

Machine Learning 101

Patrick Boily

Department of Mathematics and Statistics, University of Ottawa
Idlewyld Analytics and Consulting Services
Data Action Lab

(in collaboration with Jen Schellinck and Shintaro Hagiwara)

Instructor

Bio

- Professor ['19 – now, Dept. of Math & Stats, uOttawa]
- Lecturer ['99 – now, uOttawa | UQO | Carleton] (~55 courses/workshops)
- Manager ['12 – '19, CQADS, Carleton]
- Adjunct Research Professor ['18 – '21, Sprott School of Business, Carleton]
- Public servant ['08 – '12, CBSA | StatCan | TC | PWGSC]
- Likes symmetry and patterns
- Cheers for the Sens & fancies himself a handyman

Projects

- Global Affairs Canada
- Nuclear Waste Management Organization
- Canadian Air Transport Security Authority
- Various other clients: ~40 projects

Specialties

- Data visualization, data cleaning (unfortunately)
- Applying wide breadth of techniques to all types of data



Contents

Clustering and k -Means

- Basics
- k -Means
- Notes and Comments
- Example – Iris Dataset
- Clustering Validation (1st pass)

Hierarchical Clustering

Density-Based Clustering

Spectral Clustering

Clustering Validation

Clustering and k -Means

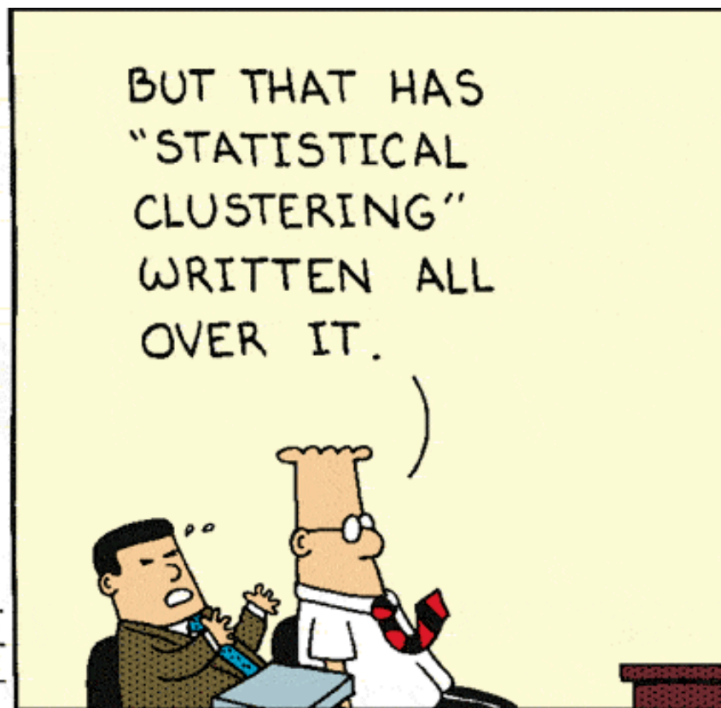


www.unitedmedia.com

S. Adams



6/13/97 © 1997 United Feature Syndicate, Inc.

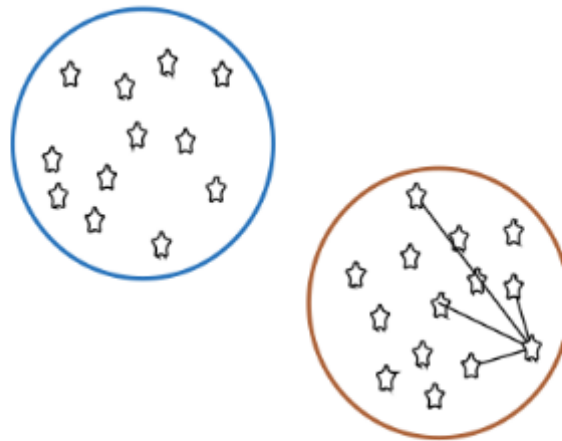


Clustering Overview

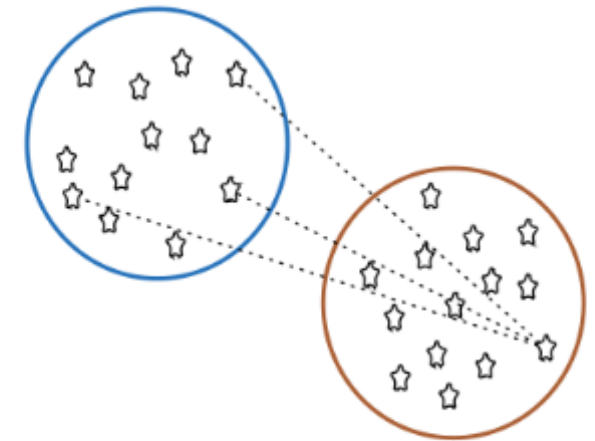
In **clustering**, the data is divided into **naturally occurring groups**. Within each group, the data points are **similar**; from group to group, they are **dissimilar**.

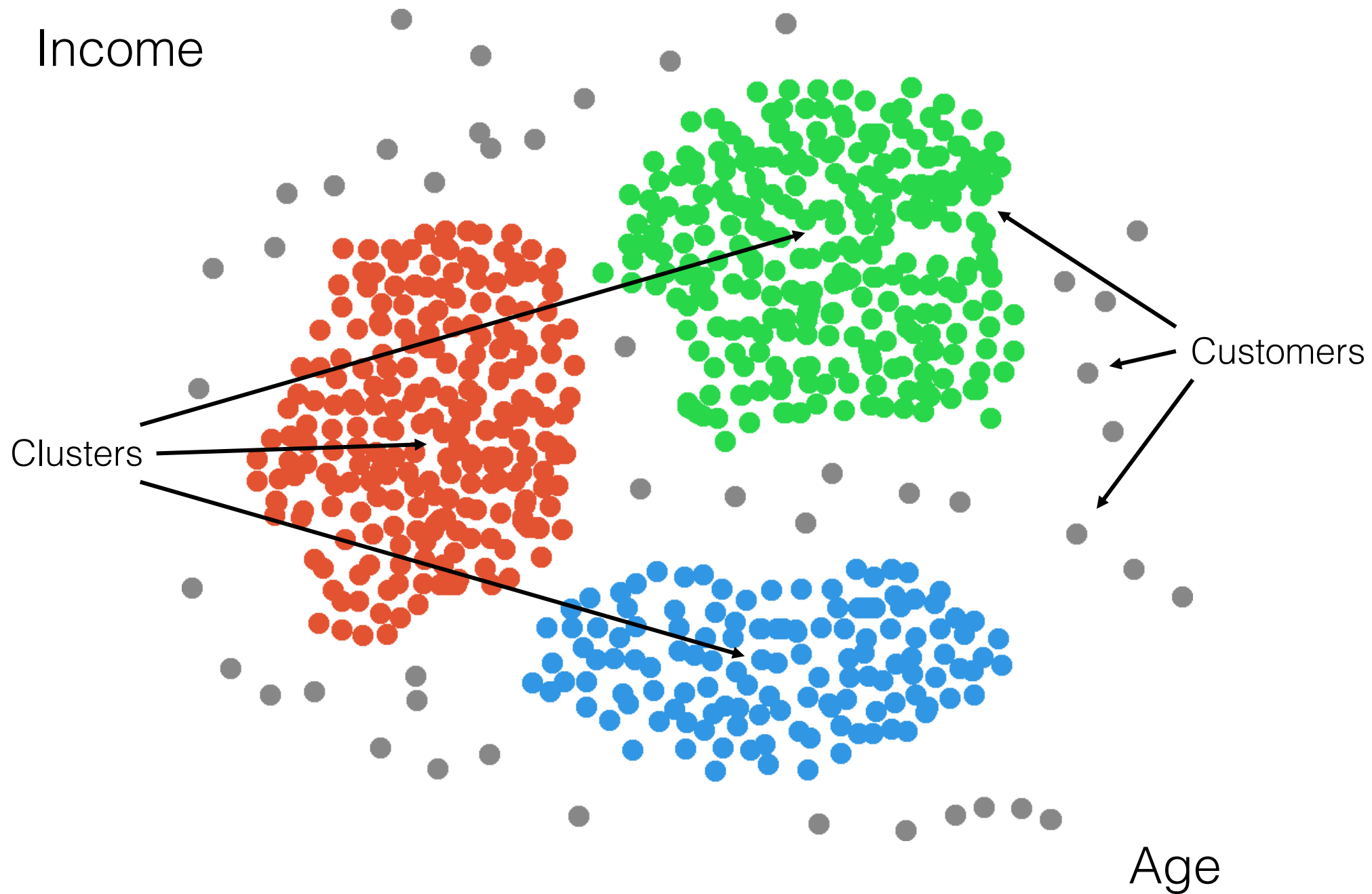
The grouping labels are not determined ahead of time, so clustering is an example of **unsupervised** learning.

average distance to points in own cluster (**low is good**)



average distance to points in neighbouring cluster (**high is good**)





Clustering Overview

Clustering is a relatively **intuitive** concept for human beings as our brains do it unconsciously

- facial recognition
- searching for patterns, etc.

In general, people are very good at **messy** data, but computers and algorithms have a harder time.

Part of the difficulty is that there is **no agreed-upon definition of what constitutes a cluster:**

- “I may not be able to define what it is, but I know one when I see one”

Clustering Overview

Clustering algorithms can be **complex** and **non-intuitive**, based on varying notions of similarities between observations.

- in spite of that, the temptation to explain clusters *a posteriori* is **strong**

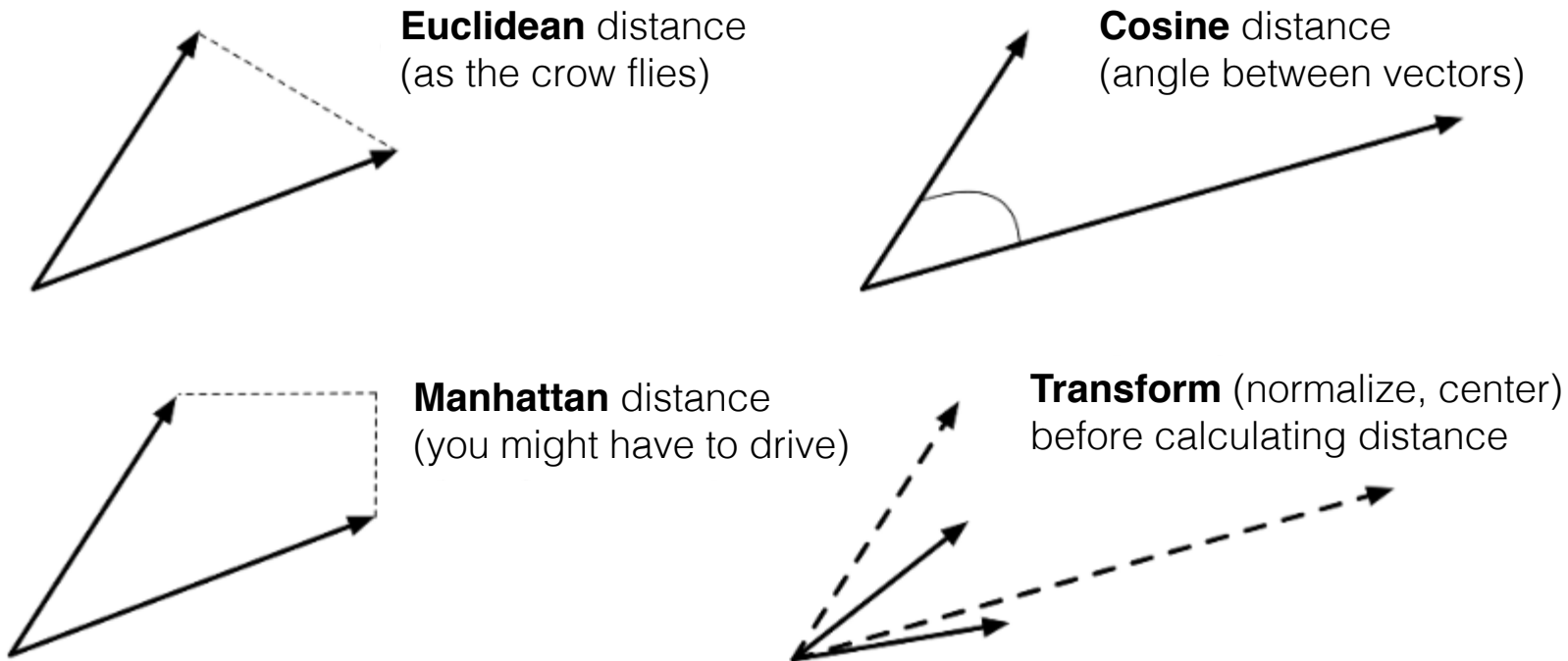
They are also (typically) **non-deterministic**:

- the same algorithm, applied twice (or more) to the same dataset, can discover completely different clusters
- the order in which the data is presented can play a role
- so can starting configurations

Discussion: What does this (potential) non-repeatability imply for validation?

Clustering Requirement

A measure of **similarity** w (or a distance d) between observations.



Typically, $w \rightarrow 1$ as $d \rightarrow 0$, and $w \rightarrow 0$ as $d \rightarrow \infty$.

Other metrics also available: Hamming, Jaccard, Pearson, etc.

Applications

Text Documents

- grouping similar documents according to their topics, based on the patterns of common and unusual words

Product Recommendations

- grouping online purchasers based on the products they have viewed, purchased, liked, or disliked
- grouping products based on customer reviews

Marketing and Business

- grouping client profiles based on their demographics and preferences

Applications

Music

- finding similar albums by grouping the customers who own them

Social Network Analysis

- recognizing communities within large groups of people

Medical Imaging

- differentiating between different tissue types in a 3D voxel

Genetic Clustering

- inferring structures in populations

Other Uses

Dividing a larger group (or area, or category) into **smaller** groups, with members of the smaller groups guaranteed to have similarities of some kind.

- tasks may then be solved separately for each of the smaller groups
- this may lead to increased accuracy once the separate results are aggregated

Creating (new) taxonomies **on the fly**, as new items are added to a group of items

- this would allow for easier product navigation on a website like Netflix, for instance.

	Y_1	Y_2	...	Y_p
01	$x_{01,1}$	$x_{01,2}$	...	$x_{01,p}$
02	$x_{02,1}$	$x_{02,2}$	...	$x_{02,p}$
03	$x_{03,1}$	$x_{03,2}$	...	$x_{03,p}$
04	$x_{04,1}$	$x_{04,2}$	...	$x_{04,p}$
05	$x_{05,1}$	$x_{05,2}$	...	$x_{05,p}$
06	$x_{06,1}$	$x_{06,2}$	...	$x_{06,p}$
07	$x_{07,1}$	$x_{07,2}$	...	$x_{07,p}$
08	$x_{08,1}$	$x_{08,2}$	...	$x_{08,p}$
...			...	
%%	$x_{\%,1}$	$x_{\%,2}$	...	$x_{\%,p}$

Clustering
Algorithm

Model

	Y_1	Y_2	...	Y_p	
01	$x_{01,1}$	$x_{01,2}$	...	$x_{01,p}$	■
02	$x_{02,1}$	$x_{02,2}$	...	$x_{02,p}$	■
03	$x_{03,1}$	$x_{03,2}$	...	$x_{03,p}$	■
04	$x_{04,1}$	$x_{04,2}$	...	$x_{04,p}$	■
05	$x_{05,1}$	$x_{05,2}$	...	$x_{05,p}$	■
06	$x_{06,1}$	$x_{06,2}$	...	$x_{06,p}$	■
07	$x_{07,1}$	$x_{07,2}$	...	$x_{07,p}$	■
08	$x_{08,1}$	$x_{08,2}$	...	$x_{08,p}$	■
...			...		
%%	$x_{\%,1}$	$x_{\%,2}$	...	$x_{\%,p}$	■

Clustering
Validation

Deployment

	▲
01	▲
02	▲
03	▲
04	▲
05	▲
06	▲
07	▲
08	▲
...	...
%%	▲

External
Info
(if available,
appropriate)

Clustering Schemes

***k*-Means**

- classical (and over-used) model
- assumptions made about the shape of clusters

Hierarchical Clustering

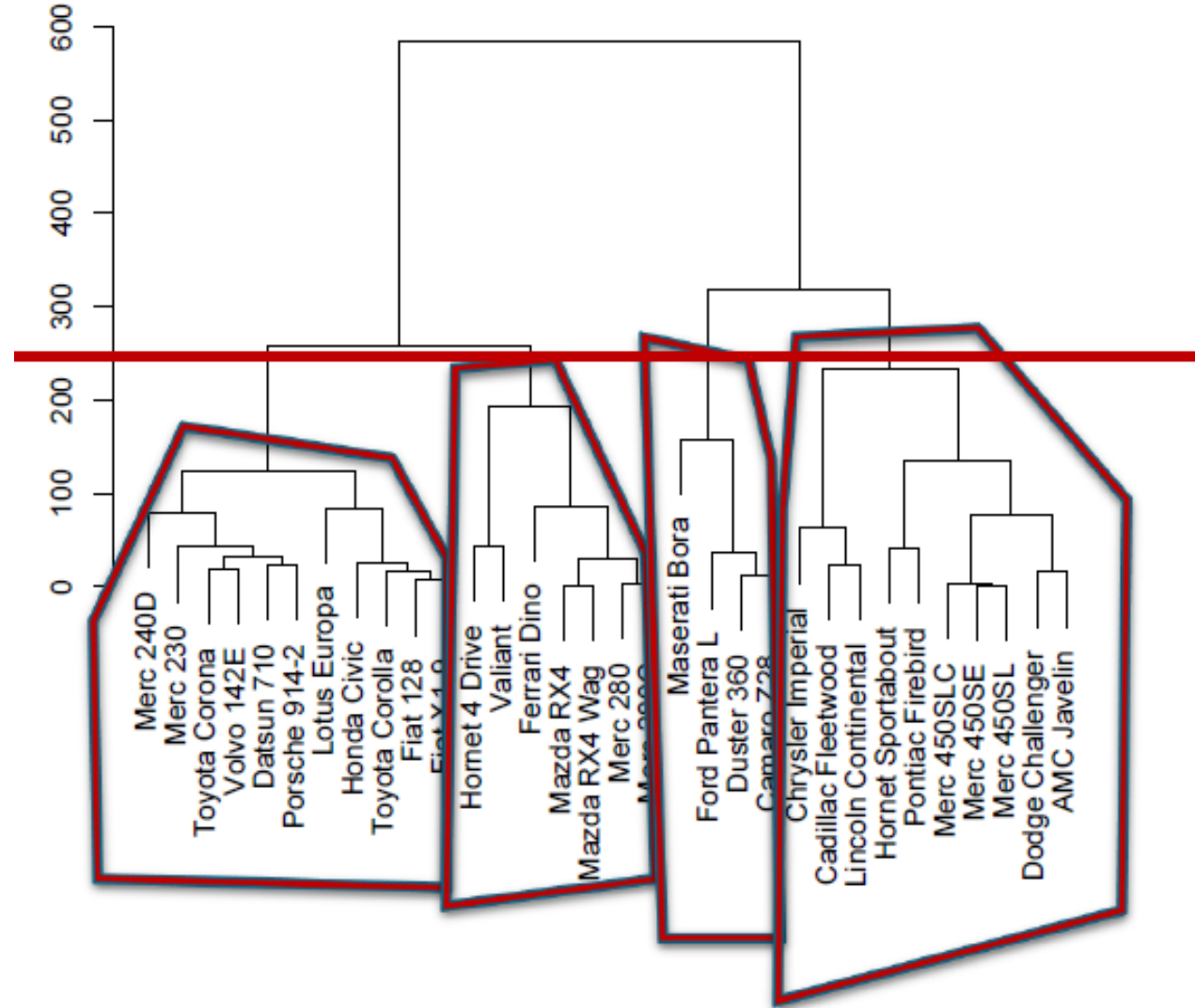
- easy to interpret, deterministic

Latent Dirichlet Allocation

- used for topic modeling

Expectation Maximization

Hierarchical Clustering



Clustering Schemes

Balanced Iterative Reducing and Clustering using Hierarchies

- aka BIRCH

Density-Based Spatial Clustering of Applications with Noise

- graph-based

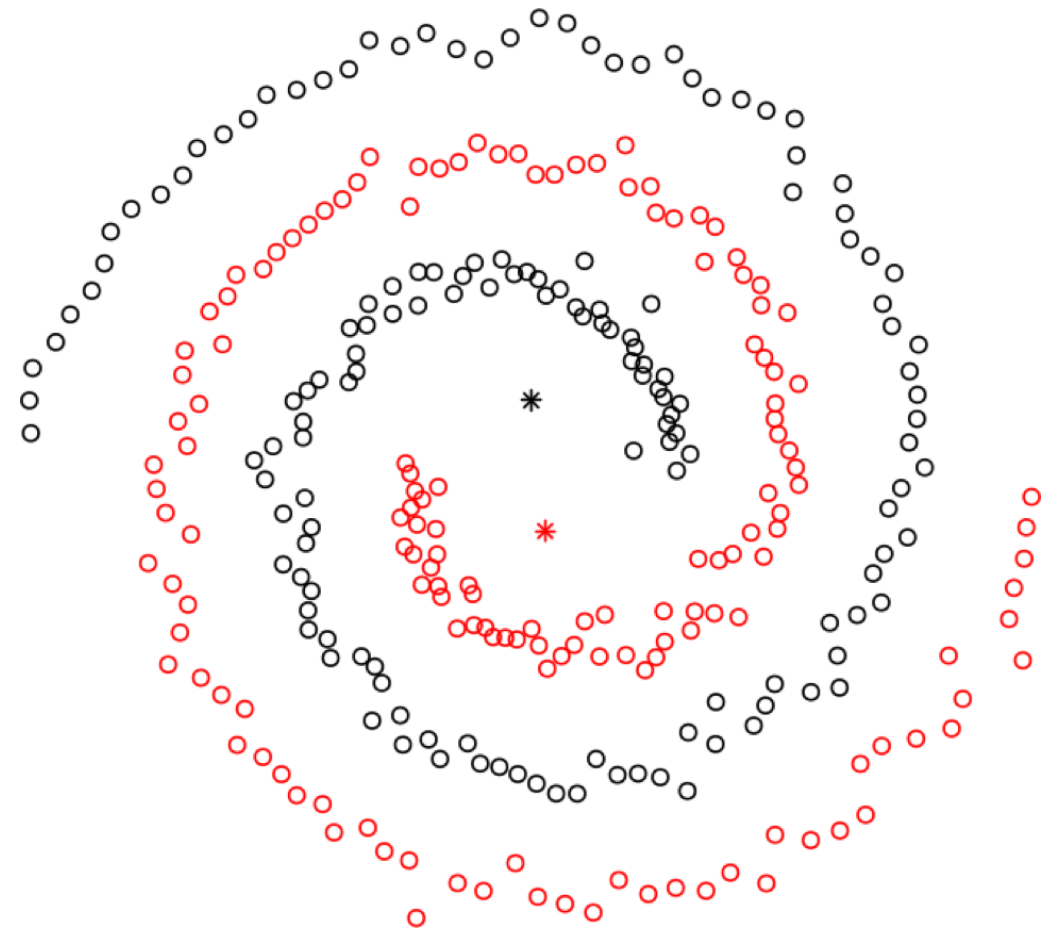
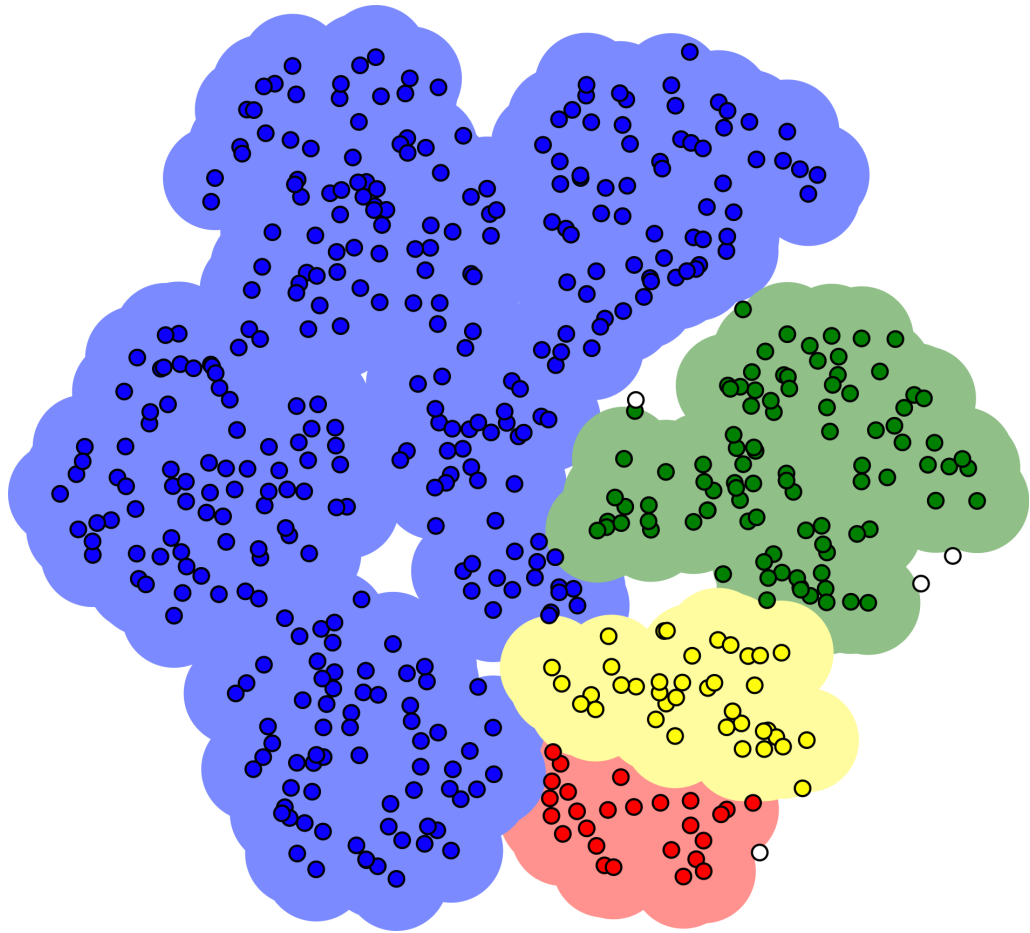
Affinity Propagation

- selects the optimal number of clusters automatically

Spectral Clustering

- recognizes non-blob clusters

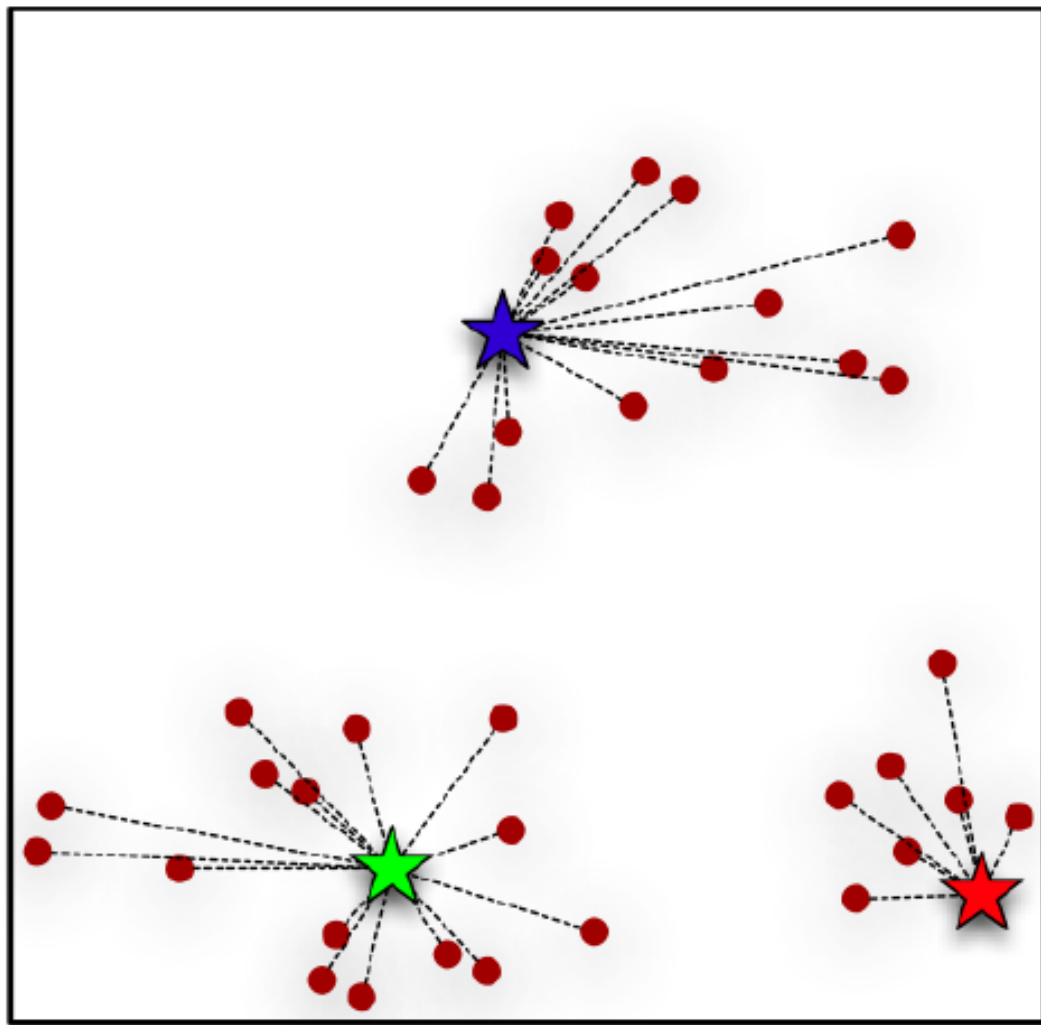
DBSCAN and Spectral Clustering



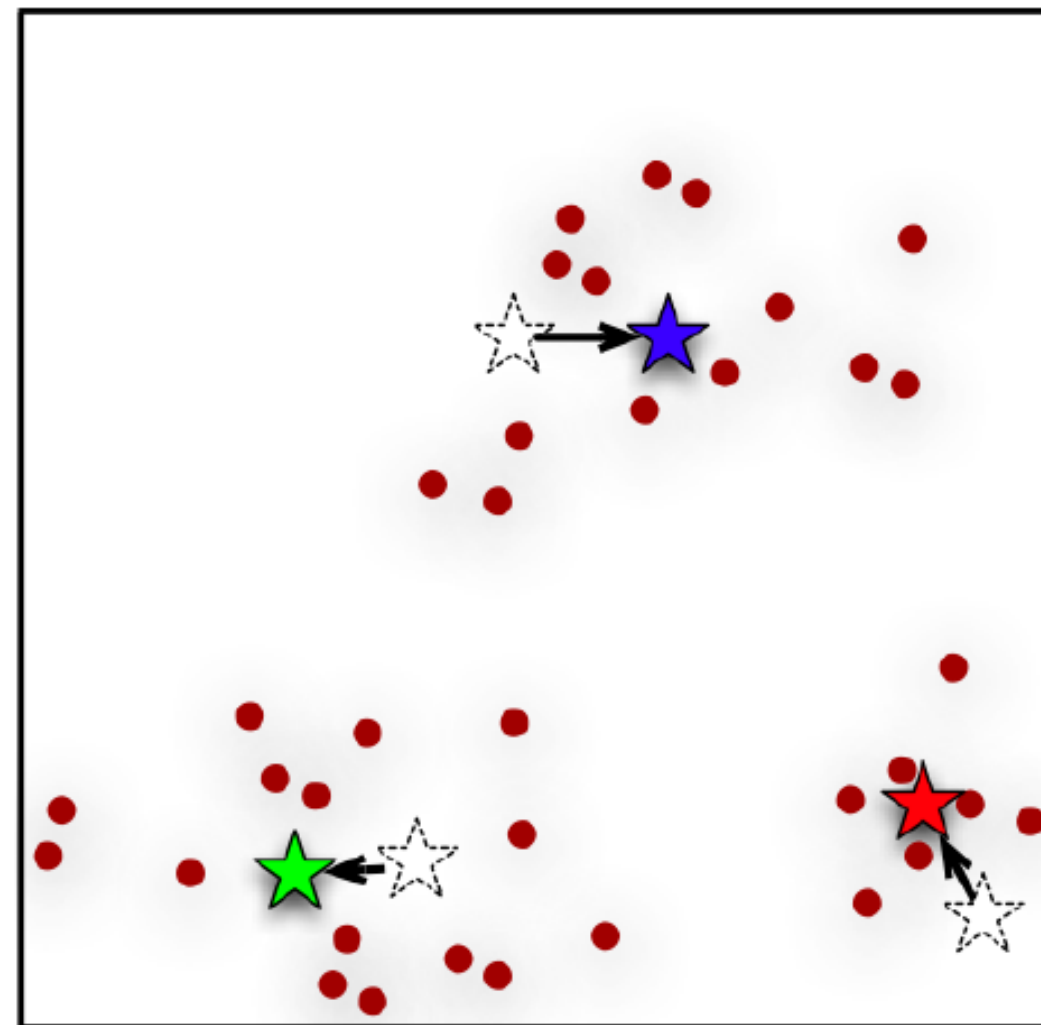
k -means is well-adapted to numerical data (although it can also be used for categorical data), but it has a tendency to force clusters of **roughly equivalent sizes**.

k -Means Algorithm

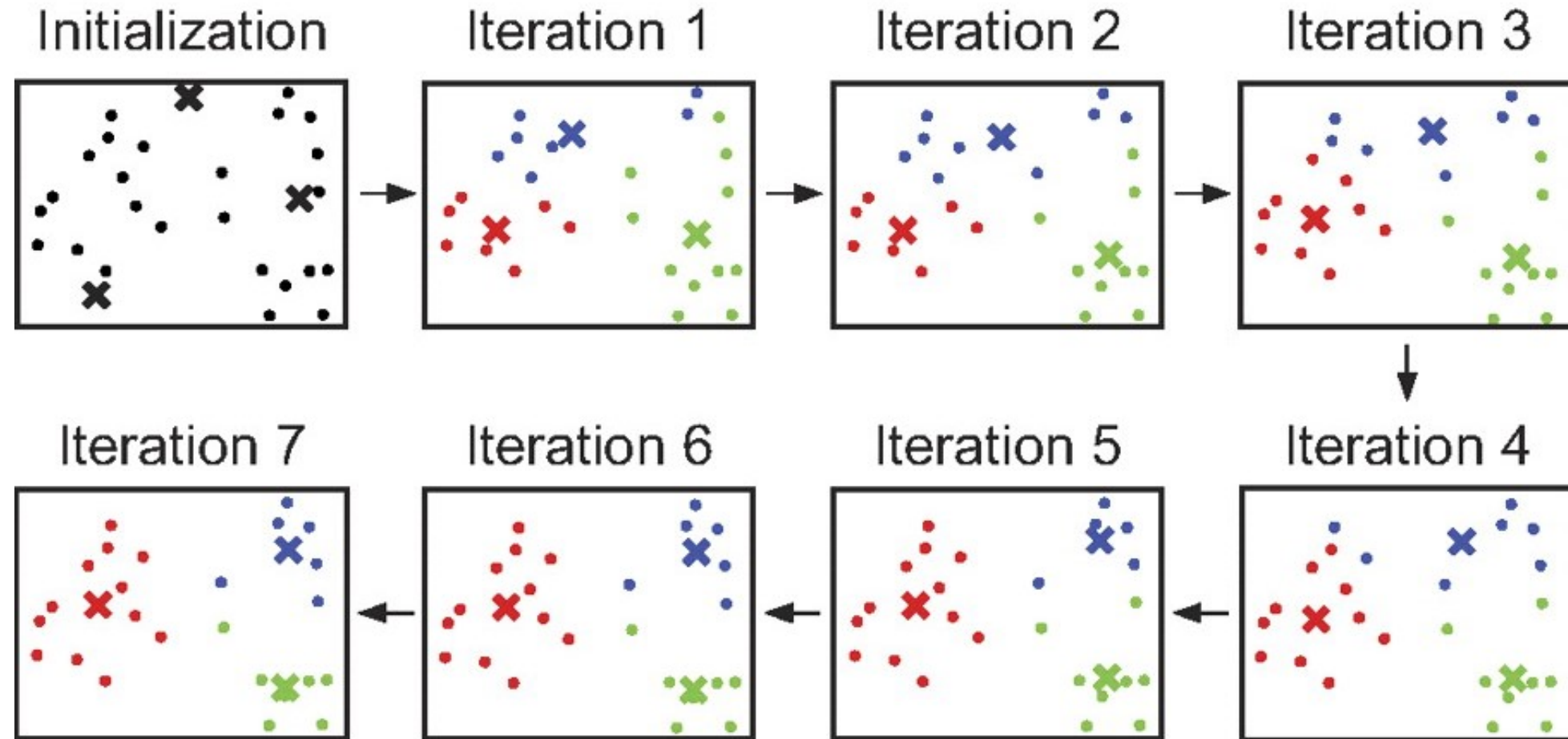
1. Select the desired **number of clusters**, say k
2. Randomly choose k instances as initial **cluster centres**
3. Calculate the **distance** from each observation to each centre
4. Place each instance in the cluster whose centre it is **nearest** to
5. Compute the **centroid** for each cluster
6. Repeat steps 3 – 5 with the new centroids
7. Repeat step 6 until the clusters are **stable**



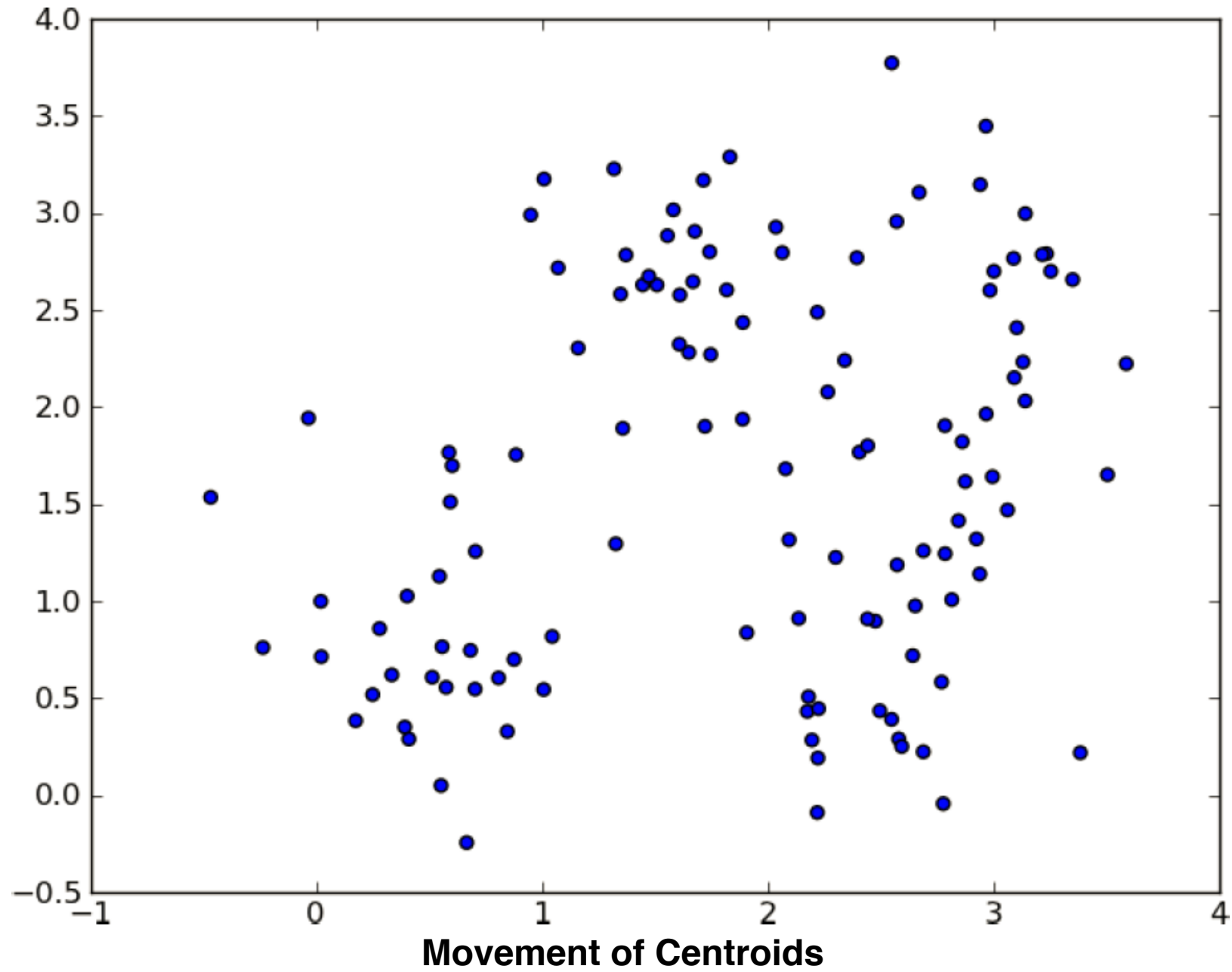
Cluster Allocation

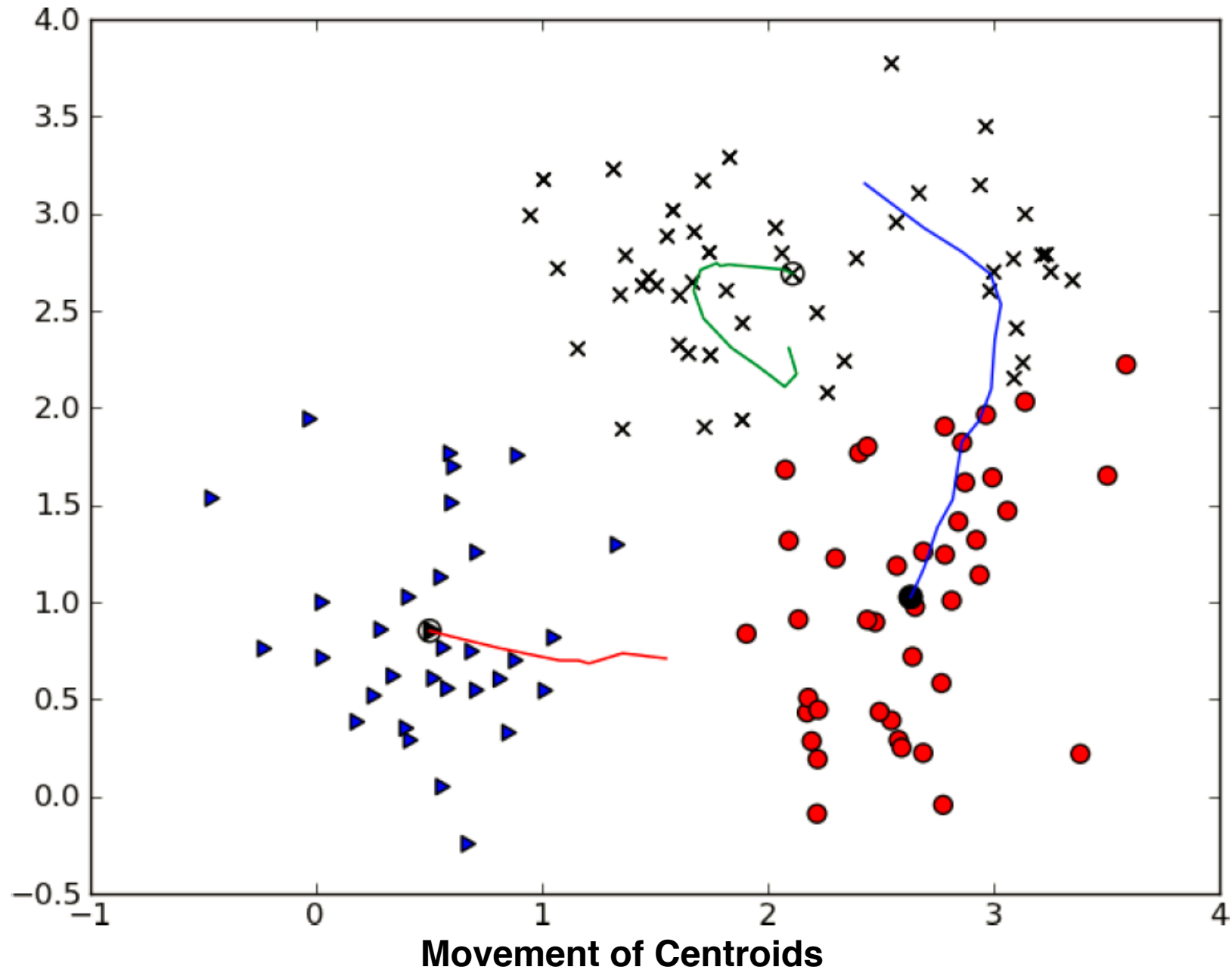


New Cluster Centroids



k -Means Iterative Process





k -Means Strengths

Easy to implement (without having to actually compute pairwise distances).

- extremely common as a consequence
- elegant and simple

In many contexts, k -means is a **natural** way to look at grouping observations.

Helps provide a **basic understanding of the data structure** in a first pass.

k -Means Limitations

Data points can only be assigned to **one** cluster.

- this can lead to overfitting
- robust solution: consider the **probability** of belonging to each cluster

Underlying clusters are assumed to be **blob-shaped**

- k -means will fail to produce useful clusters if that assumption is not met in practice

Clusters are assumed to be separate (discrete)

- k -means does not allow for **overlapping** or **hierarchical** groupings

Distance Measures (Metrics)

Categorical Variables*

- Hamming distance
- Russel/Rao index
- Jaccard
- Matching coefficient
- Dice's coefficient
- etc.

Numerical Variables

- Euclidean
- Manhattan
- Correlation
- Cosine
- Pearson
- etc.

No steadfast rule to determine which distance to use in *k*-means

Competing schemes are often produced using different metrics.

* may need to be dichotomized

Take-Away: with mixed data,
Hamming $\leftrightarrow$ Euclidean, Jaccard $\leftrightarrow$ Manhattan.

Clustering Challenges

Automation

relatively intuitive for humans, but hard to automate

Lack of a clear-cut definition

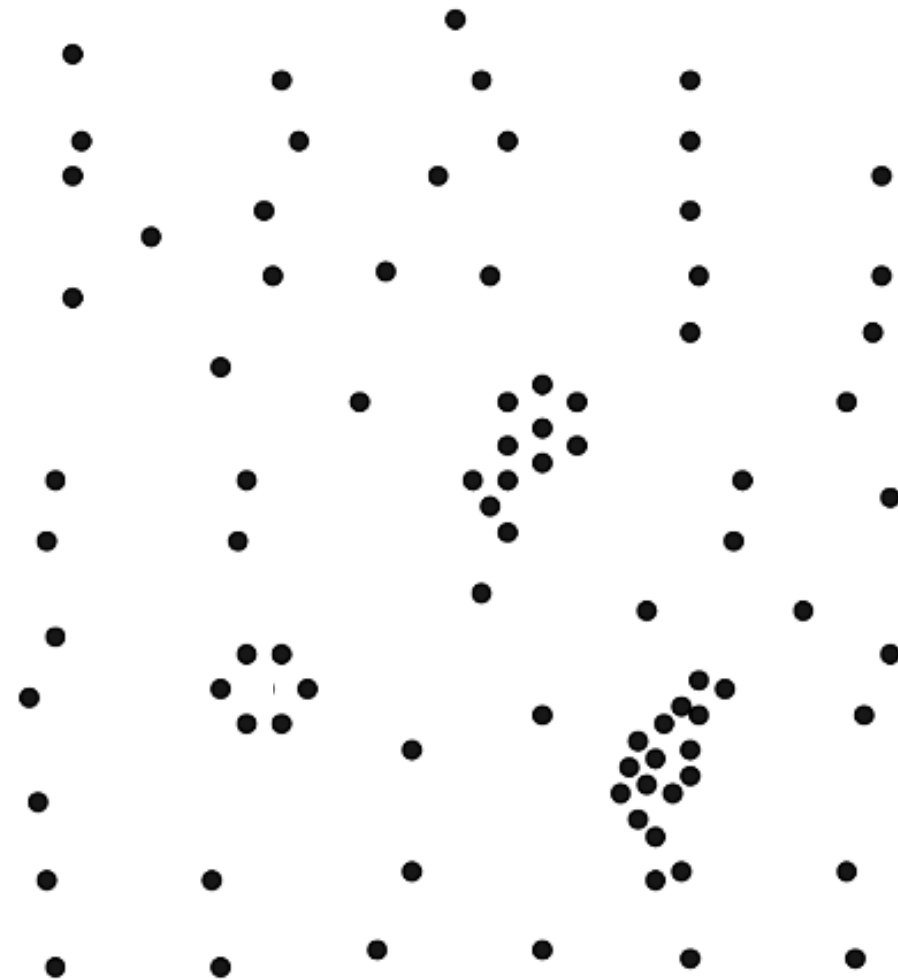
no universal agreement as to what constitutes a cluster

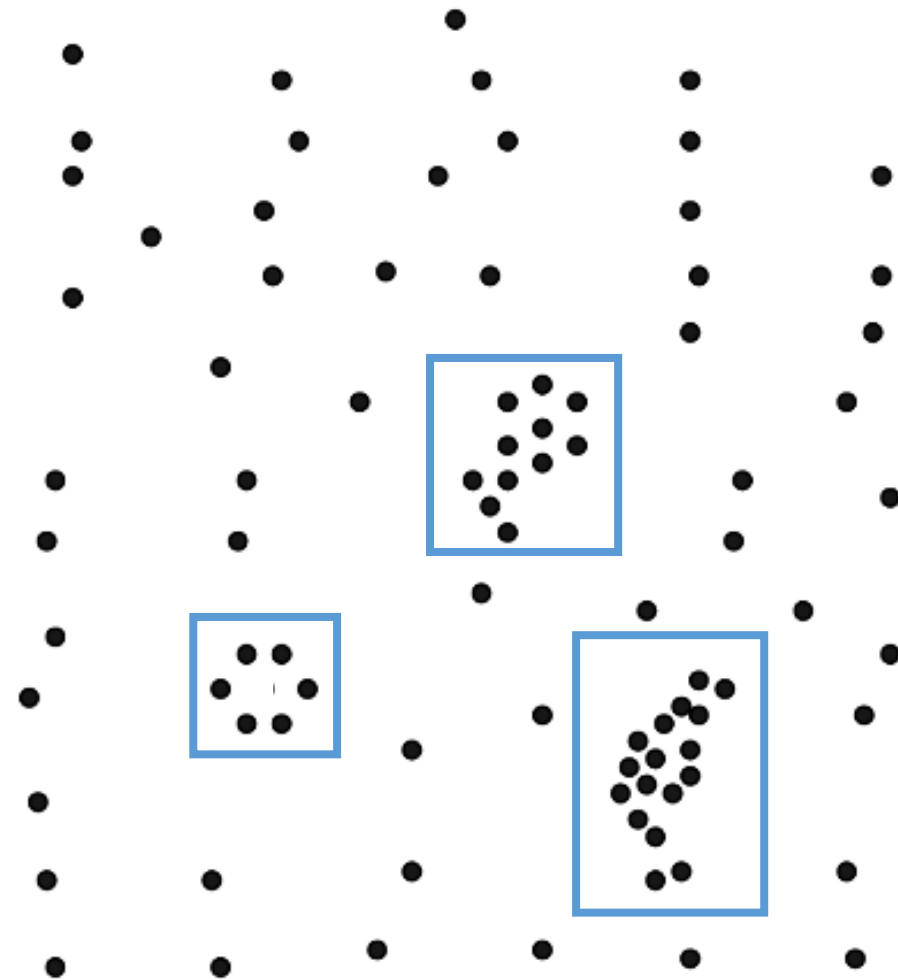
Lack of repeatability

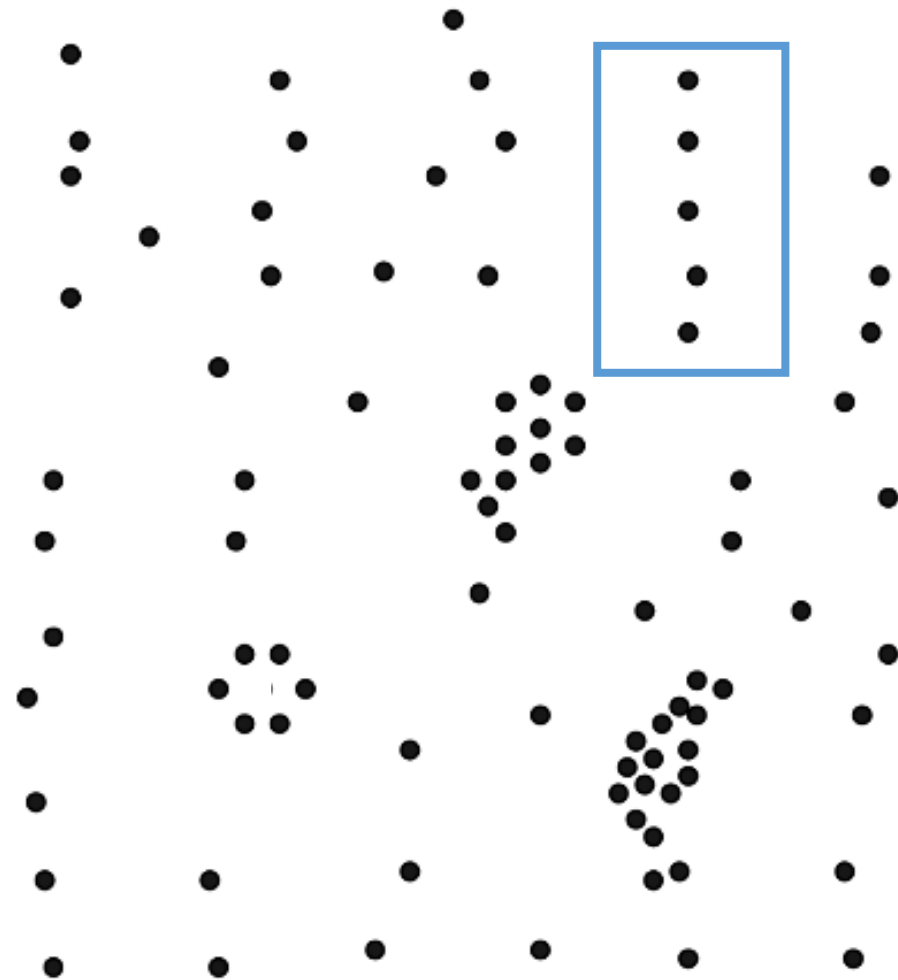
non-deterministic: the same algorithm, applied twice to the same dataset can discover completely different clusters

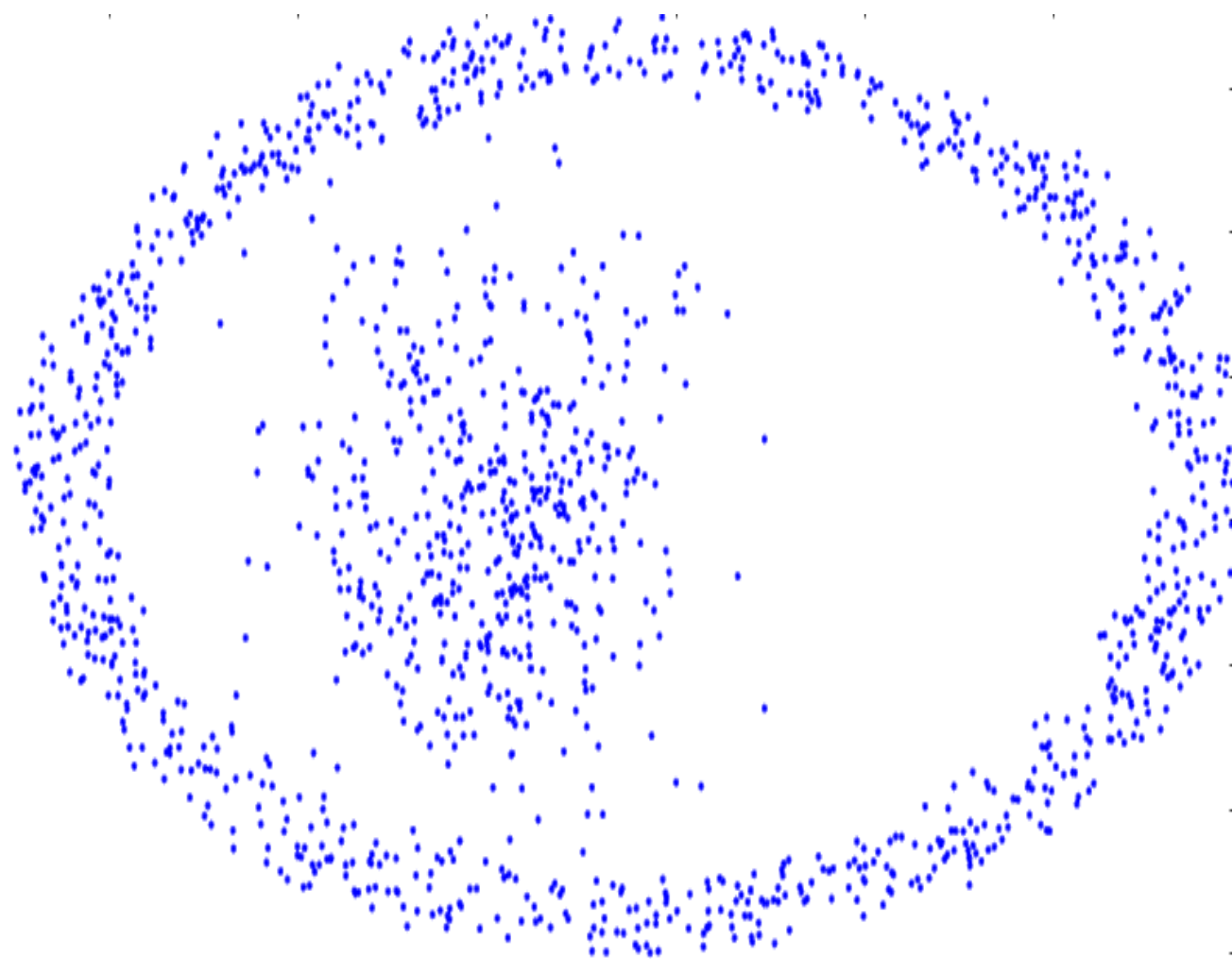
Number of clusters

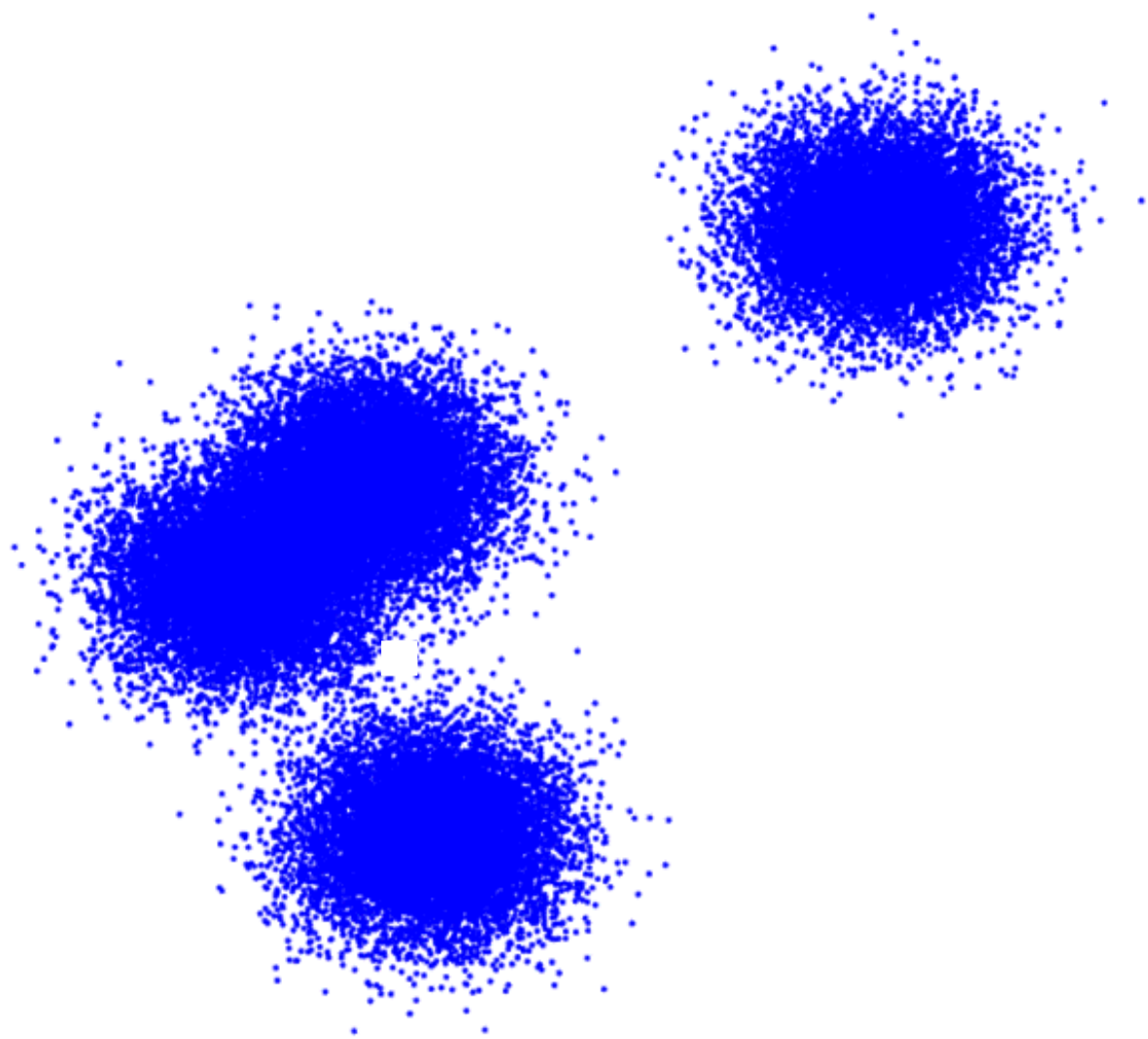
optimal number of clusters difficult to determine

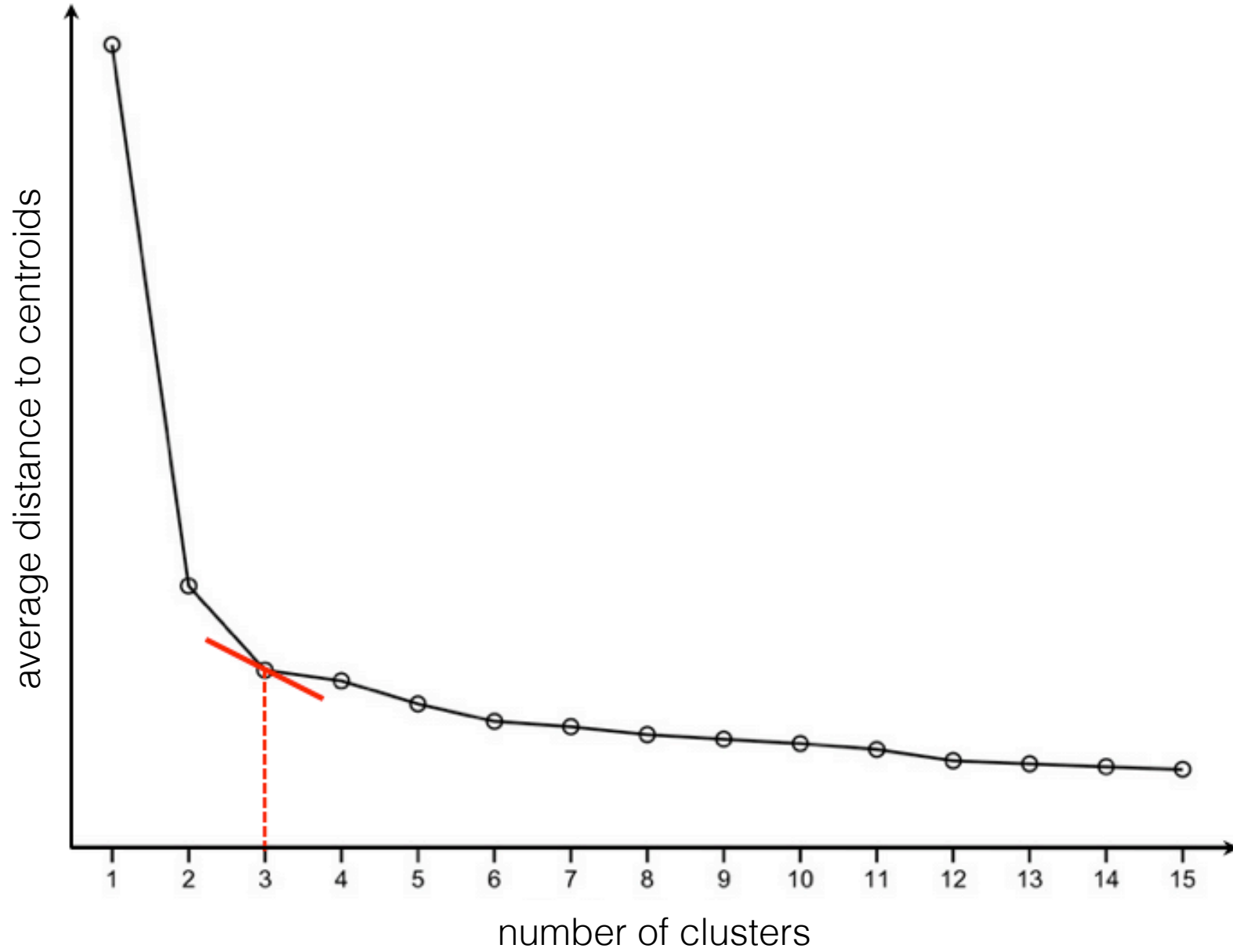












Clustering Challenges

Cluster description

should clusters be described using representative instances or average values?

Model validation

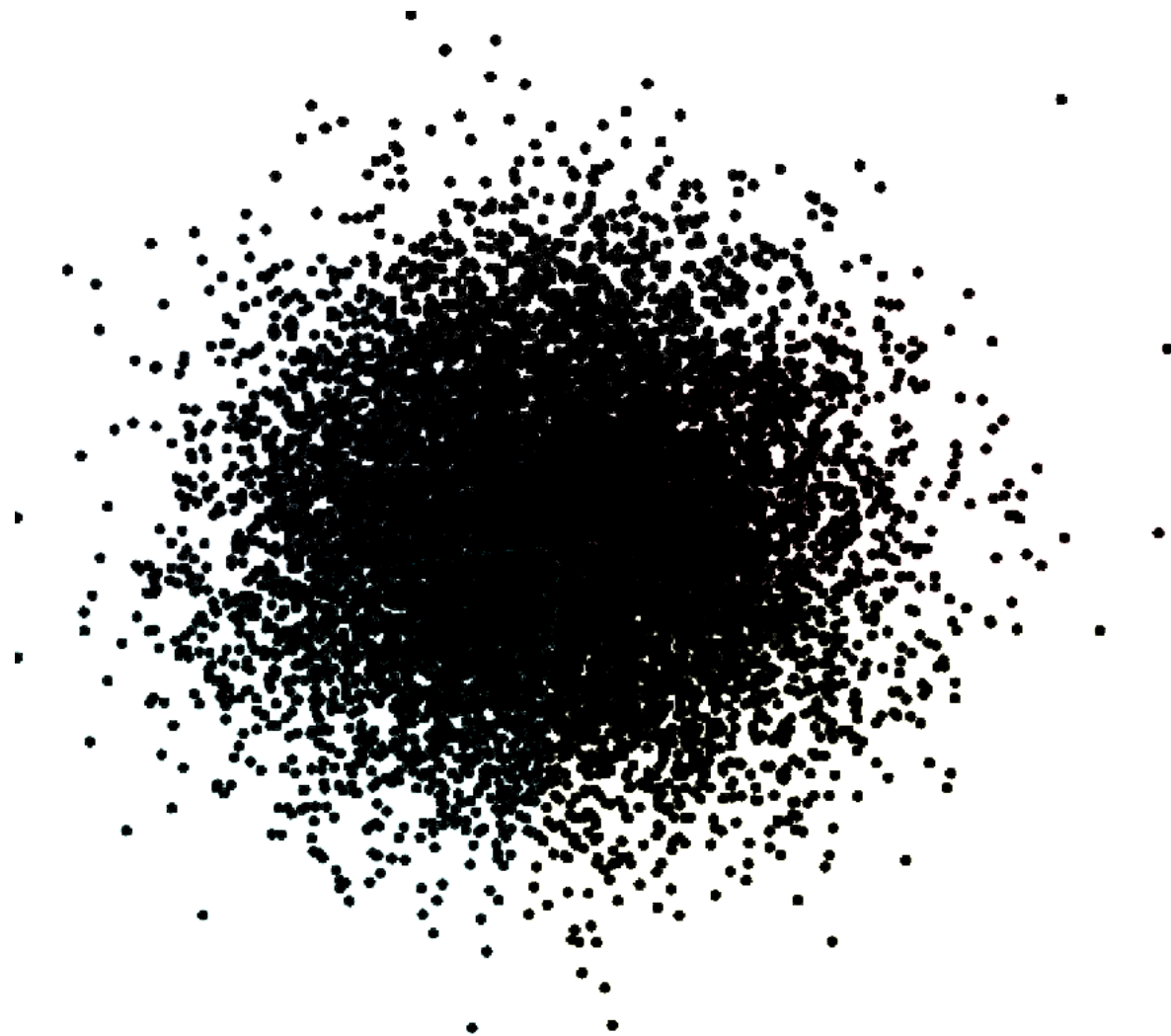
no true clustering information against which to contrast the clustering scheme, so how do we determine if it is appropriate?

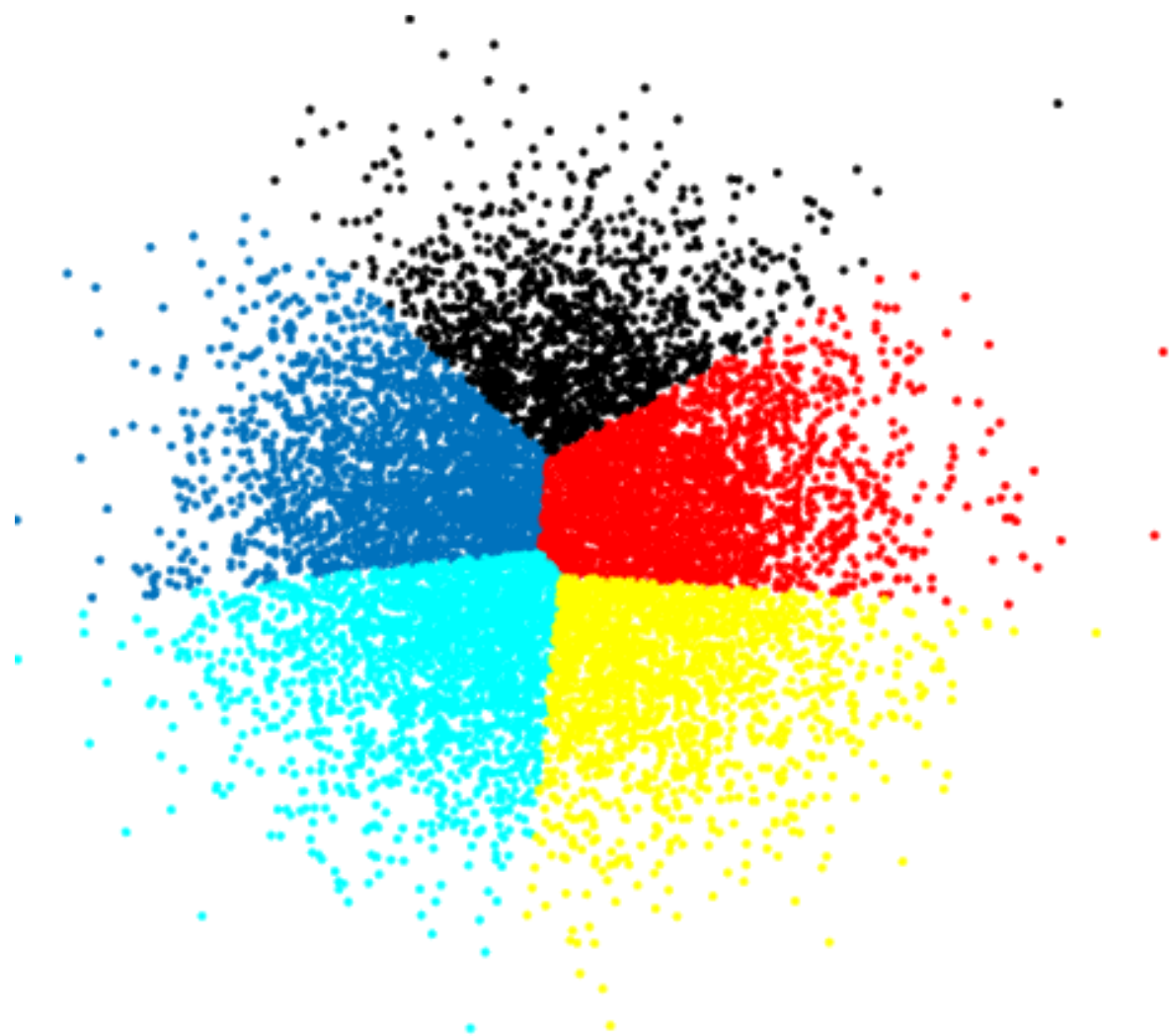
Ghost clustering

most methods will find clusters even if there are none in the data

***A posteriori* rationalization**

once clusters have been found, it is tempting to try to "explain" them ...





Take-Away: clustering looks easy in 2D or 3D spaces... but in high-dimensional spaces, **almost all pairs are equidistant!**

Data science students don't have to be gardeners, but it helps.

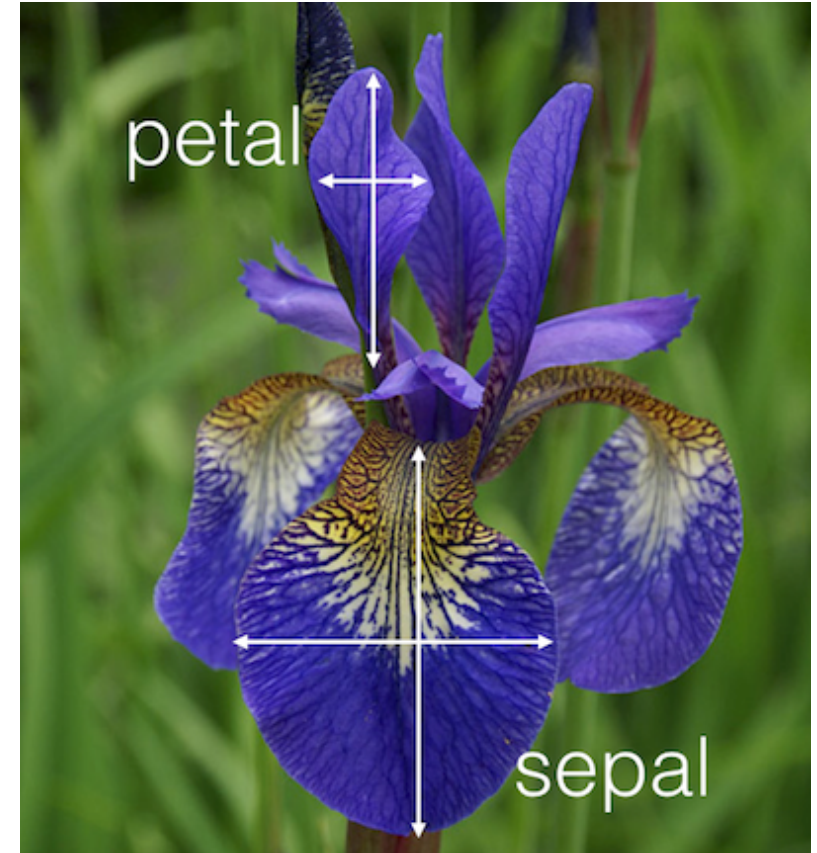
(unknown)

Example – Iris Dataset

Iris is a genus of plants with showy flowers.

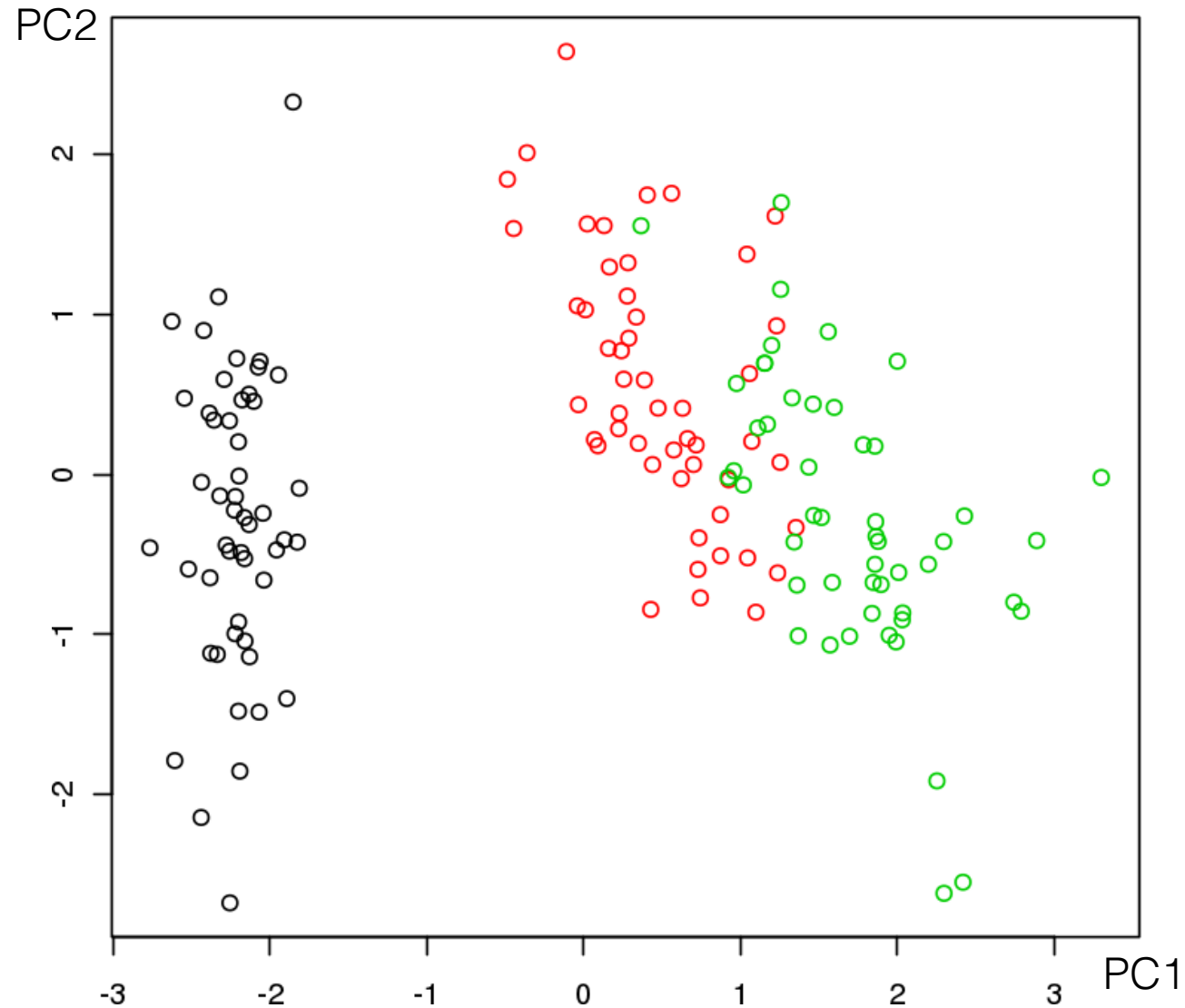
Fisher's iris dataset contains 150 observations of 5 attributes for specimens collected by Anderson, mostly from a Gaspé peninsula's pasture in the 1930s:

- **petal width**
- **petal length**
- **sepal width**
- **sepal length**
- **species**

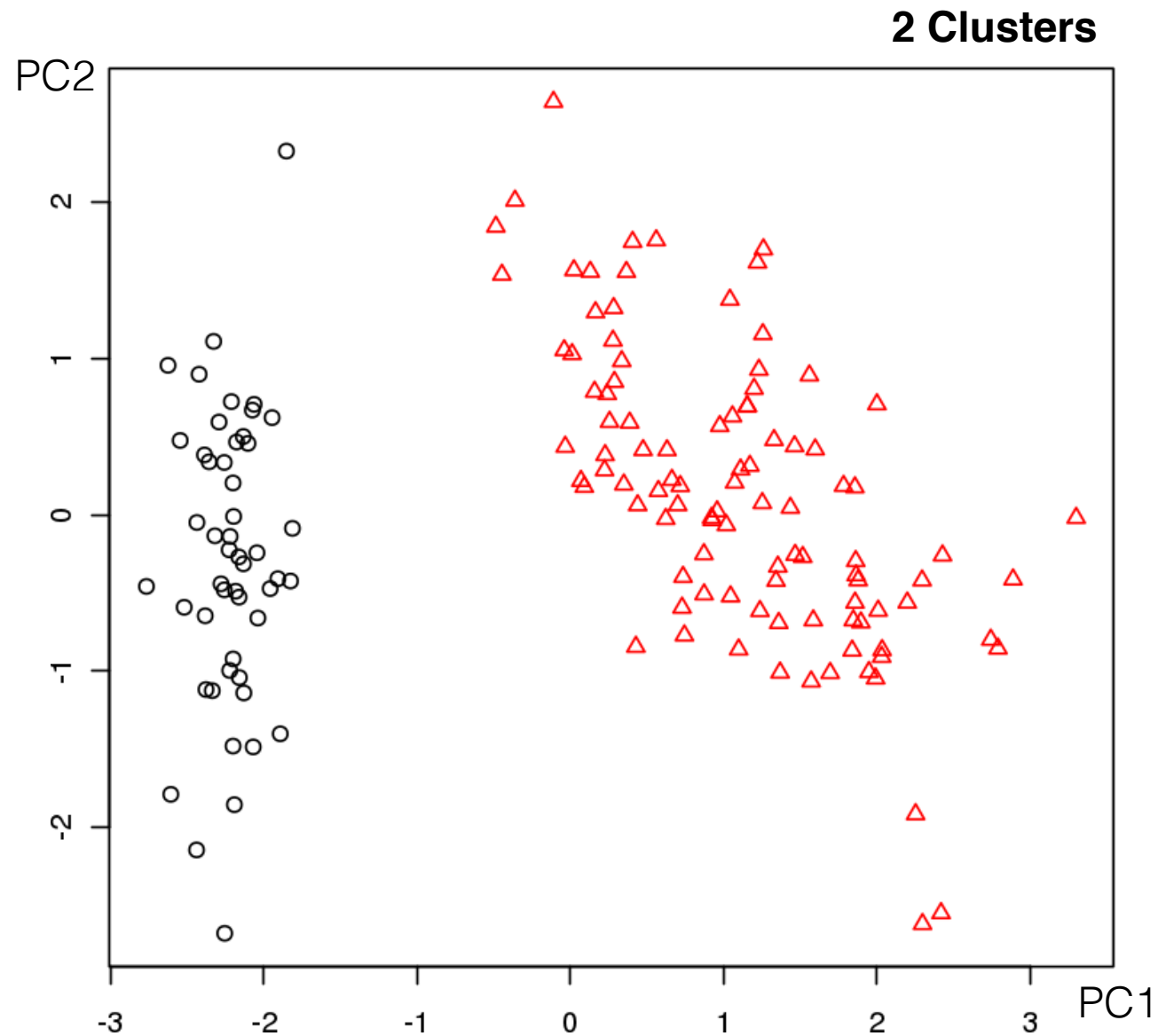


Example – Iris Dataset

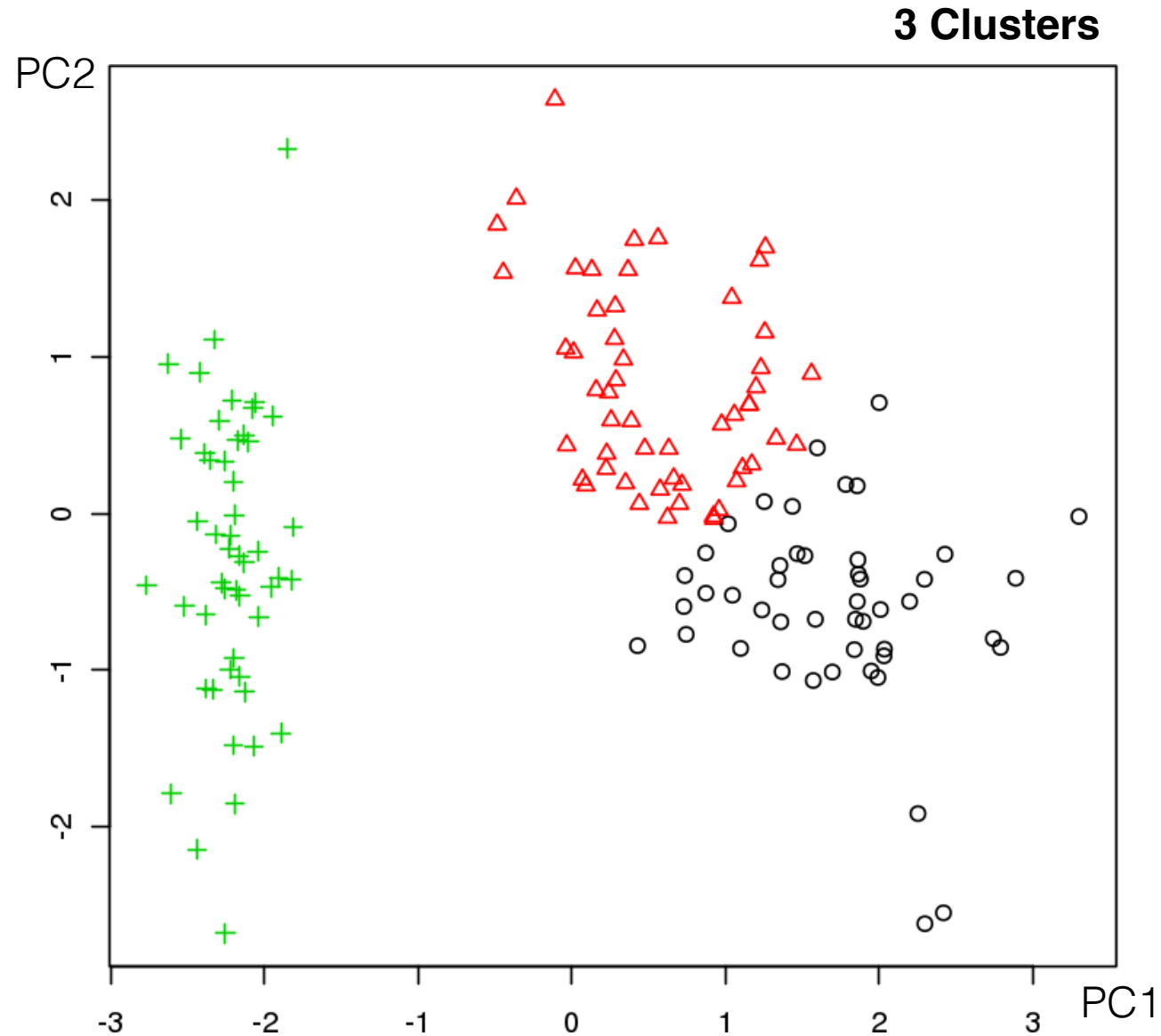
Iris Classification



Example – Iris Dataset

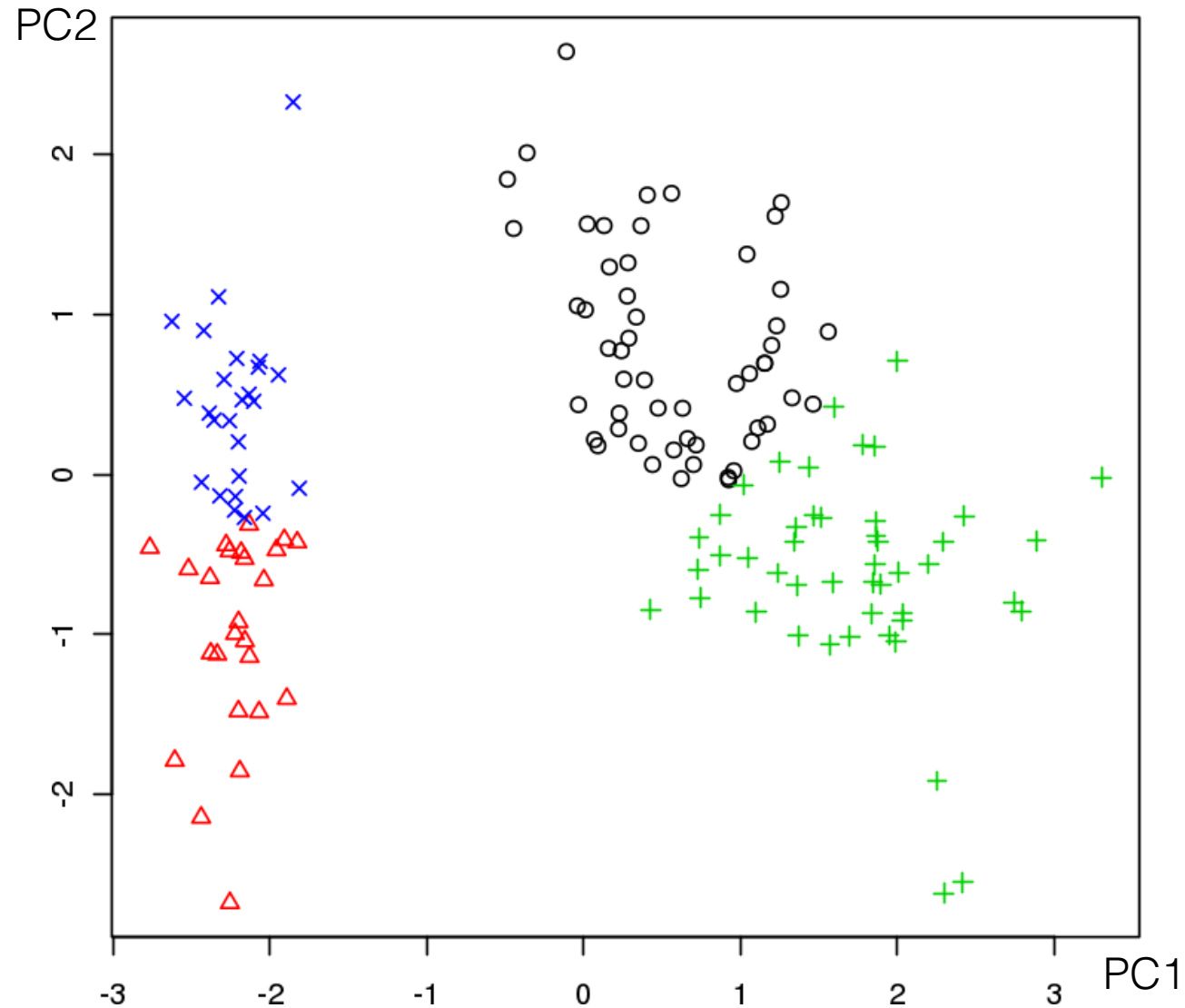


Example – Iris Dataset



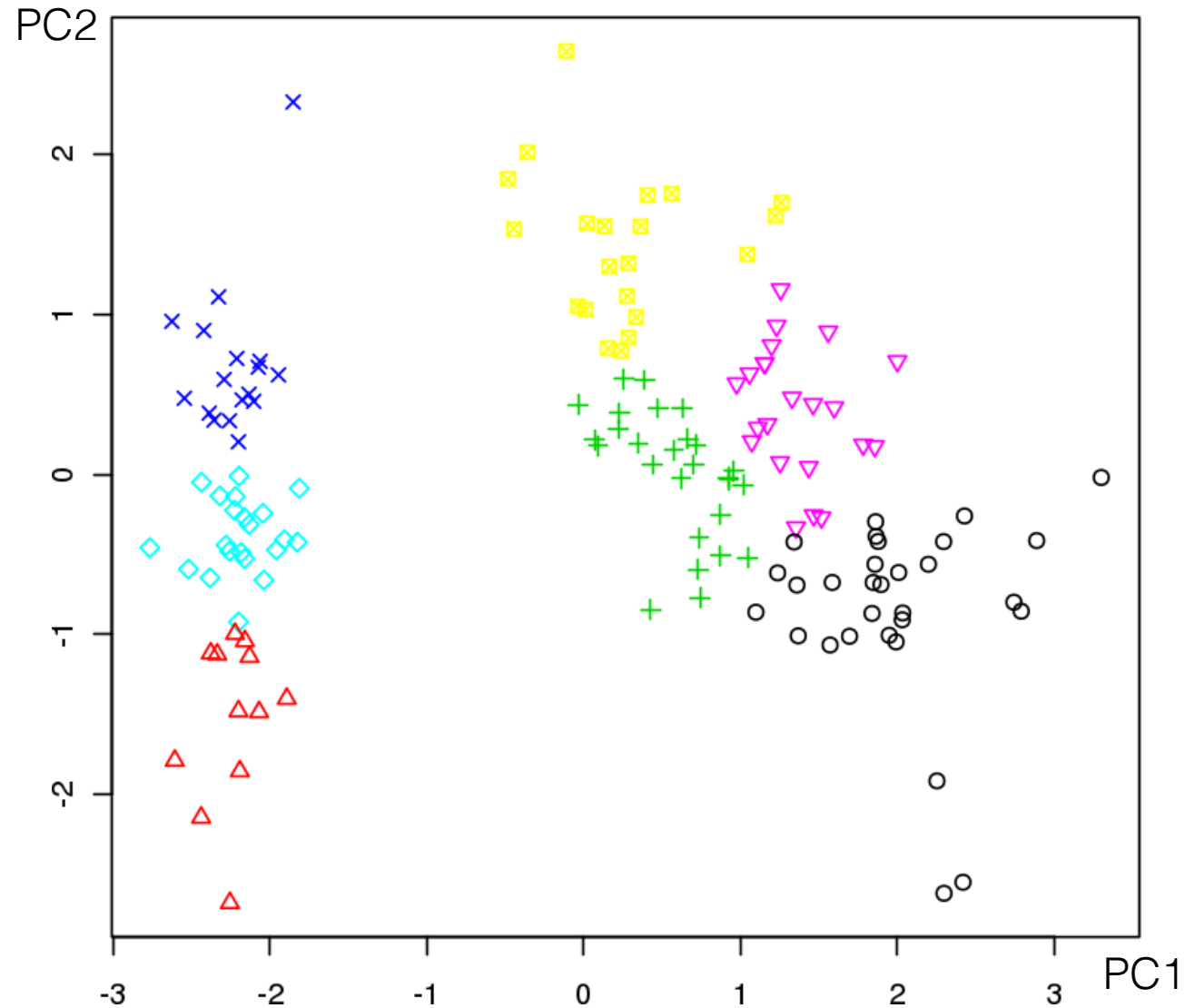
Example – Iris Dataset

4 Clusters



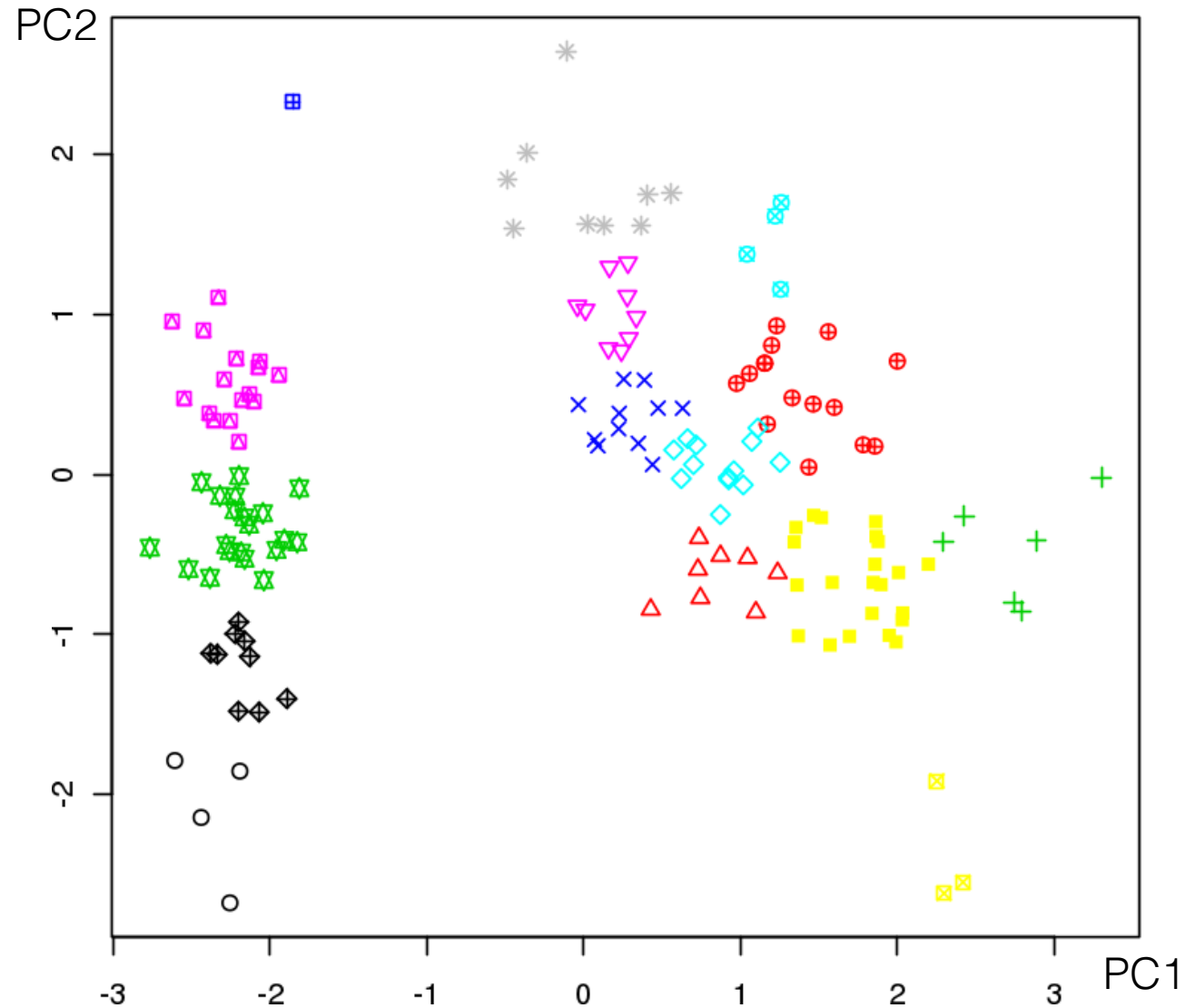
Example – Iris Dataset

7 Clusters



Example – Iris Dataset

15 Clusters



Clustering Validation

What does it mean for a clustering scheme to be **better** than another?

What does it mean for a clustering scheme to be **valid**?

What does it mean for a single cluster to be good?

How many clusters are there in the data, really?

Main challenge: what are we comparing the clustering scheme **against**? (versions of this problem plague unsupervised tasks)

Take-Away: right vs. wrong/good vs. bad is meaningless.

Optimal vs. sub-optimal is the way to go.

Clustering Validation

Optimal clustering scheme:

- maximal separation between clusters
- maximal similarity within groups
- agrees with human eye test
- useful at achieving its goals

Validation types

- external (uses additional information)
- internal (uses only the clustering results)
- relative (compares across clustering attempts)

Internal Clustering Validation

Davies-Bouldin Index can be used to determine the number of clusters in k -means

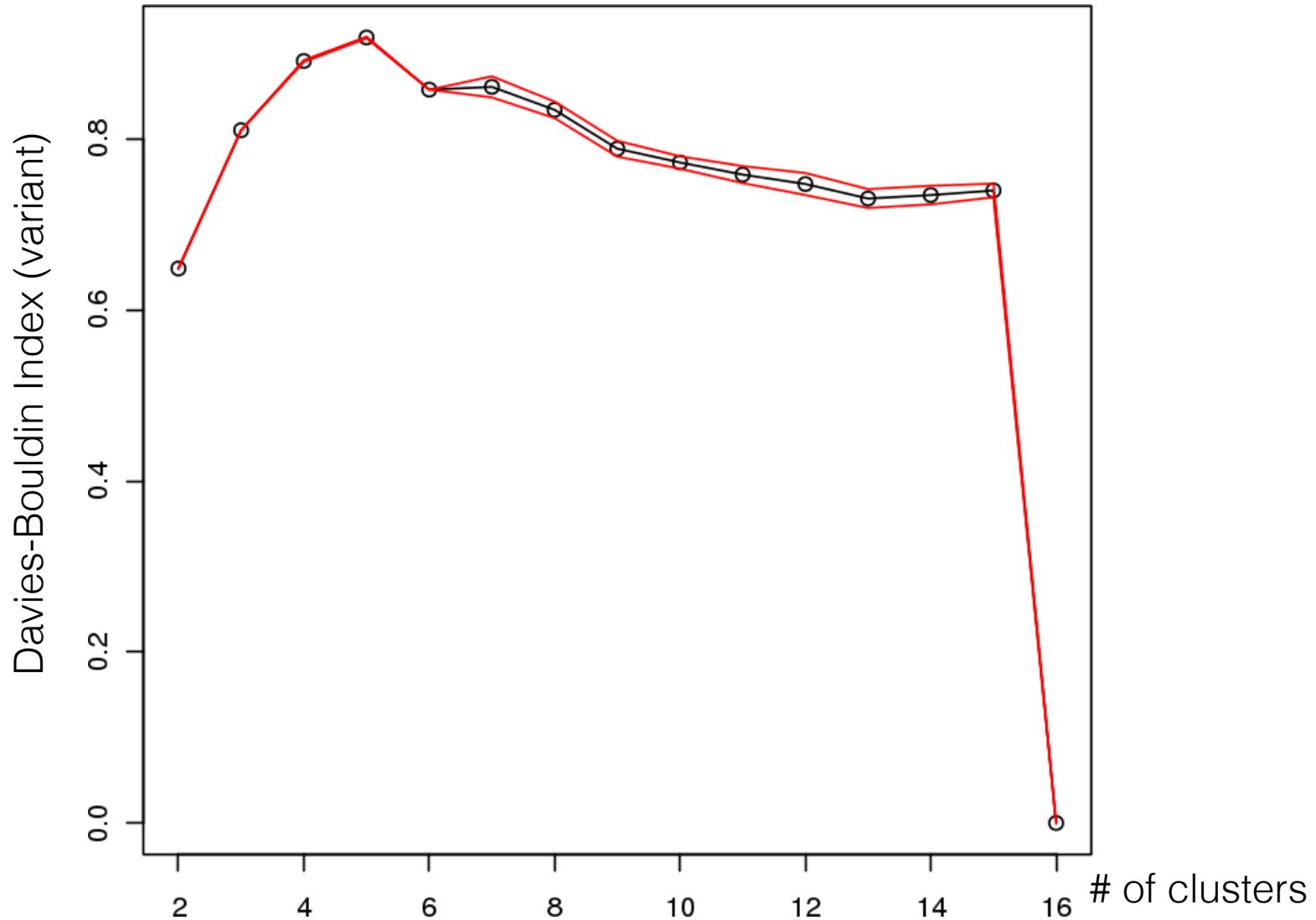
$$DB = \frac{1}{N} \sum_{i=1}^N \max_{j \neq i} \frac{s_i + s_j}{d(c_i, c_j)},$$

where N is the number of clusters, c_m is the centroid of the m^{th} cluster, and s_m is the average distance of the points in the m^{th} cluster to c_m

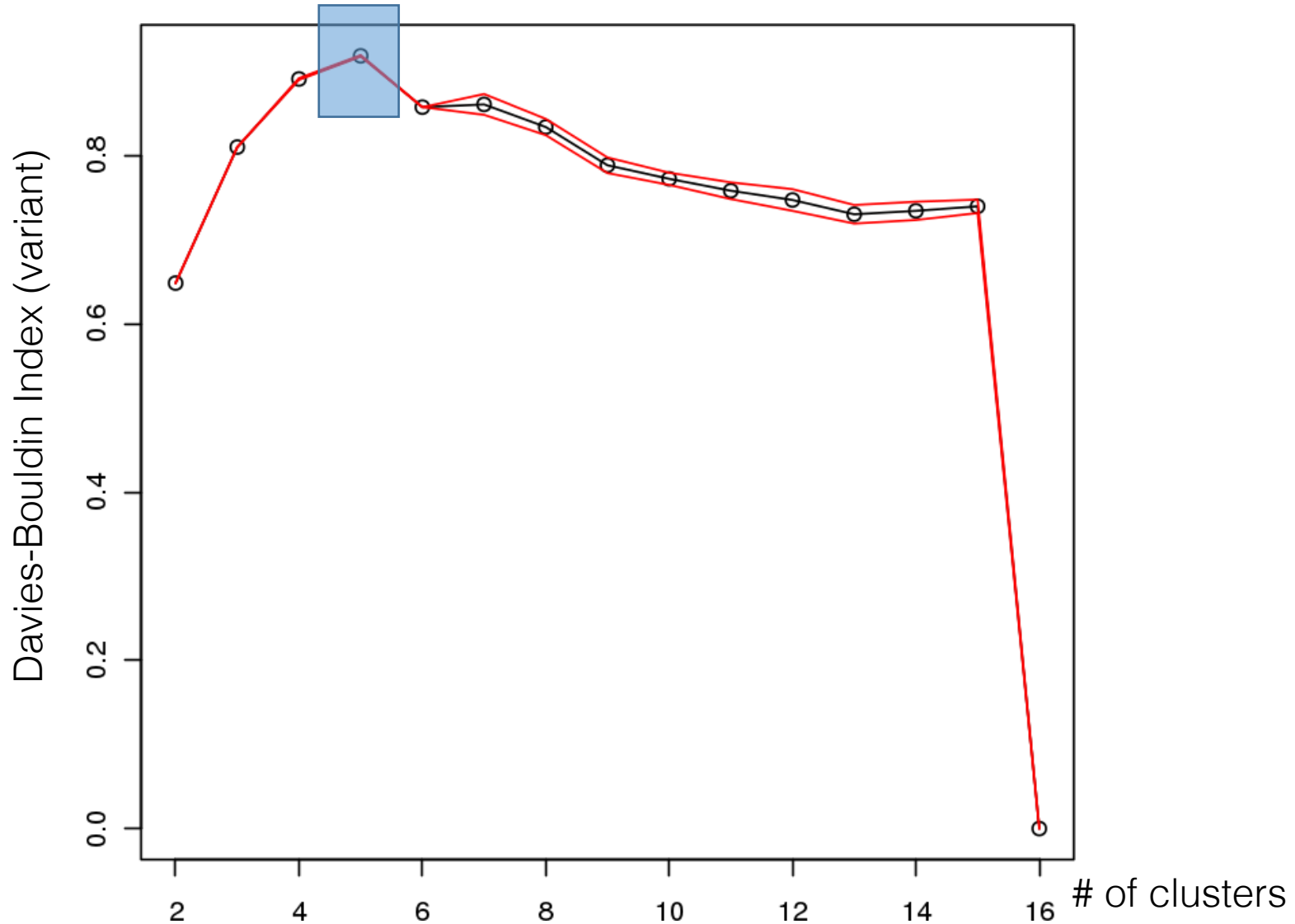
Other Methods

- Sum of Squared Errors
- Dunn's Index
- Silhouette Metric
- etc.

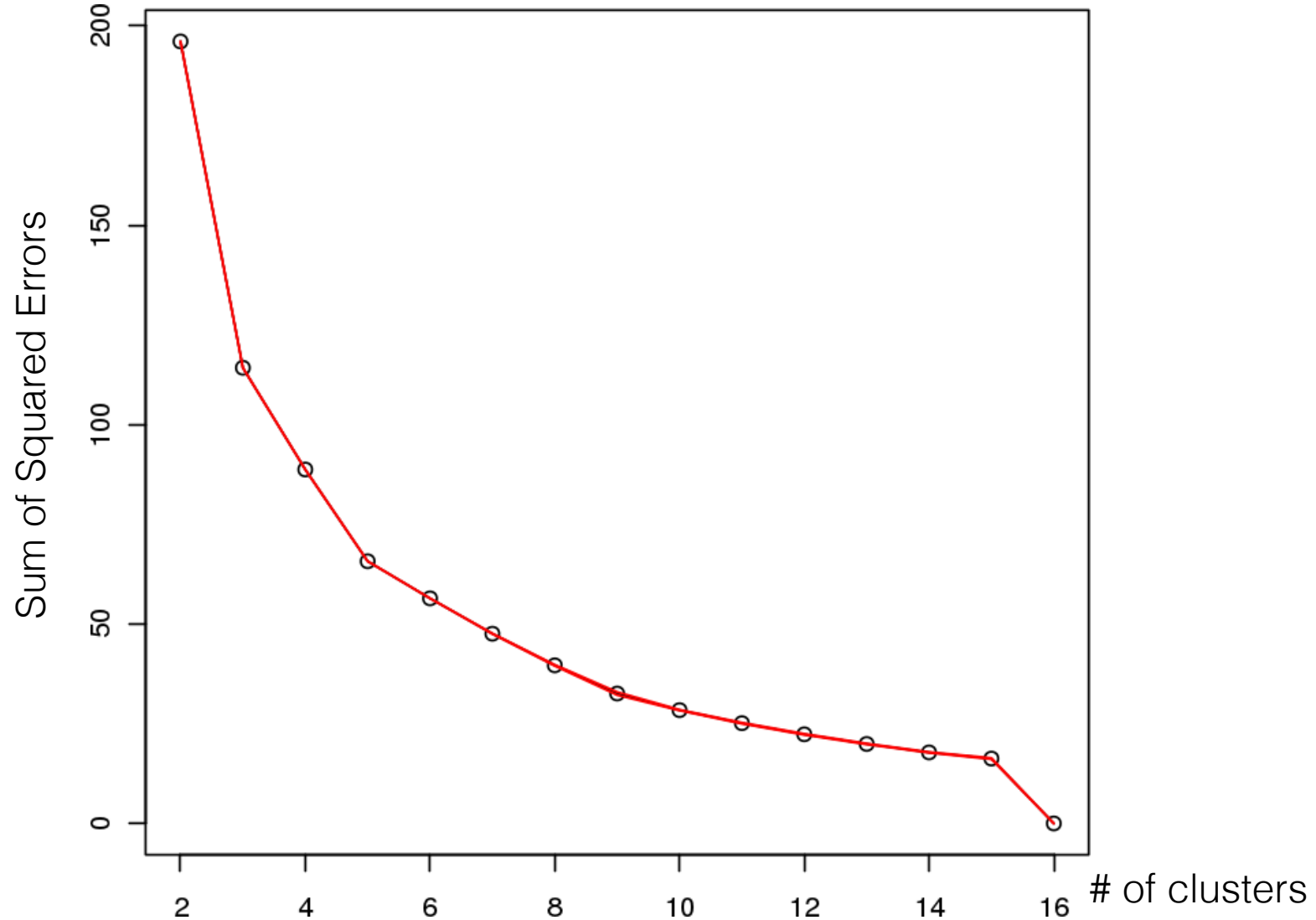
Example – Iris Dataset (revisited)



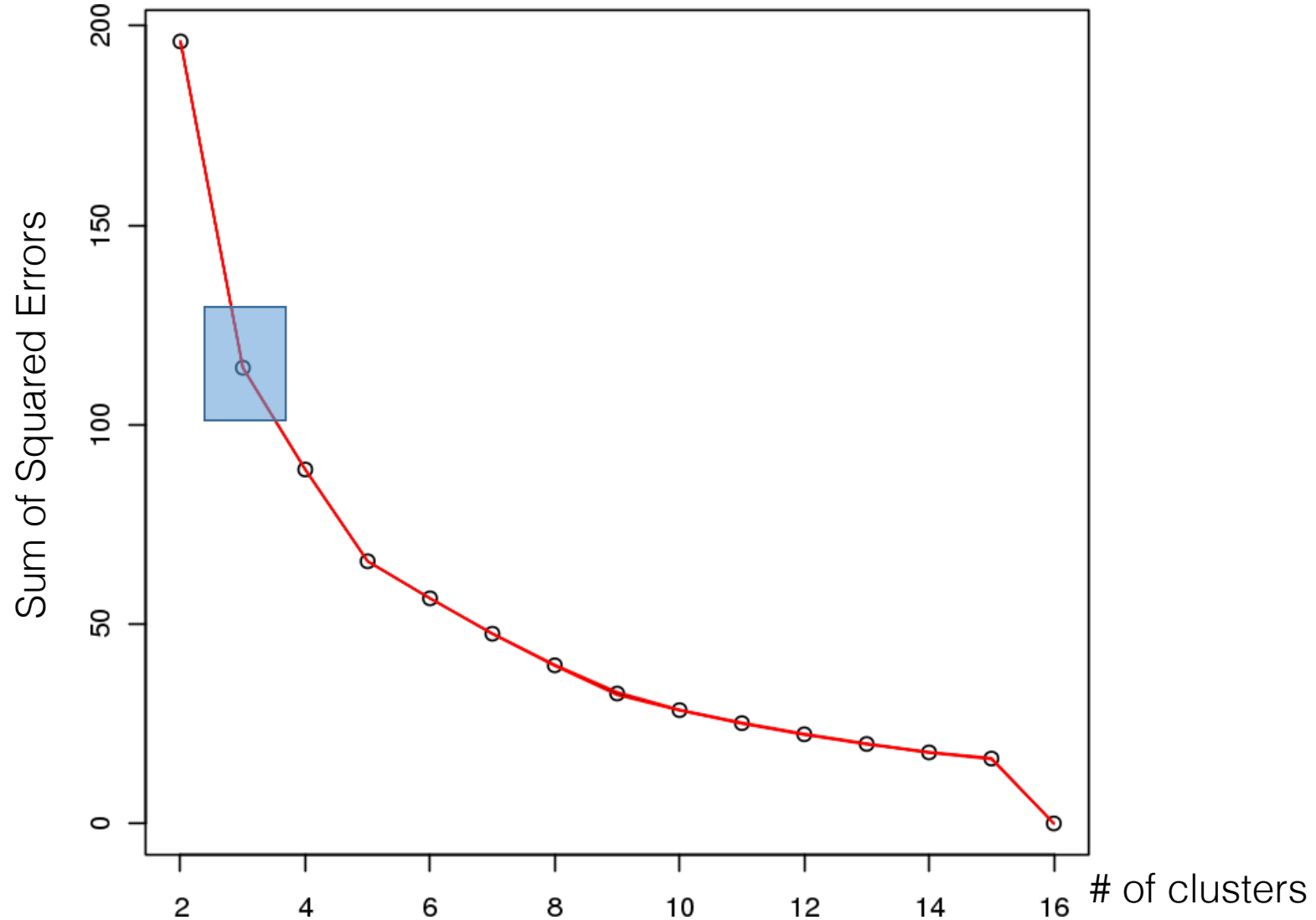
Example – Iris Dataset (revisited)



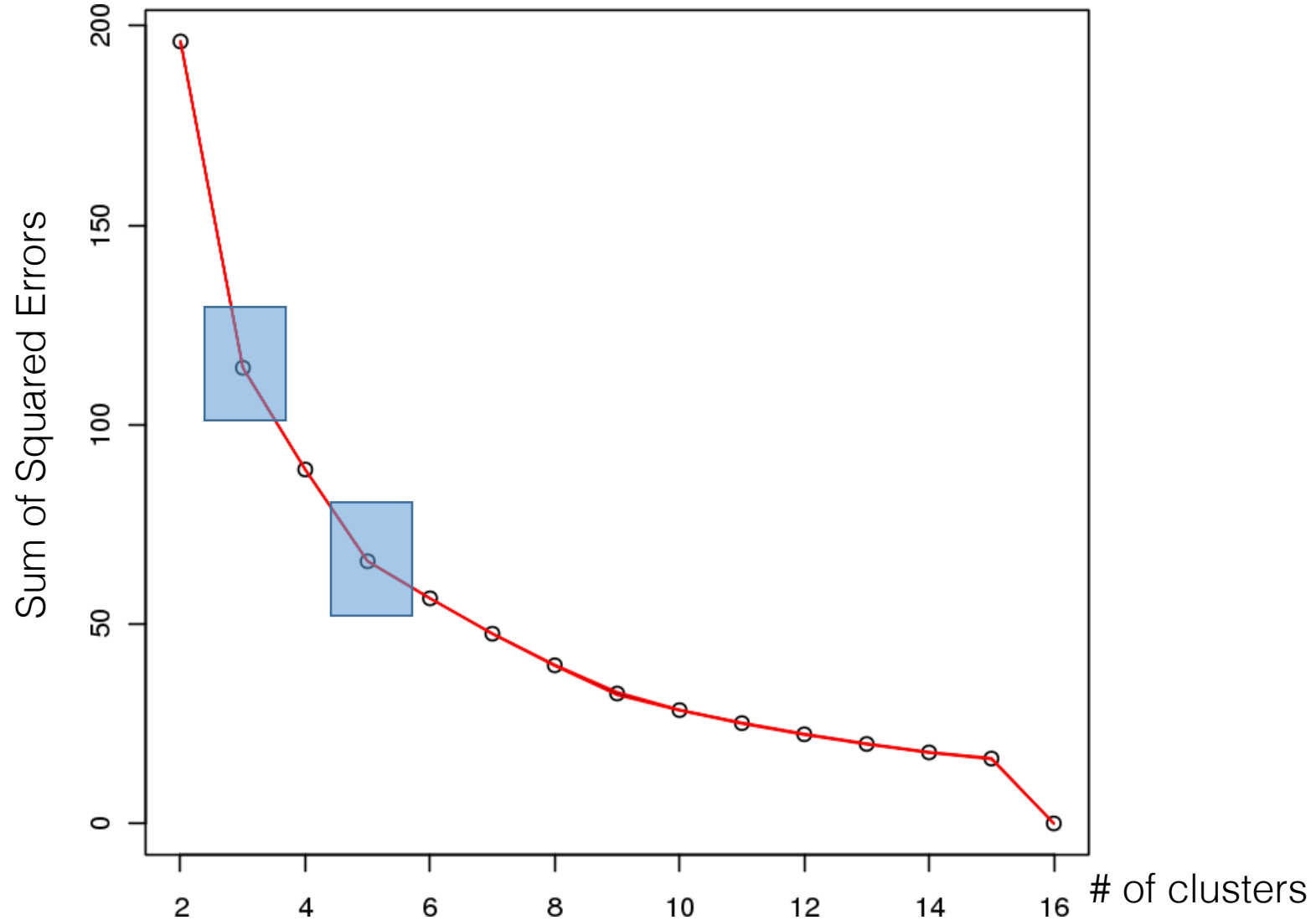
Example – Iris Dataset (revisited)



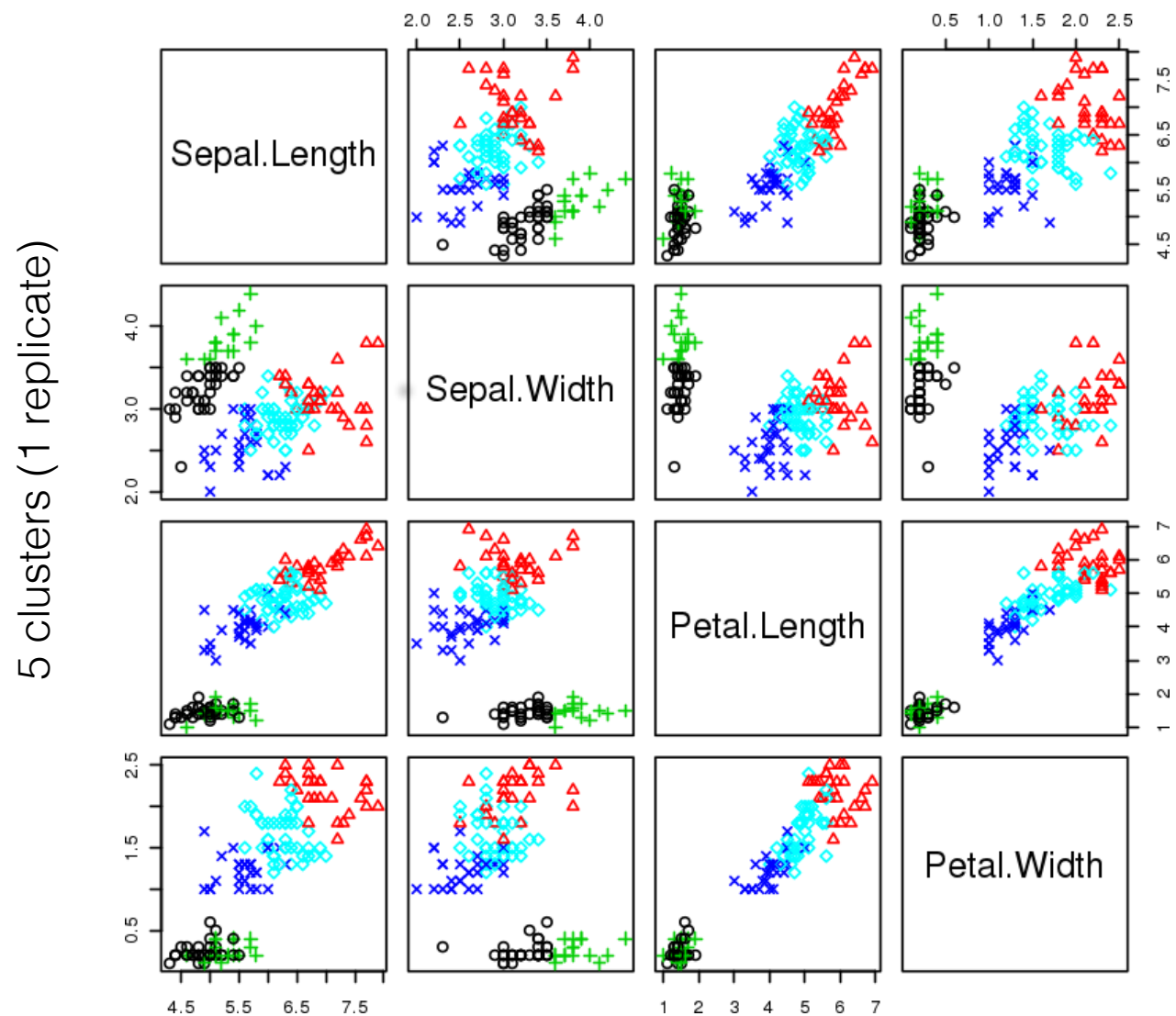
Example – Iris Dataset (revisited)



Example – Iris Dataset (revisited)



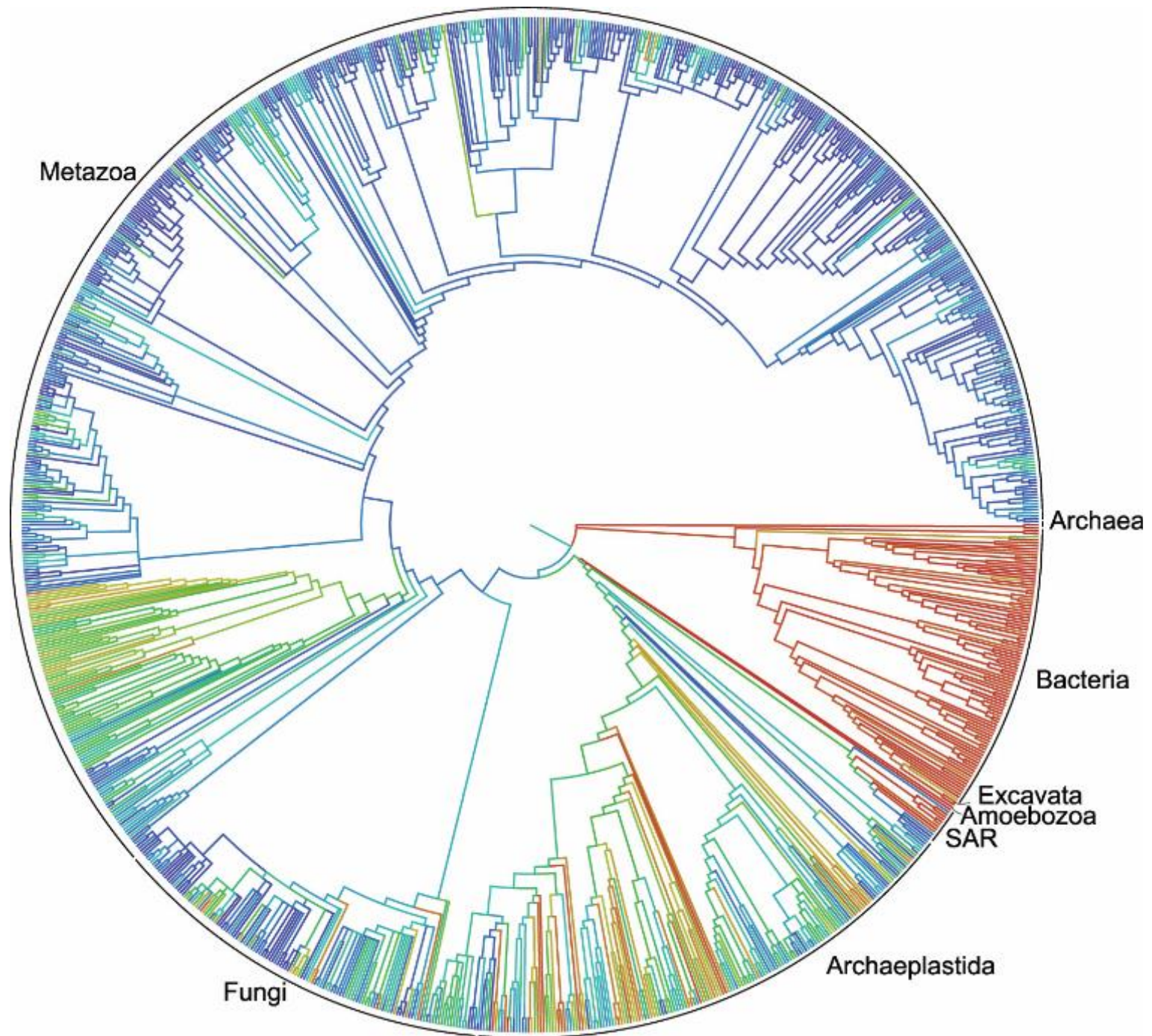
Example – Iris Dataset (revisited)



Take-Away: validating clusters is just as complicated as defining clusters.

We'll have more to say on the topic.

Hierarchical Clustering



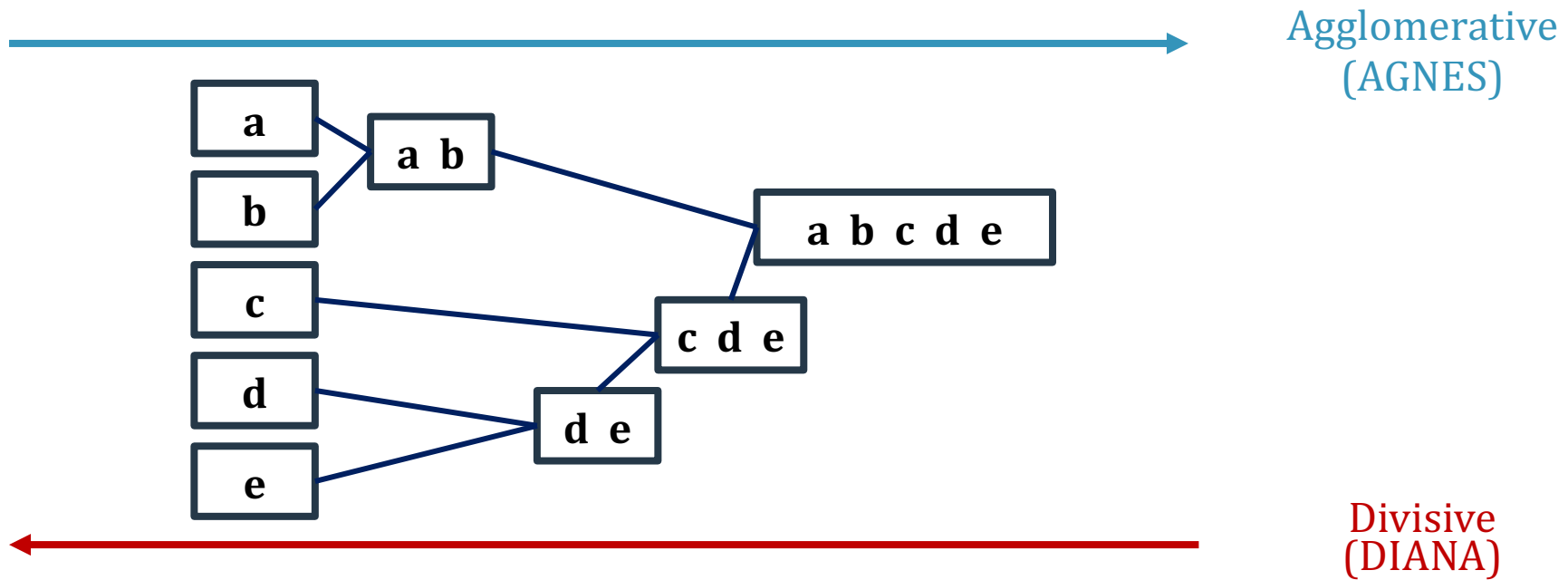
Hierarchical Clustering Overview

Hierarchical clustering (HC) clusters a dataset into a **hierarchy** of clusters (order relation is set containment).

There are two main strategies:

- **Bottom-up** (agglomerative)
initially, each observation starts in its own **separate** cluster
clusters are **merged** as the hierarchy is climbed
after the last merge, all observations are in the **same** cluster
- **Top-down** (divisive)
initially, each observation starts in the **same** cluster
clusters are **split** as the hierarchy is descended down
after the last split, each observation ends in its own **separate** cluster

Bottom-up HC is significantly faster than Top-down HC (poly. vs. exp.)



Hierarchical Clustering Overview

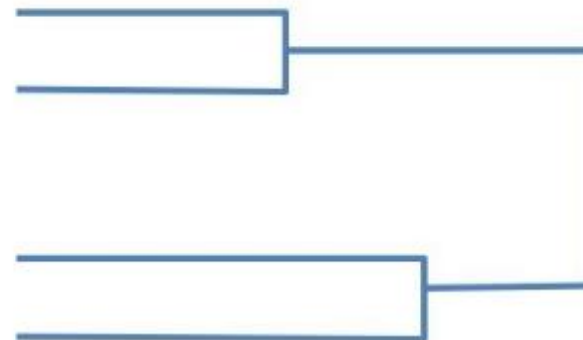
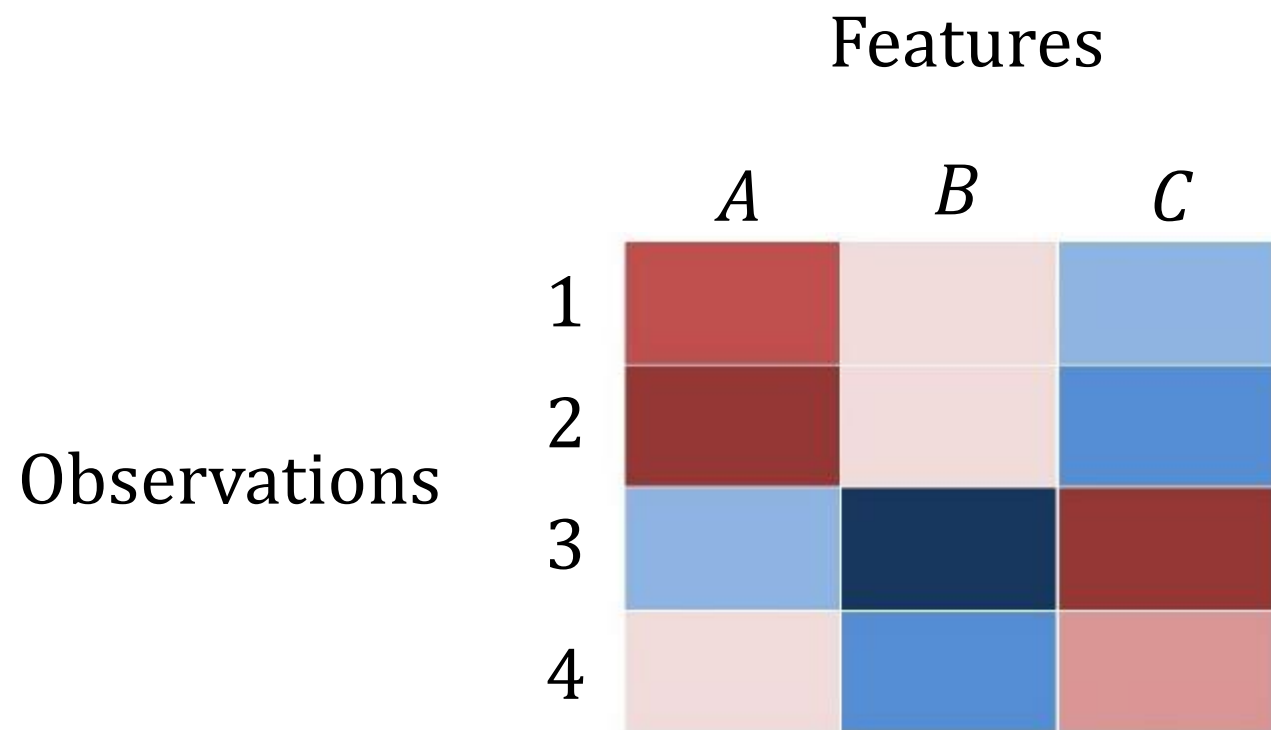
The main question: how to split, or how to merge, clusters?

This requires the notion of a distance between clusters (**linkage**).

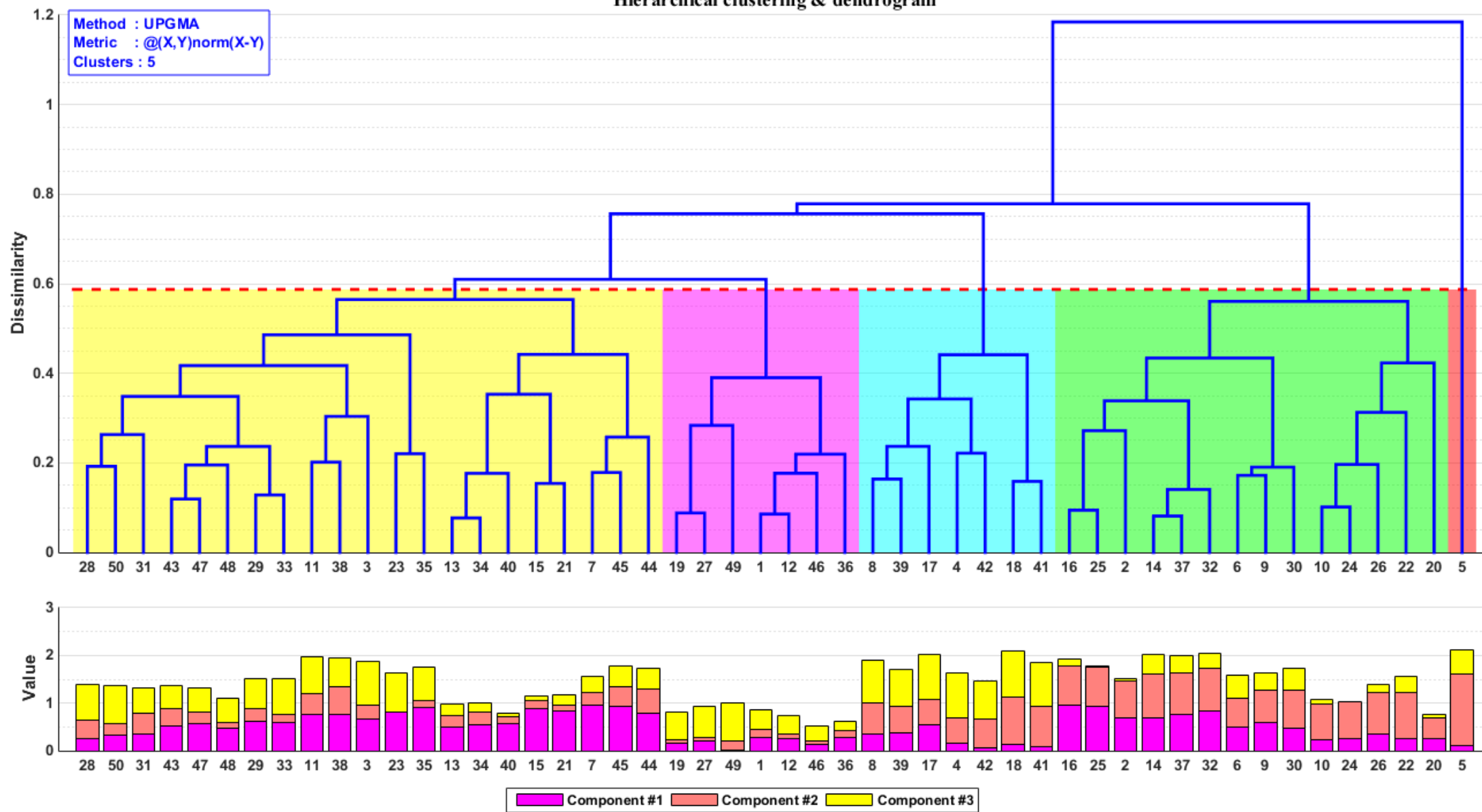
- in Bottom-up HC, **nearest pairs** of clusters are merged up the hierarchy (requires only computing distances between pairs)
- in Top-down HC, a cluster must be **optimally split** into sub-clusters down the hierarchy (much harder, computationally)

Another issue: at what level do we **report** the clustering scheme?
When do we stop climbing or descending the hierarchy?

Latent class analysis might be a better approach, in general.



Hierarchical clustering & dendrogram



[author unknown]

Back to Knowledge Discovery

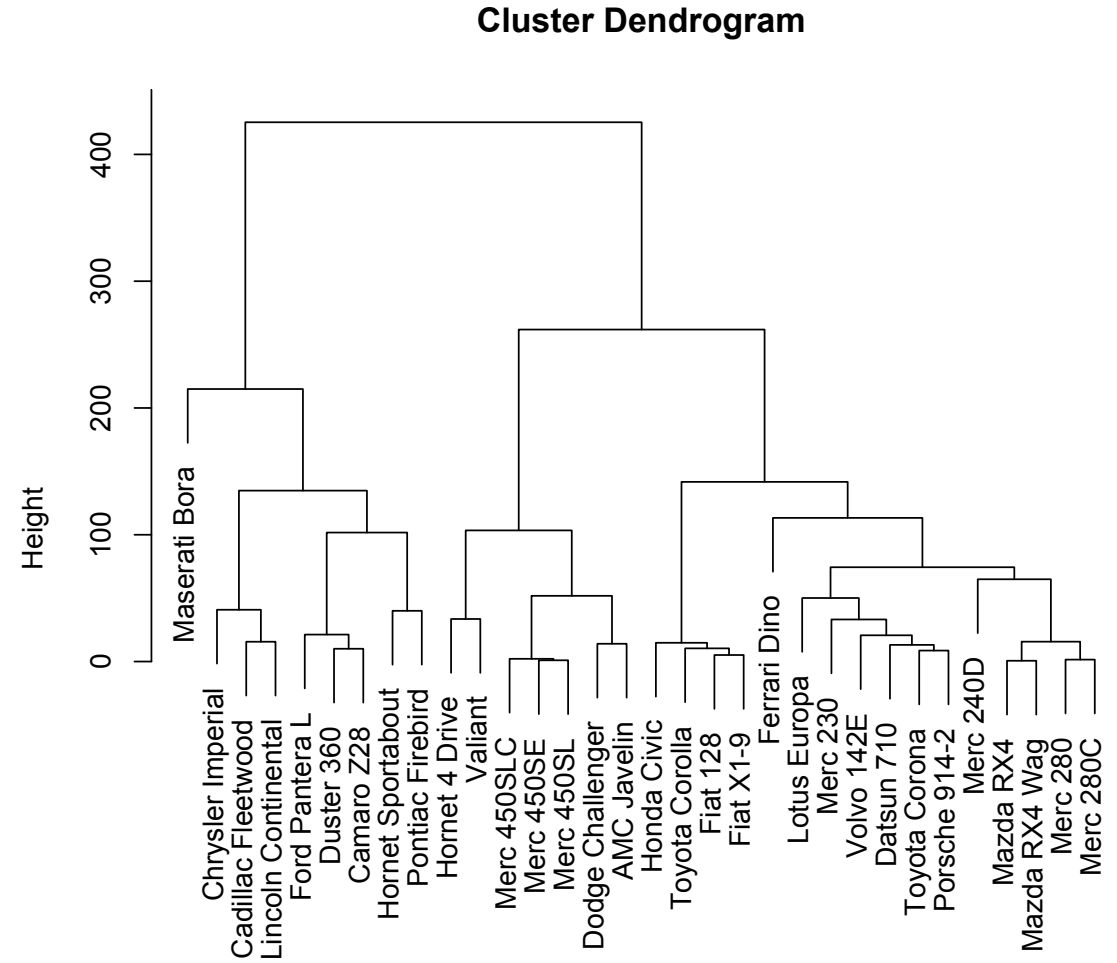
	mpg	cyl	displacement	horsepower	drat	weight	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4
Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4
Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1

More unsupervised learning:
what underlying structures can
we discover in this data?

[mtcars dataset]

Back to Knowledge Discovery

What do you notice in this diagram, structure-wise?

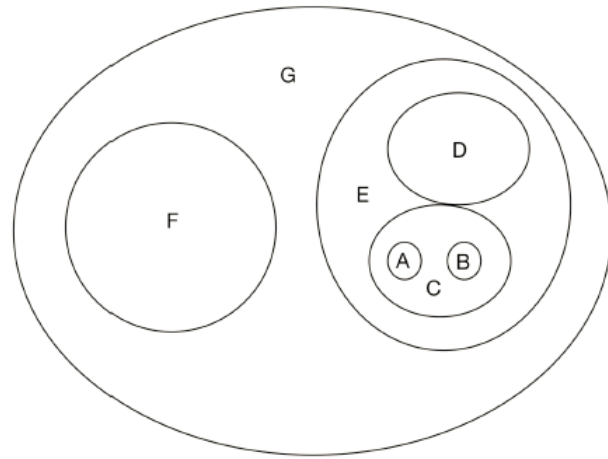
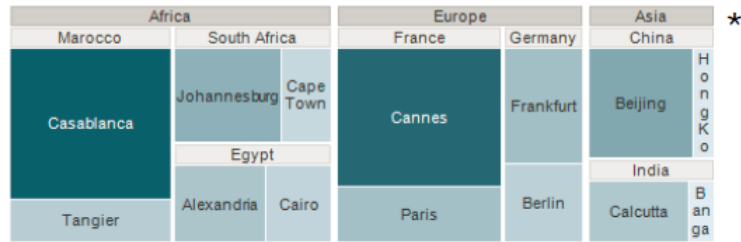


In a nutshell, **hierarchical systems** are ordered sets where elements and/or subsets are organized in a given relationship to one another, both among themselves and within the whole.

Relationships vary according to the field domain and type of system, but in general, we can describe them by the properties of elements and the laws that govern them (e.g., how they are shared and/or related).

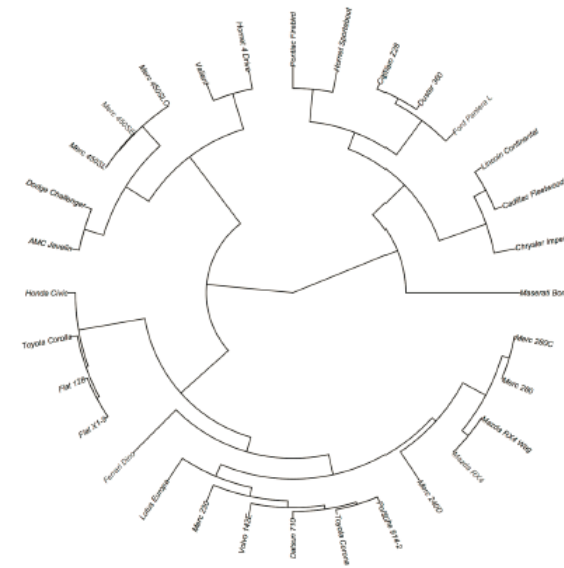
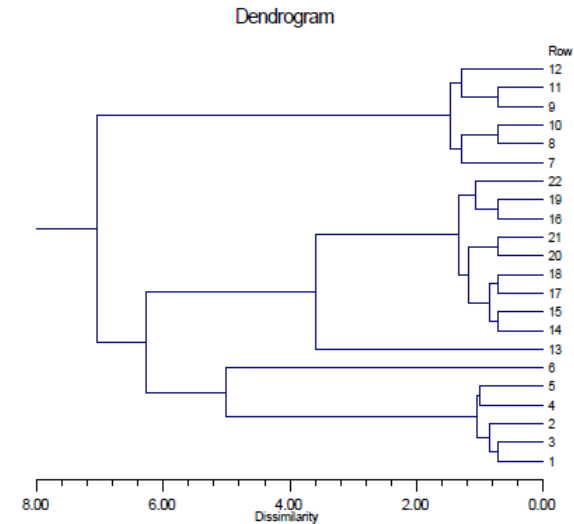
– I. Meireilles, *Design for Information*

Visualizing Hierarchy



(mammals(primates (apes(orangutan, human)), (monkeys(pygmy marmoset)), (lemurs(ruffed lemur))), (cetacea (whales(long-finned pilot whale, southern right whale)), (dolphins(striped dolphin, bottle-nose dolphin))))

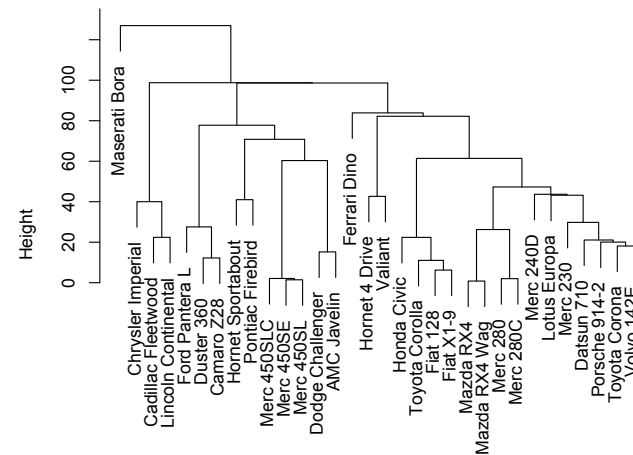
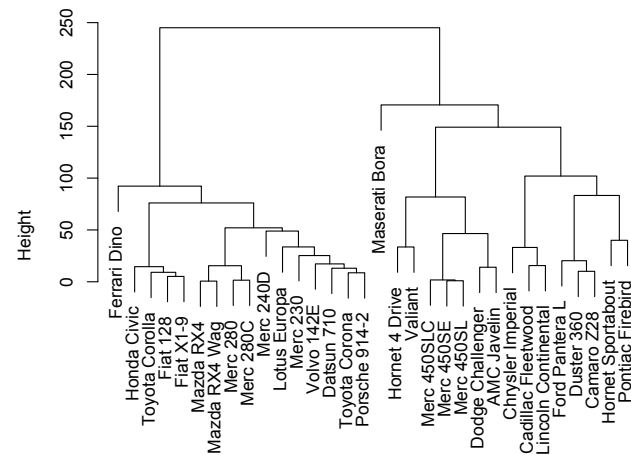
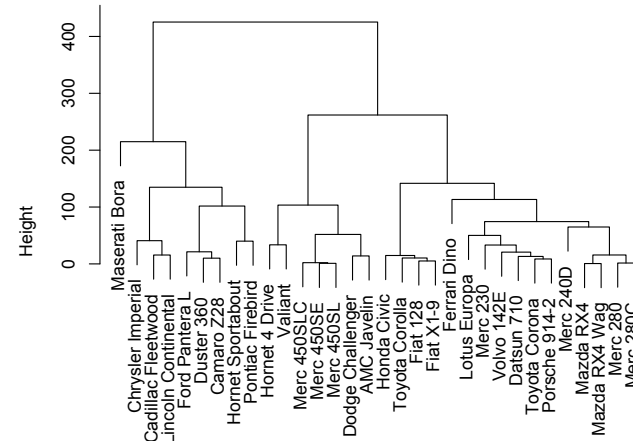
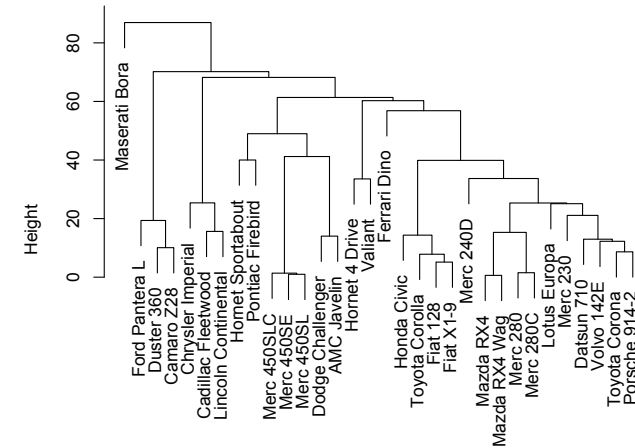
(*treemap from: TIBCO Spotfire Documentation: What is a Treemap?)



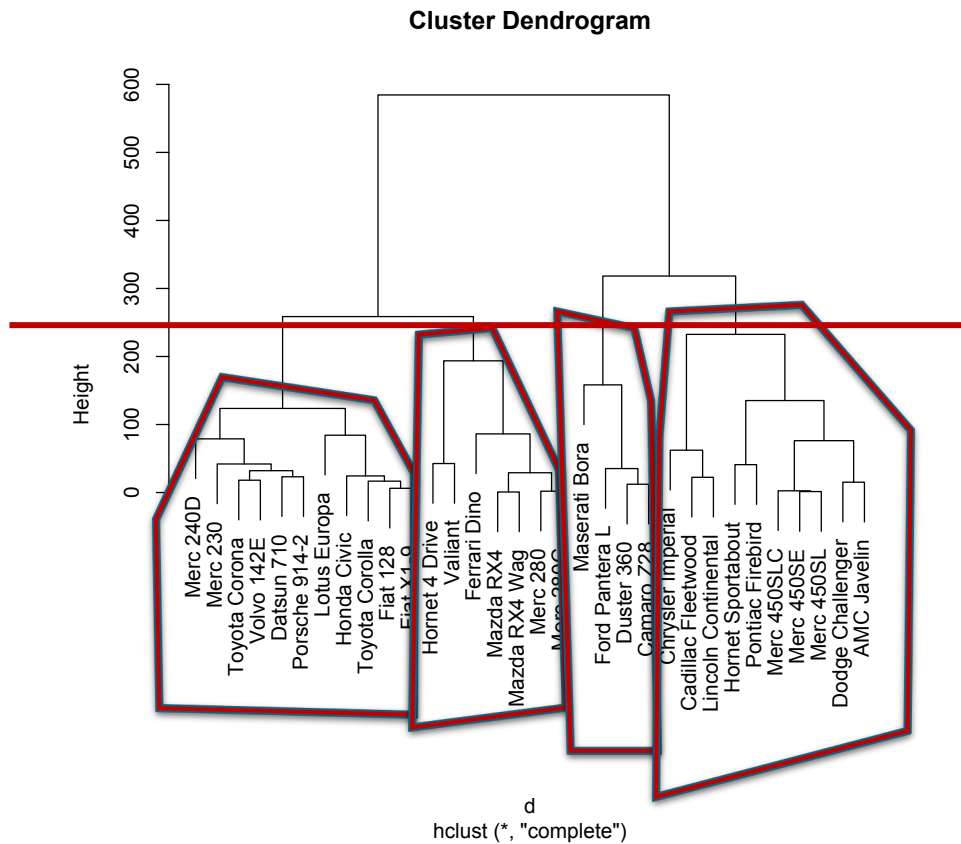
Button Press Hierarchical Clustering

What can we say about the data structure if presented with a **cluster dendrogram**?

Same data clustered using different parameter settings: which one is **optimal**?



Evaluating the Results

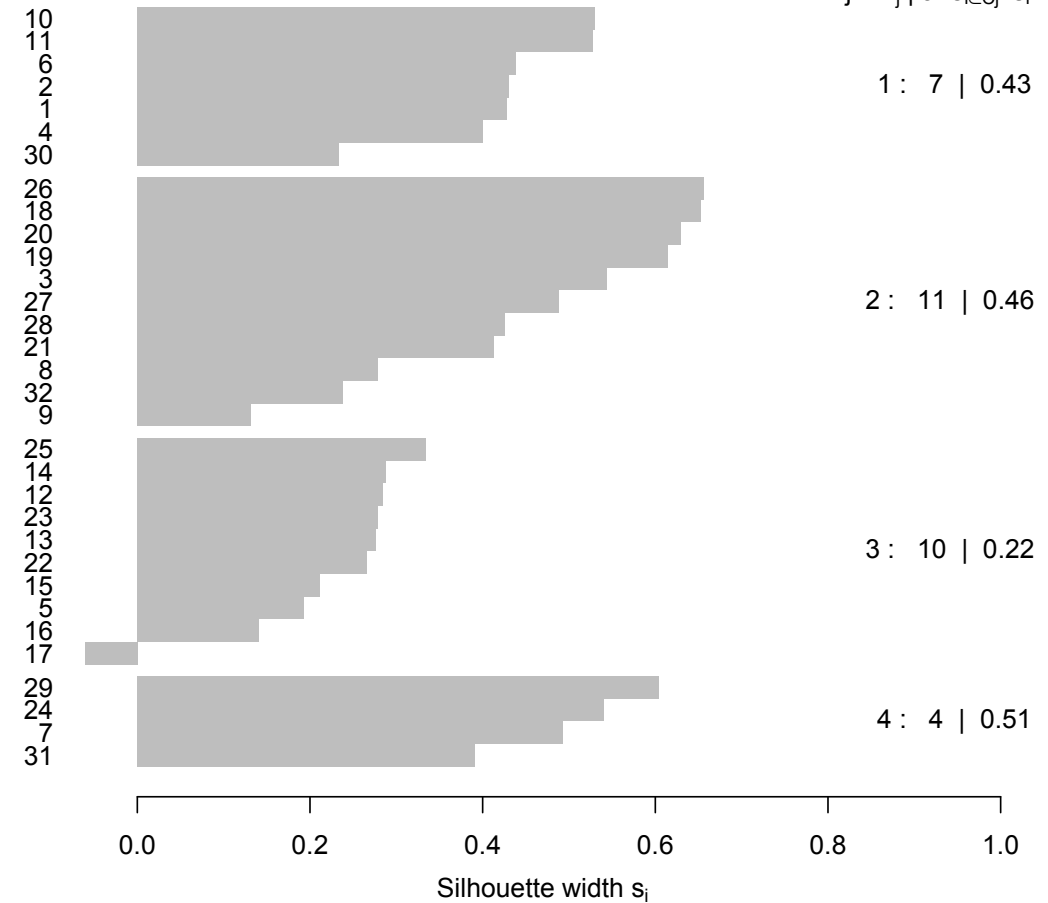


Silhouette plot of (x = cutree(hc, k = 4), dist = d)

n = 32

4 clusters C_j

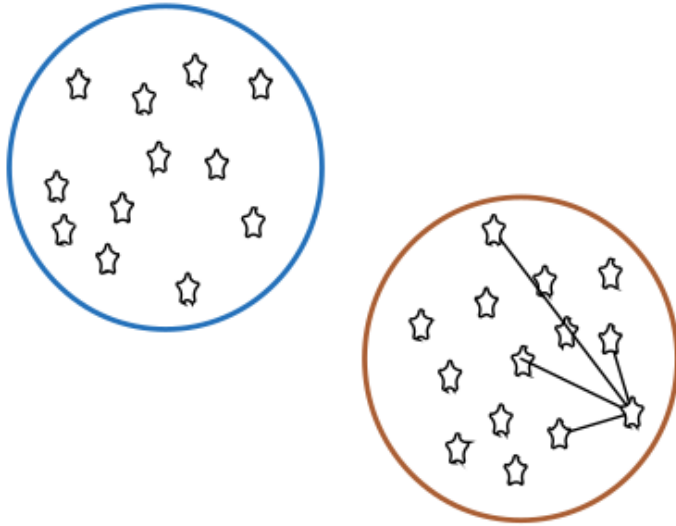
$j : n_j \mid \text{ave}_{i \in C_j} s_i$



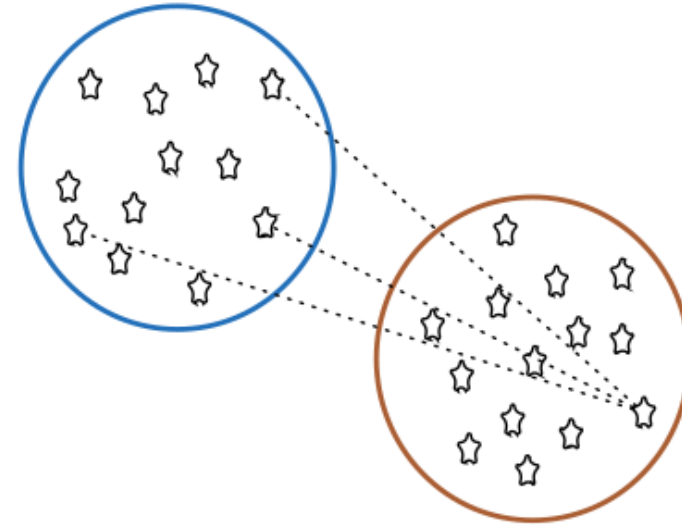
Average silhouette width : 0.38

The Silhouette Metric

average distance to points in
own cluster (**low is good**)

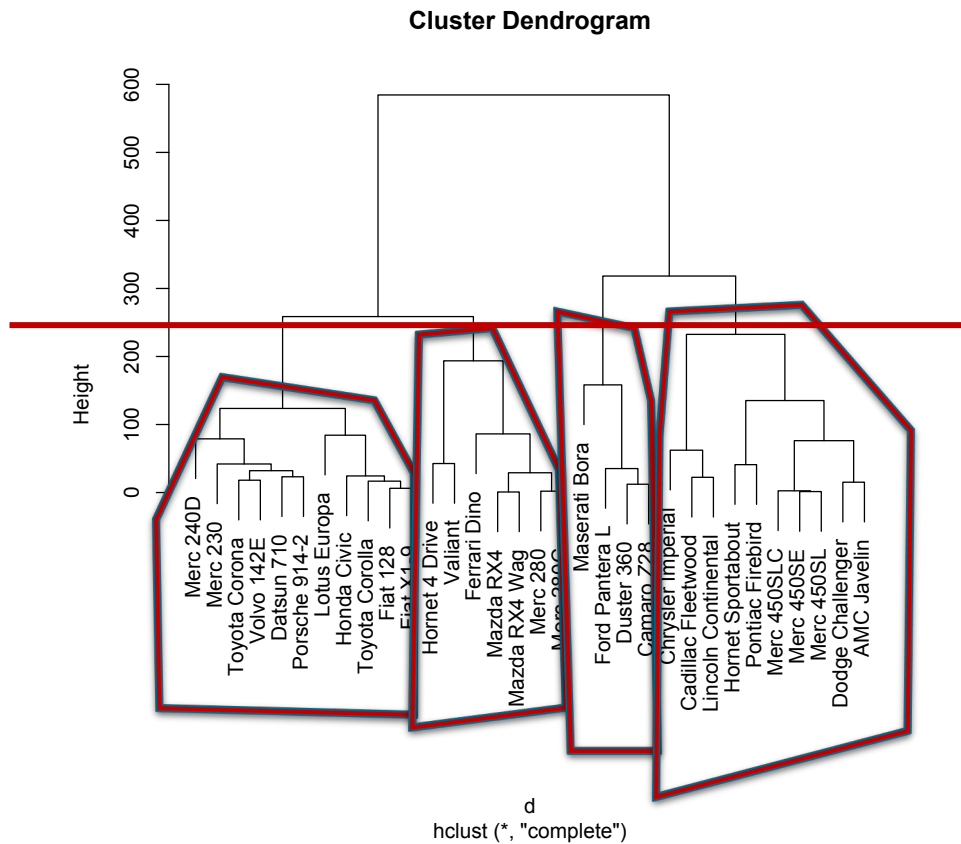


average distance to points in
neighbouring cluster (**high is good**)



$$\text{silhouette metric for a point} = \frac{\text{average dissimilarity with neighbouring cluster} - \text{average dissimilarity with own cluster}}{\text{maximum dissimilarity value (own or neighbour)}}$$

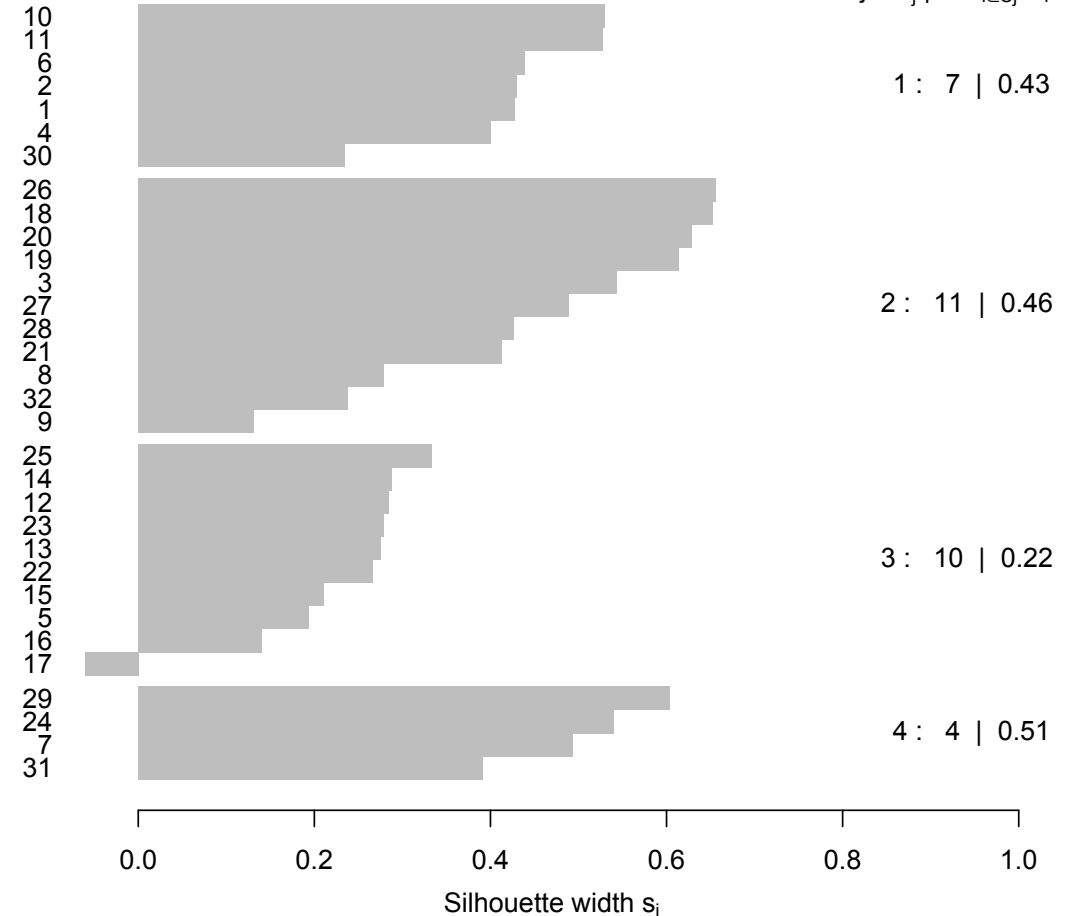
Evaluating the Results



Silhouette plot of (x = cutree(hc, k = 4), dist = d)

n = 32

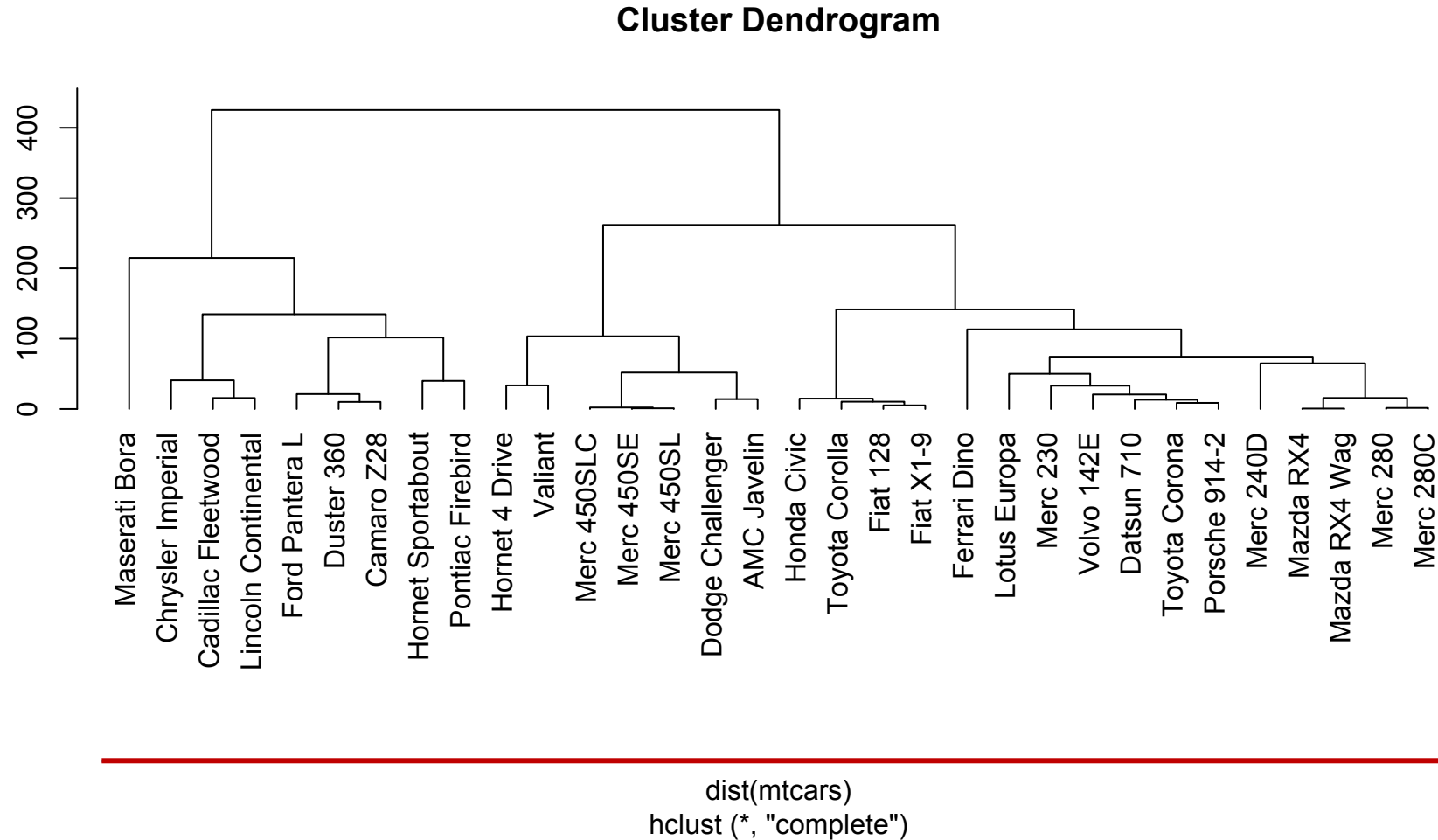
4 clusters C_j
 $j : n_j \mid \text{ave}_{i \in C_j} s_i$



Average silhouette width : 0.38

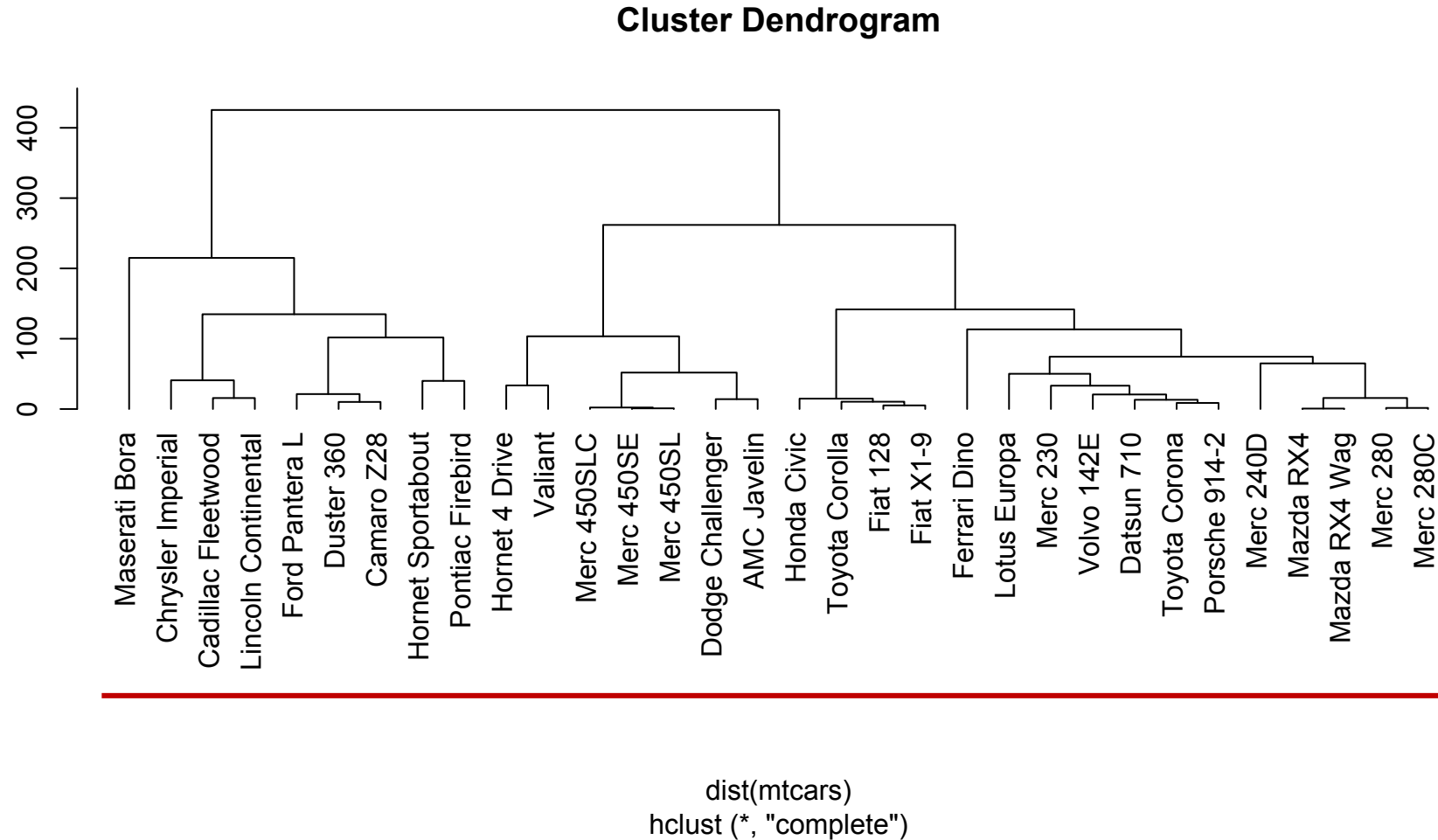
Hierarchical Clustering Algorithm

An Illustration with mtcars



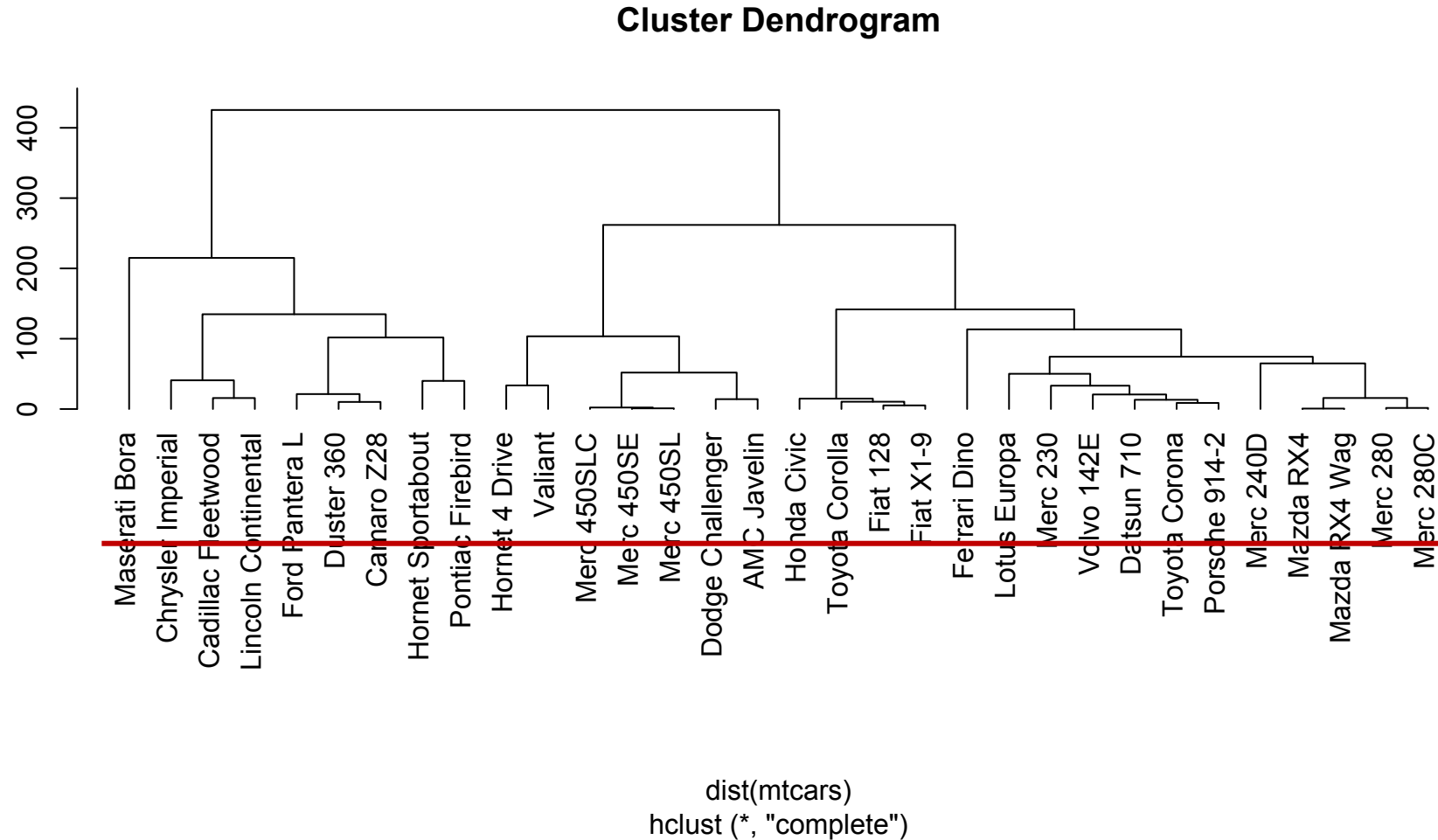
Hierarchical Clustering Algorithm

An Illustration with mtcars



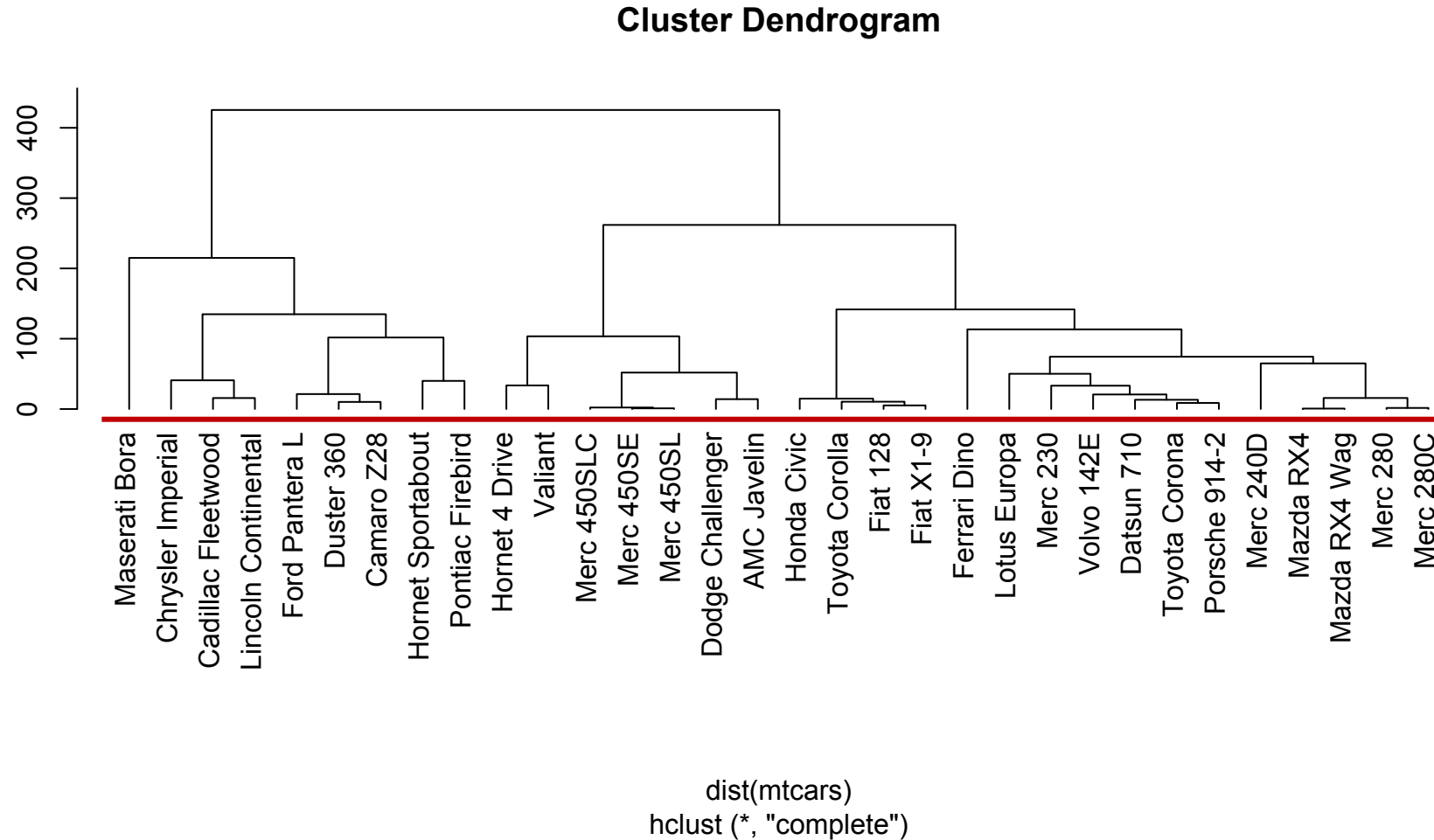
Hierarchical Clustering Algorithm

An Illustration with mtcars



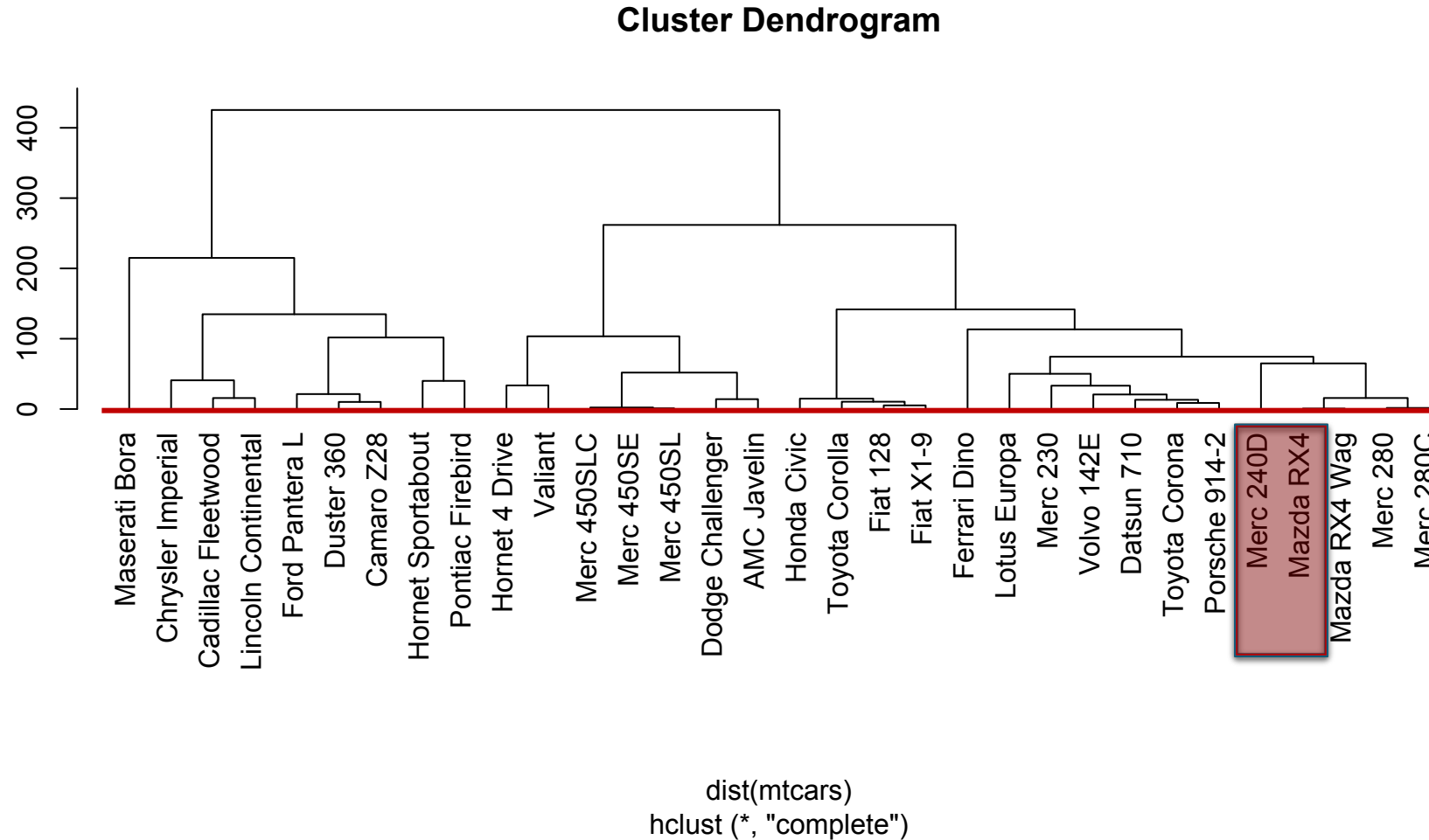
Hierarchical Clustering Algorithm

An Illustration with mtcars



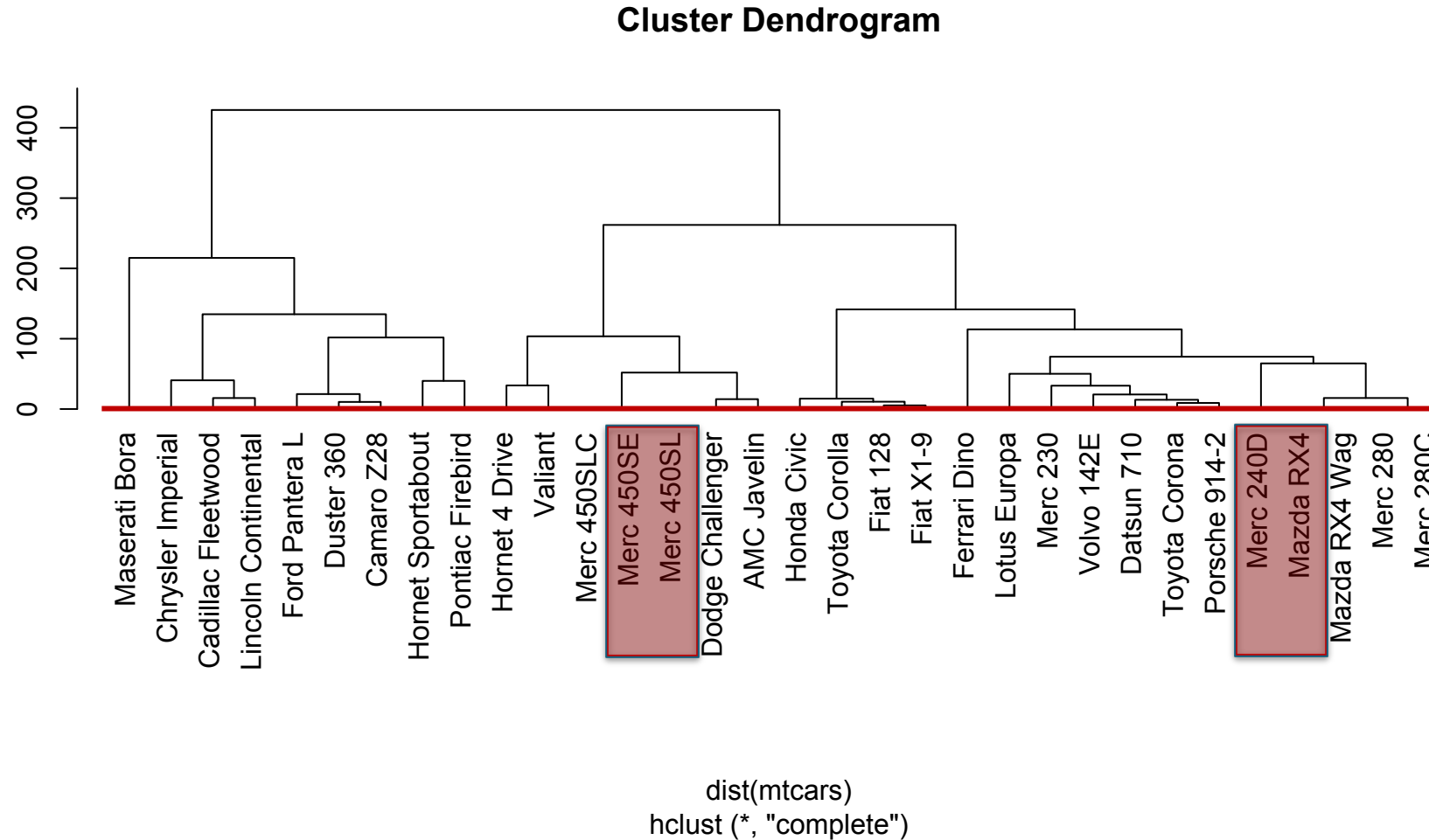
Hierarchical Clustering Algorithm

An Illustration with mtcars



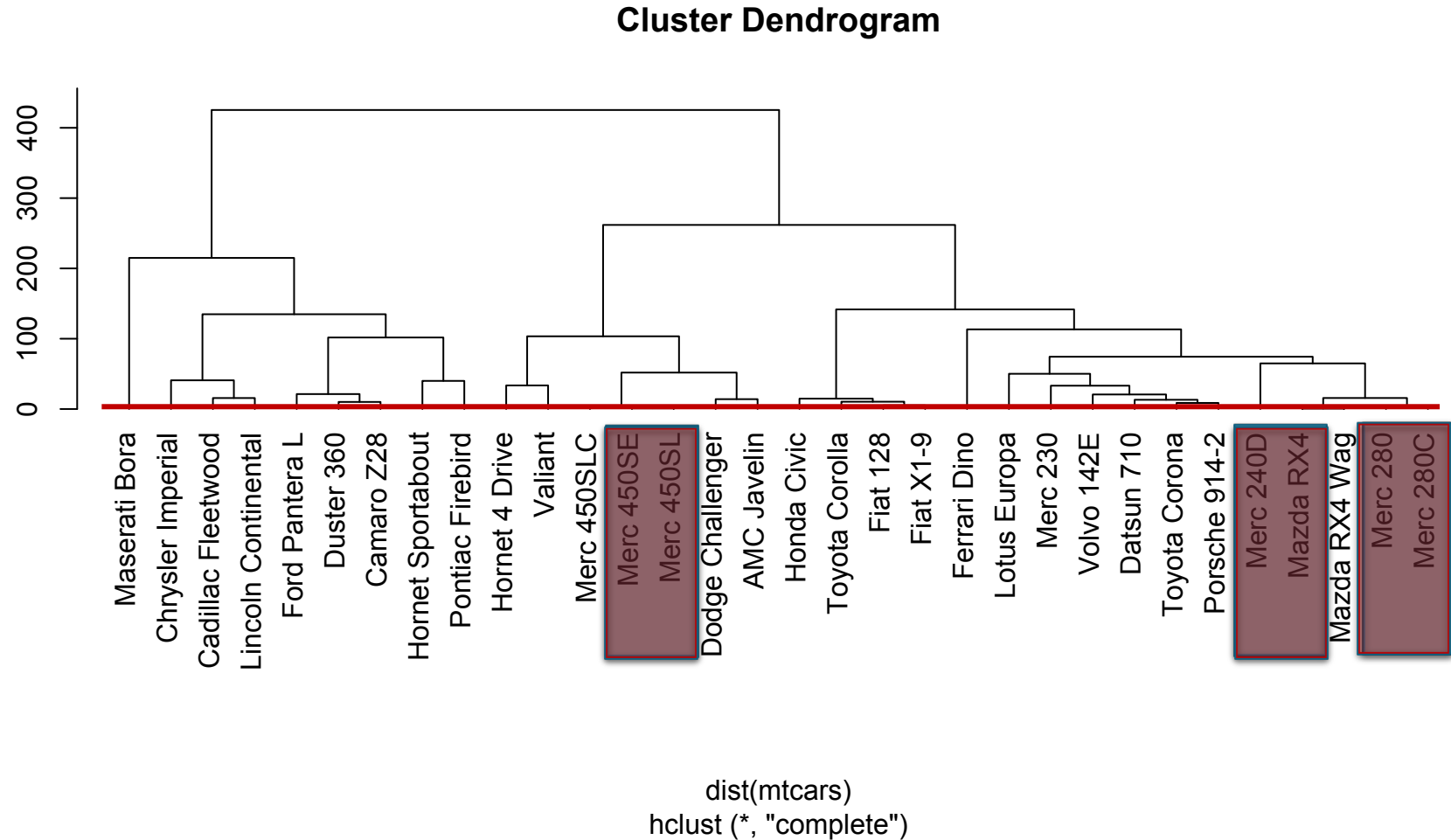
Hierarchical Clustering Algorithm

An Illustration with mtcars



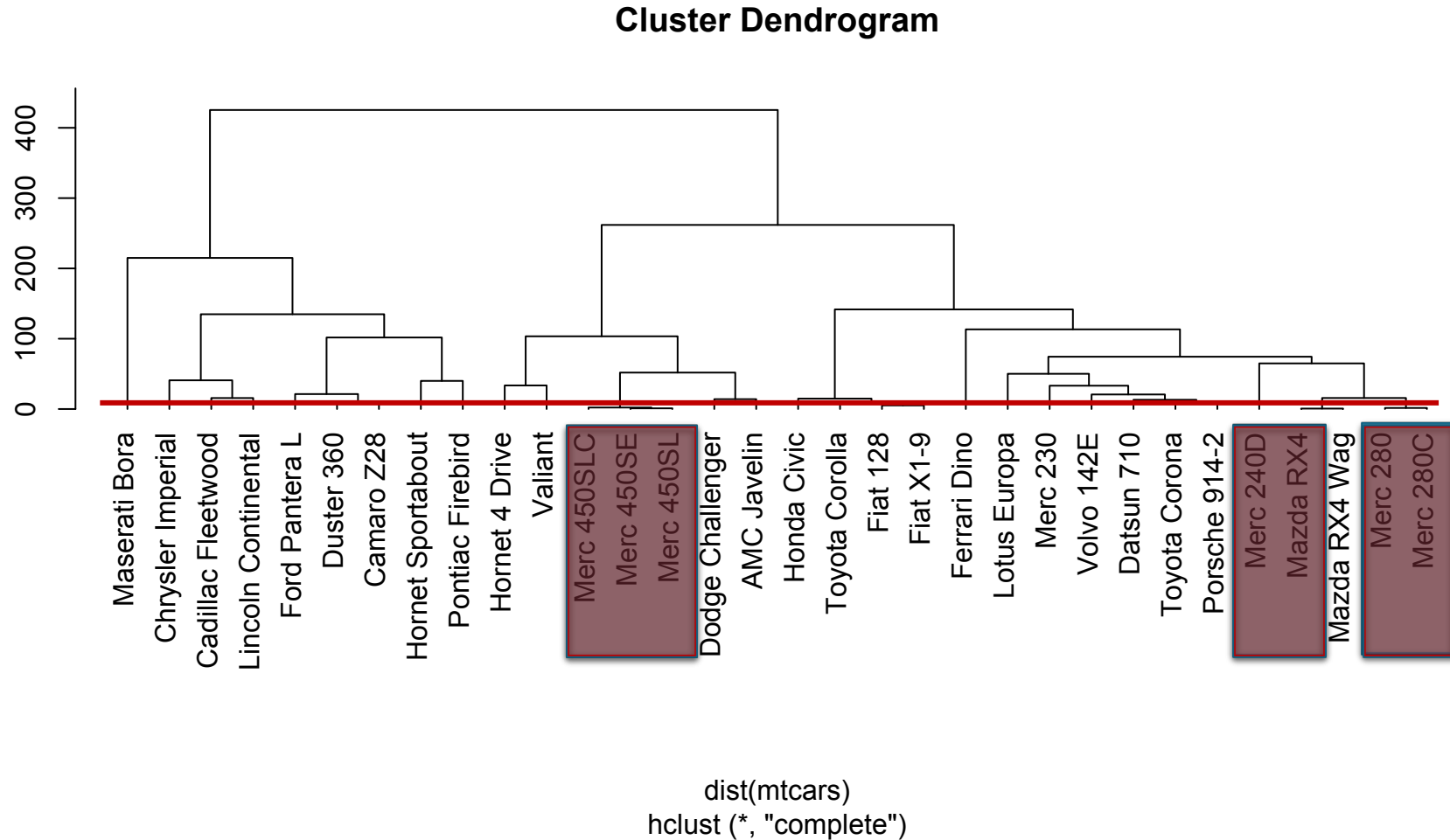
Hierarchical Clustering Algorithm

An Illustration with mtcars



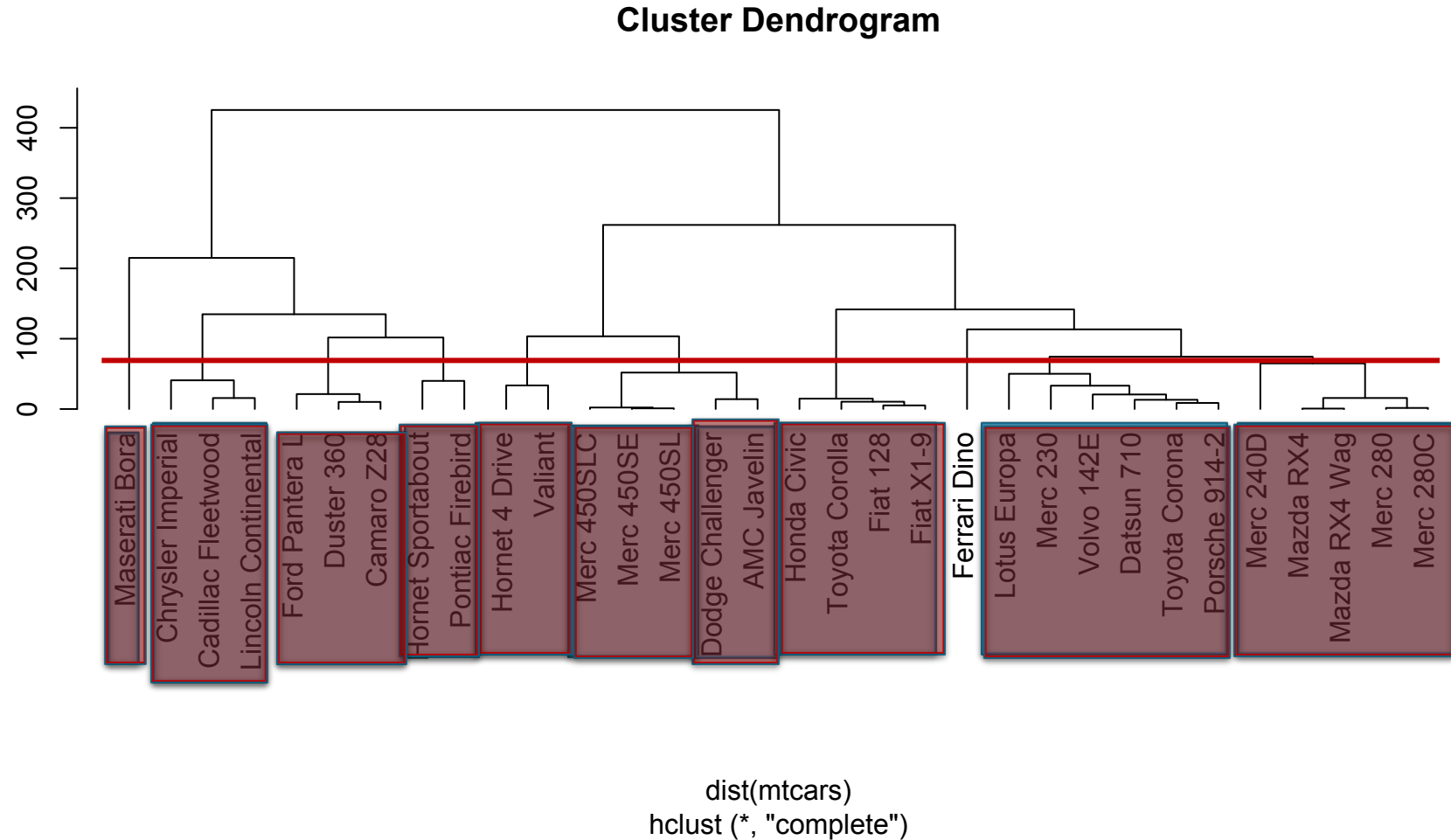
Hierarchical Clustering Algorithm

An Illustration with mtcars



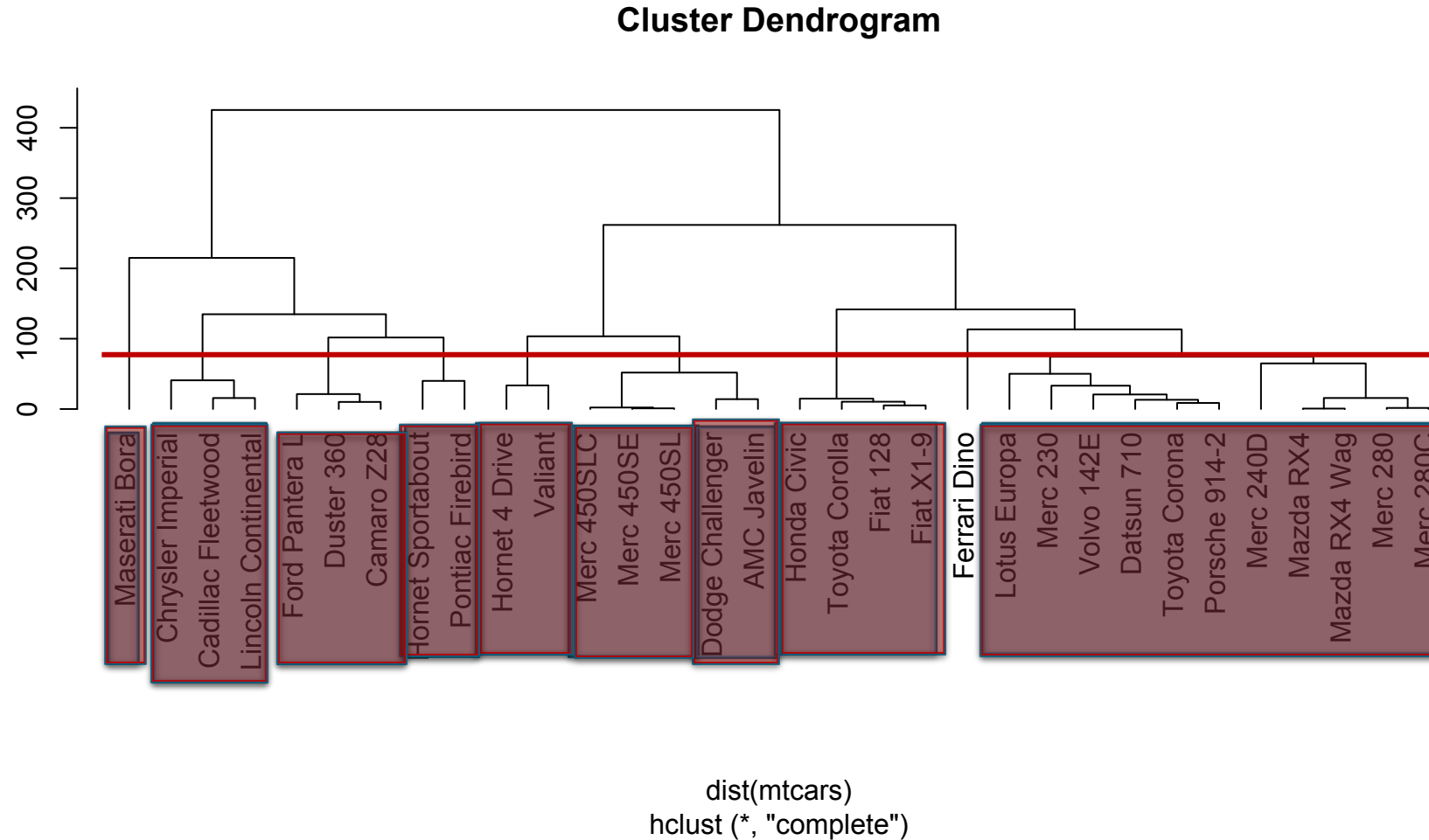
Hierarchical Clustering Algorithm

An Illustration with mtcars



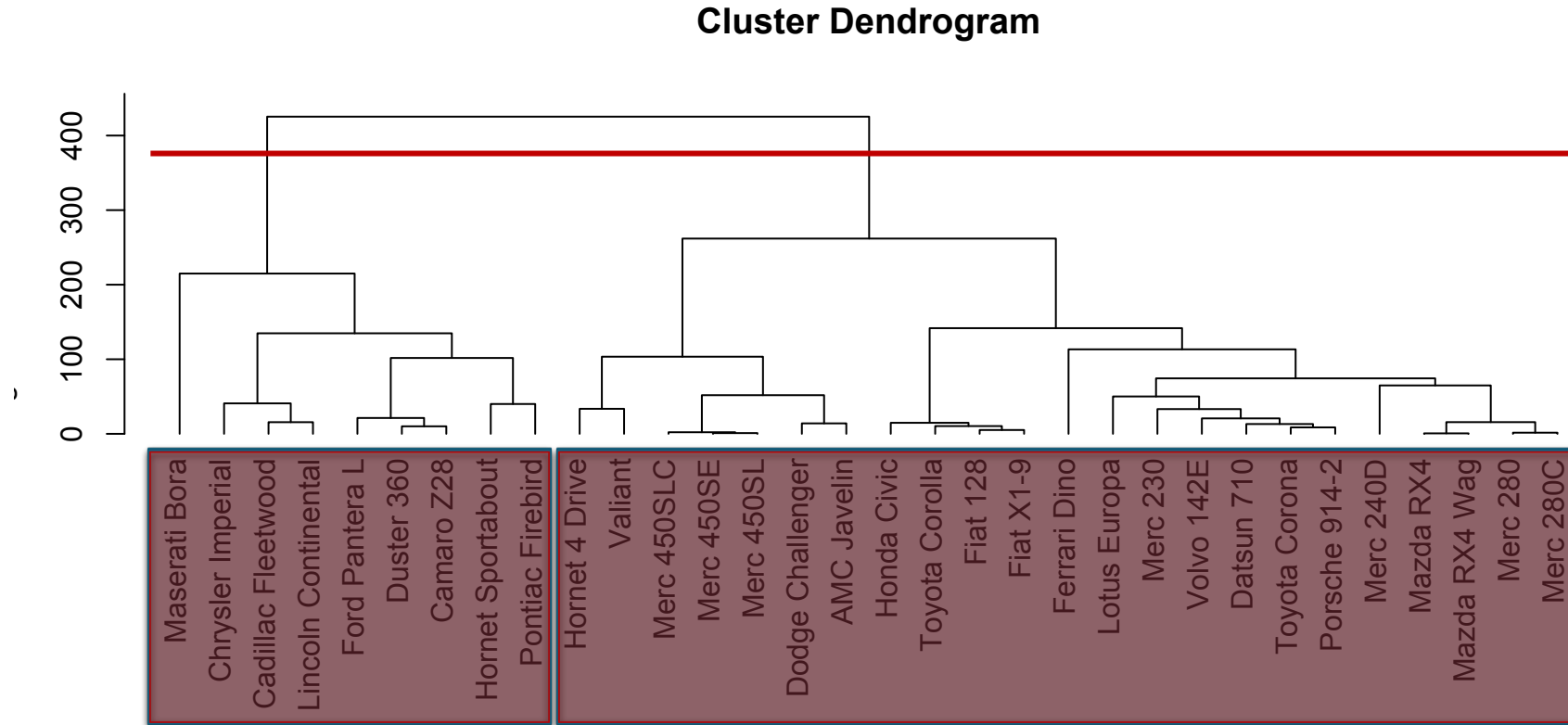
Hierarchical Clustering Algorithm

An Illustration with mtcars



Hierarchical Clustering Algorithm

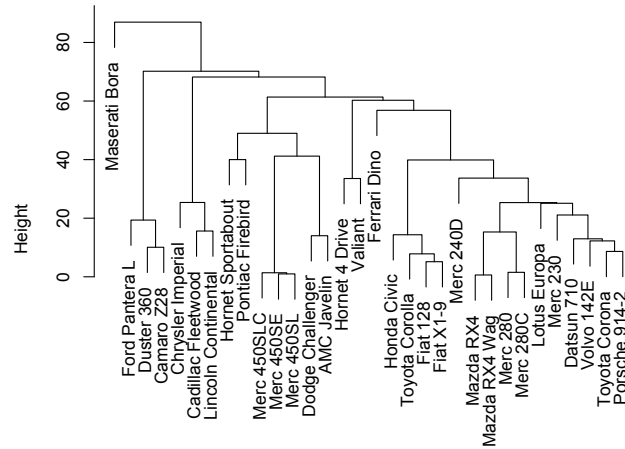
An Illustration with mtcars



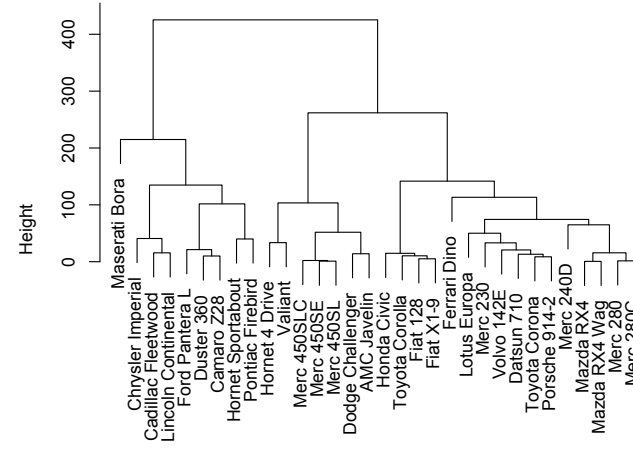
```
dist(mtcars)  
hclust (*, "complete")
```

Hierarchical Clustering Algorithm

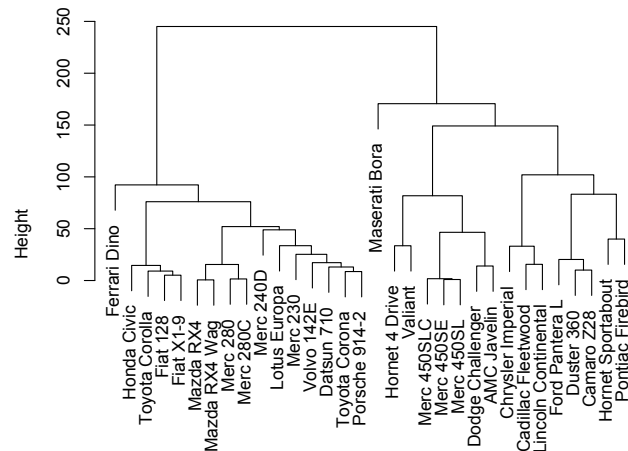
Same Cluster, Different Parameters



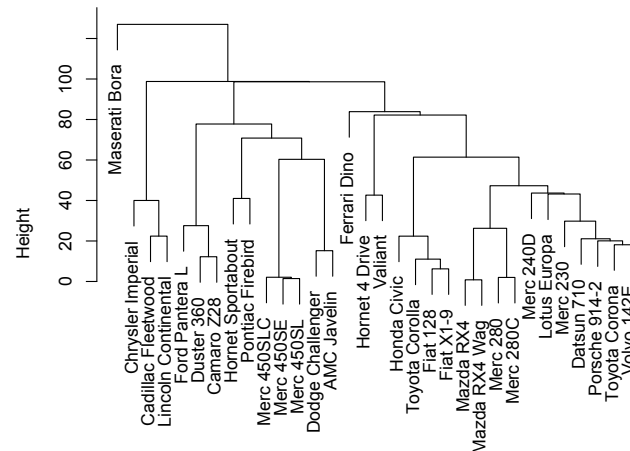
Euclidean Distance Single Link



Euclidean Distance Complete Link



Euclidean Distance Average Link



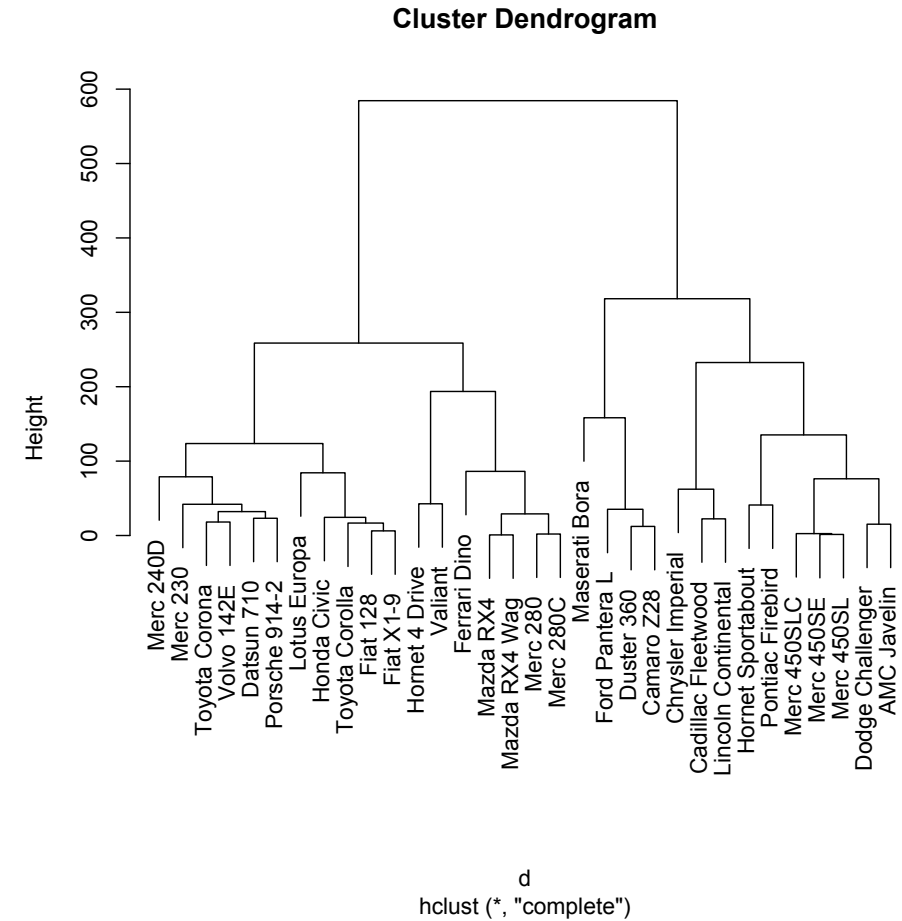
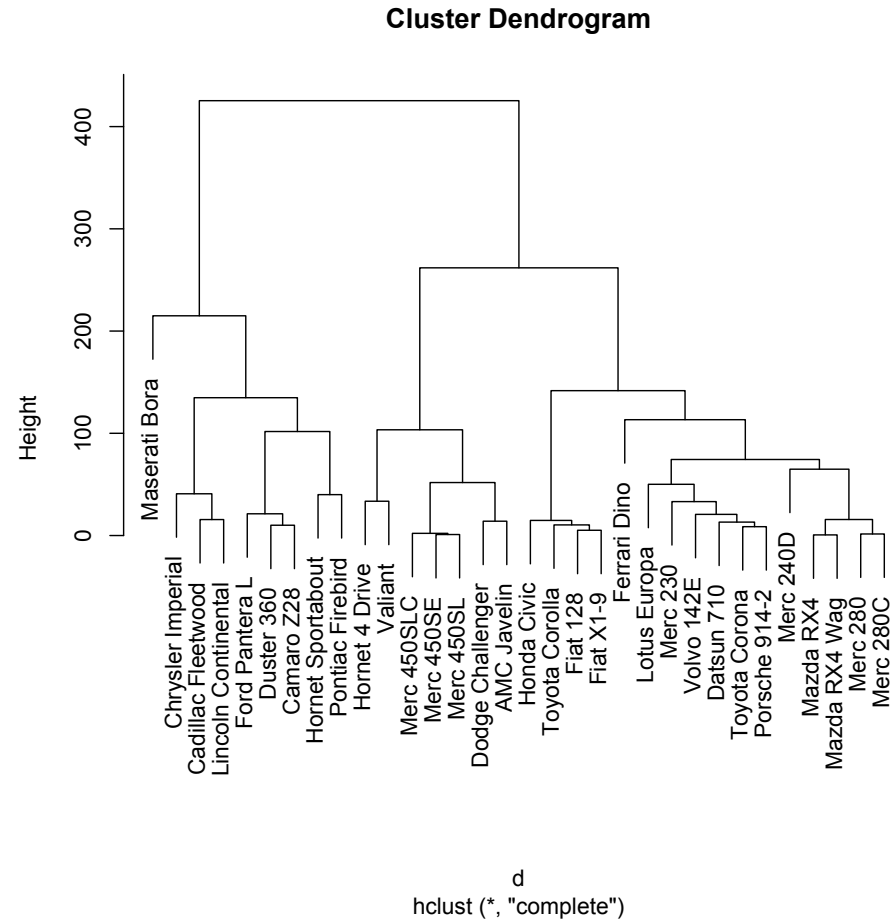
Manhattan Distance Single Link

Same data clustered
using different
parameter settings:

- **distance** metric
- **linkage** strategy

Hierarchical Clustering Algorithm

Parameters: Distance Metric



Same data clustered using two different distance metrics (euclidean, manhattan)

Hierarchical Clustering Algorithm

Similarity-Dissimilarity

Compare objects

```
> mtcars
```

	mpg	cyl	disp	hp	drat	wt
Mazda RX4	21.0	6	160.0	110	3.90	2.620
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875
Datsun 710	22.8	4	108.0	93	3.85	2.320
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440
Valiant	18.1	6	225.0	105	2.76	3.460
Duster 360	14.3	8	360.0	245	3.21	3.570
Merc 240D	24.4	4	146.7	62	3.69	3.190
Merc 230	22.8	4	140.8	95	3.92	3.150
Merc 280	19.2	6	167.6	123	3.92	3.440
Merc 280C	17.8	6	167.6	123	3.92	3.440
Merc 450SE	16.4	8	275.8	180	3.07	4.070
Merc 450SL	17.3	8	275.8	180	3.07	3.730
Merc 450SLC	15.2	8	275.8	180	3.07	3.780
Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250
Lincoln Continental	10.4	8	460.0	215	3.00	5.424
Chrysler Imperial	14.7	8	440.0	230	3.23	5.345
Fiat 128	32.4	4	78.7	66	4.08	2.200
Honda Civic	30.4	4	75.7	52	4.93	1.615
Toyota Corolla	33.9	4	71.1	65	4.22	1.835
Toyota Corona	21.5	4	120.1	97	3.70	2.465
Dodge Challenger	15.5	8	318.0	150	2.76	3.520
AMC Javelin	15.2	8	304.0	150	3.15	3.435
Camaro Z28	13.3	8	350.0	245	3.73	3.840
Pontiac Firebird	19.2	8	400.0	175	3.08	3.845
Fiat X1-9	27.3	4	79.0	66	4.08	1.935
Porsche 914-2	26.0	4	120.3	91	4.43	2.140
Lotus Europa	30.4	4	95.1	113	3.77	1.513
Ford Pantera L	15.8	8	351.0	264	4.22	3.170
Ferrari Dino	19.7	6	145.0	175	3.62	2.770
Maserati Bora	15.0	8	301.0	335	3.54	3.570
Volvo 142E	21.4	4	121.0	109	4.11	2.780

Compare values

Compare groups

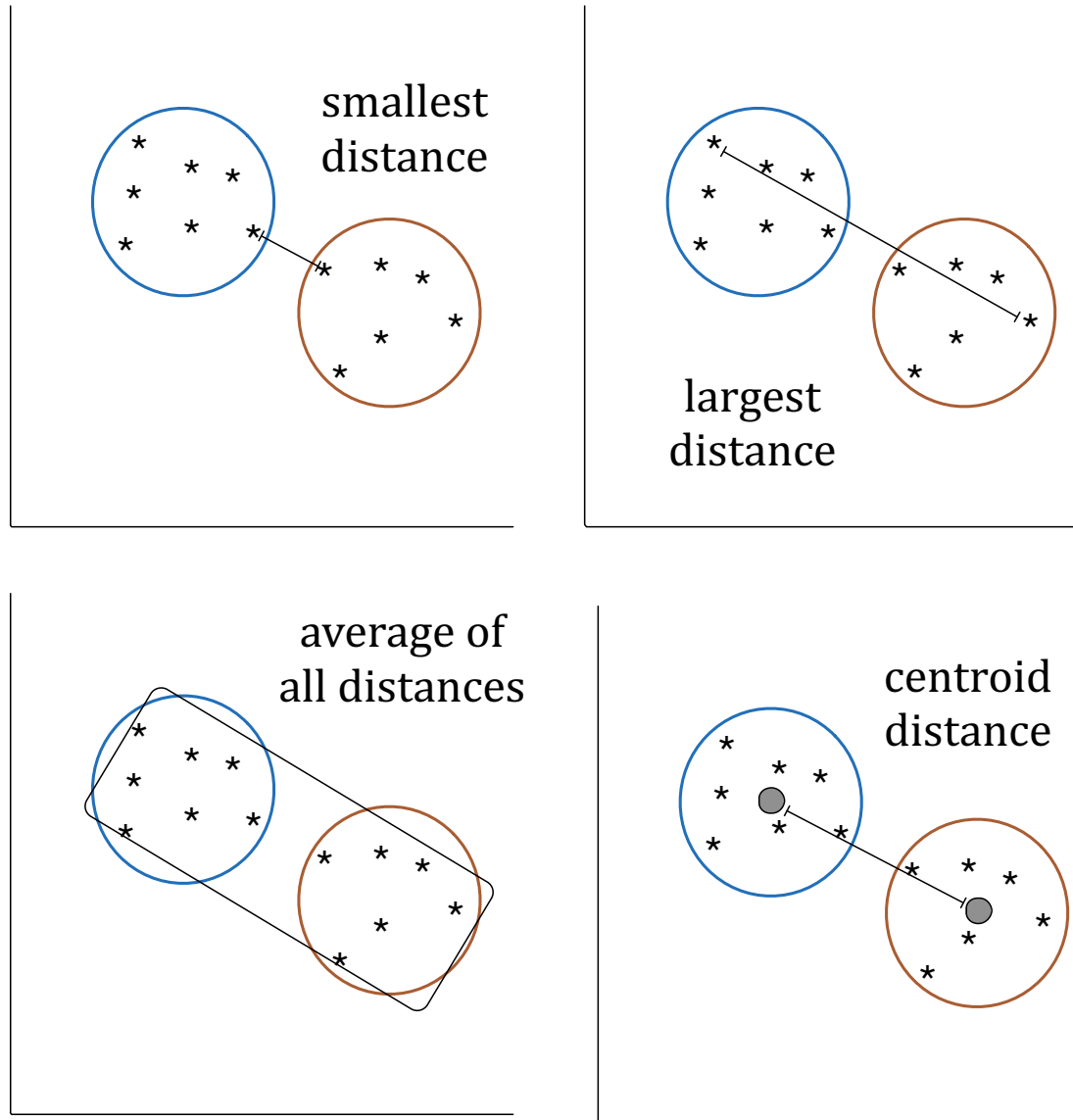
Compare variables

```
> mtcars
```

	mpg	cyl	disp	hp	drat	wt
Mazda RX4	21.0	6	160.0	110	3.90	2.620
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875
Datsun 710	22.8	4	108.0	93	3.85	2.320
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440
Valiant	18.1	6	225.0	105	2.76	3.460
Duster 360	14.3	8	360.0	245	3.21	3.570
Merc 240D	24.4	4	146.7	62	3.69	3.190
Merc 230	22.8	4	140.8	95	3.92	3.150
Merc 280	19.2	6	167.6	123	3.92	3.440
Merc 280C	17.8	6	167.6	123	3.92	3.440
Merc 450SE	16.4	8	275.8	180	3.07	4.070
Merc 450SL	17.3	8	275.8	180	3.07	3.730
Merc 450SLC	15.2	8	275.8	180	3.07	3.780
Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250
Lincoln Continental	10.4	8	460.0	215	3.00	5.424
Chrysler Imperial	14.7	8	440.0	230	3.23	5.345
Fiat 128	32.4	4	78.7	66	4.08	2.200
Honda Civic	30.4	4	75.7	52	4.93	1.615
Toyota Corolla	33.9	4	71.1	65	4.22	1.835
Toyota Corona	21.5	4	120.1	97	3.70	2.465
Dodge Challenger	15.5	8	318.0	150	2.76	3.520
AMC Javelin	15.2	8	304.0	150	3.15	3.435
Camaro Z28	13.3	8	350.0	245	3.73	3.840
Pontiac Firebird	19.2	8	400.0	175	3.08	3.845
Fiat X1-9	27.3	4	79.0	66	4.08	1.935
Porsche 914-2	26.0	4	120.3	91	4.43	2.140
Lotus Europa	30.4	4	95.1	113	3.77	1.513
Ford Pantera L	15.8	8	351.0	264	4.22	3.170
Ferrari Dino	19.7	6	145.0	175	3.62	2.770
Maserati Bora	15.0	8	301.0	335	3.54	3.570
Volvo 142E	21.4	4	121.0	109	4.11	2.780

Hierarchical Clustering Algorithm

Parameters: Linkage



The chosen linkage algorithm affects which clusters are merged, and the shape of the resulting clusters (e.g. tighter, looser)

Strengths and Limitations – Linkages

Single Linkage (smallest distance)

- can handle non-blob shapes
- sensitive to noise and outliers
- produces elongated clusters

Complete Linkage (largest distance)

- balanced clusters, with similar diameters
- not overly sensitive to noise
- tends to split large clusters
- all clusters tend to have similar diameters

Strengths and Limitations – Linkages

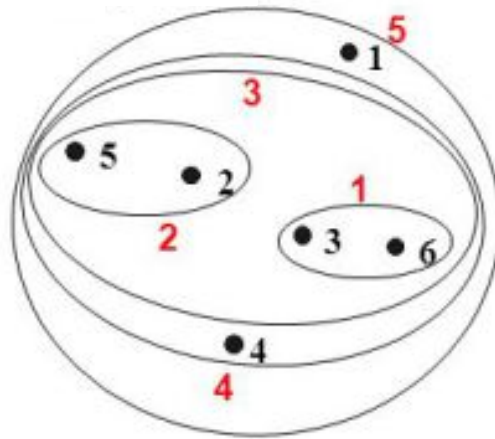
Average Linkage (average distance)

- compromise between single and complete linkages
- not too sensitive to noise and outliers
- tends to produce blob-shaped clusters

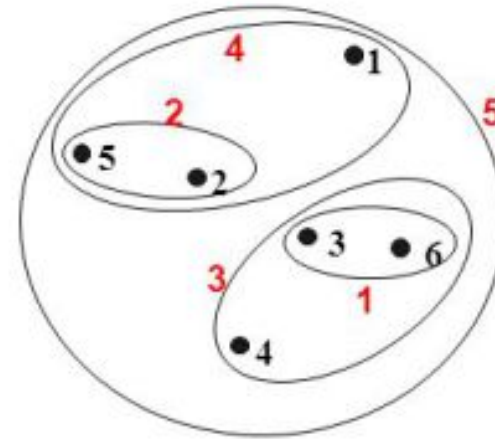
Centroid Linkage (centroid distance)

- clusters can have a lot of internal variance

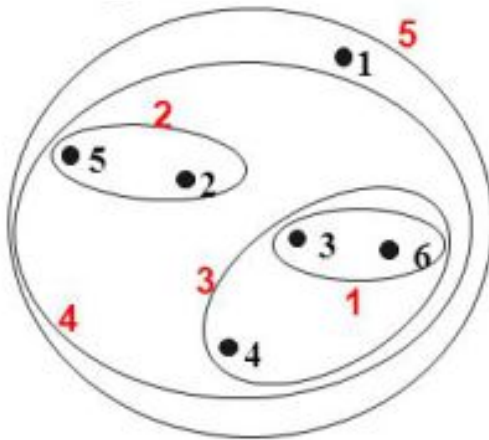
Simple Link



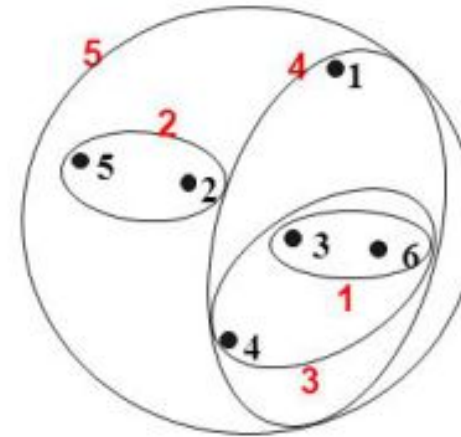
Complete Link



Average Link

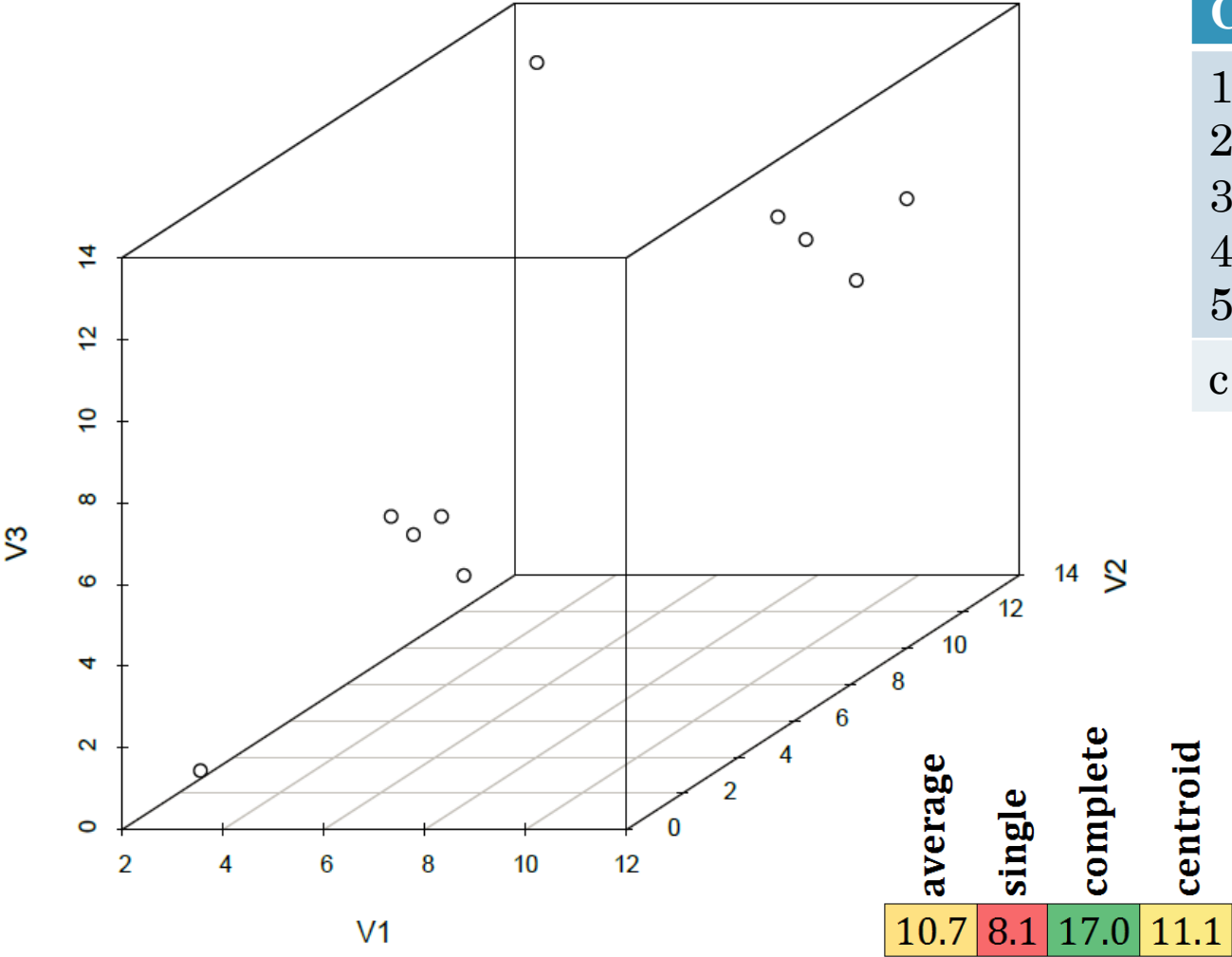


Centroid Link



Hierarchical Clustering Algorithm

Parameters: Linkage – Example



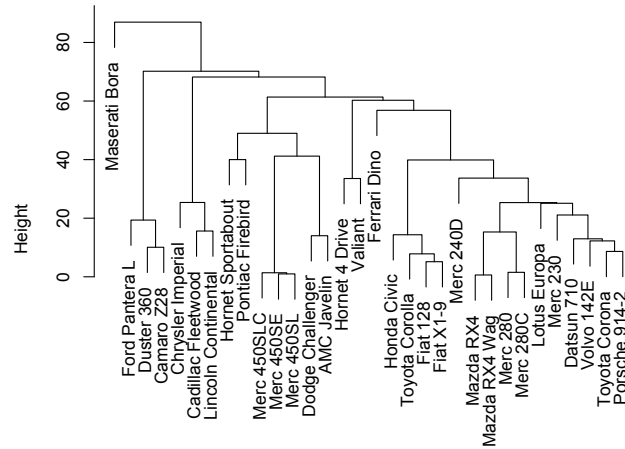
Data	
Cluster A	Cluster B
1: (5,5,5)	6: (10,10,10)
2: (5,6,5)	7: (11,10,9)
3: (4,6,5)	8: (12,10,11)
4: (6,5,4)	9: (10,9,11)
5: (3,1,1)	10: (13,13,13)
c: (4.6,4.6,4)	c: (11.2,10.4,10.8)

		Cluster A					Distance matrix
		1	2	3	4	5	
Cluster B	6	8.7	8.1	8.8	8.8	14.5	
	7	8.8	8.2	9.0	8.7	14.5	
	8	10.5	10.0	10.8	10.5	16.2	
	9	8.8	8.4	9.0	9.0	14.6	
	10	11.5	10.8	10.7	12.4	17.0	

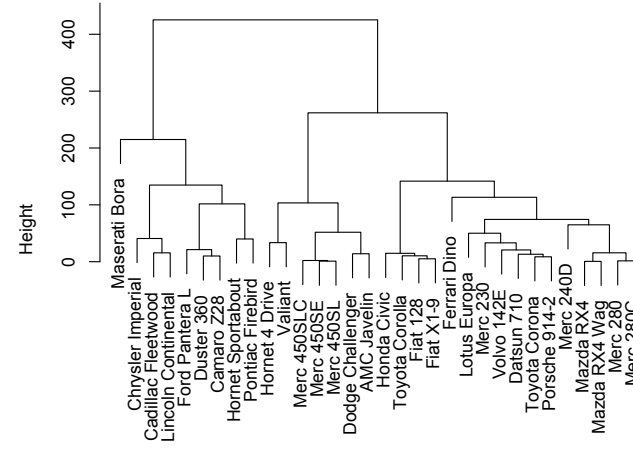
Linkage

Hierarchical Clustering Algorithm

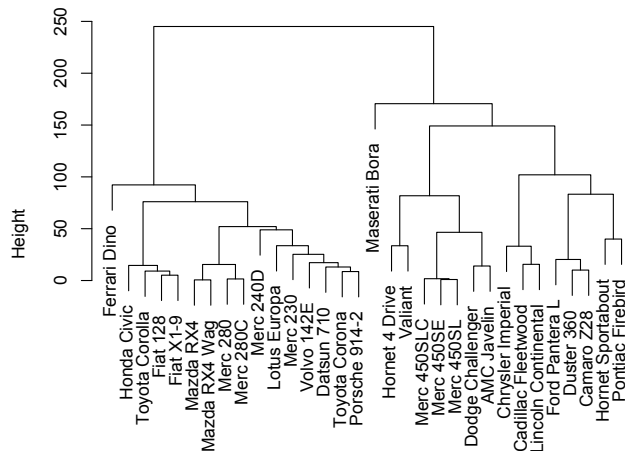
Returning to Our Clustering Results



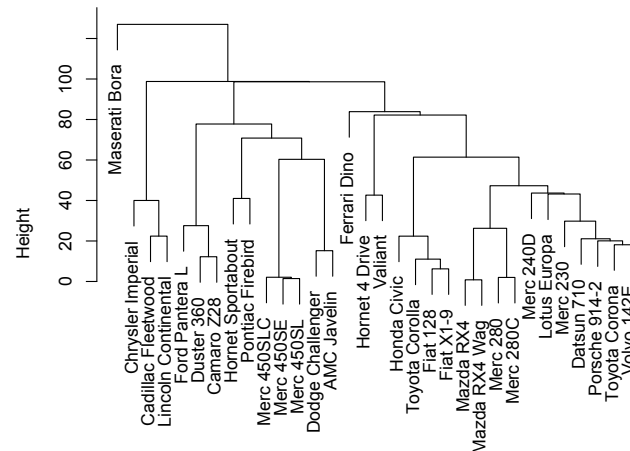
Euclidean Distance Single Link



Euclidean Distance Complete Link



Euclidean Distance Average Link



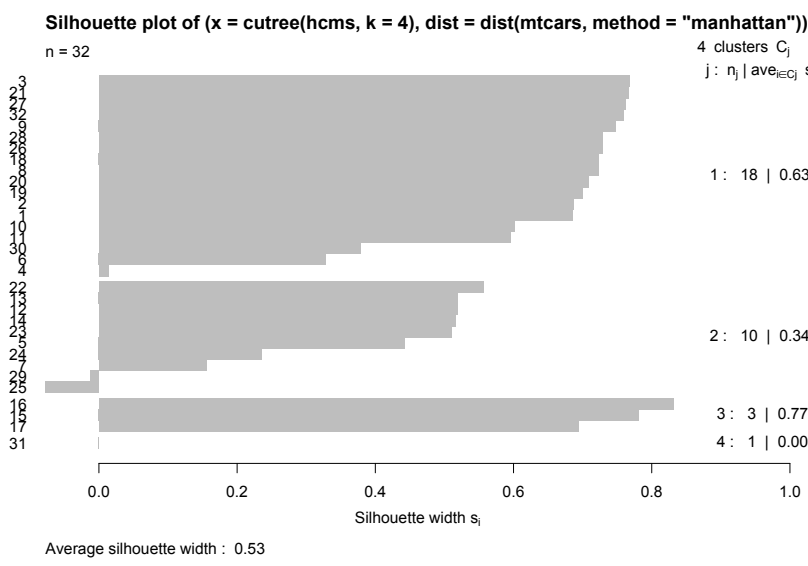
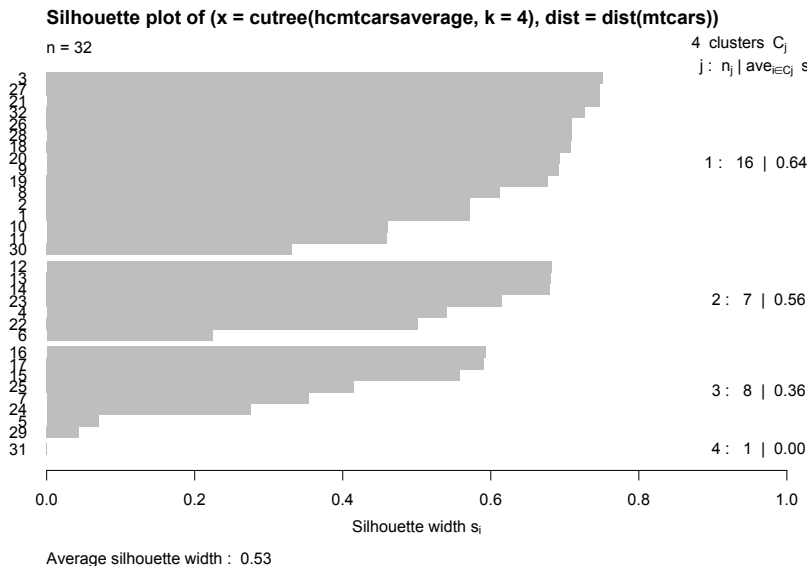
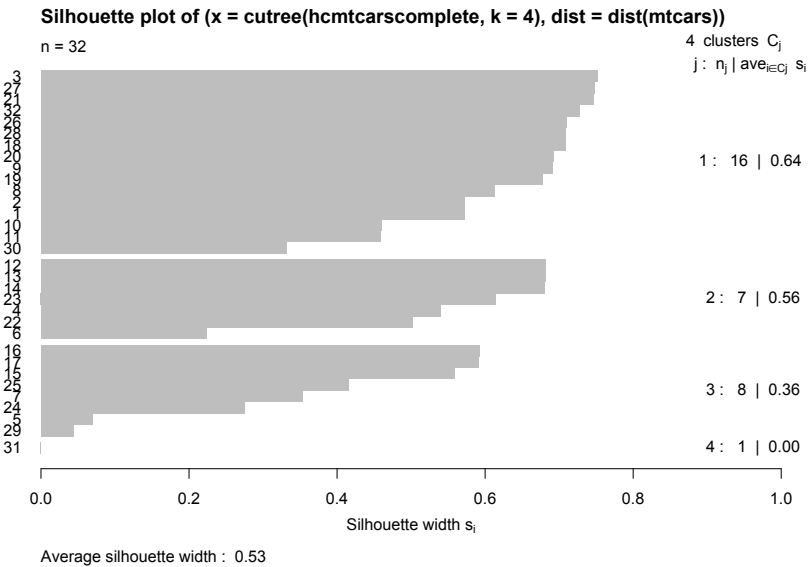
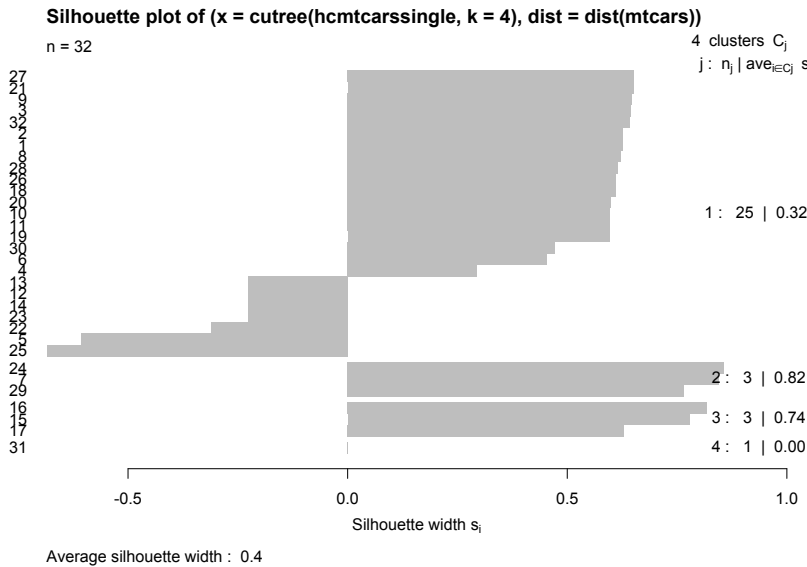
Manhattan Distance Single Link

It is often said:
choose the distance
metric that is most
meaningful for
your data.

Can evaluation
metrics also inform
this choice?

Hierarchical Clustering Algorithm

Silhouette of Clustering Results

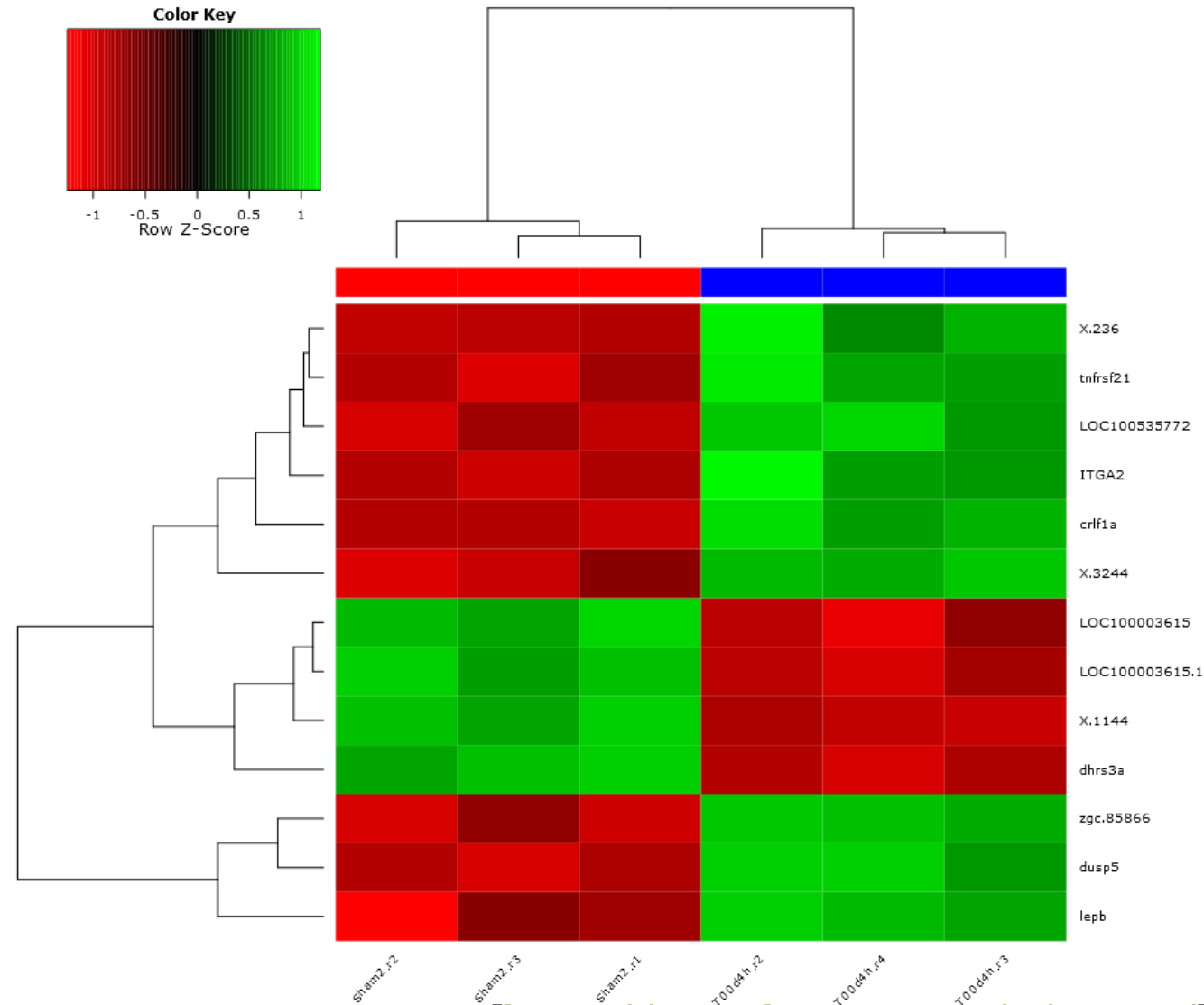


Hierarchical Clustering Notes

HC is deterministic, for a given choice of metric and linkage.

Space and time requirements do make HC unattractive for medium-to-large datasets.

Various linkage strategies: be sure to check out **Wald's method!**



Hierarchical Clustering Notes

Easy to understand and implement, but rarely optimal.

No real strong theoretical or first principle approach to specify the distance metric and linkage criteria (arbitrary decisions).

Cannot handle missing values or mixed data types.

Dendrograms can only be used to select the number of clusters when the ultrametric tree inequality holds (rarely does in practice).

Consider using **latent class analysis** instead.

HC Examples and Case Studies

Clustering Myths

Comparative Mythology

- Studying myths from different cultures to understand their similarities and possibly shared origins
- Many myths have splintered off and evolved from common sources

Julien d'Huy (2016): Used a variety of data mining techniques, including hierarchical clustering, to trace the evolution of myths.

Collection of myths broken down into common story elements.



A myth across cultures:
the hunter in the sky

HC Examples and Case Studies

Clustering Myths

Myths categorized based on presence/ absence of elements

Myths are clustered based on this categorization.

Result shows myths clustering together – could this suggest a possible common origin for these myths?

Remember, **clustering is knowledge discovery!**



A myth across cultures:
the hunter in the sky

HC Examples and Case Studies

Complex building's energy system operation patterns analysis using bag of words representation with hierarchical clustering

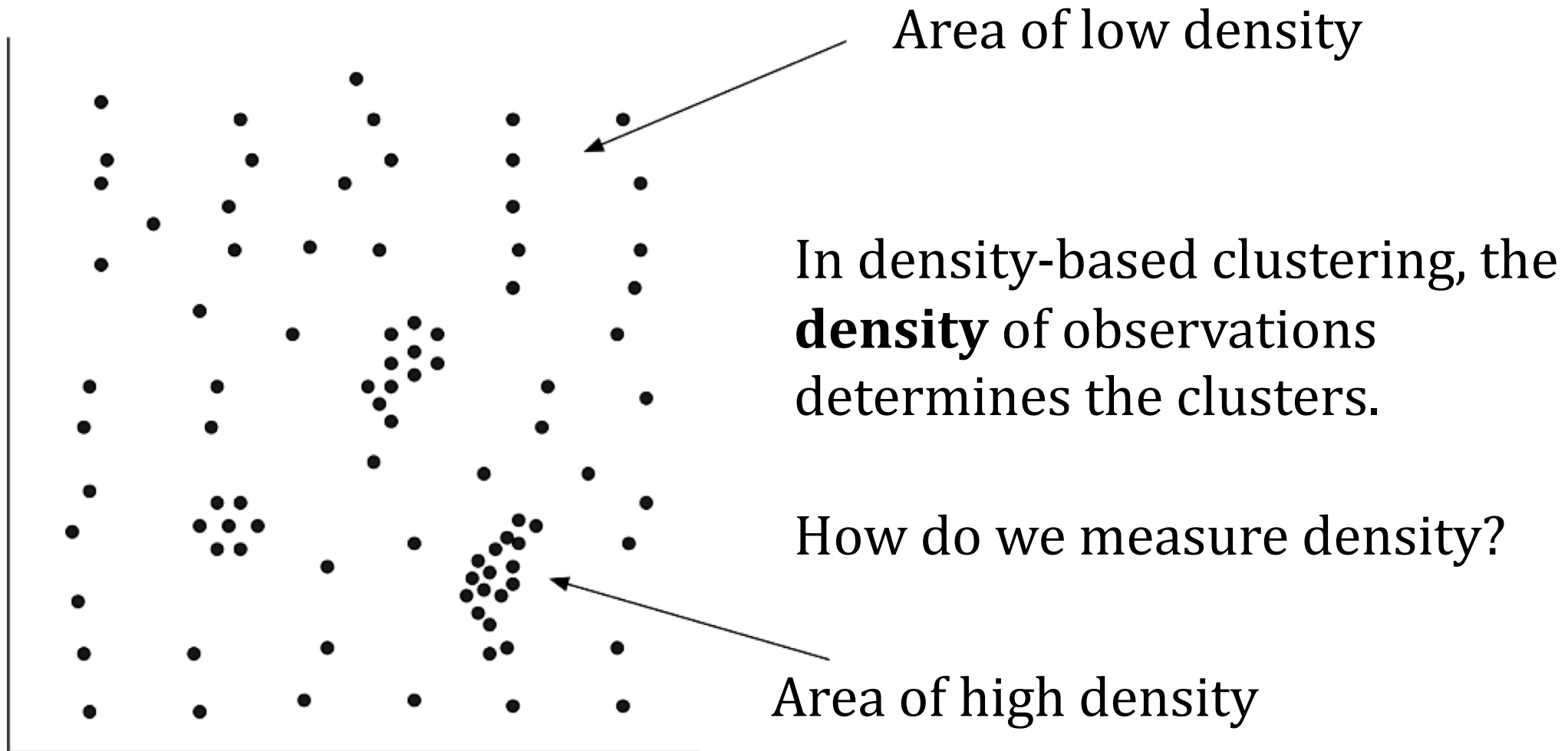
A Comparison of Antioxidant, Antibacterial, and Anticancer Activity of the Selected Thyme Species by Means of Hierarchical Clustering and Principal Component Analysis

Use of hierarchical cluster analysis to classify prisons in Ireland into mutually exclusive drug-use risk categories

Divisive Analysis (DIANA) of hierarchical clustering and GPS data for level of service criteria of urban streets

Density Based Clustering

Data Point Density



DBSCAN Algorithm – Parameters

DBSCAN uses 2 parameters:

- a **distance** parameter to create ε -neighbourhoods, and
- the **minimum number of points** in an ε -neighbourhood required to include the n'hood in the cluster being constructed (including the centre)

3 distinct types of points:

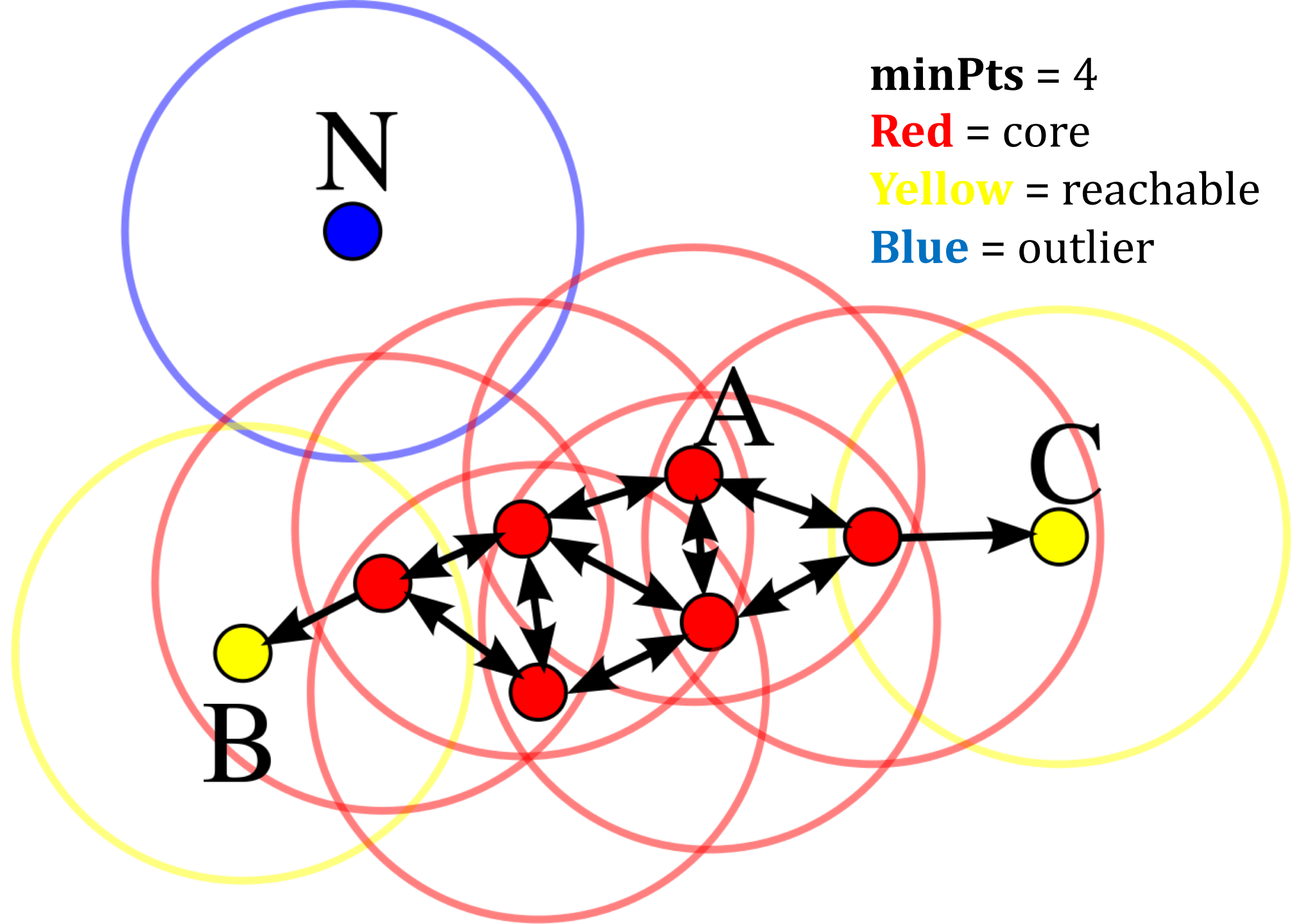
- **outliers**: out of reach of every other point
- **non-core (reachable)**: within reach of some number of points below the min. threshold
- **core**: within reach of at least the minimum number of other points

DBSCAN Algorithm – Parameters

Reachability is not a symmetric relation: **no point is reachable from a non-core point** (a non-core point may be reachable, but nothing can be reached from it).

Two points p and q are **density-connected** if there is a point o such that both p and q are reachable from o (but density-connectedness *is* symmetric).

All points within a cluster are mutually density-connected. If a point is density-reachable from any point of the cluster, **it is part of the cluster as well**.



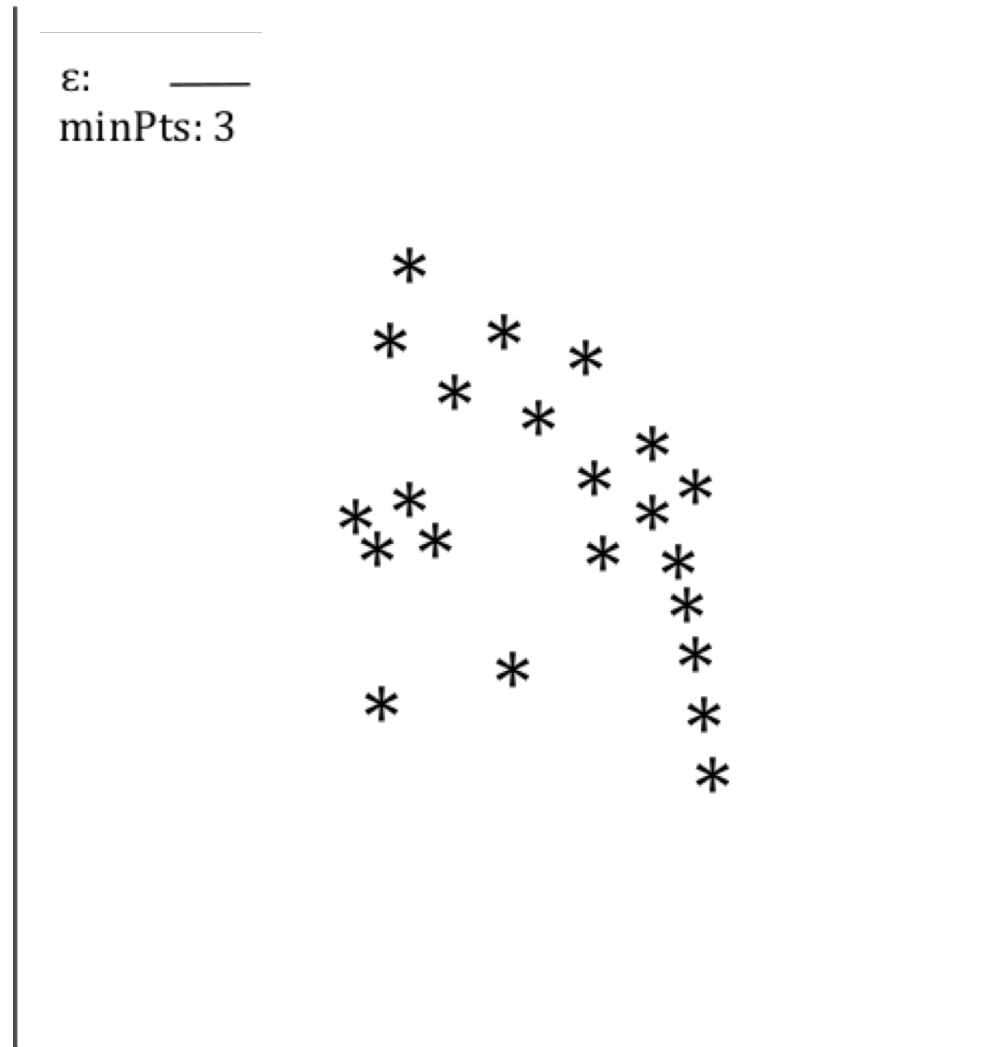
DBSCAN Algorithm

Given $\varepsilon > 0$ and minPts (as well as a distance metric d):

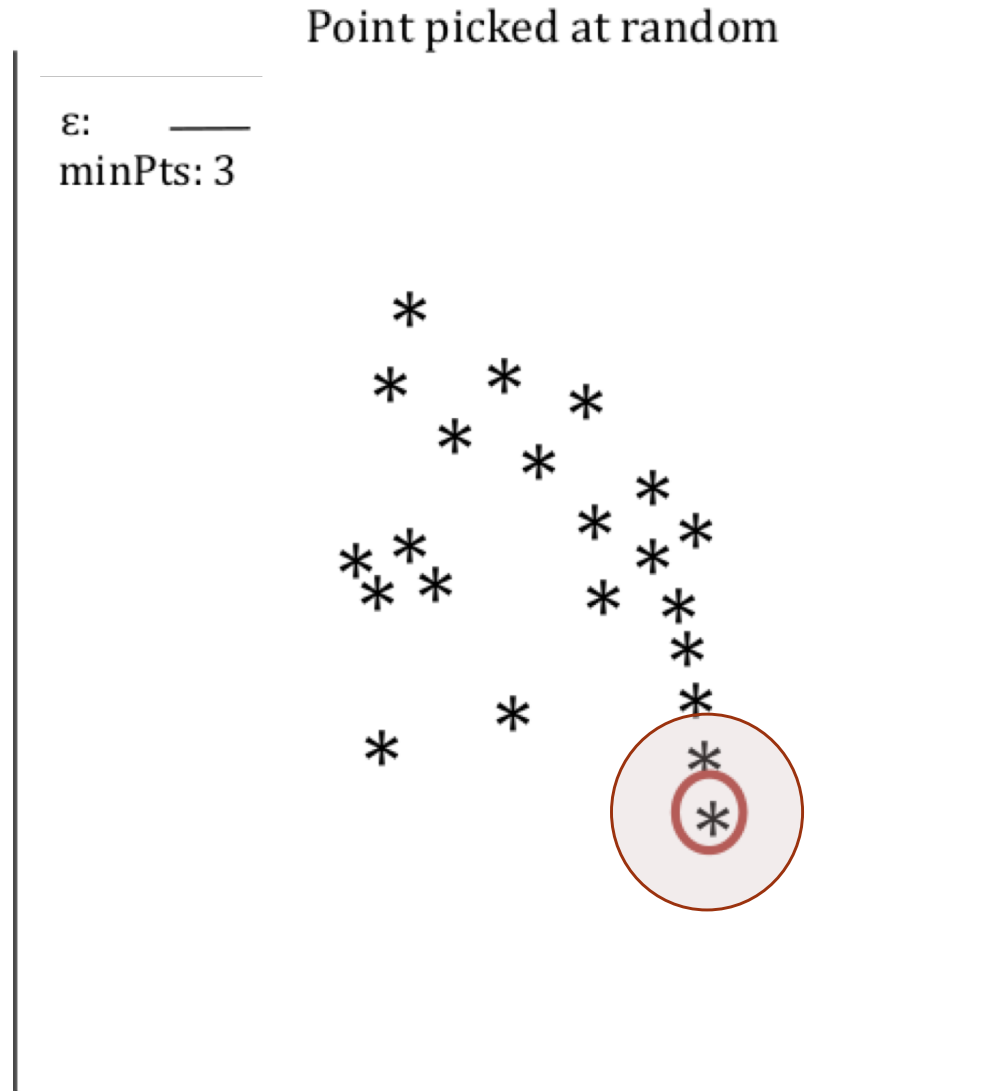
1. Find the ε -neighbours of every point, and identify the core points with more than minPts neighbours (including the core point).
2. Find the connected components of **core** points on the neighbor graph, ignoring all non-core points.
3. Assign each non-core point to a nearby cluster if the cluster is an ε neighbor, otherwise assign it to noise.

That's really all there is to it...

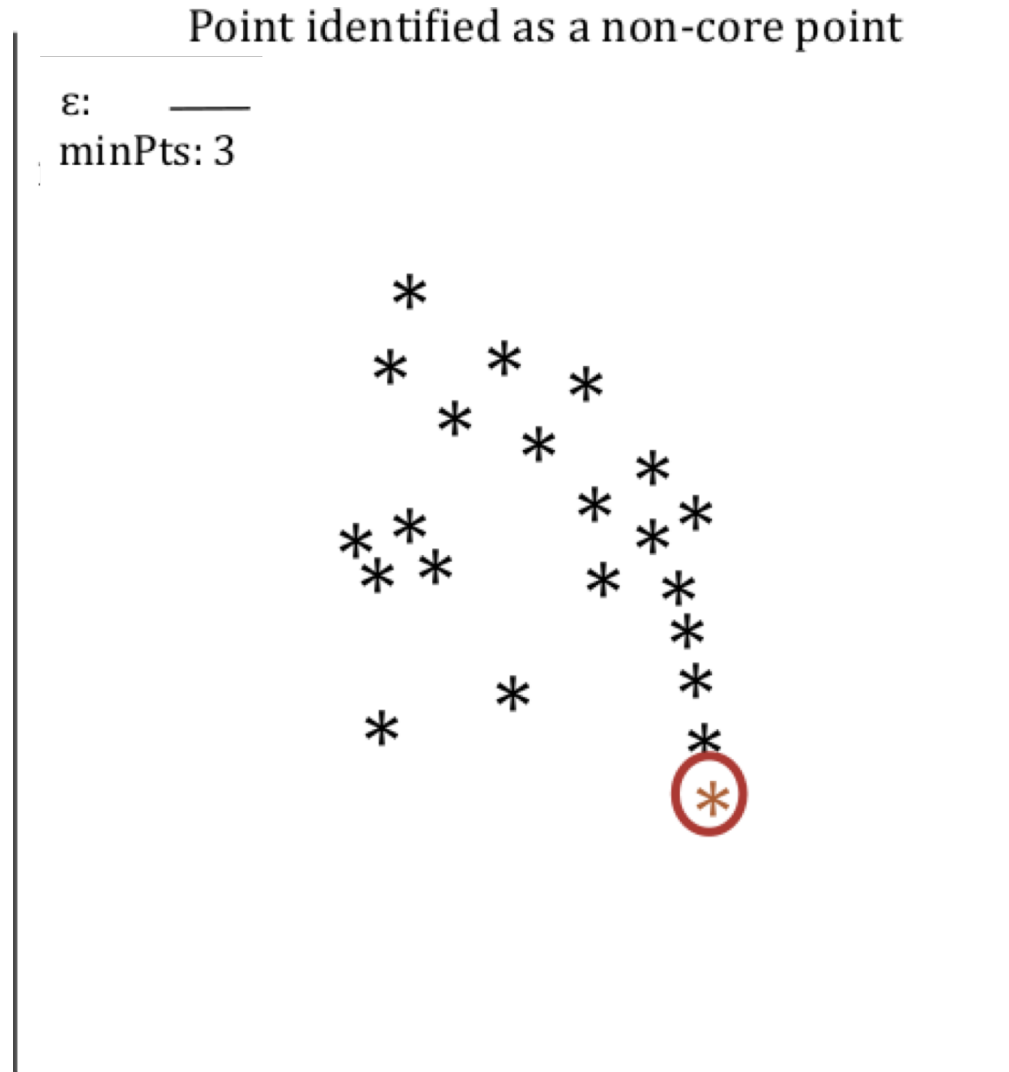
DBSCAN Example – Artificial Dataset



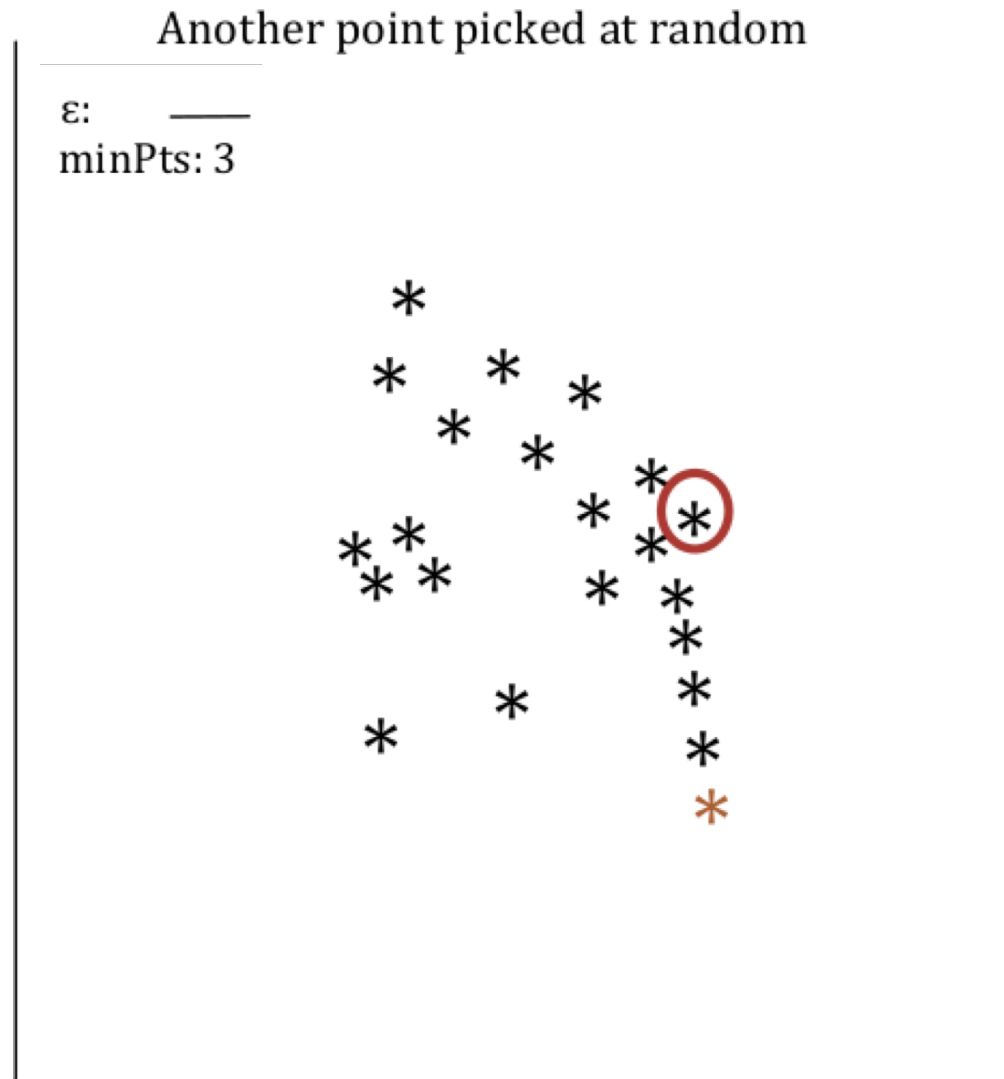
DBSCAN Example – Artificial Dataset



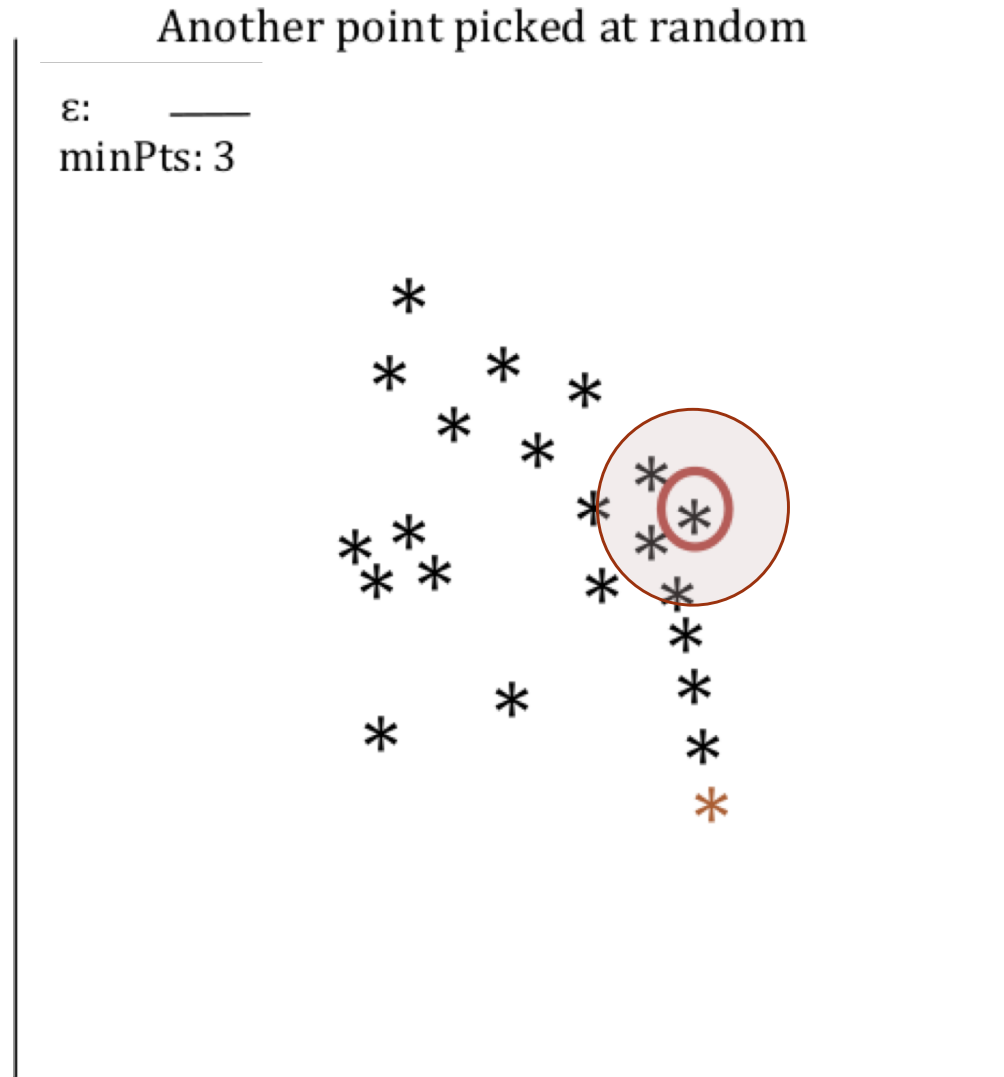
DBSCAN Example – Artificial Dataset



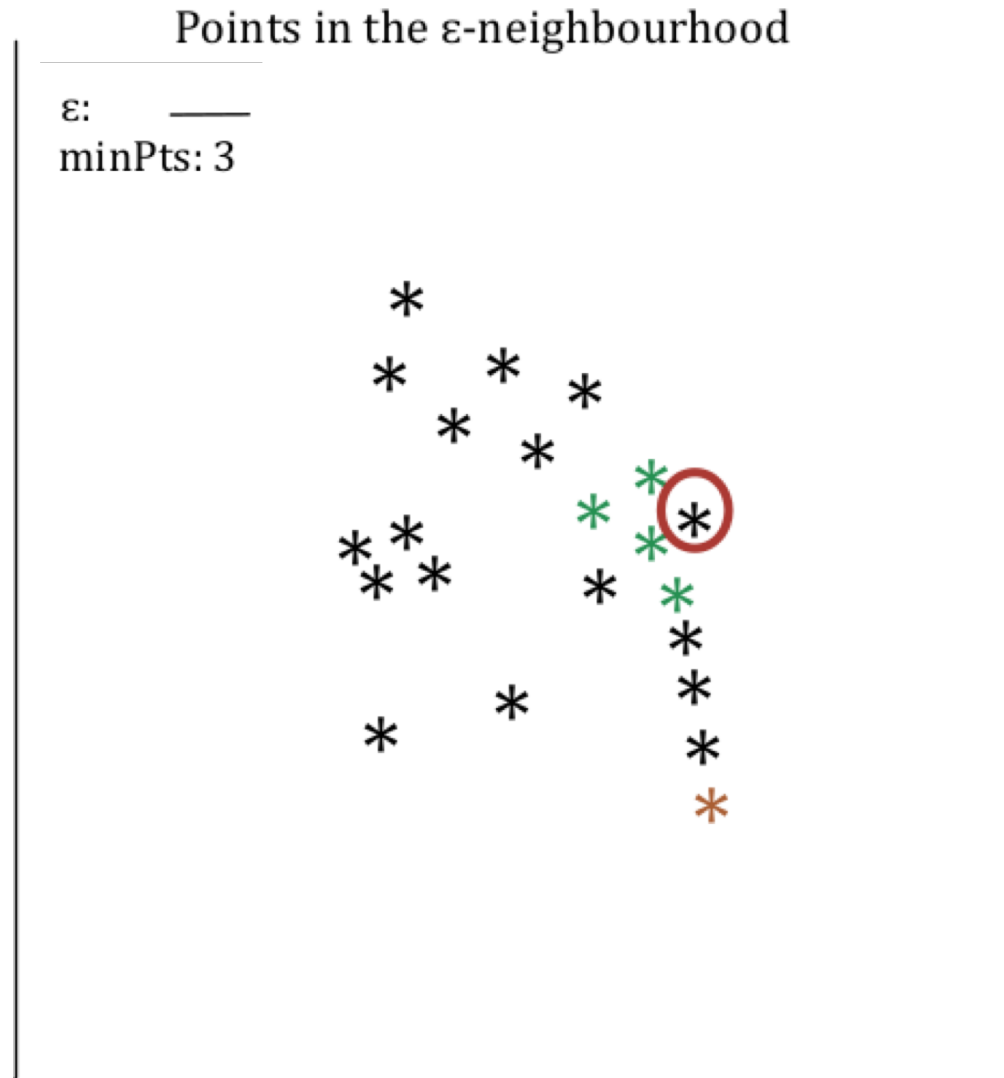
DBSCAN Example – Artificial Dataset



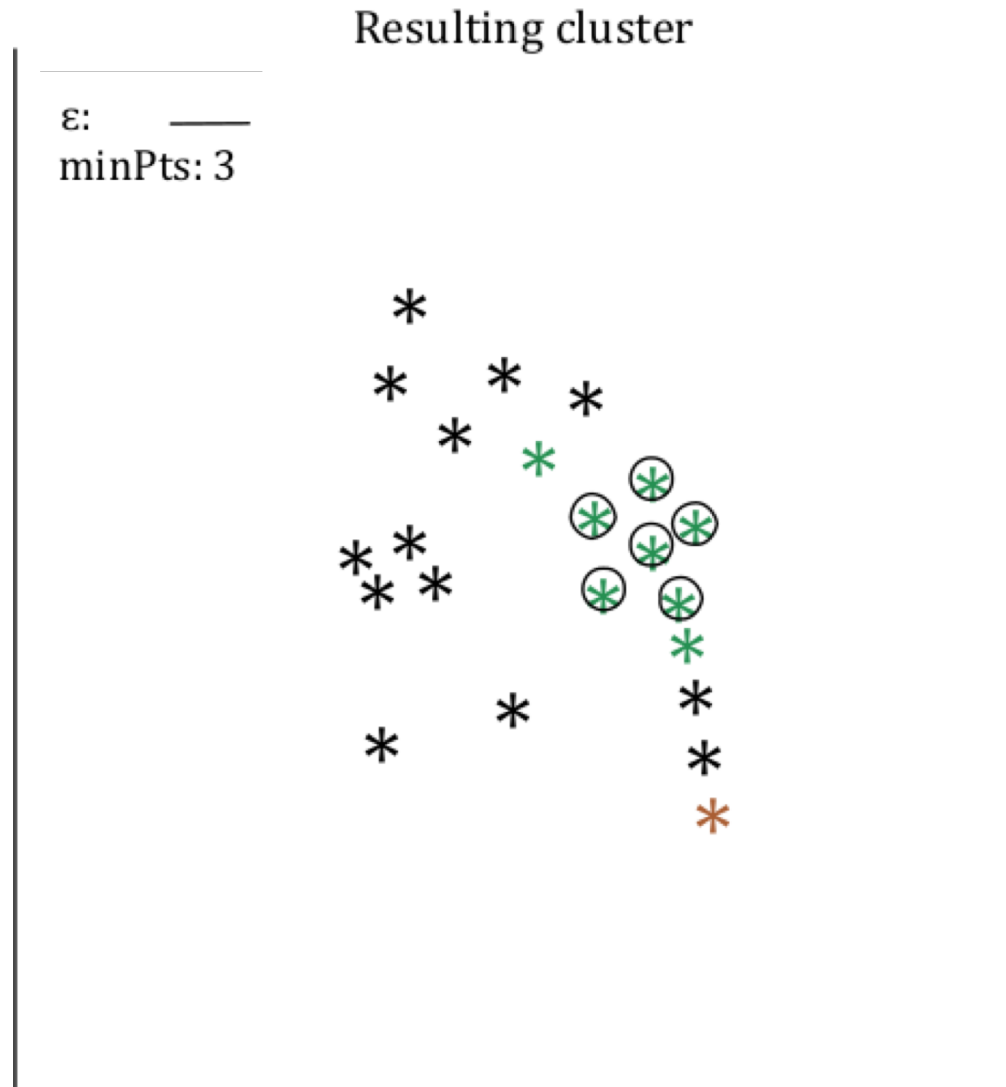
DBSCAN Example – Artificial Dataset



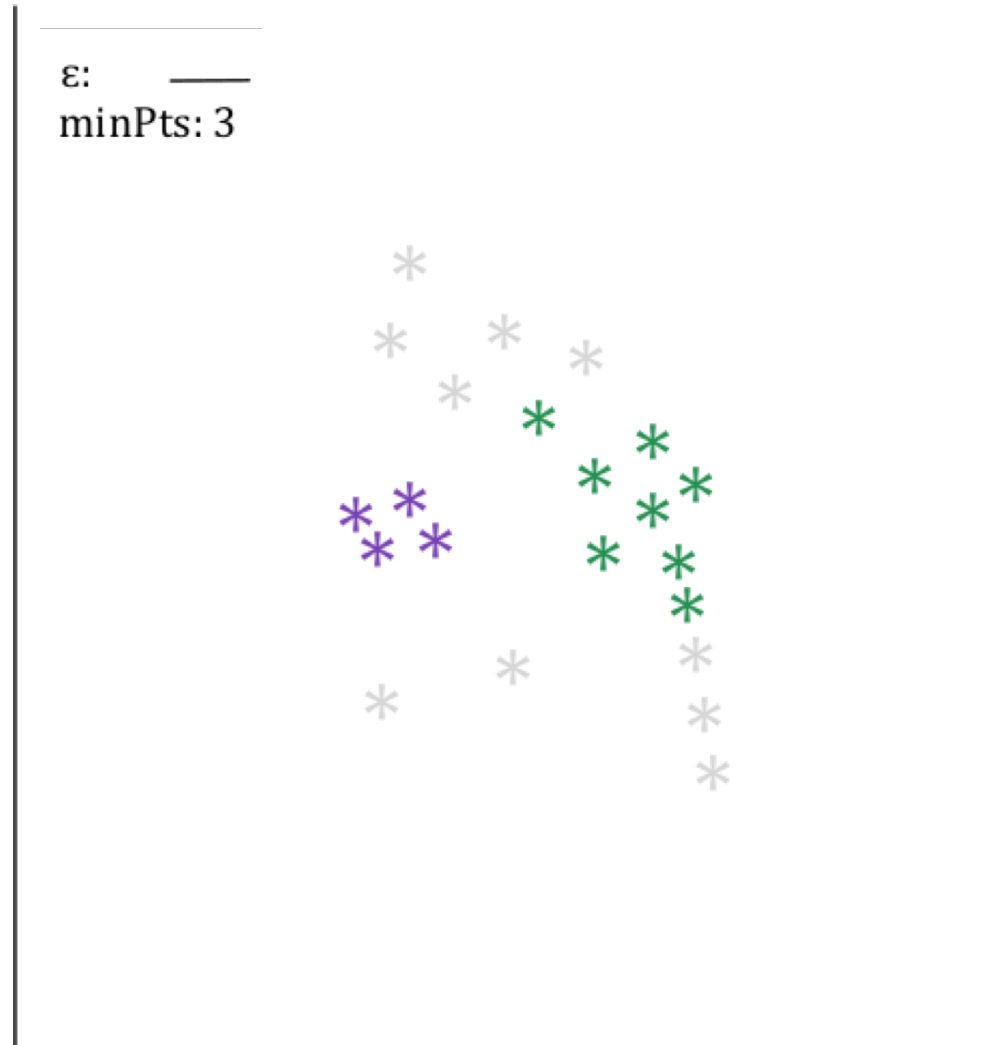
DBSCAN Example – Artificial Dataset

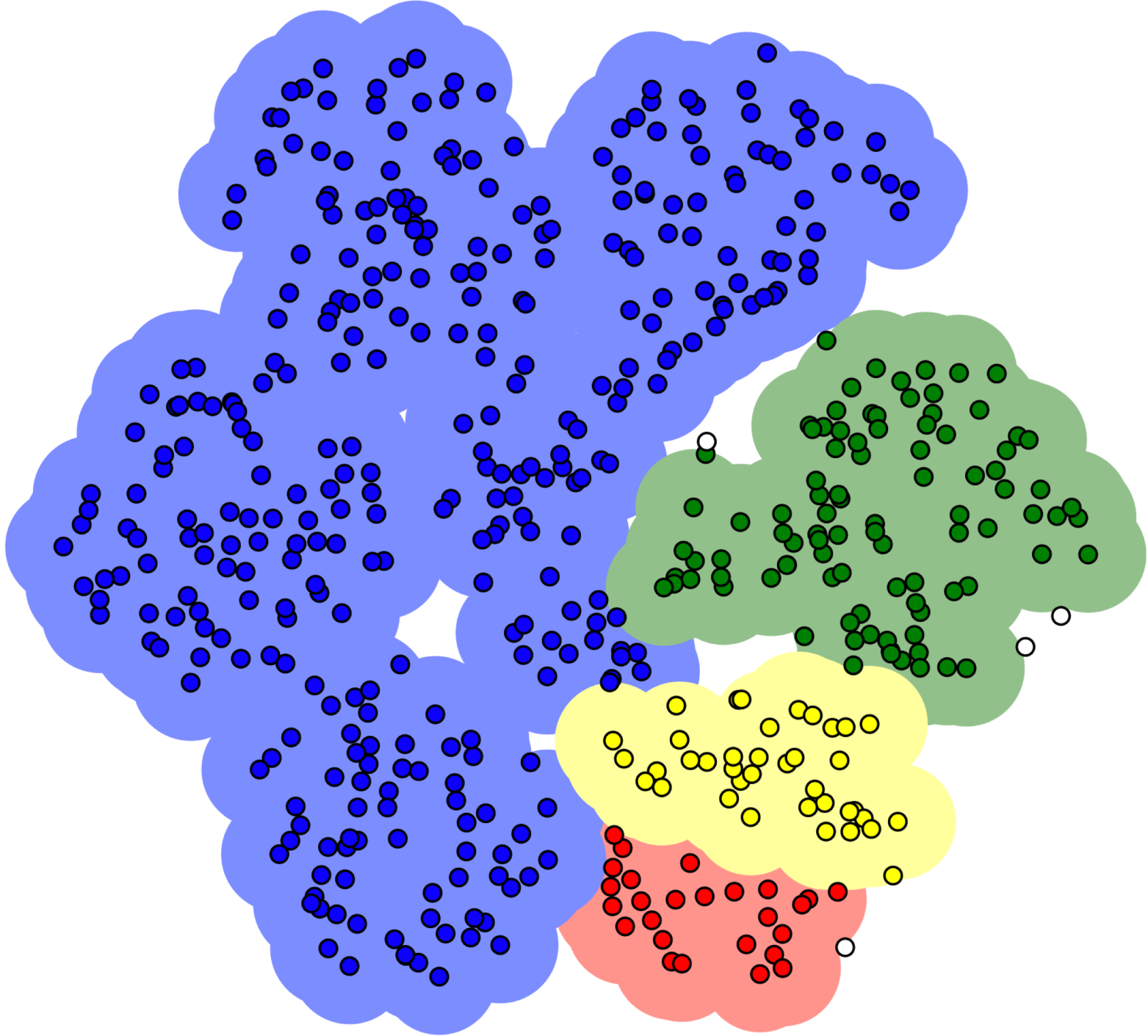


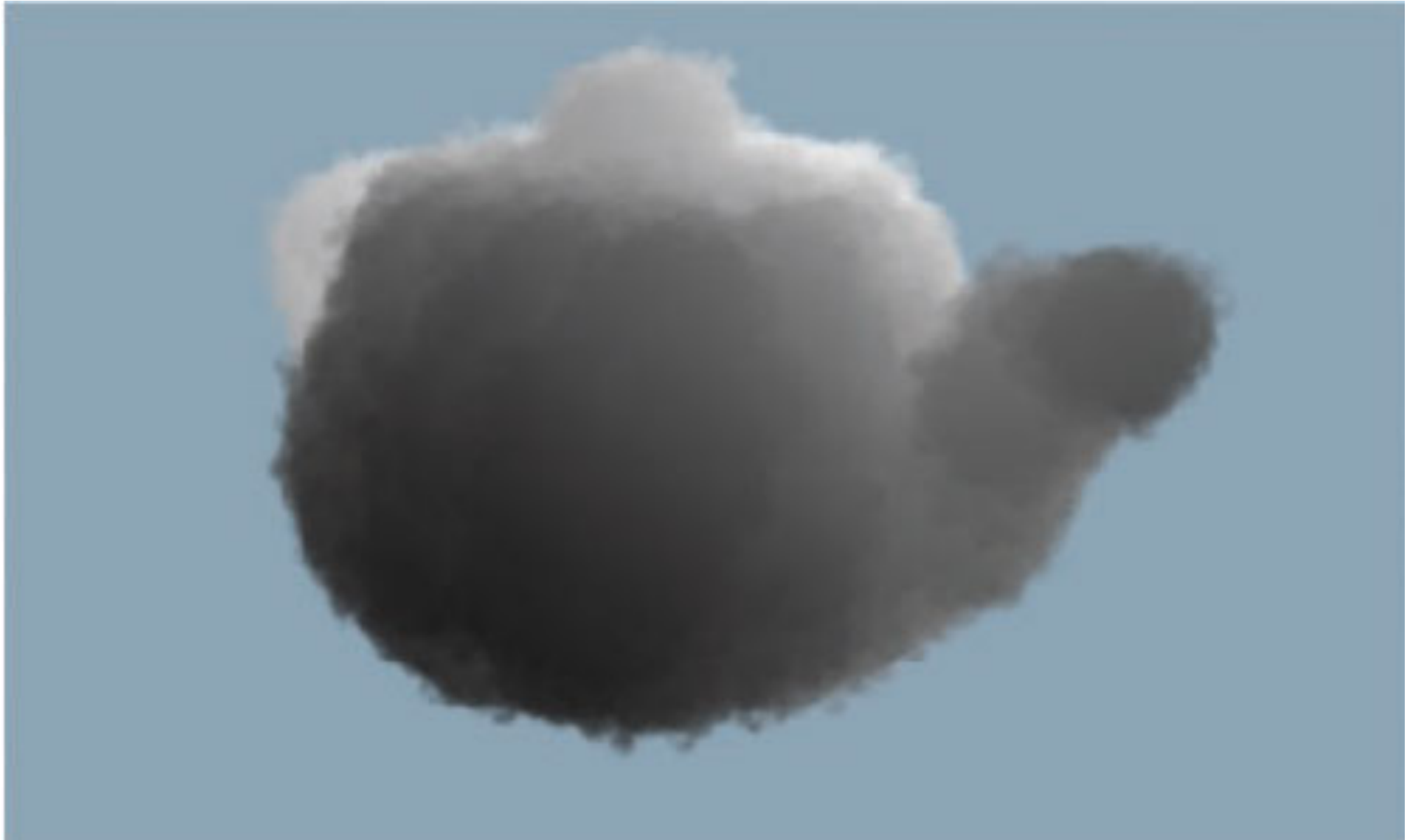
DBSCAN Example – Artificial Dataset



DBSCAN Example – Artificial Dataset







[Adapted from https://library.creativecow.net/articles/ussing_jonas/clouds_3dmax.php]

DBSCAN Clustering Challenge

	tpx	tpy
[1,]	-100	10928.249
[2,]	-99	10376.446
[3,]	-98	9696.948
[4,]	-97	10049.223
[5,]	-96	9420.883
[6,]	-95	9171.636
[7,]	-94	9118.230
[8,]	-93	9166.522
[9,]	-92	9251.633
[10,]	-91	9059.243
[11,]	-90	8390.208
[12,]	-89	8186.269
[13,]	-88	7749.231
[14,]	-87	8092.249
[15,]	-86	7518.351
[16,]	-85	7674.044
[17,]	-84	7194.916
[18,]	-83	7340.763
[19,]	-82	7456.145
[20,]	-81	6990.375



tpx

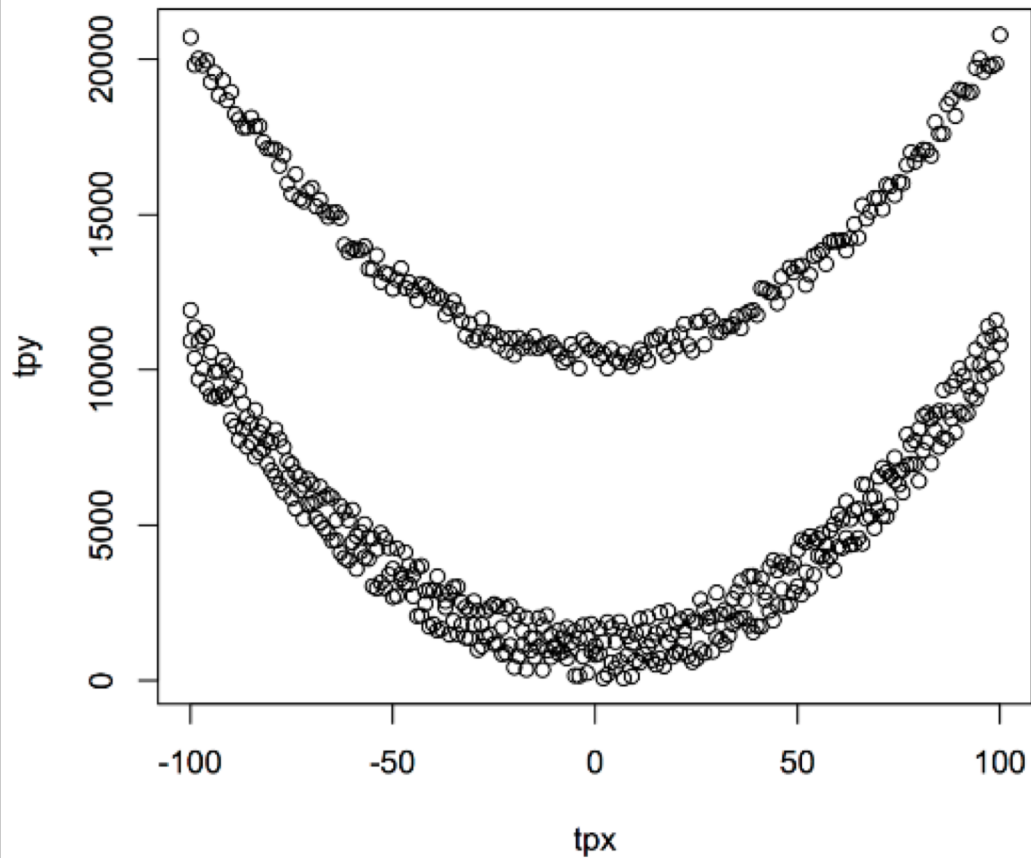


tpy

1 dimensional plot of points from each column.

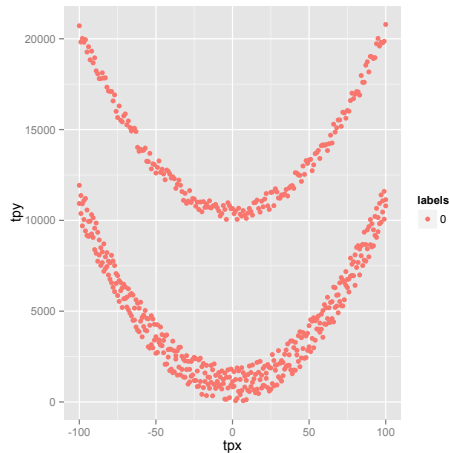
First 20 of 603 data points from an artificially-constructed dataset.

DBSCAN Clustering Challenge

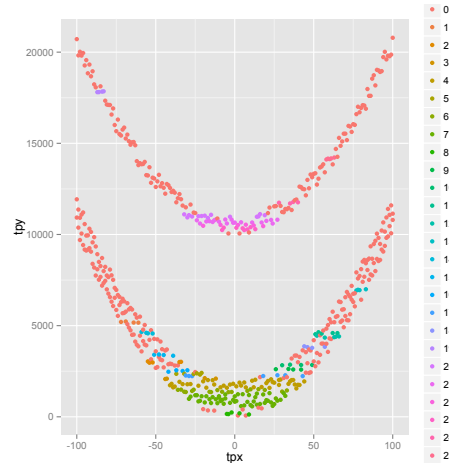


This looks like something DBSCAN
should be able to handle...
... and better than k -means, too.

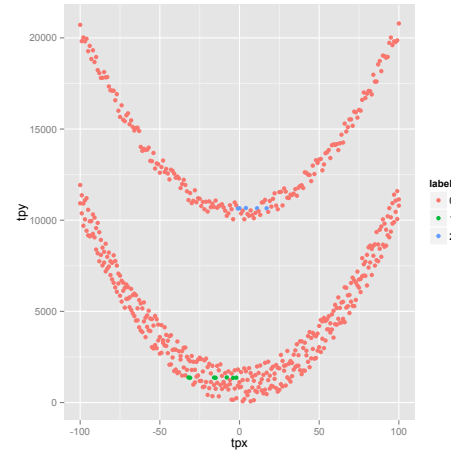
DBSCAN Clustering Challenge



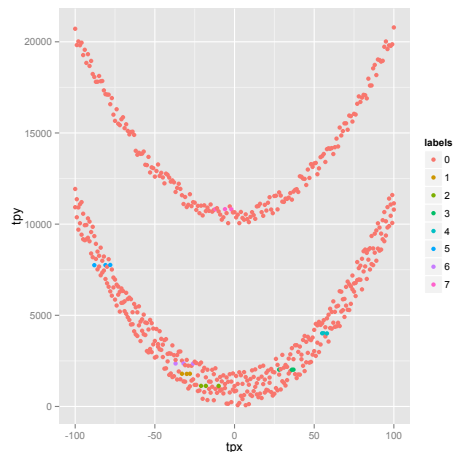
eps = 0.5 , minpts = 10 ,
0 clusters, 603 noise points



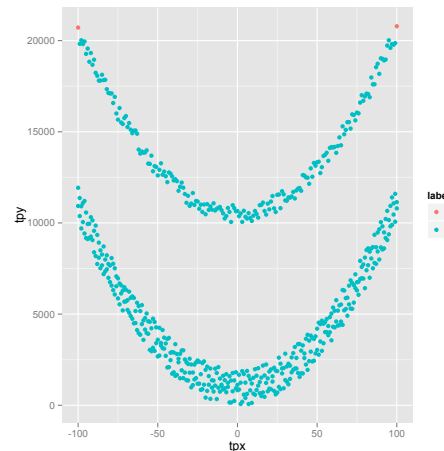
eps = 50, minpts = 4,
25 clusters, 357 noise points



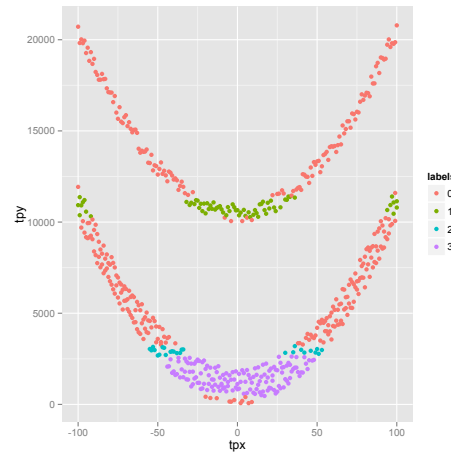
eps = 20, minpts = 5 ,
2 clusters, 591 noise points



eps = 10, minpts = 3,
7 clusters, 582 noise points



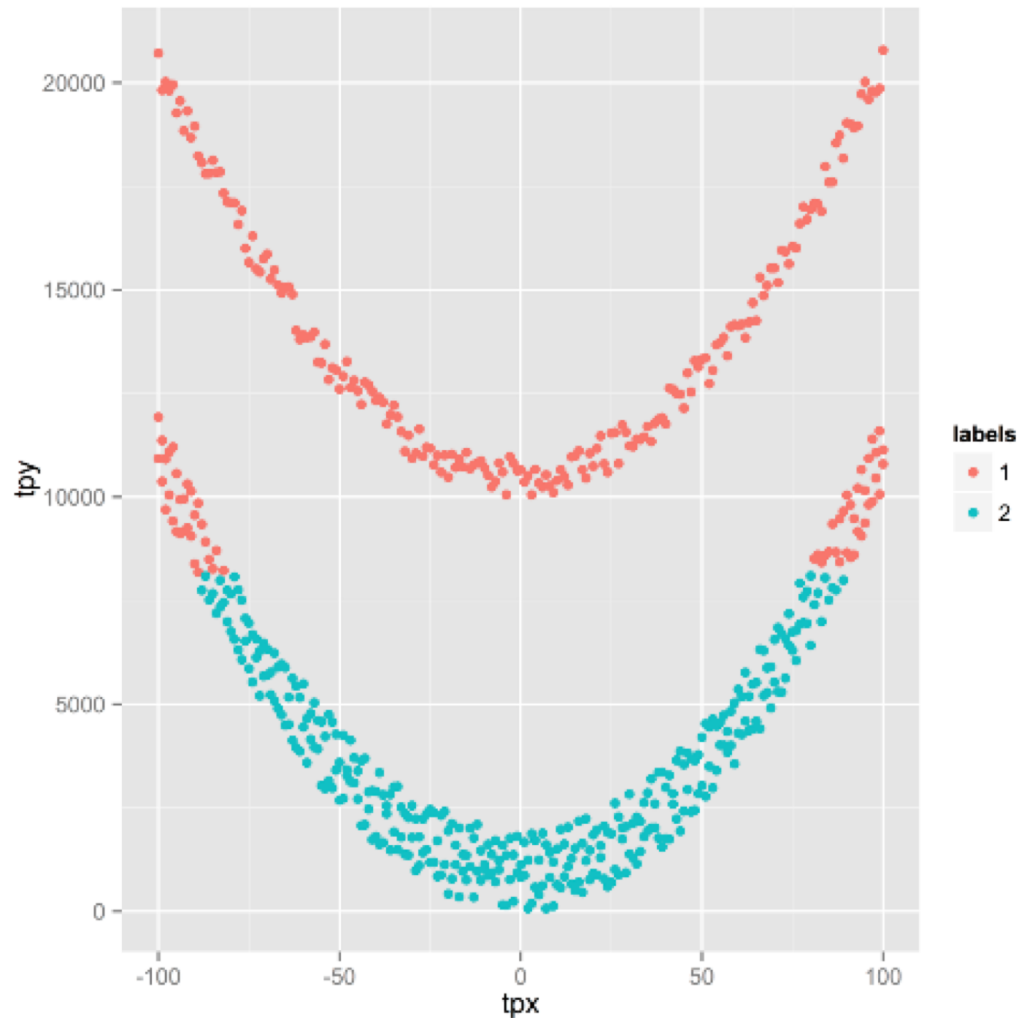
eps = 500, minpts = 5 ,
1 cluster, 2 noise points



eps = 200, minpts = 20 ,
3 clusters, 357 noise points

But it turns out that
DBSCAN isn't working
out so well for this
dataset...

DBSCAN Clustering Challenge

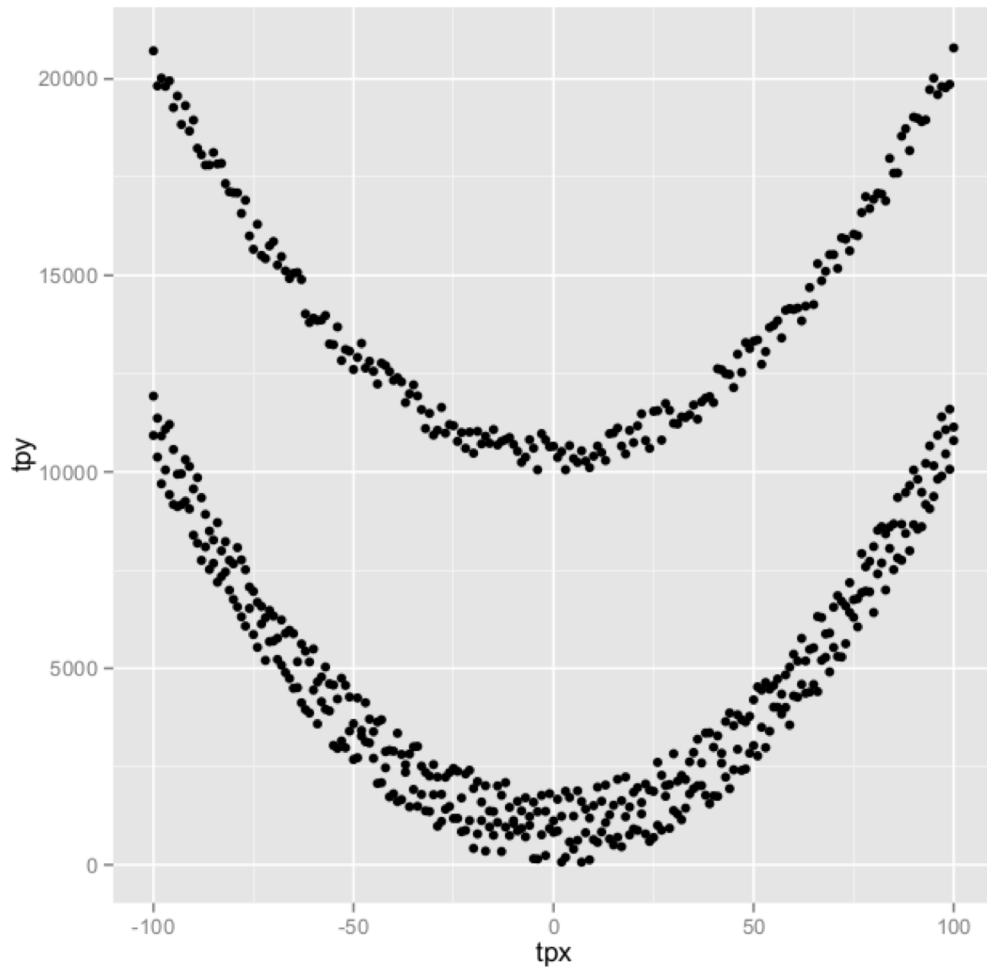


k-means appears to be doing a better job.

But is it really detecting the clusters more accurately, or just taking advantage of the separation between the two clusters?

Is there a way to get DBSCAN to work?

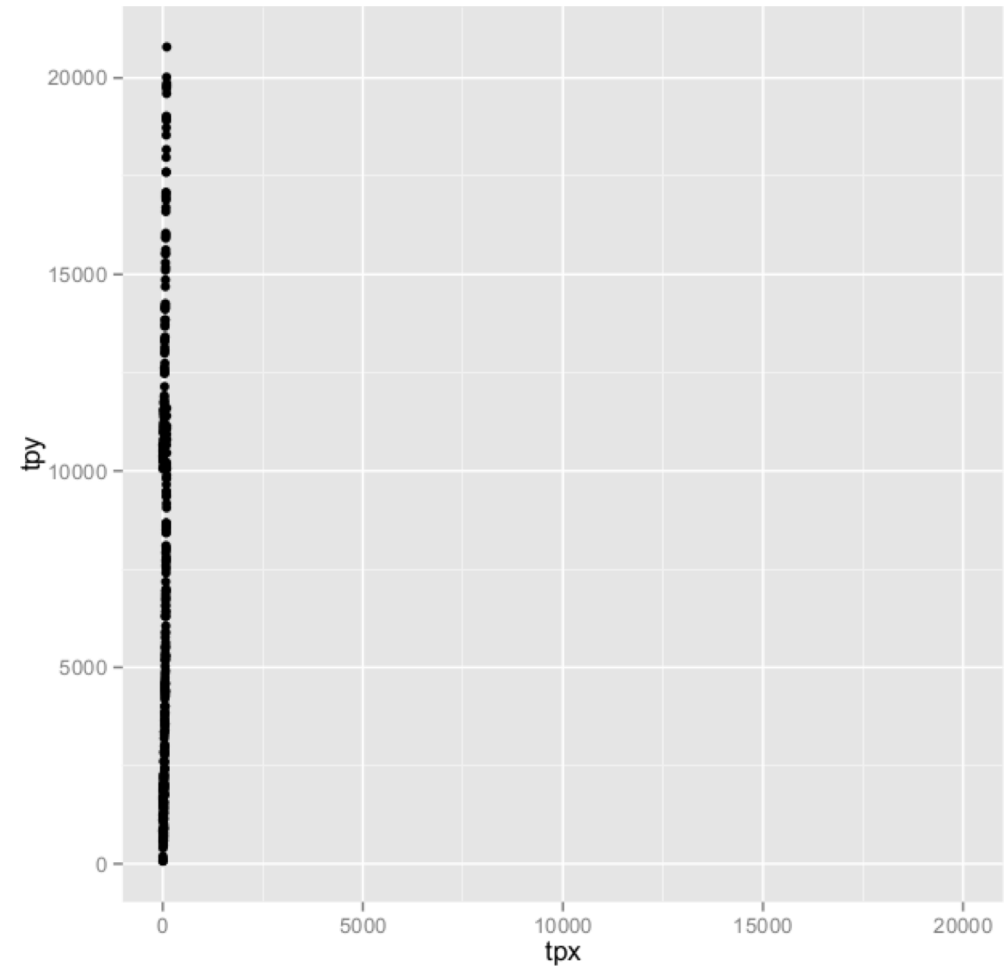
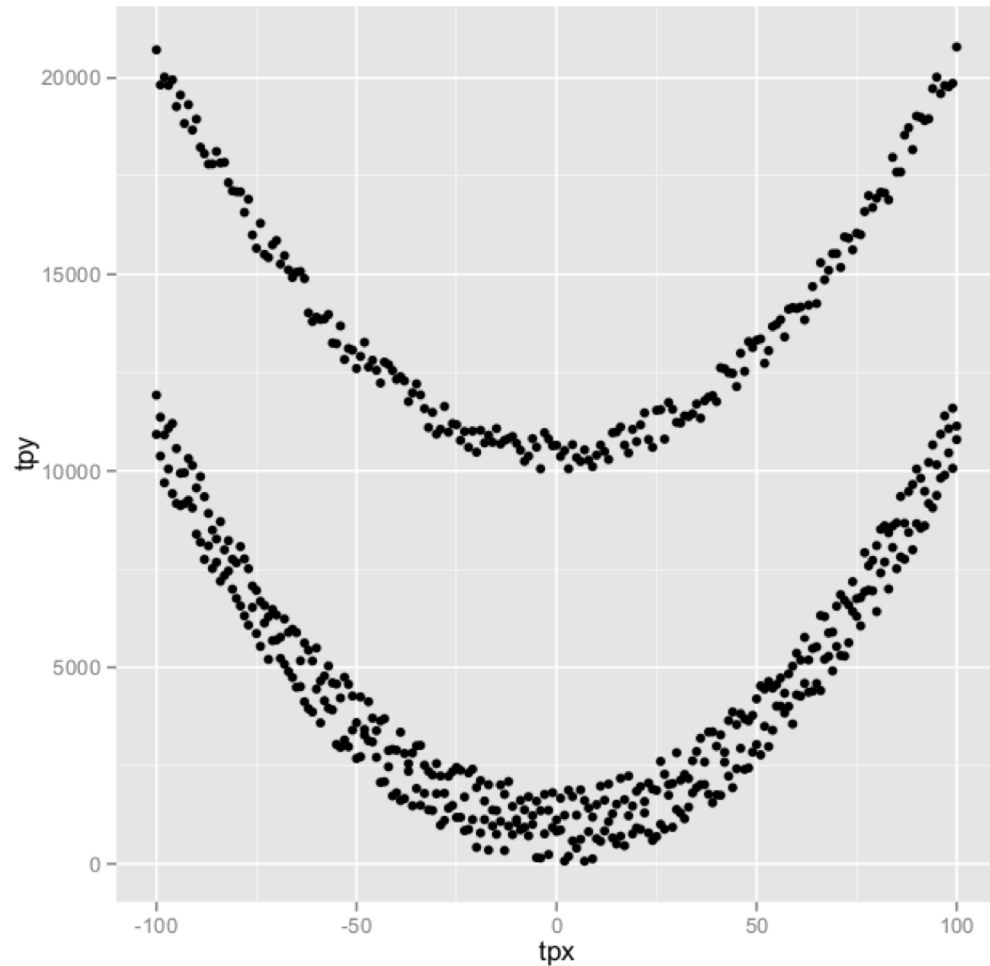
DBSCAN Clustering Challenge



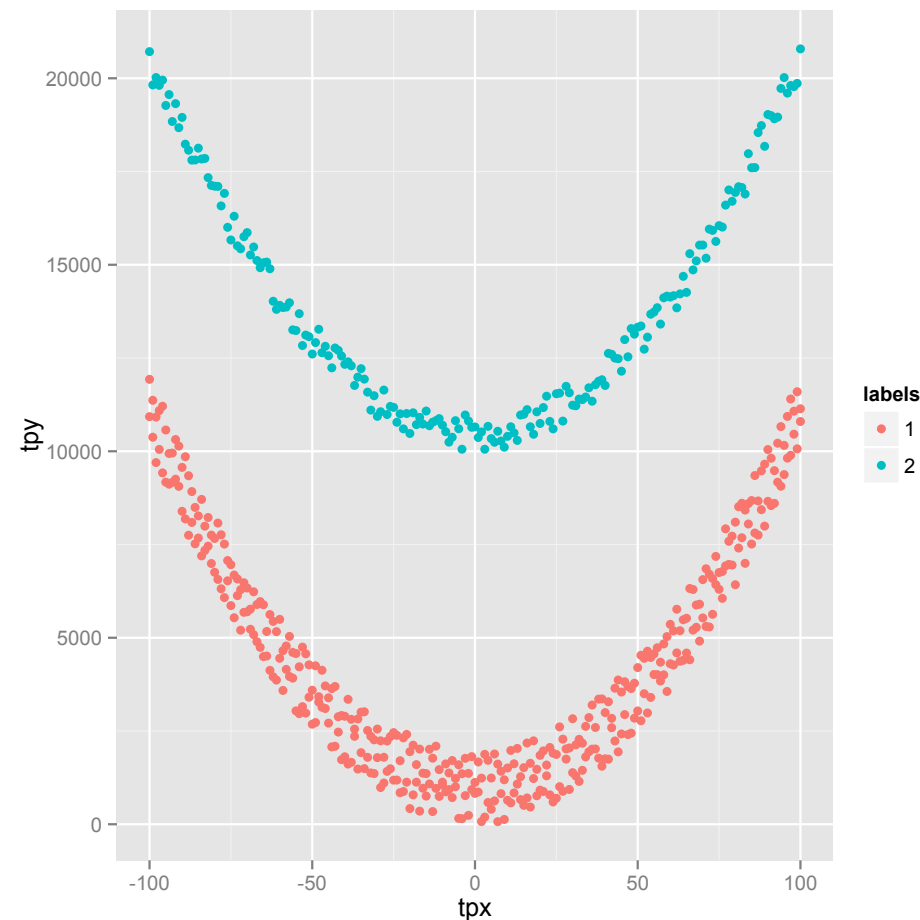
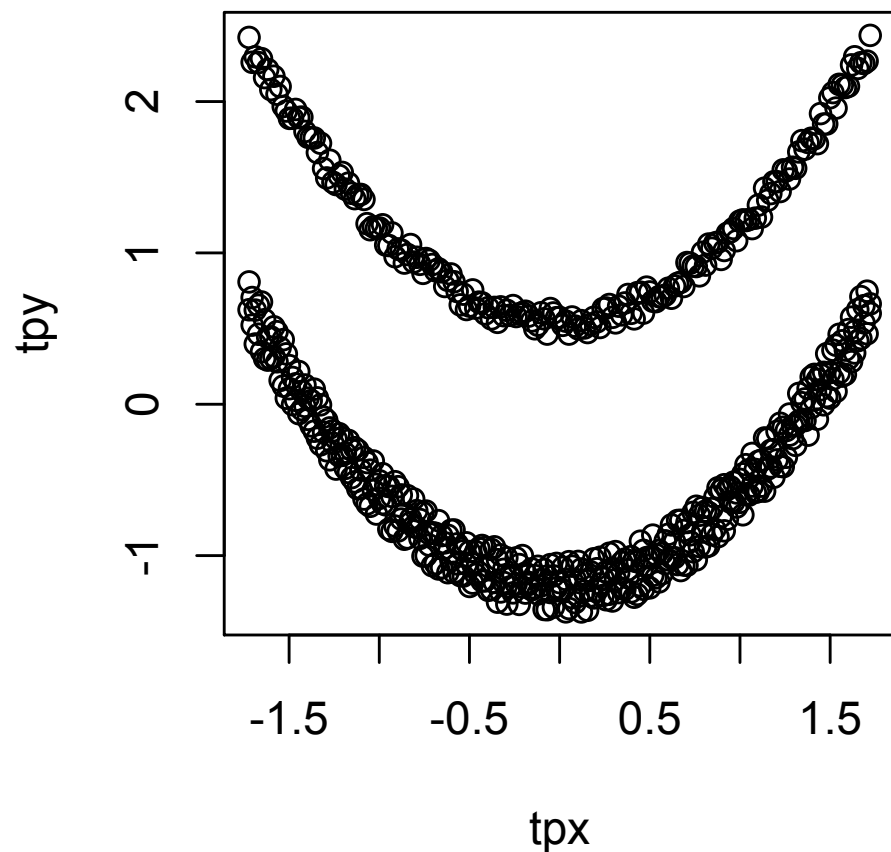
Take a closer look at the axes on this plot...

DBSCAN Clustering Challenge

Re-plotted, with the axis adjusted to match the axis:

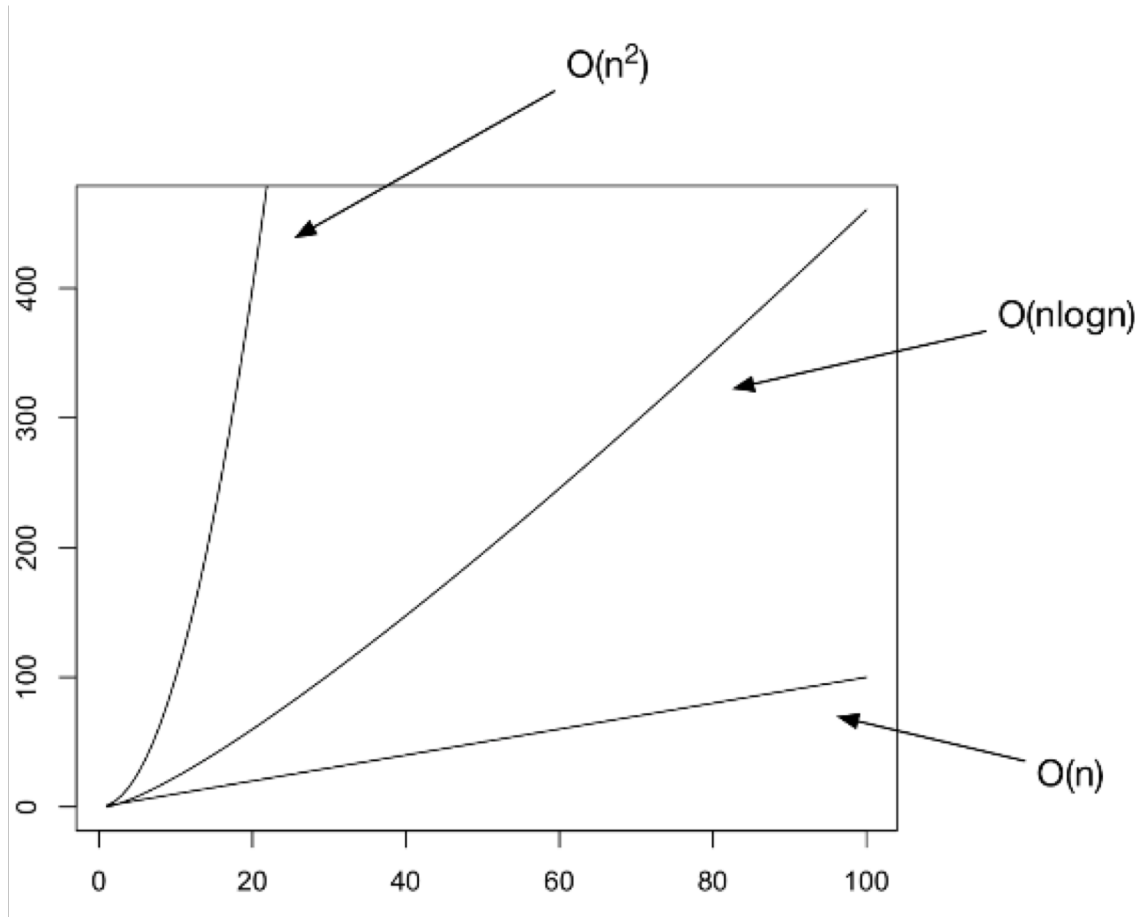


DBSCAN Clustering Challenge



When data is scaled, position and length of vectors are adjusted to normalize the distribution of the data. This pleases DBSCAN!

Comparing Algorithmic Complexity



DBSCAN can handle globular clusters and non-globular clusters — why isn't it being used all the time?

DBSCAN is $O(n \log n)$ in the best case scenario, whereas k -means is $O(nk)$ (more or less)

When the number of observations increases, DBSCAN is less efficient than k -means .

DBSCAN Advantages

No need to specify the number of clusters.

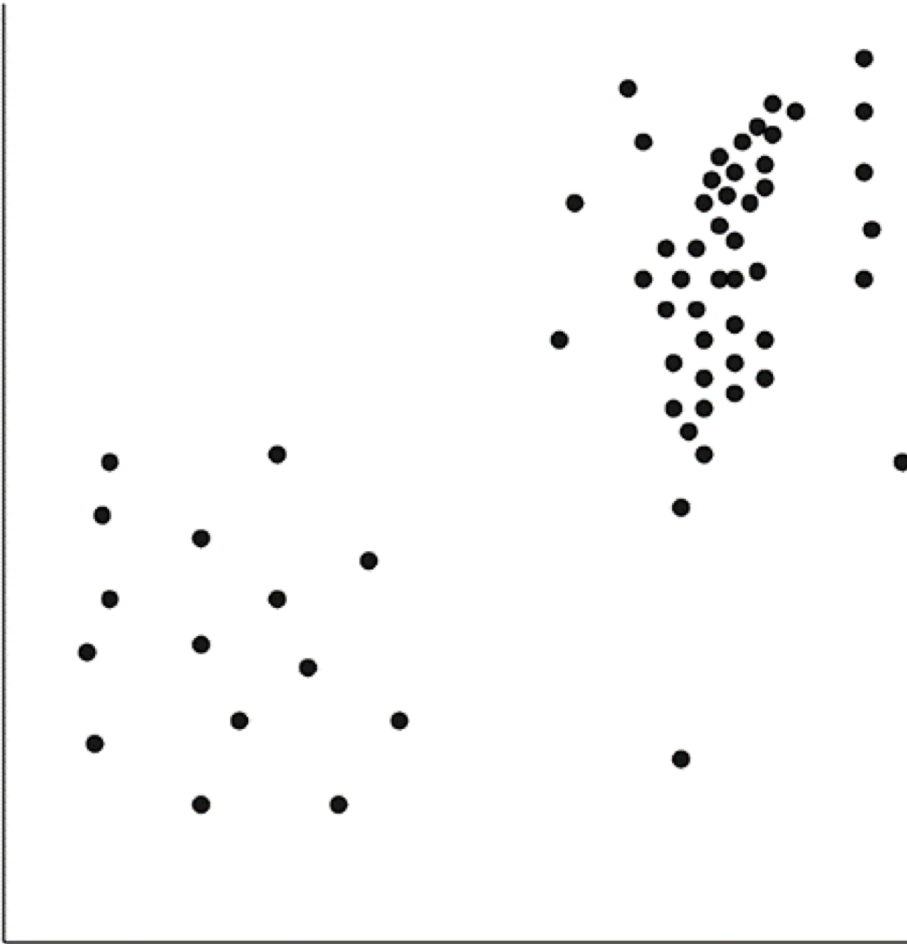
Can find arbitrarily shaped clusters.

Can recognize “noisy” points.

Robust to outliers.

Requires only two parameters (minPts and ε) which can be set by domain experts if the data is well understood.

DBSCAN Limitations



DBSCAN's clustering **kryptonite**:
datasets where cluster density is not
consistent across clusters.

Hard to set parameters that consistently
capture clusters while identifying
outliers.

Not entirely deterministic: border
points that are reachable from more
than one cluster can be part of either
cluster, depending on the order.

Parameters Estimation

minPts:

- $\text{minPts} \geq \# \text{ features} + 1$
- larger values are better for noisy data sets
- $\text{minPts} \geq 2 \times \text{dim}$ for large datasets or sets with duplicates

ε :

- if too small, a large prop. of observations is not clustered
- if too high, majority of observations are in the same cluster
- in general, small values are preferable

Distance function:

- has a major impact on the results
- should be selected before ε is chosen

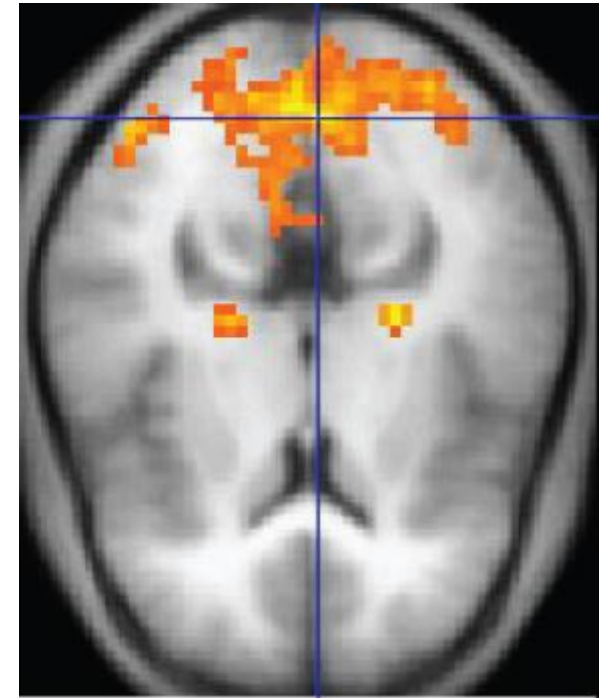
DBSCAN Examples

Detecting Alzheimer's Disease

Mild cognitive impairments (MCI) are a known to be a risk factor for development of Alzheimer's Disease

MCI are accompanied by changes in brain structure

But which changes indicate that people will go on to develop Alzheimer's?



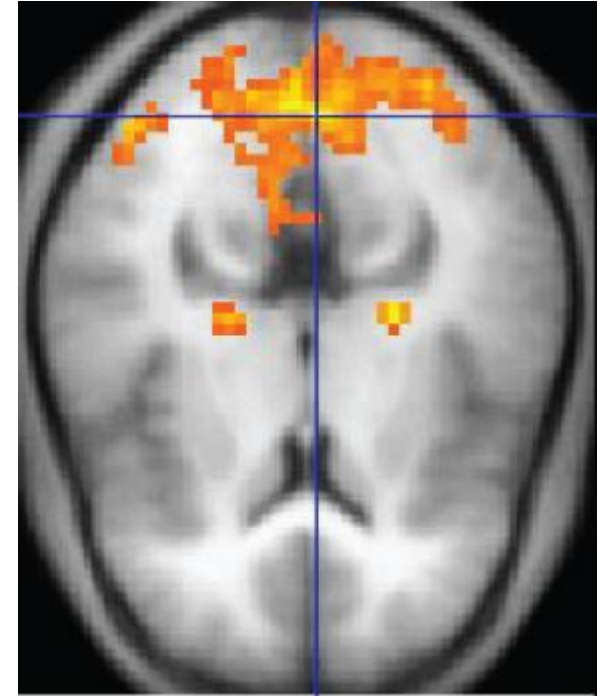
FMRI highlighting some areas of the pre-frontal cortex.

DBSCAN Examples

Detecting Alzheimer's Disease

A number of different data science techniques applied to MRI data:

- Support Vector Machines
- Bayesian Statistics
- Voting Feature Intervals
- Feature Extraction
- DBSCAN



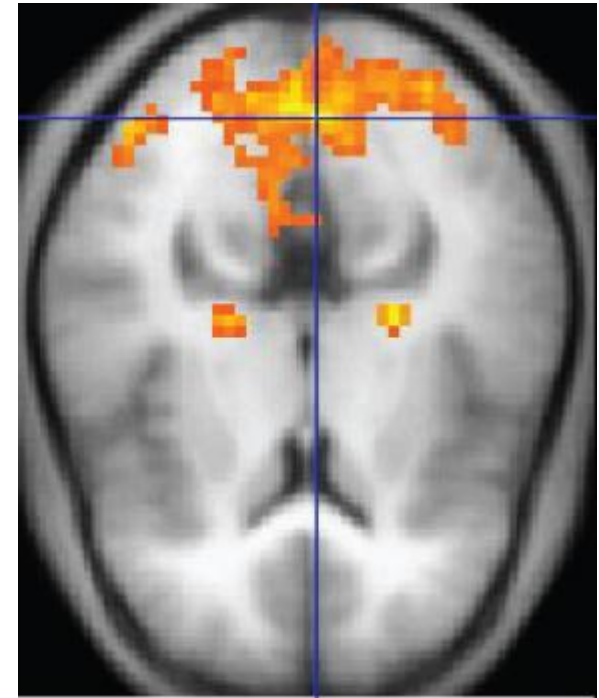
FMRI highlighting some areas of the pre-frontal cortex.

DBSCAN Examples

Detecting Alzheimer's Disease

DBSCAN is used once voxels that provide high information about the classification of the image are identified using entropy based measures

DBSCAN then groups pixels with similar spatial and information levels to determine which parts of the brain are the most important for the diagnosis



FMRI highlighting some areas of the pre-frontal cortex.

DBSCAN Examples

Some More Examples

A novel approach for predicting the length of hospital stay with DBSCAN and supervised classification algorithms

Simulation of DNA damage clustering after proton irradiation using an adapted DBSCAN algorithm

Where traffic meets DNA: mobility mining using biological sequence analysis revisited

Individual Movements and Geographical Data Mining. Clustering Algorithms for Highlighting Hotspots in Personal Navigation Routes

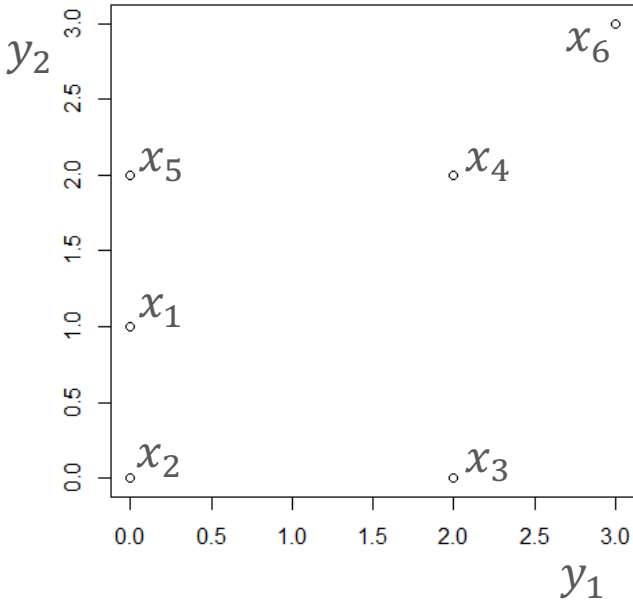
Spectral Clustering

Clustering in General

Spectral Clustering Overview

Dataset

i	y_1	y_2
x_1	0	1
x_2	0	0
x_3	2	0
x_4	2	2
x_5	0	2
x_6	3	3

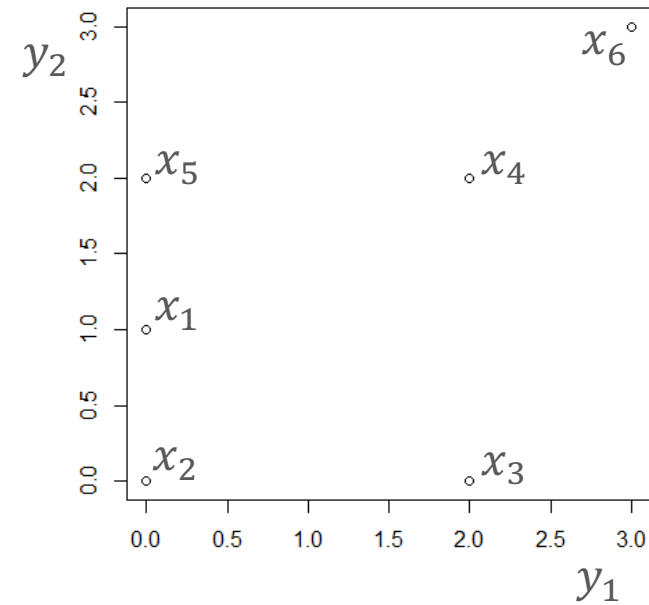


Clustering in General

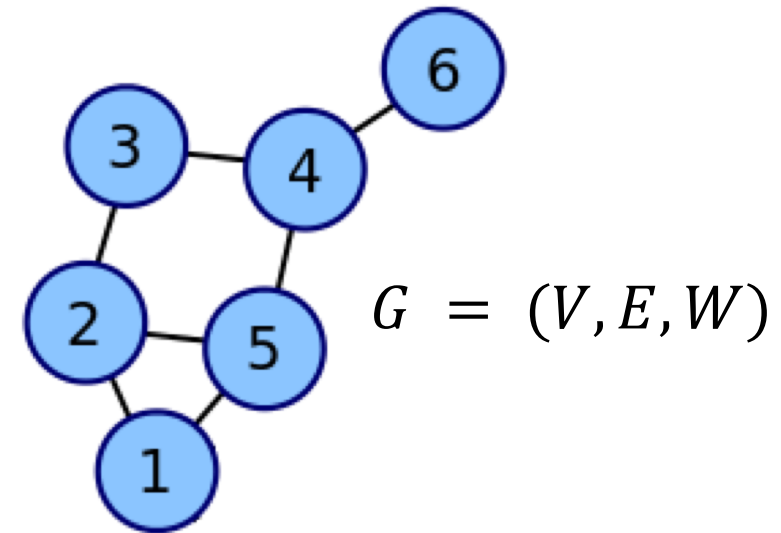
Spectral Clustering Overview

Dataset

i	y_1	y_2
x_1	0	1
x_2	0	0
x_3	2	0
x_4	2	2
x_5	0	2
x_6	3	3

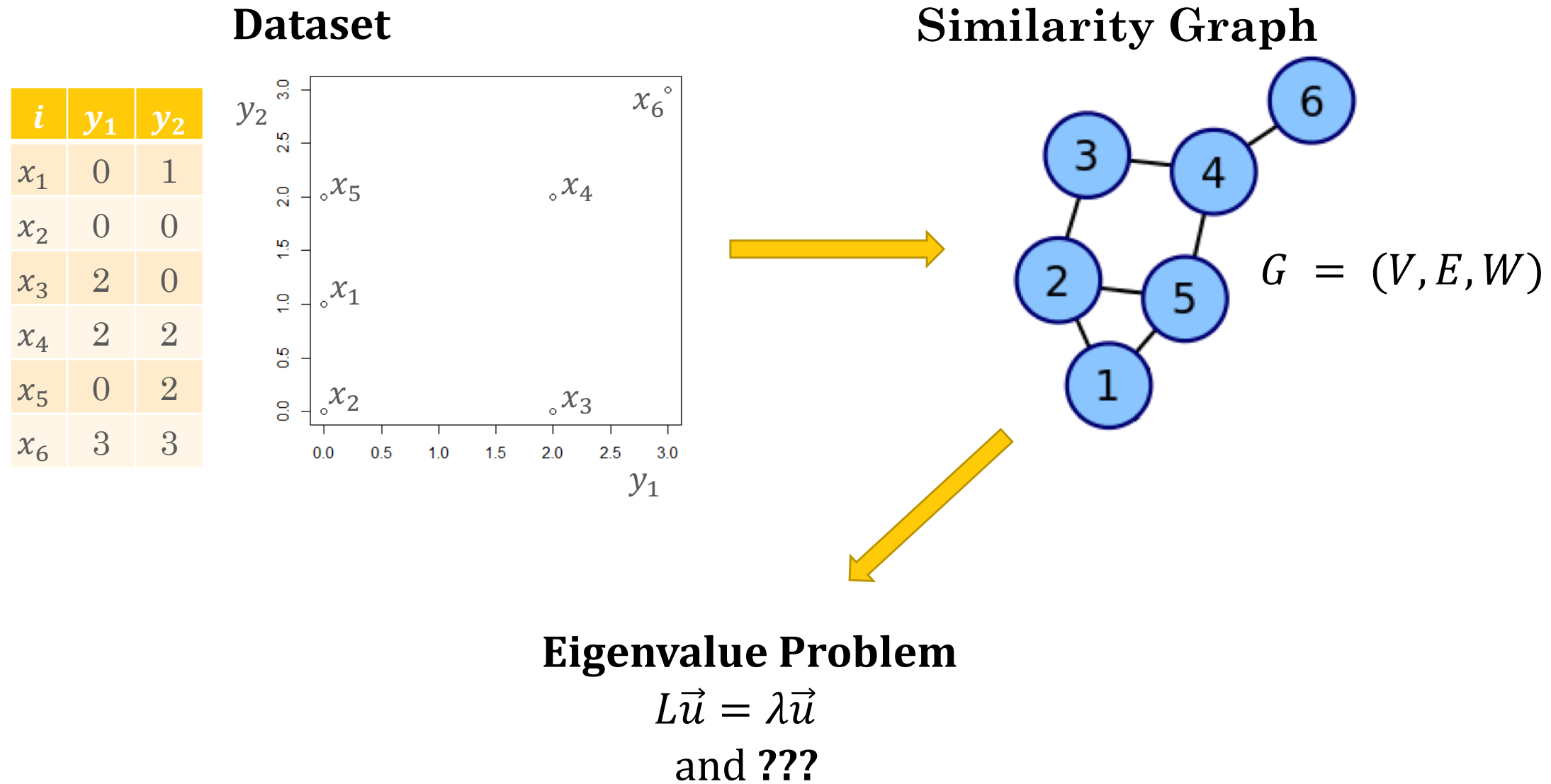


Similarity Graph



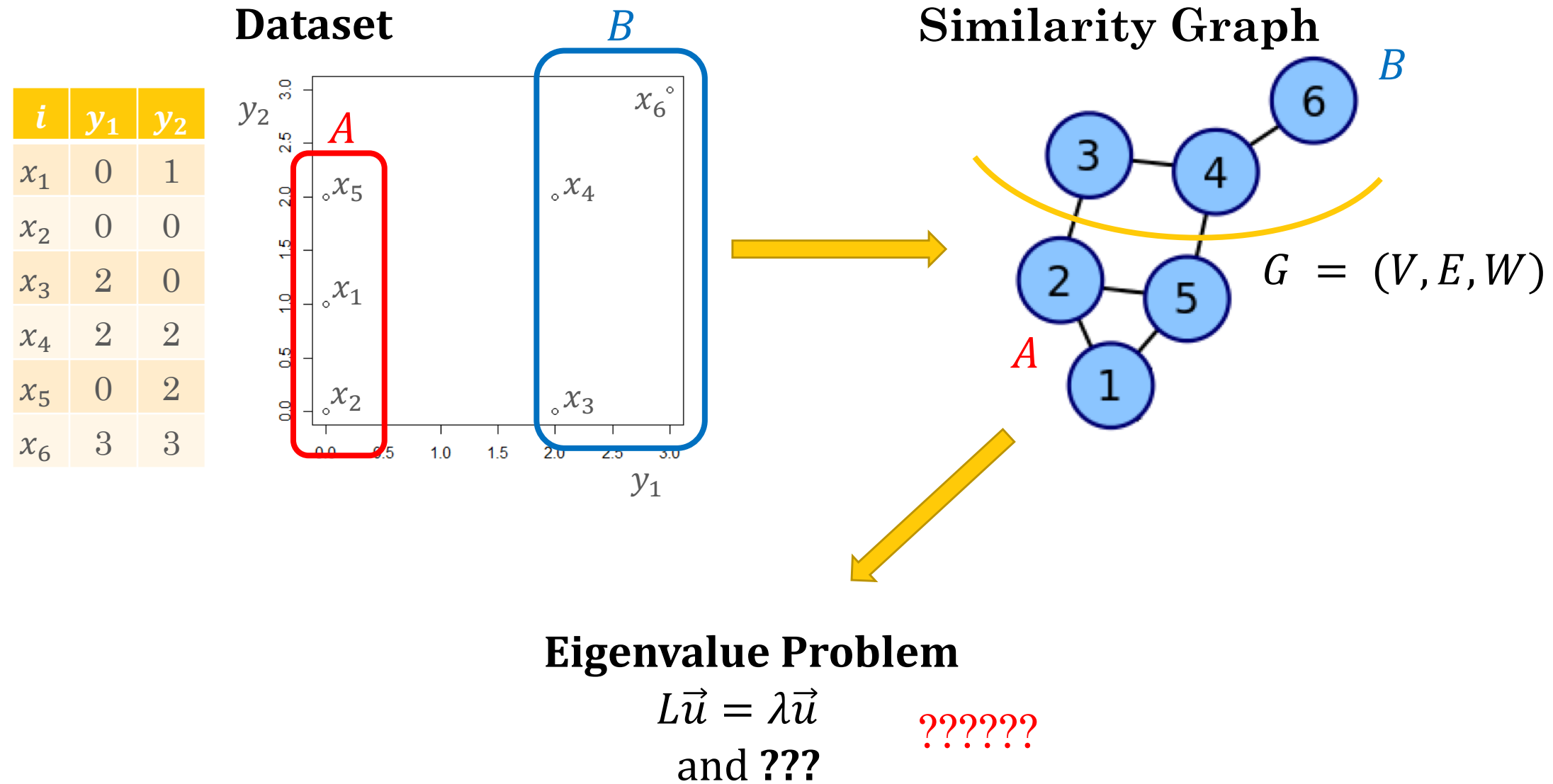
Clustering in General

Spectral Clustering Overview



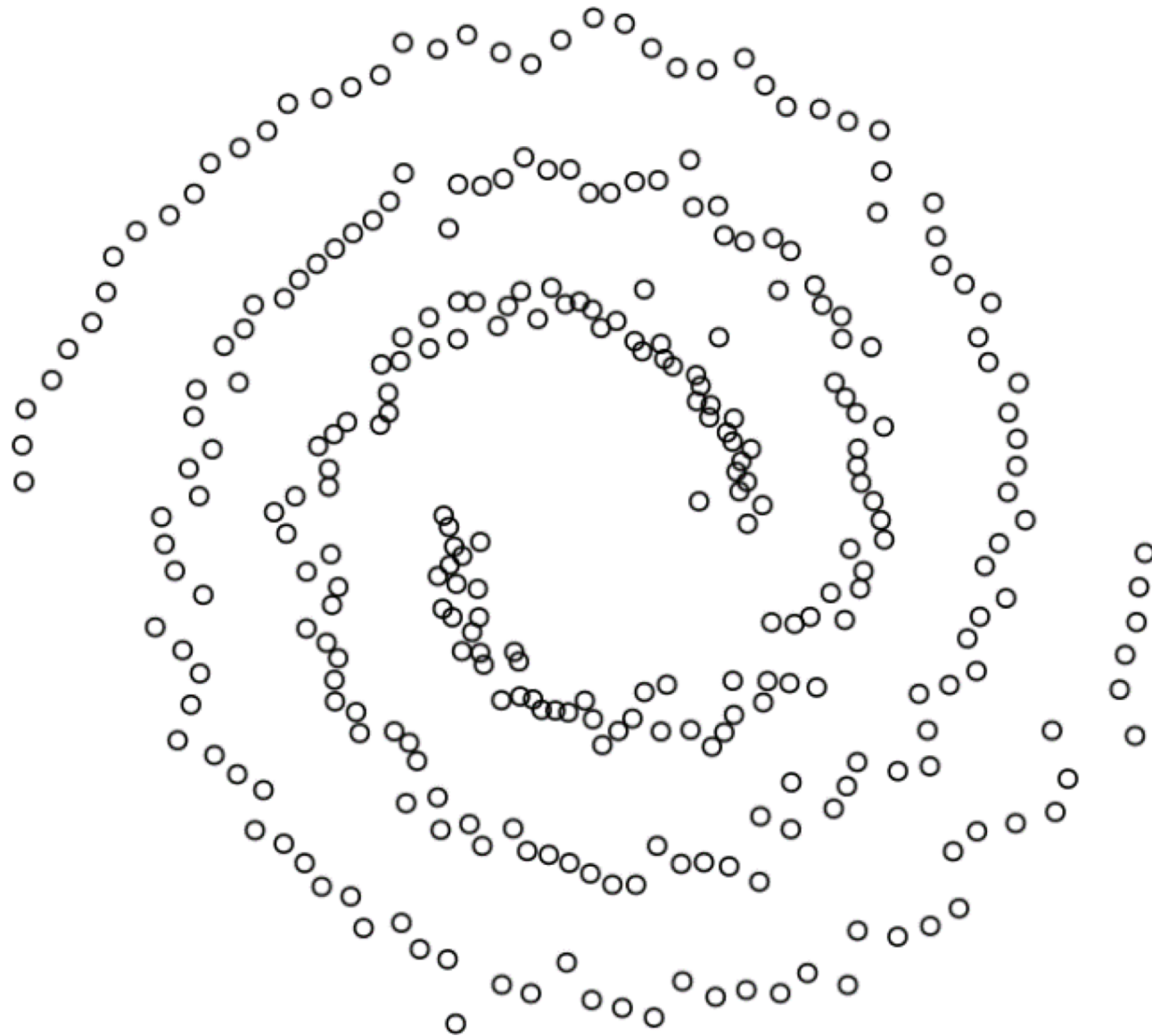
Clustering in General

Spectral Clustering Overview



Clustering in General

Motivation for Spectral Clustering



Spectral clustering makes no assumption on the shape of a cluster

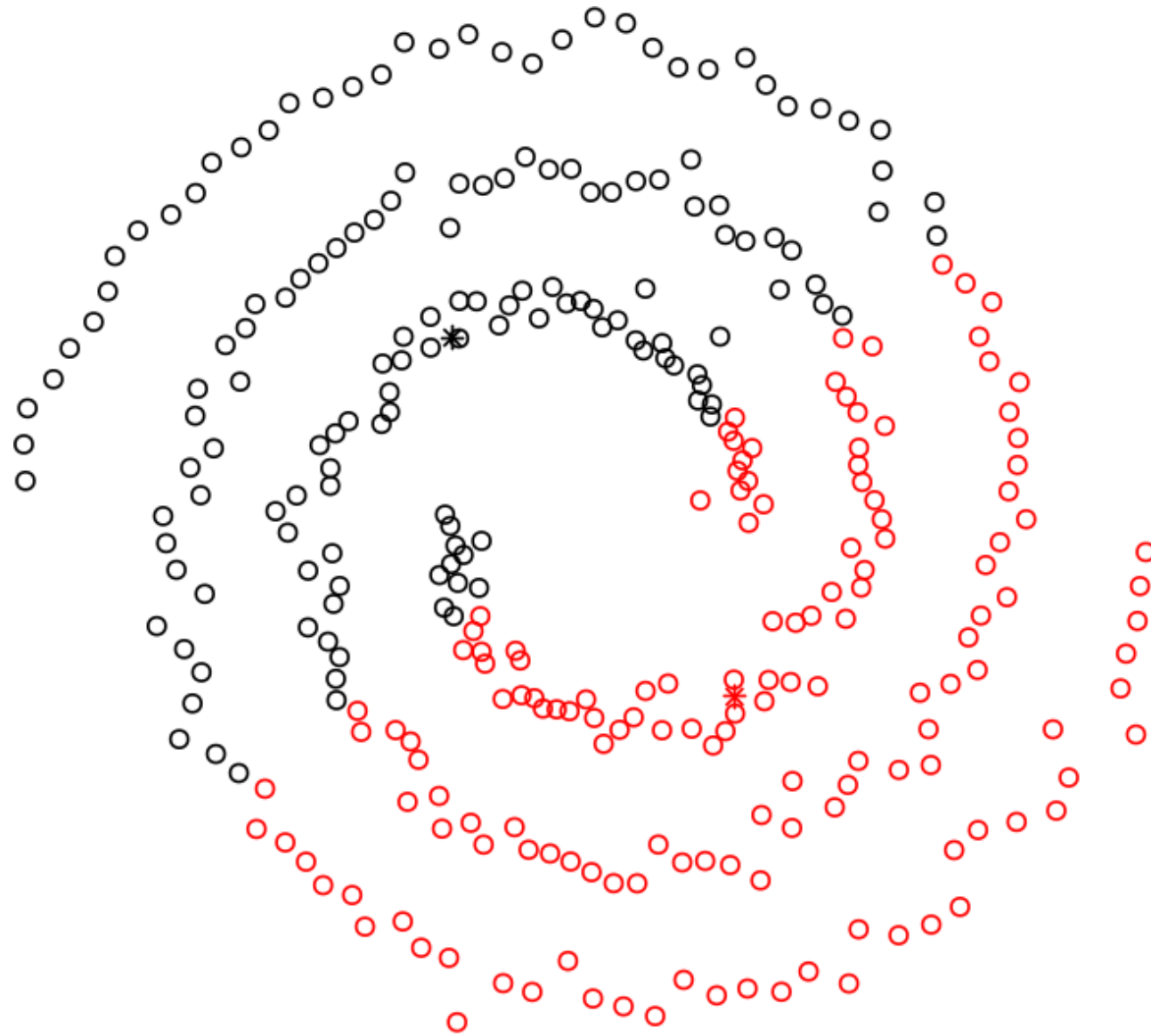
- compactness vs. connectedness

Spectral clustering can be implemented efficiently for large datasets

- computing eigenvalues is numerically “efficient”

Clustering in General

Motivation for Spectral Clustering



Spectral clustering makes no assumption on the shape of a cluster

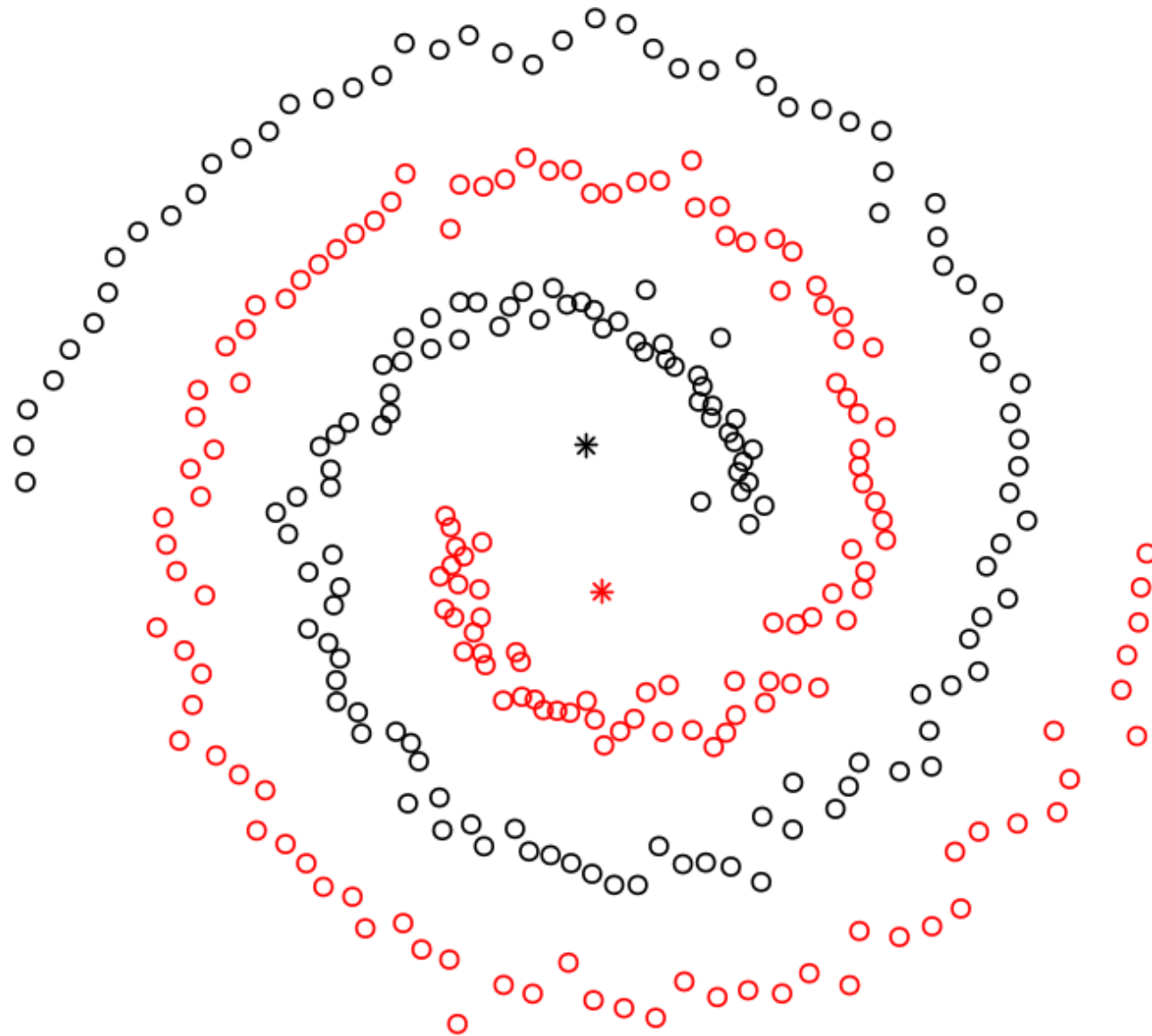
- compactness vs. connectedness

Spectral clustering can be implemented efficiently for large datasets

- computing eigenvalues is numerically “efficient”

Clustering in General

Motivation for Spectral Clustering



Spectral clustering makes no assumption on the shape of a cluster

- compactness vs. connectedness

Spectral clustering can be implemented efficiently for large datasets

- computing eigenvalues is numerically “efficient”

Data Pre-Processing

Similarity Graph

In graph theory, the notation of a **similarity graph** is $G = (V, E, W)$.

1. Data points x are **vertices** $v \in V$.
2. A pair of vertices v_i, v_j are connected by an **edge** $e_{ij} = 1$ if the **similarity weight** $w_{ij} > \tau$ for a given threshold $\tau \in [0,1)$.
3. The edges e_{ij} form the **adjacency matrix** E .
4. The weights w_{ij} form the **similarity matrix** W .
5. The (diagonal) **degree matrix** D provides information about the number of edges attached to a vertex: $d_{ii} = \sum_{j=1}^n e_{ij}$.

External requirements: threshold τ , similarity measure w .

Data Pre-Processing

Similarity Graph – Example

With the **Gower** similarity measure on data with m features

$$w_G(v_i, v_j) = 1 - \frac{1}{m} \sum_{k=1}^m \frac{|x_{i,k} - x_{j,k}|}{\text{range of } k^{\text{th}} \text{ feature}}$$

the similarity matrix of the previous data is

$$W = \begin{pmatrix} 0 & 5/6 & 1/2 & 1/2 & 5/6 & 1/6 \\ 5/6 & 0 & 2/3 & 1/3 & 2/3 & 0 \\ 1/2 & 2/3 & 0 & \boxed{2/3} & 1/3 & 1/3 \\ 1/2 & 1/3 & \boxed{2/3} & 0 & 2/3 & 2/3 \\ 5/6 & 2/3 & 1/3 & 2/3 & 0 & 1/3 \\ 1/6 & 0 & 1/3 & 2/3 & 1/3 & 0 \end{pmatrix}$$

For instance, $w_G(v_3, v_4) = \mathbf{w_{34}} = \mathbf{w_{43}} = 1 - \frac{1}{2} \left\{ \frac{|x_{3,1} - x_{4,1}|}{r_1} + \frac{|x_{3,2} - x_{4,2}|}{r_2} \right\}$.

But $r_1 = r_2 = 3$, so $w_{34} = w_{43} = 1 - \frac{1}{6} \{|2 - 2| + |0 - 2|\} = \frac{2}{3}$.

Data Pre-Processing

Similarity Graph – Example

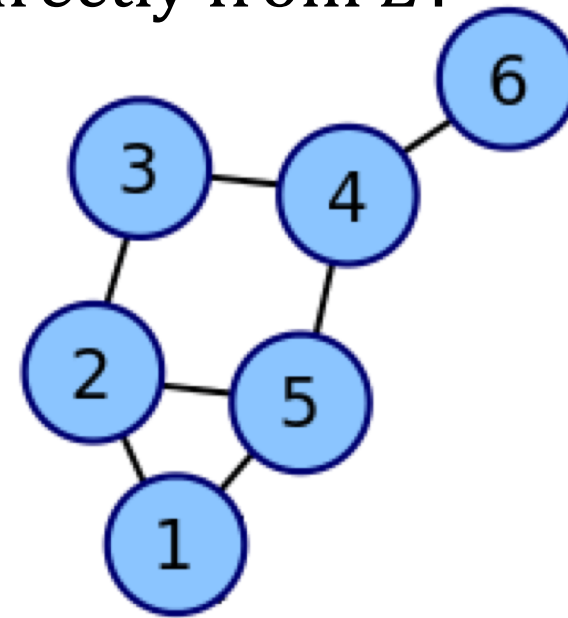
- Let's use a threshold value $\tau = 0.6$.
The adjacency matrix is thus

$$E = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

- Incidentally, the degree matrix is

$$D = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

- The similarity graph G is read directly from E :



- Now all that is left is to **partition** the graph!

Graph Partitions

Graph Cuts

- A **graph cut** partitions a graph into two sub-graphs (**clusters**) A, B .
- The goal is to partition the graph so that edges within a group have large weights (so the vertices they join are **similar**) and edges across groups have small weights (so the vertices they join are **dissimilar**).
- We focus on one way to do this: the **Normalized Cut**.

Other partition schemes: Min Cut, Ratio Cut, Min Max Cut
- An objective function $J(A, B)$ must be minimized against the set of all possible partitions (A, B) .
- The partition which minimizes J gives rise to the first clustering level.
- The procedure can be repeated as necessary on the cluster sub-graphs.

Graph Partitions

Normalized Cut – Example

Objective function:

$$J_{\text{NCut}} = \text{Cut}(A, B) \left(\frac{1}{\text{Vol}(A)} + \frac{1}{\text{Vol}(B)} \right)$$

Sum of all the weights on edges emanating from B

Sum of all the weights on edges emanating from A

Sum of all the weights on edges starting in one group and ending in the other

Advantages:

- Takes into consideration the size of partitioned groups
- Tends to avoid isolating vertices
- Takes into consideration intra-group variance

Limitations

- Not an easy optimization problem to solve (**NP-hard!!**)

Graph Partitions

Normalized Cut – Example

Objective function:

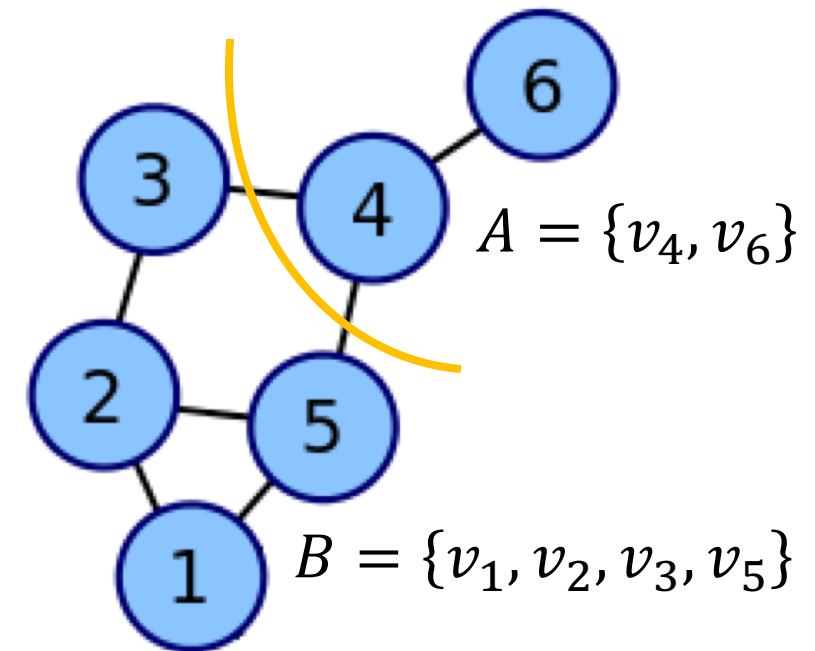
$$J_{\text{NCut}} = \text{Cut}(A, B) \left(\frac{1}{\text{Vol}(A)} + \frac{1}{\text{Vol}(B)} \right)$$

Advantages:

- Takes into consideration the size of partitioned groups
- Tends to avoid isolating vertices
- Takes into consideration intra-group variance

Limitations

- Not an easy optimization problem to solve (**NP-hard!!**)



$$\text{Cut}(A, B) = \sum_{i \in A, j \in B} w_{ij} = 3$$

$$\text{Vol}(A) = \sum_{i \in A, j \in V} w_{ij} = 13/3$$

$$\text{Vol}(B) = \sum_{i \in V, j \in B} w_{ij} = 32/3$$

$$J_{\text{NCut}}(A, B) = 0.97$$

The Eigenvalue Problem

How Spectral Clustering Got Its Name

Spectral clustering is a **compromise**: it solves an *easier* problem than Normalized Cut optimization, but with *similar* solutions.

The **Laplacian matrix** is a spectral representation of a graph.

- Simple Laplacian: $L = D - E$  Careful! There are competing definitions.
- Symmetric Laplacian: $L_S = D^{-1/2} L D^{-1/2}$
- Asymmetric Laplacian (random walk): $L_A = D^{-1} L$

In the case of two clusters, J_{NCut} is minimized when finding the eigenvector f for the **second smallest** eigenvalue of L_S , leading to the name of the method (special case of general algorithm, see later).  L is positive semi-definite and its smallest eigenvalue is 0

The clustering is recovered by sending $v_i \in A$ when $f_i > 0$, and $v_i \in B$ otherwise (or *vice-versa*).  Deterministic?

Interlude

Eigenvalues and Laplacian Matrices

An **eigenvalue** λ of a matrix T is a complex number (potentially with no imaginary part) such that $\dim \ker(T - \lambda I) > 0$.

In other words, λ is an eigenvalue of T if there exists (at least) an **eigenvector** $\vec{v} \neq 0$ such $T\vec{v} = \lambda\vec{v}$.

The Laplacian matrix L of a graph is a matrix representation of that graph.

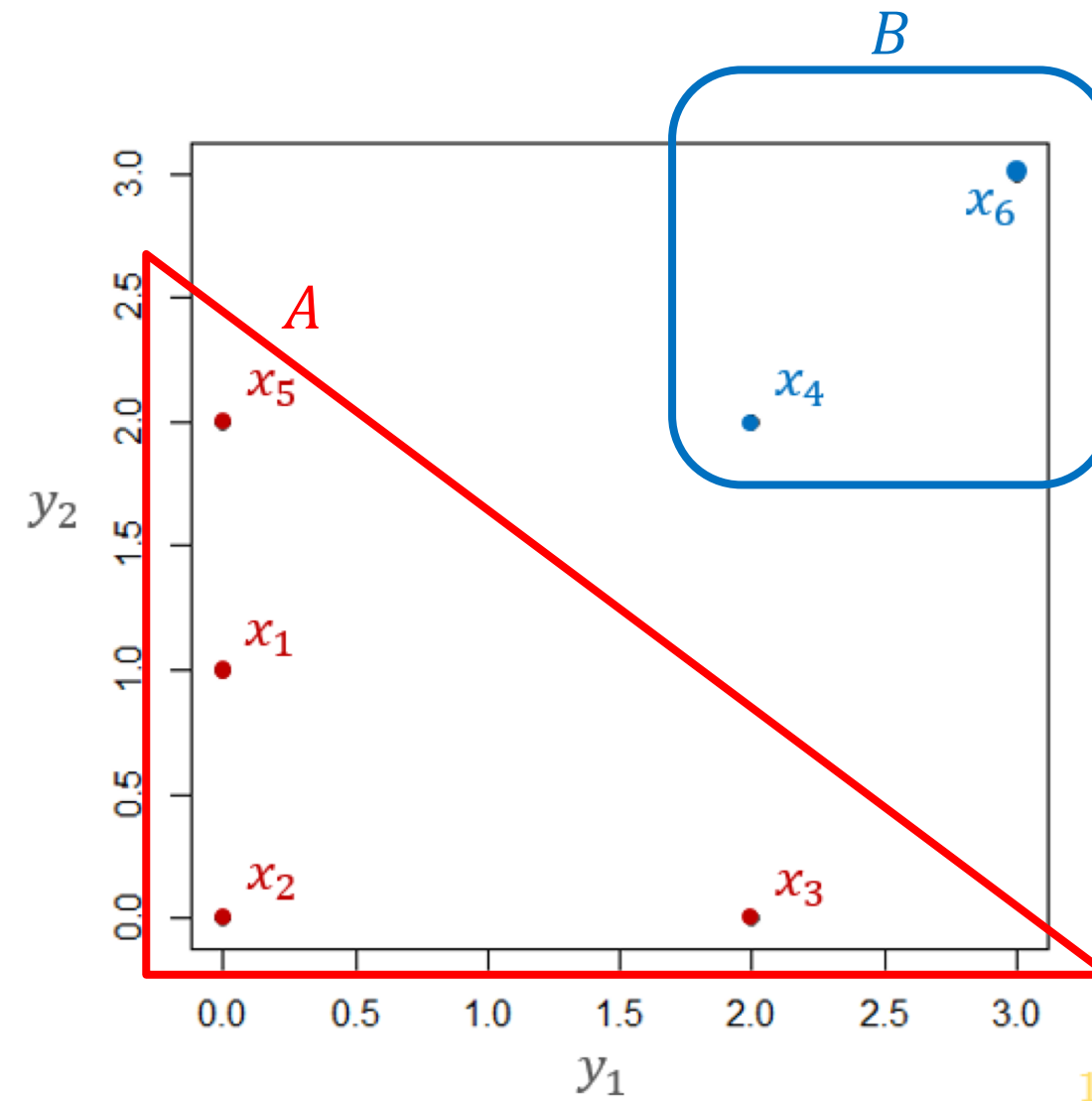
The Laplacian matrix has a bevy of nice properties that ensure that its eigenvalues behave “as they should”; for instance, the dimension of the eigenspace associated with the eigenvalue $\lambda = 0$ measures the number of connected components in the graph.

← A first guess for # of clusters?

The Eigenvalue Problem

Simple Laplacian – Example

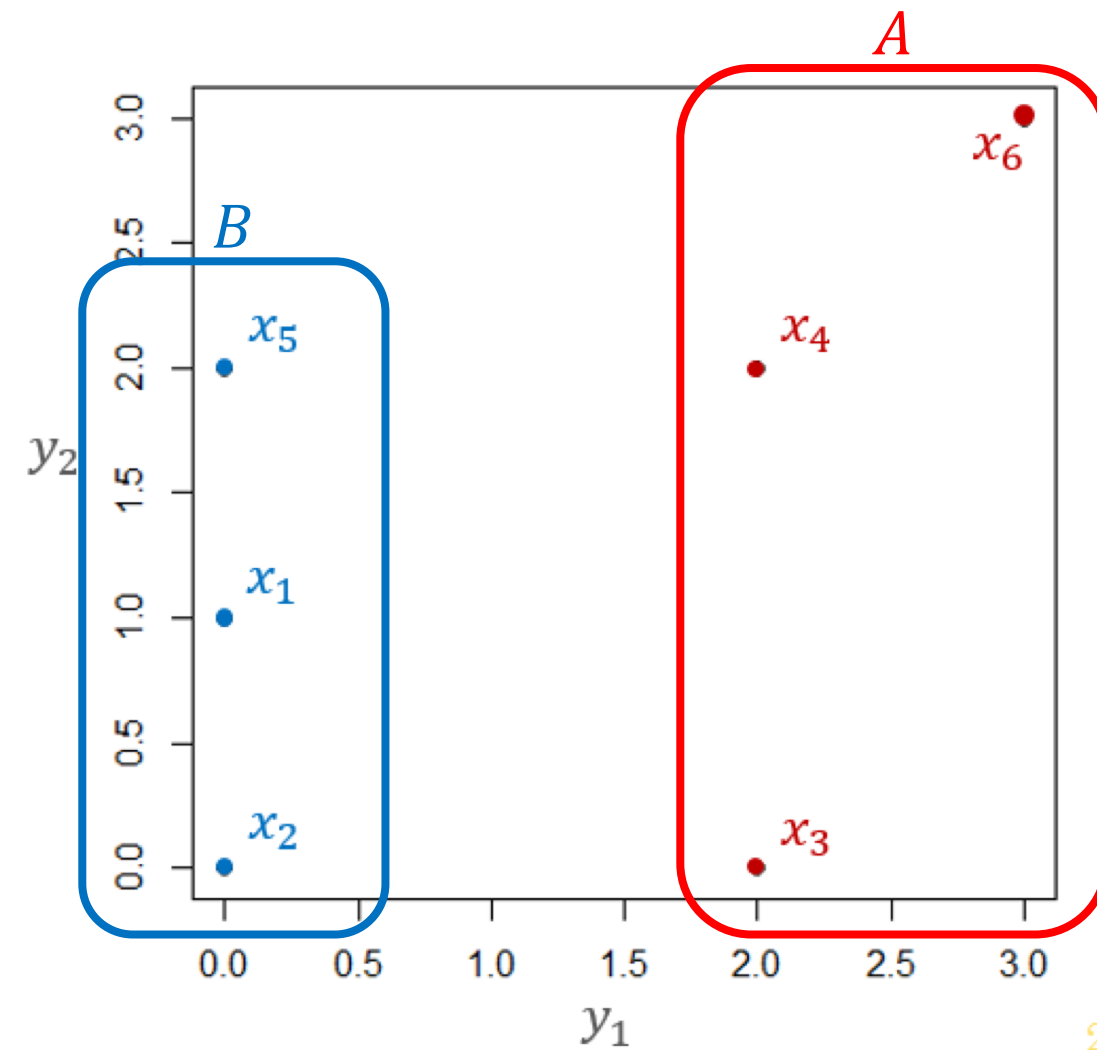
i	f_i	A/B
x_1	-0.39	A
x_2	-0.30	A
x_3	-0.18	A
x_4	0.03	B
x_5	-0.16	A
x_6	0.84	B



The Eigenvalue Problem

Symmetric Laplacian – Example

i	f_i	A/B
x_1	0.36	B
x_2	0.54	B
x_3	-0.03	A
x_4	-0.43	A
x_5	0.14	B
x_6	-0.61	A



The Eigenvalue Problem

Spectral Clustering –Algorithm

(version from von Luxburg's tutorial, with different D and L)

Algorithm to cluster $\{x_1, \dots, x_n\}$ into k clusters:

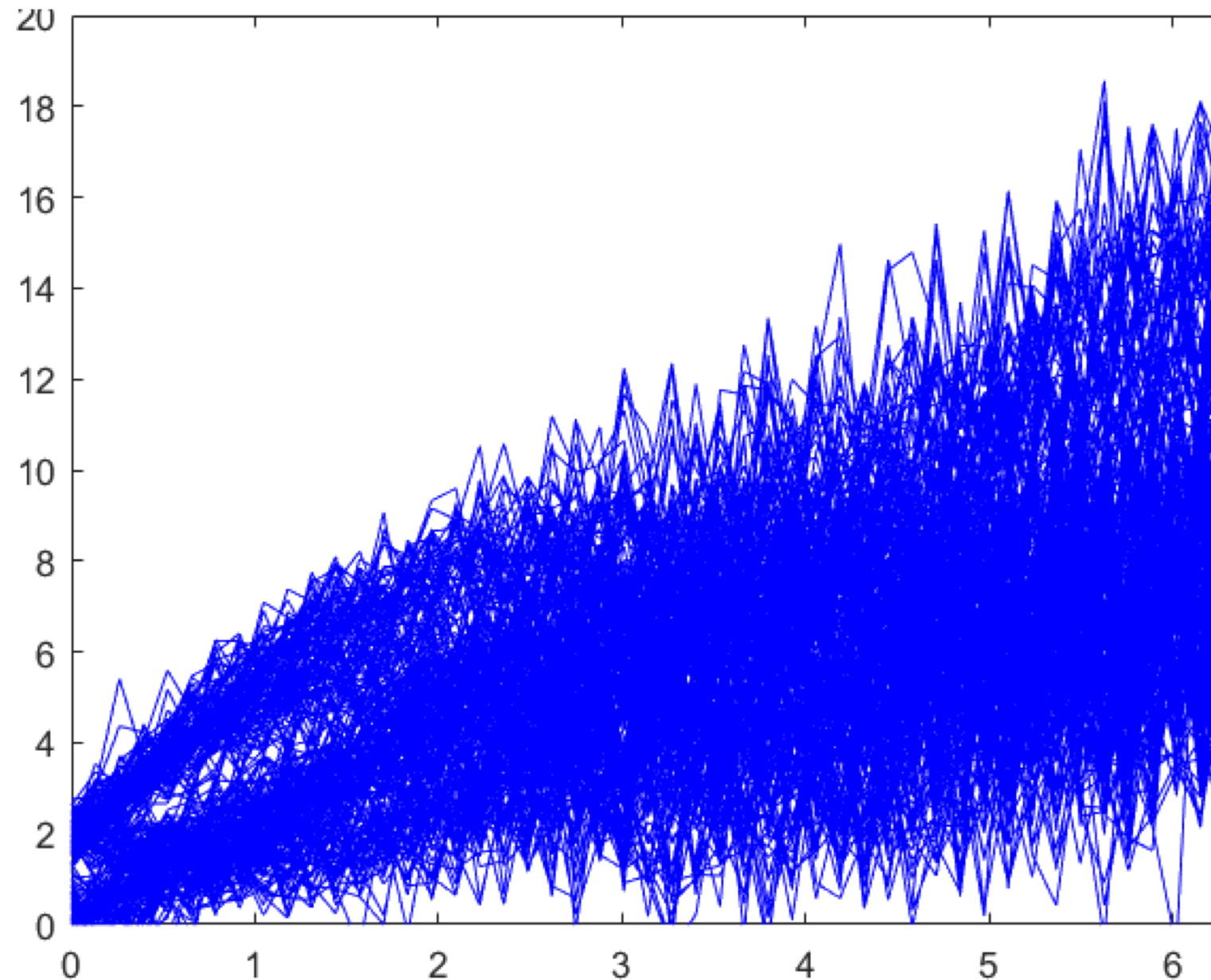
Choice of # of clusters

1. Form similarity matrix W . Choice of similarity measure
2. Define the degree matrix D . Choice of adjacency threshold
3. Construct the Laplacian matrix L . Choice of Laplacian
4. Compute the first k orthogonal eigenvectors $\{\mu_1, \dots, \mu_k\}$ of the Laplacian L corresponding to its k **smallest** eigenvalues.
5. Construct U , using $\mu_1, \dots, \mu_k$ as **columns**.
6. Normalize the **rows** of U so that they each have unit length; call the new matrix Y .
7. Cluster the rows of Y into k clusters. Choice of clustering method
8. Assign the original point x_i to cluster j if the i^{th} row of Y was assigned to cluster j .

Other algorithms: un-normalized spectral clustering, Shi and Malik's algorithm (see von Luxburg's tutorial).

Examples and Case Studies

Latent Classes – Time Series

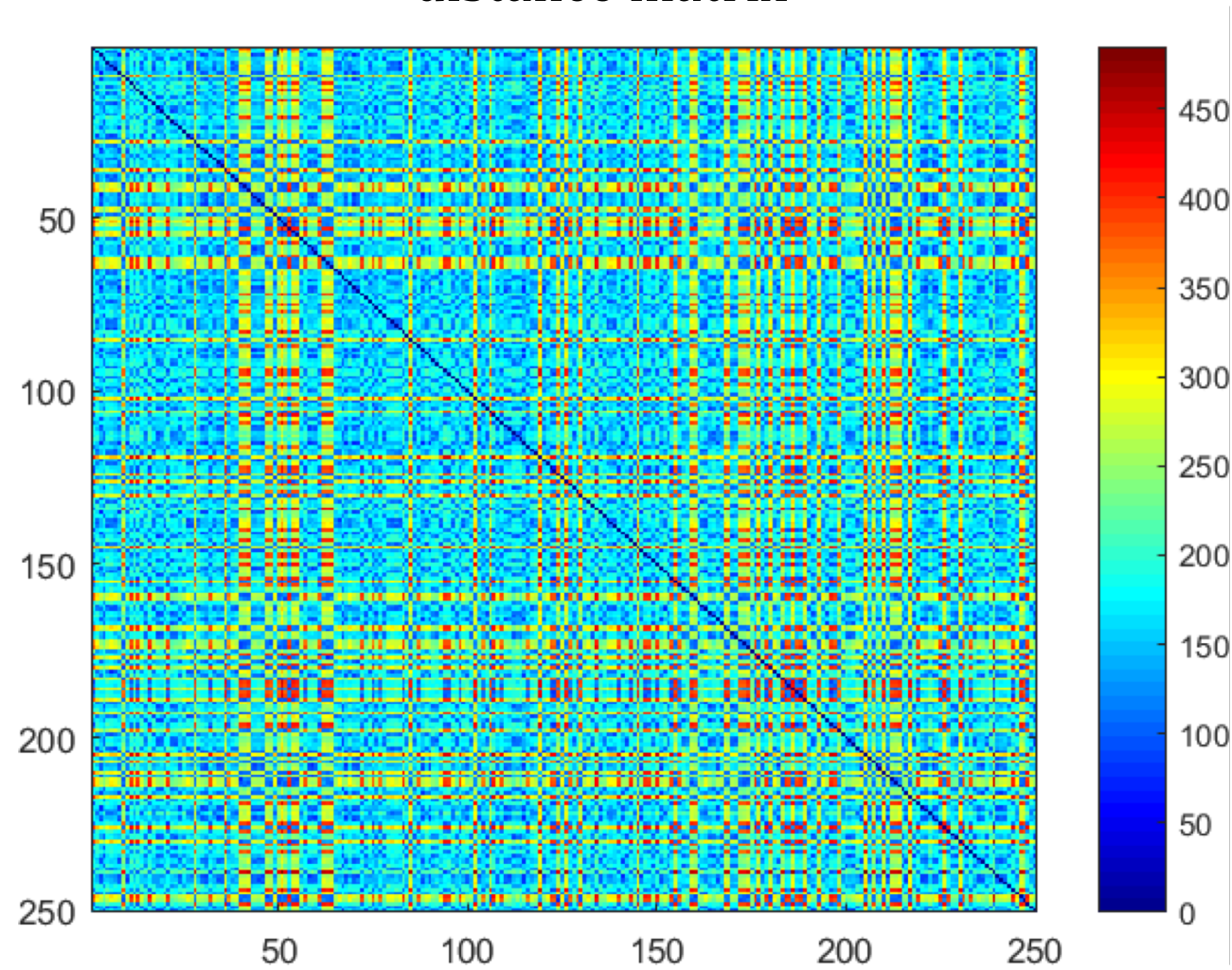


- 250 times series
- average absolute gap between series used as distance d
- Gaussian similarity measure
$$w = \exp\left(-\frac{d^2}{2\sigma^2}\right)$$
- $\sigma = 300$
- adjacency threshold $\tau = 0.9$
- $k = 5$ clusters

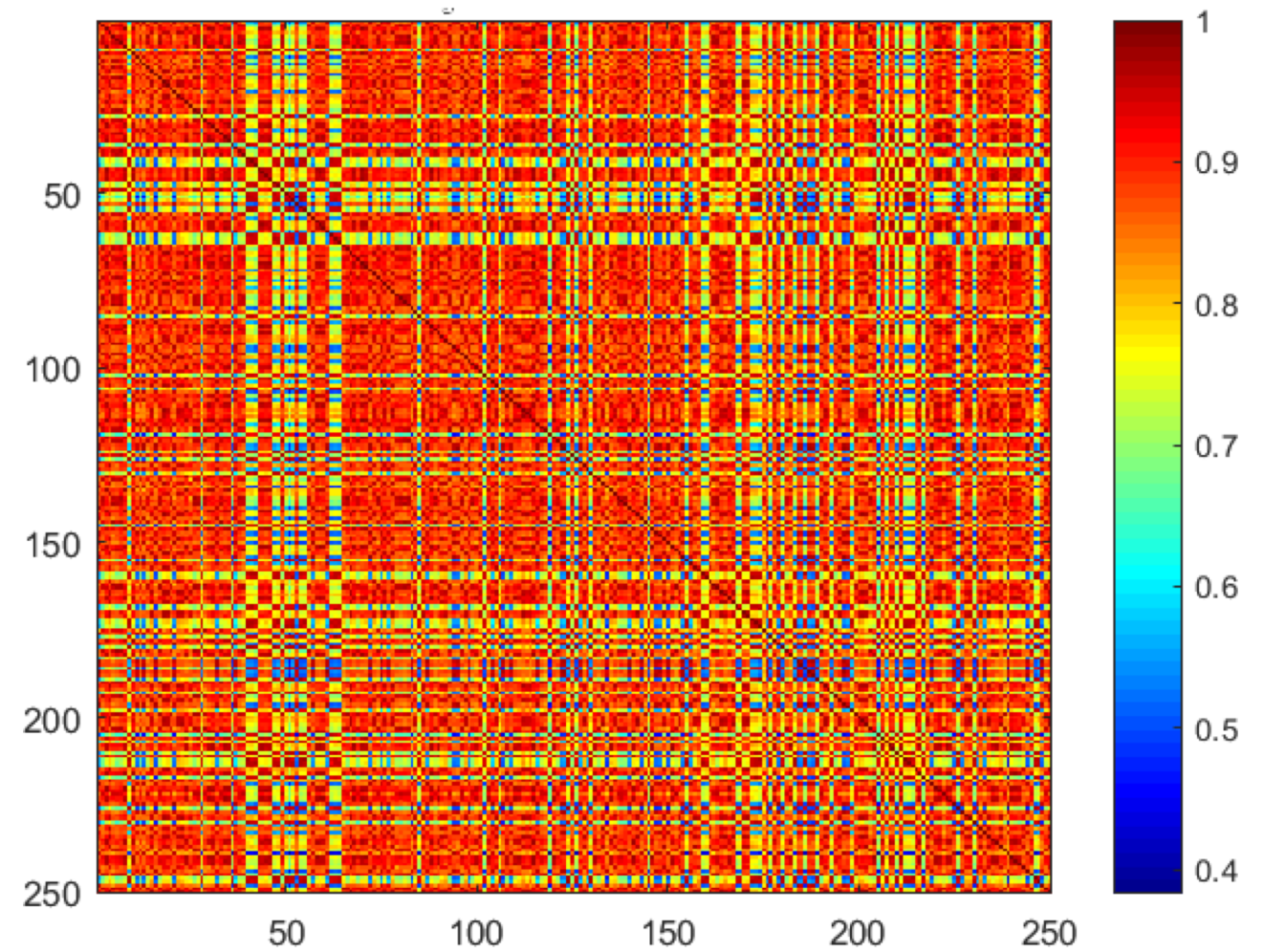
Examples and Case Studies

Latent Classes – Time Series

distance matrix



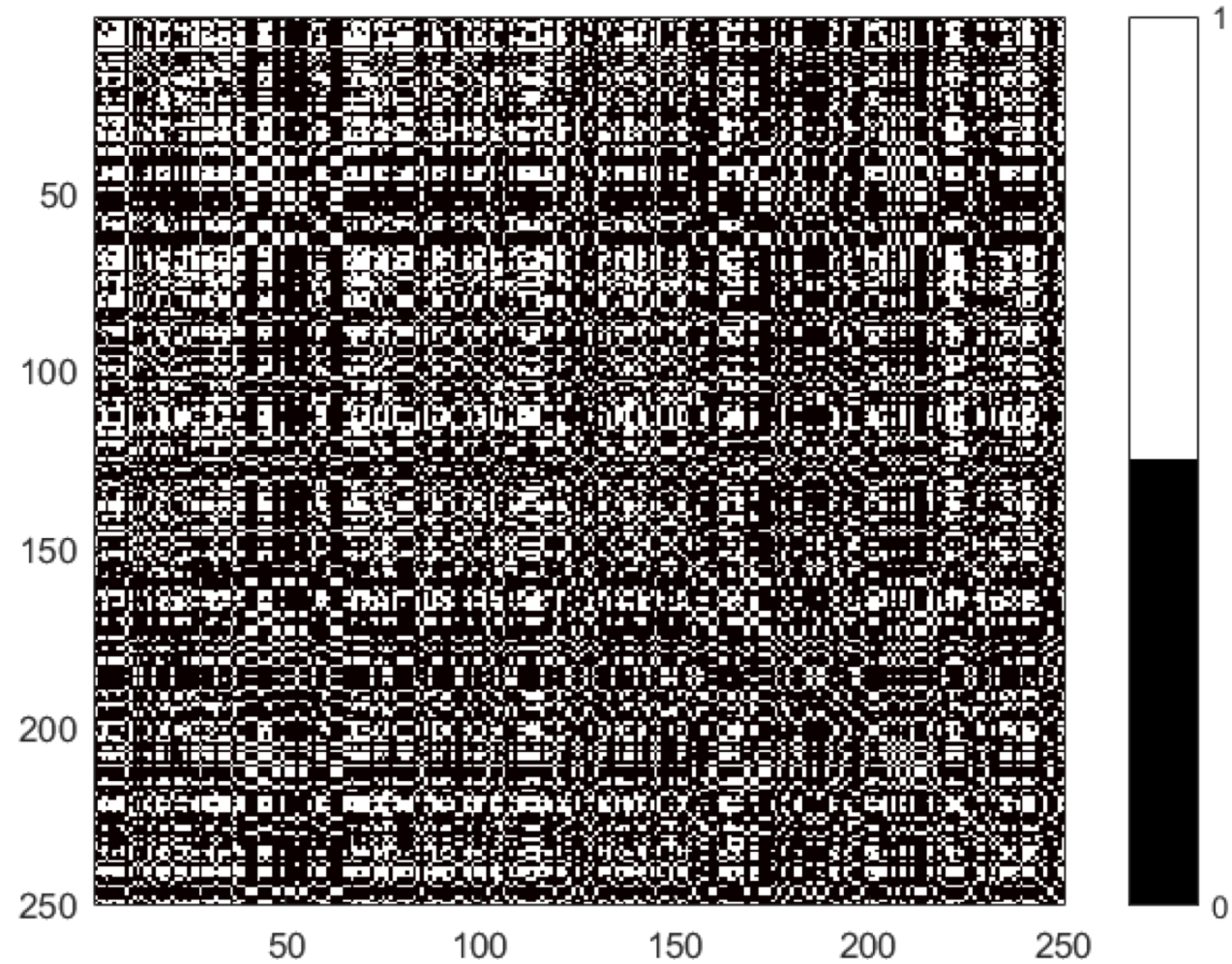
similarity matrix W



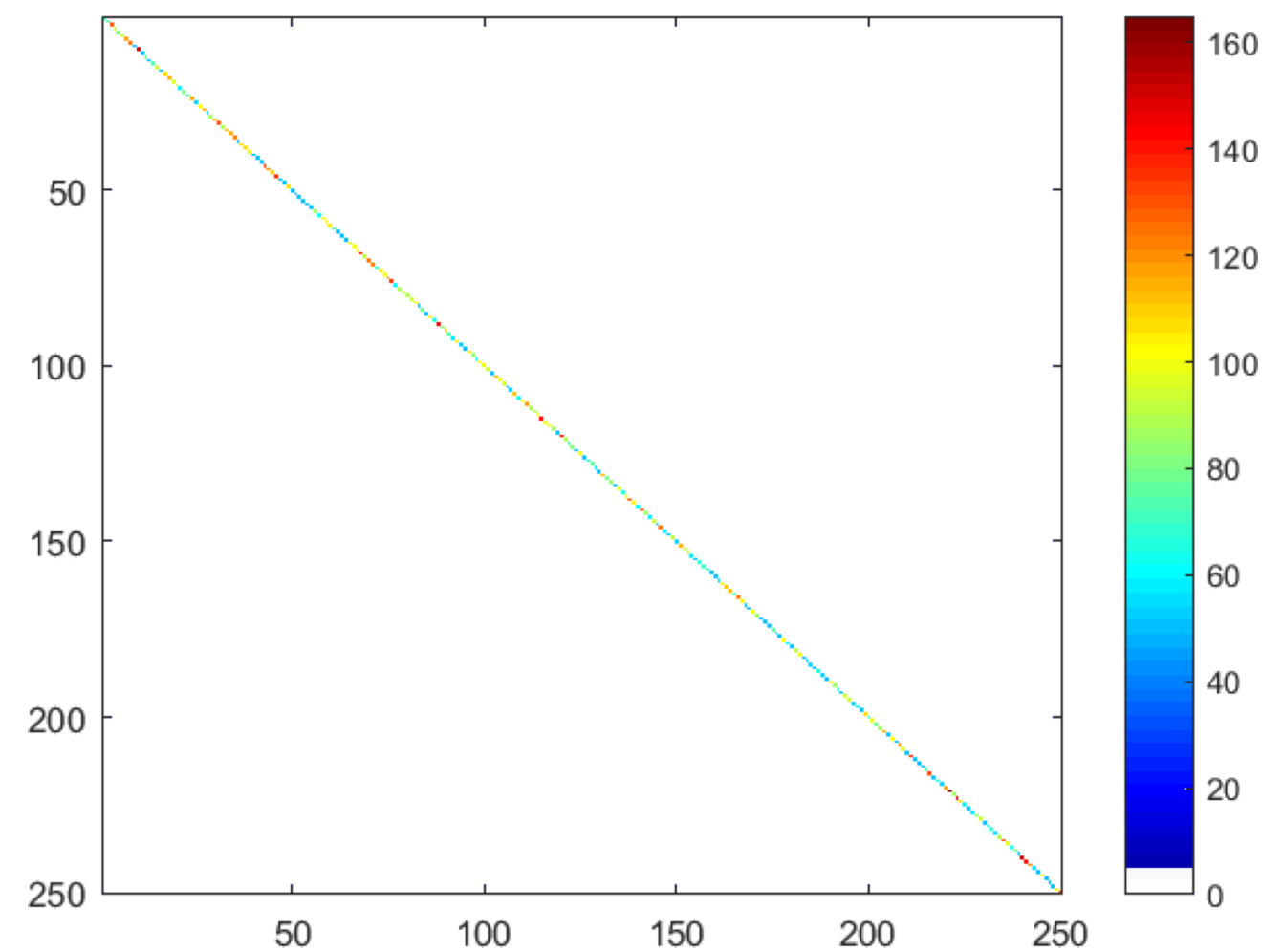
Examples and Case Studies

Latent Classes – Time Series

adjacency matrix E

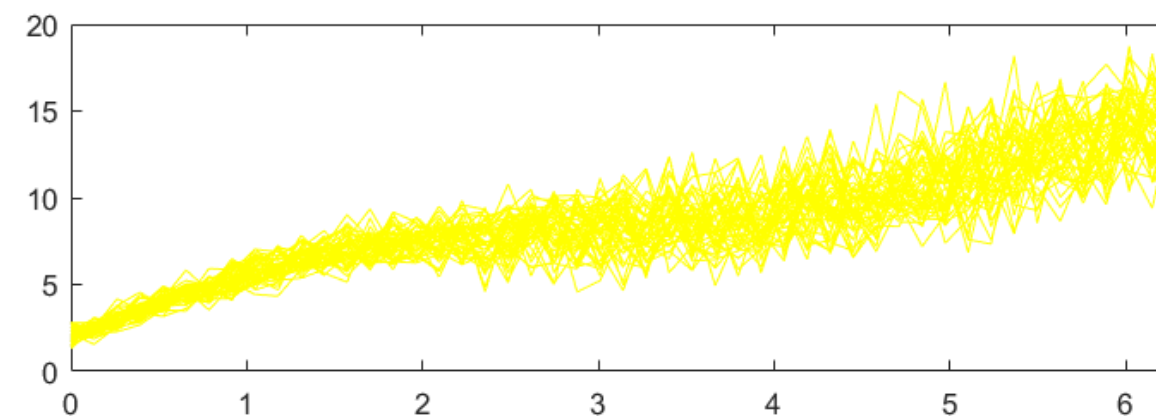
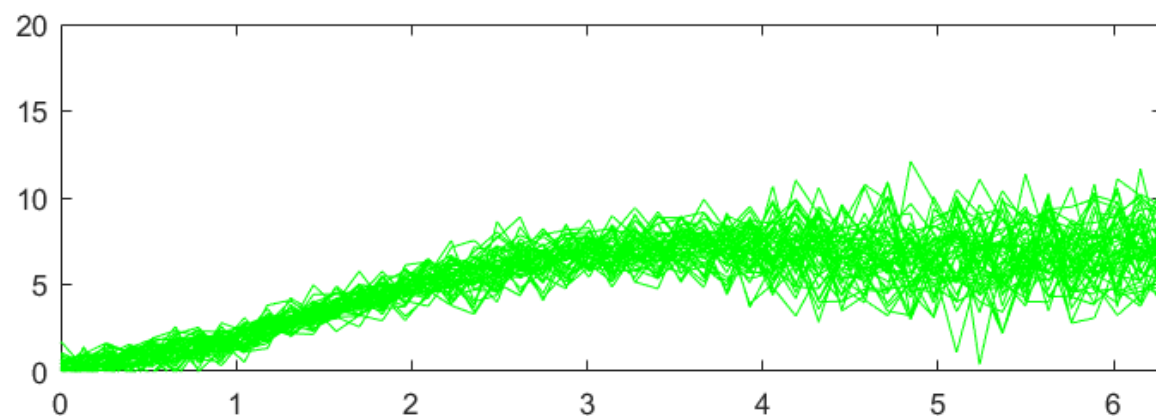
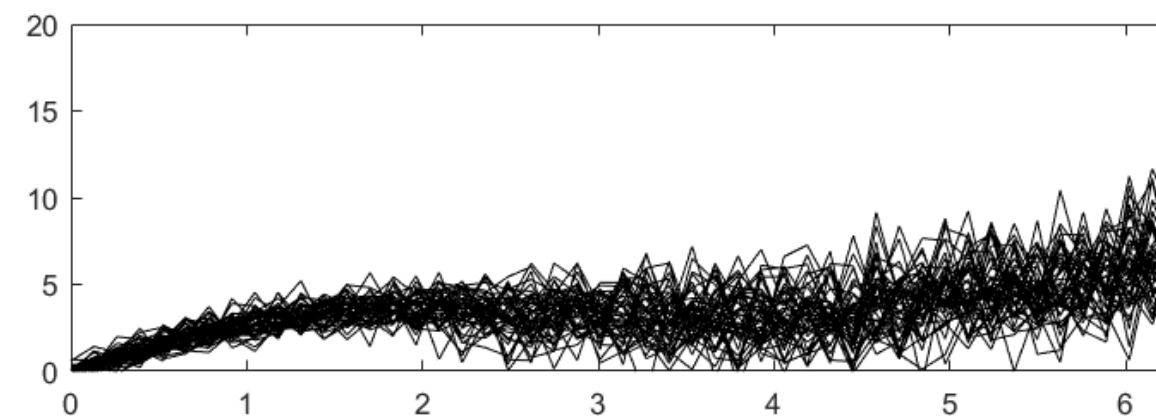
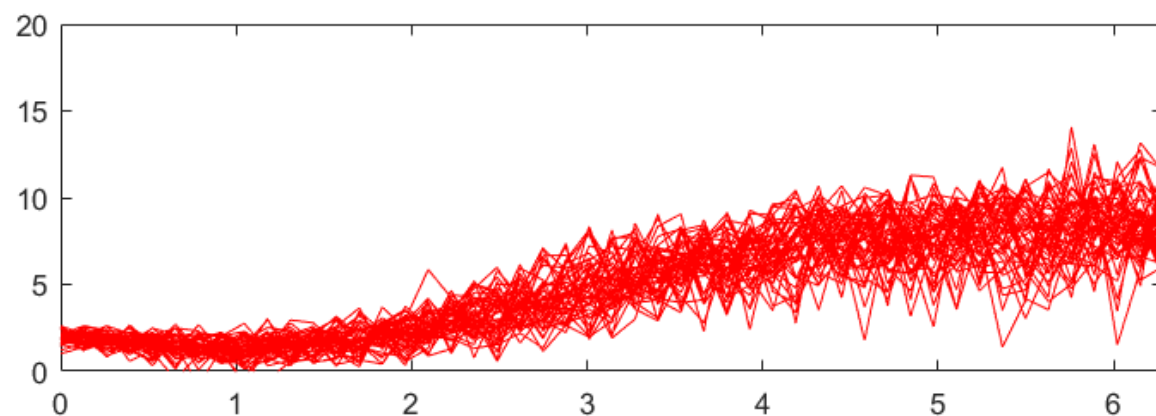
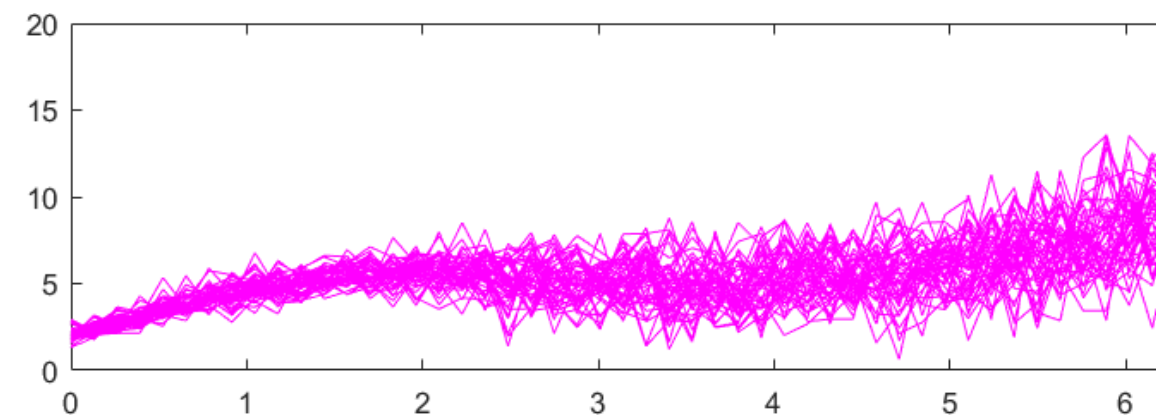
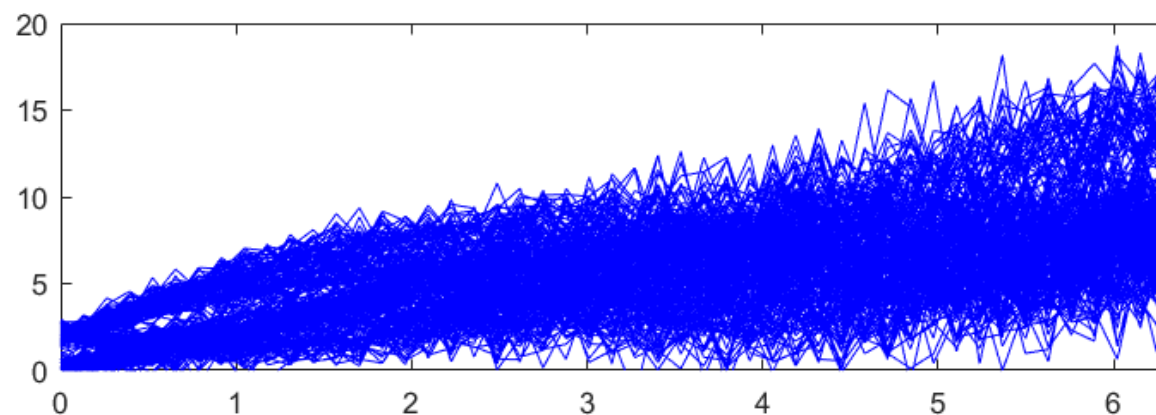


degree matrix D



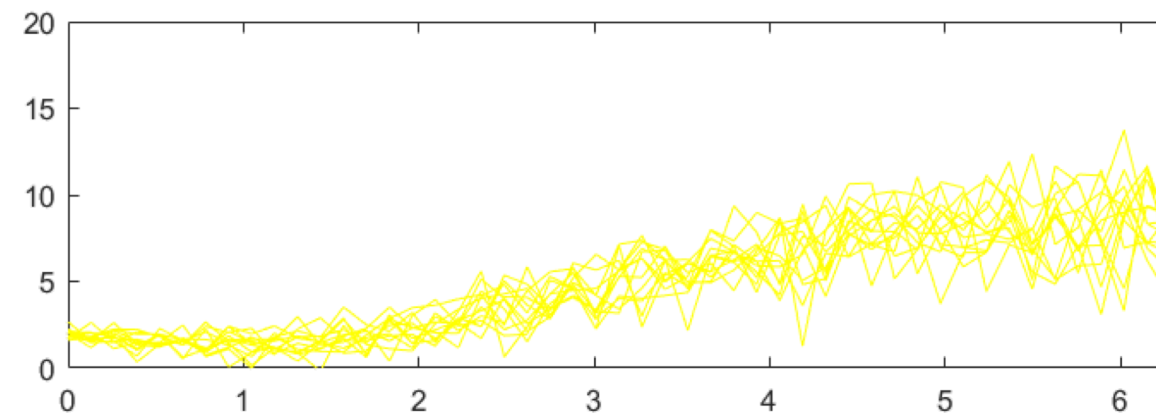
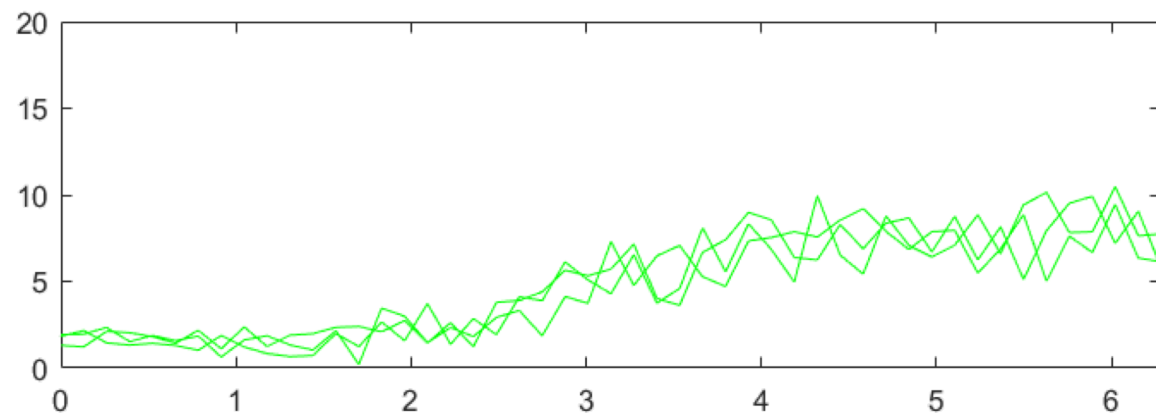
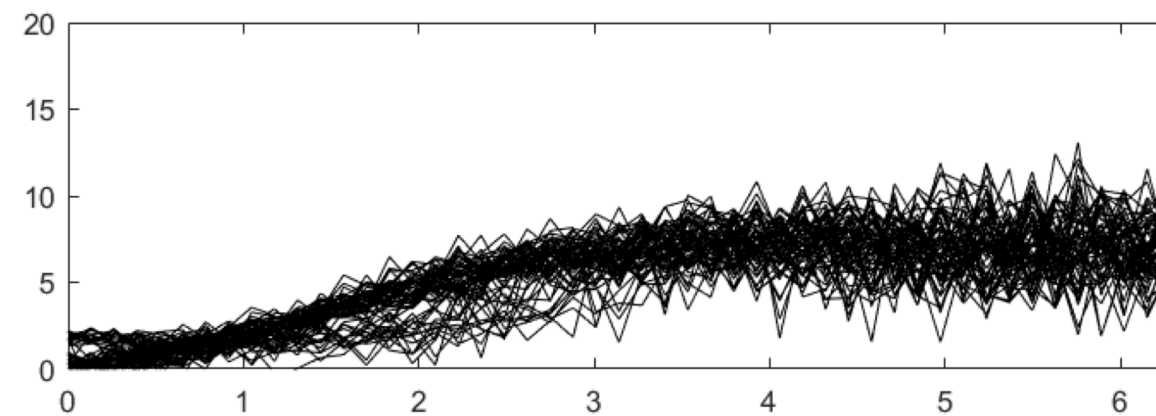
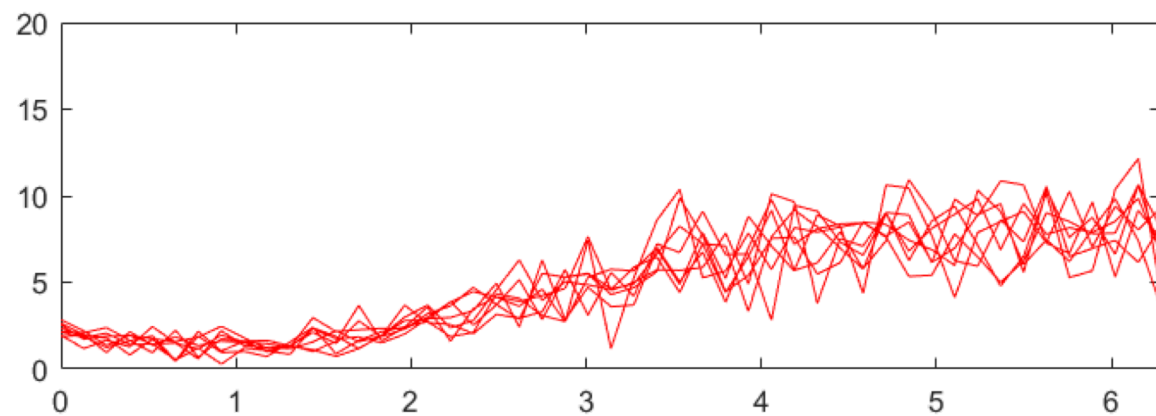
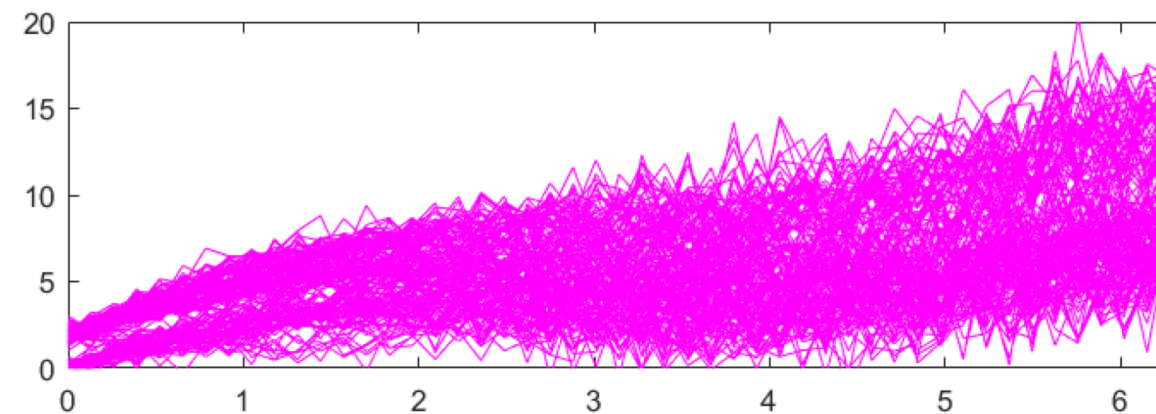
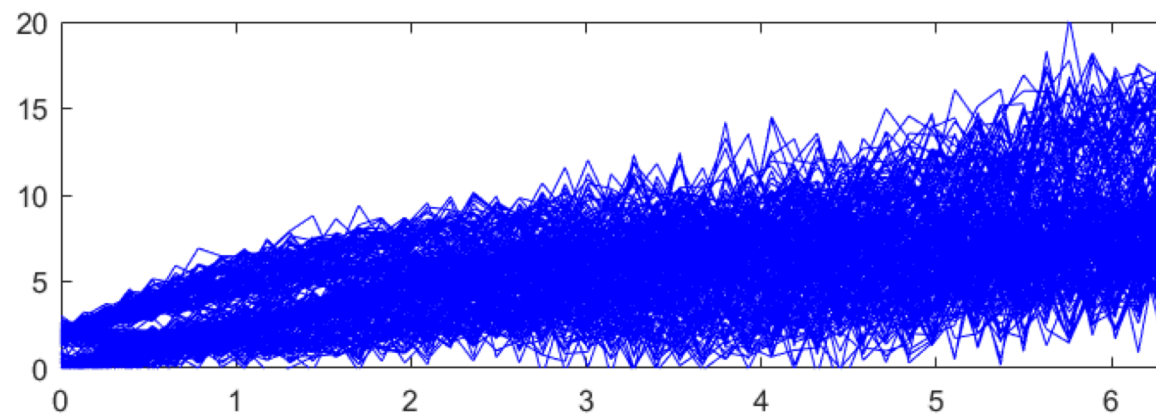
Examples and Case Studies

Latent Classes – Time Series



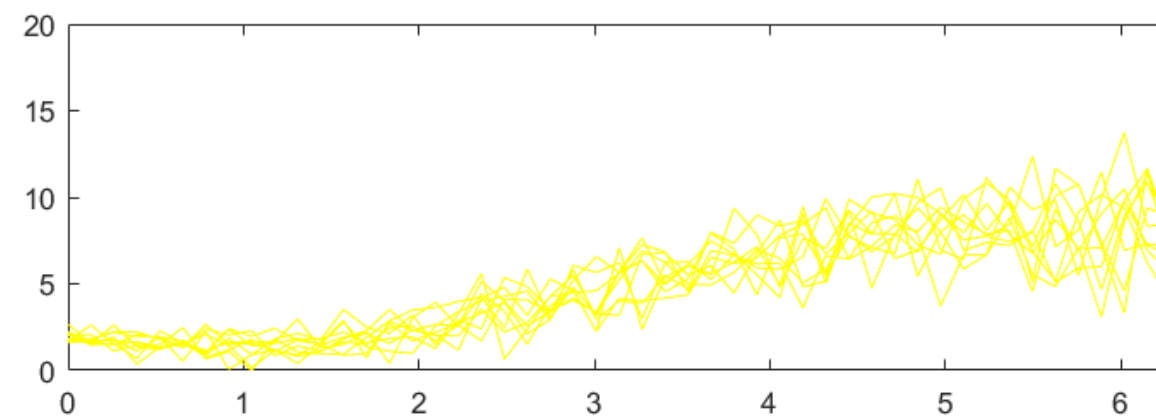
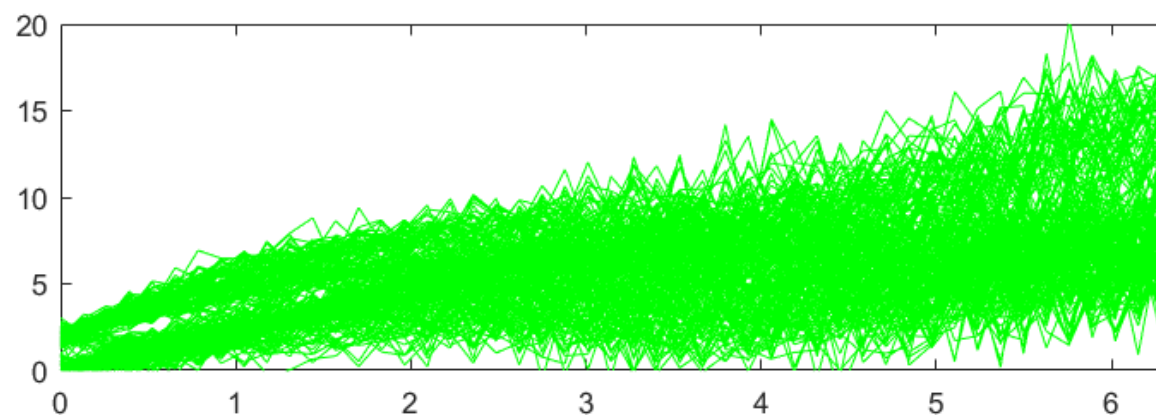
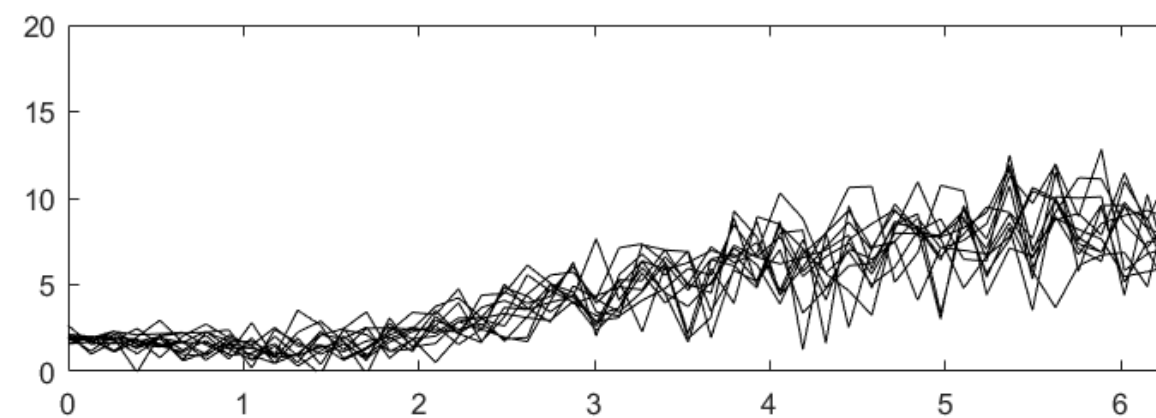
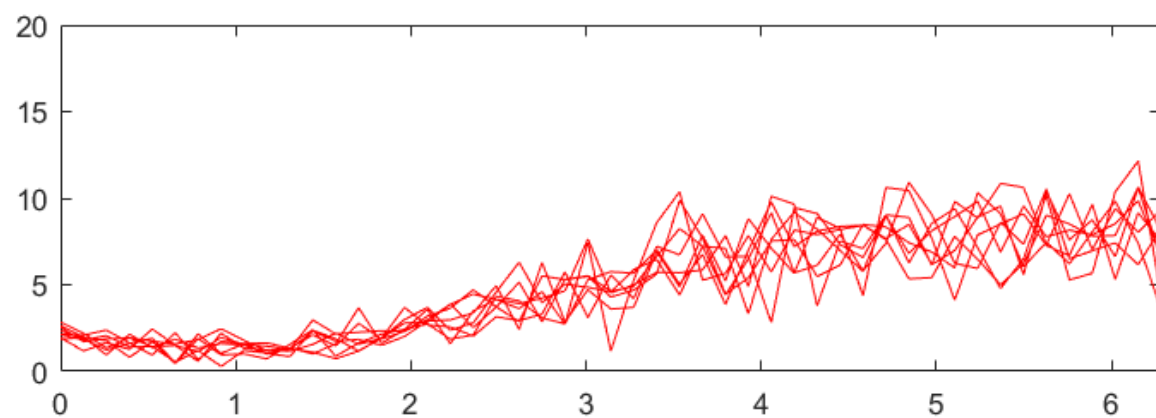
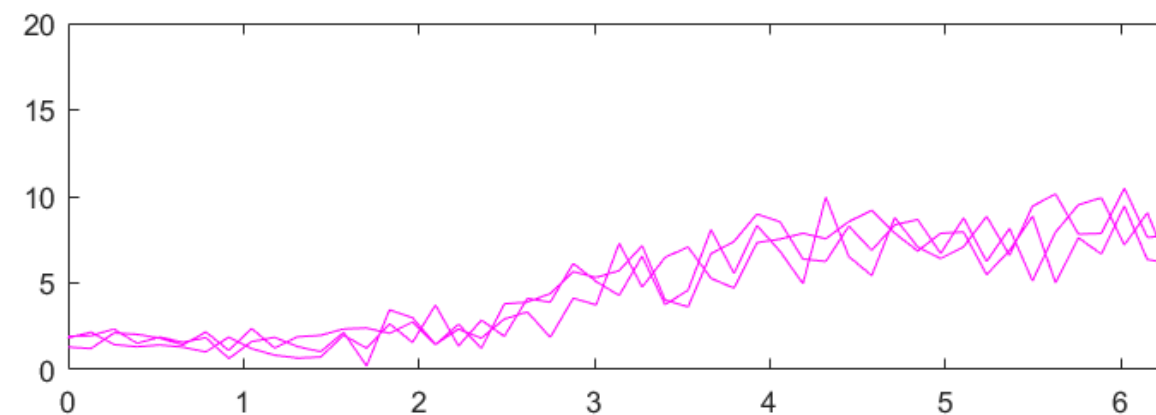
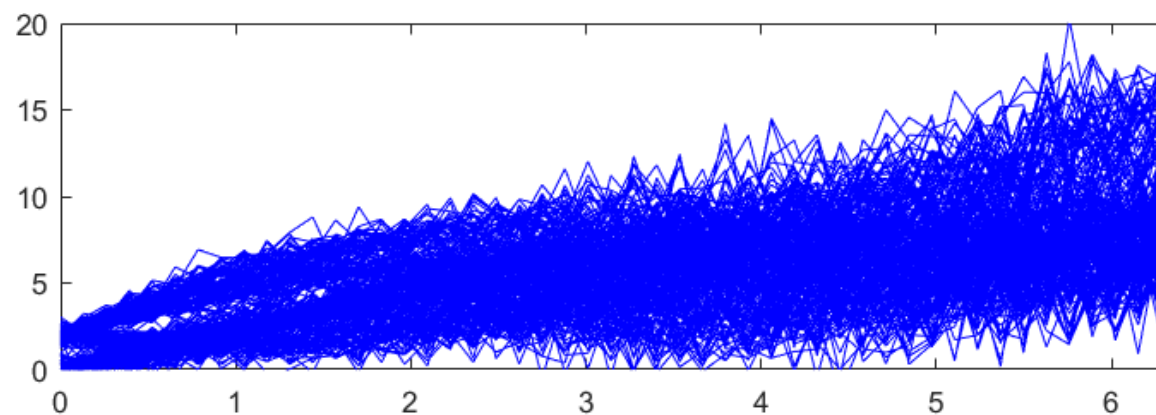
Examples and Case Studies

Latent Classes – Time Series



Examples and Case Studies

Latent Classes – Time Series



Examples and Case Studies

Signal Processing – Spectral Clustering for Speech Separation

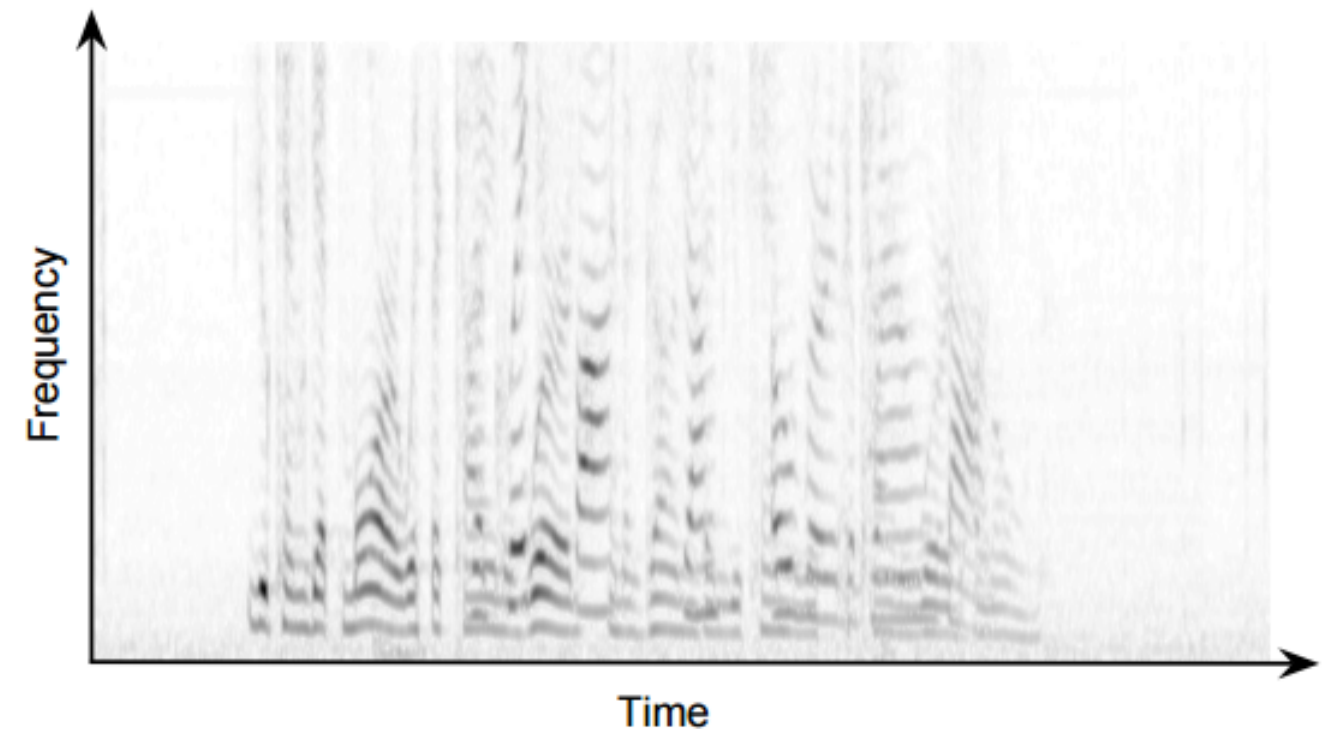
Project: Francis R. Bach and Michael I. Jordan combined prior relevant knowledge with learning similarity algorithm, to explain spectral clustering.

Goal: apply the algorithm to separate two speakers from a one-microphone blind source.

Data: Two speakers give speech and their voice signal is collected by a one-microphone blind source.

Spectrogram of speech (two simultaneous English speakers).

The gray intensity is proportional to the amplitude of the spectrogram.



Examples and Case Studies

Signal Processing – Spectral Clustering for Speech Separation

Method:

- Assume partitions are known in the given sample data.
- Perform spectral clustering on the similarity matrices
- Obtain the same partitions as assumed previously

Algorithm: Similar to NJW.

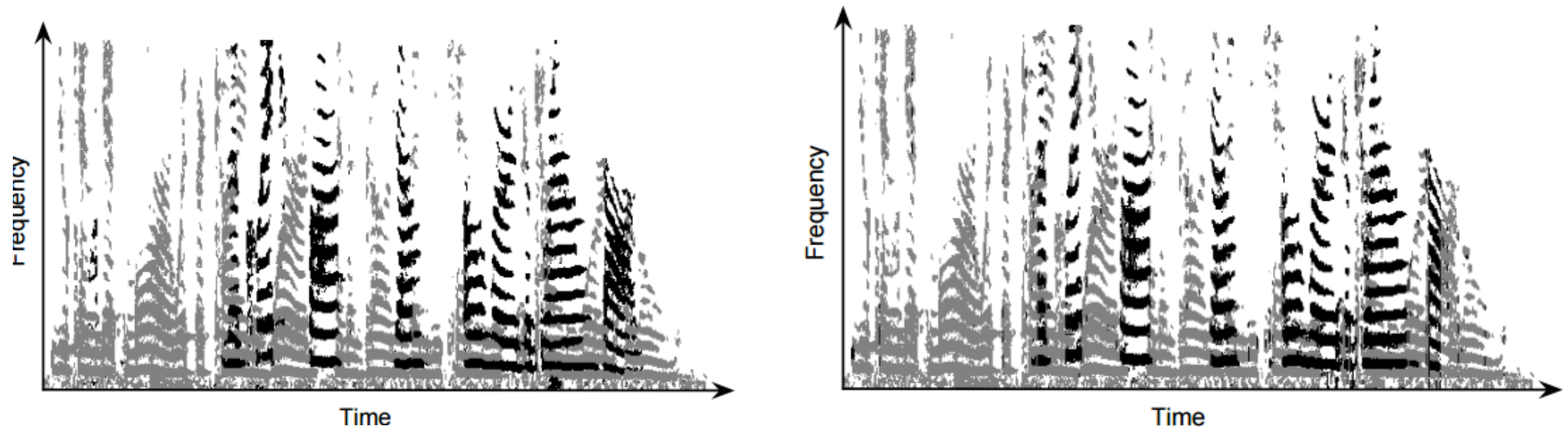
Challenges:

- Limited to the setting of ideal acoustics and equal-strength mixing of two speakers
- Training examples can be created by mixing previously captured signals
- Spectral clustering needs to be robust to irrelevant features
- Computation challenge of spectral clustering applied to speech separation

Examples and Case Studies

Signal Processing – Spectral Clustering for Speech Separation

The result is an optimized segmenter for spectrograms of speech mixtures.



Selected result: (Left) Optimal segmentation for the spectrogram of English speakers, where the two speakers are “black” and “grey”; this segmentation is obtained from the known separated signals. (Right) The blind segmentation obtained with our algorithm.

Examples and Case Studies

Sensor Detection – A Spectral Clustering Approach to Validating Sensors via Their Peers in Distributed Sensor Networks

Project: H. T. Kung and Dario Vlah describe a spectral clustering approach to identify bad sensors, by using a simple model problem.

Motivation: current status and environment affect sensors performance and impractical to bring calibrate device to test each sensor

Goal: using peer sensors to detect badly performing sensors

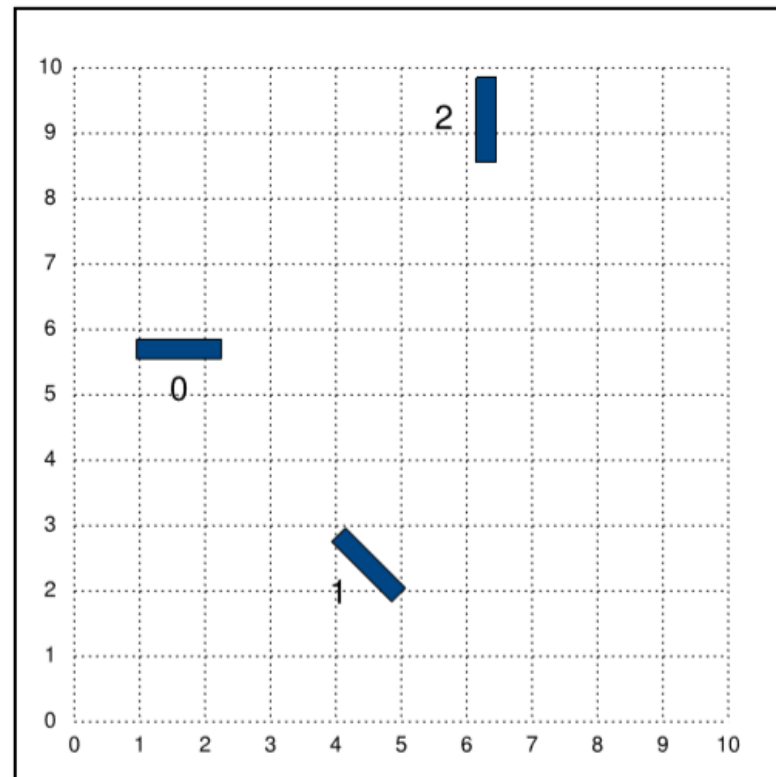
Method: simulation and spectral clustering

Examples and Case Studies

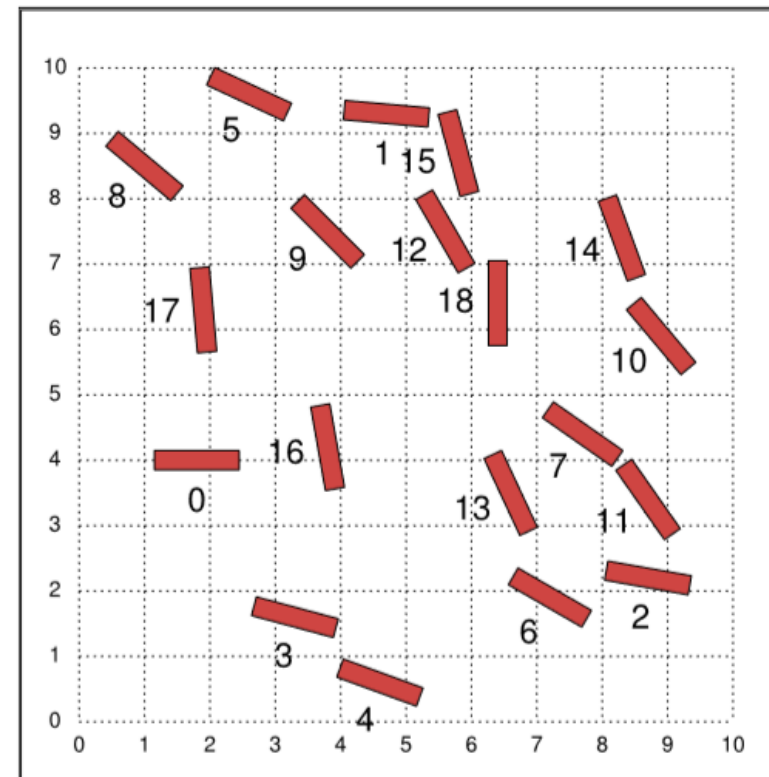
Sensor Detection – A Spectral Clustering Approach to Validating Sensors via Their Peers in Distributed Sensor Networks

Model design: sensors are indexed by their antenna orientations

- assume that the matching of a sensor and a target is based on the degree to which their antenna orientations match
- use of non-principal eigenvector with the principal one, to detect clustering structures



(a) 3 targets with 3 antenna orientations



(b) 19 sensors with 19 antenna orientations

Sensors and targets in
the same region

Examples and Case Studies

Sensor Detection – A Spectral Clustering Approach to Validating Sensors via Their Peers in Distributed Sensor Networks

Simulation of large systems on the same model design

- Data: 100 sensors and 10 targets
- Assumption 1: sensors and targets are evenly partitioned into three groups, with antenna orientations of 0, 45 and 90 degrees
- Assumption 2: some randomly selected sensors are bad sensors in the sense that their measurements can be off by any amount from -100% to +100%

Results:

- When the number k of leading eigenvectors used increases, the accuracy performance improves
- The number of false positives decreases with the number of bad sensors input to the simulator.
- Spectral clustering achieves almost perfect performance in specific circumstances.

Clustering Validation

Is a Clustering Scheme Any Good?

There is **NO** optimal validation approach.

Possibilities include:

- comparing with the optimal clustering (external)
- comparing with other clustering methods (external)
- visualizing the clusters (external)
- Davies-Bouldin, Within-SS (internal) ← 
- repeated clusterings (internal) ← 

Scenario 1: given data D , true clustering C , algorithm A produces C' :

- is C' “close” to C ?

Scenario 2: given data D , true clustering C , algorithm A produces C' ; algorithm A^* produces C^* , and so forth.

- are C' , C^* , ..., “close” to C ? Which one is “closer”?

Clustering Validation

Is a Clustering Scheme Any Good?

A distance measurement $d(C, C')$ between clusterings is needed...

Let $C = \{C_1, \dots, C_k\}$ be a **clustering** of a set of n data points $\{x_1, \dots, x_n\}$.

The **quadratic cost** is the function defined by

$$\Lambda(C) = -\text{Trace}(Z^T(C) \cdot W \cdot Z(C)),$$

where Z is the matrix representation of C :

$$Z_{ik} = \begin{cases} 1 & \text{if } x_i \in C_k \\ 0 & \text{if } x_i \notin C_k \end{cases}$$

In some sense, the clustering scheme for which $\Lambda(C)$ is minimized is **optimal** against quadratic cost.  For a given choice of similarity measure