Biostat 202: Opportunities and Challenges of "Big Data": Machine Learning 5: Clustering and Data Reduction

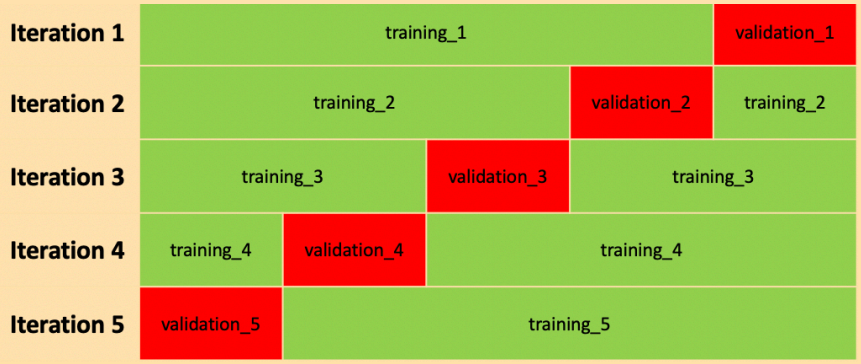
Aaron Wolfe Scheffler

Division of Biostatistics

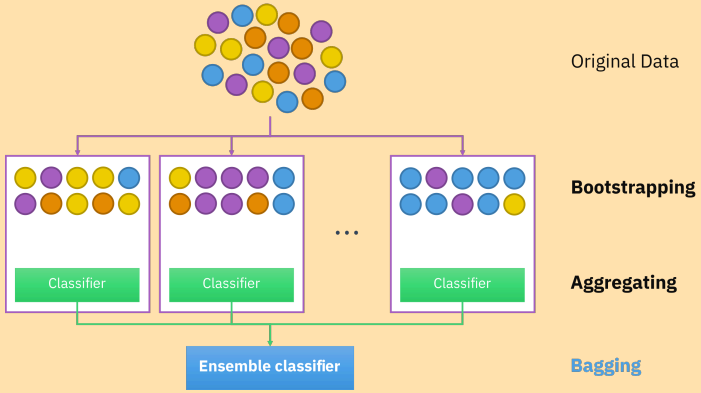
Department of Epidemiology and Biostatistics

Review of last lecture

1. Cross-validation



2. Bagging



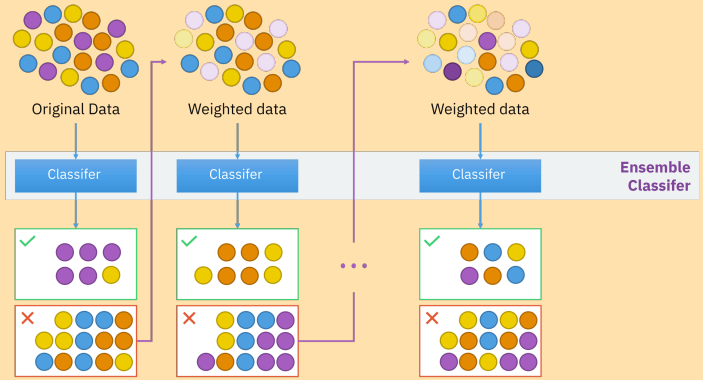
Original Data

Bootstrapping

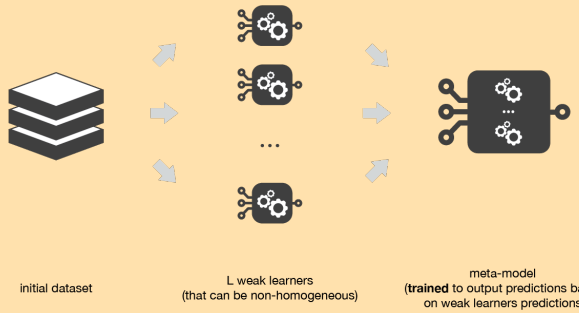
Aggregating

Bagging

3. Boosting



4. Stacking



initial dataset

L weak learners (that can be non-homogeneous)

meta-model (**trained** to output predictions based on weak learners predictions)

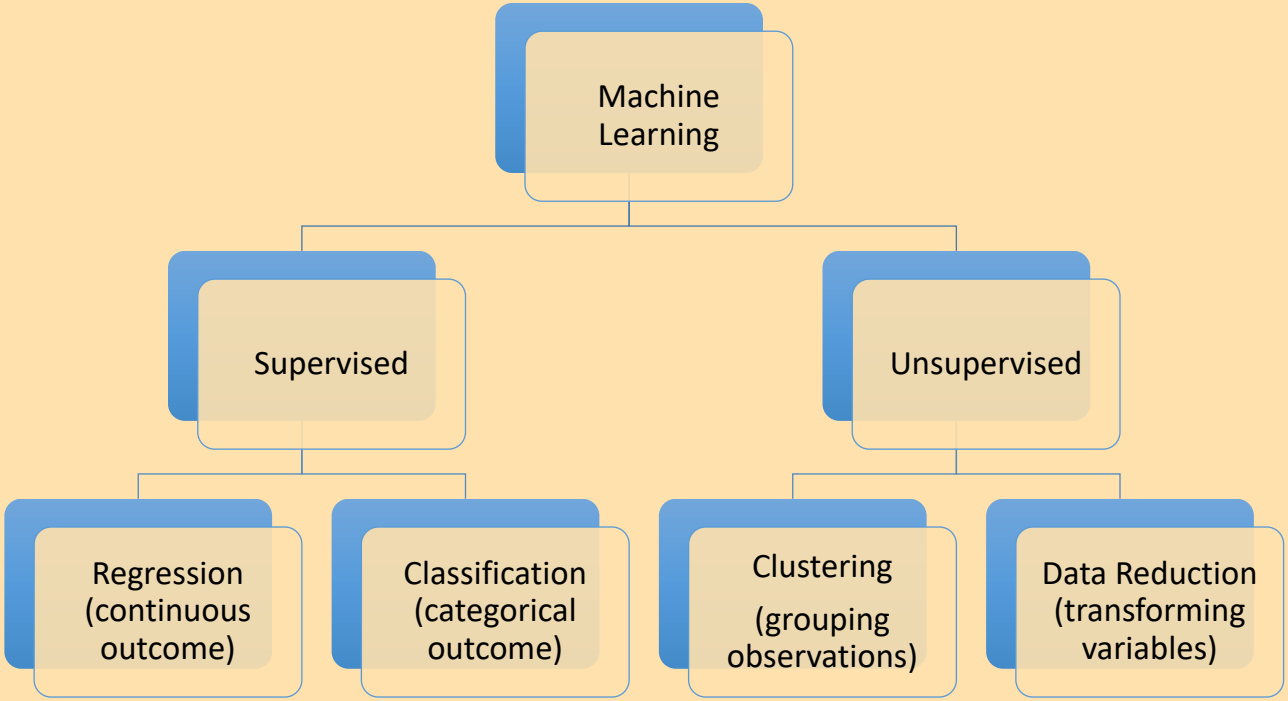
5. Feature Importance

Height at age 20 (cm)	Height at age 10 (cm)	...	Socks
182	155	...	
175	147	...	
...	...	...	
156	142	...	
153	130	...	

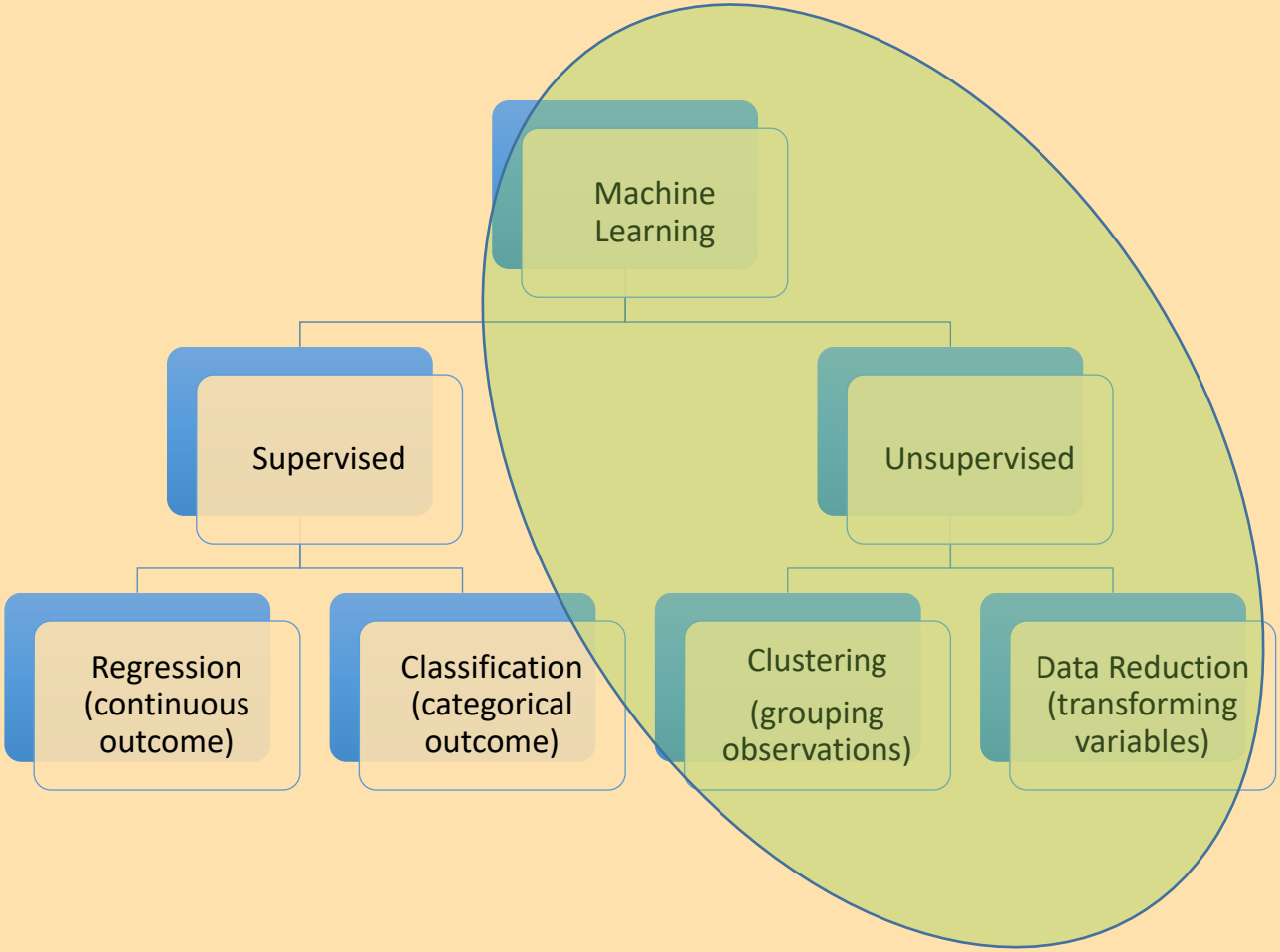
Overview of this lecture

- Clustering
  - k-means clustering
  - Hierarchical clustering
- Data reduction
  - Principal components analysis (PCA)
  - t-Distributed Stochastic Neighbor Embedding (t-SNE)

Supervised vs. unsupervised



Supervised vs. unsupervised



Supervised vs unsupervised

- **Supervised Learning**
- Have data where you know outcome in order to *train* the model
- Train the model to predict future outcomes from sets of predictors

Prediction: Regression (continuous outcomes) and classification (categorical outcomes)

- **Unsupervised Learning**
- No desired outcomes
- Separate data points into groups with "similar" properties (clustering)
- Or reduce data to fewer variables in some way that still captures important structure of the data

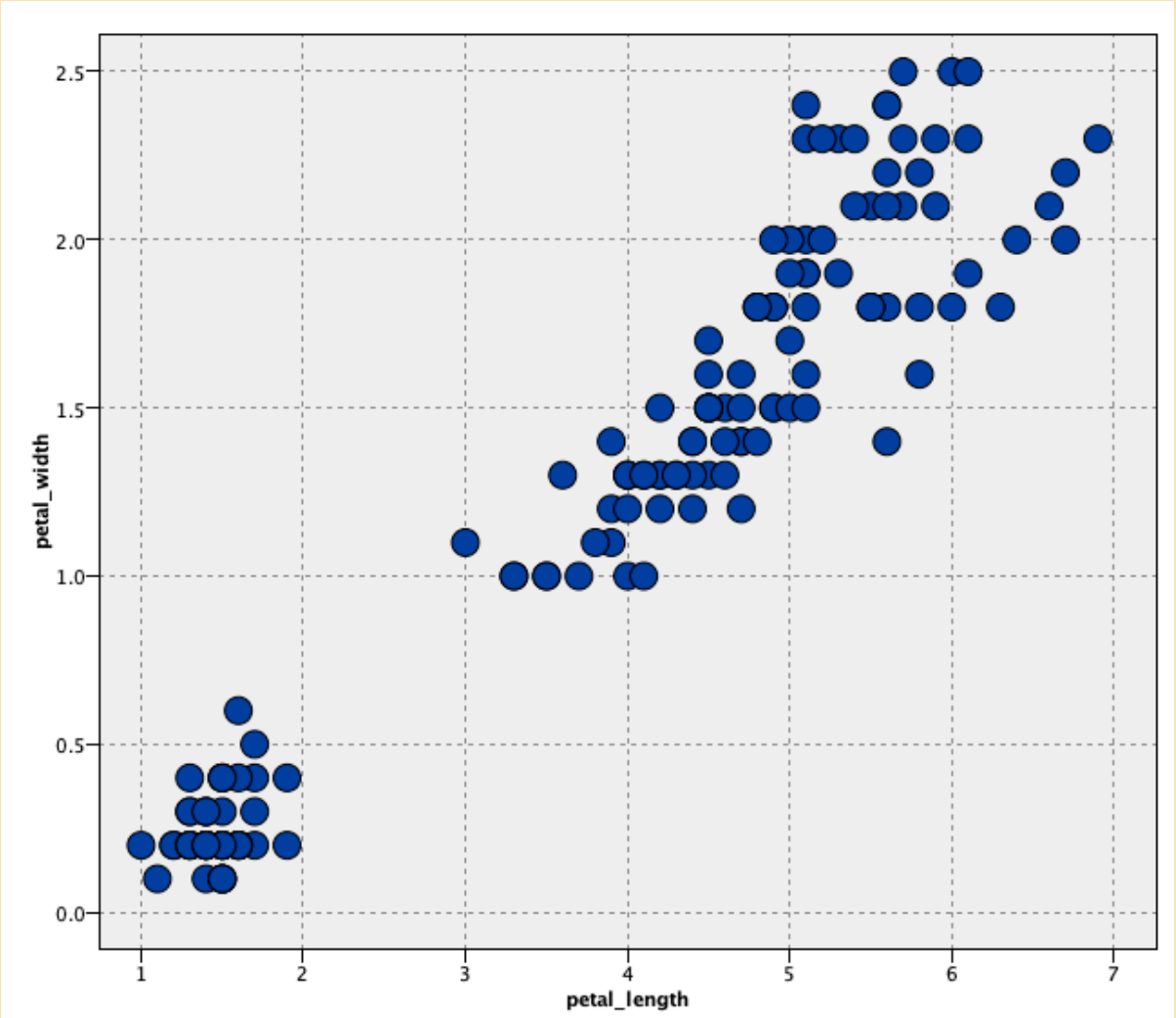
Clustering and data reduction

Clustering

Clustering problems

- Determine whether there are distinct (unknown) subclasses of brain cancer based on symptomatic and tumor characteristics
- Netflix example: you want to discover microgenres of movies and apply labels

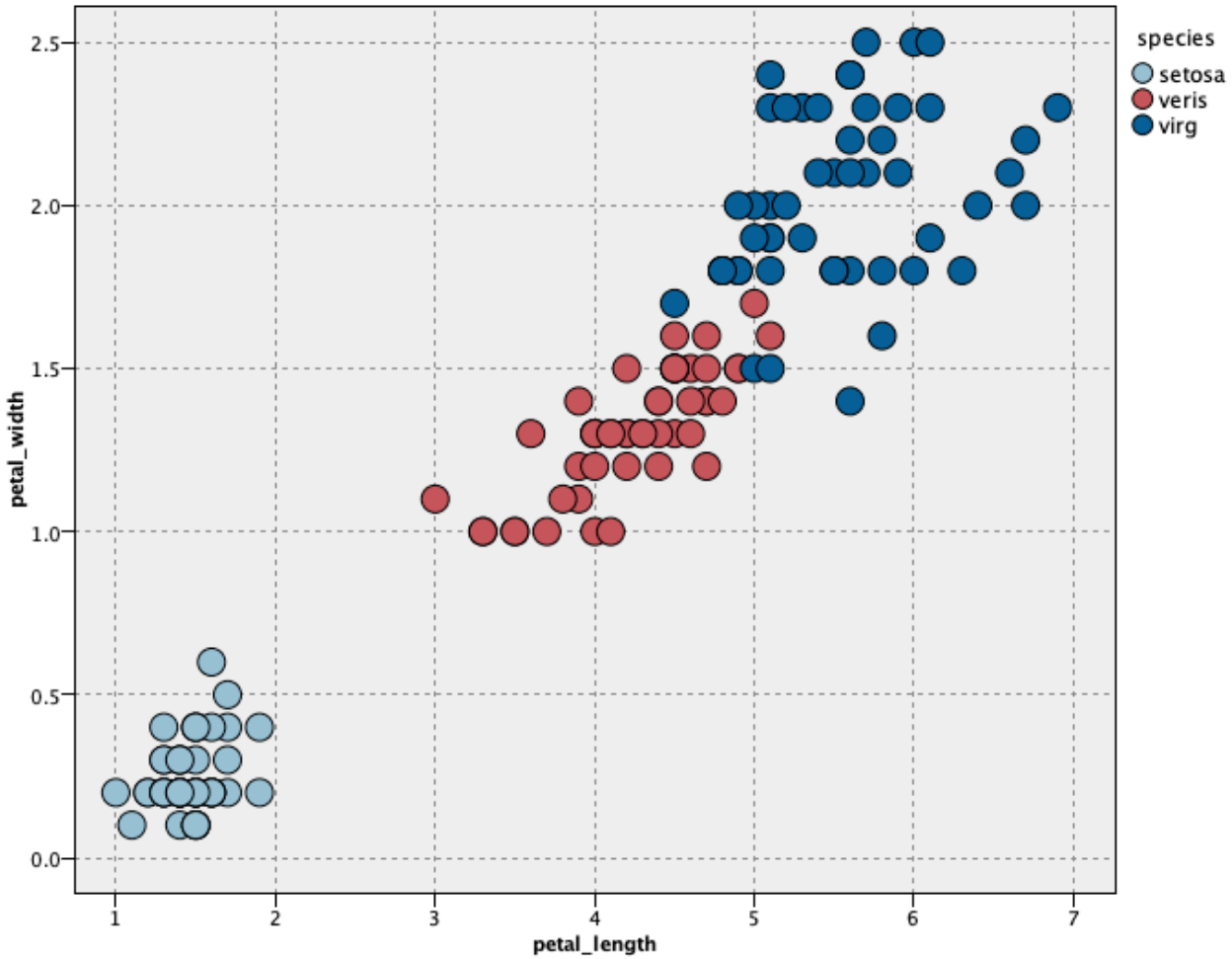
2D Clustering: Iris example



How many clusters?

Where are the clusters?

2D Clustering: Iris example

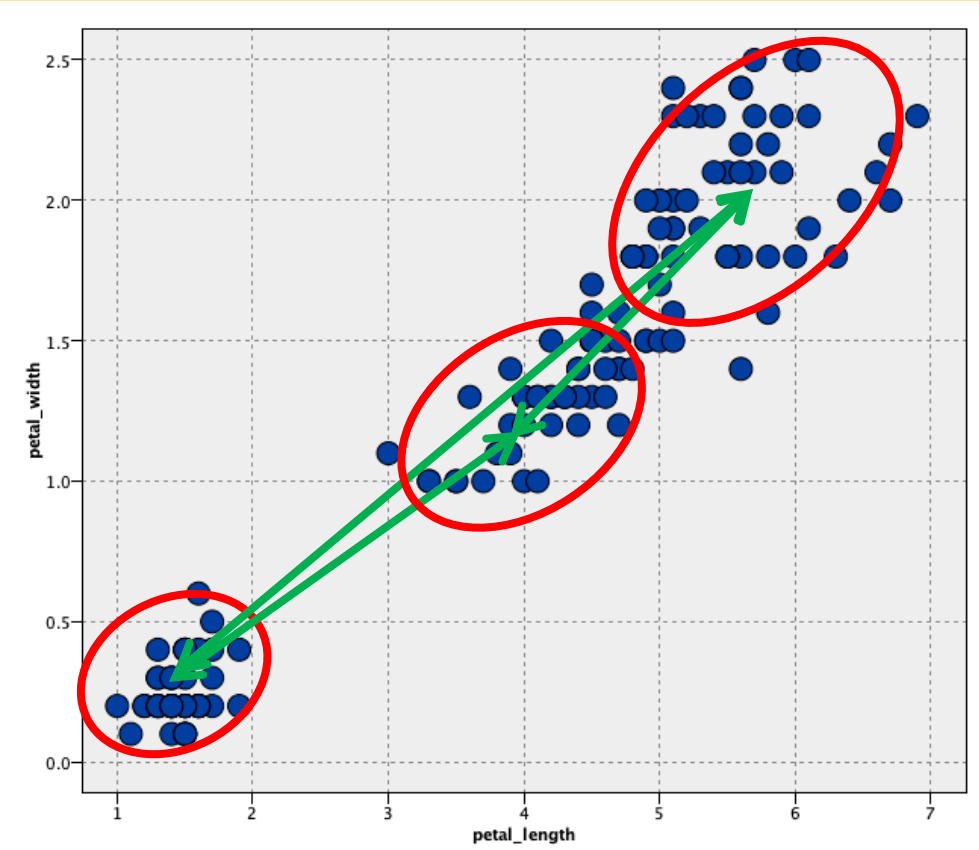


There are 3 species!

Cluster analysis (data segmentation)

- Objective is to group (or segment) collection of observations into subsets (or *clusters*)
- There is no “Target” in cluster analysis (unsupervised)
- Clusters are not predefined, they are discovered!
- Observations within a cluster are more similar (or less dissimilar) than observations in different clusters

2D clustering: Iris example



When determining clusters

We want long green lines (large distance between cluster centers)

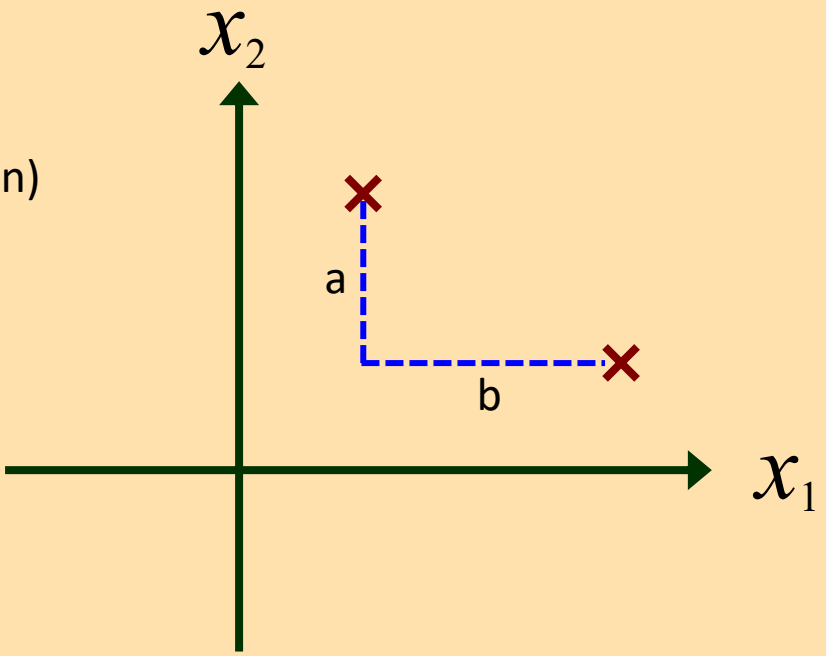
We want small ellipses (points within clusters grouped tightly together)

Goal: maximize similarity within clusters and minimize similarity between clusters

Measures of similarity/dissimilarity

- Need to define similarity (or dissimilarity) and we need to be able to measure it
- Consider 2 observations with only 2 variables x_1 and x_2 – dissimilarities could be

- Euclidean = $\sqrt{a^2 + b^2}$ (most common)
- Absolute distance = $|a| + |b|$
- Can also differentially weight attributes e.g. $\sqrt{ka^2 + b^2}$



Extension to 3D for Euclidean distance would be e.g. $\sqrt{a^2 + b^2 + c^2}$, and so on for higher dimensions

Measures of similarity/dissimilarity

- Generally need to “standardize variables” (recall from k-nearest neighbors)
- Ordinal variables are treated the same as continuous variables
- Nominal variable differences between groups need to be explicitly given. A common choice is difference 1 if the groups are the same and 0 if they are different (this needs to be standardized following calculation)
- **Goal**: maximize similarity within clusters and minimize similarity between clusters!!!

k-means clustering

k-means clustering overview

- Algorithm splits observations into K clusters, where K can equal 2, 3, 4, …
- Uses Euclidean distance measure $\sqrt{a^2 + b^2 + c^2}$…
- Common practice to try a bunch of different values for K and use some kind of measure to determine the best K after the fact, i.e. parameter tuning
- Orange uses the “Silhouette measure” to evaluate cluster homogeneity
  - Silhouette measures ranges from -1 (bad) to 1 (good)

k-means algorithm

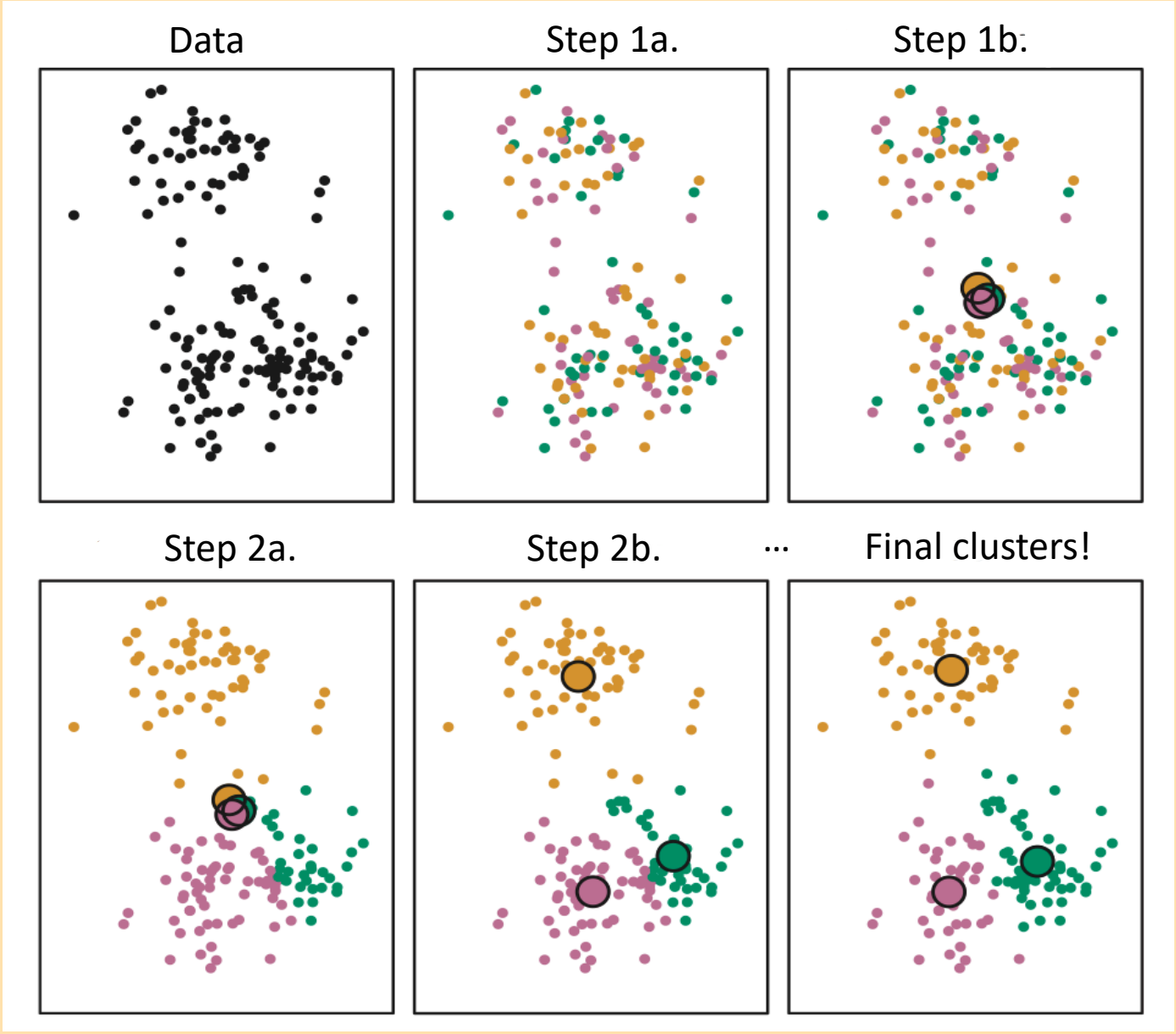
- Algorithm proceeds as follows:
  1. Randomly assign a cluster number (between 1 and K) to each of the observations/instances in the dataset. These are the *initial* cluster assignments
  2. Iterate between the following until the cluster assignments no longer change:
     a. For each cluster determine the center (centroid) of the cluster
     b. Reassign each observation to the cluster whose center is *closest* to them in terms of Euclidean distance (i.e., the center that is the most *similar* to the observation)
- The algorithm increases the within-cluster similarity relative to the between cluster similarity at every step

k-means algorithm

- The center of a cluster is simply the point associated with the mean of each of the attributes for the observations assigned to that cluster – e.g. in the Age direction the center would be located at the mean Age for all subjects in the cluster.
- Note that for different initial random assignments of observations to clusters, k-means may lead to different final clustering (it is a *local* not *global minimizer* of dissimilarity)

The final clusters may be sensitive to your starting values!

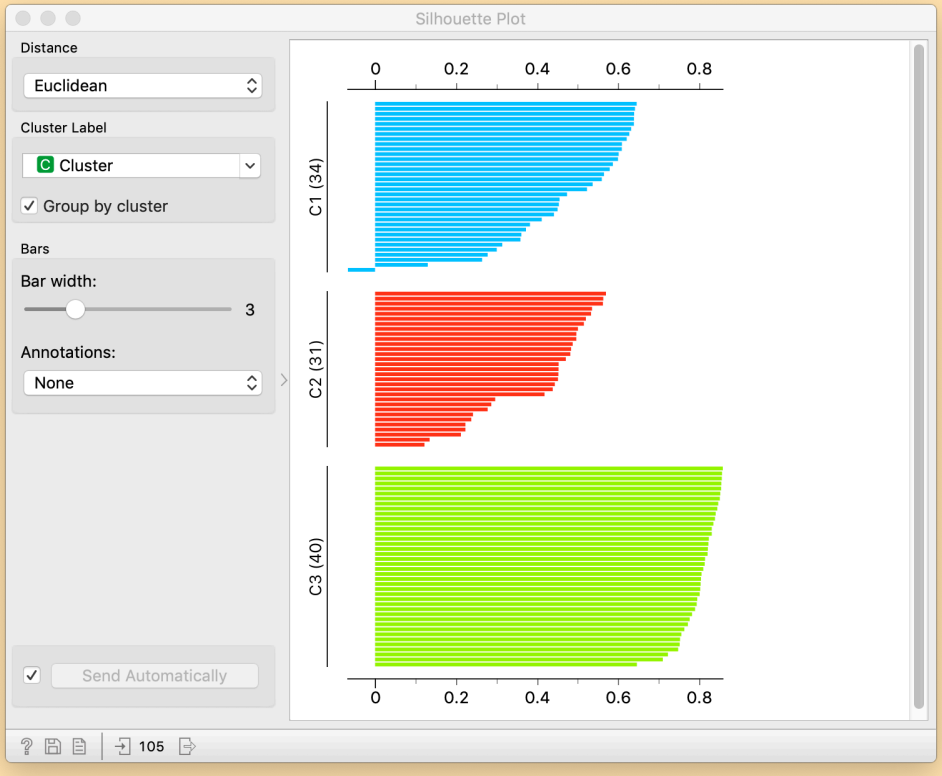
k-means algorithm (k = 3)



Adapted from ISLR, James et al., 2013 p. 389

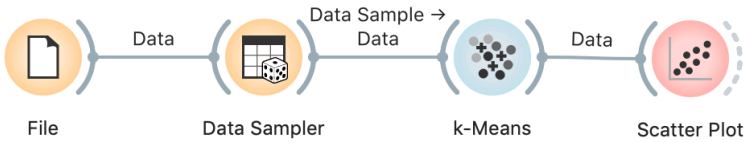
Silhouette Measure

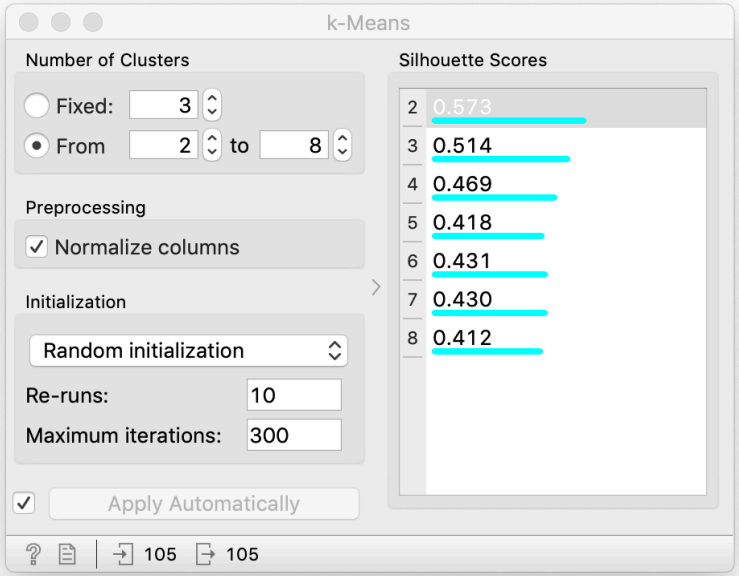
- Measures the degree of similarity (**cohesion**) of observations within a cluster **compared to** other clusters (**separation**)
- [-1 (dissimilar), 1 (similar)]
- An evaluation metric for cluster assignment
- *Goal*: high average silhouette via clustering

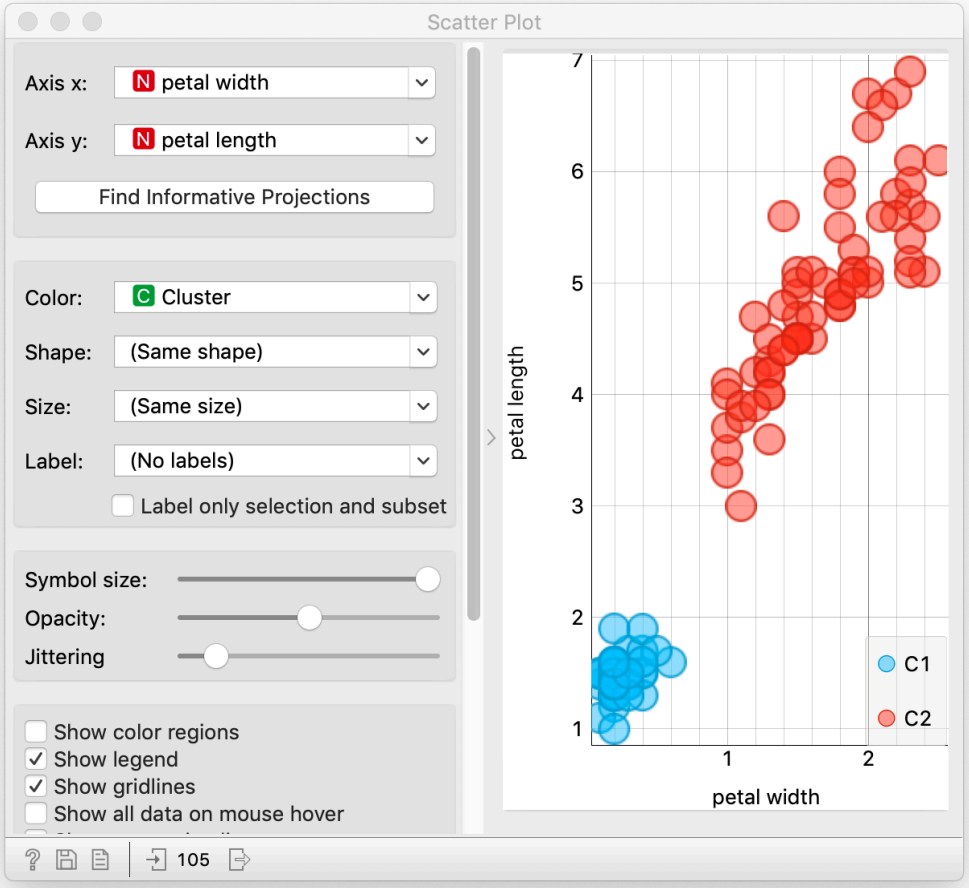


Example silhouette measures for three clusters of individuals.

K-means in Orange - Iris Example







k-means clustering

- k-means is fast – good for big data with large number of observations
- Suffers from curse of dimensionality (like KNN)
- Number of clusters must be defined by user – only heuristic methods for choosing number of clusters (Orange gives “Silhouette measure”)
- Sensitive to initial values – different starts can lead to different results – try running the algorithm repeatedly as a sensitivity analysis

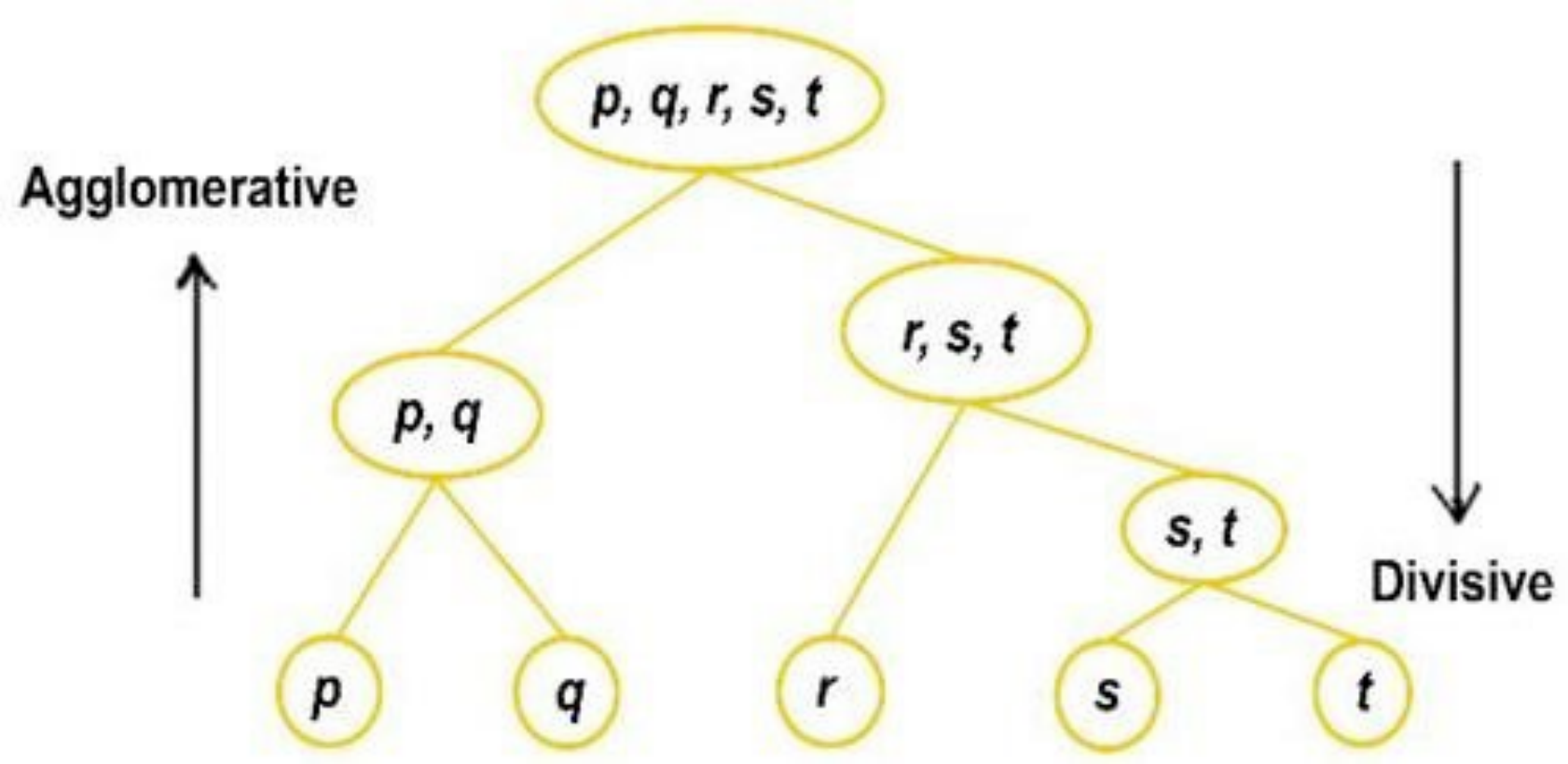
Clustering vs. classification

- Cluster analysis is NOT classification!
- In our Iris example, the clusters correspond to particular species but this comparison is provided for purely expository reasons
- Cluster analyses hypothesizes there is some latent characteristic which generates these clusters, something we might need to construct a label for after our cluster analysis (e.g. Netflix movie genres)
- So we do not wish to predict a label but rather potentially reveal a label or grouping and interpret

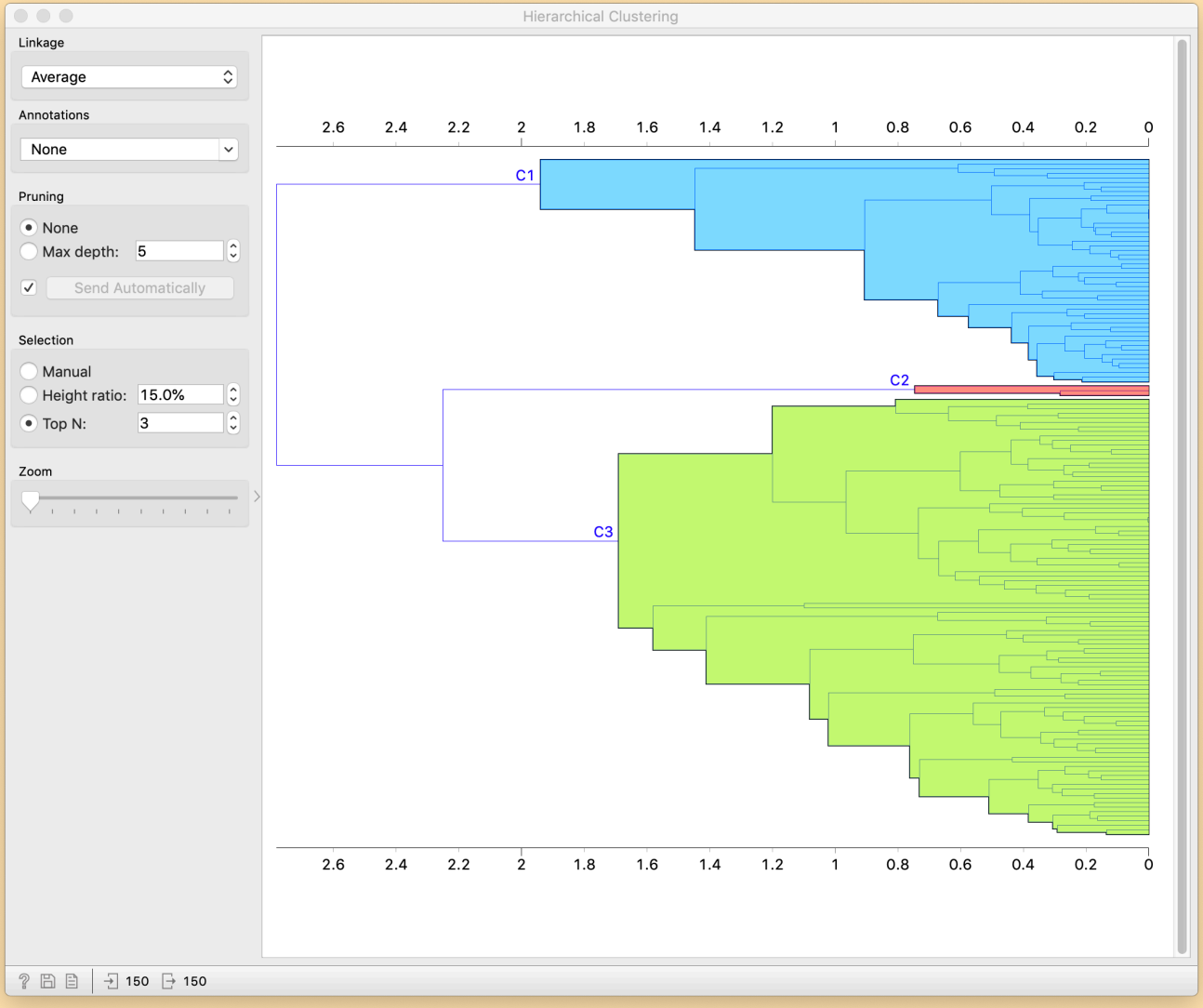
Other clustering algorithms

Hierarchical clustering method

1. Joins (**agglomerative**) or divides (**divisive**) observations
2. Hierarchically joins/divides similar/dissimilar subclusters together based on distance



Iris example – agglomerative hierarchical clustering

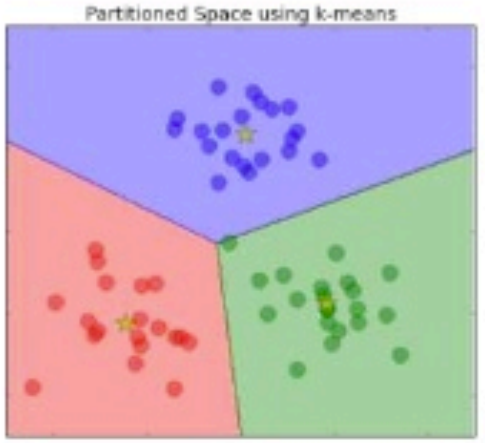


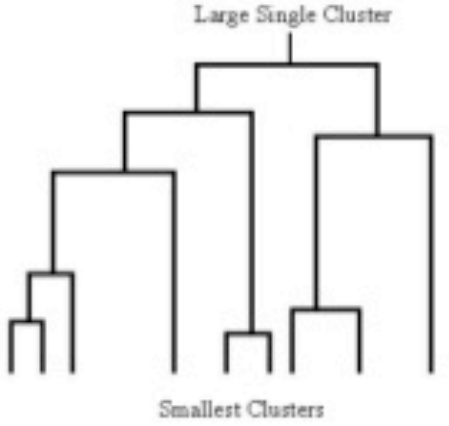
Partitional vs. Hierarchical

1. **Partitional algorithms (k-means)**
   Partition the data space
   Finds all clusters simultaneously

2. **Hierarchical algorithms**
   Generate nested cluster hierarchy
   Agglomerative (bottom-up)
   Divisive (top-down)

   Distance between clusters:
   Single-linkage, complete linkage, average-linkage





Cluster analysis (data segmentation)

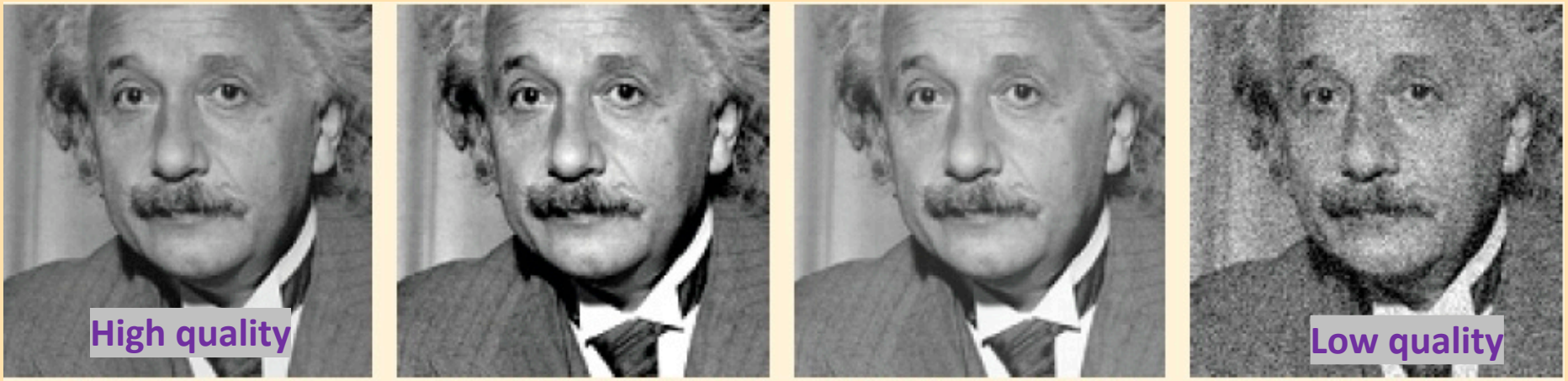
- Unsupervised learning technique that tries to group similar observations together, drive dissimilar observations apart
- Methods are differentiated by how they define similar/dissimilar
- Often need to tune the number of clusters using some evaluation metric, e.g. silhouette measure

Data Reduction Methods

Shadow Analogy

Data reduction methods

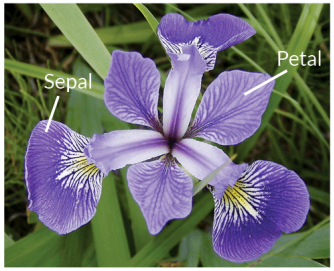
- Data reduction is the transformation of data into a corrected, simplified or ordered form
- Examples:
  - Averaging - reduce all observations to a summary number
  - Embedding - taking data from a higher dimension to a lower dimension while maintaining important characteristics among observations such as separation, distance
  - Image compression - reduce image size, maintain attributes, see below



Why data reduction?

- Interpretation:
  - Wish to **visualize** high dimensional data in a 2D/3D space
- Memory/storage:
  - Cannot store raw data, need to reduce to store/analyze
  - E.g. summarize step counts by minute rather than second
- Pre-processing:
  - Reduce dimension of data to perform an analysis
  - E.g. for k-nearest neighbors to avoid the “curse of dimensionality”

Principal components analysis (PCA)



Iris Versicolor

Iris Setosa

Iris Virginica

Data Table

Variables

- [x] Show variable labels (if present)
- [ ] Visualize numeric values
- [x] Color by instance classes

Selection

- [x] Select full rows

Restore Original Order

[x] Send Automatically

	iris	sepal length	sepal width	petal length	petal width
1	Iris-setosa	5.1	3.5	1.4	0.2
2	Iris-setosa	4.9	3.0	1.4	0.2
3	Iris-setosa	4.7	3.2	1.3	0.2
4	Iris-setosa	4.6	3.1	1.5	0.2
5	Iris-setosa	5.0	3.6	1.4	0.2
6	Iris-setosa	5.4	3.9	1.7	0.4
7	Iris-setosa	4.6	3.4	1.4	0.3
8	Iris-setosa	5.0	3.4	1.5	0.2
9	Iris-setosa	4.4	2.9	1.4	0.2
10	Iris-setosa	4.9	3.1	1.5	0.1
11	Iris-setosa	5.4	3.7	1.5	0.2
12	Iris-setosa	4.8	3.4	1.6	0.2
13	Iris-setosa	4.8	3.0	1.4	0.1
14	Iris-setosa	4.3	3.0	1.1	0.1
15	Iris-setosa	5.8	4.0	1.2	0.2
16	Iris-setosa	5.7	4.4	1.5	0.4
17	Iris-setosa	5.4	3.9	1.3	0.4
18	Iris-setosa	5.1	3.5	1.4	0.3
19	Iris-setosa	5.7	3.8	1.7	0.3
20	Iris-setosa	5.1	3.8	1.5	0.3

150

Principal components analysis: Iris example

- Each flower has four attributes (aside from species), e.g. Flower 1

Sepal Length	5.1
Sepal Width	3.5
Petal Length	1.4
Petal Width	0.2

- Can we come up with a "recipe" for each flower from some "ingredients" composed of our flower attributes?

Ingredients

Flower 1		sepal_length	sepal_width	petal_length	petal_width
Recipe (amount of ingredient)	**5.1** *	**1**	**0**	**0**	**0**
	+ **3.5** *	**0**	**1**	**0**	**0**
	+ **1.4** *	**0**	**0**	**1**	**0**
	+ **0.2** *	**0**	**0**	**0**	**1**
		5.1	3.5	1.4	0.2

Principal components analysis: Iris example

- Our ingredients only consider one attribute at a time. What if our ingredients could combine attributes that often go together? Maybe we could approximate the same "recipe" with fewer ingredients?

- Original recipe:
  (4 ingredients)

Ingredients

Flower 1		sepal_length	sepal_width	petal_length	petal_width
Recipe (amount of ingredient)	5.1 *	1	0	0	0
	+ 3.5 *	0	1	0	0
	+ 1.4 *	0	0	1	0
	+ 0.2 *	0	0	0	1
		5.1	3.5	1.4	0.2

- New recipe:
  (2 ingredients)

Ingredients

Flower 1		sepal_length	sepal_width	petal_length	petal_width
Recipe (amount of Ingredient)	5.9*	.75	.38	.51	.17
+	2.3*	-.29	-.54	.71	.35
		5.09	3.5	1.4	0.2

Wow! Almost the "same" flower with fewer ingredients!!!

Principal components analysis (PCA)

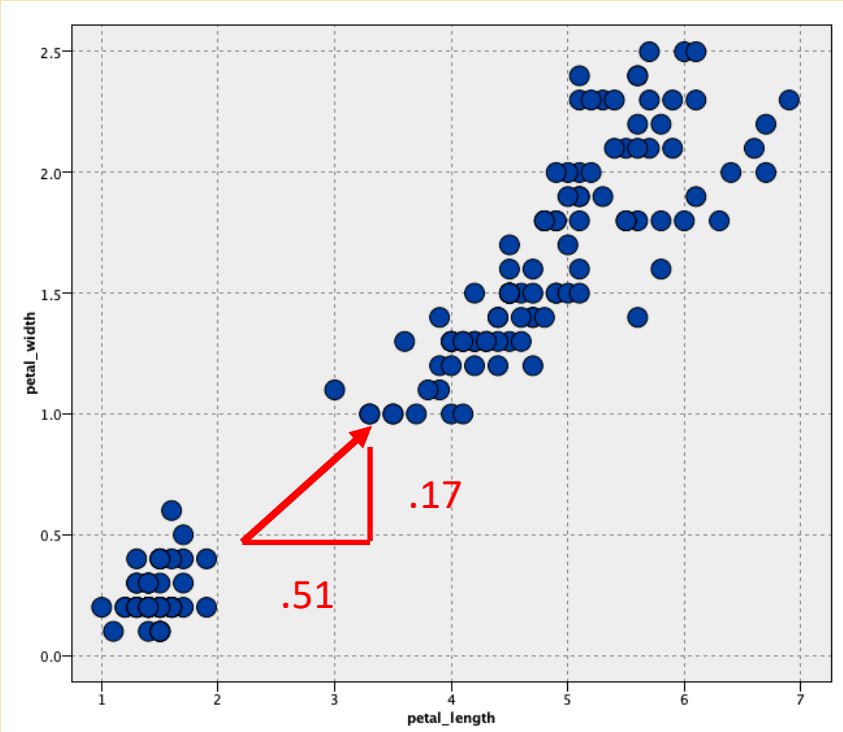
- Principal components analysis finds an optimal set of "ingredients" i.e. principal components (PC) which we can use to represent our data
- Principal components analysis also tells us the amount of each PC we need to form each observation (PC scores)
- PC are optimal in a specific way
  - Each PC captures a certain amount of variation within the data
  - The PC can be ordered in terms of importance (i.e. amount of variation explained)
  - PC are orthogonal, meaning the score for one PC vs another PC within a subject are uncorrelated
- Generally prior to PCA, need to standardize variables
- **Data reduction**: we can use a few PC to to represent our data and still have essentially the same content! Then all you need to do is store the PC and the PC scores!! Can also predict or cluster based on PC scores!!!

Principal components analysis

- How does this new mathematical vocabulary fit into “recipe” framework?

		Principal Components			
Flower 1		sepal_length	sepal_width	petal_length	petal_width
Principal Component Scores	5.9*	.75	.38	.51	.17
+	2.3*	-.29	-.54	.71	.35
		5.09	3.5	1.4	0.2

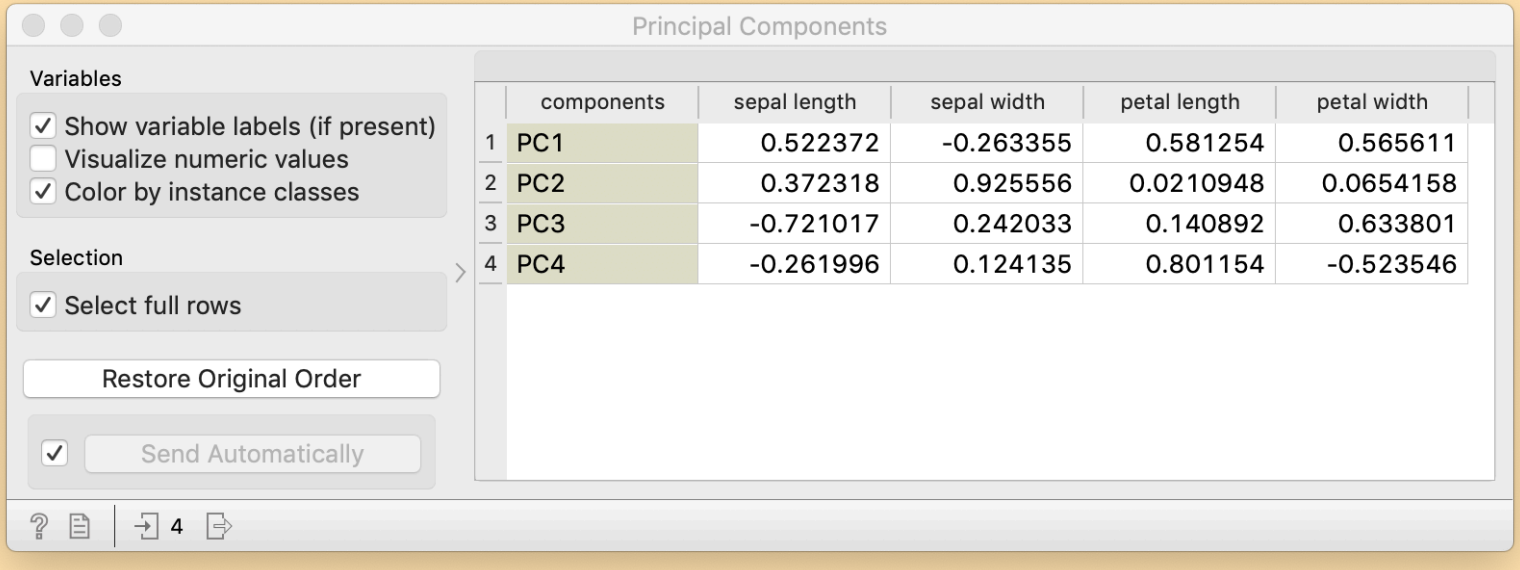
- Principal components identify the directions of “maximal spread (variation)”



Principal components analysis – Scores

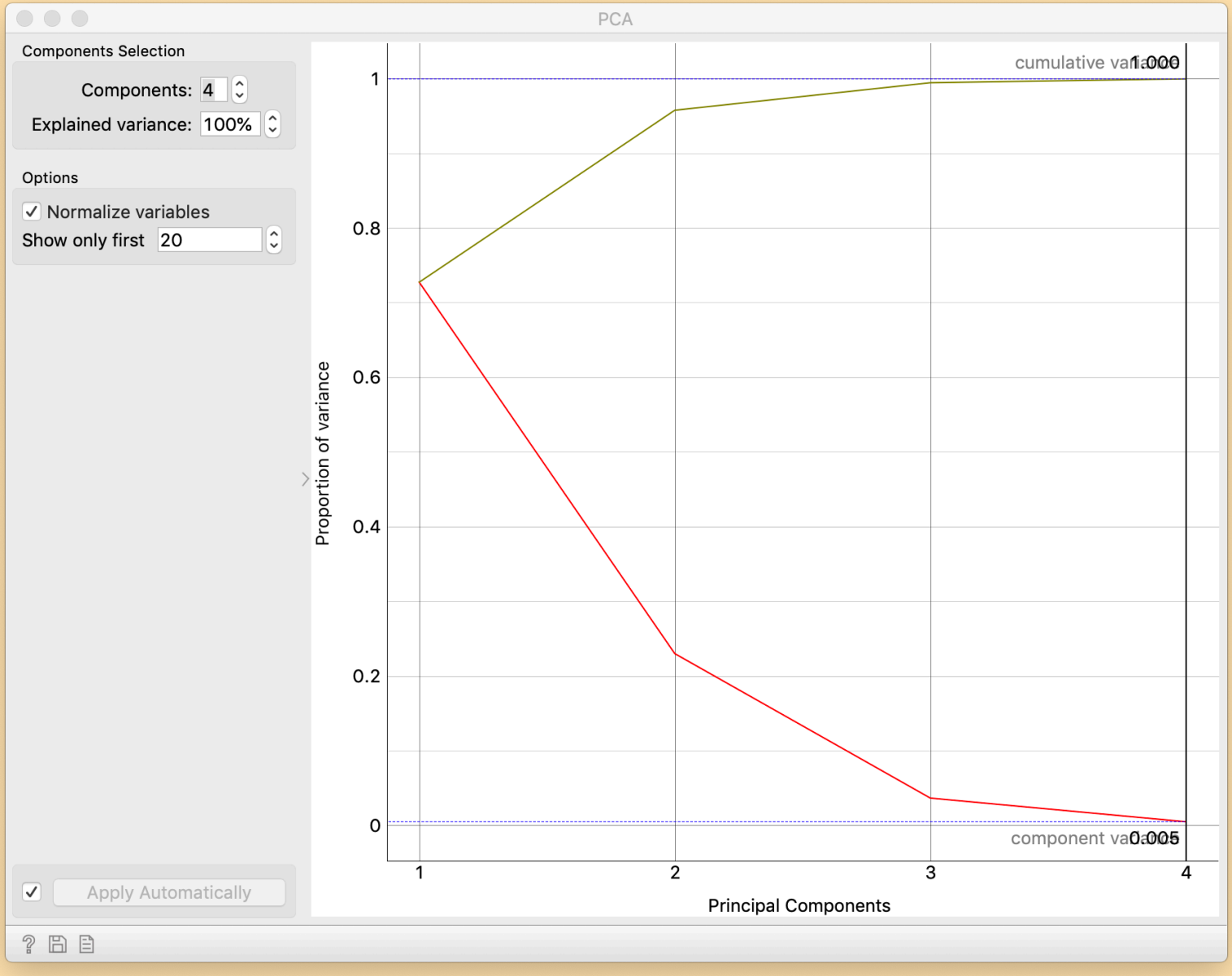
- We can even examine and compare the **principal component scores** among subjects to reveal informative patterns!

Principal Components Analysis – Principal Components (ingredients)



	components	sepal length	sepal width	petal length	petal width
1	PC1	0.522372	-0.263355	0.581254	0.565611
2	PC2	0.372318	0.925556	0.0210948	0.0654158
3	PC3	-0.721017	0.242033	0.140892	0.633801
4	PC4	-0.261996	0.124135	0.801154	-0.523546

Principal components analysis – Variance explained

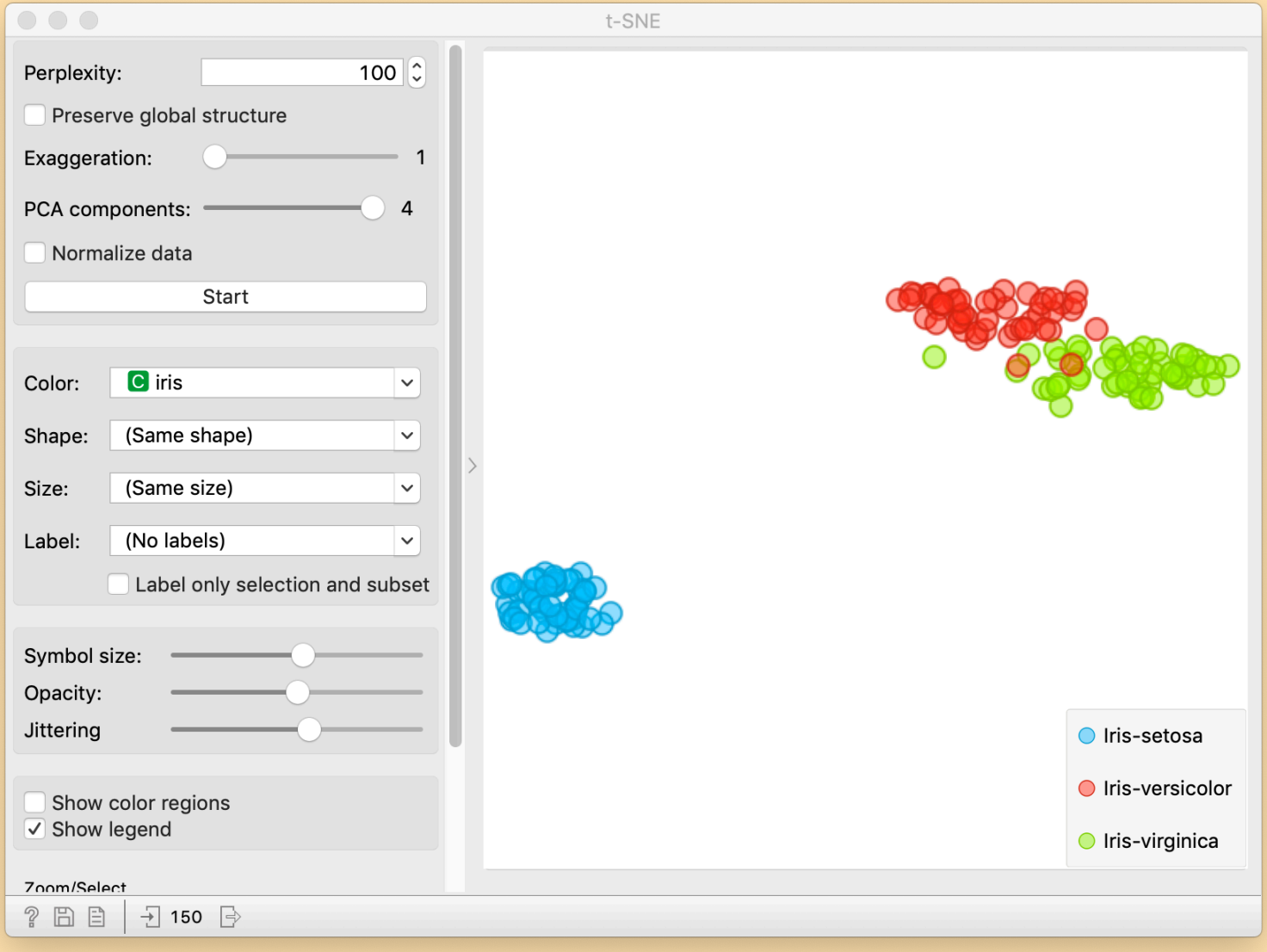


Other dimension reduction methods

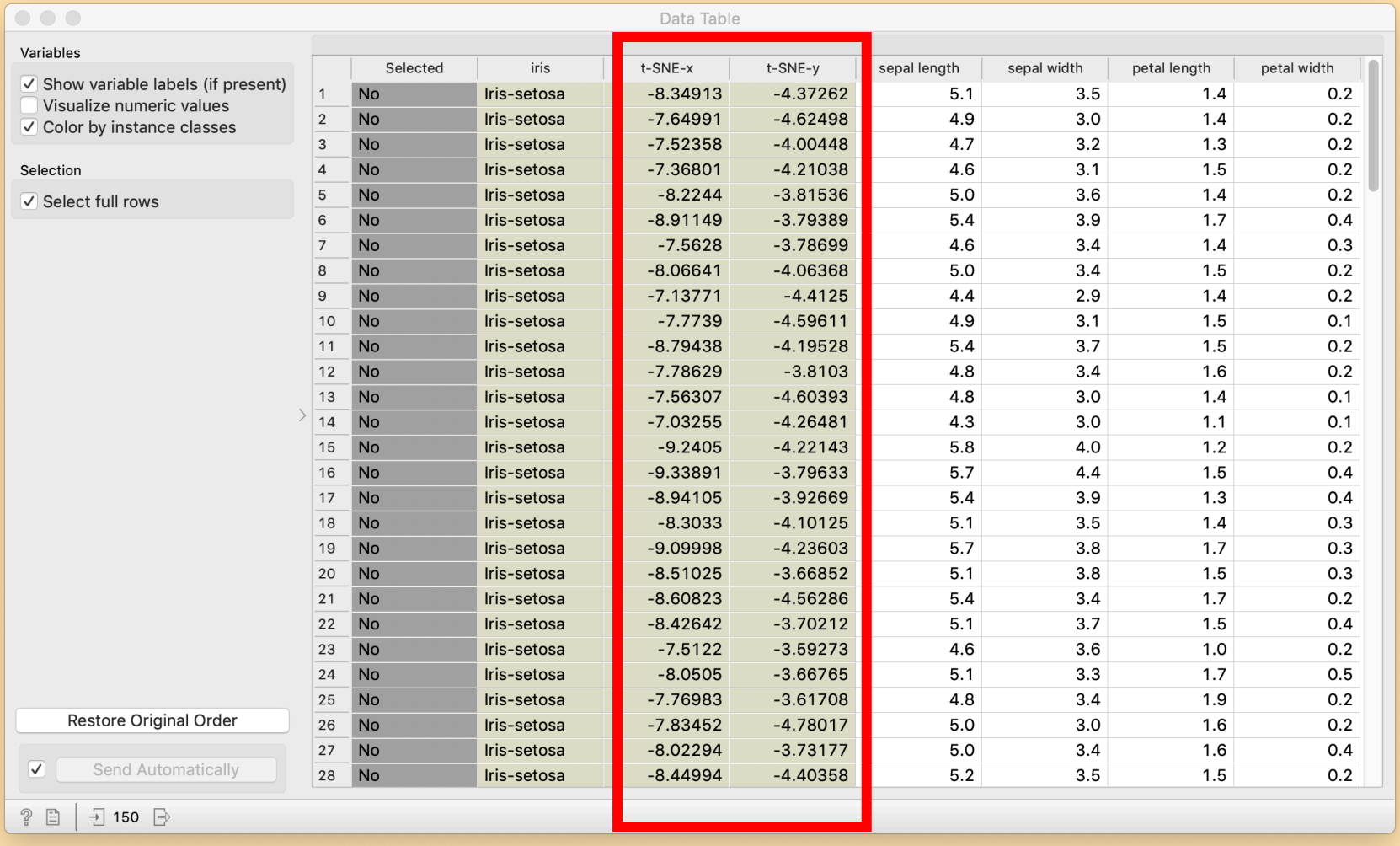
t-Distributed Stochastic Neighbor Embedding (t-SNE)

- Method for visualizing high-dimensional data in 2D or 3D space WITHOUT supervision or a target
- Maps each high-dimensional observation into a 2D/3D space such that “nearby” points in high dimensions are found near each other with high probability in the lower dimensional space (sensitive to initialization)
- Popular for identifying potential clusters through low-dimensional embedding e.g. Iris data!
- Embedding method, does not assign subjects to clusters – clustering would be a second step

t-SNE for Iris Data



t-SNE for Iris Data: coordinates



	Selected	iris	t-SNE-x	t-SNE-y	sepal length	sepal width	petal length	petal width
1	No	Iris-setosa	-8.34913	-4.37262	5.1	3.5	1.4	0.2
2	No	Iris-setosa	-7.64991	-4.62498	4.9	3.0	1.4	0.2
3	No	Iris-setosa	-7.52358	-4.00448	4.7	3.2	1.3	0.2
4	No	Iris-setosa	-7.36801	-4.21038	4.6	3.1	1.5	0.2
5	No	Iris-setosa	-8.2244	-3.81536	5.0	3.6	1.4	0.2
6	No	Iris-setosa	-8.91149	-3.79389	5.4	3.9	1.7	0.4
7	No	Iris-setosa	-7.5628	-3.78699	4.6	3.4	1.4	0.3
8	No	Iris-setosa	-8.06641	-4.06368	5.0	3.4	1.5	0.2
9	No	Iris-setosa	-7.13771	-4.4125	4.4	2.9	1.4	0.2
10	No	Iris-setosa	-7.7739	-4.59611	4.9	3.1	1.5	0.1
11	No	Iris-setosa	-8.79438	-4.19528	5.4	3.7	1.5	0.2
12	No	Iris-setosa	-7.78629	-3.8103	4.8	3.4	1.6	0.2
13	No	Iris-setosa	-7.56307	-4.60393	4.8	3.0	1.4	0.1
14	No	Iris-setosa	-7.03255	-4.26481	4.3	3.0	1.1	0.1
15	No	Iris-setosa	-9.2405	-4.22143	5.8	4.0	1.2	0.2
16	No	Iris-setosa	-9.33891	-3.79633	5.7	4.4	1.5	0.4
17	No	Iris-setosa	-8.94105	-3.92669	5.4	3.9	1.3	0.4
18	No	Iris-setosa	-8.3033	-4.10125	5.1	3.5	1.4	0.3
19	No	Iris-setosa	-9.09998	-4.23603	5.7	3.8	1.7	0.3
20	No	Iris-setosa	-8.51025	-3.66852	5.1	3.8	1.5	0.3
21	No	Iris-setosa	-8.60823	-4.56286	5.4	3.4	1.7	0.2
22	No	Iris-setosa	-8.42642	-3.70212	5.1	3.7	1.5	0.4
23	No	Iris-setosa	-7.5122	-3.59273	4.6	3.6	1.0	0.2
24	No	Iris-setosa	-8.0505	-3.66765	5.1	3.3	1.7	0.5
25	No	Iris-setosa	-7.76983	-3.61708	4.8	3.4	1.9	0.2
26	No	Iris-setosa	-7.83452	-4.78017	5.0	3.0	1.6	0.2
27	No	Iris-setosa	-8.02294	-3.73177	5.0	3.4	1.6	0.4
28	No	Iris-setosa	-8.44994	-4.40358	5.2	3.5	1.5	0.2

Dimension Reduction

- Reduces your high-dimensional data to a lower dimensional space WITHOUT regard to a target
- You can use representations in lower dimensional space for further machine learning
  - Less noise
  - Less overfitting
- Can produce computational gains in both storage and training
- Loose interpretation in reduced dimensional space

Clustering vs Dimension Reduction

Clustering

- Discover latent classes or structure in data
- Maximizes within cluster similarity
- Minimize similarity between clusters
- Cluster labels can be assigned

Dimension Reduction

- Reduces feature dimension to lower dimension
- Can capture leading directions of variation
- Does not assign a label
- Subsequent machine learning can be used on reduced dimension data

Review of this lecture

- Clustering
  - k-means clustering
  - Hierarchical clustering
- Data reduction
  - Principal components analysis (PCA)
  - t-Distributed Stochastic Neighbor Embedding (t-SNE)