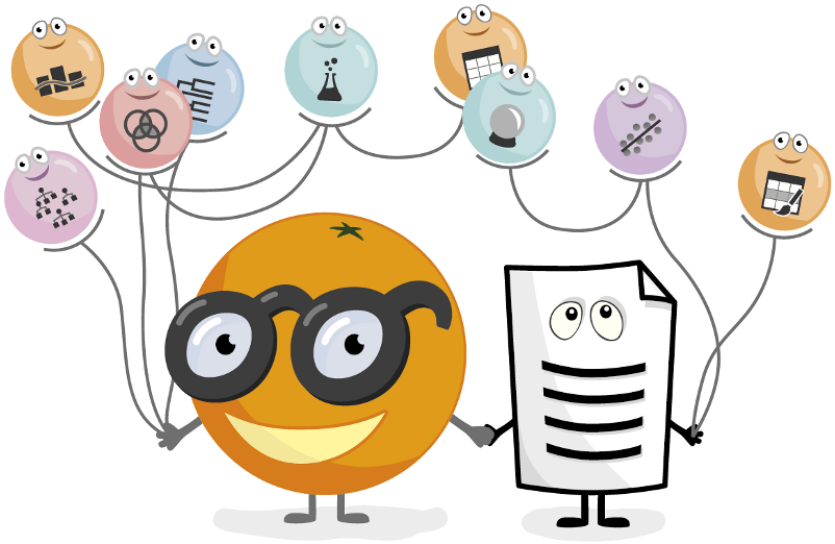


A Gentle Introduction to Data Science

Working notes for the hands-on workshop at GIS Ostrava

These notes include Orange workflows and visualizations we will construct during the course.

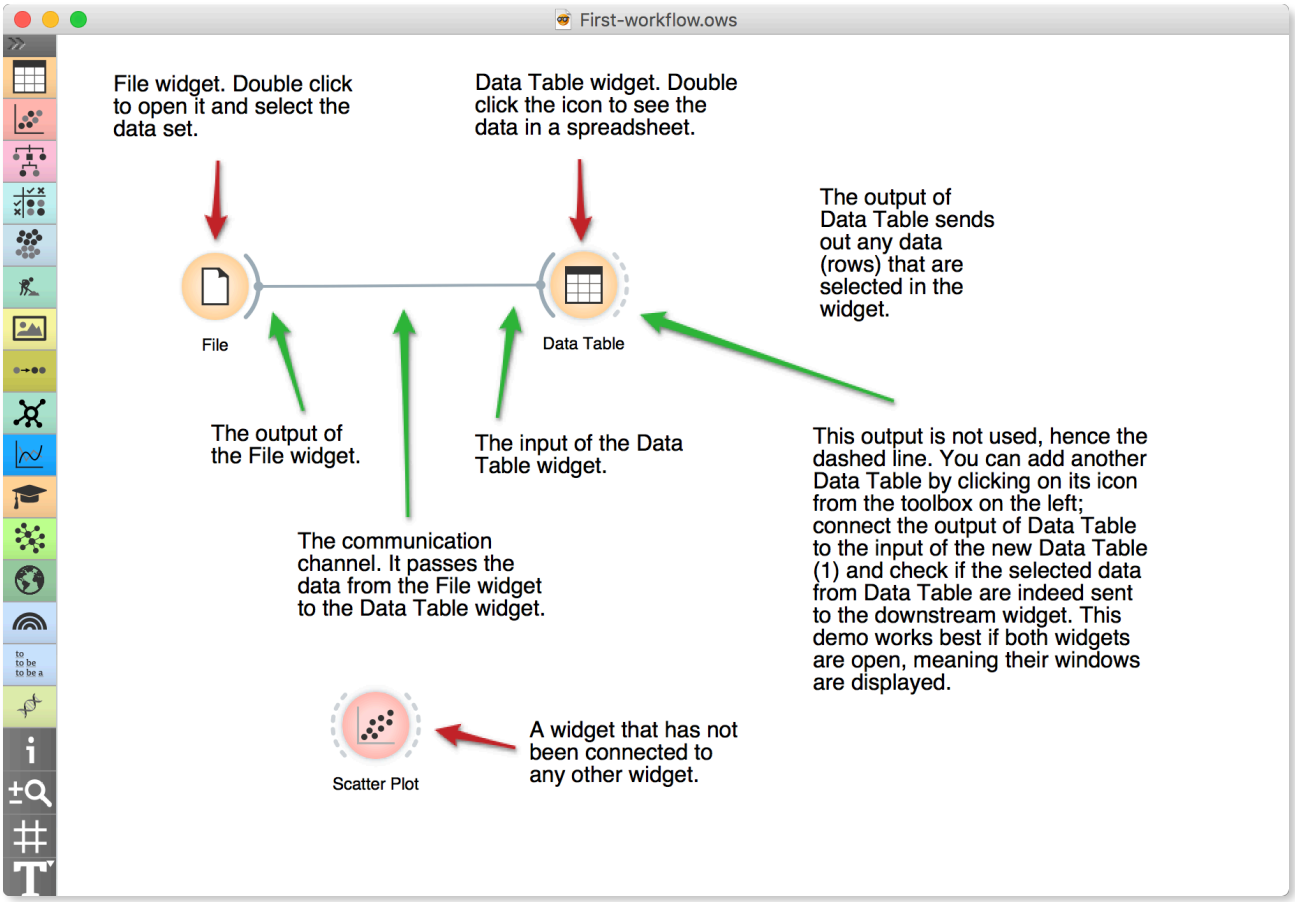
The notes were written by the members of the Bioinformatics Lab at University of Ljubljana, and are extensions of the notes by Blaz Zupan and Janez Demšar.

Welcome to the workshop on Introduction toData Science! This workshop was designed for students, academics, and curious professionals and is carried out by Blaž Zupan. You will see how common data mining tasks can be accomplished without programming. We will use Orange to construct visual data analysis workflows. Many similar data mining environments exist, but the lecturer prefers Orange for one simple reason—he is its co-author.

If you haven't already installed Orange, please download the installation package from http://orange.biolab.si.

Lesson 1: Workflows in Orange

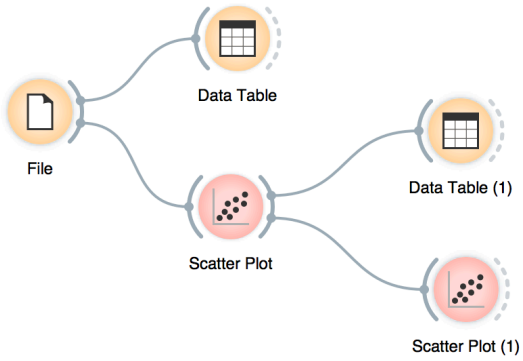
Orange workflows consist of components that read, process and visualize the data. We call them "widgets". Widgets are placed on a drawing board (the "canvas"). Widgets communicate by sending information along a communication channel. Output from one widget is used as input to another.



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.

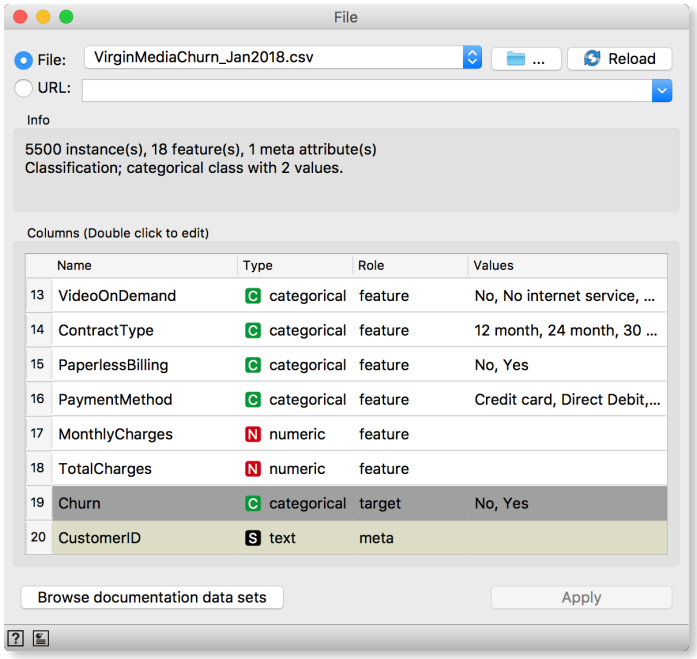
We construct workflows by dragging widgets onto the canvas and connect them by drawing a line from the transmitting widget to the receiving widget. The widget's outputs are on the right and the inputs on the left. In the workflow above, the File widget sends data to the Data Table widget.

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



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

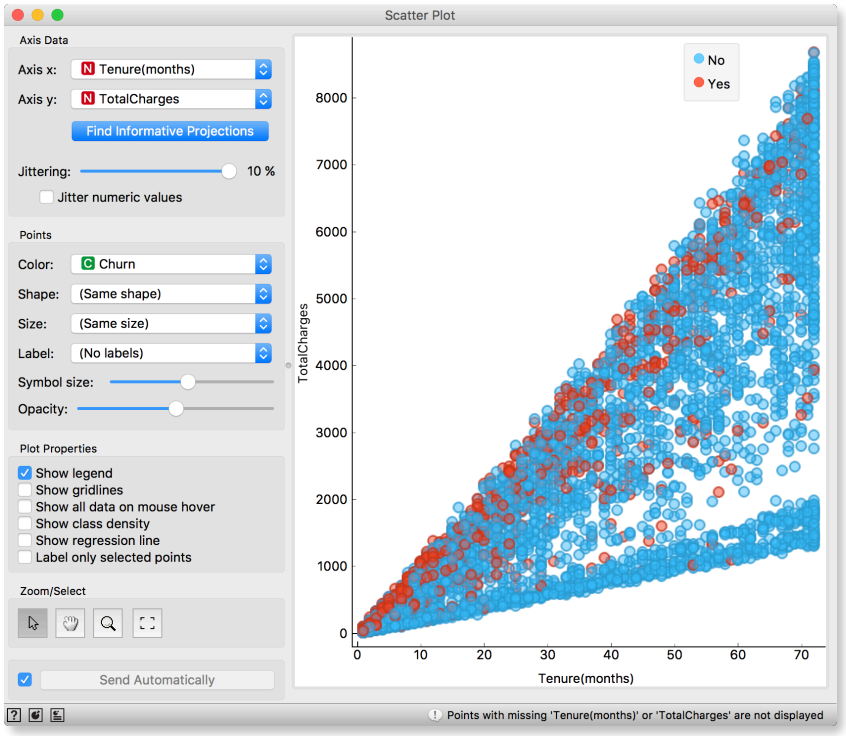
The File widget reads the data from your local disk. Open the File Widget by double clicking its icon. Load the data by clicking the folder icon and selecting *VirginMediaChurn_Jan2018.csv* from the desktop. This data set contains customer records of Virgin Media telephone service company.



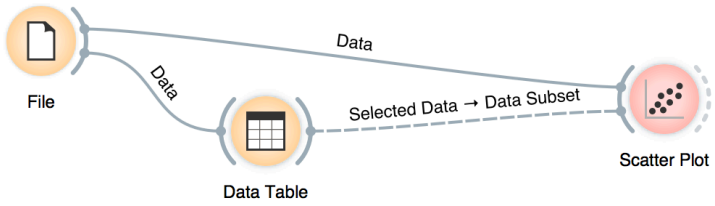
Orange workflows often start with a File widget. The Virgin Media Churn data set has 5500 rows (customers) and 20 columns. Out of the 20 columns, 18 contain information on services, 1 (marked as a "meta attributes") provides additional information on a customer (say ID, since data is anonymised) and 1 (marked as "categorical class") gives information on who stayed with the company and who discontinued the service.

After you load the data, open the other widgets. In the Scatter Plot widget, select a few data points and watch as they appear in widget Data Table (1). Use a combination of two Scatter Plot widgets, where the second scatterplot shows a detail from a smaller region selected in the first scatterplot.

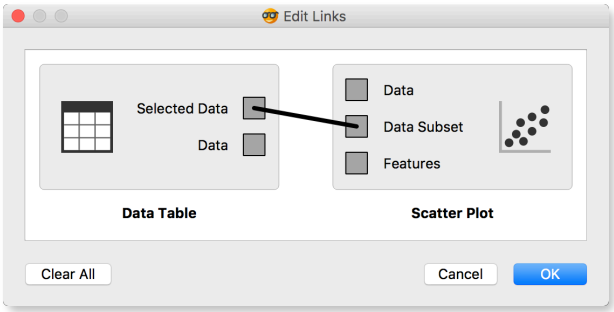
Scatter Plot is best used with numerical features. Display the relationship between Tenure(months) and TotalCharges and colour the points by Churn. What does this visualization tell us?



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



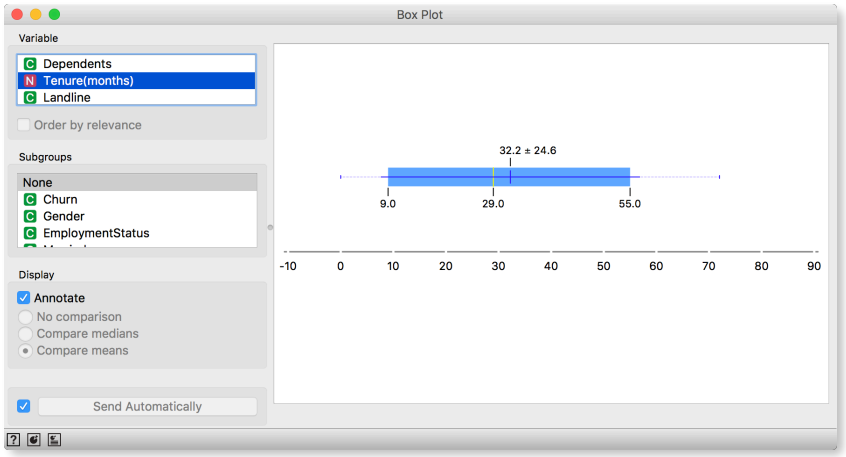
In this workflow, we have switched on the option "Show channel names between widgets" in File→Preferences.



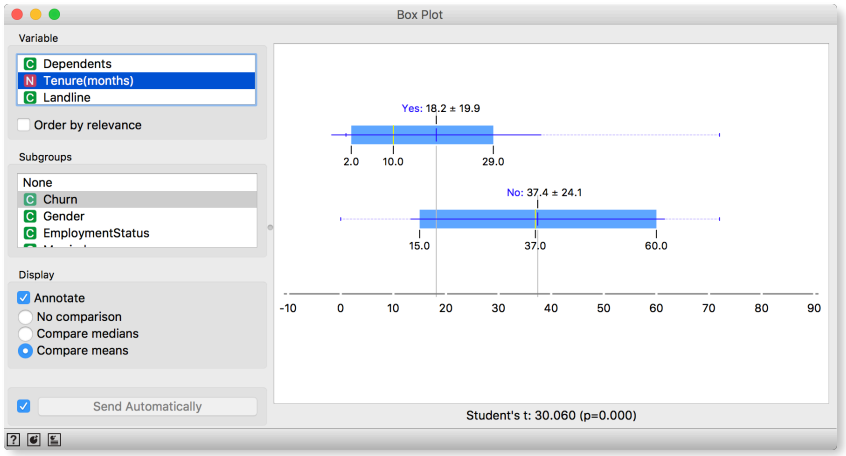
How does Orange distinguish between the primary data source and the data selection? It uses the first connected signal as the entire data set and the second one as its subset. To make changes or to check what is happening under the hood, double click on the line connecting the two widgets.

Lesson 2: Data Exploration

A nice way to get the first impression of our data is to visualize it. Box Plot can help us discover outliers, distributions and interesting features.



Looks like our customers are with the company for 32 months on average, but our standard deviation (thick blue line) is quite high, so we can expect a lot of variance in our samples.

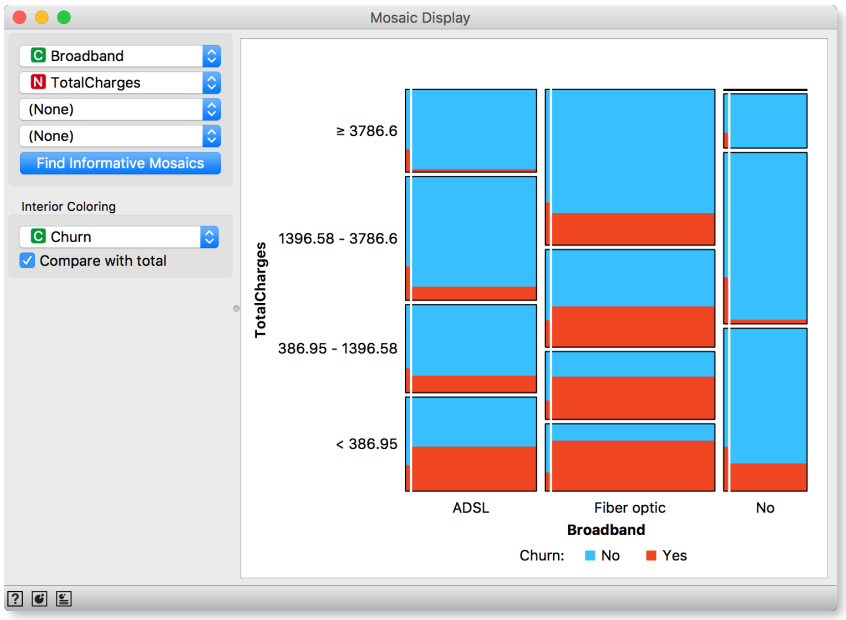


Tick 'Order by relevance' box under Variable list to order variable by how well they split subgroup values. Here, the means of Yes and No are quite apart, but their distributions still overlap.

But Box Plot is even more powerful than that. We can compare distribution by subgroups. Select Churn in Subgroups box and see how those who stay and those who leave differ by the length of tenure. Seems that those who just recently joined us, have a higher tendency to switch the operator (low loyalty).

Another great visualization is Mosaic Display. It shows us frequencies and relationships between a number of features. We have used visualization finder with Find Informative Mosaics button. This function helps us find the most interesting visualization for a given number of features, making our search for feature pairs a lot easier.

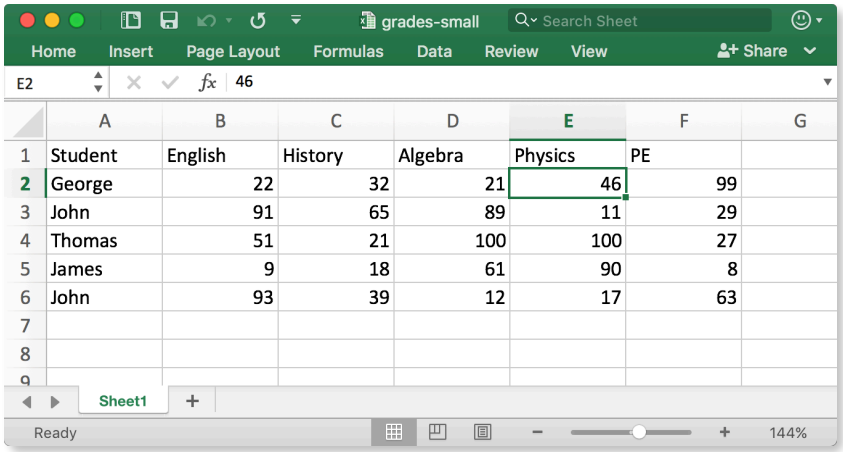
Try using 'Find Informative Mosaics' with 3 or 4 features. The area of each square corresponds to the relative number of instances in the subset.



Mosaic Display shows us relationship between Broadband feature and TotalCharges feature. Looks like those who have broadband option and total charges lower than 3786 pounds, leave more often. Remember, total charges, as we have seen in the scatter plot, is correlated with tenure! Considering this, what does this Mosaic effectively mean?

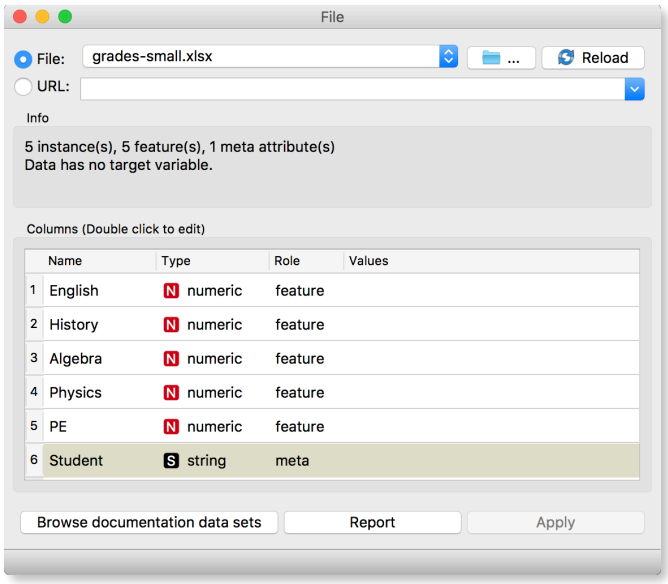
Lesson 3: Loading Your Own Data Set

Orange can read the data from spreadsheet file formats which include tab and comma separated and Excel files. Let us 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							

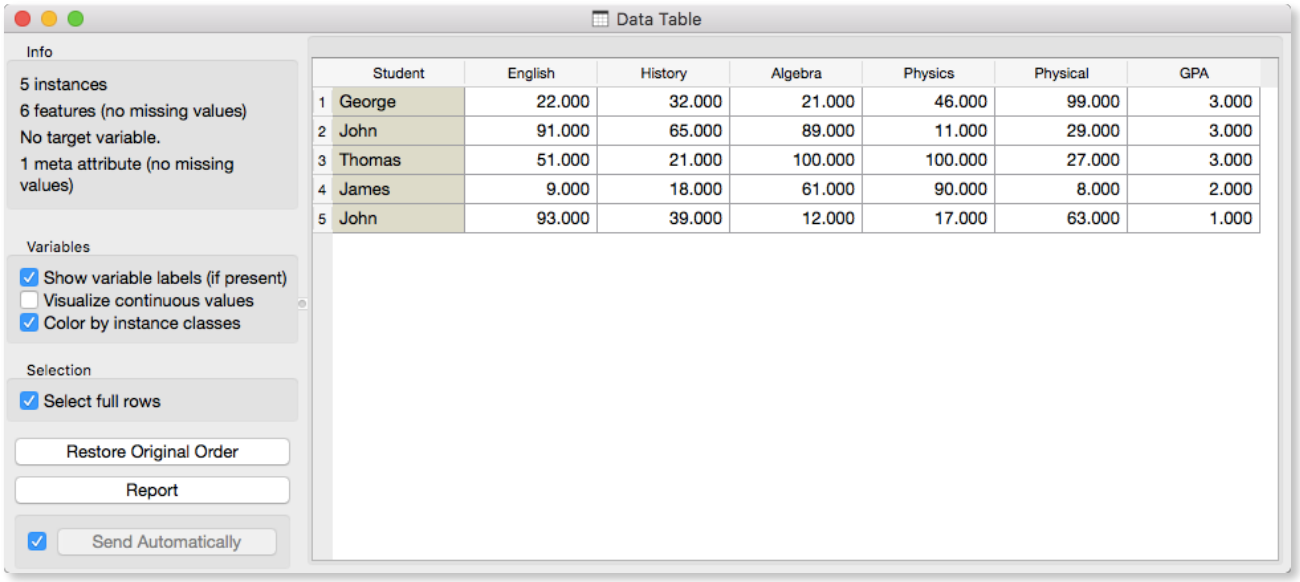
In Orange, we can use the File widget to load this data set.



Looks ok. 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. We can connect our File widget with the Data Table widget,

and double click on the Data Table to see the data in the spreadsheet format.



	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

Nice, everything is here.

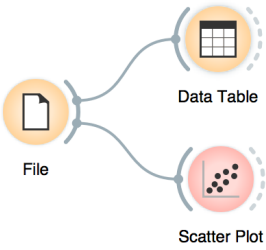
Instead of using Excel, we could also use Google Sheets, a free on-line spreadsheet alternative. Then, instead of finding the file on the local disk, we would enter its URL address in the File widget's URL entry box.

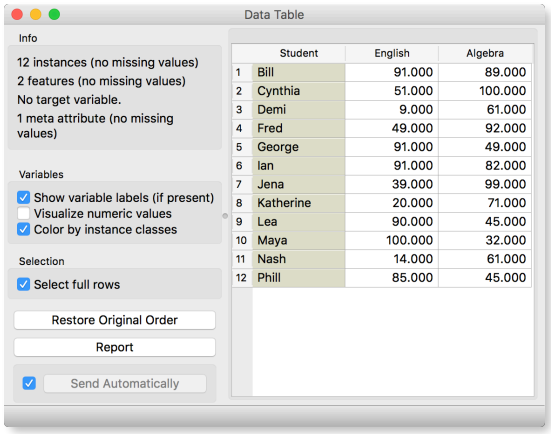
There is more to input data formatting and loading. We can define the type and kind of the data column, specify that the column is actually a web address of an image, and more. But enough for now. If you would really like to dive in for more, check out the documentation page on Loading your Data, or a video on this subject.

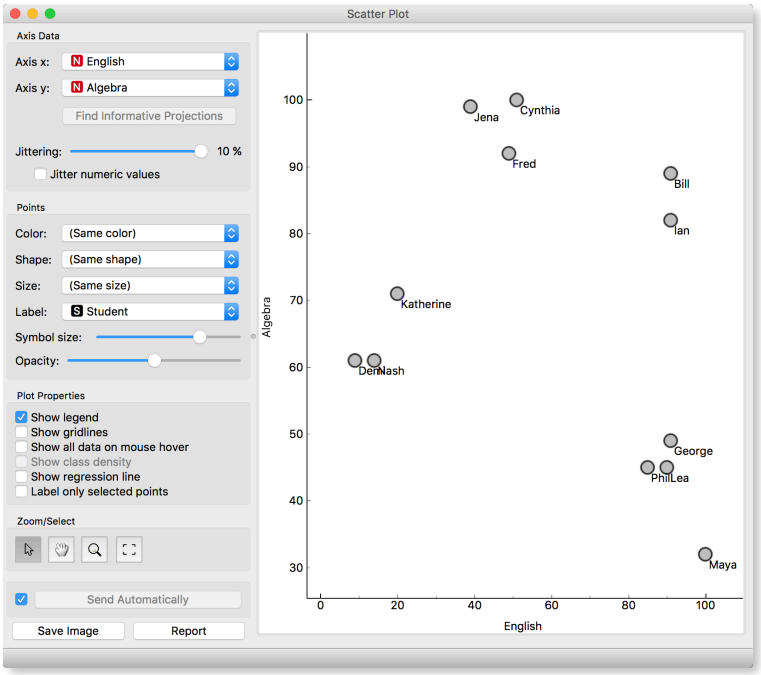
Lesson 4: Hierarchical Clustering

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.biolab.si/text/grades.tab.

Say that we are interested in finding clusters in our 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?



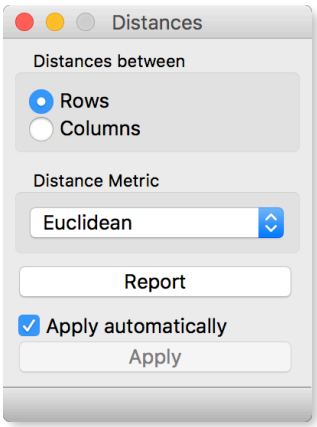




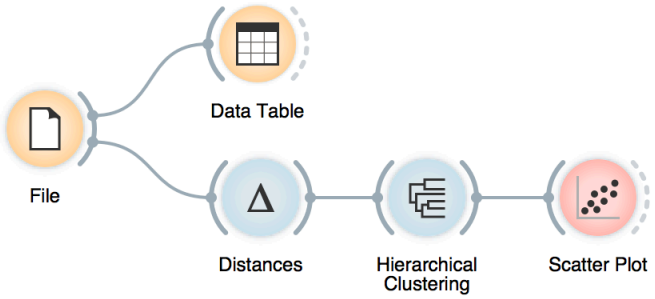
There are different ways to measure the similarity between clusters. The estimate we have described is called average linkage. We could also estimate the distance through the two closest points in each clusters (single linkage), or through the two points that are furthest away (complete linkage).

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.

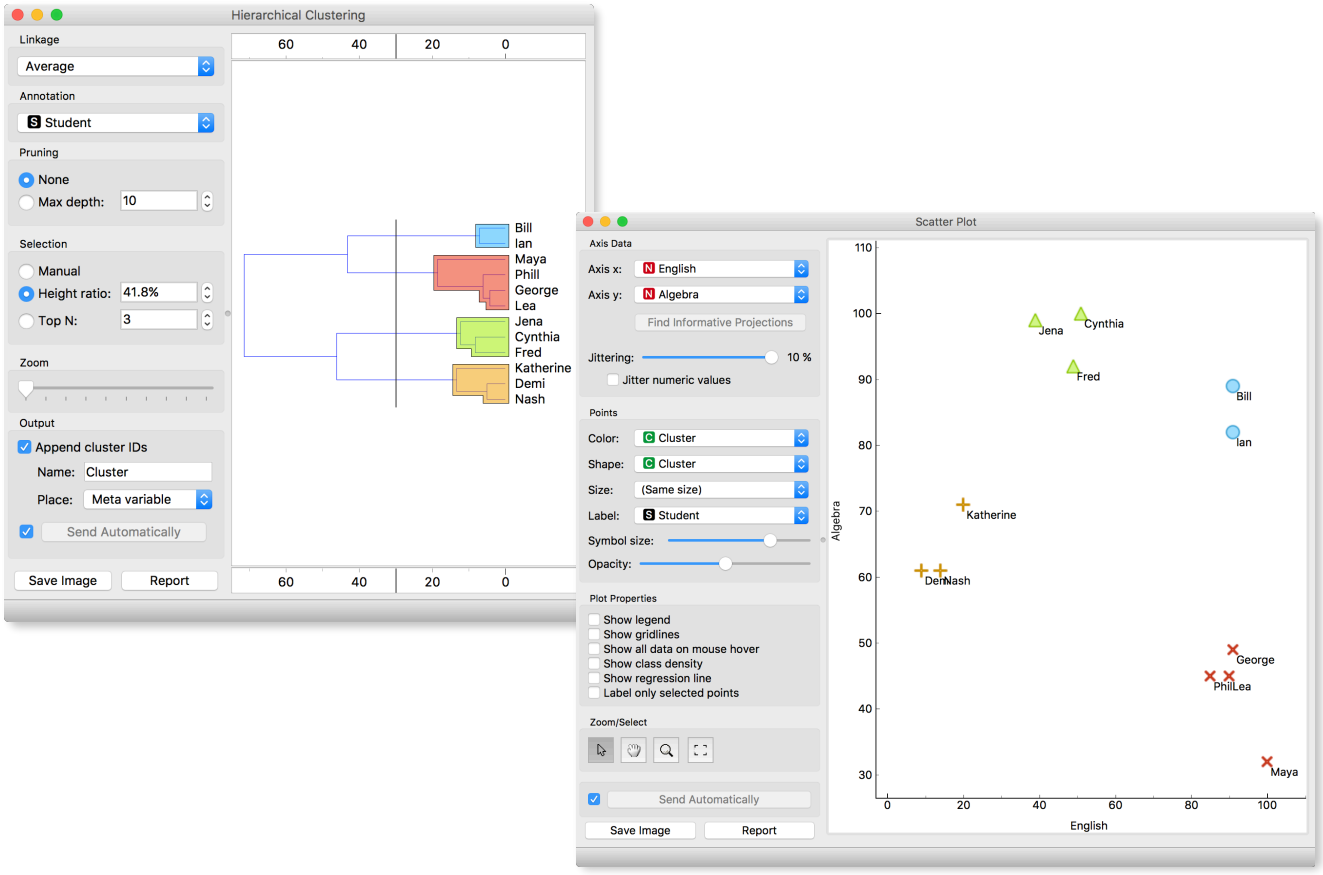
Next, we need to define a clustering algorithm. Say that we start with each data instance being its own cluster, and then, at each step, we join the clusters that are closest together. We estimate the distance between the clusters with, say, the average distance between all their pairs of data points.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



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



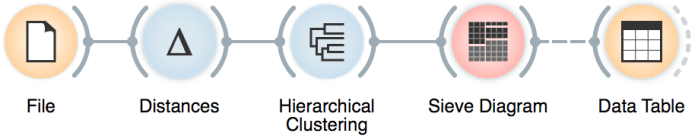
Couldn't be simpler. Load the data, measure the distances, use them in hierarchical clustering, and visualize the results in the scatter plot. Hierarchical clustering widget allows us to cut the hierarchy at a certain distance score and output the corresponding clusters:

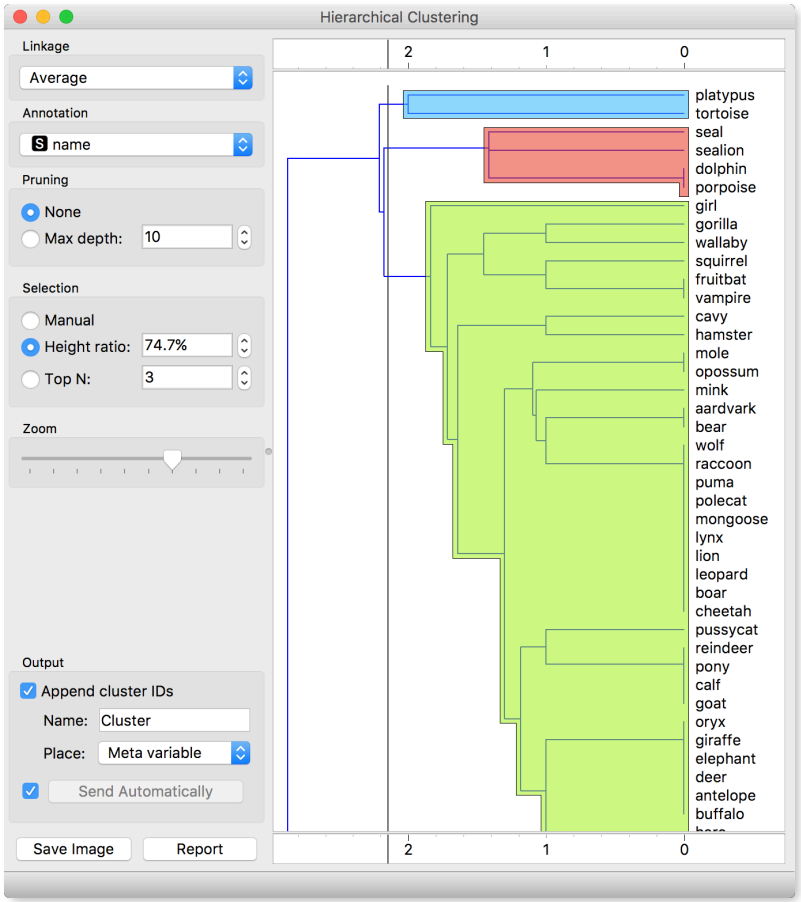


Lesson 5: Animal Kingdom

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 in our biology classes. Luckily, there's data mining and the idea that taxonomy simply stems from measuring the distance between species.

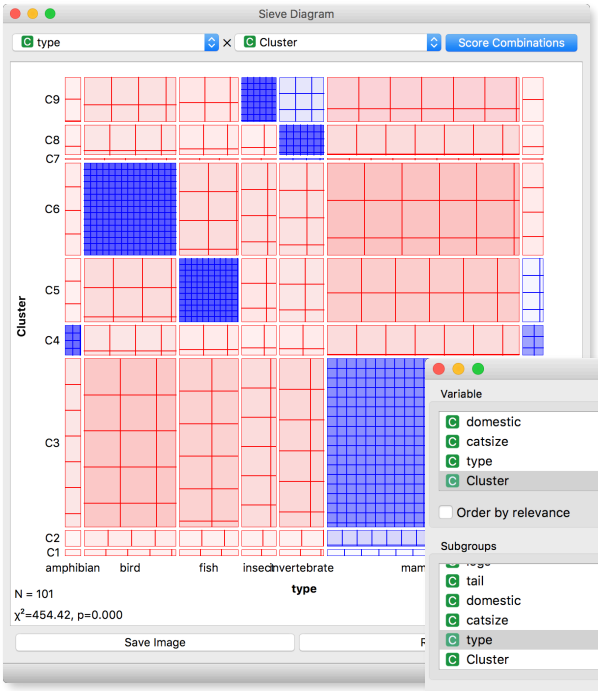
Hierarchical clustering works fast for smaller data sets. But for bigger ones it fails. Simply, it cannot be used. Why?





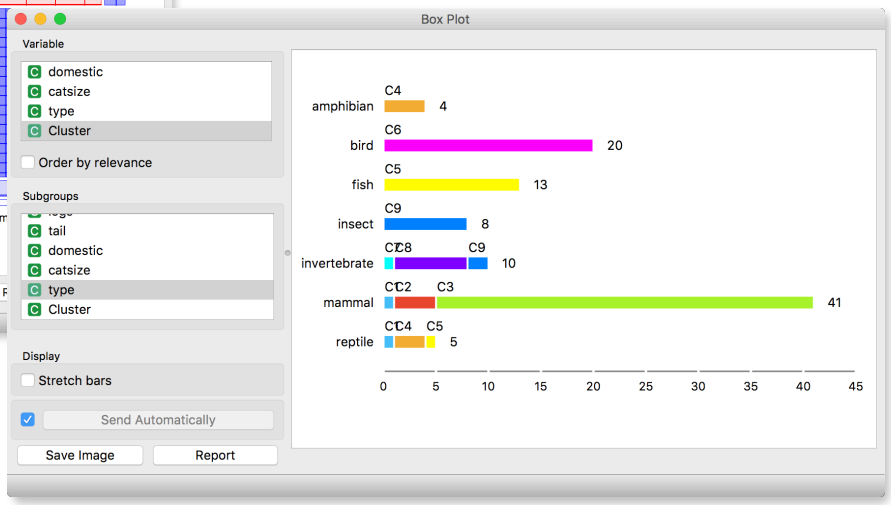
Here we use zoo data (from documentation data sets) with attributes that report on various features of animals (has hair, has feathers, lays eggs). We measure the distance and compute the clustering. Animals in this data set are annotated with type (mammal, insect, bird, and so on). It would be cool 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?



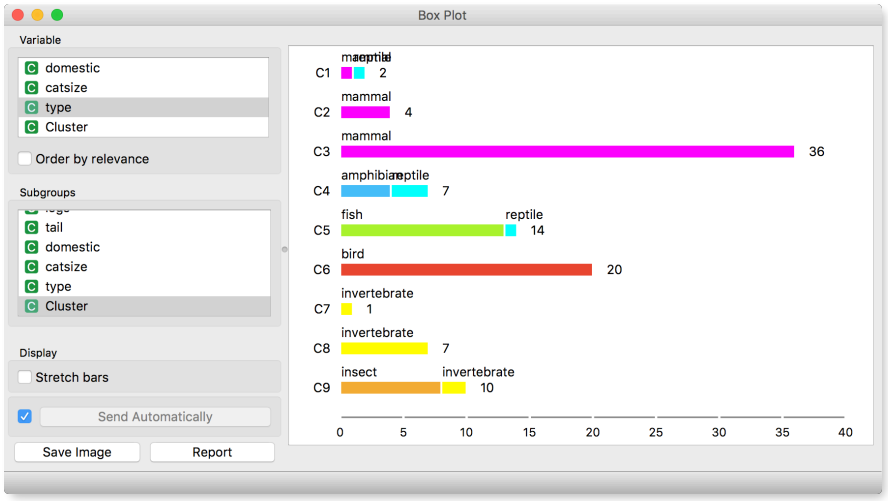
We can observe the discovered clusters in a Sieve Diagram. Birds, say, are in cluster C6. Cluster C4 consists of amphibians and some reptiles. And so forth.

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



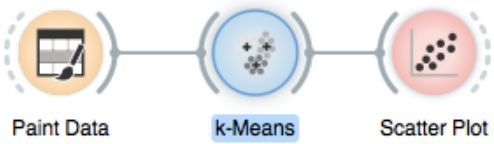
Or we can turn it around and see how different types of animals are spread across clusters.

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. Click on the clusters in the box plot and discover who they are.

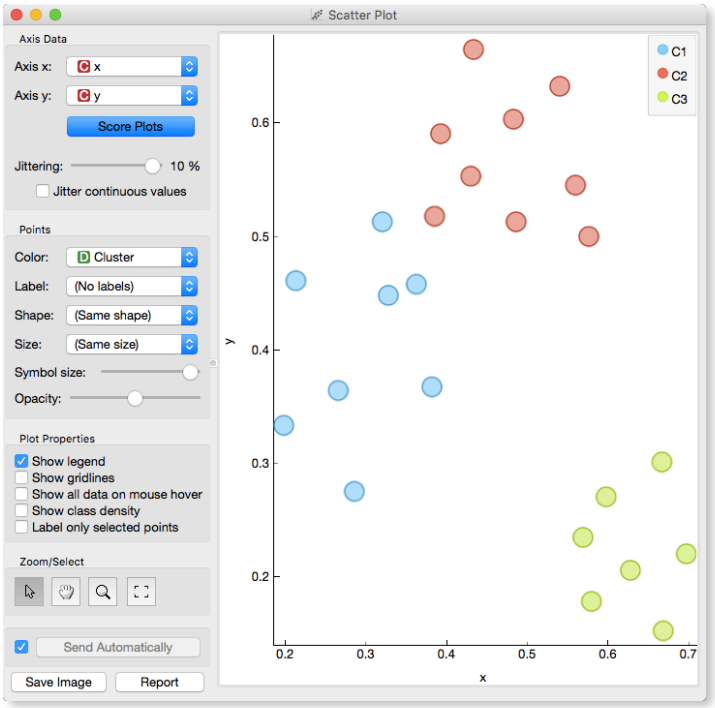


Lesson 6: Silhouettes

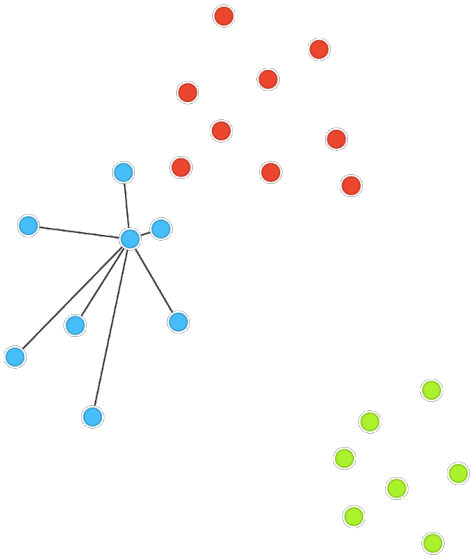
Don't get confused: we paint data and/or visualize it with Scatter plots, which show only two features. This is just for an illustration! Most data sets contain many features and methods like k-Means clustering take into account all features, not just two.



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 scatterplot.

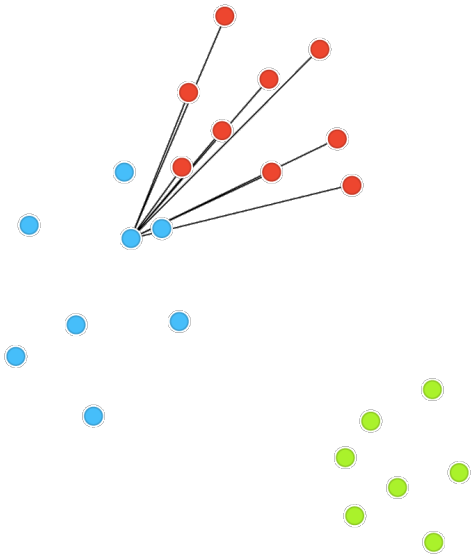


The data points in the green cluster are well separated from those in the other two. Not so for the blue and red points, where several points are on the border between the clusters. We would like to quantify the degree of how well a data point belongs to the cluster to which it is assigned.



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.



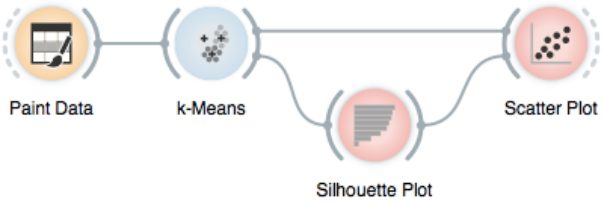
C3 is the green cluster, and all its points have large silhouettes. Not so for the other two.

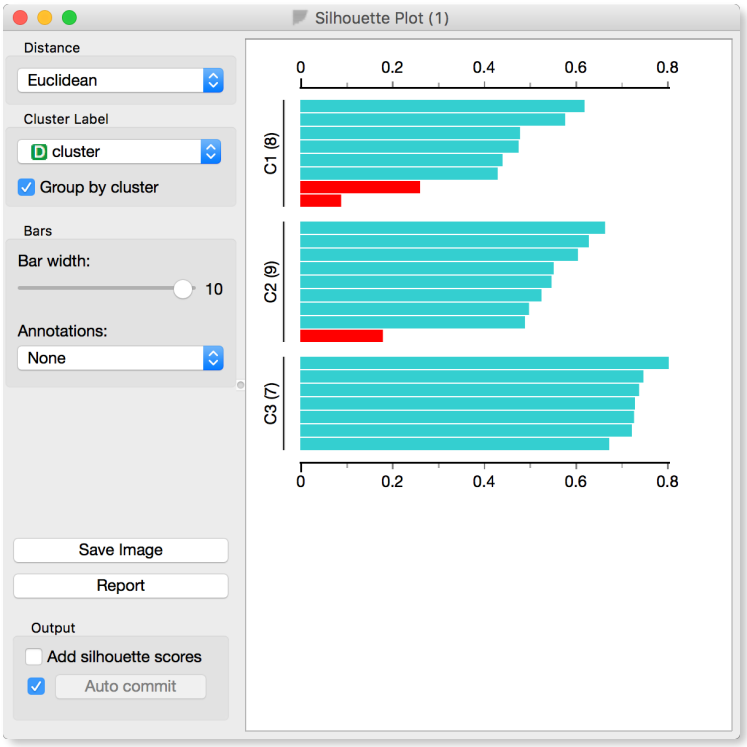
Below we selected three data instances with the worst silhouette scores. Can you guess where the lie in the scatter plot?

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, $S = (B - A) / \max\{A, B\}$. Voilá, S is our silhouette score.

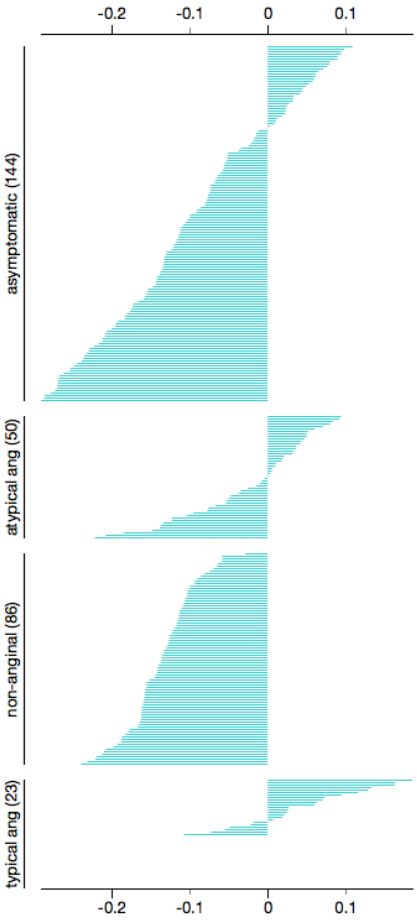
Orange has a Silhouette Plot widget that displays the values of the silhouette score for each data instance. We can also choose a particular data instance in the silhouette plot and check out its position in the scatter plot.





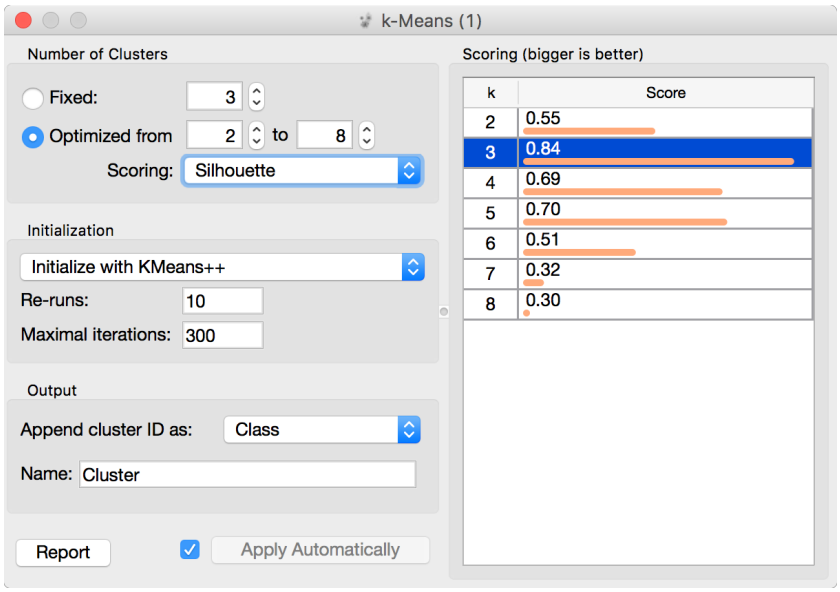
This of course looks great for data sets with two features, where the scatter plot reveals all the information. In higher-dimensional data, the scatter plot shows just two features at a time, so two points that seem close in the scatter plot may be actually far apart when all features - perhaps thousands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



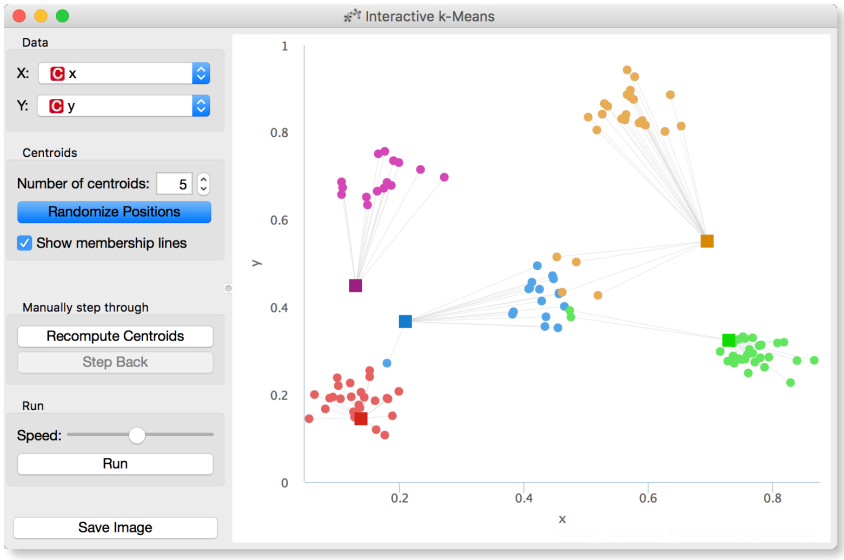
Lesson 7: k-Means Clustering

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?

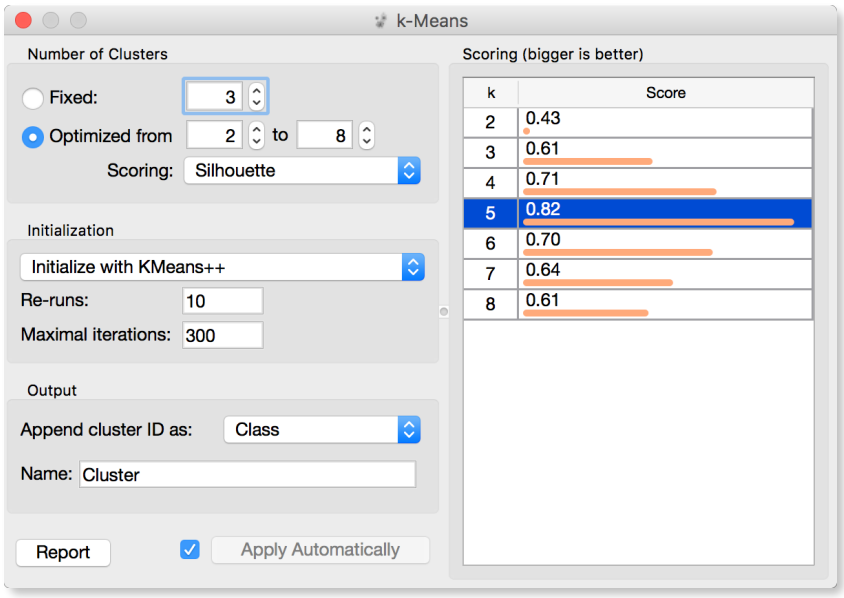
Keep pressing Recompute Centroids and Reassign Membership until it stops changes. With this simple, two-dimensional data it will take just a few iterations; with more points and features, it can take longer, but the principle is the same.

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.

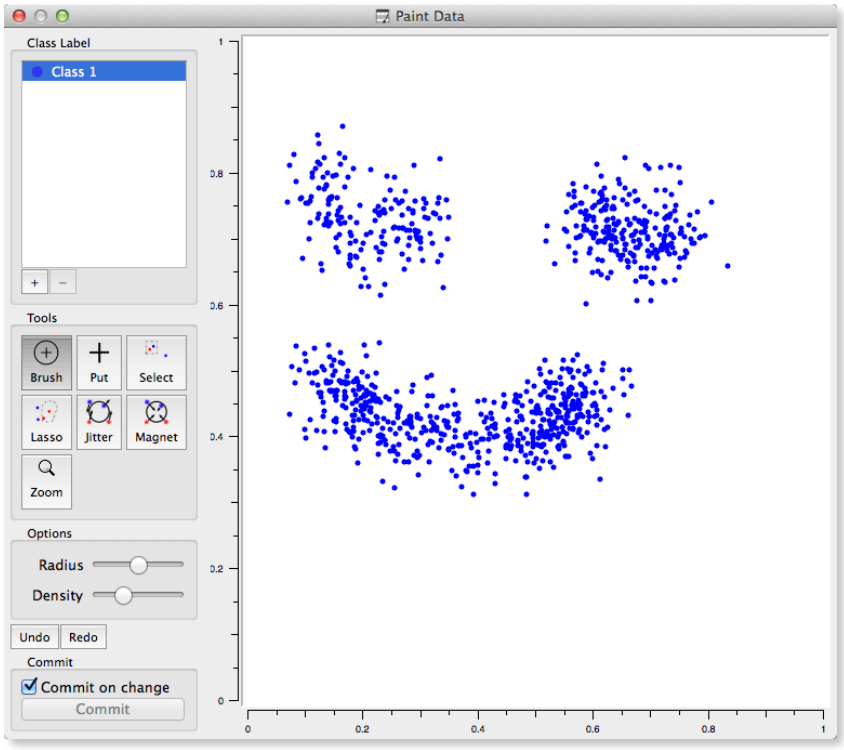


Works like charm.

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-

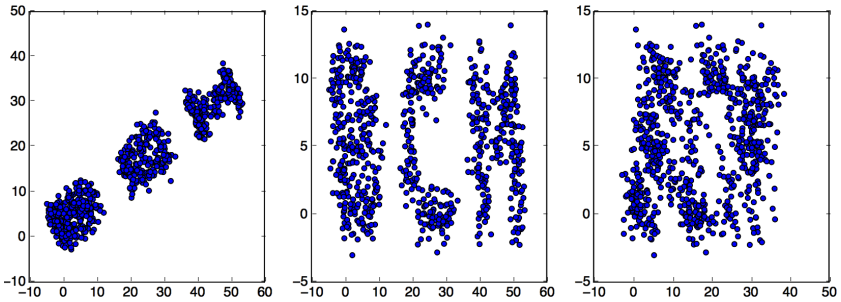
running the clustering multiple times from random positions and using the best result. Second, the silhouette sometimes fails to correctly evaluate the clustering. Nobody's perfect.

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?

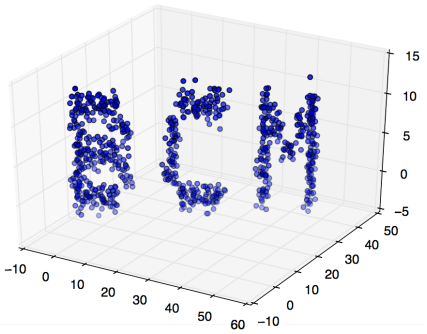


Lesson 8: Principal Component Analysis

Which of the following three scatterplots (showing x vs. y, x vs. z and y vs. z) for the same three-dimensional data gives us the best picture about the actual layout of the data in space?



Yes, the first scatter plot looks very useful: it tells us that x and y are highly correlated and that we have three clusters of somewhat irregular shape. But remember: this data is three dimensional. What is we saw it from another, perhaps better perspective?



Let's make another experiment. Go to https://in-the-sky.org/ngc3d.php, disable Auto-rotate and Show labels and select Zoom to show Local Milky Way. Now let's rotate the picture of the galaxy to find the layout of the stars.

Think about what we've done. What are the properties of the best projection?

We want the data to be as spread out as possible. If we look from the direction parallel to the galactic plane, we see just a line. We lose one dimension, essentially keeping just a single coordinate for each star. (This is unfortunately exactly the perspective we see on the night sky: most stars are in the bright band we call the milky way, and we only see the outliers.) Among all possible projections, we attempt to find the one with the highest spread across the scatter plot. This projection may not be (and usually isn't) orthogonal to any axis; it may be projection to an arbitrary plane.

We again talk about two dimensional projection only for the sake of illustration. Imagine that we have ten thousand dimensional data and we would like, for some reason, keep just ten features. Yes, we can rank the features and keep the most informative, but

what if these are correlated and tell us the same thing? Or what if our data does not have any target variable: with what should the "good features" be correlated? And what if the optimal projection is not aligned with the axes at all, so "good" features are combinations of the original ones?

We can do the same reasoning as above: we want to find a 10-dimensional (for the sake of examples) projection in which the data points are as spread as possible.

How do we do this? Let's go back to our everyday's three dimensional world and think about how to find a two-dimensional projection.

Imagine you are observing a swarm of flies; your data are their exact coordinates in the room, so the position of each fly is described by three numbers. Then you discover that your flies actually fly in a formation: they are (almost) on the same line. You could then describe the position of each fly with a single number that represents the fly's position along the line. Plus, you need to know where in the space the line lies. We call this line the first principal component. By using it, we reduce the three-dimensional space into a single dimension.

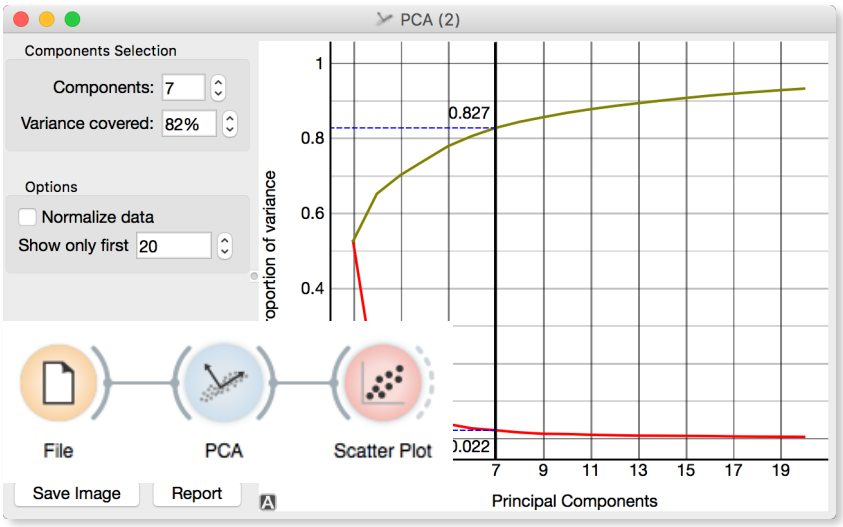
After some careful observation, you notice the flies are a bit spread in one other direction, so they do not fly along a line but along a band. Therefore, we need two numbers, one along the first and one along the — you guessed it — second principal component.

It turns out the flies are actually also spread in the third direction. Thus you need three numbers after all.

Or do you? It all depends on how spread they are in the second and in the third direction. If the spread along the second is relatively small in comparison with the first, you are fine with a single dimension. If not, you need two, but perhaps still not three.

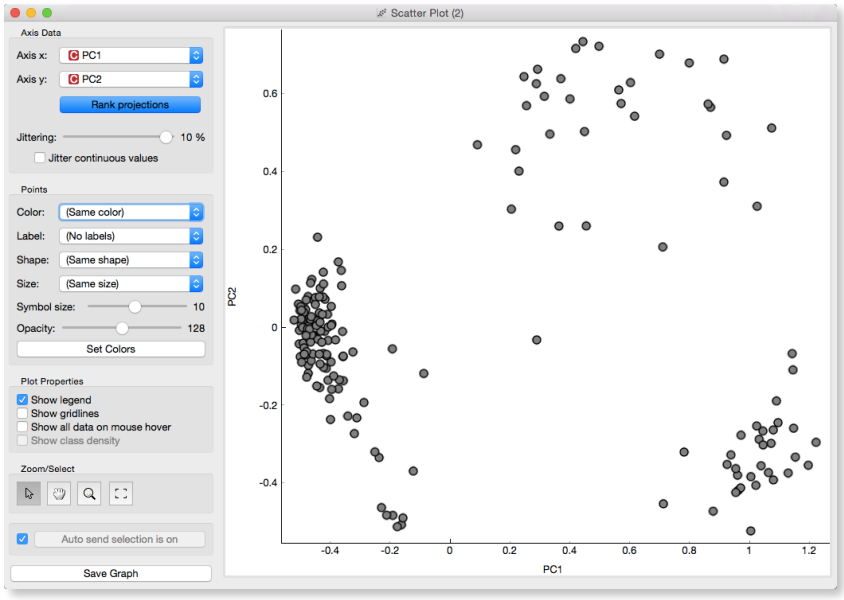
A technique that does this is called Principle Components Analysis, or PCA. The corresponding widget is simple: it receives the data and outputs the transformed data.

The widget allows you to select the number of components and helps you by showing how much information (technically: explained variance) you retain with respect to the number of



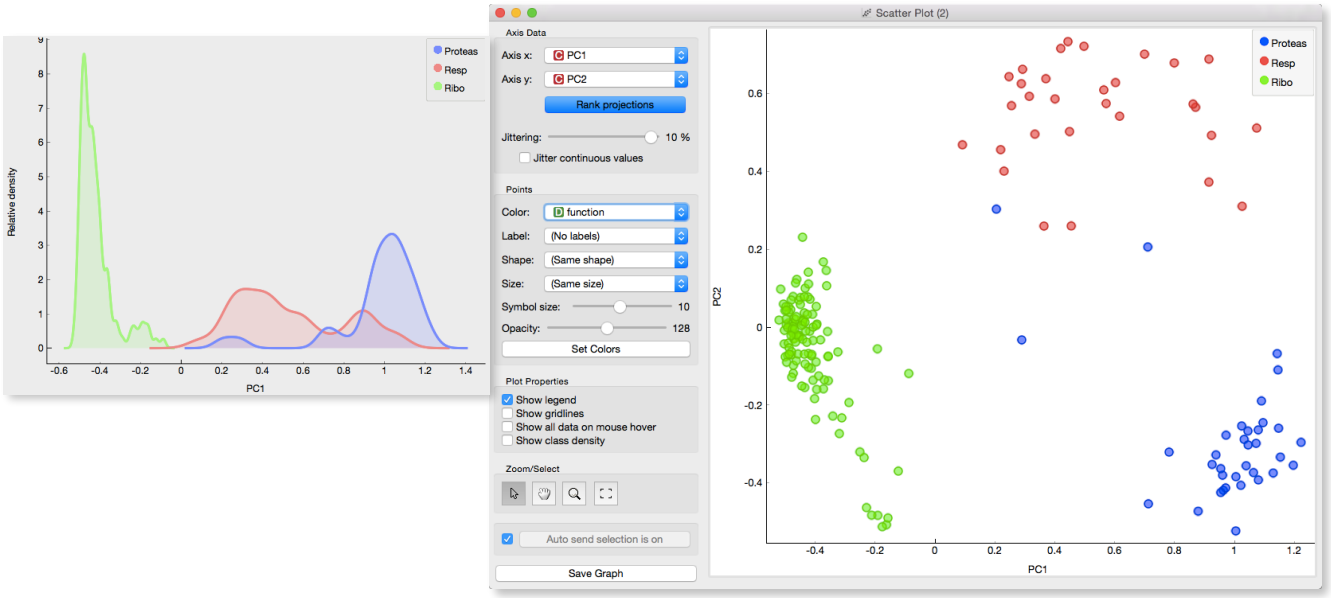
components (brownish line) and the amount of information (explained variance) in each component.

The PCA on the left shows the scree diagram for brown-selected data. Set like this, the widget replaces the 80 features with just seven - and still keeping 82.7% of information. (Note: disable "Normalize data" checkbox to get the same picture.) Let us see a scatter plot for the first two components.



The axes, PC1 and PC2, do not correspond to particular features in the original data, but to their linear combination. What we are looking at is a projection onto the plane, defined by the first two components. When you consider only two components, you can imagine that PCA put a hyperplane into multidimensional space and projecting all data into it.

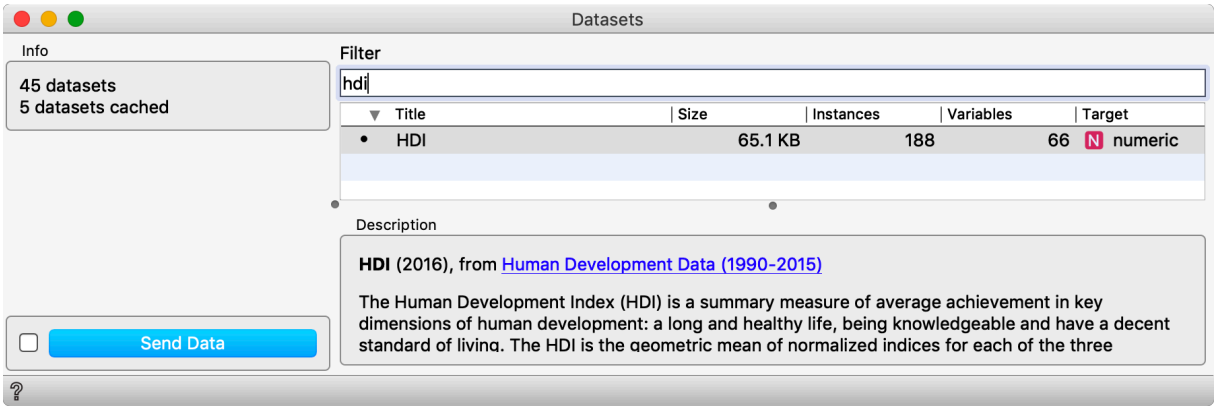
Note that this is an unsupervised method: it does not care about the class. The classes in the projection may be be well separated or not. Let's add some colors to the points and see how lucky we are this time.



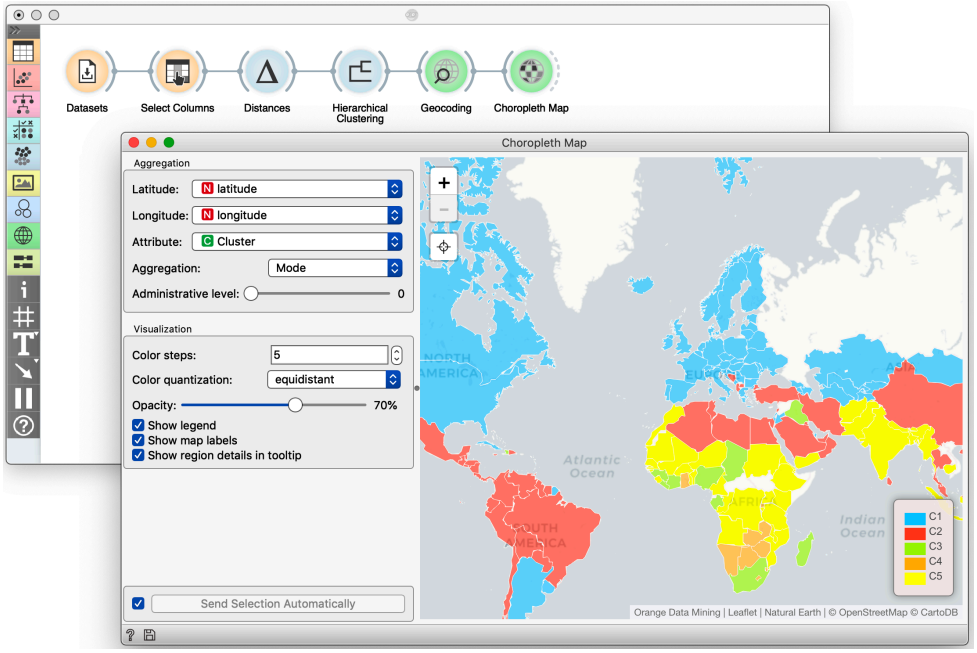
The data separated so well that these two dimensions alone may suffice for building a good classifier. No, wait, it gets even better. The data classes are separated well even along the first component.

Lesson 9: Human Development Index

The HDI was created by United Nations to emphasize that people and their capabilities should be the ultimate criteria for assessing the development of a country, not economic growth alone. We will here use the techniques that we have learned so far to analyze the data on 188 countries from 1990 to 2015. The data is readily available from Datasets widget.



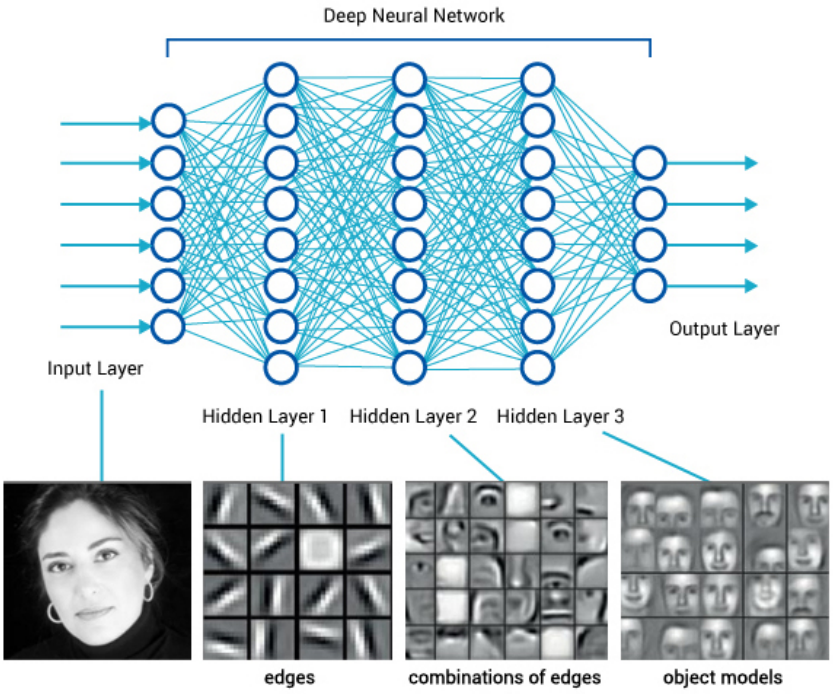
We would like to select a subset of indicators and clusters the countries accordingly. And then we would like to see the clusters on the world map. Here, you will be mostly on your own, but could construct a workflow similar to the one below. Also shown are examples of country clusters as projected on the map.



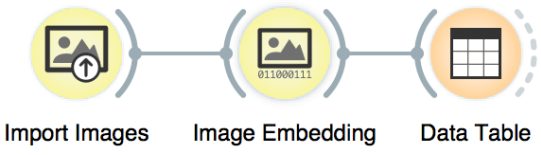
Lesson 10: Image Embedding

Every data set so far came in the matrix (tabular) form: objects (say, countries, students, flowers) were described by row vectors representing a number of features. Not all the data is like this; think about collections of text articles, stock market data sequences, voice recordings or images. It would be great if we could represent them in the same matrix format we have used so far. We would turn collections of, say, images, into matrices and explore them with the familiar prediction or clustering techniques.

This depiction of deep learning network was borrowed from



Until very recently, finding useful representation of complex objects such as images was a real pain. Now, technology called deep learning is used to develop models that transform complex objects to vectors of numbers. Consider images. When we, humans, see an image, our neural networks go from pixels, to spots, to patches, and to some higher order representations like squares, triangles, frames, all the way to representation of complex objects. Artificial neural networks used for deep learning emulate these through layers of computational units (essentially, logistic regression models and some other stuff we will ignore here). If we put an image to an input of such a network and collect the outputs from the higher levels, we get vectors containing an abstraction of the image. This is called embedding.



Deep learning requires a lot of data (thousands, possibly millions of data instances) and processing power to prepare the network. We will use one which is already prepared. Even so, embedding takes time, so Orange doesn't do it locally but uses a server invoked through the Image Embedding widget.

Image embedding describes the images with a set of 2048 features appended to the table with meta features of images.

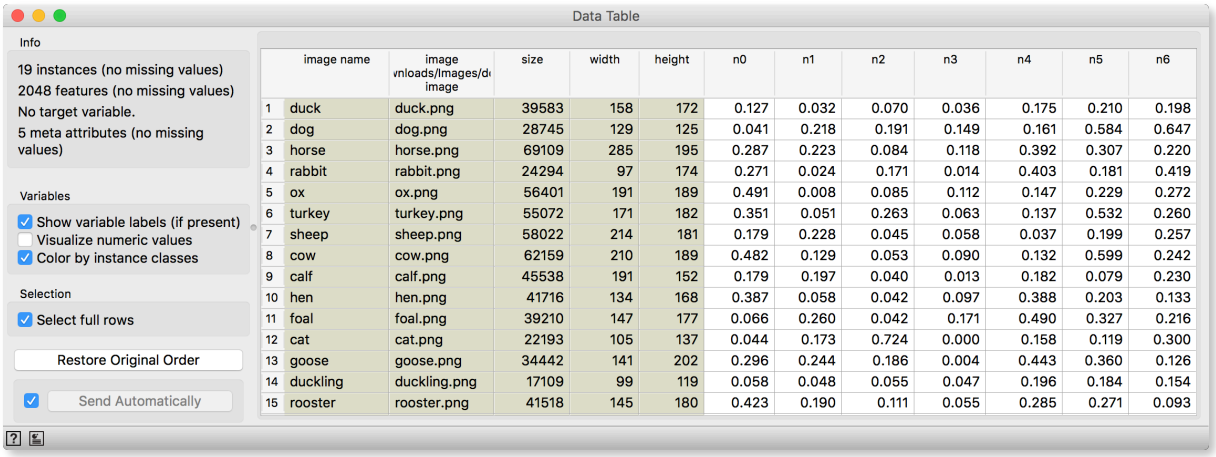
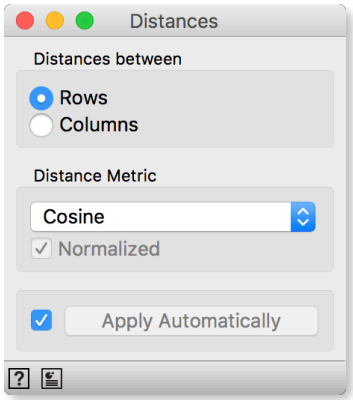
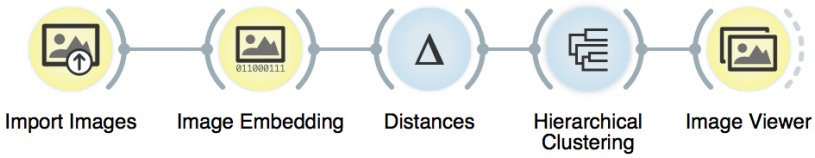


	image name	image vnloads/Images/d image	size	width	height	n0	n1	n2	n3	n4	n5	n6
1	duck	duck.png	39583	158	172	0.127	0.032	0.070	0.036	0.175	0.210	0.198
2	dog	dog.png	28745	129	125	0.041	0.218	0.191	0.149	0.161	0.584	0.647
3	horse	horse.png	69109	285	195	0.287	0.223	0.084	0.118	0.392	0.307	0.220
4	rabbit	rabbit.png	24294	97	174	0.271	0.024	0.171	0.014	0.403	0.181	0.419
5	ox	ox.png	56401	191	189	0.491	0.008	0.085	0.112	0.147	0.229	0.272
6	turkey	turkey.png	55072	171	182	0.351	0.051	0.263	0.063	0.137	0.532	0.260
7	sheep	sheep.png	58022	214	181	0.179	0.228	0.045	0.058	0.037	0.199	0.257
8	cow	cow.png	62159	210	189	0.482	0.129	0.053	0.090	0.132	0.599	0.242
9	calf	calf.png	45538	191	152	0.179	0.197	0.040	0.013	0.182	0.079	0.230
10	hen	hen.png	41716	134	168	0.387	0.058	0.042	0.097	0.388	0.203	0.133
11	foal	foal.png	39210	147	177	0.066	0.260	0.042	0.171	0.490	0.327	0.216
12	cat	cat.png	22193	105	137	0.044	0.173	0.724	0.000	0.158	0.119	0.300
13	goose	goose.png	34442	141	202	0.296	0.244	0.186	0.004	0.443	0.360	0.126
14	duckling	duckling.png	17109	99	119	0.058	0.048	0.055	0.047	0.196	0.184	0.154
15	rooster	rooster.png	41518	145	180	0.423	0.190	0.111	0.055	0.285	0.271	0.093

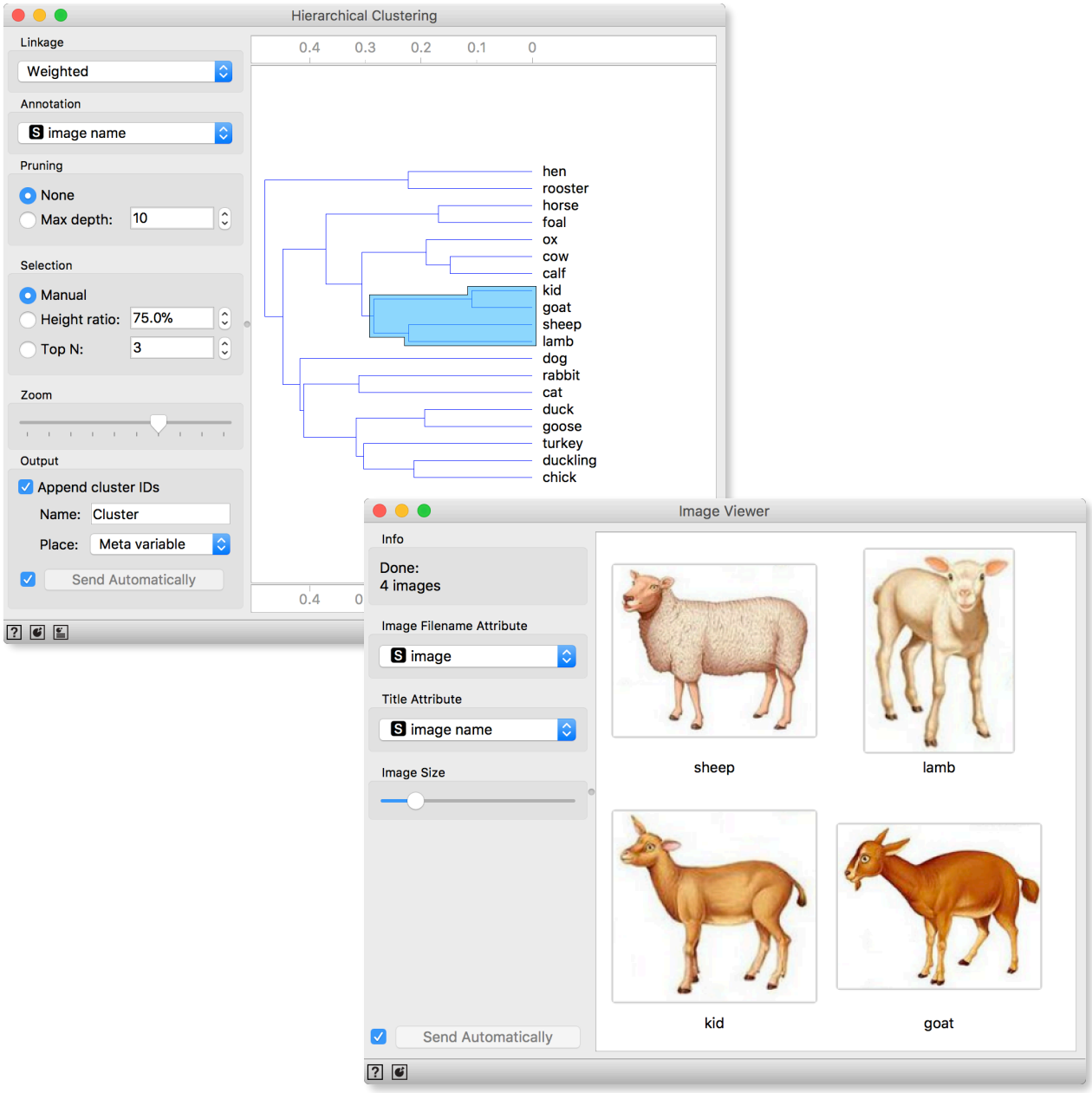
We have no idea what these features are, except that they represent some higher-abstraction concepts in the deep neural network (ok, this is not very helpful in terms of interpretation). Yet, we have just described images with vectors that we can compare and measure their similarities and distances. Distances? Right, we could do clustering. Let's cluster the images of animals and see what happens.





To recap: in the workflow about we have loaded the images from the local disk, turned them into numbers, computed the distance matrix containing distances between all pairs of images, used the distances for hierarchical clustering, and displayed the images that correspond to the selected branch of the dendrogram in the image viewer. We used cosine similarity to assess the distances (simply because of the dendrogram looked better than with the Euclidean distance).

Even the lecturers of this course were surprised at the result. Beautiful!



For the End

The Introduction to Data Science workshop ends here. We covered quite a lot of topics and hope we have taught you some crucial algorithms that should be on the stack of every data scientists. The goal was not to turn you into one, but to get you familiar with some basic techniques, tools and concepts. Data science is a huge field and it takes years of study and practice to master it. You may never become a data scientist, but as an expert in your field it should now be easier to talk and collaborate with statisticians and computer scientists. And for those who want to go ahead with data science, well, now you know where to start.