

# DATA SCIENCE:

A HANDS-ON COURSE AT BAYLOR COLLEGE OF MEDICINE

BIOINFORMATICS LAB, LJUBLJANA

We have designed this course for prospective STEM teachers. These working notes include Orange workflows and visualizations we will construct during the lectures. Throughout our training, you will see how to accomplish various data mining tasks through visual programming and use Orange to build visual data mining workflows. Many similar data mining environments exist, but the lecturers prefer Orange for one simple reason—they are its authors.

If you haven't already installed Orange, please download the installation package from <http://orangedatamining.com>.

The notes were written by Blaž Zupan and Janez Demšar with massive help from the members of the Bioinformatics Lab in Ljubljana, Slovenia, that developed Orange. We would specifically like to thank Ajda Pretnar Žagar and Marko Toplak for proofreading, editing, and converting our previous documents in Pages to LaTeX.

Copyright © 2022

PUBLISHED BY BIOINFORMATICS LAB, LJUBLJANA

TUFTE-LATEX.GOOGLECODE.COM

Licensed under the Apache License, Version 2.0 (the “License”); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.apache.org/licenses/LICENSE-2.0>. Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

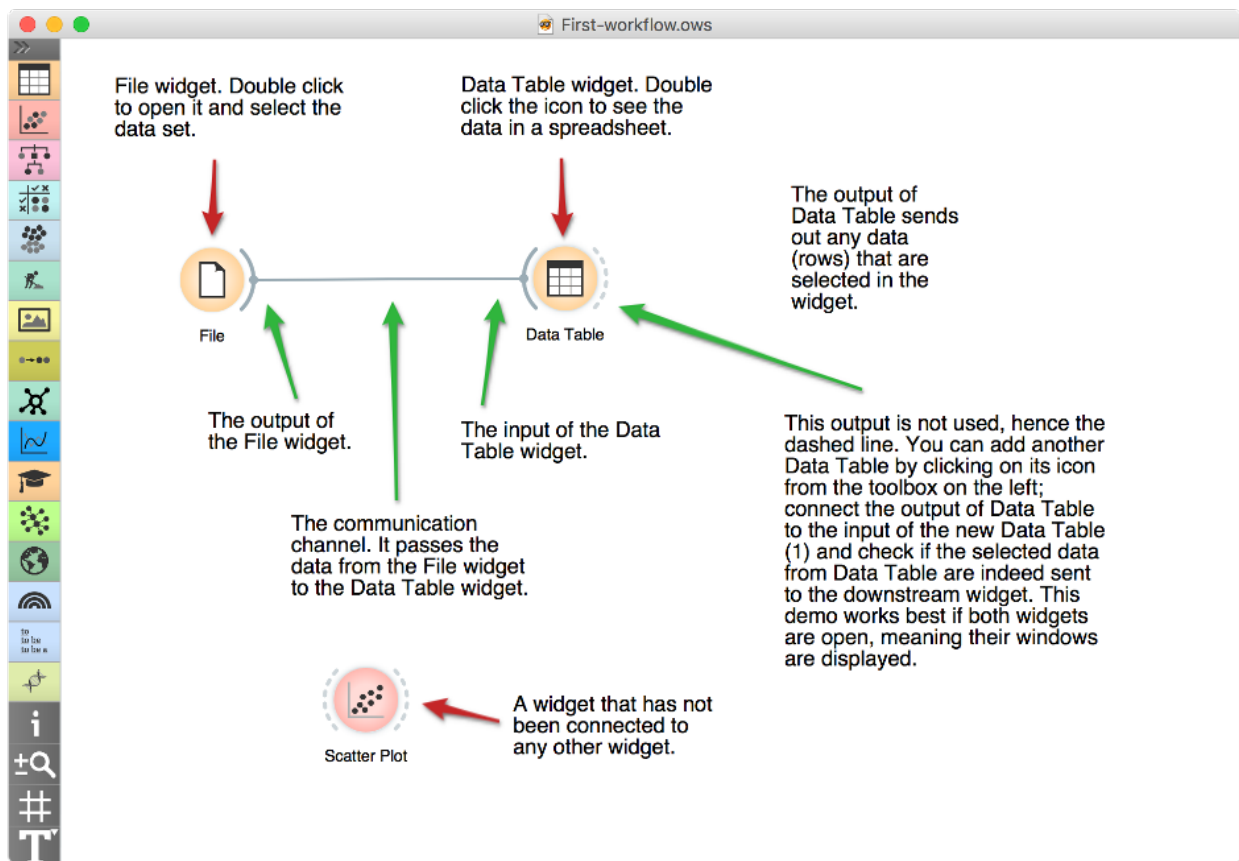
*First printing, June 2022*

# *Contents*

<i>Workflows in Orange</i>	5
<i>Basic data exploration</i>	9
<i>Saving your work</i>	12
<i>Loading data sets</i>	13
<i>Hierarchical Clustering</i>	15
<i>Animal Kingdom</i>	17
<i>Silhouettes</i>	18
<i>k-Means Clustering</i>	21

# Workflows in Orange

ORANGE WORKFLOWS consist of components that read, process, and visualize data. We refer to these components as “widgets” We place the widgets on a drawing board called the “canvas” to design a workflow. Widgets communicate by sending information along their communication channel. Output from one widget can be used as input to another.

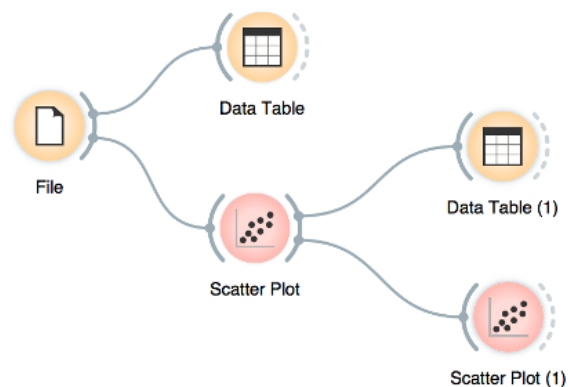


We construct workflows by dragging widgets onto the canvas and connecting them by drawing a line from the transmitting widget to the receiving widget. The widget’s outputs are on the right, and the inputs on the left. In the workflow above, the *File* widget sends data to the *Data Table* widget.

**A simple workflow with two connected widgets and one widget without connections. The outputs of a widget appear on the right, while the inputs appear on the left.**

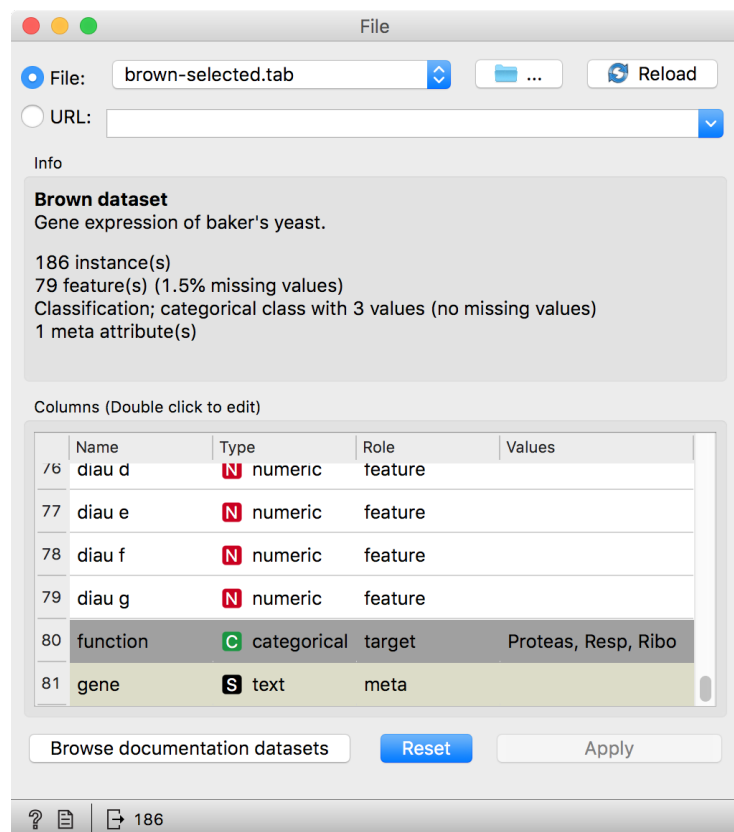
Start by constructing a workflow that consists of a File widget, two *Scatter Plot* widgets and two *Data Table* widgets:

A workflow with a File widget that reads the data from a disk and sends it to the Scatter Plot and Data Table widget. The Data Table renders the data in a spreadsheet, while the Scatter Plot visualizes it. The plot's selected data points are sent to two other widgets: Data Table (1) and Scatter Plot (1).



The *File* widget reads data from your local disk. Open the *File* widget by double-clicking its icon. Orange comes with several pre-loaded data sets. From these (“Browse documentation data sets...”), choose *brown-selected.tab*, a yeast gene expression data set.

Orange workflows often start with a File widget. The brown-selected data set comprises 186 rows (genes) and 81 columns. Out of the 81 columns, 79 contain gene expressions of baker’s yeast under various conditions, one column (marked as a “meta attribute”) provides gene names, and one column contains the “class” value or gene function.

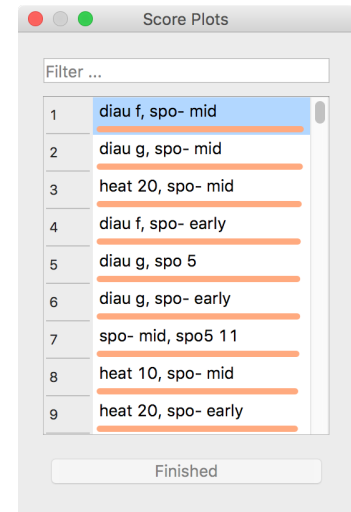


After you load the data:

1. Open the other widgets.
2. Select a few data points in the *Scatter Plot* widget and watch as they appear in the *Data Table (1)*.

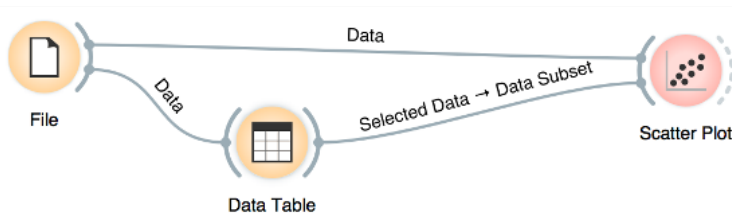
- Use a combination of two *Scatter Plot* widgets, where the second scatter plot shows a detail from a smaller region selected in the first scatter plot.

The following is a side note, but it won't hurt. The scatter plot for a pair of random features does not provide much information on gene function. Does this change with a different choice of feature pairs in the visualization? *Rank projections* at the button on the top left of the Scatter Plot widget can help you find a good feature pair. How do you think this works? Could the suggested pairs of features be helpful to a biologist?



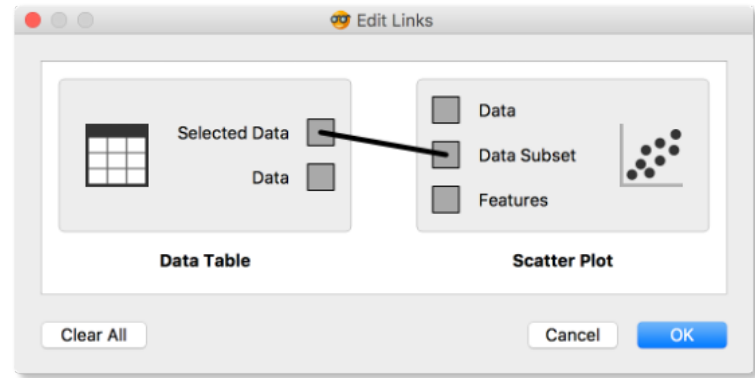
Scatter Plot and Ranking

We can connect the output of the *Data Table* widget to the *Scatter Plot* widget to highlight the chosen data instances (rows) in the scatter plot.



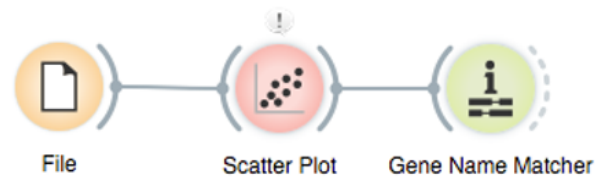
How does Orange distinguish between the primary data source and the data selection? It uses the first connected signal as the entire data set and the second one as its subset. To make changes or to check what is happening under the hood, double click on the line connecting the two widgets.

In this workflow, we have switched on the option "Show channel names between widgets" in File/Preferences.



The rows in the data set we are exploring in this lesson are gene profiles. We could perhaps use widgets from the Bioinformatics add-on to get more information on the genes we selected in any of the Orange widgets.

Orange comes with a basic set of widgets for data input, preprocessing, visualization and modeling. For other tasks, like text mining, network analysis, and bioinformatics, there are add-ons. Check them out by selecting Add-ons... from the Options menu.



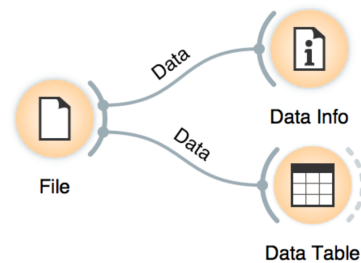


## Basic data exploration

LET US CONSIDER ANOTHER PROBLEM, this time from clinical medicine. We will dig for something interesting in the data and explore it with visualization widgets. You will get to know Orangebetter, and also learn about several interesting visualizations.

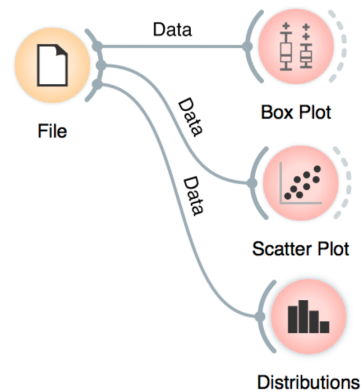
We will start with an empty canvas; to clean it from our previous lesson, use either File/New or select all the widgets and remove them (use the backspace/delete key).

Now again, add the File widget and open another documentation data set: heart\_disease. How does the data look like?



**A simple workflow to inspect the loaded dataset.**

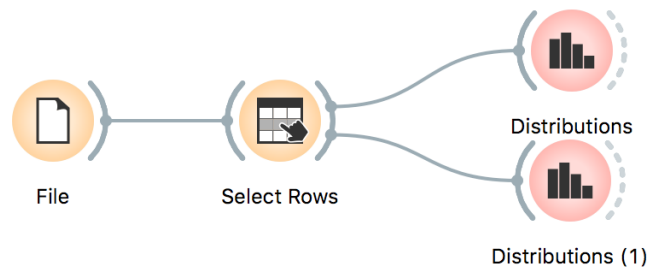
Let us check whether common visualizations tell us anything interesting. (Hint: look for gender differences. These are always interesting and occasionally even real.)



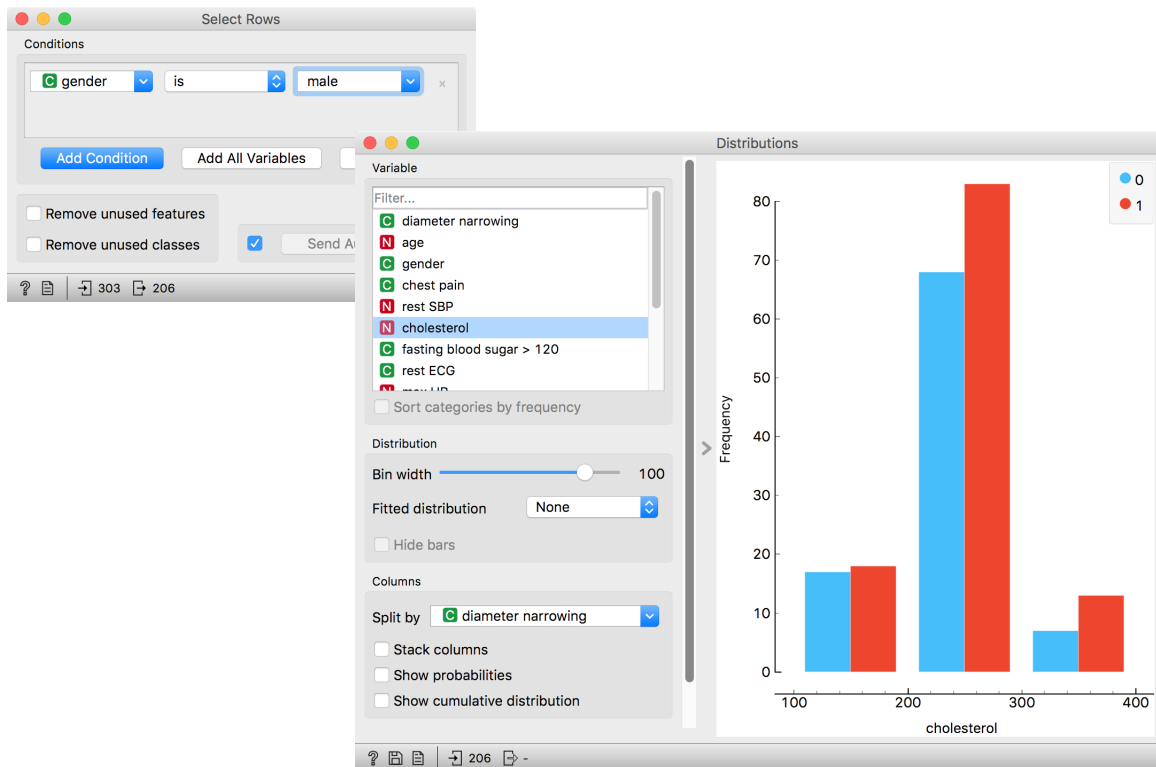
**Quick check with common statistics and visualization widgets.**

Data can also be split by the value of features, in this case the gender.

The two Distributions widgets get different data: the upper gets the selected rows and the lower gets the rest. Double-click the connection between the widgets to access setup dialog, as you've learned in the previous lesson.



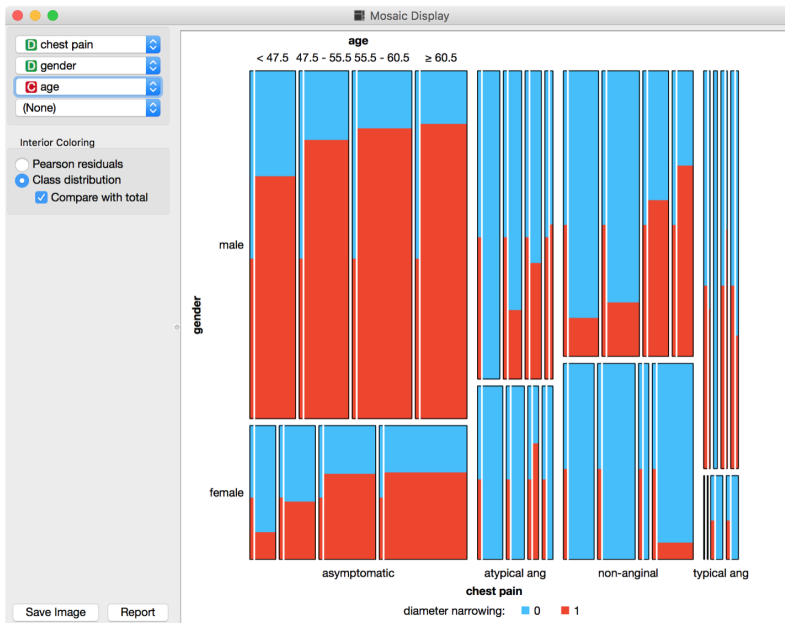
In the *Select Rows* widget, we select the female patients. You can also add other conditions. The selection of data instances provides a powerful combination with visualization of data distribution. Try having at least two widgets open simultaneously and explore the data.



There are two less-known — but great — visualizations for observing interactions between features.

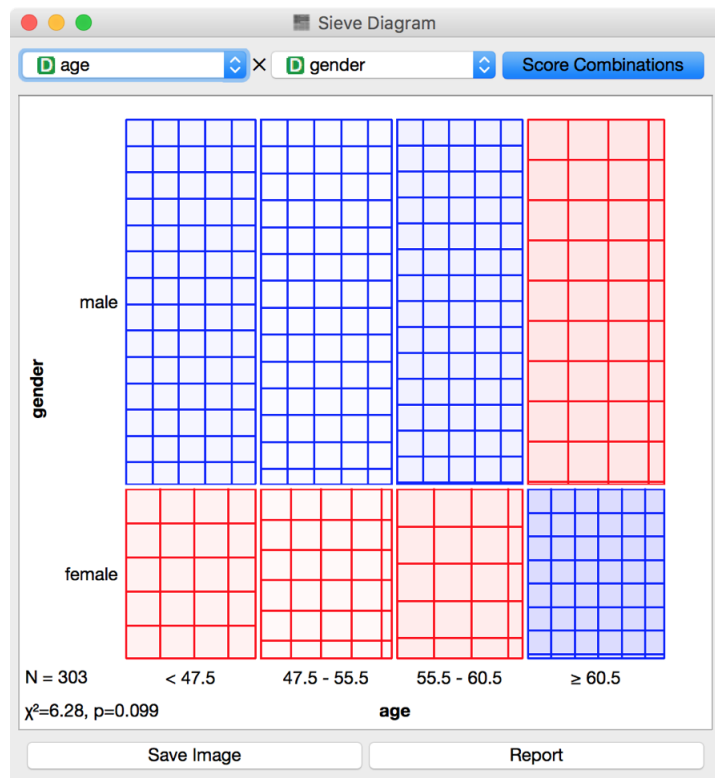
The mosaic display shows a rectangle split into columns with widths reflecting the prevalence of different types of chest pain. Each column is then further split vertically according to gender distributions within the column. The resulting rectangles are split again horizontally according to age group sizes. The red and blue areas represent each group's outcome distribution within the resulting bars, and the tiny strip to the left of each shows the overall distribution.

What can you read from this diagram?



You can play with the widget by trying different combinations of 1-4 features.

Another visualization, the Sieve diagram also splits a rectangle horizontally and vertically, but with independent cuts, so the areas correspond to the expected number of data instances if the observed variables were independent. For instance, 1/4 of patients are older than 60, and 1/3 of patients are female, so the area of the bottom right rectangle is 1/12 of the total area. With roughly 300 patients, we would expect  $1/12 \times 300 = 25$  older women in our data. Instead, there are 34. The sieve diagram shows the difference between the expected and the observed frequencies by the grid density and the color of the field.

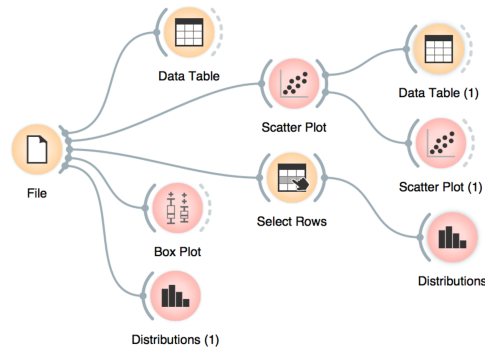


See the Score Combinations button? Try to guess what it does? And how does it score the combinations? Hint: there are some Greek letters at the bottom of the widget.

## Saving your work

AT THE END OF A LESSON, your workflow may look like this:

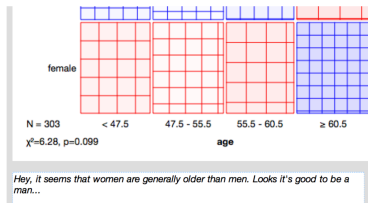
A fairly complex workflow that you would want to share or reuse at a later time.



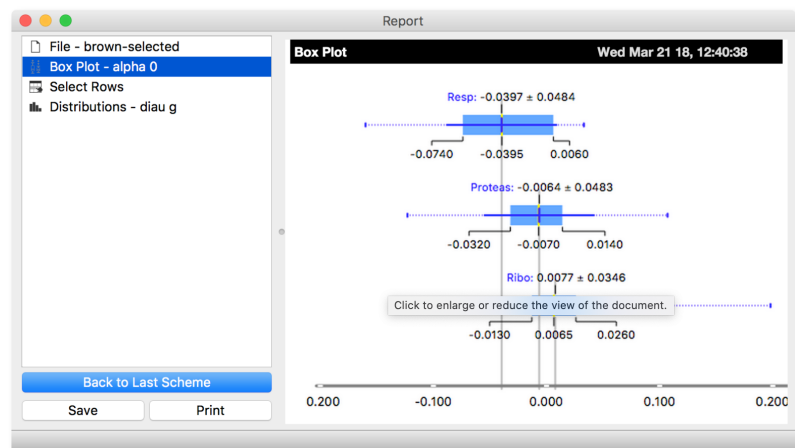
You can save this workflow using the File/Save menu and share it with your colleagues. Just don't forget to put the data files in the same directory as the file with the workflow.

Widgets also have a Report button in their bottom status bar, which you can use to keep a log of your analysis. When you find something interesting, just click it and the graph will be added to your log. You can also add reports from the widgets on the path to this one, to make sure you don't forget anything relevant.

Clicking on a section of the report window allows you to add a comment.



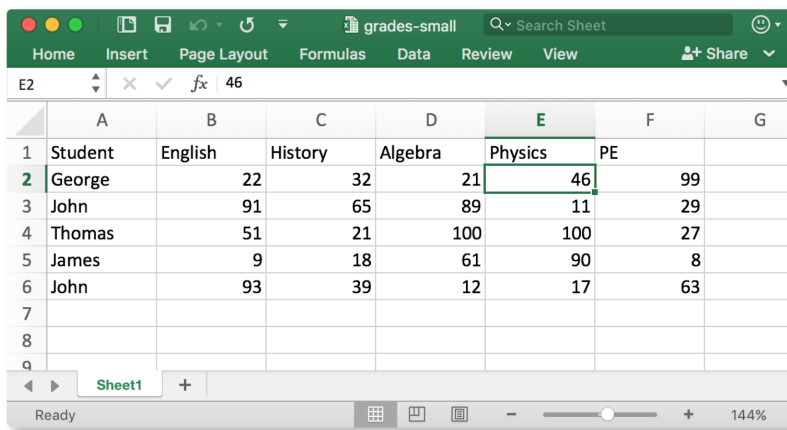
The report window and the additional text input box (bottom).



You can save the report as HTML or PDF, or a report file that includes all workflow related report items that you can later open in Orange. In this way, you and your colleagues can reproduce your analysis results.

## Loading data sets

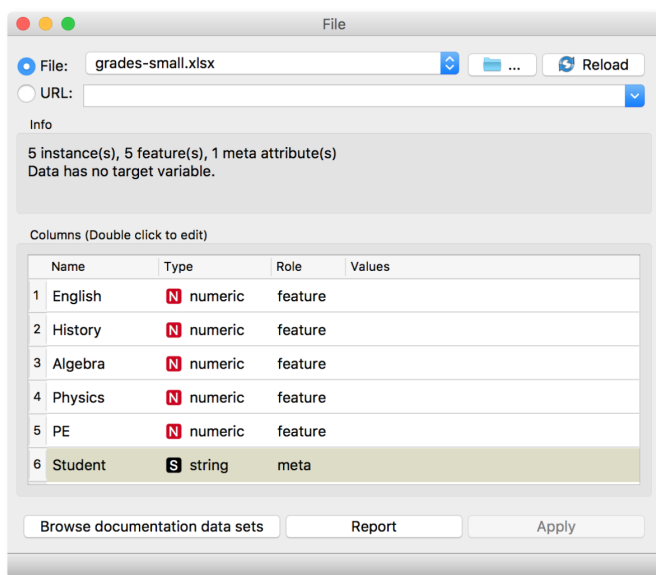
THE DATA SETS WE HAVE WORKED WITH in the previous lesson come with the Orange installation. Orange can read data from many file formats which include tab and comma separated and Excel files. To see how this works, let's prepare a data set (with school subjects and grades) in Excel and save it on a local disk.



	A	B	C	D	E	F	G
1	Student	English	History	Algebra	Physics	PE	
2	George	22	32	21	46	99	
3	John	91	65	89	11	29	
4	Thomas	51	21	100	100	27	
5	James	9	18	61	90	8	
6	John	93	39	12	17	63	
7							
8							
9							

Make a spreadsheet in Excel with the numbers shown on the left. Of course, you can use any other editor, but remember to save your file in the comma separated values (\*.csv) format.

In Orange, we can use, for example, the File widget to load this data set.



File: grades-small.xlsx

Info

5 instance(s), 5 feature(s), 1 meta attribute(s)  
Data has no target variable.

Columns (Double click to edit)

	Name	Type	Role	Values
1	English	N numeric	feature	
2	History	N numeric	feature	
3	Algebra	N numeric	feature	
4	Physics	N numeric	feature	
5	PE	N numeric	feature	
6	Student	S string	meta	

Browse documentation data sets Report Apply

The File widget allows you to select a local file or even paste a URL to a Google Spreadsheet. In the Info box, you will see a quick summary about the data you loaded. By double clicking the fields, you can also edit the types of entries and their role, that will be relevant for further processing.

Looks good! Orange has correctly guessed that student names are character strings and that this column in the data set is special, meant

to provide additional information and not to be used for any kind of modeling (more about this in the upcoming lectures). All other columns are numeric features.

It is always good to check if all the data was read correctly. Now, you can connect the *File* widget with the *Data Table* widget,

Construct a simple workflow shown on the right.



and double click on the Data Table to see the data in a spreadsheet format. Nice, everything is here.

	Student	English	History	Algebra	Physics	Physical	GPA
1	George	22.000	32.000	21.000	46.000	99.000	3.000
2	John	91.000	65.000	89.000	11.000	29.000	3.000
3	Thomas	51.000	21.000	100.000	100.000	27.000	3.000
4	James	9.000	18.000	61.000	90.000	8.000	2.000
5	John	93.000	39.000	12.000	17.000	63.000	1.000

The Data Table widget shows the loaded data set, you can select rows, which will appear on the output of the widget. It is also possible to do simple data visualizations. Explore the functionalities!

Instead of using Excel, we could also use Google Sheets, a free on-line spreadsheet alternative. Then, instead of finding the file on the local disk, we would enter its URL address to the File widget URL entry box.

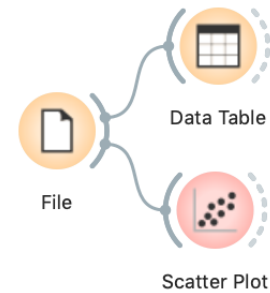
Orange's legacy native data format is a tab-delimited text file with three header rows. The first row lists the attribute names, the second row defines their type (continuous, discrete, time and string, or abbreviated c, d, t, and s), and the third row an optional role (class, meta, weight, or ignore).

There is more to input data formatting and loading. If you would really like to dive in for more, check out the documentation page on [Loading your Data](#), or a [video tutorial](#) on this subject.

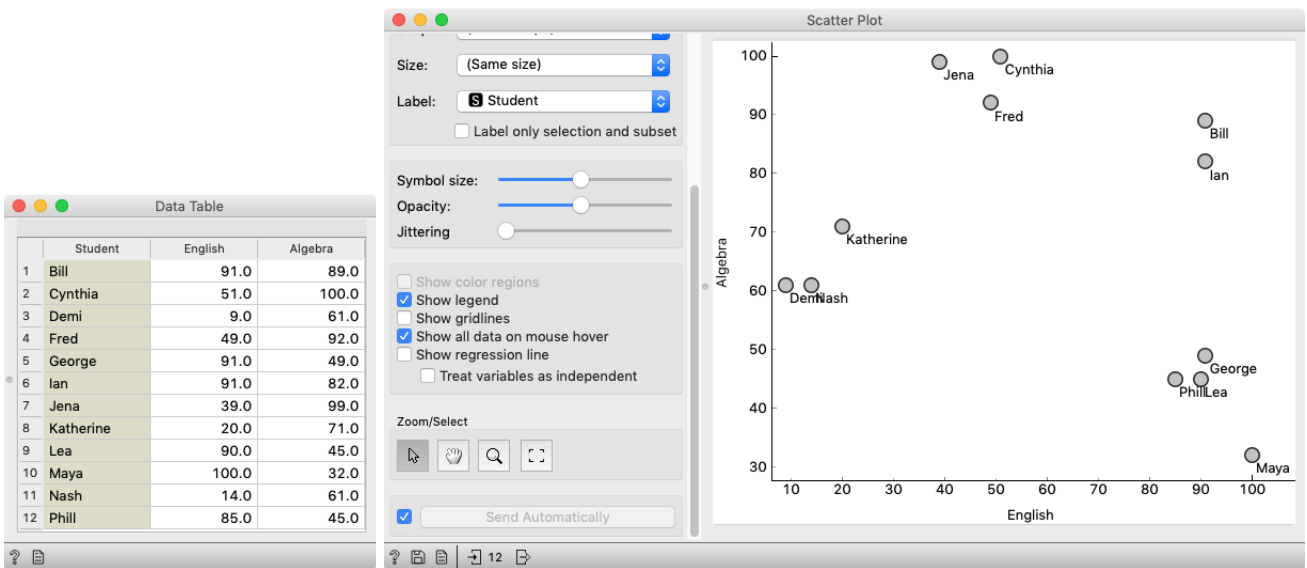
# Hierarchical Clustering

WE ARE INTERESTED IN FINDING CLUSTERS IN OUR DATA. We want to identify groups of data instances 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 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. One simple measure of similarity is the Euclidean distance. So, we would like to group data instances with small Euclidean distances.

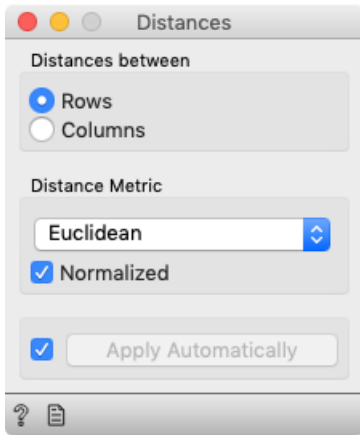


We will introduce clustering with a simple data set on students and their grades in English and Algebra. Load the data set from <http://file.biola.edu/text/grades.tab>.

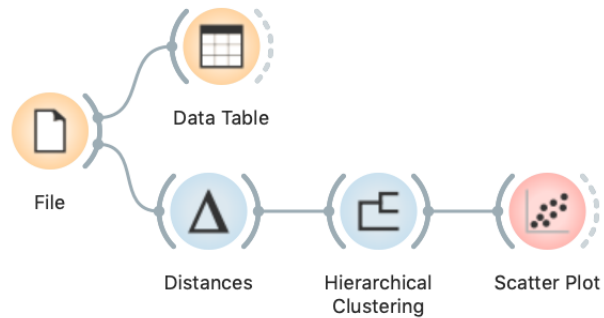


Next, we need to define a clustering algorithm. Say that we start with each data instance being its cluster, and then, at each step, we join the closest clusters. We estimate the distance between the clusters with the average distance between all their pairs of data points. This algorithm is called hierarchical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

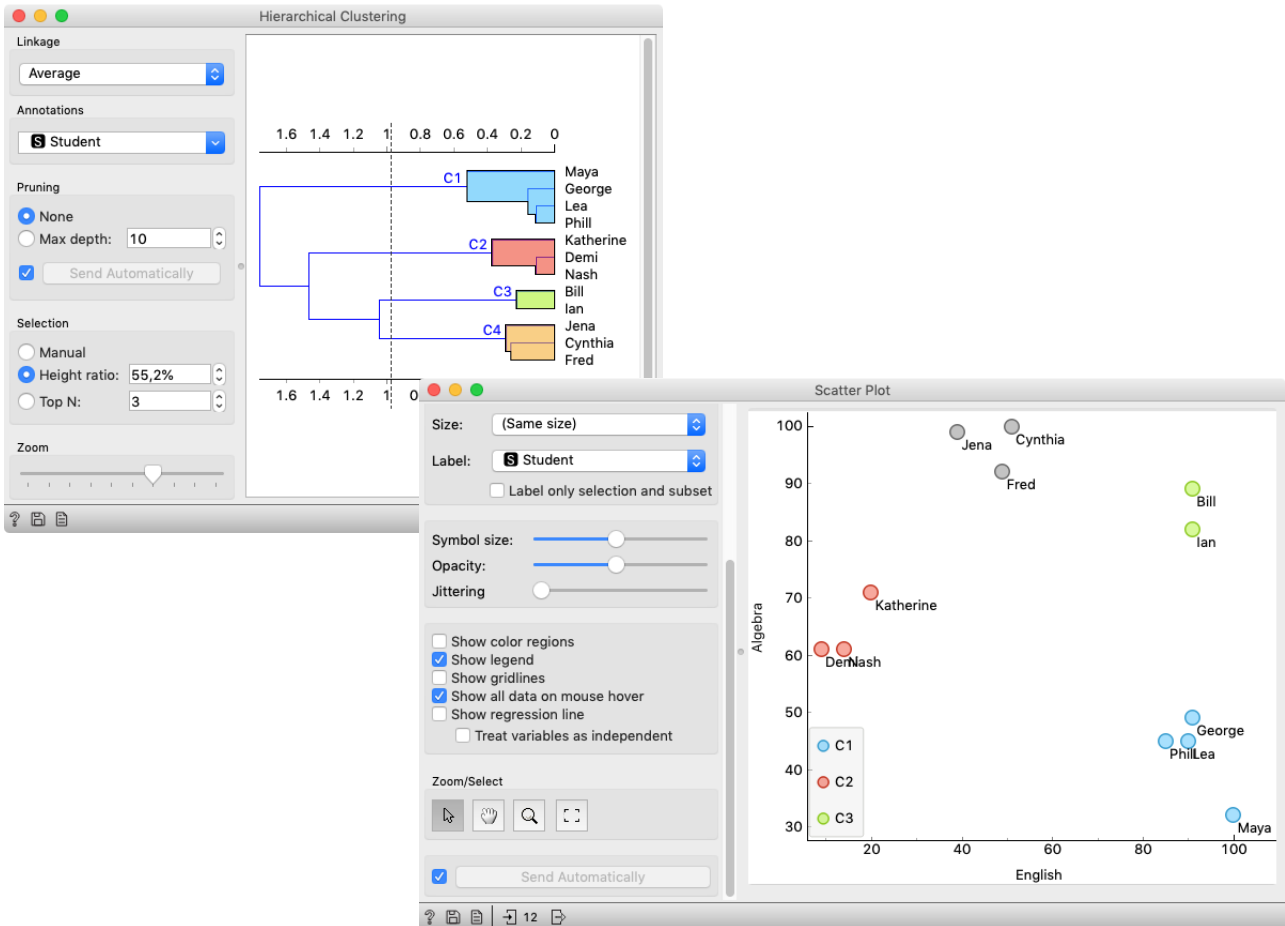
There are different ways to measure the similarity between clusters. The estimate we have described is called **average linkage**. We could also estimate the distance through the two closest points in each group (**single linkage**) or through the two points that are furthest away (**complete linkage**).



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



It couldn't be simpler. Load the data, measure the distances, use them in hierarchical clustering, and visualize the results in a scatter plot. The *Hierarchical Clustering* widget allows us to cut the hierarchy at a specific distance score and output the corresponding clusters:





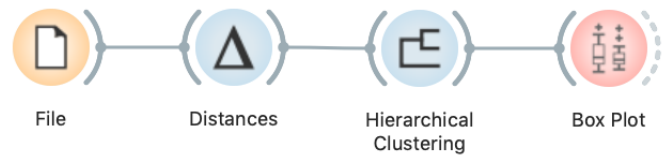
# Animal Kingdom

Your lecturers spent a 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.

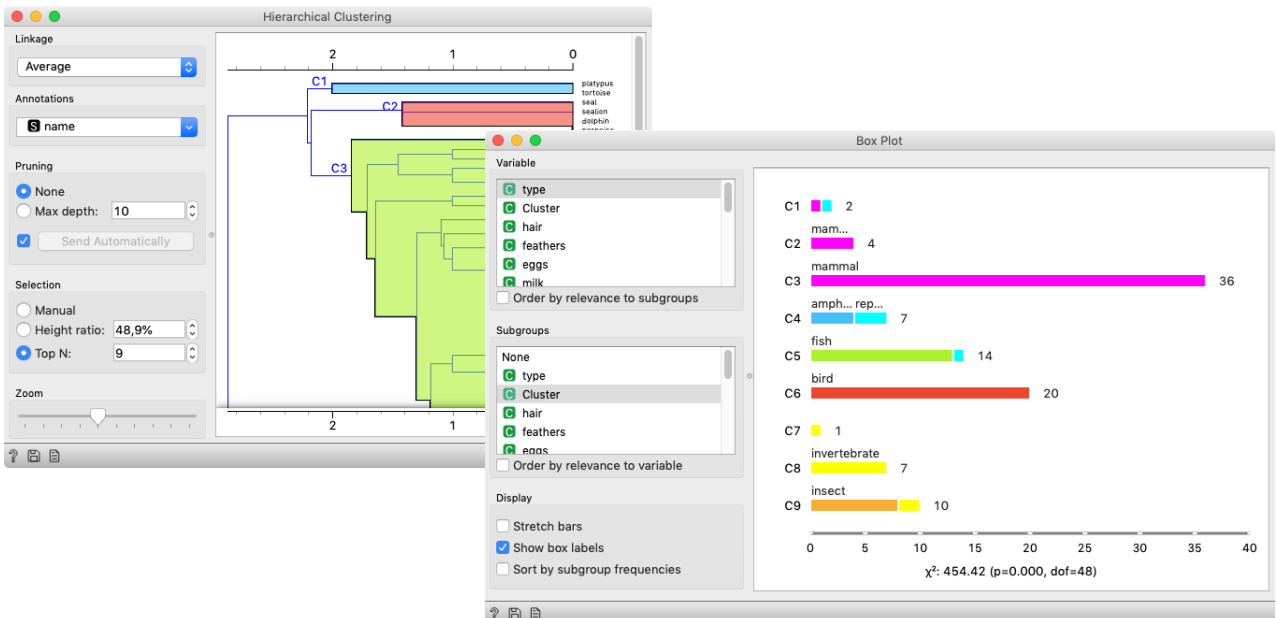
Funny stuff was we never understood the order in which the cards were laid out in the album. We later learned about taxonomy, but being more inclined to engineering we never mastered learning it in our biology classes. Luckily, there's data mining and the idea that taxonomy simply stems from measuring the distance between species.

Here we use the *zoo* data (from the documentation data sets) 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 type (mammal, insect, bird, and so on). It would be cool to know if the clustering re-discovered these groups of animals.

To split the data into clusters, let us manually set a threshold by dragging the vertical line left or right in the visualization. Can you say what is the appropriate number of groups?



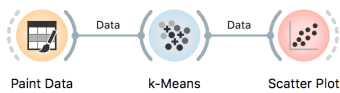
**Hierarchical clustering works fast for smaller data sets. But for bigger ones it fails. Simply, it cannot be used. Why?**



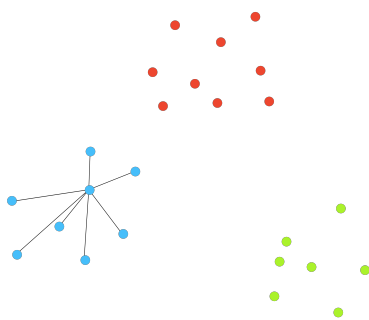
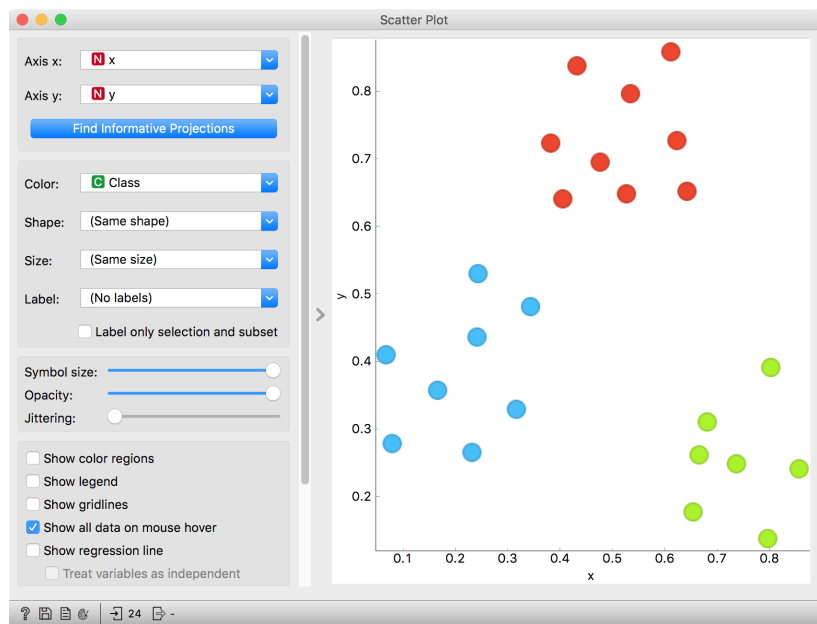
**What is wrong with those mammals? Why can't they be in one single cluster? Two reasons. First, they represent 40% of the data instances. Second, they include some weirdos. Who are they?**

# 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.



CONSIDER A TWO-FEATURE DATA SET which we have painted in the *Paint Data* widget. We send it to the k-means clustering, tell it to find three clusters, and display the clustering in the scatter plot.



Average distance A.

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.

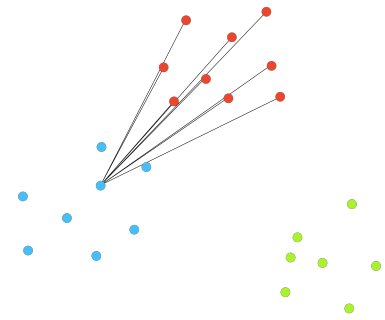
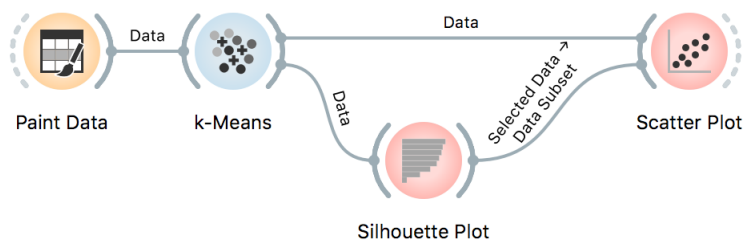
For a given data point (say the blue point in the image on the left), we can measure the distance to all the other points in its cluster and compute the average. Let us denote this average distance with  $A$ . The smaller the  $A$ , the better.

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 the 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.

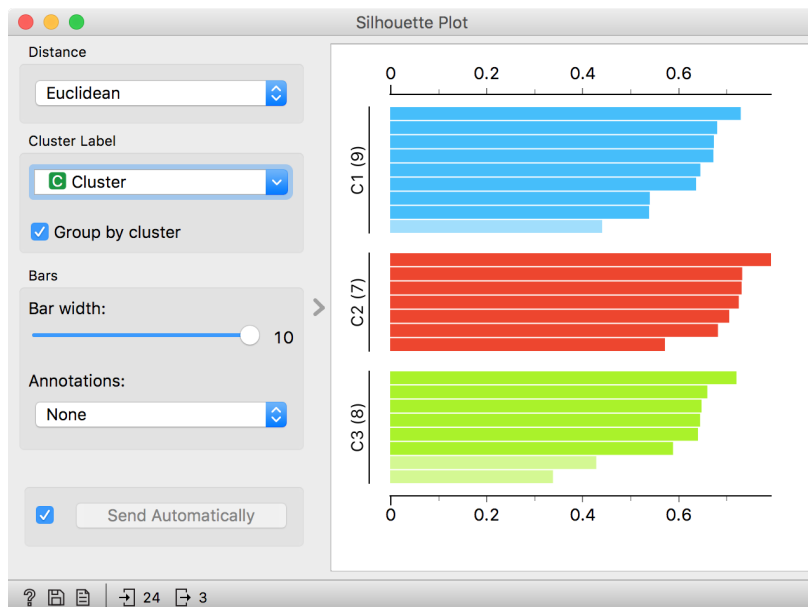
Orange has a *Silhouette Plot* widget that displays the values of the silhouette score for each data instance. We can also choose a particular data instance in the silhouette plot and check out its position in the scatter plot.

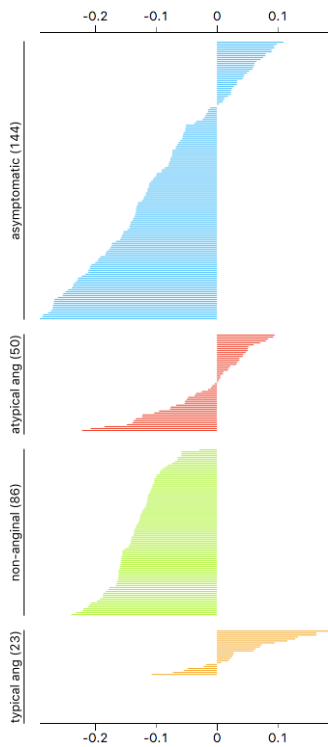


**Average distance B.**  
C3 is the green cluster, and all its points have large silhouettes. Not so for the other two.

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.

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





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 a 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.

Number of Clusters
Silhouette Scores

Fixed:

From  to

Preprocessing

Normalize columns

Initialization

Initialize with

Re-runs:

Maximum iterations:

Apply Automatically

2	0.174
3	0.128
4	0.117
5	0.118
6	0.113
7	0.105
8	0.091

# *k*-Means Clustering

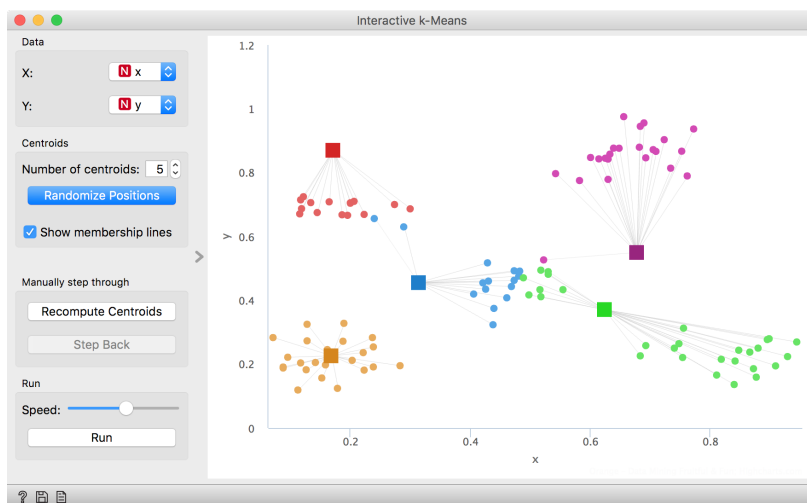
HIERARCHICAL CLUSTERING IS NOT SUITABLE FOR LARGER DATA SETS due to the prohibitive size of the distance matrix: with 30 thousand objects, the distance matrix already has almost one billion elements. An alternative approach that avoids using the distance matrix is k-means clustering.

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 big data sets with millions of data points it usually takes just a couple of ten or hundred iterations.

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

Use the *Paint Data* 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 the plot stops changing. 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.

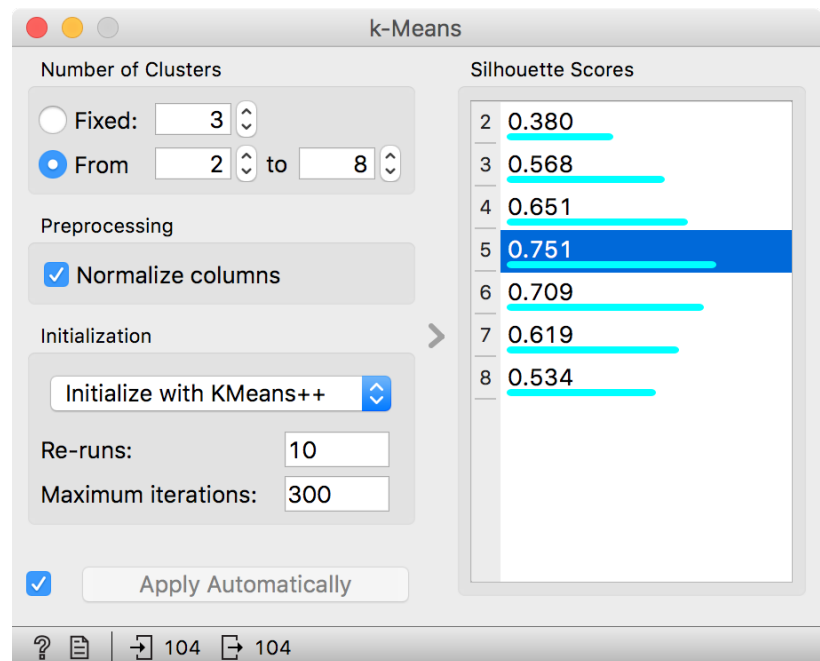
How do we set the initial number of clusters? That's simple: we choose the number that gives the optimal clustering.

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 Data 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 a 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 works really well?

