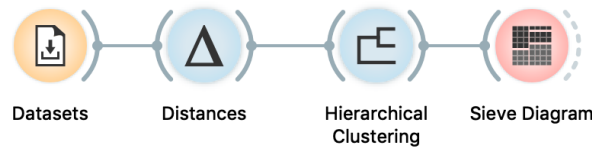


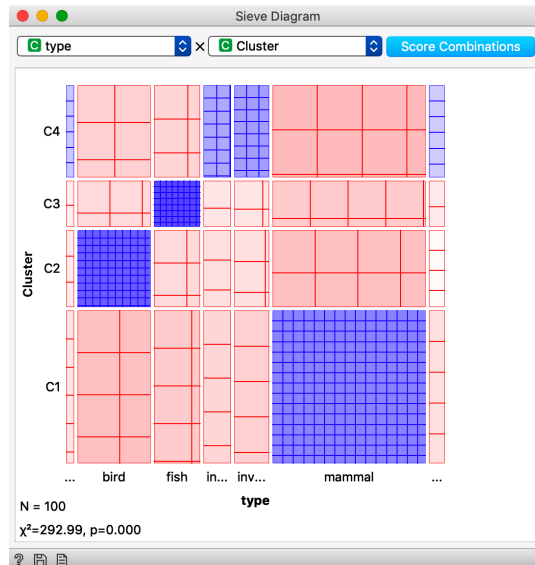
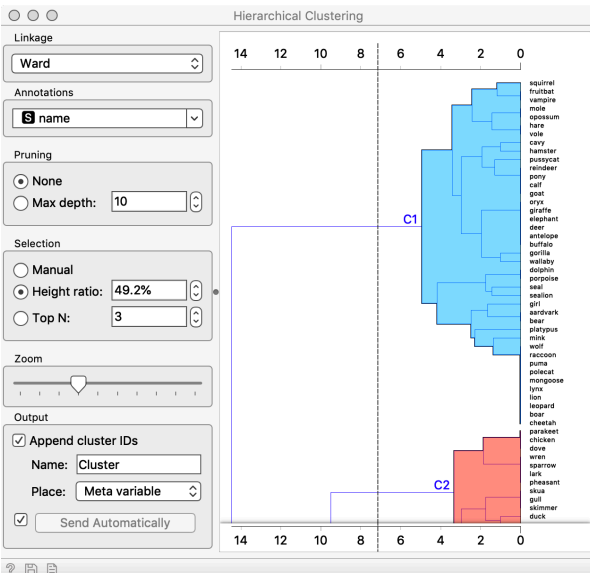
Lesson 6: Animal Kingdom



Your lecturer spent a substantial part of his youth admiring particular Croatian chocolate called the 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 I never understood the order in which the cards were laid out in the album. Later, during gymnasium's biology classes, I learned about taxonomy but never mastered it (I am blaming this one on a teacher). Luckily, there's data mining and the idea that taxonomy stems from measuring the distance between species.



Here we use zoo data with attributes that report on various features of animals (has hair, has feathers, lays eggs). We measure the distance and compute the clustering. Animals in this data set are annotated with the type (mammal, insect, bird, and so on). It would be cool to know if the clustering re-discovered these groups of animals. We can do this by marking the clusters in Hierarchical Clustering widget and then observing the results in the Sieve Diagram.

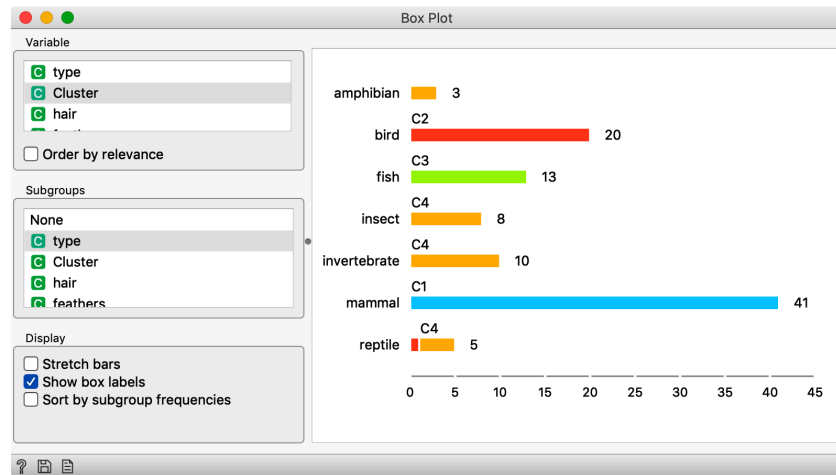


Looks great. Birds, say, are in cluster C2. Cluster C1 consists of mammals. And so forth.

Checking this in the Box plot is even cooler. We can get a distribution of animal types in each cluster:



Or we can turn it around and see how different types of animals are spread across clusters.



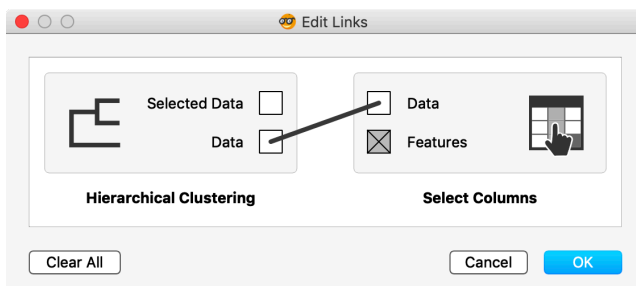
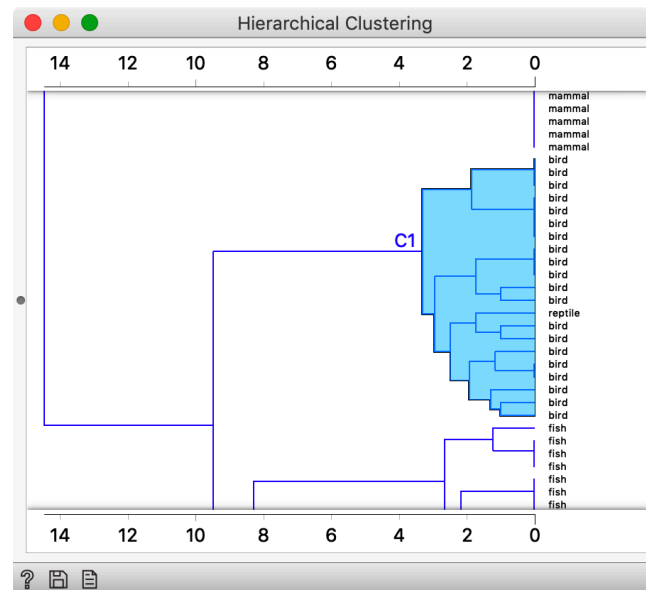
There is one reptile that, for some reason, got in the red cluster (C2). Which animal is that? Why is it clustered with birds? Click on the clusters in the box plot and discover their constituents.

Lesson 7: Cluster Interpretation

Once we have inferred the clusters, we would like to know what are the distinguishing features. We would, for instance, like to choose a branch (group) from the dendrogram, and then ask what makes the selected animals different from all the other ones. We will use the following workflow.



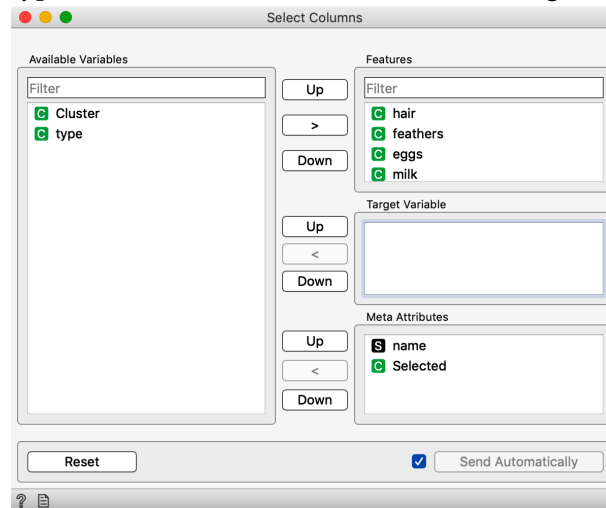
The workflow loads the zoo data, measures the distances between pairs of animals and performs hierarchical clustering. Nothing new here. In hierarchical clustering, we select a branch, say, a group of birds.



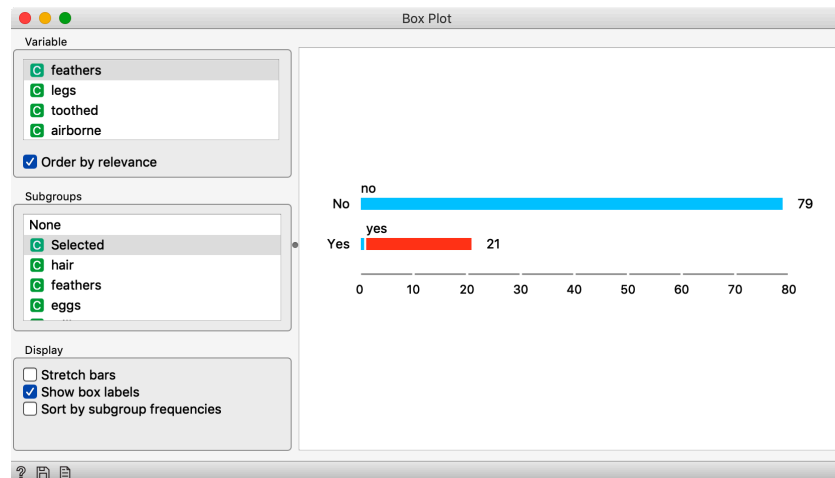
We would now like to pass the entire data set from the hierarchical clustering widget, with, hopefully, a label that denotes if the animal was in the chosen group or not. We have to rewire the connection between Hierarchical Clustering and Select Columns widget accordingly. Double click the connection, and change the wiring to match the one shown on the left.

At this stage, it would be interesting to see what is the output of Hierarchical Clustering, or better, what kind of data is transmitted to its output Data channel. Check this out using the Data Table.

From further analysis, we will get rid of the columns Cluster and type. We do this in the Select Columns widget.



We will feed the output of Select Column to the Box Plot widget. We will define the subgroups in this widget by Selected, and thus get two bars. The final trick we will use is checking Order by relevance. The features that best correlate with the subgroups, that is, with our selection, will appear on the top.



Presence of feathers best correlates with our cluster of birds. Following is a feature that reports on the number of legs. And the presence of teeth. And so on. Try choosing any other branch from the dendrogram to see how the order of features changes with our selection.

Lesson 8: k-Means Clustering

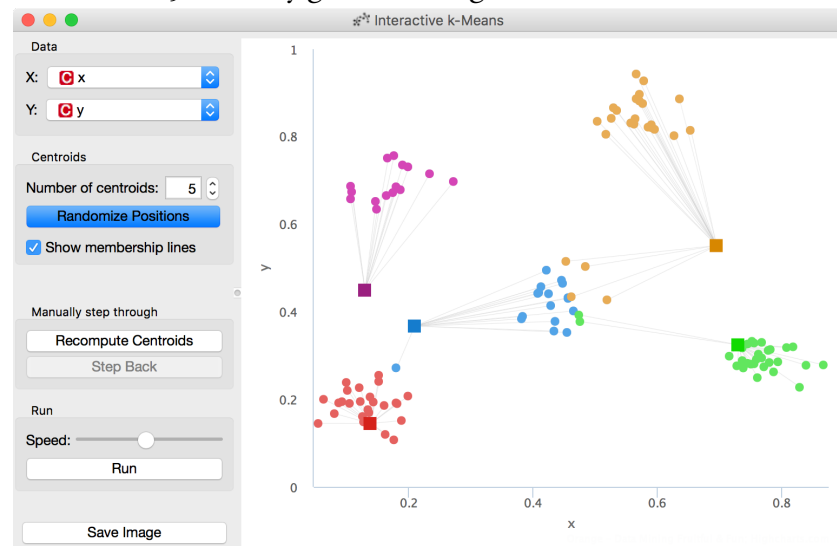
Hierarchical clustering is not suitable for larger data sets due to the prohibitive size of the distance matrix: with 30 thousand objects, the distance matrix already has almost one billion elements. An alternative approach that avoids using the distance matrix is k-means clustering.

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 converges 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 an Interactive k-Means widget, 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?



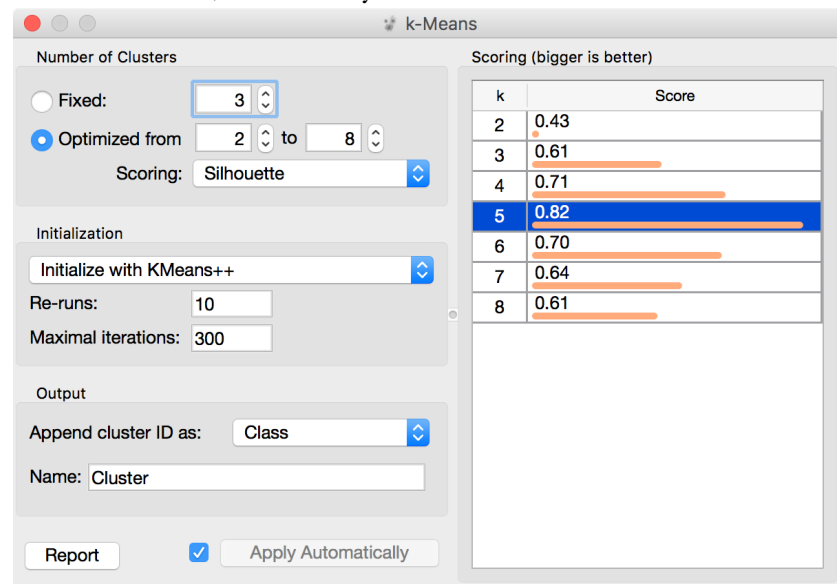
Keep pressing Recompute Centroids and Reassign Membership until it stops changes. With this simple, two-dimensional data it will take just a few iterations; with more points and features, it can take longer, but the principle is the same.

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.



k	Score
2	0.43
3	0.61
4	0.71
5	0.82
6	0.70
7	0.64
8	0.61

Works like charm.

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?

Except that it often doesn't. First, the result of k -means clustering depends on the initial selection of centers. With adverse 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 evaluate the clustering correctly. Nobody's perfect.

Lesson 9: Finding Clusters When There Are None

We saw how clustering could discover the subgroups in the data. The flip side of this is that algorithms like k-means will always find them even when they do not exist.

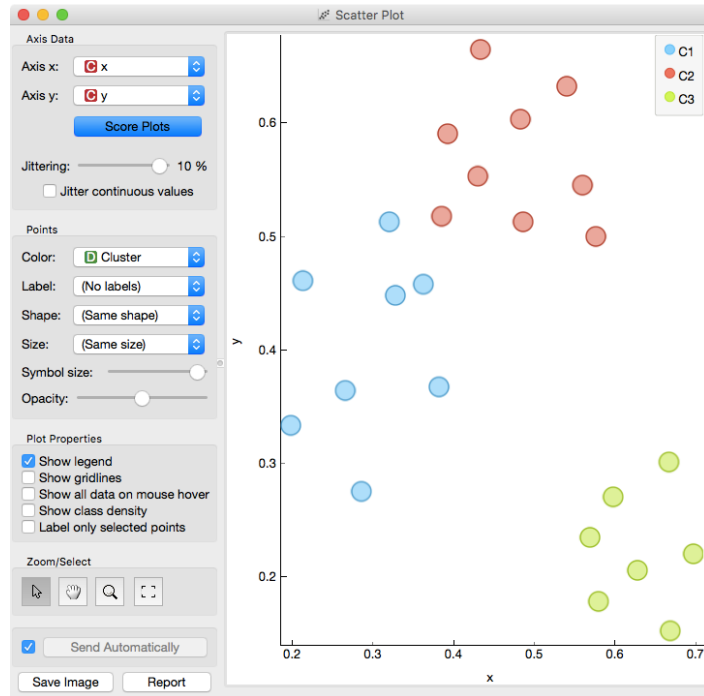
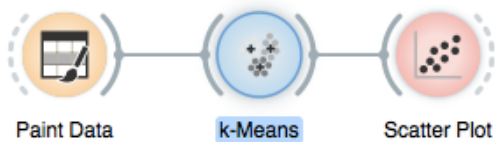


Playing with Paint Data and k-Means can be quite fun. Try painting the data where there are clusters, but k-means does not find them. Or finds the wrong ones. What kind of groups is easy to find with k-means? Are these the kind of clusters we would find in real data sets?

It is difficult to verify whether the clusters we found are "real." Data mining methods like clustering can serve only as hints that can help to form new hypotheses, which must make biological sense and be verified on new, independent data. We cannot make conclusions based only on "discovering" clusters.

Lesson 10: Silhouettes

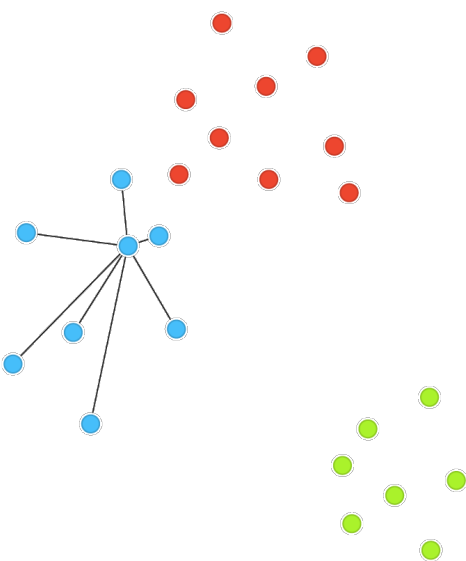
Don't get confused: we paint data and/or visualize it with Scatter plots, which show only two features. This is just for an illustration! Most data sets contain many features and methods like k-Means clustering take into account all features, not just two.

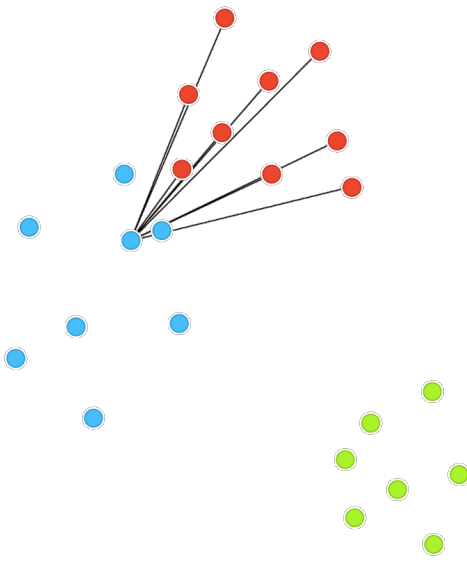


The data points in the green cluster are well separated from those in the other two. Not so for the blue and red points, where several points are on the border between the clusters. We would like to quantify the degree of how well a data point belongs to the cluster to which it is assigned.

We will invent a scoring measure for this and we will call it a *silhouette* (because this is how it's called). Our goal: a silhouette of 1 (one) will mean that the data instance is well rooted in the cluster, while the score of 0 (zero) will be assigned to data instances on the border between two clusters.

For a given data point (say the blue point in the image on the left), we can measure the distance to all the other points in its cluster and compute the average. Let us denote this average distance with A . The smaller A , the better.





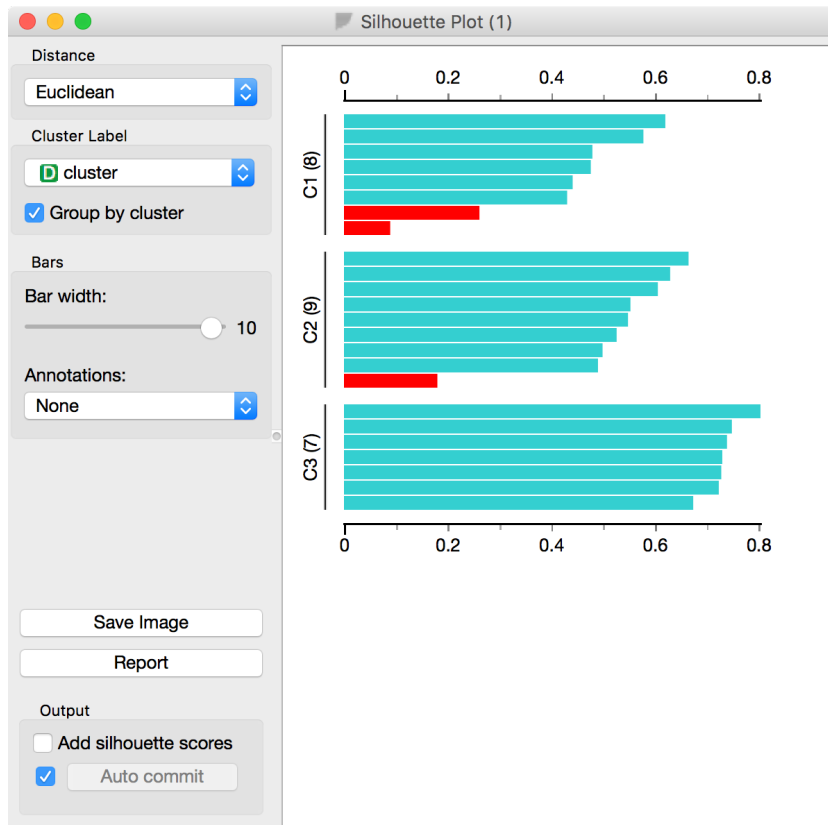
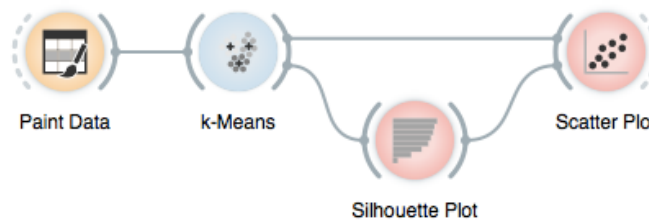
C3 is the green cluster, and all its points have large silhouettes. Not so for the other two.

Below we selected three data instances with the worst silhouette scores. Can you guess where they lie in the scatter plot?

On the other hand, we would like a data point to be far away from the points in the closest neighboring cluster. The closest cluster to our blue data point is the red cluster. We can measure the distances between the blue data point and all the points in the red cluster, and again compute the average. Let us denote this average distance as B . The larger B , the better.

The point is well rooted within its own cluster if the distance to the points from the neighboring cluster (B) is much larger than the distance to the points from its own cluster (A), hence we compute $B - A$. We normalize it by dividing it with the larger of these two numbers, $S = (B - A) / \max\{A, B\}$. Voilà, S is our silhouette score.

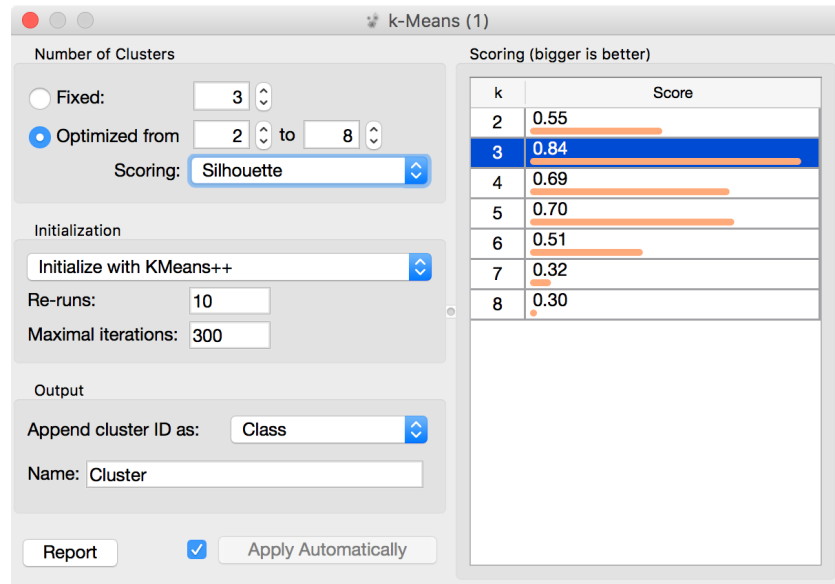
Orange has a Silhouette Plot widget that displays the values of the silhouette score for each data instance. We can also choose a particular data instance in the silhouette plot and check out its position in the scatter plot.



This of course looks great for data sets with two features, where the scatter plot reveals all the information. In higher-dimensional data, the scatter plot shows just two features at a time, so two points that seem close in the scatter plot may be actually far apart when all features - perhaps thousands of gene expressions - are taken into account.

The total quality of clustering - the silhouette of the clustering - is the average silhouette across all points. When the k-Means widget searches for the optimal number

of clusters, it tries different number of clusters and displays the corresponding silhouette scores.



Ah, one more thing: Silhouette Plot can be used on any data, not just on data sets that are the output of clustering. We could use it with the iris data set and figure out which class is well separated from the other two and, conversely, which data instances from one class are similar to those from another.

We don't have to group the instances by the class. For instance, the silhouette on the left would suggest that the patients from the heart disease data with typical anginal pain are similar to each other (with respect to the distance/similarity computed from all features), while those with other types of pain, especially non-anginal pain are not clustered together at all.

