



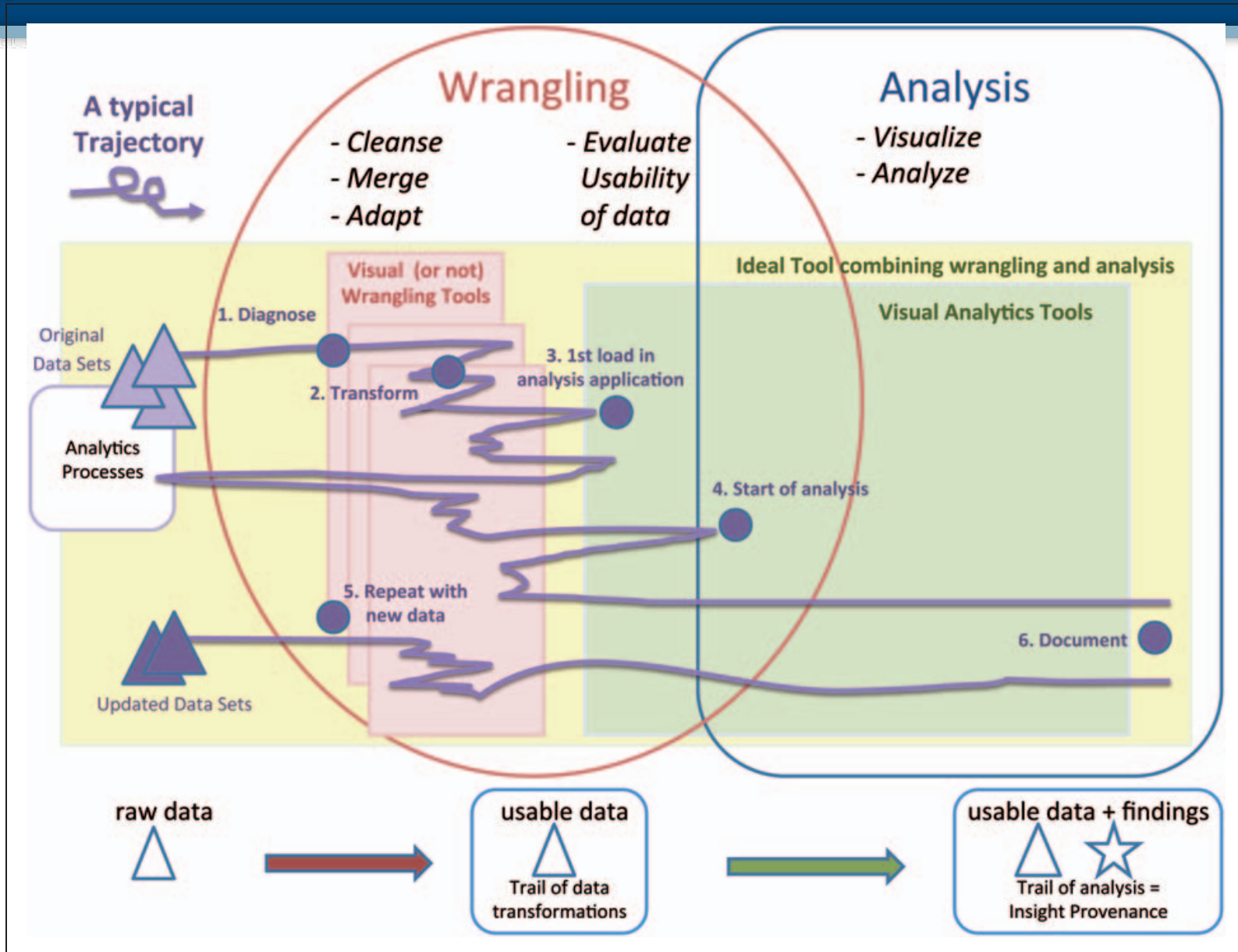
# **COMP20008 Elements of Data Processing**

## **Hierarchical clustering and dimension reduction**

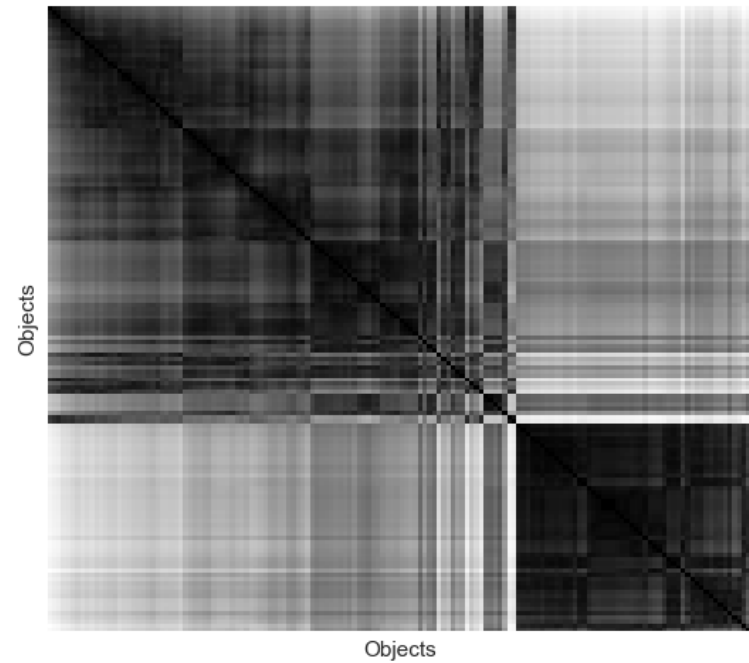
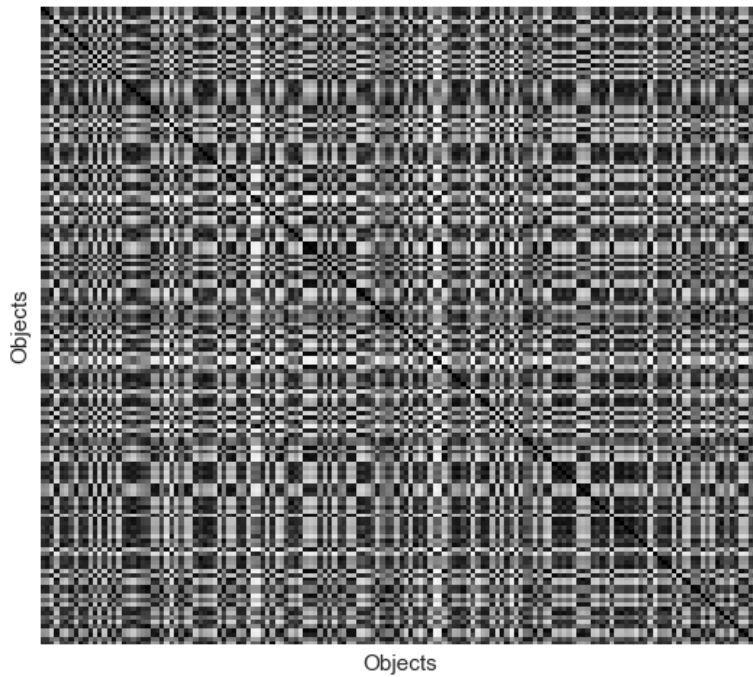


- A brief feedback
- VAT algorithm
- Hierarchical clustering
  - Another alternative for k-means to extract clusters, visualise their relationships
- Dimension reduction
  - A technique for visualising high dimensional data (many features/columns)

# Overview



- How to reorder the dissimilarity matrix?



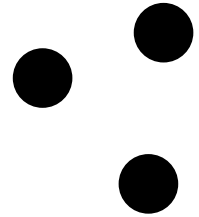


MELBOURNE



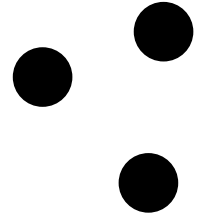
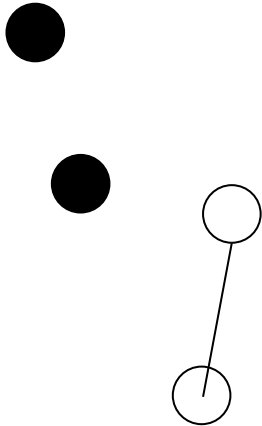


MELBOURNE





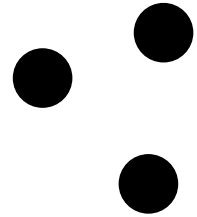
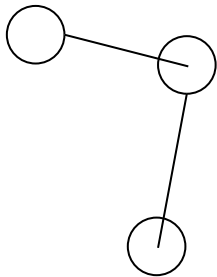
MELBOURNE





## VAT ordering example

MELBOURNE

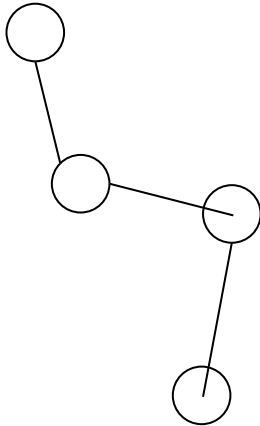
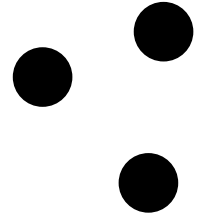






# VAT ordering example

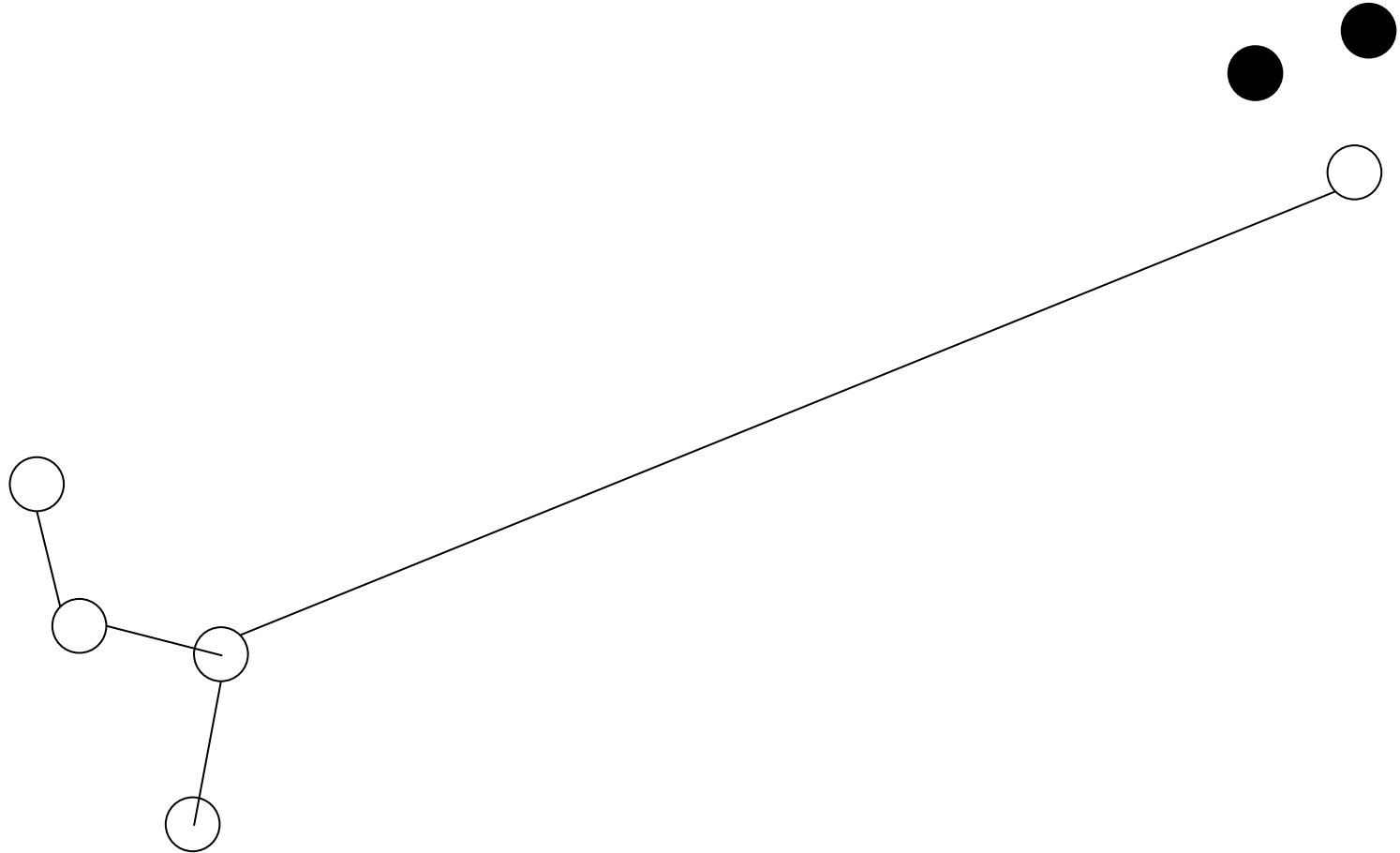
MELBOURNE





## VAT ordering example

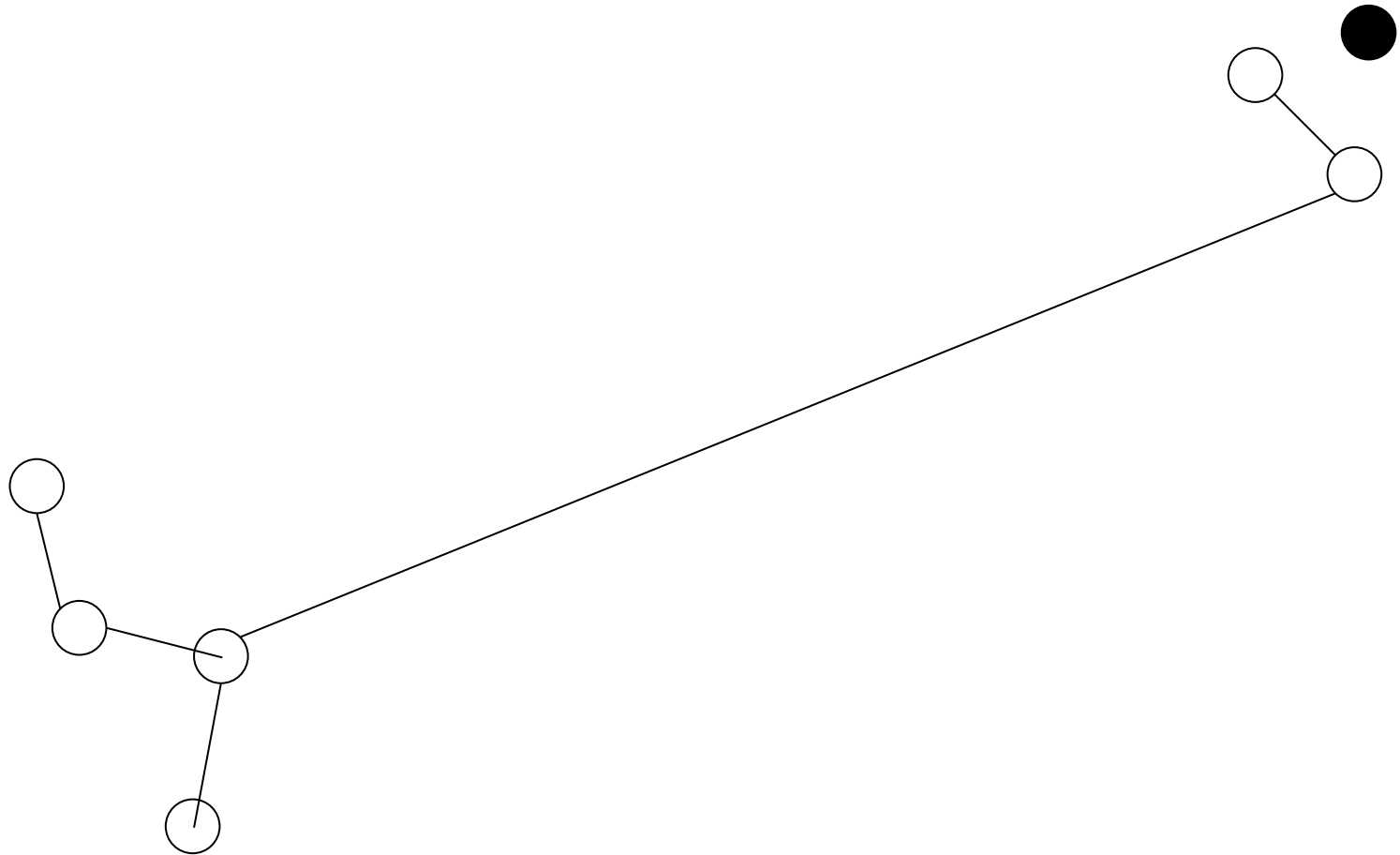
MELBOURNE





## VAT ordering example

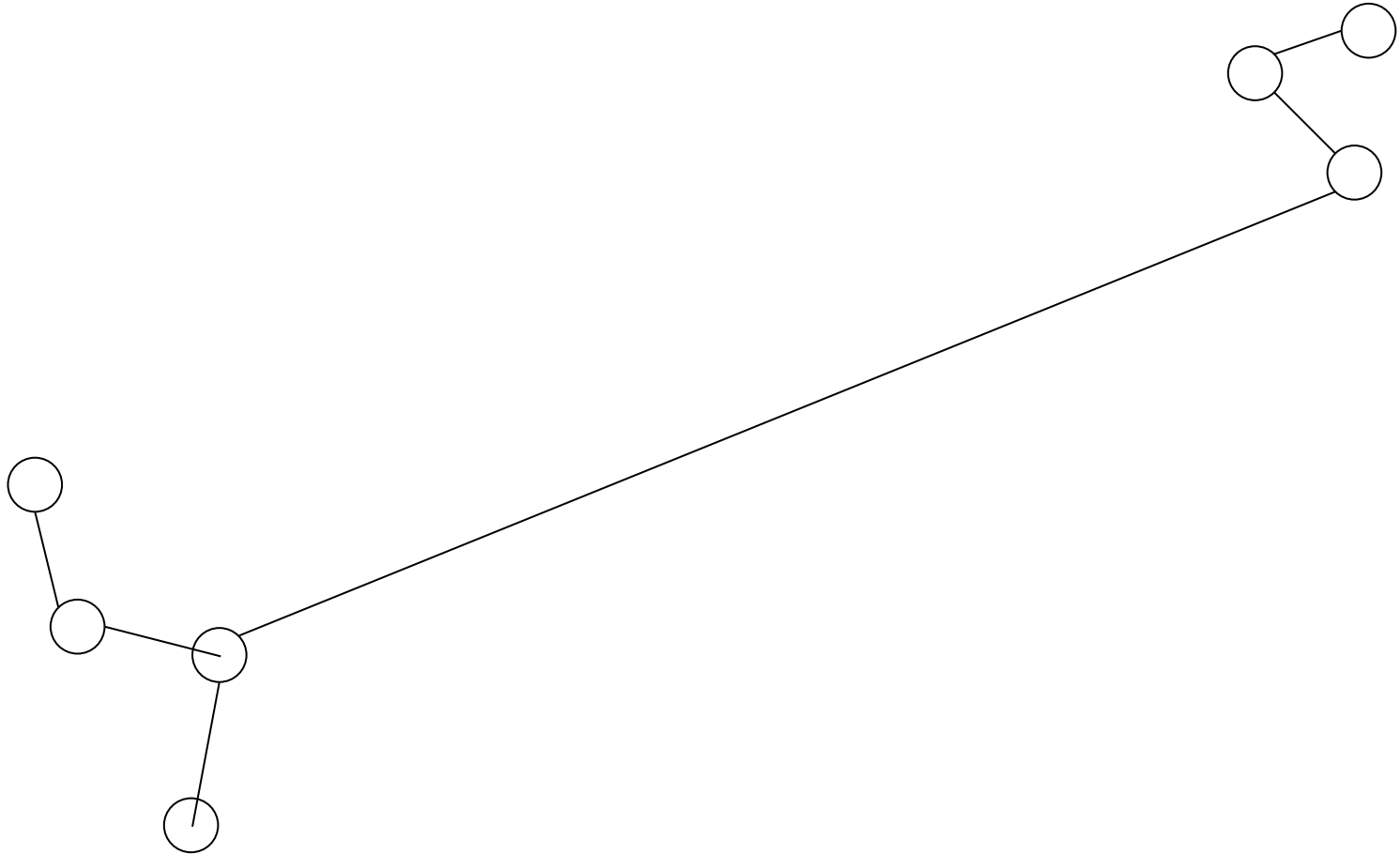
MELBOURNE





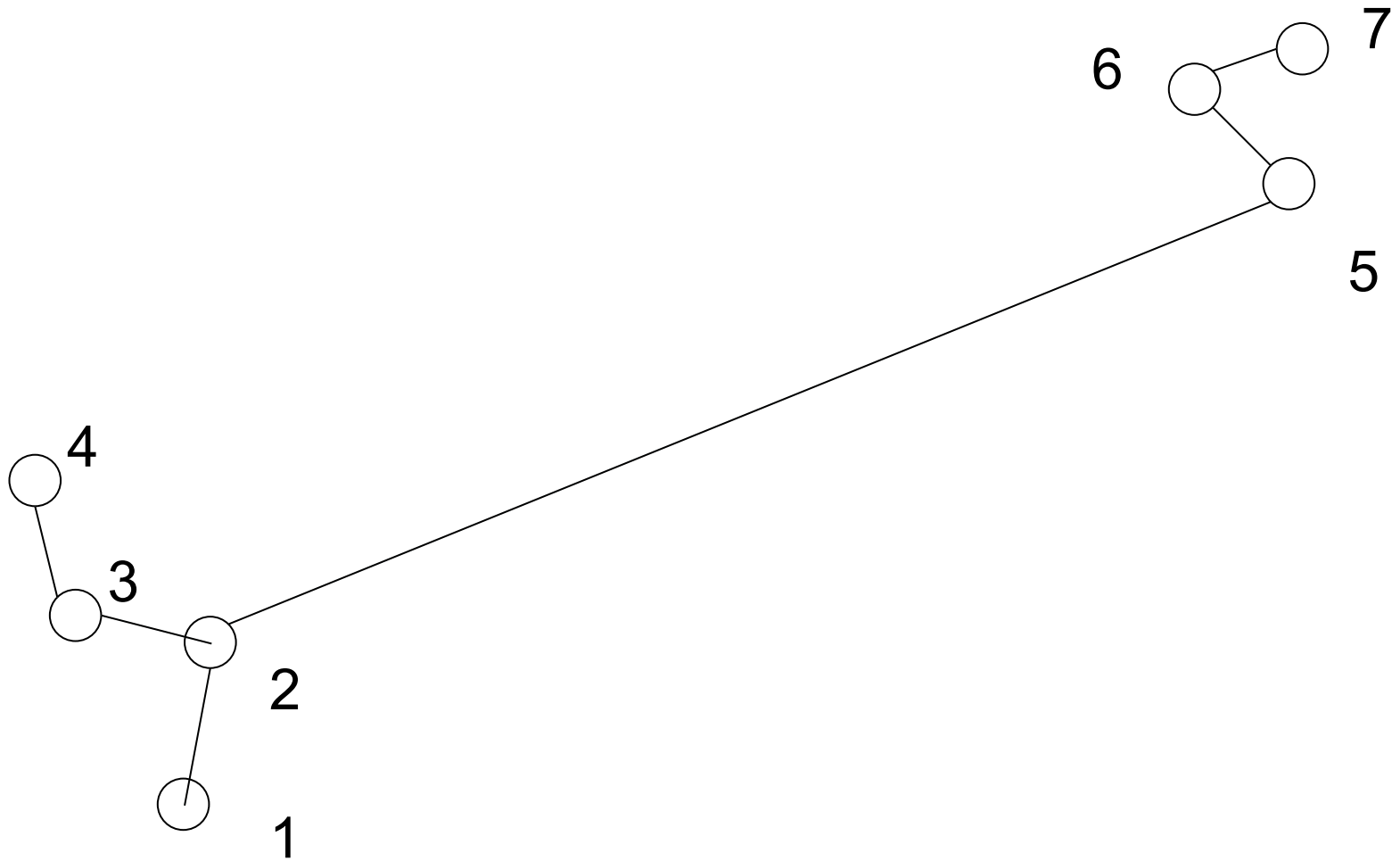
## VAT ordering example

MELBOURNE





## VAT ordering example





Given an  $N \times N$  dissimilarity matrix **D**

Let  $K = \{1, \dots, N\}$ ,  $I = J = \{\}$

Pick the two least similar objects  $o_a$  and  $o_b$  from **D**

$P(1) = a$ ;  $I = \{a\}$ ;  $J = K - \{a\}$

For  $r = 2, \dots, N$

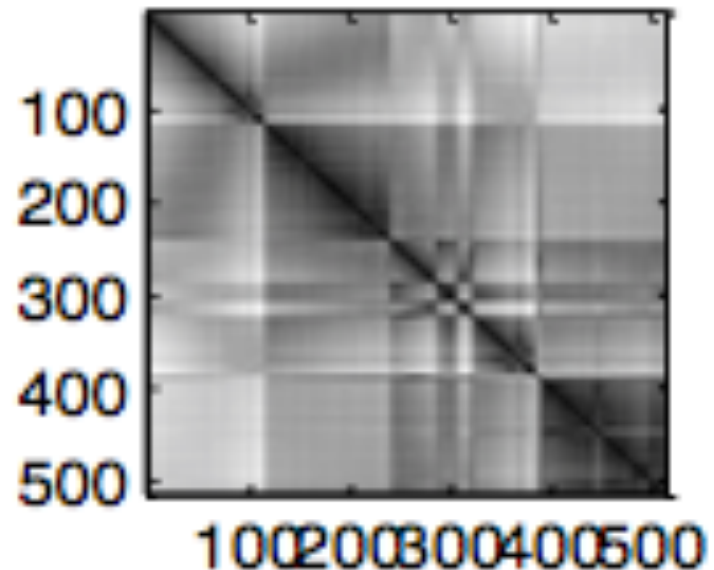
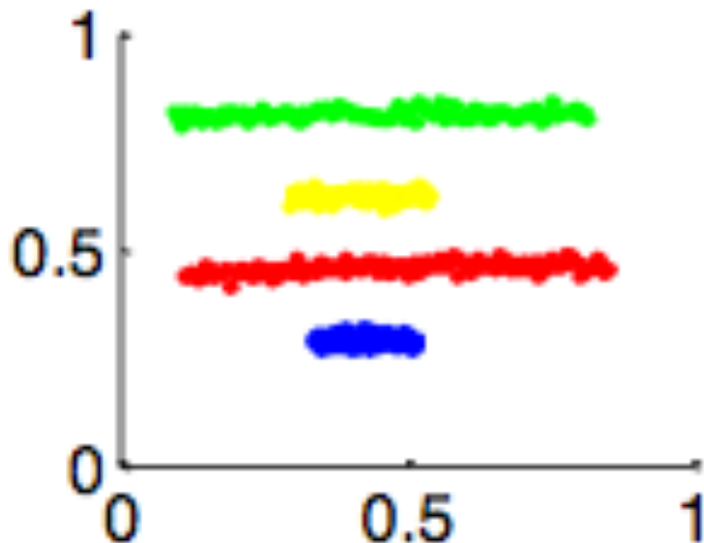
Select  $(i, j)$ : pair of most similar objects  $o_i$  and  $o_j$  from **D**

Such that  $i \in I$ ,  $j \in J$

$P(r) = j$ ;  $I = I \cup \{j\}$ ;  $J = J - \{j\}$ ;

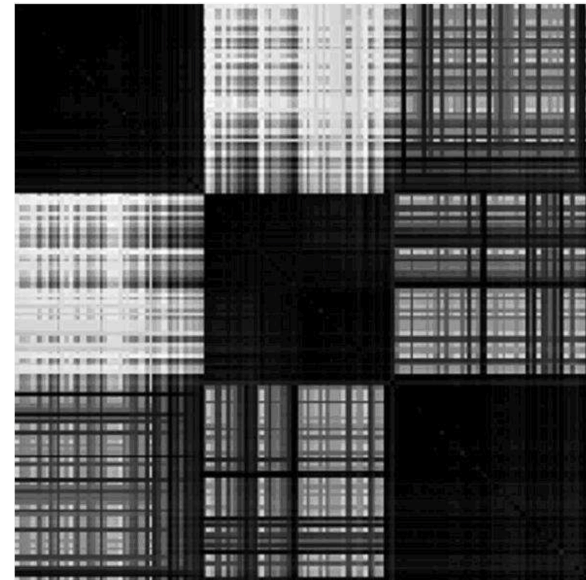
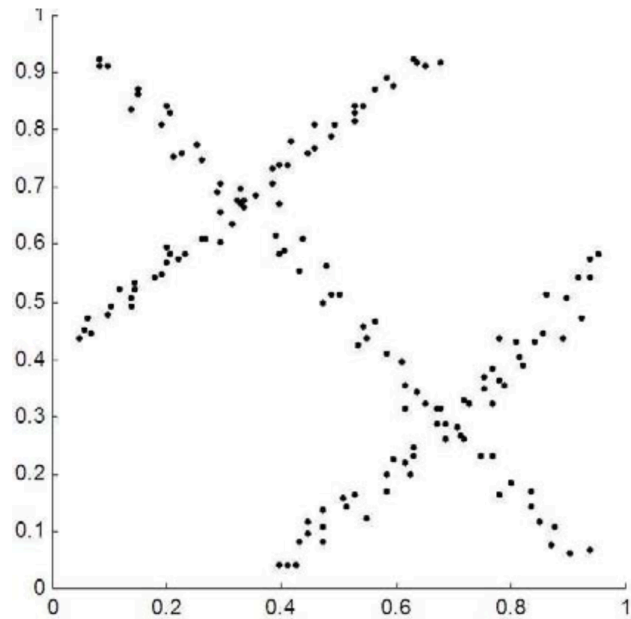
Obtain reordered dissimilarity matrix  $D^*$  from permutation  $P$

- VAT algorithm won't be effective in every situation
  - For complex shaped datasets (either significant overlap or irregular geometries between different clusters), the quality of the VAT image may significantly degrade.



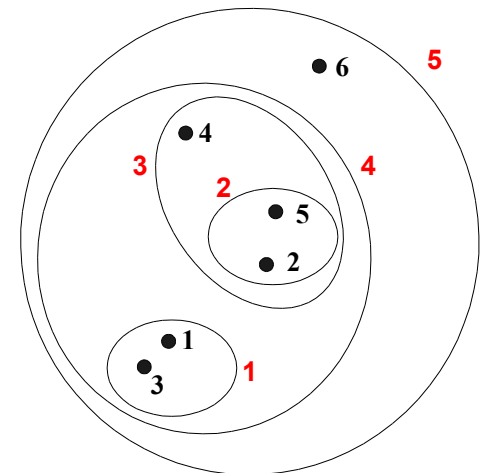
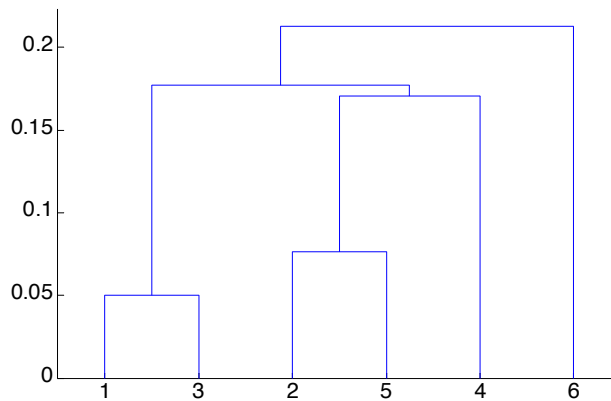


MELBOURNE





- Produces a set of nested clusters organized as a hierarchical tree
- Can be visualized as a dendrogram
  - A tree like diagram that records the sequences of merges or splits



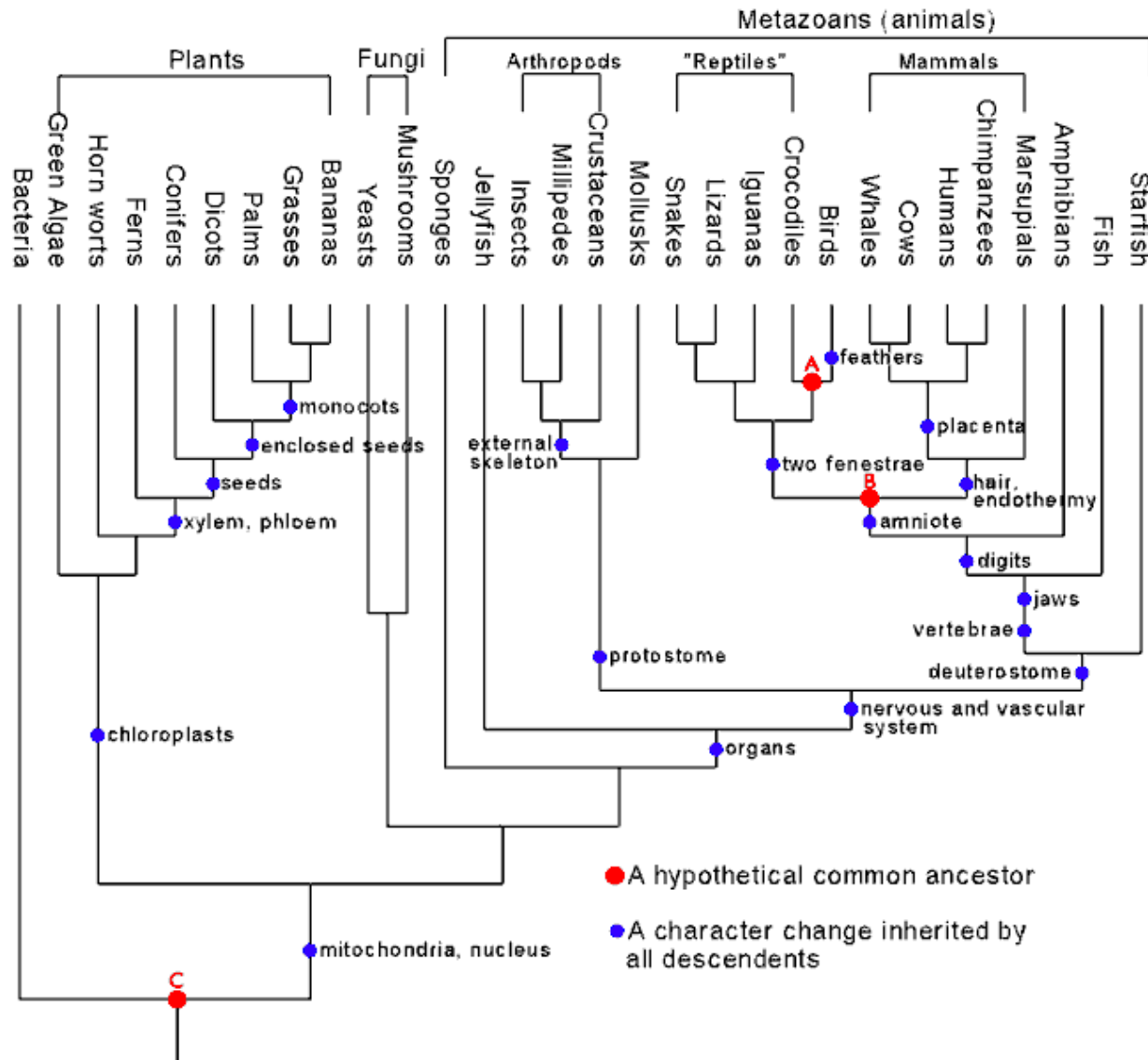


- Do not have to assume any particular number of clusters
  - Any desired number of clusters can be obtained by ‘cutting’ the dendrogram at the proper level
- They may correspond to meaningful taxonomies
  - Example in biological sciences (e.g., animal kingdom, phylogeny reconstruction, ...)



# Tree of Life

(<http://www.talkorigins.org/faqs/comdesc/phylo.html>)





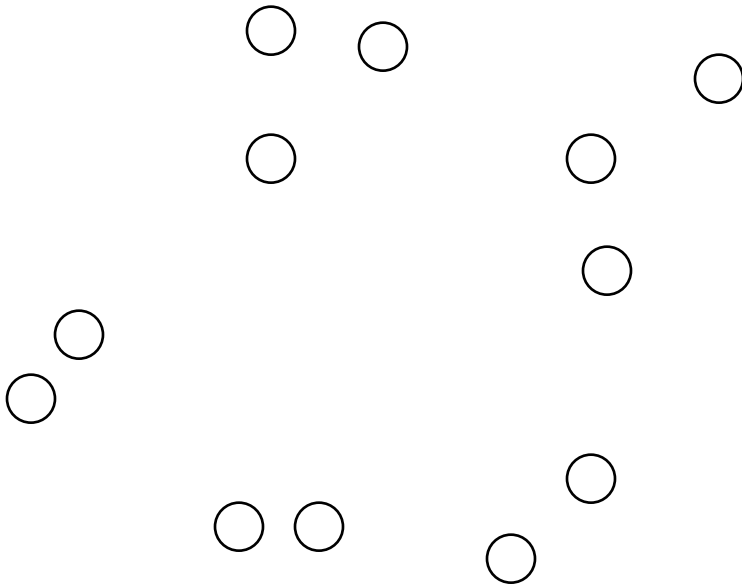
- Two main types of hierarchical clustering
  - Agglomerative:
    - Start with the points as individual clusters
    - At each step, merge the closest pair of clusters until only one cluster (or  $k$  clusters) left
  - Divisive:
    - Start with one, all-inclusive cluster
    - At each step, split a cluster until each cluster contains a point (or there are  $k$  clusters)
- Traditional hierarchical algorithms use a similarity or distance matrix
  - Merge or split one cluster at a time



- More popular hierarchical clustering technique
- Basic algorithm is straightforward
  1. Compute the proximity matrix
  2. Let each data point be a cluster
  3. **Repeat**
    - Merge the two closest clusters
    - Update the proximity matrix**Until** only a single cluster remains
- Key operation is the computation of the proximity of two clusters
  - Different approaches to defining the distance between clusters distinguish the different algorithms



- Start with clusters of individual points and a proximity matrix

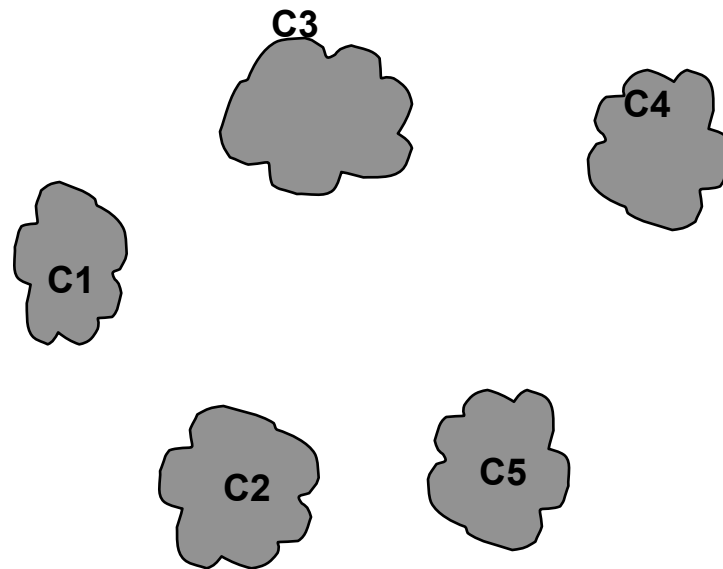


	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

**Proximity Matrix**

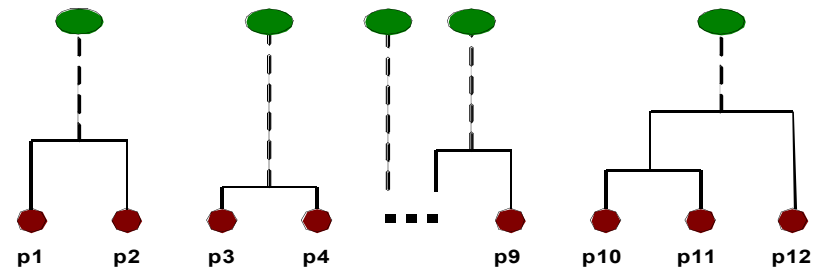


- After some merging steps, we have some clusters

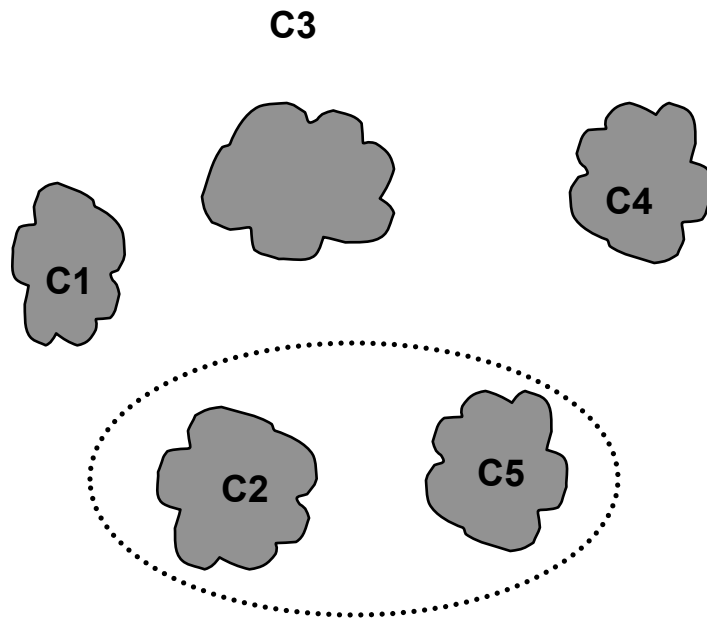


	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Proximity Matrix

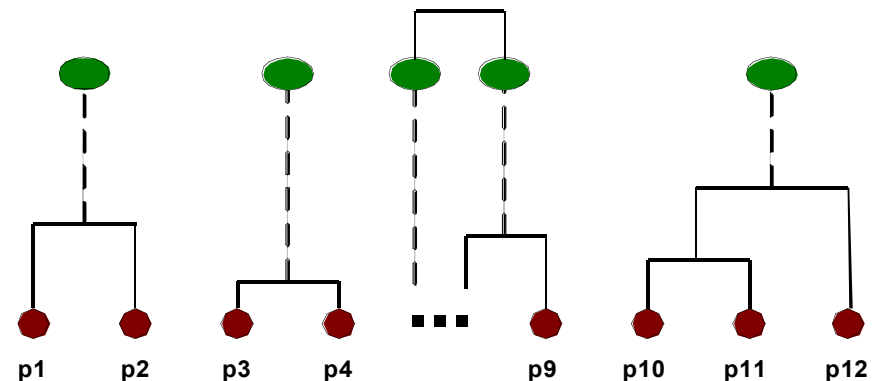


- We want to merge the two closest clusters (C2 and C5) and update the proximity matrix.



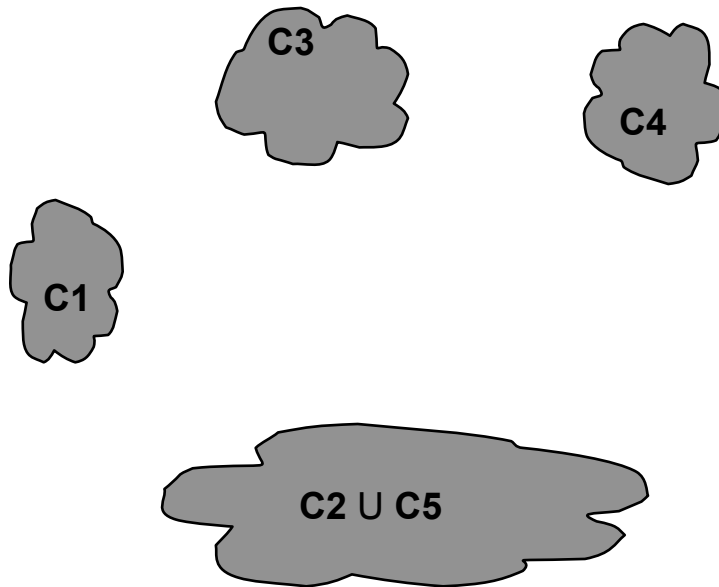
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

**Proximity Matrix**



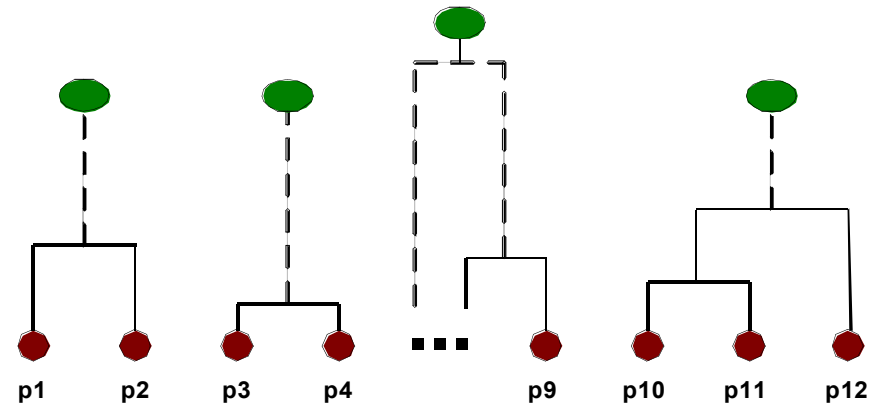


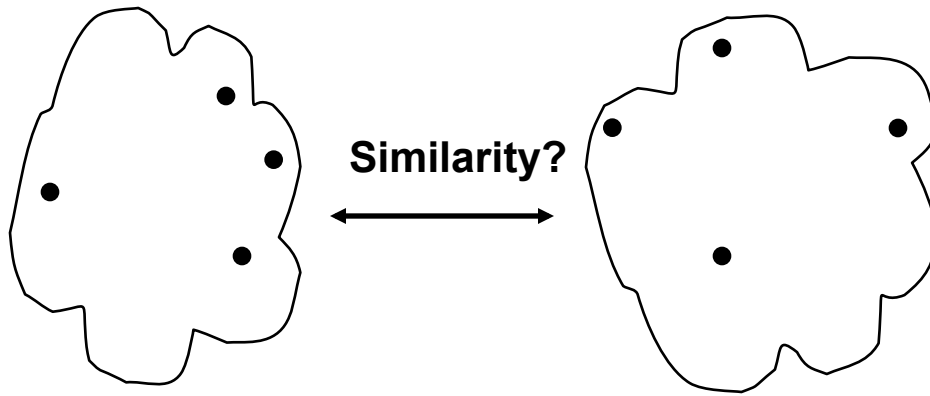
- The question is “How do we update the proximity matrix?”



			<b>C2</b>		
			<b>U</b>		
		<b>C1</b>	<b>C5</b>	<b>C3</b>	<b>C4</b>
	<b>C1</b>		?		
<b>C2 U</b>	<b>C5</b>	?	?	?	?
	<b>C3</b>		?		
	<b>C4</b>		?		

## Proximity Matrix





- We define the similarity to be the minimum distance between the clusters. This is also known as single linkage.

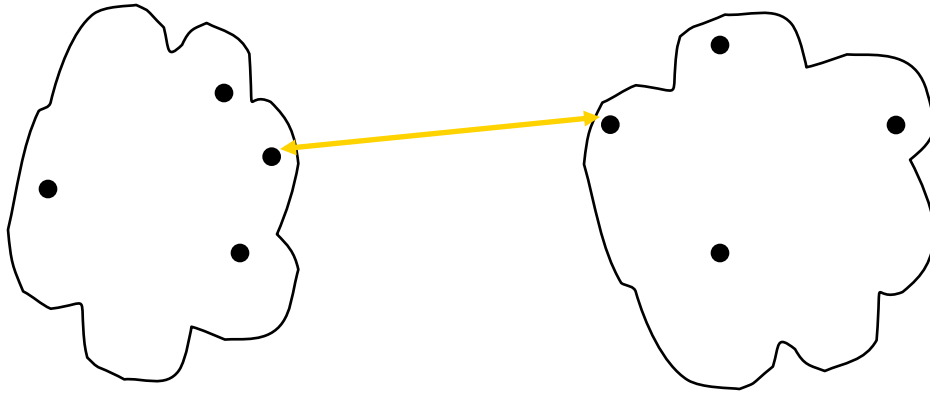
- Other choices also possible (e.g. max or average), but we won't cover these)

	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.						

**Proximity Matrix**

# How to Define Inter-Cluster Similarity

MELBOURNE



- MIN (Single Linkage)

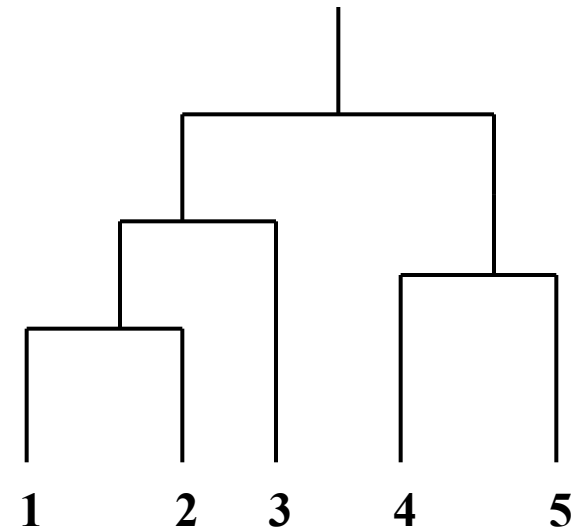
	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

• **Proximity Matrix**



- Similarity of two clusters is based on the two most similar (closest) points in the different clusters
  - Determined by one pair of points, i.e., by one link in the proximity graph.

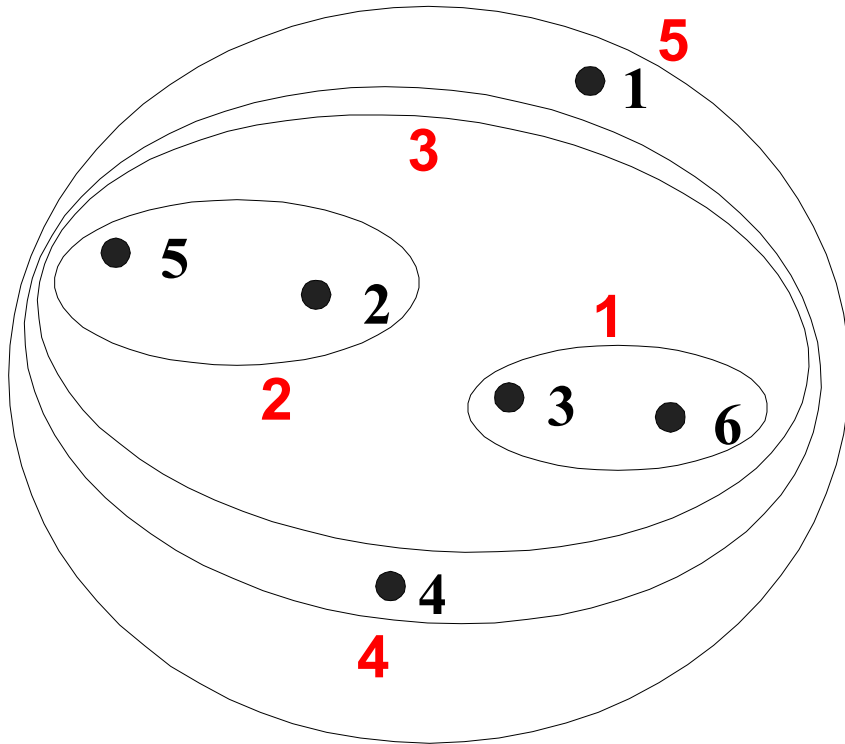
	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



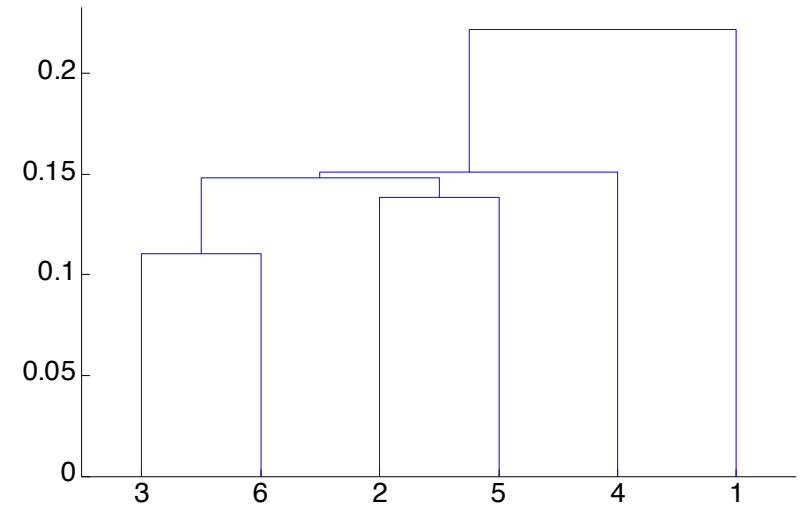


# Hierarchical Clustering: MIN

MELBOURNE



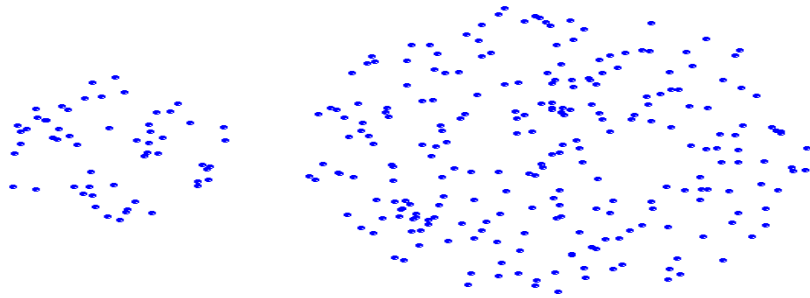
**Nested Clusters**



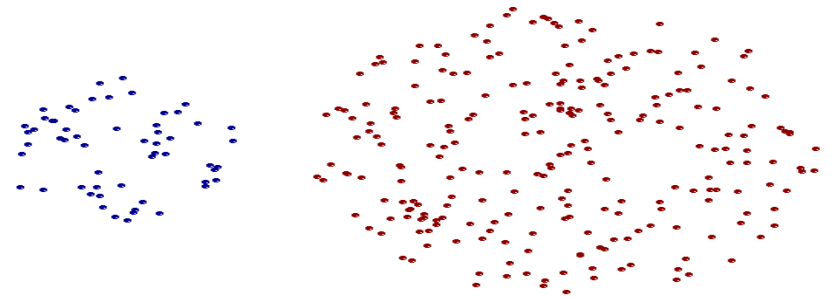
**Dendrogram**



MELBOURNE

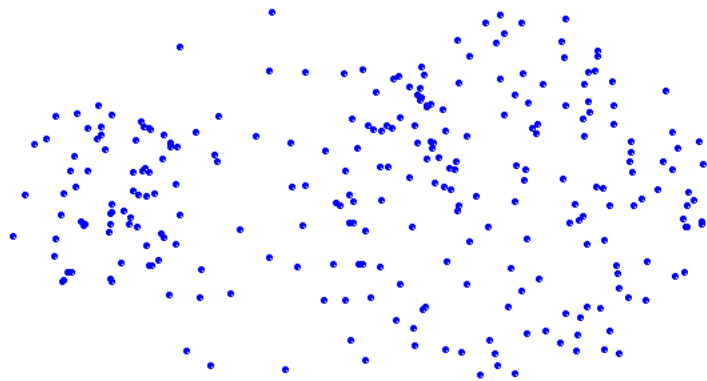


**Original Points**

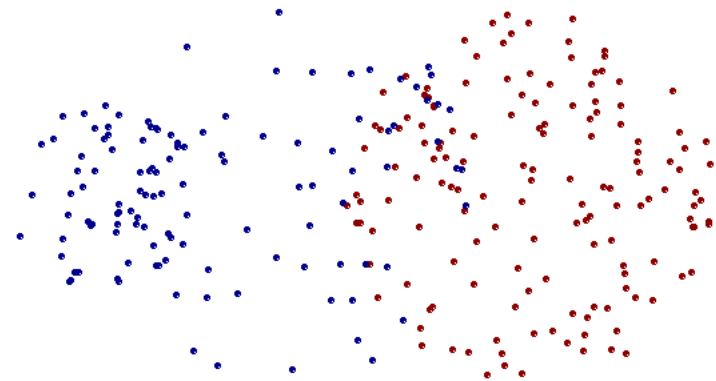


**Two Clusters**

- Can handle non-elliptical shapes



**Original Points**



**Two Clusters**

- **Sensitive to noise and outliers**



- Once a decision is made to combine two clusters, it cannot be undone
- No objective function is directly minimized





- Motivation and intuition
- Principal components analysis

- **The curse of dimensionality:** “Data analysis techniques which work well at lower dimensions, often perform poorly as the dimensionality of the analysed data increases”
- As dimensionality increases, data becomes increasingly sparse and all the distances between pairs of points begin to look the same. Impacts any algorithm that is based on distances between objects.
- For a number of data distributions, as the dimensions increase

$$\lim_{dim \rightarrow \infty} \frac{dist_{max} - dist_{min}}{dist_{min}} \rightarrow 0$$

- $dist_{max}$  is the maximum distance in the dataset between a pair of objects and  $dist_{min}$  is the minimum distance in the dataset between a pair of objects



- Purpose:
  - Avoid curse of dimensionality
  - Reduce amount of time and memory required by data processing algorithms
  - Allow data to be more easily visualized
  - May help to eliminate irrelevant features or reduce noise



- Input: A dataset with  $N$  features and  $K$  objects
- Output: A transformed dataset with  $n \ll N$  features and  $K$  objects
  - $n$  is often set to 2 or 3, so that the transformed dataset can be easily visualised
- E.g if  $n=2$

Object id	Feature1	Feature2	....	FeatureN
1	...	...	...	...
...	....	...	...	...
K	....	....	...	...

↓ Transformation

Object id	NewFeatureA	NewFeatureB
1	...	...
...	...	...
K	...	...



- The transformation must preserve the characteristics of the data
  - In particular, preserve distances between pairs of points
- If a pair of objects is close before the transformation, they should still be close after the transformation
- If a pair of objects is far apart before the transformation, they should still be far apart after the transformation
- The set of nearest neighbors of an object before the transformation should ideally be the same after the transformation



- Suppose we are given a dataset with the following  $N$  features, describing individuals in this class. Which two features would you select to represent people, in such a way that “distances” between pairs of people are likely to be preserved in the reduced dataset?
- Input:  $N=7$  features
  1. Weighted average mark (WAM)
  2. Age (years)
  3. Height (cm)
  4. Weight (kg)
  5. Number of pets owned
  6. Number of subjects passed so far
  7. Amount of sleep last night (0=little, 1=medium, 2=a lot)
- Output: Select 2 of the above features



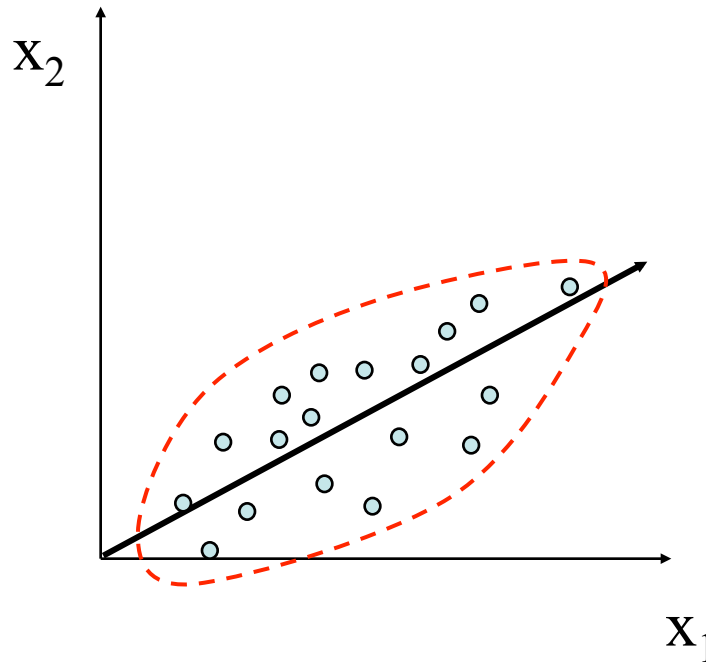
- Basic method: To reduce dimensionality, can just select a subset of the original features.
  - Scatter plots for Iris dataset shown earlier – 2D visualisations of a 3D dataset. 2 features were selected from the original 3.
- In general, when transforming a dataset from  $N$  to  $n \ll N$  features
  - *The output  $n$  features do not need to be a subset of the input  $N$  features.* Rather, they can be new features whose values are constructed using some function applied to the input  $N$  features



- Find a new set of features that better captures the variability of the data
- First dimension chosen to capture as much of the variability as possible.
- The second dimension is orthogonal to the first and, and subject to that constraint, captures as much of the remaining variability as possible,
- The third dimension is orthogonal to the first and second, and subject to that constraint, captures as much of the remaining variability as possible.
- We will not be covering the mathematical details
  - Many tutorials available on the Web if you are interested.  
Nice application of linear algebra.



- Goal is to find a projection that captures the largest amount of variation in data. Below – the 1-D direction capturing most of the variation in the data. Use this to transform from 2D to 1D.



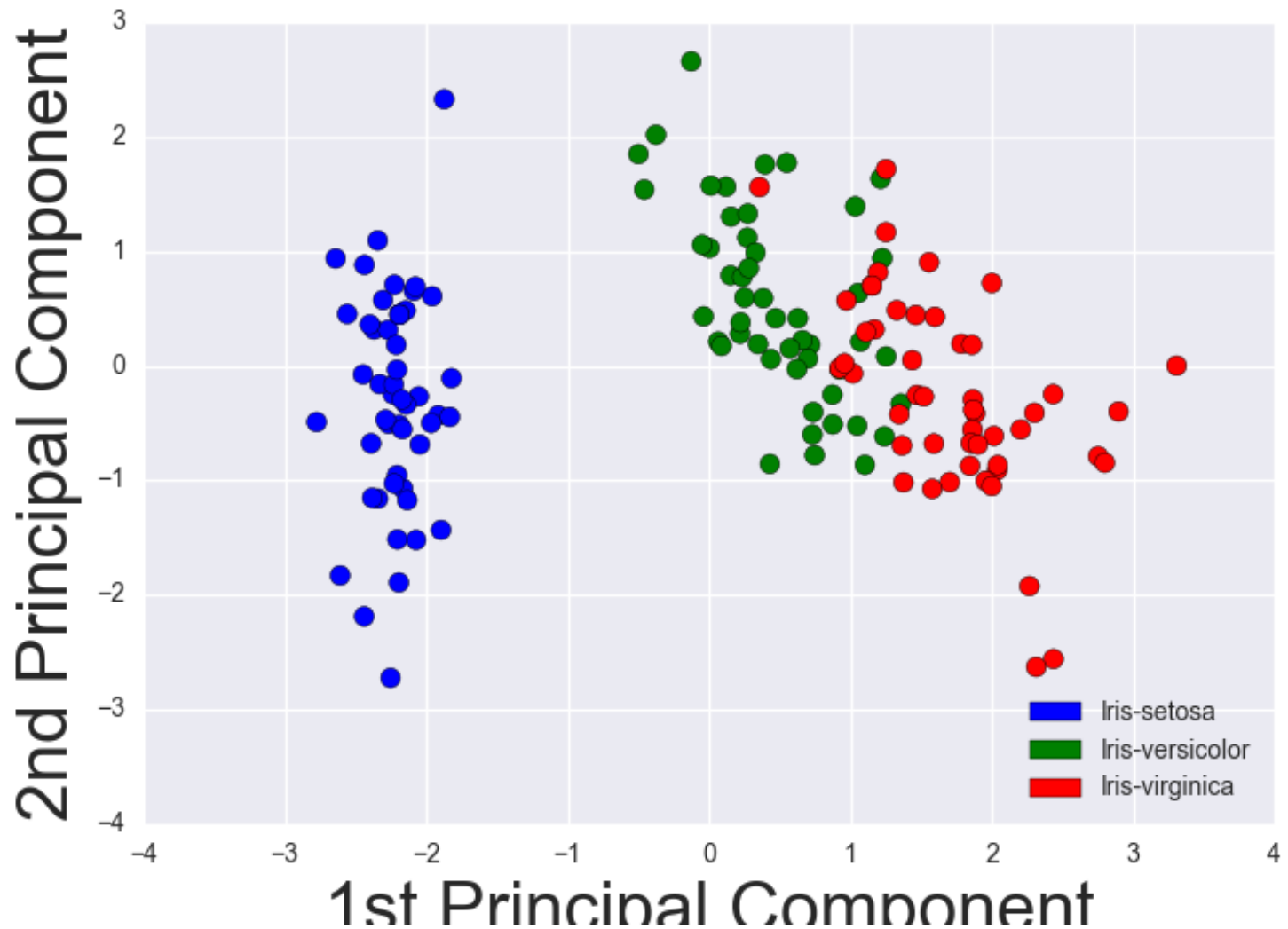


- A good visualisation for PCA
  - <http://setosa.io/ev/principal-component-analysis/>

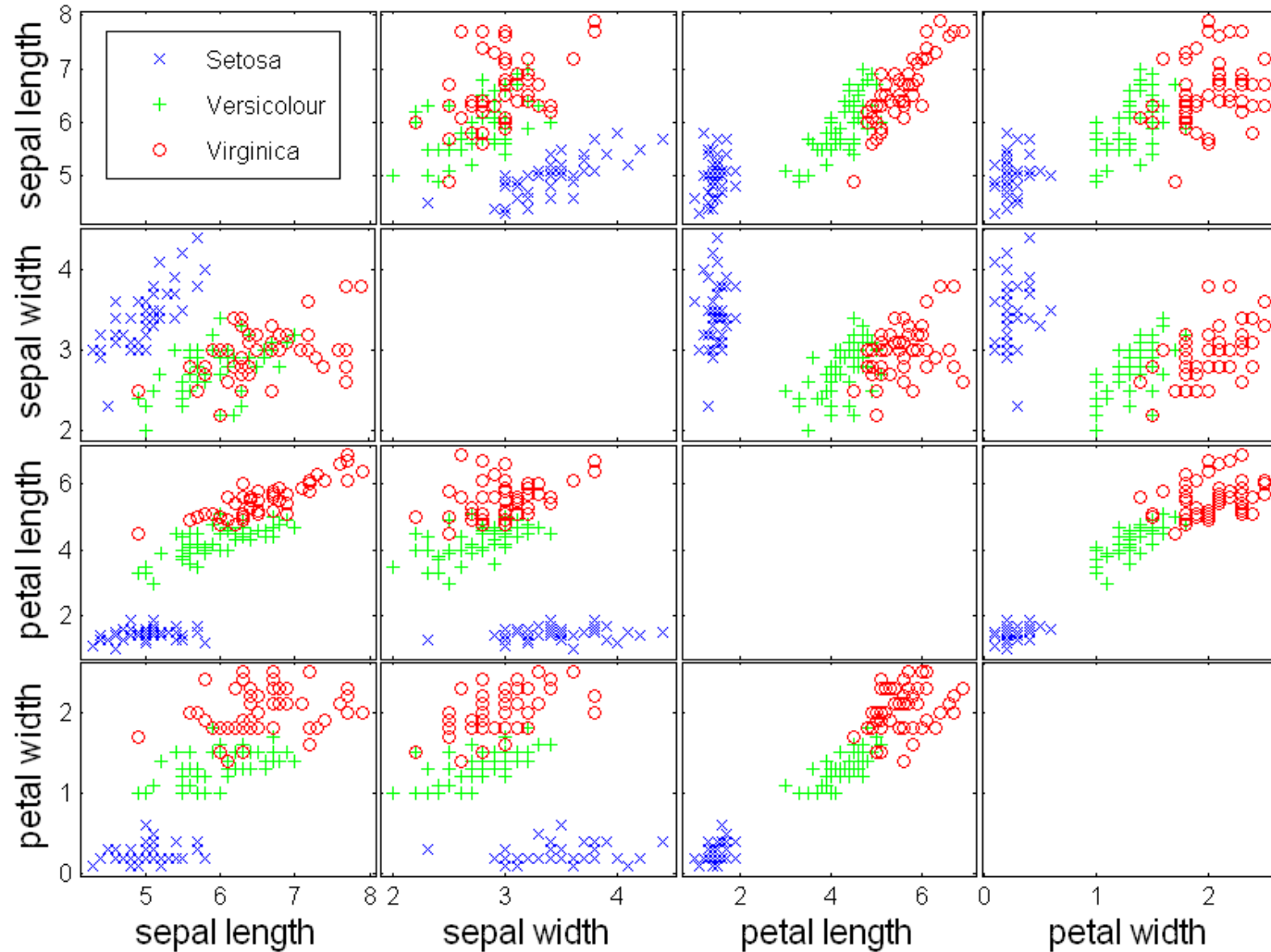


# Iris dataset: Reduced from 3 to 2 dimensions using PCA

MELBOURNE



## Contrast this against an earlier slide



Scatter plots for iris dataset



[\[Stats Main\]](#)[\[AFL Main\]](#)

[\[2013 Stats\]](#)[\[2015 Stats\]](#)

## 2014 Player Stats

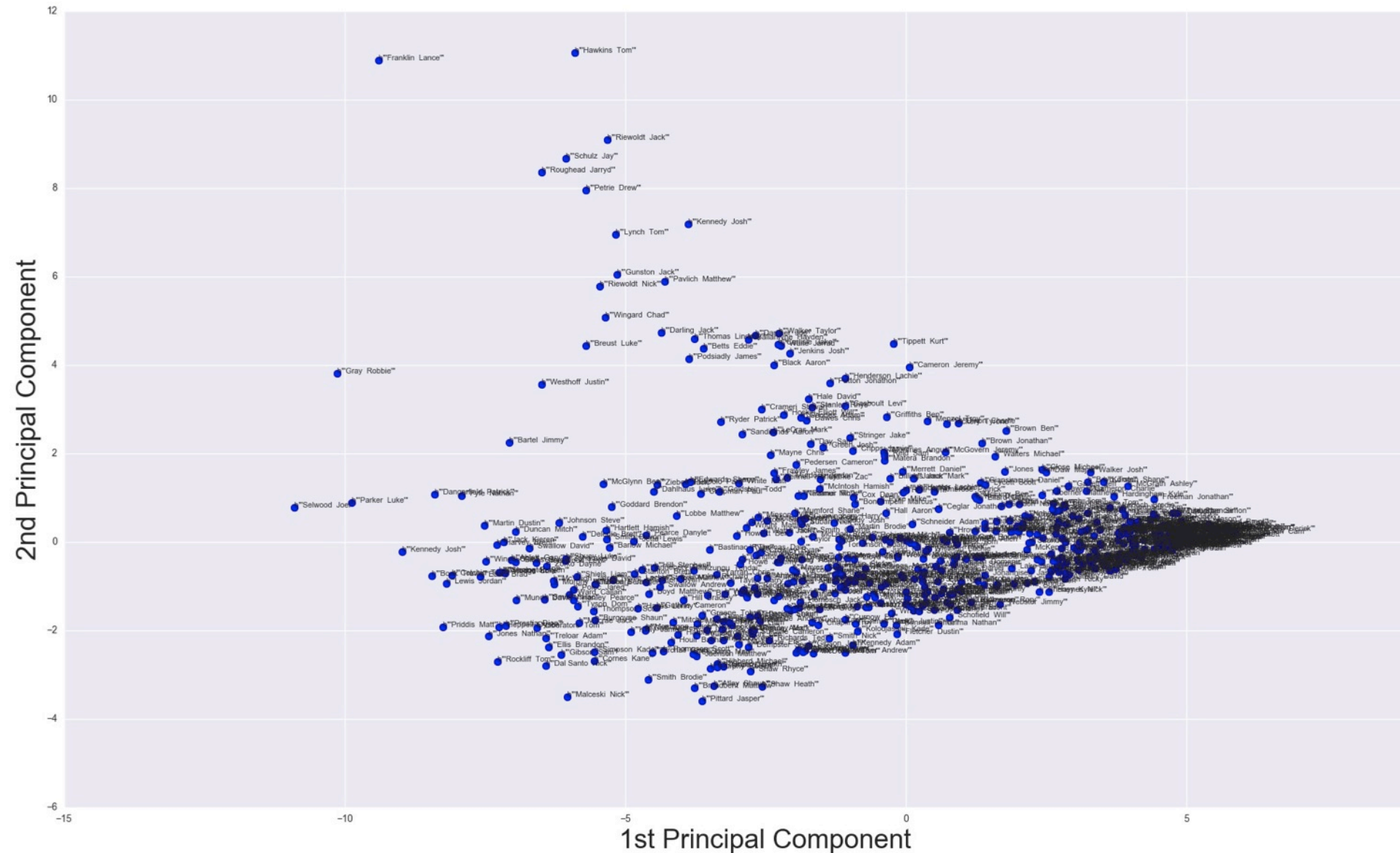
[\[2014 Stats Summary\]](#)

[\[Adelaide\]](#)[\[Brisbane Lions\]](#)[\[Carlton\]](#)[\[Collingwood\]](#)[\[Essendon\]](#)[\[Fremantle\]](#)[\[Geelong\]](#)[\[Gold Coast\]](#)[\[Greater Western Sydney\]](#)[\[Hawthorn\]](#)  
[\[Melbourne\]](#)[\[North Melbourne\]](#)[\[Port Adelaide\]](#)[\[Richmond\]](#)[\[St Kilda\]](#)[\[Sydney\]](#)[\[West Coast\]](#)[\[Western Bulldogs\]](#)  
[\[All Teams\]](#)

Abbreviations key

Adelaide <a href="#">[Game by Game]</a>																											
#	Player	GM	KI	MK	HB	DI	DA	GL	BH	HO	TK	RB	IF	CL	CG	FF	FA	BR	CP	UP	CM	MI	1%	BO	GA	%P	SU
32	<a href="#">Dangerfield, Patrick</a>	22	276	74	272	548	24.91	17	22	28	78	33	104	136	66	34	19	21	341	210	25	16	35	18	10	83.7	
9	<a href="#">Sloane, Rory</a>	22	269	105	252	521	23.68	13	9	19	147	45	99	92	50	26	15	10	275	256	9	7	64	5	21	87.2	
5	<a href="#">Thompson, Scott</a>	19	257	69	262	519	27.32	3	7	2	86	28	77	118	61	19	22	14	224	280	3	5	21	1	7	81.7	0/2
33	<a href="#">Smith, Brodie</a>	22	287	108	209	496	22.55	11	8		35	109	76	18	45	9	6	4	142	319	7	2	56	46	7	87.0	
10	<a href="#">Jaensch, Matthew</a>	22	297	126	166	463	21.05	7	5		54	89	54	7	34	19	10		106	325	16	1	57	34	3	81.3	
26	<a href="#">Douglas, Richard</a>	19	266	52	147	413	21.74	11	8	4	91	21	96	91	38	22	17		182	228	2	6	36	13	11	86.4	
11	<a href="#">Wright, Matthew</a>	20	224	89	150	374	18.70	14	8		68	22	47	39	27	30	6		141	227	4	12	26	6	17	80.0	1/2
24	<a href="#">Jacobs, Sam</a>	22	193	90	165	358	16.27	7	3	763	46	20	40	69	33	11	15	6	150	189	19	4	63	1	10	87.9	0/1
14	<a href="#">Mackay, David</a>	19	168	58	174	342	18.00	11	7		77	30	62	32	31	22	13		127	224	5	3	34	37	8	81.1	0/2
18	<a href="#">Betts, Eddie</a>	22	167	53	123	290	13.18	51	22		74	8	37	30	39	19	16	4	149	136	3	29	21	8	29	87.7	
1	<a href="#">Podsiadly, James</a>	21	189	119	101	290	13.81	26	14	2	37	17	52	2	63	14	25	4	132	165	41	35	60	1	16	90.1	
16	<a href="#">Brown, Luke</a>	22	138	55	148	286	13.00	1	1		54	37	16	8	18	13	5		81	205	1	1	42	1	4	84.5	
2	<a href="#">Crouch, Brad</a>	11	125	26	147	272	24.73	5	6	1	61	22	40	56	30	8	6		114	156	1	2	17	9	6	83.7	0/1
36	<a href="#">Martin, Brodie</a>	17	155	65	109	264	15.53	8	15		45	30	38	23	40	13	11		97	174	7	12	34	11	4	69.2	2/1
12	<a href="#">Talia, Daniel</a>	22	167	105	93	260	11.82		1		24	45	25		29	11	12		79	183	13		149	1	2	90.0	0/1
29	<a href="#">Laird, Rory</a>	16	126	65	129	255	15.94	2	2		37	21	34	15	31	8	8		81	177	1	1	25	2	2	75.4	2/0
4	<a href="#">Jenkins, Josh</a>	20	170	86	64	234	11.70	40	26	55	27	13	46	11	36	12	8	3	97	140	21	32	48	10	7	90.6	
13	<a href="#">Walker, Taylor</a>	15	138	84	82	220	14.67	34	22		24		50		47	10	21	5	102	120	23	31	20		17	90.3	
3	<a href="#">Reilly, Brent</a>	10	130	65	63	193	19.30				19	32	17	8	30	3	13		46	139	7		19	24	1	81.0	
17	<a href="#">Kerridge, Sam</a>	14	72	33	84	156	11.14	10	1		52	10	23	26	25	3	14		54	97	2	9	9	4	5	83.7	0/1

# AFL Football Dataset: PCA in 2D





- Code
  - PCA in `sklearn.decomposition`
  - Will practice in workshop



- Material partly adapted from
  - “Data Mining Concepts and Techniques”, Han et al, 2<sup>nd</sup> edition 2006.
  - “Introduction to Data Mining”, Tan et al 2005.