

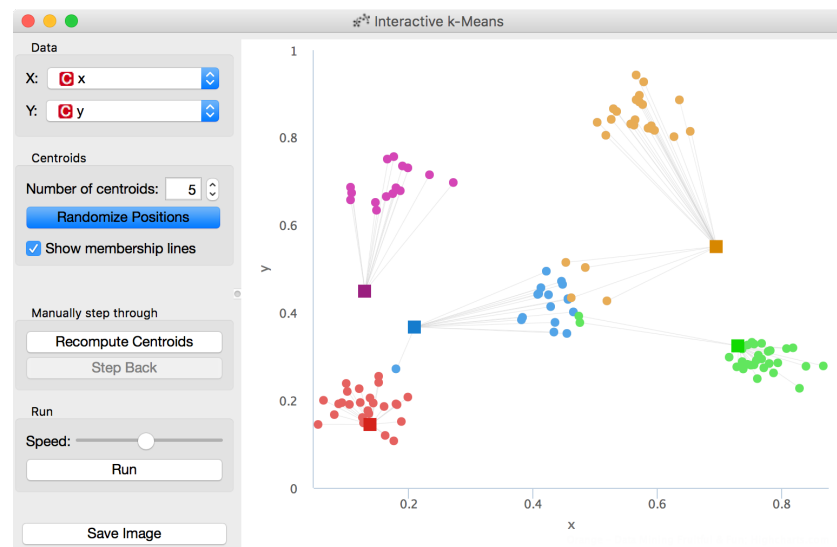
Lesson 27: k-Means Clustering

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 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?

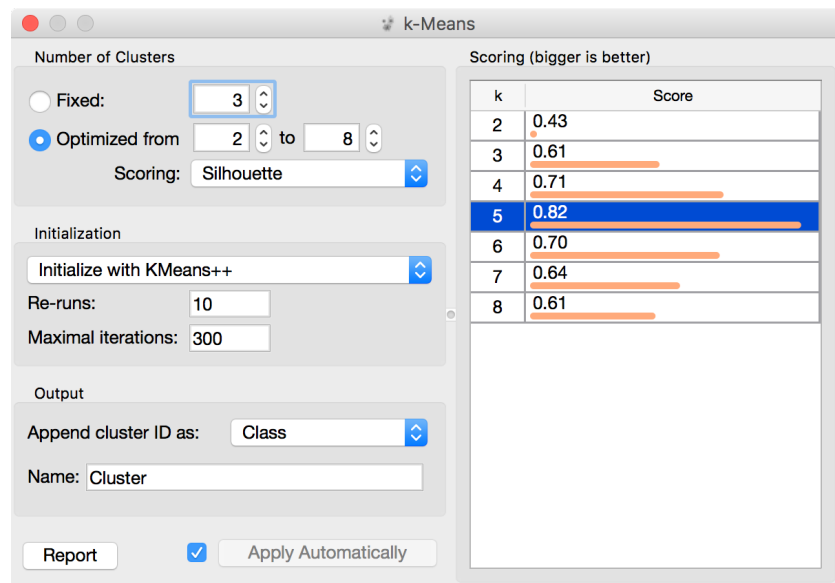
Keep pressing Recompute Centroids and Reassign Membership until it stops changes. With this simple, two-dimensional data it will take just a few iterations; with more points and features, it can take longer, but the principle is the same.

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?

Lesson 28: Finding Clusters When There Are None

We saw how clustering can 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 actually exist.

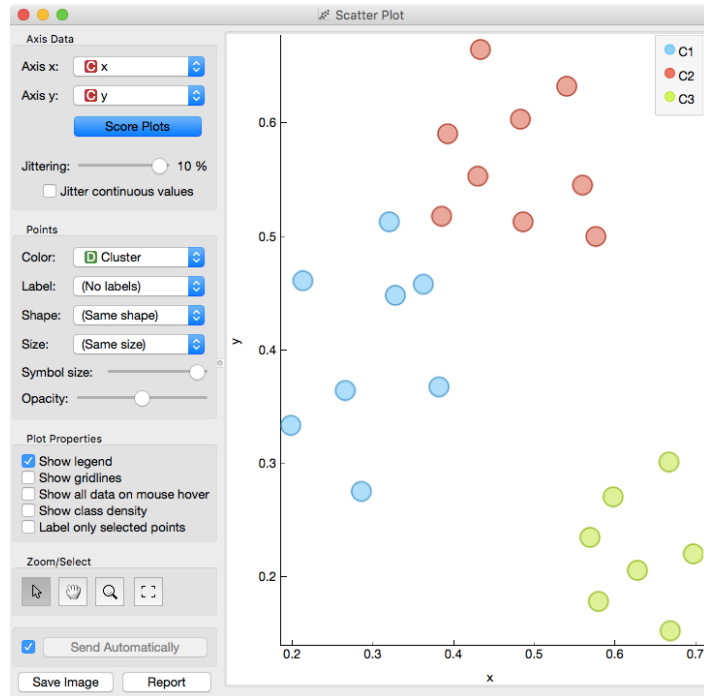
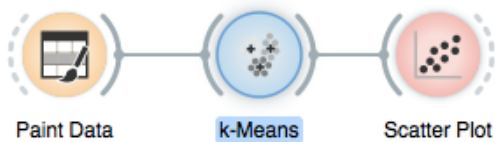


Playing with Paint Data and k-Means can be quite fun. Try painting the data where there are clusters, but k-means does not find them. Or, actually, finds the wrong ones. What kind of clusters are easy to find for k-means? Are these the kind of clusters we would actually find in real data sets?

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 forming new hypotheses, which must make biological sense and be verified on new, independent data. We cannot make conclusions based only on "discovering" clusters.

Lesson 31: Silhouettes

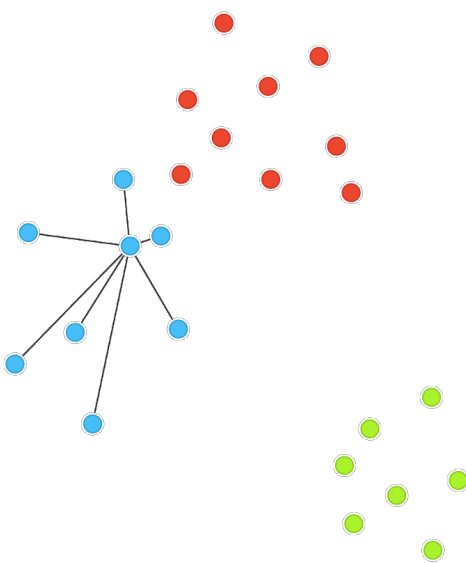
Don't get confused: we paint data and/or visualize it with Scatter plots, which show only two features. This is just for an illustration! Most data sets contain many features and methods like k-Means clustering take into account all features, not just two.

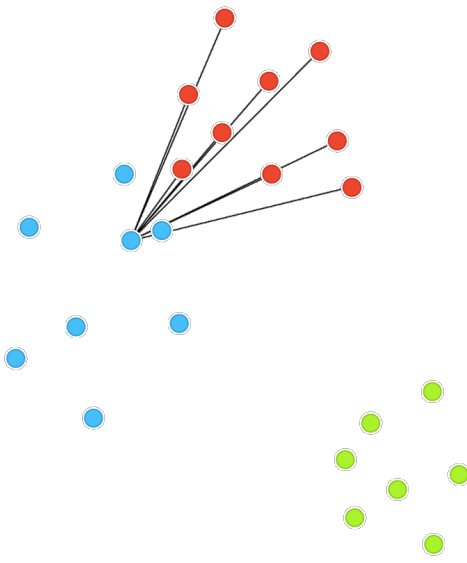


The data points in the green cluster are well separated from those in the other two. Not so for the blue and red points, where several points are on the border between the clusters. We would like to quantify the degree of how well a data point belongs to the cluster to which it is assigned.

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.





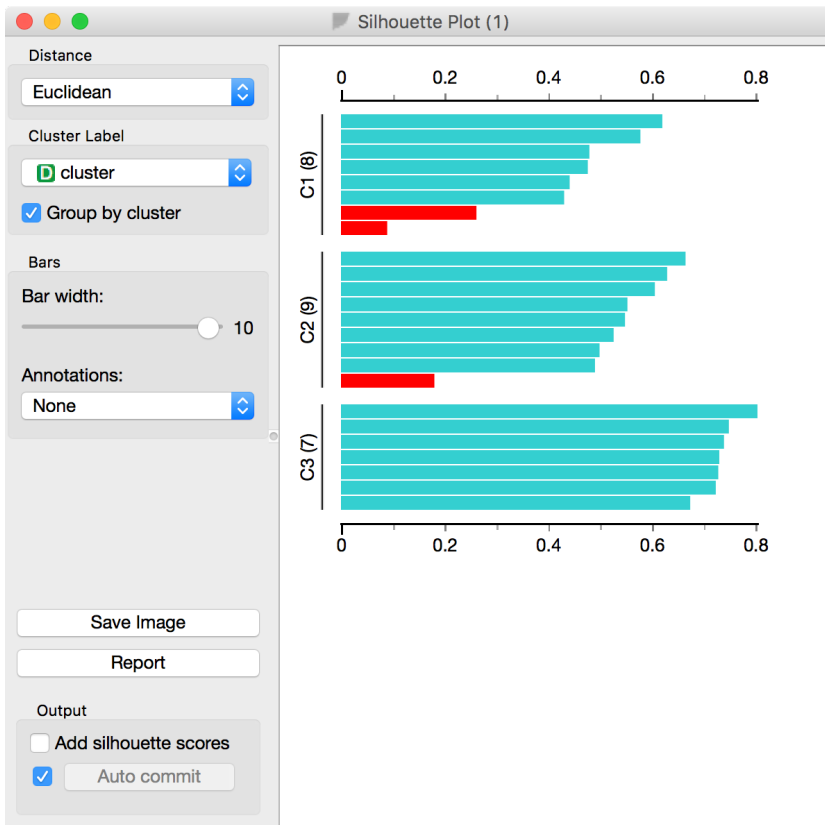
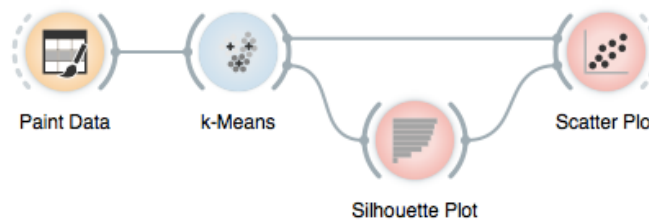
On the other hand, we would like a data point to be far away from the points in the closest neighboring cluster. The closest cluster to our blue data point is the red cluster. We can measure the distances between the blue data point and all the points in the red cluster, and again compute the average.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

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

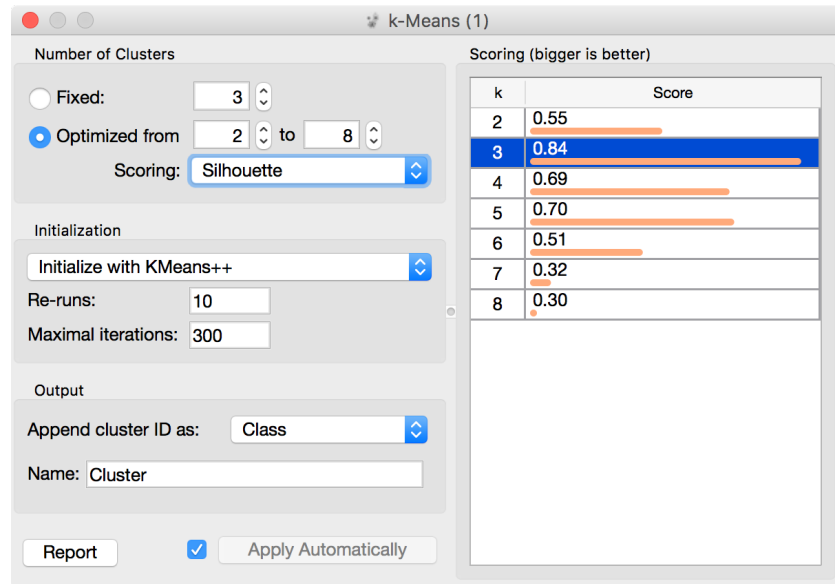
Below we selected three data instances with the worst silhouette scores. Can you guess where they lie in the scatter plot?



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.

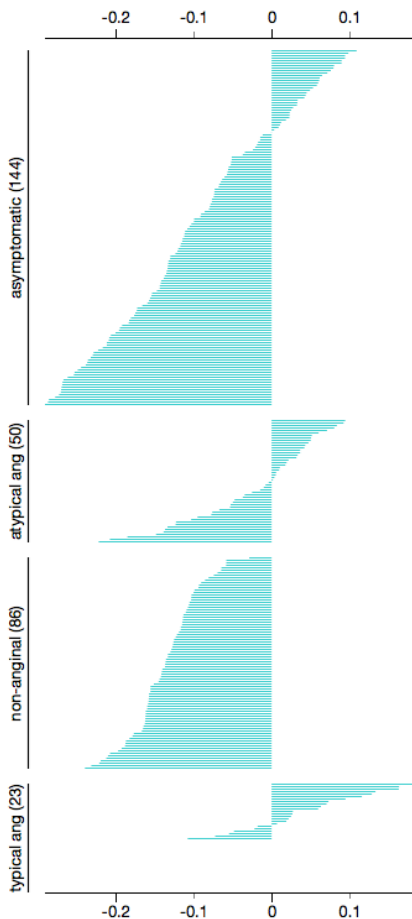
The total quality of clustering - the silhouette of the clustering - is the average silhouette across all points. When the k-Means widget searches for the optimal number

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.



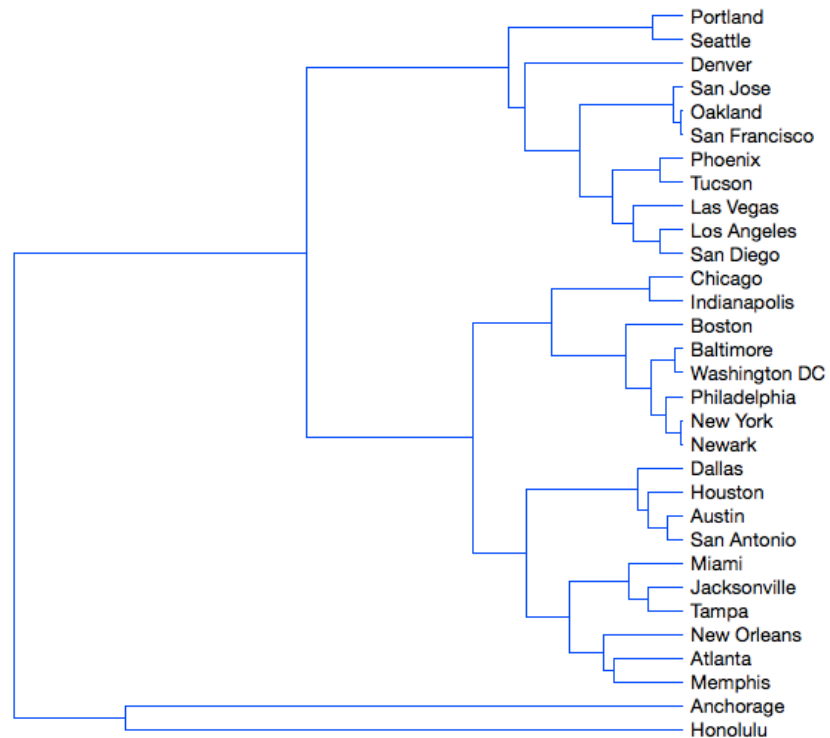
Lesson 32: Mapping the Data

Imagine a foreign visitor to the US who knows nothing about the US geography. He doesn't even have a map; the only data he has is a list of distances between the cities. Oh, yes, and he attended the Introduction to Data Mining.

If we know distances between the cities, we can cluster them.

For this example we retrieved data from http://www.mapcrow.info/united_states.html, removed the city names from the first line and replaced it with "31 labelled".

The file is available at <http://file.biolab.si/files/us-cities.dst.zip>. To load it, unzip the file and use the File Distance widget from the Prototypes add-on.



How much sense does it make? Austin and San Antonio are closer to each other than to Houston; the tree is then joined by Dallas. On the other hand, New Orleans is much closer to Houston than to Miami. And, well, good luck hitchhiking from Anchorage to Honolulu.

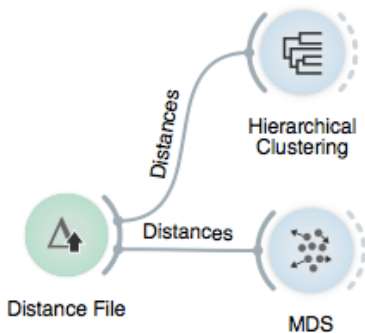
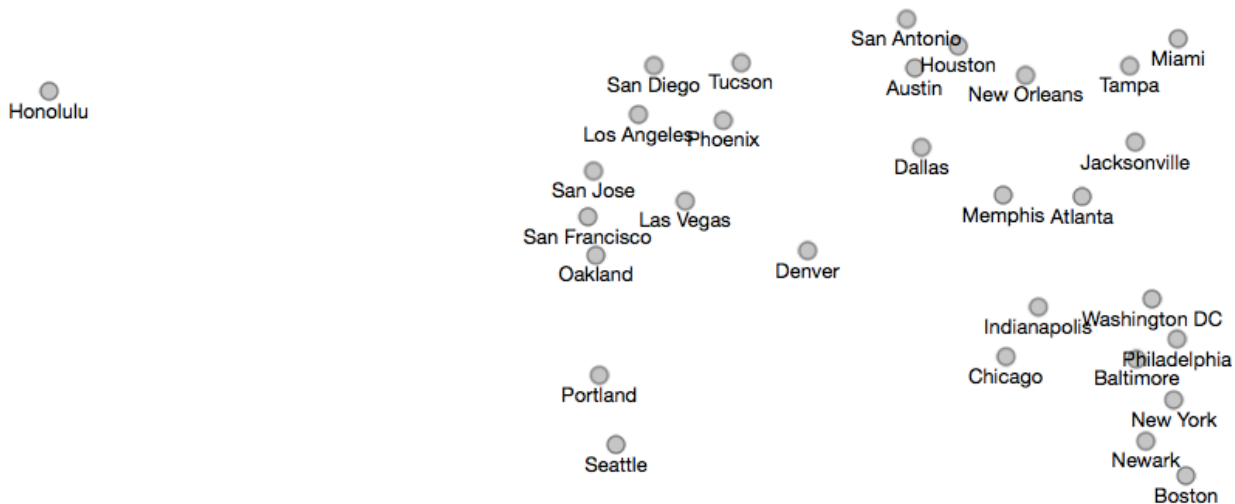
As for Anchorage and Honolulu, they are left-overs; when there were only three clusters left (Honolulu, Anchorage and the big cluster with everything else), Honolulu and Anchorage were closer to each other than to the rest. But not close — the corresponding lines in the dendrogram are really long.

We can't run k-means clustering on this data, since we only have distances, and k-means runs on real (tabular) data. Yet, k-means would have the same problem as hierarchical clustering.

The real problem is New Orleans and San Antonio: New Orleans is close to Atlanta and Memphis, Miami is close to Jacksonville and Tampa. And these two clusters are suddenly more similar to each other than to some distant cities in Texas.

In general, two points from different clusters may be more similar to each other than to some points from their corresponding clusters.

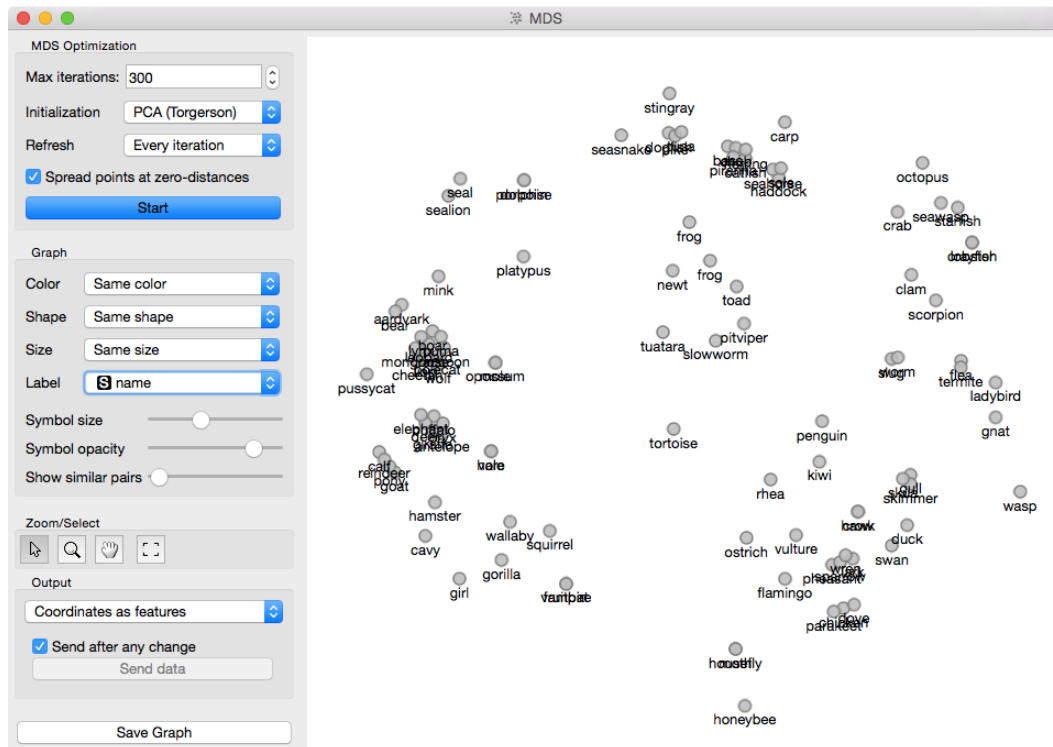
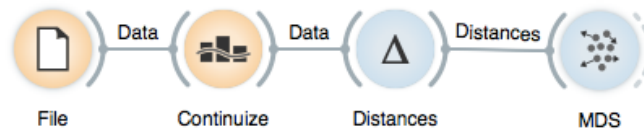
To get a better impression about the physical layout of cities, people have invented a better tool: a map! Can we reconstruct a map from a matrix of distances? Sure. Take any pair of cities and put them on paper with a distance corresponding to some scale. Add the third city and put it at the corresponding distance from the two. Continue until done. Excluding, for the sake of scale, Anchorage, we get the following map.



We have not constructed this map manually, of course. We used a widget called MDS, which stands for Multidimensional scaling.

It is actually a rather exact map of the US from the Australian perspective. You cannot get the orientation from a map of distances, but now we have a good impression about the relations between cities. It is certainly much better than clustering.

Remember the clustering of animals? Can we draw a map of animals?



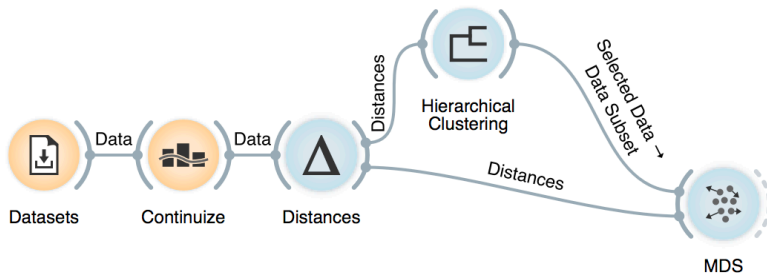
Does the map make any sense? Are similar animals together? Color the points by the types of animals and you should see.

The map of the US was accurate: one can put the points in a plane so that the distances correspond to actual distances between cities. For most data, this is usually impossible. What we get is a projection (a non-linear projection, if you care about mathematical finesses) of the data. You lose something, but you get a picture.

The MDS algorithm does not always find the optimal map. You may want to restart the MDS from random positions. Use the slider “Show similar pairs” to see whether the points that are placed together (or apart) actually belong together. In the above case, the honeybee belongs closer to the wasp, but could not fly there as in the process of optimization it bumped into the hostile region of flamingos and swans.

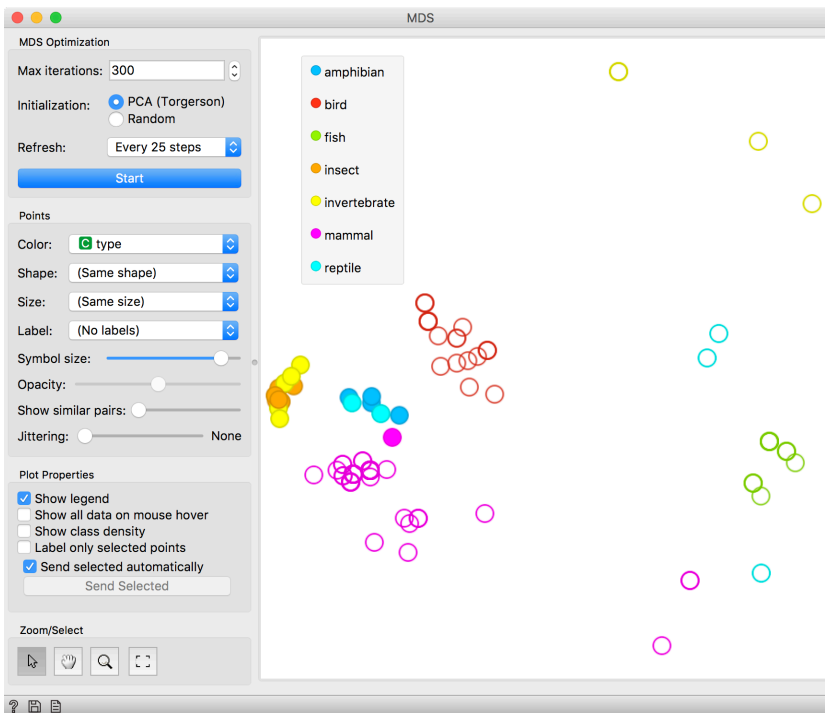
Lesson 33: All Together Now

Remember the mixed cluster in the zoo data that contained invertebrates, reptiles, amphibian, and even a mammal. Was this a homogeneous cluster? Why the mammal there? And how far is this mammal to other mammals? And why is this cluster close to the cluster of mammals?



So many questions. But we can answer them all with a combination of clustering and multi-dimensional scaling. We would like to show any cluster that we selected from a dendrogram to be shown on the map of animals presented by MDS. And we would like to use cosine distances,

so we need to take care of the composition of the workflow and proper connections between widgets.



Clustering and two-dimensional embedding make a great combination for data exploration. Clustering finds the coherent groups, and embedding, such as MDS, reveals the relations between the clusters and positions the cluster on the data map. There are other dimensionality reduction and embedding techniques that we could use, but for smaller data sets, MDS is great because it tries to preserve the distances from the original data space.

Can you change the workflow to explore the position of individual clusters found by k-means?