Day 6: Clustering

Kenneth Benoit and Slava Mikhaylov

Introduction to Data Science and Big Data Analytics

24 August 2015

Day 6 Outline

Supervised v. unsupervised learning

Scaling distance

Clustering

- k-means clustering
- Hierarchical clustering

Association rules

Supervised v. unsupervised learning

From fitting predictive models to "machine learning"

- classical statistical analysis: estimate marginal effects
- predictive models: forecast the (unknown) value of a (new or future) observation
- machine learning: make predictions on data using a more broadly defined combination of statistical models and computational algorithms

Supervised v. unsupervised learning

- *Supervised methods* require a training set that exmplify constrasting classes, identified by the researcher
  - regression models belong to this category
- *Unsupervised methods* identify patterns without requiring an explicit training step
  - often involves calibrating some critical input parameter, such as the number of categories into which items will be clustered
  - more post-hoc interpretation is required

Supervised v. unsupervised methods: examples

- Supervised: Naive Bayes, k-Nearest Neighbor, Support Vector Machines (SVM)
- Unsupervised: correspondence analysis, IRT models, factor analytic approaches

Scaling distance

Unsupervised "learning": scaling distance

- Features are treated as a quantitative matrix of variable values features
  - often normalized or standardized to allow similar computations of distance
- Many possible definitions of *distance* exist
  - 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

What a distance matrix looks like

$$\begin{bmatrix} & 0.3 & 0.4 & 0.7 \\ 0.3 & & 0.5 & 0.8 \\ 0.4 & 0.5 & & 0.45 \\ 0.7 & 0.8 & 0.45 & \end{bmatrix}$$

For instance, the dissimilarity between the first and second observations is 0.3, and the dissimilarity between the second and fourth observations is 0.8.

USArrests dataset example

```
head(USArrests, 10)
```

```
##             Murder Assault UrbanPop Rape
## Alabama       13.2     236       58 21.2
## Alaska        10.0     263       48 44.5
## Arizona        8.1     294       80 31.0
## Arkansas       8.8     190       50 19.5
## California     9.0     276       91 40.6
## Colorado       7.9     204       78 38.7
## Connecticut    3.3     110       77 11.1
## Delaware       5.9     238       72 15.8
## Florida       15.4     335       80 31.9
## Georgia       17.4     211       60 25.8
```

USArrests dataset example

```
as.matrix(dist(USArrests))[1:5, 1:5]
```

```
##             Alabama   Alaska   Arizona  Arkansas California
## Alabama     0.00000 37.17701  63.00833  46.92814   55.52477
## Alaska     37.17701  0.00000  46.59249  77.19741   45.10222
## Arizona    63.00833 46.59249   0.00000 108.85192   23.19418
## Arkansas   46.92814 77.19741 108.85192   0.00000   97.58202
## California 55.52477 45.10222  23.19418  97.58202    0.00000
```

```
as.matrix(dist(USArrests, method = "manhattan"))[1:5, 1:5]
```

```
##            Alabama Alaska Arizona Arkansas California
## Alabama        0.0   63.5    94.9     60.1       96.6
## Alaska        63.5    0.0    78.4    101.2       60.9
## Arizona       94.9   78.4     0.0    146.2       39.5
## Arkansas      60.1  101.2   146.2      0.0      148.3
## California    96.6   60.9    39.5    148.3        0.0
```

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

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

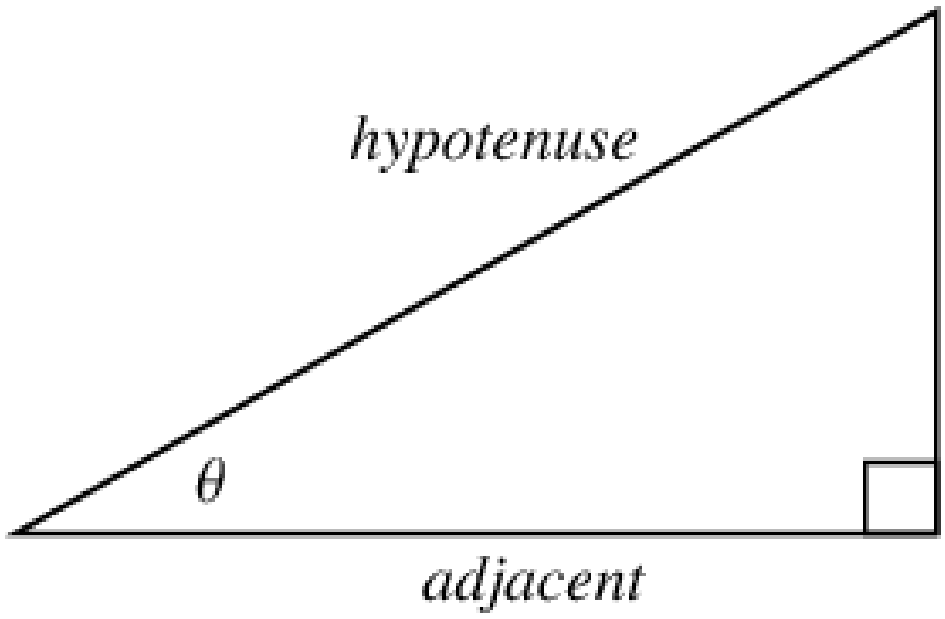
- In vector notation:

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

- 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

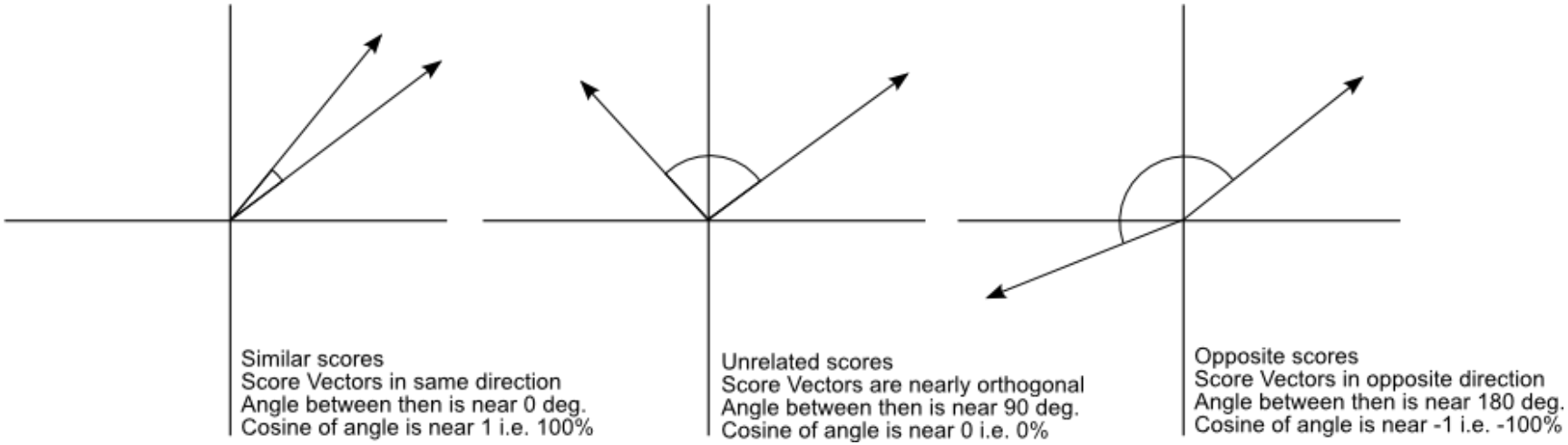
Cosine similarity

- 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

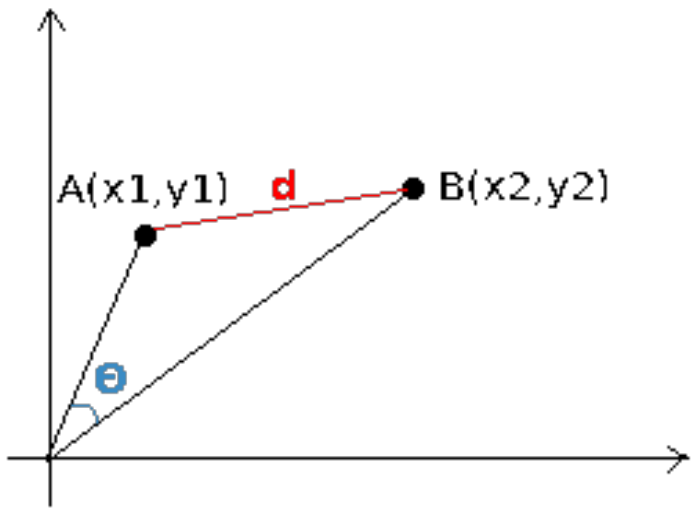
- Document 1
  Gilbert: 3, hurricane: 2, rains: 1, storm: 2, winds: 2
- Document 2
  Gilbert: 2, hurricane: 1, rains: 0, storm: 1, winds: 2

Example text: cosine similarity in R

```
require(quanteda)

## Loading required package:  quanteda
##
## Attaching package:  'quanteda'
##
## The following object is masked from 'package:base':
##
##    sample

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

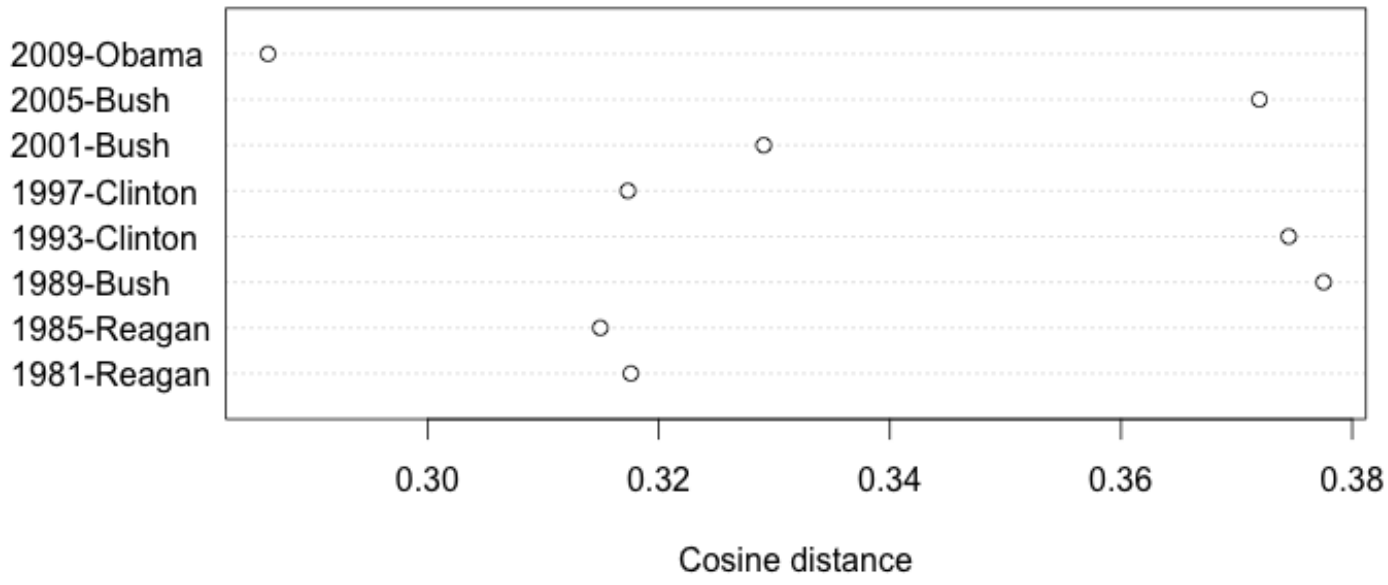
- Similar to the Cosine similarity
- Formula

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

- Ranges from 0 to 1.0

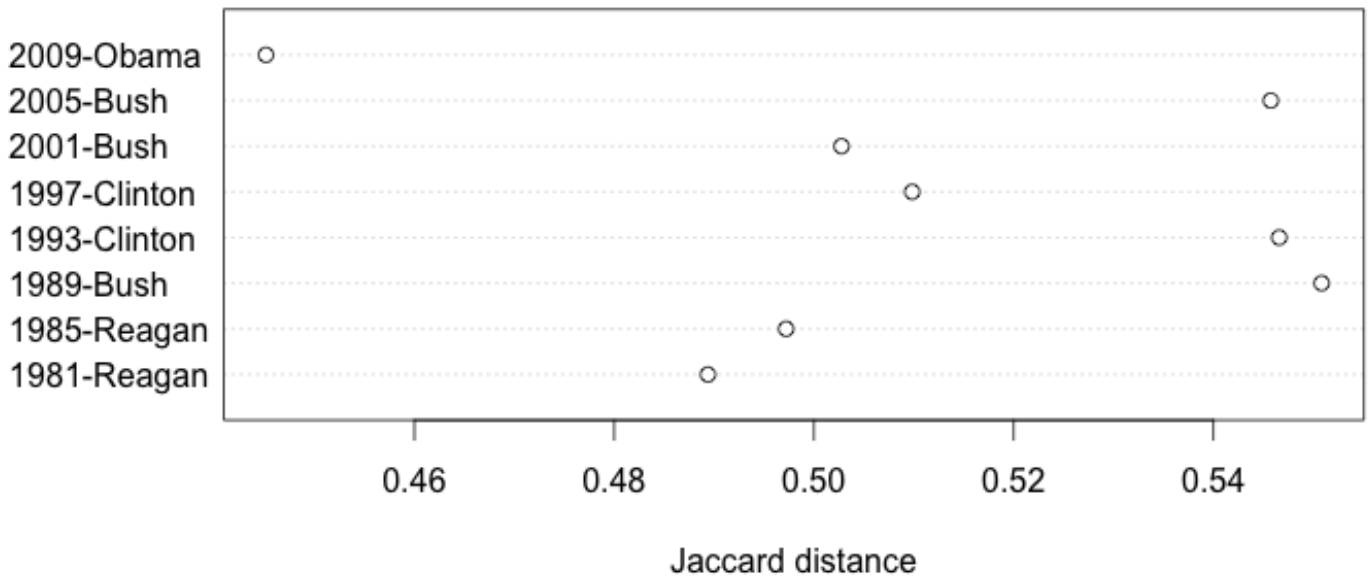
Example: Inaugural speeches, cosine distance to Obama 2014

```
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)

Clustering

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

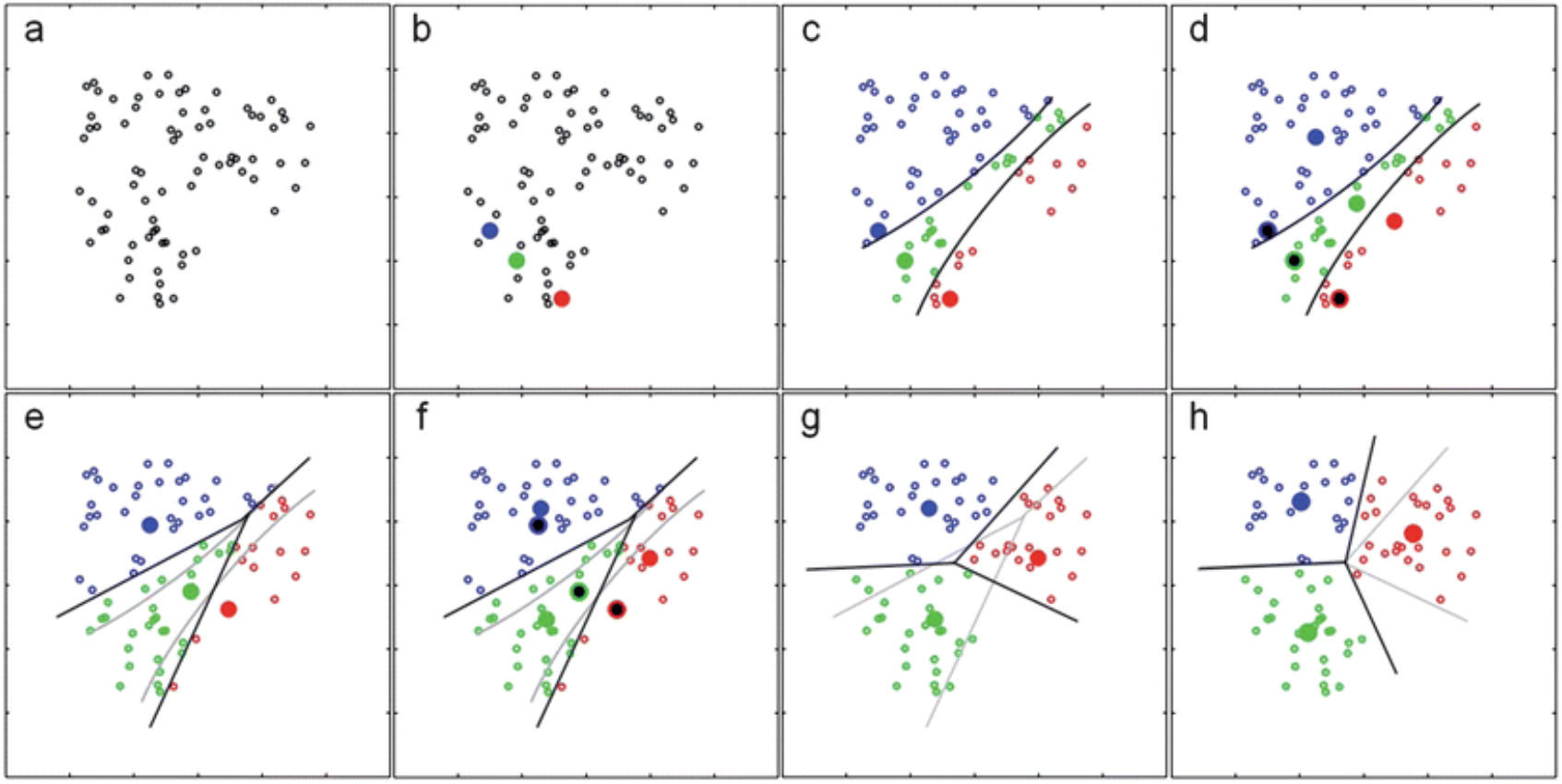
k-means clustering

- 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

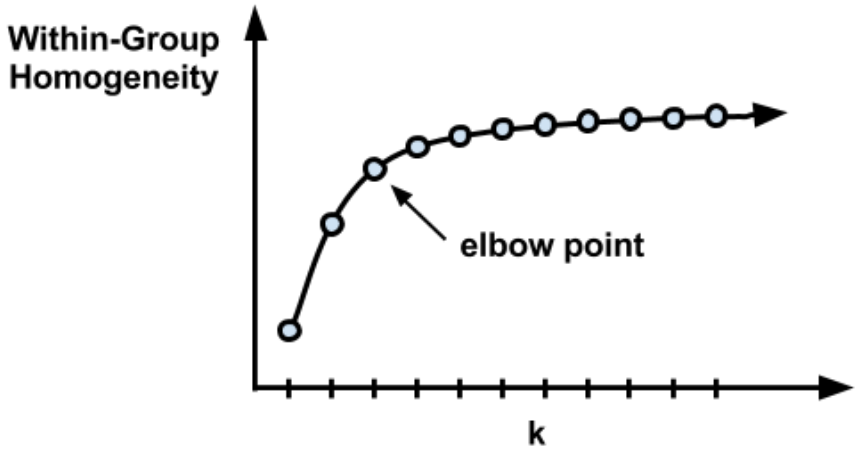


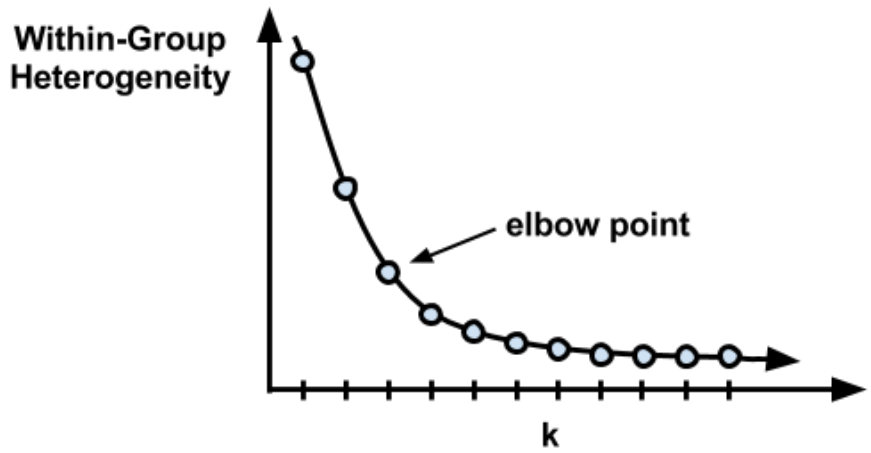
Choosing the appropriate number of clusters

- 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)

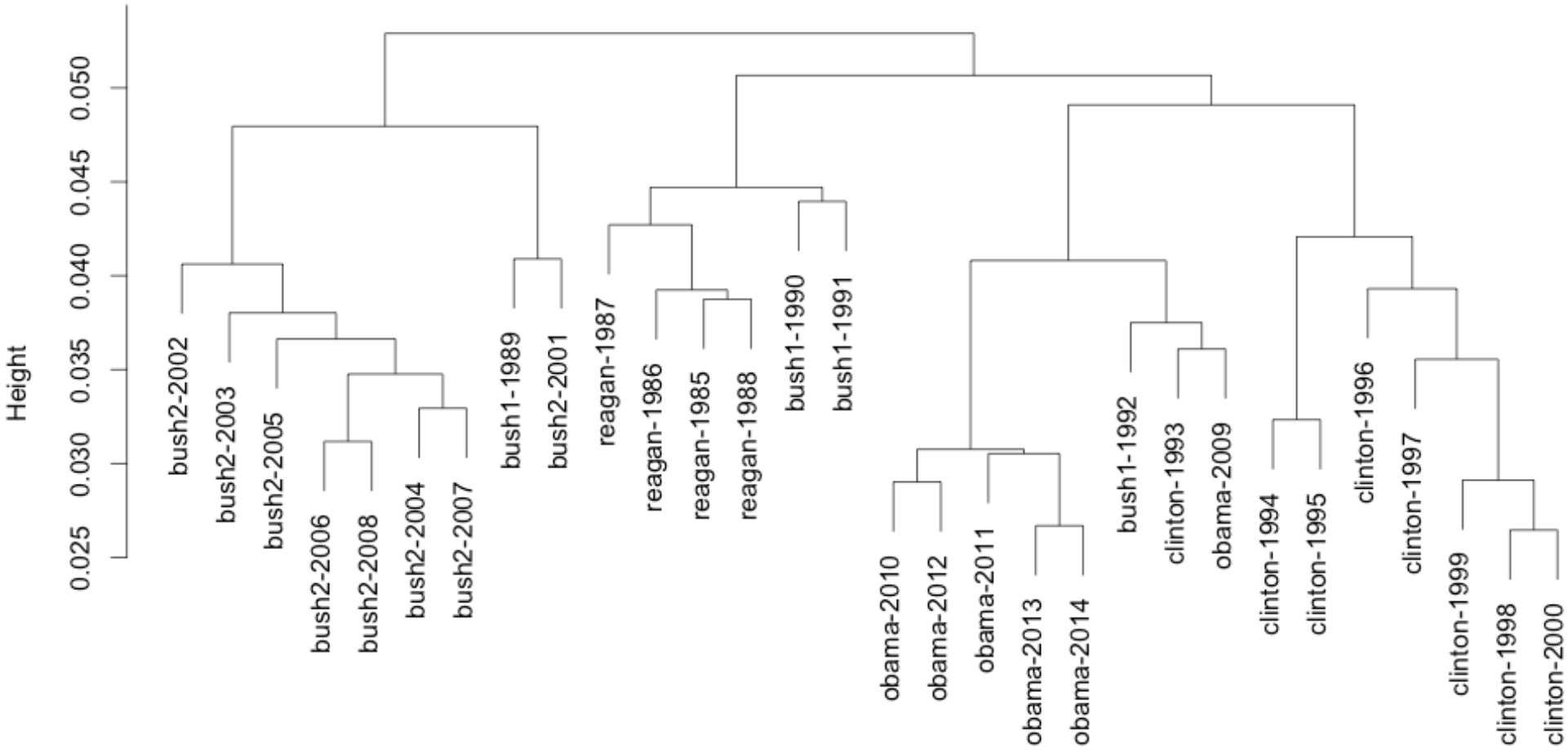
Hierarchical clustering algorithm

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: 4090
## Features occurring in fewer than 3 documents: 3536

# 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)
```

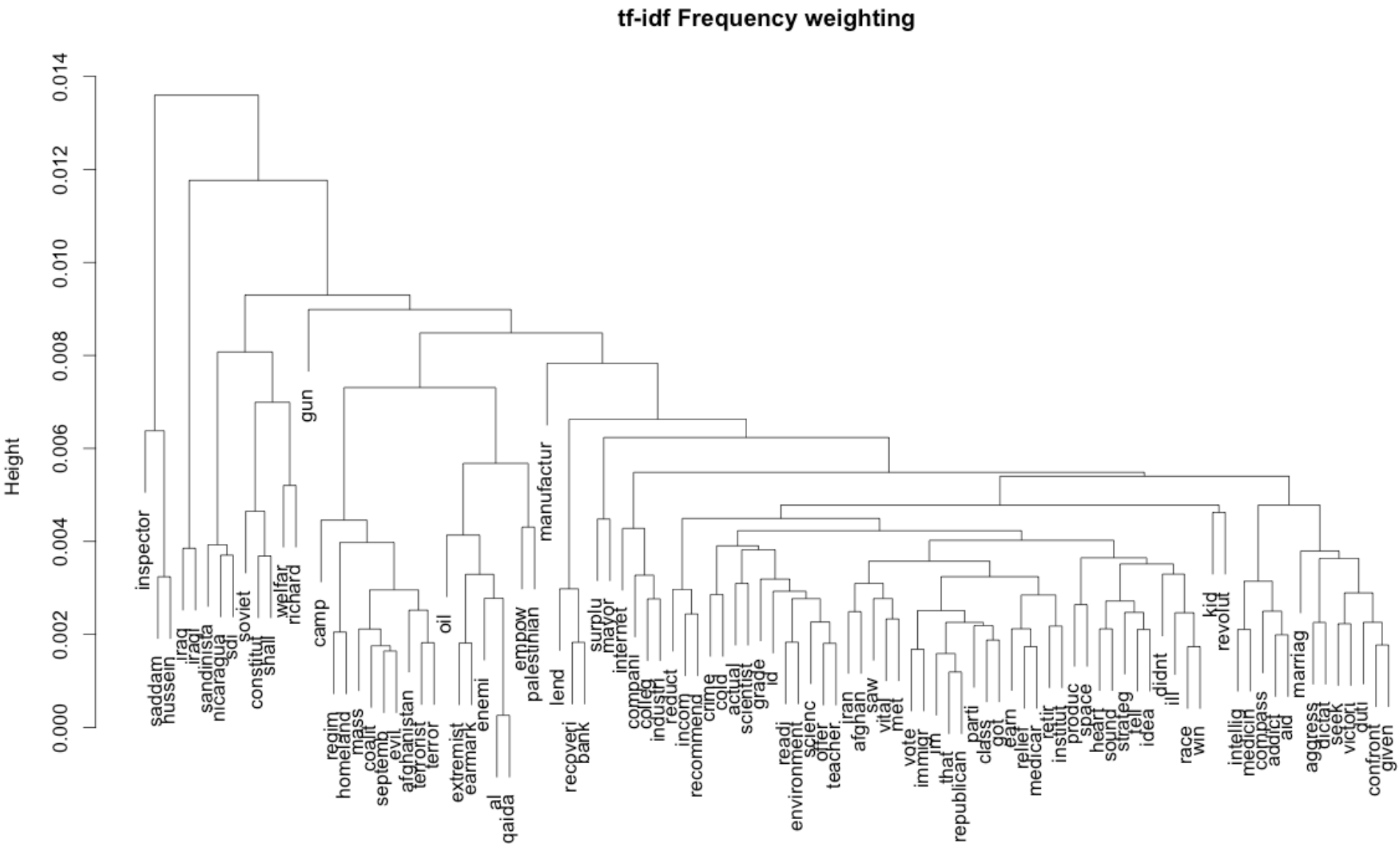
Dendrogram: Presidential State of the Union addresses



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")
```

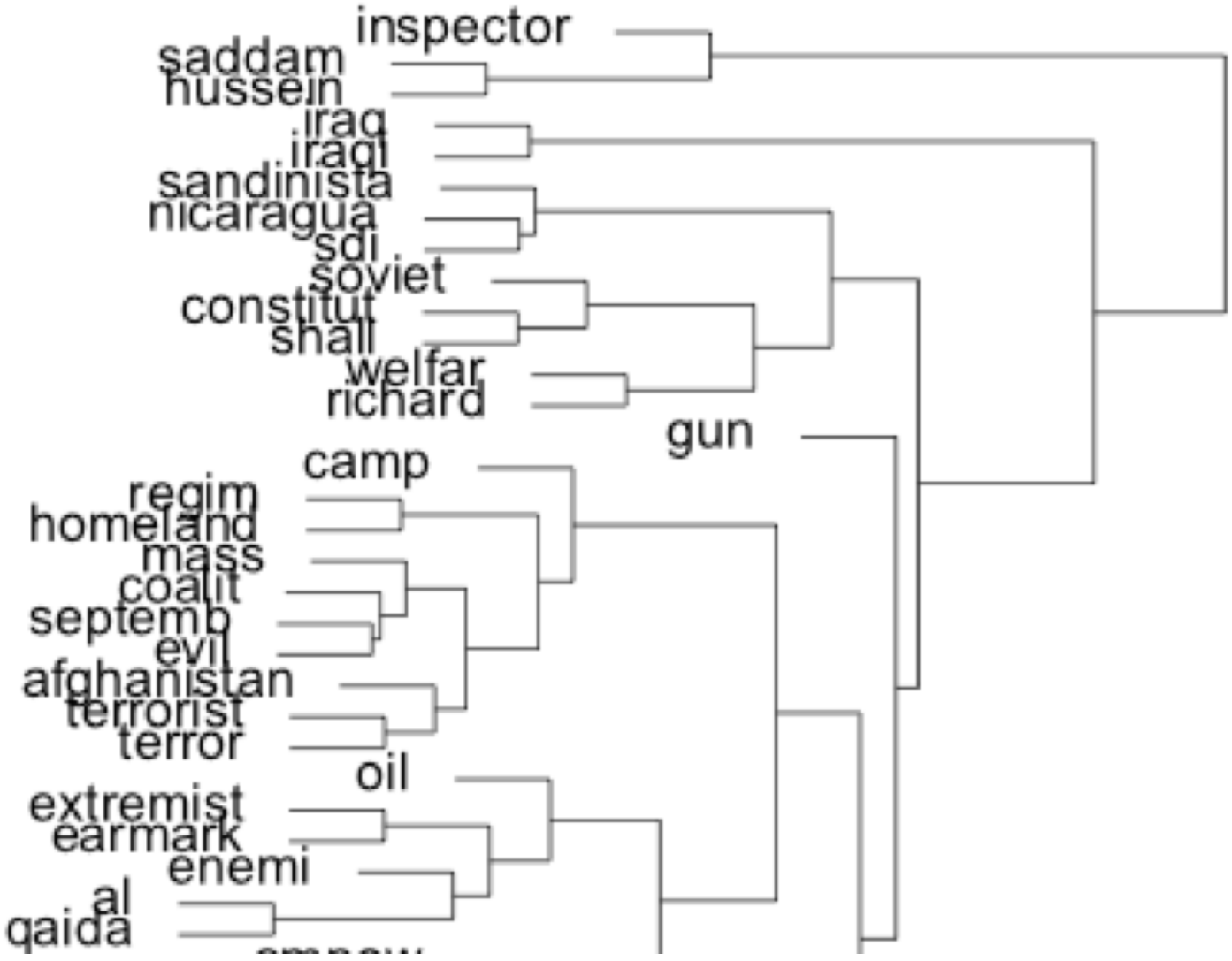
Dendrogram: Presidential State of the Union addresses



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



Association rules

Introduction to association rules

- Association rule mining is used to discover objects or attributes that frequently occur together, e.g.
  - movies or music that users prefer
  - baskets of products purchased online or in-store
- Used extensively in recommendation engines
- Terminology:

  transaction a bundle of associated items, such as a collection of movies watched or items puchased, forming the unit of analysis

  itemset the items that make up a transaction, such as purchases, web sites visited, movies watched, etc.

Introduction to association rules

- Association rule mining is used to discover objects or attributes that frequently occur together, e.g.
  - movies or music that users prefer
  - baskets of products purchased online or in-store
- Used extensively in recommendation engines
- Terminology:

  transaction a bundle of associated items, such as a collection of movies watched or items puchased, forming the unit of analysis

  itemset the items that make up a transaction, such as purchases, web sites visited, movies watched, etc.
- Many different algorithms, but we will focus on one: the *a priori* algorithm

The *apriori* algorithm

- Two core notions:

  support the support of an item X is the number of transactions that contain X divided by the total number of transactions

  confidence expresses our "confidence" in the relation *if* X, *then* Y

  - formally:

$$\text{support}(\text{union}(X, Y))/\text{support}(X)$$

- The goal is to discover the interesting rules in a dataset above some pre-defined thresholds of support and confidence, such as 10% and 60%)