conceptually similar to subtracting out the χ^2 expected cell values from the observed cell values
- Perform an SVD on this transformed matrix
  - This yields singular values **d** (with first always 1.0)
- Rescale the row (**U**) and column (**V**) vectors to obtain canonical scores (rescaled as $U_i \sqrt{f_{..}/f_{i.}}$ and $V_j \sqrt{f_{..}/f_{j.}}$)

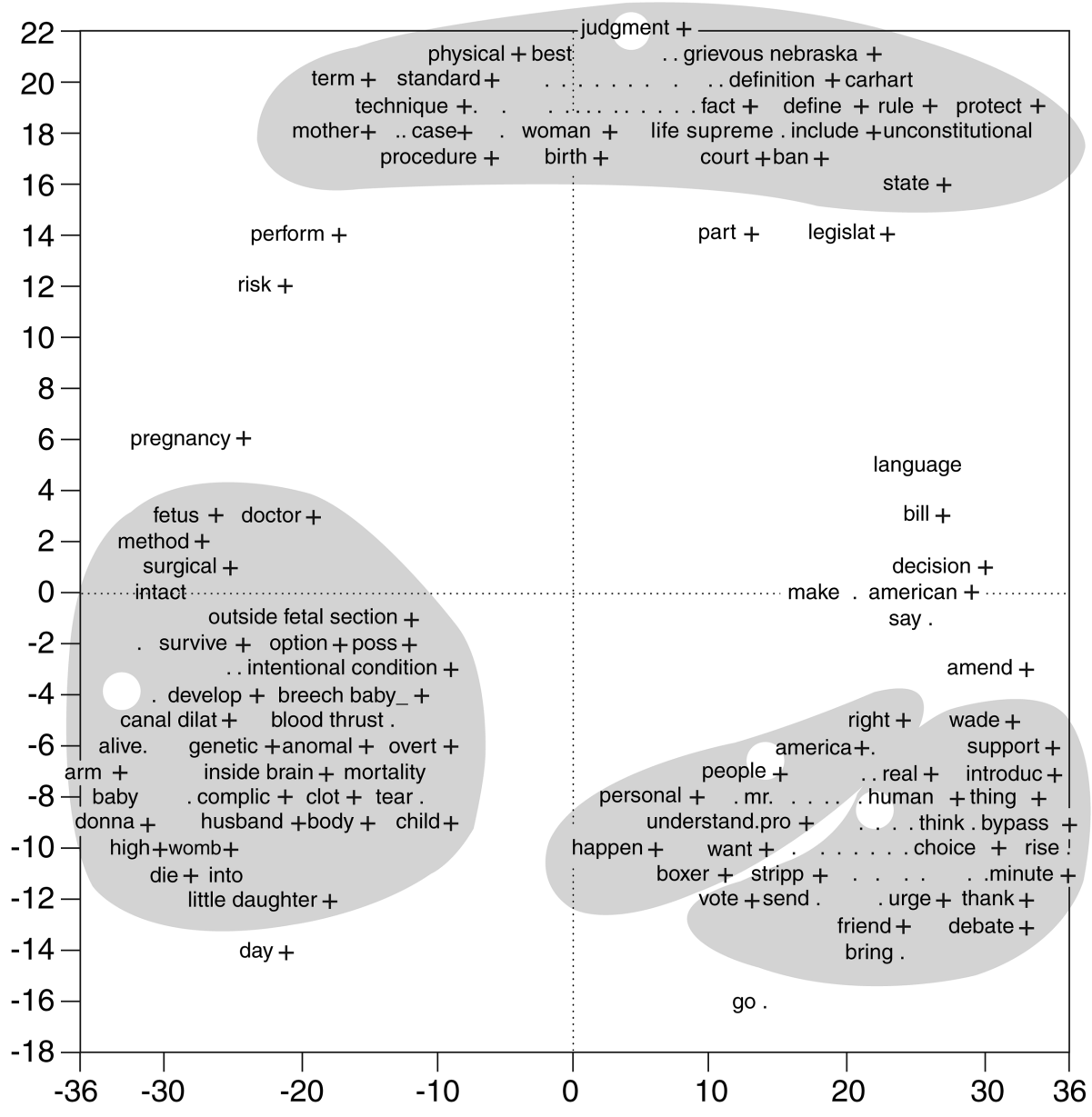
Example: Schonhardt-Bailey (2008) - speakers



	Eigenvalue	% Association	% Cumulative
Factor 1	0.30	44.4	44.4
Factor 2	0.22	32.9	77.3

Fig. 3. Correspondence analysis of classes and tags from Senate debates on Partial-Birth Abortion Ban Act

Example: Schonhardt-Bailey (2008) - words



How to get confidence intervals for CA

- There are problems with bootstrapping: (Milan and Whittaker 2004)
  - rotation of the principal components
  - inversion of singular values
  - reflection in an axis

How to account for uncertainty

- Ignore the problem and hope it will go away
  - SVD-based methods (e.g. correspondence analysis) typically do not present errors
  - and traditionally, point estimates based on other methods have not either

How to account for uncertainty

- Analytical derivatives
  - Using the multinomial formulation of the Poisson model, we can compute a Hessian for the log-likelihood function
  - The standard errors on the θ_i parameters can be computed from the covariance matrix from the log-likelihood estimation (square roots of the diagonal)
  - The covariance matrix is (asymptotically) the inverse of the negative of the Hessian
    (where the negative Hessian is the observed Fisher information matrix, a.ka. the second derivative of the log-likelihood evaluated at the maximum likelihood estimates)
  - Problem: These are *too small*

How to account for uncertainty

- Parametric bootstrapping (Slapin and Proksch, Lewis and Poole)
  Assume the distribution of the parameters, and generate data after drawing new parameters from these distributions.
  Issues:
  - slow
  - relies heavily (twice now) on parametric assumptions
  - requires some choices to be made with respect to data generation in simulations
- Non-parametric bootstrapping
-
- (and yes of course) Posterior sampling from MCMC

How to account for uncertainty

- Non-parametric bootstrapping
  - draw new versions of the texts, refit the model, save the parameters, average over the parameters
  - slow
  - not clear how the texts should be resampled

How to account for uncertainty

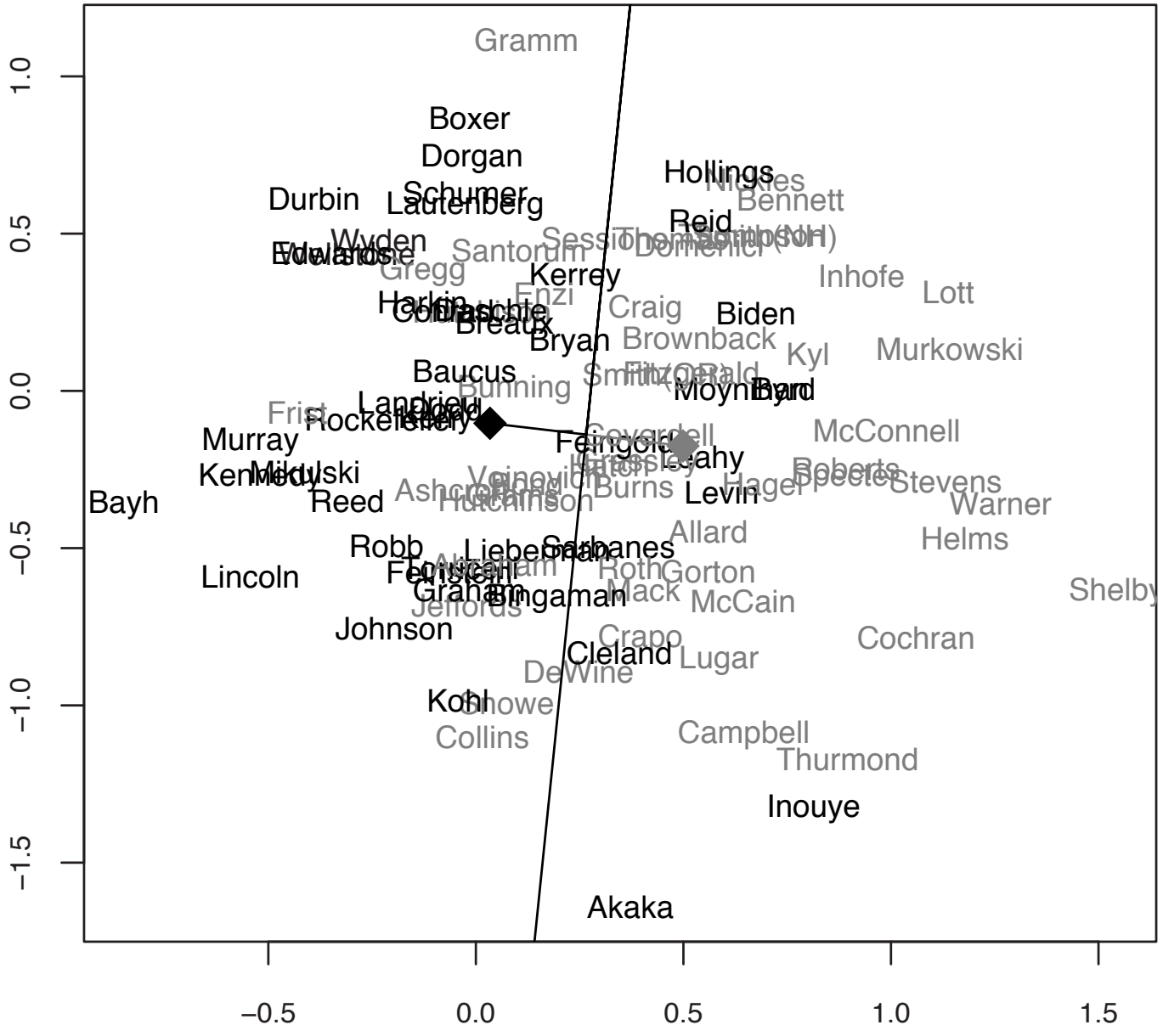
- For MCMC: from the distribution of posterior samples

Dimensions

How infer more than one dimension?
This is two questions:

- How to get two dimensions (for all policy areas) at the same time?
- How to get one dimension for each policy area?

The hazards of ex-post interpretation illustrated



Interpreting scaled dimensions

- In practice can be very subjective, involves interpretation
- Another (better) option: compare them other known descriptive variables
- Hopefully also *validate* the scale results with some human judgments
- This is necessary even for single-dimensional scaling
- And just as applicable for non-parametric methods (e.g. correspondence analysis) as for the Poisson scaling model