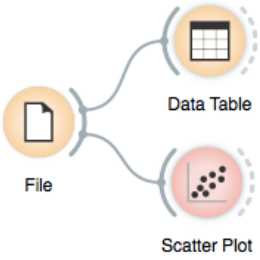
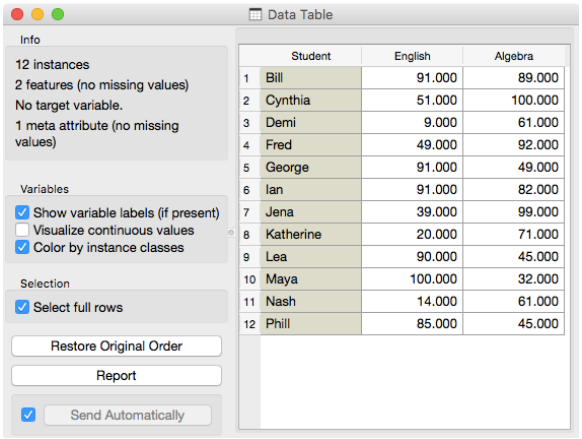
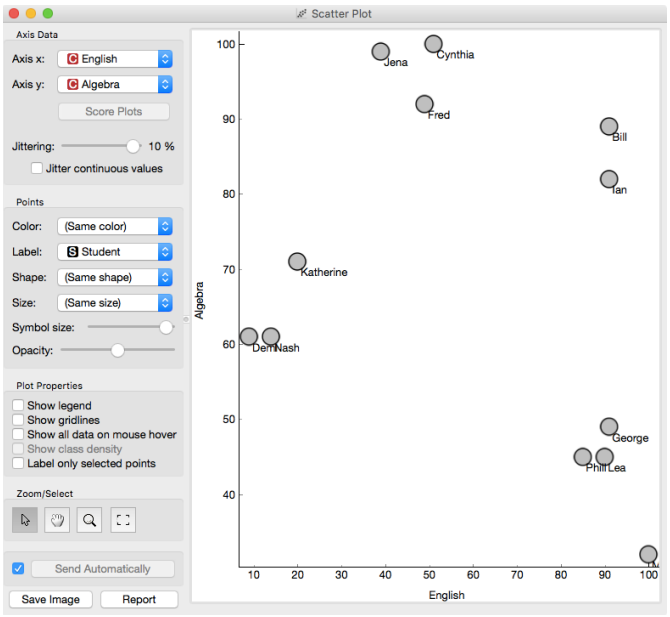
Lesson 23: Hierarchical Clustering

In the class, we will introduce clustering using a simple data set on students and their grades in English and Algebra. Load the data set from http://bit.ly/2cawh6M.





Say that we are interested in finding clusters in the data. That is, we would like to identify groups of data instances that are close together, similar to each other. Consider a simple, two-featured data set (see the side note) and plot it in the Scatter Plot. How many clusters do we have? What defines a cluster? Which data instances should belong to the same cluster? How does the clustering algorithm actually work?

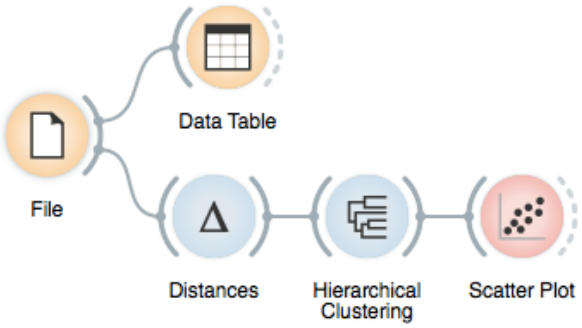


First, we need to define what we mean by "similar". We will assume that all our data instances are described (profiled) with continuous features; they are indeed, grades are given on continuous scale. One simple measure of similarity is the Euclidean distance. So, we would like to group data instances with small Euclidean distances.

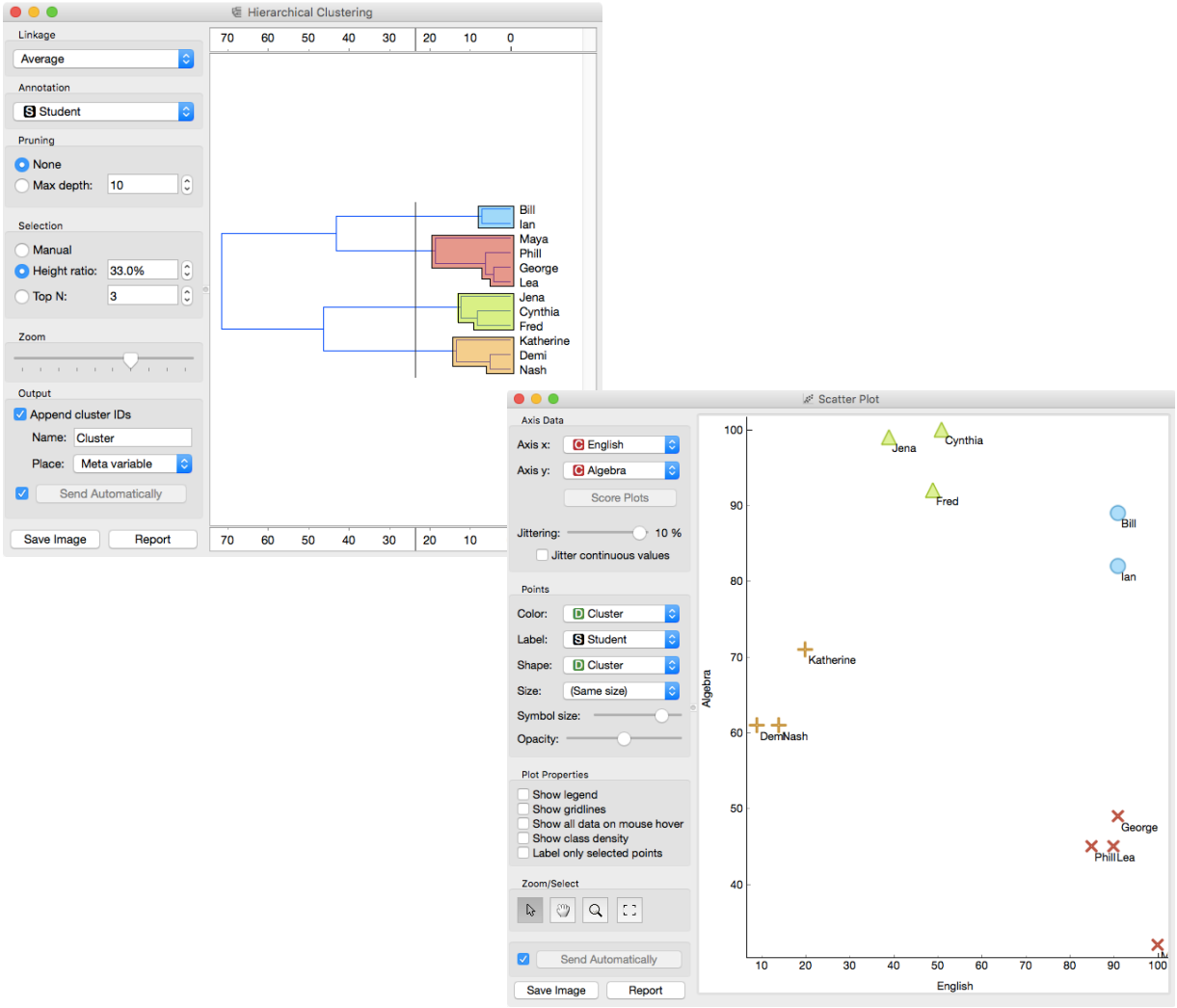
Next, we need to define a clustering algorithm. Say that we start with each data instance being its own cluster, and then, at each step we join the clusters that are closest together. We estimate the distance between the clusters with, say, the average distance between all their pairs of data points. The algorithm is called hierarchical clustering.

There are different ways to measure the similarity between clusters. The estimate we have described is called average linkage. We could also estimate the distance through the two closest points in each clusters (single linkage), or through the two points that are furthest away (complete linkage).

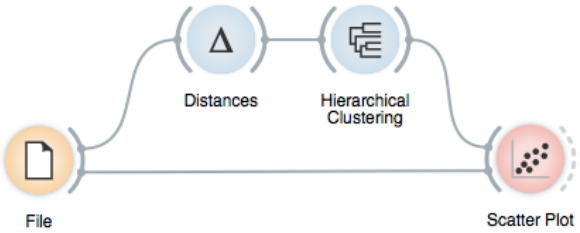
One possible way to observe the results of clustering on our small data set with grades is through the following workflow:



Couldn't be simpler. Load the data, measure distances, use them in hierarchical clustering, and visualize the results in the scatterplot. Hierarchical clustering widget allows us to cut hierarchy at a certain distance score and outputs the corresponding clusters:

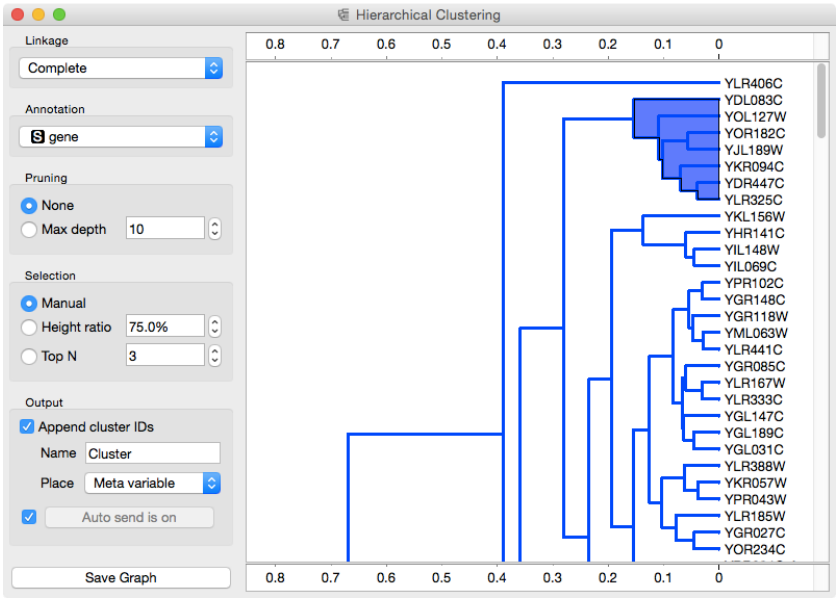


Now for a larger data set. Take brown-selected (from documentation data sets) and use the following workflow:



The Hierarchical Clustering widget visualizes the clustering in a dendrogram. One can click on a branch of a dendrogram to select a subset of the data instances. By combining it with the Scatter Plot widget, we get a great tool for exploring the effects of clustering. Try it with an appropriate pair of features to visualize (use Rank projections).

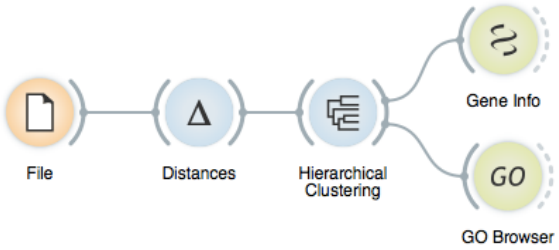
You can also try this on the Iris data.

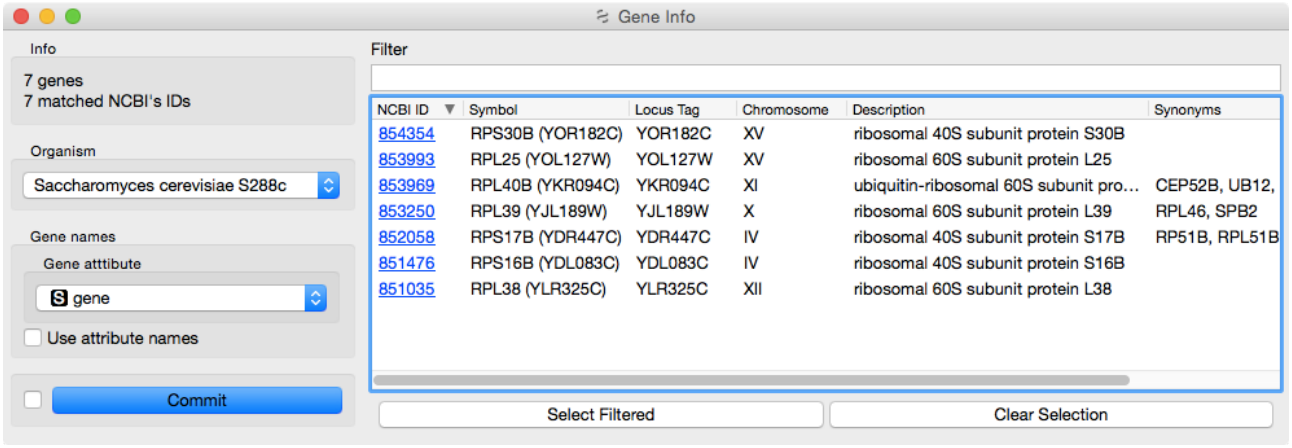


When working with gene expression data, use the Spearman correlation as a measure of distance instead of the Euclidean distance. It almost always works better.

In the brown-selected data set, the data instances are genes, and a combination with several widgets from the bioinformatics add-on becomes really appealing. We offer the following workflow more to whet you appetite, as we will discuss gene set enrichment a bit later in the course.

Hierarchical clustering works fast for smaller data sets (define smaller!). But for bigger ones it fails. Simply, cannot be used. Why?

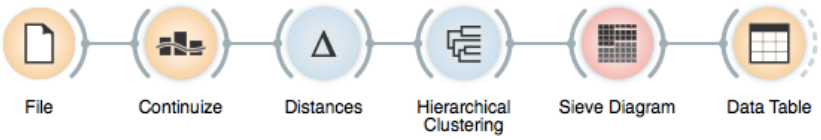


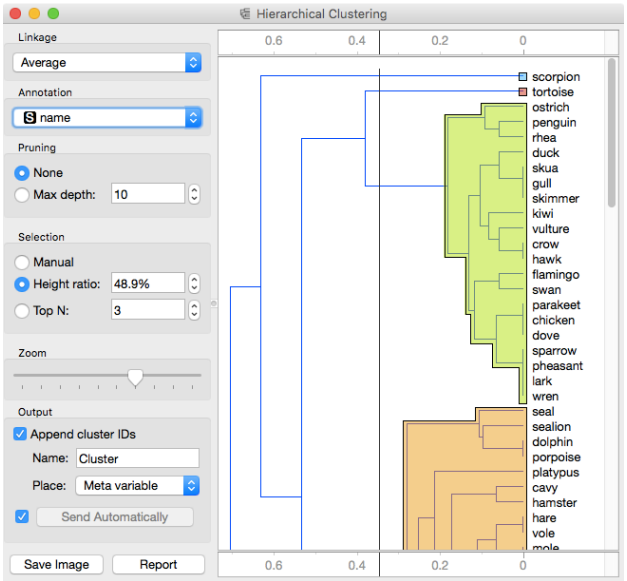


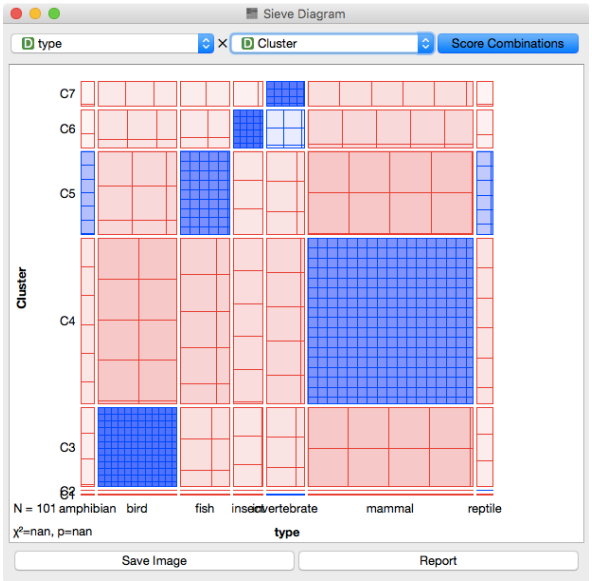
Lesson 24: Animal Kingdom

Both of your lecturers spent substantial part of their youth admiring a particular Croatian chocolate called Animal Kingdom. Each chocolate bar came with a card — a drawing of some (random) animal, and the associated album made us eat a lot of chocolate. Then our kids came, and the story repeated. Some things stay forever. Funny stuff was we never understood the order in which the cards were laid out in the album. We later learned about taxonomy, but being more inclined to engineering we never mastered learning it by heart in our biology classes. Luckily, there's data mining and the idea that taxonomy simply stems from measuring the distance between species.

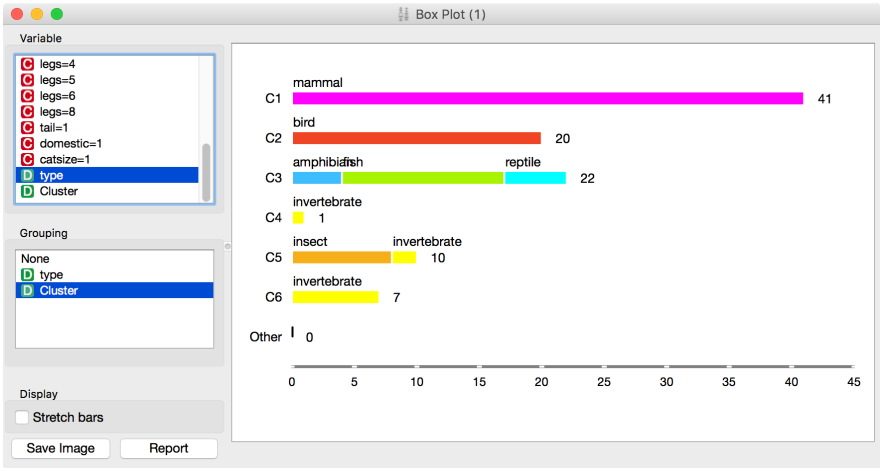
Here we use zoo data (zoo.tab from documentation data sets) with attributes that report on various features of animals (has hair, has feathers, lays eggs). Features are discrete, so we need to continuize them. Then measure the distance (say, Euclidean) and compute hierarchical clustering. Animals in this data set are annotated with type (mammal, insect, bird, and so on). It would be cool to know if the clustering actually identified groups of animals that belong to the same type. We can do this through marking the clusters in Hierarchical Clustering widget, and then observing the results in the Sieve Diagram.







Checking this in the Box plot is even cooler.



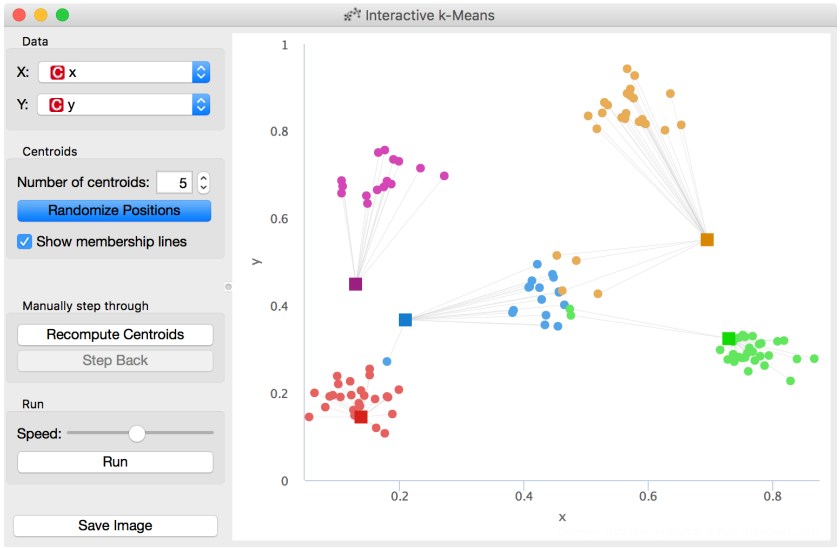
Lesson 25: k-Means Clustering

Hierarchical clustering is not suitable for larger data sets due to the prohibitive size of the distance matrix: with 30 thousand objects, the distance matrix already has almost one billion elements. An alternative approach that avoids using the distance matrix is k-means clustering.

K-means clustering randomly selects k centers (with k specified in advance). Then it alternates between two steps. In one step, it assigns each point to its closest center, thus forming k clusters. In the other, it recomputes the centers of the clusters. Repeating these two steps typically convergences quite fast; even for the big data sets with millions of data points it usually takes just a couple of tens or hundreds iterations.

Orange's add-on Educational provides a widget Interactive k-means, which illustrates the algorithm.

Use the Paint widget to paint some data - maybe five groups of points. Feed it to Interactive k-means and set the number of centroids to 5. You may get something like this.



Try rerunning the clustering from new random positions and observe how the centers conquer the territory. Exciting, isn't it?

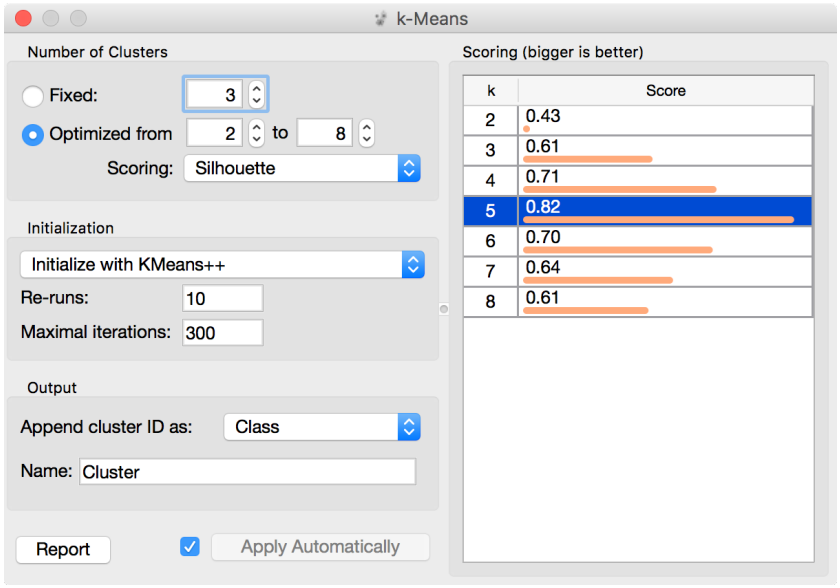
Keep pressing Recompute Centroids and Reassign Membership until it stops changes. With this simple, two-dimensional data it will take just a few iterations; with more points and features, it can take longer, but the principle is the same.

How do we set the initial number of clusters? That's simple: we choose the number that gives the optimal clustering.

Well then, how do we define the optimal clustering? This one is a bit harder. We want small distances between points in the same cluster and large distances between points from different clusters. Pick one point, and let A be its average distance to the data points in the same cluster and let B represent the average distance to the points from the closest other cluster. (The closest cluster? Just compute B for all other clusters and take the lowest value.) The value (B - A) / max(A, B) is called silhouette; the higher the silhouette, the better the point fits into its cluster. The average silhouette across all points is the silhouette of the clustering. The higher the silhouette, the better the clustering.

Now that we can assess the quality of clustering, we can run k-means with different values of parameter k (number of clusters) and select k which gives the largest silhouette.

For this, we abandon our educational toy and connect Paint to the widget k-Means. We tell it to find the optimal number of clusters between 2 and 8, as scored by the Silhouette.



Works like charm.

Except that it often doesn't. First, the result of k-means clustering depends on the initial selection of centers. With unfortunate

selection, it may get stuck in a local optimum. We solve this by re-running the clustering multiple times from random positions and using the best result. Second, the silhouette sometimes fails to correctly evaluate the clustering. Nobody's perfect.

Time to experiment. Connect the Scatter Plot to k-Means. Change the number of clusters. See if the clusters make sense. Could you paint the data where k-Means fails? Or where it really works well?