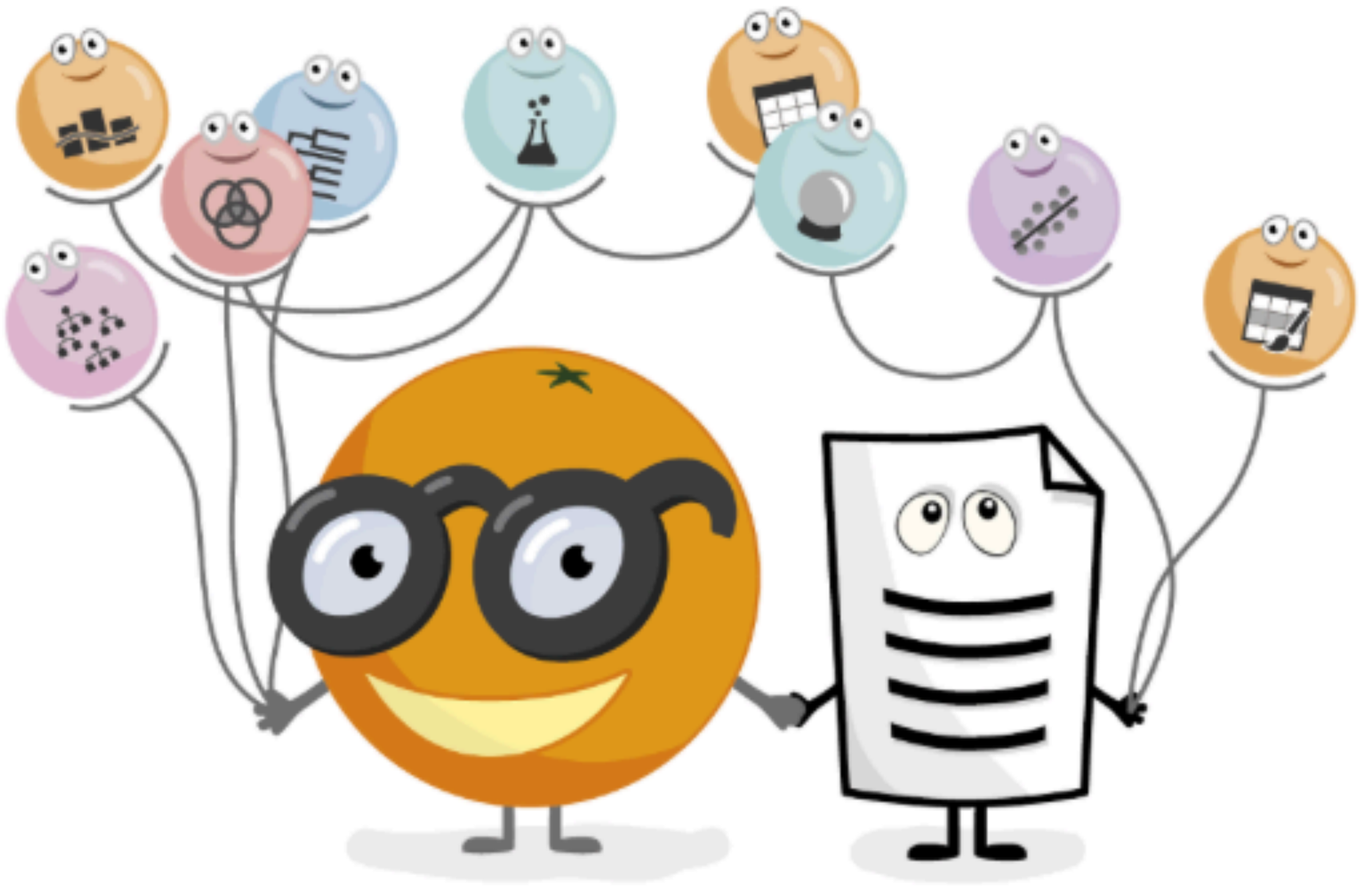
Orange Widgets



Data

File	Data Sets	SQL Table	Save Data	Data Info	Data Table	Select Columns
Select Rows	Data Sampler	Transpose	Discretize	Continuize	Create Class	Randomize
Concatenate	Paint Data	Python Script	Feature Constructor	Edit Domain	Image Viewer	Impute
Merge Data	Outliers	Preprocess	Purge Domain	Rank	Color	

Color

Set color legend for variables.

Signals

Inputs:

- **Data**

  An input data set.

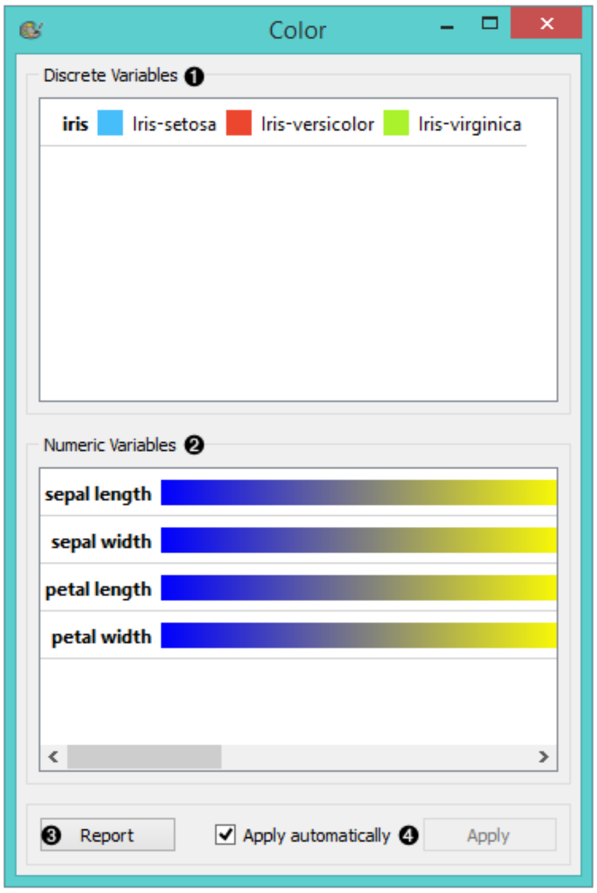
Outputs:

- **Data**

  A data set with a new color legend.

Description

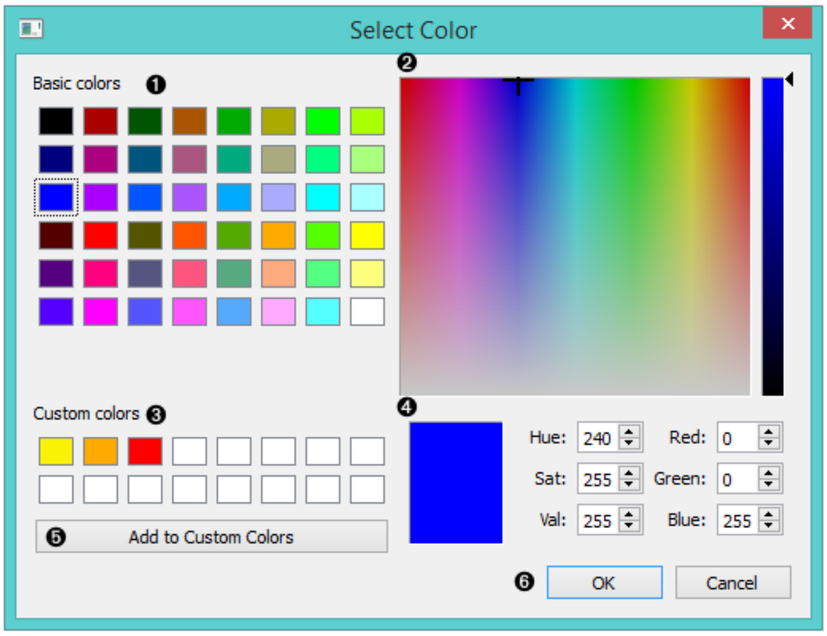
The **Color** widget enables you to set the color legend in your visualizations according to your own preferences. This option provides you with the tools for emphasizing your results and offers a great variety of color options for presenting your data. It can be combined with most visualizations widgets.



1. A list of discrete variables. You can set the color of each variable by double-clicking on it and opening the *Color palette* or the *Select color* window. The widget also enables text-editing. By clicking on a variable, you can change its name.

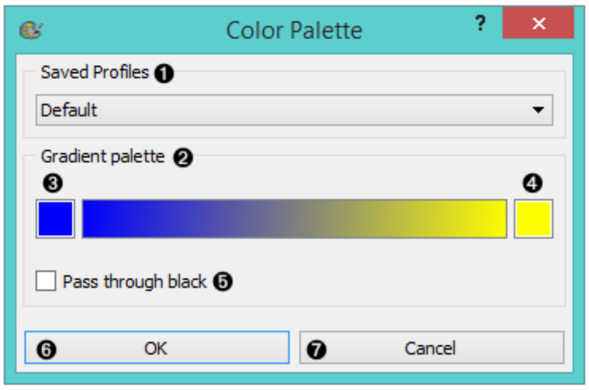
2. A list of continuous variables. You can customize the color gradients by double-clicking on them. The widget also enables text-editing. By clicking on a variable, you can change its name. If you hover over the right side side of the gradient, *Copy to all* appears. You can then apply your customized color gradient to all variables.
3. Produce a report.
4. Apply changes. If *Apply automatically* is ticked, changes will be communicated automatically. Alternatively, just click *Apply*.

Discrete variables



1. Choose a desired color from the palette of basic colors.
2. Move the cursor to choose a custom color from the color palette.
3. Choose a custom color from your previously saved color choices.
4. Specify the custom color by:
   - entering the red, green, and blue components of the color as values between 0 (darkest) and 255 (brightest)
   - entering the hue, saturation and luminescence components of the color as values in the range 0 to 255
5. Add the created color to your custom colors.
6. Click *OK* to save your choices or *Cancel* to exit the the color palette.

Numeric variables

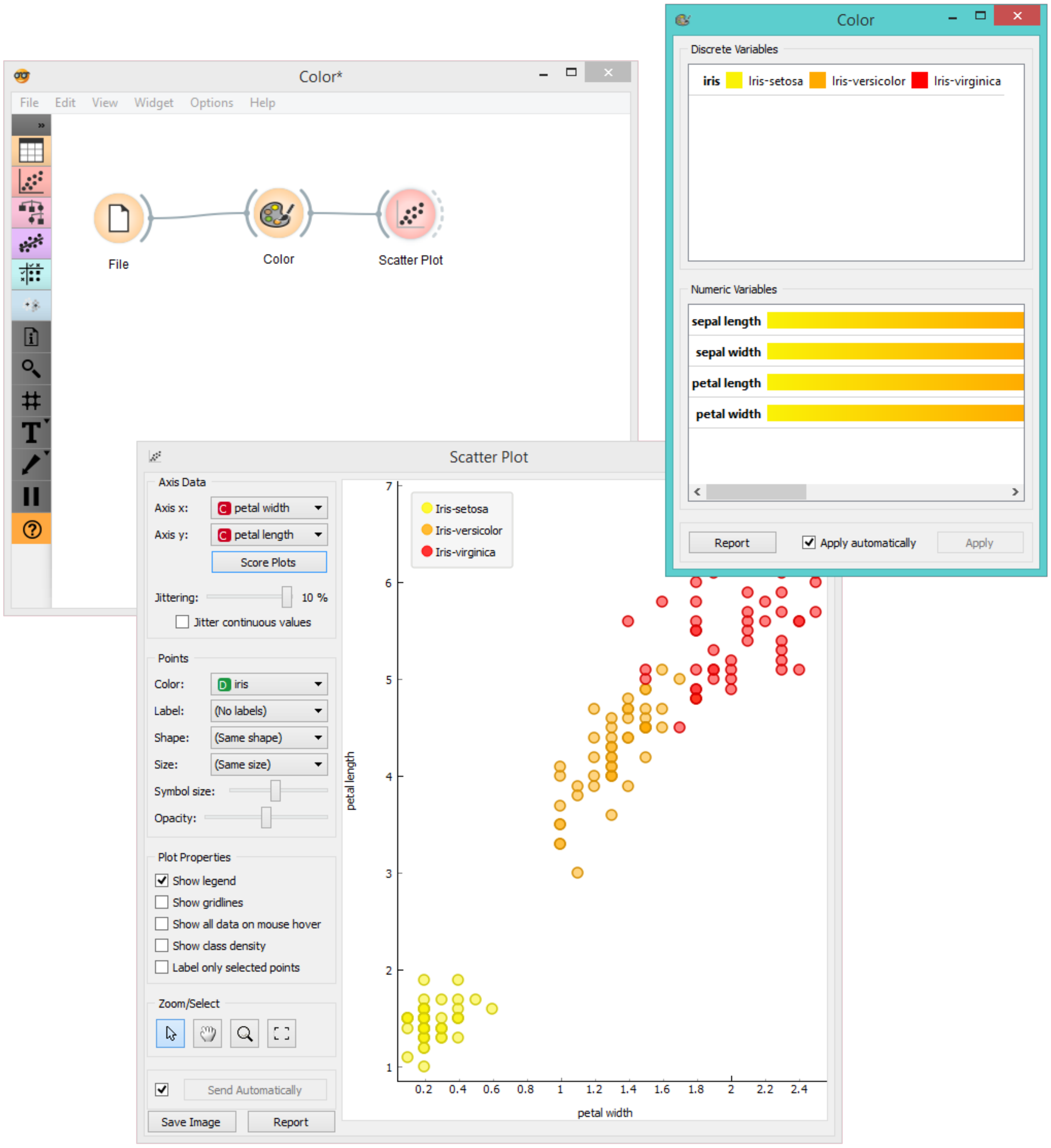


1. Choose a gradient from your saved profiles. The default profile is already set.
2. The gradient palette
3. Select the left side of the gradient. Double clicking the color opens the *Select Color* window.
4. Select the right side of the gradient. Double clicking the color opens the *Select Color* window.
5. Pass through black.

6. Click *OK* to save your choices or *Cancel* to exit the color palette.

Example

We chose to work with the *Iris* data set. We opened the color palette and selected three new colors for the three types of Irises. Then we opened the Scatter Plot widget and viewed the changes made to the scatter plot.



Discrete Variables

iris	🟨 Iris-setosa	🟧 Iris-versicolor	🟥 Iris-virginica

Numeric Variables

sepal length
sepal width
petal length
petal width

Report ✓ Apply automatically Apply

Axis Data
Axis x: petal width
Axis y: petal length
Score Plots
Jittering: 10 %
☐ Jitter continuous values

Points
Color: iris
Label: (No labels)
Shape: (Same shape)
Size: (Same size)
Symbol size:
Opacity:

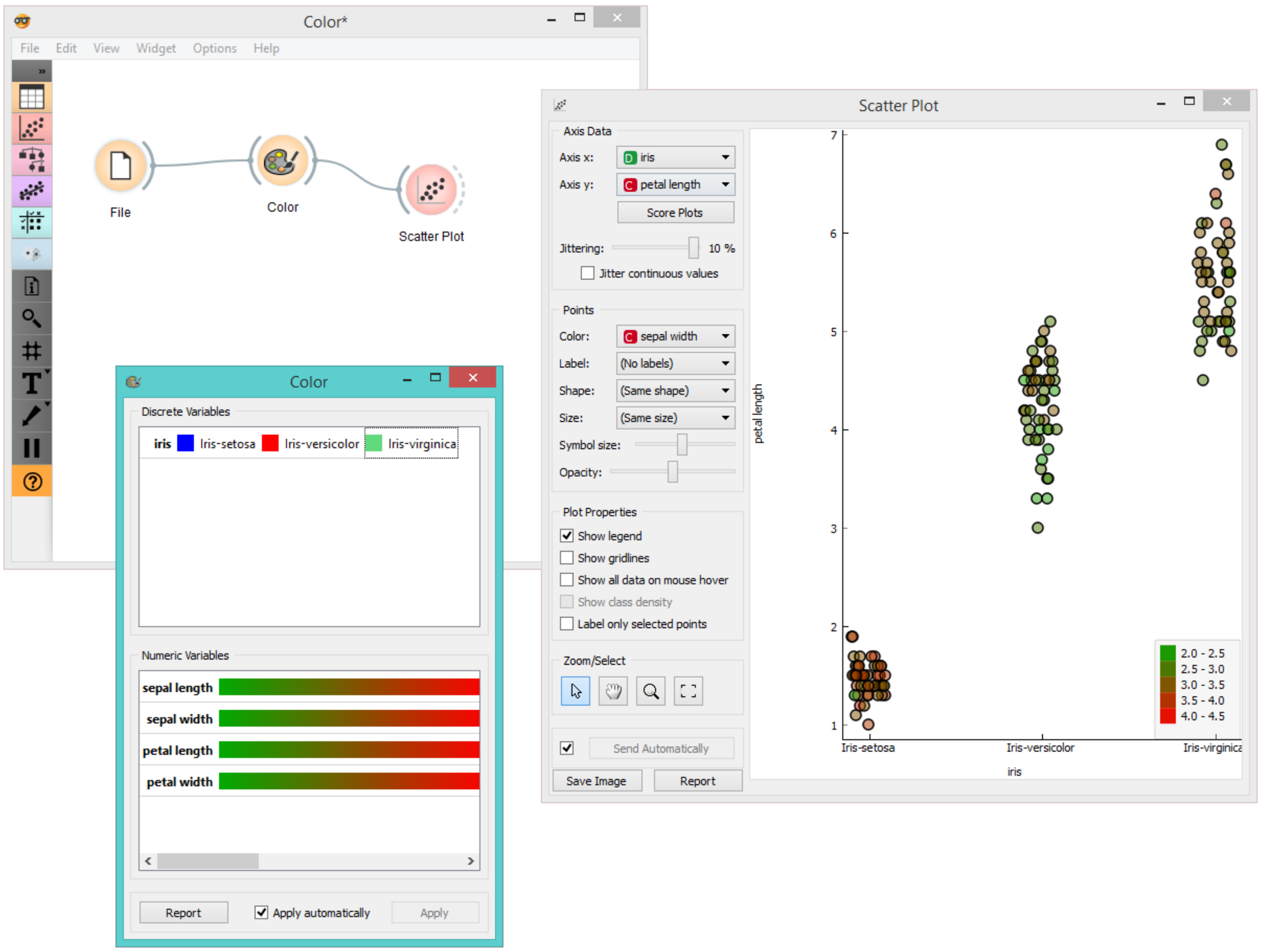
Plot Properties
☑ Show legend
☐ Show gridlines
☐ Show all data on mouse hover
☐ Show class density
☐ Label only selected points

Zoom/Select

☑ Send Automatically

Save Image Report

For our second example, we wished to demonstrate the use of the **Color** widget with continuous variables. We put different types of Irises on the x axis and petal length on the y axis. We created a new color gradient and named it greed (green + red). In order to show that sepal length is not a deciding factor in differentiating between different types of Irises, we chose to color the points according to sepal width.



Axis x: iris
Axis y: petal length
Jittering: 10 %
☐ Jitter continuous values
Color: sepal width
Label: (No labels)
Shape: (Same shape)
Size: (Same size)
Symbol size:
Opacity:
☑ Show legend
☐ Show gridlines
☐ Show all data on mouse hover
☐ Show class density
☐ Label only selected points
☑ Send Automatically
☑ Apply automatically

Concatenate

Concatenates data from multiple sources.

Signals

Inputs:

- **Primary Data**

  A data set that defines the attribute set.

- **Additional Data**

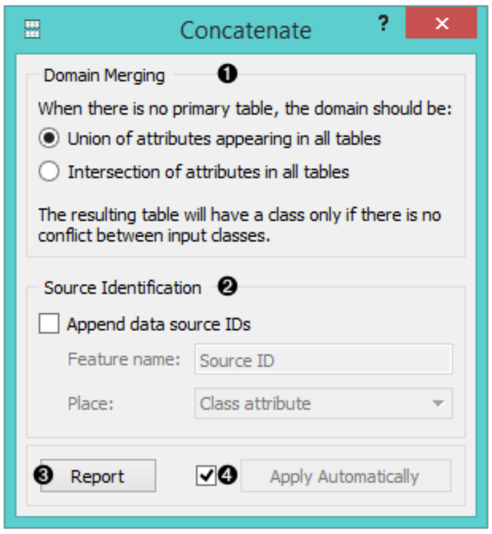
  An additional data set.

Outputs:

- **Data**

Description

The widget concatenates multiple sets of instances (data sets). The merge is “vertical”, in a sense that two sets of 10 and 5 instances yield a new set of 15 instances.



Concatenate

Domain Merging ❶

When there is no primary table, the domain should be:

(●) Union of attributes appearing in all tables

( ) Intersection of attributes in all tables

The resulting table will have a class only if there is no conflict between input classes.

Source Identification ❷

☐ Append data source IDs

Feature name: Source ID

Place: Class attribute

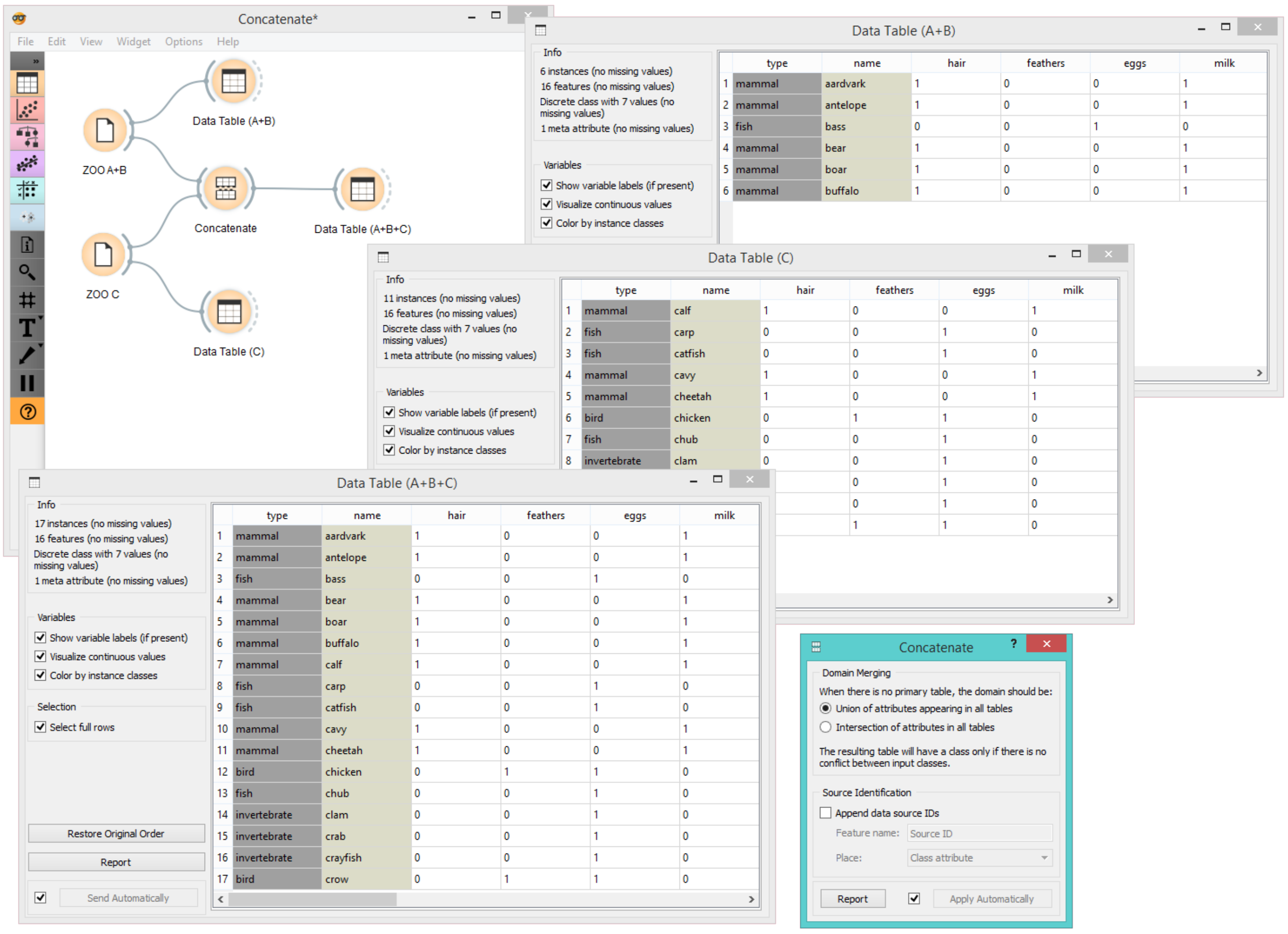
1. Set the attribute merging method.
2. Add the identification of source data sets to the output data set.
3. Produce a report.
4. If *Apply automatically* is ticked, changes are communicated automatically. Otherwise, click *Apply*.

If one of the tables is connected to the widget as the primary table, the resulting table will contain its own attributes. If there is no primary table, the attributes can be either a union of all attributes that appear in the tables specified as *Additional Tables*, or their intersection, that is, a list of attributes common to all the connected tables.

Example

As shown below, the widget can be used for merging data from two separate files. Let’s say we have two data sets with the same attributes, one containing instances from the first experiment and the other instances from the second experiment and we wish to join the two data tables together. We use the **Concatenate** widget to merge the data sets by attributes (appending new rows under existing attributes).

Below, we used a modified *Zoo* data set. In the **`first`** File widget, we loaded only the animals beginning with the letters A and B and in the **`second`** one only the animals beginning with the letter C. Upon concatenation, we observe the new data in the Data Table widget, where we see the complete table with animals from A to C.



Data Table (A+B)

Info
6 instances (no missing values)
16 features (no missing values)
Discrete class with 7 values (no missing values)
1 meta attribute (no missing values)

Variables
☑ Show variable labels (if present)
☑ Visualize continuous values
☑ Color by instance classes

	type	name	hair	feathers	eggs	milk
1	mammal	aardvark	1	0	0	1
2	mammal	antelope	1	0	0	1
3	fish	bass	0	0	1	0
4	mammal	bear	1	0	0	1
5	mammal	boar	1	0	0	1
6	mammal	buffalo	1	0	0	1

Data Table (C)

Info
11 instances (no missing values)
16 features (no missing values)
Discrete class with 7 values (no missing values)
1 meta attribute (no missing values)

Variables
☑ Show variable labels (if present)
☑ Visualize continuous values
☑ Color by instance classes

	type	name	hair	feathers	eggs	milk
1	mammal	calf	1	0	0	1
2	fish	carp	0	0	1	0
3	fish	catfish	0	0	1	0
4	mammal	cavy	1	0	0	1
5	mammal	cheetah	1	0	0	1
6	bird	chicken	0	1	1	0
7	fish	chub	0	0	1	0
8	invertebrate	clam	0	0	1	0
				0	1	0
				0	1	0
				1	1	0

Data Table (A+B+C)

Info
17 instances (no missing values)
16 features (no missing values)
Discrete class with 7 values (no missing values)
1 meta attribute (no missing values)

Variables
☑ Show variable labels (if present)
☑ Visualize continuous values
☑ Color by instance classes

Selection
☑ Select full rows

Restore Original Order
Report
☑ Send Automatically

	type	name	hair	feathers	eggs	milk
1	mammal	aardvark	1	0	0	1
2	mammal	antelope	1	0	0	1
3	fish	bass	0	0	1	0
4	mammal	bear	1	0	0	1
5	mammal	boar	1	0	0	1
6	mammal	buffalo	1	0	0	1
7	mammal	calf	1	0	0	1
8	fish	carp	0	0	1	0
9	fish	catfish	0	0	1	0
10	mammal	cavy	1	0	0	1
11	mammal	cheetah	1	0	0	1
12	bird	chicken	0	1	1	0
13	fish	chub	0	0	1	0
14	invertebrate	clam	0	0	1	0
15	invertebrate	crab	0	0	1	0
16	invertebrate	crayfish	0	0	1	0
17	bird	crow	0	1	1	0

Concatenate

Domain Merging
When there is no primary table, the domain should be:
[✓ Union of attributes appearing in all tables]
Intersection of attributes in all tables

The resulting table will have a class only if there is no conflict between input classes.

Source Identification
☐ Append data source IDs
Feature name: Source ID
Place: Class attribute

Report
☑ Apply Automatically

Continuize

Turns discrete attributes into continuous dummy variables.

Signals

Inputs:

- **Data**

  Input data set

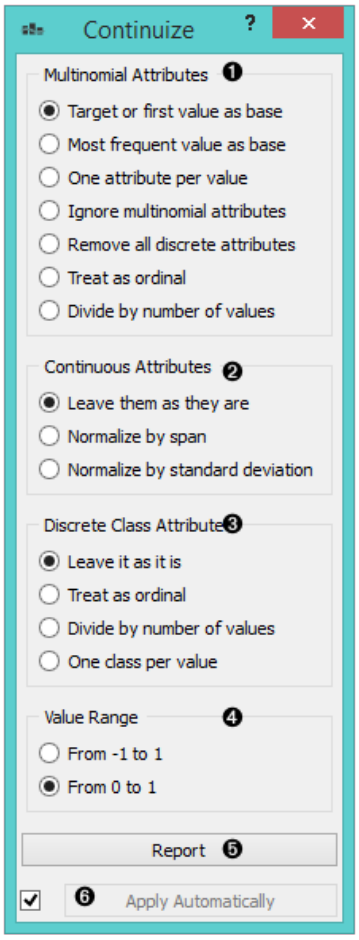
Outputs:

- **Data**

  Output data set

Description

The **Continuize** widget receives a data set in the input and outputs the same data set in which the discrete attributes (including binary attributes) are replaced with continuous ones.


Continuize

Multinomial Attributes ❶
- [✓ Target or first value as base]
- Most frequent value as base
- One attribute per value
- Ignore multinomial attributes
- Remove all discrete attributes
- Treat as ordinal
- Divide by number of values

Continuous Attributes ❷
- [✓ Leave them as they are]
- Normalize by span
- Normalize by standard deviation

Discrete Class Attribute ❸
- [✓ Leave it as it is]
- Treat as ordinal
- Divide by number of values
- One class per value

Value Range ❹
- From -1 to 1
- [✓ From 0 to 1]

Report ❺

[✓] ❻ Apply Automatically

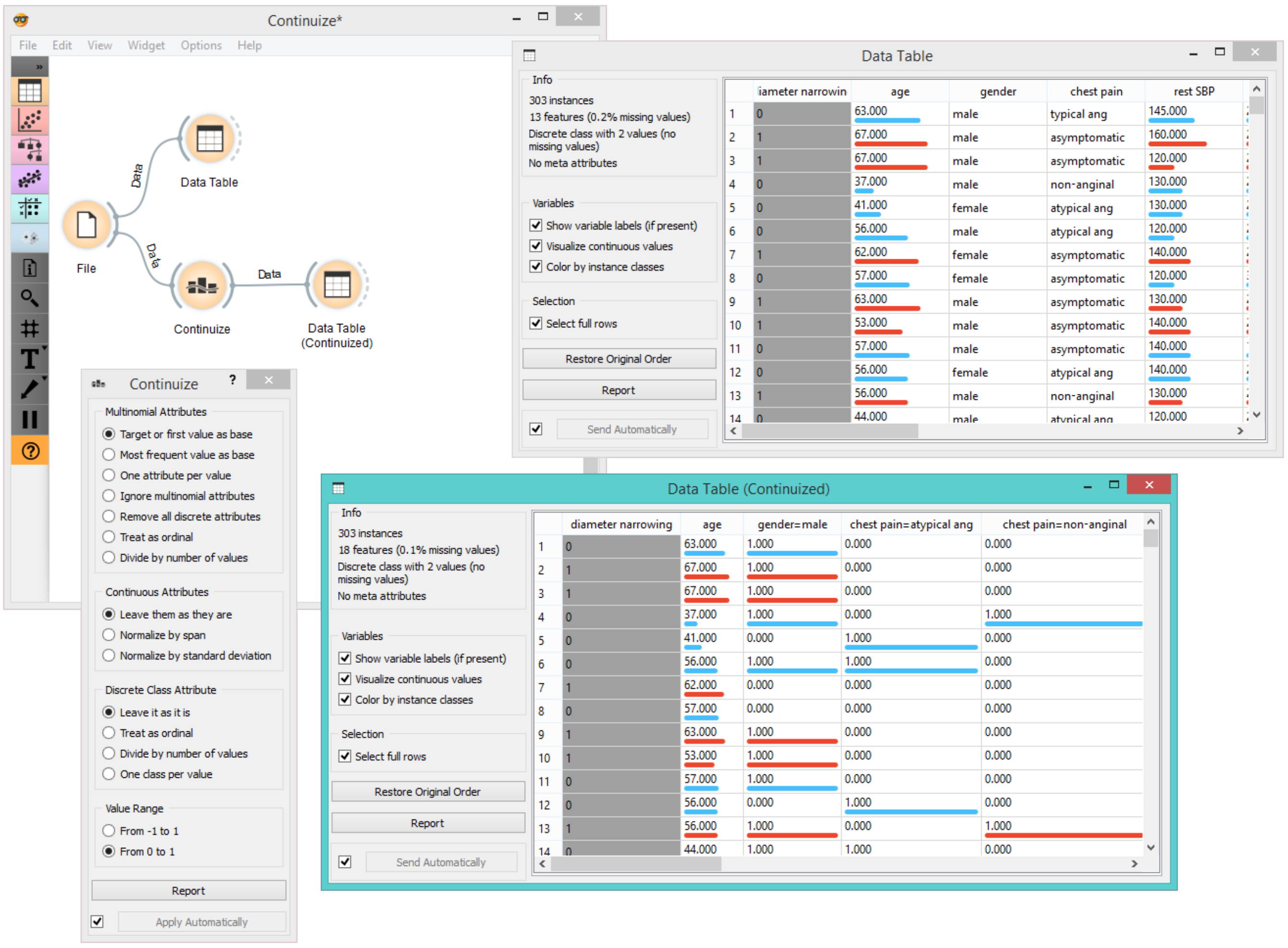
1. Continuization methods, which define the treatment of multivalued discrete attributes. Say that we have a discrete attribute status with the values low, middle and high, listed in that order. Options for their transformation are:
   - **Target or First value as base**: the attribute will be transformed into two continuous attributes, sta-

    tus=middle with values 0 or 1 signifying whether the original attribute had value middle on a particular example, and similarly, status=high. Hence, a three-valued attribute is transformed into two continuous attributes, corresponding to all except the first value of the attribute.
    - **Most frequent value as base**: similar to the above, except that the data is analyzed and the most frequent value is used as a base. So, if most examples have the value middle, the two newly constructed continuous attributes will be status=low and status=high.
    - **One attribute per value**: this would construct three continuous attributes out of a three-valued discrete one.
    - **Ignore multinominal attributes**: removes the multinominal attributes from the data.
    - **Treat as ordinal**: converts the attribute into a continuous attribute with values 0, 1, and 2.
    - **Divide by number of values**: same as above, except that the values are normalized into range 0-1. So, our case would give values 0, 0.5 and 1.

2. Define the treatment of continuous attributes. You will usually prefer the *Leave them as they are* option. The alternative is *Normalize by span*, which will subtract the lowest value found in the data and divide by the span, so all values will fit into [0, 1]. Finally, *Normalize by standard deviation* subtracts the average and divides by the deviation.
3. Define the treatment of class attributes. Besides leaving it as it is, there are also a couple of options available for multinominal attributes, except for those options which split the attribute into more than one attribute - this obviously cannot be supported since you cannot have more than one class attribute.
4. With *value range*, you can define the values of new attributes. In the above text, we supposed the range *from 0 to 1*. You can change it to *from -1 to 1*.
5. Produce a report.
6. If *Apply automatically* is ticked, changes are committed automatically. Otherwise, you have to press *Apply* after each change.

Examples

First, let's see what is the output of the **Continuize** widget. We feed the original data (the *Heart disease* data set) into the Data Table and see how they look like. Then we continuize the discrete values and observe them in another Data Table.



Continuize

Multinomial Attributes
[✓ Target or first value as base]
Most frequent value as base
One attribute per value
Ignore multinomial attributes
Remove all discrete attributes
Treat as ordinal
Divide by number of values

Continuous Attributes
[✓ Leave them as they are]
Normalize by span
Normalize by standard deviation

Discrete Class Attribute
[✓ Leave it as it is]
Treat as ordinal
Divide by number of values
One class per value

Value Range
From -1 to 1
[✓ From 0 to 1]

Report
✓ Apply Automatically

Data Table

Info
303 instances
13 features (0.2% missing values)
Discrete class with 2 values (no missing values)
No meta attributes

Variables
✓ Show variable labels (if present)
✓ Visualize continuous values
✓ Color by instance classes

Selection
✓ Select full rows

Restore Original Order
Report
✓ Send Automatically

	iameter narrowin	age	gender	chest pain	rest SBP
1	0	63.000	male	typical ang	145.000
2	1	67.000	male	asymptomatic	160.000
3	1	67.000	male	asymptomatic	120.000
4	0	37.000	male	non-anginal	130.000
5	0	41.000	female	atypical ang	130.000
6	0	56.000	male	atypical ang	120.000
7	1	62.000	female	asymptomatic	140.000
8	0	57.000	female	asymptomatic	120.000
9	1	63.000	male	asymptomatic	130.000
10	1	53.000	male	asymptomatic	140.000
11	0	57.000	male	asymptomatic	140.000
12	0	56.000	female	atypical ang	140.000
13	1	56.000	male	non-anginal	130.000
14	0	44.000	male	atypical ang	120.000

Data Table (Continuized)

Info
303 instances
18 features (0.1% missing values)
Discrete class with 2 values (no missing values)
No meta attributes

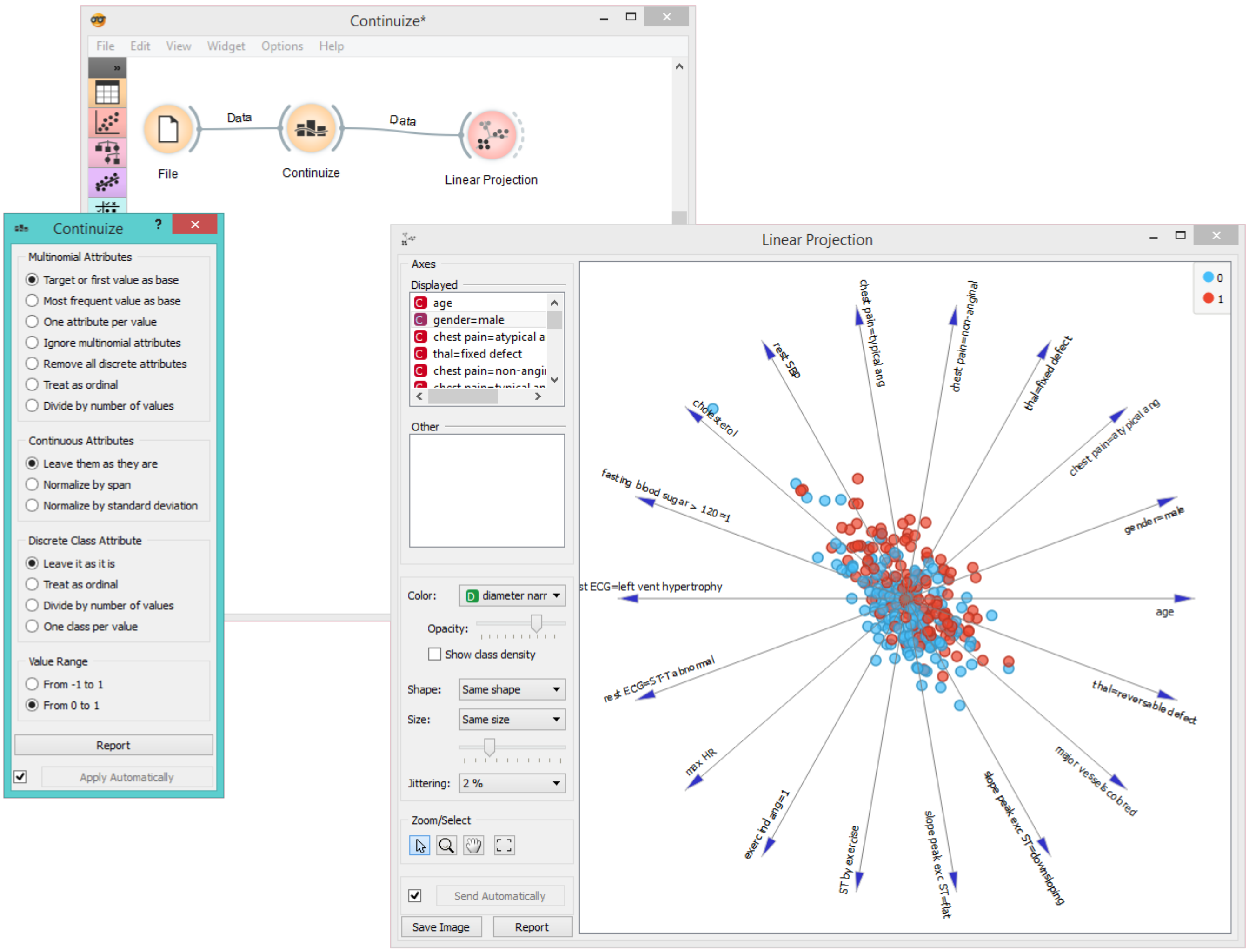
Variables
✓ Show variable labels (if present)
✓ Visualize continuous values
✓ Color by instance classes

Selection
✓ Select full rows

Restore Original Order
Report
✓ Send Automatically

	diameter narrowing	age	gender=male	chest pain=atypical ang	chest pain=non-anginal
1	0	63.000	1.000	0.000	0.000
2	1	67.000	1.000	0.000	0.000
3	1	67.000	1.000	0.000	0.000
4	0	37.000	1.000	0.000	1.000
5	0	41.000	0.000	1.000	0.000
6	0	56.000	1.000	1.000	0.000
7	1	62.000	0.000	0.000	0.000
8	0	57.000	0.000	0.000	0.000
9	1	63.000	1.000	0.000	0.000
10	1	53.000	1.000	0.000	0.000
11	0	57.000	1.000	0.000	0.000
12	0	56.000	0.000	1.000	0.000
13	1	56.000	1.000	0.000	1.000
14	0	44.000	1.000	1.000	0.000

In the second example, we show a typical use of this widget - in order to properly plot the linear projection of the data, discrete attributes need to be converted to continuous ones and that is why we put the data through the **Continuize** widget before drawing it. The attribute "*chest pain*" originally had four values and was transformed into three continuous attributes; similar happened to gender, which was transformed into a single attribute "*gender=female*".



Continuize

Multinomial Attributes
- [✓ Target or first value as base]
- Most frequent value as base
- One attribute per value
- Ignore multinomial attributes
- Remove all discrete attributes
- Treat as ordinal
- Divide by number of values

Continuous Attributes
- [✓ Leave them as they are]
- Normalize by span
- Normalize by standard deviation

Discrete Class Attribute
- [✓ Leave it as it is]
- Treat as ordinal
- Divide by number of values
- One class per value

Value Range
- From -1 to 1
- [✓ From 0 to 1]

Report

[✓] Apply Automatically

Linear Projection

Axes

Displayed
- age
- gender=male
- chest pain=atypical a
- thal=fixed defect
- chest pain=non-angi
- chest pain=typical an

Other

Color: diameter narr

Opacity:

Show class density

Shape: Same shape

Size: Same size

Jittering: 2 %

Zoom/Select

[✓] Send Automatically

Save Image Report

Create Class

Create class attribute from a string attribute.

Signals

Inputs:

- **Data**

  Attribute-valued data set.

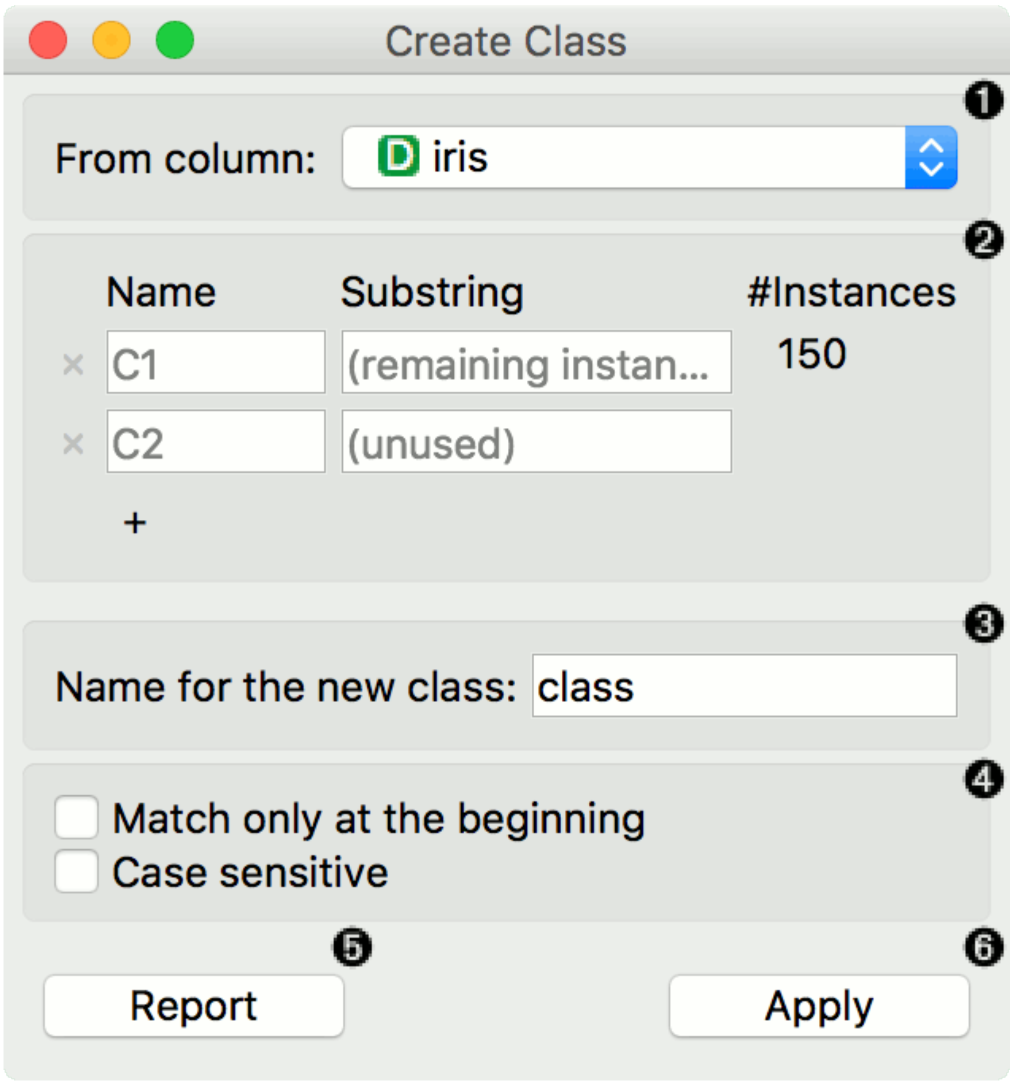
Outputs:

- **Data**

  Attribute-valued data set.

Description

Create Class creates a new class attribute from an existing discrete or string attribute. The widget matches the string value of the selected attribute and constructs a new user-defined value for matching instances.



From column: iris

	Name	Substring	#Instances
×	C1	(remaining instan...	150
×	C2	(unused)	

Name for the new class: class

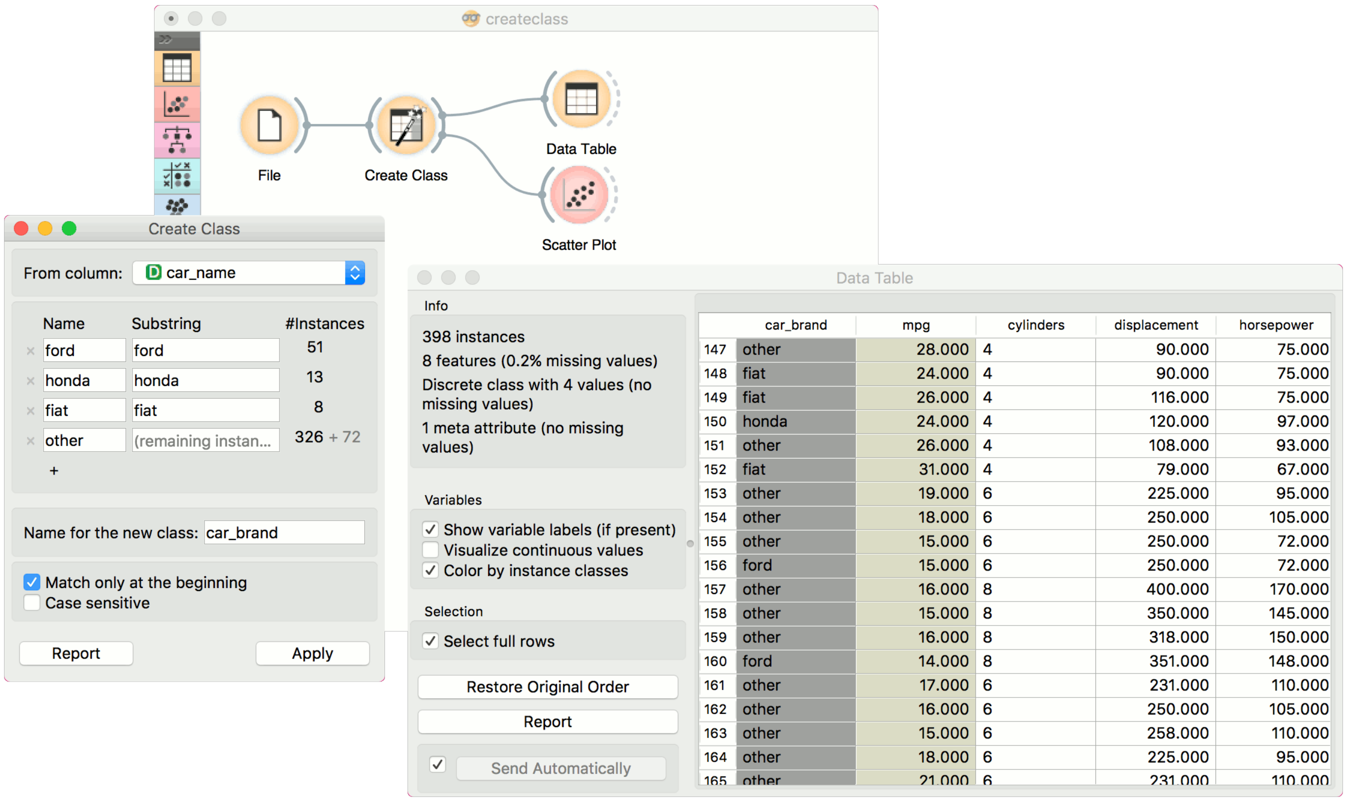
- [ ] Match only at the beginning
- [ ] Case sensitive

1. The attribute the new class is constructed from.
2. Matching: - Name: the name of the new class value - Substring: regex-defined substring that will match the values from the above-defined attribute - Instances: the number of instances matching the substring - Press '+' to add a new class value
3. Name of the new class column.
4. Match only at the beginning will begin matching from the beginning of the string. Case sensitive will match by case, too.
5. Produce a report.
6. Press *Apply* to commit the results.

Example

Here is a simple example with the *auto-mpg* data set. Pass the data to **Create Class**. Select *car_name* as a column to create the new class from. Here, we wish to create new values that match the car brand. First, we type *ford* as the new value for the matching strings. Then we define the substring that will match the data instances. This means that all instances containing *ford* in their *car_name*, will now have a value *ford* in the new class column. Next, we define the same for *honda* and *fiat*. The widget will tell us how many instance are yet unmatched (remaining instances). We will name them *other*, but you can continue creating new values by adding a condition with '+'.

We named our new class column *car_brand* and we matched at the beginning of the string.



Create Class

From column: car_name

Name	Substring	#Instances
ford	ford	51
honda	honda	13
fiat	fiat	8
other	(remaining instan...	326 + 72

Name for the new class: car_brand

☑ Match only at the beginning
☐ Case sensitive

Report Apply

Data Table

Info
398 instances
8 features (0.2% missing values)
Discrete class with 4 values (no missing values)
1 meta attribute (no missing values)

Variables
☑ Show variable labels (if present)
☐ Visualize continuous values
☑ Color by instance classes

Selection
☑ Select full rows

Restore Original Order
Report
☑ Send Automatically

	car_brand	mpg	cylinders	displacement	horsepower
147	other	28.000	4	90.000	75.000
148	fiat	24.000	4	90.000	75.000
149	fiat	26.000	4	116.000	75.000
150	honda	24.000	4	120.000	97.000
151	other	26.000	4	108.000	93.000
152	fiat	31.000	4	79.000	67.000
153	other	19.000	6	225.000	95.000
154	other	18.000	6	250.000	105.000
155	other	15.000	6	250.000	72.000
156	ford	15.000	6	250.000	72.000
157	other	16.000	8	400.000	170.000
158	other	15.000	8	350.000	145.000
159	other	16.000	8	318.000	150.000
160	ford	14.000	8	351.000	148.000
161	other	17.000	6	231.000	110.000
162	other	16.000	6	250.000	105.000
163	other	15.000	6	258.000	110.000
164	other	18.000	6	225.000	95.000
165	other	21.000	6	231.000	110.000

Finally, we can observe the new column in a Data Table or use the value as color in the Scatterplot.

Data Info

Displays information on a selected data set.

Signals

Inputs:

- **Data**

  A data set.

- **Selected Data**

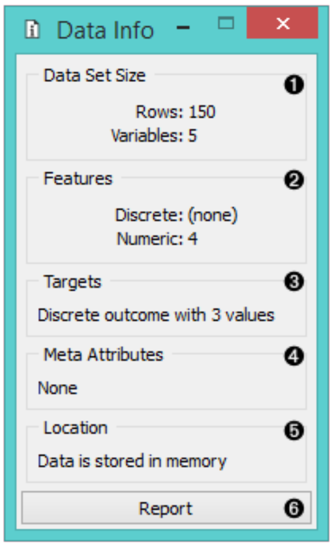
  A data subset.

Outputs:

- (None)

Description

A simple widget that presents information on data set size, features, targets, meta attributes, and location.



Rows: 150
Variables: 5

Discrete: (none)
Numeric: 4

Discrete outcome with 3 values

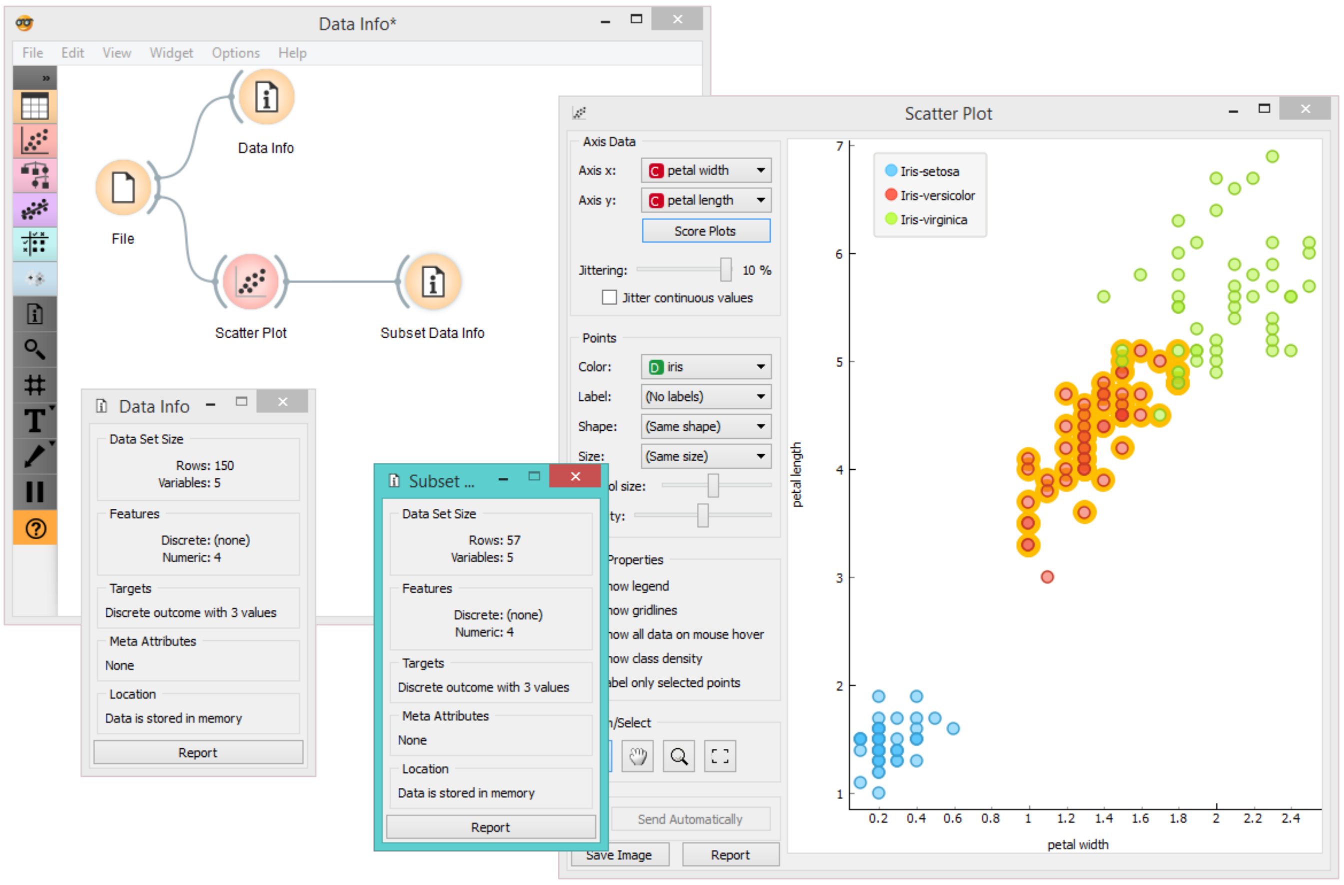
None

Data is stored in memory

1. Information on data set size
2. Information on discrete and continuous features
3. Information on targets
4. Information on meta attributes
5. Information on where the data is stored
6. Produce a report.

Example

Below, we compare the basic statistics of two **Data Info** widgets - one with information on the entire data set and the other with information on the (manually) selected subset from the Scatterplot widget. We used the *Iris* data set.



Data Info

Data Set Size
Rows: 150
Variables: 5

Features
Discrete: (none)
Numeric: 4

Targets
Discrete outcome with 3 values

Meta Attributes
None

Location
Data is stored in memory

Report

Subset ...

Data Set Size
Rows: 57
Variables: 5

Features
Discrete: (none)
Numeric: 4

Targets
Discrete outcome with 3 values

Meta Attributes
None

Location
Data is stored in memory

Report

Axis Data
Axis x: petal width
Axis y: petal length
Jittering: 10 %
Jitter continuous values

Points
Color: iris
Label: (No labels)
Shape: (Same shape)
Size: (Same size)
ol size:
ty:

Properties
how legend
how gridlines
how all data on mouse hover
how class density
abel only selected points

/Select

Data Sampler

Selects a subset of data instances from an input data set.

Signals

Inputs:

- **Data**

  Input data set to be sampled.

Outputs:

- **Data Sample**

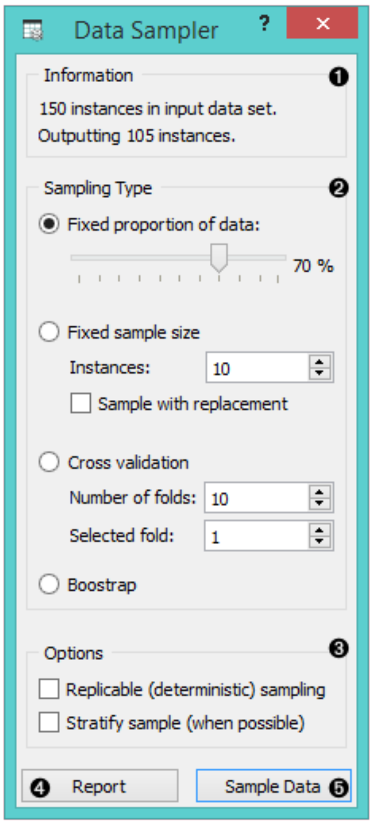
  A set of sampled data instances.

- **Remaining Data**

  All other data instances from the input data set, which are not included in the sample.

Description

The **Data Sampler** widget implements several means of sampling data from an input channel. It outputs a sampled and a complementary data set (with instances from the input set that are not included in the sampled data set). The output is processed after the input data set is provided and *Sample Data* is pressed.



Data Sampler

Information ❶

150 instances in input data set.
Outputting 105 instances.

Sampling Type ❷

(●) Fixed proportion of data: 70 %

( ) Fixed sample size
Instances: 10
[ ] Sample with replacement

( ) Cross validation
Number of folds: 10
Selected fold: 1

( ) Boostrap

Options ❸

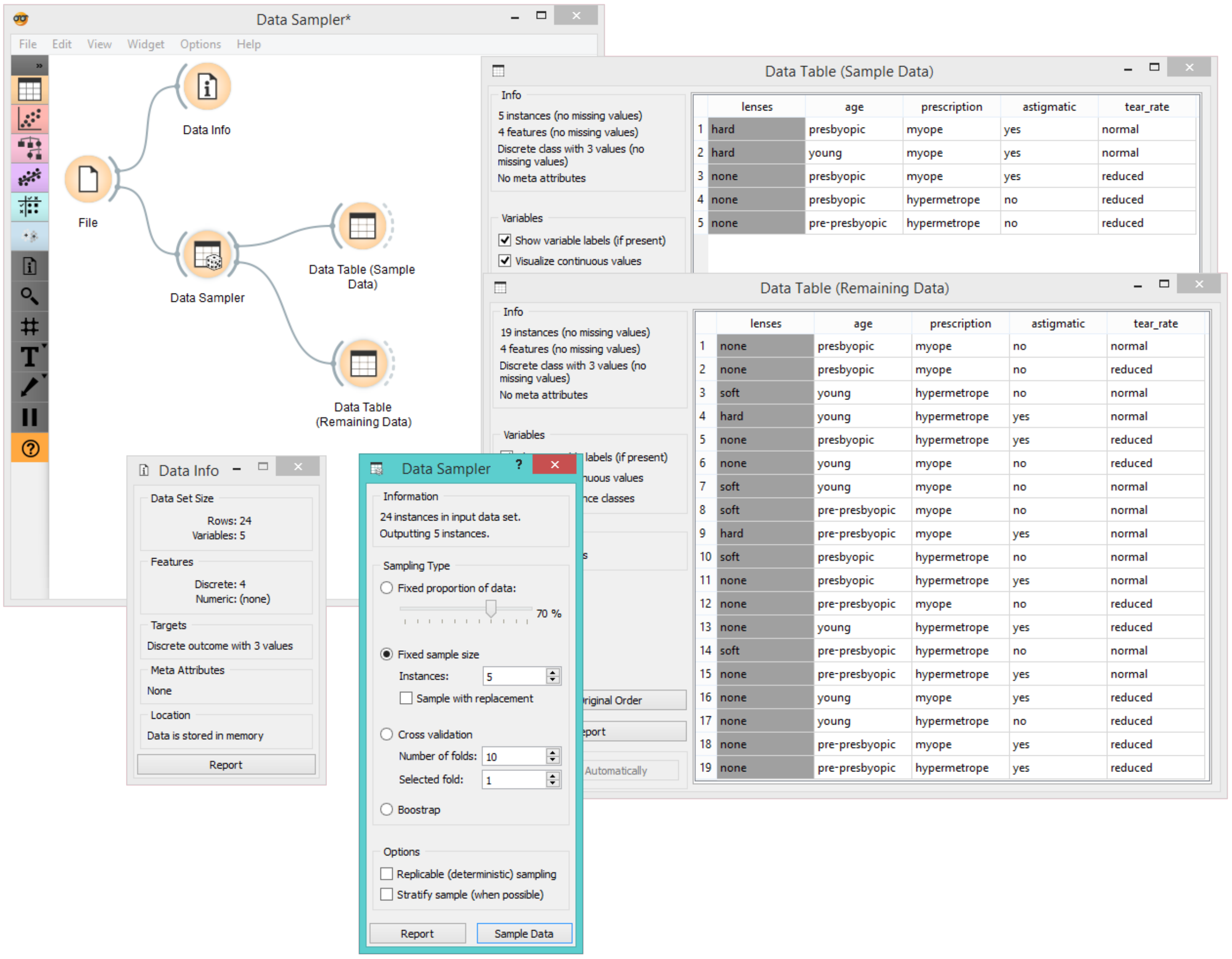
[ ] Replicable (deterministic) sampling
[ ] Stratify sample (when possible)

1. Information on the input and output data set
2. The desired sampling method:

   - **Fixed proportion of data** returns a selected percentage of the entire data (e.g. 70% of all the data)
   - **Fixed sample size** returns a selected number of data instances with a chance to set *Sample with replacement*, which always samples from the entire data set (does not subtract instances already in the subset)
   - Cross Validation partitions data instances into complementary subsets, where you can select the number of folds (subsets) and which fold you want to use as a sample.
3. *Replicable sampling* maintains sampling patterns that can be carried across users, while *stratification* mimics the composition of the input data set.
4. Produce a report.
5. Press *Sample data* to output the data sample.

Examples

First, let's see how the **Data Sampler** works. Let's look at the information on the original data set in the Data Info widget. We see there are 24 instances in the data (we used *lenses.tab*). We sampled the data with the **Data Sampler** widget and we chose to go with a fixed sample size of 5 instances for simplicity. We can observe the sampled data in the Data Table widget. The second Data Table shows the remaining 19 instances that weren't in the sample.



Data Info

Data Set Size
Rows: 24
Variables: 5

Features
Discrete: 4
Numeric: (none)

Targets
Discrete outcome with 3 values

Meta Attributes
None

Location
Data is stored in memory

Report

Data Sampler

Information
24 instances in input data set.
Outputting 5 instances.

Sampling Type
Fixed proportion of data: 70 %
[✓ Fixed sample size]
Instances: 5
Sample with replacement
Cross validation
Number of folds: 10
Selected fold: 1
Boostrap

Options
Replicable (deterministic) sampling
Stratify sample (when possible)

Report | Sample Data

Data Table (Sample Data)

Info
5 instances (no missing values)
4 features (no missing values)
Discrete class with 3 values (no missing values)
No meta attributes

Variables
✓ Show variable labels (if present)
✓ Visualize continuous values

	lenses	age	prescription	astigmatic	tear_rate
1	hard	presbyopic	myope	yes	normal
2	hard	young	myope	yes	normal
3	none	presbyopic	myope	yes	reduced
4	none	presbyopic	hypermetrope	no	reduced
5	none	pre-presbyopic	hypermetrope	no	reduced

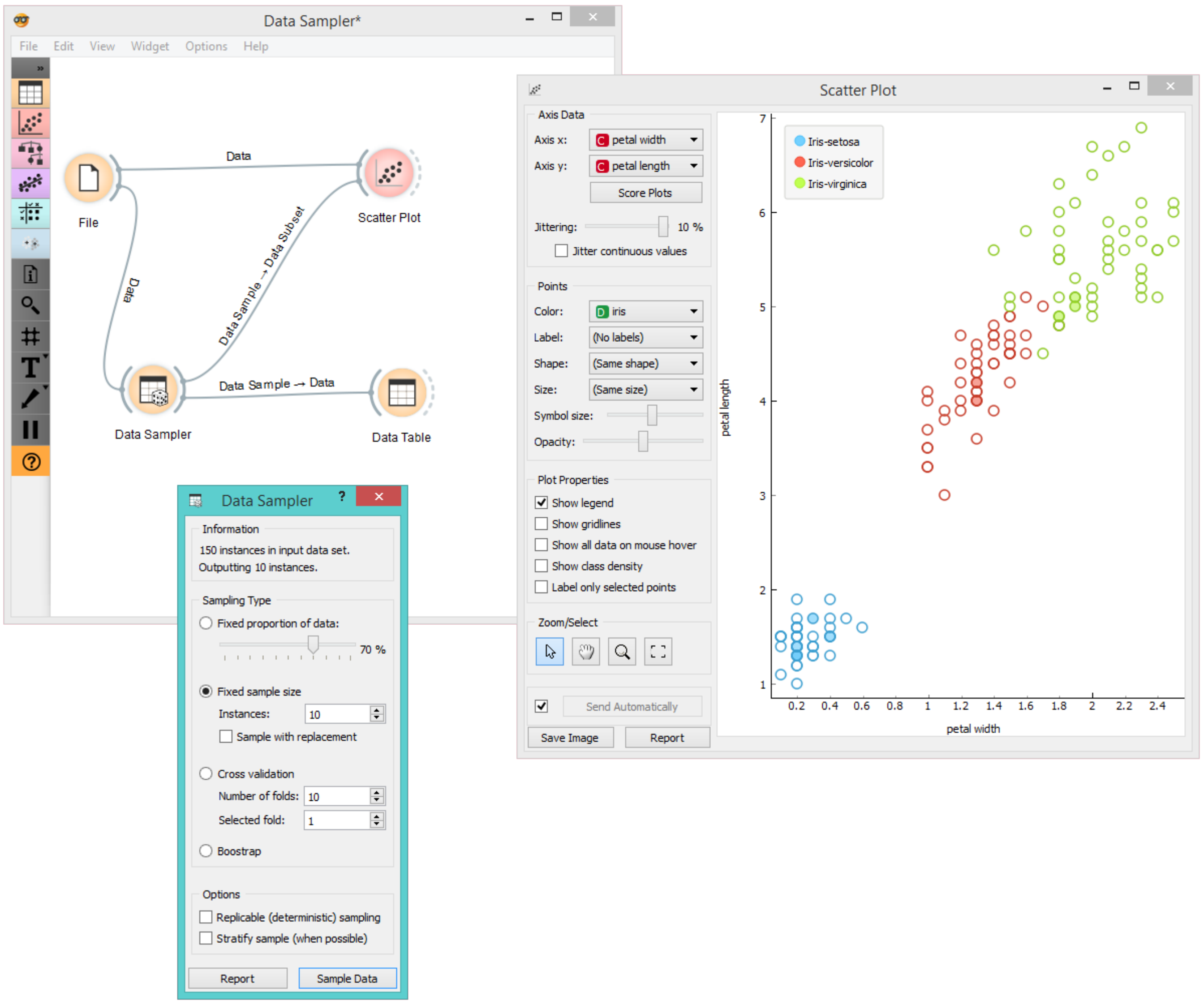
Data Table (Remaining Data)

Info
19 instances (no missing values)
4 features (no missing values)
Discrete class with 3 values (no missing values)
No meta attributes

Variables

	lenses	age	prescription	astigmatic	tear_rate
1	none	presbyopic	myope	no	normal
2	none	presbyopic	myope	no	reduced
3	soft	young	hypermetrope	no	normal
4	hard	young	hypermetrope	yes	normal
5	none	presbyopic	hypermetrope	yes	reduced
6	none	young	myope	no	reduced
7	soft	young	myope	no	normal
8	soft	pre-presbyopic	myope	no	normal
9	hard	pre-presbyopic	myope	yes	normal
10	soft	presbyopic	hypermetrope	no	normal
11	none	presbyopic	hypermetrope	yes	normal
12	none	pre-presbyopic	myope	no	reduced
13	none	young	hypermetrope	yes	reduced
14	soft	pre-presbyopic	hypermetrope	no	normal
15	none	pre-presbyopic	hypermetrope	yes	normal
16	none	young	myope	yes	reduced
17	none	young	hypermetrope	no	reduced
18	none	pre-presbyopic	myope	yes	reduced
19	none	pre-presbyopic	hypermetrope	yes	reduced

In the workflow below, we have sampled 10 data instances from the *Iris* data set and sent the original data and the sample to Scatter Plot widget for exploratory data analysis. The sampled data instances are plotted with filled circles, while the original data set is represented with empty circles.



Data Sampler

Information

150 instances in input data set.
Outputting 10 instances.

Sampling Type

- ( ) Fixed proportion of data: 70 %
- (●) Fixed sample size
  - Instances: 10
  - ☐ Sample with replacement
- ( ) Cross validation
  - Number of folds: 10
  - Selected fold: 1
- ( ) Boostrap

Options

- ☐ Replicable (deterministic) sampling
- ☐ Stratify sample (when possible)

Scatter Plot

Axis Data

- Axis x: petal width
- Axis y: petal length
- Jittering: 10 %
- ☐ Jitter continuous values

Points

- Color: iris
- Label: (No labels)
- Shape: (Same shape)
- Size: (Same size)
- Symbol size:
- Opacity:

Plot Properties

- ✓ Show legend
- ☐ Show gridlines
- ☐ Show all data on mouse hover
- ☐ Show class density
- ☐ Label only selected points

Zoom/Select

✓ Send Automatically

Data Sets

Load a data set from an online repository.

Signals

Inputs:

- (None)

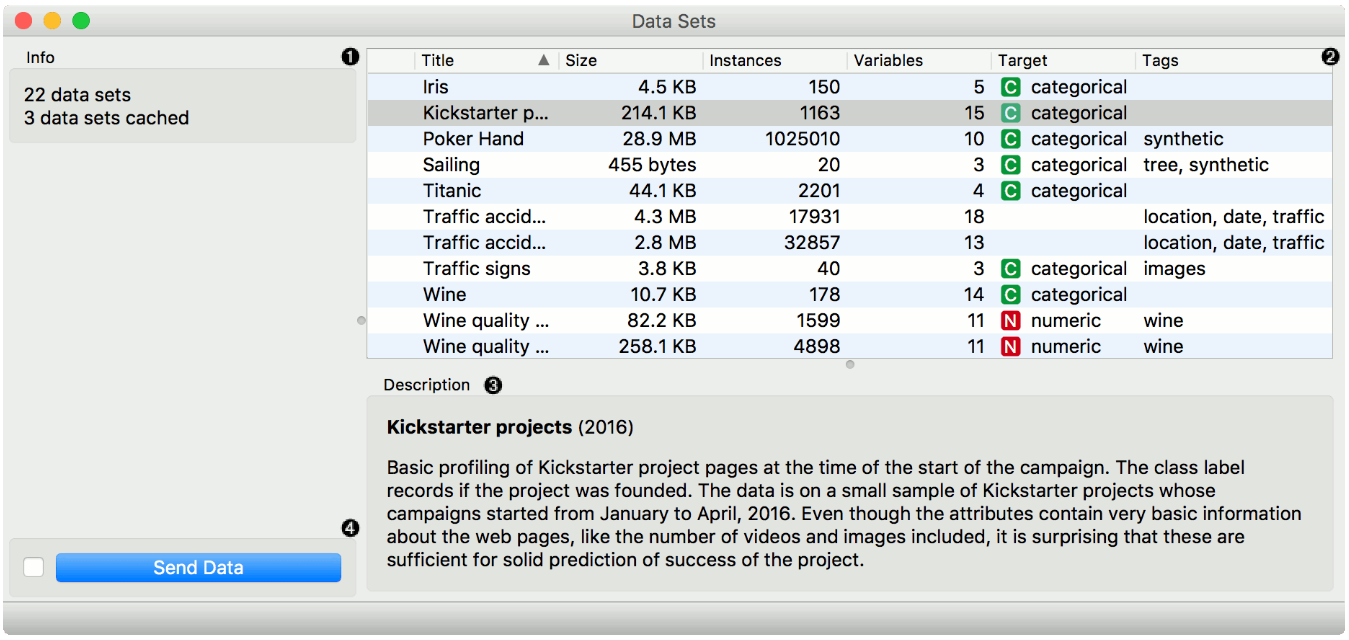
Outputs:

- **Data**

  Attribute-valued data set.

Description

Datasets widget retrives selected data set from the server and sends it to the output. File is downloaded to the local memory and thus instantly available even without the internet connection. Each data set is provided with a description and information on the data size, number of instances, number of variables, target and tags.



22 data sets
3 data sets cached

Title	Size	Instances	Variables	Target	Tags
Iris	4.5 KB	150	5	C categorical	
Kickstarter p...	214.1 KB	1163	15	C categorical	
Poker Hand	28.9 MB	1025010	10	C categorical	synthetic
Sailing	455 bytes	20	3	C categorical	tree, synthetic
Titanic	44.1 KB	2201	4	C categorical	
Traffic accid...	4.3 MB	17931	18		location, date, traffic
Traffic accid...	2.8 MB	32857	13		location, date, traffic
Traffic signs	3.8 KB	40	3	C categorical	images
Wine	10.7 KB	178	14	C categorical	
Wine quality ...	82.2 KB	1599	11	N numeric	wine
Wine quality ...	258.1 KB	4898	11	N numeric	wine

Description

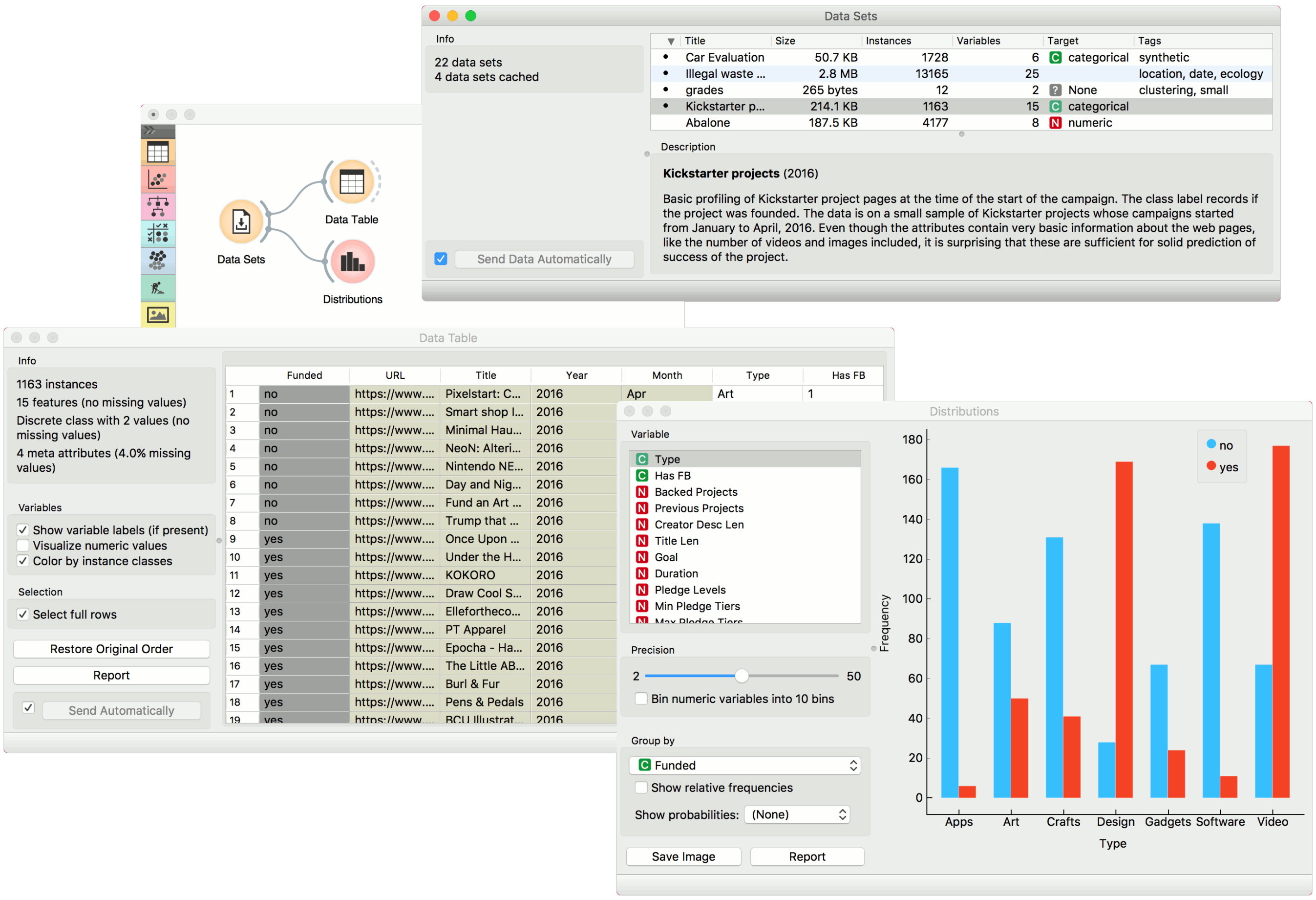
Kickstarter projects (2016)

Basic profiling of Kickstarter project pages at the time of the start of the campaign. The class label records if the project was founded. The data is on a small sample of Kickstarter projects whose campaigns started from January to April, 2016. Even though the attributes contain very basic information about the web pages, like the number of videos and images included, it is surprising that these are sufficient for solid prediction of success of the project.

1. Information on the number of data sets available and the number of them downloaded to the local memory.
2. Content of available data sets. Each data set is described with the size, number of instances and variables, type of the target variable and tags.
3. Formal description of the selected data set.
4. If *Send Data Automatically* is ticked, selected data set is communicated automatically. Alternatively, press *Send Data*.

Example

Orange workflows can start with **Data Sets** widget instead of **File** widget. In the example below, the widget retrieves a data set from an online repository (Kickstarter data), which is subsequently sent to both the Data Table and the Distributions.



Data Sets

Info

22 data sets
4 data sets cached

	Title	Size	Instances	Variables	Target	Tags
•	Car Evaluation	50.7 KB	1728	6	C categorical	synthetic
•	Illegal waste ...	2.8 MB	13165	25		location, date, ecology
•	grades	265 bytes	12	2	? None	clustering, small
•	Kickstarter p...	214.1 KB	1163	15	C categorical	
	Abalone	187.5 KB	4177	8	N numeric	

Description

Kickstarter projects (2016)

Basic profiling of Kickstarter project pages at the time of the start of the campaign. The class label records if the project was founded. The data is on a small sample of Kickstarter projects whose campaigns started from January to April, 2016. Even though the attributes contain very basic information about the web pages, like the number of videos and images included, it is surprising that these are sufficient for solid prediction of success of the project.

✓ Send Data Automatically

Data Table

Info

1163 instances
15 features (no missing values)
Discrete class with 2 values (no missing values)
4 meta attributes (4.0% missing values)

Variables

✓ Show variable labels (if present)
Visualize numeric values
✓ Color by instance classes

Selection

✓ Select full rows

Restore Original Order

Report

✓ Send Automatically

	Funded	URL	Title	Year	Month	Type	Has FB
1	no	https://www....	Pixelstart: C...	2016	Apr	Art	1
2	no	https://www....	Smart shop I...	2016			
3	no	https://www....	Minimal Hau...	2016			
4	no	https://www....	NeoN: Alteri...	2016			
5	no	https://www....	Nintendo NE...	2016			
6	no	https://www....	Day and Nig...	2016			
7	no	https://www....	Fund an Art ...	2016			
8	no	https://www....	Trump that ...	2016			
9	yes	https://www....	Once Upon ...	2016			
10	yes	https://www....	Under the H...	2016			
11	yes	https://www....	KOKORO	2016			
12	yes	https://www....	Draw Cool S...	2016			
13	yes	https://www....	Ellefortheco...	2016			
14	yes	https://www....	PT Apparel	2016			
15	yes	https://www....	Epocha - Ha...	2016			
16	yes	https://www....	The Little AB...	2016			
17	yes	https://www....	Burl & Fur	2016			
18	yes	https://www....	Pens & Pedals	2016			
19	yes	https://www....	BCU Illustrat...	2016			

Distributions

Variable

- C Type
- C Has FB
- N Backed Projects
- N Previous Projects
- N Creator Desc Len
- N Title Len
- N Goal
- N Duration
- N Pledge Levels
- N Min Pledge Tiers
- N Max Pledge Tiers

Precision

2 50

Bin numeric variables into 10 bins

Group by

C Funded

Show relative frequencies

Show probabilities: (None)

Save Image
Report

Data Table

Displays attribute-value data in a spreadsheet.

Signals

Inputs:

- **Data**

  Attribute-valued data set.

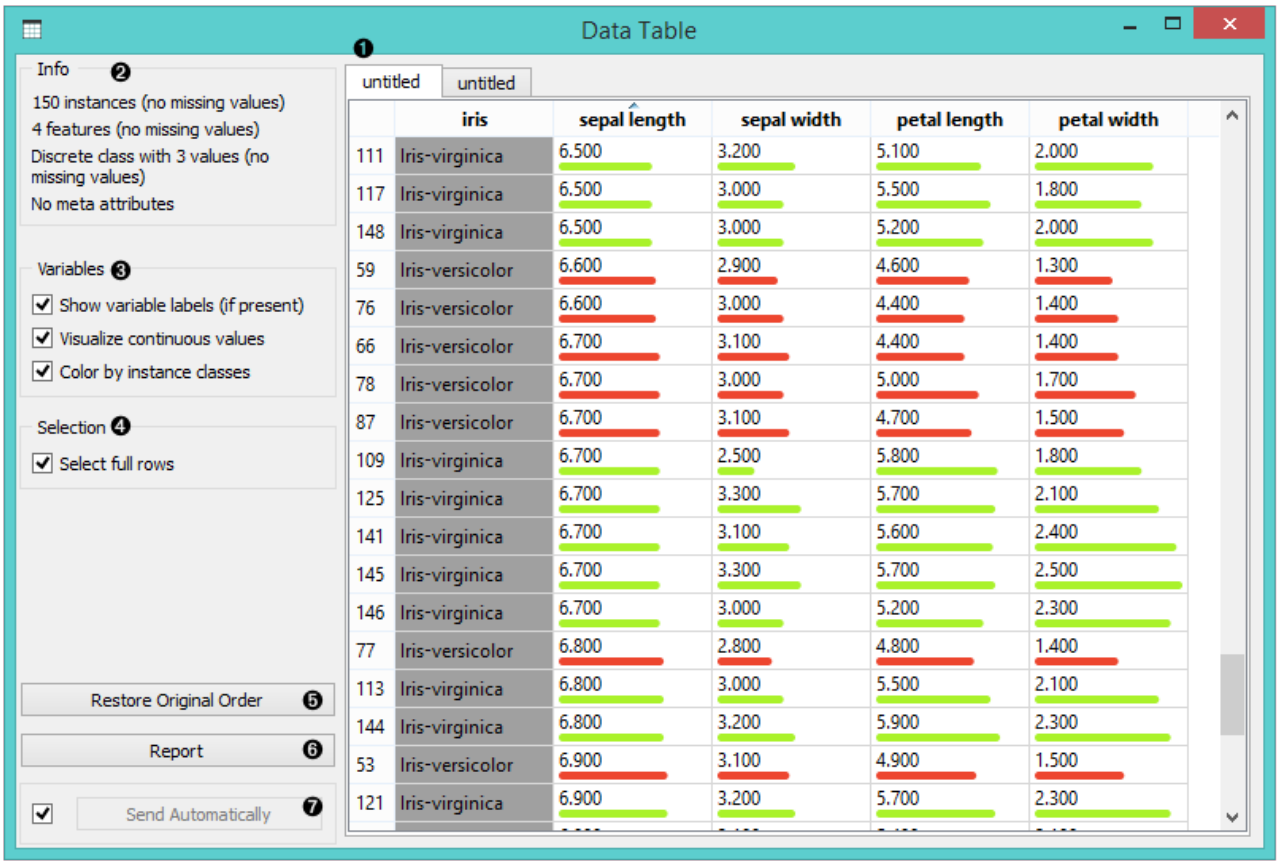
Outputs:

- **Selected Data**

  Selected data instances.

Description

The **Data Table** widget receives one or more data sets in its input and presents them as a spreadsheet. Data instances may be sorted by attribute values. The widget also supports manual selection of data instances.



150 instances (no missing values)
4 features (no missing values)
Discrete class with 3 values (no missing values)
No meta attributes

- [✓] Show variable labels (if present)
- [✓] Visualize continuous values
- [✓] Color by instance classes
- [✓] Select full rows
- [✓] Send Automatically

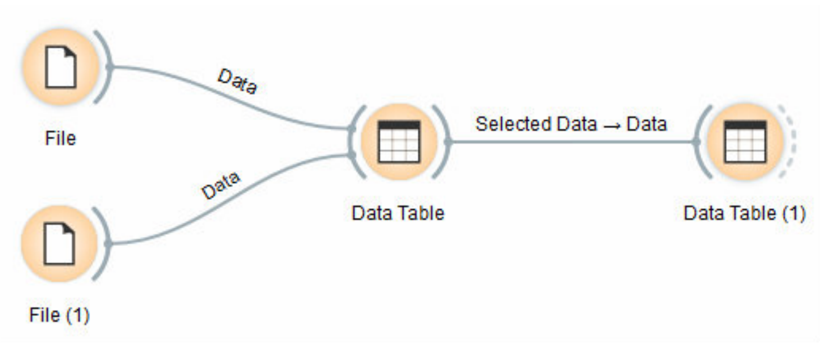
	iris	sepal length	sepal width	petal length	petal width
111	Iris-virginica	6.500	3.200	5.100	2.000
117	Iris-virginica	6.500	3.000	5.500	1.800
148	Iris-virginica	6.500	3.000	5.200	2.000
59	Iris-versicolor	6.600	2.900	4.600	1.300
76	Iris-versicolor	6.600	3.000	4.400	1.400
66	Iris-versicolor	6.700	3.100	4.400	1.400
78	Iris-versicolor	6.700	3.000	5.000	1.700
87	Iris-versicolor	6.700	3.100	4.700	1.500
109	Iris-virginica	6.700	2.500	5.800	1.800
125	Iris-virginica	6.700	3.300	5.700	2.100
141	Iris-virginica	6.700	3.100	5.600	2.400
145	Iris-virginica	6.700	3.300	5.700	2.500
146	Iris-virginica	6.700	3.000	5.200	2.300
77	Iris-versicolor	6.800	2.800	4.800	1.400
113	Iris-virginica	6.800	3.000	5.500	2.100
144	Iris-virginica	6.800	3.200	5.900	2.300
53	Iris-versicolor	6.900	3.100	4.900	1.500
121	Iris-virginica	6.900	3.200	5.700	2.300

1. The name of the data set (usually the input data file). Data instances are in rows and their attribute values in columns. In this example, the data set is sorted by the attribute "sepal length".
2. Info on current data set size and number and types of attributes
3. Values of continuous attributes can be visualized with bars; colors can be attributed to different classes.
4. Data instances (rows) can be selected and sent to the widget's output channel.

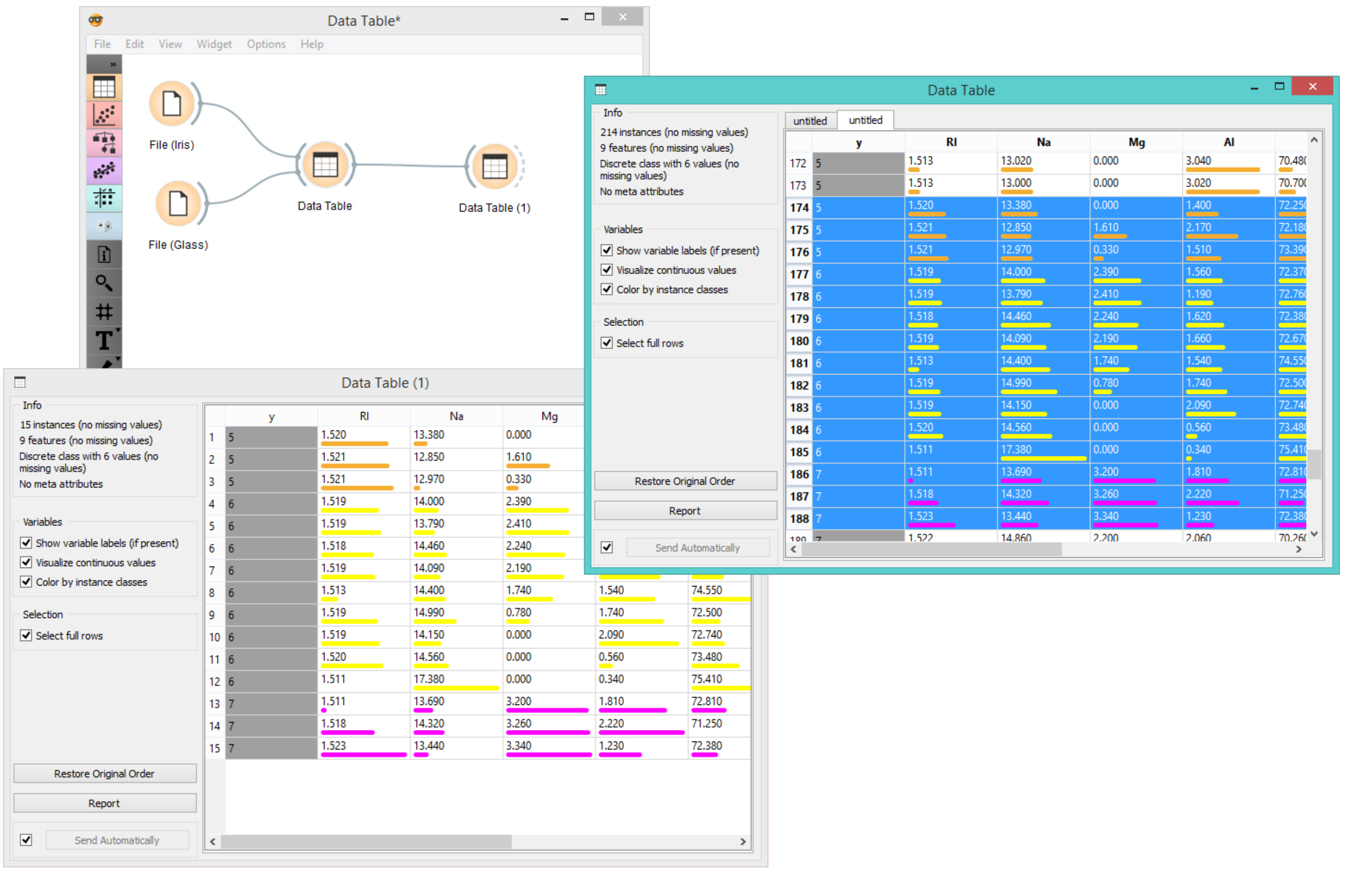
5. Use the *Restore Original Order* button to reorder data instances after attribute-based sorting.
6. Produce a report.
7. While auto-send is on, all changes will be automatically communicated to other widgets. Otherwise, press *Send Selected Rows*.

Example

We used two File widgets to read the *Iris* and *Glass* data set (provided in Orange distribution), and send them to the **Data Table** widget.



Selected data instances in the first **Data Table** are passed to the second **Data Table**. Notice that we can select which data set to view (iris or glass). Changing from one data set to another alters the communicated selection of data instances if *Commit on any change* is selected.



Data Table

Info
214 instances (no missing values)
9 features (no missing values)
Discrete class with 6 values (no missing values)
No meta attributes

Variables
☑ Show variable labels (if present)
☑ Visualize continuous values
☑ Color by instance classes

Selection
☑ Select full rows

Restore Original Order
Report
☑ Send Automatically

untitled | untitled

	y	RI	Na	Mg	Al	
172	5	1.513	13.020	0.000	3.040	70.48
173	5	1.513	13.000	0.000	3.020	70.70
174	5	1.520	13.380	0.000	1.400	72.25
175	5	1.521	12.850	1.610	2.170	72.18
176	5	1.521	12.970	0.330	1.510	73.39
177	6	1.519	14.000	2.390	1.560	72.37
178	6	1.519	13.790	2.410	1.190	72.76
179	6	1.518	14.460	2.240	1.620	72.38
180	6	1.519	14.090	2.190	1.660	72.67
181	6	1.513	14.400	1.740	1.540	74.55
182	6	1.519	14.990	0.780	1.740	72.50
183	6	1.519	14.150	0.000	2.090	72.74
184	6	1.520	14.560	0.000	0.560	73.48
185	6	1.511	17.380	0.000	0.340	75.41
186	7	1.511	13.690	3.200	1.810	72.81
187	7	1.518	14.320	3.260	2.220	71.25
188	7	1.523	13.440	3.340	1.230	72.38
[illegible]	7	1.522	14.860	2.200	2.060	70.26

Data Table (1)

Info
15 instances (no missing values)
9 features (no missing values)
Discrete class with 6 values (no missing values)
No meta attributes

Variables
☑ Show variable labels (if present)
☑ Visualize continuous values
☑ Color by instance classes

Selection
☑ Select full rows

Restore Original Order
Report
☑ Send Automatically

	y	RI	Na	Mg		
1	5	1.520	13.380	0.000		
2	5	1.521	12.850	1.610		
3	5	1.521	12.970	0.330		
4	6	1.519	14.000	2.390		
5	6	1.519	13.790	2.410		
6	6	1.518	14.460	2.240		
7	6	1.519	14.090	2.190		
8	6	1.513	14.400	1.740	1.540	74.550
9	6	1.519	14.990	0.780	1.740	72.500
10	6	1.519	14.150	0.000	2.090	72.740
11	6	1.520	14.560	0.000	0.560	73.480
12	6	1.511	17.380	0.000	0.340	75.410
13	7	1.511	13.690	3.200	1.810	72.810
14	7	1.518	14.320	3.260	2.220	71.250
15	7	1.523	13.440	3.340	1.230	72.380

Discretize

Discretizes continuous attributes from an input data set.

Signals

Inputs:

- **Data**

  Attribute-valued data set.

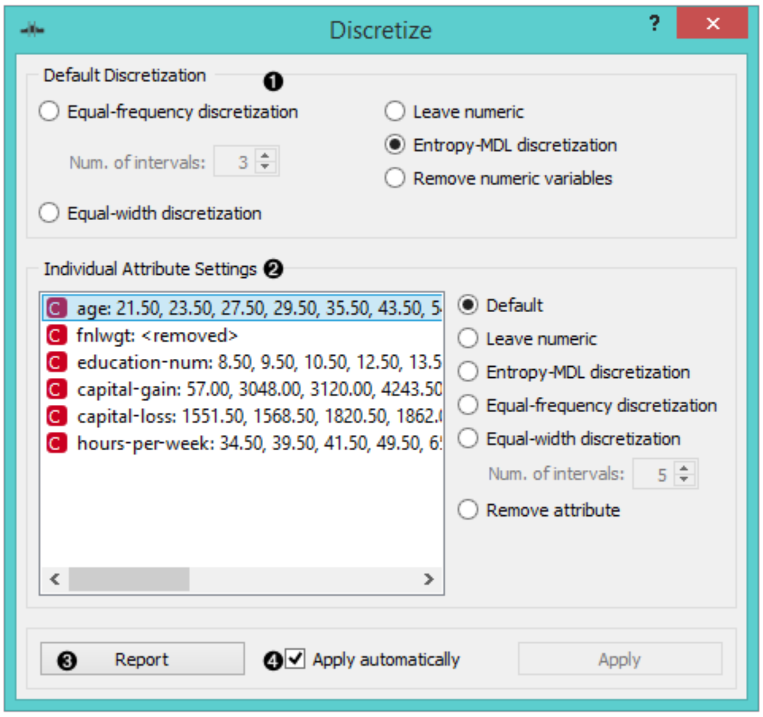
Outputs:

- **Data**

  A data set with discretized values.

Description

The **Discretize** widget discretizes continuous attributes with a selected method.



Discretize

Default Discretization 1

- Equal-frequency discretization
  - Num. of intervals: 3
- Equal-width discretization
- Leave numeric
- [✓ Entropy-MDL discretization]
- Remove numeric variables

Individual Attribute Settings 2

```
age: 21.50, 23.50, 27.50, 29.50, 35.50, 43.50, 5
fnlwgt: <removed>
education-num: 8.50, 9.50, 10.50, 12.50, 13.5
capital-gain: 57.00, 3048.00, 3120.00, 4243.50
capital-loss: 1551.50, 1568.50, 1820.50, 1862.
hours-per-week: 34.50, 39.50, 41.50, 49.50, 6
```

- [✓ Default]
- Leave numeric
- Entropy-MDL discretization
- Equal-frequency discretization
- Equal-width discretization
  - Num. of intervals: 5
- Remove attribute

3 Report | 4 [✓] Apply automatically | Apply

1. The basic version of the widget is rather simple. It allows choosing between three different discretizations.

   - Entropy-MDL, invented by Fayyad and Irani is a top-down discretization, which recursively splits the attribute at a cut maximizing information gain, until the gain is lower than the minimal description length of the cut. This discretization can result in an arbitrary number of intervals, including a single interval, in which case the attribute is discarded as useless (removed).
   - Equal-frequency splits the attribute into a given number of intervals, so that they each contain approximately the same number of instances.
   - Equal-width evenly splits the range between the smallest and the largest observed value. The *Number of intervals* can be set manually.
   - The widget can also be set to leave the attributes continuous or to remove them.

2. To treat attributes individually, go to **Individual Attribute Settings**. They show a specific discretization of each attribute and allow changes. First, the top left list shows the cut-off points for each attribute. In the snapshot, we used the entropy-MDL discretization, which determines the optimal number of intervals automatically; we can see it discretized the age into seven intervals with cut-offs at 21.50, 23.50, 27.50, 35.50, 43.50, 54.50 and 61.50, respectively, while the capital-gain got split into many intervals with several cut-offs. The final weight (fnlwgt), for instance, was left with a single interval and thus removed.

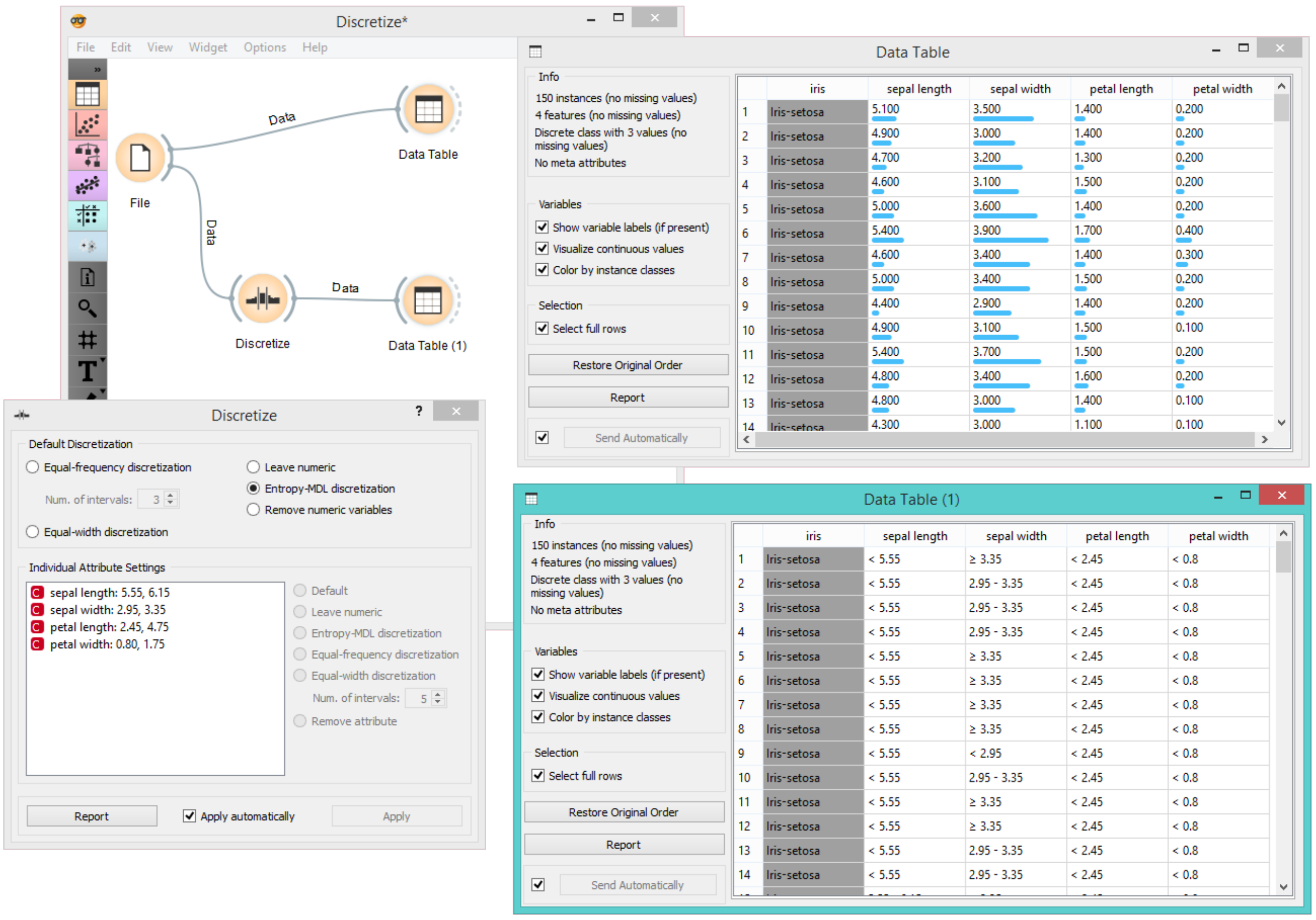
   On the right, we can select a specific discretization method for each attribute. Attribute *"fnlwgt"* would be removed by the MDL-based discretization, so to prevent its removal, we select the attribute and choose, for instance, **Equal-frequency discretization**. We could also choose to leave the attribute continuous.

3. Produce a report.

4. Tick *Apply automatically* for the widget to automatically commit changes. Alternatively, press *Apply*.

Example

In the schema below, we show the *Iris* data set with continuous attributes (as in the original data file) and with discretized attributes.



Discretize

Default Discretization

( ) Equal-frequency discretization
Num. of intervals: 3
( ) Equal-width discretization
( ) Leave numeric
(●) Entropy-MDL discretization
( ) Remove numeric variables

Individual Attribute Settings

- sepal length: 5.55, 6.15
- sepal width: 2.95, 3.35
- petal length: 2.45, 4.75
- petal width: 0.80, 1.75

( ) Default
( ) Leave numeric
( ) Entropy-MDL discretization
( ) Equal-frequency discretization
( ) Equal-width discretization
Num. of intervals: 5
( ) Remove attribute

Report | ✓ Apply automatically | Apply

Data Table

Info
150 instances (no missing values)
4 features (no missing values)
Discrete class with 3 values (no missing values)
No meta attributes

Variables
✓ Show variable labels (if present)
✓ Visualize continuous values
✓ Color by instance classes

Selection
✓ Select full rows

Restore Original Order
Report
✓ Send Automatically

	iris	sepal length	sepal width	petal length	petal width
1	Iris-setosa	5.100	3.500	1.400	0.200
2	Iris-setosa	4.900	3.000	1.400	0.200
3	Iris-setosa	4.700	3.200	1.300	0.200
4	Iris-setosa	4.600	3.100	1.500	0.200
5	Iris-setosa	5.000	3.600	1.400	0.200
6	Iris-setosa	5.400	3.900	1.700	0.400
7	Iris-setosa	4.600	3.400	1.400	0.300
8	Iris-setosa	5.000	3.400	1.500	0.200
9	Iris-setosa	4.400	2.900	1.400	0.200
10	Iris-setosa	4.900	3.100	1.500	0.100
11	Iris-setosa	5.400	3.700	1.500	0.200
12	Iris-setosa	4.800	3.400	1.600	0.200
13	Iris-setosa	4.800	3.000	1.400	0.100
14	Iris-setosa	4.300	3.000	1.100	0.100

Data Table (1)

Info
150 instances (no missing values)
4 features (no missing values)
Discrete class with 3 values (no missing values)
No meta attributes

Variables
✓ Show variable labels (if present)
✓ Visualize continuous values
✓ Color by instance classes

Selection
✓ Select full rows

Restore Original Order
Report
✓ Send Automatically

	iris	sepal length	sepal width	petal length	petal width
1	Iris-setosa	< 5.55	≥ 3.35	< 2.45	< 0.8
2	Iris-setosa	< 5.55	2.95 - 3.35	< 2.45	< 0.8
3	Iris-setosa	< 5.55	2.95 - 3.35	< 2.45	< 0.8
4	Iris-setosa	< 5.55	2.95 - 3.35	< 2.45	< 0.8
5	Iris-setosa	< 5.55	≥ 3.35	< 2.45	< 0.8
6	Iris-setosa	< 5.55	≥ 3.35	< 2.45	< 0.8
7	Iris-setosa	< 5.55	≥ 3.35	< 2.45	< 0.8
8	Iris-setosa	< 5.55	≥ 3.35	< 2.45	< 0.8
9	Iris-setosa	< 5.55	< 2.95	< 2.45	< 0.8
10	Iris-setosa	< 5.55	2.95 - 3.35	< 2.45	< 0.8
11	Iris-setosa	< 5.55	≥ 3.35	< 2.45	< 0.8
12	Iris-setosa	< 5.55	≥ 3.35	< 2.45	< 0.8
13	Iris-setosa	< 5.55	2.95 - 3.35	< 2.45	< 0.8
14	Iris-setosa	< 5.55	2.95 - 3.35	< 2.45	< 0.8

Edit Domain

Signals

Inputs:

- **Data**

  An input data set

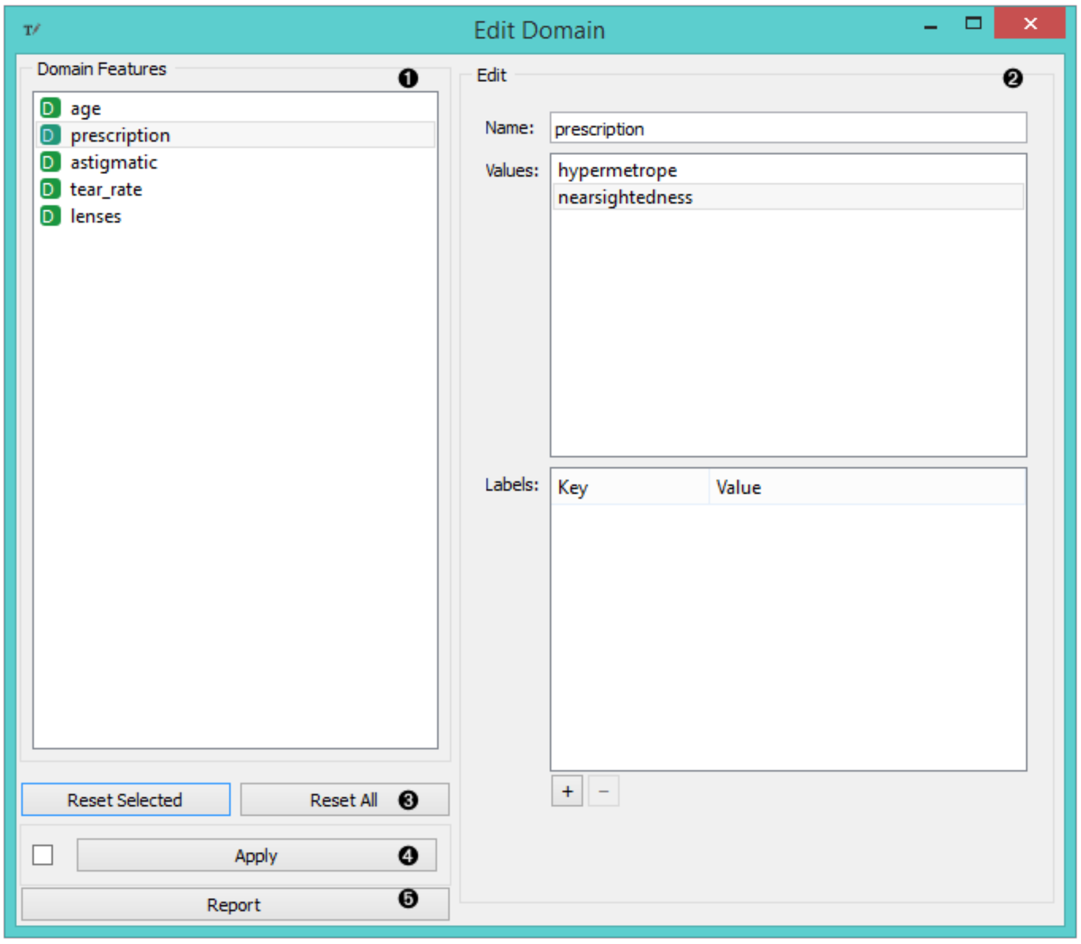
Outputs:

- **Data**

  An edited output data set

Description

This widget can be used to edit/change a data set's domain.

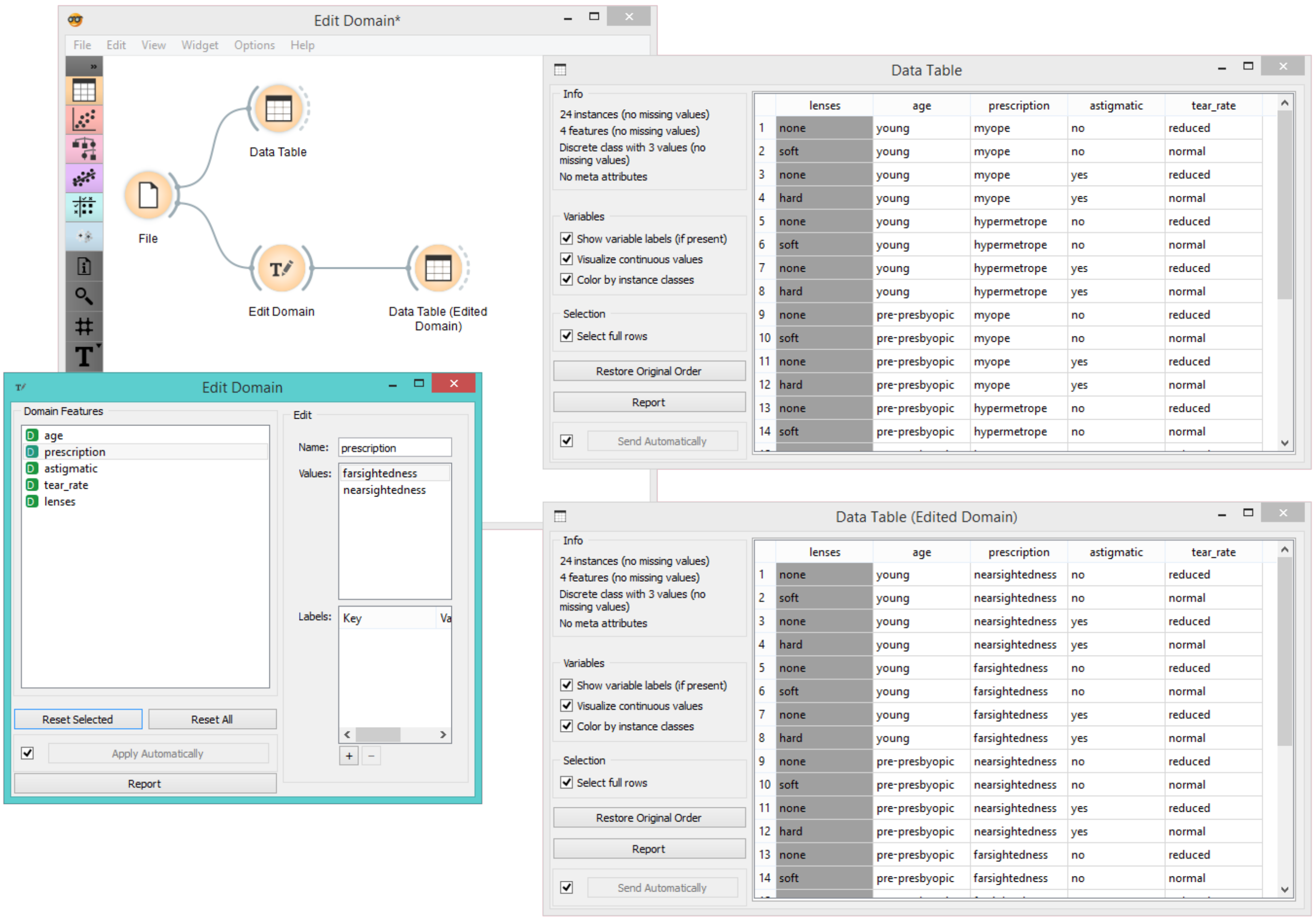


1. All features (including meta attributes) from the input data set are listed in the *Domain Features* list in the box on the left. Selecting one feature displays an editor on the right.
2. The name of the feature can be changed in the *Name* line edit. For *Discrete* features, value names can also be changed in the *Values* list box. Additonal feature annotations can be added/removed/edited in the *Labels* box. To add a new label, click the "+" button and add the *Key* and *Value* columns for the new entry. Selecting an existing label and pressing "-" will remove the annotation.

3. To revert the changes made to the feature, press the *Reset Selected* button in the *Reset* box while the feature is selected in the *Domain Features* list. Pressing *Reset All* will reset all features in the domain at the same time.
4. Pressing the *Apply* button will send the changed domain data set to the output channel.

Example

Below, we demonstrate how to simply edit an existing domain. We selected the *lenses.tab* data set and edited the *prescription* attribute. Where in the original we had the values *myope* and *hypermetrope*, we changed it into *nearsightedness* and *farsightedness* instead. For an easier comparison, we fed both the original and edited data into the Data Table widget.



Data Table

Info
24 instances (no missing values)
4 features (no missing values)
Discrete class with 3 values (no missing values)
No meta attributes

Variables
☑ Show variable labels (if present)
☑ Visualize continuous values
☑ Color by instance classes

Selection
☑ Select full rows

Restore Original Order
Report
☑ Send Automatically

	lenses	age	prescription	astigmatic	tear_rate
1	none	young	myope	no	reduced
2	soft	young	myope	no	normal
3	none	young	myope	yes	reduced
4	hard	young	myope	yes	normal
5	none	young	hypermetrope	no	reduced
6	soft	young	hypermetrope	no	normal
7	none	young	hypermetrope	yes	reduced
8	hard	young	hypermetrope	yes	normal
9	none	pre-presbyopic	myope	no	reduced
10	soft	pre-presbyopic	myope	no	normal
11	none	pre-presbyopic	myope	yes	reduced
12	hard	pre-presbyopic	myope	yes	normal
13	none	pre-presbyopic	hypermetrope	no	reduced
14	soft	pre-presbyopic	hypermetrope	no	normal

Data Table (Edited Domain)

Info
24 instances (no missing values)
4 features (no missing values)
Discrete class with 3 values (no missing values)
No meta attributes

Variables
☑ Show variable labels (if present)
☑ Visualize continuous values
☑ Color by instance classes

Selection
☑ Select full rows

Restore Original Order
Report
☑ Send Automatically

	lenses	age	prescription	astigmatic	tear_rate
1	none	young	nearsightedness	no	reduced
2	soft	young	nearsightedness	no	normal
3	none	young	nearsightedness	yes	reduced
4	hard	young	nearsightedness	yes	normal
5	none	young	farsightedness	no	reduced
6	soft	young	farsightedness	no	normal
7	none	young	farsightedness	yes	reduced
8	hard	young	farsightedness	yes	normal
9	none	pre-presbyopic	nearsightedness	no	reduced
10	soft	pre-presbyopic	nearsightedness	no	normal
11	none	pre-presbyopic	nearsightedness	yes	reduced
12	hard	pre-presbyopic	nearsightedness	yes	normal
13	none	pre-presbyopic	farsightedness	no	reduced
14	soft	pre-presbyopic	farsightedness	no	normal

Feature Constructor

Add new features to your data set.

Signals

Inputs:

- **Data**

A data set

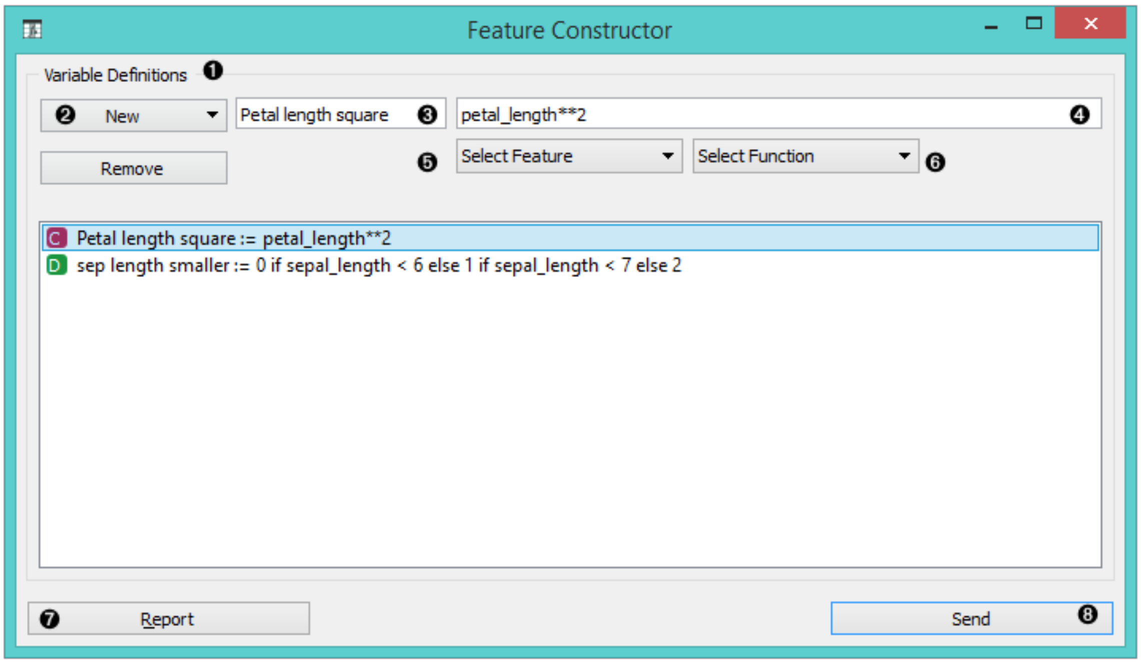
Outputs:

- **Data**

A modified data set

Description

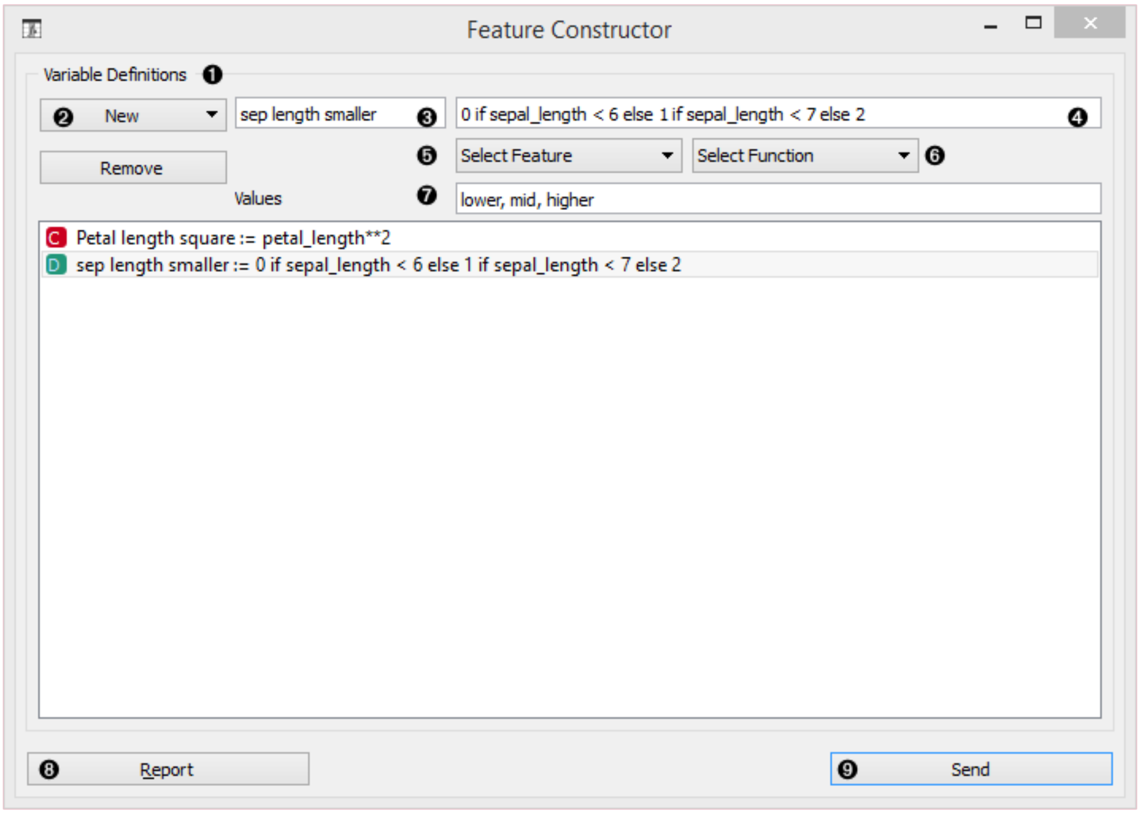
The **Feature Constructor** allows you to manually add features (columns) into your data set. The new feature can be a computation of an existing one or a combination of several (addition, subtraction, etc.). You can choose what type of feature it will be (discrete, continuous or string) and what its parameters are (name, value, expression). For continuous variables you only have to construct an expression in Python.



```
C Petal length square := petal_length**2
D sep length smaller := 0 if sepal_length < 6 else 1 if sepal_length < 7 else 2
```

1. List of constructed variables
2. Add or remove variables.
3. New feature name
4. Expression in Python
5. Select a feature.
6. Select a function.
7. Produce a report.
8. Press *Send* to communicate changes.

For discrete variables, however, there's a bit more work. First add or remove the values you want for the new feature. Then select the base value and the expression. In the example below, we have constructed an expression with 'if lower than' and defined three conditions; the program ascribes 0 (which we renamed to lower) if the original value is lower than 6, 1 (mid) if it is lower than 7 and 2 (higher) for all the other values. Notice that we use an underscore for the feature name (e.g. petal_length).

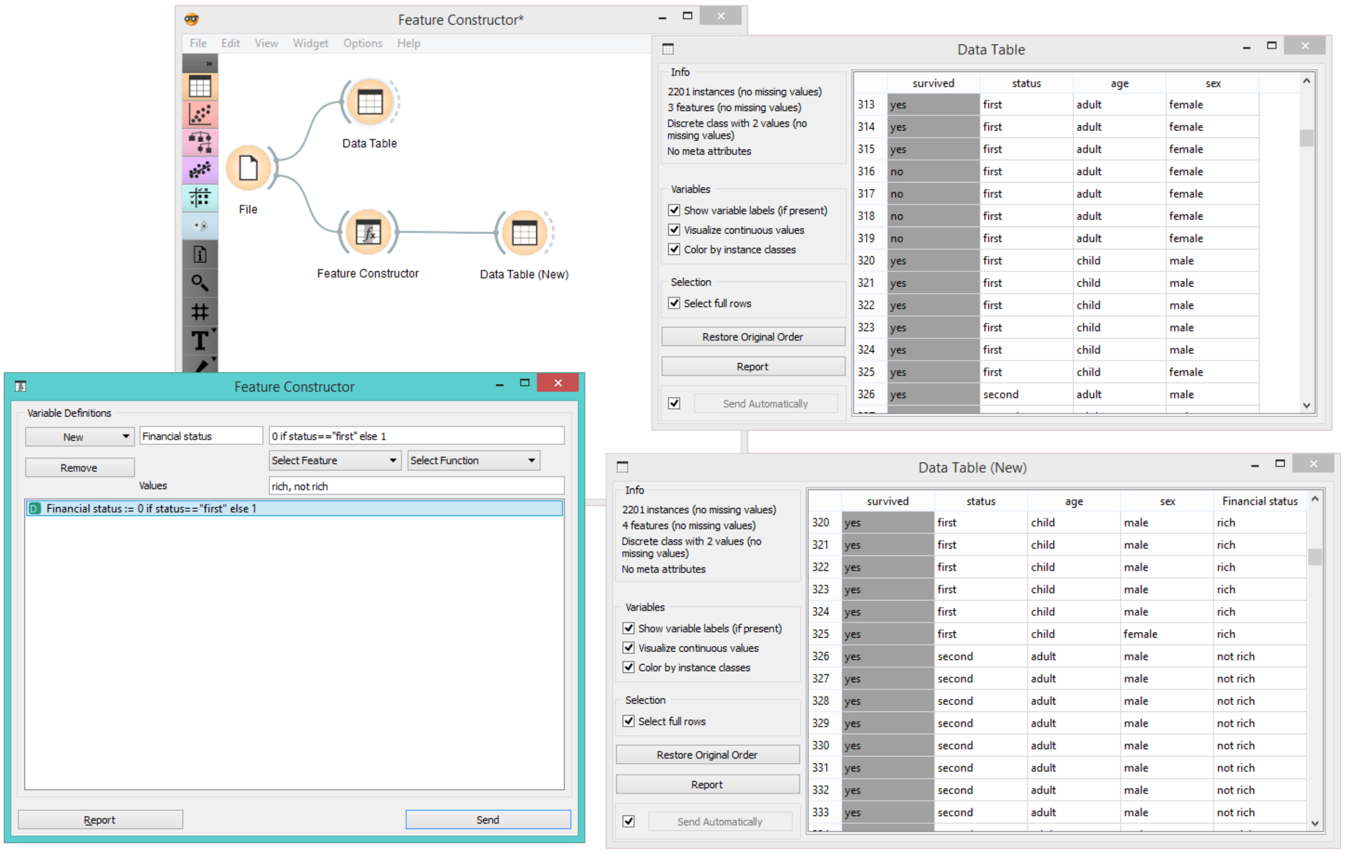


Petal length square := petal_length**2
sep length smaller := 0 if sepal_length < 6 else 1 if sepal_length < 7 else 2

1. List of variable definitions
2. Add or remove variables
3. New feature name
4. Expression in Python
5. Select a feature.
6. Select a function.
7. Assign values.
8. Produce a report.
9. Press *Send* to communicate changes.

Example

With the **Feature Constructor** you can easily adjust or combine existing features into new ones. Below, we added one new discrete feature to the *Titanic* data set. We created a new attribute called *Financial status* and set the values to be *rich* if the person belongs to the first class (status = first) and *not rich* for everybody else. We can see the new data set with Data Table widget.



Feature Constructor

Variable Definitions

New | Financial status | 0 if status=="first" else 1

Select Feature | Select Function

Remove

Values: rich, not rich

Financial status := 0 if status=="first" else 1

Report | Send

Data Table

Info
2201 instances (no missing values)
3 features (no missing values)
Discrete class with 2 values (no missing values)
No meta attributes

Variables
☑ Show variable labels (if present)
☑ Visualize continuous values
☑ Color by instance classes

Selection
☑ Select full rows

Restore Original Order

Report

☑ Send Automatically

	survived	status	age	sex
313	yes	first	adult	female
314	yes	first	adult	female
315	yes	first	adult	female
316	no	first	adult	female
317	no	first	adult	female
318	no	first	adult	female
319	no	first	adult	female
320	yes	first	child	male
321	yes	first	child	male
322	yes	first	child	male
323	yes	first	child	male
324	yes	first	child	male
325	yes	first	child	female
326	yes	second	adult	male

Data Table (New)

Info
2201 instances (no missing values)
4 features (no missing values)
Discrete class with 2 values (no missing values)
No meta attributes

Variables
☑ Show variable labels (if present)
☑ Visualize continuous values
☑ Color by instance classes

Selection
☑ Select full rows

Restore Original Order

Report

☑ Send Automatically

	survived	status	age	sex	Financial status
320	yes	first	child	male	rich
321	yes	first	child	male	rich
322	yes	first	child	male	rich
323	yes	first	child	male	rich
324	yes	first	child	male	rich
325	yes	first	child	female	rich
326	yes	second	adult	male	not rich
327	yes	second	adult	male	not rich
328	yes	second	adult	male	not rich
329	yes	second	adult	male	not rich
330	yes	second	adult	male	not rich
331	yes	second	adult	male	not rich
332	yes	second	adult	male	not rich
333	yes	second	adult	male	not rich

Hints

If you are unfamiliar with Python math language, here's a quick introduction.

- +, - to add, subtract
- * to multiply
- / to divide
- % to divide and return the remainder
- ** for exponent (for square root square by 0.5)
- // for floor division
- <, >, <=, >= less than, greater than, less or equal, greater or equal
- == for equal
- != for not equal

As in the example: (*value*) if (*feature name*) < (*value*), else (*value*) if (*feature name*) < (*value*), else (*value*)

[Use value 1 if feature is less than specified value, else use value 2 if feature is less than specified value 2, else use value 3.]

See more here.

File

Reads attribute-value data from an input file.

Signals

Inputs:

- (None)

Outputs:

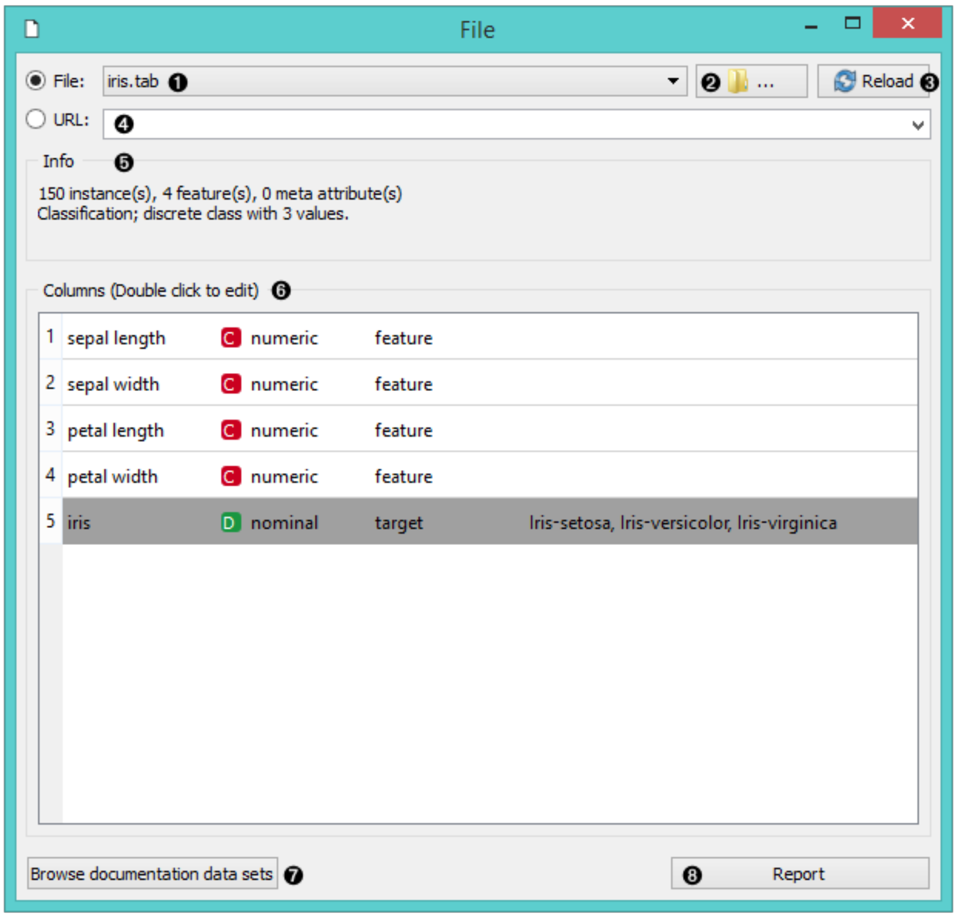
- **Data**

  Attribute-valued data from the input file

Description

The **File** widget reads the input data file (data table with data instances) and sends the data set to its output channel. The history of most recently opened files is maintained in the widget. The widget also includes a directory with sample data sets that come pre-installed with Orange.

The widget reads data from Excel (**.xlsx**), simple tab-delimited (**.txt**), comma-separated files (**.csv**) or URLs.


File

● File: iris.tab ❶ ❷ ... Reload ❸

○ URL: ❹

Info ❺

150 instance(s), 4 feature(s), 0 meta attribute(s)
Classification; discrete class with 3 values.

Columns (Double click to edit) ❻

1	sepal length	C	numeric	feature	
2	sepal width	C	numeric	feature	
3	petal length	C	numeric	feature	
4	petal width	C	numeric	feature	
5	iris	D	nominal	target	Iris-setosa, Iris-versicolor, Iris-virginica

Browse documentation data sets ❼

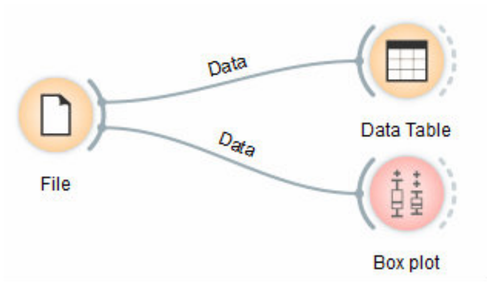
❽ Report

1. Browse through previously opened data files, or load any of the sample ones.
2. Browse for a data file.
3. Reloads currently selected data file.
4. Insert data from URL adresses, including data from Google Sheets.

5. Information on the loaded data set: data set size, number and types of data features.
6. Additional information on the features in the data set. Features can be edited by double-clicking on them. The user can change the attribute names, select the type of variable per each attribute (*Continuous*, *Nominal*, *String*, *Datetime*), and choose how to further define the attributes (as *Features*, *Targets* or *Meta*). The user can also decide to ignore an attribute.
7. Browse documentation data sets.
8. Produce a report.

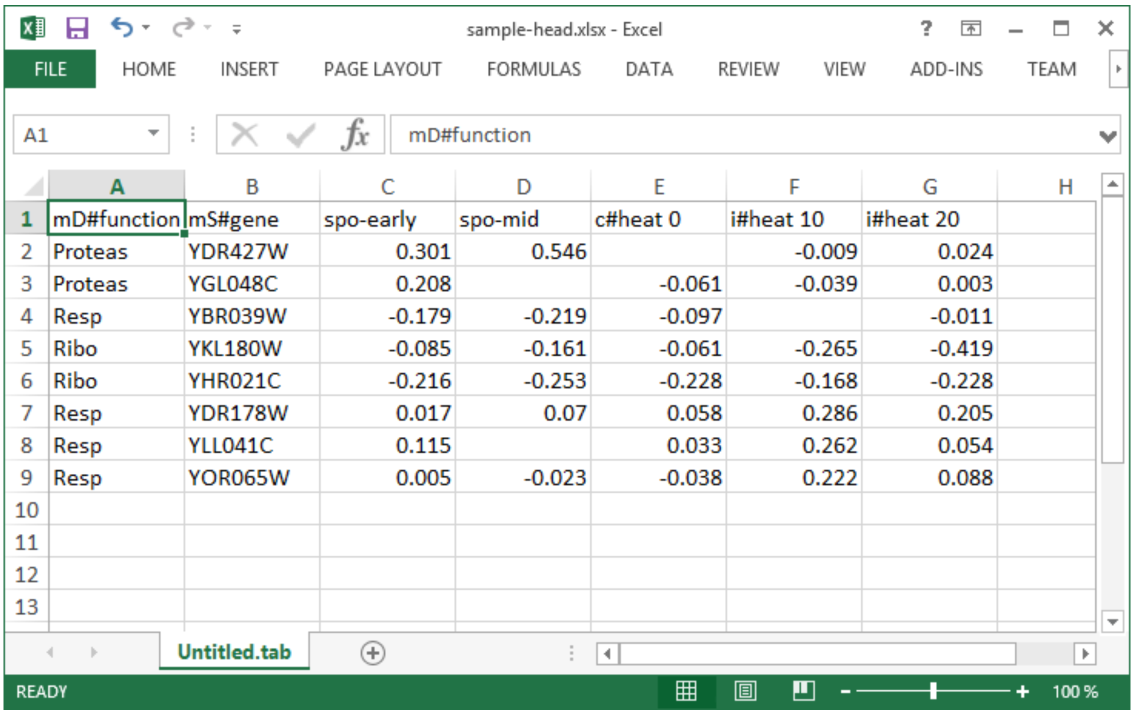
Example

Most Orange workflows would probably start with the **File** widget. In the schema below, the widget is used to read the data that is sent to both the Data Table and the Box Plot widget.



Loading your data

- Orange can import any comma, .xlsx or tab-delimited data file or URL. Use the File widget and then, if needed, select class and meta attributes.
- To specify the domain and the type of the attribute, attribute names can be preceded with a label followed by a hash. Use c for class and m for meta attribute, i to ignore a column, and C, D, S for continuous, discrete and string attribute types. Examples: C#mpg, mS#name, i#dummy. Make sure to set **Import Options** in File widget and set the header to **Orange simplified header**.
- Orange's native format is a tab-delimited text file with three header rows. The first row contains attribute names, the second the type (**continuous**, **discrete** or **string**), and the third the optional element (**class**, **meta** or **string**).



	A	B	C	D	E	F	G	H
1	mD#function	mS#gene	spo-early	spo-mid	c#heat 0	i#heat 10	i#heat 20	
2	Proteas	YDR427W	0.301	0.546		-0.009	0.024	
3	Proteas	YGL048C	0.208		-0.061	-0.039	0.003	
4	Resp	YBR039W	-0.179	-0.219	-0.097		-0.011	
5	Ribo	YKL180W	-0.085	-0.161	-0.061	-0.265	-0.419	
6	Ribo	YHR021C	-0.216	-0.253	-0.228	-0.168	-0.228	
7	Resp	YDR178W	0.017	0.07	0.058	0.286	0.205	
8	Resp	YLL041C	0.115		0.033	0.262	0.054	
9	Resp	YOR065W	0.005	-0.023	-0.038	0.222	0.088	
10								
11								
12								
13								

Read more on loading your data here.

Image Viewer

Displays images that come with a data set.

Signals

Inputs:

- **Data**

  A data set with images.

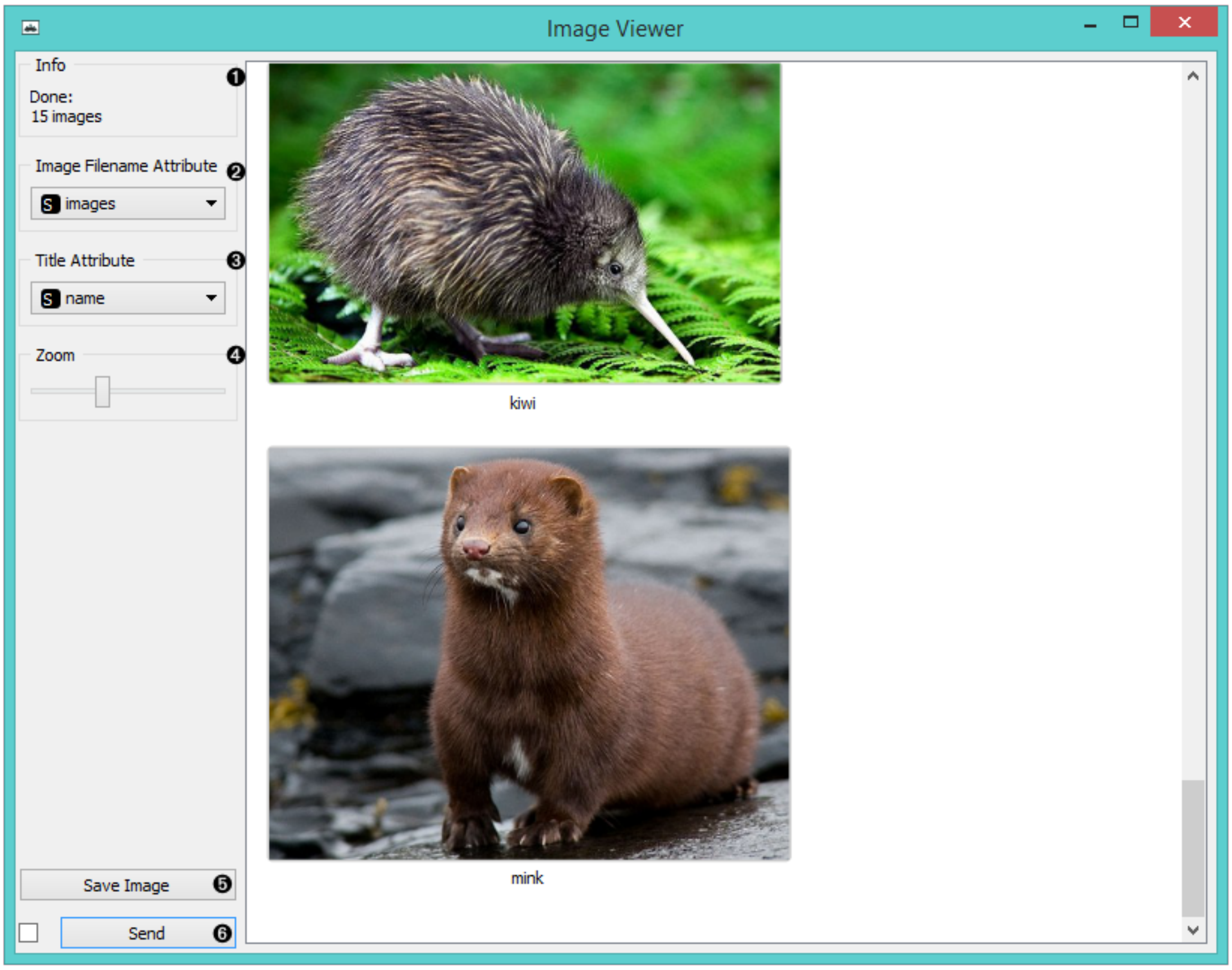
Outputs:

- **Data**

  Images that come with the data.

Description

The **Image Viewer** widget can display images from a data set, which are stored locally or on the internet. It can be used for image comparison, while looking for similarities or discrepancies between selected data instances (e.g. bacterial growth or bitmap representations of handwriting).



1. Information on the data set
2. Select the column with image data (links).
3. Select the column with image titles.
4. Zoom in or out.
5. Saves the visualization in a file.
6. Tick the box on the left to commit changes automatically. Alternatively, click *Send*.

Examples

A very simple way to use this widget is to connect the File widget with **Image Viewer** and see all the images that come with your data set.

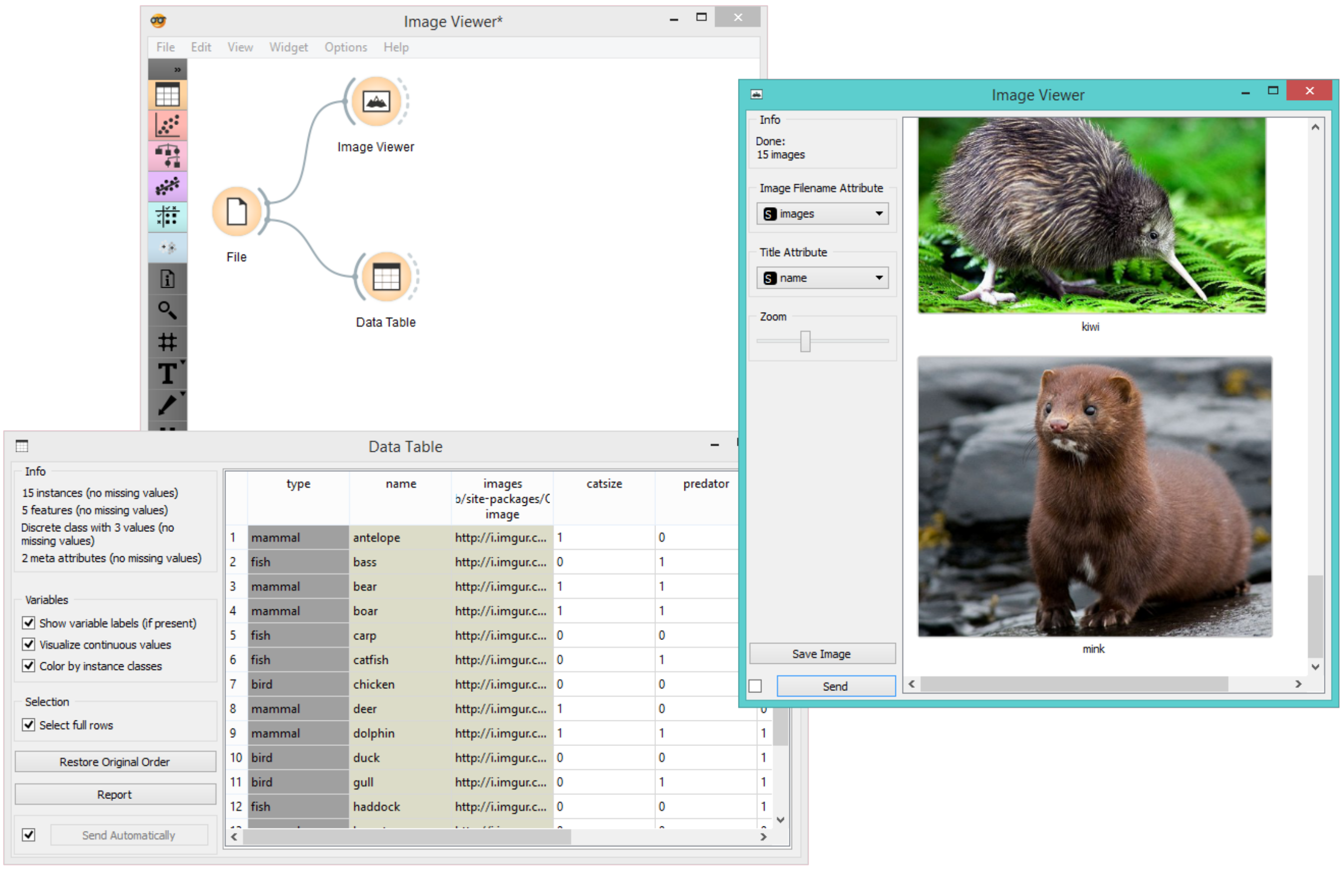


Image Viewer

Info
Done:
15 images

Image Filename Attribute: images

Title Attribute: name

Zoom

Save Image

☐ Send

Data Table

Info
15 instances (no missing values)
5 features (no missing values)
Discrete class with 3 values (no missing values)
2 meta attributes (no missing values)

Variables
✓ Show variable labels (if present)
✓ Visualize continuous values
✓ Color by instance classes

Selection
✓ Select full rows

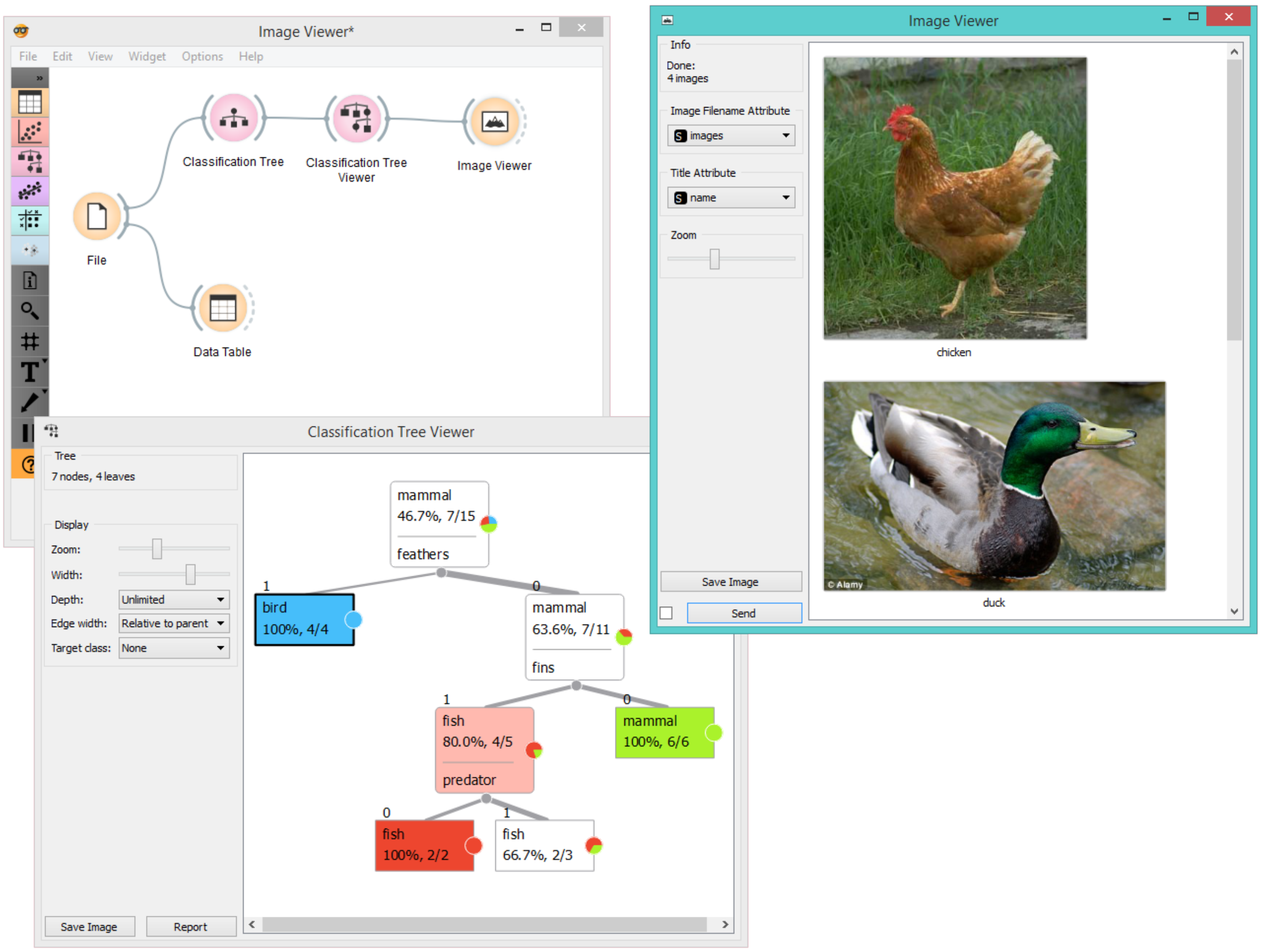
Restore Original Order

Report

✓ Send Automatically

	type	name	images b/site-packages/C image	catsize	predator	
1	mammal	antelope	http://i.imgur.c...	1	0	
2	fish	bass	http://i.imgur.c...	0	1	
3	mammal	bear	http://i.imgur.c...	1	1	
4	mammal	boar	http://i.imgur.c...	1	1	
5	fish	carp	http://i.imgur.c...	0	0	
6	fish	catfish	http://i.imgur.c...	0	1	
7	bird	chicken	http://i.imgur.c...	0	0	
8	mammal	deer	http://i.imgur.c...	1	0	0
9	mammal	dolphin	http://i.imgur.c...	1	1	1
10	bird	duck	http://i.imgur.c...	0	0	1
11	bird	gull	http://i.imgur.c...	0	1	1
12	fish	haddock	http://i.imgur.c...	0	0	1

Alternatively, you can visualize only selected instances, as shown in the example below.



Tree
7 nodes, 4 leaves

Display
Zoom:
Width:
Depth: Unlimited
Edge width: Relative to parent
Target class: None

Info
Done:
4 images

Image Filename Attribute: images

Title Attribute: name

Zoom

Impute

Replaces unknown values in the data.

Signals

Inputs

- **Data**

  A data set.

- **Learner for Imputation**

  A learning algorithm to be used when values are imputed with a predictive model. This algorithm, if given, substitutes the default (1-NN).

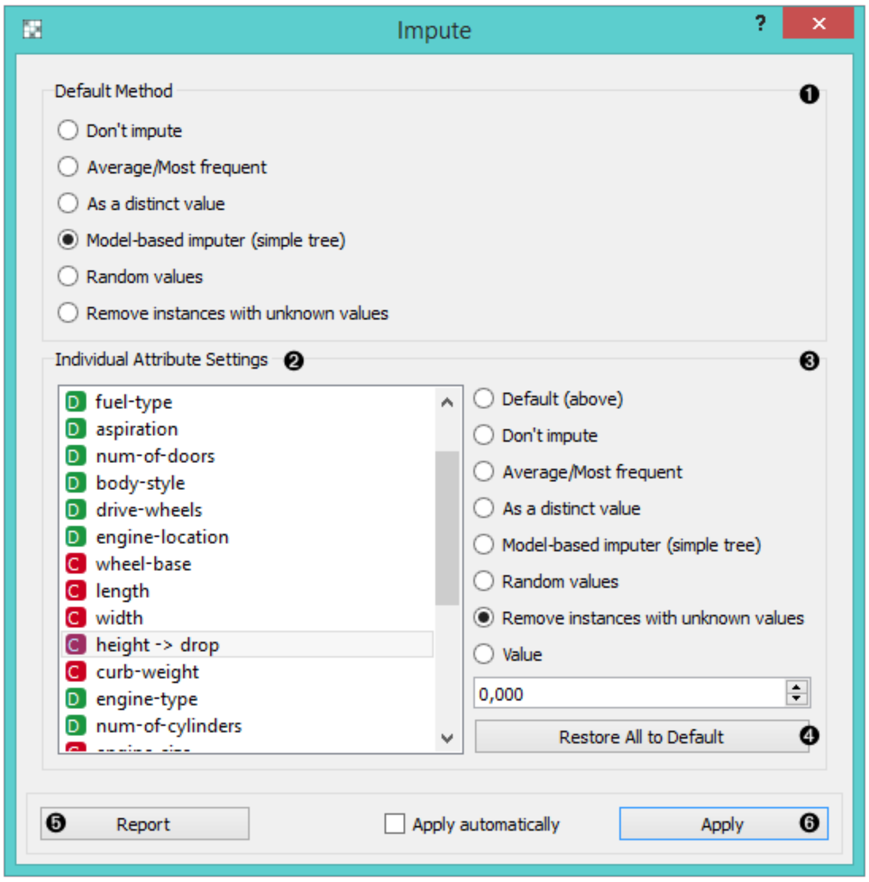
Outputs

- **Data**

  The same data set as in the input, but with the missing values imputed.

Description

Some Orange's algorithms and visualizations cannot handle unknown values in the data. This widget does what statisticians call imputation: it substitutes missing values by values either computed from the data or set by the user.



Impute

Default Method ❶
- Don't impute
- Average/Most frequent
- As a distinct value
- [✓ Model-based imputer (simple tree)]
- Random values
- Remove instances with unknown values

Individual Attribute Settings ❷ ❸

- D fuel-type
- D aspiration
- D num-of-doors
- D body-style
- D drive-wheels
- D engine-location
- C wheel-base
- C length
- C width
- C height -> drop
- C curb-weight
- D engine-type
- D num-of-cylinders

- Default (above)
- Don't impute
- Average/Most frequent
- As a distinct value
- Model-based imputer (simple tree)
- Random values
- [✓ Remove instances with unknown values]
- Value

0,000

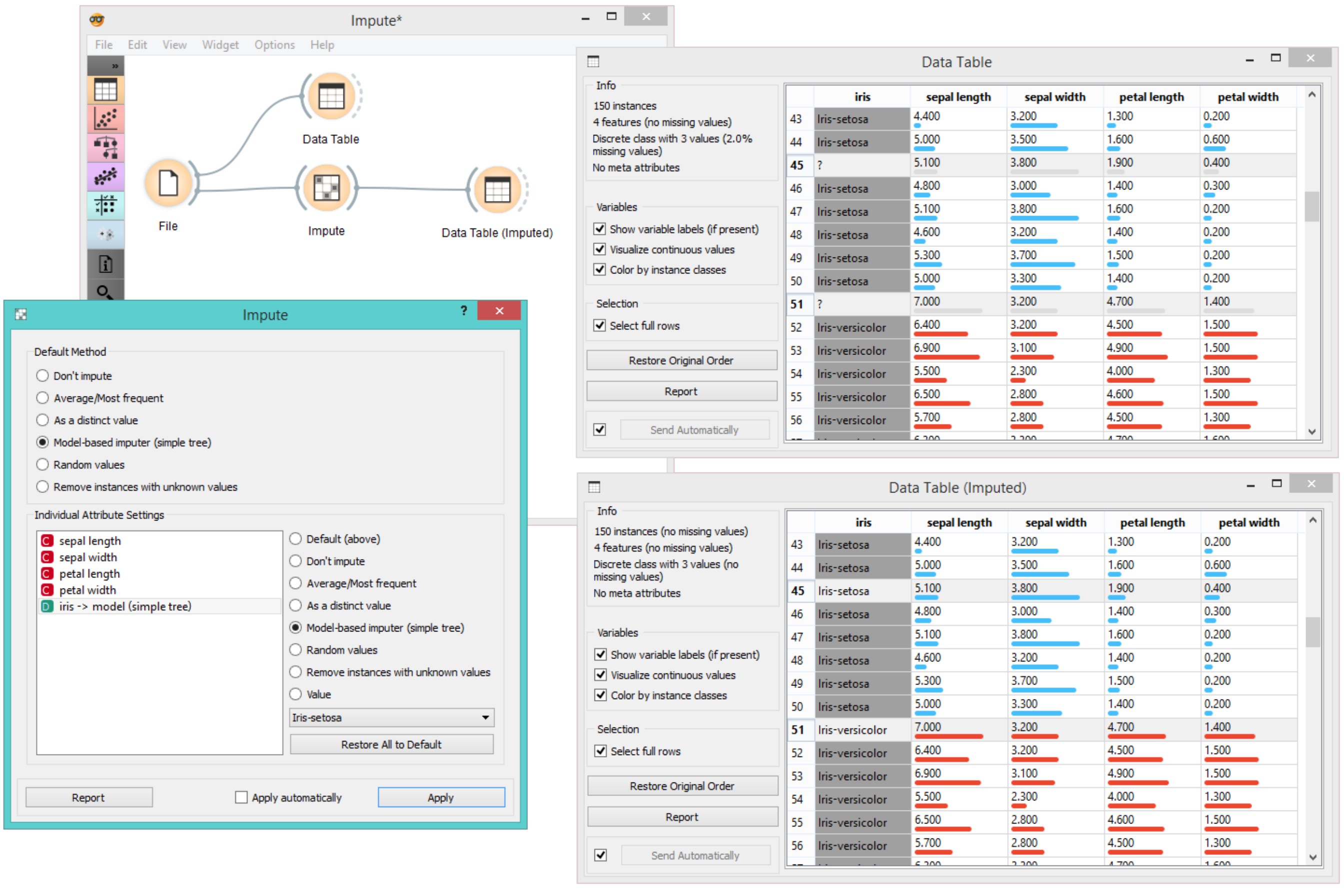
Restore All to Default ❹

❺ Report | Apply automatically | Apply ❻

1. In the top-most box, *Default method*, the user can specify a general imputation technique for all attributes.
   - **Don't Impute** does nothing with the missing values.
   - **Average/Most-frequent** uses the average value (for continuous attributes) or the most common value (for discrete attributes).
   - **As a distinct value** creates new values to substitute the missing ones.
   - **Model-based imputer** constructs a model for predicting the missing value, based on values of other attributes; a separate model is constructed for each attribute. The default model is 1-NN learner, which takes the value from the most similar example (this is sometimes referred to as hot deck imputation). This algorithm can be substituted by one that the user connects to the input signal Learner for Imputation. Note, however, that if there are discrete and continuous attributes in the data, the algorithm needs to be capable of handling them both; at the moment only 1-NN learner can do that. (In the future, when Orange has more regressors, the Impute widget may have separate input signals for discrete and continuous models.)
   - **Random values** computes the distributions of values for each attribute and then imputes by picking random values from them.
   - **Remove examples with missing values** removes the example containing missing values. This check also applies to the class attribute if *Impute class values* is checked.
2. It is possible to specify individual treatment for each attribute, which overrides the default treatment set. One can also specify a manually defined value used for imputation. In the screenshot, we decided not to impute the values of "*normalized-losses*" and "*make*", the missing values of "*aspiration*" will be replaced by random values, while the missing values of "*body-style*" and "*drive-wheels*" are replaced by "*hatchback*" and "*fwd*", respectively. If the values of "*length*", "*width*" or "*height*" are missing, the example is discarded. Values of all other attributes use the default method set above (model-based imputer, in our case).
3. The imputation methods for individual attributes are the same as default. methods.
4. *Restore All to Default* resets the individual attribute treatments to default.
5. Produce a report.
6. All changes are committed immediately if *Apply automatically* is checked. Otherwise, *Apply* needs to be ticked to apply any new settings.

Example

To demonstrate how the **Impute** widget works, we played around with the *Iris* data set and deleted some of the data. We used the **Impute** widget and selected the *Model-based imputer* to impute the missing values. In another Data Table, we see how the question marks turned into distinct values ("Iris-setosa, "Iris-versicolor").



Impute

Default Method
- ( ) Don't impute
- ( ) Average/Most frequent
- ( ) As a distinct value
- (●) Model-based imputer (simple tree)
- ( ) Random values
- ( ) Remove instances with unknown values

Individual Attribute Settings
- C sepal length
- C sepal width
- C petal length
- C petal width
- D iris -> model (simple tree)

- ( ) Default (above)
- ( ) Don't impute
- ( ) Average/Most frequent
- ( ) As a distinct value
- (●) Model-based imputer (simple tree)
- ( ) Random values
- ( ) Remove instances with unknown values
- ( ) Value

Iris-setosa

Restore All to Default

Report | ☐ Apply automatically | Apply

Data Table

Info
150 instances
4 features (no missing values)
Discrete class with 3 values (2.0% missing values)
No meta attributes

Variables
- ☑ Show variable labels (if present)
- ☑ Visualize continuous values
- ☑ Color by instance classes

Selection
- ☑ Select full rows

Restore Original Order
Report
☑ Send Automatically

	iris	sepal length	sepal width	petal length	petal width
43	Iris-setosa	4.400	3.200	1.300	0.200
44	Iris-setosa	5.000	3.500	1.600	0.600
45	?	5.100	3.800	1.900	0.400
46	Iris-setosa	4.800	3.000	1.400	0.300
47	Iris-setosa	5.100	3.800	1.600	0.200
48	Iris-setosa	4.600	3.200	1.400	0.200
49	Iris-setosa	5.300	3.700	1.500	0.200
50	Iris-setosa	5.000	3.300	1.400	0.200
51	?	7.000	3.200	4.700	1.400
52	Iris-versicolor	6.400	3.200	4.500	1.500
53	Iris-versicolor	6.900	3.100	4.900	1.500
54	Iris-versicolor	5.500	2.300	4.000	1.300
55	Iris-versicolor	6.500	2.800	4.600	1.500
56	Iris-versicolor	5.700	2.800	4.500	1.300

Data Table (Imputed)

Info
150 instances (no missing values)
4 features (no missing values)
Discrete class with 3 values (no missing values)
No meta attributes

Variables
- ☑ Show variable labels (if present)
- ☑ Visualize continuous values
- ☑ Color by instance classes

Selection
- ☑ Select full rows

Restore Original Order
Report
☑ Send Automatically

	iris	sepal length	sepal width	petal length	petal width
43	Iris-setosa	4.400	3.200	1.300	0.200
44	Iris-setosa	5.000	3.500	1.600	0.600
45	Iris-setosa	5.100	3.800	1.900	0.400
46	Iris-setosa	4.800	3.000	1.400	0.300
47	Iris-setosa	5.100	3.800	1.600	0.200
48	Iris-setosa	4.600	3.200	1.400	0.200
49	Iris-setosa	5.300	3.700	1.500	0.200
50	Iris-setosa	5.000	3.300	1.400	0.200
51	Iris-versicolor	7.000	3.200	4.700	1.400
52	Iris-versicolor	6.400	3.200	4.500	1.500
53	Iris-versicolor	6.900	3.100	4.900	1.500
54	Iris-versicolor	5.500	2.300	4.000	1.300
55	Iris-versicolor	6.500	2.800	4.600	1.500
56	Iris-versicolor	5.700	2.800	4.500	1.300

Merge Data

Merges two data sets, based on values of selected attributes.

Signals

Inputs:

- **Data**

  Attribute-valued data set.

- **Extra Data**

  Attribute-valued data set.

Outputs:

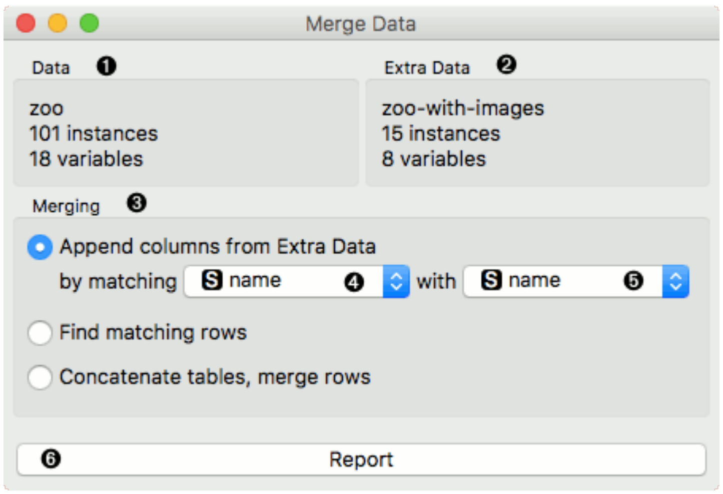
- **Data**

  Instances from input data to which attributes from input extra data are added.

Description

The **Merge Data** widget is used to horizontally merge two data sets, based on values of selected attributes. In the input, two data sets are required, data and extra data. The widget allows selection of an attribute from each domain, which will be used to perform the merging. The widget produces one output. It corresponds to instances from the input data to which attributes from input extra data are appended.

Merging is done by values of selected (merging) attributes. First, the value of the merging attribute from Data is taken and instances from Extra Data are searched for matching values. If more than a single instance from Extra Data was to be found, the attribute is removed from available merging attributes.


Merge Data

Data ❶
zoo
101 instances
18 variables

Extra Data ❷
zoo-with-images
15 instances
8 variables

Merging ❸
[✓ Append columns from Extra Data]
by matching name ❹ with name ❺
Find matching rows
Concatenate tables, merge rows

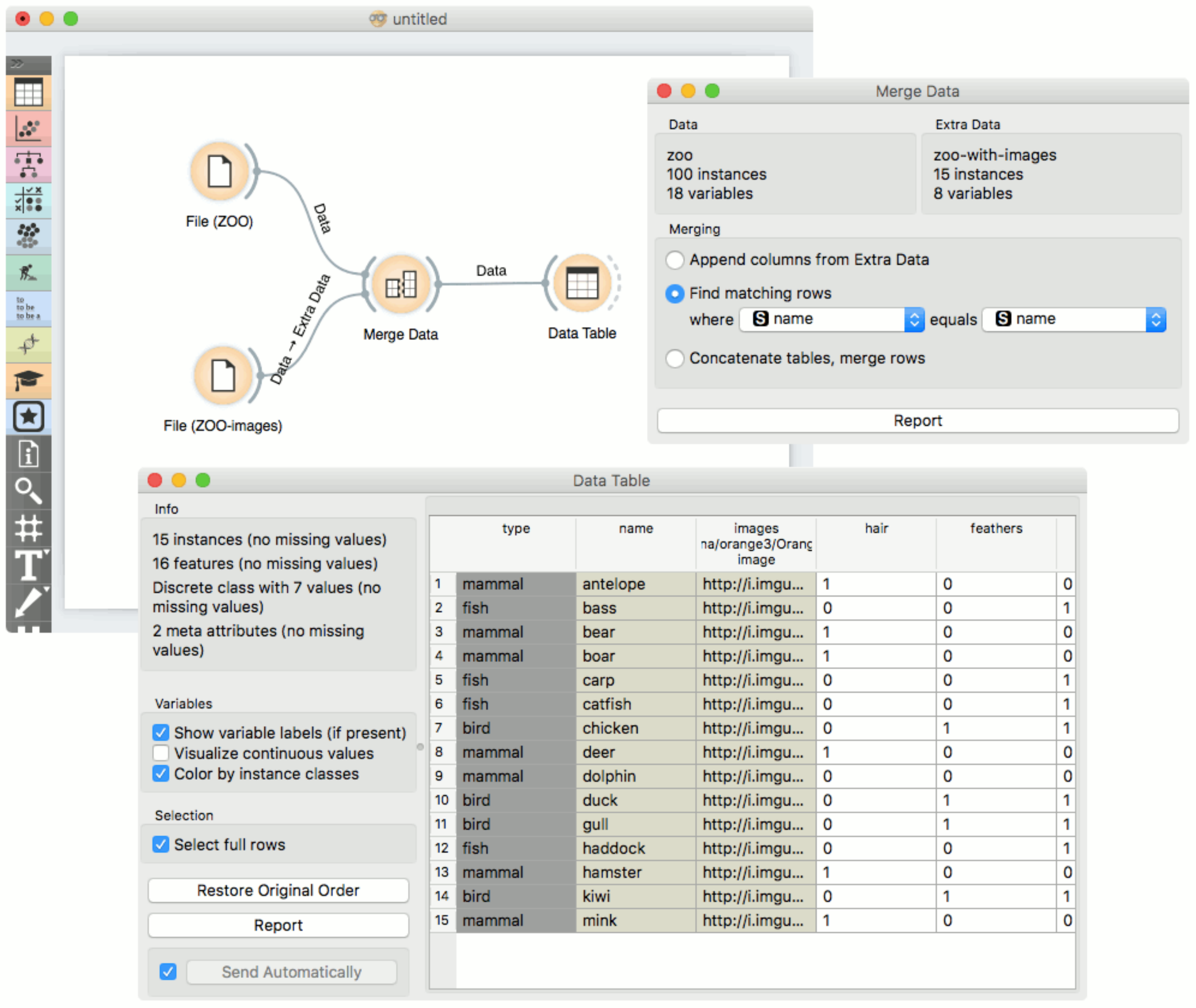
❻ Report

1. Information on Data
2. Information on Extra Data
3. Merging type. **Append columns from Extra Data** outputs all instances from Data appended by matching instances from Extra Data. When no match is found, unknown values are appended. **Find matching rows** outputs similar as above, except hen no match is found, instances are excluded. **Concatenate tables, merge rows** outputs all instances from both inputs, even though the match may not be found. In that case unknown values

are assigned.
4. List of comparable attributes from Data
5. List of comparable attributes from Extra Data
6. Produce a report.

Example

Merging two data sets results in appending new attributes to the original file, based on a selected common attribute. In the example below, we wanted to merge the **zoo.tab** file containing only factual data with **`zoo-with-images.tab`** containing images. Both files share a common string attribute *names*. Now, we create a workflow connecting the two files. The *zoo.tab* data is connected to **Data** input of the **Merge Data** widget, and the *zoo-with-images.tab* data to the **Extra Data** input. Outputs of the **Merge Data** widget is then connected to the Data Table widget. In the latter, the **Merged Data** channels are shown, where image attributes are added to the original data.



Merge Data

Data
zoo
100 instances
18 variables

Extra Data
zoo-with-images
15 instances
8 variables

Merging
Append columns from Extra Data
[✓ Find matching rows]
where name equals name
Concatenate tables, merge rows

Report

Data Table

Info
15 instances (no missing values)
16 features (no missing values)
Discrete class with 7 values (no missing values)
2 meta attributes (no missing values)

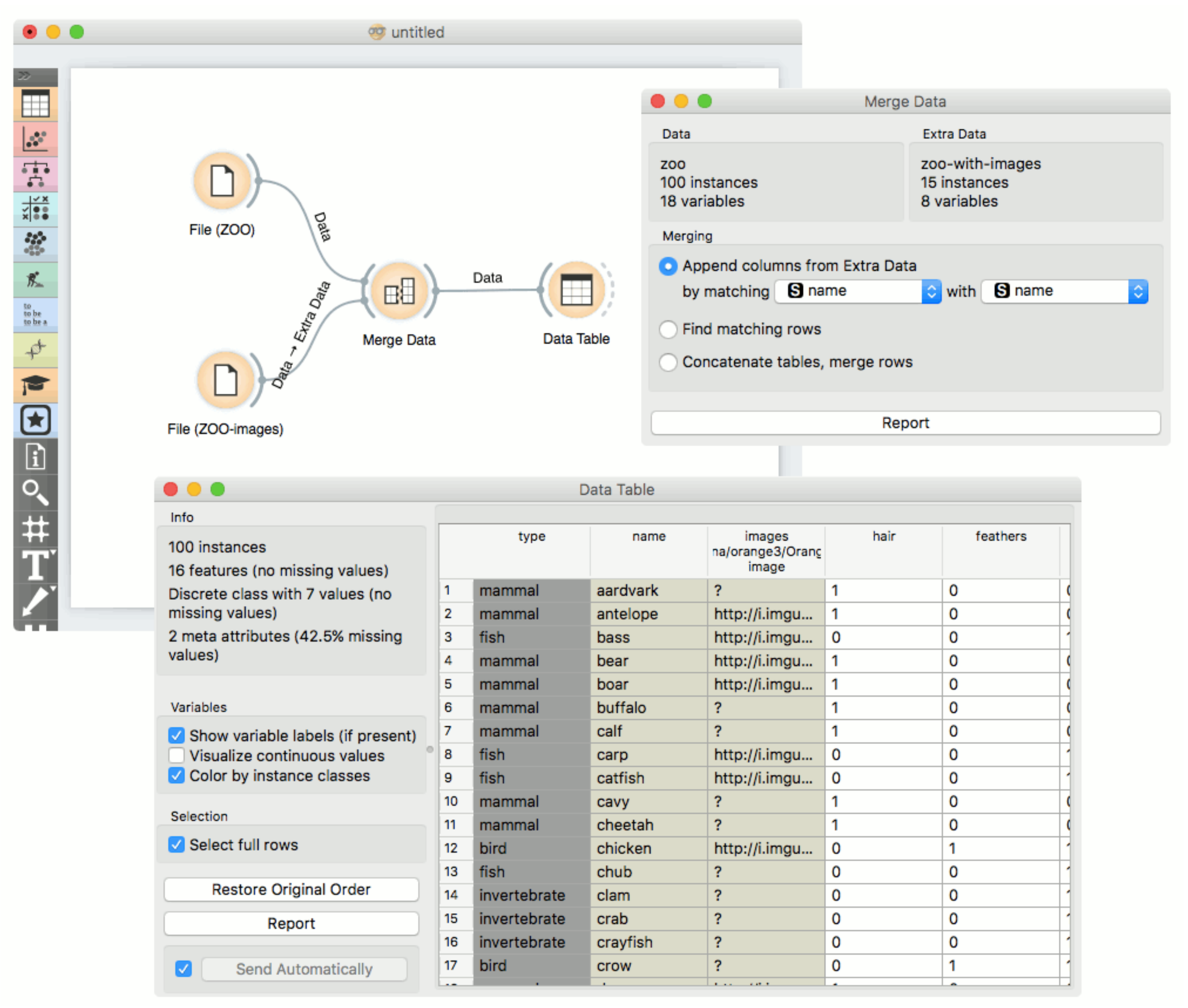
Variables
✓ Show variable labels (if present)
Visualize continuous values
✓ Color by instance classes

Selection
✓ Select full rows

Restore Original Order
Report
✓ Send Automatically

	type	name	images na/orange3/Orang image	hair	feathers	
1	mammal	antelope	http://i.imgu...	1	0	0
2	fish	bass	http://i.imgu...	0	0	1
3	mammal	bear	http://i.imgu...	1	0	0
4	mammal	boar	http://i.imgu...	1	0	0
5	fish	carp	http://i.imgu...	0	0	1
6	fish	catfish	http://i.imgu...	0	0	1
7	bird	chicken	http://i.imgu...	0	1	1
8	mammal	deer	http://i.imgu...	1	0	0
9	mammal	dolphin	http://i.imgu...	0	0	0
10	bird	duck	http://i.imgu...	0	1	1
11	bird	gull	http://i.imgu...	0	1	1
12	fish	haddock	http://i.imgu...	0	0	1
13	mammal	hamster	http://i.imgu...	1	0	0
14	bird	kiwi	http://i.imgu...	0	1	1
15	mammal	mink	http://i.imgu...	1	0	0

The case where we want to include all instances in the output, even those where no match by attribute *names* was found, is shown in the following workflow.



Merge Data

Data
zoo
100 instances
18 variables

Extra Data
zoo-with-images
15 instances
8 variables

Merging
- [✓ Append columns from Extra Data] by matching name with name
- Find matching rows
- Concatenate tables, merge rows

Report

Data Table

Info
100 instances
16 features (no missing values)
Discrete class with 7 values (no missing values)
2 meta attributes (42.5% missing values)

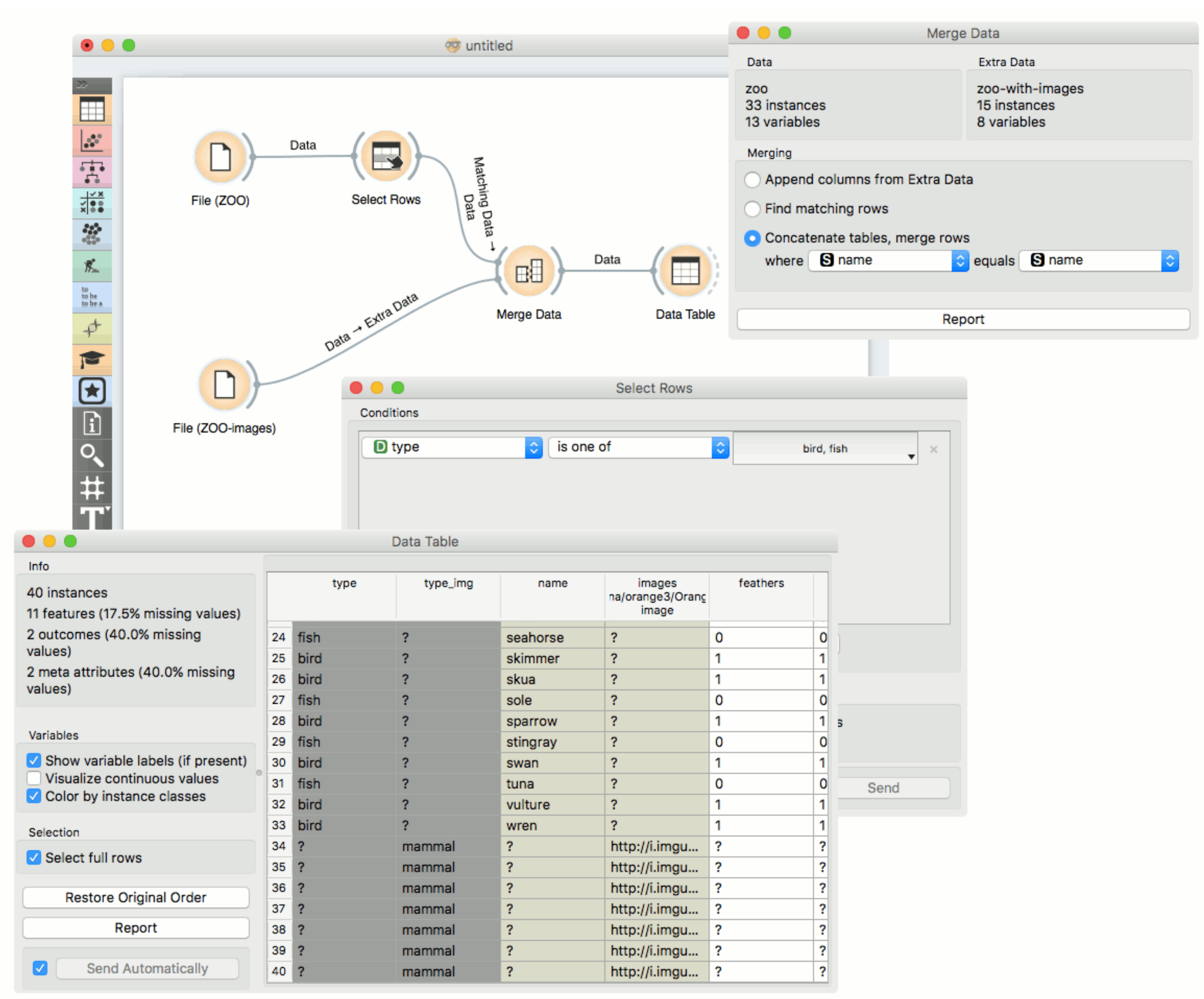
Variables
- [✓] Show variable labels (if present)
- [ ] Visualize continuous values
- [✓] Color by instance classes

Selection
- [✓] Select full rows

Restore Original Order
Report
[✓] Send Automatically

	type	name	images na/orange3/Orang image	hair	feathers
1	mammal	aardvark	?	1	0
2	mammal	antelope	http://i.imgu...	1	0
3	fish	bass	http://i.imgu...	0	0
4	mammal	bear	http://i.imgu...	1	0
5	mammal	boar	http://i.imgu...	1	0
6	mammal	buffalo	?	1	0
7	mammal	calf	?	1	0
8	fish	carp	http://i.imgu...	0	0
9	fish	catfish	http://i.imgu...	0	0
10	mammal	cavy	?	1	0
11	mammal	cheetah	?	1	0
12	bird	chicken	http://i.imgu...	0	1
13	fish	chub	?	0	0
14	invertebrate	clam	?	0	0
15	invertebrate	crab	?	0	0
16	invertebrate	crayfish	?	0	0
17	bird	crow	?	0	1

The third type of merging is shown in the next workflow. The output consist of both inputs, with unknown values assigned where no match was found.



Merge Data

Data
zoo
33 instances
13 variables

Extra Data
zoo-with-images
15 instances
8 variables

Merging
- Append columns from Extra Data
- Find matching rows
- [✓ Concatenate tables, merge rows]
  where name equals name

Report

Select Rows

Conditions
type | is one of | bird, fish

Send

Data Table

Info
40 instances
11 features (17.5% missing values)
2 outcomes (40.0% missing values)
2 meta attributes (40.0% missing values)

Variables
- [✓] Show variable labels (if present)
- [ ] Visualize continuous values
- [✓] Color by instance classes

Selection
- [✓] Select full rows

Restore Original Order
Report
[✓] Send Automatically

	type	type_img	name	images na/orange3/Oranı image	feathers	
24	fish	?	seahorse	?	0	0
25	bird	?	skimmer	?	1	1
26	bird	?	skua	?	1	1
27	fish	?	sole	?	0	0
28	bird	?	sparrow	?	1	1
29	fish	?	stingray	?	0	0
30	bird	?	swan	?	1	1
31	fish	?	tuna	?	0	0
32	bird	?	vulture	?	1	1
33	bird	?	wren	?	1	1
34	?	mammal	?	http://i.imgu...	?	?
35	?	mammal	?	http://i.imgu...	?	?
36	?	mammal	?	http://i.imgu...	?	?
37	?	mammal	?	http://i.imgu...	?	?
38	?	mammal	?	http://i.imgu...	?	?
39	?	mammal	?	http://i.imgu...	?	?
40	?	mammal	?	http://i.imgu...	?	?

Hint

If the two data sets consist of equally-named attributes (other than the ones used to perform the merging), Orange will check by default for consistency of the values of these attributes and report an error in case of non-matching values. In order to avoid the consistency checking, make sure that new attributes are created for each data set: you may use the '*Columns with the same name in different files represent different variables*' option in the File widget for loading the data.

Outliers

Simple outlier detection by comparing distances between instances.

Signals

Inputs:

- **Data**

  A data set

- **Distances**

  A distance matrix

Outputs:

- **Outliers**

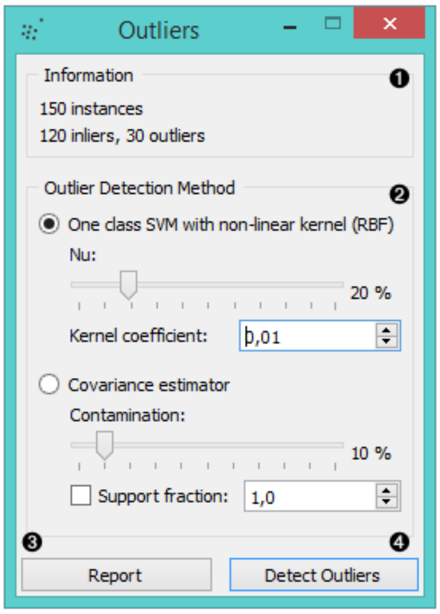
  A data set containing instances scored as outliers

- **Inliers**

  A data set containing instances not scored as outliers

Description

The **Outliers** widget applies one of the two methods for outlier detection. Both methods apply classification to the data set, one with SVM (multiple kernels) and the other with elliptical envelope. *One-class SVM with non-linear kernels (RBF)* performs well with non-Gaussian distributions, while *Covariance estimator* works only for data with Gaussian distribution.


Outliers

Information
150 instances
120 inliers, 30 outliers

Outlier Detection Method
(●) One class SVM with non-linear kernel (RBF)
Nu: 20 %
Kernel coefficient: 0,01
( ) Covariance estimator
Contamination: 10 %
☐ Support fraction: 1,0

1. Information on the input data, number of inliers and outliers based on the selected model.
2. Select the *Outlier detection method*:
   - **One class SVM with non-linear kernel (RBF)**: classifies data as similar or different from the core class

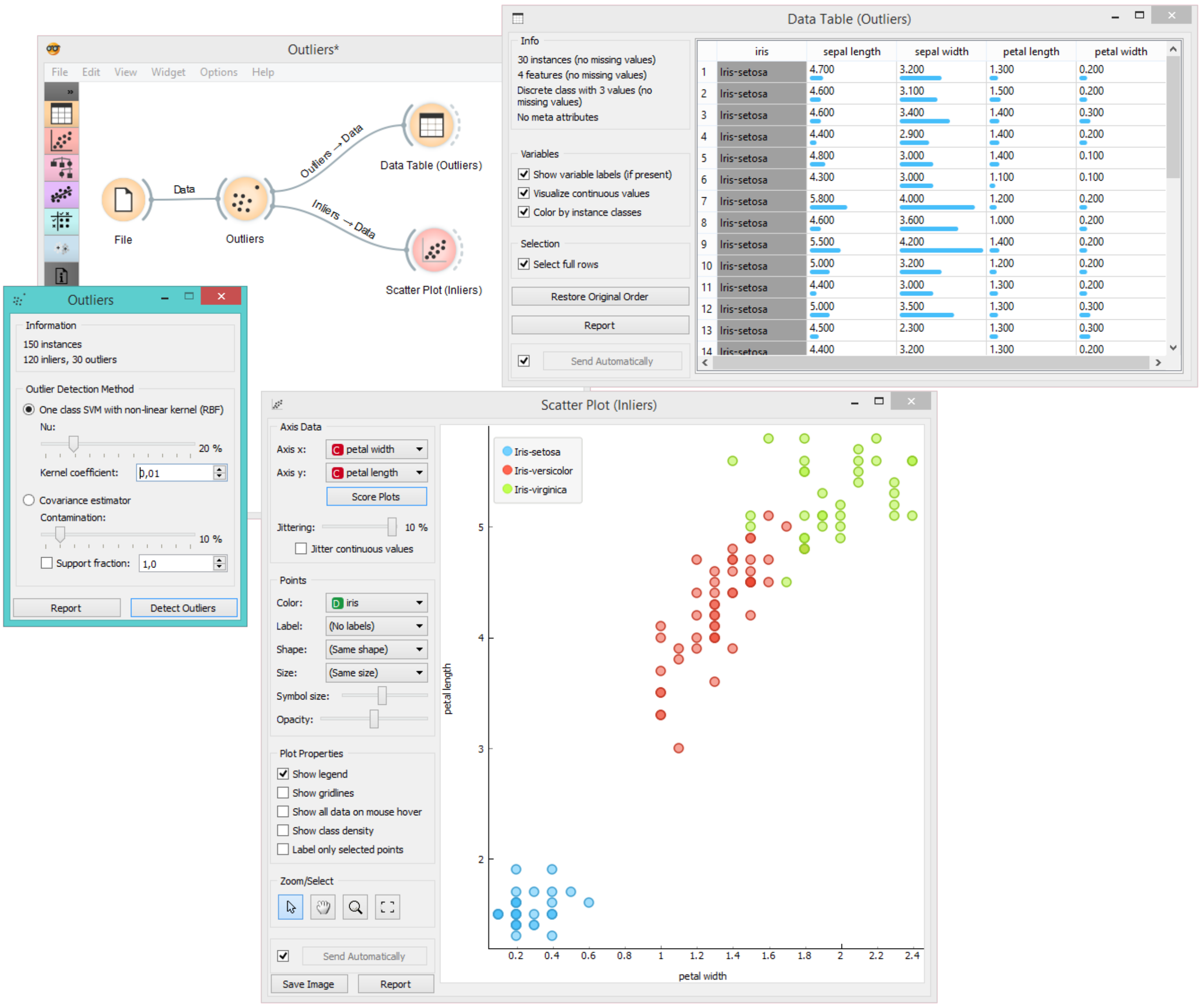
- **Nu** is a parameter for the upper bound on the fraction of training errors and a lower bound of the fraction of support vectors
- **Kernel coefficient** is a gamma parameter, which specifies how much influence a single data instance has

- **Covariance estimator**: fits ellipsis to central points with Mahalanobis distance metric
  - **Contamination** is the proportion of outliers in the data set
  - **Support fraction** specifies the proportion of points included in the estimate

3. Produce a report.
4. Click *Detect outliers* to output the data.

Example

Below, is a simple example of how to use this widget. We used the *Iris* data set to detect the outliers. We chose the *one class SVM with non-linear kernel (RBF)* method, with Nu set at 20% (less training errors, more support vectors). Then we observed the outliers in the Data Table widget, while we sent the inliers to the Scatter Plot.



Outliers

Information
150 instances
120 inliers, 30 outliers

Outlier Detection Method
(●) One class SVM with non-linear kernel (RBF)
Nu: 20 %
Kernel coefficient: 0,01
( ) Covariance estimator
Contamination: 10 %
☐ Support fraction: 1,0

Report | Detect Outliers

Data Table (Outliers)

Info
30 instances (no missing values)
4 features (no missing values)
Discrete class with 3 values (no missing values)
No meta attributes

Variables
☑ Show variable labels (if present)
☑ Visualize continuous values
☑ Color by instance classes

Selection
☑ Select full rows

Restore Original Order
Report
☑ Send Automatically

	iris	sepal length	sepal width	petal length	petal width
1	Iris-setosa	4.700	3.200	1.300	0.200
2	Iris-setosa	4.600	3.100	1.500	0.200
3	Iris-setosa	4.600	3.400	1.400	0.300
4	Iris-setosa	4.400	2.900	1.400	0.200
5	Iris-setosa	4.800	3.000	1.400	0.100
6	Iris-setosa	4.300	3.000	1.100	0.100
7	Iris-setosa	5.800	4.000	1.200	0.200
8	Iris-setosa	4.600	3.600	1.000	0.200
9	Iris-setosa	5.500	4.200	1.400	0.200
10	Iris-setosa	5.000	3.200	1.200	0.200
11	Iris-setosa	4.400	3.000	1.300	0.200
12	Iris-setosa	5.000	3.500	1.300	0.300
13	Iris-setosa	4.500	2.300	1.300	0.300
14	Iris-setosa	4.400	3.200	1.300	0.200

Scatter Plot (Inliers)

Axis Data
Axis x: petal width
Axis y: petal length
Score Plots
Jittering: 10 %
☐ Jitter continuous values

Points
Color: iris
Label: (No labels)
Shape: (Same shape)
Size: (Same size)
Symbol size:
Opacity:

Plot Properties
☑ Show legend
☐ Show gridlines
☐ Show all data on mouse hover
☐ Show class density
☐ Label only selected points

Zoom/Select

☑ Send Automatically
Save Image | Report

Paint Data

Paints data on a 2D plane. You can place individual data points or use a brush to paint larger data sets.

Signals

Inputs

- (None)

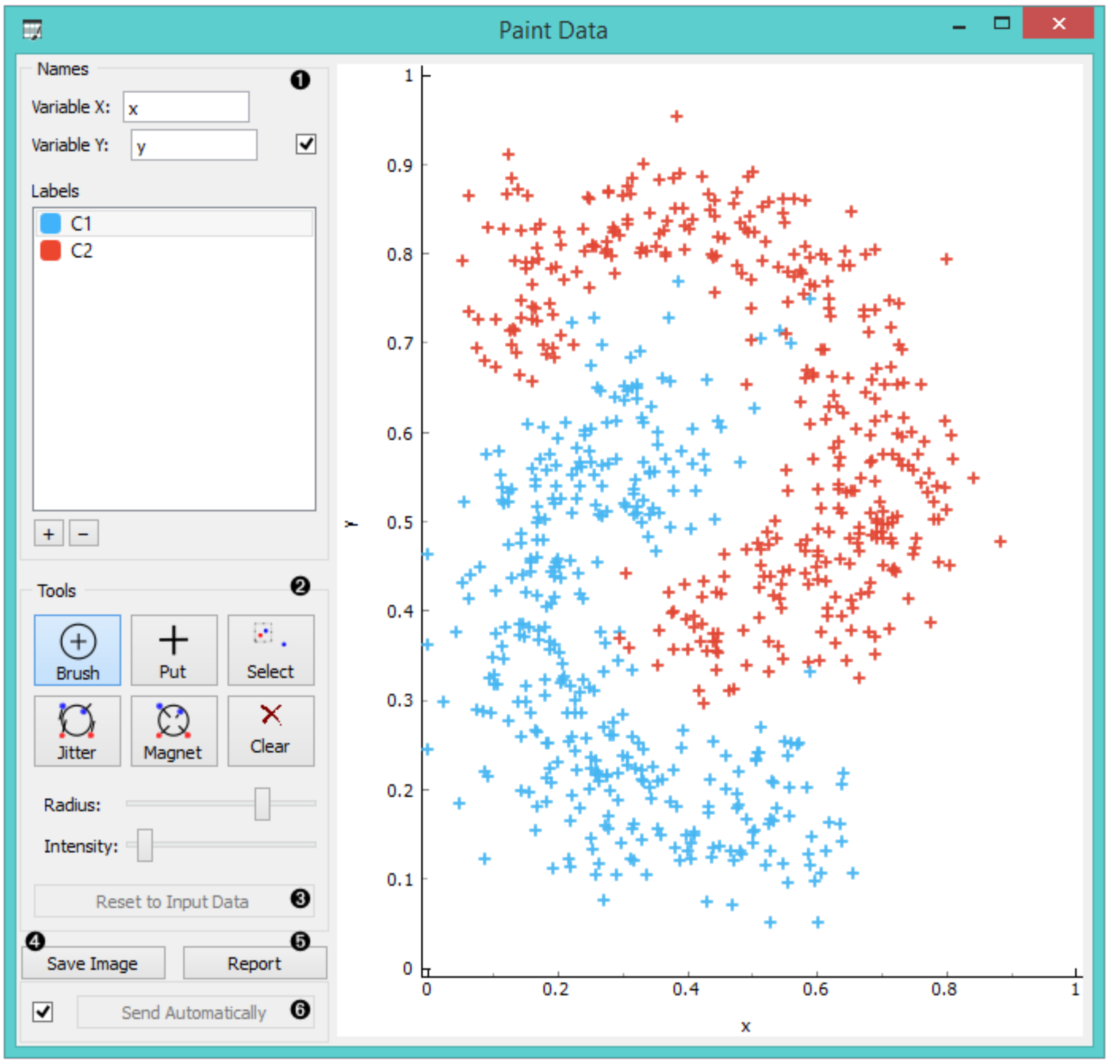
Outputs

- **Data**

  Attribute-valued data set created in the widget

Description

The widget supports the creation of a new data set by visually placing data points on a two-dimension plane. Data points can be placed on the plane individually (*Put*) or in a larger number by brushing (*Brush*). Data points can belong to classes if the data is intended to be used in supervised learning.



Paint Data

Names
Variable X: x
Variable Y: y ✓

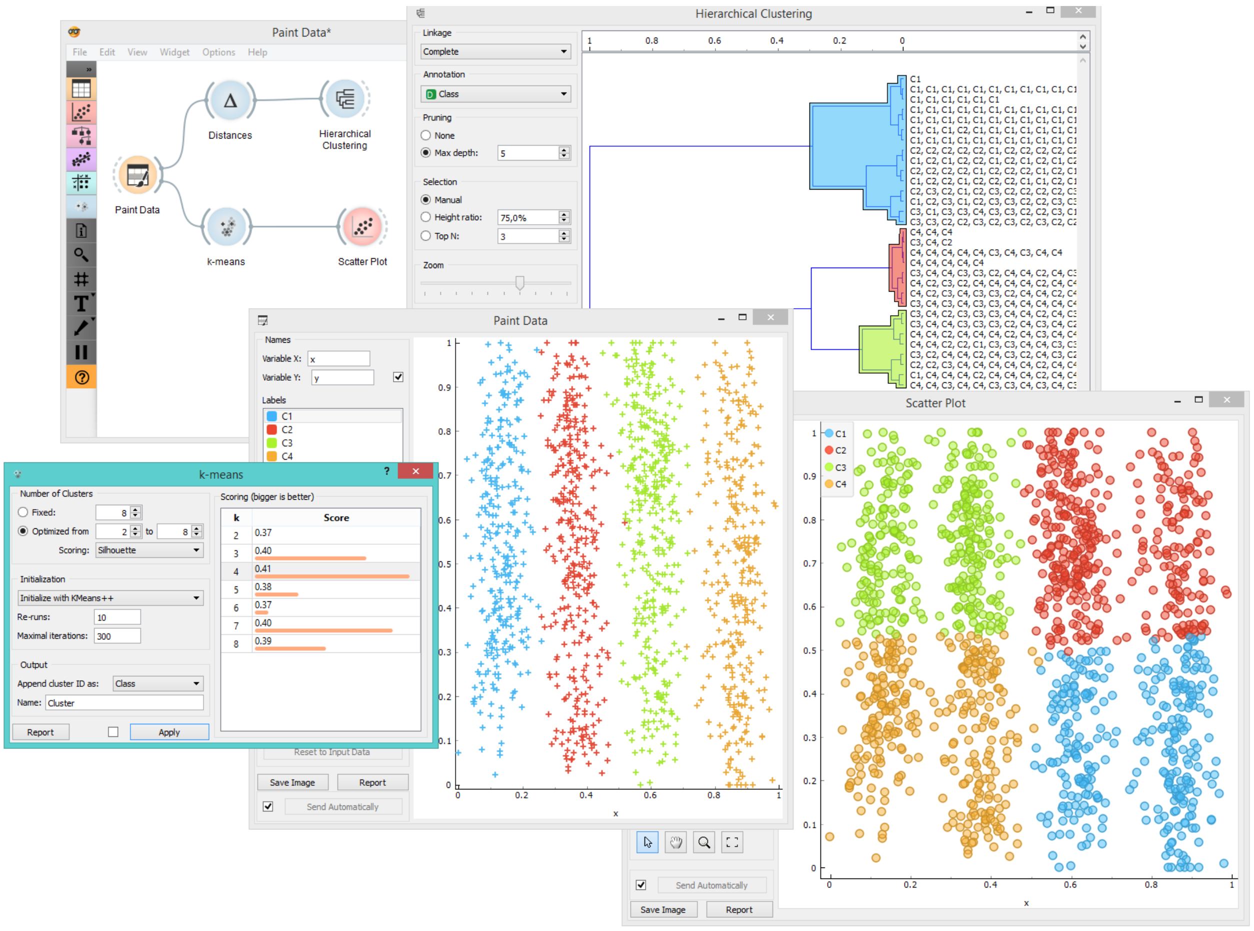
Labels
🟦 C1
🟥 C2

1. Name the axes and select a class to paint data instances. You can add or remove classes. Use only one class to

create classless, unsupervised data sets.
2. Drawing tools. Paint data points with *Brush* (multiple data instances) or *Put* (individual data instance). Select data points with *Select* and remove them with the Delete/Backspace key. Reposition data points with Jitter (spread) and *Magnet* (focus). Use *Zoom* and scroll to zoom in or out. Below, set the radius and intensity for Brush, Put, Jitter and Magnet tools.
3. Reset to Input Data.
4. *Save Image* saves the image to your computer in a .svg or .png format.
5. Produce a report.
6. Tick the box on the left to automatically commit changes to other widgets. Alternatively, press *Send* to apply them.

Example

In the example below, we have painted a data set with 4 classes. Such data set is great for demonstrating k-means and hierarchical clustering methods. In the screenshot, we see that k-means, overall, recognizes clusters better than hierarchical clustering. It returns a score rank, where the best score (the one with the highest value) means the most likely number of clusters. Hierarchical clustering, however, doesn't group the right classes together. This is a great tool for learning and exploring statistical concepts.



Hierarchical Clustering
Linkage: Complete
Annotation: Class
Pruning: ( ) None [✓ Max depth: 5]
Selection: [✓ Manual] ( ) Height ratio: 75,0% ( ) Top N: 3

Paint Data
Names
Variable X: x
Variable Y: y
Labels: 🟦 C1, 🟥 C2, 🟩 C3, 🟧 C4

k-means
Number of Clusters
( ) Fixed: 8
[✓ Optimized from 2 to 8]
Scoring: Silhouette
Initialization: Initialize with KMeans++
Re-runs: 10
Maximal iterations: 300
Output
Append cluster ID as: Class
Name: Cluster
Report ☐ Apply

Scoring (bigger is better)

k	Score
2	0.37
3	0.40
4	0.41
5	0.38
6	0.37
7	0.40
8	0.39

Preprocess

Preprocesses data with selected methods.

Signals

Inputs:

- **Data**

  A data set.

Outputs:

- **Preprocessor**

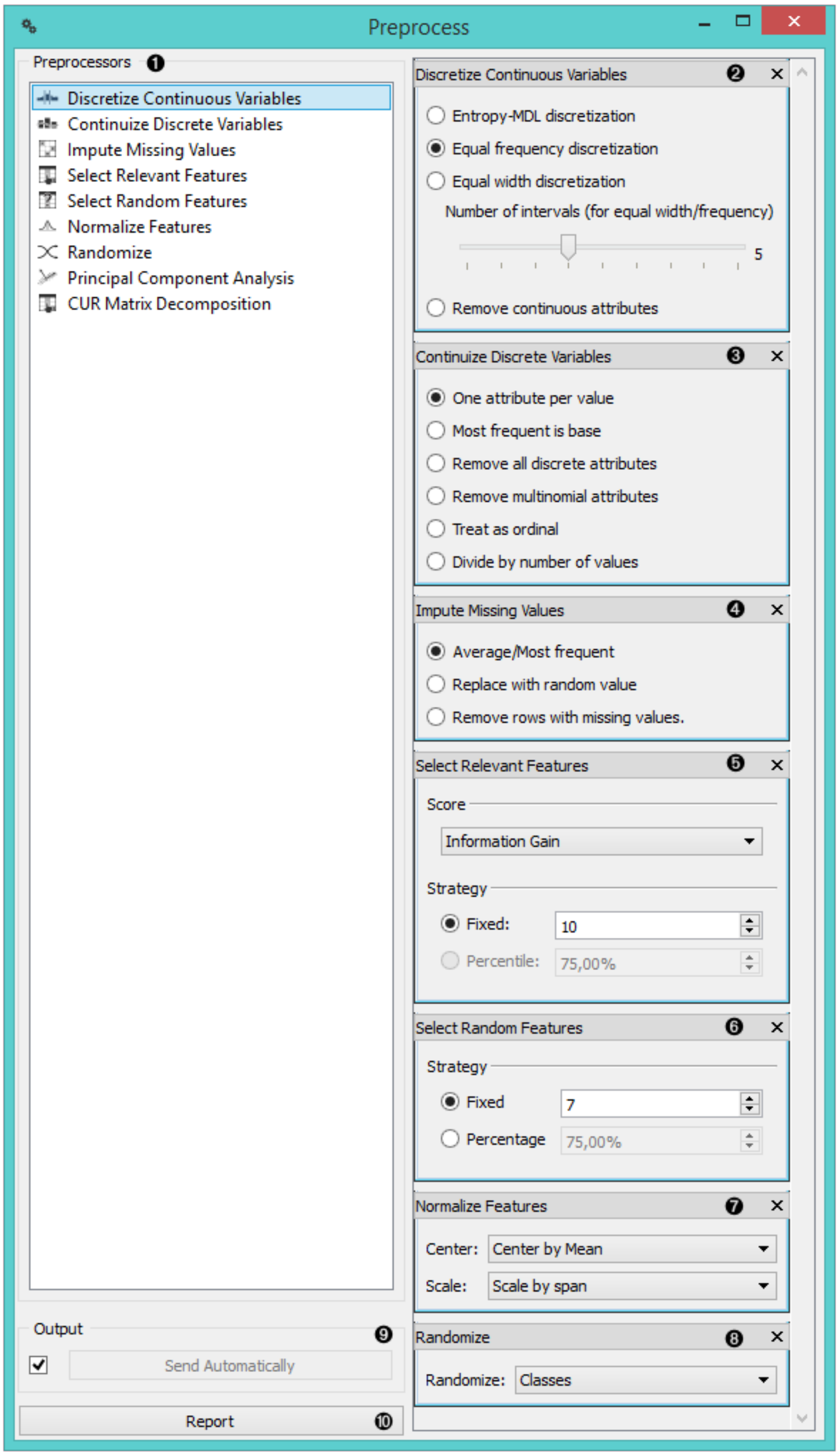
  A preprocessing method.

- **Preprocessed Data**

  Data preprocessed with selected methods.

Description

Preprocessing is crucial for achieving better-quality analysis results. The **Preprocess** widget offers five preprocessing methods to improve data quality. In this widget, you can immediately discretize continuous values or continuize discrete ones, impute missing values, select relevant features or center and scale them. Basically, this widget combines four separate widgets for simpler processing.



Discretize Continuous Variables
- Entropy-MDL discretization
- [✓ Equal frequency discretization]
- Equal width discretization
- Number of intervals (for equal width/frequency): 5
- Remove continuous attributes

Continuize Discrete Variables
- [✓ One attribute per value]
- Most frequent is base
- Remove all discrete attributes
- Remove multinomial attributes
- Treat as ordinal
- Divide by number of values

Impute Missing Values
- [✓ Average/Most frequent]
- Replace with random value
- Remove rows with missing values.

Select Relevant Features
- Score: Information Gain
- Strategy: [✓ Fixed: 10] Percentile: 75,00%

Select Random Features
- Strategy: [✓ Fixed 7] Percentage 75,00%

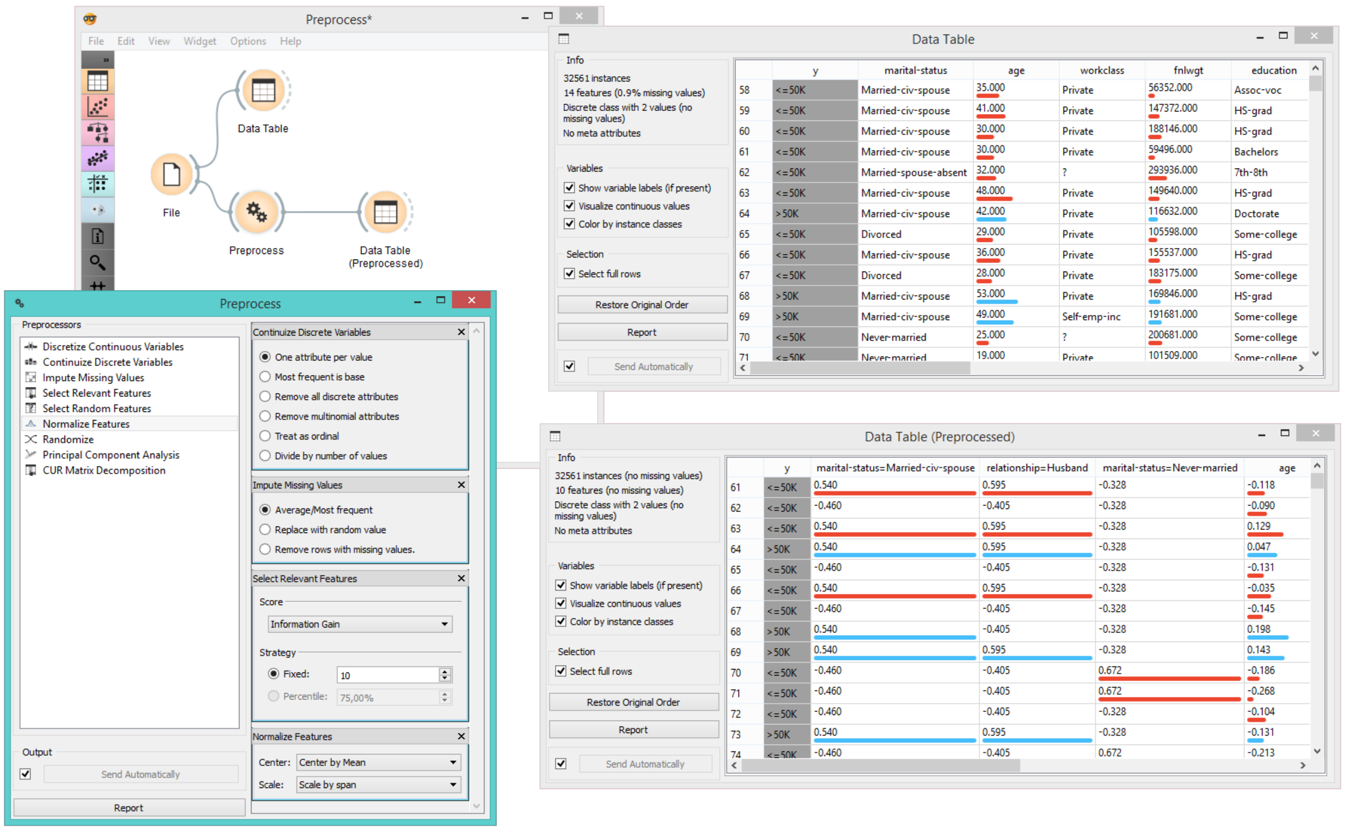
Normalize Features
- Center: Center by Mean
- Scale: Scale by span

Randomize
- Randomize: Classes

1. List of preprocessors. You drag the preprocessors you wish to use to the right side of the widget.
2. Discretization of continuous values
3. Continuization of discrete values
4. Impute missing values or remove them.
5. Select the most relevant features by information gain, gain ratio, Gini index.
6. Select random features
7. Normalize features
8. Randomize
9. When the box is ticked (*Send Automatically*), the widget will communicate changes automatically. Alternatively, click *Send*.
10. Produce a report.

Example

In the example below, we have used the *adult* data set and preprocessed the data. We continuized discrete values (age, education and marital status...) as *one attribute per value*, we imputed missing values (replacing ? with average values), selected 10 most relevant attributes by *Information gain*, centered them by mean and scaled by span. We can observe the changes in the Data Table and compare it to the non-processed data.



Preprocess

Preprocessors: Discretize Continuous Variables, Continuize Discrete Variables, Impute Missing Values, Select Relevant Features, Select Random Features, Normalize Features, Randomize, Principal Component Analysis, CUR Matrix Decomposition

Continuize Discrete Variables: [✓ One attribute per value] / Most frequent is base / Remove all discrete attributes / Remove multinomial attributes / Treat as ordinal / Divide by number of values

Impute Missing Values: [✓ Average/Most frequent] / Replace with random value / Remove rows with missing values.

Select Relevant Features: Score: Information Gain; Strategy: [✓ Fixed: 10] / Percentile: 75,00%

Normalize Features: Center: Center by Mean; Scale: Scale by span

Output: ✓ Send Automatically

Report

Data Table

Info: 32561 instances; 14 features (0.9% missing values); Discrete class with 2 values (no missing values); No meta attributes

Variables: ✓ Show variable labels (if present); ✓ Visualize continuous values; ✓ Color by instance classes

Selection: ✓ Select full rows

Restore Original Order | Report | ✓ Send Automatically

	y	marital-status	age	workclass	fnlwgt	education
58	<=50K	Married-civ-spouse	35.000	Private	56352.000	Assoc-voc
59	<=50K	Married-civ-spouse	41.000	Private	147372.000	HS-grad
60	<=50K	Married-civ-spouse	30.000	Private	188146.000	HS-grad
61	<=50K	Married-civ-spouse	30.000	Private	59496.000	Bachelors
62	<=50K	Married-spouse-absent	32.000	?	293936.000	7th-8th
63	<=50K	Married-civ-spouse	48.000	Private	149640.000	HS-grad
64	>50K	Married-civ-spouse	42.000	Private	116632.000	Doctorate
65	<=50K	Divorced	29.000	Private	105598.000	Some-college
66	<=50K	Married-civ-spouse	36.000	Private	155537.000	HS-grad
67	<=50K	Divorced	28.000	Private	183175.000	Some-college
68	>50K	Married-civ-spouse	53.000	Private	169846.000	HS-grad
69	>50K	Married-civ-spouse	49.000	Self-emp-inc	191681.000	Some-college
70	<=50K	Never-married	25.000	?	200681.000	Some-college
71	<=50K	Never-married	19.000	Private	101509.000	Some-college

Data Table (Preprocessed)

Info: 32561 instances (no missing values); 10 features (no missing values); Discrete class with 2 values (no missing values); No meta attributes

Variables: ✓ Show variable labels (if present); ✓ Visualize continuous values; ✓ Color by instance classes

Selection: ✓ Select full rows

Restore Original Order | Report | ✓ Send Automatically

	y	marital-status=Married-civ-spouse	relationship=Husband	marital-status=Never-married	age
61	<=50K	0.540	0.595	-0.328	-0.118
62	<=50K	-0.460	-0.405	-0.328	-0.090
63	<=50K	0.540	0.595	-0.328	0.129
64	>50K	0.540	0.595	-0.328	0.047
65	<=50K	-0.460	-0.405	-0.328	-0.131
66	<=50K	0.540	0.595	-0.328	-0.035
67	<=50K	-0.460	-0.405	-0.328	-0.145
68	>50K	0.540	-0.405	-0.328	0.198
69	>50K	0.540	0.595	-0.328	0.143
70	<=50K	-0.460	-0.405	0.672	-0.186
71	<=50K	-0.460	-0.405	0.672	-0.268
72	<=50K	-0.460	-0.405	-0.328	-0.104
73	>50K	0.540	0.595	-0.328	-0.131
74	<=50K	-0.460	-0.405	0.672	-0.213

Purge Domain

Removes unused attribute values and useless attributes, sorts the remaining values.

Signals

Inputs:

- **Data**

  A data set.

Outputs:

- **Data**

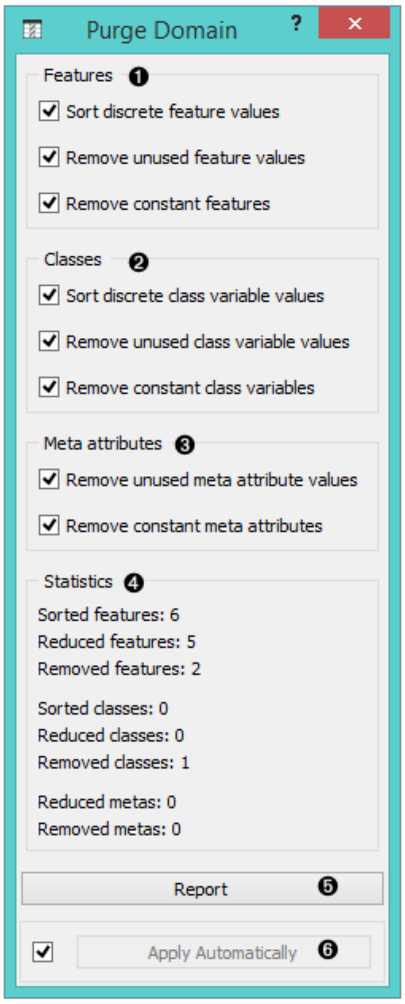
  A filtered data set

Description

Definitions of nominal attributes sometimes contain values which don’t appear in the data. Even if this does not happen in the original data, filtering the data, selecting examplary subsets and alike can remove all examples for which the attribute has some particular value. Such values clutter data presentation, especially various visualizations, and should be removed.

After purging an attribute, it may become single-valued or, in extreme case, have no values at all (if the value of this attribute was undefined for all examples). In such cases, the attribute can be removed.

A different issue is the order of attribute values: if the data is read from a file in a format in which values are not declared in advance, they are sorted “in order of appearance”. Sometimes we would prefer to have them sorted alphabetically.


Purge Domain
Features 1
Sort discrete feature values
Remove unused feature values
Remove constant features
Classes 2
Sort discrete class variable values
Remove unused class variable values
Remove constant class variables
Meta attributes 3
Remove unused meta attribute values
Remove constant meta attributes
Statistics 4
Report 5
Apply Automatically 6

Sorted features: 6
Reduced features: 5
Removed features: 2

Sorted classes: 0
Reduced classes: 0
Removed classes: 1

Reduced metas: 0
Removed metas: 0

1. Purge attributes.
2. Purge classes.
3. Purge meta attributes.
4. Information on the filtering process
5. Produce a report.
6. If *Apply automatically* is ticked, the widget will output data at each change of widget settings.

Such purification is done by the widget **Purge Domain**. Ordinary attributes and class attributes are treated separately. For each, we can decide if we want the values sorted or not. Next, we may allow the widget to remove attributes with less than two values or remove the class attribute if there are less than two classes. Finally, we can instruct the widget to check which values of attributes actually appear in the data and remove the unused values. The widget cannot remove values if it is not allowed to remove the attributes, since having attributes without values makes no sense.

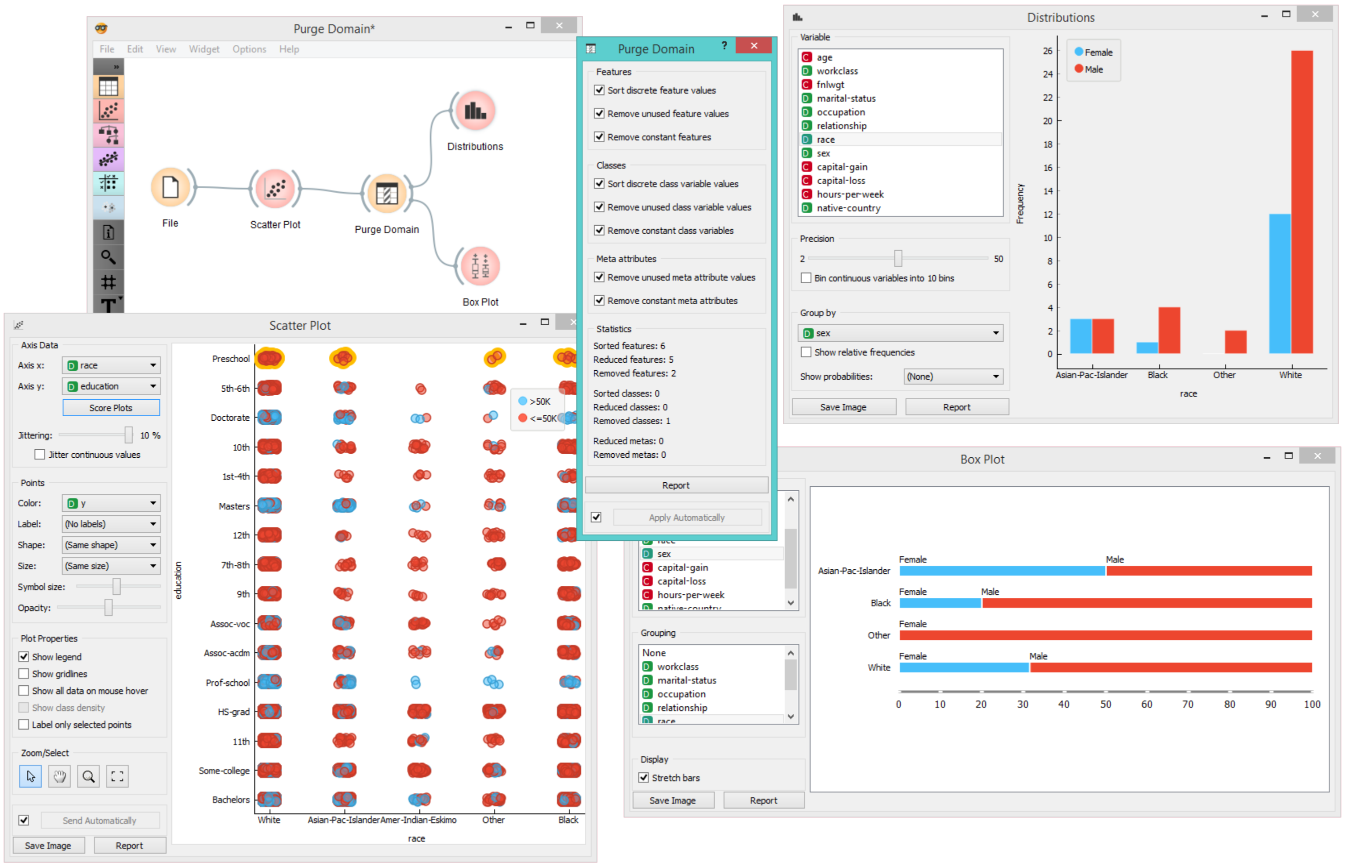
The new, reduced attributes get the prefix "R", which distinguishes them from the original ones. The values of new attributes can be computed from the old ones, but not the other way around. This means that if you construct a classifier from the new attributes, you can use it to classify the examples described by the original attributes. But not the opposite: constructing a classifier from the old attributes and using it on examples described by the reduced ones won't work. Fortunately, the latter is seldom the case. In a typical setup, one would explore the data, visualize it, filter it, purify it… and then test the final model on the original data.

Example

The **Purge Domain** widget would typically appear after data filtering, for instance when selecting a subset of visualized examples.

In the above schema, we play with the *adult.tab* data set: we visualize it and select a portion of the data, which contains only four out of the five original classes. To get rid of the empty class, we put the data through **Purge Domain** before going on to the Box Plot widget. The latter shows only the four classes which are in the **Purge Data** output.

To see the effect of data purification, uncheck *Remove unused class variable values* and observe the effect this has on Box Plot.



Purge Domain

Features
- [✓] Sort discrete feature values
- [✓] Remove unused feature values
- [✓] Remove constant features

Classes
- [✓] Sort discrete class variable values
- [✓] Remove unused class variable values
- [✓] Remove constant class variables

Meta attributes
- [✓] Remove unused meta attribute values
- [✓] Remove constant meta attributes

Statistics
Sorted features: 6
Reduced features: 5
Removed features: 2

Sorted classes: 0
Reduced classes: 0
Removed classes: 1

Reduced metas: 0
Removed metas: 0

[✓] Apply Automatically

Scatter Plot

Axis Data
Axis x: race
Axis y: education
Jittering: 10 %
[ ] Jitter continuous values

Points
Color: y
Label: (No labels)
Shape: (Same shape)
Size: (Same size)

Plot Properties
- [✓] Show legend
- [ ] Show gridlines
- [ ] Show all data on mouse hover
- [ ] Show class density
- [ ] Label only selected points

Distributions

Variable: race
Precision: 2 … 50
[ ] Bin continuous variables into 10 bins
Group by: sex
[ ] Show relative frequencies
Show probabilities: (None)

Box Plot

Grouping: race
Display: [✓] Stretch bars

Python Script

Extends functionalities through Python scripting.

Signals

Inputs:

- **in_data (Orange.data.Table)**

  Input data set bound to `in_data` variable in the script's local namespace.

- **in_distance (Orange.core.SymMatrix)**

  Input symmetric matrix bound to `in_distance` variable in the script's local namespace.

- **in_learner (Orange.classification.Learner)**

  Input learner bound to `in_learner` variable in the script's local namespace.

- **in_classifier (Orange.classification.Learner)**

  Input classifier bound to `in_classifier` variable in the script's local namespace.

- **in_object (object)**

  Input python object bound to `in_object` variable in the script's local namespace.

Outputs:

- **out_data (Orange.data.Table)**

  Data set retrieved from `out_data` variable in the script's local namespace after execution.

- **out_distance (Orange.core.SymMatrix)**

  Symmetric matrix retrieved from `out_distance` variable in the script's local namespace after execution.

- **out_learner (Orange.classification.Learner)**

  Learner retrieved from `out_learner` variable in the script's local namespace.

- **out_classifier (Orange.classification.Learner)**

  Classifier retrieved from `out_classifier` variable in the script's local namespace after execution.

- **out_object (object)**

  Python object retrieved from `out_object` variable in the script's local namespace after execution.

Description

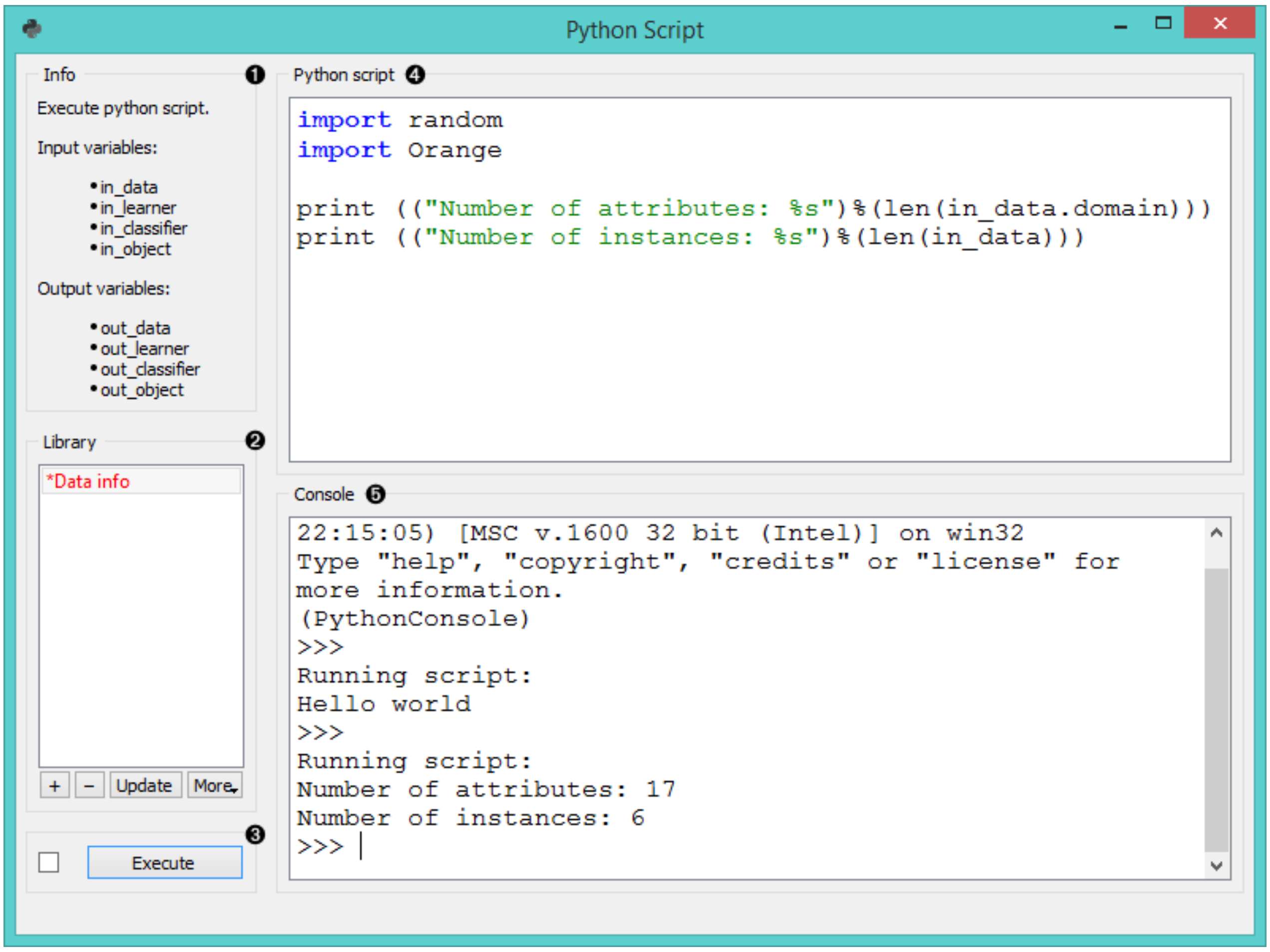
Python Script widget can be used to run a python script in the input, when a suitable functionality is not implemented in an existing widget. The script has `in_data`, `in_distance`, `in_learner`, `in_classifier` and `in_object` variables (from input signals) in its local namespace. If a signal is not connected or it did not yet receive any data, those variables contain `None`.

After the script is executed, `out_data`, `out_distance`, ... variables from the script's local namespace are extracted and used as outputs of the widget. The widget can be further connected to other widgets for visualizing the output.

For instance the following script would simply pass on all signals it receives:

```
out_data = in_data
out_distance = in_distance
out_learner = in_learner
out_classifier = in_classifier
out_object = in_object
```

> Note:
>
> You should not modify the input objects in place.



```
import random
import Orange

print (("Number of attributes: %s")%(len(in_data.domain)))
print (("Number of instances: %s")%(len(in_data)))
```

```
22:15:05) [MSC v.1600 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for
more information.
(PythonConsole)
>>>
Running script:
Hello world
>>>
Running script:
Number of attributes: 17
Number of instances: 6
>>> |
```

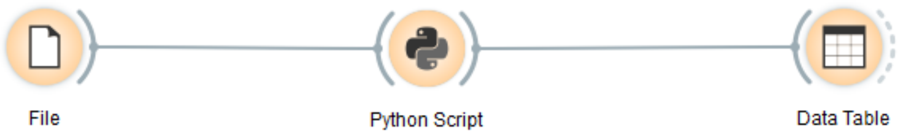
1. Info box contains names of basic operators for Orange Python script.
2. The *Library* control can be used to manage multiple scripts. Pressing "+" will add a new entry and open it in the *Python script* editor. When the script is modified, its entry in the *Library* will change to indicate it has unsaved changes. Pressing *Update* will save the script (keyboard shortcut ctrl + s). A script can be removed by selecting it and pressing the "-" button.
3. Pressing *Execute* in the *Run* box executes the script (using `exec`). Any script output (from `print`) is captured and displayed in the *Console* below the script. If *Auto execute* is checked, the script is run any time inputs to the widget change.
4. The *Python script* editor on the left can be used to edit a script (it supports some rudimentary syntax highlighting).
5. Console displays the output of the script.

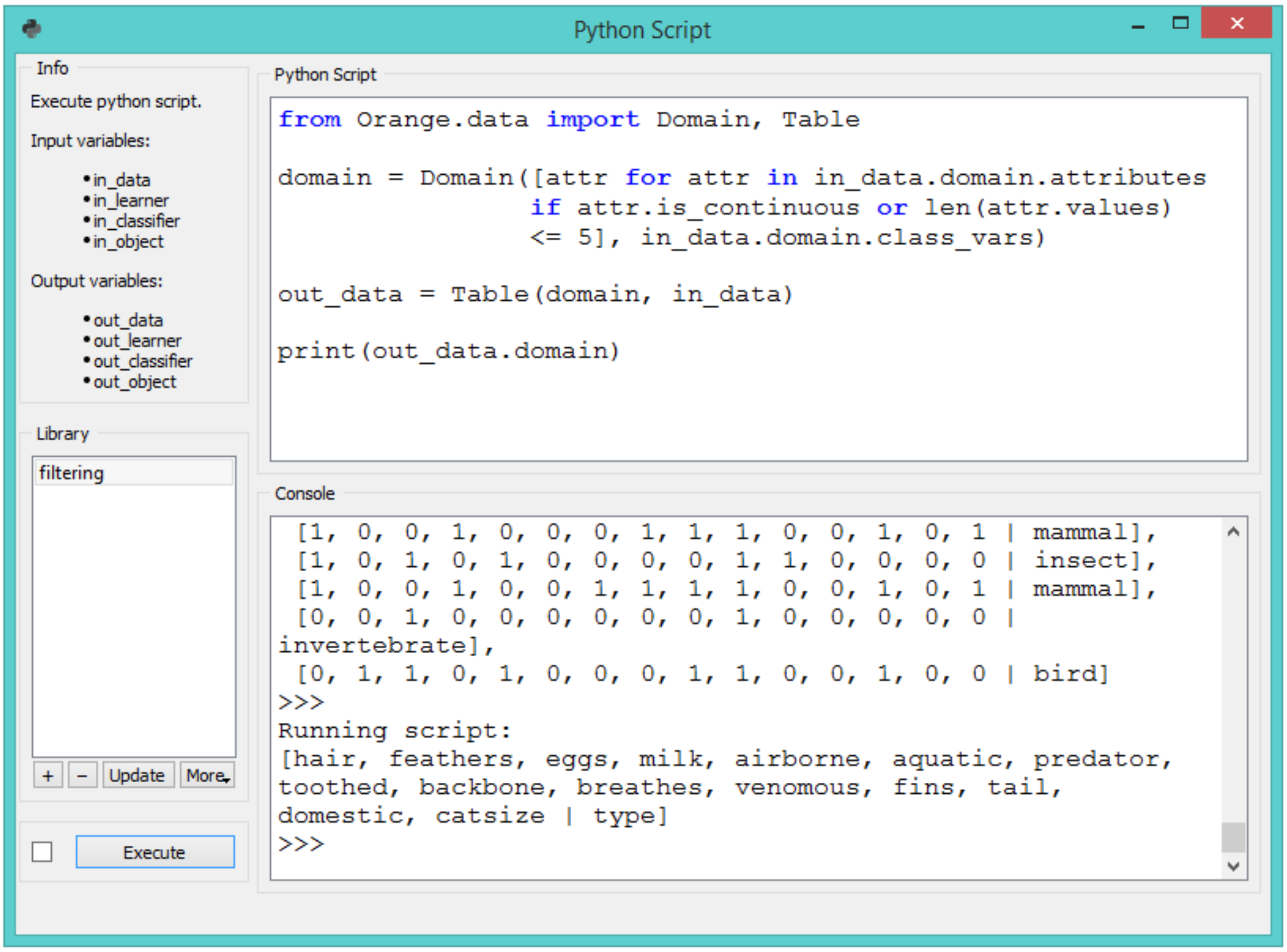
Examples

Python Script widget is intended to extend functionalities for advanced users.

One can, for example, do batch filtering by attributes. We used zoo.tab for the example and we filtered out all the attributes that have more than 5 discrete values. This in our case removed only 'leg' attribute, but imagine an example where one would have many such attributes.

```
from Orange.data import Domain, Table
domain = Domain([attr for attr in in_data.domain.attributes
                 if attr.is_continuous or len(attr.values) <= 5],
                in_data.domain.class_vars)
out_data = Table(domain, in_data)
```



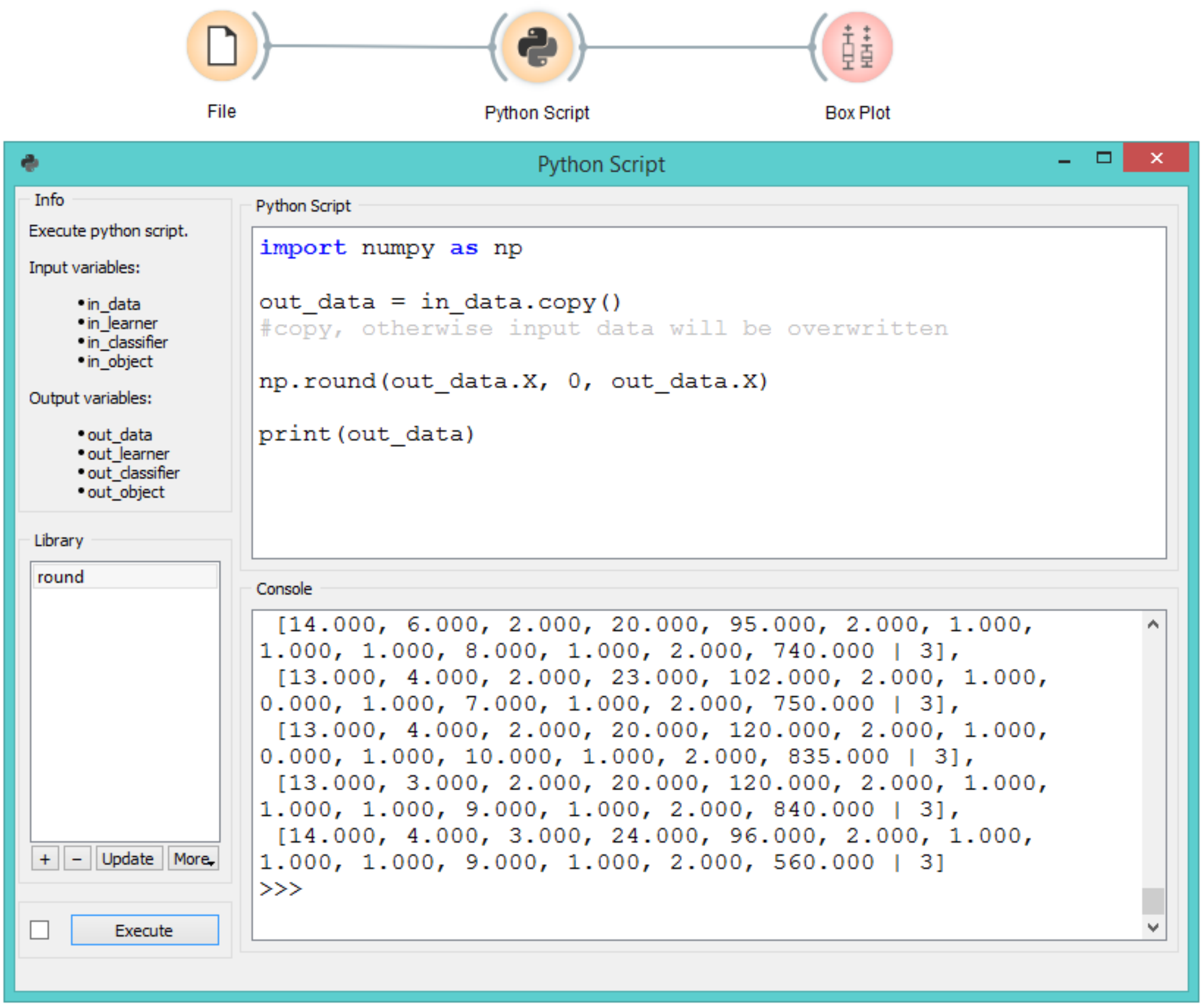


```
from Orange.data import Domain, Table

domain = Domain([attr for attr in in_data.domain.attributes
                 if attr.is_continuous or len(attr.values)
                 <= 5], in_data.domain.class_vars)

out_data = Table(domain, in_data)

print(out_data.domain)
```

```
 [1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1 | mammal],
 [1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0 | insect],
 [1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1 | mammal],
 [0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0 |
invertebrate],
 [0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0 | bird]
>>>
Running script:
[hair, feathers, eggs, milk, airborne, aquatic, predator,
toothed, backbone, breathes, venomous, fins, tail,
domestic, catsize | type]
>>>
```

The second example shows how to round all the values in a few lines of code. This time we used wine.tab and rounded all the values to whole numbers.

```
import numpy as np
out_data = in_data.copy()
#copy, otherwise input data will be overwritten
np.round(out_data.X, 0, out_data.X)
```

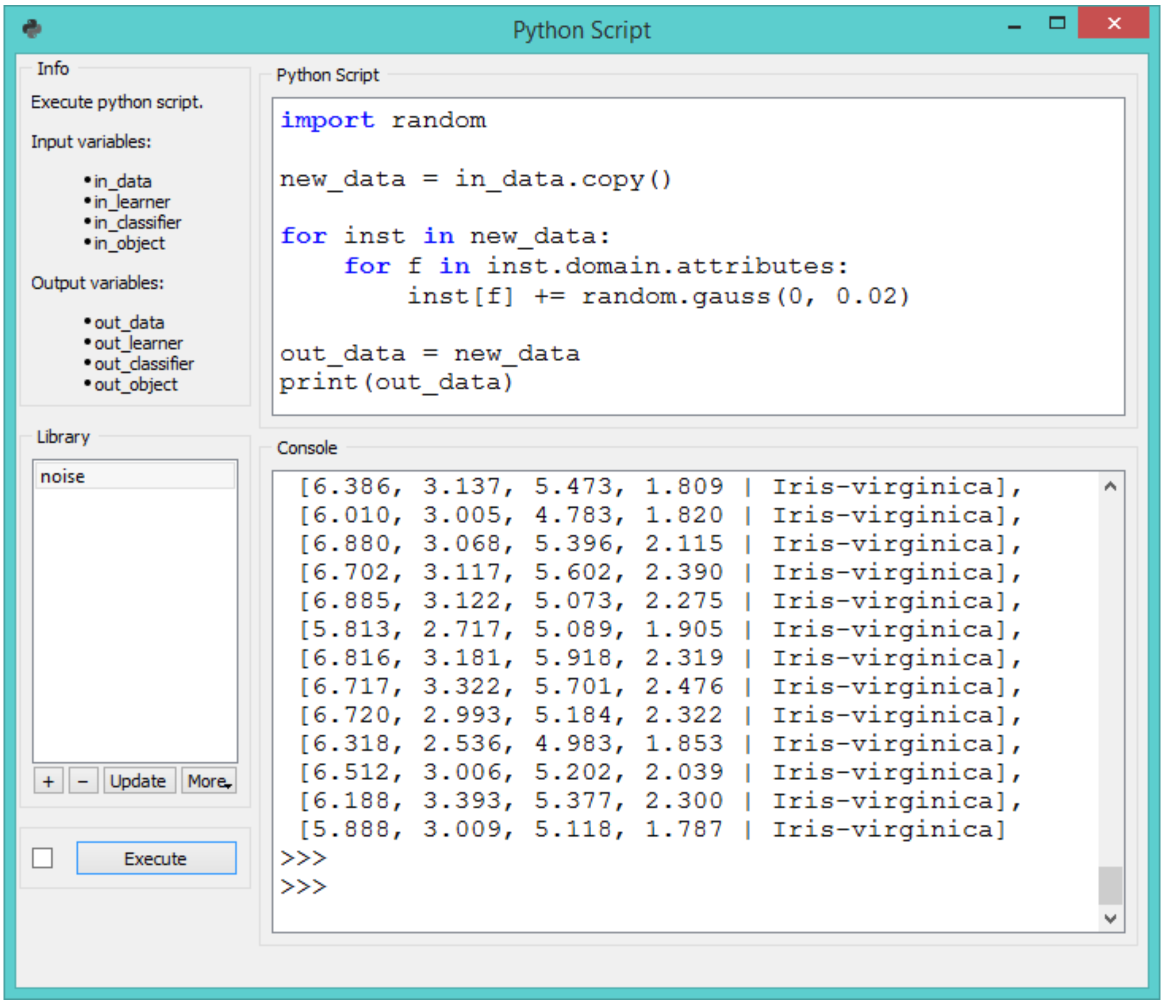


```
import numpy as np

out_data = in_data.copy()
#copy, otherwise input data will be overwritten

np.round(out_data.X, 0, out_data.X)

print(out_data)
```

```
 [14.000, 6.000, 2.000, 20.000, 95.000, 2.000, 1.000,
1.000, 1.000, 8.000, 1.000, 2.000, 740.000 | 3],
 [13.000, 4.000, 2.000, 23.000, 102.000, 2.000, 1.000,
0.000, 1.000, 7.000, 1.000, 2.000, 750.000 | 3],
 [13.000, 4.000, 2.000, 20.000, 120.000, 2.000, 1.000,
0.000, 1.000, 10.000, 1.000, 2.000, 835.000 | 3],
 [13.000, 3.000, 2.000, 20.000, 120.000, 2.000, 1.000,
1.000, 1.000, 9.000, 1.000, 2.000, 840.000 | 3],
 [14.000, 4.000, 3.000, 24.000, 96.000, 2.000, 1.000,
1.000, 1.000, 9.000, 1.000, 2.000, 560.000 | 3]
>>>
```

The third example introduces some gaussian noise to the data. Again we make a copy of the input data, then walk th-rough all the values with a double for loop and add random noise.

```
import random
from Orange.data import Domain, Table
new_data = in_data.copy()
for inst in new_data:
  for f in inst.domain.attributes:
    inst[f] += random.gauss(0, 0.02)
out_data = new_data
```



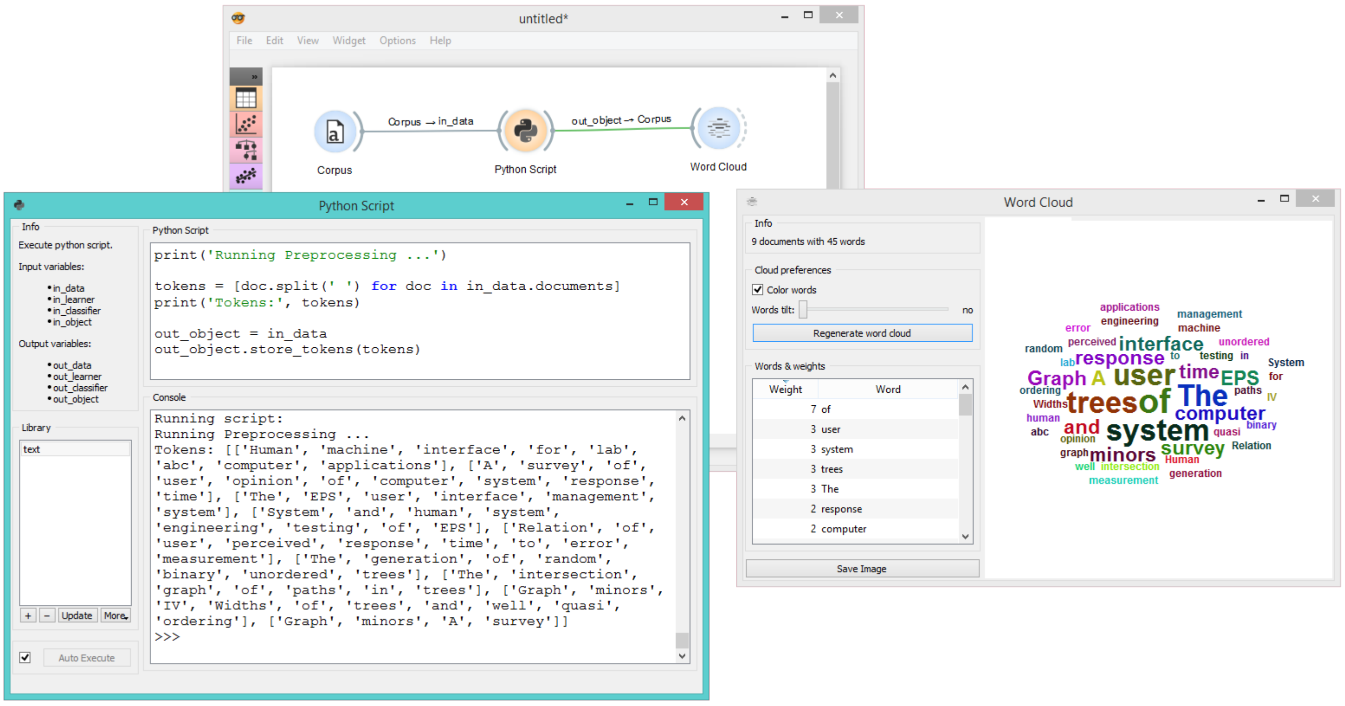
```
import random

new_data = in_data.copy()

for inst in new_data:
    for f in inst.domain.attributes:
        inst[f] += random.gauss(0, 0.02)

out_data = new_data
print(out_data)
```

```
 [6.386, 3.137, 5.473, 1.809 | Iris-virginica],
 [6.010, 3.005, 4.783, 1.820 | Iris-virginica],
 [6.880, 3.068, 5.396, 2.115 | Iris-virginica],
 [6.702, 3.117, 5.602, 2.390 | Iris-virginica],
 [6.885, 3.122, 5.073, 2.275 | Iris-virginica],
 [5.813, 2.717, 5.089, 1.905 | Iris-virginica],
 [6.816, 3.181, 5.918, 2.319 | Iris-virginica],
 [6.717, 3.322, 5.701, 2.476 | Iris-virginica],
 [6.720, 2.993, 5.184, 2.322 | Iris-virginica],
 [6.318, 2.536, 4.983, 1.853 | Iris-virginica],
 [6.512, 3.006, 5.202, 2.039 | Iris-virginica],
 [6.188, 3.393, 5.377, 2.300 | Iris-virginica],
 [5.888, 3.009, 5.118, 1.787 | Iris-virginica]
>>>
>>>
```

The final example uses Orange3-Text add-on. **Python Script** is very useful for custom preprocessing in text mining, extracting new features from strings, or utilizing advanced nltk or gensim functions. Below, we simply tokenized our input data from deerwester.tab by splitting them by whitespace.

```
print('Running Preprocessing ...')
tokens = [doc.split(' ') for doc in in_data.documents]
print('Tokens:', tokens)
out_object = in_data
out_object.store_tokens(tokens)
```

You can add a lot of other preprocessing steps to further adjust the output. The output of **Python Script** can be used with any widget that accepts the type of output your script produces. In this case, connection is green, which signalizes the right type of input for Word Cloud widget.



```
print('Running Preprocessing ...')

tokens = [doc.split(' ') for doc in in_data.documents]
print('Tokens:', tokens)

out_object = in_data
out_object.store_tokens(tokens)
```

```
Running script:
Running Preprocessing ...
Tokens: [['Human', 'machine', 'interface', 'for', 'lab',
'abc', 'computer', 'applications'], ['A', 'survey', 'of',
'user', 'opinion', 'of', 'computer', 'system', 'response',
'time'], ['The', 'EPS', 'user', 'interface', 'management',
'system'], ['System', 'and', 'human', 'system',
'engineering', 'testing', 'of', 'EPS'], ['Relation', 'of',
'user', 'perceived', 'response', 'time', 'to', 'error',
'measurement'], ['The', 'generation', 'of', 'random',
'binary', 'unordered', 'trees'], ['The', 'intersection',
'graph', 'of', 'paths', 'in', 'trees'], ['Graph', 'minors',
'IV', 'Widths', 'of', 'trees', 'and', 'well', 'quasi',
'ordering'], ['Graph', 'minors', 'A', 'survey']]
>>>
```

Weight	Word
7	of
3	user
3	system
3	trees
3	The
2	response
2	computer

Randomize

Shuffles classes, attributes and/or metas of an input data set.

Signals

Inputs:

- **Data**

  Data set.

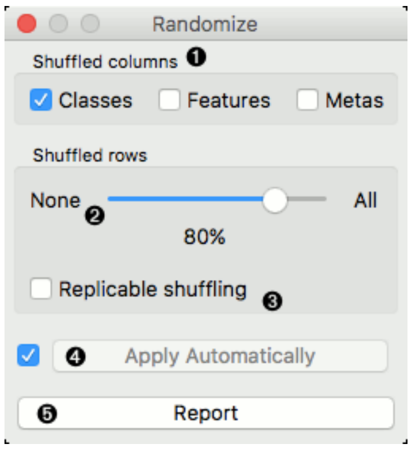
Outputs:

- **Data**

  Randomized data set.

Description

The **Randomize** widget receives a data set in the input and outputs the same data set in which the classes, attributes or/and metas are shuffled.

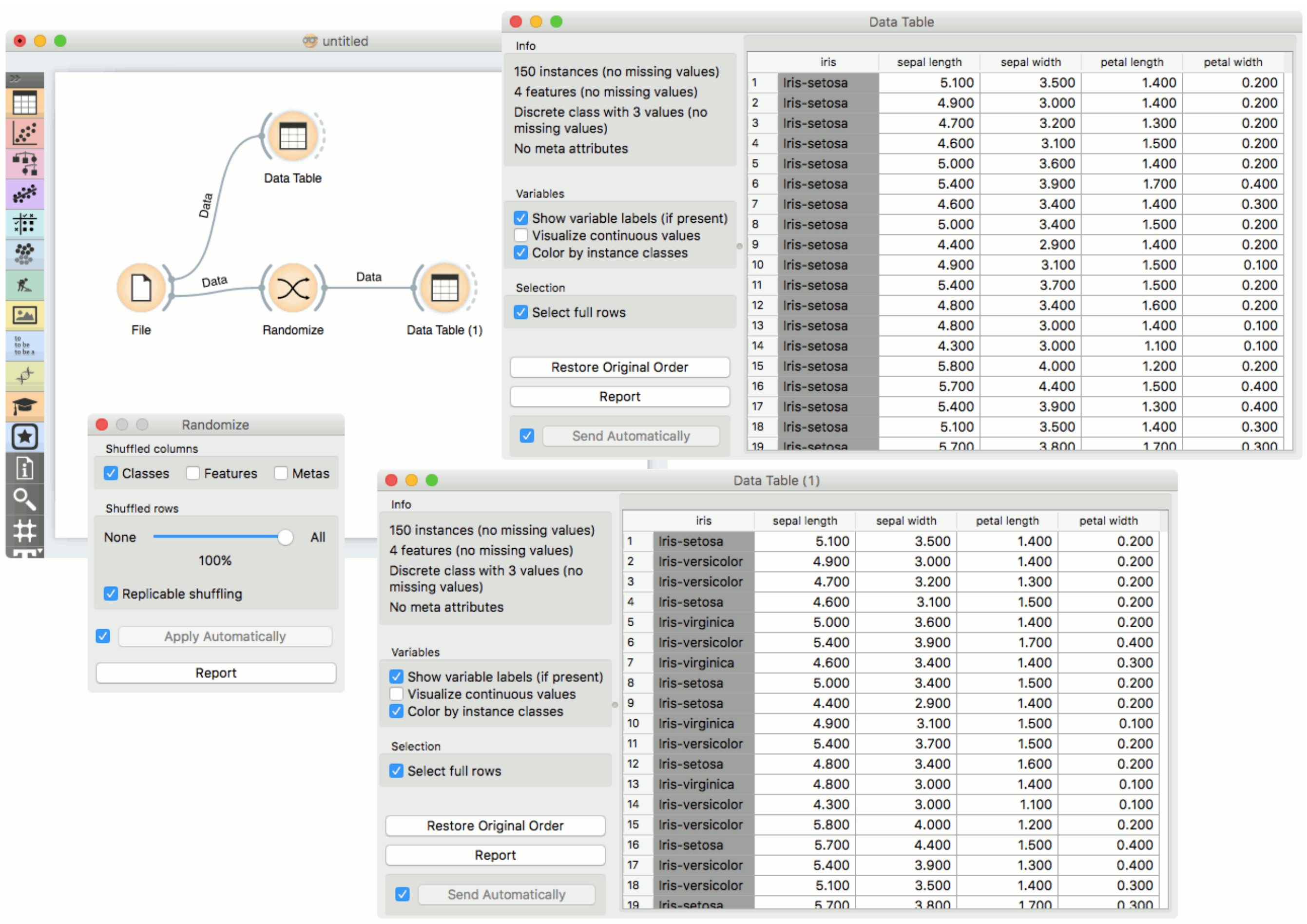


Classes ✓, Features, Metas
80%
Replicable shuffling 3

1. Select group of columns of the data set you want to shuffle.
2. Select proportion of the data set you want to shuffle.
3. Produce replicable output.
4. If *Apply automatically* is ticked, changes are committed automatically. Otherwise, you have to press *Apply* after each change.
5. Produce a report.

Example

The **Randomize** widget is usually placed right after (e.g. File widget. The basic usage is shown in the following workflow, where values of class variable of Iris data set are randomly shuffled.



Randomize

Shuffled columns: [✓ Classes] Features Metas

Shuffled rows: None — All, 100%

[✓] Replicable shuffling

[✓] Apply Automatically

Report

Data Table

Info

150 instances (no missing values)
4 features (no missing values)
Discrete class with 3 values (no missing values)
No meta attributes

Variables: [✓] Show variable labels (if present); Visualize continuous values; [✓] Color by instance classes

Selection: [✓] Select full rows

Restore Original Order

Report

[✓] Send Automatically

	iris	sepal length	sepal width	petal length	petal width
1	Iris-setosa	5.100	3.500	1.400	0.200
2	Iris-setosa	4.900	3.000	1.400	0.200
3	Iris-setosa	4.700	3.200	1.300	0.200
4	Iris-setosa	4.600	3.100	1.500	0.200
5	Iris-setosa	5.000	3.600	1.400	0.200
6	Iris-setosa	5.400	3.900	1.700	0.400
7	Iris-setosa	4.600	3.400	1.400	0.300
8	Iris-setosa	5.000	3.400	1.500	0.200
9	Iris-setosa	4.400	2.900	1.400	0.200
10	Iris-setosa	4.900	3.100	1.500	0.100
11	Iris-setosa	5.400	3.700	1.500	0.200
12	Iris-setosa	4.800	3.400	1.600	0.200
13	Iris-setosa	4.800	3.000	1.400	0.100
14	Iris-setosa	4.300	3.000	1.100	0.100
15	Iris-setosa	5.800	4.000	1.200	0.200
16	Iris-setosa	5.700	4.400	1.500	0.400
17	Iris-setosa	5.400	3.900	1.300	0.400
18	Iris-setosa	5.100	3.500	1.400	0.300
19	Iris-setosa	5.700	3.800	1.700	0.300

Data Table (1)

Info

150 instances (no missing values)
4 features (no missing values)
Discrete class with 3 values (no missing values)
No meta attributes

Variables: [✓] Show variable labels (if present); Visualize continuous values; [✓] Color by instance classes

Selection: [✓] Select full rows

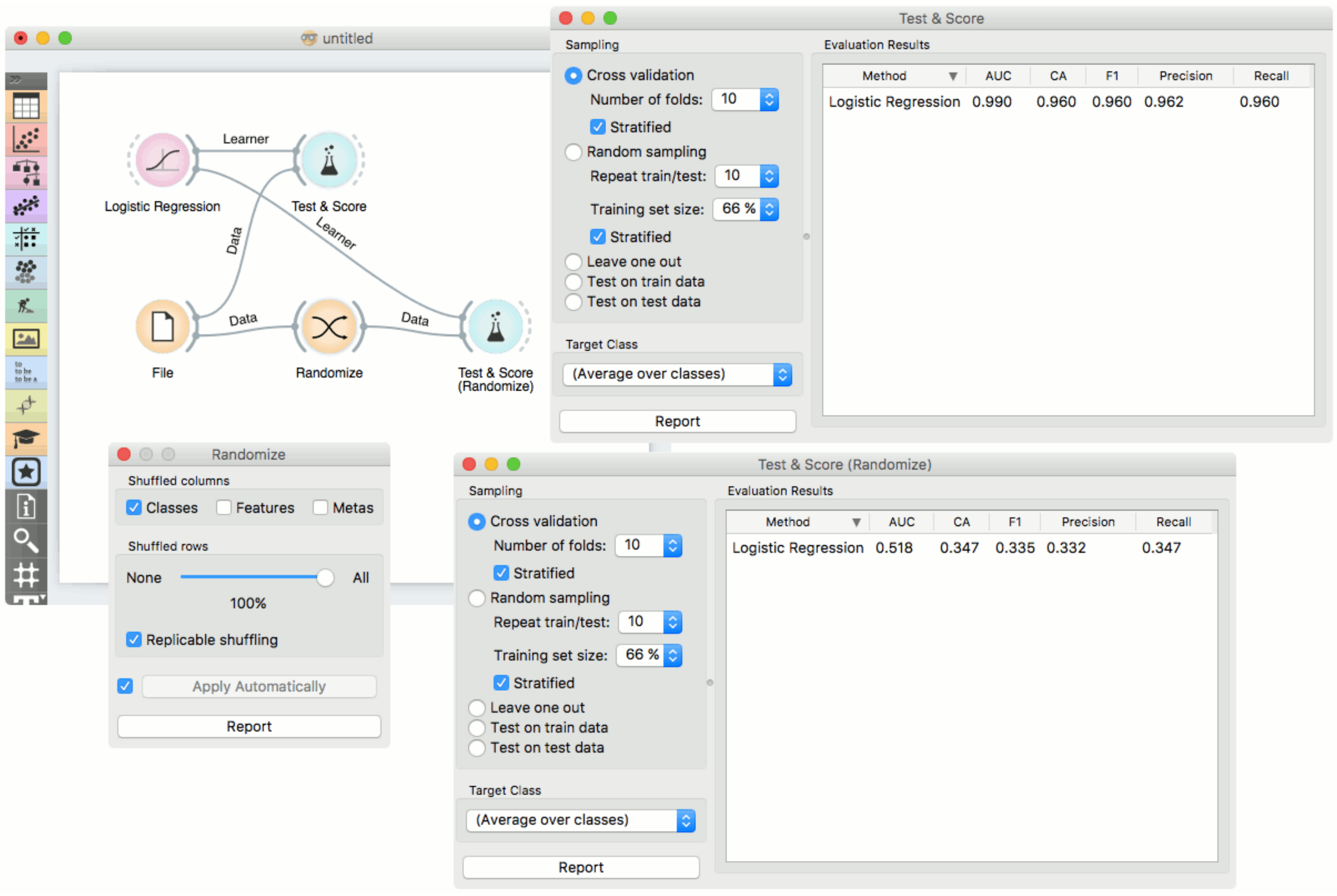
Restore Original Order

Report

[✓] Send Automatically

	iris	sepal length	sepal width	petal length	petal width
1	Iris-setosa	5.100	3.500	1.400	0.200
2	Iris-versicolor	4.900	3.000	1.400	0.200
3	Iris-versicolor	4.700	3.200	1.300	0.200
4	Iris-setosa	4.600	3.100	1.500	0.200
5	Iris-virginica	5.000	3.600	1.400	0.200
6	Iris-versicolor	5.400	3.900	1.700	0.400
7	Iris-virginica	4.600	3.400	1.400	0.300
8	Iris-setosa	5.000	3.400	1.500	0.200
9	Iris-setosa	4.400	2.900	1.400	0.200
10	Iris-virginica	4.900	3.100	1.500	0.100
11	Iris-versicolor	5.400	3.700	1.500	0.200
12	Iris-setosa	4.800	3.400	1.600	0.200
13	Iris-virginica	4.800	3.000	1.400	0.100
14	Iris-versicolor	4.300	3.000	1.100	0.100
15	Iris-versicolor	5.800	4.000	1.200	0.200
16	Iris-setosa	5.700	4.400	1.500	0.400
17	Iris-versicolor	5.400	3.900	1.300	0.400
18	Iris-versicolor	5.100	3.500	1.400	0.300
19	Iris-setosa	5.700	3.800	1.700	0.300

In the next example we show how shuffling class values influences model performance on the same data set as above.



Randomize

Shuffled columns: [✓ Classes] Features Metas

Shuffled rows: None — All, 100%

[✓] Replicable shuffling

[✓] Apply Automatically

Report

Test & Score

Sampling: [✓ Cross validation], Number of folds: 10, [✓] Stratified; Random sampling, Repeat train/test: 10, Training set size: 66 %, [✓] Stratified; Leave one out; Test on train data; Test on test data

Target Class: (Average over classes)

Report

Evaluation Results

Method	AUC	CA	F1	Precision	Recall
Logistic Regression	0.990	0.960	0.960	0.962	0.960

Test & Score (Randomize)

Sampling: [✓ Cross validation], Number of folds: 10, [✓] Stratified; Random sampling, Repeat train/test: 10, Training set size: 66 %, [✓] Stratified; Leave one out; Test on train data; Test on test data

Target Class: (Average over classes)

Report

Evaluation Results

Method	AUC	CA	F1	Precision	Recall
Logistic Regression	0.518	0.347	0.335	0.332	0.347

Rank

Ranking of attributes in classification or regression data sets.

Signals

Inputs:

- **Data**

  An input data set.

- **Scorer** (multiple)

  Models that implement the feature scoring interface, such as linear / logistic regression, random forest, stochastic gradient descent, etc.

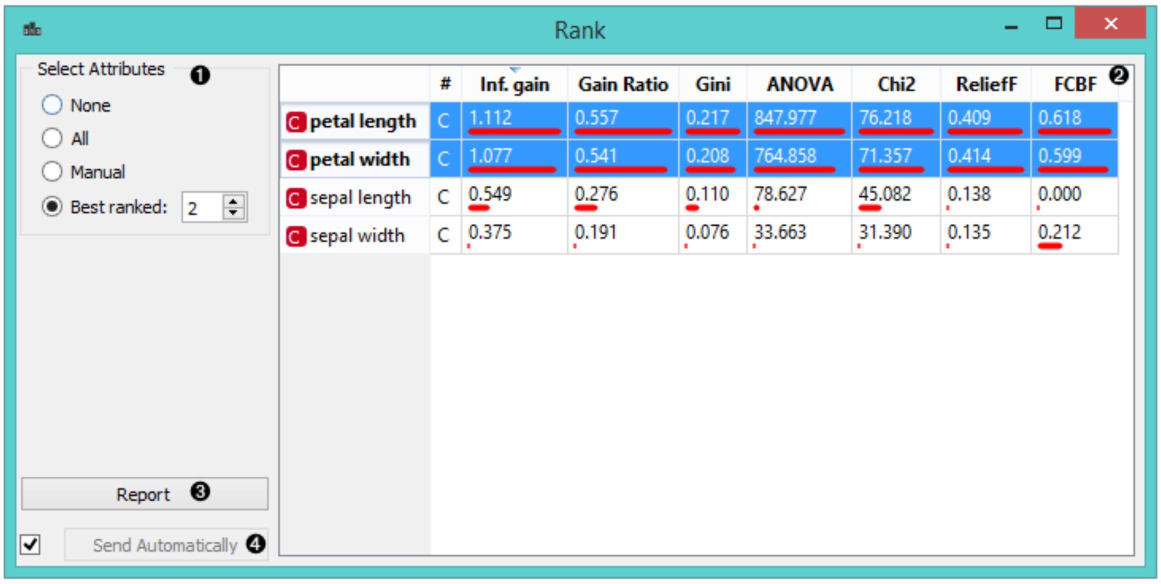
Outputs:

- **Reduced Data**

  A data set whith selected attributes.

Description

The **Rank** widget considers class-labeled data sets (classification or regression) and scores the attributes according to their correlation with the class.



Select Attributes 1

None
All
Manual
[✓ Best ranked: 2]

✓ Send Automatically

	#	Inf. gain	Gain Ratio	Gini	ANOVA	Chi2	ReliefF	FCBF 2
C petal length	C	1.112	0.557	0.217	847.977	76.218	0.409	0.618
C petal width	C	1.077	0.541	0.208	764.858	71.357	0.414	0.599
C sepal length	C	0.549	0.276	0.110	78.627	45.082	0.138	0.000
C sepal width	C	0.375	0.191	0.076	33.663	31.390	0.135	0.212

1. Select attributes from the data table.
2. Data table with attributes (rows) and their scores by different scoring methods (columns)
3. Produce a report.
4. If '*Send Automatically*' is ticked, the widget automatically communicates changes to other widgets.

Scoring methods

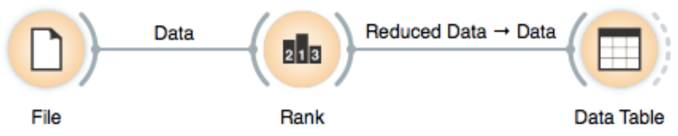
1. Information Gain: the expected amount of information (reduction of entropy)

2. Gain Ratio: a ratio of the information gain and the attribute's intrinsic information, which reduces the bias towards multivalued features that occurs in information gain
3. Gini: the inequality among values of a frequency distribution
4. ANOVA: the difference between average vaules of the feature in different classes
5. Chi2: dependence between the feature and the class as measure by the chi-square statistice
6. ReliefF: the ability of an attribute to distinguish between classes on similar data instances
7. FCBF (Fast Correlation Based Filter): entropy-based measure, which also identifies redundancy due to pairwise correlations between features

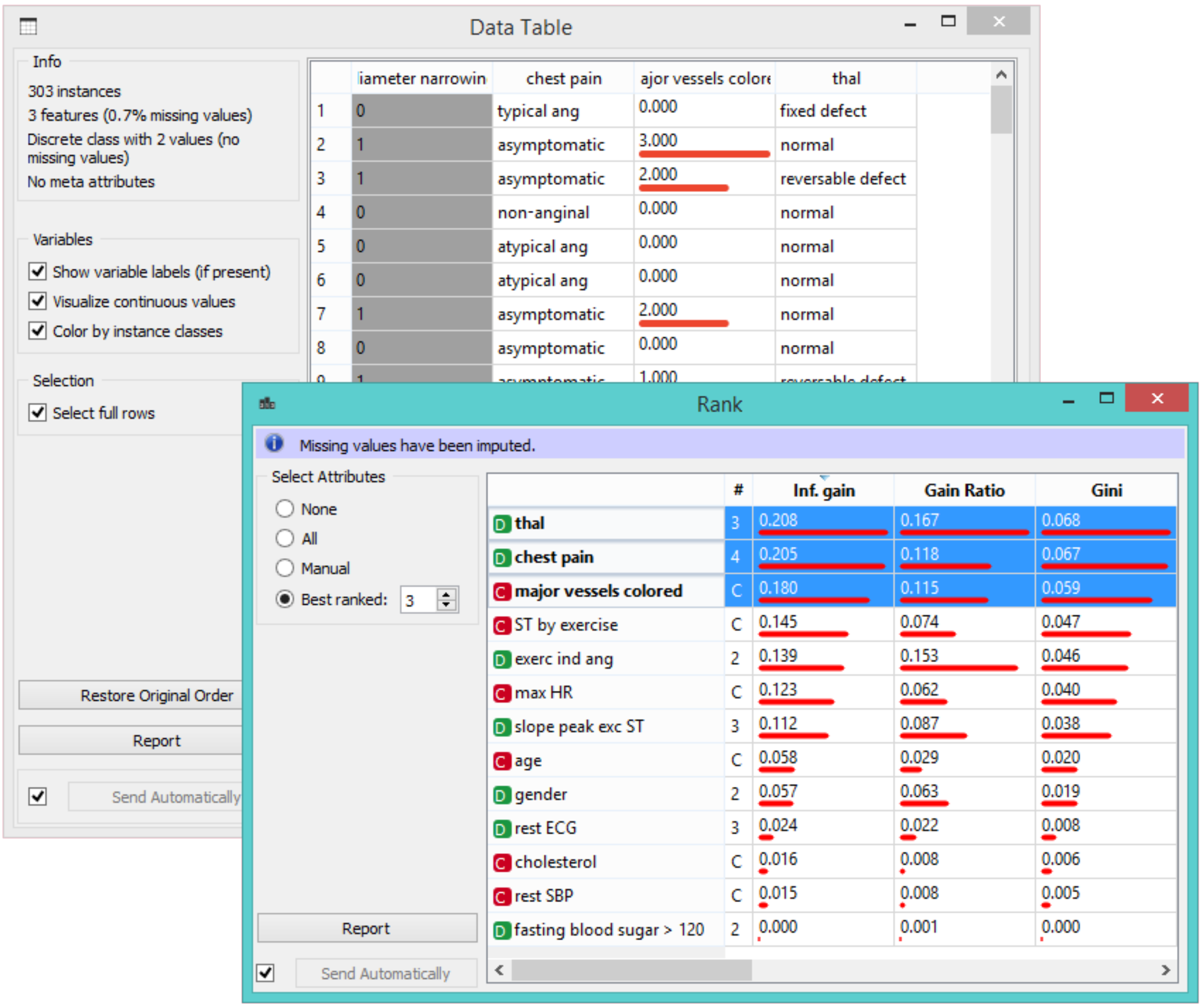
Additionally, you can connect certain learners that enable scoring the features according to how important they are in models that the learners build (e.g. Linear / Logistic Regression, Random Forest, SGD, …).

Example: Attribute Ranking and Selection

Below, we have used the **Rank** widget immediately after the File widget to reduce the set of data attributes and include only the most informative ones:



Notice how the widget outputs a data set that includes only the best-scored attributes:



Data Table

Info
303 instances
3 features (0.7% missing values)
Discrete class with 2 values (no missing values)
No meta attributes

Variables
☑ Show variable labels (if present)
☑ Visualize continuous values
☑ Color by instance classes

Selection
☑ Select full rows

Restore Original Order
Report
☑ Send Automatically

	iameter narrowin	chest pain	ajor vessels colore	thal
1	0	typical ang	0.000	fixed defect
2	1	asymptomatic	3.000	normal
3	1	asymptomatic	2.000	reversable defect
4	0	non-anginal	0.000	normal
5	0	atypical ang	0.000	normal
6	0	atypical ang	0.000	normal
7	1	asymptomatic	2.000	normal
8	0	asymptomatic	0.000	normal
9	1	asymptomatic	1.000	reversable defect

Rank

Missing values have been imputed.

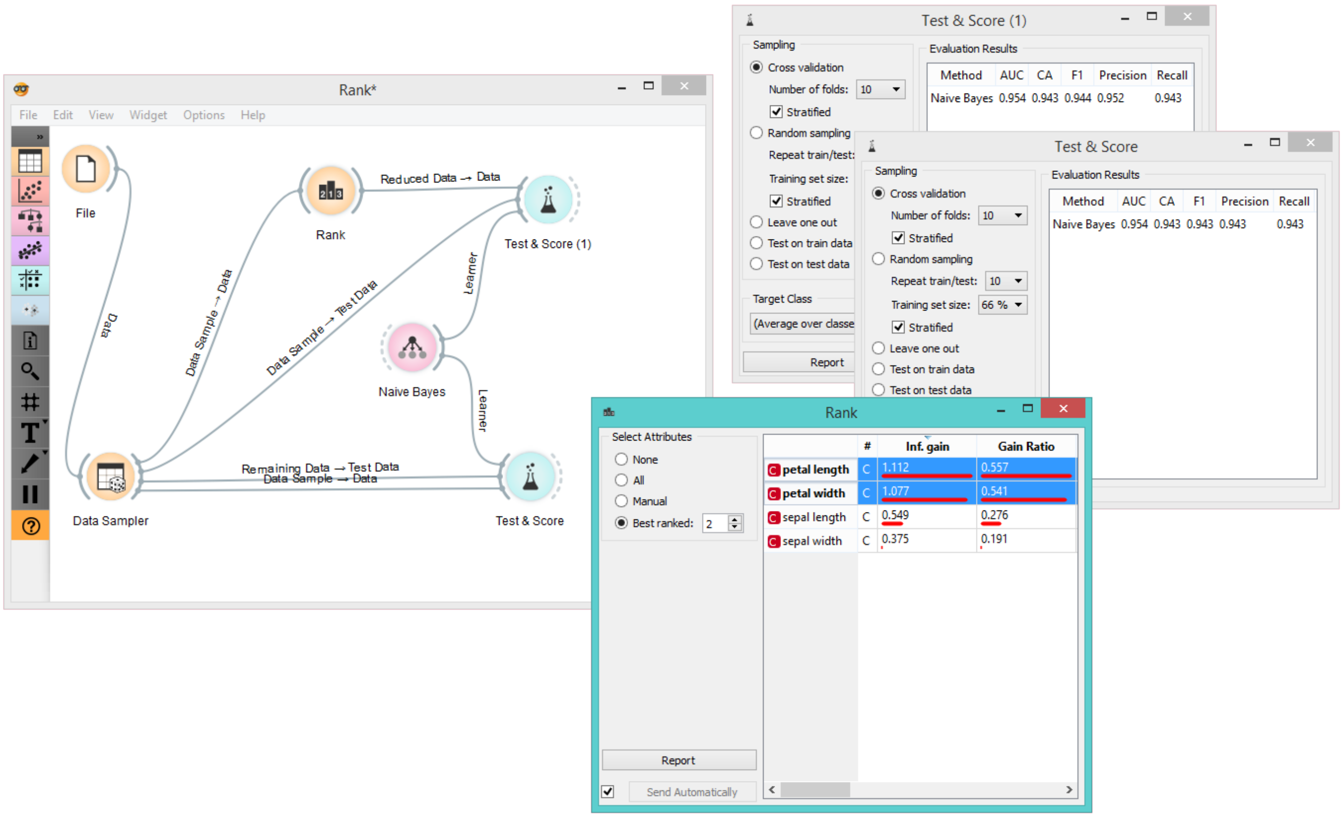
Select Attributes
○ None
○ All
○ Manual
● Best ranked: 3

Report
☑ Send Automatically

	#	Inf. gain	Gain Ratio	Gini
D thal	3	0.208	0.167	0.068
D chest pain	4	0.205	0.118	0.067
C major vessels colored	C	0.180	0.115	0.059
C ST by exercise	C	0.145	0.074	0.047
D exerc ind ang	2	0.139	0.153	0.046
C max HR	C	0.123	0.062	0.040
D slope peak exc ST	3	0.112	0.087	0.038
C age	C	0.058	0.029	0.020
D gender	2	0.057	0.063	0.019
D rest ECG	3	0.024	0.022	0.008
C cholesterol	C	0.016	0.008	0.006
C rest SBP	C	0.015	0.008	0.005
D fasting blood sugar > 120	2	0.000	0.001	0.000

Example: Feature Subset Selection for Machine Learning

What follows is a bit more complicated example. In the workflow below, we first split the data into a training set and a test set. In the upper branch, the training data passes through the **Rank** widget to select the most informative attributes, while in the lower branch there is no feature selection. Both feature selected and original data sets are passed to their own Test & Score widgets, which develop a *Naive Bayes* classifier and score it on a test set.



Test & Score (1)

Sampling: [✓ Cross validation] Number of folds: 10; [✓ Stratified]; Random sampling; Repeat train/test:; Training set size:; [✓ Stratified]; Leave one out; Test on train data; Test on test data

Target Class: (Average over classes)

Evaluation Results

Method	AUC	CA	F1	Precision	Recall
Naive Bayes	0.954	0.943	0.944	0.952	0.943

Test & Score

Sampling: [✓ Cross validation] Number of folds: 10; [✓ Stratified]; Random sampling; Repeat train/test: 10; Training set size: 66 %; [✓ Stratified]; Leave one out; Test on train data; Test on test data

Evaluation Results

Method	AUC	CA	F1	Precision	Recall
Naive Bayes	0.954	0.943	0.943	0.943	0.943

Rank

Select Attributes: None; All; Manual; [✓ Best ranked: 2]

	#	Inf. gain	Gain Ratio
petal length	C	1.112	0.557
petal width	C	1.077	0.541
sepal length	C	0.549	0.276
sepal width	C	0.375	0.191

For data sets with many features, a naive Bayesian classifier feature selection, as shown above, would often yield a better predictive accuracy.

Save Data

Saves data to a file.

Signals

Inputs:

- **Data**

  A data set.

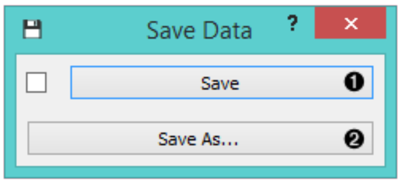
Outputs:

- (None)

Description

The **Save Data** widget considers a data set provided in the input channel and saves it to a data file with a specified name. It can save the data as a tab-delimited or a comma-separated file.

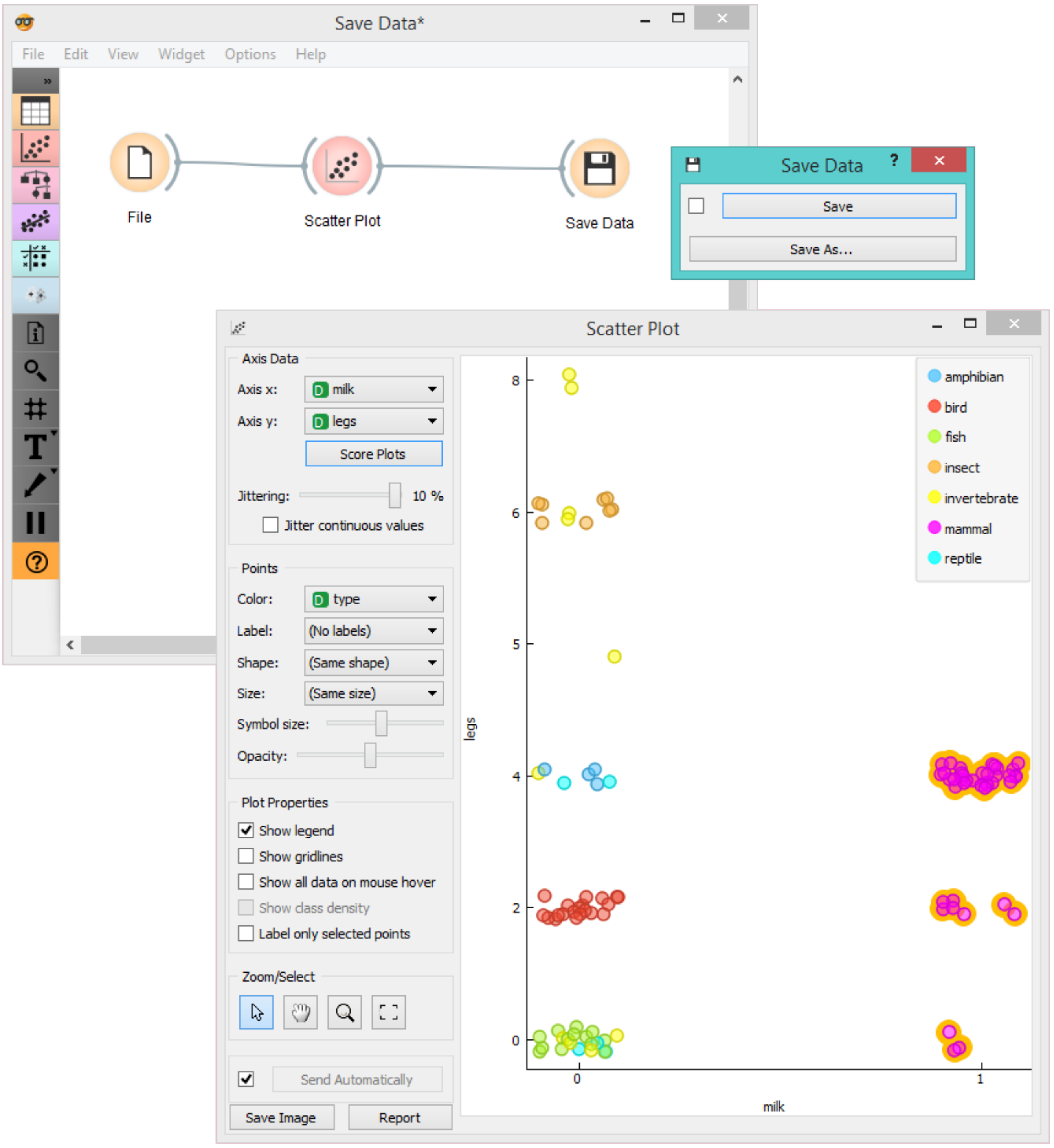
The widget does not save the data every time it receives a new signal in the input as this would constantly (and, mostly, inadvertently) overwrite the file. Instead, the data is saved only after a new file name is set or the user pushes the *Save* button.



1. Save by overwriting the existing file.
2. *Save as* to create a new file.

Example

In the workflow below, we used the *Zoo* data set. We loaded the data into the Scatter Plot widget, with which we selected a subset of data instances and pushed them to the **Save Data** widget to store them in a file.



Axis Data

Axis x: milk

Axis y: legs

Jittering: 10 %

☐ Jitter continuous values

Points

Color: type

Label: (No labels)

Shape: (Same shape)

Size: (Same size)

Symbol size:

Opacity:

Plot Properties

☑ Show legend

☐ Show gridlines

☐ Show all data on mouse hover

☐ Show class density

☐ Label only selected points

Zoom/Select

☑ Send Automatically

● amphibian
● bird
● fish
● insect
● invertebrate
● mammal
● reptile

Select Columns

Manual selection of data attributes and composition of data domain.

Signals

Inputs:

- **Data**

  Attribute-valued data set.

Outputs:

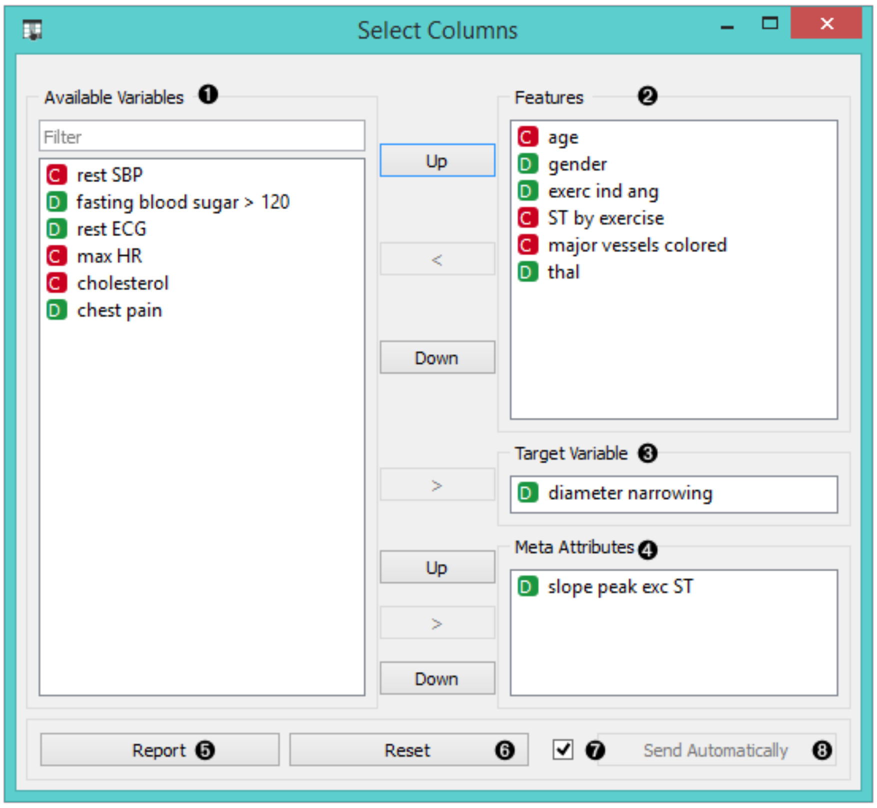
- **Data**

  Attribute-valued data set composed using the domain specification from the widget.

Description

The **Select Columns** widget is used to manually compose your data domain. The user can decide which attributes will be used and how. Orange distinguishes between ordinary attributes, (optional) class attributes and meta attributes. For instance, for building a classification model, the domain would be composed of a set of attributes and a discrete class attribute. Meta attributes are not used in modelling, but several widgets can use them as instance labels.

Orange attributes have a type and are either discrete, continuous or a character string. The attribute type is marked with a symbol appearing before the name of the attribute (D, C, S, respectively).

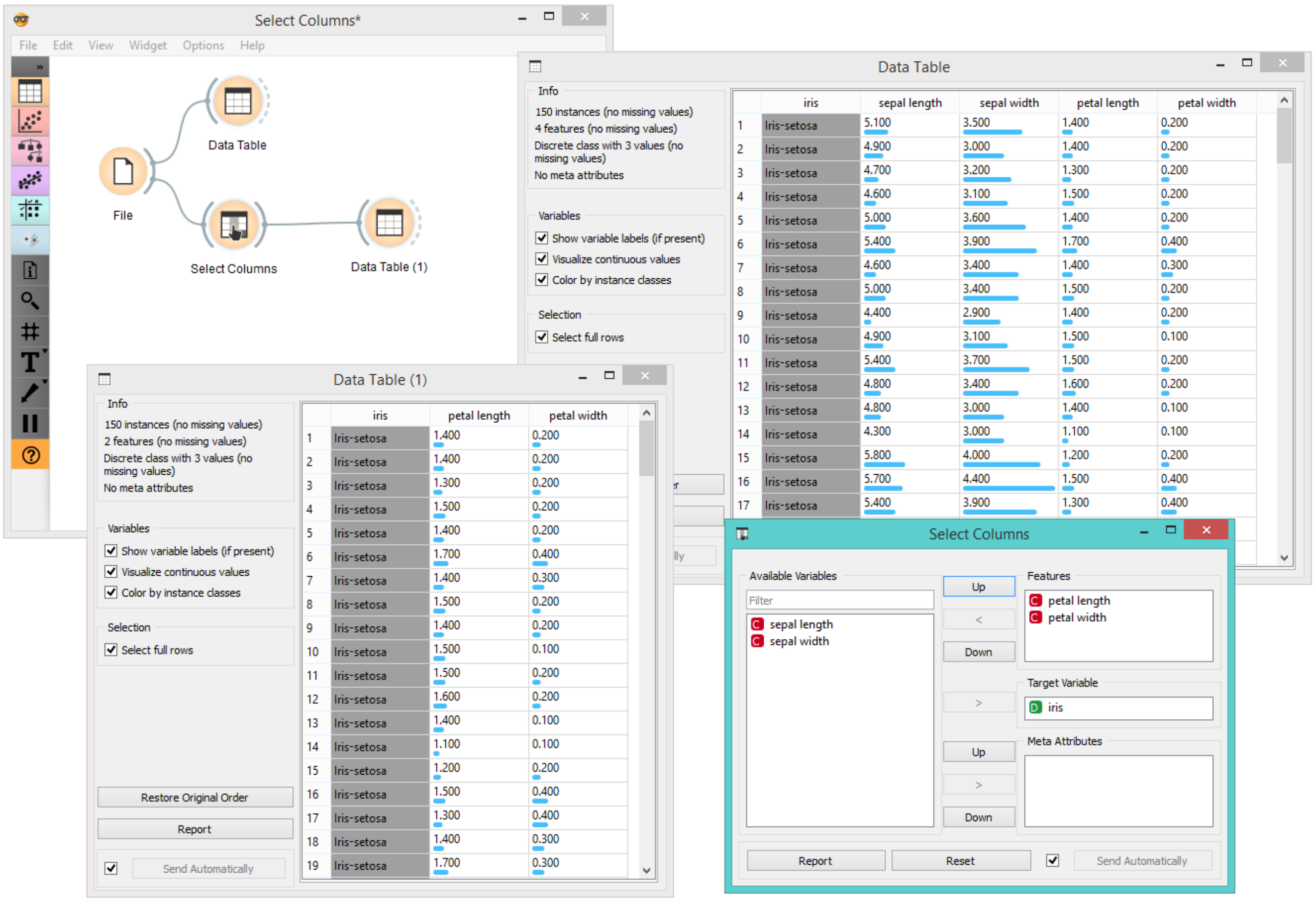


1. Left-out data attributes that will not be in the output data file
2. Data attributes in the new data file

3. Target variable. If none, the new data set will be without a target variable.
4. Meta attributes of the new data file. These attributes are included in the data set but are, for most methods, not considered in the analysis.
5. Produce a report.
6. Reset the domain composition to that of the input data file.
7. Tick if you wish to auto-apply changes of the data domain.
8. Apply changes of the data domain and send the new data file to the output channel of the widget.

Examples

In the workflow below, the *Iris* data from the File widget is fed into the **Select Columns** widget, where we select to output only two attributes (namely petal width and petal length). We view both the original data set and the data set with selected columns in the Data Table widget.



Data Table

Info
150 instances (no missing values)
4 features (no missing values)
Discrete class with 3 values (no missing values)
No meta attributes

Variables
☑ Show variable labels (if present)
☑ Visualize continuous values
☑ Color by instance classes

Selection
☑ Select full rows

	iris	sepal length	sepal width	petal length	petal width
1	Iris-setosa	5.100	3.500	1.400	0.200
2	Iris-setosa	4.900	3.000	1.400	0.200
3	Iris-setosa	4.700	3.200	1.300	0.200
4	Iris-setosa	4.600	3.100	1.500	0.200
5	Iris-setosa	5.000	3.600	1.400	0.200
6	Iris-setosa	5.400	3.900	1.700	0.400
7	Iris-setosa	4.600	3.400	1.400	0.300
8	Iris-setosa	5.000	3.400	1.500	0.200
9	Iris-setosa	4.400	2.900	1.400	0.200
10	Iris-setosa	4.900	3.100	1.500	0.100
11	Iris-setosa	5.400	3.700	1.500	0.200
12	Iris-setosa	4.800	3.400	1.600	0.200
13	Iris-setosa	4.800	3.000	1.400	0.100
14	Iris-setosa	4.300	3.000	1.100	0.100
15	Iris-setosa	5.800	4.000	1.200	0.200
16	Iris-setosa	5.700	4.400	1.500	0.400
17	Iris-setosa	5.400	3.900	1.300	0.400

Data Table (1)

Info
150 instances (no missing values)
2 features (no missing values)
Discrete class with 3 values (no missing values)
No meta attributes

Variables
☑ Show variable labels (if present)
☑ Visualize continuous values
☑ Color by instance classes

Selection
☑ Select full rows

Restore Original Order
Report
☑ Send Automatically

	iris	petal length	petal width
1	Iris-setosa	1.400	0.200
2	Iris-setosa	1.400	0.200
3	Iris-setosa	1.300	0.200
4	Iris-setosa	1.500	0.200
5	Iris-setosa	1.400	0.200
6	Iris-setosa	1.700	0.400
7	Iris-setosa	1.400	0.300
8	Iris-setosa	1.500	0.200
9	Iris-setosa	1.400	0.200
10	Iris-setosa	1.500	0.100
11	Iris-setosa	1.500	0.200
12	Iris-setosa	1.600	0.200
13	Iris-setosa	1.400	0.100
14	Iris-setosa	1.100	0.100
15	Iris-setosa	1.200	0.200
16	Iris-setosa	1.500	0.400
17	Iris-setosa	1.300	0.400
18	Iris-setosa	1.400	0.300
19	Iris-setosa	1.700	0.300

Select Columns

Available Variables
Filter
C sepal length
C sepal width

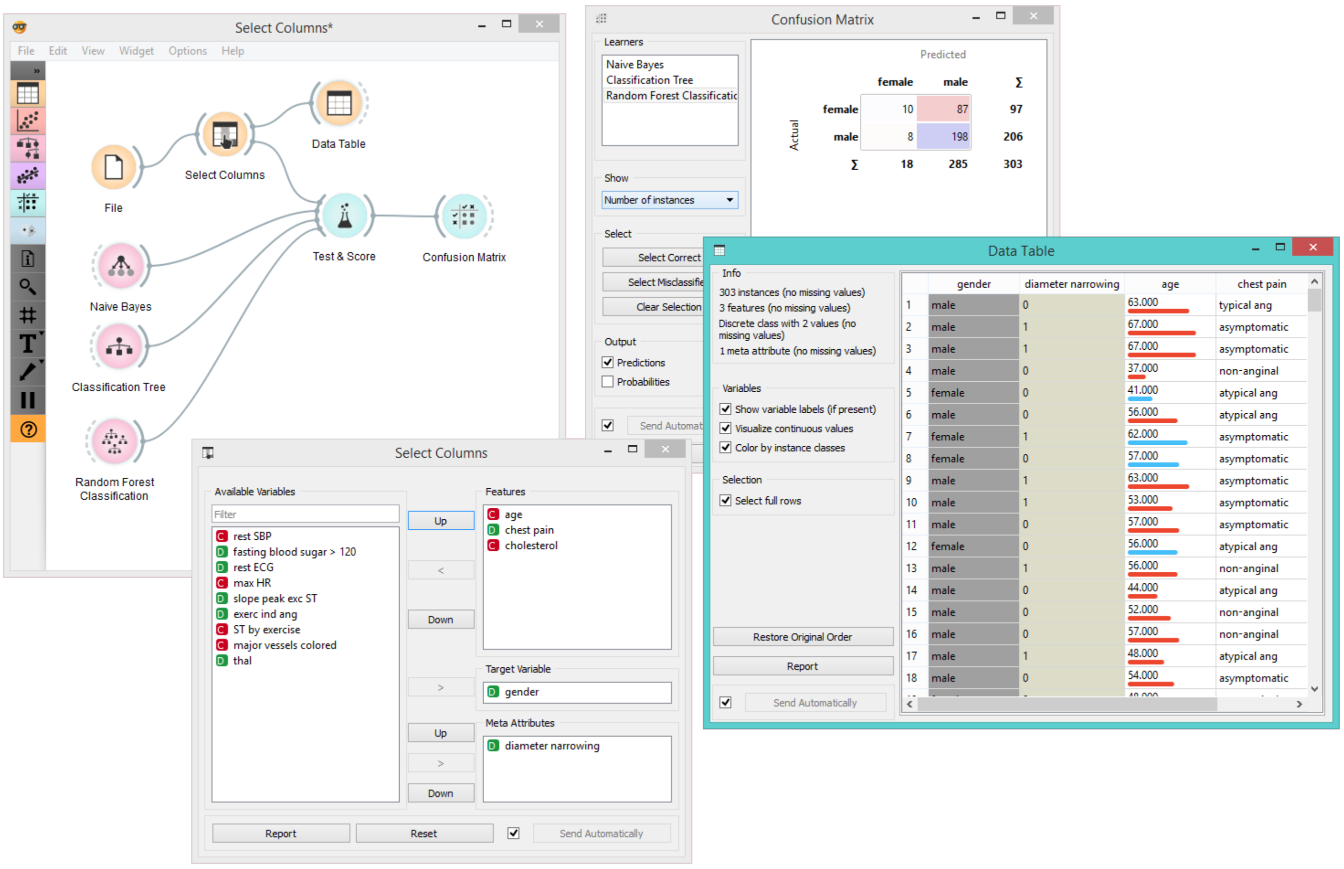
Features
C petal length
C petal width

Target Variable
D iris

Meta Attributes

Up < Down > Up > Down
Report Reset ☑ Send Automatically

For a more complex use of the widget, we composed a workflow to redefine the classification problem in the *heart-disease* data set. Originally, the task was to predict if the patient has a coronary artery diameter narrowing. We changed the problem to that of gender classification, based on age, chest pain and cholesterol level, and informatively kept the diameter narrowing as a meta attribute.



Confusion Matrix

Learners
Naive Bayes
Classification Tree
Random Forest Classificatic

Show: Number of instances

Select
Select Correct
Select Misclassifie
Clear Selection

Output
[✓] Predictions
[ ] Probabilities
[✓] Send Automat

		Predicted		
Actual		female	male	∑
	female	10	87	97
	male	8	198	206
	∑	18	285	303

Data Table

Info
303 instances (no missing values)
3 features (no missing values)
Discrete class with 2 values (no missing values)
1 meta attribute (no missing values)

Variables
[✓] Show variable labels (if present)
[✓] Visualize continuous values
[✓] Color by instance classes

Selection
[✓] Select full rows

Restore Original Order
Report
[✓] Send Automatically

	gender	diameter narrowing	age	chest pain
1	male	0	63.000	typical ang
2	male	1	67.000	asymptomatic
3	male	1	67.000	asymptomatic
4	male	0	37.000	non-anginal
5	female	0	41.000	atypical ang
6	male	0	56.000	atypical ang
7	female	1	62.000	asymptomatic
8	female	0	57.000	asymptomatic
9	male	1	63.000	asymptomatic
10	male	1	53.000	asymptomatic
11	male	0	57.000	asymptomatic
12	female	0	56.000	atypical ang
13	male	1	56.000	non-anginal
14	male	0	44.000	atypical ang
15	male	0	52.000	non-anginal
16	male	0	57.000	non-anginal
17	male	1	48.000	atypical ang
18	male	0	54.000	asymptomatic

Select Columns

Available Variables
Filter
C rest SBP
D fasting blood sugar > 120
D rest ECG
C max HR
D slope peak exc ST
D exerc ind ang
C ST by exercise
C major vessels colored
D thal

Features
C age
D chest pain
C cholesterol

Target Variable
D gender

Meta Attributes
D diameter narrowing

Select Rows

Selects data instances based on conditions over data features.

Signals

Inputs:

- **Data**

  Data set.

Outputs:

- **Matching Data**

  Instances that match the conditions.

- **Non-Matching Data**

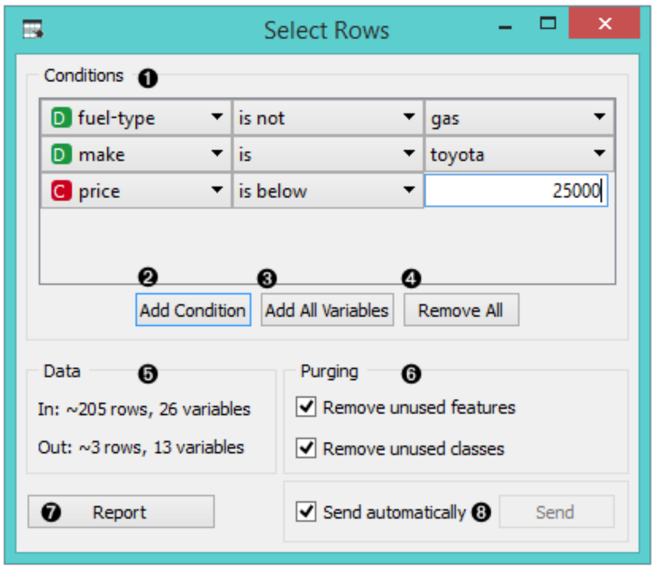
  Instances that do not match the conditions.

Description

This widget selects a subset from an input data set, based on user-defined conditions. Instances that match the selection rule are placed in the output *Matching Data* channel.

Criteria for data selection are presented as a collection of conjuncted terms (i.e. selected items are those matching all the terms in '*Conditions*').

Condition terms are defined through selecting an attribute, selecting an operator from a list of operators, and, if needed, defining the value to be used in the condition term. Operators are different for discrete, continuous and string attributes.



D fuel-type	is not	gas
D make	is	toyota
C price	is below	25000

In: ~205 rows, 26 variables

Out: ~3 rows, 13 variables

✓ Remove unused features

✓ Remove unused classes

✓ Send automatically

1. Conditions you want to apply, their operators and related values
2. Add a new condition to the list of conditions.
3. Add all the possible variables at once.

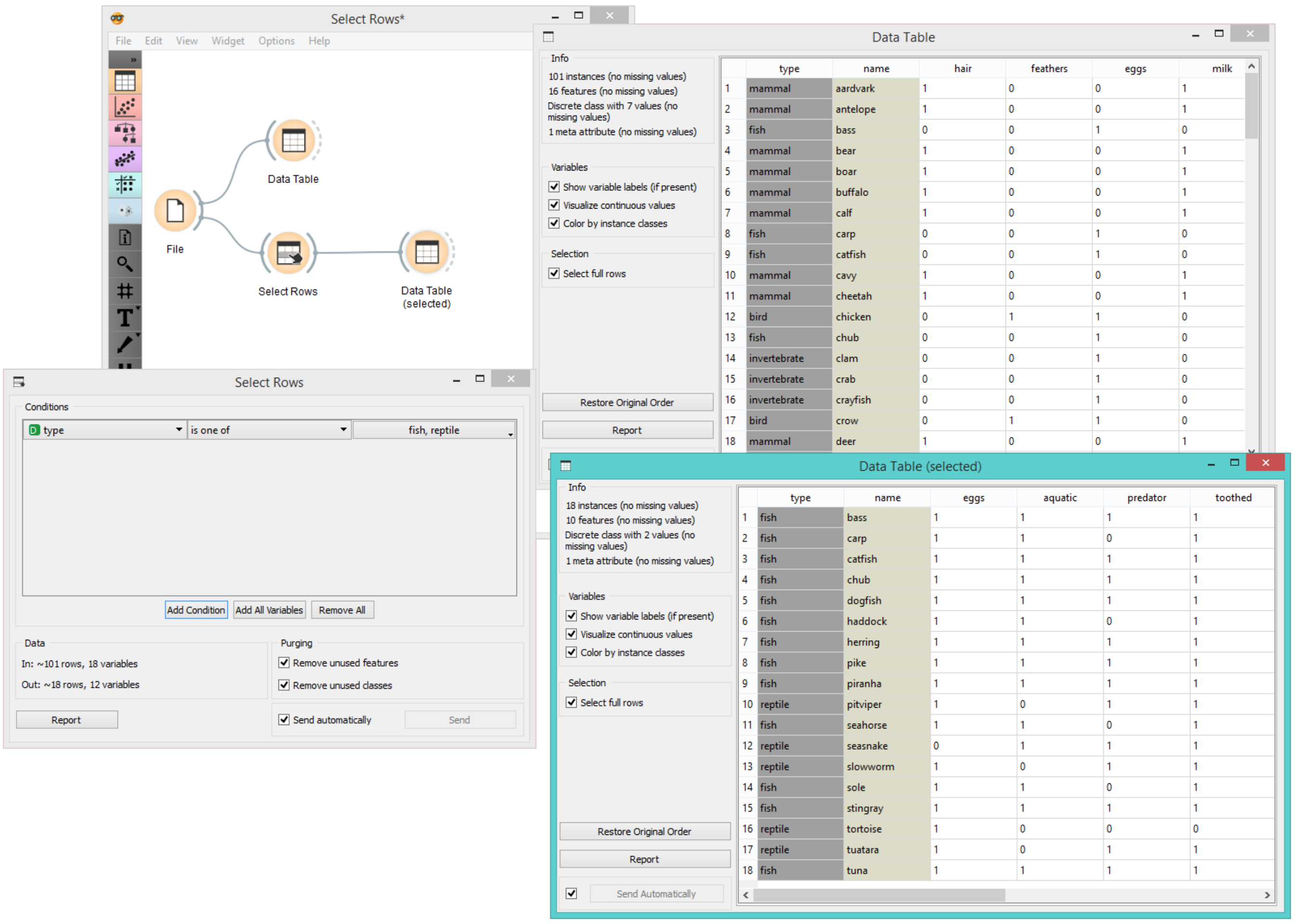
4. Remove all the listed variables at once.
5. Information on the input data set and information on instances that match the condition(s)
6. Purge the output data.
7. When the *Send automatically* box is ticked, all changes will be automatically communicated to other widgets.
8. Produce a report.

Any change in the composition of the condition will update the information pane (*Data Out*).

If *Send automatically* is selected, then the output is updated on any change in the composition of the condition or any of its terms.

Example

In the workflow below, we used the *Zoo* data from the File widget and fed it into the **Select Rows** widget. In the widget, we chose to output only two animal types, namely fish and reptiles. We can inspect both the original data set and the data set with selected rows in the Data Table widget.



Select Rows

Conditions

type | is one of | fish, reptile

Add Condition | Add All Variables | Remove All

Data

In: ~101 rows, 18 variables

Out: ~18 rows, 12 variables

Purging

☑ Remove unused features

☑ Remove unused classes

Report

☑ Send automatically

Send

Data Table

Info

101 instances (no missing values)
16 features (no missing values)
Discrete class with 7 values (no missing values)
1 meta attribute (no missing values)

Variables

☑ Show variable labels (if present)

☑ Visualize continuous values

☑ Color by instance classes

Selection

☑ Select full rows

Restore Original Order

Report

	type	name	hair	feathers	eggs	milk
1	mammal	aardvark	1	0	0	1
2	mammal	antelope	1	0	0	1
3	fish	bass	0	0	1	0
4	mammal	bear	1	0	0	1
5	mammal	boar	1	0	0	1
6	mammal	buffalo	1	0	0	1
7	mammal	calf	1	0	0	1
8	fish	carp	0	0	1	0
9	fish	catfish	0	0	1	0
10	mammal	cavy	1	0	0	1
11	mammal	cheetah	1	0	0	1
12	bird	chicken	0	1	1	0
13	fish	chub	0	0	1	0
14	invertebrate	clam	0	0	1	0
15	invertebrate	crab	0	0	1	0
16	invertebrate	crayfish	0	0	1	0
17	bird	crow	0	1	1	0
18	mammal	deer	1	0	0	1

Data Table (selected)

Info

18 instances (no missing values)
10 features (no missing values)
Discrete class with 2 values (no missing values)
1 meta attribute (no missing values)

Variables

☑ Show variable labels (if present)

☑ Visualize continuous values

☑ Color by instance classes

Selection

☑ Select full rows

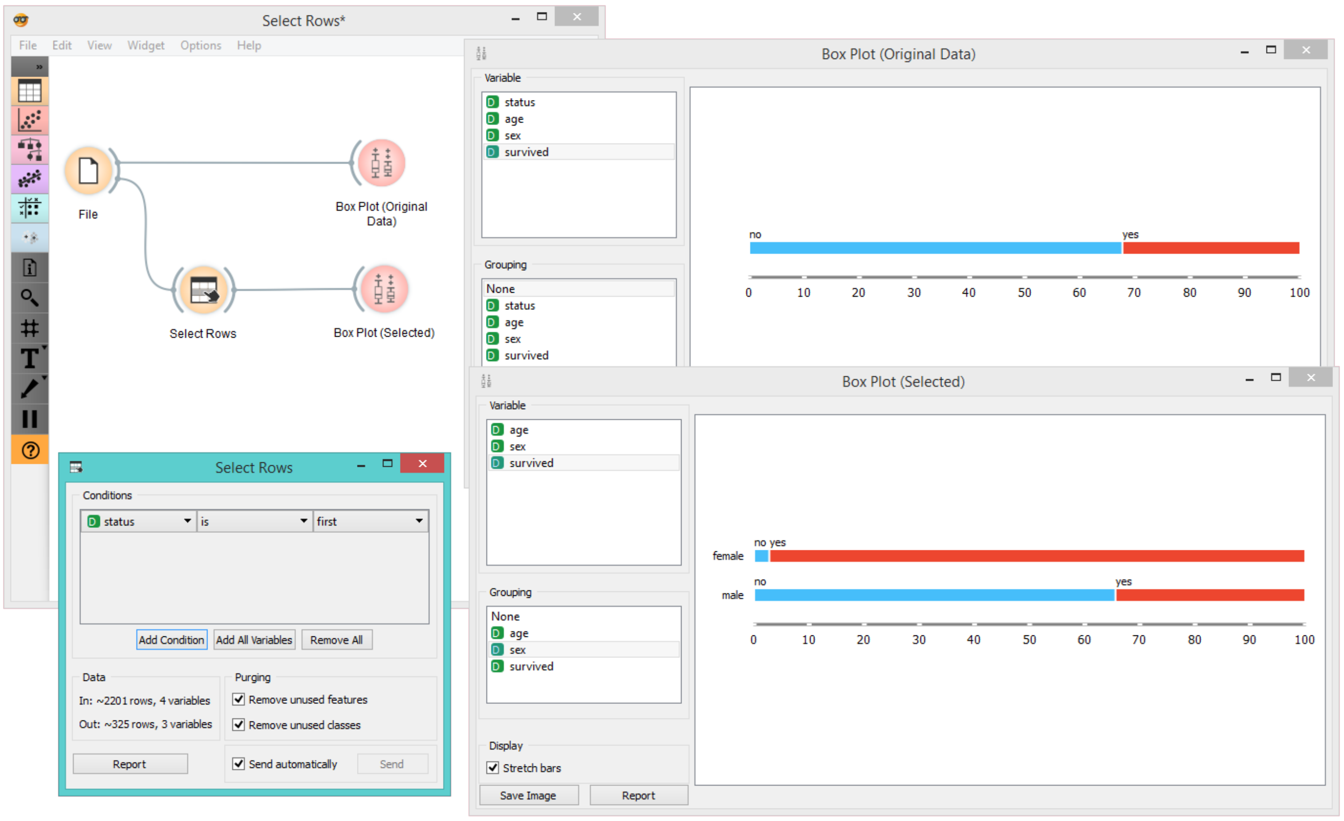
Restore Original Order

Report

☑ Send Automatically

	type	name	eggs	aquatic	predator	toothed
1	fish	bass	1	1	1	1
2	fish	carp	1	1	0	1
3	fish	catfish	1	1	1	1
4	fish	chub	1	1	1	1
5	fish	dogfish	1	1	1	1
6	fish	haddock	1	1	0	1
7	fish	herring	1	1	1	1
8	fish	pike	1	1	1	1
9	fish	piranha	1	1	1	1
10	reptile	pitviper	1	0	1	1
11	fish	seahorse	1	1	0	1
12	reptile	seasnake	0	1	1	1
13	reptile	slowworm	1	0	1	1
14	fish	sole	1	1	0	1
15	fish	stingray	1	1	1	1
16	reptile	tortoise	1	0	0	0
17	reptile	tuatara	1	0	1	1
18	fish	tuna	1	1	1	1

In the next example, we used the data from the *Titanic* data set and similarly fed it into the Box Plot widget. We first observed the entire data set based on survival. Then we selected only first class passengers in the **Select Rows** widget and fed it again into the Box Plot. There we could see all the first class passengers listed by their survival rate and grouped by gender.



Conditions: status is first

Data
In: ~2201 rows, 4 variables
Out: ~325 rows, 3 variables

Purging
☑ Remove unused features
☑ Remove unused classes

☑ Send automatically

SQL Table

Reads data from an SQL database.

Signals

Inputs:

- (None)

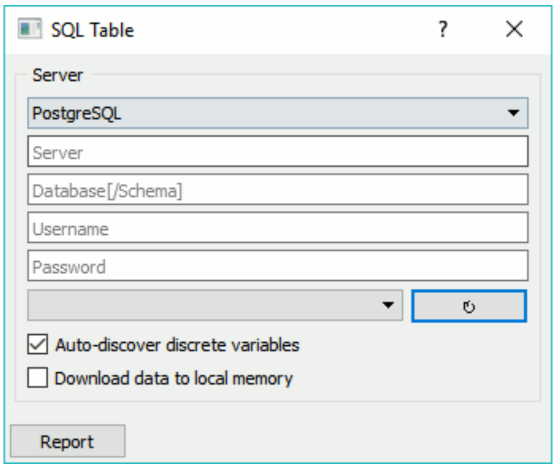
Outputs:

- **Data**

  Attribute-valued data from the database

Description

The **SQL** widget accesses data stored in an SQL database. It can connect to PostgreSQL (requires psycopg2 module) or SQL Server (requires pymssql module).



Transpose

Transposes a data table.

Signals

Inputs:

- **Data**

  A data set.

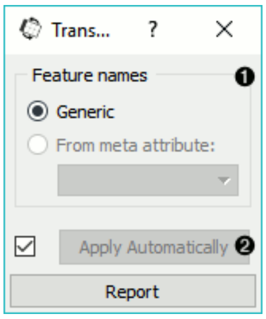
Outputs:

- **Data**

  Transposed data set

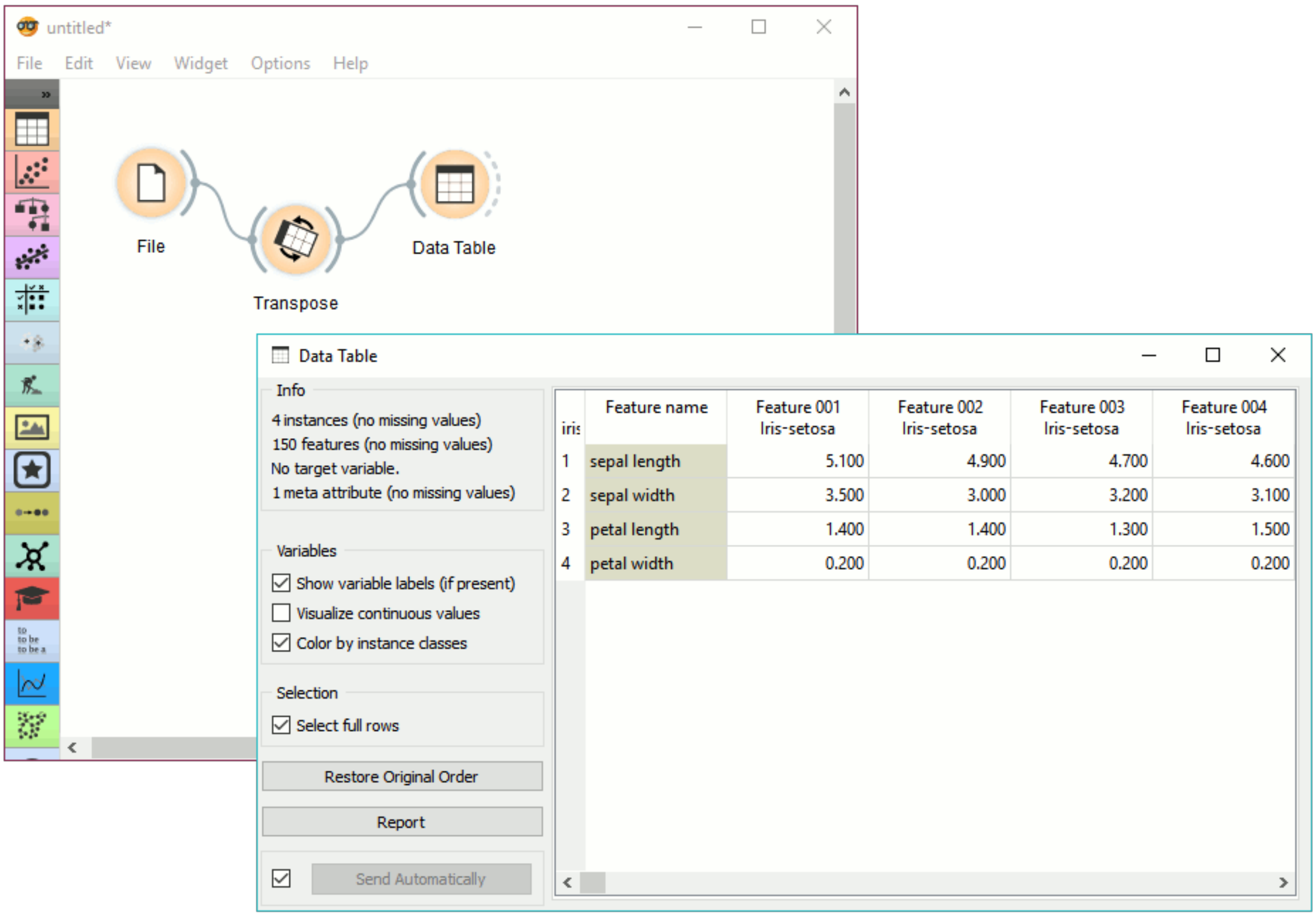
Description

Transpose widget transposes data table.



Example

This is a simple workflow showing how to use **Transpose**. Connect the widget to File widget. The output of **Transpose** is a transposed data table with rows as columns and columns as rows. You can observe the result in a Data Table.



Data Table

Info

4 instances (no missing values)
150 features (no missing values)
No target variable.
1 meta attribute (no missing values)

Variables

☑ Show variable labels (if present)
☐ Visualize continuous values
☑ Color by instance classes

Selection

☑ Select full rows

Restore Original Order

Report

☑ Send Automatically

iris	Feature name	Feature 001 Iris-setosa	Feature 002 Iris-setosa	Feature 003 Iris-setosa	Feature 004 Iris-setosa
1	sepal length	5.100	4.900	4.700	4.600
2	sepal width	3.500	3.000	3.200	3.100
3	petal length	1.400	1.400	1.300	1.500
4	petal width	0.200	0.200	0.200	0.200

Visualize

Box Plot	Distributions	Heat Map	Scatter Plot	Venn Diagram	Linear Projection	Scatter Map
Sieve Diagram	Pythagorean Tree	Pythagorean Forest	CN2 Rule Viewer	Mosaic Display	Silhouette Plot	Tree Viewer
Geo Map	Nomogram	FreeViz				

Box Plot

Shows distribution of attribute values.

Signals

Inputs:

- **Data**

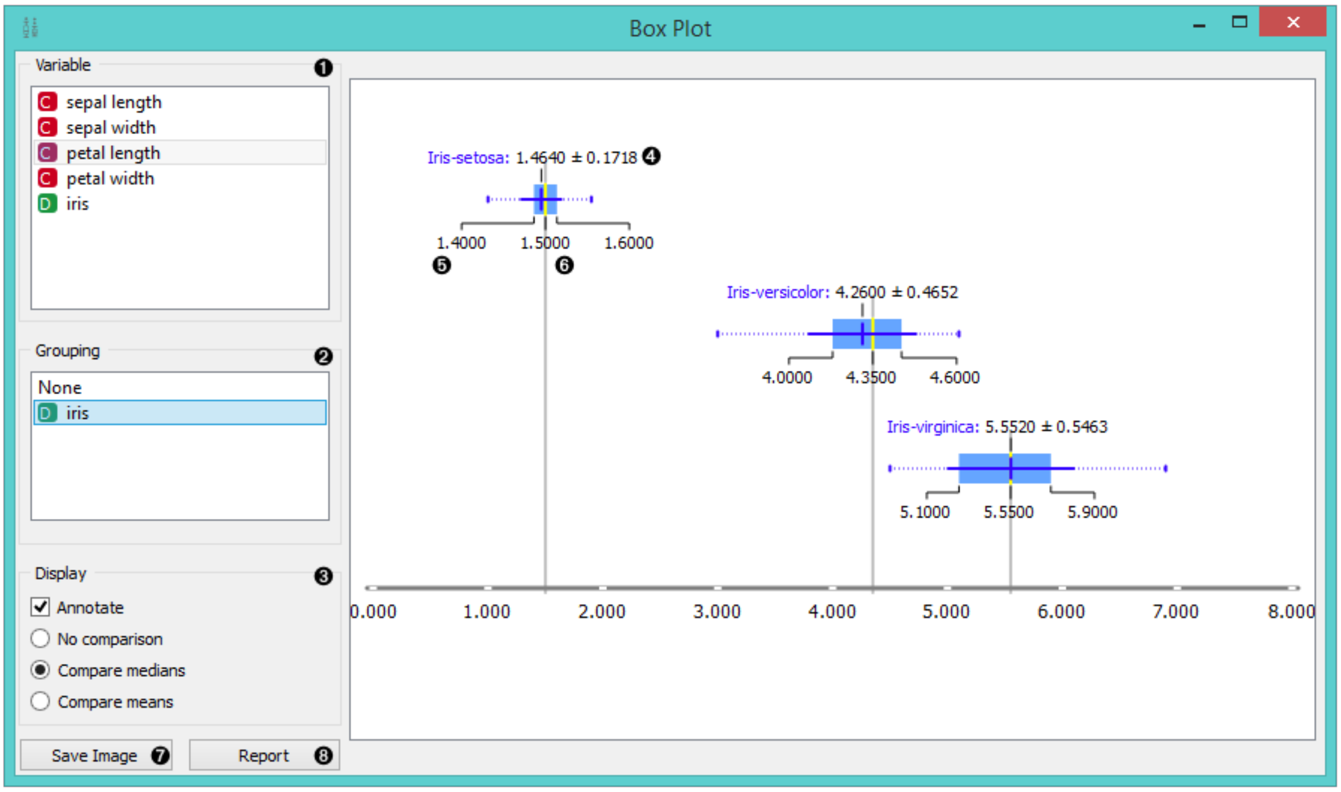
  An input data set

Outputs:

- (None)

Description

The **Box Plot** widget shows the distributions of attribute values. It is a good practice to check any new data with this widget to quickly discover any anomalies, such as duplicated values (e.g. gray and grey), outliers, and alike.



Box Plot

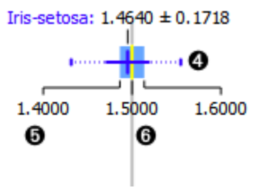
Variable ❶
- sepal length
- sepal width
- petal length
- petal width
- iris

Grouping ❷
- None
- iris

Display ❸
- [✓] Annotate
- ○ No comparison
- ● Compare medians
- ○ Compare means

Save Image ❼ Report ❽

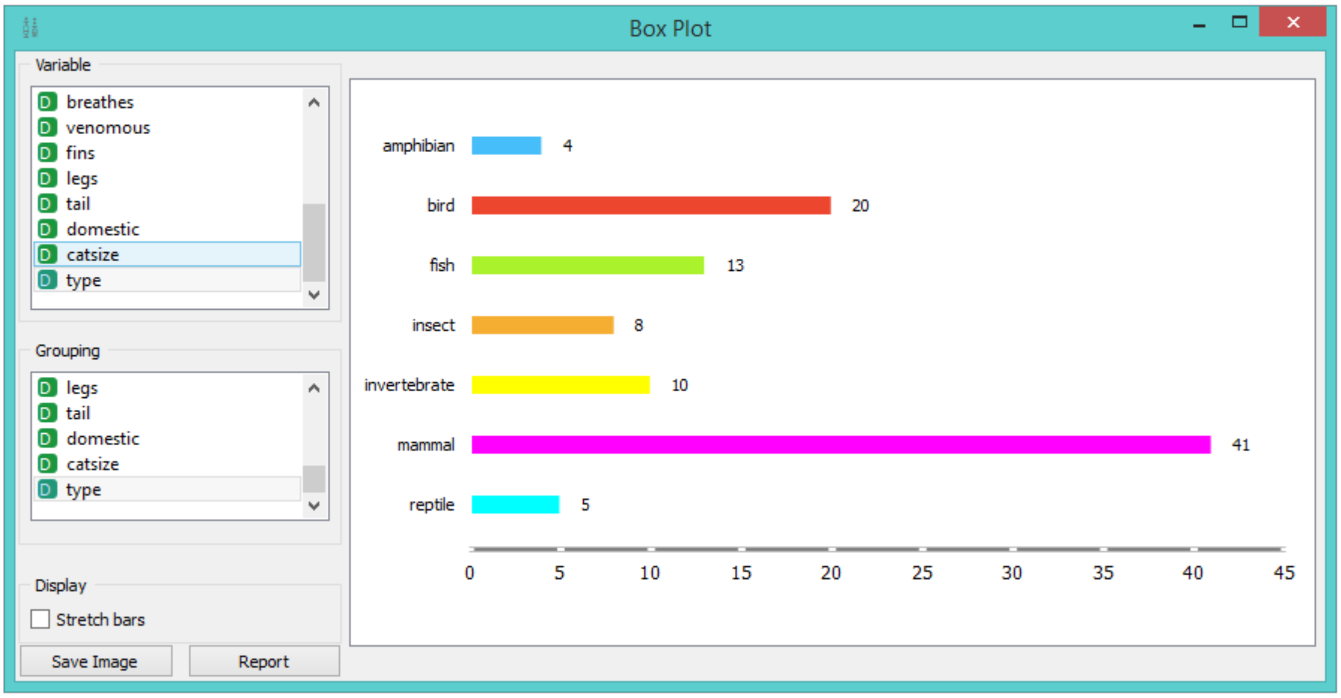
1. Select the variable you want to see plotted.
2. Choose *Grouping* to see box plots displayed by class.
3. When instances are grouped by class, you can change the display mode. Annotated boxes will display the end values, the mean and the median, while compare medians and compare means will, naturally, compare the selected value between class groups.



For continuous attributes the widget displays:

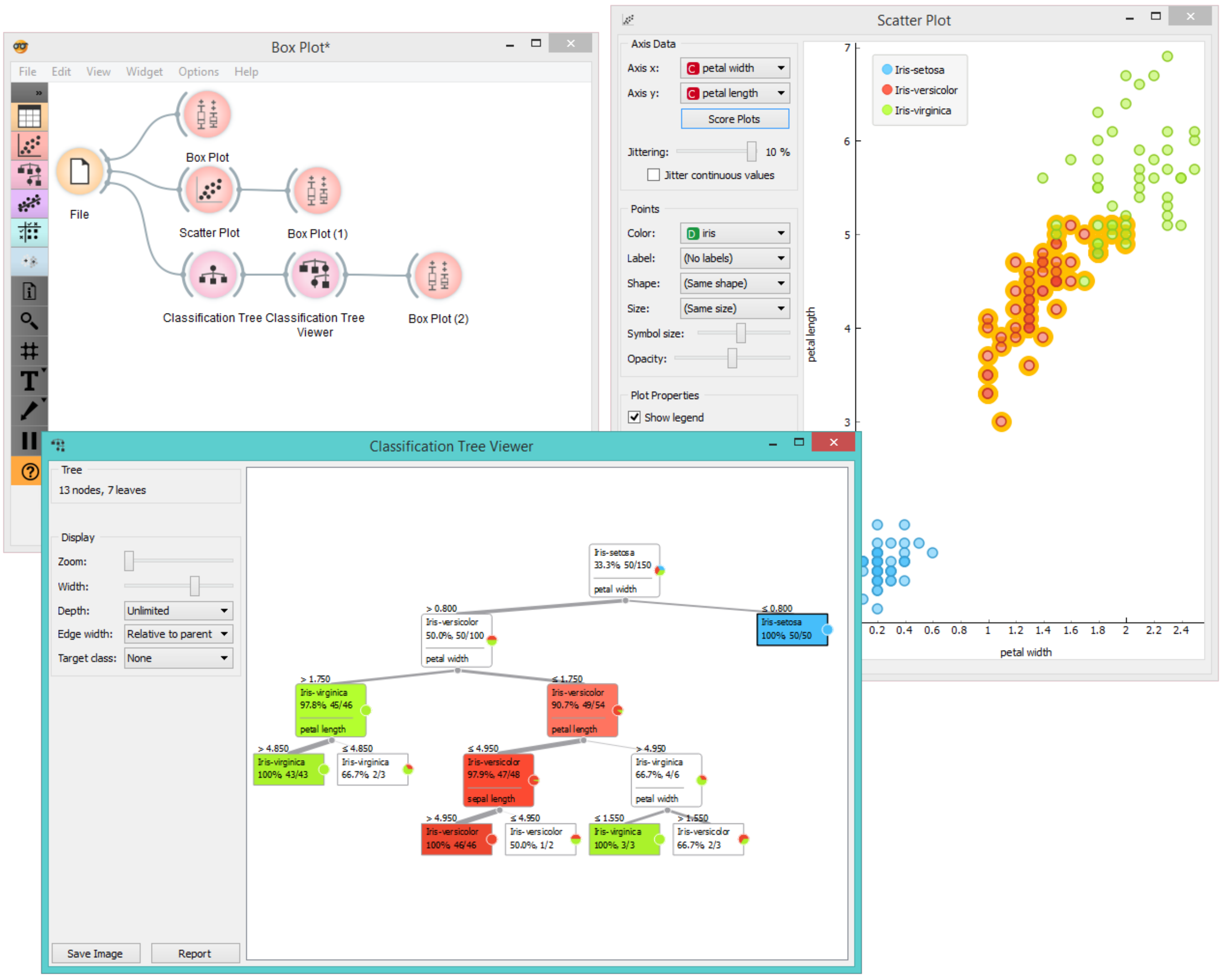
4. The mean (the dark blue vertical line)
5. Border values for the standard deviation of the mean. The blue highlighted area is the entire standard deviation of the mean.
6. The median (yellow vertical line). The thin blue line represents the area between the first (25%) and the third (75%) quantile, while the thin dotted line represents the entire range of values (from the lowest to the highest value in the data set for the selected parameter).
7. Save image.
8. Produce a report.

For discrete attributes, the bars represent the number of instances with each particular attribute value. The plot shows the number of different animal types in the *Zoo* data set: there are 41 mammals, 13 fish, 20 birds and so on.



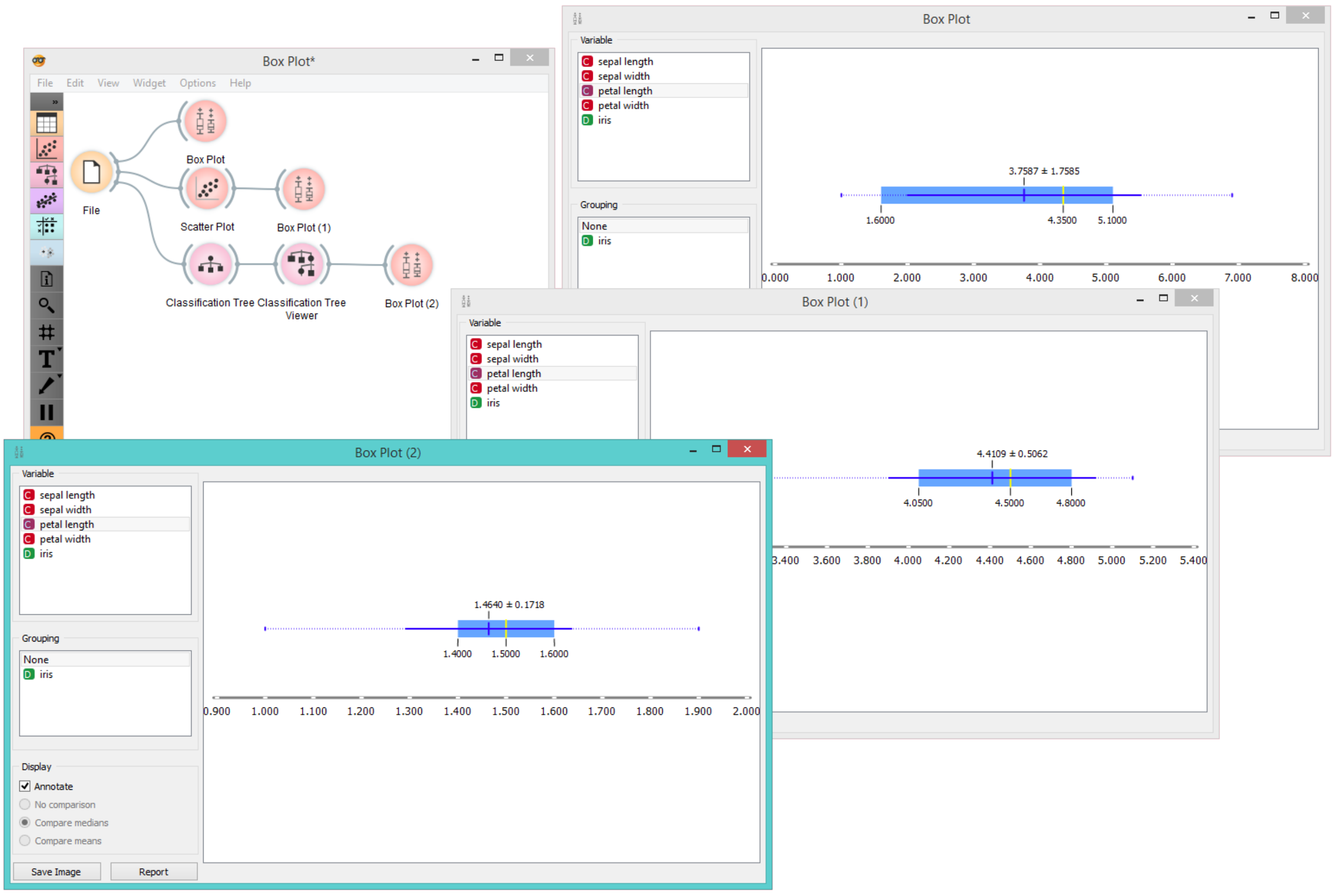
Example

The **Box Plot** widget is most commonly used immediately after the File widget to observe the statistical properties of a data set. It is also useful for finding the properties of a specific data set, for instance a set of instances manually defined in another widget (e.g. Scatterplot) or instances belonging to some cluster or a classification tree node, as shown in the schema below.



Axis Data
Axis x: petal width
Axis y: petal length
Jittering: 10 %
Jitter continuous values
Points
Color: iris
Label: (No labels)
Shape: (Same shape)
Size: (Same size)
Symbol size:
Opacity:
Plot Properties
✓ Show legend

Tree
13 nodes, 7 leaves
Display
Zoom:
Width:
Depth: Unlimited
Edge width: Relative to parent
Target class: None



Variable
sepal length
sepal width
petal length
petal width
iris
Grouping
None
iris
Display
✓ Annotate
No comparison
[✓ Compare medians]
Compare means

CN2 Rule Viewer

CN2 Rule Viewer

Signals

Inputs:

- **Data**

  Data set to filter.

- **CN2 Rule Classifier**

  CN2 Rule Classifier, including a list of induced rules.

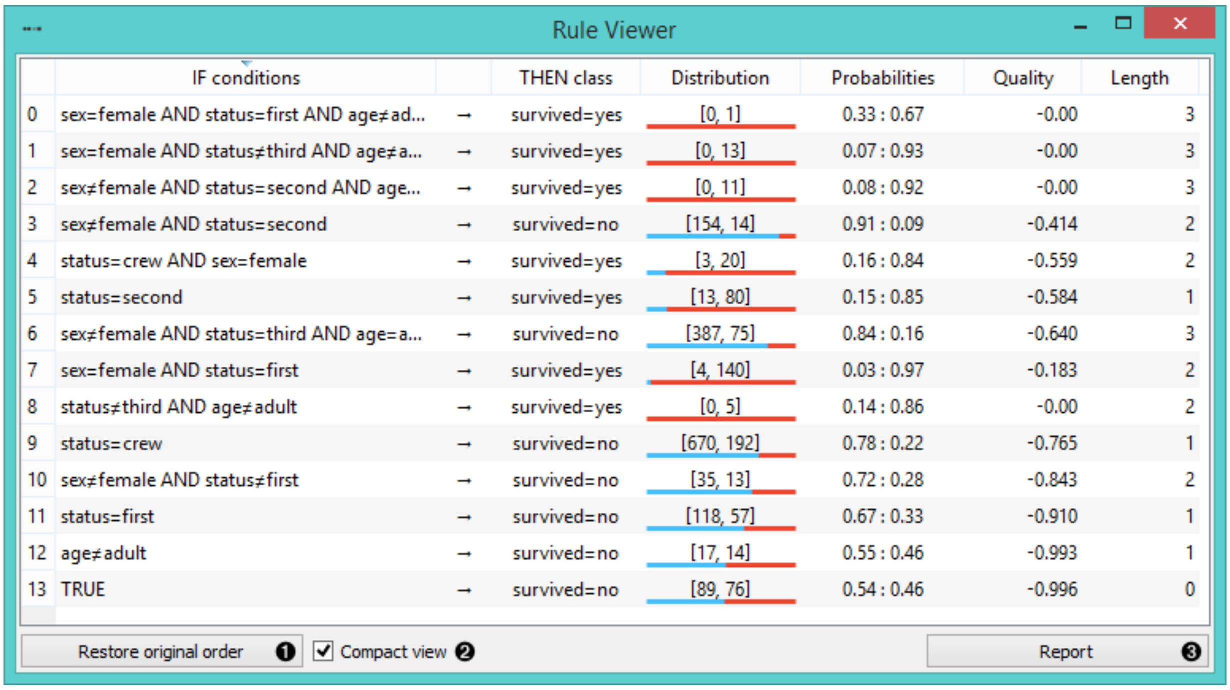
Outputs:

- **Filtered Data**

  If data is connected, upon active selection (at least one rule is selected), filtered data is emitted. Output are data instances covered by all selected rules.

Description

A widget that displays CN2 classification rules. If data is also connected, upon rule selection, one can analyze which instances abide to the conditions.



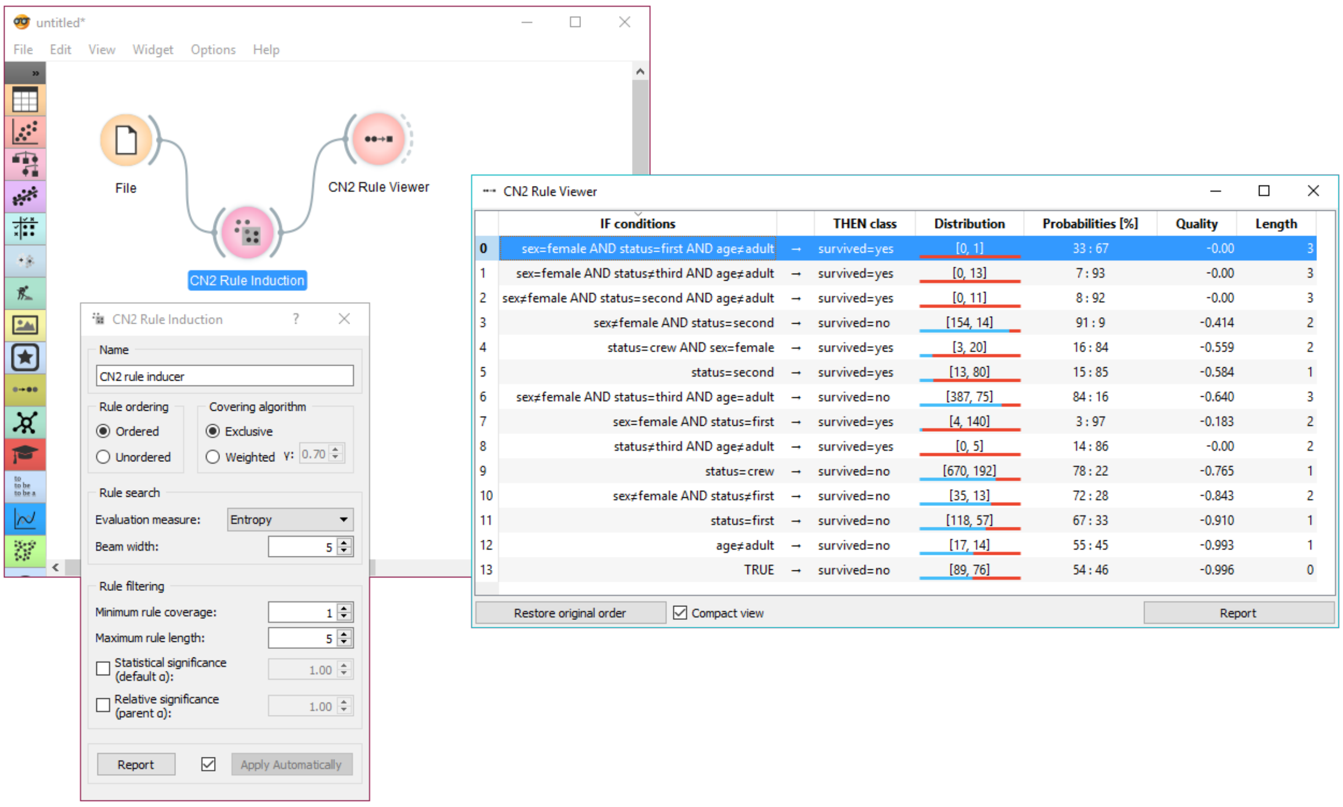
	IF conditions		THEN class	Distribution	Probabilities	Quality	Length
0	sex=female AND status=first AND age≠ad...	→	survived=yes	[0, 1]	0.33 : 0.67	-0.00	3
1	sex=female AND status≠third AND age≠a...	→	survived=yes	[0, 13]	0.07 : 0.93	-0.00	3
2	sex≠female AND status=second AND age...	→	survived=yes	[0, 11]	0.08 : 0.92	-0.00	3
3	sex≠female AND status=second	→	survived=no	[154, 14]	0.91 : 0.09	-0.414	2
4	status=crew AND sex=female	→	survived=yes	[3, 20]	0.16 : 0.84	-0.559	2
5	status=second	→	survived=yes	[13, 80]	0.15 : 0.85	-0.584	1
6	sex≠female AND status=third AND age=a...	→	survived=no	[387, 75]	0.84 : 0.16	-0.640	3
7	sex=female AND status=first	→	survived=yes	[4, 140]	0.03 : 0.97	-0.183	2
8	status≠third AND age≠adult	→	survived=yes	[0, 5]	0.14 : 0.86	-0.00	2
9	status=crew	→	survived=no	[670, 192]	0.78 : 0.22	-0.765	1
10	sex≠female AND status≠first	→	survived=no	[35, 13]	0.72 : 0.28	-0.843	2
11	status=first	→	survived=no	[118, 57]	0.67 : 0.33	-0.910	1
12	age≠adult	→	survived=no	[17, 14]	0.55 : 0.46	-0.993	1
13	TRUE	→	survived=no	[89, 76]	0.54 : 0.46	-0.996	0

1. Original order of induced rules can be restored.
2. When rules are many and complex, the view can appear packed. For this reason, *compact view* was implemented, which allows a flat presentation and a cleaner inspection of rules.
3. Click *Report* to bring up a detailed description of the rule induction algorithm and its parameters, the data domain, and induced rules.

Additionally, upon selection, rules can be copied to clipboard by pressing the default system shortcut (ctrl+C, cmd+C).

Examples

In the schema below, the most common use of the widget is presented. First, the data is read and a CN2 rule classifier is trained. We are using *titanic* data set for the rule constrution. The rules are then viewed using the Rule Viewer. To explore different CN2 algorithms and understand how adjusting parameters influences the learning process, **Rule Viewer** should be kept open and in sight, while setting the CN2 learning algorithm (the presentation will be updated promptly).



CN2 Rule Induction

Name: CN2 rule inducer

Rule ordering: [✓ Ordered] Unordered

Covering algorithm: [✓ Exclusive] Weighted γ: 0.70

Rule search

Evaluation measure: Entropy

Beam width: 5

Rule filtering

Minimum rule coverage: 1

Maximum rule length: 5

☐ Statistical significance (default α): 1.00

☐ Relative significance (parent α): 1.00

Report ☑ Apply Automatically

CN2 Rule Viewer

	IF conditions		THEN class	Distribution	Probabilities [%]	Quality	Length
0	sex=female AND status=first AND age≠adult	→	survived=yes	[0, 1]	33 : 67	-0.00	3
1	sex=female AND status≠third AND age≠adult	→	survived=yes	[0, 13]	7 : 93	-0.00	3
2	sex≠female AND status=second AND age≠adult	→	survived=yes	[0, 11]	8 : 92	-0.00	3
3	sex≠female AND status=second	→	survived=no	[154, 14]	91 : 9	-0.414	2
4	status=crew AND sex=female	→	survived=yes	[3, 20]	16 : 84	-0.559	2
5	status=second	→	survived=yes	[13, 80]	15 : 85	-0.584	1
6	sex≠female AND status=third AND age=adult	→	survived=no	[387, 75]	84 : 16	-0.640	3
7	sex=female AND status=first	→	survived=yes	[4, 140]	3 : 97	-0.183	2
8	status≠third AND age≠adult	→	survived=yes	[0, 5]	14 : 86	-0.00	2
9	status=crew	→	survived=no	[670, 192]	78 : 22	-0.765	1
10	sex≠female AND status≠first	→	survived=no	[35, 13]	72 : 28	-0.843	2
11	status=first	→	survived=no	[118, 57]	67 : 33	-0.910	1
12	age≠adult	→	survived=no	[17, 14]	55 : 45	-0.993	1
13	TRUE	→	survived=no	[89, 76]	54 : 46	-0.996	0

Restore original order ☑ Compact view Report

Selecting a rule outputs filtered data instances. These can be viewed in a Data Table.

Distributions

Displays value distributions for a single attribute.

Signals

Inputs:

- **Data**

  An input data set.

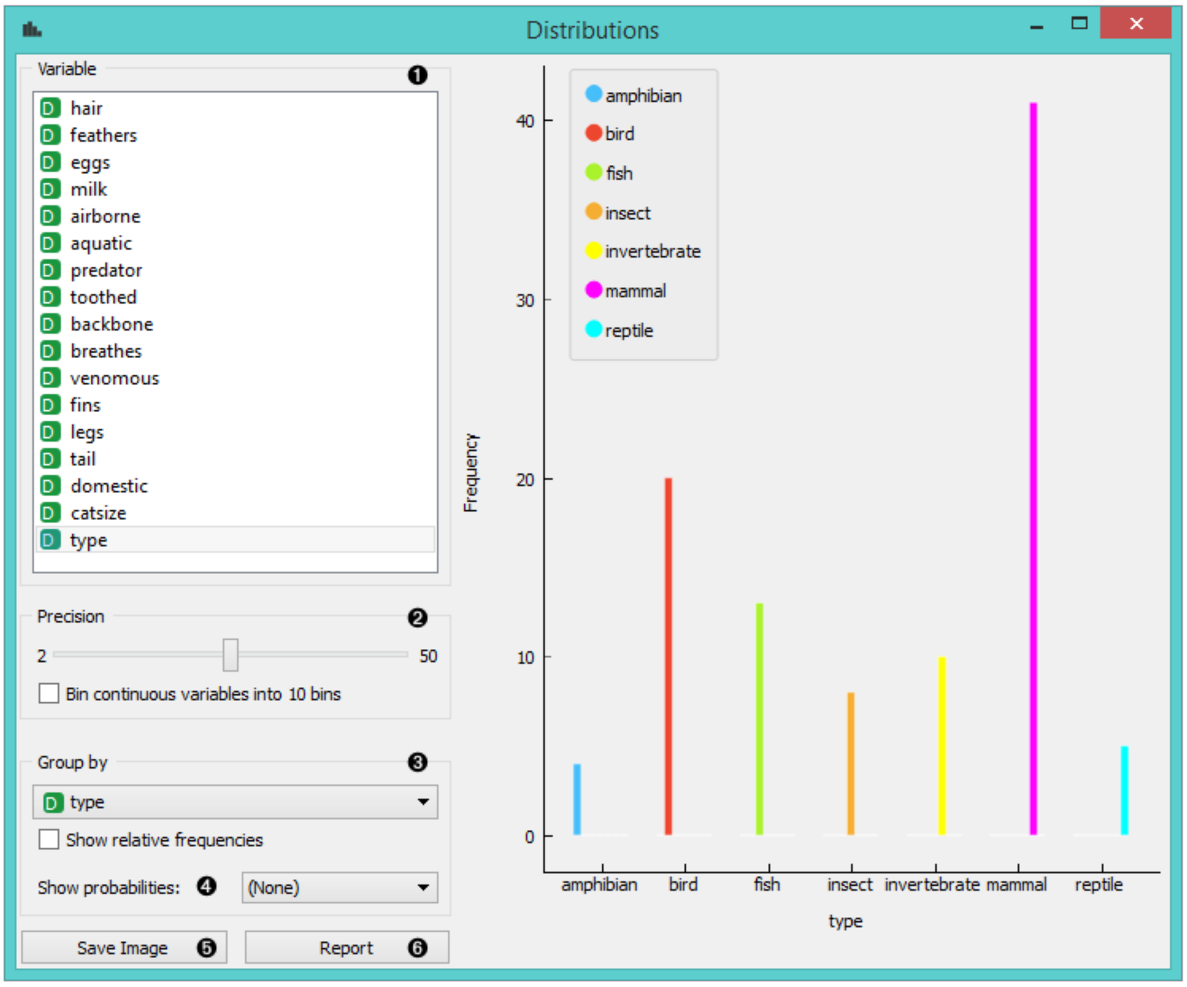
Outputs:

- (None)

Description

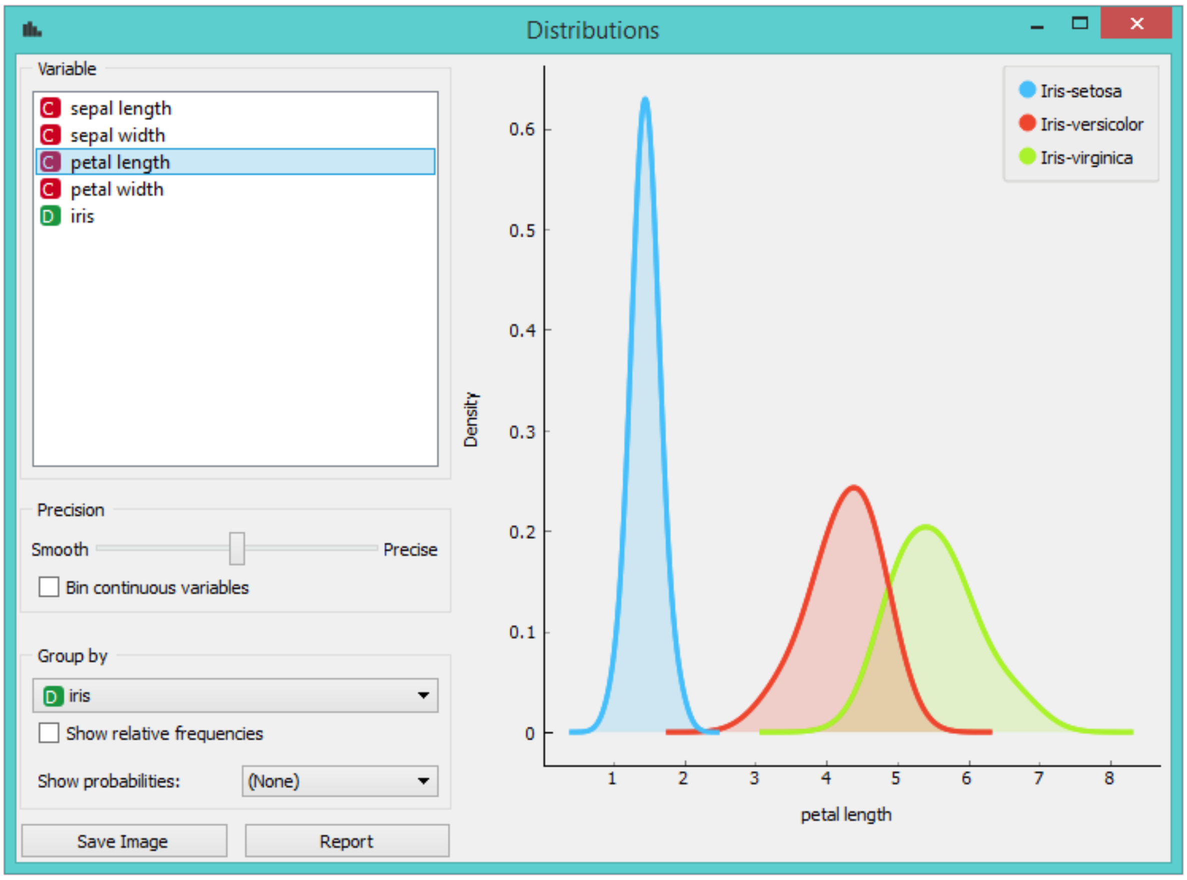
The **Distributions** widget displays the value distribution of discrete or continuous attributes. If the data contains a class variable, distributions may be conditioned on the class.

For discrete attributes, the graph displayed by the widget shows how many times (e.g., in how many instances) each attribute value appears in the data. If the data contains a class variable, class distributions for each of the attribute values will be displayed as well (like in the snapshot below). In order to create this graph, we used the *Zoo* data set.

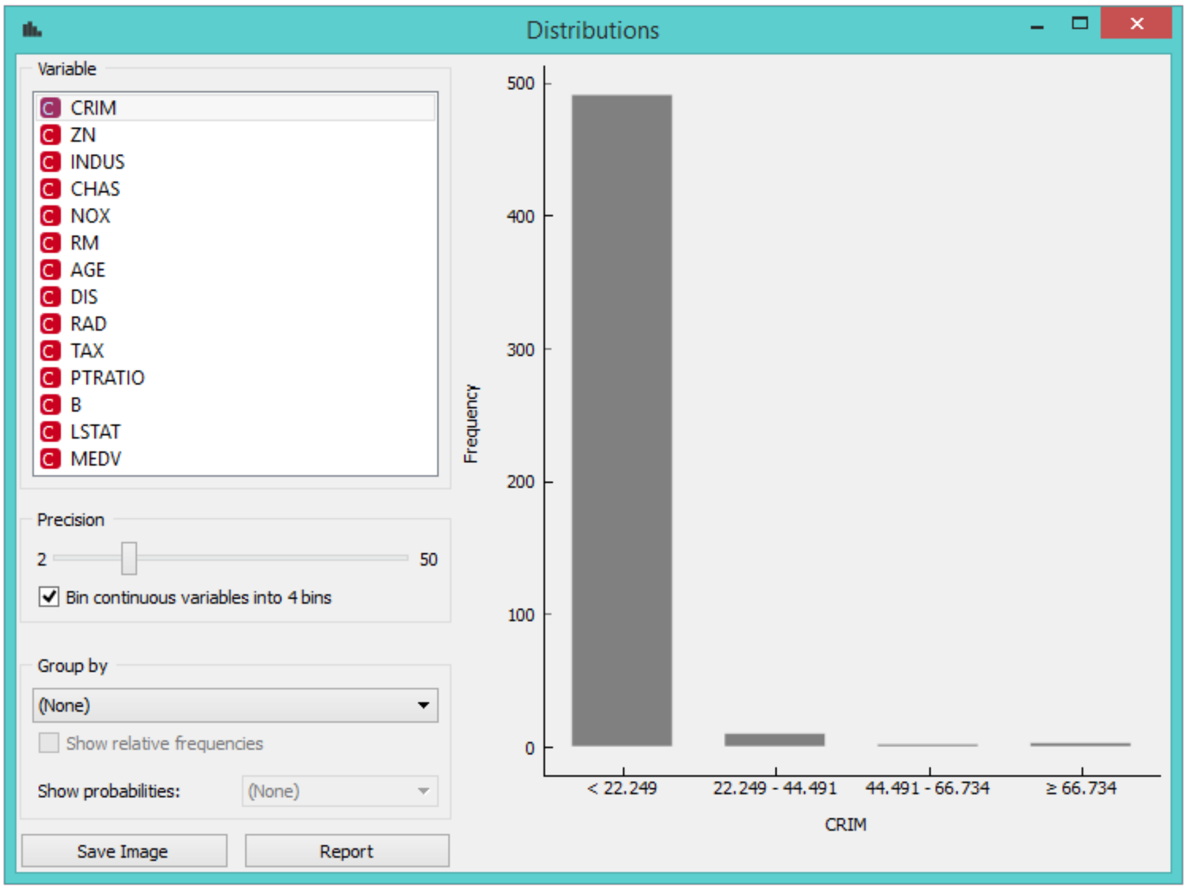


1. A list of variables for distributions display
2. If *Bin continuous variables* is ticked, the widget will discretize continuous variables by assigning them to intervals. The number of intervals is set by precision scale. Alternatively, you can set smoothness for the distribution curves of continuous variables.
3. The widget may be requested to display value distributions only for instances of certain class (*Group by*). *Show relative frequencies* will scale the data by percentage of the data set.
4. Show probabilities.
5. *Save image* saves the graph to your computer in a .svg or .png format.
6. Produce a report.

For continuous attributes, the attribute values are displayed as a function graph. Class probabilities for continuous attributes are obtained with gaussian kernel density estimation, while the appearance of the curve is set with the *Precision* bar (smooth or precise). For the purpose of this example, we used the *Iris* data set.



In class-less domains, the bars are displayed in gray. Here we set *Bin continuous variables into 10 bins*, which distributes variables into 10 intervals and displays averages of these intervals as histograms (see 2. above). We used the *Housing* data set.


Distributions
Variable
CRIM
ZN
INDUS
CHAS
NOX
RM
AGE
DIS
RAD
TAX
PTRATIO
B
LSTAT
MEDV
Precision
2
50
Bin continuous variables into 4 bins
Group by
(None)
Show relative frequencies
Show probabilities:
(None)
Save Image
Report
Frequency
500
400
300
200
100
0
< 22.249
22.249 - 44.491
44.491 - 66.734
≥ 66.734
CRIM

FreeViz

Displays FreeViz projection.

Signals

Inputs:

- **Data**

  An input data set.

- **Data Subset**

  A subset of instances from the input data set.

Outputs:

- **Selected Data**

  A subset of instances that the user manually selected from the freeviz plot.

- **Data**

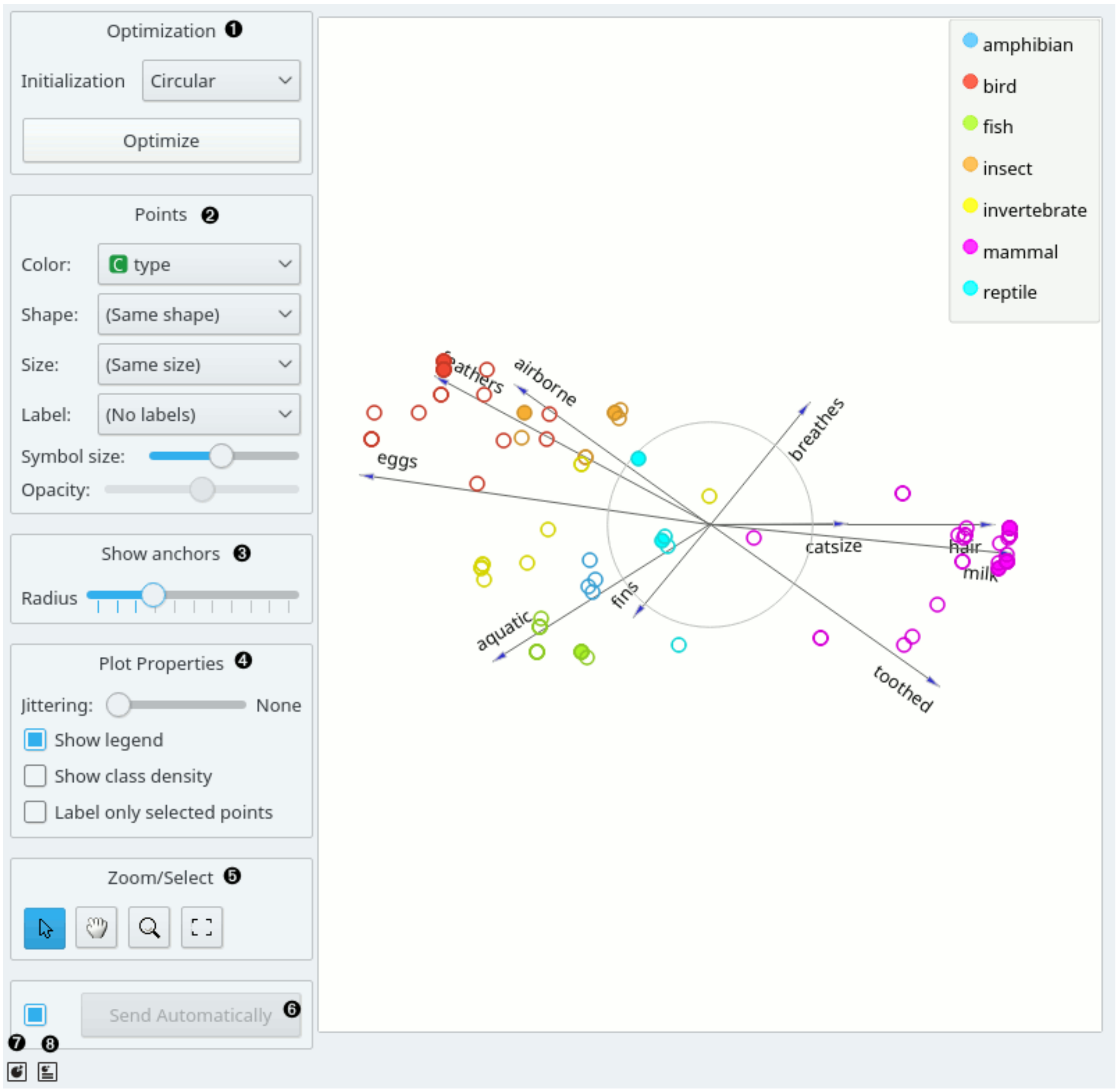
  Data with an additional column showing whether a point is selected. If more than one group is selected then also the group name is written instead.

- **Components**

  FreeViz vectors

Description

FreeViz uses a paradigm borrowed from particle physics: points in the same class attract each other, those from different class repel each other, and the resulting forces are exerted on the anchors of the attributes, that is, on unit vectors of each of the dimensional axis. The points cannot move (are projected in the projection space), but the attribute anchors can, so the optimization process is a hill-climbing optimization where at the end the anchors are placed such that forces are in equilibrium. The button Optimize is used to invoke the optimization process. The result of the optimization may depend on the initial placement of the anchors, which can be set in a circle, arbitrary or even manually. The later also works at any stage of optimization, and we recommend to play with this option in order to understand how a change of one anchor affects the positions of the data points. In any linear projection, projections of unit vector that are very short compared to the others indicate that their associated attribute is not very informative for particular classification task. Those vectors, that is, their corresponding anchors, may be hidden from the visualization using Radius slider in Show anchors box.



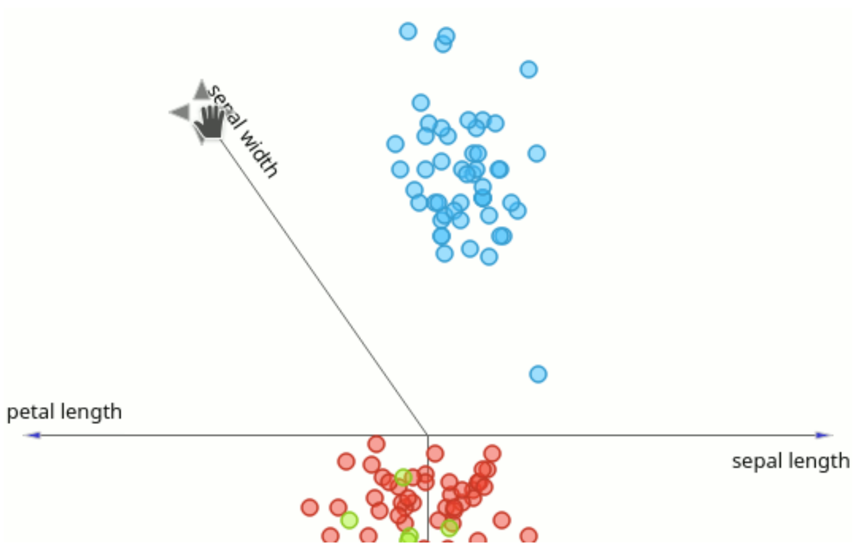
1. Two initial positions of anchors are possible: random and circular. Optimization moves anchors in an optimal position.
2. Set the color of the displayed points (you will get colors for discrete values and grey-scale points for continuous). Set label, shape and size to differentiate between points. Set symbol size and opacity for all data points.
3. Anchors inside a circle are hidden. Circle radius can be be changed using a slider.
4. Adjust *plot properties*:
   - Set jittering to prevent

   the dots from overlapping (especially for discrete attributes).

   - *Show legend* displays a legend on the right. Click and drag the legend to move it.
   - *Show class density* colors the graph by class (see the screenshot below).
   - *Label only selected points* allows you to select individual data instances and label them.
5. *Select, zoom, pan and zoom to fit* are the options for exploring the graph. The manual selection of data instances works as an angular/square selection tool. Double click to move the projection. Scroll in or out for zoom.
6. If *Send automatically* is ticked, changes are communicated automatically. Alternatively, press *Send*.
7. *Save Image* saves the created image to your computer in a .svg or .png format.

8. Produce a report.

Manually move anchors

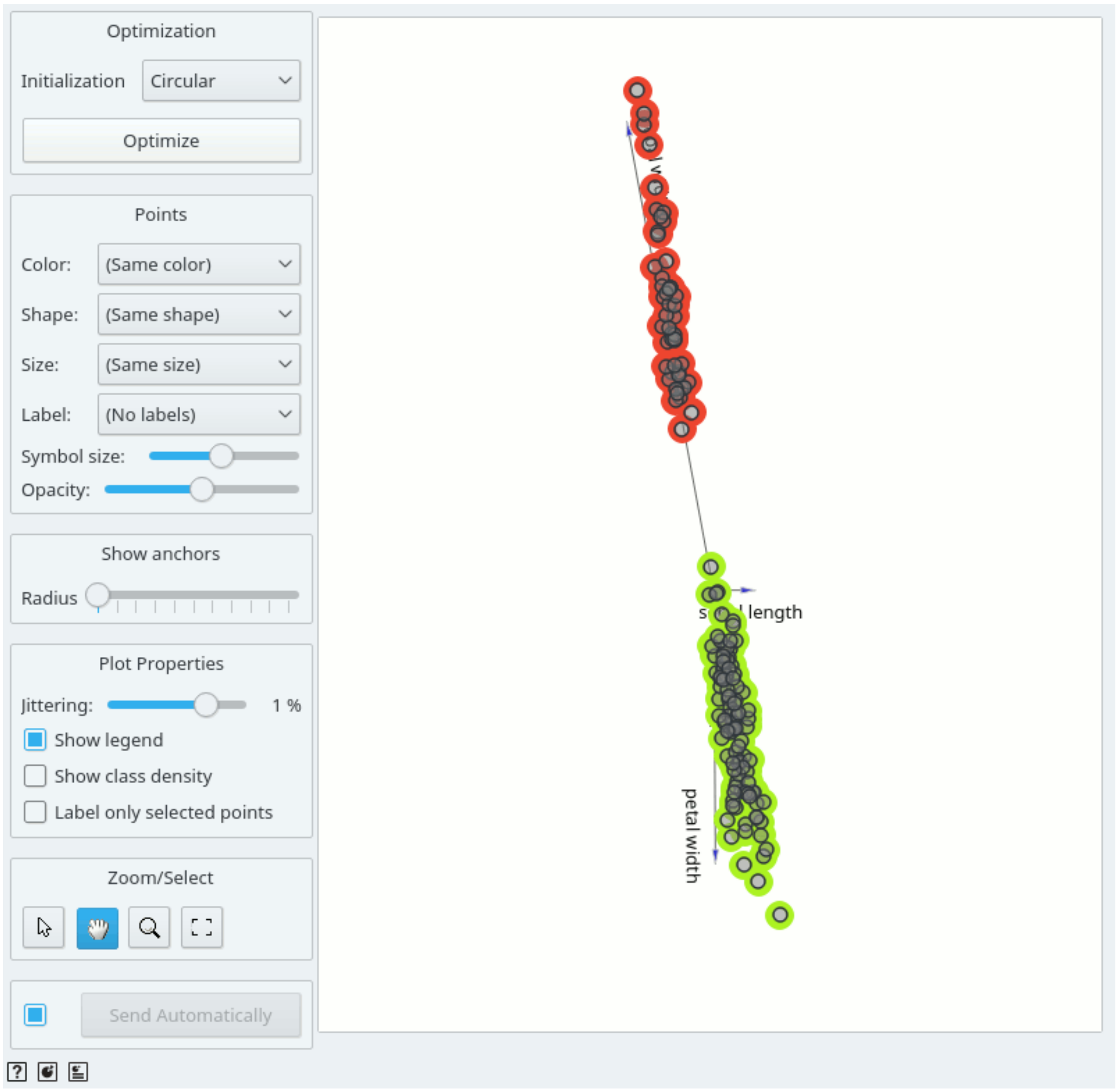


One can manually move anchors. Use a mouse pointer and hover above the end of an anchor. Click the left button and then you can move selected anchor where ever you want.

Selection

Selection can be used to manually defined subgroups in the data. Use Shift modifier when selecting data instances to put them into a new group. Shift + Ctrl (or Shift + Cmd on macOs) appends instances to the last group.

Signal data outputs a data table with an additional column that contains group indices.



Optimization

Initialization: Circular

Optimize

Points

Color: (Same color)

Shape: (Same shape)

Size: (Same size)

Label: (No labels)

Symbol size:

Opacity:

Show anchors

Radius

Plot Properties

Jittering: 1 %

☑ Show legend

☐ Show class density

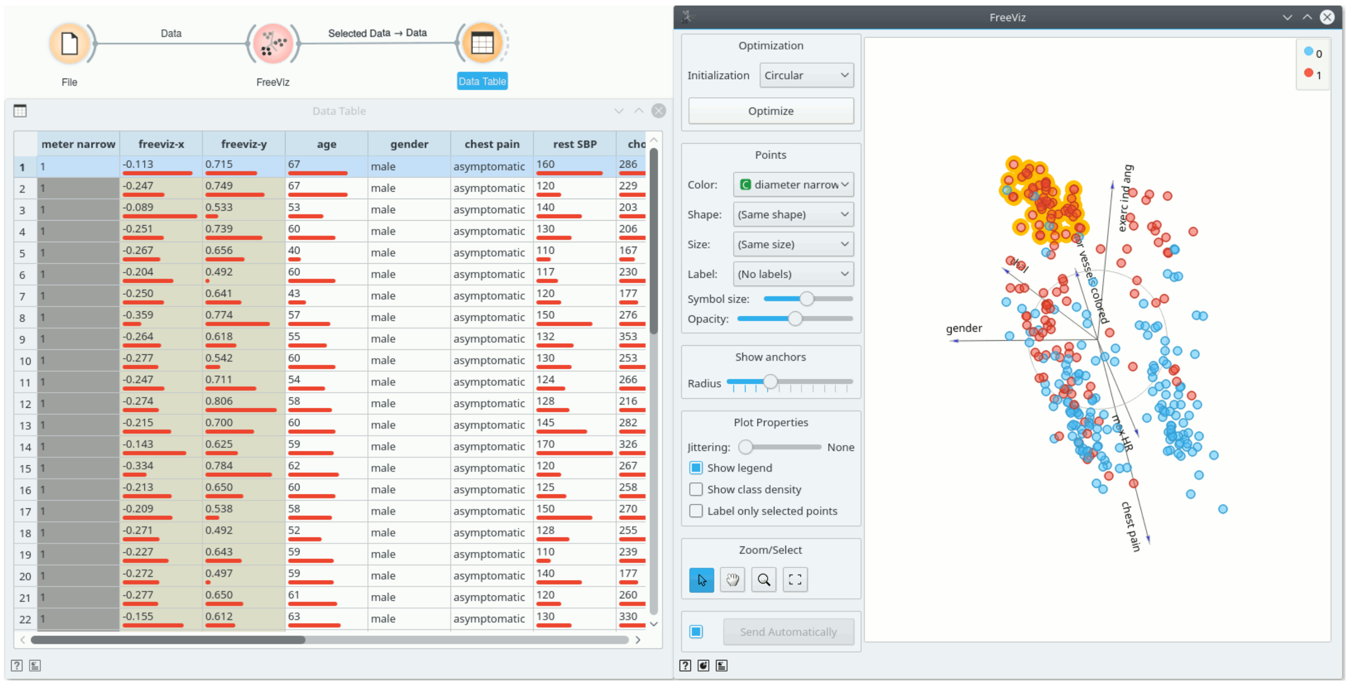
☐ Label only selected points

Zoom/Select

☑ Send Automatically

Explorative Data Analysis

The **FreeViz**, as the rest of Orange widgets, supports zooming-in and out of part of the plot and a manual selection of data instances. These functions are available in the lower left corner of the widget. The default tool is *Select*, which selects data instances within the chosen rectangular area. *Pan* enables you to move the plot around the pane. With *Zoom* you can zoom in and out of the pane with a mouse scroll, while *Reset zoom* resets the visualization to its optimal size. An example of a simple schema, where we selected data instances from a rectangular region and sent them to the Data Table widget, is shown below.



Data Table

	meter narrow	freeviz-x	freeviz-y	age	gender	chest pain	rest SBP	cho
1	1	-0.113	0.715	67	male	asymptomatic	160	286
2	1	-0.247	0.749	67	male	asymptomatic	120	229
3	1	-0.089	0.533	53	male	asymptomatic	140	203
4	1	-0.251	0.739	60	male	asymptomatic	130	206
5	1	-0.267	0.656	40	male	asymptomatic	110	167
6	1	-0.204	0.492	60	male	asymptomatic	117	230
7	1	-0.250	0.641	43	male	asymptomatic	120	177
8	1	-0.359	0.774	57	male	asymptomatic	150	276
9	1	-0.264	0.618	55	male	asymptomatic	132	353
10	1	-0.277	0.542	60	male	asymptomatic	130	253
11	1	-0.247	0.711	54	male	asymptomatic	124	266
12	1	-0.274	0.806	58	male	asymptomatic	128	216
13	1	-0.215	0.700	60	male	asymptomatic	145	282
14	1	-0.143	0.625	59	male	asymptomatic	170	326
15	1	-0.334	0.784	62	male	asymptomatic	120	267
16	1	-0.213	0.650	60	male	asymptomatic	125	258
17	1	-0.209	0.538	58	male	asymptomatic	150	270
18	1	-0.271	0.492	52	male	asymptomatic	128	255
19	1	-0.227	0.643	59	male	asymptomatic	110	239
20	1	-0.272	0.497	59	male	asymptomatic	140	177
21	1	-0.277	0.650	61	male	asymptomatic	120	260
22	1	-0.155	0.612	63	male	asymptomatic	130	330

FreeViz

Optimization
Initialization: Circular
Optimize

Points
Color: diameter narrow
Shape: (Same shape)
Size: (Same size)
Label: (No labels)
Symbol size:
Opacity:

Show anchors
Radius

Plot Properties
Jittering: None
[✓] Show legend
[ ] Show class density
[ ] Label only selected points

Zoom/Select

[✓] Send Automatically

Geo Map

Show data points on a world map.

Signals

Inputs:

- **Data**

  An input data set.

- **Data Subset**

  A subset of instances from the input data set.

- **Learner**

  A learning algorithm (classification or regression).

Outputs:

- **Selected Data**

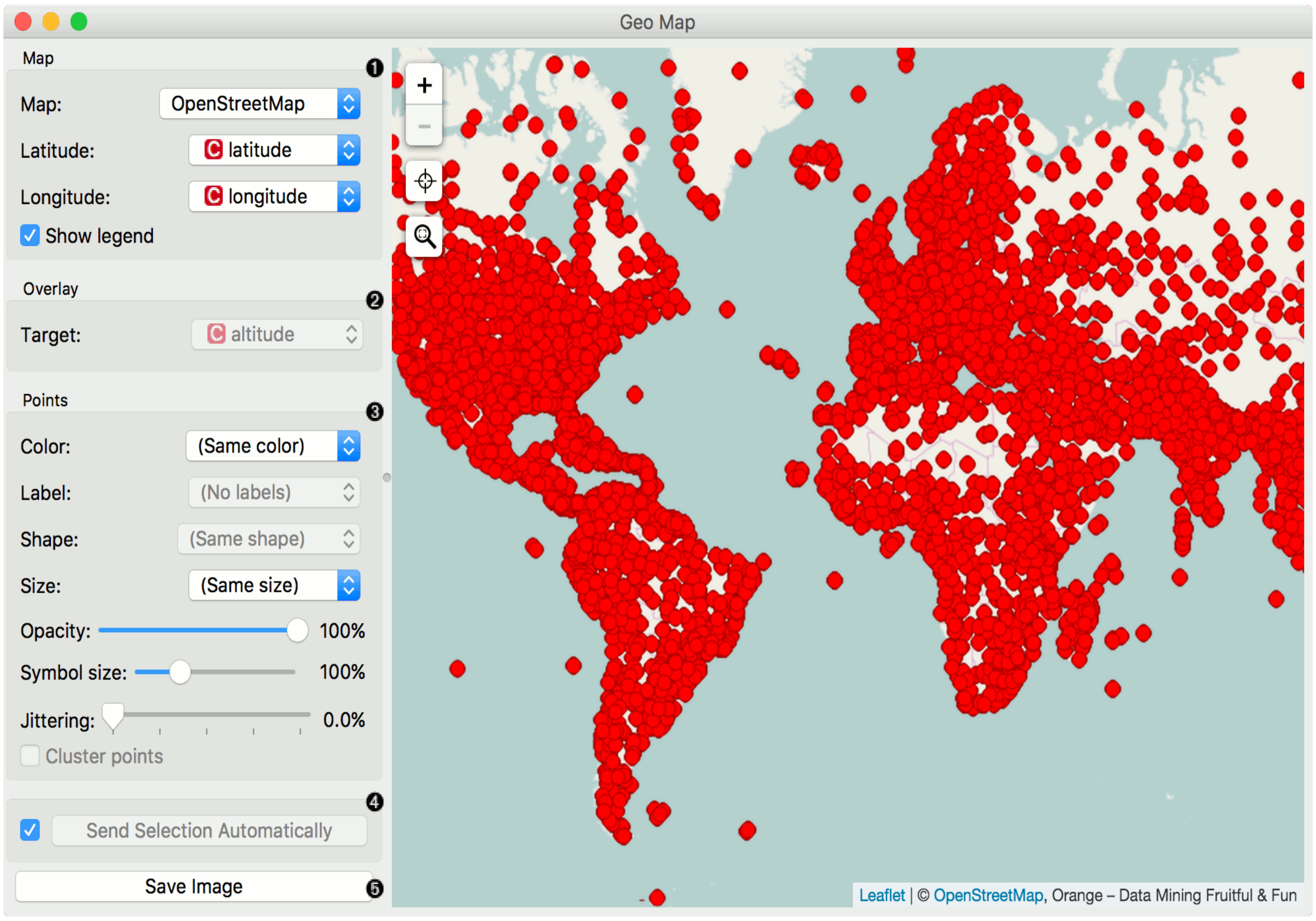
  A subset of instances that the user has manually selected from the map.

- **Data**

  Data set with an appended meta attribute specifying selected and unselected data.

Description

Geo Map widget maps geo-spatial data on a world map. It only works on data sets containing latitude and longitude variables. It also enables class predictions when a learner is provided on the input.



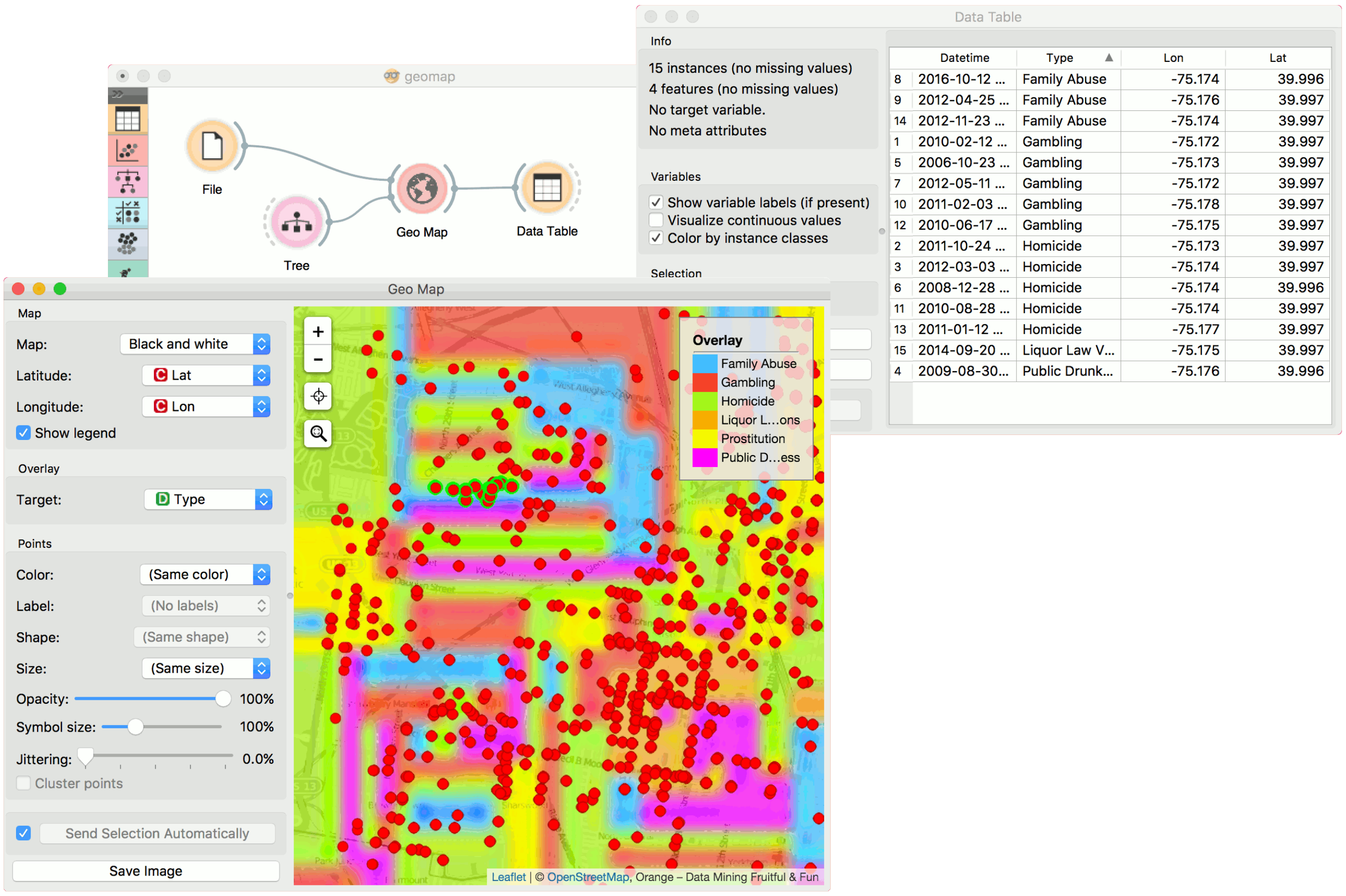
1. Define map properties: - Set the type of *map*: Black and White, OpenStreetMap, Topographic, Satellite, Print, Light, Dark, Railyways and Watercolor. - Set latitude and longitude attributes, if the widget didn't recognize them automatically. Latitude values should be between -90(S) and 90(N) and longitude values between -180(W) and 180(E).
2. Overlay: - Set the target (class) for predictive mapping. A learner has to be provided on the input. The classifier is trained on latitude and longitude pairs only (i.e. it maps lat/lon pairs to the selected attribute).
3. Set point parameters: - *Color*: color of data points by attribute values - *Label*: label data points with an attribute (available when zoomed in) - *Shape*: shape of data points by attribute (available when zoomed in) - *Size*: size of data points by attribute - Opacity: set transparency of data points - Symbol size: size of data points (small to large) - Jittering: disperse overlaid data points - Cluster points: cluster neighboring points with naive greedy clustering (available when less than 600 points are in view)
4. If *Send Selection Automatically* is ticked, changes are communicated automatically. Alternatively, click *Send Selection*. *Save image* saves the image to your computer in a .svg or .png format.

Note:

To select a subset of points from the map, hold Shift and draw a rectangle around the point you want to output.

Examples

In the first example we will model class predictions on a map. We will use *philadelphia-crime* data set, load it with File widget and connect it to **Map**. We can already observe the mapped points in Map. Now, we connect Tree to Map and set target variable to Type. This will display the predicted type of crime for a specific region of Philadelphia city (each region will be colored with a corresponding color code, explained in a legend on the right).



Data Table

Info

15 instances (no missing values)
4 features (no missing values)
No target variable.
No meta attributes

Variables

✓ Show variable labels (if present)
Visualize continuous values
✓ Color by instance classes

Selection

	Datetime	Type ▲	Lon	Lat
8	2016-10-12 ...	Family Abuse	-75.174	39.996
9	2012-04-25 ...	Family Abuse	-75.176	39.997
14	2012-11-23 ...	Family Abuse	-75.174	39.997
1	2010-02-12 ...	Gambling	-75.172	39.997
5	2006-10-23 ...	Gambling	-75.173	39.997
7	2012-05-11 ...	Gambling	-75.172	39.997
10	2011-02-03 ...	Gambling	-75.178	39.997
12	2010-06-17 ...	Gambling	-75.175	39.997
2	2011-10-24 ...	Homicide	-75.173	39.997
3	2012-03-03 ...	Homicide	-75.174	39.997
6	2008-12-28 ...	Homicide	-75.174	39.996
11	2010-08-28 ...	Homicide	-75.174	39.997
13	2011-01-12 ...	Homicide	-75.177	39.997
15	2014-09-20 ...	Liquor Law V...	-75.175	39.997
4	2009-08-30...	Public Drunk...	-75.176	39.996

Geo Map

Map

Map: Black and white
Latitude: Lat
Longitude: Lon
✓ Show legend

Overlay

Target: Type

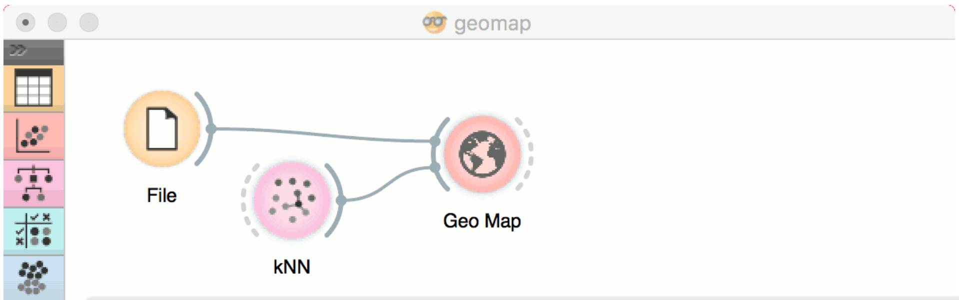
Points

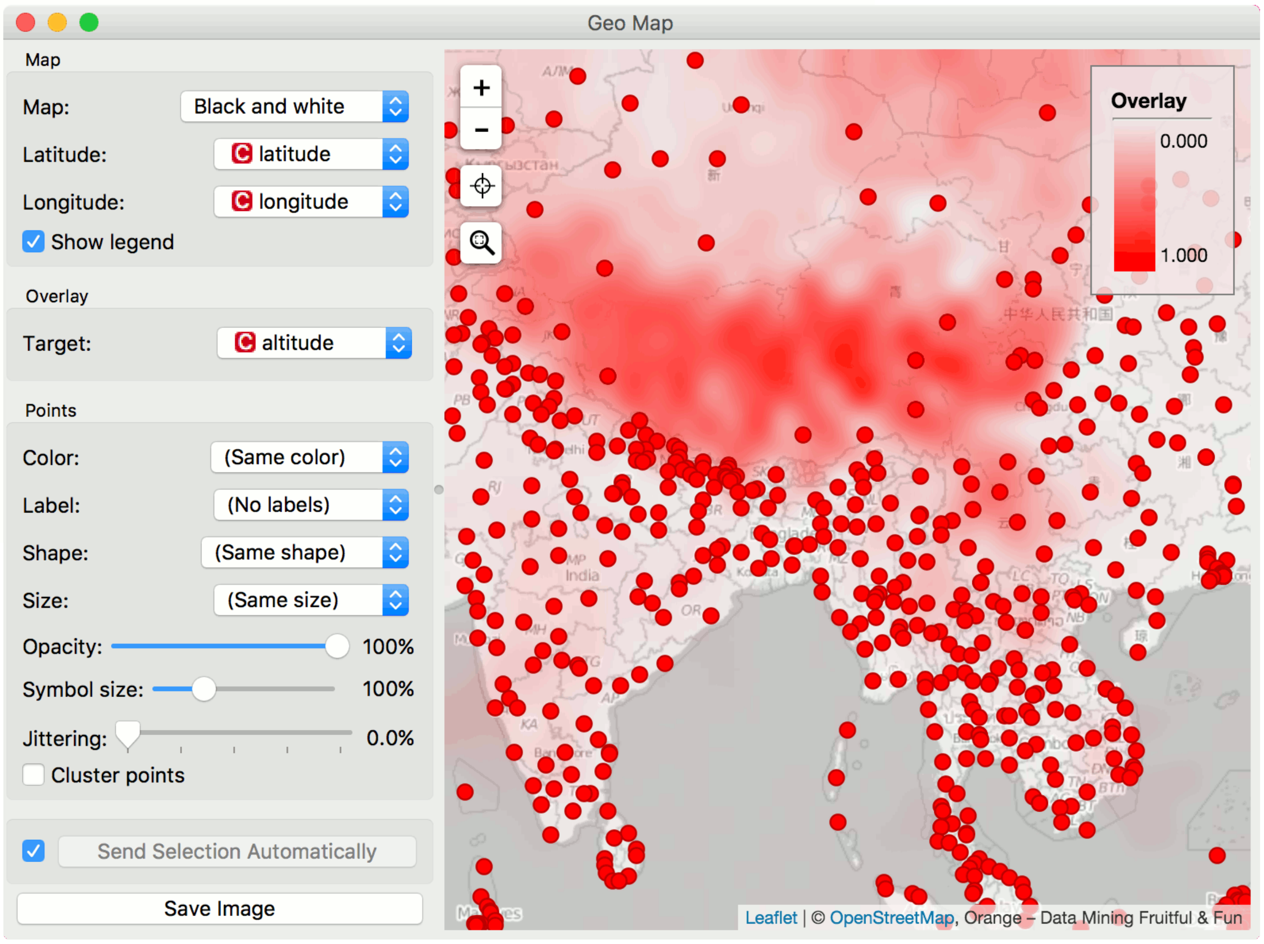
Color: (Same color)
Label: (No labels)
Shape: (Same shape)
Size: (Same size)
Opacity: 100%
Symbol size: 100%
Jittering: 0.0%
Cluster points

✓ Send Selection Automatically

Save Image

The second example uses global-airports.csv data. Say we somehow want to predict the altitude of the area based solely on the latitude and longitude. We again load the data with File widget and connect it to Map. Then we use a regressor, say, KNN and connect it to Map as well. Now we set target to altitude and use Black and White map type. The model guessed the Himalaya, but mades some errors elsewhere.


geomap
File
kNN
Geo Map


Geo Map
Map
Map: Black and white
Latitude: latitude
Longitude: longitude
Show legend
Overlay
Target: altitude
Points
Color: (Same color)
Label: (No labels)
Shape: (Same shape)
Size: (Same size)
Opacity: 100%
Symbol size: 100%
Jittering: 0.0%
Cluster points
Send Selection Automatically
Save Image
Overlay
0.000
1.000
Leaflet | © OpenStreetMap, Orange – Data Mining Fruitful & Fun

Heat Map

Plots a heat map for a pair of attributes.

Signals

Inputs:

- **Data**

  An input data set.

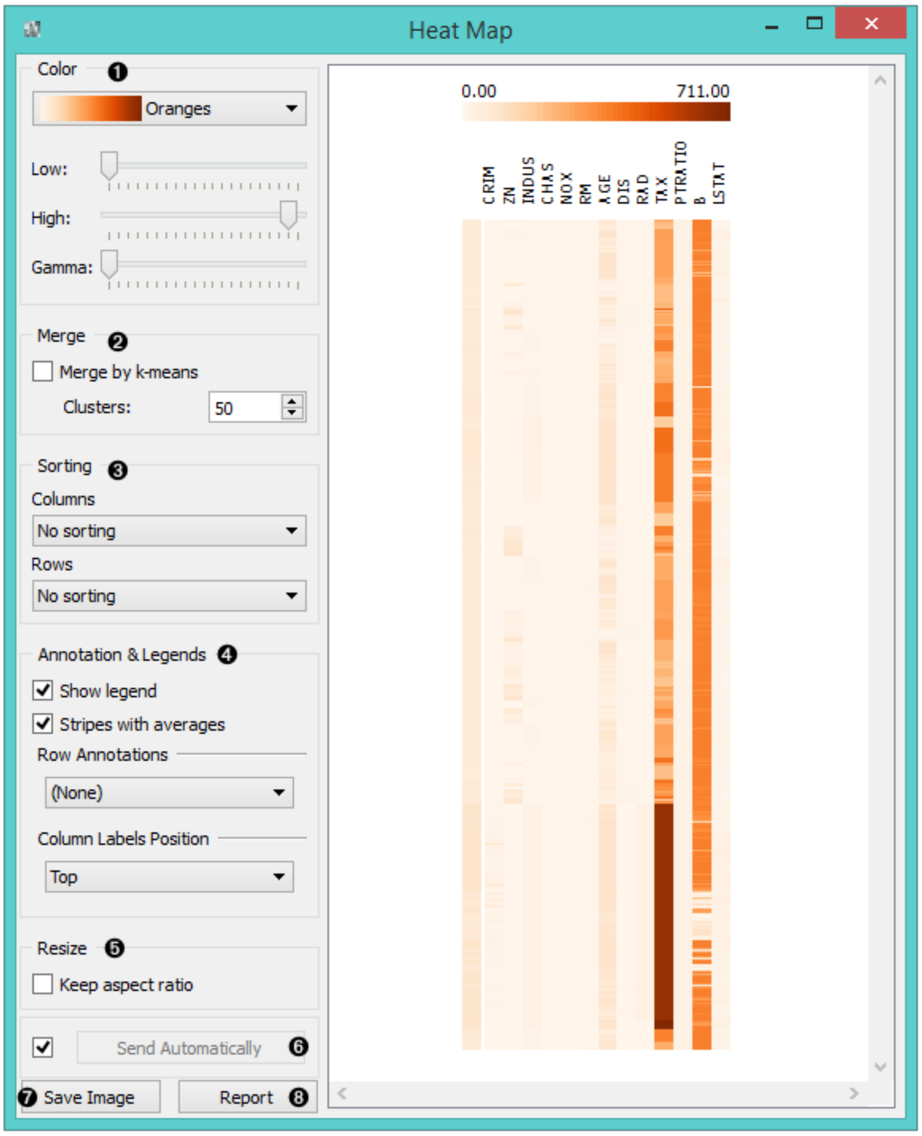
Outputs:

- **Selected Data**

  A subset of instances that the user has manually selected from the map.

Description

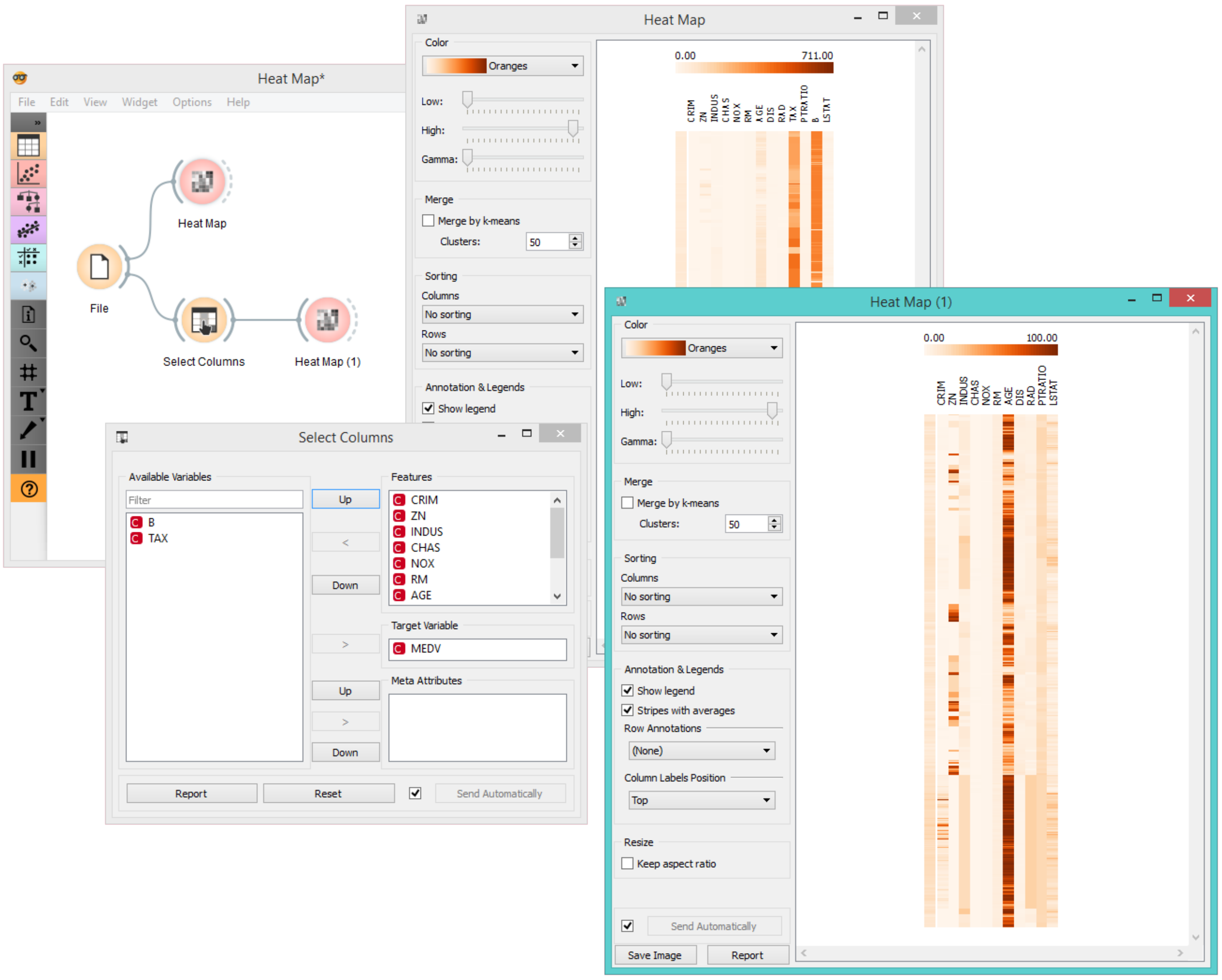
Heat map is a graphical method for visualizing attribute values by class in a two-way matrix. It only works on data sets containing continuous variables. The values are represented by color: the higher a certain value is, the darker the represented color. By combining class and attributes on x and y axes, we see where the attribute values are the strongest and where the weakest, thus enabling us to find typical features (discrete) or value range (continuous) for each class.



1. The color scheme legend. **Low** and **High** are thresholds for the color palette (low for attributes with low values and high for attributes with high values).
2. Merge data.
3. Sort columns and rows: - **No Sorting** (lists attributes as found in the data set) - **Clustering** (clusters data by similarity) - **Clustering with ordered leaves** (maximizes the sum of similarities of adjacent elements)
4. Set what is displayed in the plot in **Annotation & Legend**. - If *Show legend* is ticked, a color chart will be displayed above the map. - If *Stripes with averages* is ticked, a new line with attribute averages will be displayed on the left. - **Row Annotations** adds annotations to each instance on the right. - **Column Label Positions** places column labels in a selected place (None, Top, Bottom, Top and Bottom).
5. If *Keep aspect ratio* is ticked, each value will be displayed with a square (proportionate to the map).
6. If *Send Automatically* is ticked, changes are communicated automatically. Alternatively, click *Send*.
7. *Save image* saves the image to your computer in a .svg or .png format.
8. Produce a report.

Example

The **Heat Map** below displays attribute values for the *Housing* data set. The aforementioned data set concerns the housing values in the suburbs of Boston. The first thing we see in the map are the 'B' and 'Tax' attributes, which are the only two colored in dark orange. The 'B' attribute provides information on the proportion of blacks by town and the 'Tax' attribute informs us about the full-value property-tax rate per $10,000. In order to get a clearer heat map, we then use the Select Columns widget and remove the two attributes from the data set. Then we again feed the data to the **Heat map**. The new projection offers additional information. By removing 'B' and 'Tax', we can see other deciding factors, namely 'Age' and 'ZN'. The 'Age' attribute provides information on the proportion of owner-occupied units built prior to 1940 and the 'ZN' attribute informs us about the proportion of non-retail business acres per town.



The **Heat Map** widget is a nice tool for discovering relevant features in the data. By removing some of the more pronounced features, we came across new information, which was hiding in the background.

References

[Housing Data Set](#)

Linear Projection

A linear projection method with explorative data analysis.

Signals

Inputs:

- **Data**

  An input data set

- **Data Subset**

  A subset of data instances

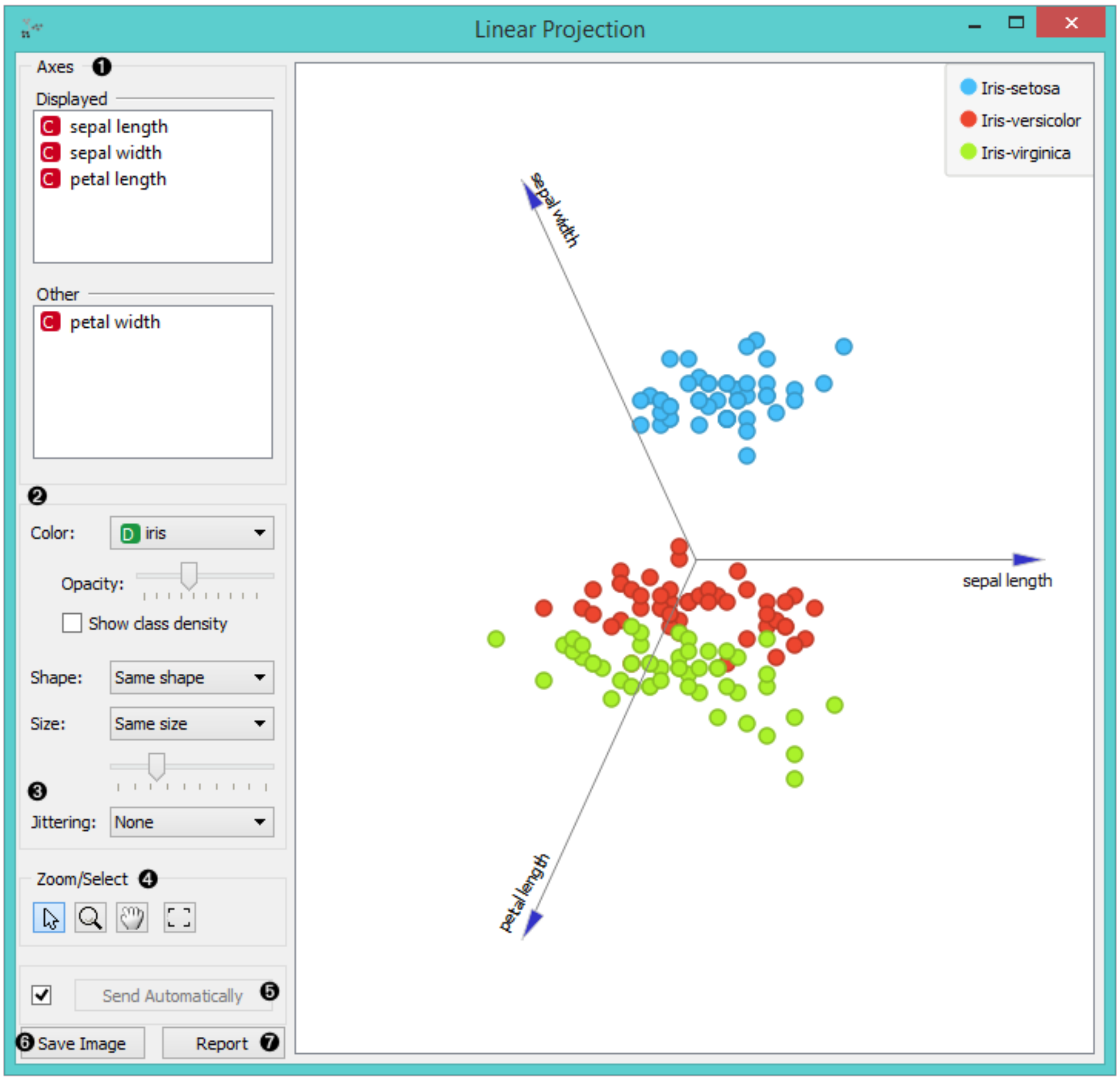
Outputs:

- **Selected Data**

  A data subset that the user has manually selected in the projection.

Description

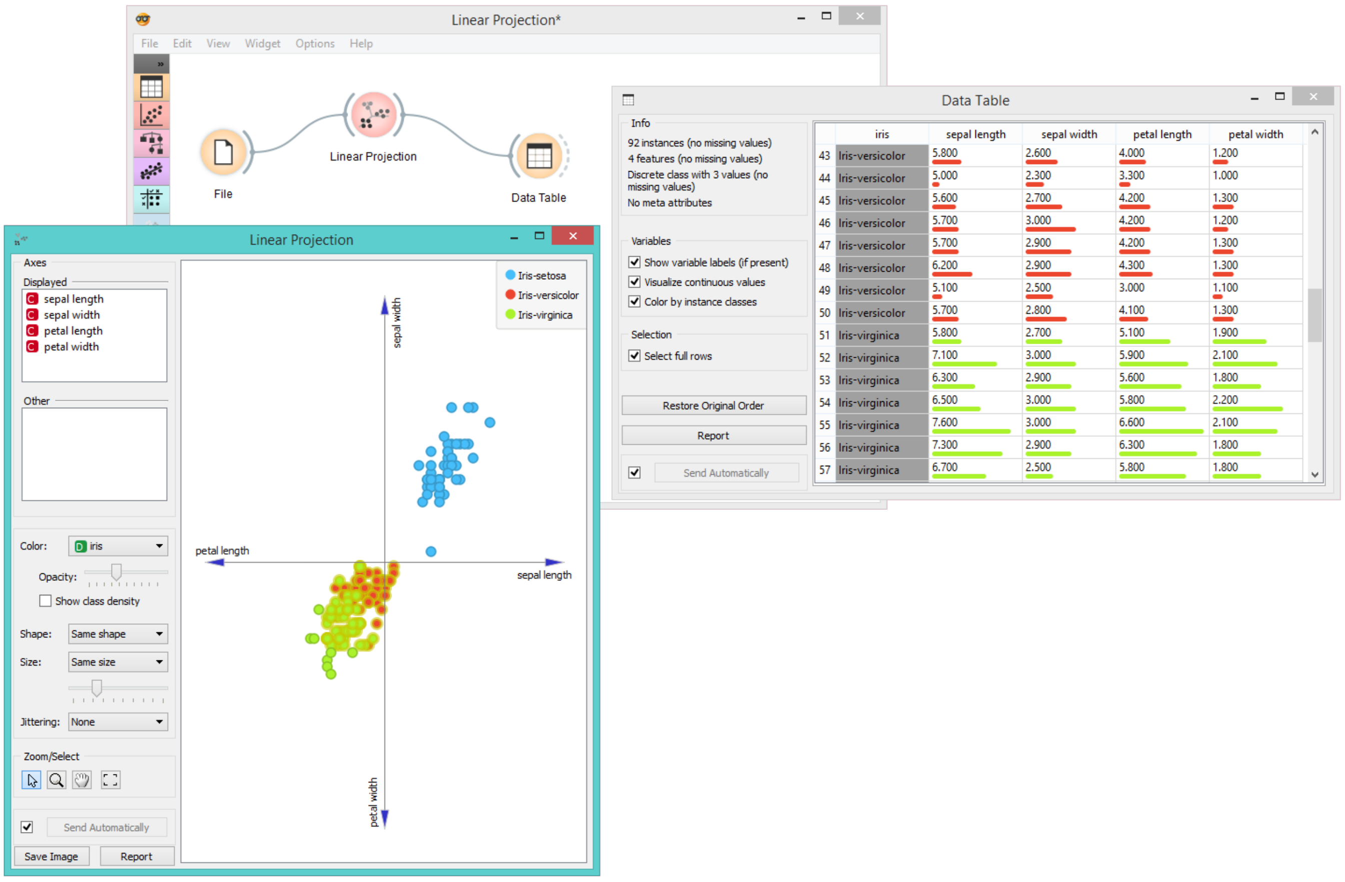
This widget displays linear projections of class-labeled data. Consider, for a start, a projection of the *Iris* data set shown below. Notice that it is the sepal width and sepal length that already separate *Iris setosa* from the other two, while the petal length is the attribute best separating *Iris versicolor* from *Iris virginica*.



1. Axes in the projection that are displayed and other available axes.
2. Set the color of the displayed dots (you will get colored dots for discrete values and grey-scale dots for continuous). Set opacity, shape and size to differentiate between instances.
3. Set jittering to prevent the dots from overlapping (especially for discrete attributes).
4. *Select*, *zoom*, *pan* and *zoom to fit* options for exploring the graph. Manual selection of data instances works as a non-angular/free-hand selection tool. Double click to move the projection. Scroll in or out for zoom.
5. When the box is ticked (*Auto commit is on*), the widget will communicate the changes automatically. Alternatively, click *Commit*.
6. *Save Image* saves the created image to your computer in a .svg or .png format.
7. Produce a report.

Example

The **Linear Projection** widget works just like other visualization widgets. Below, we connected it to the File widget to see the set projected on a 2-D plane. Then we selected the data for further analysis and connected it to the Data Table widget to see the details of the selected subset.



92 instances (no missing values)
4 features (no missing values)
Discrete class with 3 values (no missing values)
No meta attributes

	iris	sepal length	sepal width	petal length	petal width
43	Iris-versicolor	5.800	2.600	4.000	1.200
44	Iris-versicolor	5.000	2.300	3.300	1.000
45	Iris-versicolor	5.600	2.700	4.200	1.300
46	Iris-versicolor	5.700	3.000	4.200	1.200
47	Iris-versicolor	5.700	2.900	4.200	1.300
48	Iris-versicolor	6.200	2.900	4.300	1.300
49	Iris-versicolor	5.100	2.500	3.000	1.100
50	Iris-versicolor	5.700	2.800	4.100	1.300
51	Iris-virginica	5.800	2.700	5.100	1.900
52	Iris-virginica	7.100	3.000	5.900	2.100
53	Iris-virginica	6.300	2.900	5.600	1.800
54	Iris-virginica	6.500	3.000	5.800	2.200
55	Iris-virginica	7.600	3.000	6.600	2.100
56	Iris-virginica	7.300	2.900	6.300	1.800
57	Iris-virginica	6.700	2.500	5.800	1.800

References

Koren Y., Carmel L. (2003). Visualization of labeled data using linear transformations. In Proceedings of IEEE Information Visualization 2003, (InfoVis'03). Available here.

Boulesteix A.-L., Strimmer K. (2006). Partial least squares: a versatile tool for the analysis of high-dimensional genomic data. Briefings in Bioinformatics, 8(1), 32-44. Abstract here.

Mosaic Display

Display data in a mosaic plot.

Signals

Inputs:

- **Data**

  An input data set.

- **Data subset**

  An input data subset.

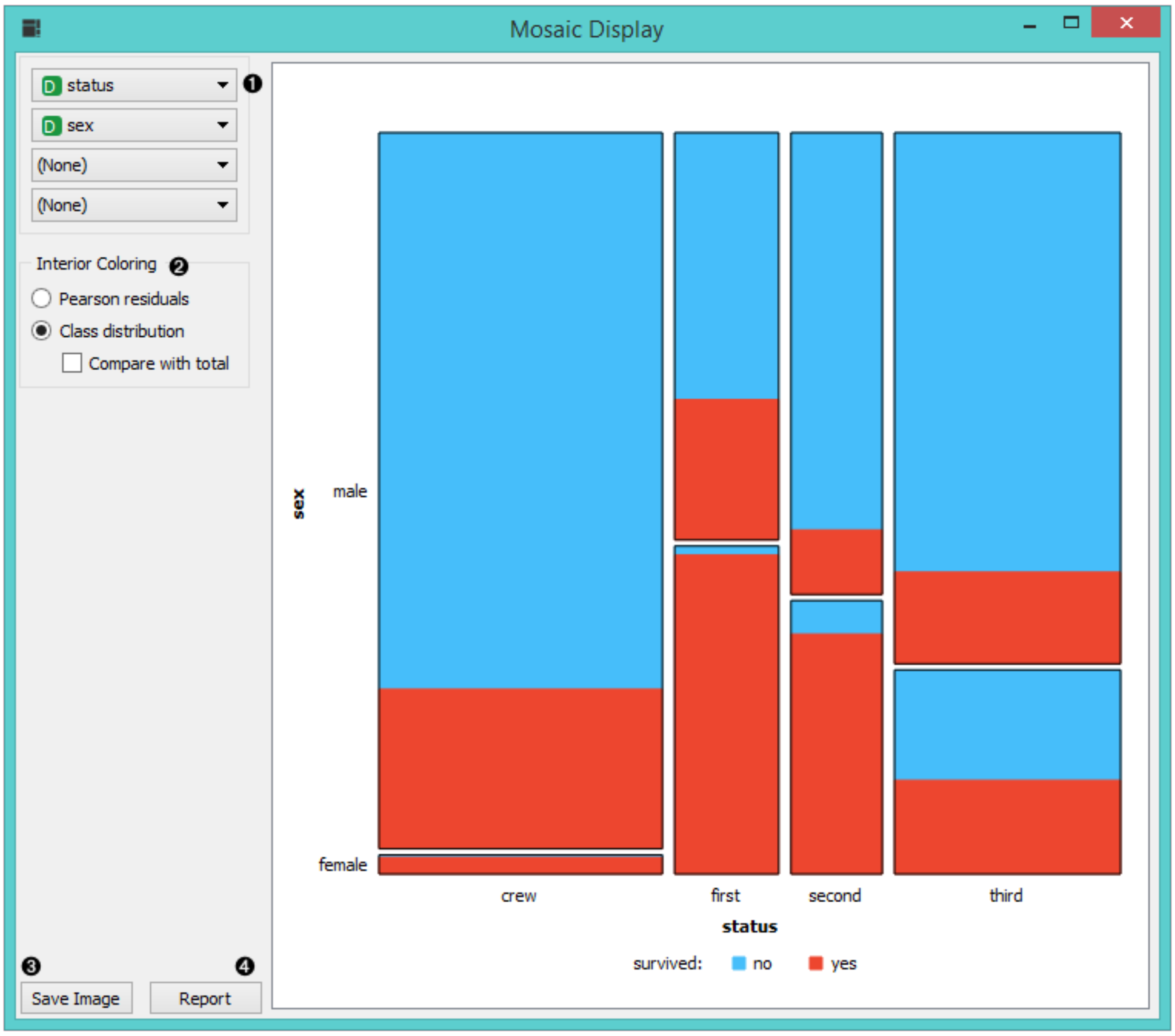
Outputs:

- **Selected data**

  A subset of instances that the user has manually selected from the plot.

Description

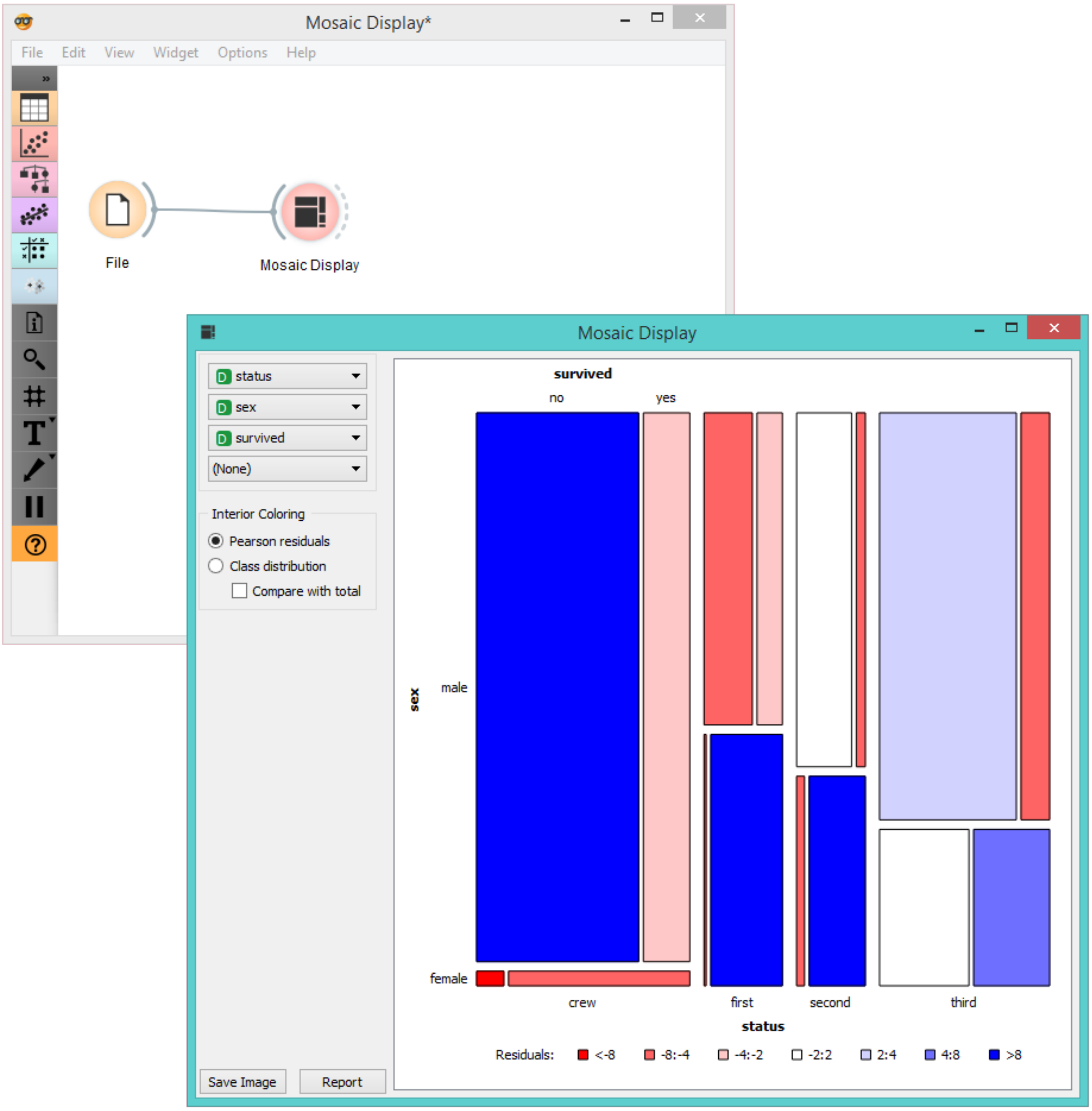
The **Mosaic plot** is a graphical representation of a two-way frequency table or a contingency table. It is used for visualizing data from two or more qualitative variables and was introduced in 1981 by Hartigan and Kleiner and expanded and refined by Friendly in 1994. It provides the user with the means to more efficiently recognize relationships between different variables. If you wish to read up on the history of Mosaic Display, additional reading is available here.



1. Select the variables you wish to see plotted.
2. Select interior coloring. You can color the interior according to class or you can use the *Pearson residual*, which is the difference between observed and fitted values, divided by an estimate of the standard deviation of the observed value. If *Compare to total* is clicked, a comparison is made to all instances.
3. *Save image* saves the created image to your computer in a .svg or .png format.
4. Produce a report.

Example

We loaded the *titanic* data set and connected it to the **Mosaic Display** widget. We decided to focus on two variables, namely status, sex and survival. We colored the interiors according to Pearson residuals in order to demonstrate the difference between observed and fitted values.



We can see that the survival rates for men and women clearly deviate from the fitted value.

Nomogram

Nomograms for visualization of Naive Bayes and Logistic Regression classifiers.

Signals

Inputs:

- **Classifier**

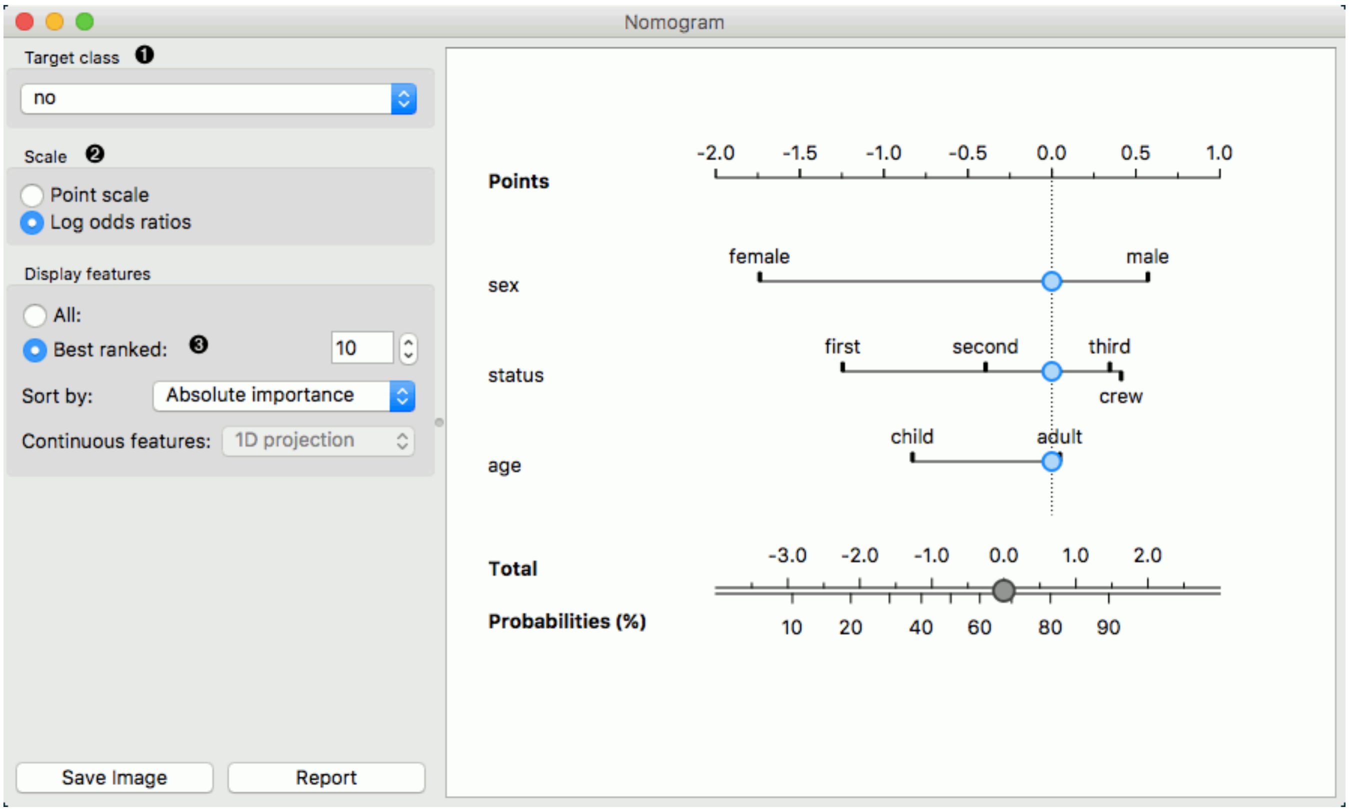
  A trained classifier (Naive Bayes or Logistic regression).

- **Data**

  Data instance.

Description

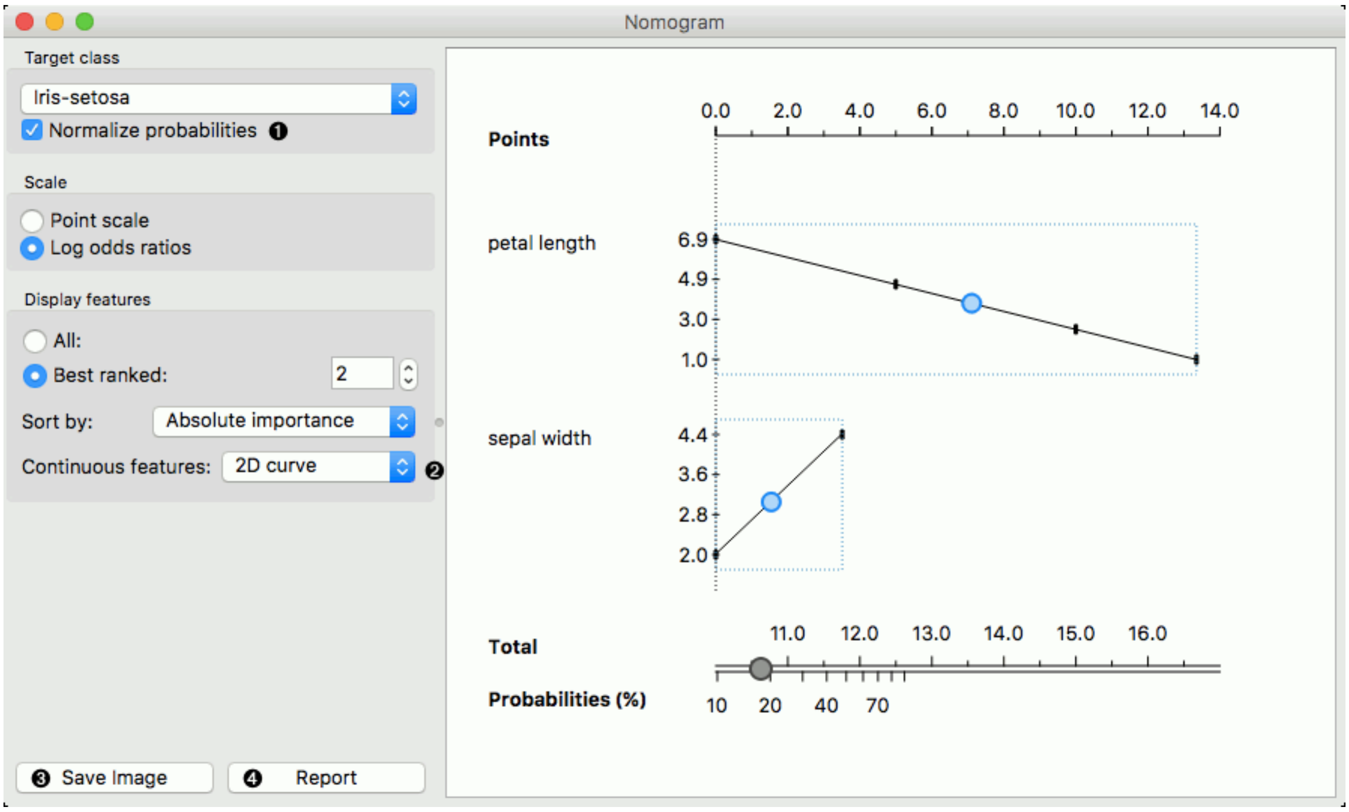
The **Nomogram** enables some classifier's (more precisely Naive Bayes classifier and Logistic Regression classifier) visual representation. It offers an insight into the structure of the training data and effects of the attributes on the class probabilities. Besides visualization of the classifier, the widget offers interactive support to prediction of class probabilities. A snapshot below shows the nomogram of the Titanic data set, that models the probability for a passenger not to survive the disaster of the Titanic.



1. Select the target class you want to model the probability for.
2. By default Scale is set to Log odds ration. For easier understanding and interpretation option Point scale can be used. The unit is obtained by re-scaling the log odds so that the maximal absolute log odds ratio in the nomogram represents 100 points.
3. When there are to many attributes in the plotted data set, you can choose to display only best ranked ones. It is possible to choose from 'No sorting', 'Name', 'Absolute importance', 'Positive influence' and 'Negative influence'

for Naive Bayes representation and from ‘No sorting’, ‘Name’ and ‘Absolute importance’ for Logistic Regression representation.

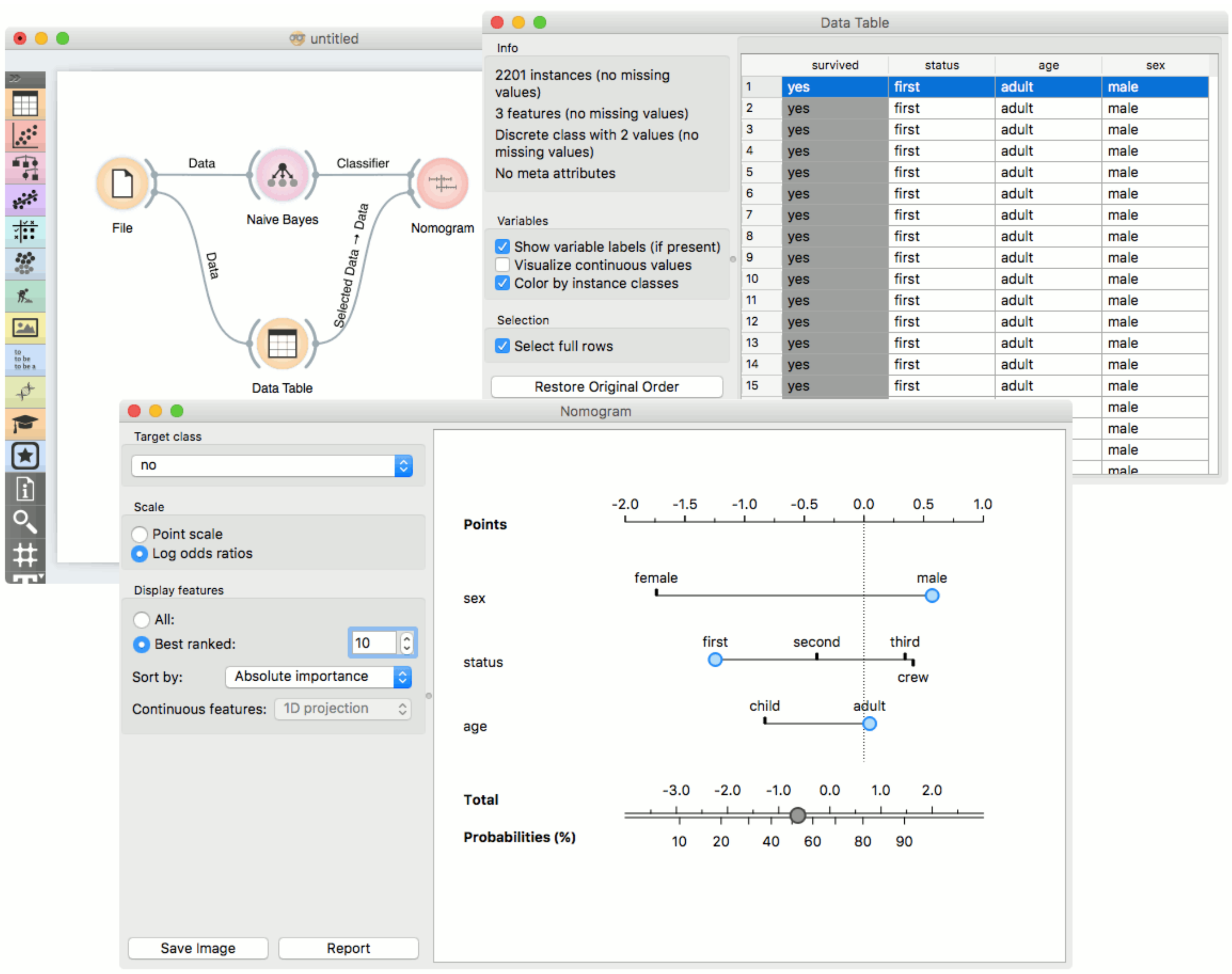
To represent nomogram for Logistic Regressing classifier Iris data set is used:



1. The probability for the chosen target class is computed by 1. vs. all principle, which should be taken in consideration when dealing with multiclass data (alternating probabilities do not sum to 1). To avoid this inconvenience, you can choose to normalize probabilities.
2. Continuous attributes can be plotted in 2D (only for Logistic Regression).
3. Save image.
4. Produce a report.

Example

The **Nomogram** widget should be used immediately after trained classifier widget (e.g. Naive Bayes. It can also be passed a data instance using any widget that enables selection (e.g. Data Table) as shown in the workflow below.



Data Table

Info

2201 instances (no missing values)
3 features (no missing values)
Discrete class with 2 values (no missing values)
No meta attributes

Variables

- [✓] Show variable labels (if present)
- [ ] Visualize continuous values
- [✓] Color by instance classes

Selection

- [✓] Select full rows

	survived	status	age	sex
1	yes	first	adult	male
2	yes	first	adult	male
3	yes	first	adult	male
4	yes	first	adult	male
5	yes	first	adult	male
6	yes	first	adult	male
7	yes	first	adult	male
8	yes	first	adult	male
9	yes	first	adult	male
10	yes	first	adult	male
11	yes	first	adult	male
12	yes	first	adult	male
13	yes	first	adult	male
14	yes	first	adult	male
15	yes	first	adult	male

Nomogram

Target class: no

Scale
- ( ) Point scale
- (●) Log odds ratios

Display features
- ( ) All:
- (●) Best ranked: 10

Sort by: Absolute importance

Continuous features: 1D projection

Referring to the Titanic data set once again, 1490 (68%) of passengers on Titanic, of 2201 in total, died. To make a prediction, the contribution of each attribute is measured as a point score and the individual point scores are summed to determine the probability. When the value of the attribute is unknown, its contribution is 0 points. Therefore, not knowing anything about the passenger, the total point score is 0, and the corresponding probability equals to the unconditional prior. The nomogram in the example shows the case when we know that the passenger is a male adult from the first class. The points sum to -0.36, with a corresponding probability of not surviving of about 53%.

Pythagorean Forest

Pythagorean forest for visualising random forests.

Signals

Inputs:

- **Random Forest**

  Classification / regression tree models as random forest.

Outputs:

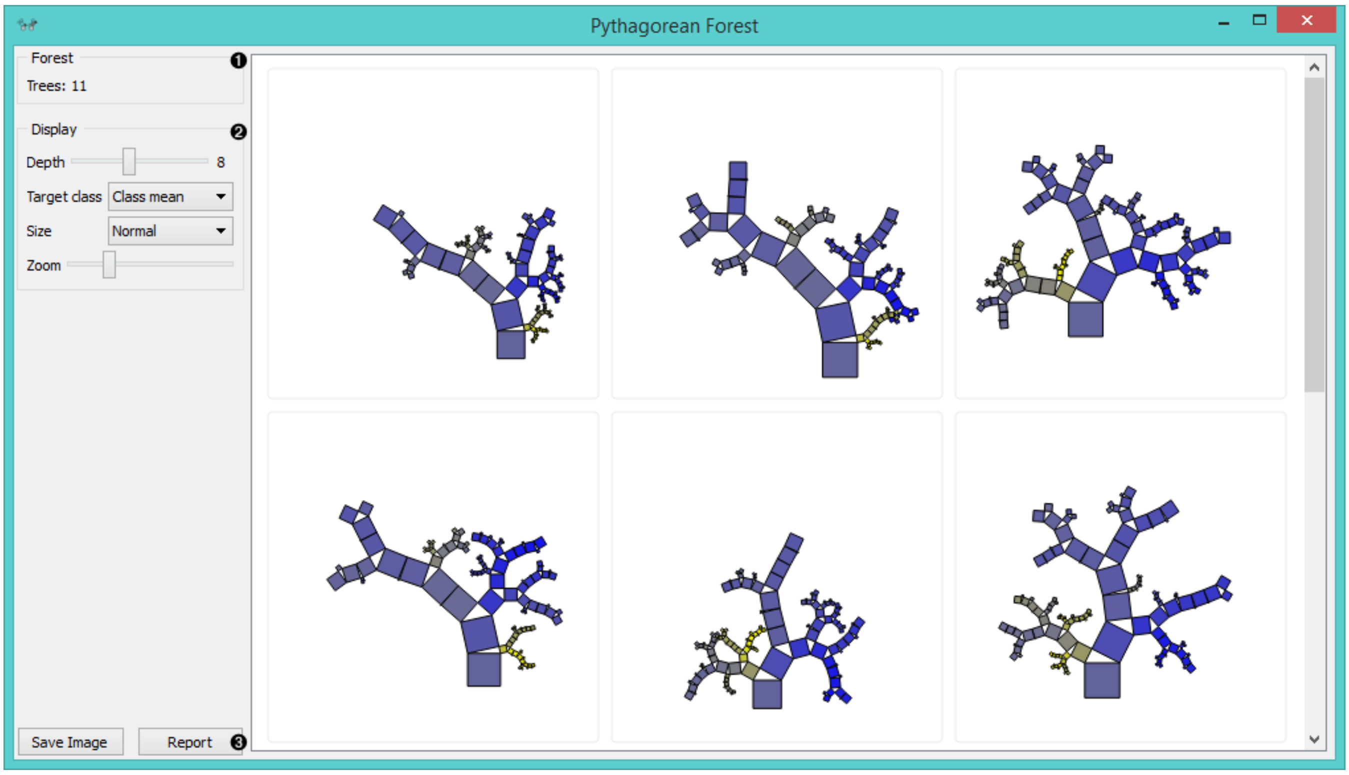
- **Tree**

  A selected classification / regression tree model.

Description

Pythagorean Forest shows all learned decision tree models from Random Forest widget. It displays then as Pythagorean trees, each visualization pertaining to one randomly constructed tree. In the visualization, you can select a tree and display it in Pythagorean Tree wigdet. The best tree is the one with the shortest and most strongly colored branches. This means few attributes split the branches well.

Widget displays both classification and regression results. Classification requires discrete target variable in the data set, while regression requires a continuous target variable. Still, they both should be fed a Tree on the input.



1. Information on the input random forest model.

2. Display parameters:

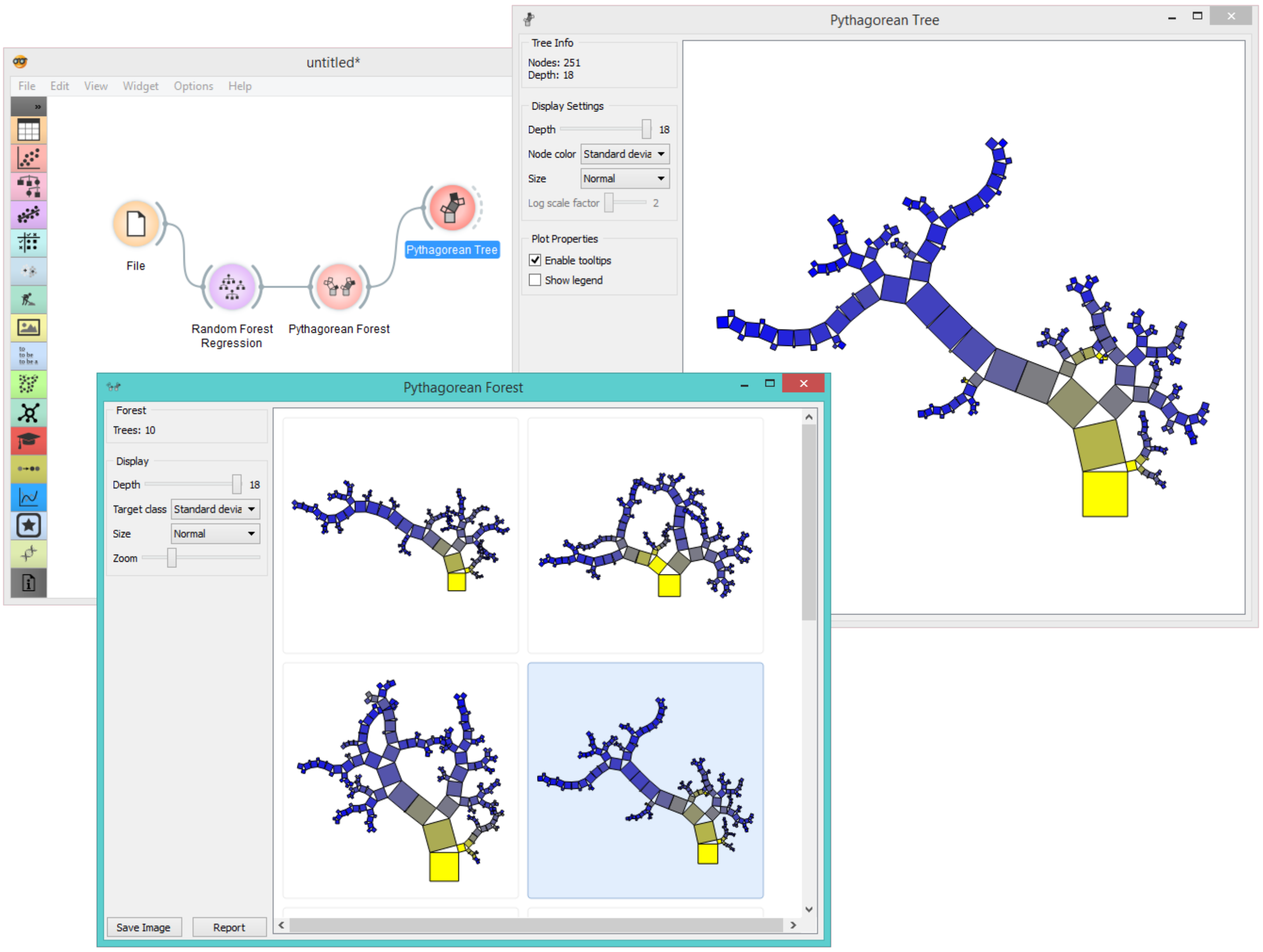
- *Depth*: set the depth to which the trees are grown.
- *Target class*: set the target class for coloring the trees. If *None* is selected, tree will be white. If the input is a classification tree, you can color nodes by their respective class. If the input is a regression tree, the options are *Class mean*, which will color tree nodes by the class mean value and *Standard deviation*, which will color then by the standard deviation value of the node.
- *Size*: set the size of the nodes. *Normal* will keep nodes the size of the subset in the node. *Square root* and *Logarithmic* are the respective transformations of the node size.
- *Zoom*: allows you to se the size of the tree visualizations.

3. *Save Image*: save the visualization to your computer as a *.svg* or *.png* file. *Report*: produce a report.

Example

Pythagorean Forest is great for visualizing several built trees at once. In the example below, we've used *housing* data set and plotted all 10 trees we've grown with Random Forest. When changing the parameters in Random Forest, visualization in Pythagorean Forest will change as well.

Then we've selected a tree in the visualization and inspected it further with Pythagorean Tree widget.



Pythagorean Tree

Tree Info
Nodes: 251
Depth: 18

Display Settings
Depth 18
Node color Standard devia
Size Normal
Log scale factor 2

Plot Properties
☑ Enable tooltips
☐ Show legend

Pythagorean Forest

Forest
Trees: 10

Display
Depth 18
Target class Standard devia
Size Normal
Zoom

References

Beck, F., Burch, M., Munz, T., Di Silvestro, L. and Weiskopf, D. (2014). Generalized Pythagoras Trees for Visualizing Hierarchies. In IVAPP '14 Proceedings of the 5th International Conference on Information Visualization Theory and Applications, 17-28.

Pythagorean Tree

Pythagorean tree visualisation for classification or regression trees.

Signals

Inputs:

- **Tree**

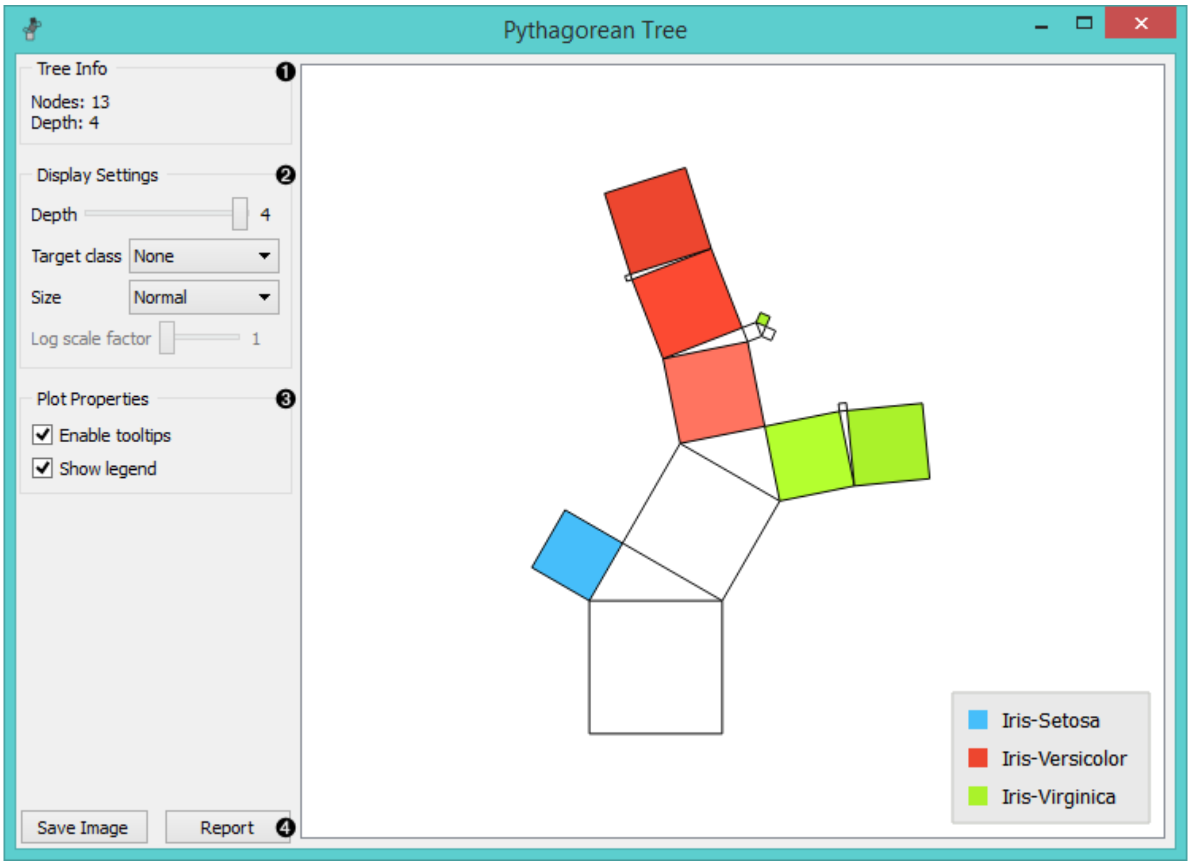
  A decision tree model.

- **Selected Data**

  A subset of instances that the user has manually selected from the Pythagorean tree.

Description

Pythagorean Trees are plane fractals that can be used to depict general tree hierarchies as presented in an article by Fabian Beck and co-authors. In our case, they are used for visualizing and exploring tree models, such as Tree.

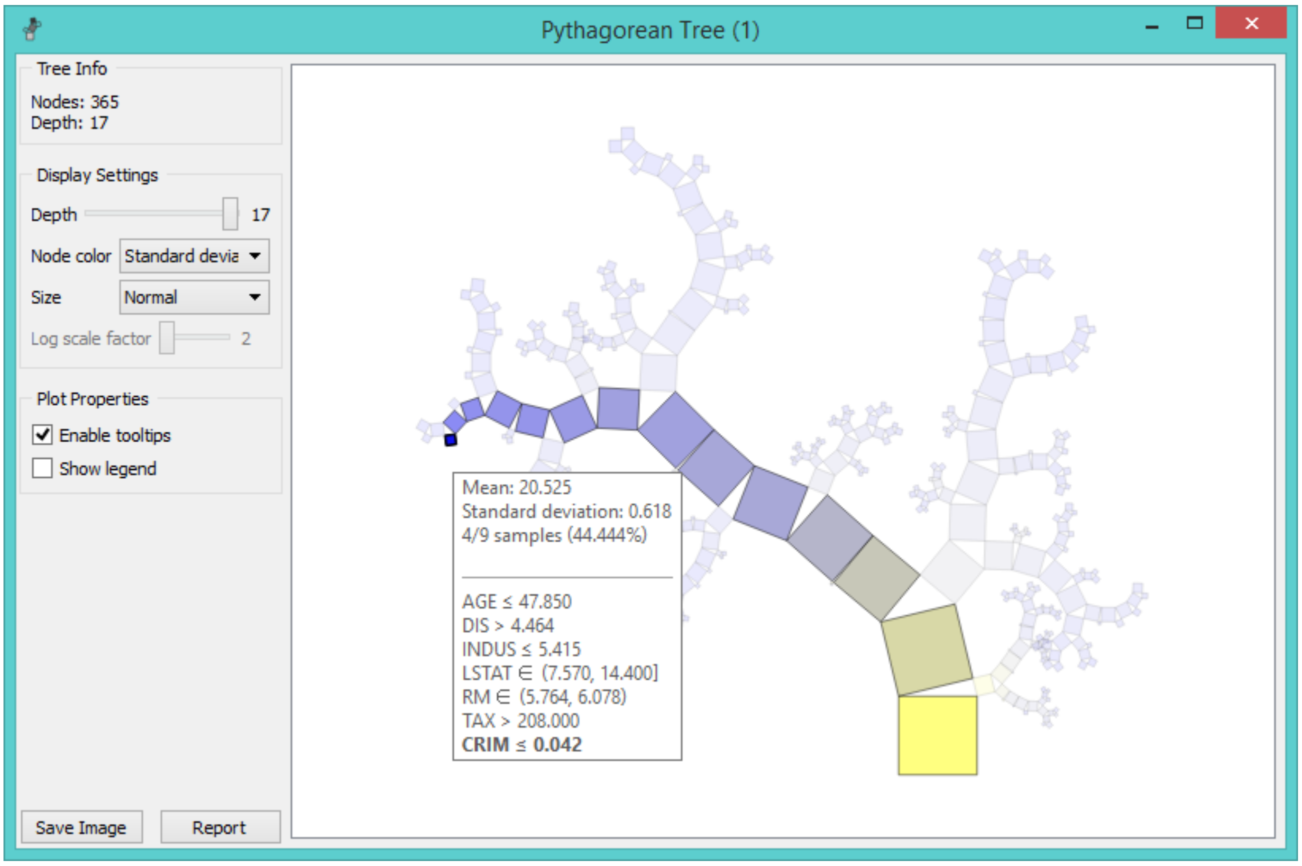


1. Information on the input tree model.
2. Visualization parameters:
   - *Depth*: set the depth of displayed trees.
   - *Target class* (for classification trees): the intensity of the color for nodes of the tree will correspond to the probability of the target class. If *None* is selected, the color of the node will denote the most probable class.

- *Node color* (for regression trees): node colors can correspond to mean or standard deviation of class value of the training data instances in the node.
- *Size*: define a method to compute the size of the square representing the node. *Normal* will keep node sizes correspond to the size of training data subset in the node. *Square root* and *Logarithmic* are the respective transformations of the node size.
- *Log scale factor* is only enabled when *logarithmic* transformation is selected. You can set the log factor between 1 and 10.

3. Plot properties:
    - *Enable tooltips*: display node information upon hovering.
    - *Show legend*: shows color legend for the plot.
4. Reporting:
    - *Save Image*: save the visualization to a SVG or PNG file.
    - *Report*: add visualization to the report.

Pythagorean Tree can visualize both classification and regression trees. Below is an example for regression tree. The only difference between the two is that regression tree doesn't enable coloring by class, but can color by class mean or standard deviation.



Nodes: 365
Depth: 17

Depth 17
Node color Standard devia
Size Normal
Log scale factor 2

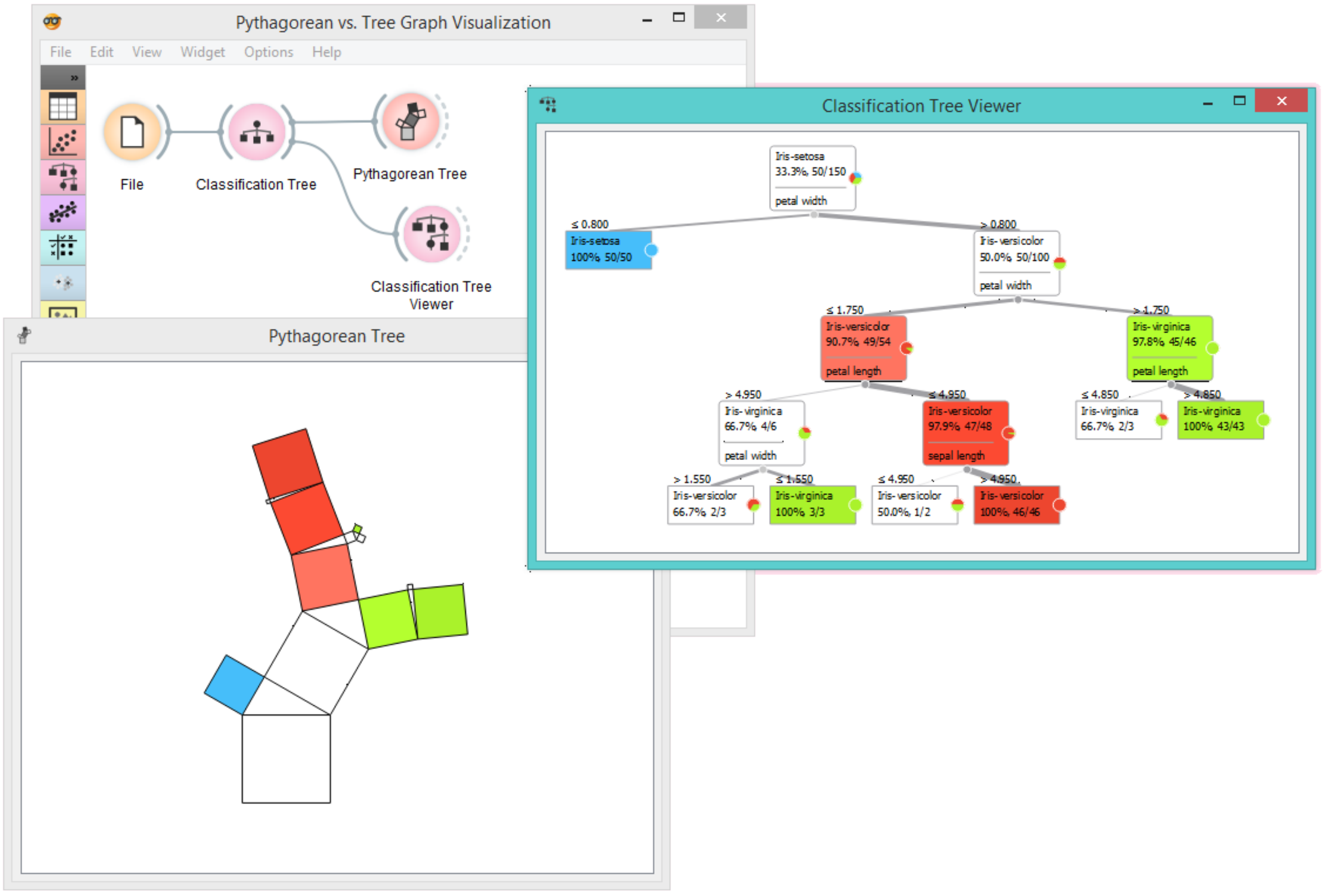
Enable tooltips ✓
Show legend

Mean: 20.525
Standard deviation: 0.618
4/9 samples (44.444%)

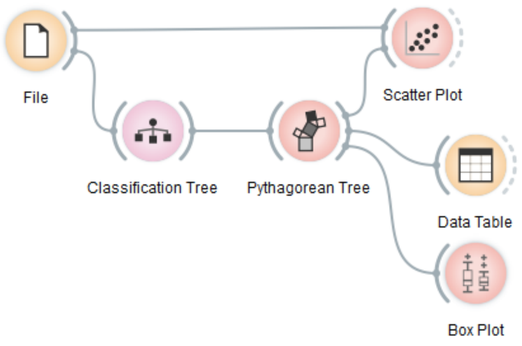
AGE ≤ 47.850
DIS > 4.464
INDUS ≤ 5.415
LSTAT ∈ (7.570, 14.400]
RM ∈ (5.764, 6.078)
TAX > 208.000
CRIM ≤ 0.042

Example

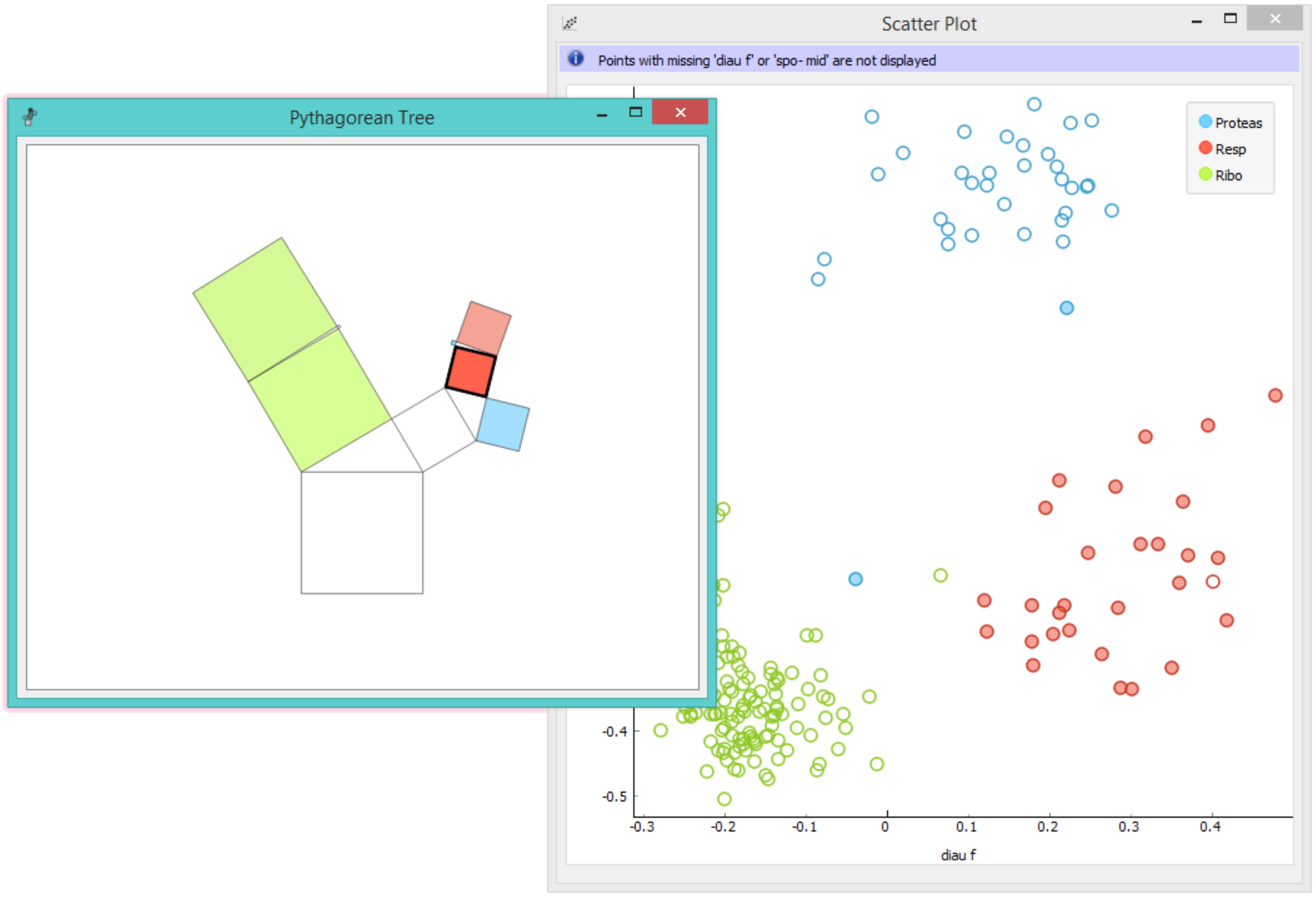
The workflow from the screenshot below demonstrates the difference between Tree Viewer and Pythagorean Tree. They can both visualize Tree, but Pythagorean visualization takes less space and is more compact, even for a small Iris flower data set. For both visualization widgets, we have hidden the control area on the left by clicking on the splitter between control and visualization area.



Pythagorean Tree is interactive: click on any of the nodes (squares) to select training data instances that were associated with that node. The following workflow explores these feature.



The selected data instances are shown as a subset in the Scatter Plot, sent to the Data Table and examined in the Box Plot. We have used brown-selected data set in this example. The tree and scatter plot are shown below; the selected node in the tree has a black outline.



References

Beck, F., Burch, M., Munz, T., Di Silvestro, L. and Weiskopf, D. (2014). Generalized Pythagoras Trees for Visualizing Hierarchies. In IVAPP '14 Proceedings of the 5th International Conference on Information Visualization Theory and Applications, 17-28.

Scatter Map

Plots a scatter map for a pair of continuous attributes.

Signals

Inputs:

- **Data**

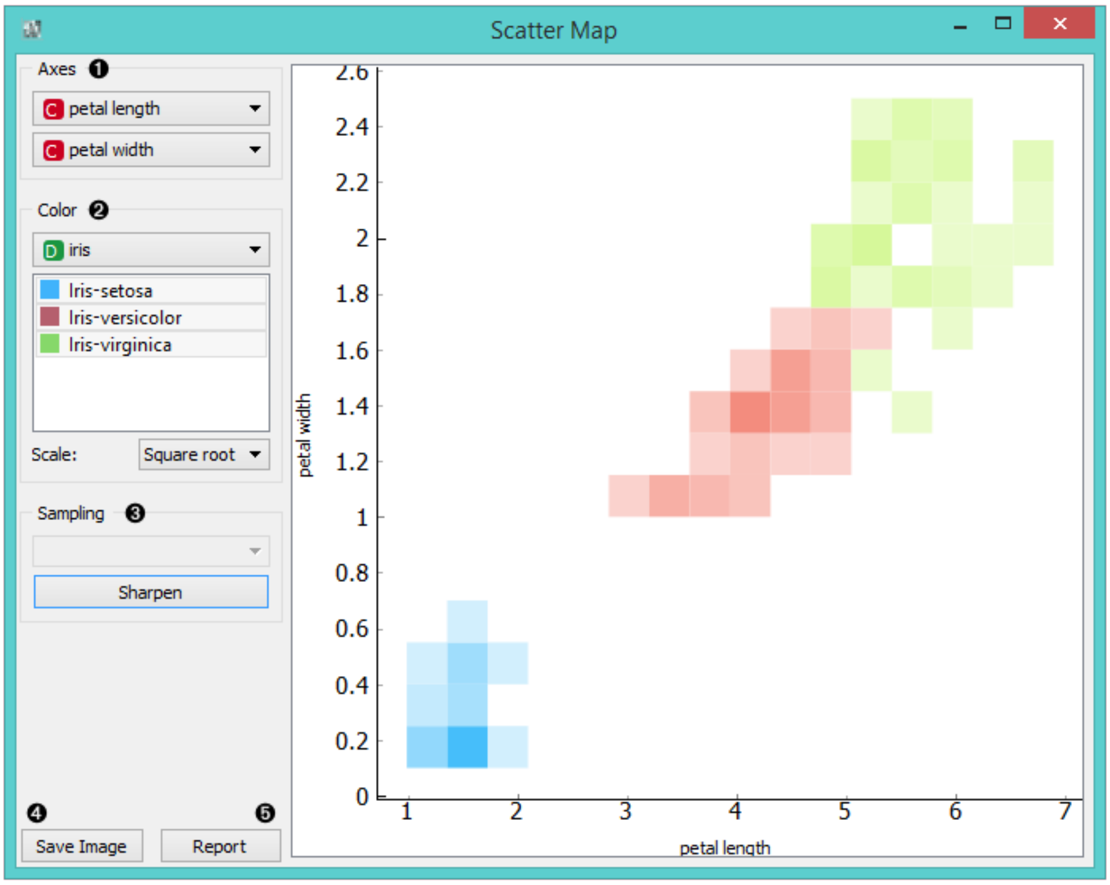
  An input data set

Outputs:

- None

Description

A Scatter map is a graphical method for visualizing frequencies in a two-way matrix by color. The higher the occurrence of a certain value, the darker the represented color. By combining two values on x and y axes, we see where the attribute combination is the strongest and where the weakest, thus enabling the user to find strong correlations or representative instances.

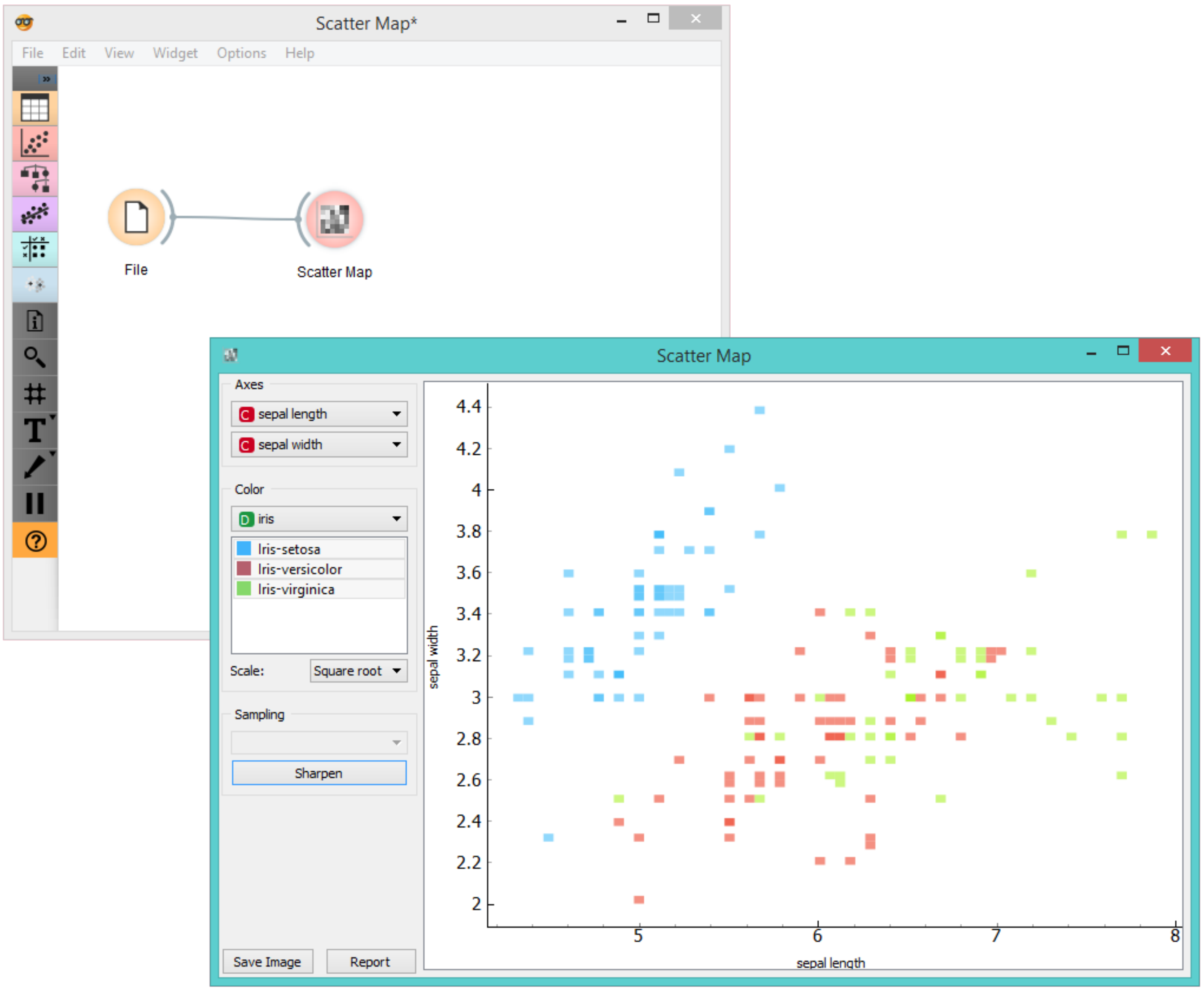


1. Select the x and y attribute to be plotted.
2. Color the plot by attribute. You can also select which attribute instances you wish to see in the visualization by clicking on them. At the bottom, you can select the color scale strength (linear, square root or logarithmic).
3. *Sampling* is enabled only when the widget is connected to the *SQL Table* widget. You can set the sampling time

for large data to speed up the analysis. *Sharpen* works for all data types and it will resize (sharpen) the squares in the plot.
4. *Save Image* saves the created image to your computer in a .svg or .png format.
5. Produce a report.

Example

Below, you can see an example workflow for the **Scatter Map** widget. Notice that the widget only works with continuous data, so you need to first continuize the data attributes you want to visualize. The Scatter map below displays two attributes from the *Iris* data set, namely the petal width and petal length. Here, we can see the distribution of width and length values per Iris type. You can see that the variety *Iris setosa* is distinctly separated from the other two varieties by petal width and length and that the most typical values for these attributes are around 0.2 for petal width and between 1.4 and 1.7 for petal length. This shows that petal width and length are good attributes for telling Iris setosa apart from the other two varieties.



Scatter Plot

Scatterplot visualization with explorative analysis and intelligent data visualization enhancements.

Signals

Inputs:

- **Data**

  An input data set.

- **Data Subset**

  A subset of instances from the input data set.

- **Features**

  A list of attributes.

Outputs:

- **Selected Data**

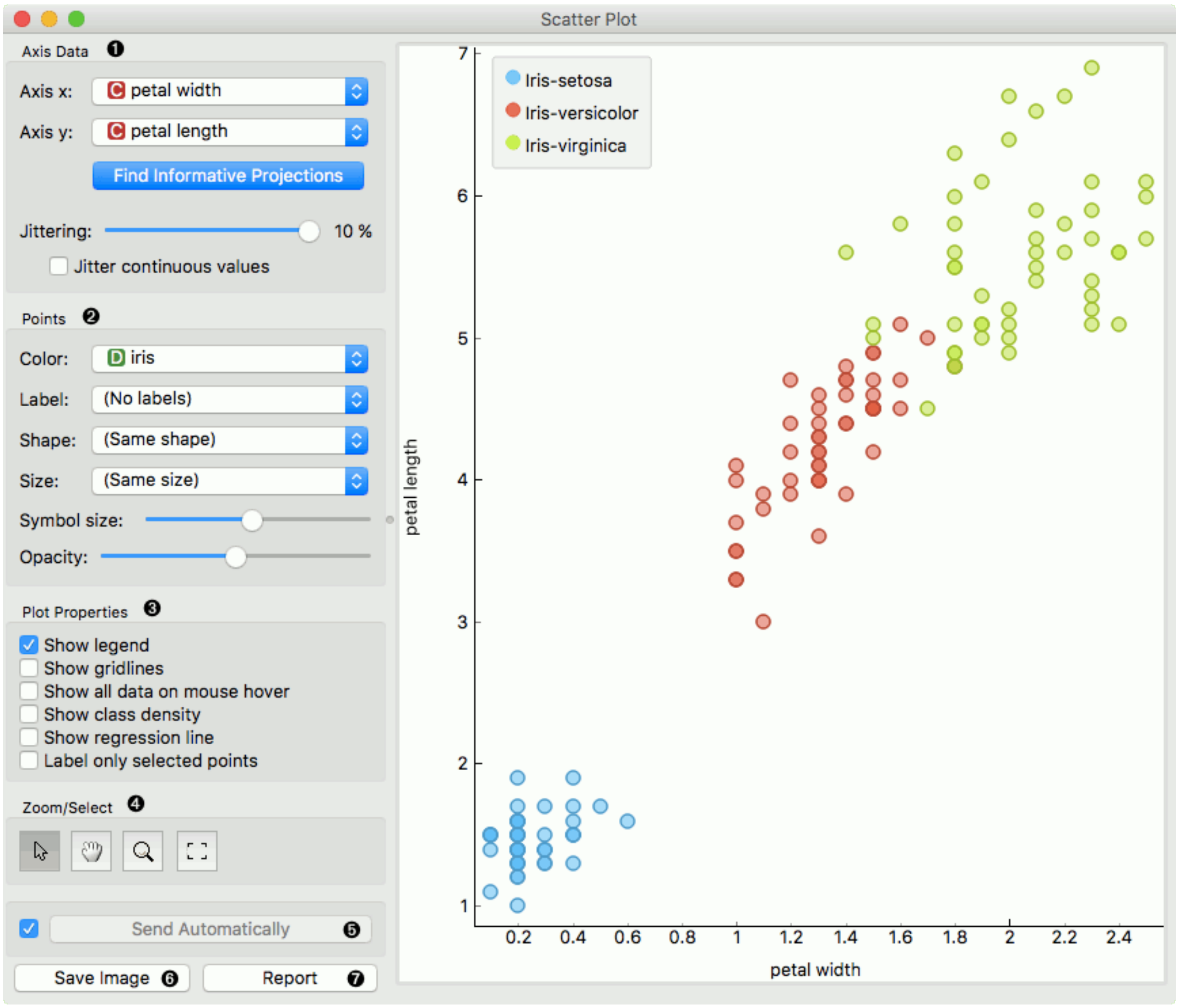
  A subset of instances that the user manually selected from the scatterplot.

- **Data**

  Data with an additional column showing whether a point is selected.

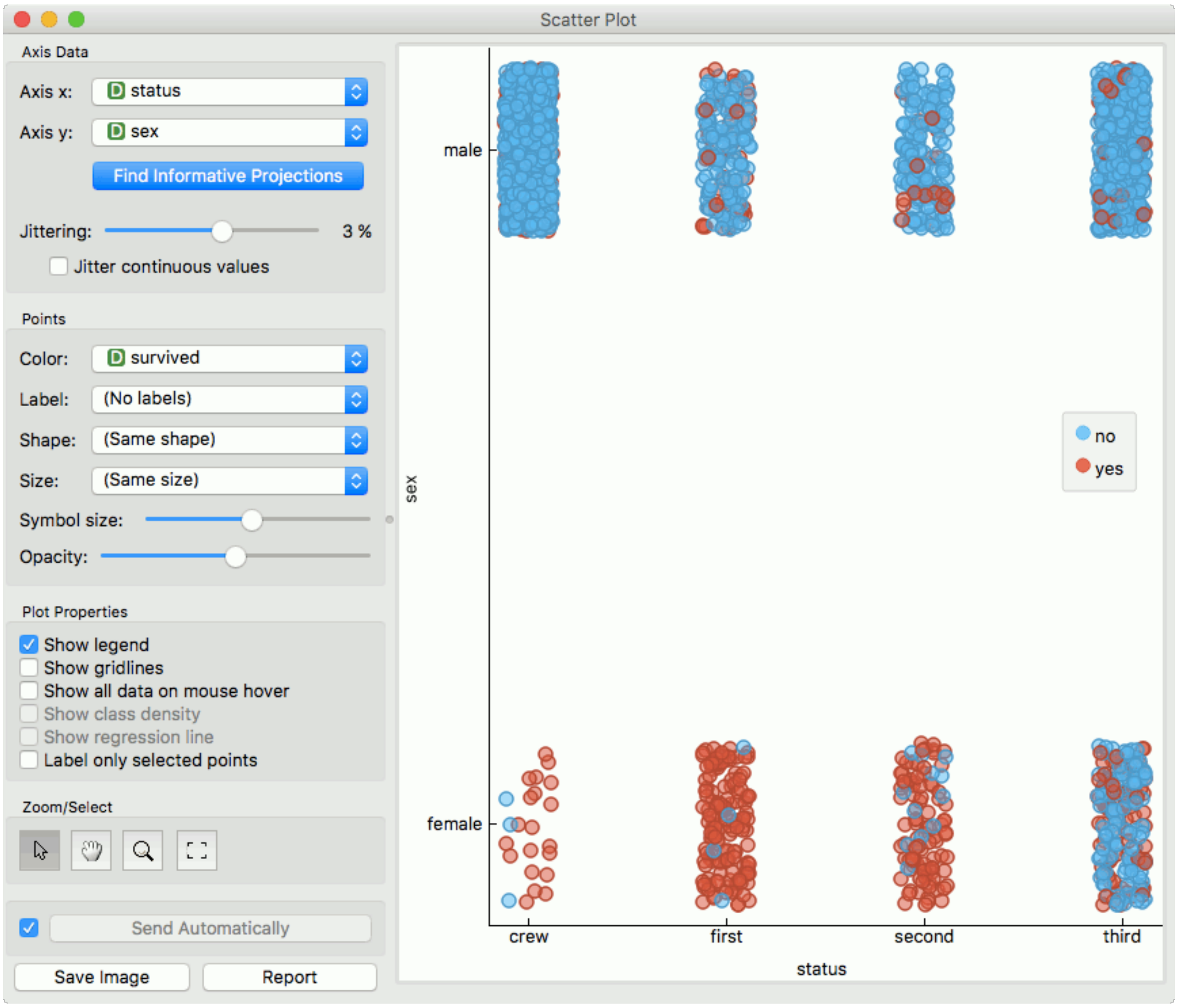
Description

The **Scatterplot** widget provides a 2-dimensional scatterplot visualization for both continuous and discrete-valued attributes. The data is displayed as a collection of points, each having the value of the x-axis attribute determining the position on the horizontal axis and the value of the y-axis attribute determining the position on the vertical axis. Various properties of the graph, like color, size and shape of the points, axis titles, maximum point size and jittering can be adjusted on the left side of the widget. A snapshot below shows the scatterplot of the *Iris* data set with the coloring matching of the class attribute.



1. Select the x and y attribute. Optimize your projection by using **Rank Projections**. This feature scores attribute pairs by average classification accuracy and returns the top scoring pair with a simultaneous visualization update. Set jittering to prevent the dots overlapping. If *Jitter continuous values* is ticked, continuous instances will be dispersed.
2. Set the color of the displayed points (you will get colors for discrete values and grey-scale points for continuous). Set label, shape and size to differentiate between points. Set symbol size and opacity for all data points. Set the desired colors scale.
3. Adjust *plot properties*:
    - *Show legend* displays a legend on the right. Click and drag the legend to move it.
    - *Show gridlines* displays the grid behind the plot.
    - *Show all data on mouse hover* enables information bubbles if the cursor is placed on a dot.
    - *Show class density* colors the graph by class (see the screenshot below).
    - *Show regression line* draws the regression line for pair of continuous attributes.
    - *Label only selected points* allows you to select individual data instances and label them.
4. *Select, zoom, pan and zoom to fit* are the options for exploring the graph. The manual selection of data instances works as an angular/square selection tool. Double click to move the projection. Scroll in or out for zoom.
5. If *Send automatically* is ticked, changes are communicated automatically. Alternatively, press *Send*.
6. *Save Image* saves the created image to your computer in a .svg or .png format.
7. Produce a report.

For discrete attributes, jittering circumvents the overlap of points which have the same value for both axes, and therefore the density of points in the region corresponds better to the data. As an example, the scatterplot for the Titanic data set, reporting on the gender of the passengers and the traveling class is shown below; without jittering, the scatterplot would display only eight distinct points.



Axis Data

Axis x: status

Axis y: sex

Jittering: 3 %

- [ ] Jitter continuous values

Points

Color: survived

Label: (No labels)

Shape: (Same shape)

Size: (Same size)

Symbol size:

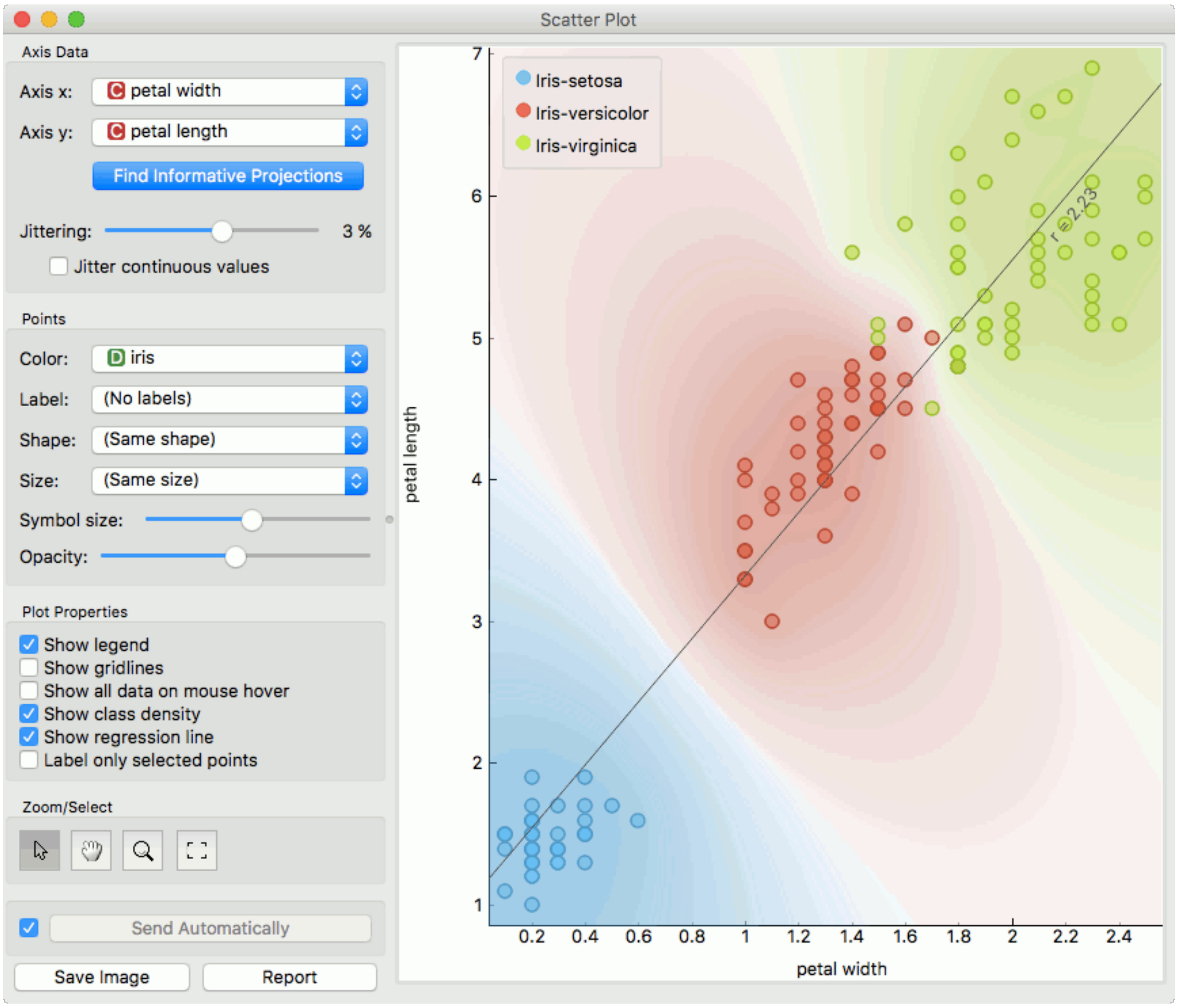
Opacity:

Plot Properties

- [x] Show legend
- [ ] Show gridlines
- [ ] Show all data on mouse hover
- [ ] Show class density
- [ ] Show regression line
- [ ] Label only selected points

Zoom/Select

Here is an example of the **Scatter Plot** widget if the *Show class density* and *Show regression line* boxes are ticked.

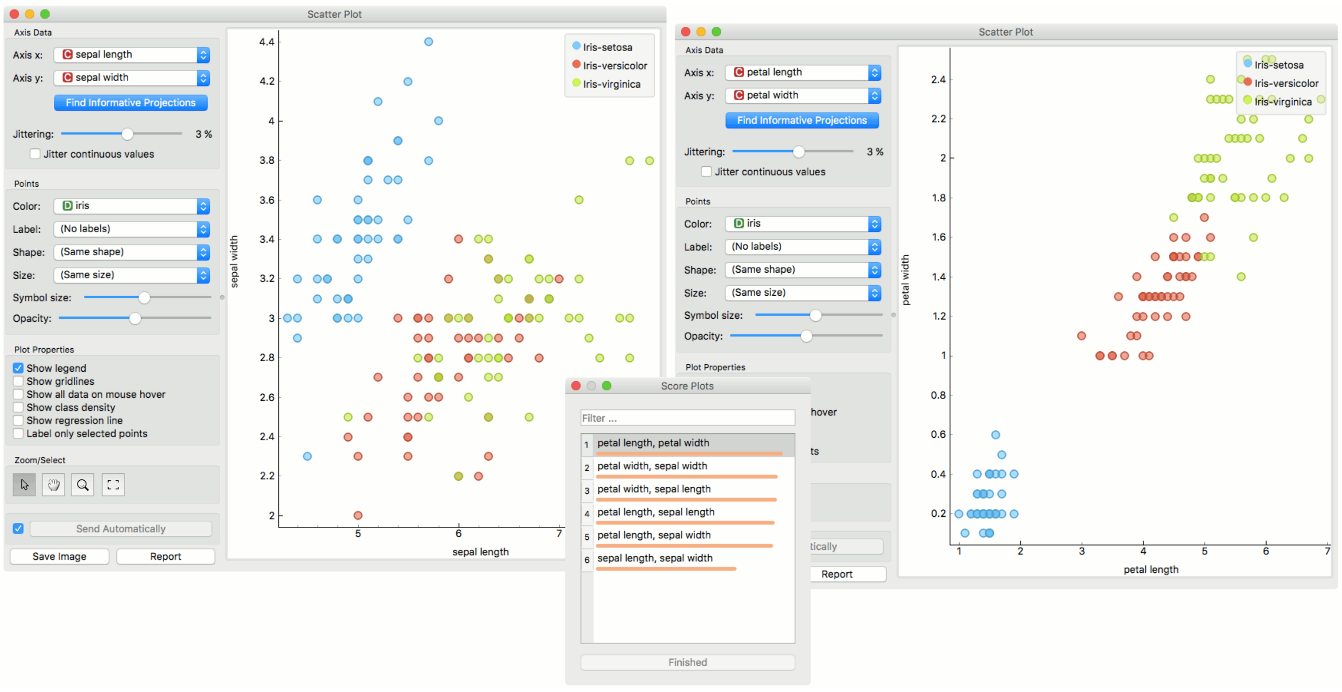


Intelligent Data Visualization

If a data set has many attributes, it is impossible to manually scan through all the pairs to find interesting or useful scatterplots. Orange implements intelligent data visualization with the **Find Informative Projections** option in the widget. The goal of optimization is to find scatterplot projections where instances are well separated.

To use this method, go to the *Find Informative Projections* option in the widget, open the subwindow and press *Start Evaluation*. The feature will return a list of attribute pairs by average classification accuracy score.

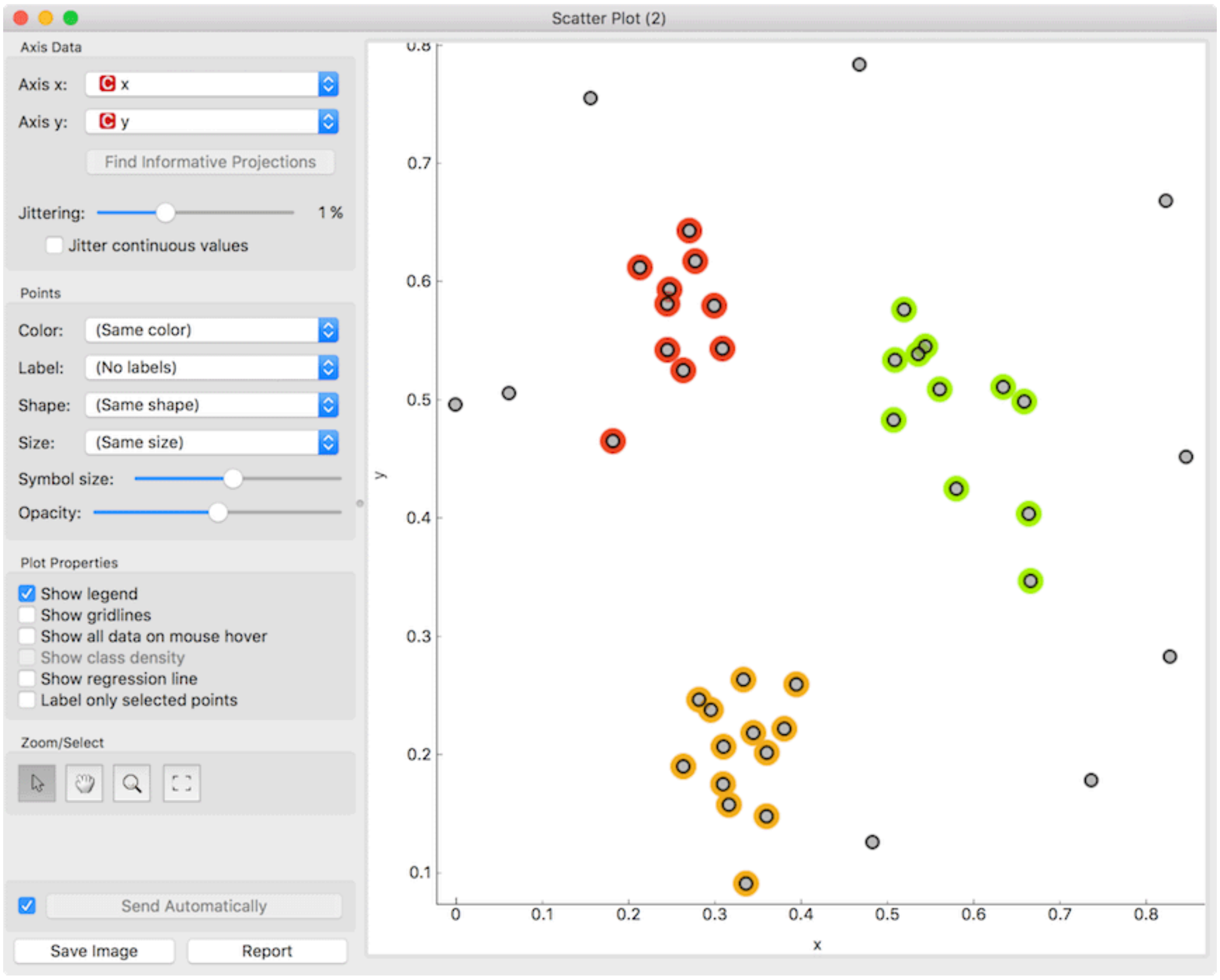
Below, there is an example demonstrating the utility of ranking. The first scatterplot projection was set as the default sepal width to sepal length plot (we used the Iris data set for simplicity). Upon running *Find Informative Projections* optimization, the scatterplot converted to a much better projection of petal width to petal length plot.



Selection

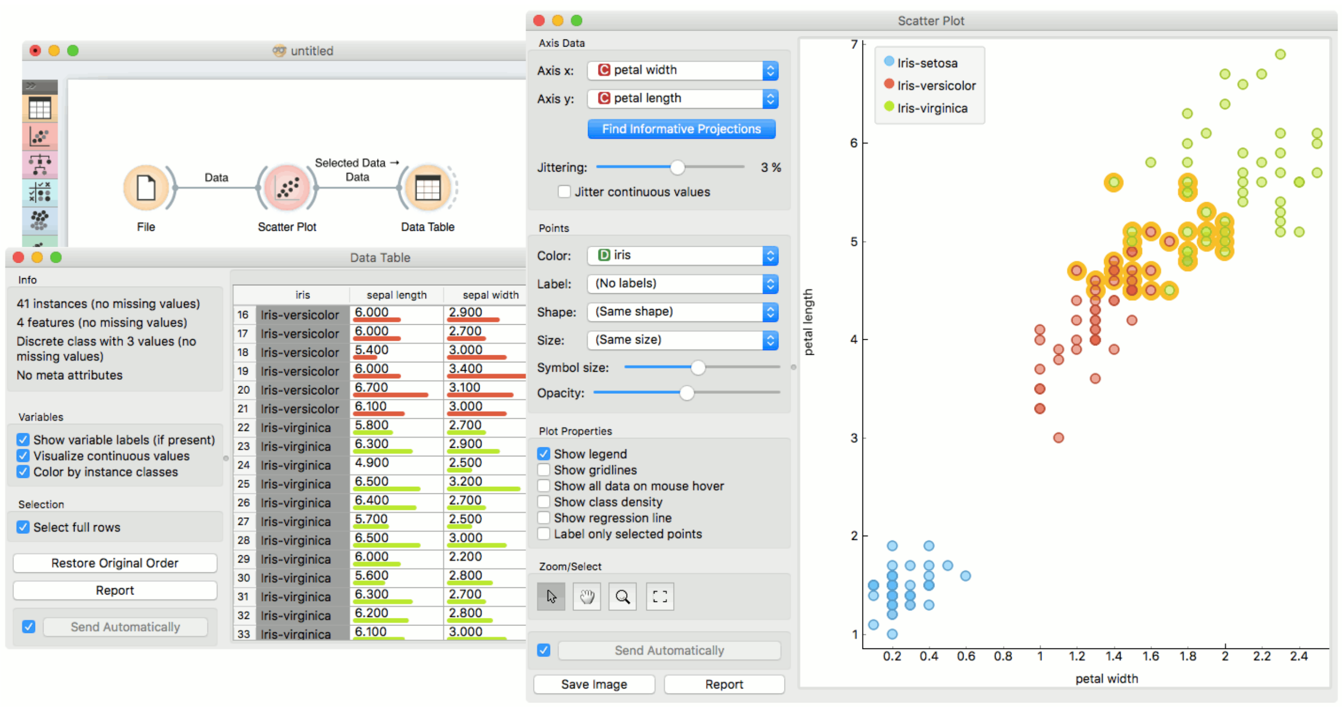
Selection can be used to manually defined subgroups in the data. Use Shift modifier when selecting data instances to put them into a new group. Shift + Ctrl (or Shift + Cmd on macOs) appends instances to the last group.

Signal data outputs a data table with an additional column that contains group indices.



Explorative Data Analysis

The **Scatterplot**, as the rest of Orange widgets, supports zooming-in and out of part of the plot and a manual selection of data instances. These functions are available in the lower left corner of the widget. The default tool is *Select*, which selects data instances within the chosen rectangular area. *Pan* enables you to move the scatterplot around the pane. With *Zoom* you can zoom in and out of the pane with a mouse scroll, while *Reset zoom* resets the visualization to its optimal size. An example of a simple schema, where we selected data instances from a rectangular region and sent them to the Data Table widget, is shown below. Notice that the scatterplot doesn't show all 52 data instances, because some data instances overlap (they have the same values for both attributes used).



Data Table

Info

41 instances (no missing values)
4 features (no missing values)
Discrete class with 3 values (no missing values)
No meta attributes

Variables

- [x] Show variable labels (if present)
- [x] Visualize continuous values
- [x] Color by instance classes

Selection

- [x] Select full rows

- [x] Send Automatically

	iris	sepal length	sepal width
16	Iris-versicolor	6.000	2.900
17	Iris-versicolor	6.000	2.700
18	Iris-versicolor	5.400	3.000
19	Iris-versicolor	6.000	3.400
20	Iris-versicolor	6.700	3.100
21	Iris-versicolor	6.100	3.000
22	Iris-virginica	5.800	2.700
23	Iris-virginica	6.300	2.900
24	Iris-virginica	4.900	2.500
25	Iris-virginica	6.500	3.200
26	Iris-virginica	6.400	2.700
27	Iris-virginica	5.700	2.500
28	Iris-virginica	6.500	3.000
29	Iris-virginica	6.000	2.200
30	Iris-virginica	5.600	2.800
31	Iris-virginica	6.300	2.700
32	Iris-virginica	6.200	2.800
33	Iris-virginica	6.100	3.000

Axis Data

Axis x: petal width
Axis y: petal length
Jittering: 3 %
- [ ] Jitter continuous values

Points

Color: iris
Label: (No labels)
Shape: (Same shape)
Size: (Same size)
Symbol size:
Opacity:

Plot Properties

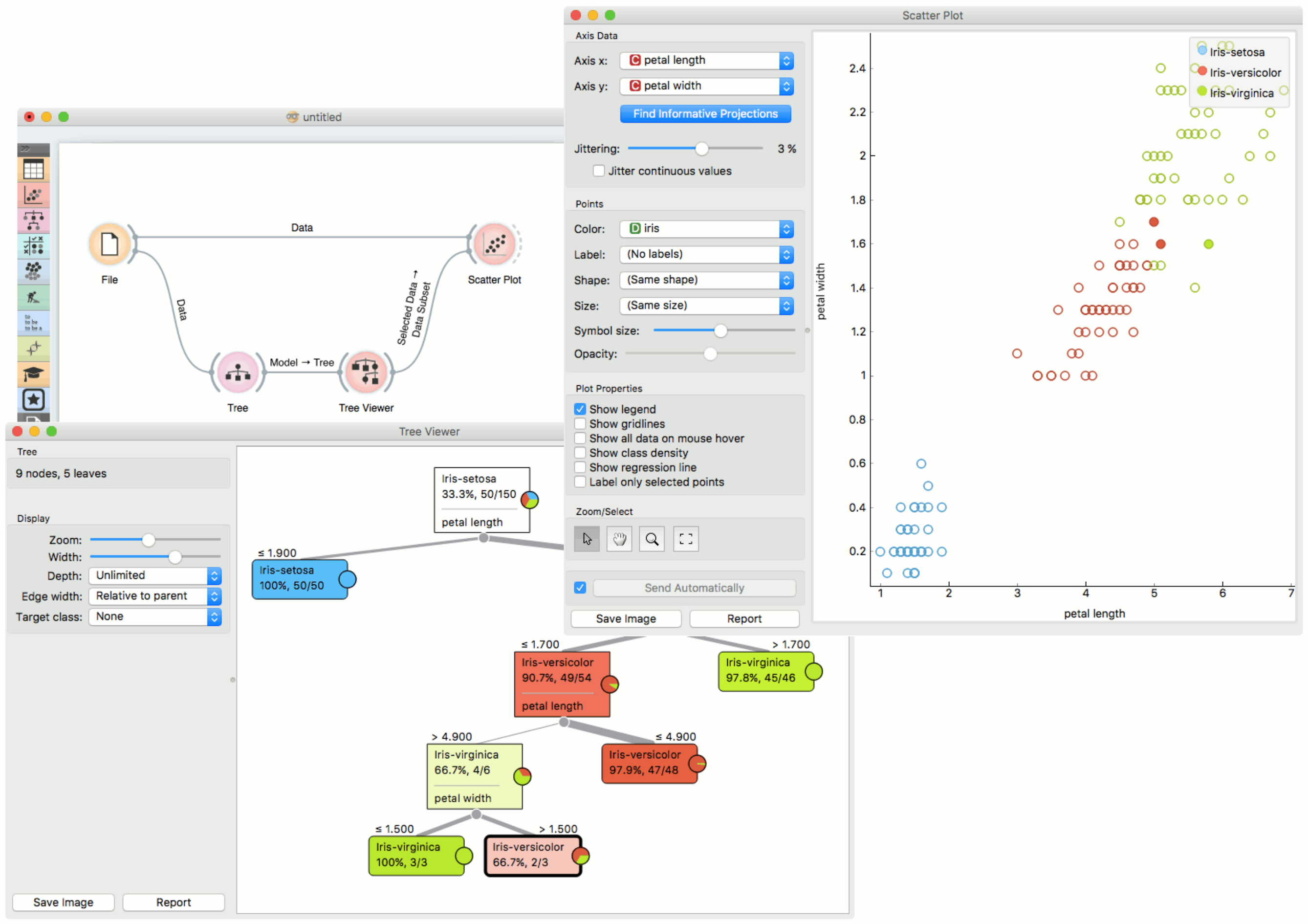
- [x] Show legend
- [ ] Show gridlines
- [ ] Show all data on mouse hover
- [ ] Show class density
- [ ] Show regression line
- [ ] Label only selected points

Zoom/Select

- [x] Send Automatically

Example

The **Scatterplot** can be combined with any widget that outputs a list of selected data instances. In the example below, we combine Tree and **Scatterplot** to display instances taken from a chosen decision tree node (clicking on any node of the tree will send a set of selected data instances to the scatterplot and mark selected instances with filled symbols).



Tree Viewer

Tree

9 nodes, 5 leaves

Display

Zoom:
Width:
Depth: Unlimited
Edge width: Relative to parent
Target class: None

Axis Data

Axis x: petal length
Axis y: petal width
Jittering: 3 %
- [ ] Jitter continuous values

Points

Color: iris
Label: (No labels)
Shape: (Same shape)
Size: (Same size)
Symbol size:
Opacity:

Plot Properties

- [x] Show legend
- [ ] Show gridlines
- [ ] Show all data on mouse hover
- [ ] Show class density
- [ ] Show regression line
- [ ] Label only selected points

Zoom/Select

- [x] Send Automatically

Sieve Diagram

Plots a sieve diagram for a pair of attributes.

Signals

Inputs:

- **Data**

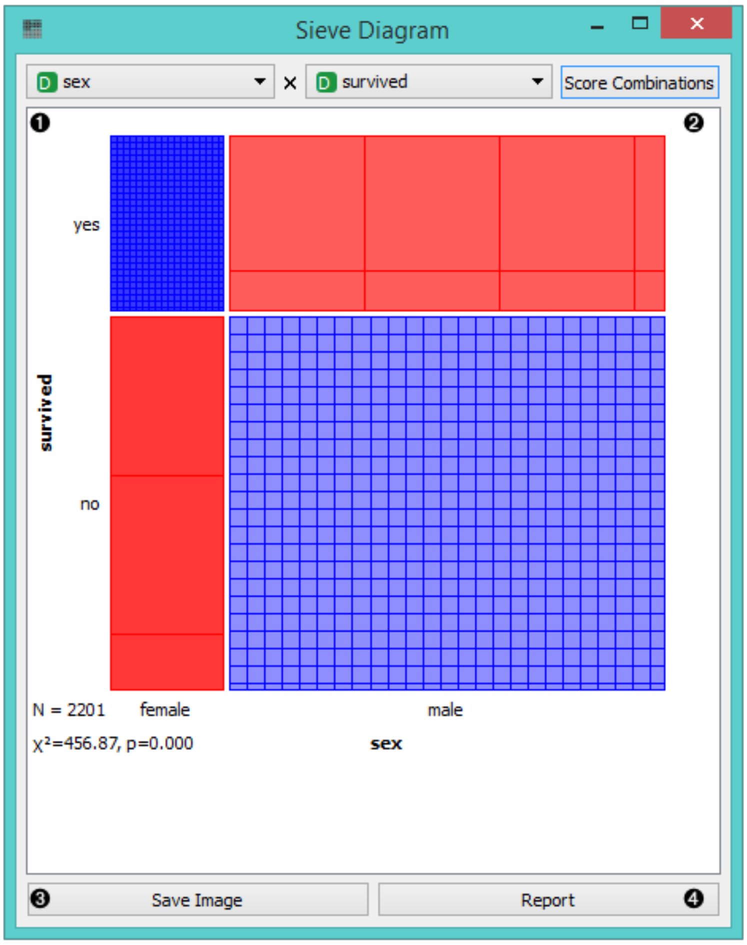
  An input data set

Outputs:

- None

Description

A **Sieve diagram** is a graphical method for visualizing frequencies in a two-way contingency table and comparing them to expected frequencies under assumption of independence. It was proposed by Riedwyl and Schüpbach in a technical report in 1983 and later called a parquet diagram (Riedwyl and Schüpbach, 1994). In this display, the area of each rectangle is proportional to the expected frequency, while the observed frequency is shown by the number of squares in each rectangle. The difference between observed and expected frequency (proportional to the standard Pearson residual) appears as the density of shading, using color to indicate whether the deviation from independence is positive (blue) or negative (red).

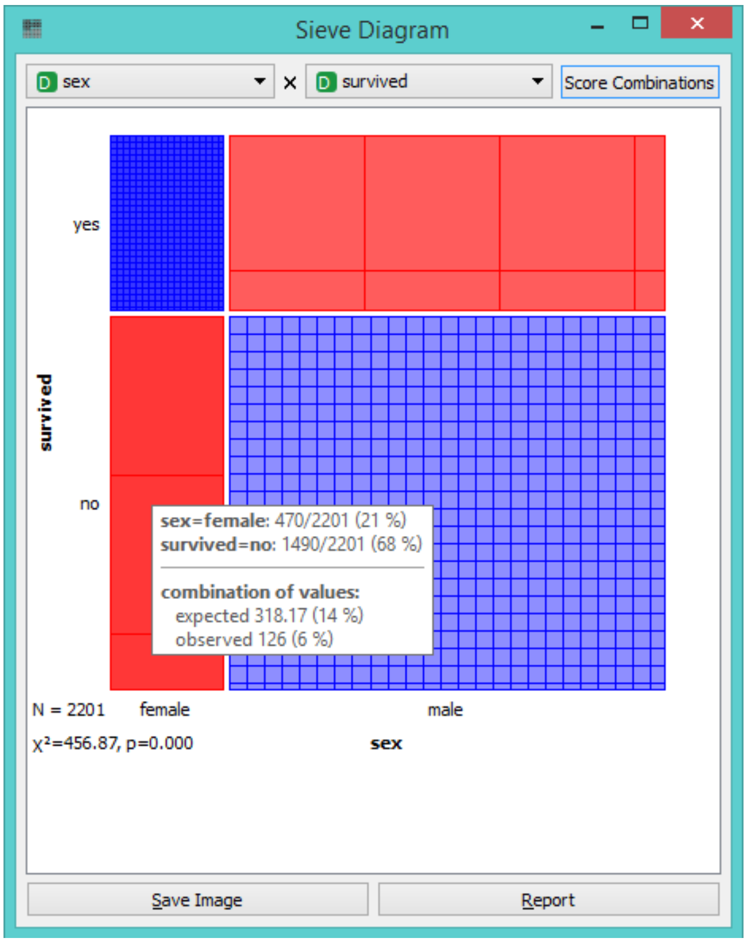


N = 2201

χ²=456.87, p=0.000

1. Select the attributes you want to display in the sieve plot.
2. Score combinations enables you to fin the best possible combination of attributes.
3. *Save Image* saves the created image to your computer in a .svg or .png format.
4. Produce a report.

The snapshot below shows a sieve diagram for the *Titanic* data set and has the attributes *sex* and *survived* (the latter is a class attribute in this data set). The plot shows that the two variables are highly associated, as there are substantial differences between observed and expected frequencies in all of the four quadrants. For example, and as highlighted in the balloon, the chance for surviving the accident was much higher for female passengers than expected (0.06 vs. 0.15).

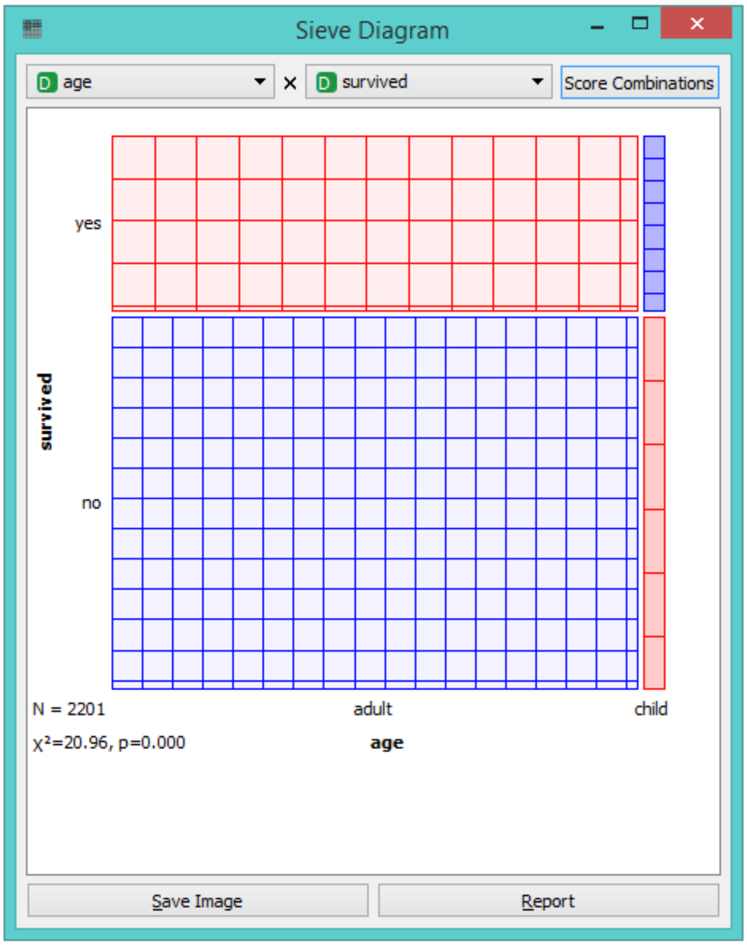


sex=female: 470/2201 (21 %)
survived=no: 1490/2201 (68 %)

combination of values:
expected 318.17 (14 %)
observed 126 (6 %)

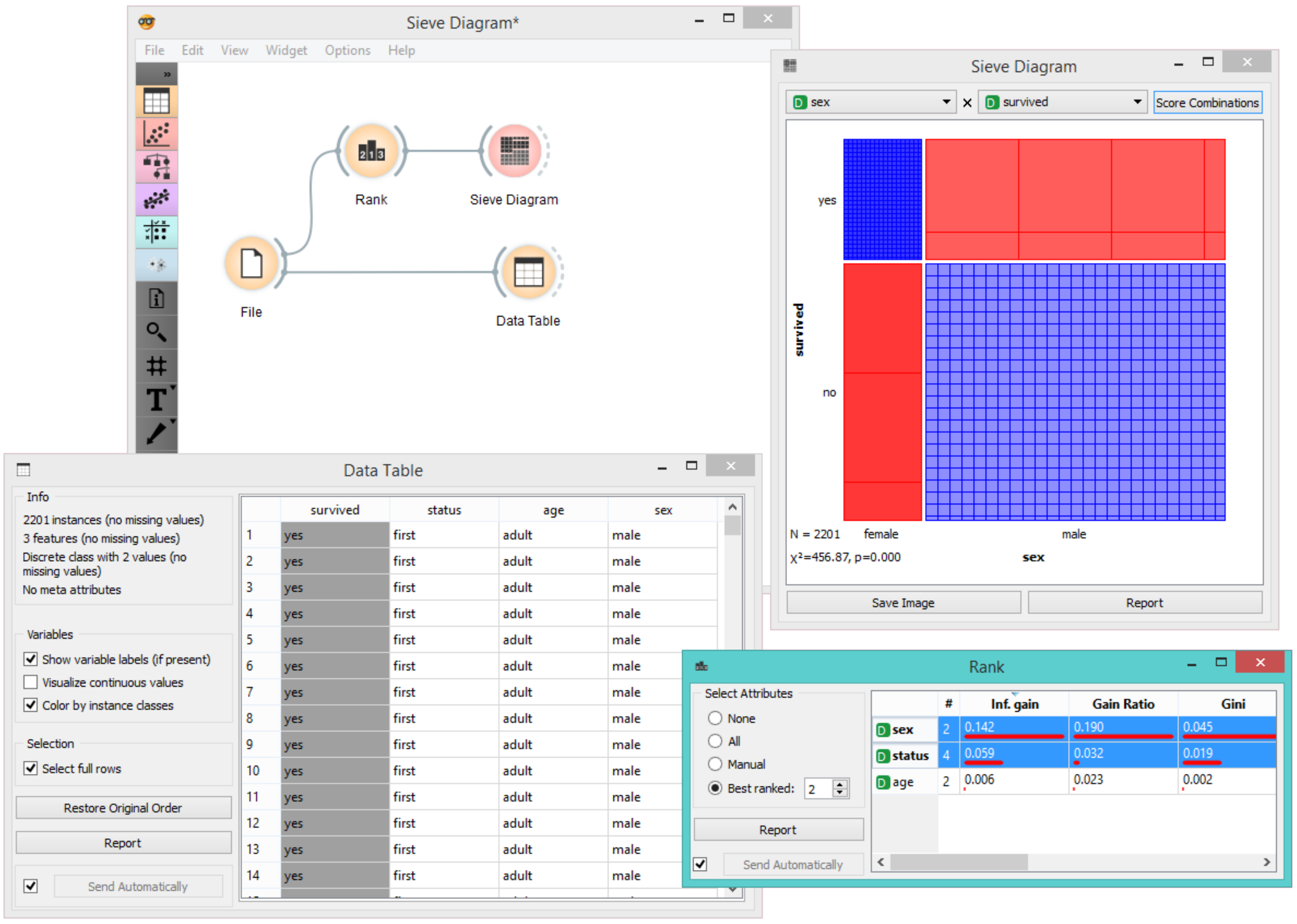
N = 2201
χ²=456.87, p=0.000

Pairs of attributes with interesting associations have a strong shading, such as the diagram shown in the above snapshot. For contrast, a sieve diagram of the least interesting pair (age vs. survival) is shown below.



Example

Below, we see a simple schema using the *Titanic* data set, where we use the Rank widget to select the best attributes (the ones with the highest information gain, gain ratio or gini index) and feed them into the **Sieve Diagram**. This displays the sieve plot for the two best attributes, which in our case are sex and status. We see that the survival rate on the Titanic was very high for women of the first class and very low for female crew members.



Data Table

Info
2201 instances (no missing values)
3 features (no missing values)
Discrete class with 2 values (no missing values)
No meta attributes

Variables
- [x] Show variable labels (if present)
- [ ] Visualize continuous values
- [x] Color by instance classes

Selection
- [x] Select full rows

	survived	status	age	sex
1	yes	first	adult	male
2	yes	first	adult	male
3	yes	first	adult	male
4	yes	first	adult	male
5	yes	first	adult	male
6	yes	first	adult	male
7	yes	first	adult	male
8	yes	first	adult	male
9	yes	first	adult	male
10	yes	first	adult	male
11	yes	first	adult	male
12	yes	first	adult	male
13	yes	first	adult	male
14	yes	first	adult	male

Sieve Diagram

sex × survived

survived: yes, no
sex: female, male

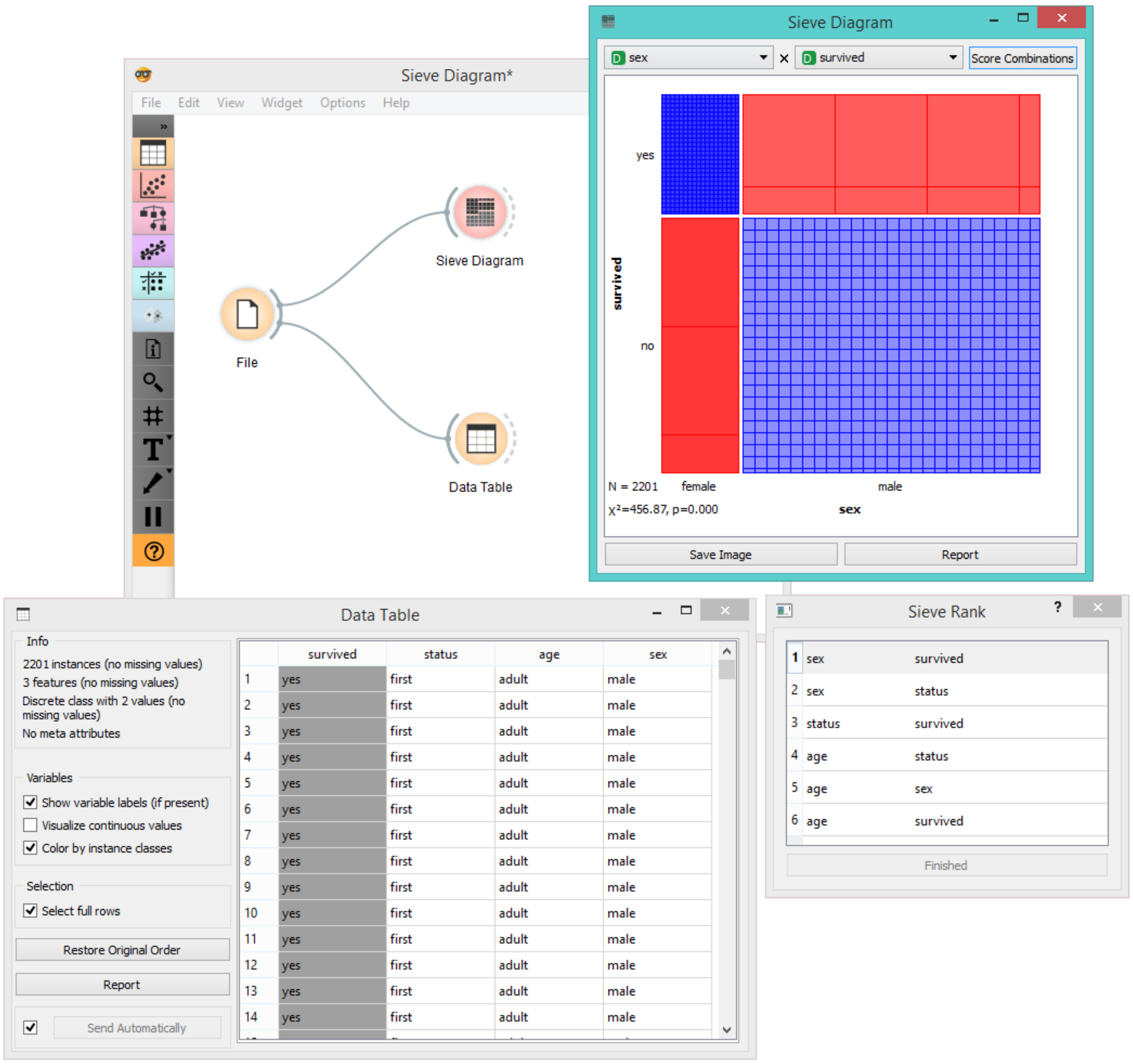
N = 2201
χ²=456.87, p=0.000

Rank

Select Attributes
- None
- All
- Manual
- [✓ Best ranked: 2]

	#	Inf. gain	Gain Ratio	Gini
sex	2	0.142	0.190	0.045
status	4	0.059	0.032	0.019
age	2	0.006	0.023	0.002

The **Sieve Diagram** also features the *Score Combinations* option, which makes the ranking of attributes even easier.



N = 2201

χ^2=456.87, p=0.000

Info

2201 instances (no missing values)
3 features (no missing values)
Discrete class with 2 values (no missing values)
No meta attributes

Variables

☑ Show variable labels (if present)
☐ Visualize continuous values
☑ Color by instance classes

Selection

☑ Select full rows

Restore Original Order

Report

☑ Send Automatically

	survived	status	age	sex
1	yes	first	adult	male
2	yes	first	adult	male
3	yes	first	adult	male
4	yes	first	adult	male
5	yes	first	adult	male
6	yes	first	adult	male
7	yes	first	adult	male
8	yes	first	adult	male
9	yes	first	adult	male
10	yes	first	adult	male
11	yes	first	adult	male
12	yes	first	adult	male
13	yes	first	adult	male
14	yes	first	adult	male

1	sex	survived
2	sex	status
3	status	survived
4	age	status
5	age	sex
6	age	survived

References

Riedwyl, H., and Schüpbach, M. (1994). Parquet diagram to plot contingency tables. In Softstat '93: Advances in Statistical Software, F. Faulbaum (Ed.). New York: Gustav Fischer, 293-299.

Silhouette Plot

A graphical representation of consistency within clusters of data.

Signals

Inputs

- **Data**

  A data set.

Outputs

- **Selected Data**

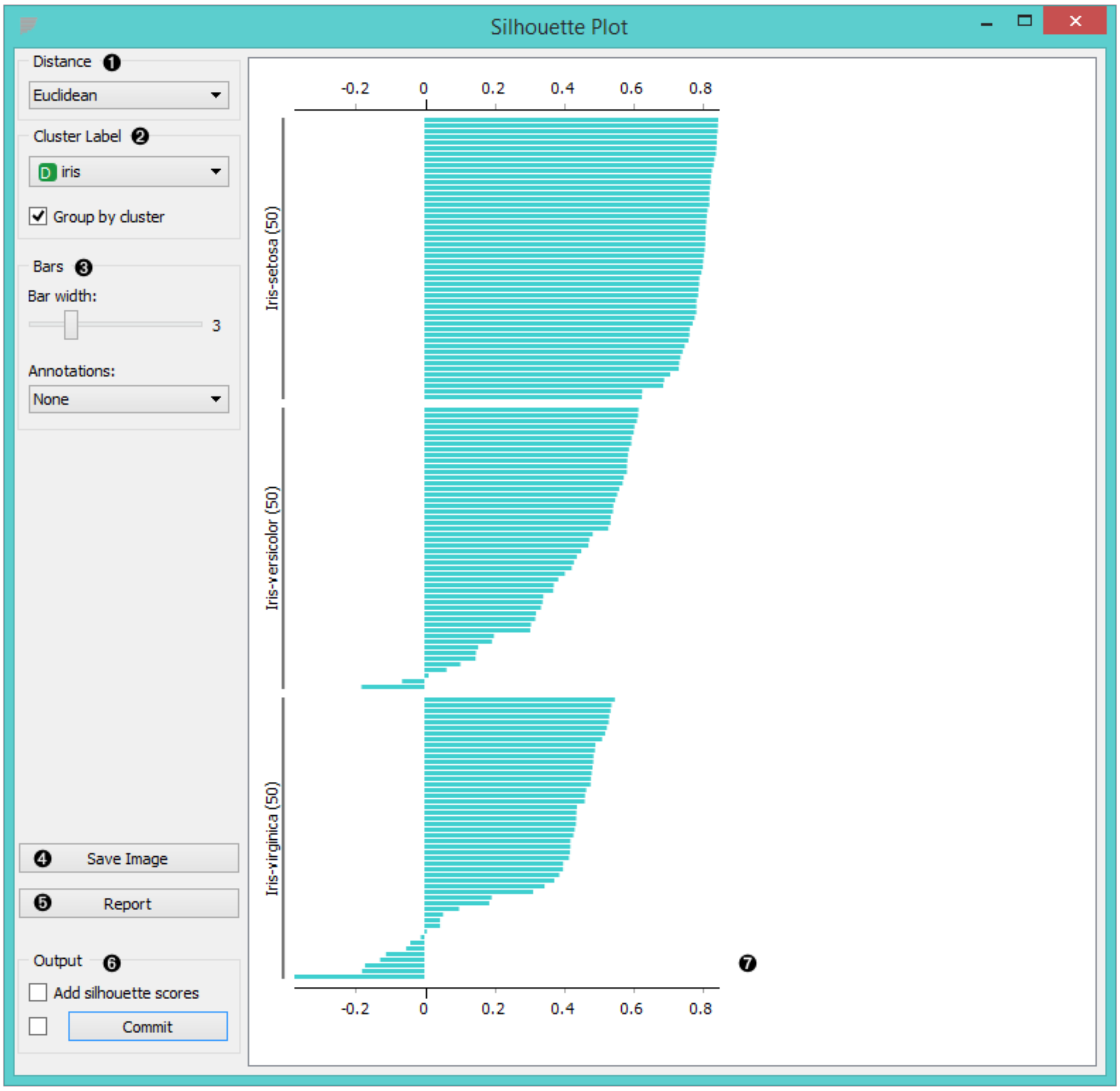
  A subset of instances that the user has manually selected from the plot.

- **Other Data**

  Remaining data.

Description

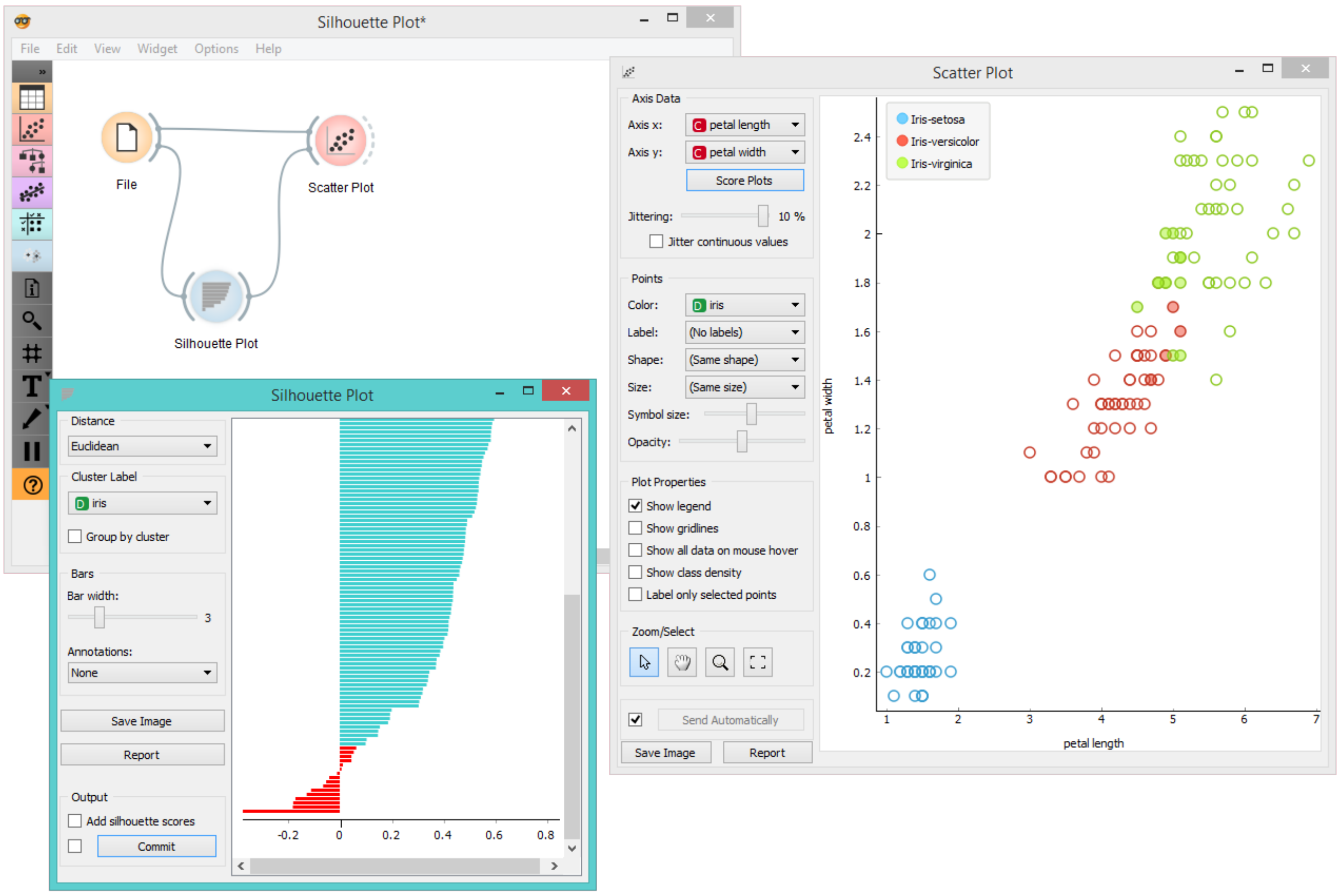
The **Silhouette Plot** widget offers a graphical representation of consistency within clusters of data and provides the user with the means to visually assess cluster quality. The silhouette score is a measure of how similar an object is to its own cluster in comparison to other clusters and is crucial in the creation of a silhoutte plot. The silhouette score close to 1 indicates that the data instance is close to the center of the cluster and instances posessing the silhouette scores close to 0 are on the border between two clusters.



1. Choose the distance metric. You can choose between:
   - Euclidean ("straight line", distance between two points)
   - Manhattan (the sum of absolute differences for all attributes)
2. Select the cluster label. You can decide whether to group the instances by cluster or not.
3. Display options:
   - *Choose bar width.*
   - *Annotations*: annotate the silhouette plot.
4. *Save Image* saves the created silhouette plot to your computer in a *.png* or *.svg* format.
5. Produce a report.
6. Output:
   - *Add silhouette scores* (good clusters have higher silhoutte scores)
   - By clicking *Commit*, changes are comminicated to the output of the widget. Alternatively, tick the box on the left and changes will be communicated automatically.
7. The created silhouette plot.

Example

In the snapshot below, we have decided to use the **Silhoutte Plot** on the *iris* data set. We selected data intances with low silhouette scores and passed them on as a subset to the Scatter Plot widget. This visualization only confirms the accuracy of the **Silhouette Plot** widget, as you can clearly see that the subset lies in the border between two clusters.



Silhouette Plot

Distance: Euclidean
Cluster Label: iris
Group by cluster: ☐
Bars — Bar width: 3
Annotations: None
Output — Add silhouette scores: ☐

Scatter Plot

Axis Data — Axis x: petal length; Axis y: petal width
Jittering: 10 %
Jitter continuous values: ☐
Points — Color: iris; Label: (No labels); Shape: (Same shape); Size: (Same size)
Plot Properties — Show legend: ✓; Show gridlines: ☐; Show all data on mouse hover: ☐; Show class density: ☐; Label only selected points: ☐

If you are interested in other uses of the **Silhouette Plot** widget, feel free to explore our blog post.

Tree Viewer

A visualization of classification and regression trees.

Signals

Inputs:

- **Tree**

  A decision tree.

Outputs:

- **Selected Data**

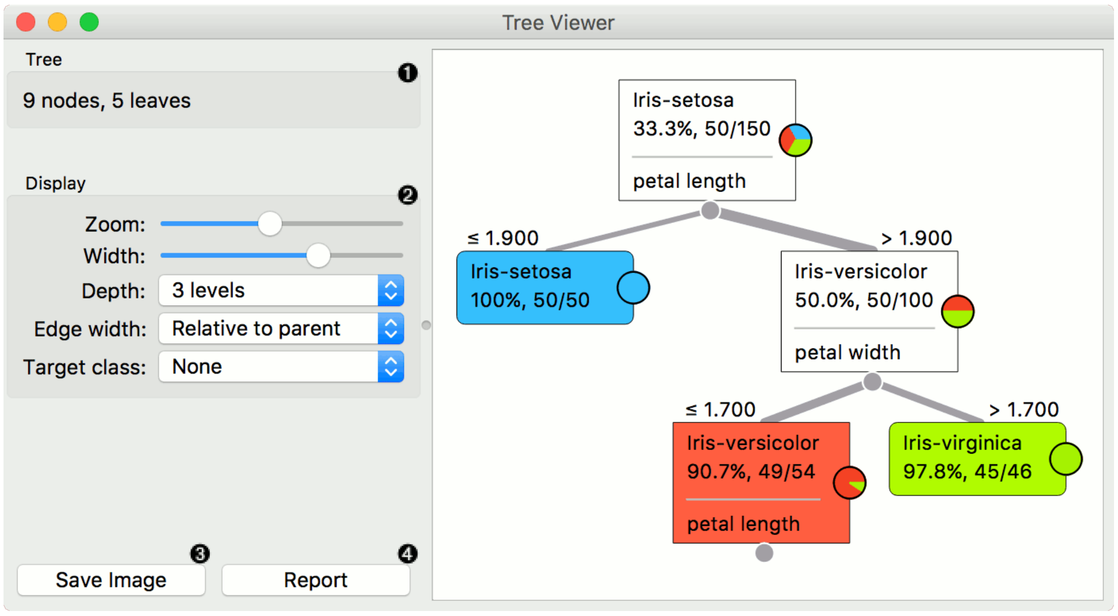
  Data from a selected tree node.

- **Data**

  Data set with an additional attribute for selection labels.

Description

This is a versatile widget with 2-D visualization of classification and regression trees. The user can select a node, instructing the widget to output the data associated with the node, thus enabling explorative data analysis.

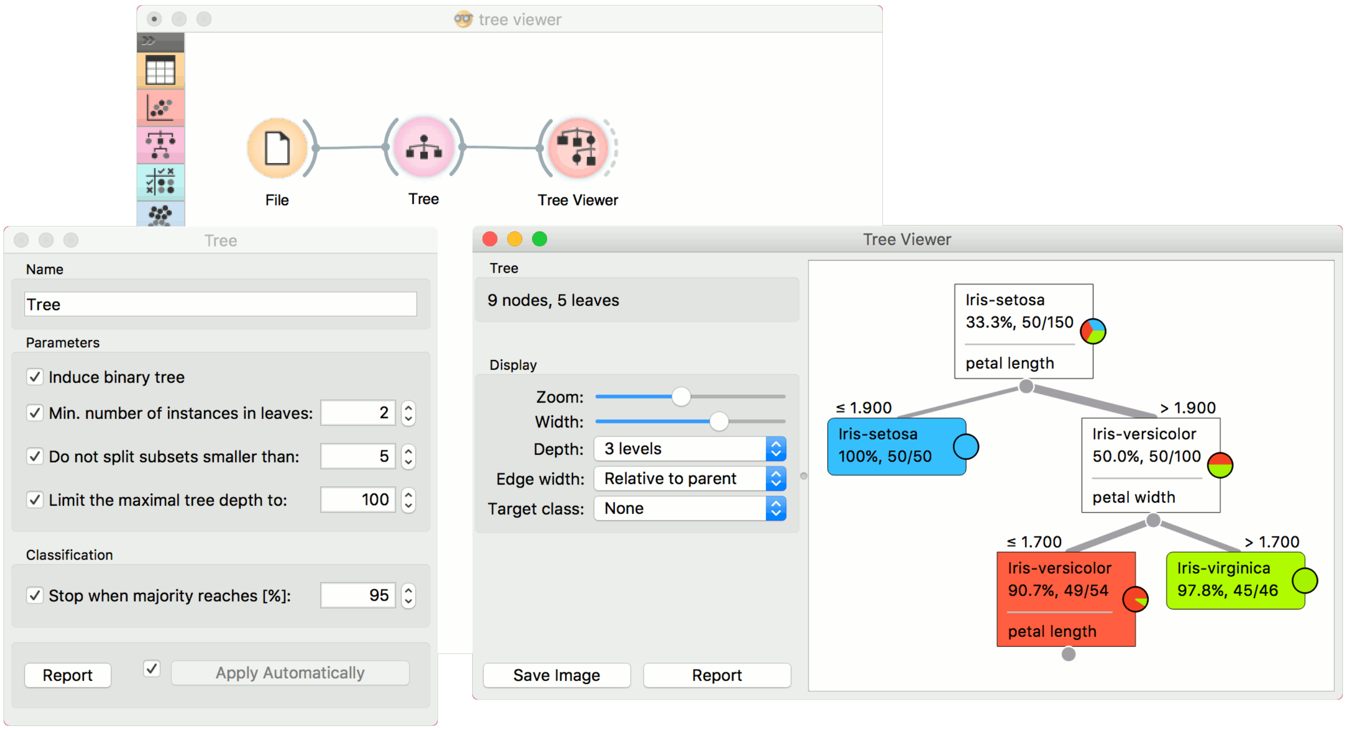


1. Information on the input.
2. Display options:
   - Zoom in and zoom out
   - Select the tree width. The nodes display information bubbles when hovering over them.
   - Select the depth of your tree.
   - Select edge width. The edges between the nodes in the tree graph are drawn based on the selected edge width.
     - All the edges will be of equal width if *Fixed* is chosen.
     - When *Relative to root* is selected, the width of the edge will correspond to the proportion of instances

in the corresponding node with respect to all the instances in the training data. Under this selection, the edge will get thinner and thinner when traversing toward the bottom of the tree.
        - *Relative to parent* makes the edge width correspond to the proportion of instances in the nodes with respect to the instances in their parent node.
    - Define the target class, which you can change based on classes in the data.
3. Press *Save image* to save the created tree graph to your computer as a *.svg* or *.png* file.
4. Produce a report.

Examples

Below, is a simple classification schema, where we have read the data, constructed the decision tree and viewed it in our **Tree Viewer**. If both the viewer and Tree are open, any re-run of the tree induction algorithm will immediately affect the visualization. You can thus use this combination to explore how the parameters of the induction algorithm influence the structure of the resulting tree.



Tree

Name: Tree

Parameters
- ✓ Induce binary tree
- ✓ Min. number of instances in leaves: 2
- ✓ Do not split subsets smaller than: 5
- ✓ Limit the maximal tree depth to: 100

Classification
- ✓ Stop when majority reaches [%]: 95

Report ✓ Apply Automatically

Tree Viewer

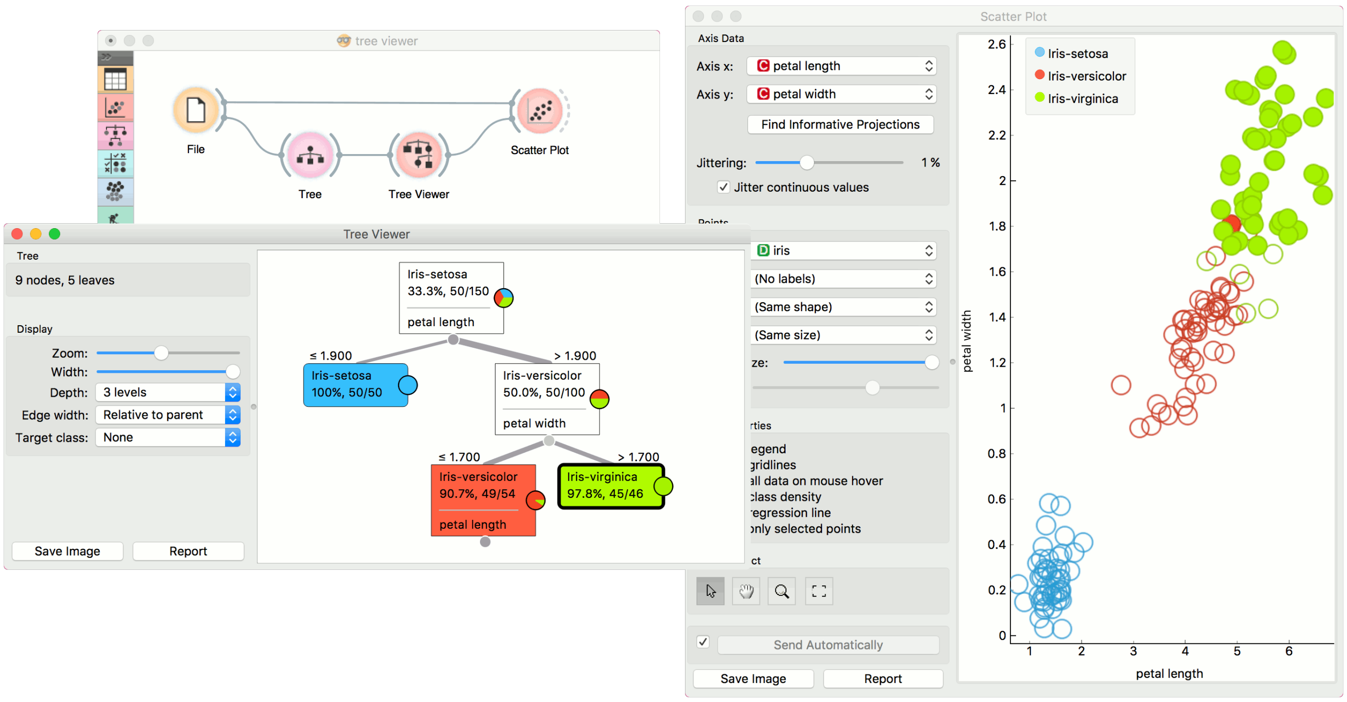
Tree: 9 nodes, 5 leaves

Display
- Zoom:
- Width:
- Depth: 3 levels
- Edge width: Relative to parent
- Target class: None

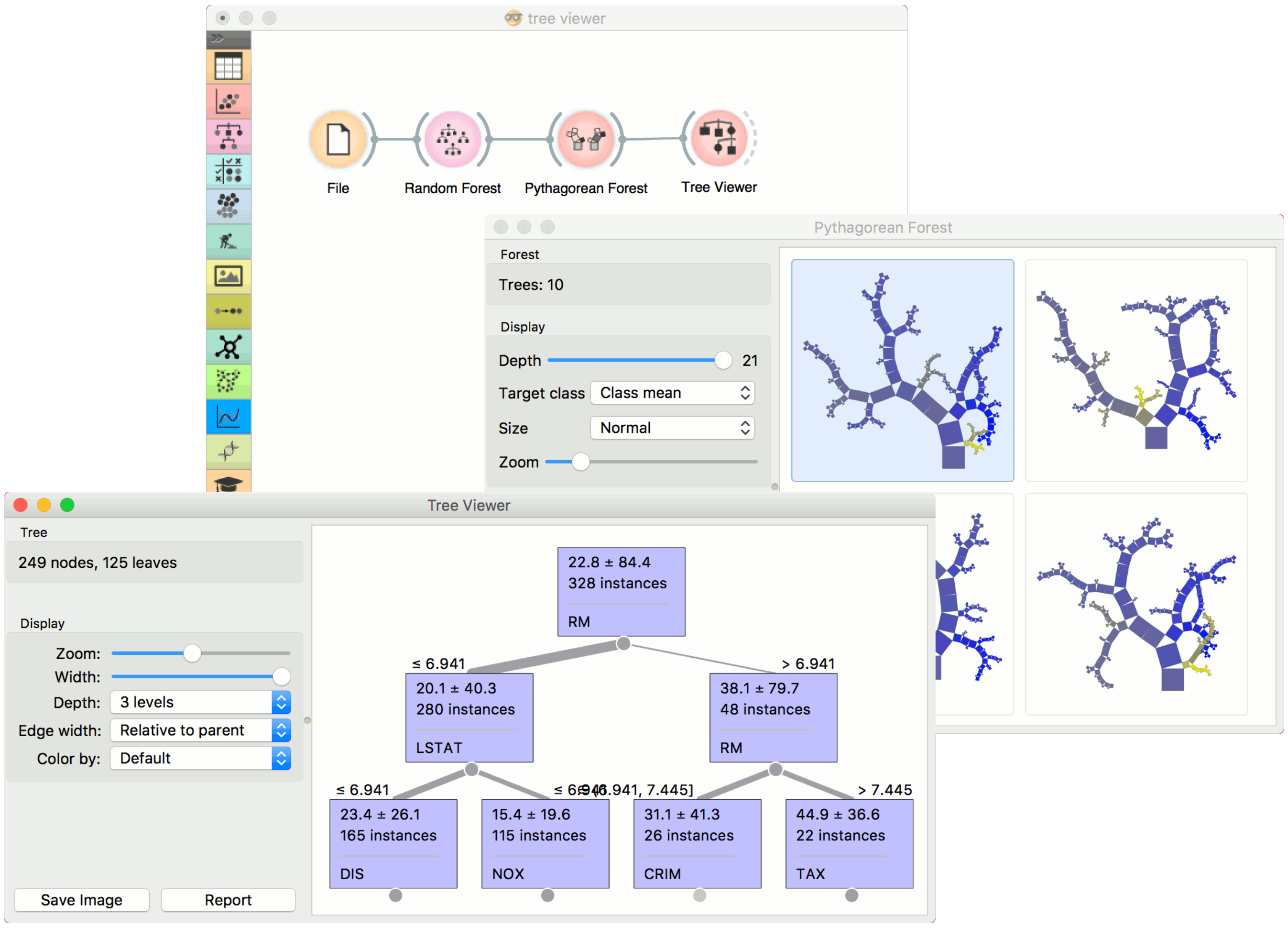
Save Image Report

Clicking on any node will output the related data instances. This is explored in the schema below that shows the subset in the data table and in the Scatterplot. Make sure that the tree data is passed as a data subset; this can be done by connecting the **Scatterplot** to the File widget first, and connecting it to the **Tree Viewer** widget next. Selected data will be displayed as bold dots.

Tree Viewer can also export labelled data. Connect Data Table to **Tree Viewer** and set the link between widgets to *Data* instead of *Selected Data*. This will send the entire data to **Data Table** with an additional meta column labelling selected data instances (*Yes* for selected and *No* for the remaining).



Finally, **Tree Viewer** can be used also for visualizing regression trees. Connect Random Forest to File widget using *housing.tab* data set. Then connect Pythagorean Forest to **Random Forest**. In **Pythagorean Forest** select a regression tree you wish to further analyze and pass it to the **Tree Viewer**. The widget will display the constructed tree. For visualizing larger trees, especially for regression, Pythagorean Tree could be a better option.



Venn Diagram

Plots a Venn diagram for two or more data subsets.

Signals

Inputs:

- **Data**

  An input data set

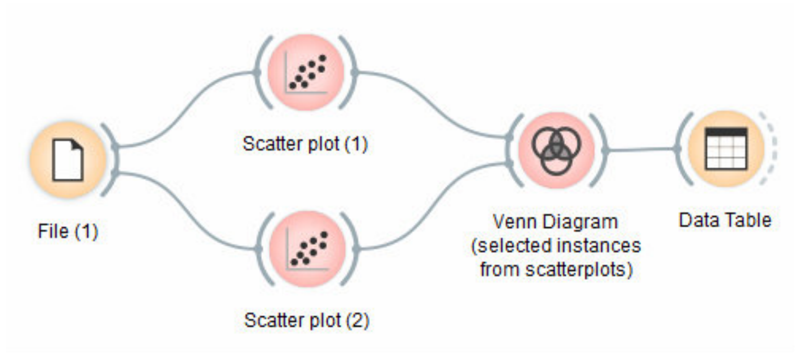
Outputs:

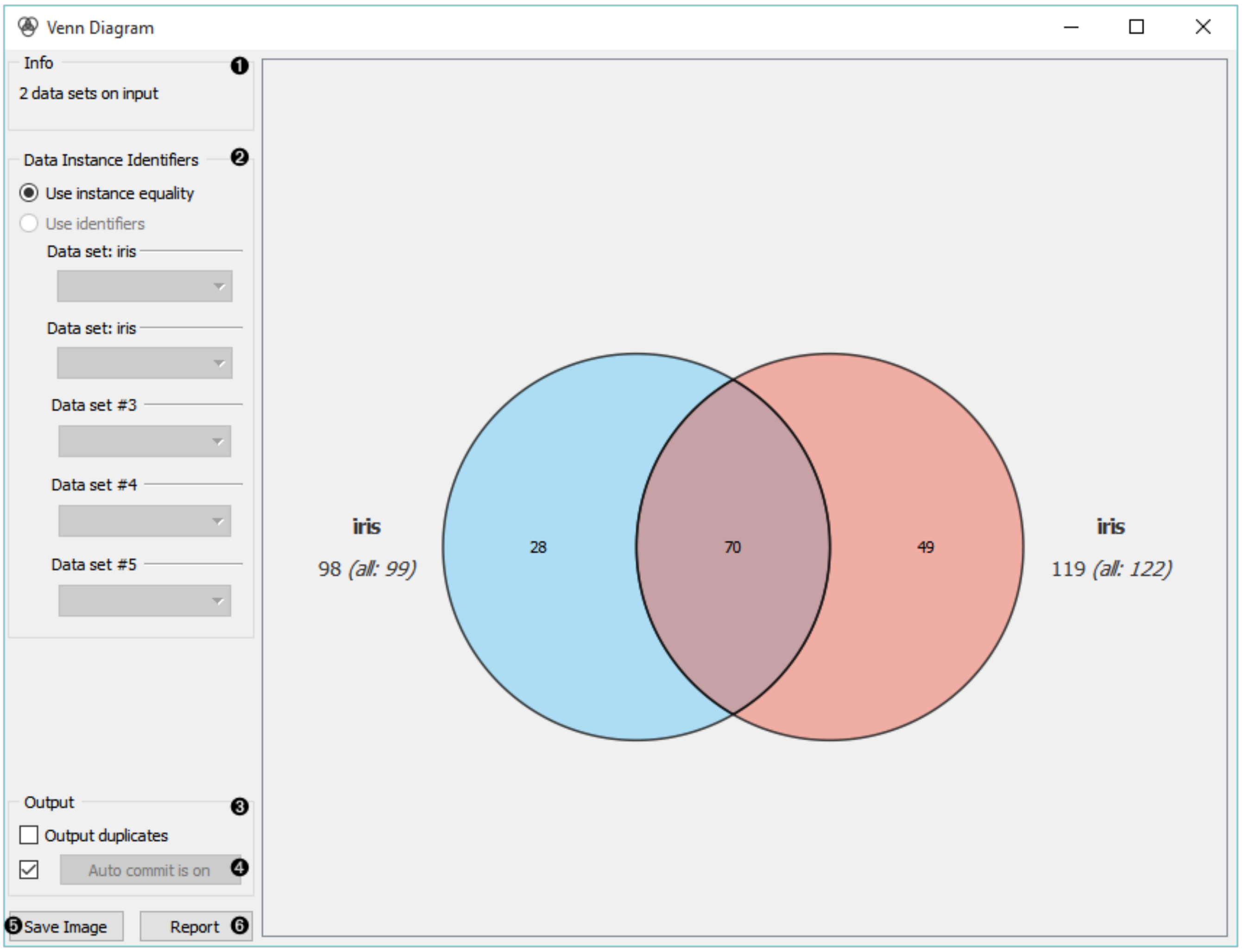
- **Selected Data**

  A subset of instances that the user has manually selected from the diagram.

Description

The **Venn Diagram** widget displays logical relations between data sets. This projection shows two or more data sets represented by circles of different colors. The intersections are subsets that belong to more than one data set. To further analyze or visualize the subset, click on the intersection.

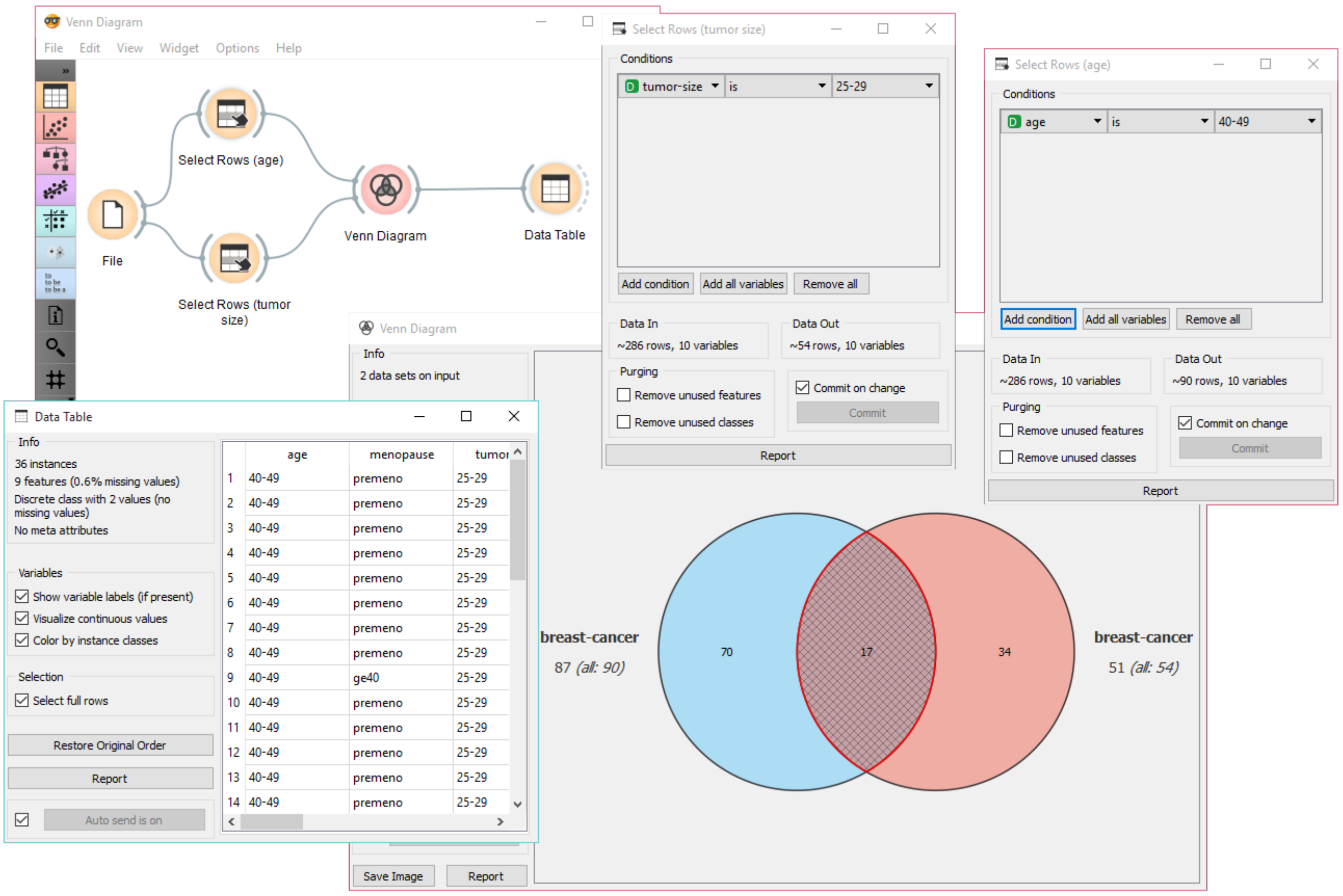




1. Information on the input data.
2. Select the identifiers by which to compare the data.
3. Tick *Output duplicates* if you wish to remove duplicates.
4. If *Auto commit* is on, changes are automatically communicated to other widgets. Alternatively, click *Commit*.
5. *Save Image* saves the created image to your computer in a .svg or .png format.
6. Produce a report.

Examples

The easiest way to use the **Venn Diagram** is to select data subsets and find matching instances in the visualization. We use the *breast-cancer* data set to select two subsets with Select Rows widget - the first subset is that of breast cancer patients aged between 40 and 49 and the second is that of patients with a tumor size between 20 and 29. The **Venn Diagram** helps us find instances that correspond to both criteria, which can be found in the intersection of the two circles.



Select Rows (tumor size)

Conditions: tumor-size is 25-29

Add condition | Add all variables | Remove all

Data In: ~286 rows, 10 variables

Data Out: ~54 rows, 10 variables

Purging: ☐ Remove unused features ☐ Remove unused classes

☑ Commit on change | Commit

Report

Select Rows (age)

Conditions: age is 40-49

Add condition | Add all variables | Remove all

Data In: ~286 rows, 10 variables

Data Out: ~90 rows, 10 variables

Purging: ☐ Remove unused features ☐ Remove unused classes

☑ Commit on change | Commit

Report

Venn Diagram

Info: 2 data sets on input

Data Table

Info: 36 instances; 9 features (0.6% missing values); Discrete class with 2 values (no missing values); No meta attributes

Variables: ☑ Show variable labels (if present) ☑ Visualize continuous values ☑ Color by instance classes

Selection: ☑ Select full rows

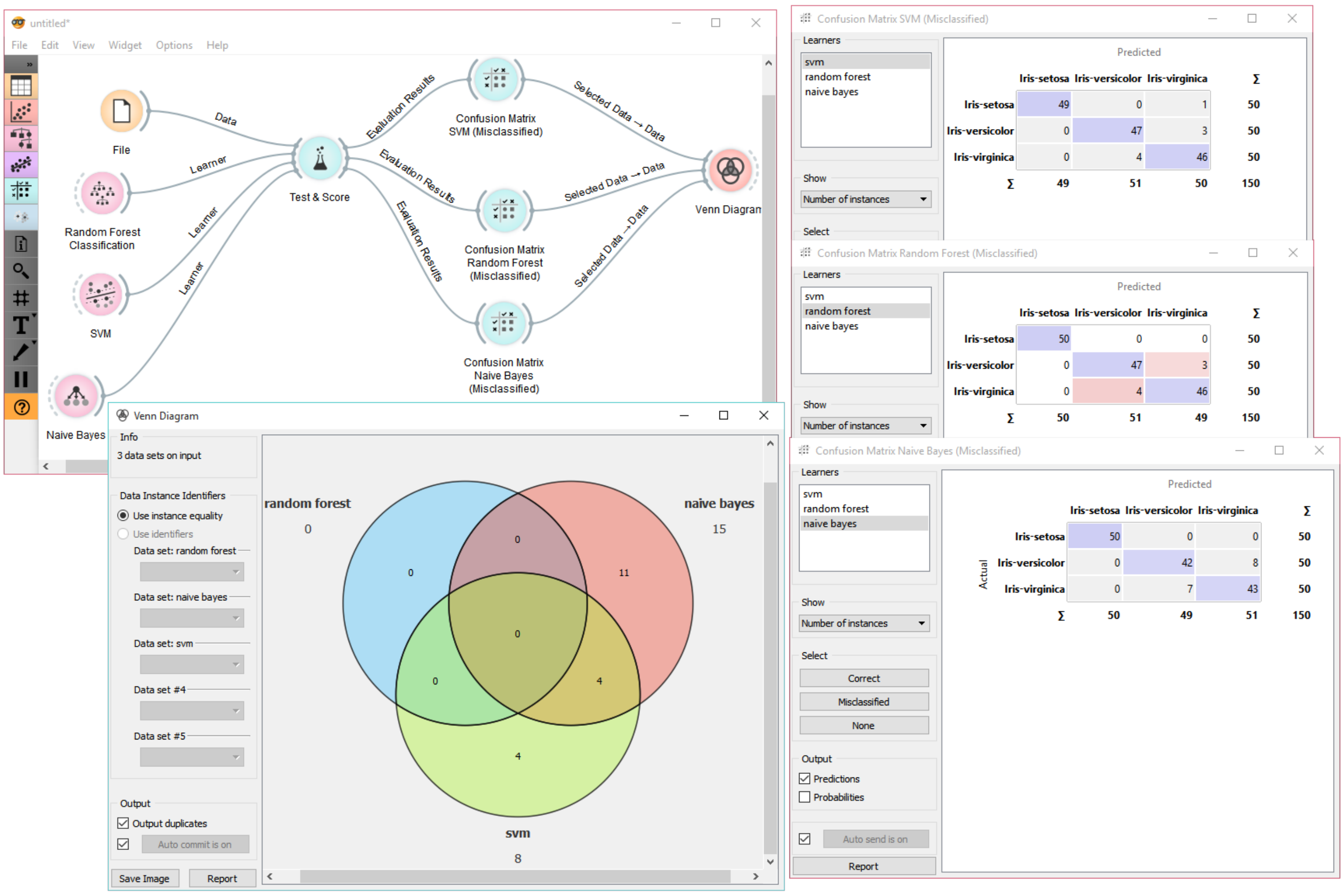
Restore Original Order

Report

☑ Auto send is on

	age	menopause	tumor
1	40-49	premeno	25-29
2	40-49	premeno	25-29
3	40-49	premeno	25-29
4	40-49	premeno	25-29
5	40-49	premeno	25-29
6	40-49	premeno	25-29
7	40-49	premeno	25-29
8	40-49	premeno	25-29
9	40-49	ge40	25-29
10	40-49	premeno	25-29
11	40-49	premeno	25-29
12	40-49	premeno	25-29
13	40-49	premeno	25-29
14	40-49	premeno	25-29

The **Venn Diagram** widget can be also used for exploring different prediction models. In the following example, we analysed 3 prediction methods, namely Naive Bayes, SVM and Random Forest, according to their misclassified instances. By selecting misclassifications in the three Confusion Matrix widgets and sending them to Venn diagram, we can see all the misclassification instances visualized per method used. Then we open **Venn Diagram** and select, for example, the misclassified instances that were identified by all three methods (in our case 2). This is represented as an intersection of all three circles. Click on the intersection to see this two instances marked in the Scatterplot widget. Try selecting different diagram sections to see how the scatterplot visualization changes.



Confusion Matrix SVM (Misclassified)

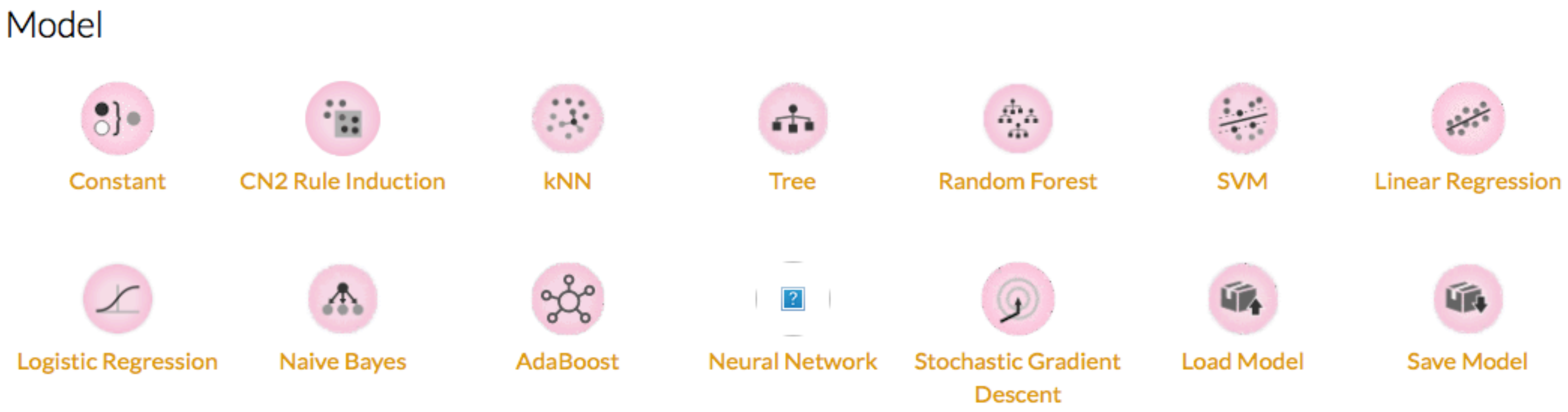
	Iris-setosa	Iris-versicolor	Iris-virginica	Σ
Iris-setosa	49	0	1	50
Iris-versicolor	0	47	3	50
Iris-virginica	0	4	46	50
Σ	49	51	50	150

Confusion Matrix Random Forest (Misclassified)

	Iris-setosa	Iris-versicolor	Iris-virginica	Σ
Iris-setosa	50	0	0	50
Iris-versicolor	0	47	3	50
Iris-virginica	0	4	46	50
Σ	50	51	49	150

Confusion Matrix Naive Bayes (Misclassified)

	Iris-setosa	Iris-versicolor	Iris-virginica	Σ
Iris-setosa	50	0	0	50
Iris-versicolor	0	42	8	50
Iris-virginica	0	7	43	50
Σ	50	49	51	150


Model
Constant
CN2 Rule Induction
kNN
Tree
Random Forest
SVM
Linear Regression
Logistic Regression
Naive Bayes
AdaBoost
Neural Network
Stochastic Gradient Descent
Load Model
Save Model

AdaBoost

An ensemble meta-algorithm that combines weak learners and adapts to the ‘hardness’ of each training sample.

Signals

Inputs:

- **Data**

  A data set.

- **Preprocessor**

  Preprocessing method(s)

- **Learner**

  A learning algorithm.

Outputs:

- **Learner**

  AdaBoost learning algorithm with settings as specified in the dialog.

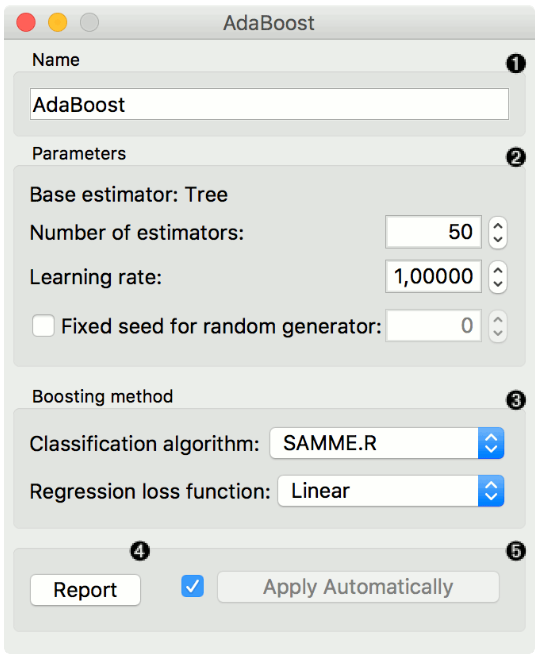
- **Model**

  A trained model. Output signal sent only if input *Data* is present.

Description

The AdaBoost (short for “Adaptive boosting”) widget is a machine-learning algorithm, formulated by Yoav Freund and Robert Schapire. It can be used with other learning algorithms to boost their performance. It does so by tweaking the weak learners.

AdaBoost works for both classification and regression.


Name

AdaBoost

Parameters

Base estimator: Tree

Number of estimators: 50

Learning rate: 1,00000

Fixed seed for random generator: 0

Boosting method

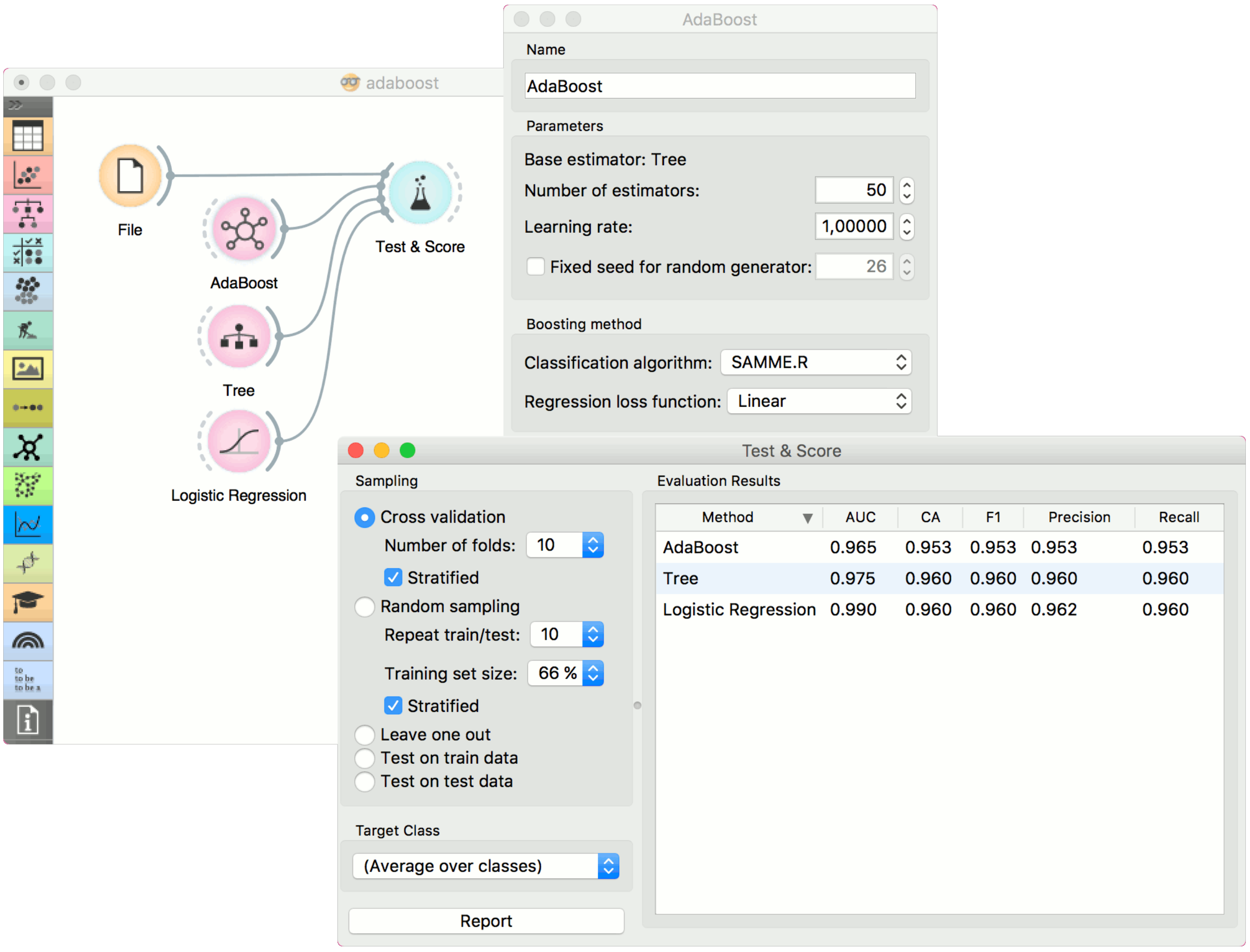
Classification algorithm: SAMME.R

Regression loss function: Linear

1. The learner can be given a name under which it will appear in other widgets. The default name is “AdaBoost”.
2. Set the parameters. The base estimator is a tree and you can set:
   - *Number of estimators*
   - *Learning rate*: it determines to what extent the newly acquired information will override the old information (0 = the agent will not learn anything, 1 = the agent considers only the most recent information)
   - *Fixed seed for random generator*: set a fixed seed to enable reproducing the results.
3. Boosting method.
   - *Classification algorithm* (if classification on input): SAMME (updates base estimator’s weights with classification results) or SAMME.R (updates base estimator’s weight with probability estimates).
   - *Regression loss function* (if regression on input): Linear (), Square (), Exponential ().
4. Produce a report.
5. Click *Apply* after changing the settings. That will put the new learner in the output and, if the training examples are given, construct a new model and output it as well. To communicate changes automatically tick *Apply Automatically*.

Examples

For classification, we loaded the *iris* data set. We used *AdaBoost*, Tree and Logistic Regression and evaluated the models’ performance in Test & Score.



AdaBoost

Name: AdaBoost

Parameters

Base estimator: Tree

Number of estimators: 50

Learning rate: 1,00000

Fixed seed for random generator: 26

Boosting method

Classification algorithm: SAMME.R

Regression loss function: Linear

Test & Score

Sampling

- [✓ Cross validation]
  - Number of folds: 10
  - [✓ Stratified]
- Random sampling
  - Repeat train/test: 10
  - Training set size: 66 %
  - [✓ Stratified]
- Leave one out
- Test on train data
- Test on test data

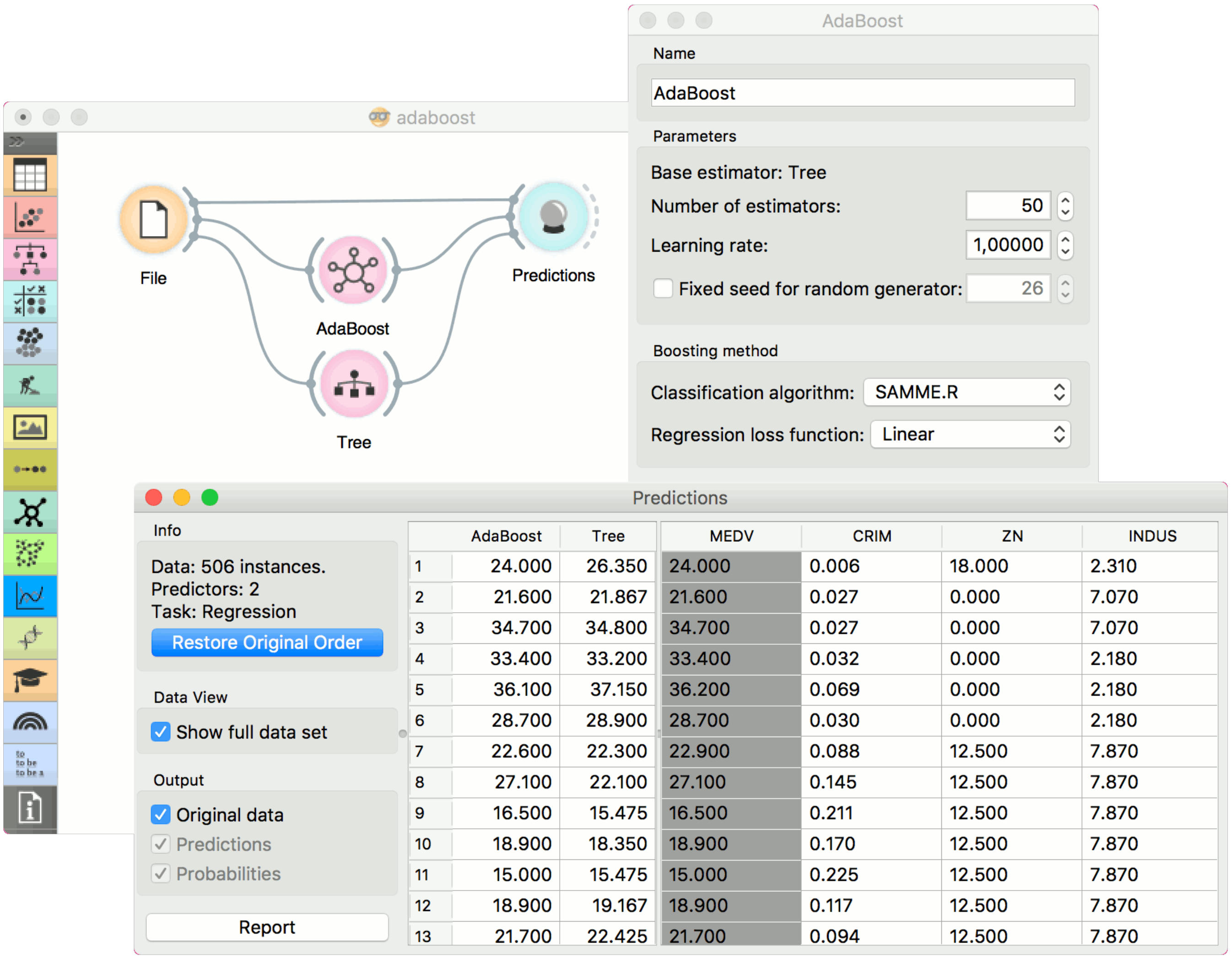
Target Class: (Average over classes)

Report

Evaluation Results

Method	AUC	CA	F1	Precision	Recall
AdaBoost	0.965	0.953	0.953	0.953	0.953
Tree	0.975	0.960	0.960	0.960	0.960
Logistic Regression	0.990	0.960	0.960	0.962	0.960

For regression, we loaded the *housing* data set, sent the data instances to two different models (**AdaBoost** and Tree) and output them to the Predictions widget.



AdaBoost

Name

AdaBoost

Parameters

Base estimator: Tree

Number of estimators: 50

Learning rate: 1,00000

Fixed seed for random generator: 26

Boosting method

Classification algorithm: SAMME.R

Regression loss function: Linear

Predictions

Info

Data: 506 instances.
Predictors: 2
Task: Regression

Restore Original Order

Data View

✓ Show full data set

Output

✓ Original data
✓ Predictions
✓ Probabilities

Report

	AdaBoost	Tree	MEDV	CRIM	ZN	INDUS
1	24.000	26.350	24.000	0.006	18.000	2.310
2	21.600	21.867	21.600	0.027	0.000	7.070
3	34.700	34.800	34.700	0.027	0.000	7.070
4	33.400	33.200	33.400	0.032	0.000	2.180
5	36.100	37.150	36.200	0.069	0.000	2.180
6	28.700	28.900	28.700	0.030	0.000	2.180
7	22.600	22.300	22.900	0.088	12.500	7.870
8	27.100	22.100	27.100	0.145	12.500	7.870
9	16.500	15.475	16.500	0.211	12.500	7.870
10	18.900	18.350	18.900	0.170	12.500	7.870
11	15.000	15.475	15.000	0.225	12.500	7.870
12	18.900	19.167	18.900	0.117	12.500	7.870
13	21.700	22.425	21.700	0.094	12.500	7.870

CN2 Rule Induction

Induce rules from data using CN2 algorithm.

Signals

Inputs

- **Data**

  Data set.

- **Preprocessor**

  Preprocessing method(s)

Outputs

- **Learner**

  The CN2 learning algorithm with settings as specified in the dialog.

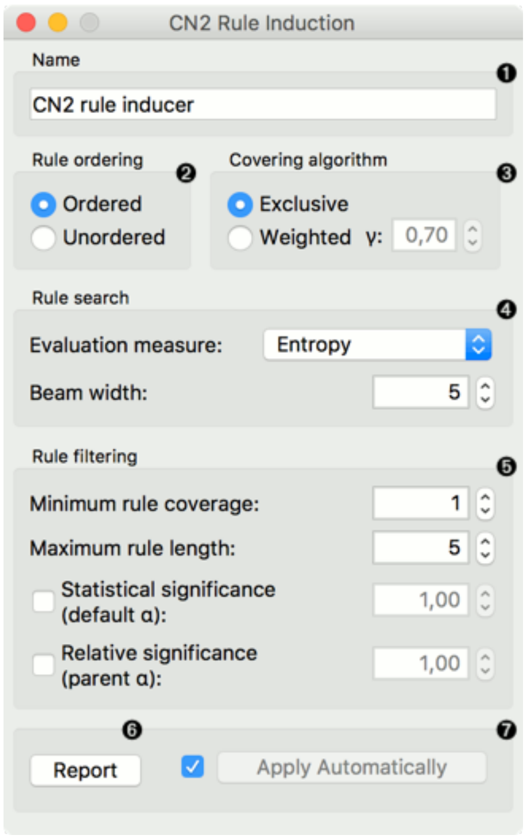
- **CN2 Rule Classifier**

  A trained model. Output signal sent only if input *Data* is present.

Description

The CN2 algorithm is a classification technique designed for the efficient induction of simple, comprehensible rules of form "if *cond* then predict *class*", even in domains where noise may be present.

CN2 Rule Induction works only for classification.



Name: CN2 rule inducer

Rule ordering: [✓ Ordered] Unordered

Covering algorithm: [✓ Exclusive] Weighted γ: 0,70

Rule search

Evaluation measure: Entropy

Beam width: 5

Rule filtering

Minimum rule coverage: 1

Maximum rule length: 5

Statistical significance (default α): 1,00

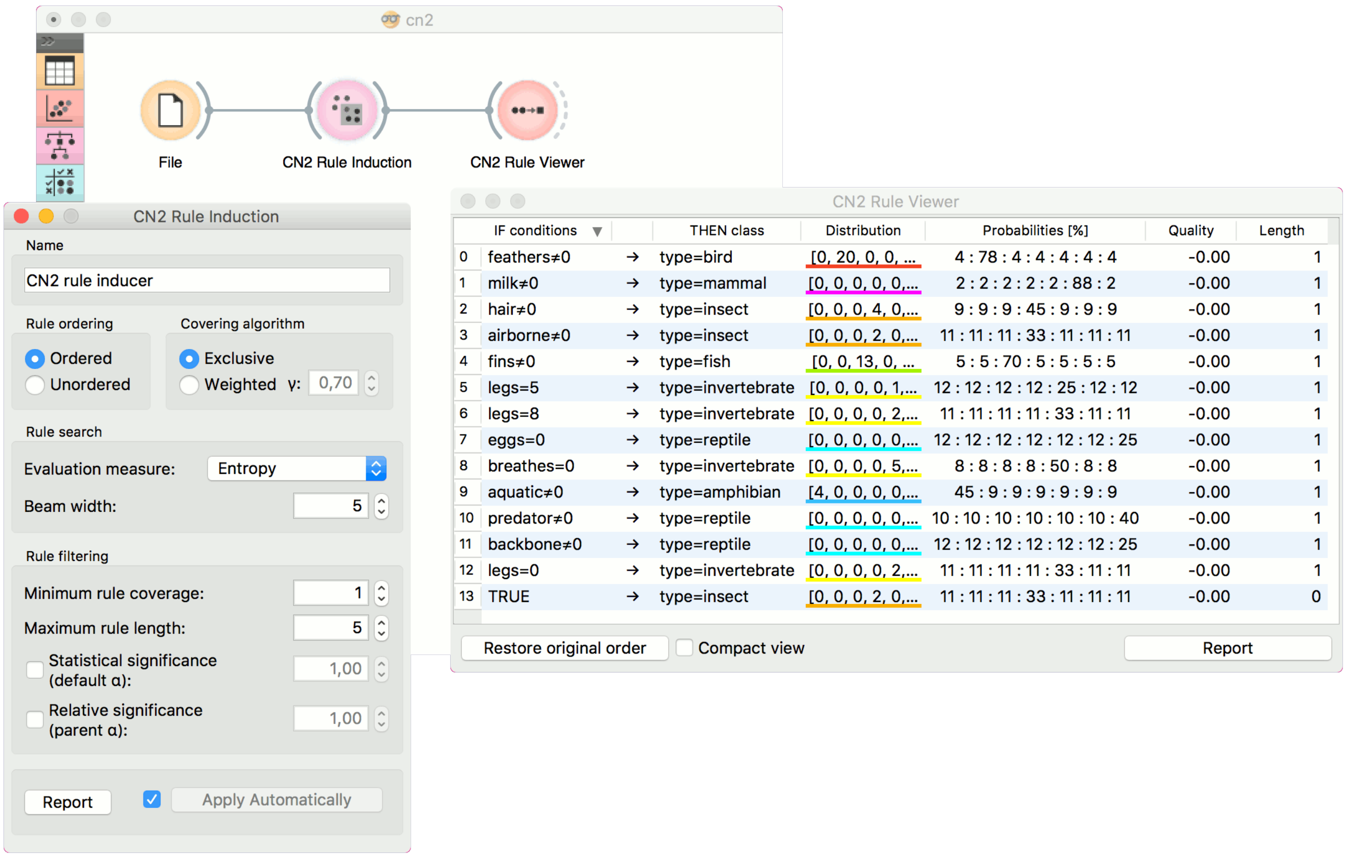
Relative significance (parent α): 1,00

[✓ Apply Automatically]

1. Name under which the learner appears in other widgets. The default name is *CN2 Rule Induction*.
2. *Rule ordering*:
   - **Ordered**: induce ordered rules (decision list). Rule conditions are found and the majority class is assigned in the rule head.
   - **Unordered**: induce unordered rules (rule set). Learn rules for each class individually, in regard to the original learning data.
3. *Covering algorithm*:
   - **Exclusive**: after covering a learning instance, remove it from further consideration.
   - **Weighted**: after covering a learning instance, decrease its weight (multiplication by *gamma*) and in-turn decrease its impact on further iterations of the algorithm.
4. *Rule search*:
   - **Evaluation measure**: select a heuristic to evaluate found hypotheses:
     a. Entropy (measure of unpredictability of content)
     b. Laplace Accuracy
     c. Weighted Relative Accuracy
   - **Beam width**; remember the best rule found thus far and monitor a fixed number of alternatives (the beam).
5. *Rule filtering*:
   - **Minimum rule coverage**: found rules must cover at least the minimum required number of covered examples. Unordered rules must cover this many target class examples.
   - **Maximum rule length**: found rules may combine at most the maximum allowed number of selectors (conditions).
   - **Default alpha**: significance testing to prune out most specialised (less frequently applicable) rules in regard to the initial distribution of classes.
   - **Parent alpha**: significance testing to prune out most specialised (less frequently applicable) rules in regard to the parent class distribution.
6. Tick 'Apply Automatically' to auto-communicate changes to other widgets and to immediately train the classifier if learning data is connected. Alternatively, press 'Apply' after configuration.

Examples

For the example below, we have used *zoo* data set and passed it to **CN2 Rule Induction**. We can review and interpret the built model with CN2 Rule Viewer widget.



CN2 Rule Induction

Name: CN2 rule inducer

Rule ordering: [✓ Ordered] Unordered

Covering algorithm: [✓ Exclusive] Weighted γ: 0,70

Rule search

Evaluation measure: Entropy

Beam width: 5

Rule filtering

Minimum rule coverage: 1

Maximum rule length: 5

☐ Statistical significance (default α): 1,00

☐ Relative significance (parent α): 1,00

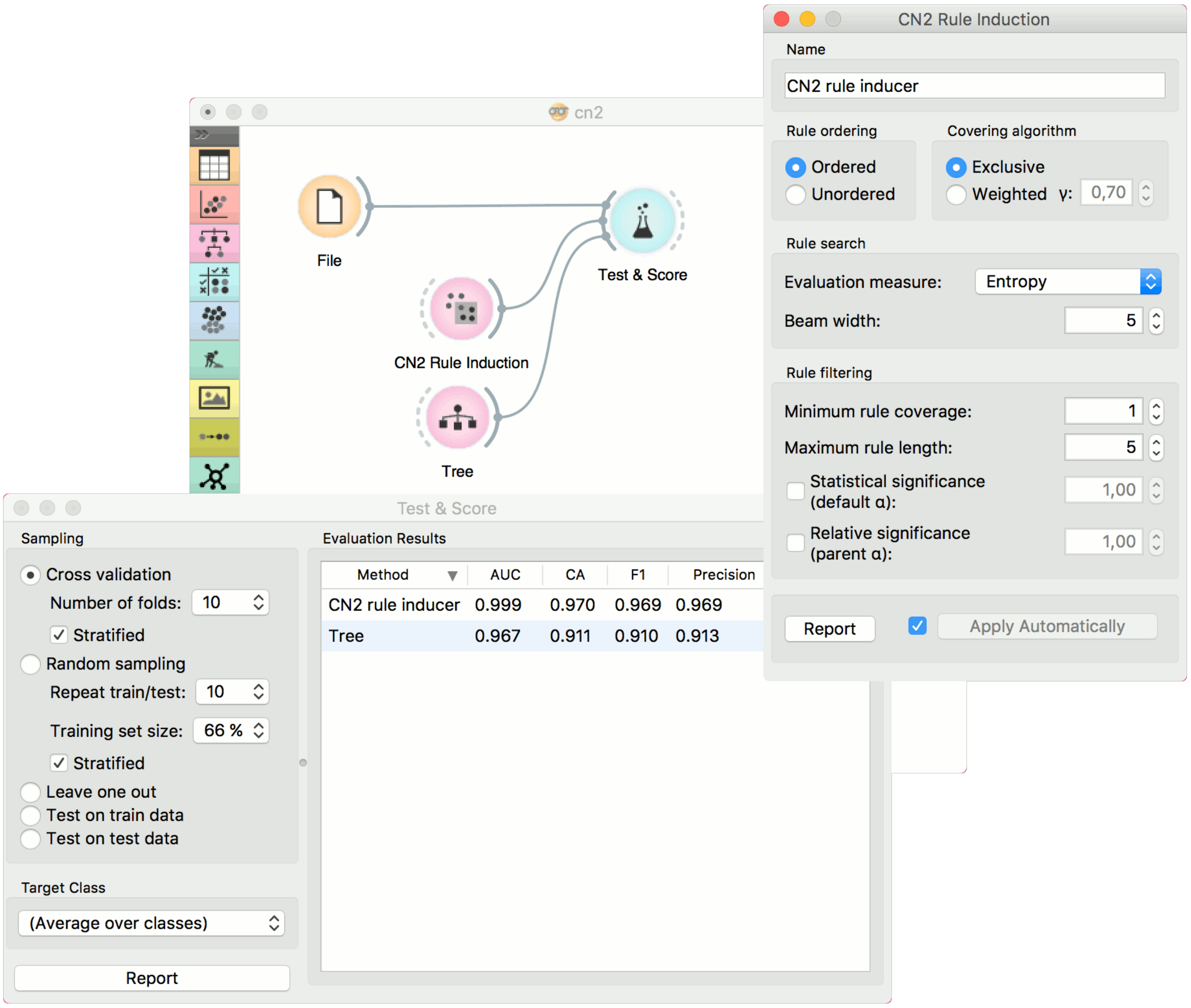
Report ☑ Apply Automatically

CN2 Rule Viewer

	IF conditions		THEN class	Distribution	Probabilities [%]	Quality	Length
0	feathers≠0	→	type=bird	[0, 20, 0, 0, ...	4 : 78 : 4 : 4 : 4 : 4 : 4	-0.00	1
1	milk≠0	→	type=mammal	[0, 0, 0, 0, 0,...	2 : 2 : 2 : 2 : 2 : 88 : 2	-0.00	1
2	hair≠0	→	type=insect	[0, 0, 0, 4, 0,...	9 : 9 : 9 : 45 : 9 : 9 : 9	-0.00	1
3	airborne≠0	→	type=insect	[0, 0, 0, 2, 0,...	11 : 11 : 11 : 33 : 11 : 11 : 11	-0.00	1
4	fins≠0	→	type=fish	[0, 0, 13, 0, ...	5 : 5 : 70 : 5 : 5 : 5 : 5	-0.00	1
5	legs=5	→	type=invertebrate	[0, 0, 0, 0, 1,...	12 : 12 : 12 : 12 : 25 : 12 : 12	-0.00	1
6	legs=8	→	type=invertebrate	[0, 0, 0, 0, 2,...	11 : 11 : 11 : 11 : 33 : 11 : 11	-0.00	1
7	eggs=0	→	type=reptile	[0, 0, 0, 0, 0,...	12 : 12 : 12 : 12 : 12 : 12 : 25	-0.00	1
8	breathes=0	→	type=invertebrate	[0, 0, 0, 0, 5,...	8 : 8 : 8 : 8 : 50 : 8 : 8	-0.00	1
9	aquatic≠0	→	type=amphibian	[4, 0, 0, 0, 0,...	45 : 9 : 9 : 9 : 9 : 9 : 9	-0.00	1
10	predator≠0	→	type=reptile	[0, 0, 0, 0, 0,...	10 : 10 : 10 : 10 : 10 : 10 : 40	-0.00	1
11	backbone≠0	→	type=reptile	[0, 0, 0, 0, 0,...	12 : 12 : 12 : 12 : 12 : 12 : 25	-0.00	1
12	legs=0	→	type=invertebrate	[0, 0, 0, 0, 2,...	11 : 11 : 11 : 11 : 33 : 11 : 11	-0.00	1
13	TRUE	→	type=insect	[0, 0, 0, 2, 0,...	11 : 11 : 11 : 33 : 11 : 11 : 11	-0.00	0

Restore original order ☐ Compact view Report

The second workflow tests evaluates **CN2 Rule Induction** and Tree in Test & Score.



CN2 Rule Induction

Name: CN2 rule inducer

Rule ordering: [✓ Ordered] Unordered

Covering algorithm: [✓ Exclusive] Weighted γ: 0,70

Rule search

Evaluation measure: Entropy

Beam width: 5

Rule filtering

Minimum rule coverage: 1

Maximum rule length: 5

☐ Statistical significance (default α): 1,00

☐ Relative significance (parent α): 1,00

Report ☑ Apply Automatically

Test & Score

Sampling

[✓ Cross validation]

Number of folds: 10

☑ Stratified

Random sampling

Repeat train/test: 10

Training set size: 66 %

☑ Stratified

Leave one out

Test on train data

Test on test data

Target Class: (Average over classes)

Report

Evaluation Results

Method	AUC	CA	F1	Precision
CN2 rule inducer	0.999	0.970	0.969	0.969
Tree	0.967	0.911	0.910	0.913

References

1. "Separate-and-Conquer Rule Learning", Johannes Fürnkranz, Artificial Intelligence Review 13, 3-54, 1999
2. "The CN2 Induction Algorithm", Peter Clark and Tim Niblett, Machine Learning Journal, 3 (4), pp261-283, (1989)
3. "Rule Induction with CN2: Some Recent Improvements", Peter Clark and Robin Boswell, Machine Learning - Proceedings of the 5th European Conference (EWSL-91), pp151-163, 1991
4. "Subgroup Discovery with CN2-SD", Nada Lavrač et al., Journal of Machine Learning Research 5 (2004), 153-188, 2004

Constant

Predict the most frequent class or mean value from the training set.

Signals

Inputs:

- **Data**

  A data set

- **Preprocessor**

  Preprocessing method(s)

Outputs:

- **Learner**

  A majority/mean learning algorithm

- **Model**

  A trained model. Output signal sent only if input *Data* is present.

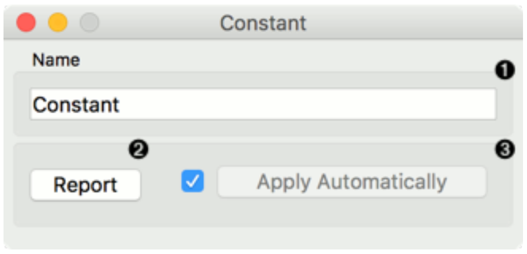
Description

This learner produces a model that always predicts the majority for classification tasks and mean value for regression tasks.

For classification, when predicting the class value with Predictions, the widget will return relative frequencies of the classes in the training set. When there are two or more majority classes, the classifier chooses the predicted class randomly, but always returns the same class for a particular example.

For regression, it *learns* the mean of the class variable and returns a predictor with the same mean value.

The widget is typically used as a baseline for other models.



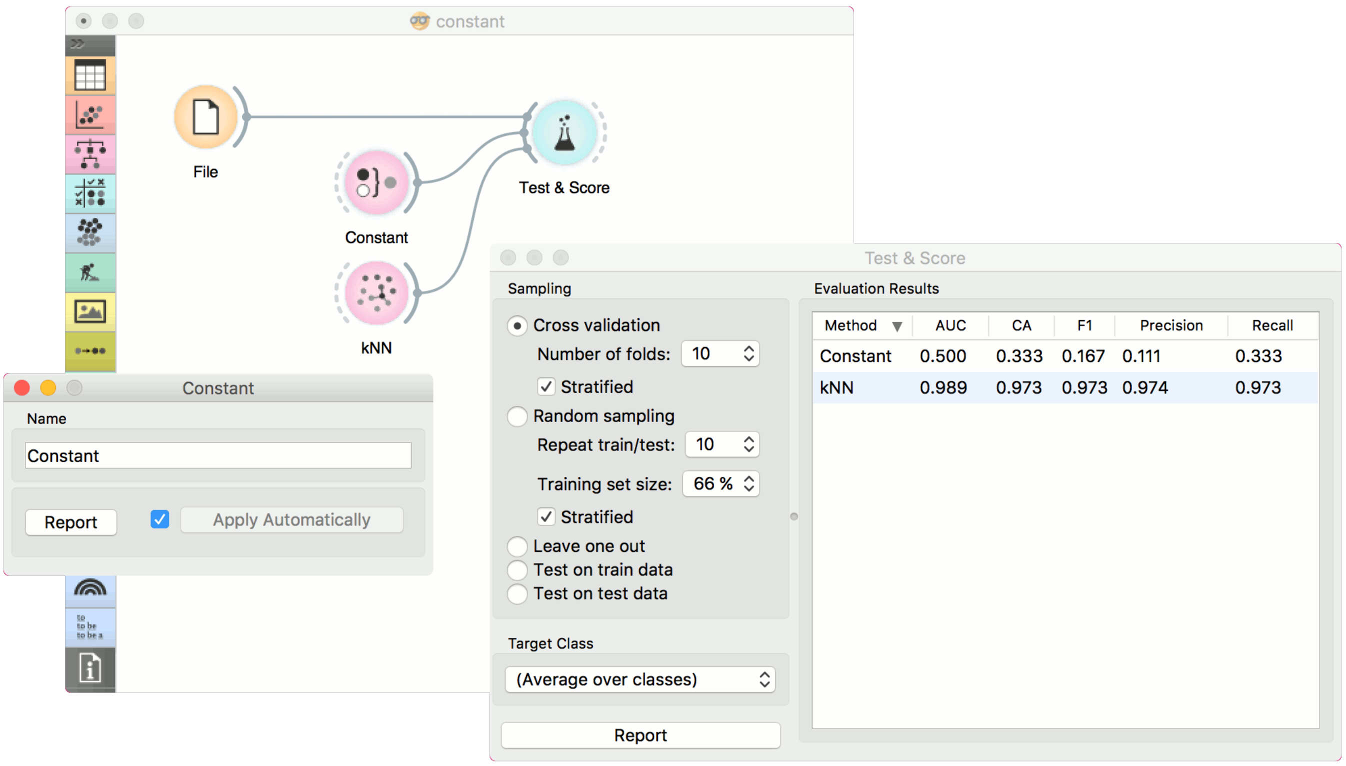
This widget provides the user with two options:

1. The name under which it will appear in other widgets. Default name is “Constant”.
2. Produce a report.

If you change the widget’s name, you need to click *Apply*. Alternatively, tick the box on the left side and changes will be communicated automatically.

Examples

In a typical classification example, we would use this widget to compare the scores of other learning algorithms (such as kNN) with the default scores. Use *iris* data set and connect it to Test & Score. Then connect **Constant** and kNN to Test & Score and observe how well kNN performs against a constant baseline.



Constant

Name: Constant

Report ☑ Apply Automatically

Test & Score

Sampling
- ● Cross validation
  - Number of folds: 10
  - ✓ Stratified
- Random sampling
  - Repeat train/test: 10
  - Training set size: 66 %
  - ✓ Stratified
- Leave one out
- Test on train data
- Test on test data

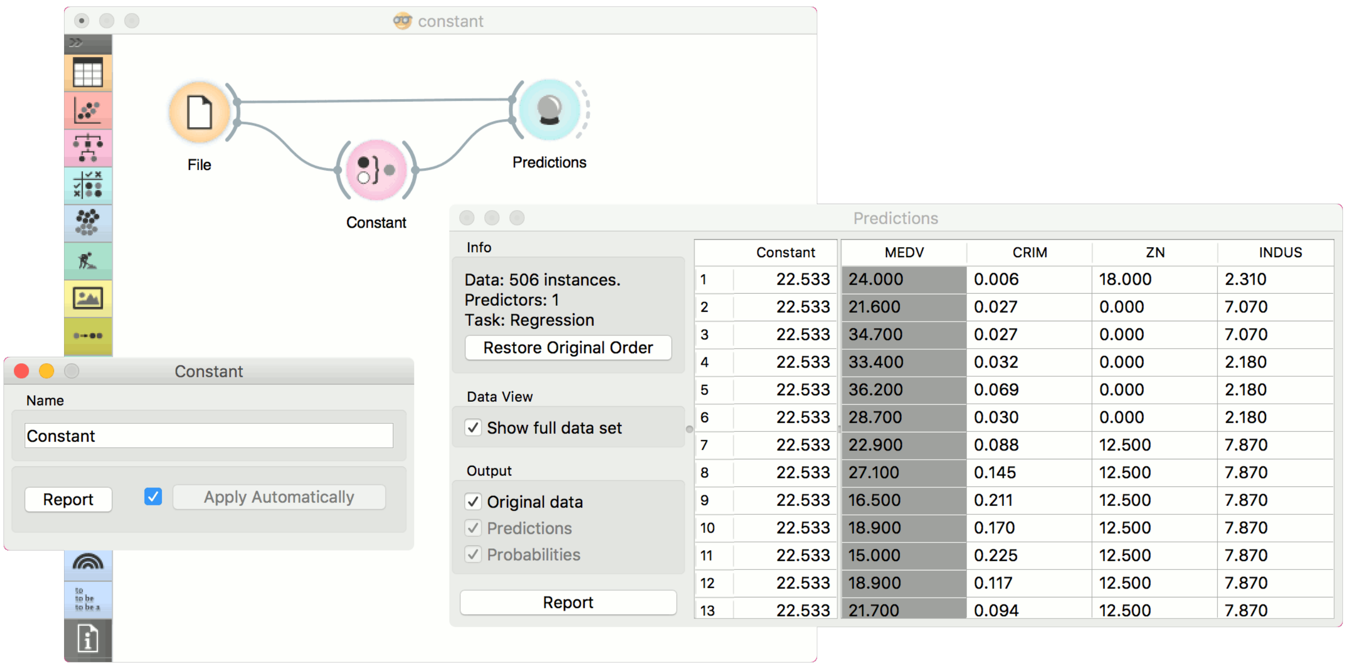
Target Class: (Average over classes)

Report

Evaluation Results

Method	AUC	CA	F1	Precision	Recall
Constant	0.500	0.333	0.167	0.111	0.333
kNN	0.989	0.973	0.973	0.974	0.973

For regression, we use **Constant** to construct a predictor in Predictions. We used the *housing* data set. In **Predictions**, you can see that *Mean Learner* returns one (mean) value for all instances.



Constant

Name: Constant

Report ☑ Apply Automatically

Predictions

Info

Data: 506 instances.
Predictors: 1
Task: Regression

Restore Original Order

Data View
- ✓ Show full data set

Output
- ✓ Original data
- ✓ Predictions
- ✓ Probabilities

Report

	Constant	MEDV	CRIM	ZN	INDUS
1	22.533	24.000	0.006	18.000	2.310
2	22.533	21.600	0.027	0.000	7.070
3	22.533	34.700	0.027	0.000	7.070
4	22.533	33.400	0.032	0.000	2.180
5	22.533	36.200	0.069	0.000	2.180
6	22.533	28.700	0.030	0.000	2.180
7	22.533	22.900	0.088	12.500	7.870
8	22.533	27.100	0.145	12.500	7.870
9	22.533	16.500	0.211	12.500	7.870
10	22.533	18.900	0.170	12.500	7.870
11	22.533	15.000	0.225	12.500	7.870
12	22.533	18.900	0.117	12.500	7.870
13	22.533	21.700	0.094	12.500	7.870

kNN

Predict according to the nearest training instances.

Signals

Inputs:

- **Data**

  A data set

- **Preprocessor**

  Preprocessing method(s)

Outputs:

- **Learner**

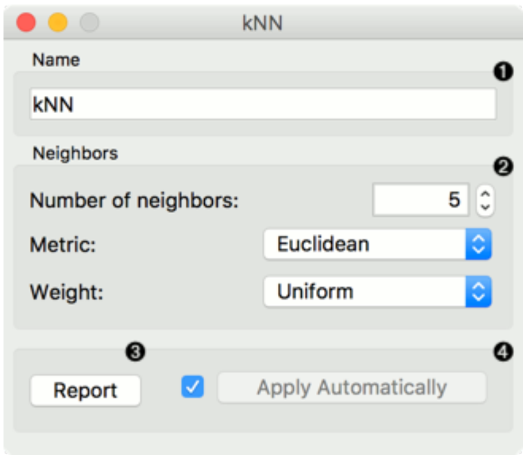
  A kNN learning algorithm with settings as specified in the dialog.

- **Model**

  A trained model. Output signal sent only if input *Data* is present.

Description

The **kNN** widget uses the kNN algorithm that searches for k closest training examples in feature space and uses their average as prediction.



kNN

Name ❶

kNN

Neighbors ❷

Number of neighbors: 5

Metric: Euclidean

Weight: Uniform

❸ ❹

1. A name under which it will appear in other widgets. The default name is “kNN”.

2. Set the number of nearest neighbors, the distance parameter (metric) and weights as model criteria. Metric can be:

   - Euclidean (“straight line”, distance between two points)
   - Manhattan (sum of absolute differences of all attributes)
   - Maximal (greatest of absolute differences between attributes)
   - Mahalanobis (distance between point and distribution).

The *Weights* you can use are:

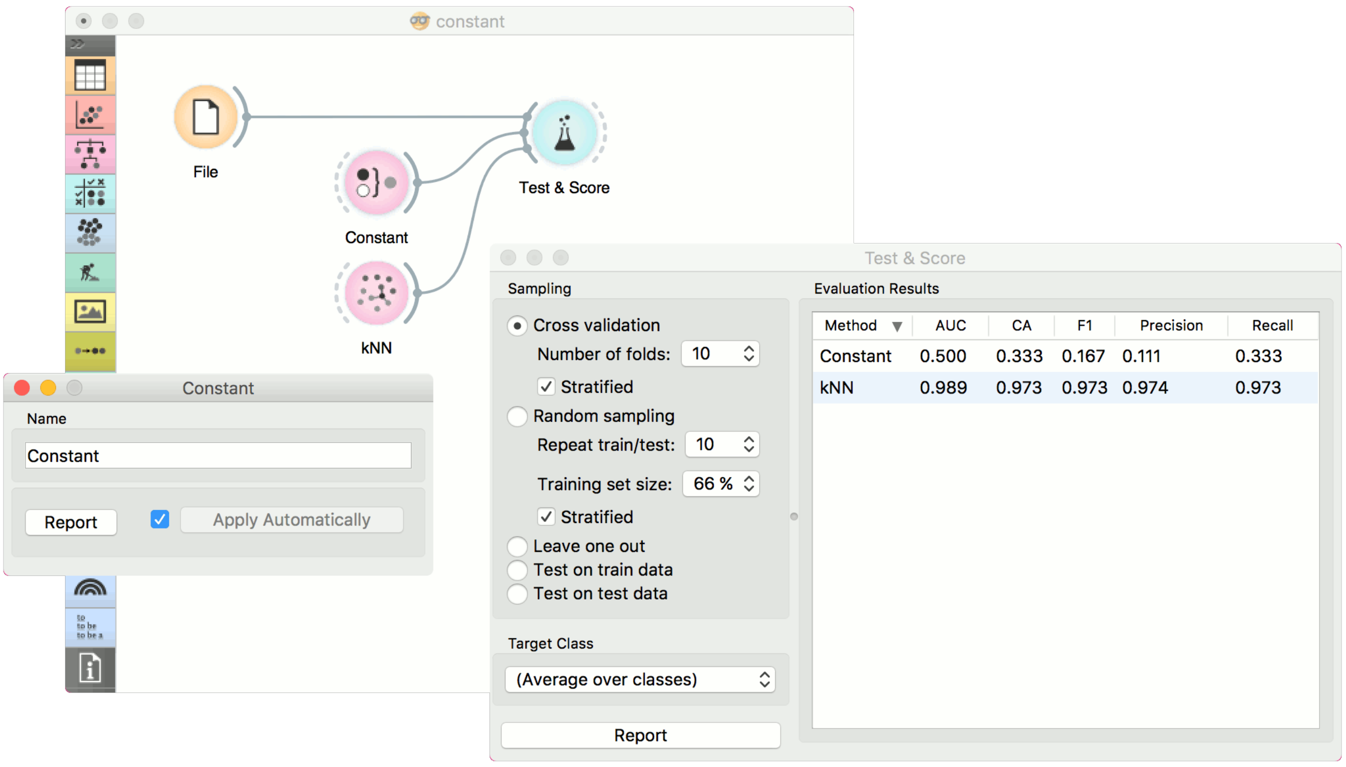
- **Uniform**: all points in each neighborhood are weighted equally.
- **Distance**: closer neighbors of a query point have a greater influence than the neighbors further away.

3. Produce a report.

4. When you change one or more settings, you need to click *Apply*, which will put a new learner on the output and, if the training examples are given, construct a new model and output it as well. Changes can also be applied automatically by clicking the box on the left side of the *Apply* button.

Examples

The first example is a classification task on *iris* data set. We compare the results of k-Nearest neighbors with the default model Constant, which always predicts the majority class.



Constant

Name: Constant

Report | ✓ Apply Automatically

Test & Score

Sampling

- (●) Cross validation
  - Number of folds: 10
  - ✓ Stratified
- ( ) Random sampling
  - Repeat train/test: 10
  - Training set size: 66 %
  - ✓ Stratified
- ( ) Leave one out
- ( ) Test on train data
- ( ) Test on test data

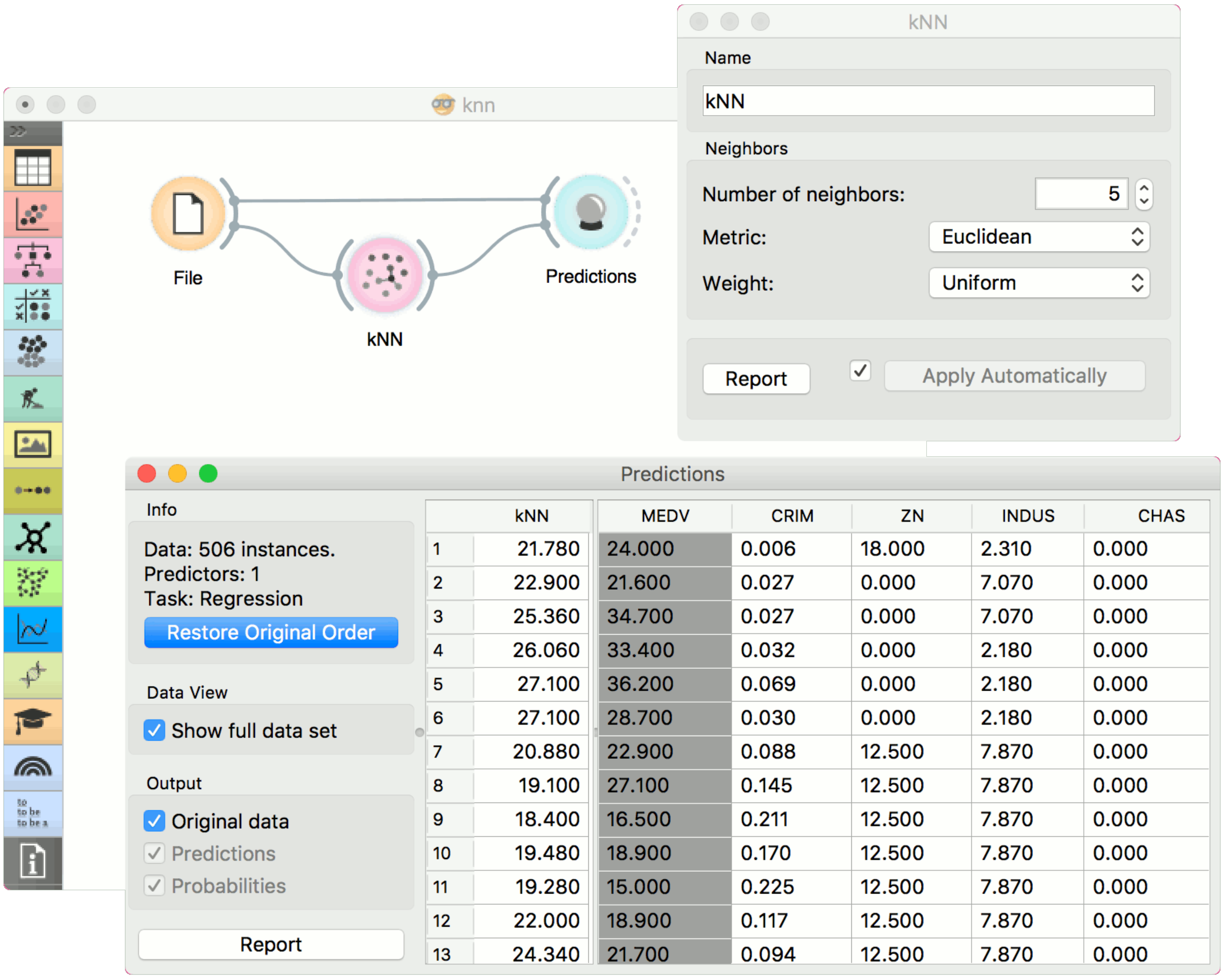
Target Class: (Average over classes)

Report

Evaluation Results

Method	AUC	CA	F1	Precision	Recall
Constant	0.500	0.333	0.167	0.111	0.333
kNN	0.989	0.973	0.973	0.974	0.973

The second example is a regression task. This workflow shows how to use the *Learner* output. For the purpose of this example, we used the *housing* data set. We input the **kNN** prediction model into Predictions and observe the predicted values.



kNN

Name

kNN

Neighbors

Number of neighbors: 5

Metric: Euclidean

Weight: Uniform

Report ✓ Apply Automatically

Predictions

Info

Data: 506 instances.
Predictors: 1
Task: Regression

Restore Original Order

Data View

✓ Show full data set

Output

✓ Original data
✓ Predictions
✓ Probabilities

Report

	kNN	MEDV	CRIM	ZN	INDUS	CHAS
1	21.780	24.000	0.006	18.000	2.310	0.000
2	22.900	21.600	0.027	0.000	7.070	0.000
3	25.360	34.700	0.027	0.000	7.070	0.000
4	26.060	33.400	0.032	0.000	2.180	0.000
5	27.100	36.200	0.069	0.000	2.180	0.000
6	27.100	28.700	0.030	0.000	2.180	0.000
7	20.880	22.900	0.088	12.500	7.870	0.000
8	19.100	27.100	0.145	12.500	7.870	0.000
9	18.400	16.500	0.211	12.500	7.870	0.000
10	19.480	18.900	0.170	12.500	7.870	0.000
11	19.280	15.000	0.225	12.500	7.870	0.000
12	22.000	18.900	0.117	12.500	7.870	0.000
13	24.340	21.700	0.094	12.500	7.870	0.000

Linear Regression

A linear regression algorithm with optional L1 (LASSO), L2 (ridge) or L1L2 (elastic net) regularization.

Signals

Inputs:

- **Data**

  A data set

- **Preprocessor**

  A preprocessed data set.

Outputs:

- **Learner**

  A linear regression learning algorithm with settings as specified in the dialog.

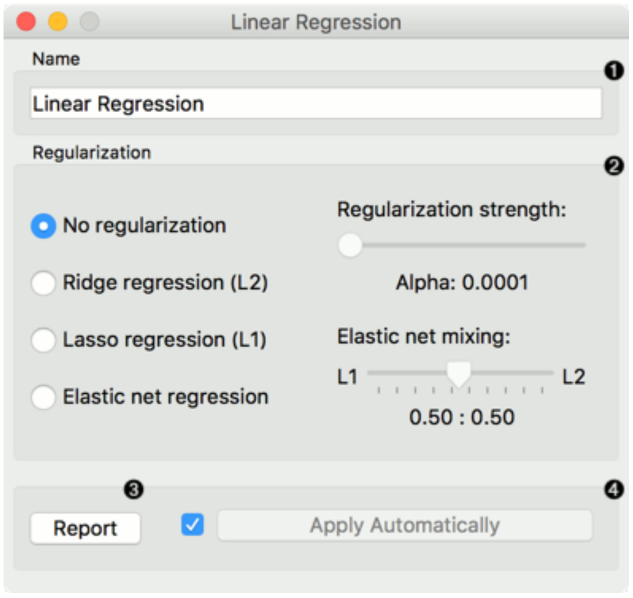
- **Predictor**

  A trained regressor. Output signal sent only if input *Data* is present.

Description

The **Linear Regression** widget constructs a learner/predictor that learns a linear function from its input data. The model can identify the relationship between a predictor xi and the response variable y. Additionally, Lasso and Ridge regularization parameters can be specified. Lasso regression minimizes a penalized version of the least squares loss function with L1-norm penalty and Ridge regularization with L2-norm penalty.

Linear regreesion works only on regression tasks.


Linear Regression

Name: Linear Regression

Regularization

[✓ No regularization]
Ridge regression (L2)
Lasso regression (L1)
Elastic net regression

Regularization strength:
Alpha: 0.0001

Elastic net mixing:
L1 — L2
0.50 : 0.50

Report
[✓] Apply Automatically

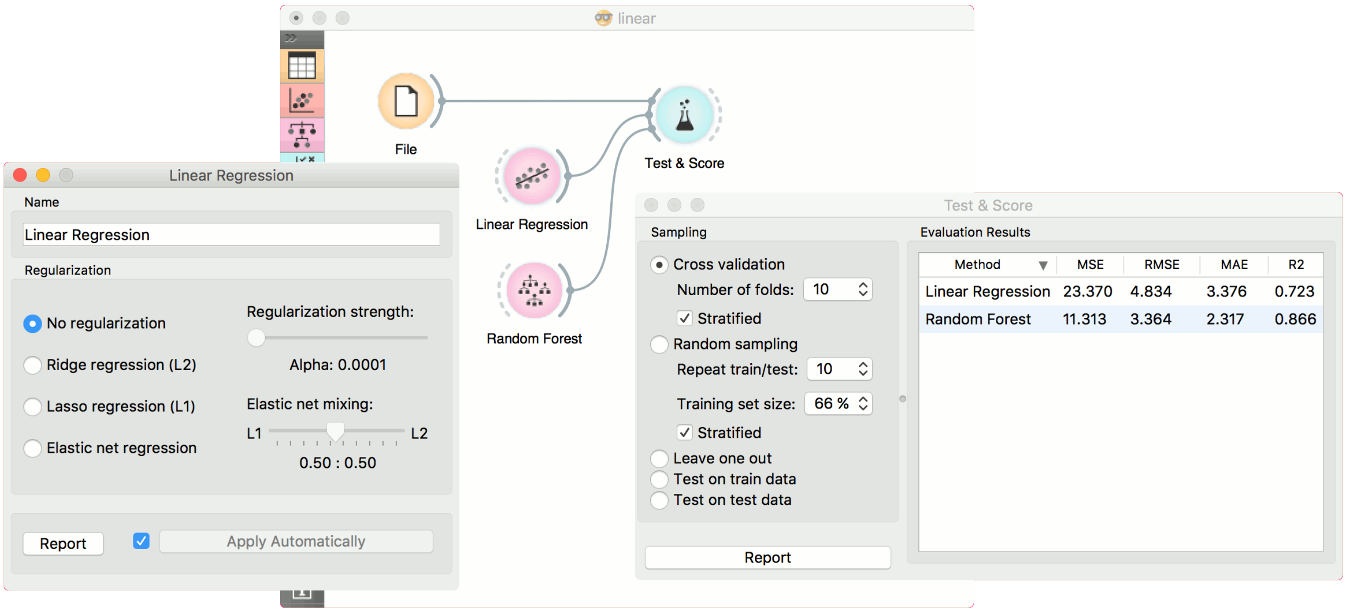
1. The learner/predictor name
2. Choose a model to train:

- no regularization
- a Ridge regularization (L2-norm penalty)
- a Lasso bound (L1-norm penalty)
- an Elastic net regularization

3. Produce a report.
4. Press *Apply* to commit changes. If *Apply Automatically* is ticked, changes are committed automatically.

Example

Below, is a simple workflow with *housing* data set. We trained **Linear Regression** and Random Forest and evaluated their performance in Test&Score.



Linear Regression

Name: Linear Regression

Regularization

[✓ No regularization] Ridge regression (L2) Lasso regression (L1) Elastic net regression

Regularization strength: Alpha: 0.0001

Elastic net mixing: L1 0.50 : 0.50 L2

Report ✓ Apply Automatically

Test & Score

Sampling

[✓ Cross validation]
Number of folds: 10
✓ Stratified
Random sampling
Repeat train/test: 10
Training set size: 66 %
✓ Stratified
Leave one out
Test on train data
Test on test data

Report

Evaluation Results

Method	MSE	RMSE	MAE	R2
Linear Regression	23.370	4.834	3.376	0.723
Random Forest	11.313	3.364	2.317	0.866

Load Model

Load a model from an input file.

Signals

Inputs:

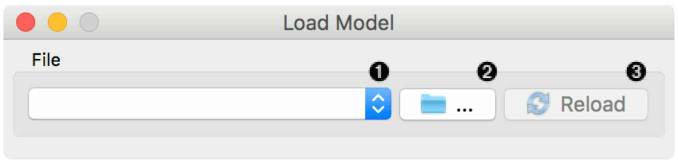
- None

Outputs:

- **Model**

  A model with selected parameters.

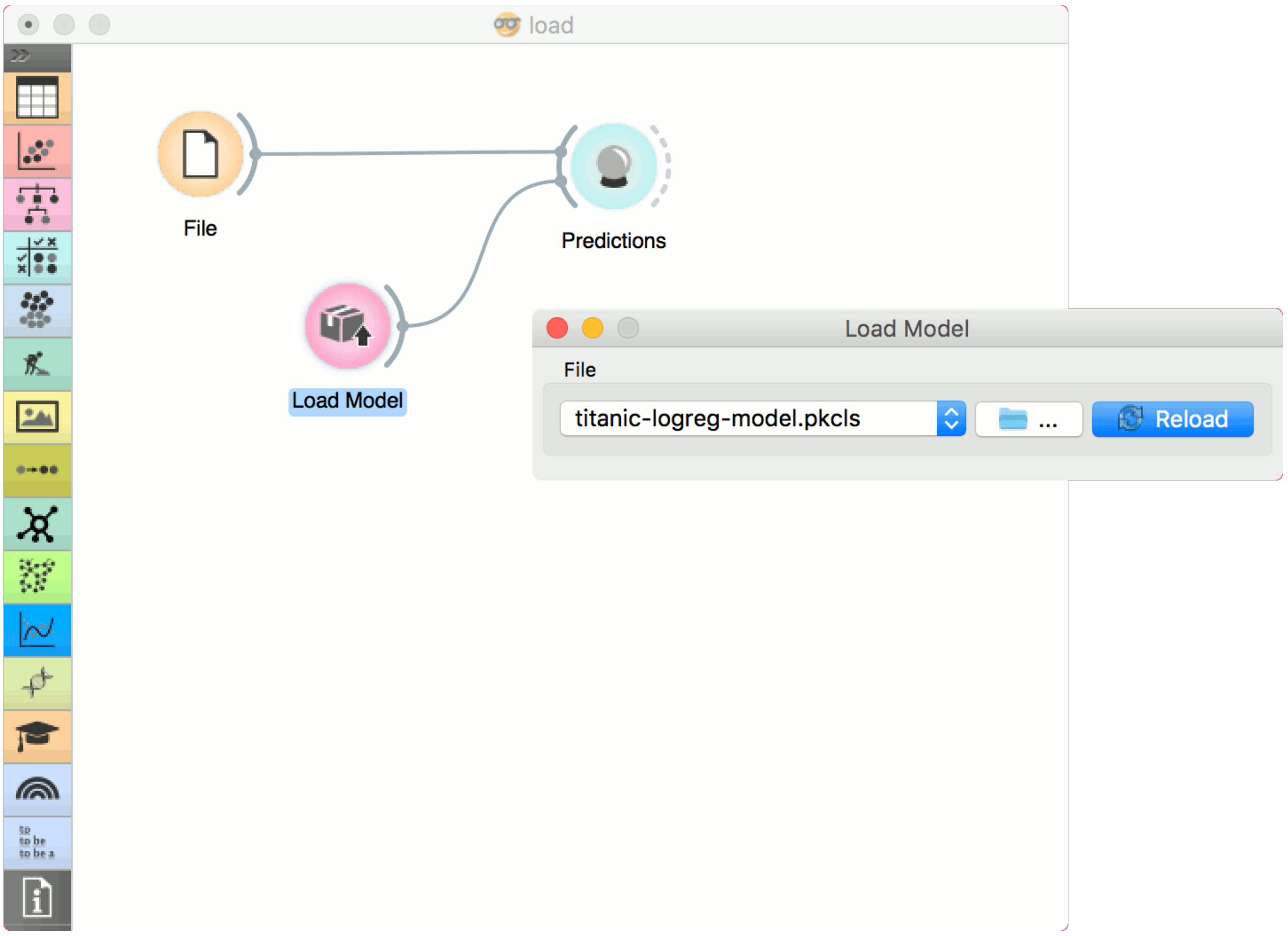
Description



1. Choose from a list of previously used models.
2. Browse for saved models.
3. Reload the selected model.

Example

When you want to use a custom-set model that you've saved before, open the **Load Model** widget and select the de-sired file with the *Browse* icon. This widget loads the exisiting model into Predictions widget. Data sets used with **Load Model** have to contain compatible attributes!


load
File
Predictions
Load Model
Load Model
File
titanic-logreg-model.pkcls
...
Reload

Logistic Regression

The logistic regression classification algorithm with LASSO (L1) or ridge (L2) regularization.

Signals

Inputs:

- **Data**

  A data set

- **Preprocessor**

  Preprocessing method(s)

Outputs:

- **Learner**

  A logistic regression learning algorithm with settings as specified in the dialog.

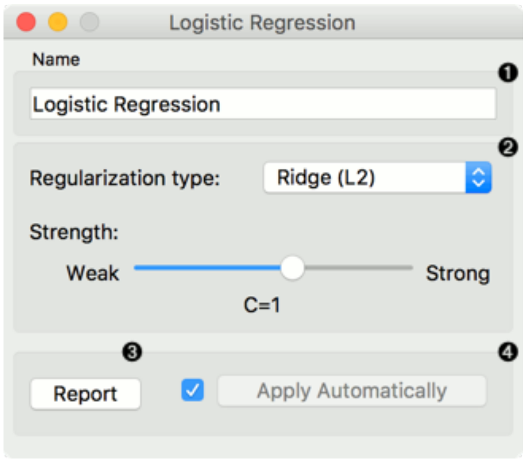
- **Logistic Regression Classifier**

  A trained classifier. Output signal sent only if input *Data* is present.

Description

Logistic Regression learns a Logistic Regression model from the data.

It only works for classification tasks.

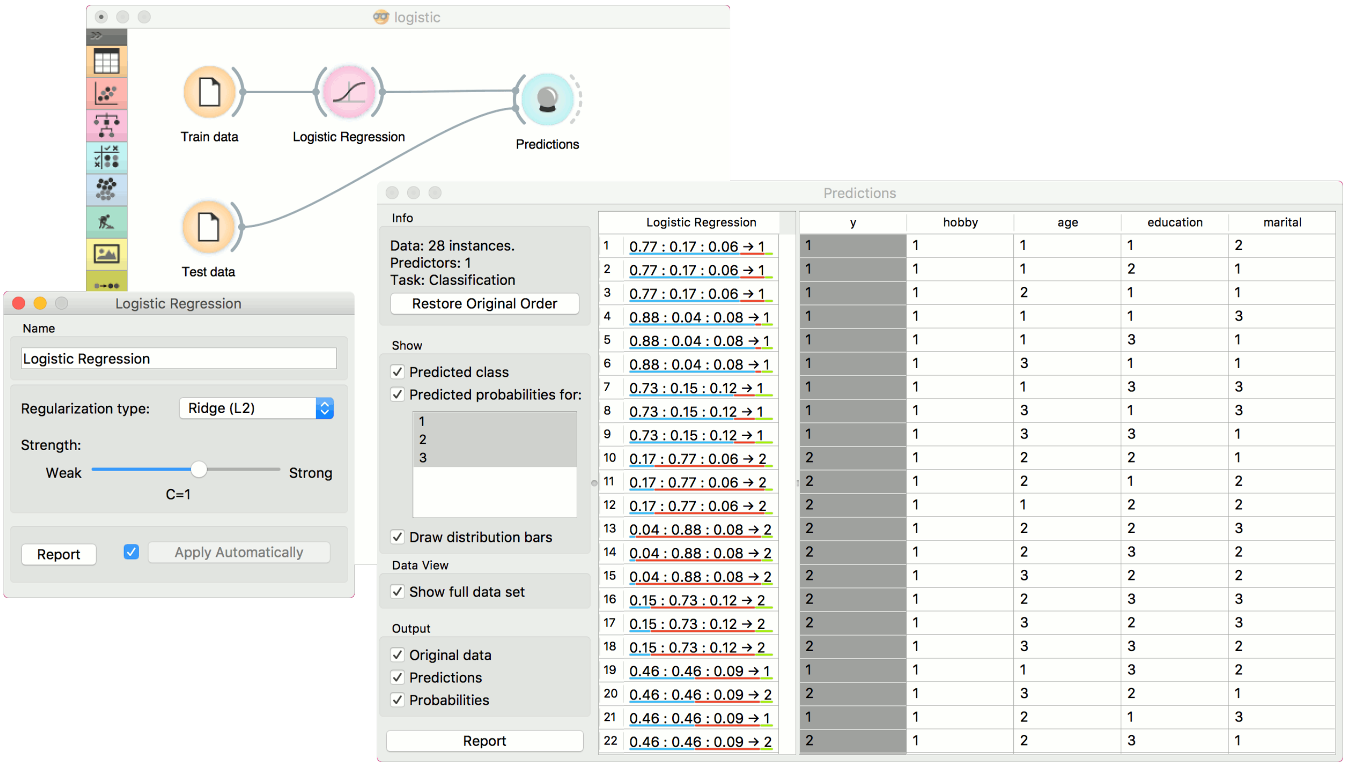


1. A name under which the learner appears in other widgets. The default name is “Logistic Regression”.
2. Regularization type (either L1 or L2). Set the cost strength (default is C=1).
3. Press *Apply* to commit changes. If *Apply Automatically* is ticked, changes will be communicated automatically.

Example

The widget is used just as any other widget for inducing a classifier. This is an example demonstrating prediction results with logistic regression on the *hayes-roth* data set. We first load *hayes-roth_learn* in the File widget and pass the data to **Logistic Regression**. Then we pass the trained model to Predictions.

Now we want to predict class value on a new data set. We load *hayes-roth_test* in the second **File** widget and connect it to **Predictions**. We can now observe class values predicted with **Logistic Regression** directly in **Predictions**.



Logistic Regression

Name: Logistic Regression

Regularization type: Ridge (L2)

Strength: Weak — Strong, C=1

Report ✓ Apply Automatically

Predictions

Info

Data: 28 instances.
Predictors: 1
Task: Classification

Restore Original Order

Show

✓ Predicted class
✓ Predicted probabilities for:
1
2
3
✓ Draw distribution bars

Data View

✓ Show full data set

Output

✓ Original data
✓ Predictions
✓ Probabilities

Report

	Logistic Regression	y	hobby	age	education	marital
1	0.77 : 0.17 : 0.06 → 1	1	1	1	1	2
2	0.77 : 0.17 : 0.06 → 1	1	1	1	2	1
3	0.77 : 0.17 : 0.06 → 1	1	1	2	1	1
4	0.88 : 0.04 : 0.08 → 1	1	1	1	1	3
5	0.88 : 0.04 : 0.08 → 1	1	1	1	3	1
6	0.88 : 0.04 : 0.08 → 1	1	1	3	1	1
7	0.73 : 0.15 : 0.12 → 1	1	1	1	3	3
8	0.73 : 0.15 : 0.12 → 1	1	1	3	1	3
9	0.73 : 0.15 : 0.12 → 1	1	1	3	3	1
10	0.17 : 0.77 : 0.06 → 2	2	1	2	2	1
11	0.17 : 0.77 : 0.06 → 2	2	1	2	1	2
12	0.17 : 0.77 : 0.06 → 2	2	1	1	2	2
13	0.04 : 0.88 : 0.08 → 2	2	1	2	2	3
14	0.04 : 0.88 : 0.08 → 2	2	1	2	3	2
15	0.04 : 0.88 : 0.08 → 2	2	1	3	2	2
16	0.15 : 0.73 : 0.12 → 2	2	1	2	3	3
17	0.15 : 0.73 : 0.12 → 2	2	1	3	2	3
18	0.15 : 0.73 : 0.12 → 2	2	1	3	3	2
19	0.46 : 0.46 : 0.09 → 1	1	1	1	3	2
20	0.46 : 0.46 : 0.09 → 2	2	1	3	2	1
21	0.46 : 0.46 : 0.09 → 1	1	1	2	1	3
22	0.46 : 0.46 : 0.09 → 2	2	1	2	3	1

Naive Bayes

A fast and simple probabilistic classifier based on Bayes' theorem with the assumption of feature independence.

Signals

Inputs:

- **Data**

  A data set

- **Preprocessor**

  Preprocessing method(s)

Outputs:

- **Learner**

  A naive bayes learning algorithm with settings as specified in the dialog.

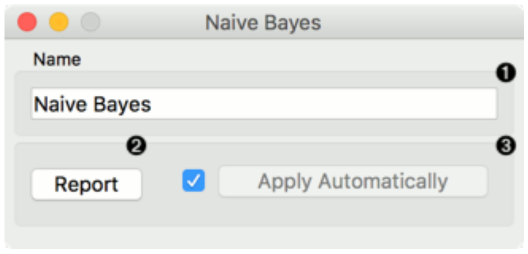
- **Model**

  A trained classifier. Output signal sent only if input *Data* is present.

Description

Naive Bayes learns a Naive Bayesian model from the data.

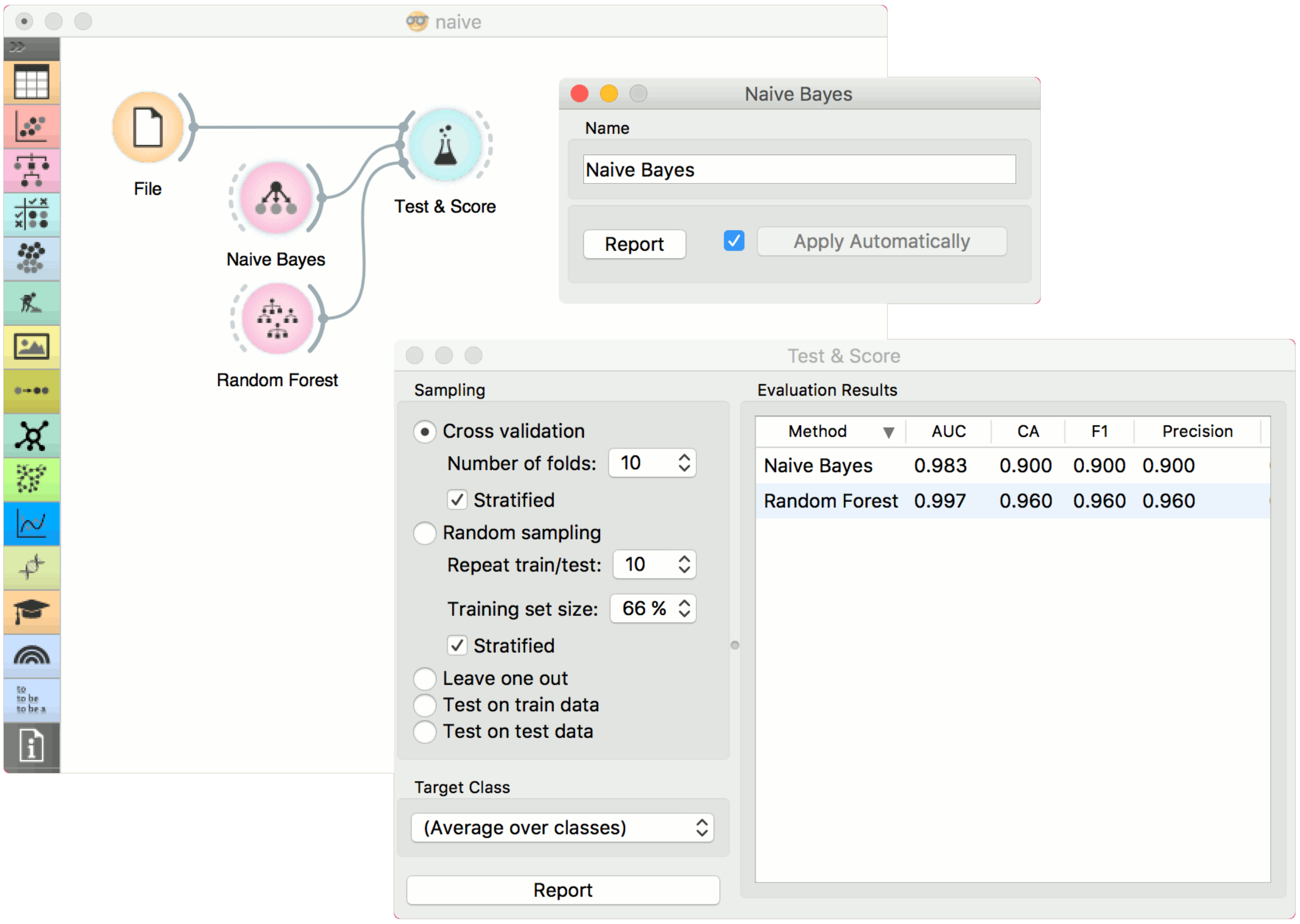
It only works for classification tasks.



This widget has two options: the name under which it will appear in other widgets and producing a report. The default name is *Naive Bayes*. When you change it, you need to press *Apply*.

Examples

Here, we present two uses of this widget. First, we compare the results of the **Naive Bayes** with another model, the Random Forest. We connect *iris* data from File to Test&Score. We also connect **Naive Bayes** and Random Forest to **Test & Score** and observe their prediction scores.



Naive Bayes

Name

Naive Bayes

Report ✓ Apply Automatically

Test & Score

Sampling

- ● Cross validation
  - Number of folds: 10
  - ✓ Stratified
- ○ Random sampling
  - Repeat train/test: 10
  - Training set size: 66 %
  - ✓ Stratified
- ○ Leave one out
- ○ Test on train data
- ○ Test on test data

Target Class

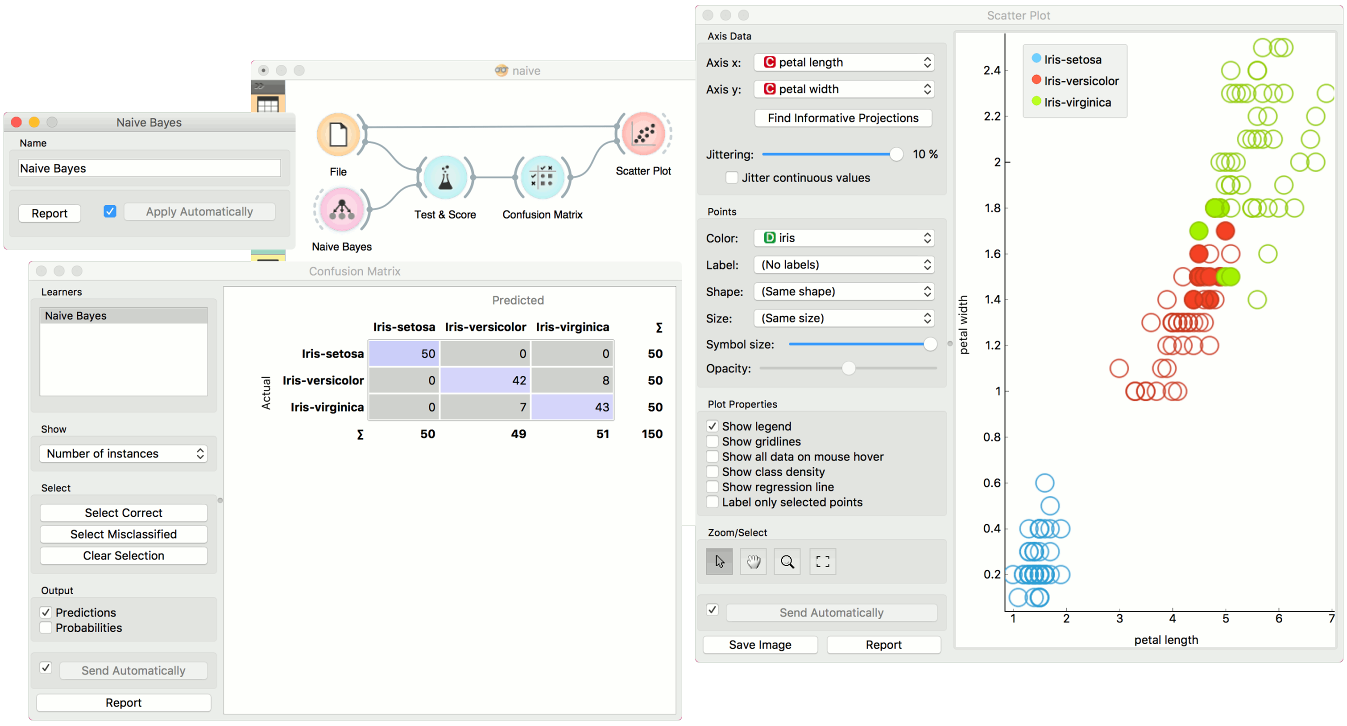
(Average over classes)

Report

Evaluation Results

Method	AUC	CA	F1	Precision
Naive Bayes	0.983	0.900	0.900	0.900
Random Forest	0.997	0.960	0.960	0.960

The second schema shows the quality of predictions made with **Naive Bayes**. We feed the Test&Score widget a Naive Bayes learner and then send the data to the Confusion Matrix. We also connect Scatterplot with **File**. Then we select the misclassified instances in the **Confusion Matrix** and show feed them to Scatterplot. The bold dots in the scatterplot are the misclassified instances from **Naive Bayes**.



Naive Bayes

Name

Naive Bayes

Report ✓ Apply Automatically

Confusion Matrix

Learners: Naive Bayes

Show: Number of instances

Select: Select Correct / Select Misclassified / Clear Selection

Output: ✓ Predictions, Probabilities

✓ Send Automatically

Report

Actual \ Predicted	Iris-setosa	Iris-versicolor	Iris-virginica	Σ
Iris-setosa	50	0	0	50
Iris-versicolor	0	42	8	50
Iris-virginica	0	7	43	50
Σ	50	49	51	150

Scatter Plot

Axis Data
- Axis x: petal length
- Axis y: petal width
- Find Informative Projections
- Jittering: 10 %
- Jitter continuous values

Points
- Color: iris
- Label: (No labels)
- Shape: (Same shape)
- Size: (Same size)
- Symbol size:
- Opacity:

Plot Properties
- ✓ Show legend
- Show gridlines
- Show all data on mouse hover
- Show class density
- Show regression line
- Label only selected points

Zoom/Select

✓ Send Automatically

Save Image Report

Neural Network

A multi-layer perceptron (MLP) algorithm with backpropagation.

Signals

Inputs:

- **Data**

  A data set

- **Preprocessor**

  Preprocessing method(s)

Outputs:

- **Learner**

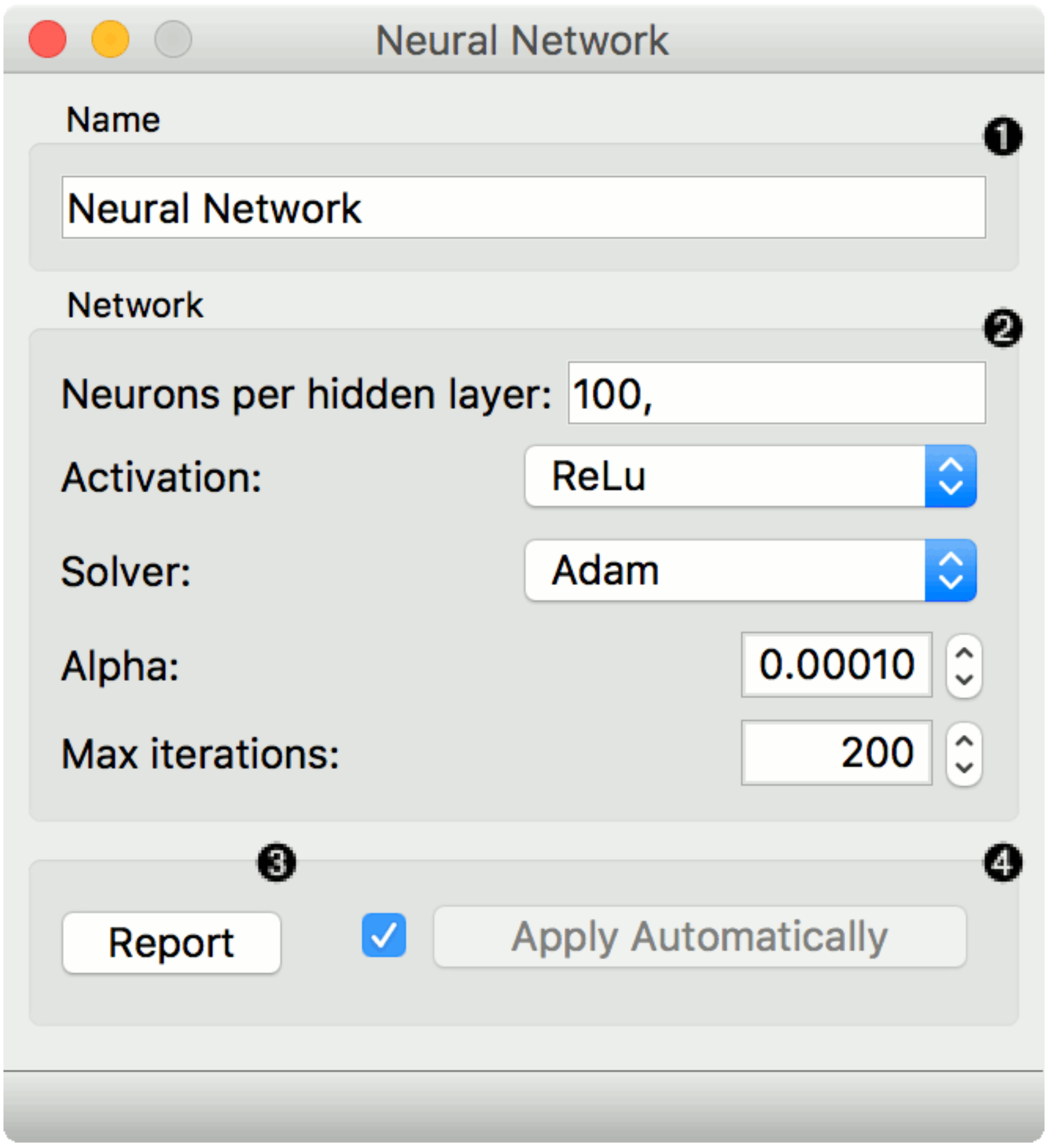
  A MLP learning algorithm with settings as specified in the dialog.

- **Model**

  A trained model. Output signal sent only if input *Data* is present.

Description

The **Neural Network** widget uses sklearn's Multi-layer Perceptron algorithm that can learn non-linear models as well as linear.



Name: Neural Network

Network

Neurons per hidden layer: 100,

Activation: ReLu

Solver: Adam

Alpha: 0.00010

Max iterations: 200

✓ Apply Automatically

1. A name under which it will appear in other widgets. The default name is "Neural Network".

2. Set model parameters: - Neurons per hidden layer: defined as the ith element represents the number of neurons in the ith hidden layer. E.g. a neural network with 3 layers can be defined as 2, 3, 2. - Activation function for the hidden layer:

    - Identity: no-op activation, useful to implement linear bottleneck
    - Logistic: the logistic sigmoid function
    - tanh: the hyperbolic tan function
    - ReLu: the rectified linear unit function

   - Solver for weight optimization: - L-BFGS-B: an optimizer in the family of quasi-Newton methods - SGD: stochastic gradient descent - Adam: stochastic gradient-based optimizer
   - Alpha: L2 penalty (regularization term) parameter
   - Max iterations: maximum number of iterations

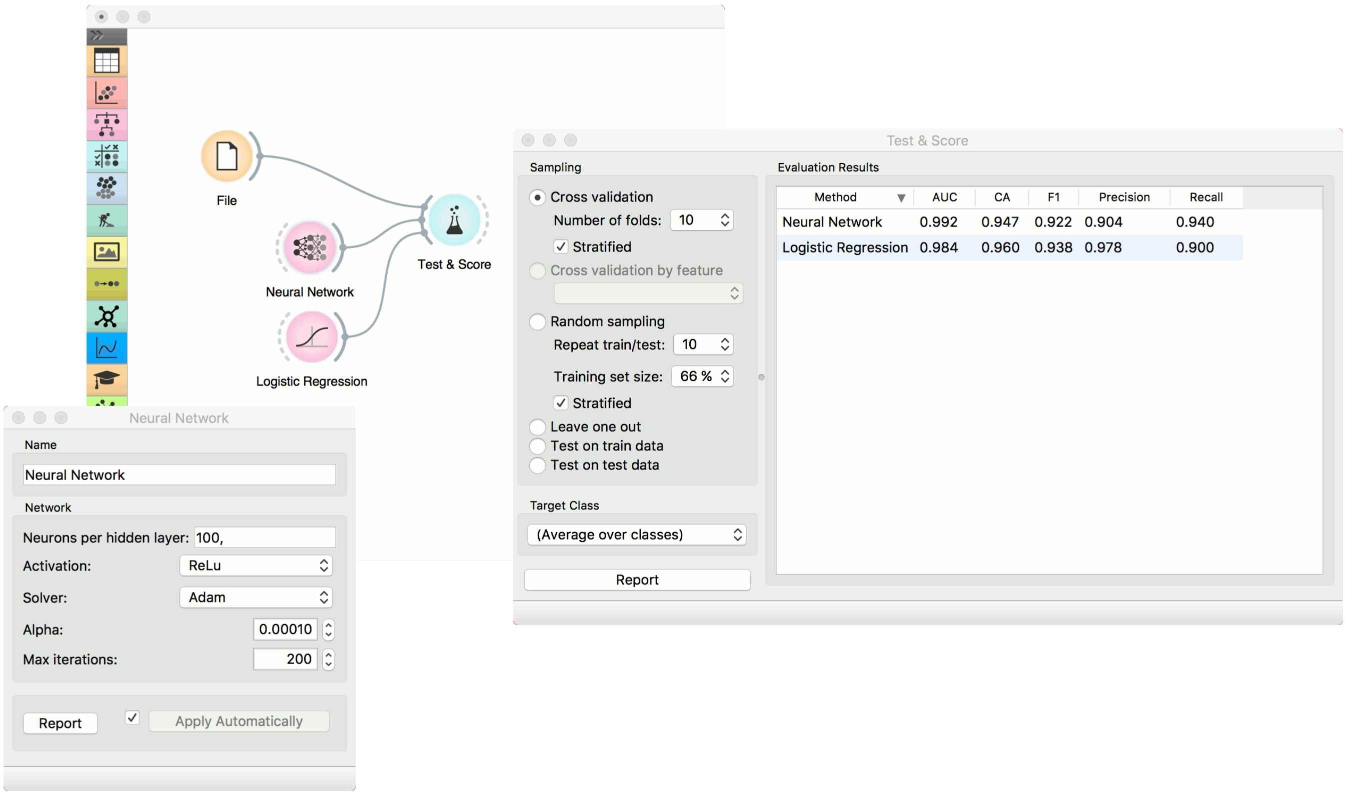
   Other parameters are set to sklearn's defaults.

3. Produce a report.

4. When the box is ticked (*Apply Automatically*), the widget will communicate changes automatically. Alternatively, click *Apply*.

Examples

The first example is a classification task on *iris* data set. We compare the results of **Neural Network** with the Logistic Regression.



Test & Score

Sampling

- (●) Cross validation
  - Number of folds: 10
  - ✓ Stratified
- ( ) Cross validation by feature
- ( ) Random sampling
  - Repeat train/test: 10
  - Training set size: 66 %
  - ✓ Stratified
- ( ) Leave one out
- ( ) Test on train data
- ( ) Test on test data

Target Class: (Average over classes)

Report

Evaluation Results

Method	AUC	CA	F1	Precision	Recall
Neural Network	0.992	0.947	0.922	0.904	0.940
Logistic Regression	0.984	0.960	0.938	0.978	0.900

Neural Network

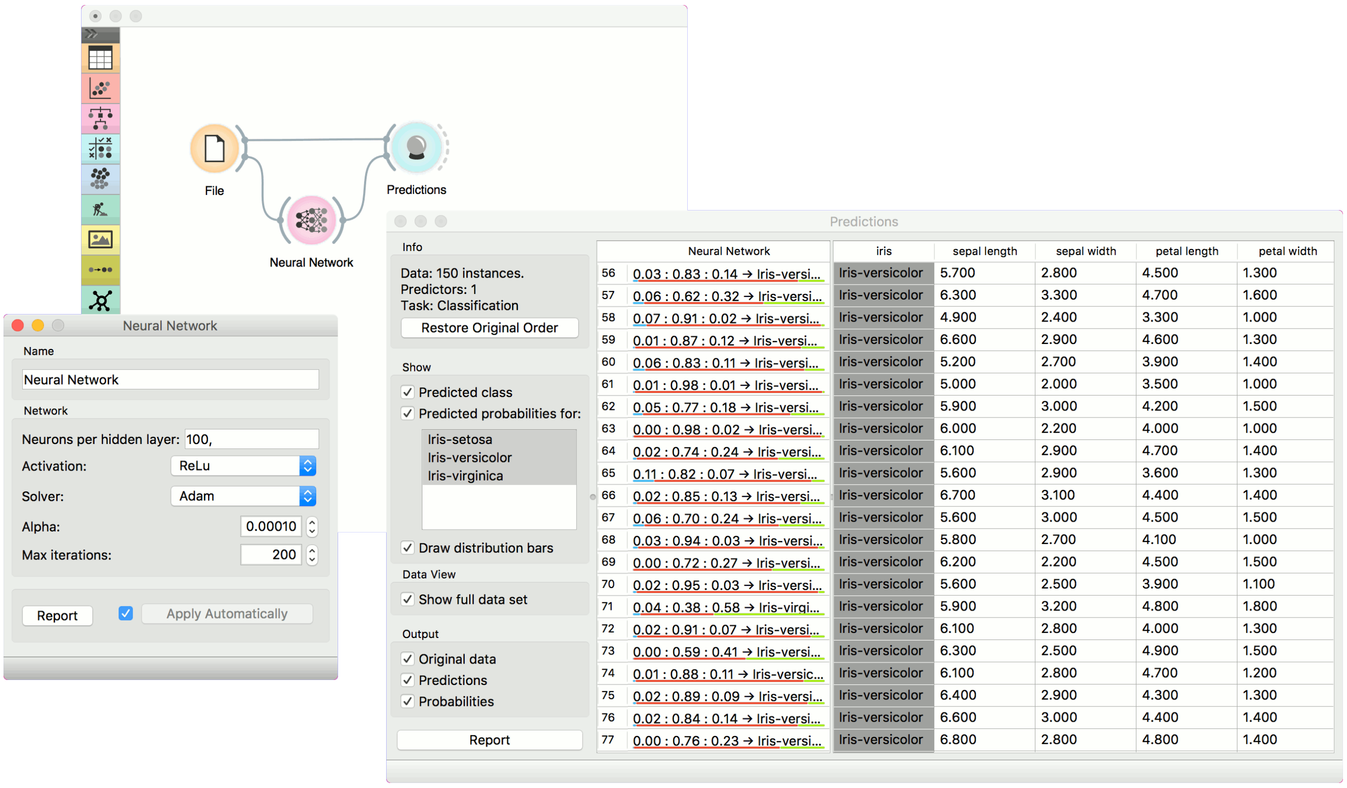
Name: Neural Network

Network

- Neurons per hidden layer: 100,
- Activation: ReLu
- Solver: Adam
- Alpha: 0.00010
- Max iterations: 200

Report ✓ Apply Automatically

The second example is a prediction task, still using the *iris* data. This workflow shows how to use the *Learner* output. We input the **Neural Network** prediction model into Predictions and observe the predicted values.



Neural Network

Name: Neural Network

Network

- Neurons per hidden layer: 100,
- Activation: ReLu
- Solver: Adam
- Alpha: 0.00010
- Max iterations: 200

Report ✓ Apply Automatically

Predictions

Info

Data: 150 instances.
Predictors: 1
Task: Classification

Restore Original Order

Show

- ✓ Predicted class
- ✓ Predicted probabilities for:
  - Iris-setosa
  - Iris-versicolor
  - Iris-virginica
- ✓ Draw distribution bars

Data View

- ✓ Show full data set

Output

- ✓ Original data
- ✓ Predictions
- ✓ Probabilities

Report

	Neural Network	iris	sepal length	sepal width	petal length	petal width
56	0.03 : 0.83 : 0.14 → Iris-versi...	Iris-versicolor	5.700	2.800	4.500	1.300
57	0.06 : 0.62 : 0.32 → Iris-versi...	Iris-versicolor	6.300	3.300	4.700	1.600
58	0.07 : 0.91 : 0.02 → Iris-versi...	Iris-versicolor	4.900	2.400	3.300	1.000
59	0.01 : 0.87 : 0.12 → Iris-versi...	Iris-versicolor	6.600	2.900	4.600	1.300
60	0.06 : 0.83 : 0.11 → Iris-versi...	Iris-versicolor	5.200	2.700	3.900	1.400
61	0.01 : 0.98 : 0.01 → Iris-versi...	Iris-versicolor	5.000	2.000	3.500	1.000
62	0.05 : 0.77 : 0.18 → Iris-versi...	Iris-versicolor	5.900	3.000	4.200	1.500
63	0.00 : 0.98 : 0.02 → Iris-versi...	Iris-versicolor	6.000	2.200	4.000	1.000
64	0.02 : 0.74 : 0.24 → Iris-versi...	Iris-versicolor	6.100	2.900	4.700	1.400
65	0.11 : 0.82 : 0.07 → Iris-versi...	Iris-versicolor	5.600	2.900	3.600	1.300
66	0.02 : 0.85 : 0.13 → Iris-versi...	Iris-versicolor	6.700	3.100	4.400	1.400
67	0.06 : 0.70 : 0.24 → Iris-versi...	Iris-versicolor	5.600	3.000	4.500	1.500
68	0.03 : 0.94 : 0.03 → Iris-versi...	Iris-versicolor	5.800	2.700	4.100	1.000
69	0.00 : 0.72 : 0.27 → Iris-versi...	Iris-versicolor	6.200	2.200	4.500	1.500
70	0.02 : 0.95 : 0.03 → Iris-versi...	Iris-versicolor	5.600	2.500	3.900	1.100
71	0.04 : 0.38 : 0.58 → Iris-virgi...	Iris-versicolor	5.900	3.200	4.800	1.800
72	0.02 : 0.91 : 0.07 → Iris-versi...	Iris-versicolor	6.100	2.800	4.000	1.300
73	0.00 : 0.59 : 0.41 → Iris-versi...	Iris-versicolor	6.300	2.500	4.900	1.500
74	0.01 : 0.88 : 0.11 → Iris-versic...	Iris-versicolor	6.100	2.800	4.700	1.200
75	0.02 : 0.89 : 0.09 → Iris-versi...	Iris-versicolor	6.400	2.900	4.300	1.300
76	0.02 : 0.84 : 0.14 → Iris-versi...	Iris-versicolor	6.600	3.000	4.400	1.400
77	0.00 : 0.76 : 0.23 → Iris-versi...	Iris-versicolor	6.800	2.800	4.800	1.400

Random Forest

Predict using an ensemble of decision trees.

Signals

Inputs:

- **Data**

  A data set

- **Preprocessor**

  Preprocessing method(s)

Outputs:

- **Learner**

  A random forest learning algorithm with settings as specified in the dialog.

- **Model**

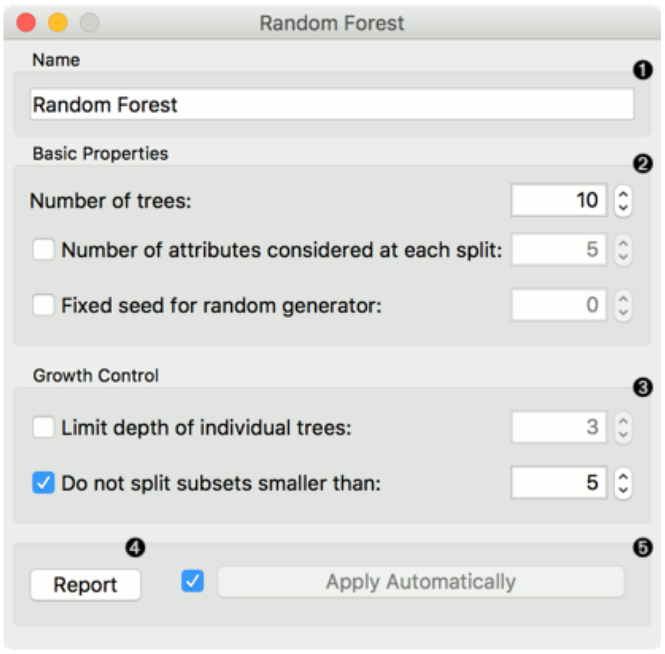
  A trained model. Output signal sent only if input *Data* is present.

Description

Random forest is an ensemble learning method used for classification, regression and other tasks. It was first proposed by Tin Kam Ho and further developed by Leo Breiman (Breiman, 2001) and Adele Cutler.

Random Forest builds a set of decision trees. Each tree is developed from a bootstrap sample from the training data. When developing individual trees, an arbitrary subset of attributes is drawn (hence the term “Random”), from which the best attribute for the split is selected. The final model is based on the majority vote from individually developed trees in the forest.

Random Forest works for both classification and regression tasks.


Random Forest

Name

Random Forest

Basic Properties

Number of trees: 10

Number of attributes considered at each split: 5

Fixed seed for random generator: 0

Growth Control

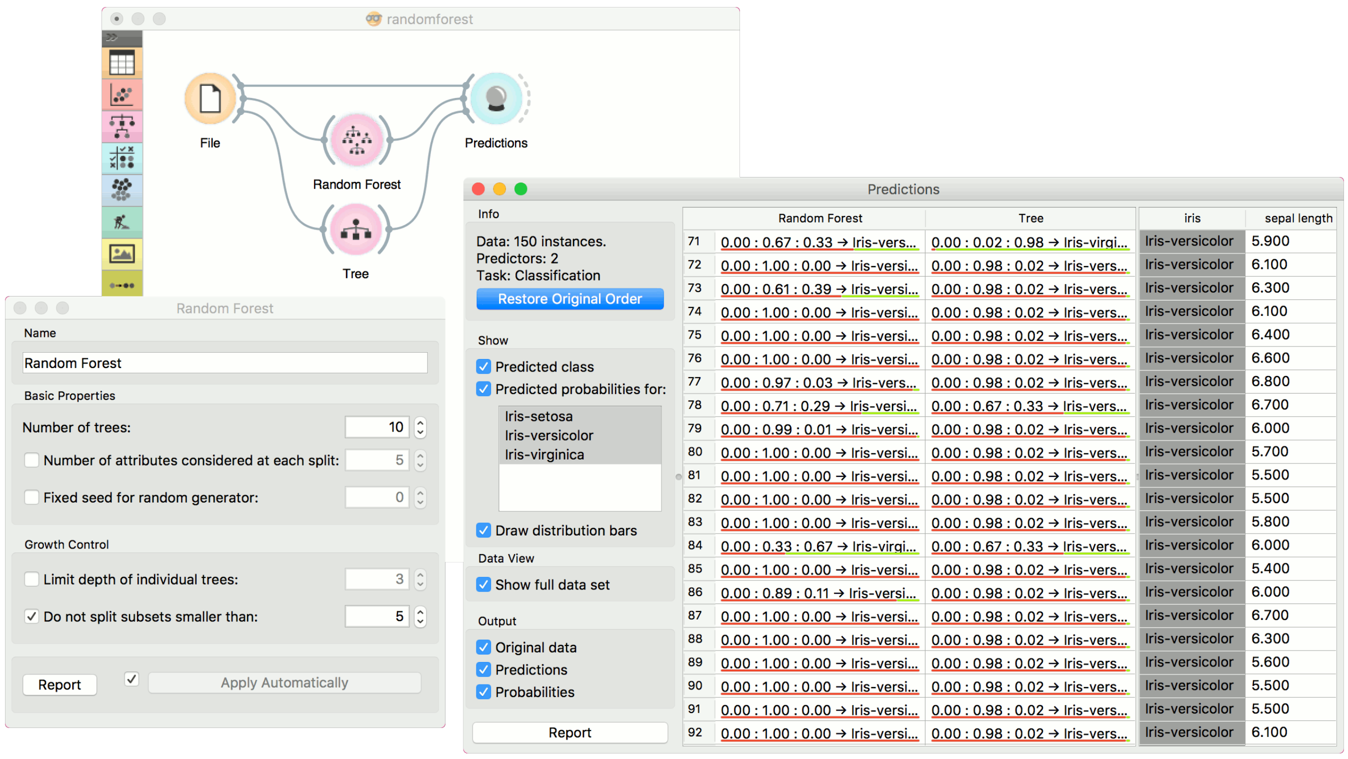
Limit depth of individual trees: 3

Do not split subsets smaller than: 5

1. Specify the name of the model. The default name is “Random Forest”.
2. Specify how many decision trees will be included in the forest (*Number of trees in the forest*), and how many attributes will be arbitrarily drawn for consideration at each node. If the latter is not specified (option *Number of attributes*... left unchecked), this number is equal to the square root of the number of attributes in the data. You can also choose to fix the seed for tree generation (*Fixed seed for random generator*), which enables replicability of the results.
3. Original Brieman’s proposal is to grow the trees without any pre-prunning, but since pre-pruning often works quite well and is faster, the user can set the depth to which the trees will be grown (*Limit depth of individual trees*). Another pre-pruning option is to select the smallest subset that can be split (*Do not split subsets smaller than*).
4. Produce a report.
5. Click *Apply* to communicate the changes to other widgets. Alternatively, tick the box on the left side of the *Apply* button and changes will be communicated automatically.

Examples

For classification tasks, we use *iris* data set. Connect it to Predictions. Then, connect File to **Random Forest** and Tree and connect them further to Predictions. Finally, observe the predictions for the two models.



Random Forest

Name: Random Forest

Basic Properties
- Number of trees: 10
- [ ] Number of attributes considered at each split: 5
- [ ] Fixed seed for random generator: 0

Growth Control
- [ ] Limit depth of individual trees: 3
- [✓] Do not split subsets smaller than: 5

[✓] Apply Automatically

Predictions

Info
Data: 150 instances.
Predictors: 2
Task: Classification

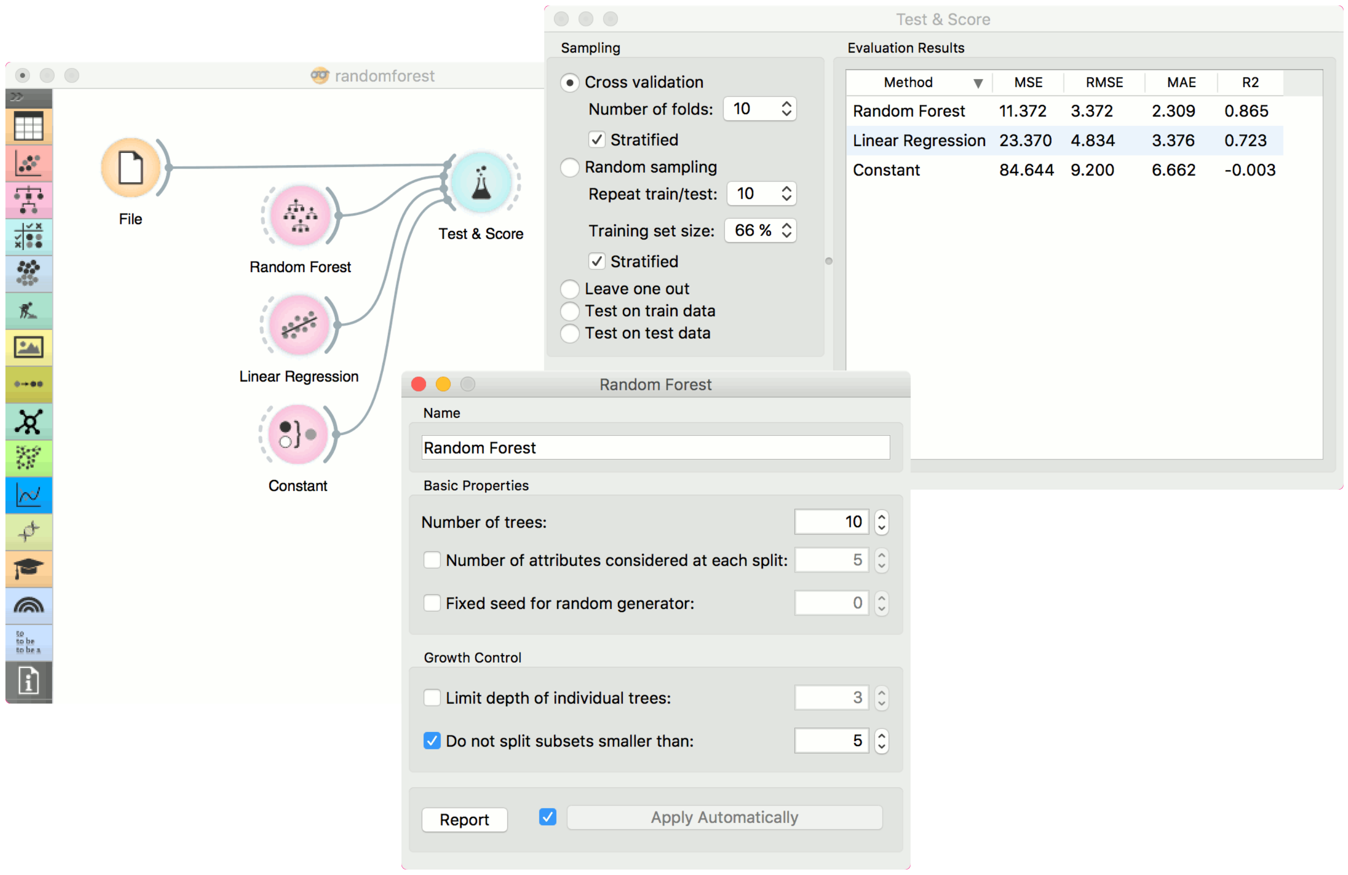
Show
- [✓] Predicted class
- [✓] Predicted probabilities for:
  - Iris-setosa
  - Iris-versicolor
  - Iris-virginica
- [✓] Draw distribution bars

Data View
- [✓] Show full data set

Output
- [✓] Original data
- [✓] Predictions
- [✓] Probabilities

	Random Forest	Tree	iris	sepal length
71	0.00 : 0.67 : 0.33 → Iris-vers...	0.00 : 0.02 : 0.98 → Iris-virgi...	Iris-versicolor	5.900
72	0.00 : 1.00 : 0.00 → Iris-versi...	0.00 : 0.98 : 0.02 → Iris-vers...	Iris-versicolor	6.100
73	0.00 : 0.61 : 0.39 → Iris-versi...	0.00 : 0.98 : 0.02 → Iris-vers...	Iris-versicolor	6.300
74	0.00 : 1.00 : 0.00 → Iris-versi...	0.00 : 0.98 : 0.02 → Iris-vers...	Iris-versicolor	6.100
75	0.00 : 1.00 : 0.00 → Iris-versi...	0.00 : 0.98 : 0.02 → Iris-vers...	Iris-versicolor	6.400
76	0.00 : 1.00 : 0.00 → Iris-versi...	0.00 : 0.98 : 0.02 → Iris-vers...	Iris-versicolor	6.600
77	0.00 : 0.97 : 0.03 → Iris-vers...	0.00 : 0.98 : 0.02 → Iris-vers...	Iris-versicolor	6.800
78	0.00 : 0.71 : 0.29 → Iris-versi...	0.00 : 0.67 : 0.33 → Iris-vers...	Iris-versicolor	6.700
79	0.00 : 0.99 : 0.01 → Iris-versi...	0.00 : 0.98 : 0.02 → Iris-vers...	Iris-versicolor	6.000
80	0.00 : 1.00 : 0.00 → Iris-versi...	0.00 : 0.98 : 0.02 → Iris-vers...	Iris-versicolor	5.700
81	0.00 : 1.00 : 0.00 → Iris-versi...	0.00 : 0.98 : 0.02 → Iris-vers...	Iris-versicolor	5.500
82	0.00 : 1.00 : 0.00 → Iris-versi...	0.00 : 0.98 : 0.02 → Iris-vers...	Iris-versicolor	5.500
83	0.00 : 1.00 : 0.00 → Iris-versi...	0.00 : 0.98 : 0.02 → Iris-vers...	Iris-versicolor	5.800
84	0.00 : 0.33 : 0.67 → Iris-virgi...	0.00 : 0.67 : 0.33 → Iris-vers...	Iris-versicolor	6.000
85	0.00 : 1.00 : 0.00 → Iris-versi...	0.00 : 0.98 : 0.02 → Iris-vers...	Iris-versicolor	5.400
86	0.00 : 0.89 : 0.11 → Iris-versi...	0.00 : 0.98 : 0.02 → Iris-vers...	Iris-versicolor	6.000
87	0.00 : 1.00 : 0.00 → Iris-versi...	0.00 : 0.98 : 0.02 → Iris-vers...	Iris-versicolor	6.700
88	0.00 : 1.00 : 0.00 → Iris-versi...	0.00 : 0.98 : 0.02 → Iris-vers...	Iris-versicolor	6.300
89	0.00 : 1.00 : 0.00 → Iris-versi...	0.00 : 0.98 : 0.02 → Iris-vers...	Iris-versicolor	5.600
90	0.00 : 1.00 : 0.00 → Iris-versi...	0.00 : 0.98 : 0.02 → Iris-vers...	Iris-versicolor	5.500
91	0.00 : 1.00 : 0.00 → Iris-versi...	0.00 : 0.98 : 0.02 → Iris-vers...	Iris-versicolor	5.500
92	0.00 : 1.00 : 0.00 → Iris-versi...	0.00 : 0.98 : 0.02 → Iris-vers...	Iris-versicolor	6.100

For regressions tasks, we will use *housing* data. Here, we will compare different models, namely **Random Forest**, Linear Regression and Constant, in the Test&Score widget.



Test & Score

Sampling
- (●) Cross validation
  - Number of folds: 10
  - [✓] Stratified
- ( ) Random sampling
  - Repeat train/test: 10
  - Training set size: 66 %
  - [✓] Stratified
- ( ) Leave one out
- ( ) Test on train data
- ( ) Test on test data

Evaluation Results

Method	MSE	RMSE	MAE	R2
Random Forest	11.372	3.372	2.309	0.865
Linear Regression	23.370	4.834	3.376	0.723
Constant	84.644	9.200	6.662	-0.003

Random Forest

Name: Random Forest

Basic Properties
- Number of trees: 10
- [ ] Number of attributes considered at each split: 5
- [ ] Fixed seed for random generator: 0

Growth Control
- [ ] Limit depth of individual trees: 3
- [✓] Do not split subsets smaller than: 5

[✓] Apply Automatically

References

Breiman, L. (2001). Random Forests. In Machine Learning, 45(1), 5-32. Available here

Save Model

Save a trained model to an output file.

Signals

Inputs:

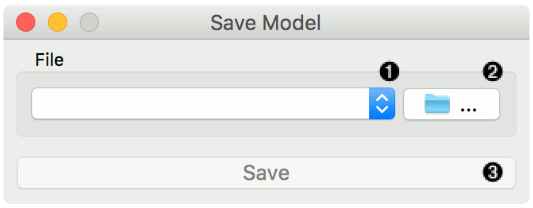
- **Model**

  A model with selected parameters

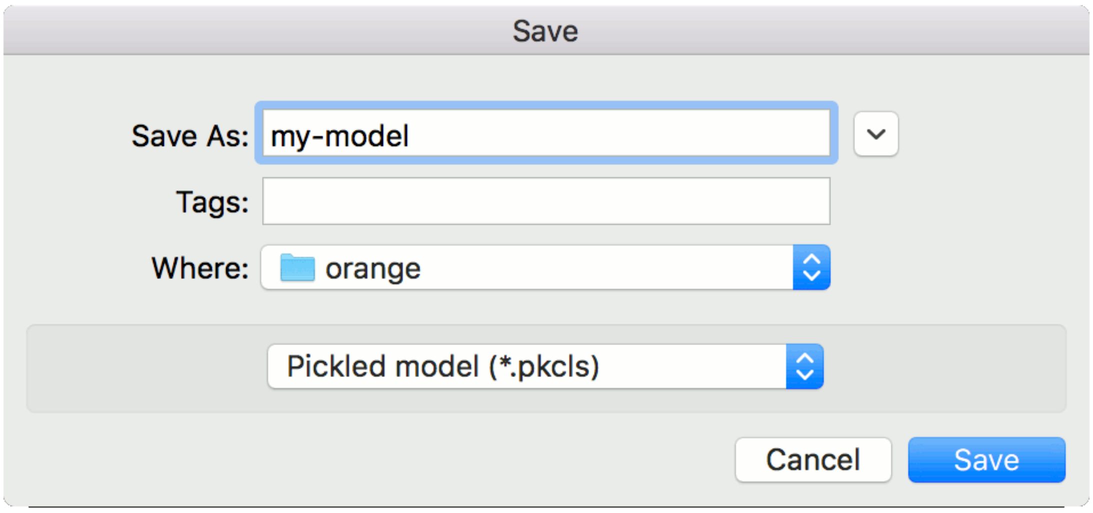
Outputs:

- None

Description



1. Choose from previously saved models.
2. Save the created model with the *Browse* icon. Click on the icon and enter the name of the file. The model will be saved to a pickled file.



Save As: my-model

Tags:

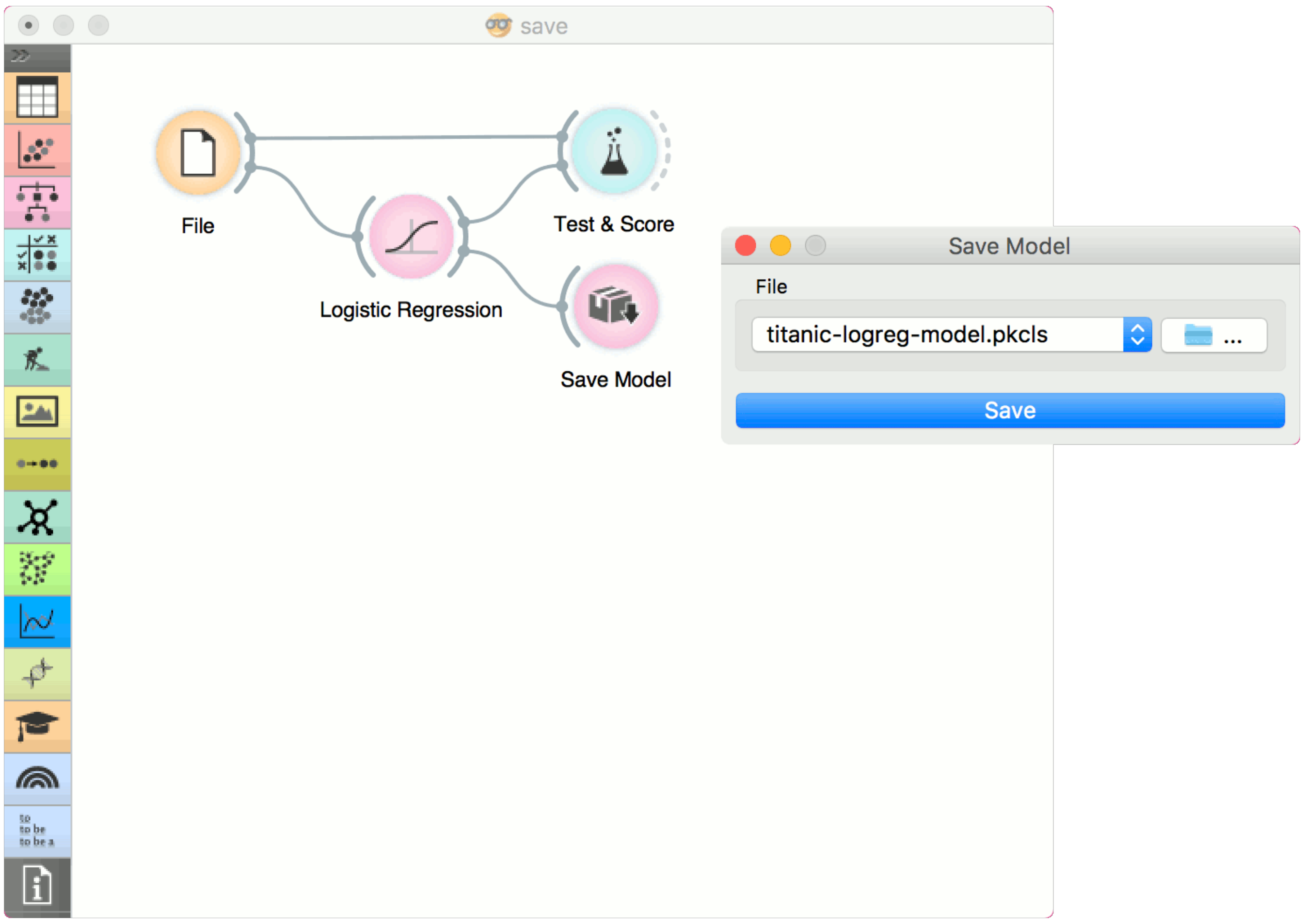
Where: orange

Pickled model (*.pkcls)

3. Save the model.

Example

When you want to save a custom-set model, feed the data to the model (e.g. Logistic Regression) and connect it to **Save Model**. Name the model; load it later into workflows with Load Model. Data sets used with **Load Model** have to contain compatible attributes.



Stochastic Gradient Descent

Minimize an objective function using a stochastic approximation of gradient descent.

Signals

Inputs:

- **Data**

  A data set.

- **Preprocessor**

  Preprocessing method(s)

Outputs:

- **Learner**

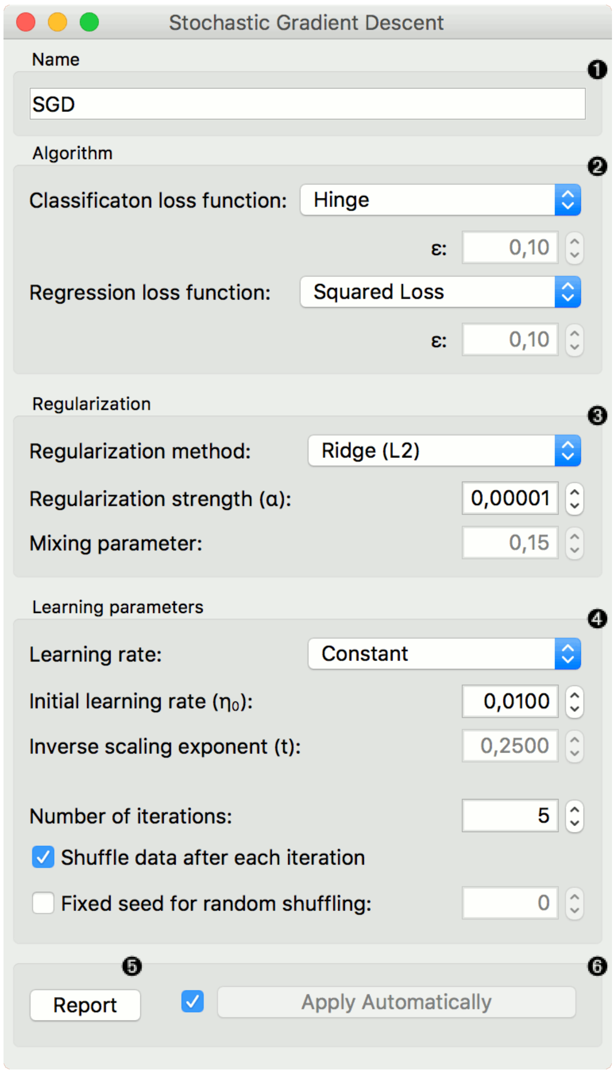
  A SGD learning algorithm with settings as specified in the dialog.

- **Model**

  A trained model. Output signal sent only if input *Data* is present.

Description

The **Stochastic Gradient Descent** widget uses stochastic gradient descent that minimizes a chosen loss function with a linear function. The algorithm approximates a true gradient by considering one sample at a time, and simultaneously updates the model based on the gradient of the loss function. For regression, it returns predictors as minimizers of the sum, i.e. M-estimators, and is especially useful for large-scale and sparse data sets.



Name

SGD

Algorithm

Classificaton loss function: Hinge

ε: 0,10

Regression loss function: Squared Loss

ε: 0,10

Regularization

Regularization method: Ridge (L2)

Regularization strength (α): 0,00001

Mixing parameter: 0,15

Learning parameters

Learning rate: Constant

Initial learning rate (η_0): 0,0100

Inverse scaling exponent (t): 0,2500

Number of iterations: 5

☑ Shuffle data after each iteration

☐ Fixed seed for random shuffling: 0

1. Specify the name of the model. The default name is "SGD".

2. Algorithm parameters. Classification loss function:

   - Hinge (linear SVM)
   - Logistic Regression (logistic regression SGD)
   - Modified Huber (smooth loss that brings tolerance to outliers as well as probability estimates)
   - *Squared Hinge* (quadratically penalized hinge)
   - Perceptron (linear loss used by the perceptron algorithm)
   - Squared Loss (fitted to ordinary least-squares)
   - Huber (switches to linear loss beyond ε)
   - Epsilon insensitive (ignores errors within ε, linear beyond it)
   - *Squared epsilon insensitive* (loss is squared beyond ε-region).

   Regression loss function:

   - Squared Loss (fitted to ordinary least-squares)
   - Huber (switches to linear loss beyond ε)
   - Epsilon insensitive (ignores errors within ε, linear beyond it)
   - *Squared epsilon insensitive* (loss is squared beyond ε-region).

3. Regularization norms to prevent overfitting:

   - None.
   - Lasso (L1) (L1, leading to sparse solutions)
   - Ridge (L2) (L2, standard regularizer)
   - Elastic net (mixing both penalty norms).

   Regularization strength defines how much regularization will be applied (the less we regularize, the more we allow the model to fit the data) and the mixing parameter what the ratio between L1 and L2 loss will be (if set to 0 then the loss is L2, if set to 1 then it is L1).

4. Learning parameters.

   - Learning rate:

     - *Constant*: learning rate stays the same through all epochs (passes)
     - Optimal: a heuristic proposed by Leon Bottou
     - Inverse scaling: earning rate is inversely related to the number of iterations

   - Initial learning rate.

   - Inverse scaling exponent: learning rate decay.

   - Number of iterations: the number of passes through the training data.

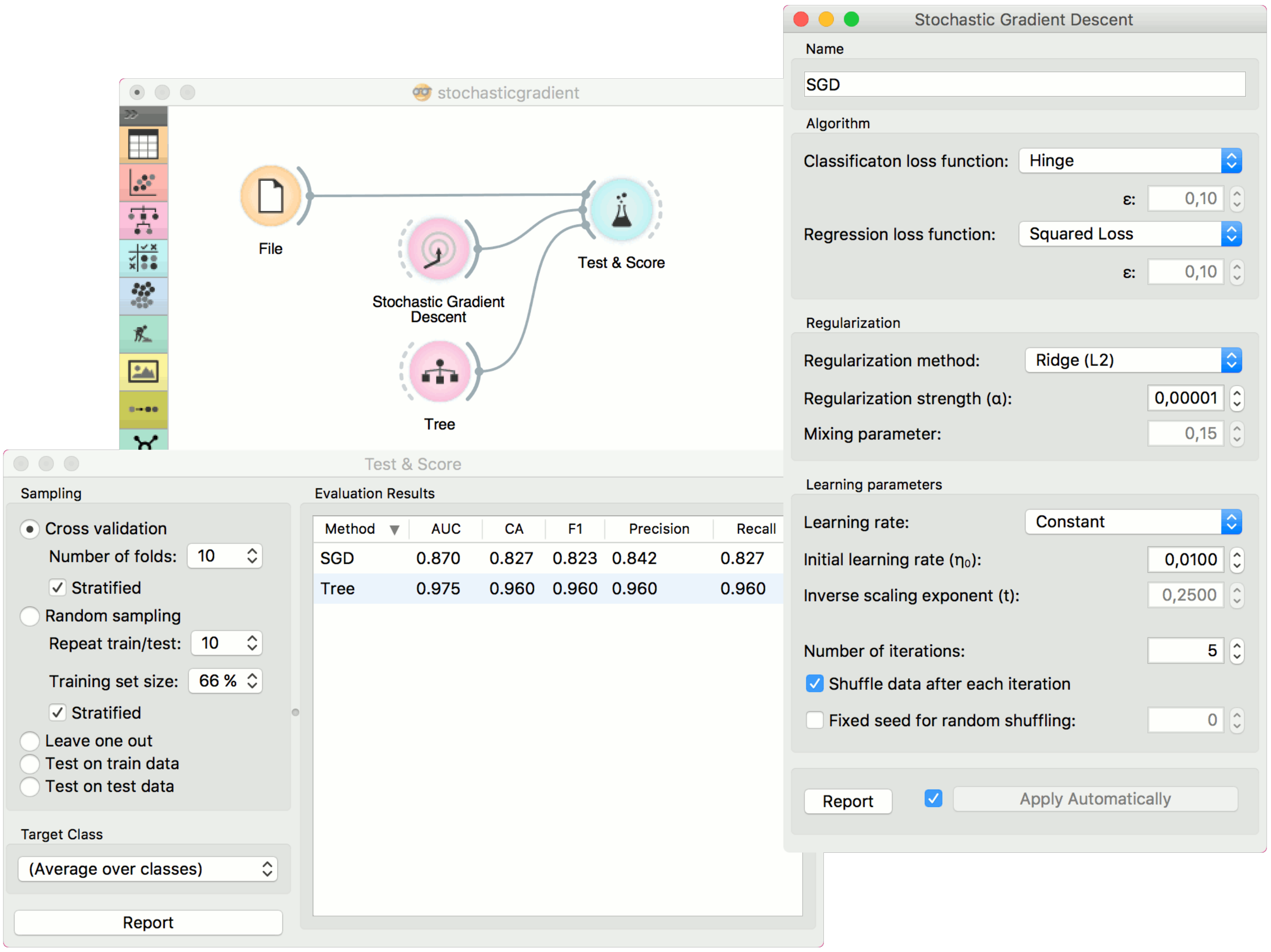
   - If *Shuffle data after each iteration* is on, the order of data instances is mixed after each pass.

   - If *Fixed seed for random shuffling* is on, the algorithm will use a fixed random seed and enable replicating the results.

7. Produce a report.
8. Press *Apply* to commit changes. Alternatively, tick the box on the left side of the *Apply* button and changes will be communicated automatically.

Examples

For the classification task, we will use *iris* data set and test two models on it. We connected Stochastic Gradient Descent and Tree to Test & Score. We also connected File to **Test & Score** and observed model performance in the widget.



Stochastic Gradient Descent

Name

SGD

Algorithm

Classificaton loss function: Hinge

ε: 0,10

Regression loss function: Squared Loss

ε: 0,10

Regularization

Regularization method: Ridge (L2)

Regularization strength (α): 0,00001

Mixing parameter: 0,15

Learning parameters

Learning rate: Constant

Initial learning rate (η_0): 0,0100

Inverse scaling exponent (t): 0,2500

Number of iterations: 5

☑ Shuffle data after each iteration

☐ Fixed seed for random shuffling: 0

Report ☑ Apply Automatically

Test & Score

Sampling

● Cross validation

Number of folds: 10

✓ Stratified

○ Random sampling

Repeat train/test: 10

Training set size: 66 %

✓ Stratified

○ Leave one out

○ Test on train data

○ Test on test data

Target Class

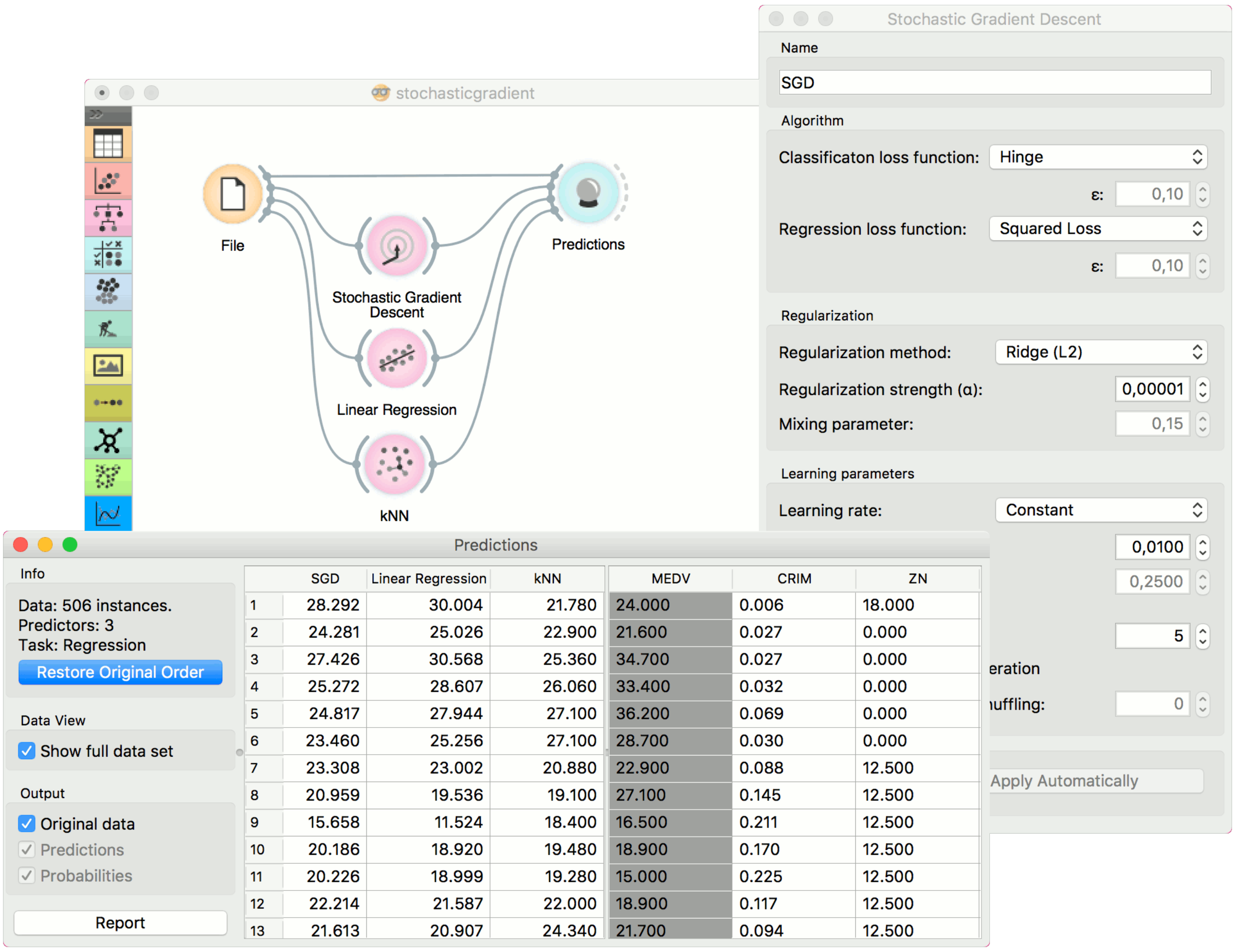
(Average over classes)

Report

Evaluation Results

Method	AUC	CA	F1	Precision	Recall
SGD	0.870	0.827	0.823	0.842	0.827
Tree	0.975	0.960	0.960	0.960	0.960

For the regression task, we will compare three different models to see which predict what kind of results. For the purpose of this example, the *housing* data set is used. We connect the File widget to **Stochastic Gradient Descent**, Linear Regression and kNN widget and all four to the Predictions widget.



Stochastic Gradient Descent

Name

SGD

Algorithm

Classificaton loss function: Hinge

ε: 0,10

Regression loss function: Squared Loss

ε: 0,10

Regularization

Regularization method: Ridge (L2)

Regularization strength (α): 0,00001

Mixing parameter: 0,15

Learning parameters

Learning rate: Constant

0,0100

0,2500

5

eration

ıuffling: 0

Apply Automatically

Predictions

Info

Data: 506 instances.
Predictors: 3
Task: Regression

Restore Original Order

Data View

☑ Show full data set

Output

☑ Original data
☑ Predictions
☑ Probabilities

Report

	SGD	Linear Regression	kNN	MEDV	CRIM	ZN
1	28.292	30.004	21.780	24.000	0.006	18.000
2	24.281	25.026	22.900	21.600	0.027	0.000
3	27.426	30.568	25.360	34.700	0.027	0.000
4	25.272	28.607	26.060	33.400	0.032	0.000
5	24.817	27.944	27.100	36.200	0.069	0.000
6	23.460	25.256	27.100	28.700	0.030	0.000
7	23.308	23.002	20.880	22.900	0.088	12.500
8	20.959	19.536	19.100	27.100	0.145	12.500
9	15.658	11.524	18.400	16.500	0.211	12.500
10	20.186	18.920	19.480	18.900	0.170	12.500
11	20.226	18.999	19.280	15.000	0.225	12.500
12	22.214	21.587	22.000	18.900	0.117	12.500
13	21.613	20.907	24.340	21.700	0.094	12.500

SVM

Support Vector Machines map inputs to higher-dimensional feature spaces.

Signals

Inputs:

- **Data**

  A data set.

- **Preprocessor**

  Preprocessing method(s)

Outputs:

- **Learner**

  A support vector machine learning algorithm with settings as specified in the dialog.

- **Model**

  A trained model. Output signal sent only if input *Data* is present.

- **Support Vectors**

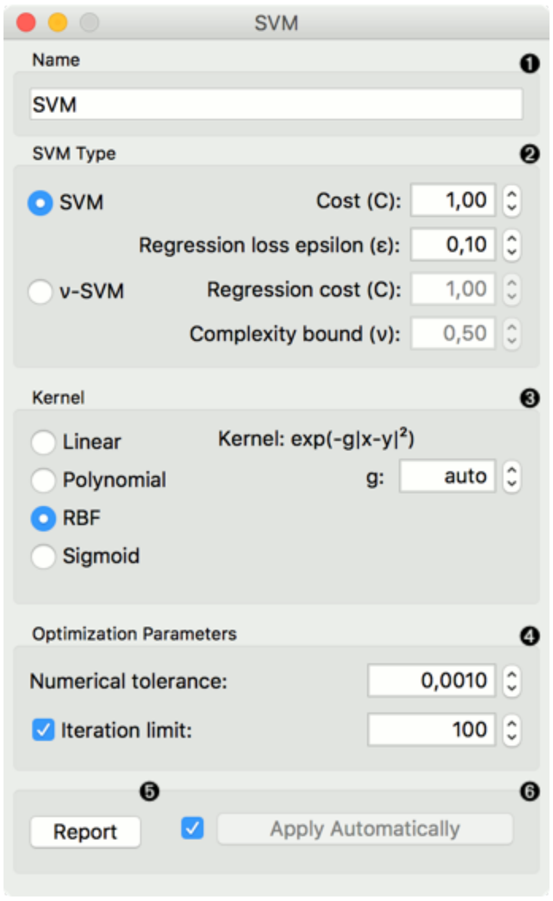
  A subset of data instances from the training set that were used as support vectors in the trained model.

Description

Support vector machine (SVM) is a machine learning technique that separates the attribute space with a hyperplane, thus maximizing the margin between the instances of different classes or class values. The technique often yields supreme predictive performance results. Orange embeds a popular implementation of SVM from the LIBSVM package. This widget is its graphical user interface.

For regression tasks, **SVM** performs linear regression in a high dimension feature space using an ε-insensitive loss. Its estimation accuracy depends on a good setting of C, ε and kernel parameters. The widget outputs class predictions based on a SVM Regression.

The widget works for both classification and regression tasks.



1. The learner can be given a name under which it will appear in other widgets. The default name is "SVM".

2. SVM type with test error settings. *SVM* and *ν-SVM* are based on different minimization of the error function. On the right side, you can set test error bounds:

   - SVM:

     - Cost: penalty term for loss and applies for classification and regression tasks.
     - ε: a parameter to the epsilon-SVR model, applies to regression tasks. Defines the distance from true values within which no penalty is associated with predicted values.

   - ν-SVM:

     - Cost: penalty term for loss and applies only to regression tasks
     - ν: a parameter to the ν-SVR model, applies to classification and regression tasks. An upper bound on the fraction of training errors and a lower bound of the fraction of support vectors.

3. Kernel is a function that transforms attribute space to a new feature space to fit the maximum-margin hyperplane, thus allowing the algorithm to create the model with:

   - Linear
   - Polynomial
   - RBF and
   - Sigmoid

   kernels. Functions that specify the kernel are presented upon selecting them, and the constants involved are:

   - **g** for the gamma constant in kernel function (the recommended value is 1/k, where k is the number of the attributes, but since there may be no training set given to the widget the default is 0 and the user has to set this option manually),
   - **c** for the constant c0 in the kernel function (default 0), and
   - **d** for the degree of the kernel (default 3).

4. Set permitted deviation from the expected value in *Numerical Tolerance*. Tick the box next to *Iteration Limit* to set the maximum number of iterations permitted.

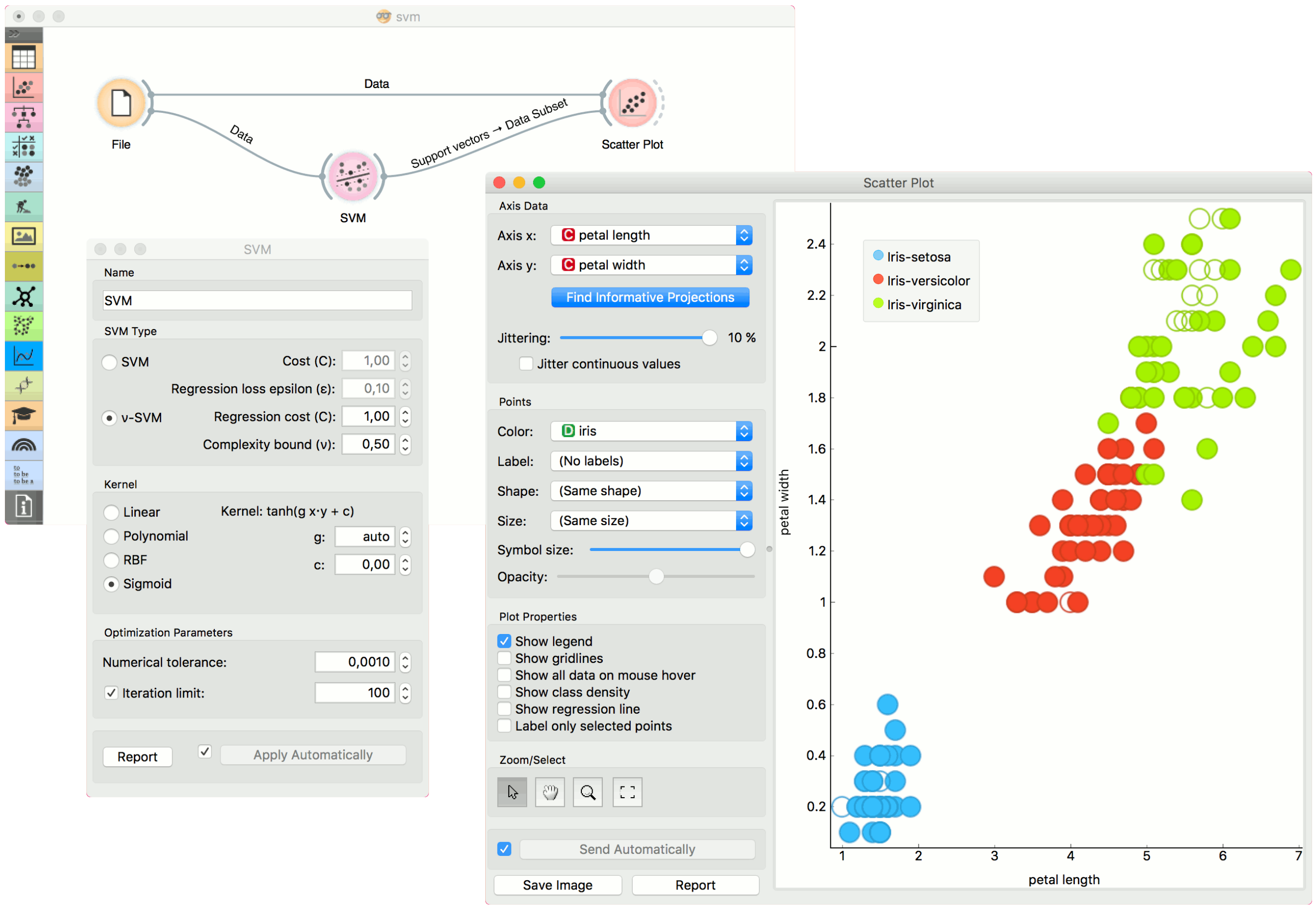
5. Produce a report.

6. Click *Apply* to commit changes. If you tick the box on the left side of the *Apply* button, changes will be communicated automatically.

Examples

In the first (regression) example, we have used *housing* data set and split the data into two data subsets (*Data Sample* and *Remaining Data*) with Data Sampler. The sample was sent to SVM which produced a *Model*, which was then used in Predictions to predict the values in *Remaining Data*. A similar schema can be used if the data is already in two separate files; in this case, two File widgets would be used instead of the File - Data Sampler combination.

The second example shows how to use **SVM** in combination with Scatterplot. The following workflow trains a SVM model on *iris* data and outputs support vectors, which are those data instances that were used as support vectors in the learning phase. We can observe which are these data instances in a scatter plot visualization. Note that for the workflow to work correctly, you must set the links between widgets as demonstrated in the screenshot below.



SVM

Name: SVM

SVM Type
- SVM — Cost (C): 1,00; Regression loss epsilon (ε): 0,10
- [✓ v-SVM] — Regression cost (C): 1,00; Complexity bound (ν): 0,50

Kernel
- Linear
- Polynomial
- RBF
- [✓ Sigmoid]

Kernel: tanh(g x·y + c)
g: auto
c: 0,00

Optimization Parameters
Numerical tolerance: 0,0010
✓ Iteration limit: 100

Scatter Plot

Axis Data
Axis x: petal length
Axis y: petal width
Jittering: 10 %
Jitter continuous values

Points
Color: iris
Label: (No labels)
Shape: (Same shape)
Size: (Same size)
Symbol size:
Opacity:

Plot Properties
✓ Show legend
Show gridlines
Show all data on mouse hover
Show class density
Show regression line
Label only selected points

Zoom/Select

References

Introduction to SVM on StatSoft.

Tree

A tree algorithm with forward pruning.

Signals

Inputs:

- **Data**

  A data set

- **Preprocessor**

  Preprocessing method(s)

Outputs:

- **Learner**

  A decision tree learning algorithm with settings as specified in the dialog.

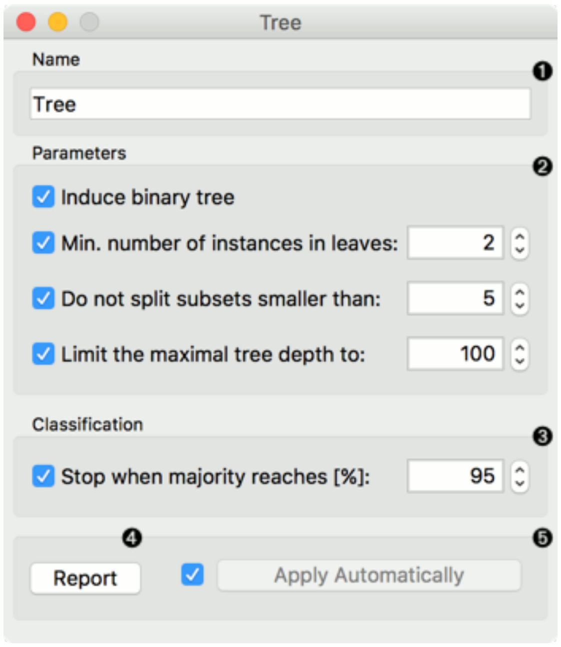
- **Model**

  A subset of data instances from the training set that were used as support vectors in the trained model.

Description

Tree is a simple algorithm that splits the data into nodes by class purity. It is a precursor to Random Forest. Tree in Orange is designed in-house and can handle both discrete and continuous data sets.

It can also be used for both classification and regression tasks.



Name: Tree

✓ Induce binary tree
✓ Min. number of instances in leaves: 2
✓ Do not split subsets smaller than: 5
✓ Limit the maximal tree depth to: 100
✓ Stop when majority reaches [%]: 95
✓ Apply Automatically

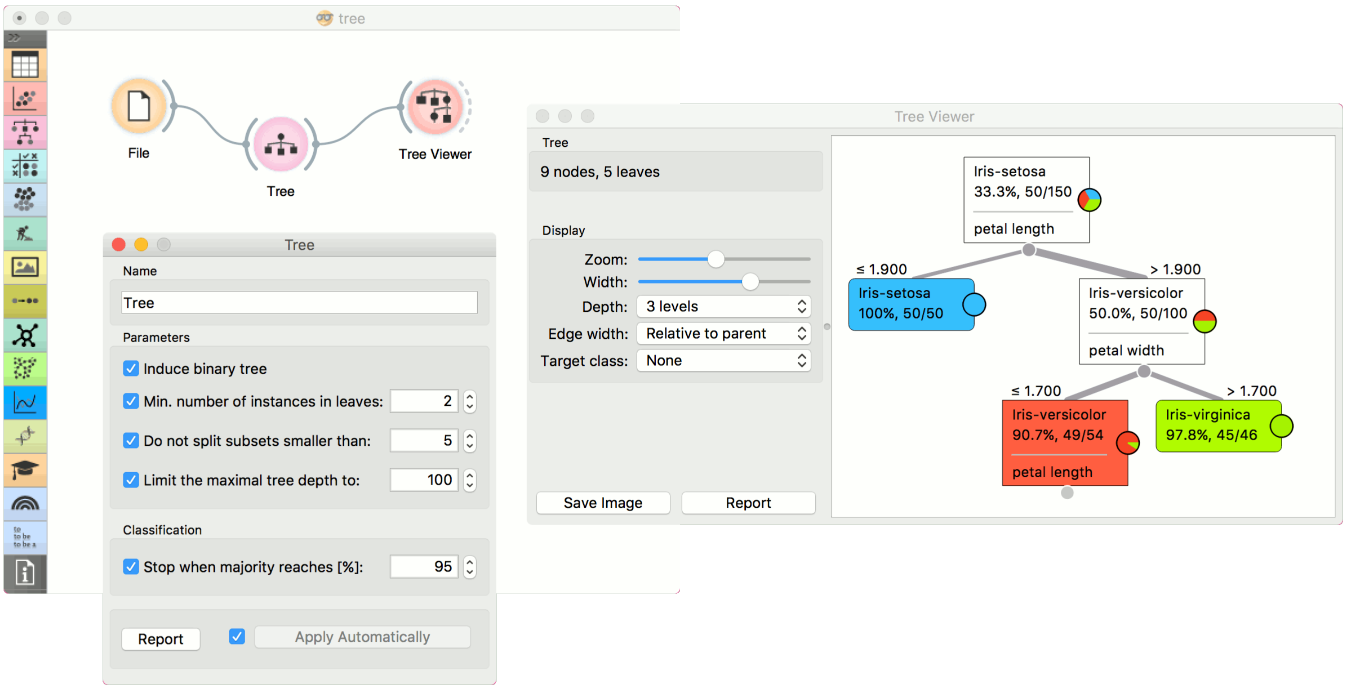
1. The learner can be given a name under which it will appear in other widgets. The default name is “Tree”.
2. Tree parameters: - **Induce binary tree**: build a binary tree (split into two child nodes) - **Min. number of instances in leaves**: if checked, the algorithm will never construct a split which would put less than the specified

number of training examples into any of the branches. - **Do not split subsets smaller than**: forbids the algorithm to split the nodes with less than the given number of instances. - **Limit the maximal tree depth**: limits the depth of the classification tree to the specified number of node levels.

3. **Stop when majority reaches [%]**: stop splitting the nodes after a specified majority threshold is reached
4. Produce a report. After changing the settings, you need to click *Apply*, which will put the new learner on the output and, if the training examples are given, construct a new classifier and output it as well. Alternatively, tick the box on the left and changes will be communicated automatically.

Examples

There are two typical uses for this widget. First, you may want to induce a model and check what it looks like in Tree Viewer.



Tree

Name

Tree

Parameters

✓ Induce binary tree

✓ Min. number of instances in leaves: 2

✓ Do not split subsets smaller than: 5

✓ Limit the maximal tree depth to: 100

Classification

✓ Stop when majority reaches [%]: 95

Tree Viewer

Tree

9 nodes, 5 leaves

Display

Zoom:

Width:

Depth: 3 levels

Edge width: Relative to parent

Target class: None

Iris-setosa
33.3%, 50/150
petal length

≤ 1.900

Iris-setosa
100%, 50/50

> 1.900

Iris-versicolor
50.0%, 50/100
petal width

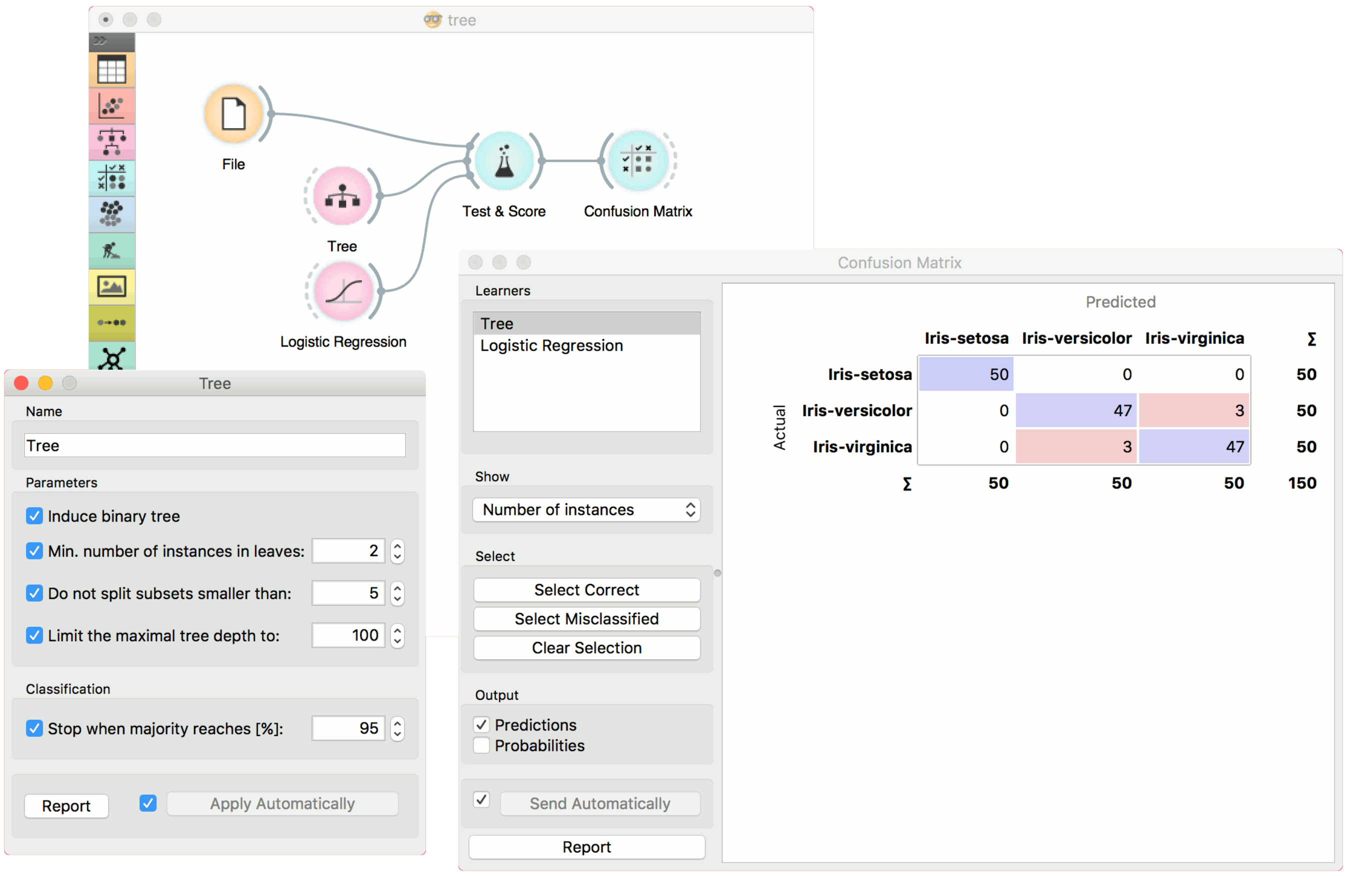
≤ 1.700

Iris-versicolor
90.7%, 49/54
petal length

> 1.700

Iris-virginica
97.8%, 45/46

The second schema trains a model and evaluates its performance against Logistic Regression.



Tree

Name: Tree

Parameters
- [✓] Induce binary tree
- [✓] Min. number of instances in leaves: 2
- [✓] Do not split subsets smaller than: 5
- [✓] Limit the maximal tree depth to: 100

Classification
- [✓] Stop when majority reaches [%]: 95

Report [✓] Apply Automatically

Confusion Matrix

Learners: Tree, Logistic Regression

Show: Number of instances

Select: Select Correct, Select Misclassified, Clear Selection

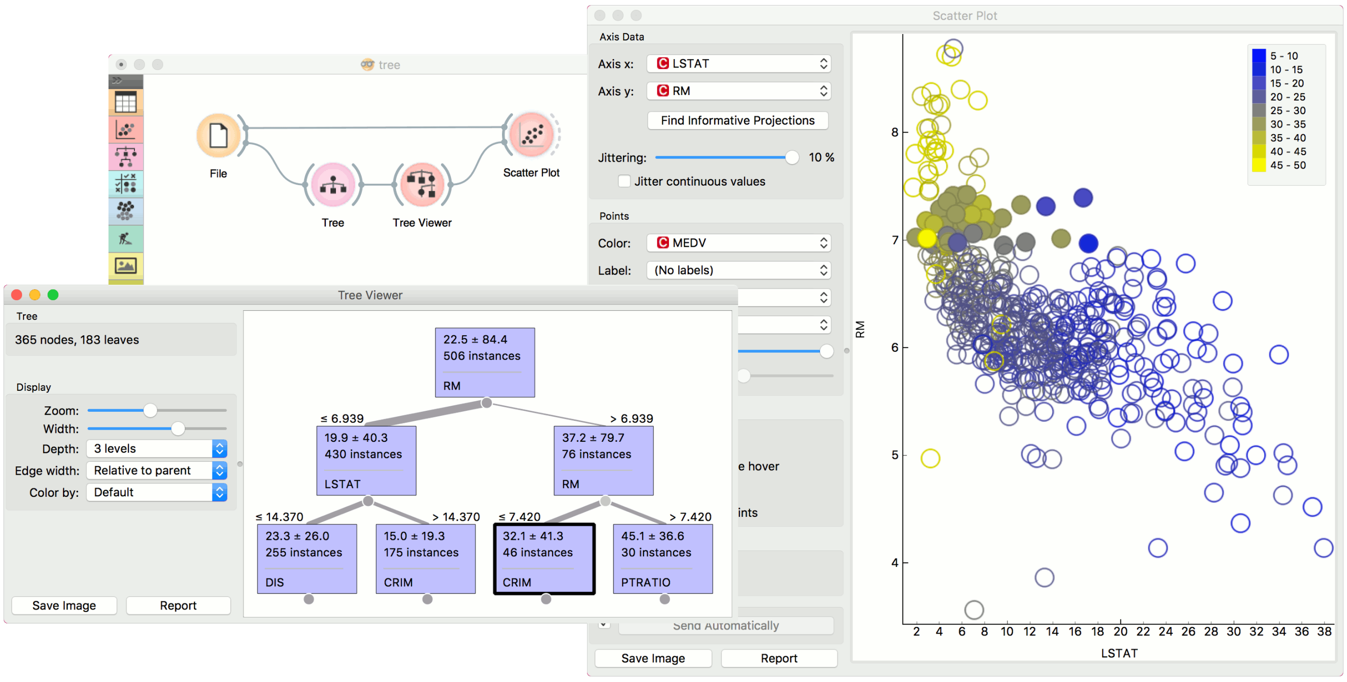
Output
- [✓] Predictions
- [ ] Probabilities

[✓] Send Automatically

Report

Actual \ Predicted	Iris-setosa	Iris-versicolor	Iris-virginica	Σ
Iris-setosa	50	0	0	50
Iris-versicolor	0	47	3	50
Iris-virginica	0	3	47	50
Σ	50	50	50	150

We used the *iris* data set in both examples. However, **Tree** works for regression tasks as well. Use *housing* data set and pass it to **Tree**. The selected tree node from Tree Viewer is presented in the Scatter Plot and we can see that the selected examples exhibit the same features.



Tree Viewer

Tree: 365 nodes, 183 leaves

Display
- Zoom:
- Width:
- Depth: 3 levels
- Edge width: Relative to parent
- Color by: Default

Save Image, Report

Scatter Plot

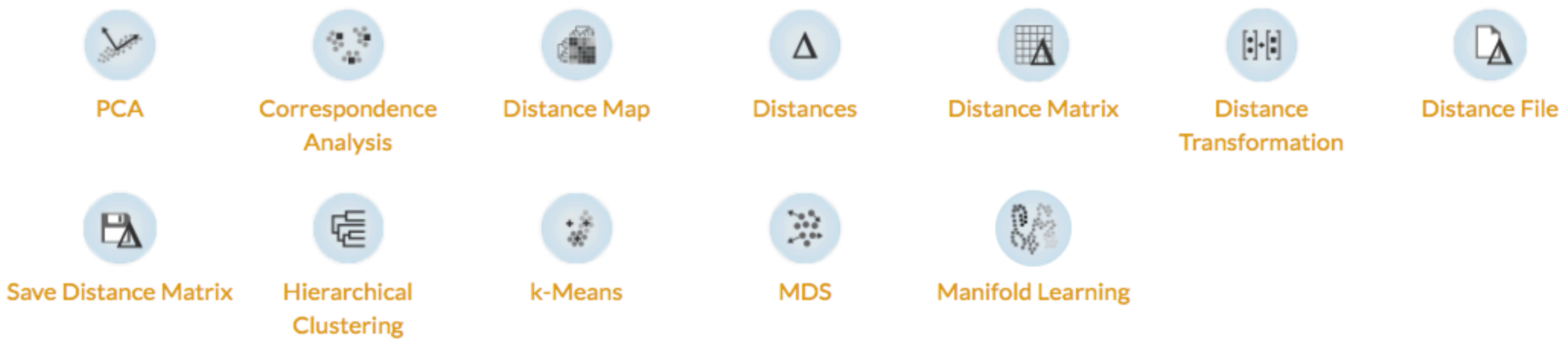
Axis Data
- Axis x: LSTAT
- Axis y: RM
- Find Informative Projections
- Jittering: 10 %
- [ ] Jitter continuous values

Points
- Color: MEDV
- Label: (No labels)

Send Automatically

Save Image, Report

Unsupervised



Correspondence Analysis

Signals

Inputs:

- **Data**

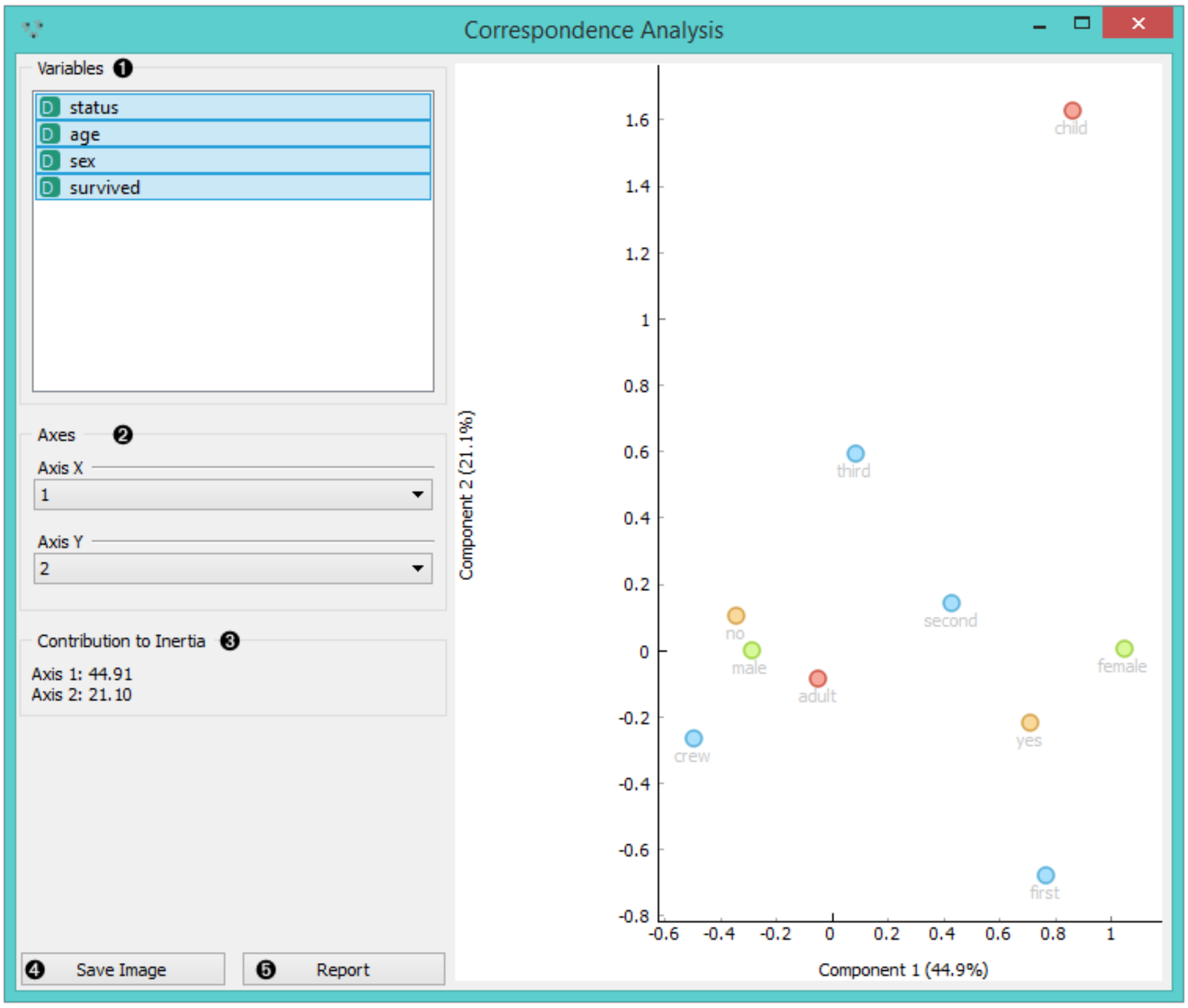
  A data set.

Outputs:

- None

Description

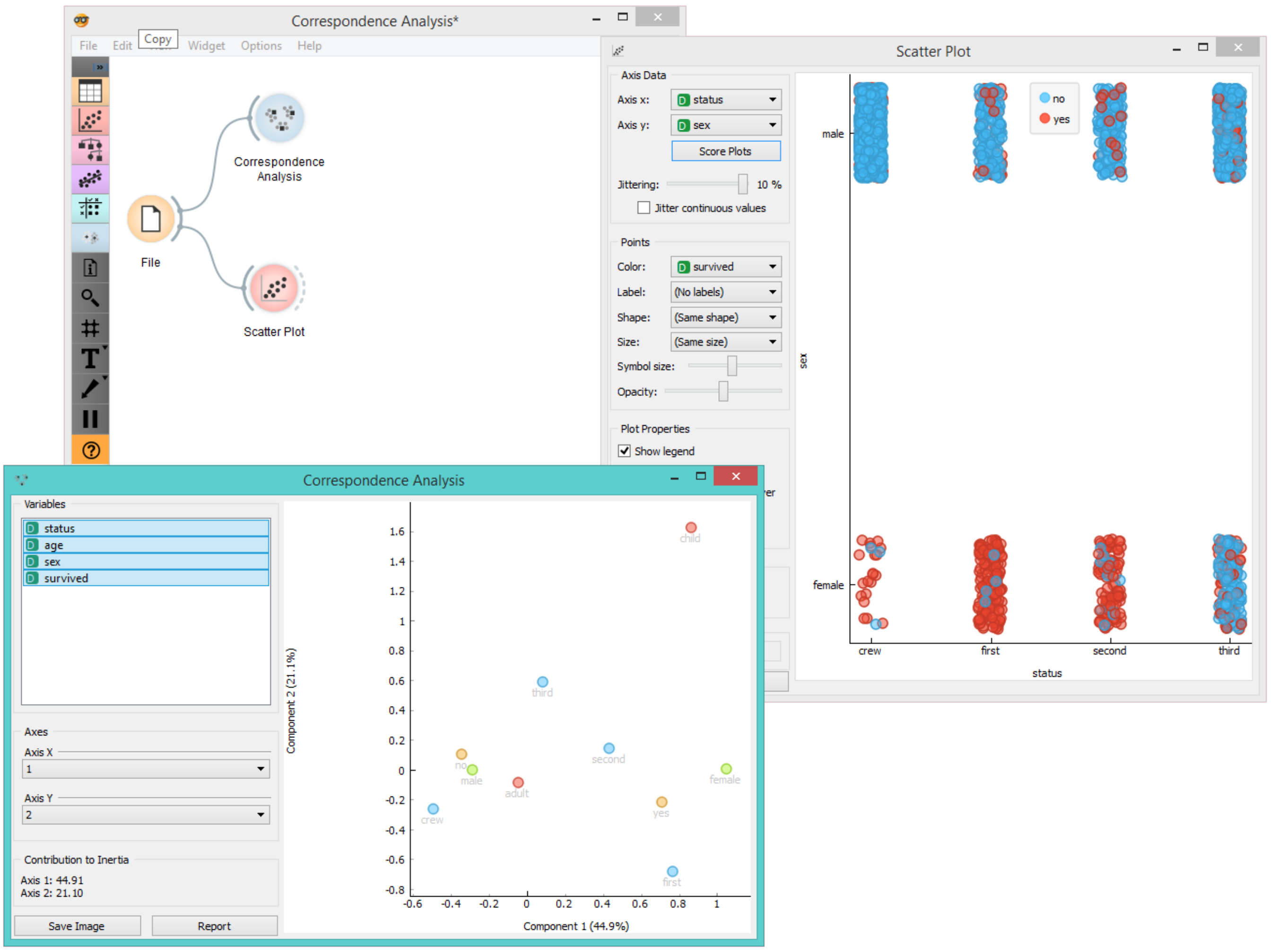
Correspondence Analysis (CA) computes the CA linear transformation of the input data. While it is similar to PCA, CA computes linear transformation on discrete rather than on continuous data.



1. Select the variables you want to see plotted.
2. Select the component for each axis.
3. Inertia values (percentage of independence from transformation, i.e. variables are in the same dimension).
4. Produce a report.

Example

Below, is a simple comparison between the **Correspondence Analysis** and Scatter plot widgets on the *Titanic* data set. While the Scatter plot shows fairly well which class and sex had a good survival rate and which one didn't, **Correspondence Analysis** can plot several variables in a 2-D graph, thus making it easy to see the relations between variable values. It is clear from the graph that "no", "male" and "crew" are related to each other. The same goes for "yes", "female" and "first".



Distance File

Loads an existing distance file.

Signals

Inputs:

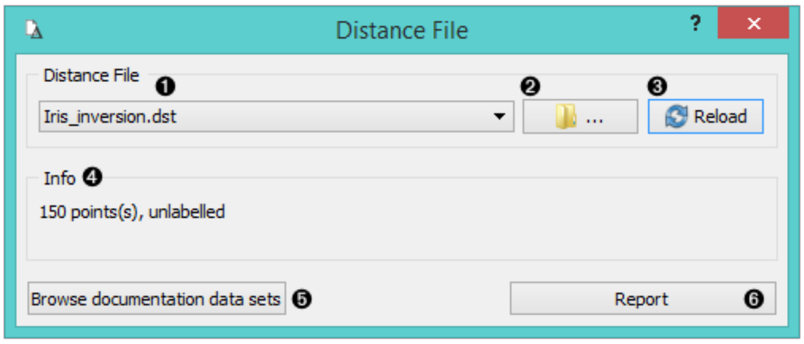
- None

Outputs:

- **Distance File**

  A distance matrix.

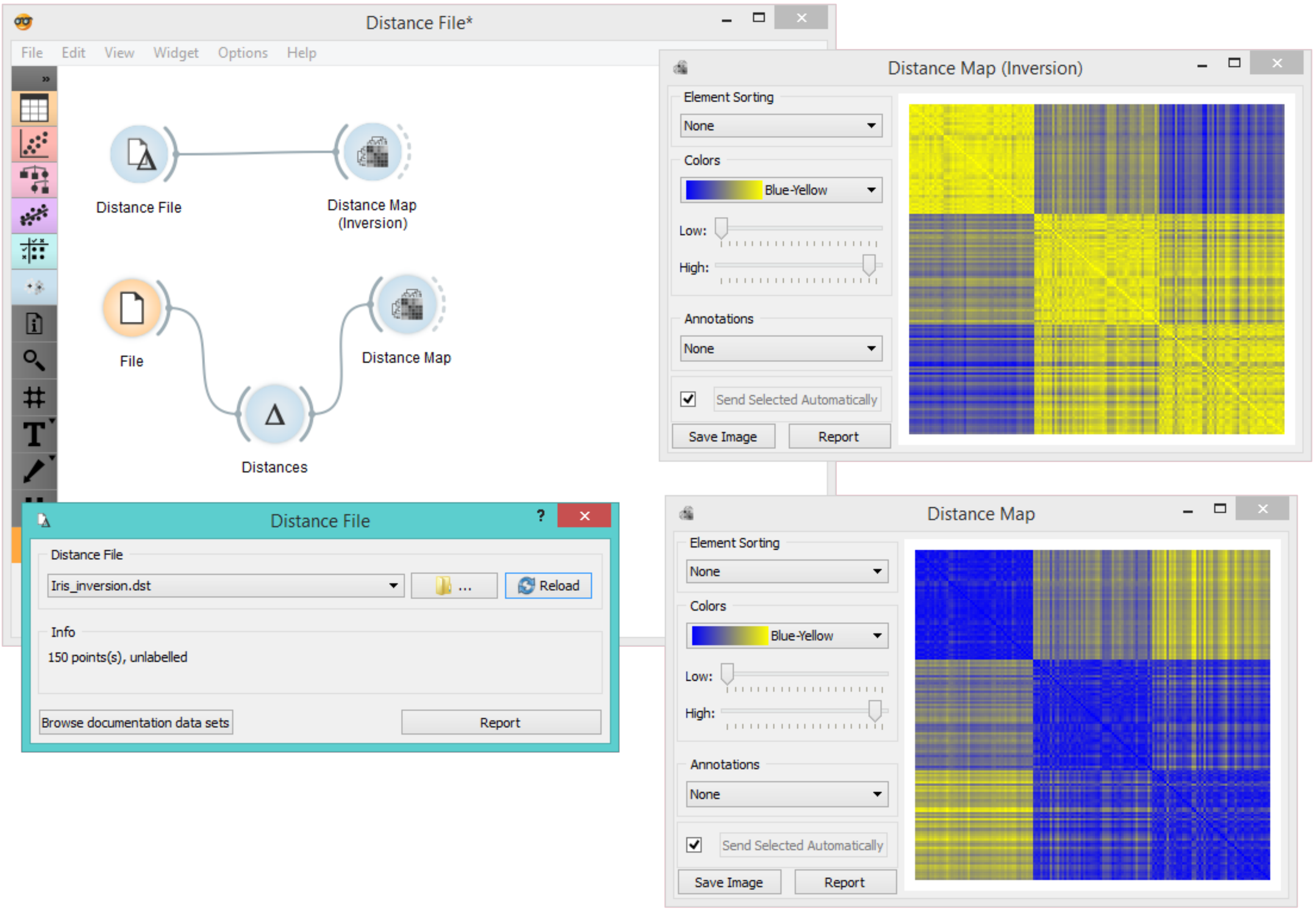
Description



1. Choose from a list of previously saved distance files.
2. Browse for saved distance files.
3. Reload the selected distance file.
4. Information about the distance file (number of points, labelled/unlabelled)
5. Browse documentation data sets.
6. Produce a report.

Example

When you want to use a custom-set distance file that you've saved before, open the **Distance File** widget and select the desired file with the *Browse* icon. This widget loads the existing distance file. In the snapshot below, we loaded the transformed *Iris* distance matrix from the Save Distance Matrix example. We displayed the transformed data matrix in the Distance Map widget. We also decided to display a distance map of the original *Iris* data set for comparison.


Distance File*
File Edit View Widget Options Help
Distance File
Distance Map (Inversion)
File
Distance Map
Distances
Distance Map (Inversion)
Element Sorting
None
Colors
Blue-Yellow
Low:
High:
Annotations
None
Send Selected Automatically
Save Image
Report
Distance File
Distance File
Iris_inversion.dst
...
Reload
Info
150 points(s), unlabelled
Browse documentation data sets
Report
Distance Map
Element Sorting
None
Colors
Blue-Yellow
Low:
High:
Annotations
None
Send Selected Automatically
Save Image
Report

Distance Map

Visualizes distances between items.

Signals

Inputs:

- **Distances**

  A distance matrix.

Outputs:

- **Data**

  Instances corresponding to the selected elements of the matrix.

- **Features**

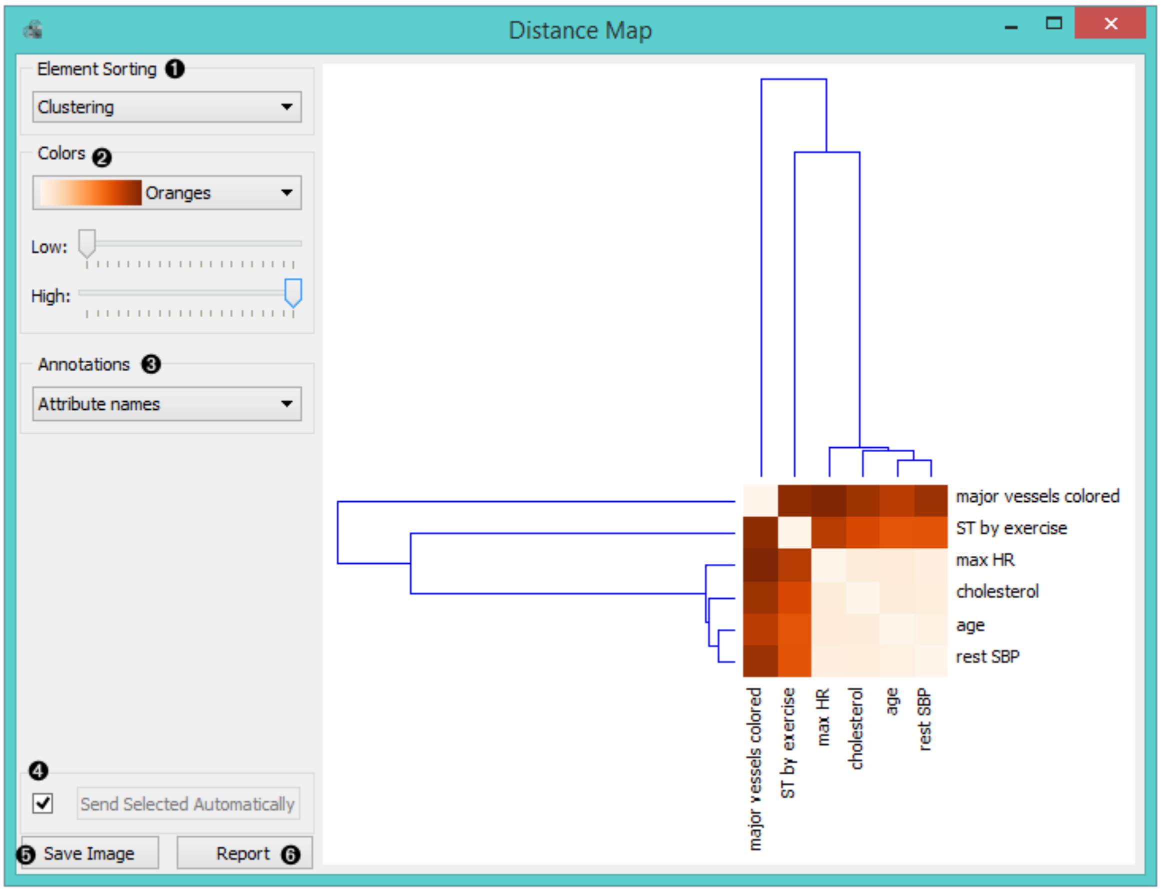
  Attributes corresponding to the selected elements of the matrix.

Description

The **Distance Map** visualizes distances between objects. The visualization is the same as if we printed out a table of numbers, except that the numbers are replaced by colored spots.

Distances are most often those between instances ("*rows*" in the Distances widget) or attributes ("*columns*" in Distances widget). The only suitable input for **Distance Map** is the Distances widget. For the output, the user can select a region of the map and the widget will output the corresponding instances or attributes. Also note that the **Distances** widget ignores discrete values and calculates distances only for continuous data, thus it can only display distance map for discrete data if you Continuize them first.

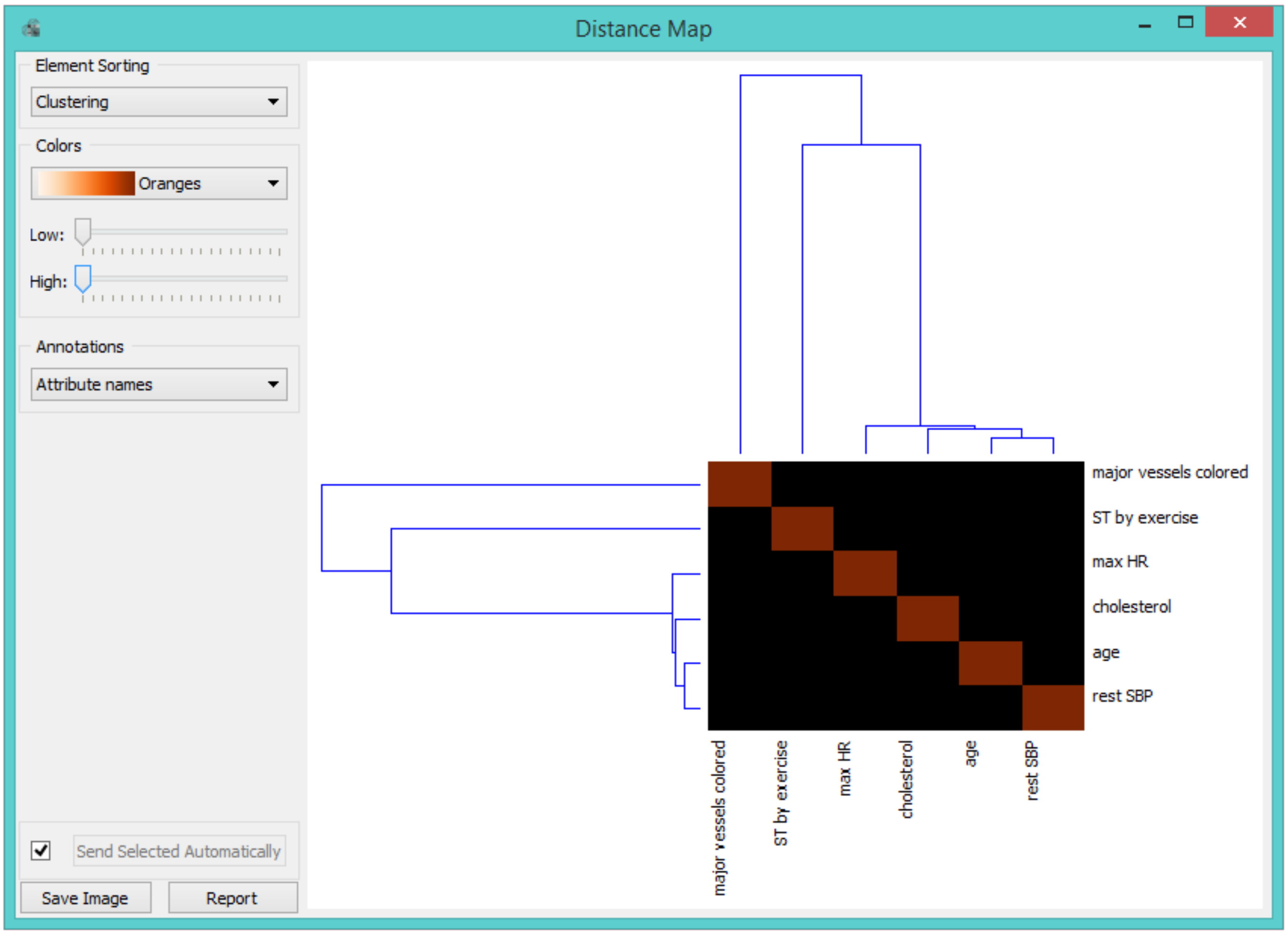
The snapshot shows distances between columns in the *heart disease* data, where smaller distances are represented with light and larger with dark orange. The matrix is symmetric and the diagonal is a light shade of orange - no attribute is different from itself. Symmetricity is always assumed, while the diagonal may also be non-zero.



1. *Element sorting* arranges elements in the map by
   - None (lists instances as found in the data set)
   - **Clustering** (clusters data by similarity)
   - **Clustering with ordered leaves** (maximizes the sum of similarities of adjacent elements)
2. *Colors*
   - **Colors** (select the color palette for your distance map)
   - **Low** and **High** are thresholds for the color palette (low for instances or attributes with low distances and high for instances or attributes with high distances).
3. Select *Annotations*.
4. If *Send Selected Automatically* is on, the data subset is communicated automatically, otherwise you need to press *Send Selected*.
5. Press *Save Image* if you want to save the created image to your computer.
6. Produce a report.

Normally, a color palette is used to visualize the entire range of distances appearing in the matrix. This can be changed by setting the low and high threshold. In this way we ignore the differences in distances outside this interval and visualize the interesting part of the distribution.

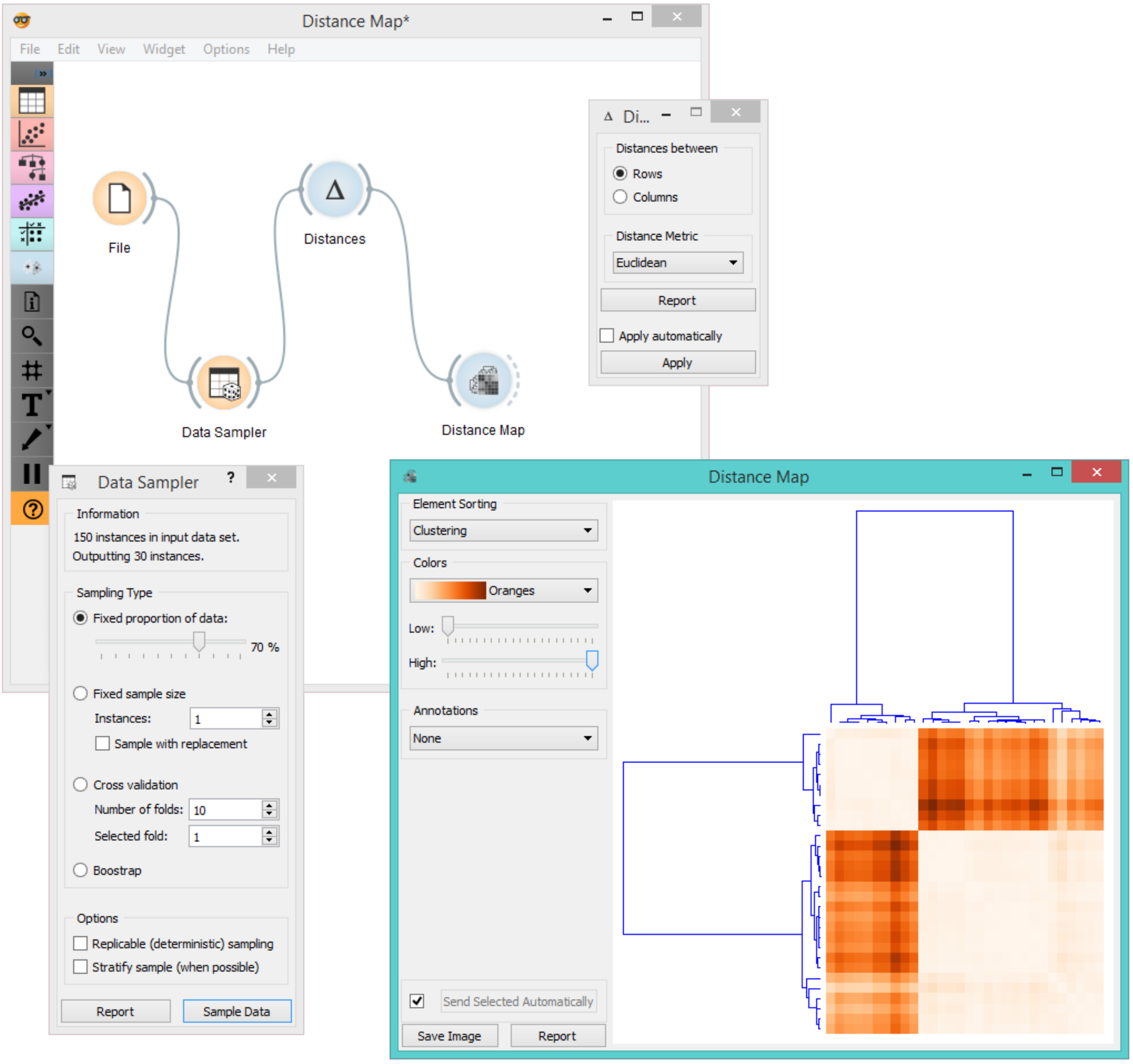
Below, we visualized the most correlated attributes (distances by columns) in the *heart disease* data set by setting the color threshold for high distances to the minimum. We get a predominantly black square, where attributes with the lowest distance scores are represented by a lighter shade of the selected color schema (in our case: orange). Beside the diagonal line, we see that in our example *ST by exercise* and *major vessels colored* are the two attributes closest together.



The user can select a region in the map with the usual click-and-drag of the cursor. When a part of the map is selected, the widget outputs all items from the selected cells.

Examples

The first workflow shows a very standard use of the **Distance Map** widget. We select 70% of the original *Iris* data as our sample and view the distances between rows in **Distance Map**.



Distances between
- [✓ Rows]
- Columns

Distance Metric: Euclidean

☐ Apply automatically

Information
150 instances in input data set.
Outputting 30 instances.

Sampling Type
- [✓ Fixed proportion of data:] 70 %
- Fixed sample size
  - Instances: 1
  - ☐ Sample with replacement
- Cross validation
  - Number of folds: 10
  - Selected fold: 1
- Boostrap

Options
- ☐ Replicable (deterministic) sampling
- ☐ Stratify sample (when possible)

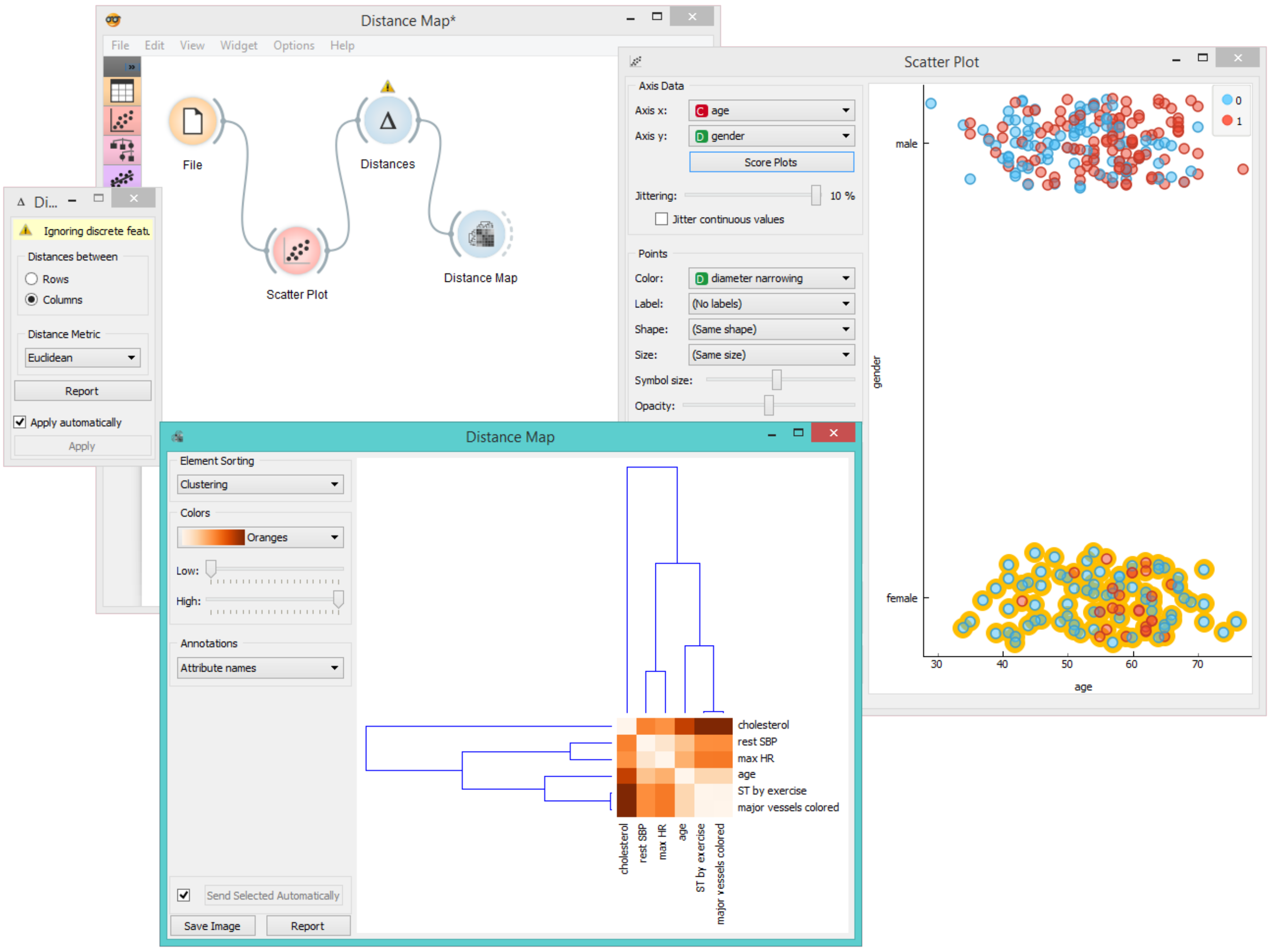
Element Sorting: Clustering

Colors: Oranges
Low:
High:

Annotations: None

☑ Send Selected Automatically

In the second example, we use the *heart disease* data again and select a subset of women only from the Scatter Plot. Then, we visualize distances between columns in the **Distance Map**. Since the subset also contains some discrete data, the Distances widget warns us it will ignore the discrete features, thus we will see only continuous instances/attributes in the map.


Ignoring discrete featu

Distances between: Rows [✓ Columns]

Distance Metric: Euclidean

[✓] Apply automatically

Axis x: age

Axis y: gender

Jittering: 10 %

Jitter continuous values

Color: diameter narrowing

Label: (No labels)

Shape: (Same shape)

Size: (Same size)

Element Sorting: Clustering

Colors: Oranges

Annotations: Attribute names

[✓] Send Selected Automatically

Distance Matrix

Visualizes distance measures in a distance matrix.

Signals

Inputs:

- **Distances**

  A distance matrix.

Outputs:

- **Distances**

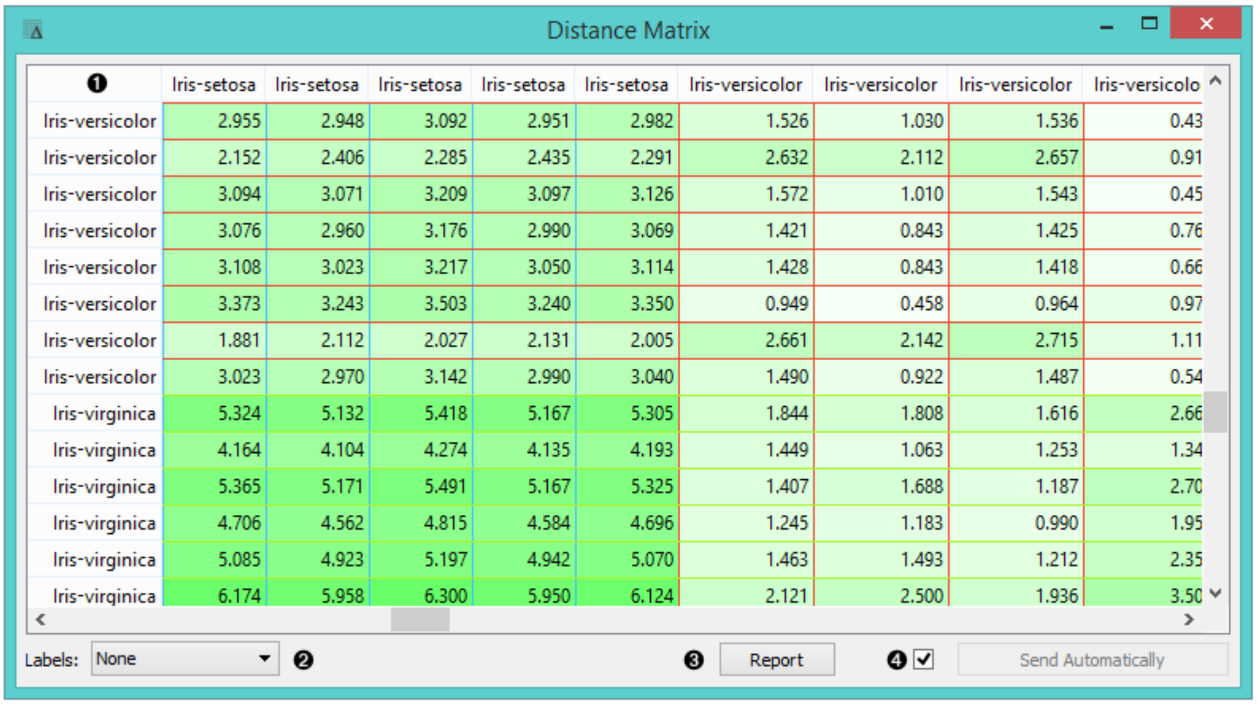
  A distance matrix.

- **Table**

  Distance measures in a distance matrix.

Description

The **Distance Matrix** widget creates a distance matrix, which is a two-dimensional array containing the distances, taken pairwise, between the elements of a set. The number of elements in the data set defines the size of the matrix. Data matrices are essential for hierarchical clustering and they are extremely useful in bioinformatics as well, where they are used to represent protein structures in a coordinate-independent manner.



❶	Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa	Iris-versicolor	Iris-versicolor	Iris-versicolor	Iris-versicolo
Iris-versicolor	2.955	2.948	3.092	2.951	2.982	1.526	1.030	1.536	0.43
Iris-versicolor	2.152	2.406	2.285	2.435	2.291	2.632	2.112	2.657	0.91
Iris-versicolor	3.094	3.071	3.209	3.097	3.126	1.572	1.010	1.543	0.45
Iris-versicolor	3.076	2.960	3.176	2.990	3.069	1.421	0.843	1.425	0.76
Iris-versicolor	3.108	3.023	3.217	3.050	3.114	1.428	0.843	1.418	0.66
Iris-versicolor	3.373	3.243	3.503	3.240	3.350	0.949	0.458	0.964	0.97
Iris-versicolor	1.881	2.112	2.027	2.131	2.005	2.661	2.142	2.715	1.11
Iris-versicolor	3.023	2.970	3.142	2.990	3.040	1.490	0.922	1.487	0.54
Iris-virginica	5.324	5.132	5.418	5.167	5.305	1.844	1.808	1.616	2.66
Iris-virginica	4.164	4.104	4.274	4.135	4.193	1.449	1.063	1.253	1.34
Iris-virginica	5.365	5.171	5.491	5.167	5.325	1.407	1.688	1.187	2.70
Iris-virginica	4.706	4.562	4.815	4.584	4.696	1.245	1.183	0.990	1.95
Iris-virginica	5.085	4.923	5.197	4.942	5.070	1.463	1.493	1.212	2.35
Iris-virginica	6.174	5.958	6.300	5.950	6.124	2.121	2.500	1.936	3.50

❷ ❸ ❹ ✓

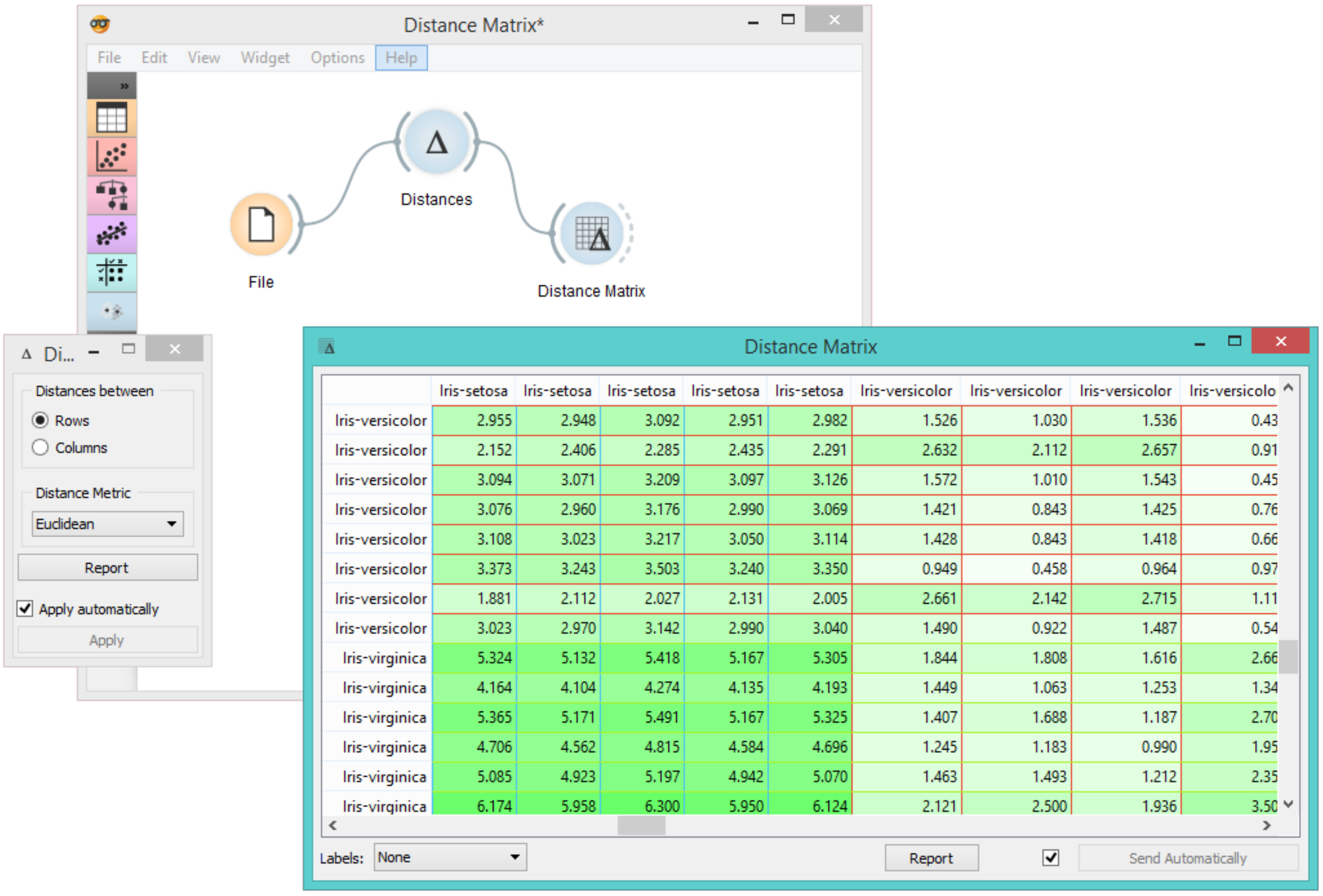
1. Elements in the data set and the distances between them
2. Label the table. The options are: *none*, *enumeration*, *according to variables*.
3. Produce a report.
4. Click *Send* to communicate changes to other widgets. Alternatively, tick the box in front of the *Send* button and

changes will be communicated automatically (*Send Automatically*).

The only two suitable inputs for **Distance Matrix** are the Distances widget and the Distance Transformation widget. The output of the widget is a data table containing the distance matrix. The user can decide how to label the table and the distance matrix (or instances in the distance matrix) can then be visualized or displayed in a separate data table.

Example

The example below displays a very standard use of the **Distance Matrix** widget. We compute the distances between rows in the sample from the *Iris* data set and output them in the **Distance Matrix**. It comes as no surprise that Iris Virginica and Iris Setosa are the furthest apart.



	Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa	Iris-versicolor	Iris-versicolor	Iris-versicolor	Iris-versicolo
Iris-versicolor	2.955	2.948	3.092	2.951	2.982	1.526	1.030	1.536	0.43
Iris-versicolor	2.152	2.406	2.285	2.435	2.291	2.632	2.112	2.657	0.91
Iris-versicolor	3.094	3.071	3.209	3.097	3.126	1.572	1.010	1.543	0.45
Iris-versicolor	3.076	2.960	3.176	2.990	3.069	1.421	0.843	1.425	0.76
Iris-versicolor	3.108	3.023	3.217	3.050	3.114	1.428	0.843	1.418	0.66
Iris-versicolor	3.373	3.243	3.503	3.240	3.350	0.949	0.458	0.964	0.97
Iris-versicolor	1.881	2.112	2.027	2.131	2.005	2.661	2.142	2.715	1.11
Iris-versicolor	3.023	2.970	3.142	2.990	3.040	1.490	0.922	1.487	0.54
Iris-virginica	5.324	5.132	5.418	5.167	5.305	1.844	1.808	1.616	2.66
Iris-virginica	4.164	4.104	4.274	4.135	4.193	1.449	1.063	1.253	1.34
Iris-virginica	5.365	5.171	5.491	5.167	5.325	1.407	1.688	1.187	2.70
Iris-virginica	4.706	4.562	4.815	4.584	4.696	1.245	1.183	0.990	1.95
Iris-virginica	5.085	4.923	5.197	4.942	5.070	1.463	1.493	1.212	2.35
Iris-virginica	6.174	5.958	6.300	5.950	6.124	2.121	2.500	1.936	3.50

Distance Transformation

Transforms distances in a data set.

Signals

Inputs:

- **Distances**

  A distance matrix

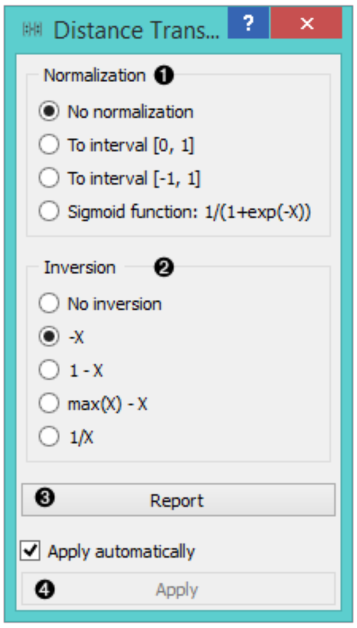
Outputs:

- **Distances**

  A distance matrix

Description

The **Distances Transformation** widget is used for the normalization and inversion of distance matrices. The normalization of data is necessary to bring all the variables into proportion with one another.


Distance Trans...

Normalization ❶
[✓ No normalization]
To interval [0, 1]
To interval [-1, 1]
Sigmoid function: 1/(1+exp(-X))

Inversion ❷
No inversion
[✓ -X]
1 - X
max(X) - X
1/X

❸ Report

[✓ Apply automatically]

❹ Apply

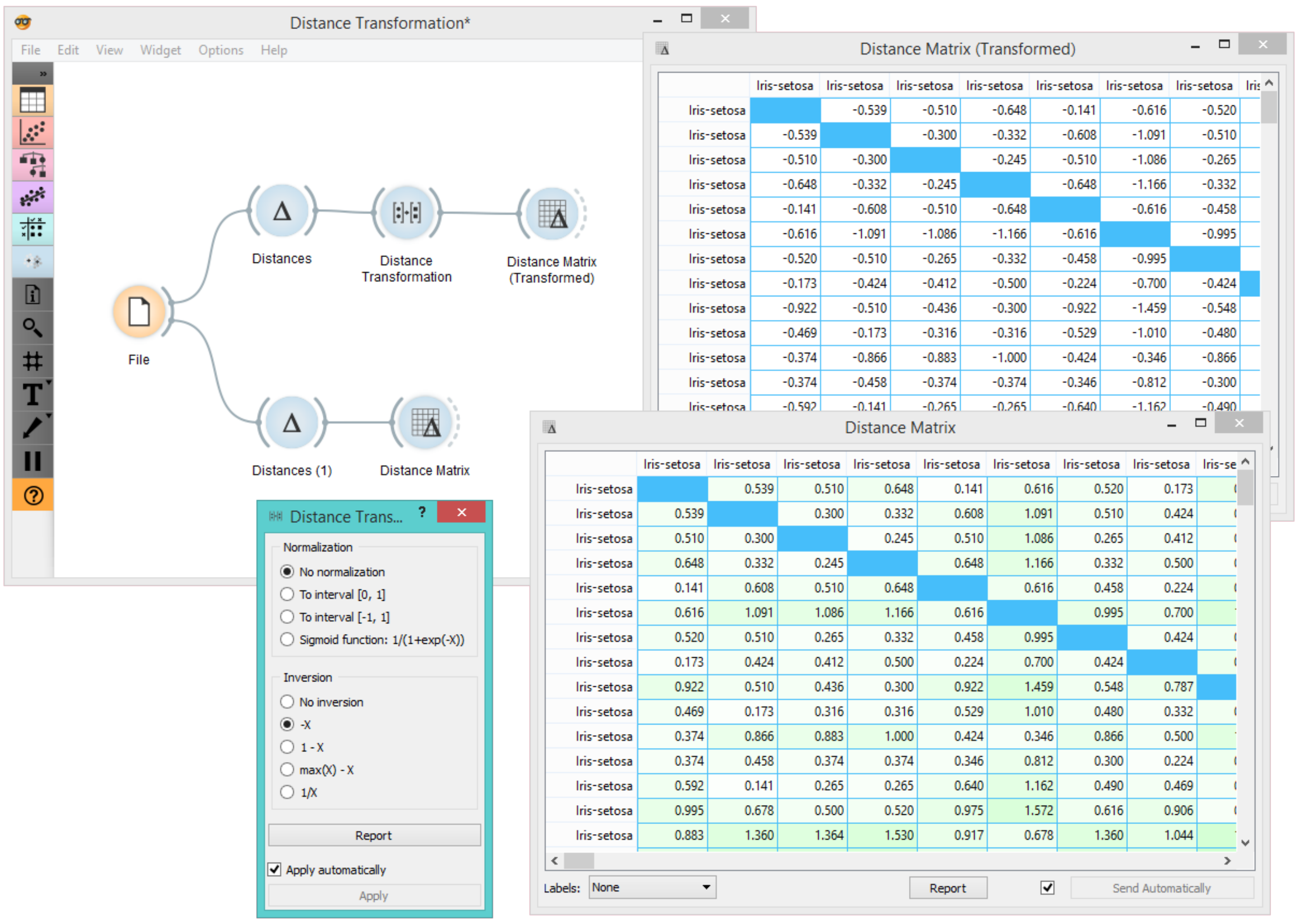
1. Choose the type of Normalization:
   - **No normalization**
   - **To interval [0, 1]**
   - **To interval [-1, 1]**
   - Sigmoid function: 1/(1+exp(-X))
2. Choose the type of Inversion:
   - **No inversion**
   - **-X**
   - **1 - X**
   - **max(X) - X**

- **1/X**

3. Produce a report.
4. After changing the settings, you need to click *Apply* to commit changes to other widgets. Alternatively, tick *Apply automatically*.

Example

In the snapshot below, you can see how transformation affects the distance matrix. We loaded the *Iris* data set and calculated the distances between rows with the help of the Distances widget. In order to demonstrate how **Distance Transformation** affects the Distance Matrix, we created the worflow below and compared the transformed distance matrix with the "original" one.



Normalization
(● No normalization) ○ To interval [0, 1] ○ To interval [-1, 1] ○ Sigmoid function: 1/(1+exp(-X))

Inversion
○ No inversion (● -X) ○ 1 - X ○ max(X) - X ○ 1/X

[✓] Apply automatically

Distance Matrix (Transformed)

	Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa
Iris-setosa		-0.539	-0.510	-0.648	-0.141	-0.616	-0.520
Iris-setosa	-0.539		-0.300	-0.332	-0.608	-1.091	-0.510
Iris-setosa	-0.510	-0.300		-0.245	-0.510	-1.086	-0.265
Iris-setosa	-0.648	-0.332	-0.245		-0.648	-1.166	-0.332
Iris-setosa	-0.141	-0.608	-0.510	-0.648		-0.616	-0.458
Iris-setosa	-0.616	-1.091	-1.086	-1.166	-0.616		-0.995
Iris-setosa	-0.520	-0.510	-0.265	-0.332	-0.458	-0.995	
Iris-setosa	-0.173	-0.424	-0.412	-0.500	-0.224	-0.700	-0.424
Iris-setosa	-0.922	-0.510	-0.436	-0.300	-0.922	-1.459	-0.548
Iris-setosa	-0.469	-0.173	-0.316	-0.316	-0.529	-1.010	-0.480
Iris-setosa	-0.374	-0.866	-0.883	-1.000	-0.424	-0.346	-0.866
Iris-setosa	-0.374	-0.458	-0.374	-0.374	-0.346	-0.812	-0.300
Iris-setosa	-0.592	-0.141	-0.265	-0.265	-0.640	-1.162	-0.490

Distance Matrix

	Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa
Iris-setosa		0.539	0.510	0.648	0.141	0.616	0.520	0.173
Iris-setosa	0.539		0.300	0.332	0.608	1.091	0.510	0.424
Iris-setosa	0.510	0.300		0.245	0.510	1.086	0.265	0.412
Iris-setosa	0.648	0.332	0.245		0.648	1.166	0.332	0.500
Iris-setosa	0.141	0.608	0.510	0.648		0.616	0.458	0.224
Iris-setosa	0.616	1.091	1.086	1.166	0.616		0.995	0.700
Iris-setosa	0.520	0.510	0.265	0.332	0.458	0.995		0.424
Iris-setosa	0.173	0.424	0.412	0.500	0.224	0.700	0.424	
Iris-setosa	0.922	0.510	0.436	0.300	0.922	1.459	0.548	0.787
Iris-setosa	0.469	0.173	0.316	0.316	0.529	1.010	0.480	0.332
Iris-setosa	0.374	0.866	0.883	1.000	0.424	0.346	0.866	0.500
Iris-setosa	0.374	0.458	0.374	0.374	0.346	0.812	0.300	0.224
Iris-setosa	0.592	0.141	0.265	0.265	0.640	1.162	0.490	0.469
Iris-setosa	0.995	0.678	0.500	0.520	0.975	1.572	0.616	0.906
Iris-setosa	0.883	1.360	1.364	1.530	0.917	0.678	1.360	1.044

Distances

Computes distances between rows/columns in a data set.

Signals

Inputs:

- **Data**

  A data set

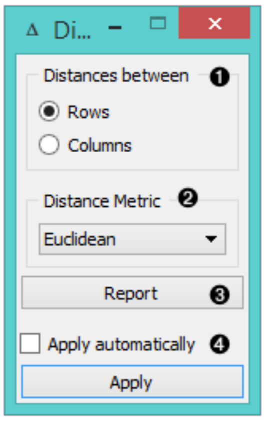
Outputs:

- **Distances**

  A distance matrix

Description

The **Distances** widget computes distances between rows or columns in a data set.



1. Choose whether to measure distances between rows or columns.
2. Choose the *Distance Metric*:
   - Euclidean ("straight line", distance between two points)
   - Manhattan (the sum of absolute differences for all attributes)
   - Cosine (the cosine of the angle between two vectors of an inner product space)
   - Jaccard (the size of the intersection divided by the size of the union of the sample sets)
   - Spearman (linear correlation between the rank of the values, remapped as a distance in a [0, 1] interval)
   - Spearman absolute (linear correlation between the rank of the absolute values, remapped as a distance in a [0, 1] interval)
   - Pearson (linear correlation between the values, remapped as a distance in a [0, 1] interval)
   - Pearson absolute (linear correlation between the absolute values, remapped as a distance in a [0, 1] interval)

   In case of missing values, the widget automatically imputes the average value of the row or the column.

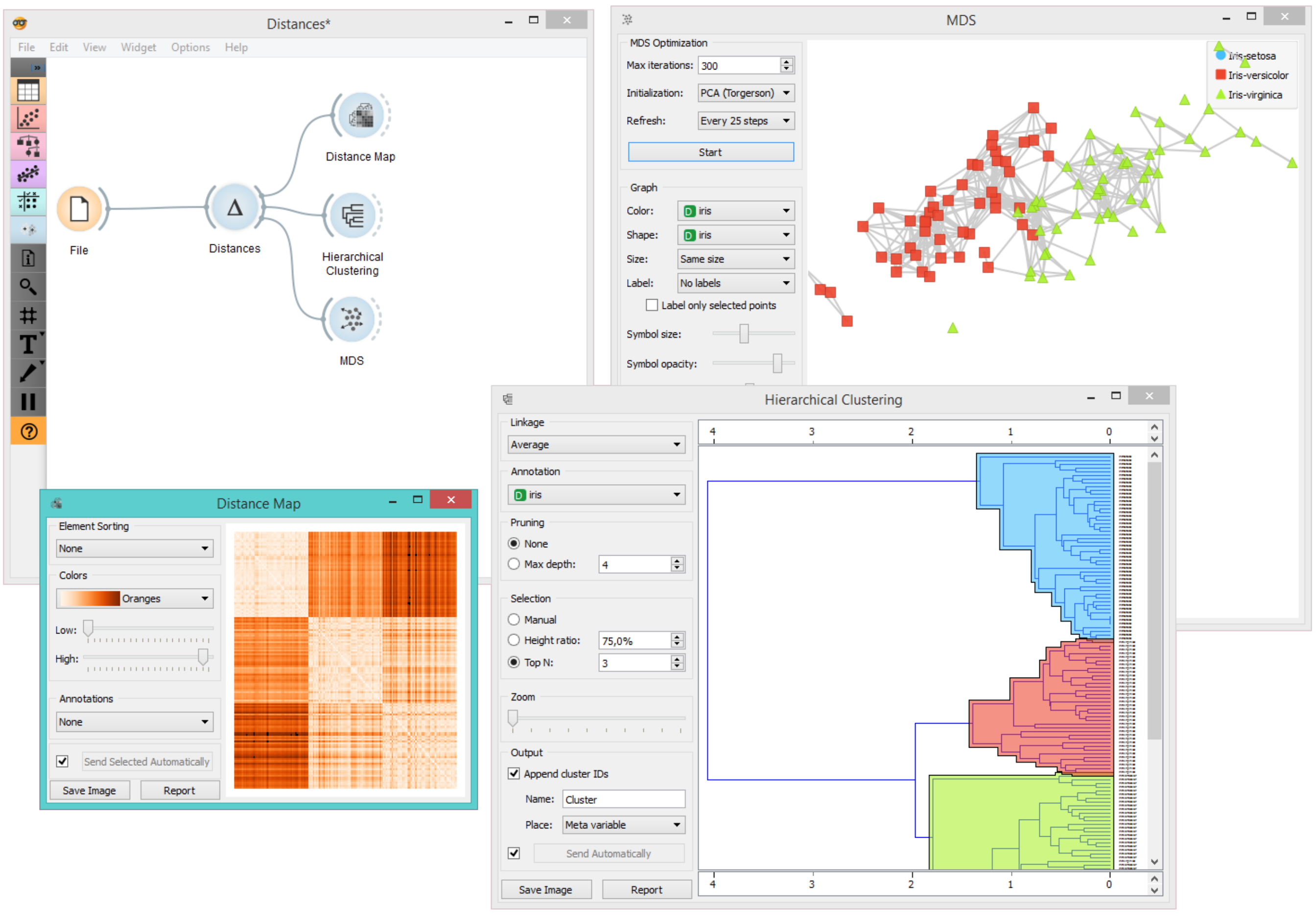
   Since the widget cannot compute distances between discrete and continuous attributes, it only uses continuous attributes and ignores the discrete ones. If you want to use discrete attributes, continuize them with the Continuize widget first.

3. Produce a report.

4. Tick *Apply Automatically* to automatically commit changes to other widgets. Alternatively, press '*Apply*'.

Example

This widget needs to be connected to another widget to display results, for instance to Distance Map to visualize distances, Hierarchical Clustering to cluster the attributes, or MDS to visualize the distances in a plane.



MDS

MDS Optimization
Max iterations: 300
Initialization: PCA (Torgerson)
Refresh: Every 25 steps
Color: iris
Shape: iris
Size: Same size
Label: No labels
☐ Label only selected points

Hierarchical Clustering

Linkage: Average
Annotation: iris
Pruning: [✓ None] Max depth: 4
Selection: Manual, Height ratio: 75,0%, [✓ Top N: 3]
Output: ☑ Append cluster IDs
Name: Cluster
Place: Meta variable
☑ Send Automatically

Distance Map

Element Sorting: None
Colors: Oranges
Low:
High:
Annotations: None
☑ Send Selected Automatically

Hierarchical Clustering

Groups items using a hierarchical clustering algorithm.

Signals

Inputs:

- **Distances**

  A distance matrix

Outputs:

- **Selected Data**

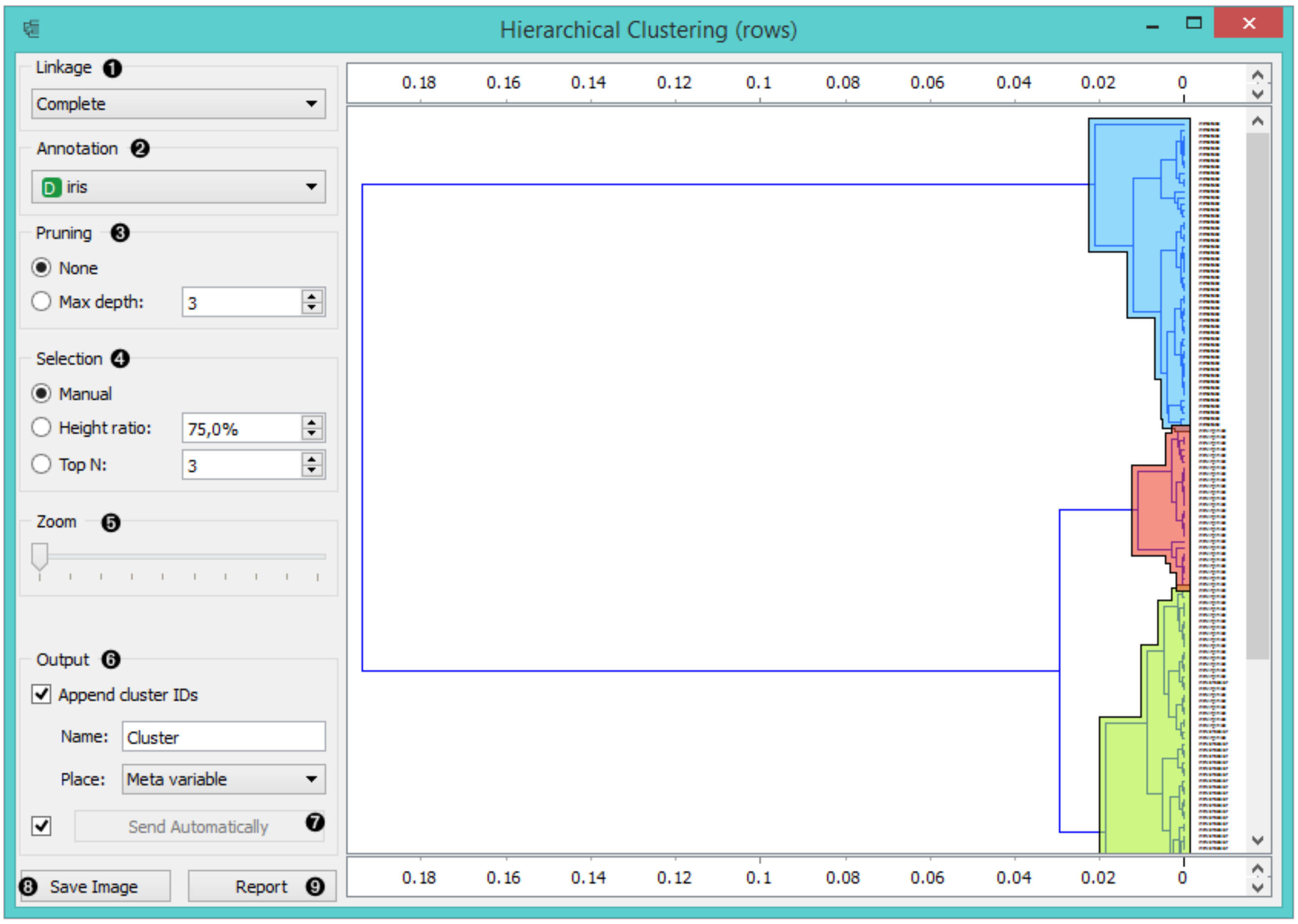
  A data subset

- **Other Data**

  Remaining data

Description

The widget computes hierarchical clustering of arbitrary types of objects from a matrix of distances and shows a corresponding dendrogram.

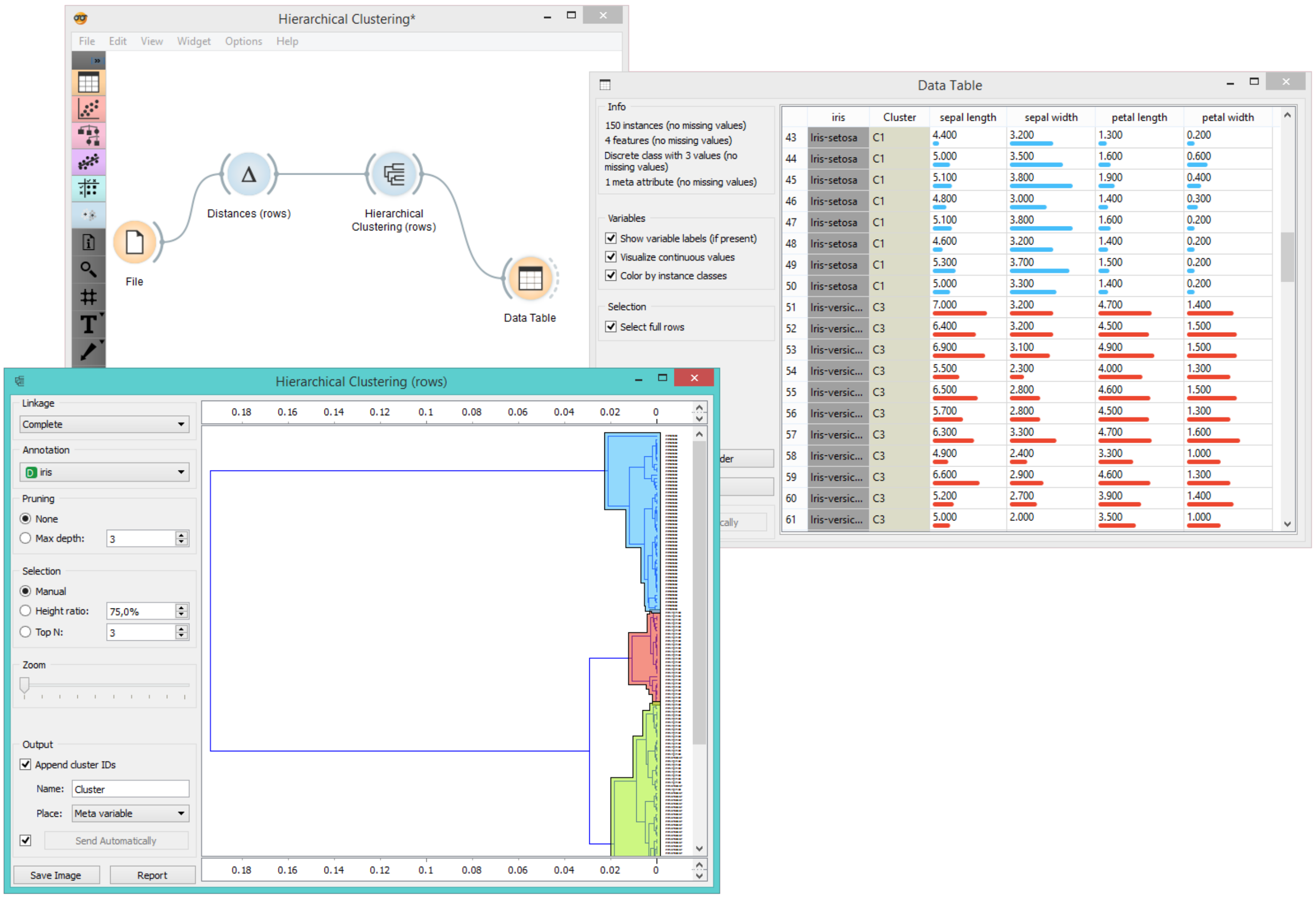


Linkage ❶: Complete
Annotation ❷: iris
Pruning ❸: [✓ None] Max depth: 3
Selection ❹: [✓ Manual] Height ratio: 75,0% Top N: 3
Zoom ❺
Output ❻: ✓ Append cluster IDs
Name: Cluster
Place: Meta variable
✓ Send Automatically ❼
❽ Save Image
Report ❾

1. The widget supports four ways of measuring distances between clusters:
   - **Single linkage** computes the distance between the closest elements of the two clusters
   - **Average linkage** computes the average distance between elements of the two clusters
   - **Weighted linkage** uses the WPGMA method
   - **Complete linkage** computes the distance between the clusters' most distant elements
2. Labels of nodes in the dendrogram can be chosen in the **Annotation** box.
3. Huge dendrograms can be pruned in the *Pruning* box by selecting the maximum depth of the dendrogram. This only affects the display, not the actual clustering.
4. The widget offers three different selection methods:
   - **Manual** (Clicking inside the dendrogram will select a cluster. Multiple clusters can be selected by holding Ctrl/Cmd. Each selected cluster is shown in a different color and is treated as a separate cluster in the output.)
   - **Height ratio** (Clicking on the bottom or top ruler of the dendrogram places a cutoff line in the graph. Items to the right of the line are selected.)
   - **Top N** (Selects the number of top nodes.)
5. Use *Zoom* and scroll to zoom in or out.
6. If the items being clustered are instances, they can be added a cluster index (*Append cluster IDs*). The ID can appear as an ordinary **Attribute**, **Class attribute** or a **Meta attribute**. In the second case, if the data already has a class attribute, the original class is placed among meta attributes.
7. The data can be automatically output on any change (*Auto send is on*) or, if the box isn't ticked, by pushing *Send Data*.
8. Clicking this button produces an image that can be saved.
9. Produce a report.

Examples

The workflow below shows the output of **Hierarchical Clustering** for the *Iris* data set in Data Table widget. We see that if we choose *Append cluster IDs* in hierarchical clustering, we can see an additional column in the Data Table named *Cluster*. This is a way to check how hierarchical clustering clustered individual instances.



Linkage: Complete

Annotation: iris

Pruning
- (●) None
- ( ) Max depth: 3

Selection
- (●) Manual
- ( ) Height ratio: 75,0%
- ( ) Top N: 3

Zoom

Output
- ☑ Append cluster IDs
- Name: Cluster
- Place: Meta variable
- ☑ Send Automatically

Data Table

Info

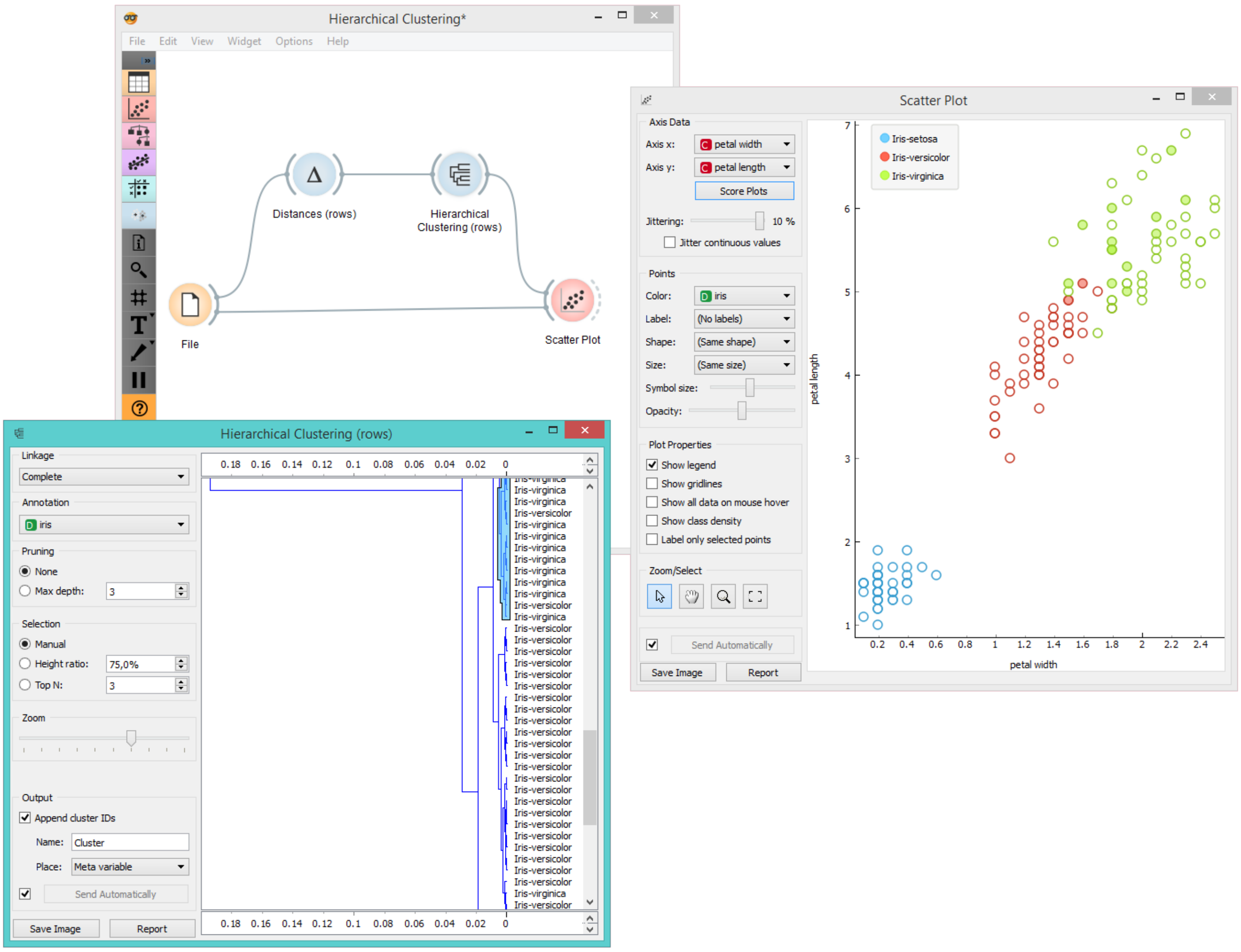
150 instances (no missing values)
4 features (no missing values)
Discrete class with 3 values (no missing values)
1 meta attribute (no missing values)

Variables
- ☑ Show variable labels (if present)
- ☑ Visualize continuous values
- ☑ Color by instance classes

Selection
- ☑ Select full rows

	iris	Cluster	sepal length	sepal width	petal length	petal width
43	Iris-setosa	C1	4.400	3.200	1.300	0.200
44	Iris-setosa	C1	5.000	3.500	1.600	0.600
45	Iris-setosa	C1	5.100	3.800	1.900	0.400
46	Iris-setosa	C1	4.800	3.000	1.400	0.300
47	Iris-setosa	C1	5.100	3.800	1.600	0.200
48	Iris-setosa	C1	4.600	3.200	1.400	0.200
49	Iris-setosa	C1	5.300	3.700	1.500	0.200
50	Iris-setosa	C1	5.000	3.300	1.400	0.200
51	Iris-versic...	C3	7.000	3.200	4.700	1.400
52	Iris-versic...	C3	6.400	3.200	4.500	1.500
53	Iris-versic...	C3	6.900	3.100	4.900	1.500
54	Iris-versic...	C3	5.500	2.300	4.000	1.300
55	Iris-versic...	C3	6.500	2.800	4.600	1.500
56	Iris-versic...	C3	5.700	2.800	4.500	1.300
57	Iris-versic...	C3	6.300	3.300	4.700	1.600
58	Iris-versic...	C3	4.900	2.400	3.300	1.000
59	Iris-versic...	C3	6.600	2.900	4.600	1.300
60	Iris-versic...	C3	5.200	2.700	3.900	1.400
61	Iris-versic...	C3	5.000	2.000	3.500	1.000

In the second example, we loaded the *Iris* data set again, but this time we added the Scatter Plot, showing all the instances from the File widget, while at the same time receiving the selected instances signal from **Hierarchical Clustering**. This way we can observe the position of the selected cluster(s) in the projection.



Hierarchical Clustering (rows)

Linkage: Complete
Annotation: iris
Pruning: [✓ None] Max depth: 3
Selection: [✓ Manual] Height ratio: 75,0% Top N: 3
Zoom
Output: [✓ Append cluster IDs]
Name: Cluster
Place: Meta variable
[✓] Send Automatically

Scatter Plot

Axis Data
Axis x: petal width
Axis y: petal length
Score Plots
Jittering: 10 %
Jitter continuous values

Points
Color: iris
Label: (No labels)
Shape: (Same shape)
Size: (Same size)
Symbol size:
Opacity:

Plot Properties
[✓] Show legend
Show gridlines
Show all data on mouse hover
Show class density
Label only selected points

Zoom/Select
[✓] Send Automatically

k-Means

Groups items using the k-Means clustering algorithm.

Signals

Inputs:

- **Data**

  A data set.

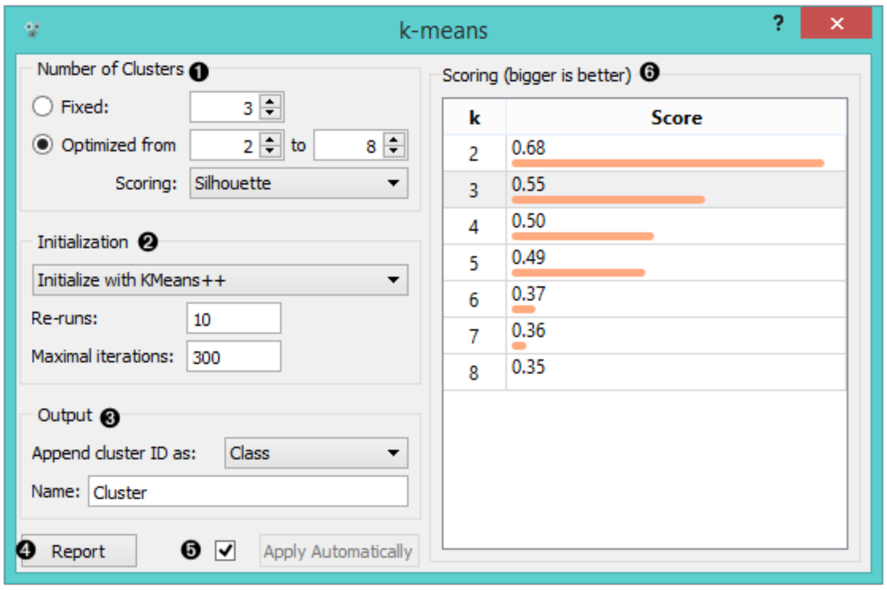
Outputs:

- **Data**

  A data set with cluster index as a class attribute.

Description

The widget applies the k-Means clustering algorithm to the data and outputs a new data set in which the cluster index is used as a class attribute. The original class attribute, if it exists, is moved to meta attributes. Scores of clustering results for various k are also shown in the widget.



Fixed: 3
[✓ Optimized from] 2 to 8
Scoring: Silhouette

Initialize with KMeans++
Re-runs: 10
Maximal iterations: 300

Append cluster ID as: Class
Name: Cluster

[✓ Apply Automatically]

k	Score
2	0.68
3	0.55
4	0.50
5	0.49
6	0.37
7	0.36
8	0.35

1. Select the number of clusters.
   - **Fixed**: algorithm clusters data in a specified number of clusters.
   - **Optimized**: widget shows clustering scores for the selected cluster range.
   - Silhouette (contrasts average distance to elements in the same cluster with the average distance to elements in other clusters)
   - **Inter-cluster distance** (measures distances between clusters, normally between centroids)
   - **Distance to** centroids (measures distances to the arithmetic means of clusters)
2. Select the initialization method (the way the algorithm begins clustering):
   - k-Means++ (first center is selected randomly, subsequent are chosen from the remaining points with probability proportioned to squared distance from the closest center)

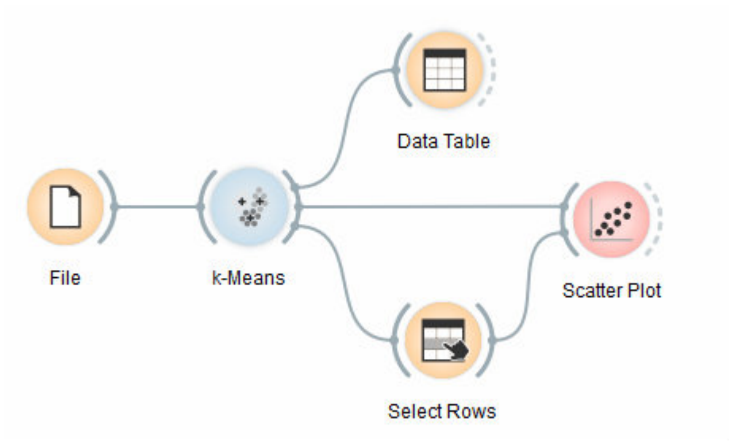
- **Random initialization** (clusters are assigned randomly at first and then updated with further iterations)

  Re-runs (how many times the algorithm is run) and **maximal iterations** (the maximum number of iteration within each algorithm run) can be set manually.

3. The widget outputs a new data set with appended cluster information. Select how to append cluster information (as class, feature or meta attribute) and name the column.
4. If *Apply Automatically* is ticked, the widget will commit changes automatically. Alternatively, click *Apply*.
5. Produce a report.
6. Check scores of clustering results for various k.

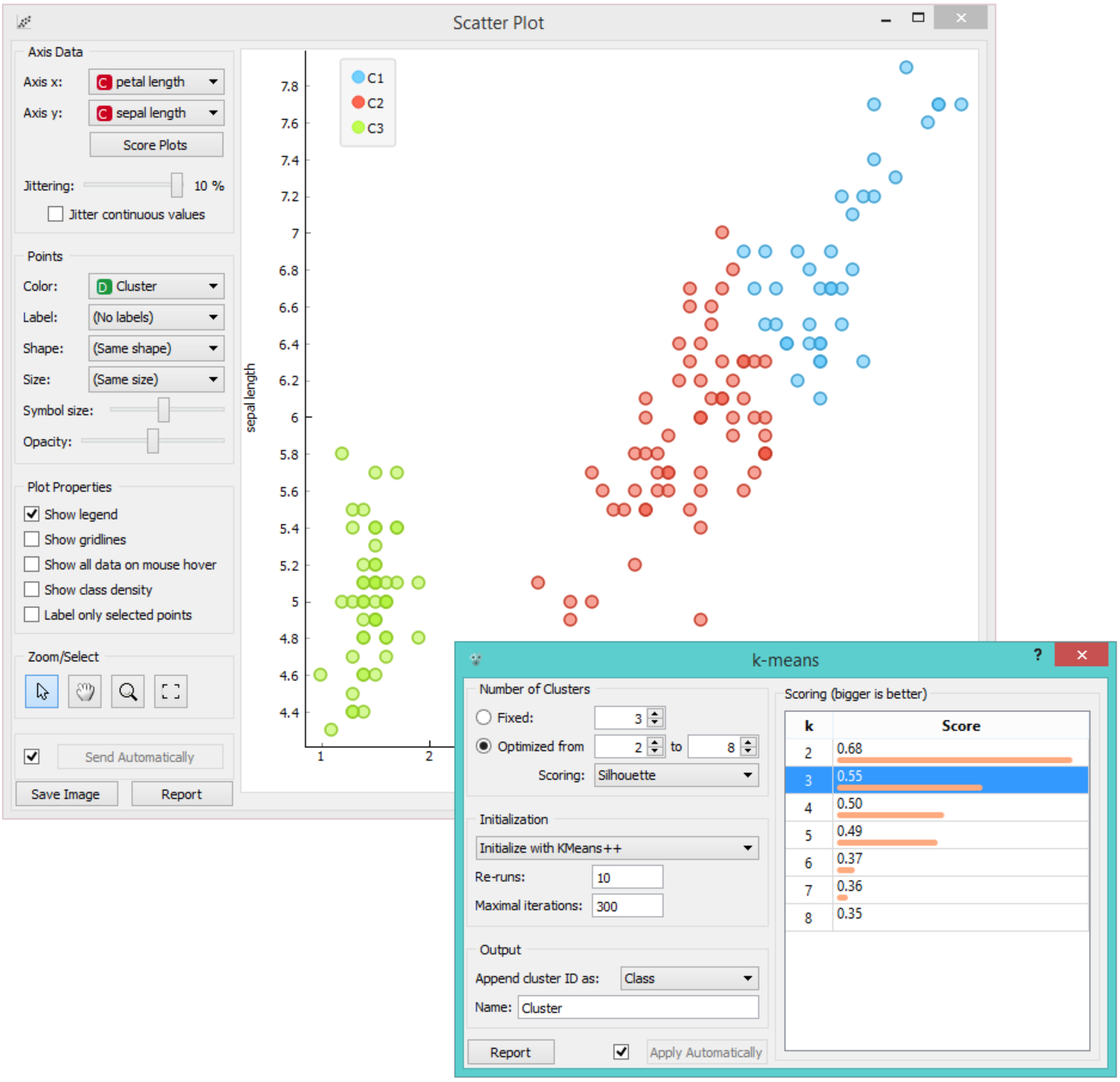
Examples

We are going to explore the widget with the following schema.



First, we load the *Iris* data set, divide it into three clusters and show it in the Data Table, where we can observe which instance went into which cluster. The interesting parts are the Scatter Plot and Select Rows.

Since **k-Means** added the cluster index as a class attribute, the scatter plot will color the points according to the clusters they are in.



Axis Data
Axis x: petal length
Axis y: sepal length
Score Plots
Jittering: 10 %
☐ Jitter continuous values

Points
Color: Cluster
Label: (No labels)
Shape: (Same shape)
Size: (Same size)
Symbol size:
Opacity:

Plot Properties
☑ Show legend
☐ Show gridlines
☐ Show all data on mouse hover
☐ Show class density
☐ Label only selected points

Zoom/Select

☑ Send Automatically
Save Image Report

Number of Clusters
○ Fixed: 3
● Optimized from 2 to 8
Scoring: Silhouette

Initialization
Initialize with KMeans++
Re-runs: 10
Maximal iterations: 300

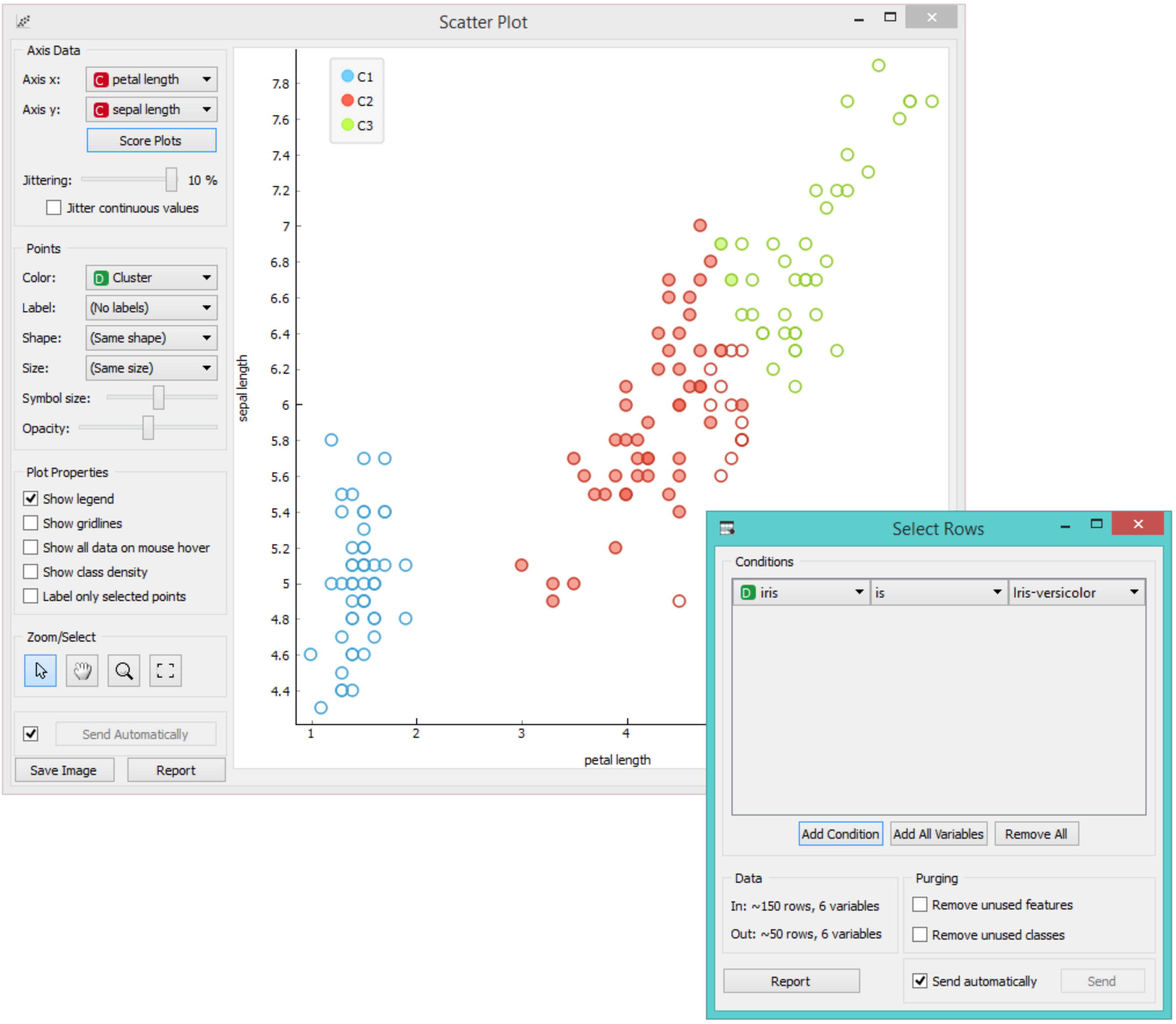
Output
Append cluster ID as: Class
Name: Cluster

Report ☑ Apply Automatically

Scoring (bigger is better)

k	Score
2	0.68
3	0.55
4	0.50
5	0.49
6	0.37
7	0.36
8	0.35

What we are really interested in is how well the clusters induced by the (unsupervised) clustering algorithm match the actual classes in the data. We thus take Select Rows widget, in which we can select individual classes and have the corresponding points marked in the scatter plot. The match is perfect for *setosa*, and pretty good for the other two classes.

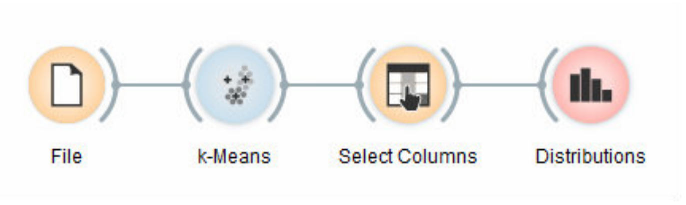


Axis x: petal length
Axis y: sepal length
Jittering: 10 %
☐ Jitter continuous values
Color: Cluster
Label: (No labels)
Shape: (Same shape)
Size: (Same size)
✓ Show legend
☐ Show gridlines
☐ Show all data on mouse hover
☐ Show class density
☐ Label only selected points
✓ Send Automatically

iris is Iris-versicolor
In: ~150 rows, 6 variables
Out: ~50 rows, 6 variables
☐ Remove unused features
☐ Remove unused classes
✓ Send automatically

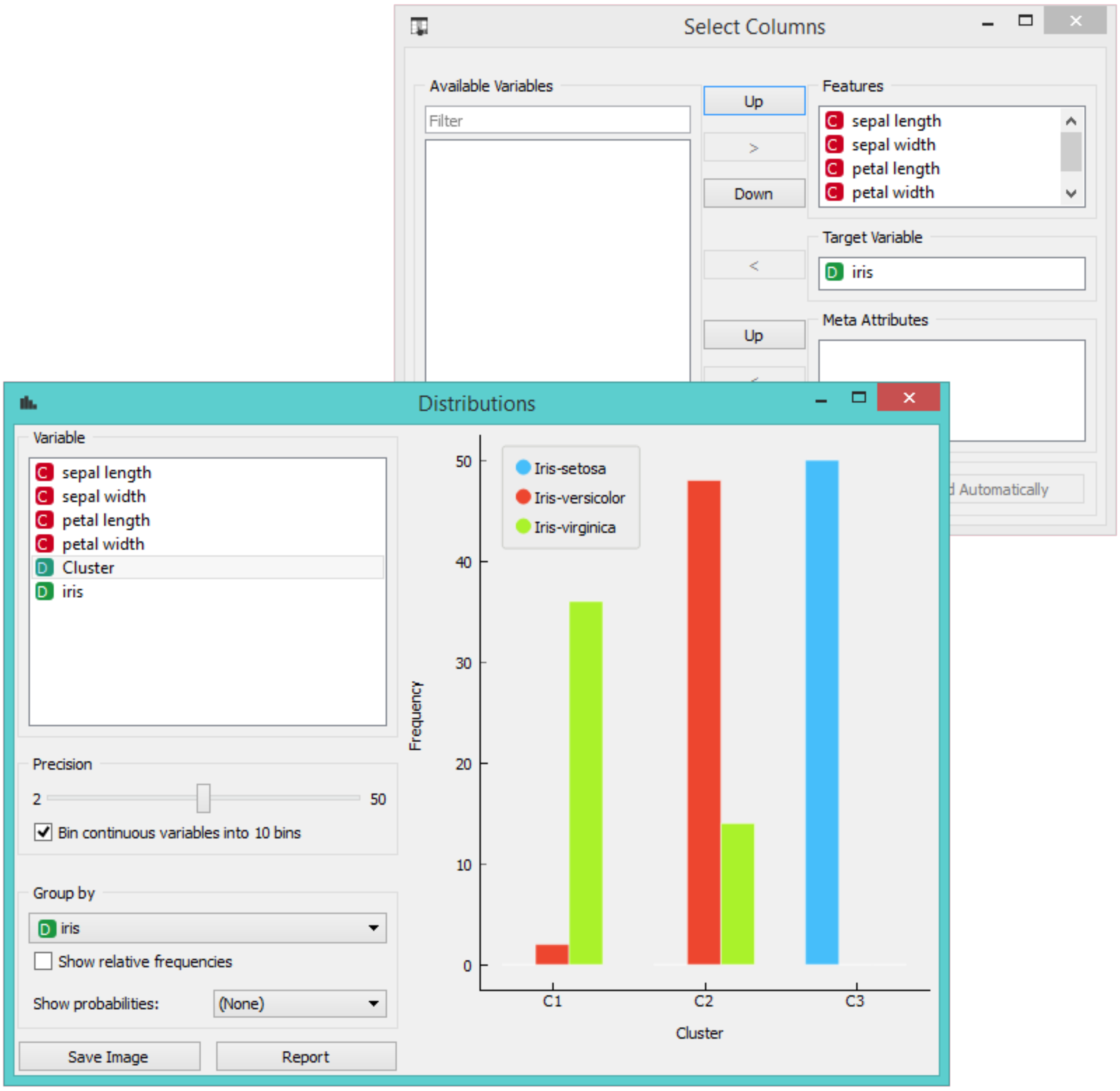
You may have noticed that we left the **Remove unused values/attributes** and **Remove unused classes** in Select Rows unchecked. This is important: if the widget modifies the attributes, it outputs a list of modified instances and the scatter plot cannot compare them to the original data.

Perhaps a simpler way to test the match between clusters and the original classes is to use the Distributions widget.



The only (minor) problem here is that this widget only visualizes normal (and not meta) attributes. We solve this by using Select Columns: we reinstate the original class *Iris* as the class and put the cluster index among the attributes.

The match is perfect for *setosa*: all instances of setosa are in the third cluster (blue). 48 *versicolors* are in the second cluster (red), while two ended up in the first. For *virginicae*, 36 are in the first cluster and 14 in the second.



Features: sepal length, sepal width, petal length, petal width

Target Variable: iris

Variable: sepal length, sepal width, petal length, petal width, Cluster, iris

Precision: 2 — 50

☑ Bin continuous variables into 10 bins

Group by: iris

☐ Show relative frequencies

Show probabilities: (None)

Manifold Learning

Nonlinear dimensionality reduction.

Signals

Inputs:

- **Data**

  A data set

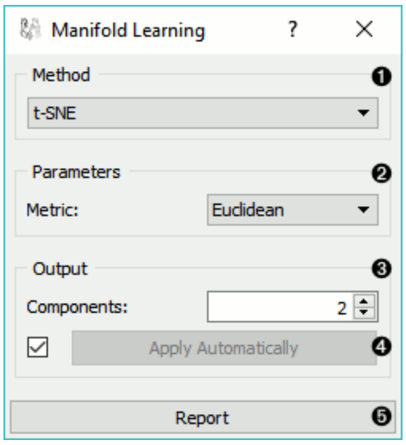
Outputs:

- **Transformed Data**

  A data set with new, reduced coordinates.

Description

Manifold Learning is a technique which finds a non-linear manifold within the higher-dimensional space. The widget then outputs new coordinates which correspond to a two-dimensional space. Such data can be later visualized with Scatter Plot or other visualization widgets.



Manifold Learning

Method
t-SNE

Parameters
Metric: Euclidean

Output
Components: 2
☑ Apply Automatically

Report

1. Method for manifold learning:
   - t-SNE
   - MDS, see also MDS widget
   - Isomap
   - Locally Linear Embedding
   - Spectral Embedding
2. Set parameters for the method:
   - t-SNE (distance measures):
     - *Euclidean* distance
     - *Manhattan*
     - *Chebyshev*
     - *Jaccard*
     - *Mahalanobis*
     - *Cosine*

- MDS (iterations and initialization):
    - *max interations*: maximum number of optimization interations
    - *initialization*: method for initialization of the algorithm (PCA or random)
- Isomap:
    - number of *neighbors*
- Locally Linear Embedding:
    - *method*:
        - standard
        - modified
        - hessian eigenmap
        - local
    - number of *neighbors*
    - *max iterations*
- Spectral Embedding:
    - *affinity*:
        - nearest neighbors
        - RFB kernel

3. Output: the number of reduced features (components).
4. If *Apply automatically* is ticked, changes will be propagated automatically. Alternatively, click *Apply*.
5. Produce a report.

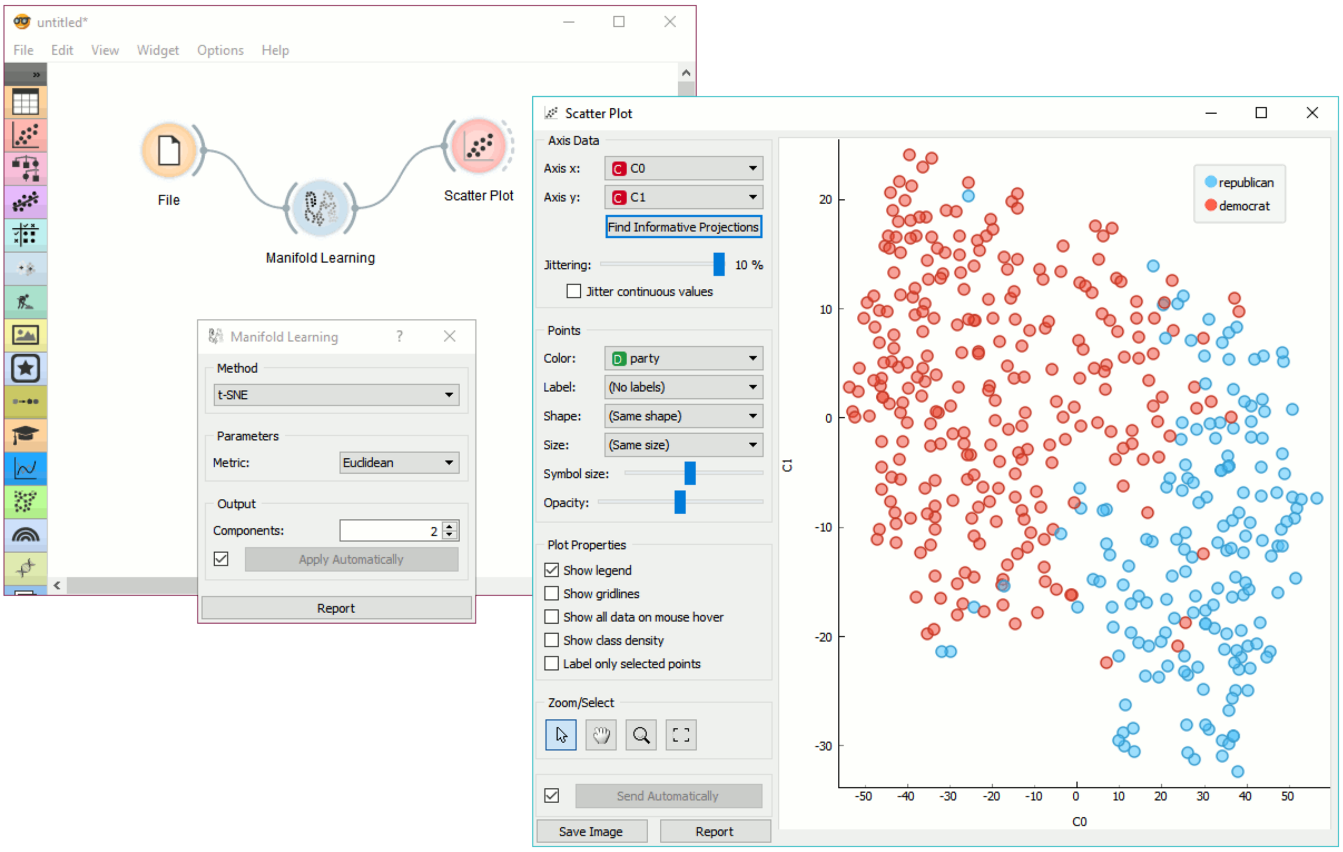
Manifold Learning widget produces different embeddings for high-dimensional data.

... figure:: images/collage-manifold.png

From left to right, top to bottom: t-SNE, MDS, Isomap, Locally Linear Embedding and Spectral Embedding.

Example

Manifold Learning widget transforms high-dimensional data into a lower dimensional approximation. This makes it great for visualizing data sets with many features. We used *voting.tab* to map 16-dimensional data onto a 2D graph. Then we used Scatter Plot to plot the embeddings.



Manifold Learning

Method: t-SNE

Parameters
Metric: Euclidean

Output
Components: 2
☑ Apply Automatically

Report

Axis Data
Axis x: C0
Axis y: C1
Find Informative Projections
Jittering: 10 %
☐ Jitter continuous values

Points
Color: party
Label: (No labels)
Shape: (Same shape)
Size: (Same size)
Symbol size:
Opacity:

Plot Properties
☑ Show legend
☐ Show gridlines
☐ Show all data on mouse hover
☐ Show class density
☐ Label only selected points

Zoom/Select

☑ Send Automatically

Save Image
Report

MDS

Multidimensional scaling (MDS) projects items onto a plane fitted to given distances between points.

Signals

Inputs:

- **Distances**

  A distance matrix

- **Data**

  A data set

Outputs:

- **Data**

  A data set with MDS coordinates.

- **Data subset**

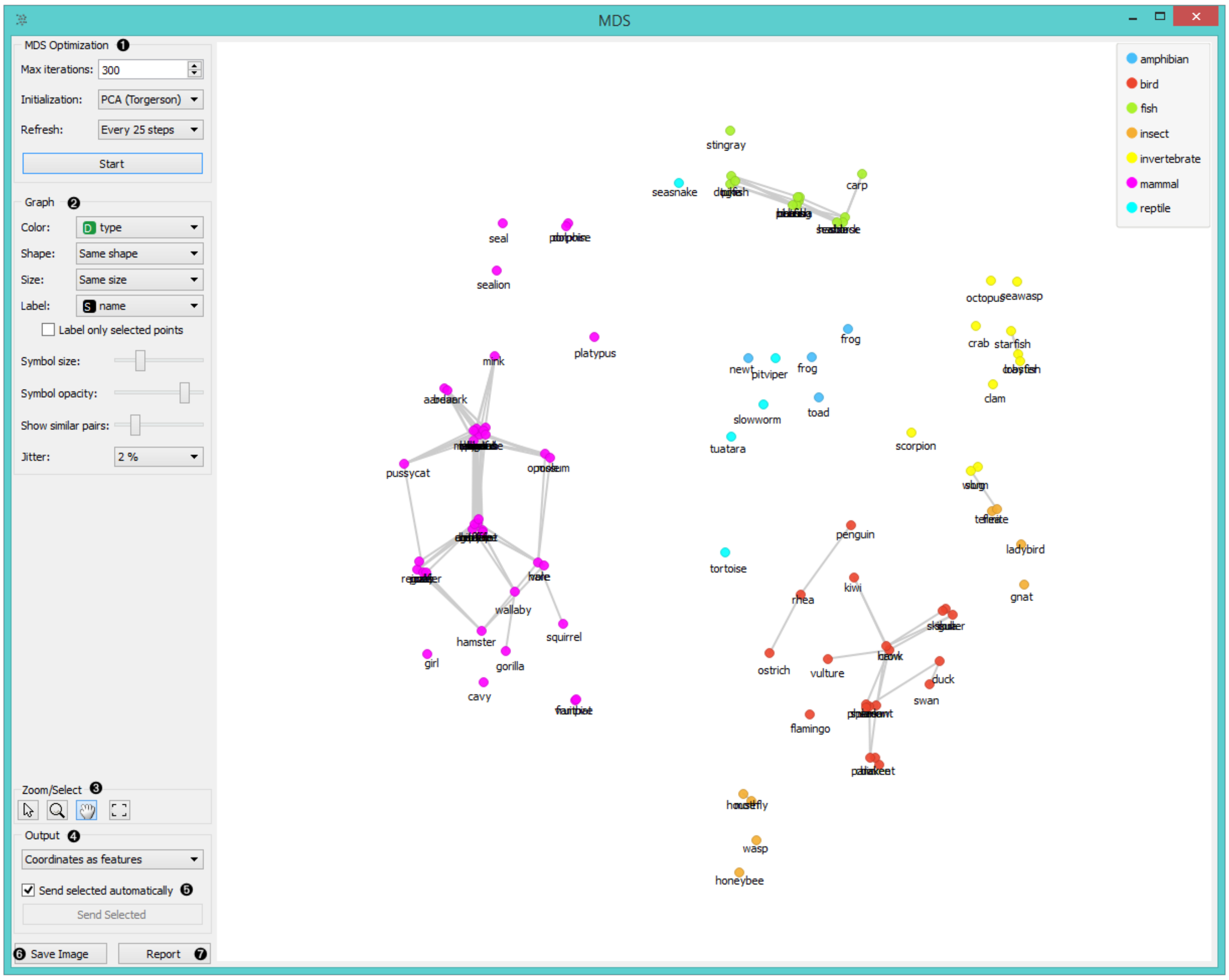
  Selected data

Description

Multidimensional scaling is a technique which finds a low-dimensional (in our case a two-dimensional) projection of points, where it tries to fit distances between points as well as possible. The perfect fit is typically impossible to obtain since the data is high-dimensional or the distances are not Euclidean.

In the input, the widget needs either a data set or a matrix of distances. When visualizing distances between rows, you can also adjust the color of the points, change their shape, mark them, and output them upon selection.

The algorithm iteratively moves the points around in a kind of a simulation of a physical model: if two points are too close to each other (or too far away), there is a force pushing them apart (or together). The change of the point's position at each time interval corresponds to the sum of forces acting on it.



MDS

MDS Optimization ❶

Max iterations: 300

Initialization: PCA (Torgerson)

Refresh: Every 25 steps

Start

Graph ❷

Color: type

Shape: Same shape

Size: Same size

Label: name

☐ Label only selected points

Symbol size:

Symbol opacity:

Show similar pairs:

Jitter: 2 %

Zoom/Select ❸

Output ❹

Coordinates as features

☑ Send selected automatically ❺

Send Selected

❻ Save Image

Report ❼

1. The widget redraws the projection during optimization. Optimization is run automatically in the beginning and later by pushing *Start*.

   - **Max iterations**: The optimization stops either when the projection changes only minimally at the last iteration or when a maximum number of iterations has been reached.
   - **Initialization**: PCA (Torgerson) positions the initial points along principal coordinate axes. *Random* sets the initial points to a random position and then readjusts them.
   - **Refresh**: Set how often you want to refresh the visualization. It can be at *Every iteration*, *Every 5/10/25/50 steps* or never (*None*). Setting a lower refresh interval makes the animation more visually appealing, but can be slow if the number of points is high.

2. Defines how the points are visualized. These options are available only when visalizing distances between rows (selected in the Distances widget).

   - **Color**: Color of points by attribute (gray for continuous, colored for discrete).
   - **Shape**: Shape of points by attribute (only for discrete).
   - **Size**: Set the size of points (*Same size* or select an attribute) or let the size depend on the value of the continuous attribute the point represents (Stress).
   - **Label**: Discrete attributes can serve as a label.
   - **Symbol size**: Adjust the size of the dots.
   - **Symbol opacity**: Adjust the transparency level of the dots.
   - **Show similar pairs**: Adjust the strength of network lines.
   - **Jitter**: Set jittering to prevent the dots from overlapping.

3. Adjust the graph with *Zoom/Select*. The arrow enables you to select data instances. The magnifying glass enables zooming, which can be also done by scrolling in and out. The hand allows you to move the graph around. The rectangle readjusts the graph proportionally.
4. Select the desired output:

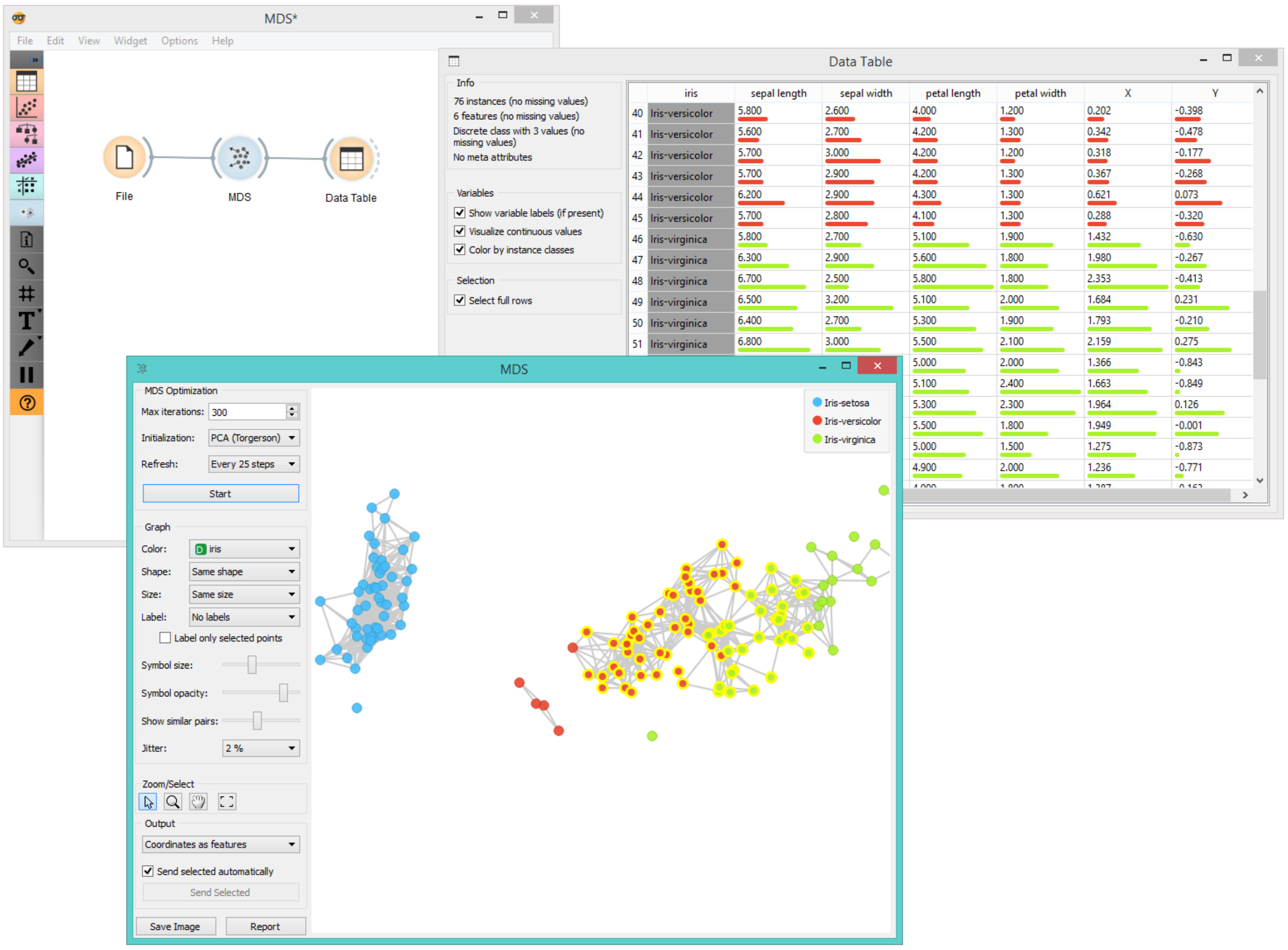
- **Original features only** (input data set)
- **Coordinates only** (MDS coordinates)
- **Coordinates as features** (input data set + MDS coordinates as regular attributes)
- **Coordinates as meta attributes** (input data set + MDS coordinates as meta attributes)

5. Sending the instances can be automatic if *Send selected automatically* is ticked. Alternatively, click *Send selected*.
6. **Save Image** allows you to save the created image either as .svg or .png file to your device.
7. Produce a report.

The MDS graph performs many of the functions of the Visualizations widget. It is in many respects similar to the Scatter Plot widget, so we recommend reading that widget's description as well.

Example

The above graphs were drawn using the following simple schema. We used the *iris.tab* data set. Using the Distances widget we input the distance matrix into the **MDS** widget, where we see the *Iris* data displayed in a 2-dimensional plane. We can see the appended coordinates in the Data Table widget.



Data Table

Info
76 instances (no missing values)
6 features (no missing values)
Discrete class with 3 values (no missing values)
No meta attributes

Variables
☑ Show variable labels (if present)
☑ Visualize continuous values
☑ Color by instance classes

Selection
☑ Select full rows

	iris	sepal length	sepal width	petal length	petal width	X	Y
40	Iris-versicolor	5.800	2.600	4.000	1.200	0.202	-0.398
41	Iris-versicolor	5.600	2.700	4.200	1.300	0.342	-0.478
42	Iris-versicolor	5.700	3.000	4.200	1.200	0.318	-0.177
43	Iris-versicolor	5.700	2.900	4.200	1.300	0.367	-0.268
44	Iris-versicolor	6.200	2.900	4.300	1.300	0.621	0.073
45	Iris-versicolor	5.700	2.800	4.100	1.300	0.288	-0.320
46	Iris-virginica	5.800	2.700	5.100	1.900	1.432	-0.630
47	Iris-virginica	6.300	2.900	5.600	1.800	1.980	-0.267
48	Iris-virginica	6.700	2.500	5.800	1.800	2.353	-0.413
49	Iris-virginica	6.500	3.200	5.100	2.000	1.684	0.231
50	Iris-virginica	6.400	2.700	5.300	1.900	1.793	-0.210
51	Iris-virginica	6.800	3.000	5.500	2.100	2.159	0.275
				5.000	2.000	1.366	-0.843
				5.100	2.400	1.663	-0.849
				5.300	2.300	1.964	0.126
				5.500	1.800	1.949	-0.001
				5.000	1.500	1.275	-0.873
				4.900	2.000	1.236	-0.771

MDS

MDS Optimization
Max iterations: 300
Initialization: PCA (Torgerson)
Refresh: Every 25 steps
Start

Graph
Color: iris
Shape: Same shape
Size: Same size
Label: No labels
☐ Label only selected points
Symbol size:
Symbol opacity:
Show similar pairs:
Jitter: 2 %

Zoom/Select

Output
Coordinates as features
☑ Send selected automatically
Send Selected

Save Image
Report

References

Wickelmaier, F. (2003). An Introduction to MDS. Sound Quality Research Unit, Aalborg University. Available here.

PCA

PCA linear transformation of input data.

Signals

Inputs:

- **Data**

  A data set.

Outputs:

- **Transformed Data**

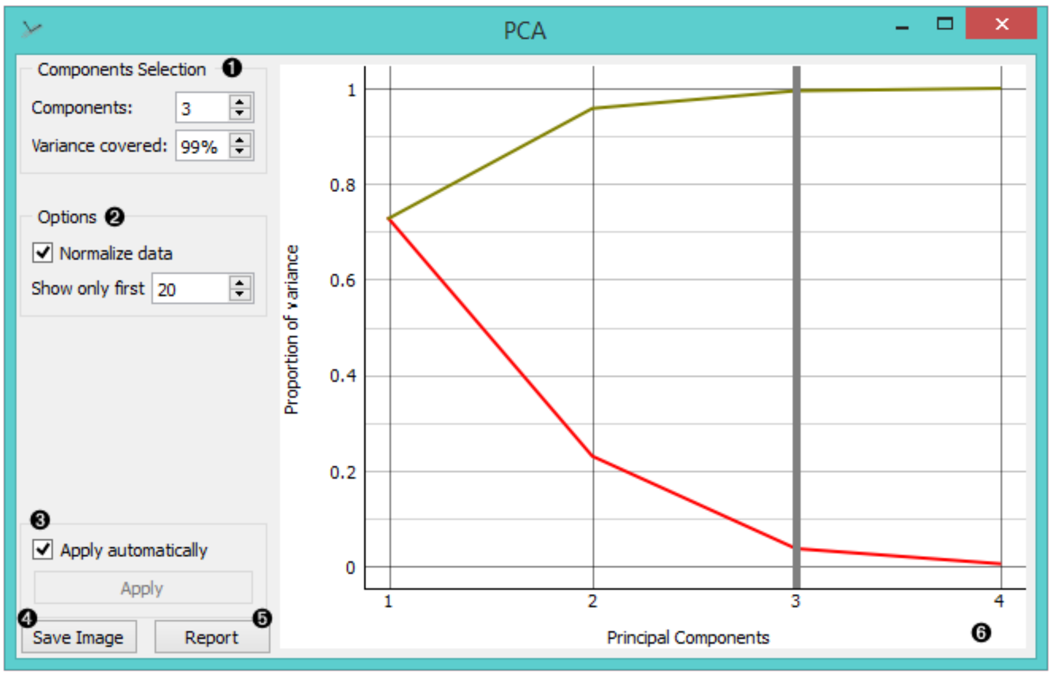
  PCA transformed input data.

- **Components**

  Eigenvectors.

Description

Principal Component Analysis (PCA) computes the PCA linear transformation of the input data. It outputs either a transformed data set with weights of individual instances or weights of principal components.



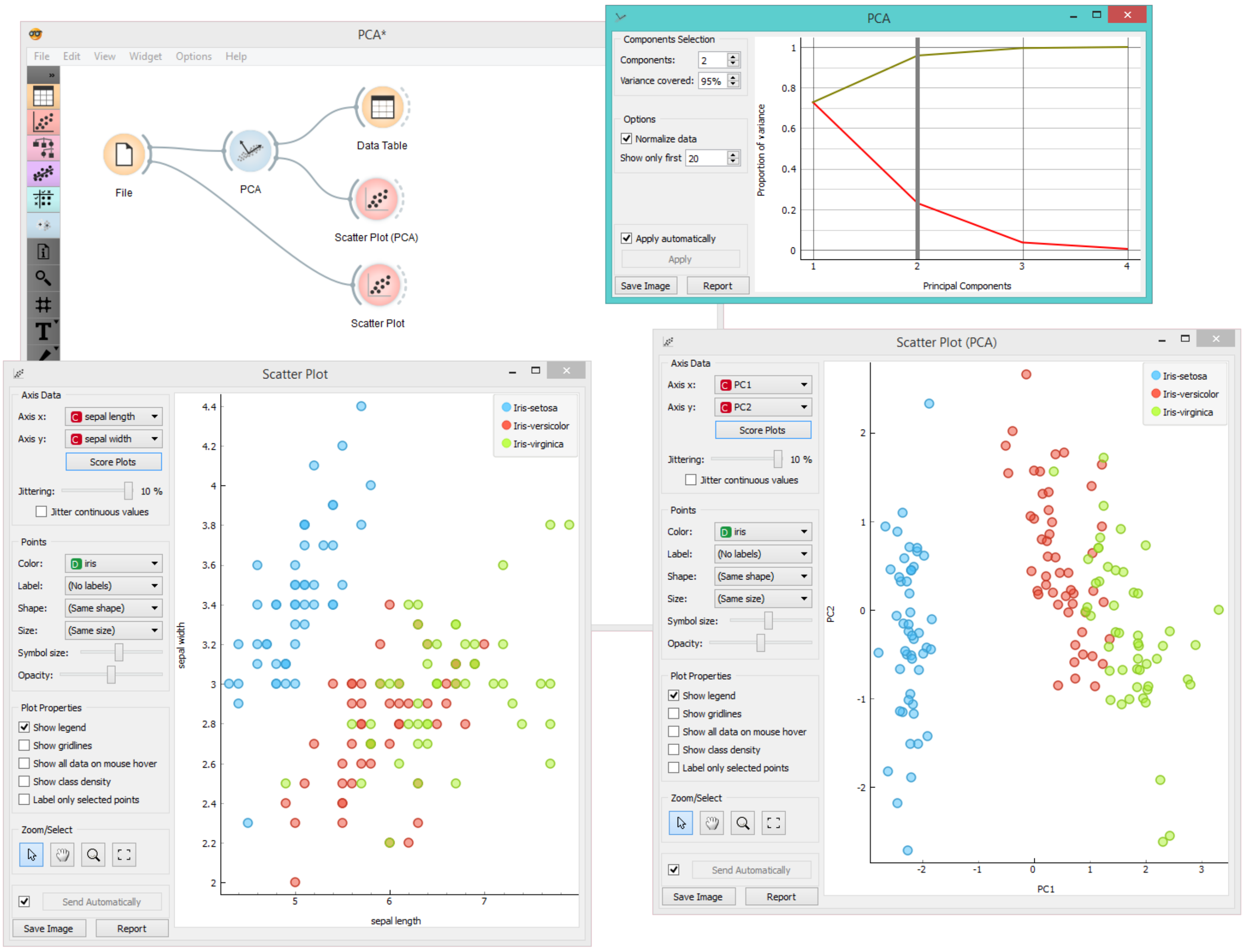
1. Select how many principal components you wish in your output. It is best to choose as few as possible with variance covered as high as possible. You can also set how much variance you wish to cover with your principal components.
2. You can normalize data to adjust the values to common scale.
3. When *Apply Automatically* is ticked, the widget will automatically communicate all changes. Alternatively, click *Apply*.
4. Press *Save Image* if you want to save the created image to your computer.

5. Produce a report.
6. Principal components graph, where the red (lower) line is the variance covered per component and the green (upper) line is cumulative variance covered by components.

The number of components of the transformation can be selected either in the *Components Selection* input box or by dragging the vertical cutoff line in the graph.

Examples

PCA can be used to simplify visualizations of large data sets. Below, we used the *Iris* data set to show how we can improve the visualization of the data set with PCA. The transformed data in the Scatter Plot show a much clearer distinction between classes than the default settings.

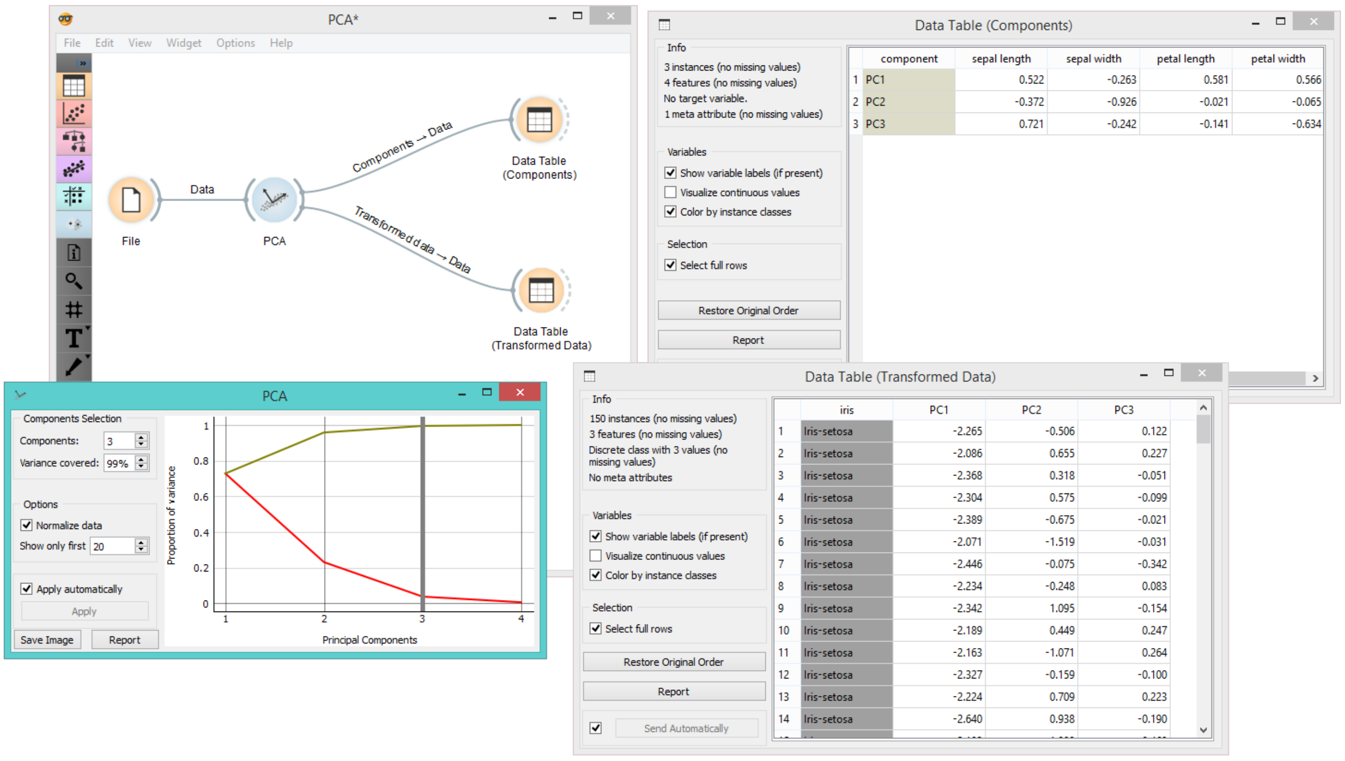


PCA: Components: 2; Variance covered: 95%; ✓ Normalize data; Show only first 20; ✓ Apply automatically

Scatter Plot: Axis x: sepal length; Axis y: sepal width; Jittering: 10 %; Color: iris; Label: (No labels); Shape: (Same shape); Size: (Same size); ✓ Show legend; ✓ Send Automatically

Scatter Plot (PCA): Axis x: PC1; Axis y: PC2; Jittering: 10 %; Color: iris; Label: (No labels); Shape: (Same shape); Size: (Same size); ✓ Show legend; ✓ Send Automatically

The widget provides two outputs: transformed data and principal components. Transformed data are weights for individual instances in the new coordinate system, while components are the system descriptors (weights for princial components). When fed into the Data Table, we can see both outputs in numerical form. We used two data tables in order to provide a more clean visualization of the workflow, but you can also choose to edit the links in such a way that you display the data in just one data table. You only need to create two links and connect the *Transformed data* and *Components* inputs to the *Data* output.



PCA

Components Selection
Components: 3
Variance covered: 99%

Options
☑ Normalize data
Show only first 20

☑ Apply automatically

Data Table (Components)

Info
3 instances (no missing values)
4 features (no missing values)
No target variable.
1 meta attribute (no missing values)

Variables
☑ Show variable labels (if present)
☐ Visualize continuous values
☑ Color by instance classes

Selection
☑ Select full rows

	component	sepal length	sepal width	petal length	petal width
1	PC1	0.522	-0.263	0.581	0.566
2	PC2	-0.372	-0.926	-0.021	-0.065
3	PC3	0.721	-0.242	-0.141	-0.634

Data Table (Transformed Data)

Info
150 instances (no missing values)
3 features (no missing values)
Discrete class with 3 values (no missing values)
No meta attributes

Variables
☑ Show variable labels (if present)
☐ Visualize continuous values
☑ Color by instance classes

Selection
☑ Select full rows

☑ Send Automatically

	iris	PC1	PC2	PC3
1	Iris-setosa	-2.265	-0.506	0.122
2	Iris-setosa	-2.086	0.655	0.227
3	Iris-setosa	-2.368	0.318	-0.051
4	Iris-setosa	-2.304	0.575	-0.099
5	Iris-setosa	-2.389	-0.675	-0.021
6	Iris-setosa	-2.071	-1.519	-0.031
7	Iris-setosa	-2.446	-0.075	-0.342
8	Iris-setosa	-2.234	-0.248	0.083
9	Iris-setosa	-2.342	1.095	-0.154
10	Iris-setosa	-2.189	0.449	0.247
11	Iris-setosa	-2.163	-1.071	0.264
12	Iris-setosa	-2.327	-0.159	-0.100
13	Iris-setosa	-2.224	0.709	0.223
14	Iris-setosa	-2.640	0.938	-0.190

Save Distance Matrix

Saves a distance matrix.

Signals

Inputs:

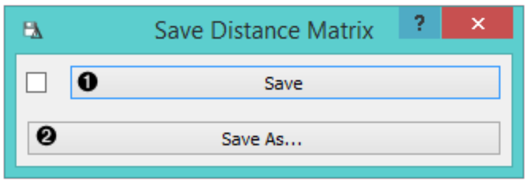
- **Distances**

  A distance matrix.

Outputs:

- None

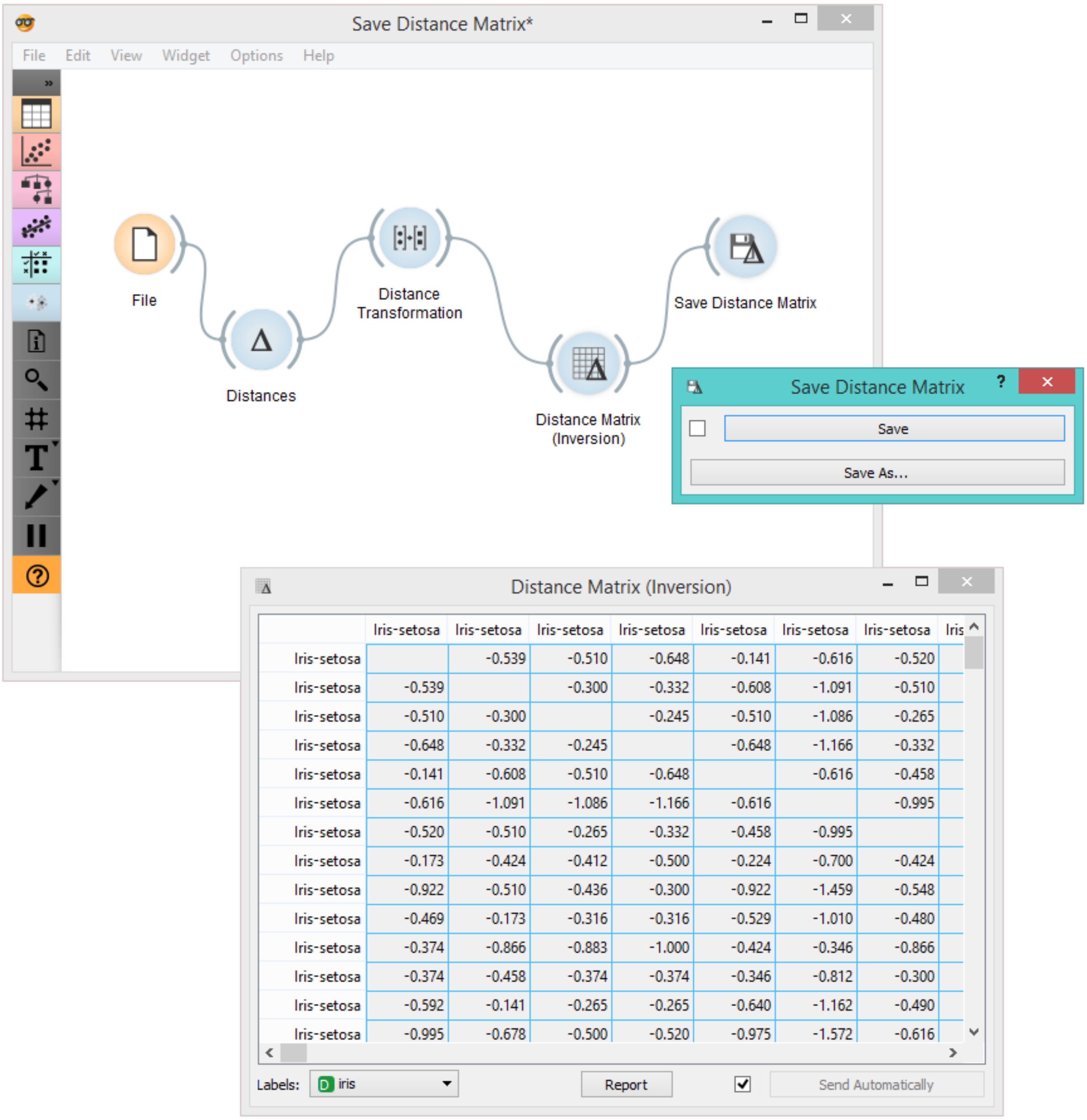
Description



1. By clicking *Save*, you choose from previously saved distance matrices. Alternatively, tick the box on the left side of the *Save* button and changes will be communicated automatically.
2. By clicking *Save as*, you save the distance matrix to your computer, you only need to enter the name of the file and click *Save*. The distance matrix will be saved as type *.dst*.

Example

In the snapshot below, we used the Distance Transformation widget to transform the distances in the *Iris* data set. We then chose to save the transformed version to our computer, so we could use it later on. We decided to output all data instances. You can choose to output just a minor subset of the data matrix. Pairs are marked automatically. If you wish to know what happened to our changed file, go here



	Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa	Iris
Iris-setosa		-0.539	-0.510	-0.648	-0.141	-0.616	-0.520	
Iris-setosa	-0.539		-0.300	-0.332	-0.608	-1.091	-0.510	
Iris-setosa	-0.510	-0.300		-0.245	-0.510	-1.086	-0.265	
Iris-setosa	-0.648	-0.332	-0.245		-0.648	-1.166	-0.332	
Iris-setosa	-0.141	-0.608	-0.510	-0.648		-0.616	-0.458	
Iris-setosa	-0.616	-1.091	-1.086	-1.166	-0.616		-0.995	
Iris-setosa	-0.520	-0.510	-0.265	-0.332	-0.458	-0.995		
Iris-setosa	-0.173	-0.424	-0.412	-0.500	-0.224	-0.700	-0.424	
Iris-setosa	-0.922	-0.510	-0.436	-0.300	-0.922	-1.459	-0.548	
Iris-setosa	-0.469	-0.173	-0.316	-0.316	-0.529	-1.010	-0.480	
Iris-setosa	-0.374	-0.866	-0.883	-1.000	-0.424	-0.346	-0.866	
Iris-setosa	-0.374	-0.458	-0.374	-0.374	-0.346	-0.812	-0.300	
Iris-setosa	-0.592	-0.141	-0.265	-0.265	-0.640	-1.162	-0.490	
Iris-setosa	-0.995	-0.678	-0.500	-0.520	-0.975	-1.572	-0.616	

Evaluation

Calibration Plot

Confusion Matrix

Lift Curve

Predictions

ROC Analysis

Test & Score

Calibration Plot

Shows the match between classifiers' probability predictions and actual class probabilities.

Signals

Inputs:

- **Evaluation Results**

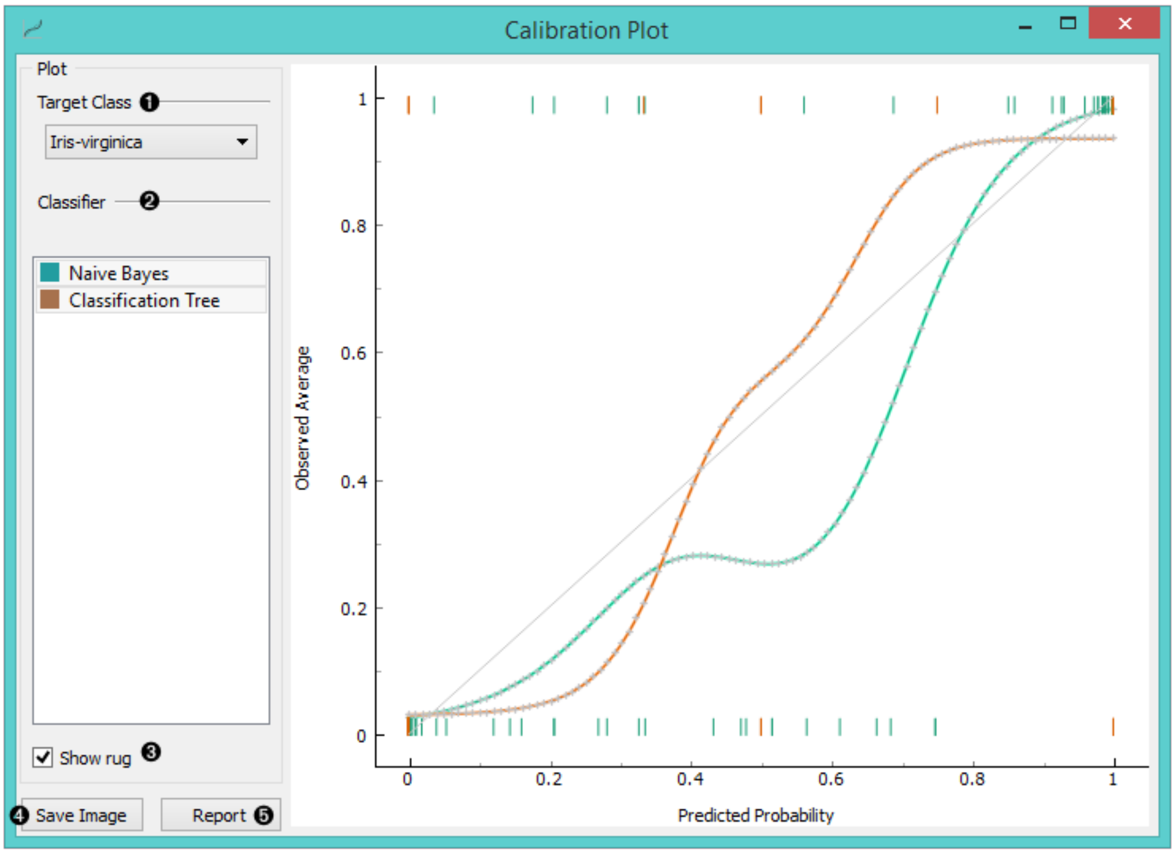
  Results of testing classification algorithms.

Outputs:

- None

Description

The Calibration Plot plots class probabilities against those predicted by the classifier(s).



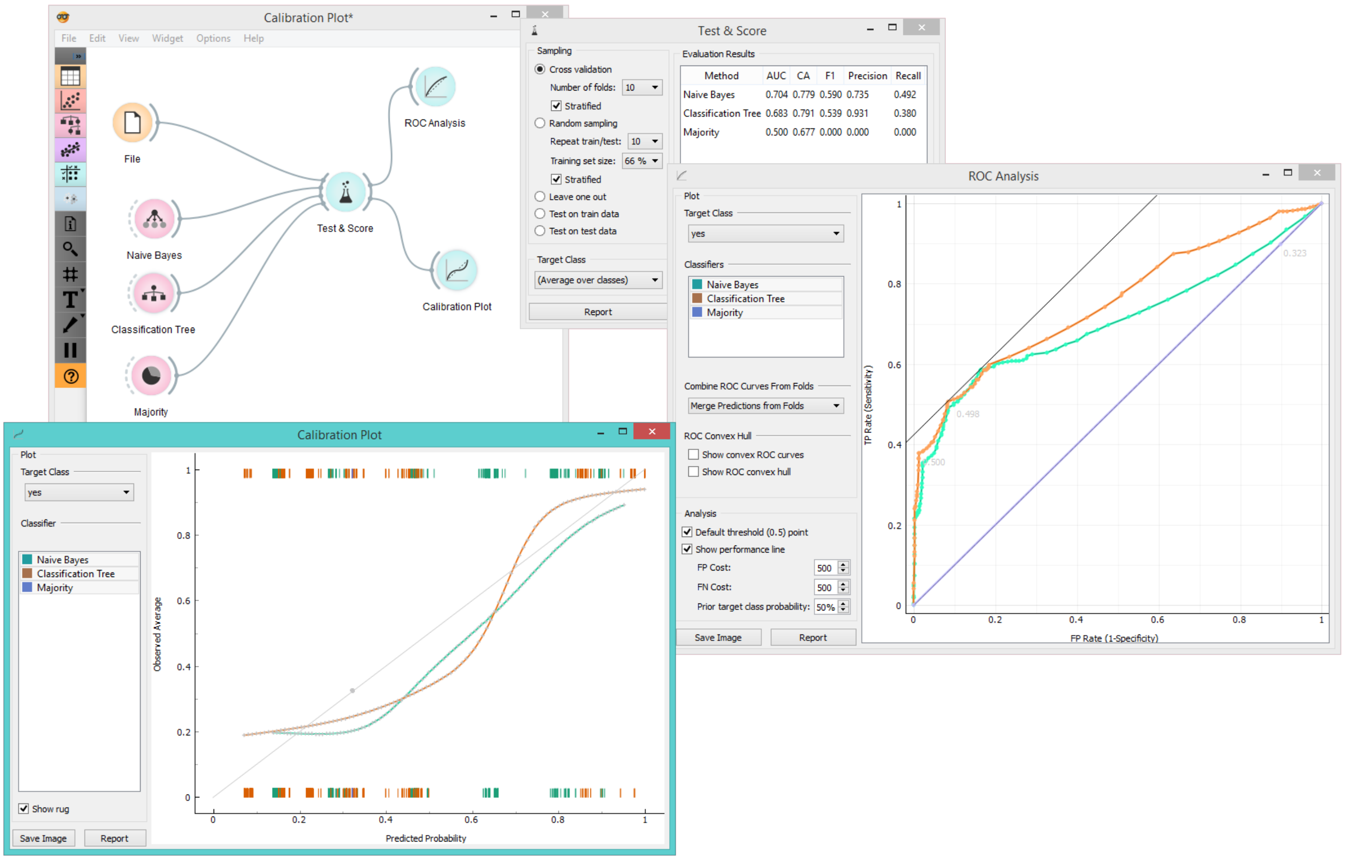
1. Select the desired target class from the drop down menu.
2. Choose which classifiers to plot. The diagonal represents optimal behaviour; the closer the classifier's curve gets, the more accurate its prediction probabilities are. Thus we would use this widget to see whether a classifier is overly optimistic (gives predominantly positive results) or pesimitistic (gives predominantly negative results).
3. If *Show rug* is enabled, ticks are displayed at the bottom and the top of the graph, which represent negative and positive examples respectively. Their position corresponds to the classifier's probability prediction and the color shows the classifier. At the bottom of the graph, the points to the left are those which are (correctly) assigned a low probability of the target class, and those to the right are incorrectly assigned high probabilities. At the top of

the graph, the instances to the right are correctly assigned high probabilities and vice versa.
4. Press *Save Image* if you want to save the created image to your computer in a .svg or .png format.
5. Produce a report.

Example

At the moment, the only widget which gives the right type of signal needed by the **Calibration Plot** is Test&Score. The Calibration Plot will hence always follow Test&Score and, since it has no outputs, no other widgets follow it.

Here is a typical example, where we compare three classifiers (namely Naive Bayes, Tree and Constant) and input them into Test&Score. We used the *Titanic* data set. Test&Score then displays evaluation results for each classifier. Then we draw **Calibration Plot** and ROC Analysis widgets from Test&Score to further analyze the performance of classifiers. **Calibration Plot** enables you to see prediction accuracy of class probabilities in a plot.



Test & Score

Sampling
- ◉ Cross validation
  - Number of folds: 10
  - ✓ Stratified
- ○ Random sampling
  - Repeat train/test: 10
  - Training set size: 66 %
  - ✓ Stratified
- ○ Leave one out
- ○ Test on train data
- ○ Test on test data

Target Class: (Average over classes)

Evaluation Results

Method	AUC	CA	F1	Precision	Recall
Naive Bayes	0.704	0.779	0.590	0.735	0.492
Classification Tree	0.683	0.791	0.539	0.931	0.380
Majority	0.500	0.677	0.000	0.000	0.000

ROC Analysis

Plot
- Target Class: yes
- Classifiers: 🟩 Naive Bayes, 🟫 Classification Tree, 🟦 Majority
- Combine ROC Curves From Folds: Merge Predictions from Folds
- ROC Convex Hull: ☐ Show convex ROC curves, ☐ Show ROC convex hull
- Analysis: ✓ Default threshold (0.5) point, ✓ Show performance line
  - FP Cost: 500
  - FN Cost: 500
  - Prior target class probability: 50%

Calibration Plot

Plot
- Target Class: yes
- Classifier: 🟩 Naive Bayes, 🟫 Classification Tree, 🟦 Majority
- ✓ Show rug

Confusion Matrix

Shows proportions between the predicted and actual class.

Signals

Inputs:

- **Evaluation results**

  Results of testing the algorithms; typically from **Test Learners**

Outputs:

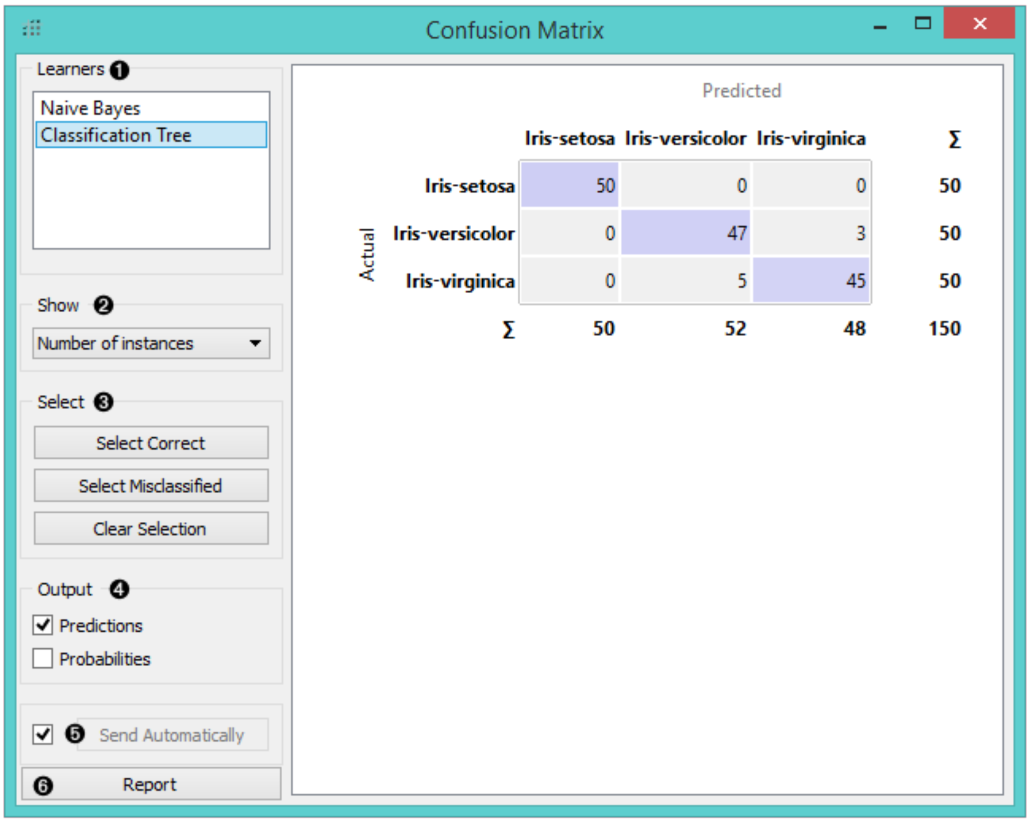
- **Selected Data**

  A data subset from the selected cells in the confusion matrix.

Description

The Confusion Matrix gives the number/proportion of instances between the predicted and actual class. The selection of the elements in the matrix feeds the corresponding instances into the output signal. This way, one can observe which specific instances were misclassified and how.

The widget usually gets the evaluation results from Test & Score; an example of the schema is shown below.



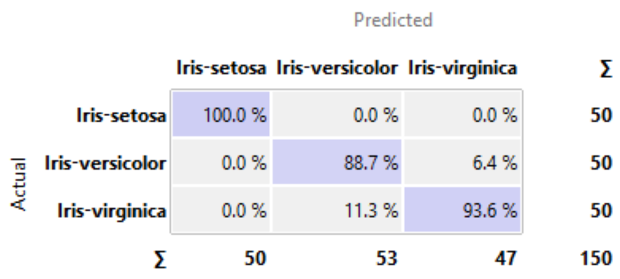
Actual \ Predicted	Iris-setosa	Iris-versicolor	Iris-virginica	Σ
Iris-setosa	50	0	0	50
Iris-versicolor	0	47	3	50
Iris-virginica	0	5	45	50
Σ	50	52	48	150

1. When evaluation results contain data on multiple learning algorithms, we have to choose one in the *Learners* box.

The snapshot shows the confusion matrix for Tree and Naive Bayesian models trained and tested on the *iris* data. The righthand side of the widget contains the matrix for the naive Bayesian model (since this model is selected on the left). Each row corresponds to a correct class, while columns represent the predicted classes. For instance, four instances of *Iris-versicolor* were misclassified as *Iris-virginica*. The rightmost column gives the number of instances from each class (there are 50 irises of each of the three classes) and the bottom row gives the number of instances classified into each class (e.g., 48 instances were classified into virginica).

2. In *Show*, we select what data we would like to see in the matrix.

    - **Number of instances** shows correctly and incorrectly classified instances numerically.
    - **Proportions of predicted** shows how many instances classified as, say, *Iris-versicolor* are in which true class; in the table we can read the 0% of them are actually setosae, 88.5% of those classified as versicolor are versicolors, and 7.7% are virginicae.
    - **Proportions of actual** shows the opposite relation: of all true versicolors, 92% were classified as versicolors and 8% as virginicae.



Actual \ Predicted	Iris-setosa	Iris-versicolor	Iris-virginica	Σ
Iris-setosa	100.0 %	0.0 %	0.0 %	50
Iris-versicolor	0.0 %	88.7 %	6.4 %	50
Iris-virginica	0.0 %	11.3 %	93.6 %	50
Σ	50	53	47	150

3. In *Select*, you can choose the desired output.

    - **Correct** sends all correctly classified instances to the output by selecting the diagonal of the matrix.

    - **Misclassified** selects the misclassified instances.

    - **None** annuls the selection.

    As mentioned before, one can also select individual cells of the table to select specific kinds of misclassified instances (e.g. the versicolors classified as virginicae).

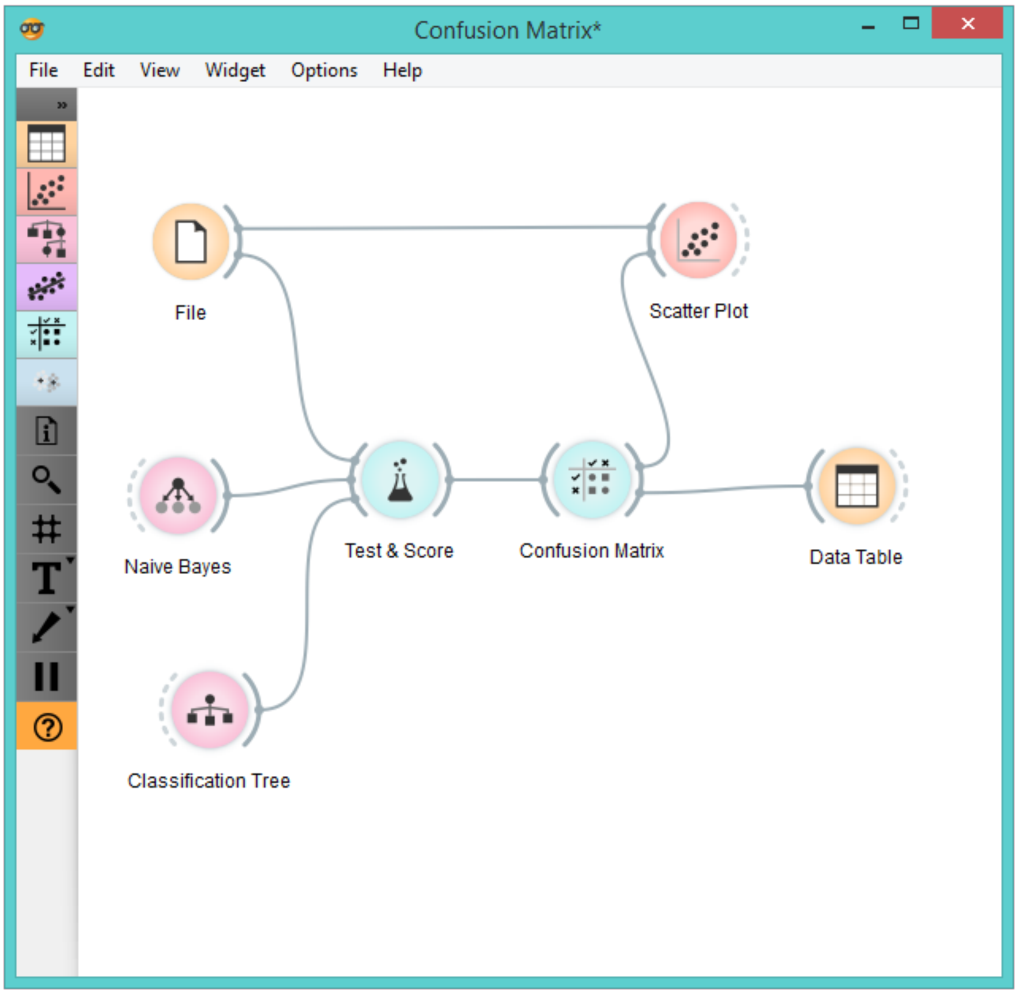
4. When sending selected instances, the widget can add new attributes, such as predicted classes or their probabilities, if the corresponding options *Predictions* and/or *Probabilities* are checked.

5. The widget outputs every change if *Send Automatically* is ticked. If not, the user will need to click *Send Selected* to commit the changes.

6. Produce a report.

Example

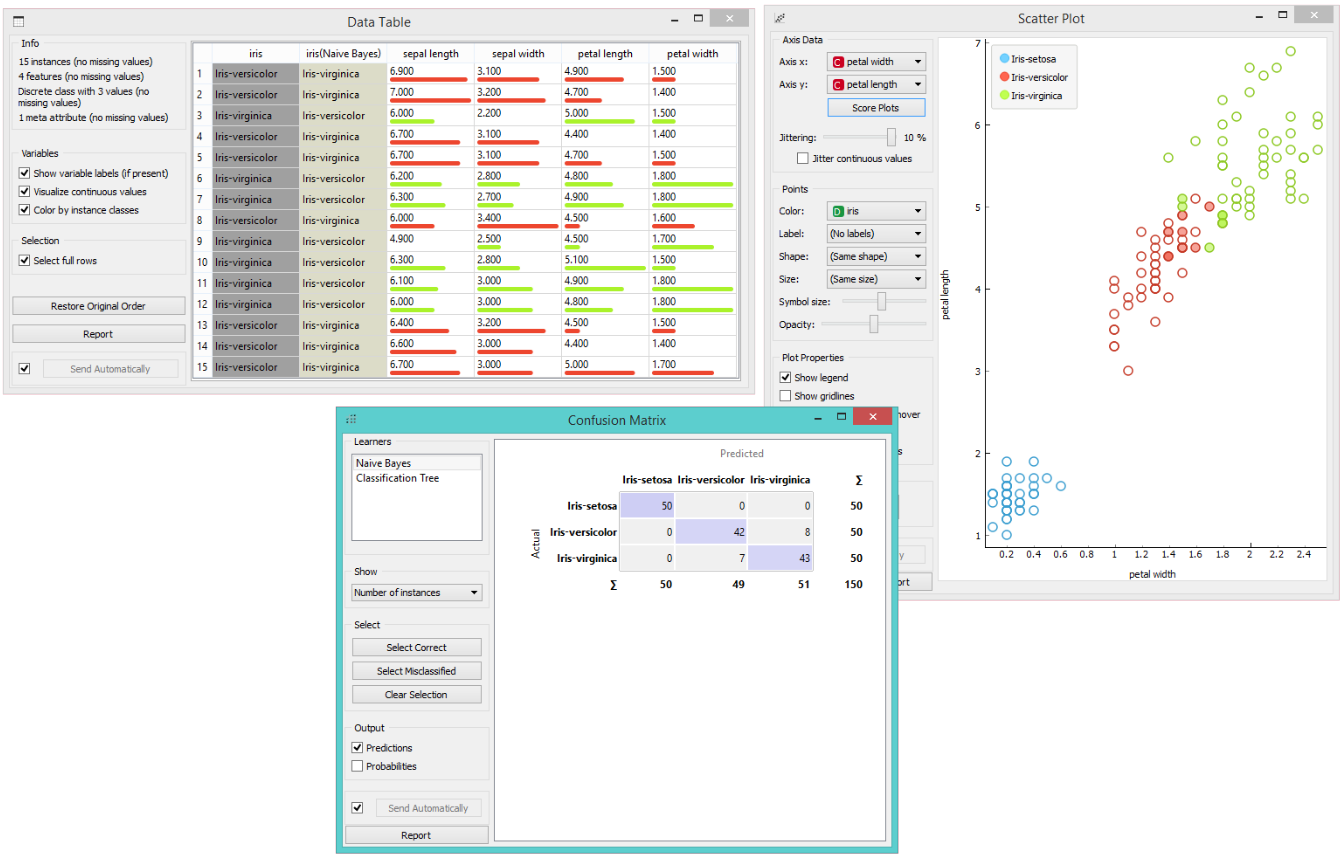
The following workflow demonstrates what this widget can be used for.



Test & Score gets the data from File and two learning algorithms from Naive Bayes and Tree. It performs cross-validation or some other train-and-test procedures to get class predictions by both algorithms for all (or some) data instances. The test results are fed into the **Confusion Matrix**, where we can observe how many instances were misclassified and in which way.

In the output, we used Data Table to show the instances we selected in the confusion matrix. If we, for instance, click *Misclassified*, the table will contain all instances which were misclassified by the selected method.

The Scatterplot gets two sets of data. From the File widget it gets the complete data, while the confusion matrix sends only the selected data, misclassifications for instance. The scatter plot will show all the data, with bold symbols representing the selected data.



15 instances (no missing values)
4 features (no missing values)
Discrete class with 3 values (no missing values)
1 meta attribute (no missing values)

- [✓] Show variable labels (if present)
- [✓] Visualize continuous values
- [✓] Color by instance classes
- [✓] Select full rows
- [✓] Send Automatically

	iris	iris(Naive Bayes)	sepal length	sepal width	petal length	petal width
1	Iris-versicolor	Iris-virginica	6.900	3.100	4.900	1.500
2	Iris-versicolor	Iris-virginica	7.000	3.200	4.700	1.400
3	Iris-virginica	Iris-versicolor	6.000	2.200	5.000	1.500
4	Iris-versicolor	Iris-virginica	6.700	3.100	4.400	1.400
5	Iris-versicolor	Iris-virginica	6.700	3.100	4.700	1.500
6	Iris-virginica	Iris-versicolor	6.200	2.800	4.800	1.800
7	Iris-virginica	Iris-versicolor	6.300	2.700	4.900	1.800
8	Iris-versicolor	Iris-virginica	6.000	3.400	4.500	1.600
9	Iris-virginica	Iris-versicolor	4.900	2.500	4.500	1.700
10	Iris-virginica	Iris-versicolor	6.300	2.800	5.100	1.500
11	Iris-virginica	Iris-versicolor	6.100	3.000	4.900	1.800
12	Iris-virginica	Iris-versicolor	6.000	3.000	4.800	1.800
13	Iris-versicolor	Iris-virginica	6.400	3.200	4.500	1.500
14	Iris-versicolor	Iris-virginica	6.600	3.000	4.400	1.400
15	Iris-versicolor	Iris-virginica	6.700	3.000	5.000	1.700

Axis x: petal width
Axis y: petal length
Jittering: 10 %
- [ ] Jitter continuous values

Color: iris
Label: (No labels)
Shape: (Same shape)
Size: (Same size)
Symbol size:
Opacity:

- [✓] Show legend
- [ ] Show gridlines

Learners: Naive Bayes, Classification Tree
Show: Number of instances

Actual \ Predicted	Iris-setosa	Iris-versicolor	Iris-virginica	Σ
Iris-setosa	50	0	0	50
Iris-versicolor	0	42	8	50
Iris-virginica	0	7	43	50
Σ	50	49	51	150

- [✓] Predictions
- [ ] Probabilities
- [✓] Send Automatically

Lift Curve

Measures the performance of a chosen classifier against a random classifier.

Signals

Inputs:

- **Evaluation Results**

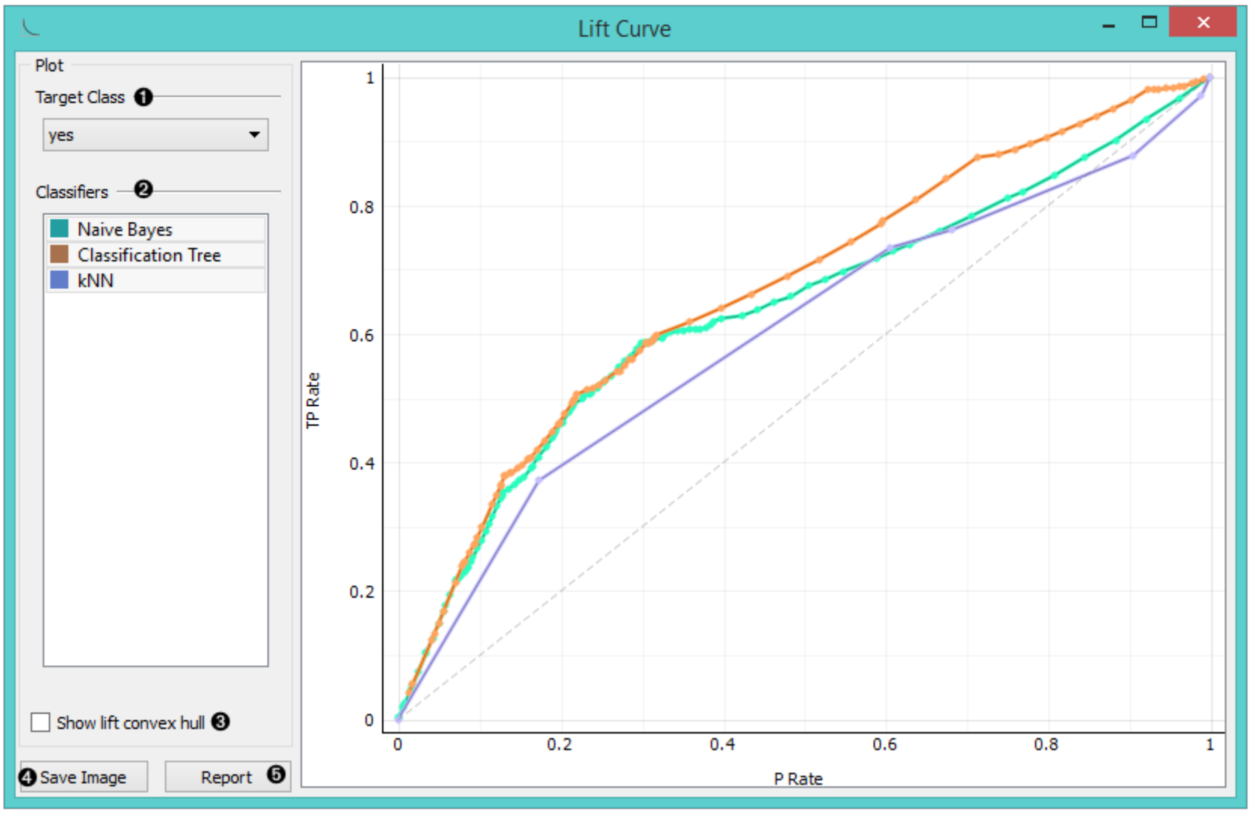
  Results of classifiers' tests on data.

Outputs:

- None

Description

The **Lift curve** shows the relation between the number of instances which were predicted positive and those that are indeed positive and thus measures the performance of a chosen classifier against a random classifier. The graph is constructed with the cumulative number of cases (in descending order of probability) on the x-axis and the cumulative number of true positives on the y-axis. Lift curve is often used in segmenting the population, e.g., plotting the number of responding customers against the number of all customers contacted. You can also determine the optimal classifier and its threshold from the graph.



1. Choose the desired *Target class*. The default class is chosen alphabetically.
2. If test results contain more than one classifier, the user can choose which curves she or he wants to see plotted. Click on a classifier to select or deselect the curve.
3. *Show lift convex hull* plots a convex hull over lift curves for all classifiers (yellow curve). The curve shows the optimal classifier (or combination thereof) for each desired TP/P rate.
4. Press *Save Image* if you want to save the created image to your computer in a .svg or .png format.

5. Produce a report.
6. 2-D pane with **P rate** (population) as x-axis and **TP rate** (true positives) as a y-axis. The diagonal line represents the behaviour of a random classifier. Click and drag to move the pane and scroll in or out to zoom. Click on the "*A*" sign at the bottom left corner to realign the pane.

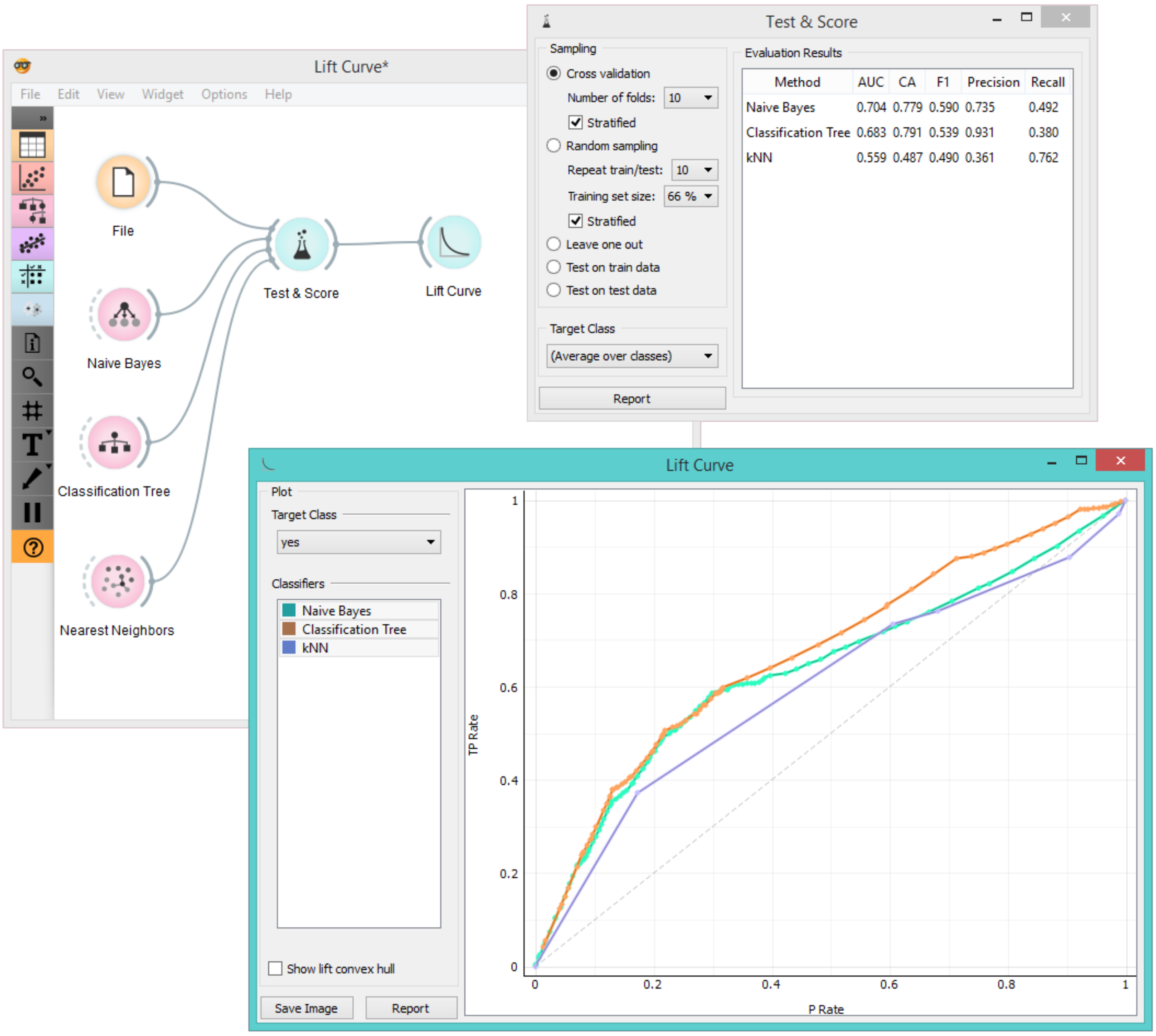
Note:

The perfect classifier would have a steep slope towards 1 until all classes are guessed correctly and then run straight along 1 on y-axis to (1,1).

Example

At the moment, the only widget which gives the right type of the signal needed by the **Lift Curve** is Test&Score.

In the example below, we try to see the prediction quality for the class 'survived' on the *Titanic* data set. We compared three different classifiers in the Test Learners widget and sent them to Lift Curve to see their performance against a random model. We see the Tree classifier is the best out of the three, since it best aligns with *lift convex hull*. We also see that its performance is the best for the first 30% of the population (in order of descending probability), which we can set as the threshold for optimal classification.



Sampling
- (●) Cross validation
  - Number of folds: 10
  - [✓] Stratified
- ( ) Random sampling
  - Repeat train/test: 10
  - Training set size: 66 %
  - [✓] Stratified
- ( ) Leave one out
- ( ) Test on train data
- ( ) Test on test data

Target Class: (Average over classes)

Evaluation Results

Method	AUC	CA	F1	Precision	Recall
Naive Bayes	0.704	0.779	0.590	0.735	0.492
Classification Tree	0.683	0.791	0.539	0.931	0.380
kNN	0.559	0.487	0.490	0.361	0.762

Plot

Target Class: yes

Classifiers
- 🟩 Naive Bayes
- 🟫 Classification Tree
- 🟦 kNN

[ ] Show lift convex hull

References

Handouts of the University of Notre Dame on Data Mining - Lift Curve. Available here.

Predictions

Shows models' predictions on the data.

Signals

Inputs

- **Data**

  A data set.

- **Predictors**

  Predictors to be used on the data.

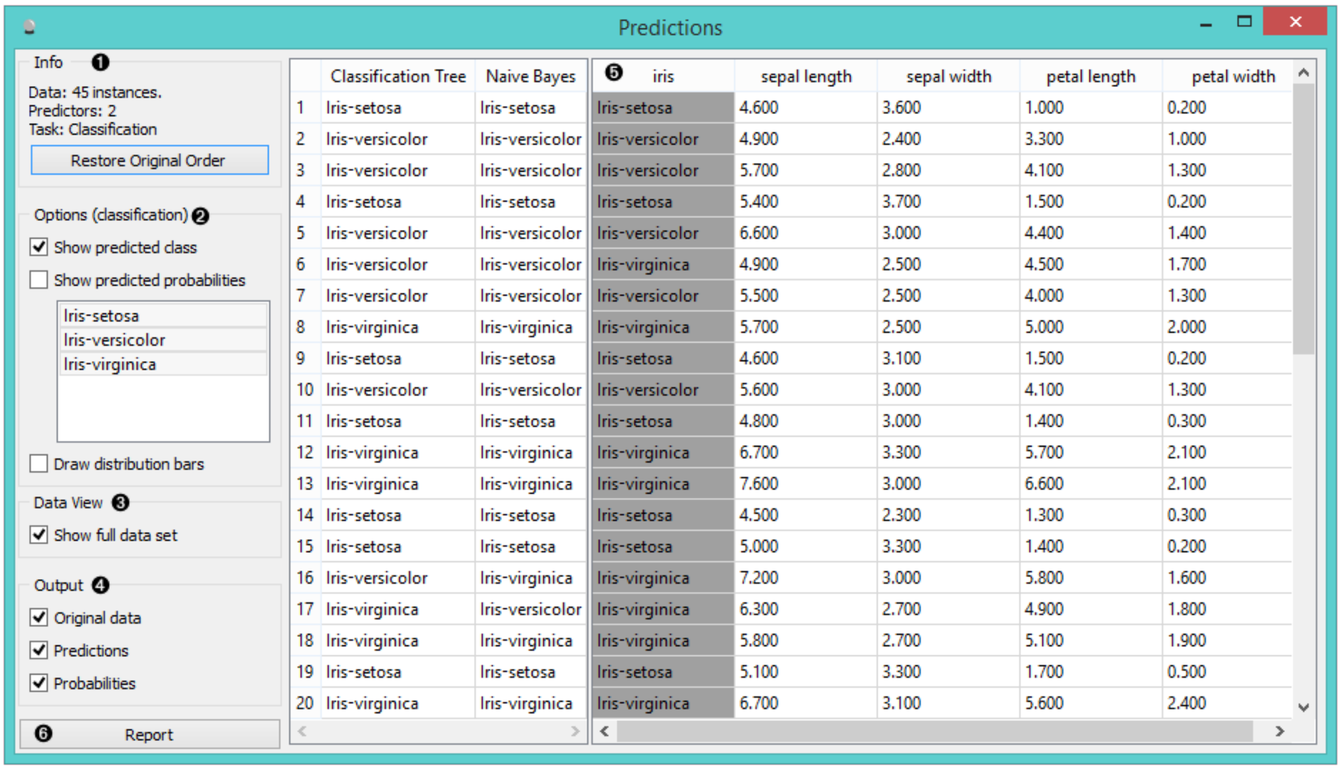
Outputs

- **Predictions**

  Original data with added predictions.

Description

The widget receives a data set and one or more predictors (classifiers, not learning algorithms - see the example below). It outputs the data and the predictions.


Predictions

Info ❶
Data: 45 instances.
Predictors: 2
Task: Classification
Restore Original Order

Options (classification) ❷
- [✓] Show predicted class
- [ ] Show predicted probabilities
  - Iris-setosa
  - Iris-versicolor
  - Iris-virginica
- [ ] Draw distribution bars

Data View ❸
- [✓] Show full data set

Output ❹
- [✓] Original data
- [✓] Predictions
- [✓] Probabilities

❻ Report

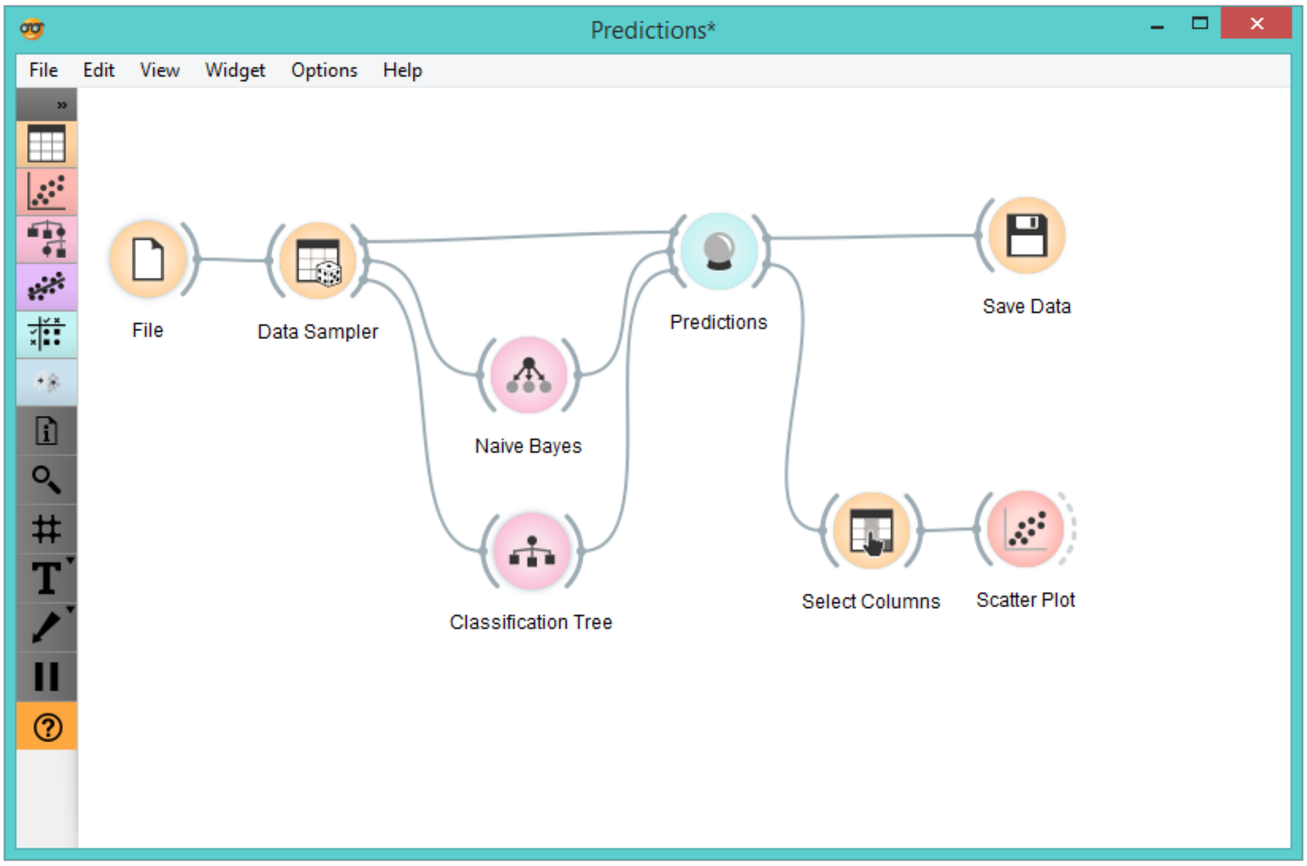
	Classification Tree	Naive Bayes	❺ iris	sepal length	sepal width	petal length	petal width
1	Iris-setosa	Iris-setosa	Iris-setosa	4.600	3.600	1.000	0.200
2	Iris-versicolor	Iris-versicolor	Iris-versicolor	4.900	2.400	3.300	1.000
3	Iris-versicolor	Iris-versicolor	Iris-versicolor	5.700	2.800	4.100	1.300
4	Iris-setosa	Iris-setosa	Iris-setosa	5.400	3.700	1.500	0.200
5	Iris-versicolor	Iris-versicolor	Iris-versicolor	6.600	3.000	4.400	1.400
6	Iris-versicolor	Iris-versicolor	Iris-virginica	4.900	2.500	4.500	1.700
7	Iris-versicolor	Iris-versicolor	Iris-versicolor	5.500	2.500	4.000	1.300
8	Iris-virginica	Iris-virginica	Iris-virginica	5.700	2.500	5.000	2.000
9	Iris-setosa	Iris-setosa	Iris-setosa	4.600	3.100	1.500	0.200
10	Iris-versicolor	Iris-versicolor	Iris-versicolor	5.600	3.000	4.100	1.300
11	Iris-setosa	Iris-setosa	Iris-setosa	4.800	3.000	1.400	0.300
12	Iris-virginica	Iris-virginica	Iris-virginica	6.700	3.300	5.700	2.100
13	Iris-virginica	Iris-virginica	Iris-virginica	7.600	3.000	6.600	2.100
14	Iris-setosa	Iris-setosa	Iris-setosa	4.500	2.300	1.300	0.300
15	Iris-setosa	Iris-setosa	Iris-setosa	5.000	3.300	1.400	0.200
16	Iris-versicolor	Iris-virginica	Iris-virginica	7.200	3.000	5.800	1.600
17	Iris-virginica	Iris-versicolor	Iris-virginica	6.300	2.700	4.900	1.800
18	Iris-virginica	Iris-virginica	Iris-virginica	5.800	2.700	5.100	1.900
19	Iris-setosa	Iris-setosa	Iris-setosa	5.100	3.300	1.700	0.500
20	Iris-virginica	Iris-virginica	Iris-virginica	6.700	3.100	5.600	2.400

1. Information on the input
2. The user can select the options for classification. If *Show predicted class* is ticked, the appended data table provides information on predicted class. If *Show predicted probabilities* is ticked, the appended data table provides information on probabilities predicted by the classifiers. The user can also select the predicted class he or she

wants displayed in the appended data table. The option *Draw distribution bars* provides a nice visualization of the predictions.
3. By ticking the *Show full data set*, the user can append the entire data table to the *Predictions* widget.
4. Select the desired output.
5. The appended data table
6. Produce a report.

Despite its simplicity, the widget allows for quite an interesting analysis of decisions of predictive models; there is a simple demonstration at the bottom of the page. Confusion Matrix is a related widget and although many things can be done with any of them, there are tasks for which one of them might be much more convenient than the other. The output of the widget is another data set, where predictions are appended as new meta attributes. You can select which features you wish to output (original data, predictions, probabilities). The resulting data set can be appended to the widget, but you can still choose to display it in a separate data table.

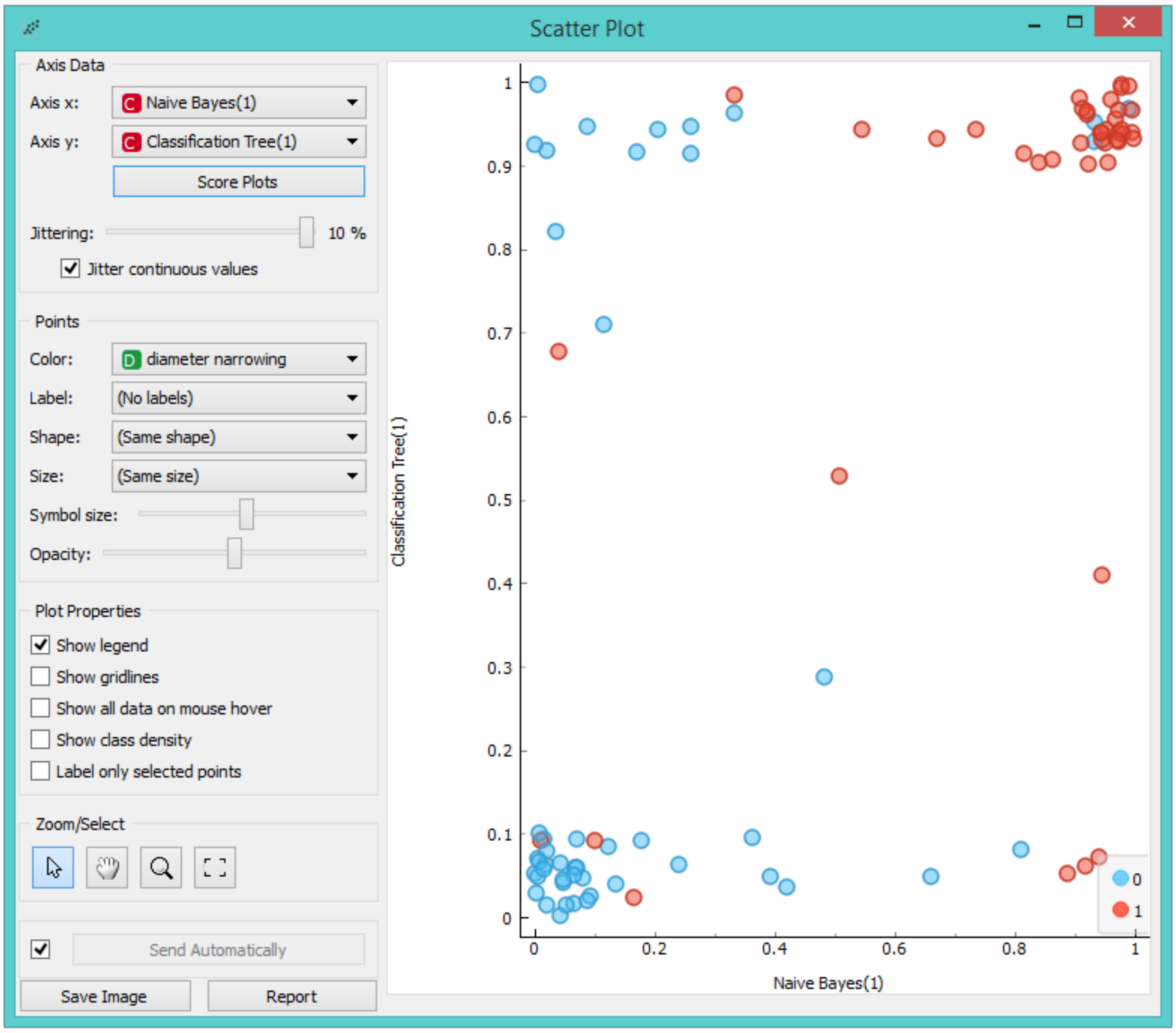
Example



We randomly split the *heart-disease* data into two subsets. The larger subset, containing 70 % of data instances, is sent to Naive Bayes and Tree, so they can produce the corresponding model. Models are then sent into **Predictions**, among with the remaining 30 % of the data. Predictions shows how these examples are classified.

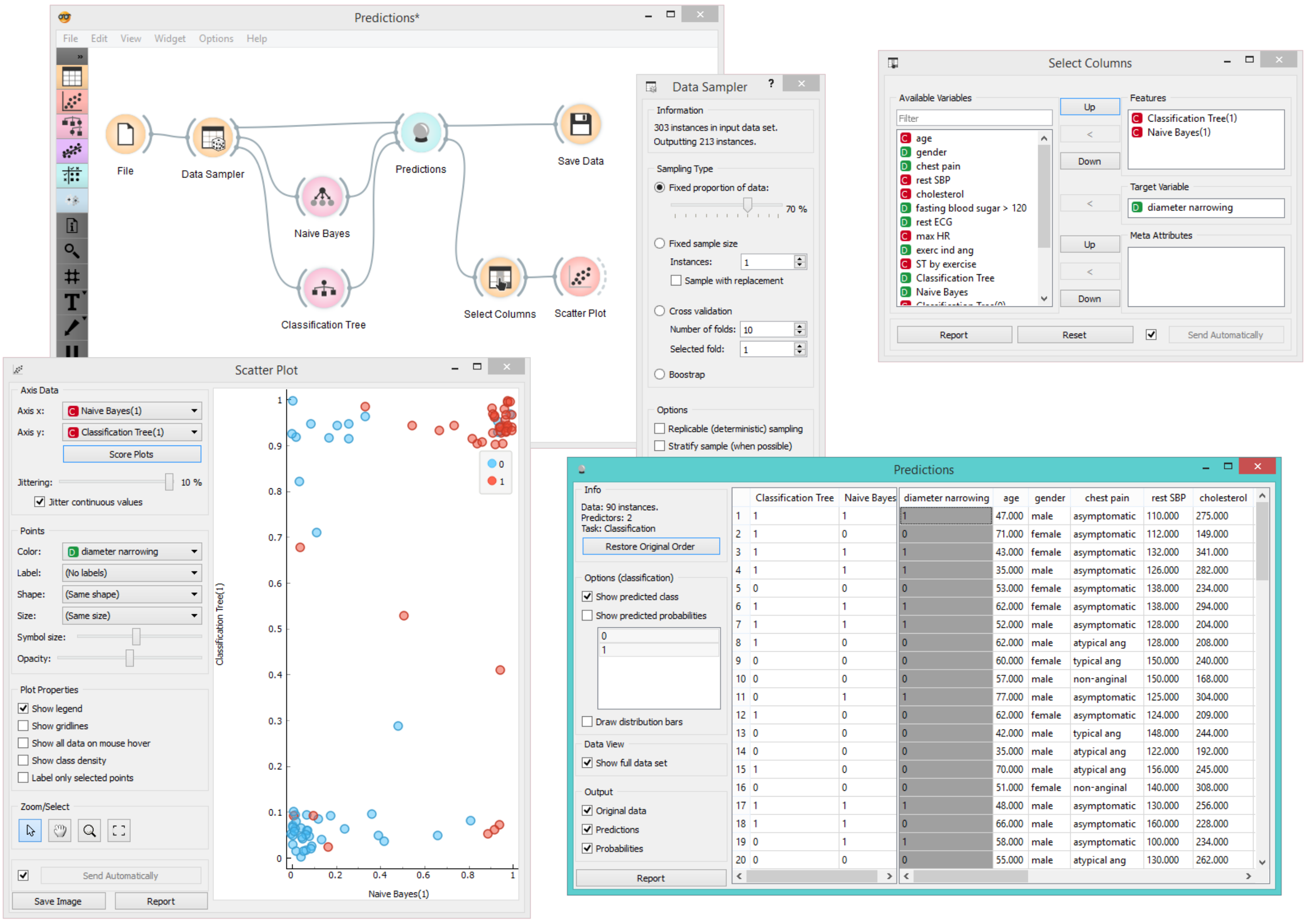
To save the predictions, we simply attach the Save widget to **Predictions**. The final file is a data table and can be saved as in a .tab or .tsv format.

Finally, we can analyze the models' predictions. For that, we first take Select Columns with which we move the meta attributes with probability predictions to features. The transformed data is then given to the Scatterplot, which we set to use the attributes with probabilities as the x and y axes, while the class is (already by default) used to color the data points.



To get the above plot, we selected *Jitter continuous values*, since the decision tree gives just a few distinct probabilities. The blue points in the bottom left corner represent the people with no diameter narrowing, which were correctly classified by both models. The upper right red points represent the patients with narrowed vessels, which were correctly classified by both.

Note that this analysis is done on a rather small sample, so these conclusions may be ungrounded. Here is the entire workflow:



Data Sampler

Information
303 instances in input data set.
Outputting 213 instances.

Sampling Type
(●) Fixed proportion of data: 70 %
( ) Fixed sample size
Instances: 1
☐ Sample with replacement
( ) Cross validation
Number of folds: 10
Selected fold: 1
( ) Boostrap

Options
☐ Replicable (deterministic) sampling
☐ Stratify sample (when possible)

Select Columns

Available Variables: age, gender, chest pain, rest SBP, cholesterol, fasting blood sugar > 120, rest ECG, max HR, exerc ind ang, ST by exercise, Classification Tree, Naive Bayes

Features: Classification Tree(1), Naive Bayes(1)

Target Variable: diameter narrowing

Meta Attributes:

☑ Send Automatically

Scatter Plot

Axis Data
Axis x: Naive Bayes(1)
Axis y: Classification Tree(1)
Jittering: 10 %
☑ Jitter continuous values

Points
Color: diameter narrowing
Label: (No labels)
Shape: (Same shape)
Size: (Same size)

Plot Properties
☑ Show legend
☐ Show gridlines
☐ Show all data on mouse hover
☐ Show class density
☐ Label only selected points

Predictions

Info
Data: 90 instances.
Predictors: 2
Task: Classification

Options (classification)
☑ Show predicted class
☐ Show predicted probabilities
☐ Draw distribution bars

Data View
☑ Show full data set

Output
☑ Original data
☑ Predictions
☑ Probabilities

	Classification Tree	Naive Bayes	diameter narrowing	age	gender	chest pain	rest SBP	cholesterol
1	1	1	1	47.000	male	asymptomatic	110.000	275.000
2	1	0	0	71.000	female	asymptomatic	112.000	149.000
3	1	1	1	43.000	female	asymptomatic	132.000	341.000
4	1	1	1	35.000	male	asymptomatic	126.000	282.000
5	0	0	0	53.000	female	asymptomatic	138.000	234.000
6	1	1	1	62.000	female	asymptomatic	138.000	294.000
7	1	1	1	52.000	male	asymptomatic	128.000	204.000
8	1	0	0	62.000	male	atypical ang	128.000	208.000
9	0	0	0	60.000	female	typical ang	150.000	240.000
10	0	0	0	57.000	male	non-anginal	150.000	168.000
11	0	1	1	77.000	male	asymptomatic	125.000	304.000
12	1	0	0	62.000	female	asymptomatic	124.000	209.000
13	0	0	0	42.000	male	typical ang	148.000	244.000
14	0	0	0	35.000	male	atypical ang	122.000	192.000
15	1	0	0	70.000	male	atypical ang	156.000	245.000
16	0	0	0	51.000	female	non-anginal	140.000	308.000
17	1	1	1	48.000	male	asymptomatic	130.000	256.000
18	1	1	0	66.000	male	asymptomatic	160.000	228.000
19	0	1	1	58.000	male	asymptomatic	100.000	234.000
20	0	0	0	55.000	male	atypical ang	130.000	262.000

Another example of using this widget is given in the documentation for the widget Confusion Matrix.

ROC Analysis

Plots a true positive rate against a false positive rate of a test.

Signals

Inputs:

- **Evaluation Results**

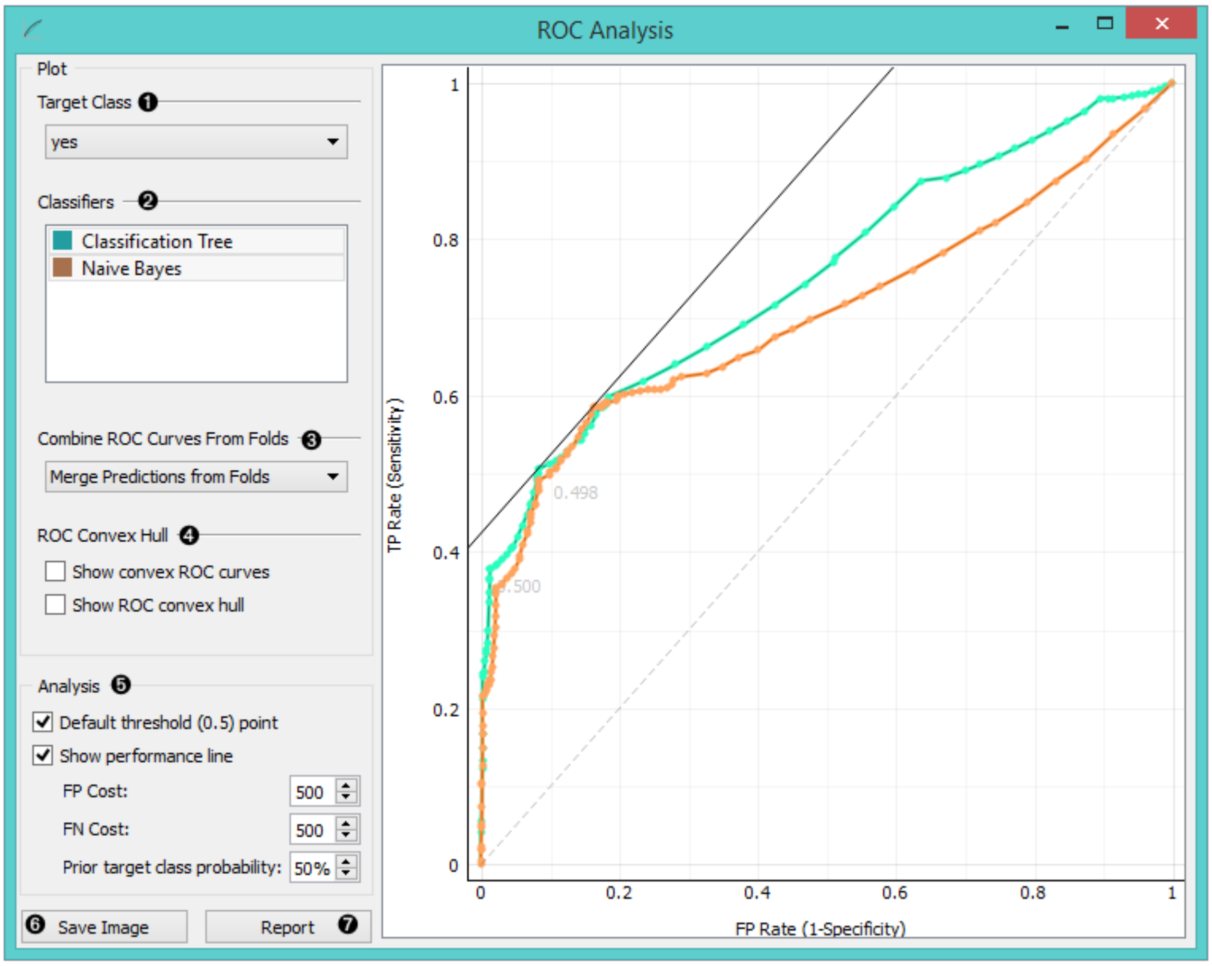
  Results of classifiers' tests on data

Outputs:

- None

Description

The widget shows ROC curves for the tested models and the corresponding convex hull. It serves as a mean of comparison between classification models. The curve plots a false positive rate on an x-axis (1-specificity; probability that target=1 when true value=0) against a true positive rate on a y-axis (sensitivity; probability that target=1 when true value=1). The closer the curve follows the left-hand border and then the top border of the ROC space, the more accurate the classifier. Given the costs of false positives and false negatives, the widget can also determine the optimal classifier and threshold.



ROC Analysis

Plot

Target Class ❶: yes

Classifiers ❷:
- 🟦 Classification Tree
- 🟫 Naive Bayes

Combine ROC Curves From Folds ❸: Merge Predictions from Folds

ROC Convex Hull ❹
- ☐ Show convex ROC curves
- ☐ Show ROC convex hull

Analysis ❺
- ☑ Default threshold (0.5) point
- ☑ Show performance line

FP Cost: 500

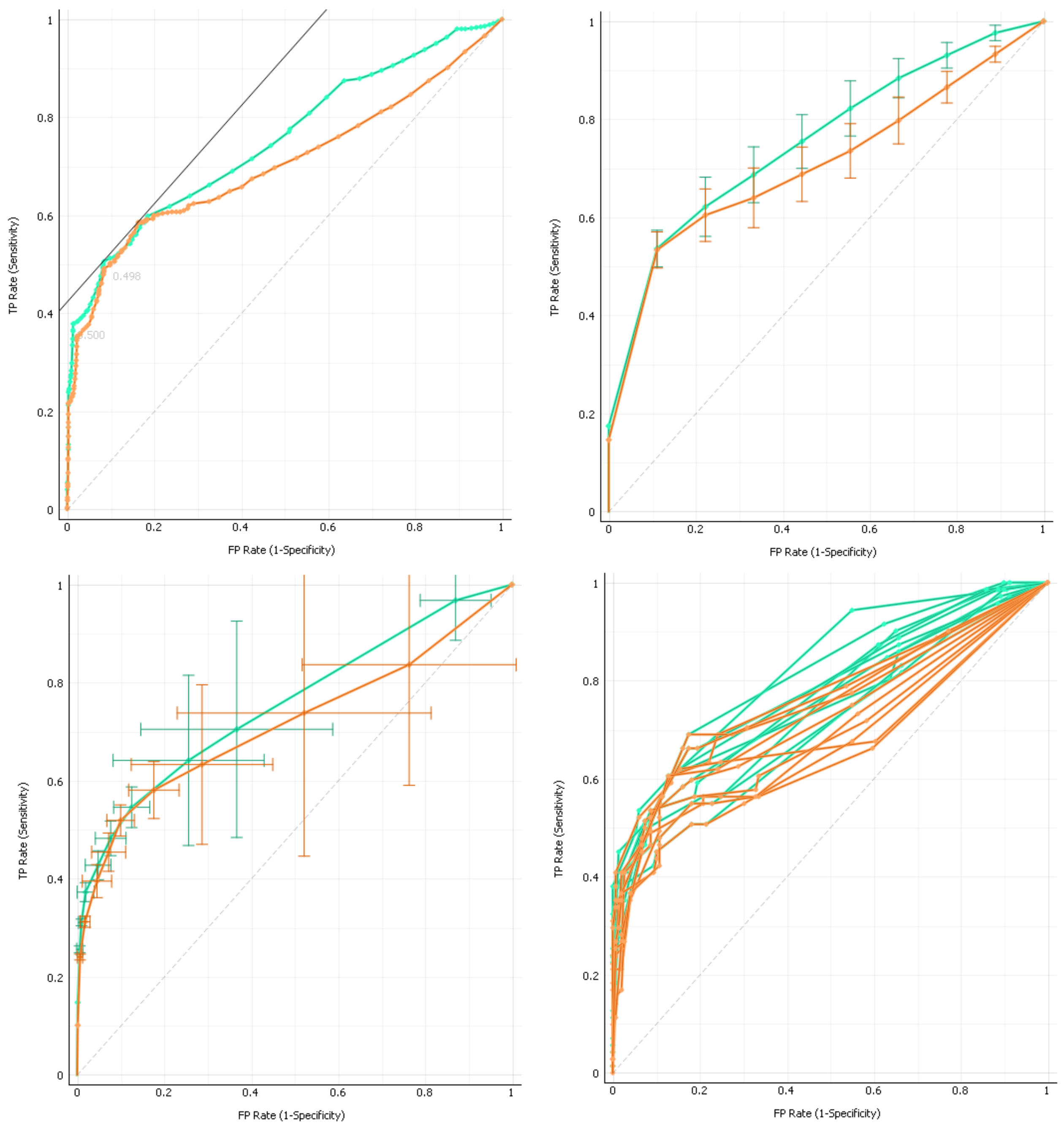
FN Cost: 500

Prior target class probability: 50%

1. Choose the desired *Target Class*. The default class is chosen alphabetically.

2. If test results contain more than one classifier, the user can choose which curves she or he wants to see plotted. Click on a classifier to select or deselect it.

3. When the data comes from multiple iterations of training and testing, such as k-fold cross validation, the results can be (and usually are) averaged.

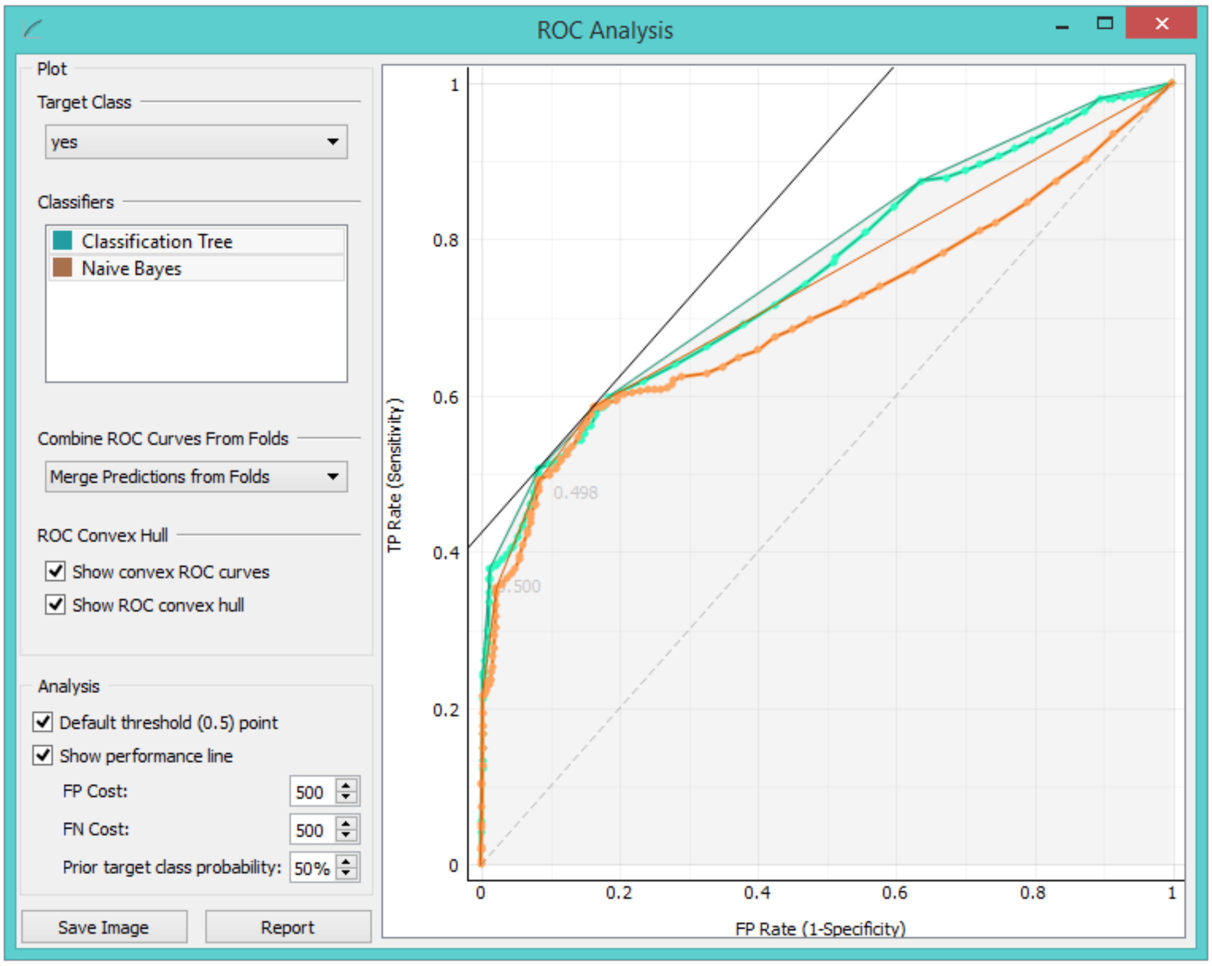


The averaging options are:

- **Merge predictions from folds** (top left), which treats all the test data as if they came from a single iteration
- **Mean TP rate** (top right) averages the curves vertically, showing the corresponding confidence intervals
- **Mean TP and FP at threshold** (bottom left) traverses over threshold, averages the positions of curves and shows horizontal and vertical confidence intervals
- **Show individual curves** (bottom right) does not average but prints all the curves instead

4. Option *Show convex ROC curves* refers to convex curves over each individual classifier (the thin lines positioned over curves). *Show ROC convex hull* plots a convex hull combining all classifiers (the gray area below the curves). Plotting both types of convex curves makes sense since selecting a threshold in a concave part of the curve cannot

yield optimal results, disregarding the cost matrix. Besides, it is possible to reach any point on the convex curve by combining the classifiers represented by the points on the border of the concave region.



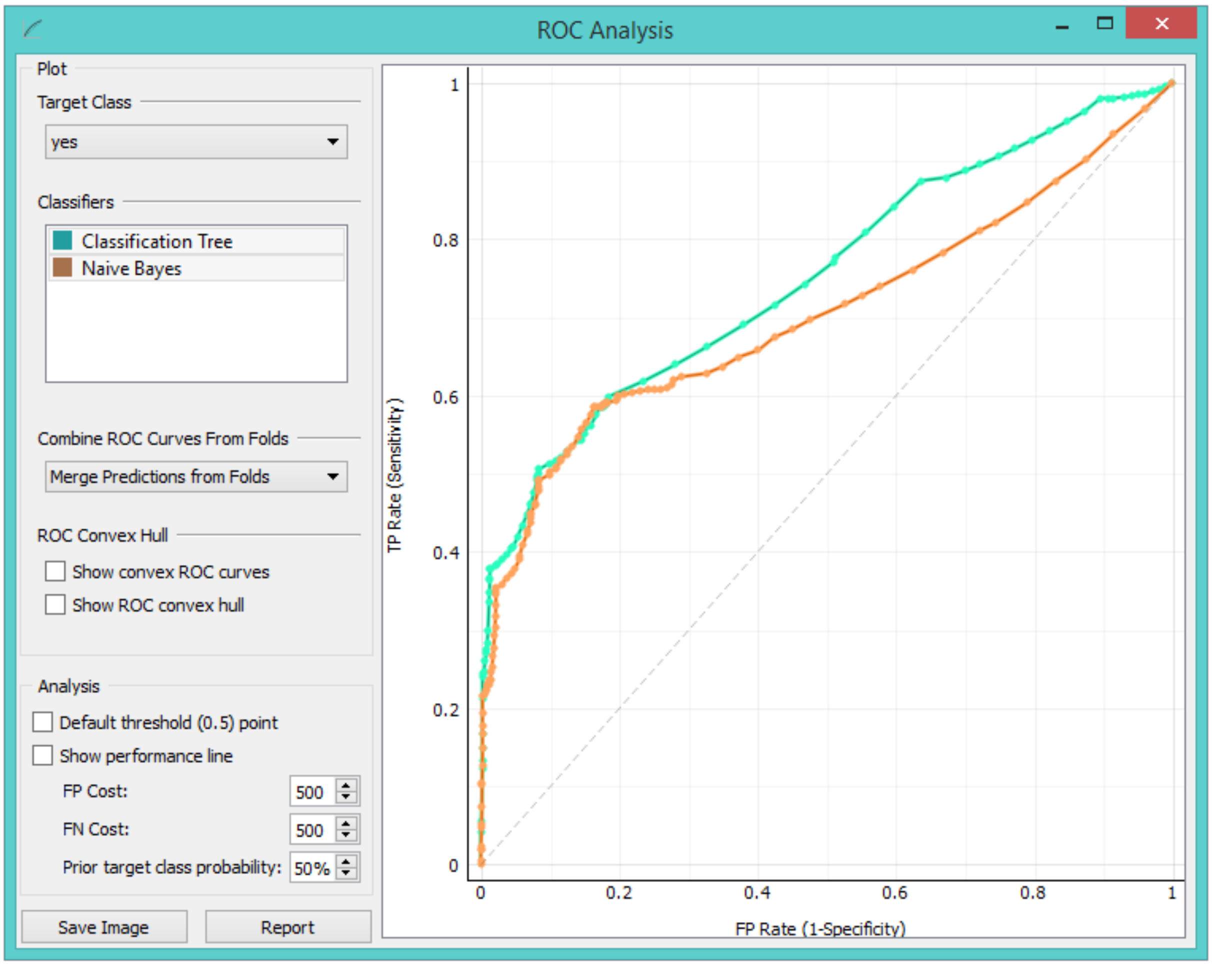
The diagonal dotted line represents the behaviour of a random classifier. The full diagonal line represents iso-performance. A black "*A*" symbol at the bottom of the graph proportionally readjusts the graph.

5. The final box is dedicated to the analysis of the curve. The user can specify the cost of false positives (FP) and false negatives (FN), and the prior target class probability.

   Default threshold (0.5) point shows the point on the ROC curve achieved by the classifier if it predicts the target class if its probability equals or exceeds 0.5.

   Show performance line shows iso-performance in the ROC space so that all the points on the line give the same profit/loss. The line further to the upper left is better than the one down and right. The direction of the line depends upon costs and probabilities. This gives a recipe for depicting the optimal threshold for the given costs: this is the point where the tangent with the given inclination touches the curve and it is marked in the plot. If we push the iso-performance higher or more to the left, the points on the iso-performance line cannot be reached by the learner. Going down or to the right, decreases the performance.

   The widget allows setting the costs from 1 to 1000. Units are not important, as are not the magnitudes. What matters is the relation between the two costs, so setting them to 100 and 200 will give the same result as 400 and 800.



Target Class: yes

Classifiers:
- 🟦 Classification Tree
- 🟫 Naive Bayes

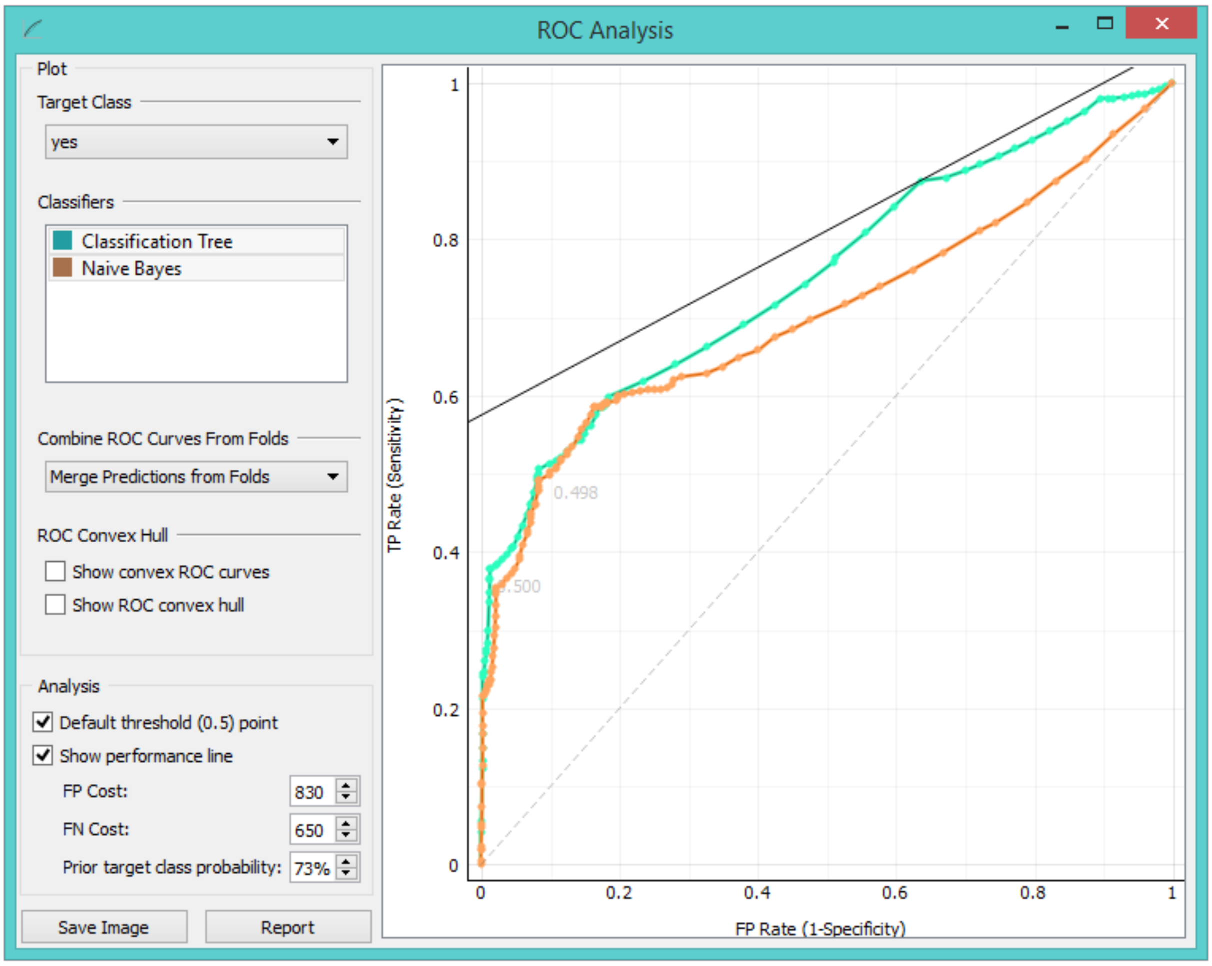
Combine ROC Curves From Folds: Merge Predictions from Folds

- [ ] Show convex ROC curves
- [ ] Show ROC convex hull

- [ ] Default threshold (0.5) point
- [ ] Show performance line

FP Cost: 500
FN Cost: 500
Prior target class probability: 50%

Defaults: both costs equal (500), Prior target class probability 50% (from the data).



Target Class: yes

Classifiers:
- 🟦 Classification Tree
- 🟫 Naive Bayes

Combine ROC Curves From Folds: Merge Predictions from Folds

- [ ] Show convex ROC curves
- [ ] Show ROC convex hull

- [x] Default threshold (0.5) point
- [x] Show performance line

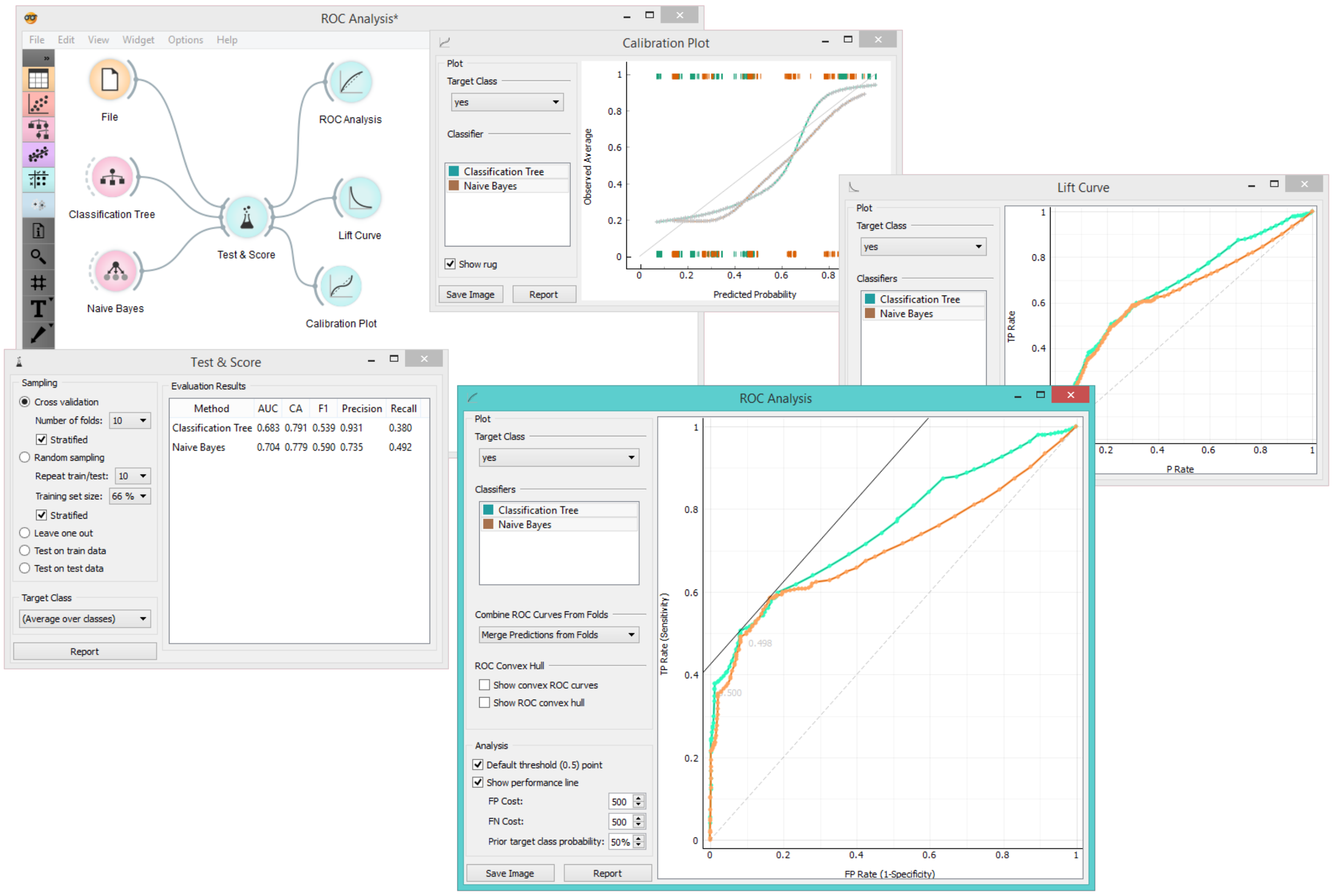
FP Cost: 830
FN Cost: 650
Prior target class probability: 73%

False positive cost: 830, False negative cost 650, Prior target class probability 73%.

6. Press *Save Image* if you want to save the created image to your computer in a .svg or .png format.
7. Produce a report.

Example

At the moment, the only widget which gives the right type of signal needed by the **ROC Analysis** is Test&Score. Below, we compare two classifiers, namely Tree and Naive Bayes, in **Test&Score** and then compare their performance in **ROC Analysis**, Life Curve and Calibration Plot.



Sampling
- [✓ Cross validation]
  - Number of folds: 10
  - ✓ Stratified
- Random sampling
  - Repeat train/test: 10
  - Training set size: 66 %
  - ✓ Stratified
- Leave one out
- Test on train data
- Test on test data

Target Class: (Average over classes)

Report

Evaluation Results

Method	AUC	CA	F1	Precision	Recall
Classification Tree	0.683	0.791	0.539	0.931	0.380
Naive Bayes	0.704	0.779	0.590	0.735	0.492

Test & Score

Tests learning algorithms on data.

Signals

Inputs

- **Data**

  Data for training and, if there is no separate test data set, also testing.

- **Test Data**

  Separate data for testing.

- **Learner**

  One or more learning algorithms.

Outputs

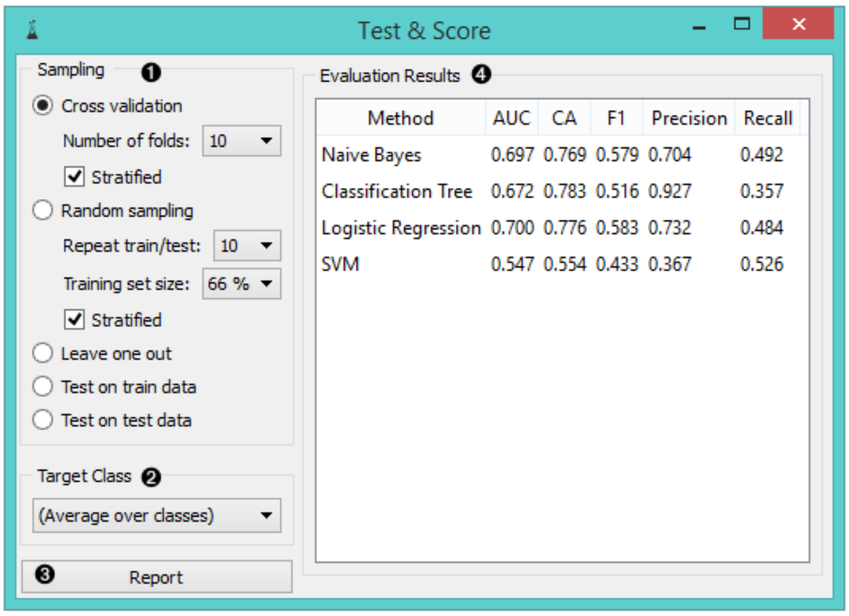
- **Evaluation results**

  Results of testing the algorithms.

Description

The widget tests learning algorithms. Different sampling schemes are available, including using separate test data. The widget does two things. First, it shows a table with different classifier performance measures, such as classification accuracy and area under the curve. Second, it outputs evaluation results, which can be used by other widgets for analyzing the performance of classifiers, such as ROC Analysis or Confusion Matrix.

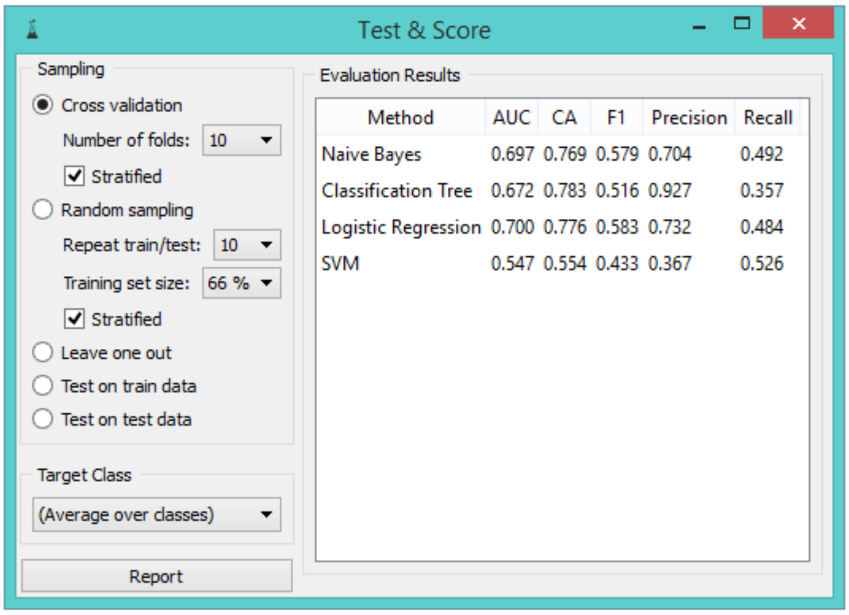
The *Learner* signal has an uncommon property: it can be connected to more than one widget to test multiple learners with the same procedures.



Method	AUC	CA	F1	Precision	Recall
Naive Bayes	0.697	0.769	0.579	0.704	0.492
Classification Tree	0.672	0.783	0.516	0.927	0.357
Logistic Regression	0.700	0.776	0.583	0.732	0.484
SVM	0.547	0.554	0.433	0.367	0.526

1. The widget supports various sampling methods.
   - Cross-validation splits the data into a given number of folds (usually 5 or 10). The algorithm is tested by holding out examples from one fold at a time; the model is induced from other folds and examples from the held out fold are classified. This is repeated for all the folds.
   - **Leave-one-out** is similar, but it holds out one instance at a time, inducing the model from all others and then classifying the held out instances. This method is obviously very stable, reliable ... and very slow.
   - **Random sampling** randomly splits the data into the training and testing set in the given proportion (e.g. 70:30); the whole procedure is repeated for a specified number of times.
   - **Test on train data** uses the whole data set for training and then for testing. This method practically always gives wrong results.
   - **Test on test data**: the above methods use the data from *Data* signal only. To input another data set with testing examples (for instance from another file or some data selected in another widget), we select *Separate Test Data* signal in the communication channel and select Test on test data.
2. Only *Test on test data* requires a target class, e.g. having the disease or being of subvariety *Iris setosa*. When *Target class* is (None), the methods return the average value. Target class can be selected at the bottom of the widget.
3. Produce a report.
4. The widget will compute a number of performance statistics:

Classification

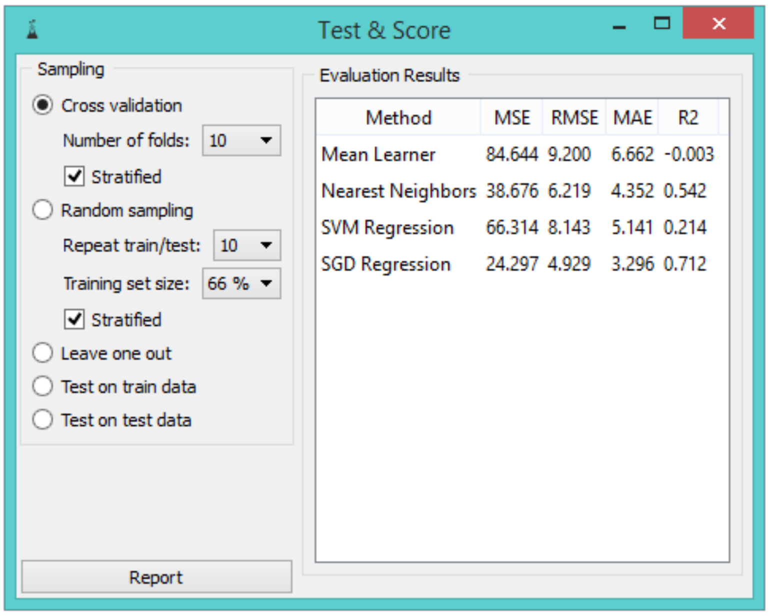


Number of folds: 10
Stratified ✓
Repeat train/test: 10
Training set size: 66 %
Stratified ✓
Target Class: (Average over classes)

Method	AUC	CA	F1	Precision	Recall
Naive Bayes	0.697	0.769	0.579	0.704	0.492
Classification Tree	0.672	0.783	0.516	0.927	0.357
Logistic Regression	0.700	0.776	0.583	0.732	0.484
SVM	0.547	0.554	0.433	0.367	0.526

- Area under ROC is the area under the receiver-operating curve.
- Classification accuracy is the proportion of correctly classified examples.
- F-1 is a weighted harmonic mean of precision and recall (see below).
- Precision is the proportion of true positives among instances classified as positive, e.g. the proportion of *Iris virginica* correctly identified as Iris virginica.
- Recall is the proportion of true positives among all positive instances in the data, e.g. the number of sick among all diagnosed as sick.

Regression



Cross validation [✓]
Number of folds: 10
Stratified [✓]
Random sampling
Repeat train/test: 10
Training set size: 66 %
Stratified [✓]
Leave one out
Test on train data
Test on test data

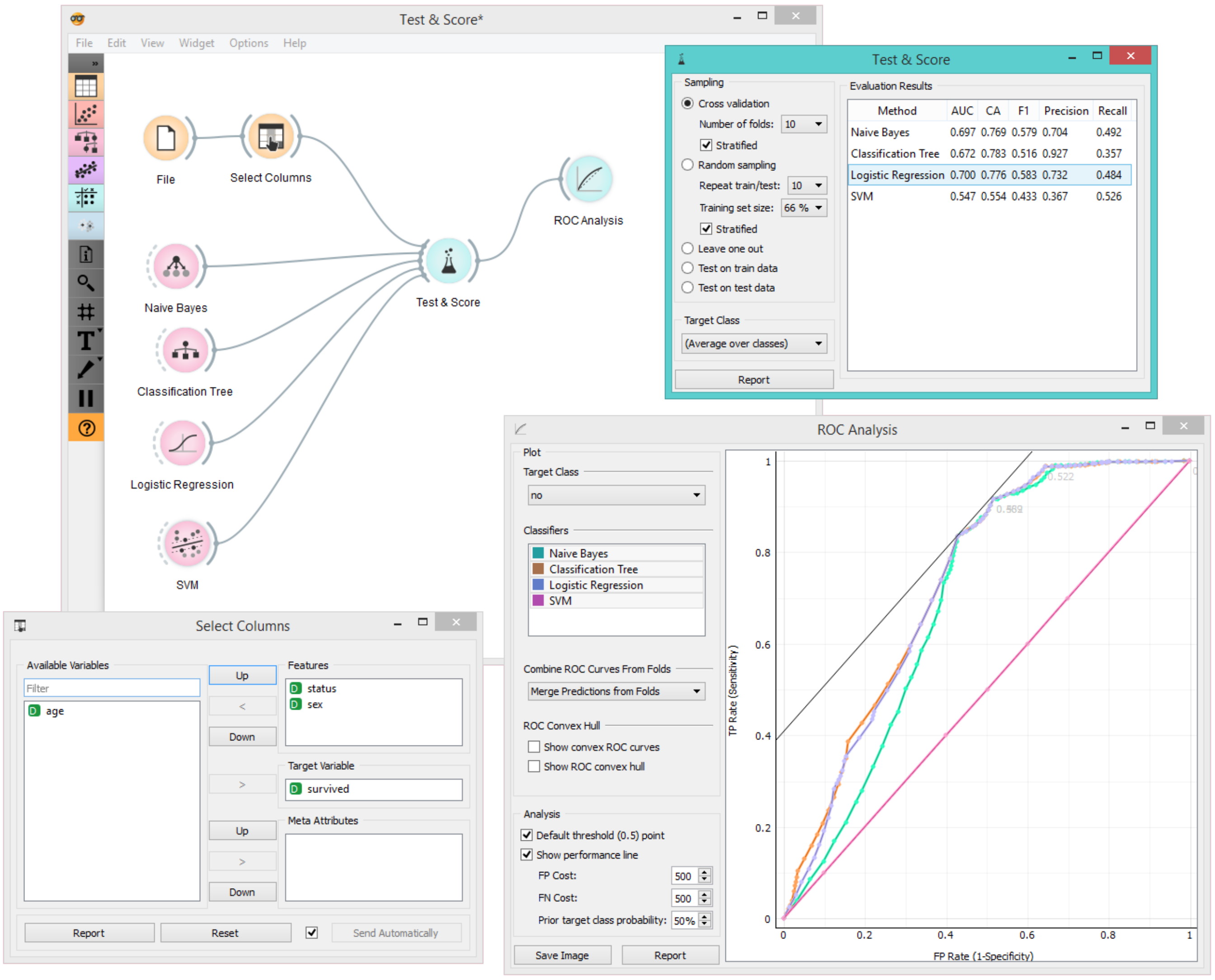
Evaluation Results

Method	MSE	RMSE	MAE	R2
Mean Learner	84.644	9.200	6.662	-0.003
Nearest Neighbors	38.676	6.219	4.352	0.542
SVM Regression	66.314	8.143	5.141	0.214
SGD Regression	24.297	4.929	3.296	0.712

- MSE measures the average of the squares of the errors or deviations (the difference between the estimator and what is estimated).
- RMSE is the square root of the arithmetic mean of the squares of a set of numbers (a measure of imperfection of the fit of the estimator to the data)
- MAE is used to measure how close forecasts or predictions are to eventual outcomes.
- R2 is interpreted as the proportion of the variance in the dependent variable that is predictable from the independent variable.

Example

In a typical use of the widget, we give it a data set and a few learning algorithms and we observe their performance in the table inside the **Test & Score** widget and in the ROC. The data is often preprocessed before testing; in this case we did some manual feature selection (Select Columns widget) on *Titanic* data set, where we want to know only the sex and status of the survived and omit the age.



Test & Score

Sampling
- [✓ Cross validation]
  - Number of folds: 10
  - ☑ Stratified
- Random sampling
  - Repeat train/test: 10
  - Training set size: 66 %
  - ☑ Stratified
- Leave one out
- Test on train data
- Test on test data

Target Class: (Average over classes)

Report

Evaluation Results

Method	AUC	CA	F1	Precision	Recall
Naive Bayes	0.697	0.769	0.579	0.704	0.492
Classification Tree	0.672	0.783	0.516	0.927	0.357
Logistic Regression	0.700	0.776	0.583	0.732	0.484
SVM	0.547	0.554	0.433	0.367	0.526

Select Columns

Available Variables
Filter
- age

Features
- status
- sex

Target Variable
- survived

Meta Attributes

Up < Down > Up > Down

Report Reset ☑ Send Automatically

ROC Analysis

Plot
Target Class: no

Classifiers
- 🟦 Naive Bayes
- 🟫 Classification Tree
- 🟦 Logistic Regression
- 🟪 SVM

Combine ROC Curves From Folds: Merge Predictions from Folds

ROC Convex Hull
- ☐ Show convex ROC curves
- ☐ Show ROC convex hull

Analysis
- ☑ Default threshold (0.5) point
- ☑ Show performance line
- FP Cost: 500
- FN Cost: 500
- Prior target class probability: 50%

Another example of using this widget is presented in the documentation for the Confusion Matrix widget.

Data Fusion

IMDb Actors

Chaining

Completion Scoring

Fusion Graph

Latent Factors

Matrix Sampler

Mean Fuser

Movie Genres

Movie Ratings

Table to Relation

Chaining

Profiles objects of one type in the latent space of another object type through chaining of latent matrices along paths in a data fusion graph.

Signals

Inputs:

- **Fitted Fusion Graph**

  Fitted collective latent data model.

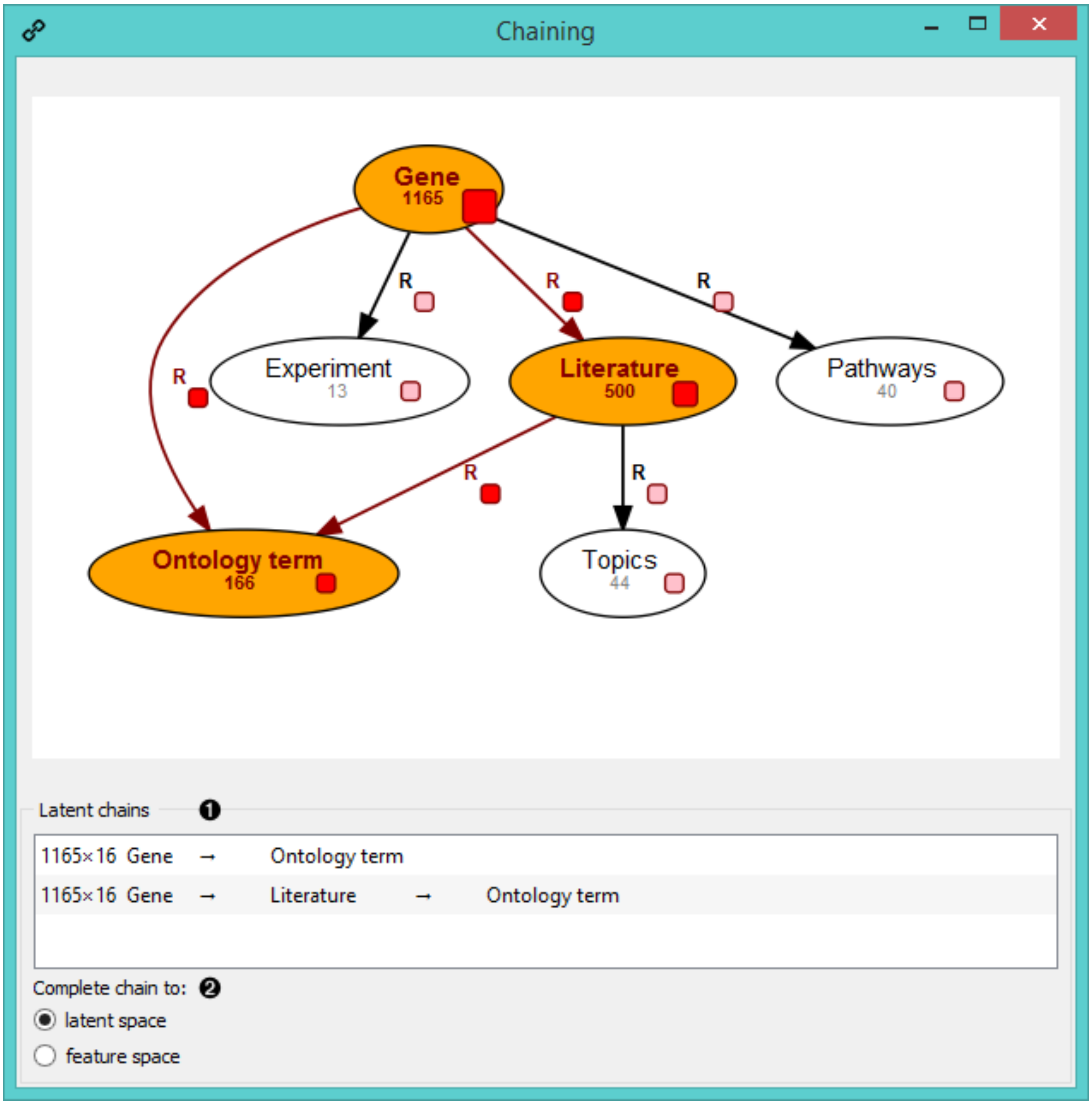
Outputs:

- **Relation**

  Relationships between two groups of objects.

Description

Chaining constructs data profiles of objects of one type that are expressed in the latent space of another object type. This is done by appropriately multiplying the latent matrices along paths that connect start and end nodes in the fusion graph. The widget displays a fitted fusion graph on the right, where you can select the start and end node (object type) that are then used in chaining.



Latent chains ❶

1165×16 Gene	→	Ontology term		
1165×16 Gene	→	Literature	→	Ontology term

Complete chain to: ❷

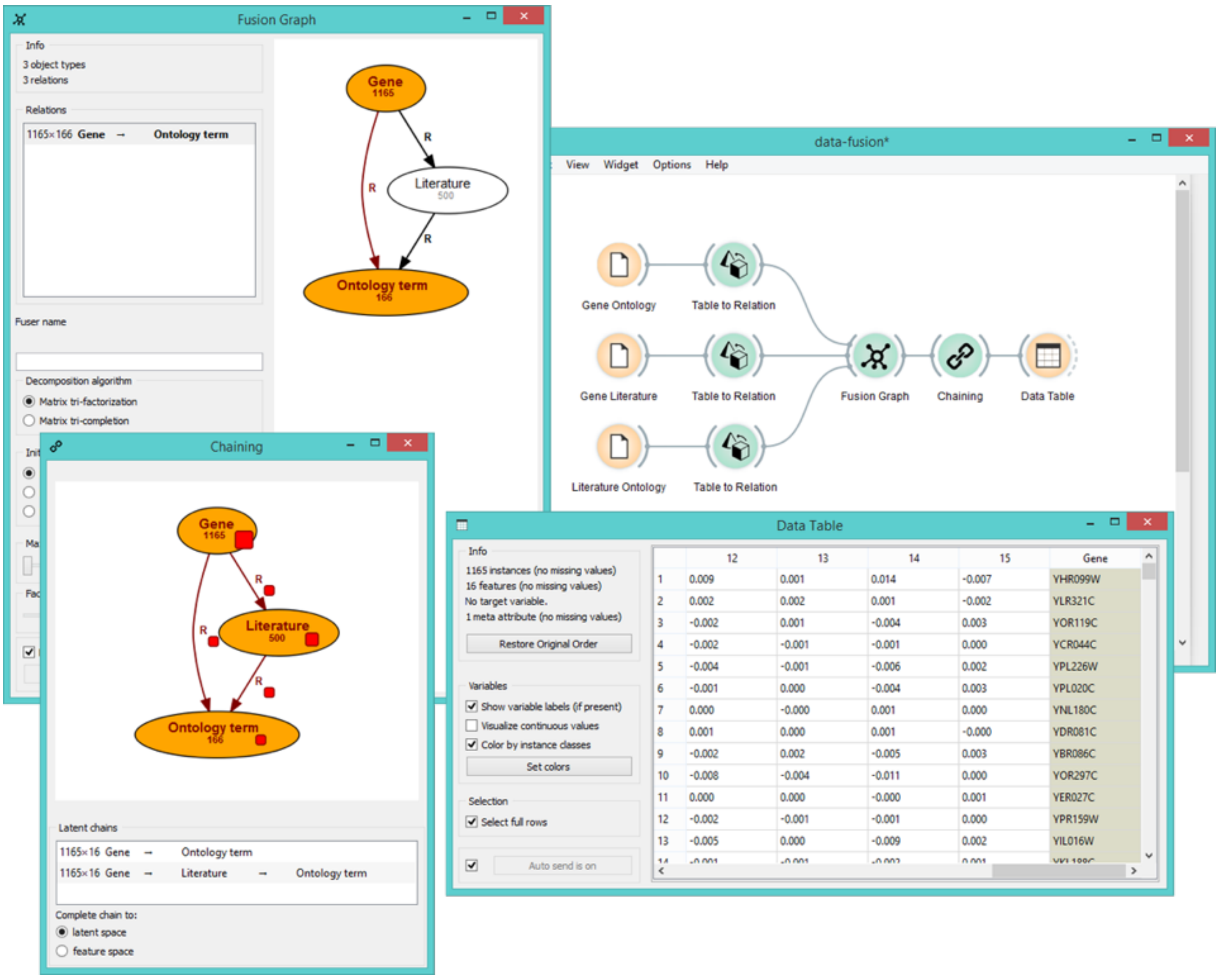
(●) latent space

( ) feature space

1. The widget displays all chains that connect selected start node with the selected end node (in orange). Click on the chain you wish to output.
2. Select what type of chain you wish to output:
   - **latent space** (widget outputs data profiles in the latent space)
   - **feature space** (widget outputs data profiles in the original domain space)

Example

This widget is great for constructing profiles that relate objects, which are not directly connected in a fusion graph. In the example below we have three data sets: annotations of genes from the Gene Ontology, literature on genes and literature on ontology terms. We use **Chaining** to see how genes relate to ontology terms.



Info
3 object types
3 relations

Relations
1165×166 **Gene** → **Ontology term**

Fuser name

Decomposition algorithm
- [✓] Matrix tri-factorization
- Matrix tri-completion

Latent chains
1165×16 Gene → Ontology term
1165×16 Gene → Literature → Ontology term

Complete chain to:
- [✓] latent space
- feature space

Info
1165 instances (no missing values)
16 features (no missing values)
No target variable.
1 meta attribute (no missing values)

Restore Original Order

Variables
- [✓] Show variable labels (if present)
- Visualize continuous values
- [✓] Color by instance classes

Set colors

Selection
- [✓] Select full rows

[✓] Auto send is on

	12	13	14	15	Gene
1	0.009	0.001	0.014	-0.007	YHR099W
2	0.002	0.002	0.001	-0.002	YLR321C
3	-0.002	0.001	-0.004	0.003	YOR119C
4	-0.002	-0.001	-0.001	0.000	YCR044C
5	-0.004	-0.001	-0.006	0.002	YPL226W
6	-0.001	0.000	-0.004	0.003	YPL020C
7	0.000	-0.000	0.001	0.000	YNL180C
8	0.001	0.000	0.001	-0.000	YDR081C
9	-0.002	0.002	-0.005	0.003	YBR086C
10	-0.008	-0.004	-0.011	0.000	YOR297C
11	0.000	0.000	-0.000	0.001	YER027C
12	-0.002	-0.001	-0.001	0.000	YPR159W
13	-0.005	0.000	-0.009	0.002	YIL016W

Completion Scoring

Scores the quality of matrix completion using root mean squared error (RSME) metric.

Signals

Inputs:

- **Fitted fusion graph**

  Fitted collective latent data model.

- **Relation**

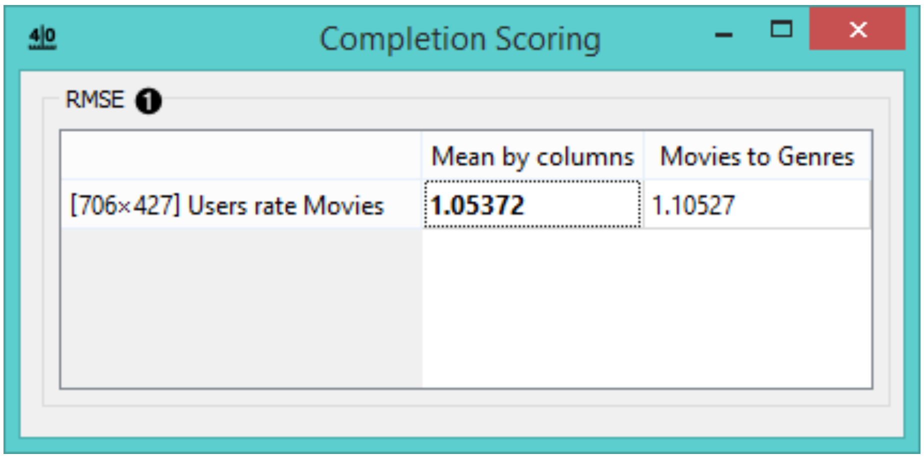
  Relationships between two groups of objects.

Outputs:

- (None)

Description

This widget assesses the quality of matrix completion based on root mean squared error metric (RMSE). Each row contains scores representing matrix completion quality of different relations. Results for prediction models are in columns.

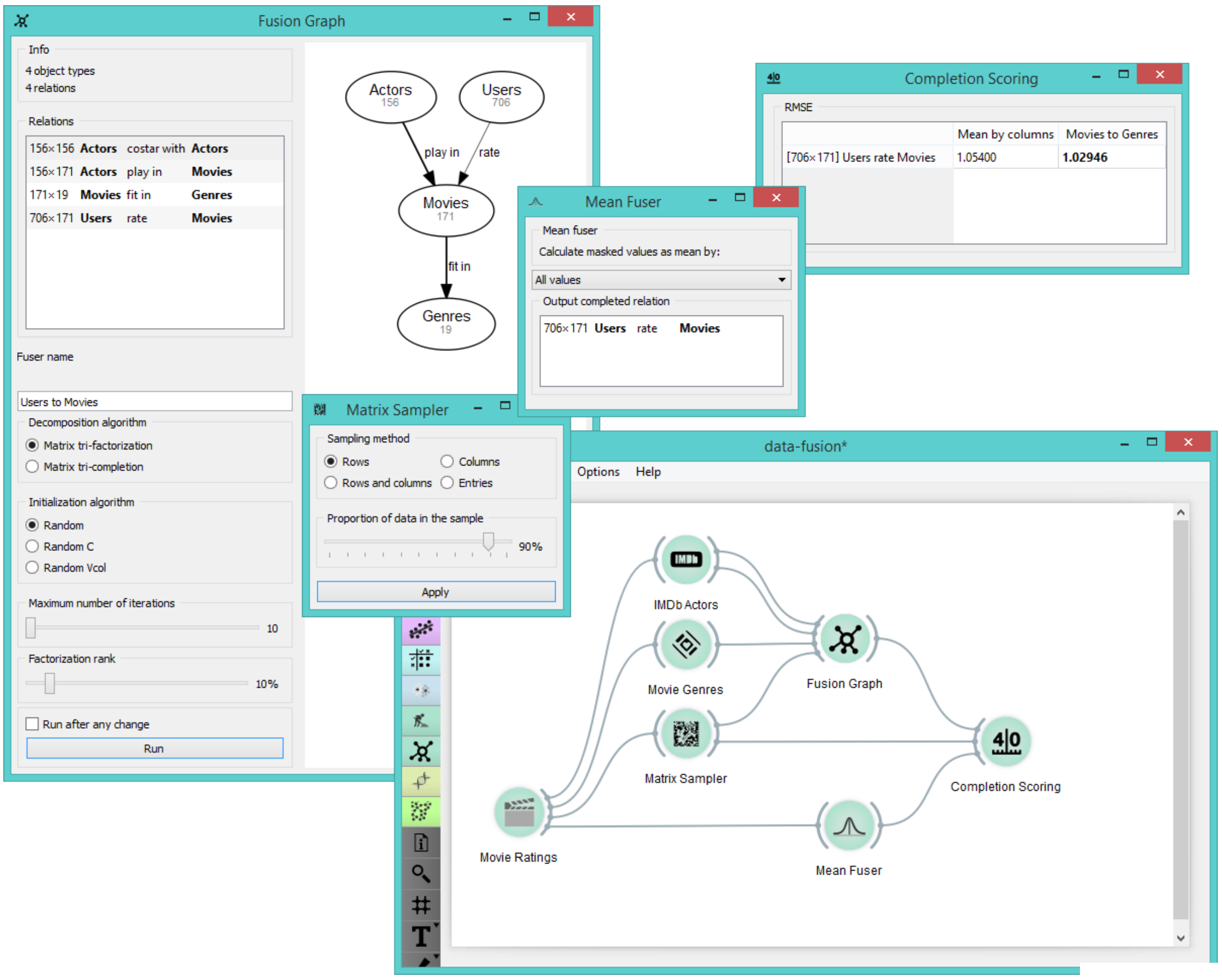


	Mean by columns	Movies to Genres
[706×427] Users rate Movies	1.05372	1.10527

1. The RMSE value chart for the input relation matrix.

Example

Completion Scoring widget assesses the quality of matrix completion using the RMSE metric. Connect it with **Matrix Sampler** to score prediction models (previously learnt on in-sample data) on out-of-the-sample data. You can also use **Mean Fuser** to get a mean score for latent values.



v: latest

Fusion Graph

Constructs a data fusion graph and runs collective matrix factorization algorithm.

Signals

Inputs:

- **Relation**

  Relationships between two groups of objects.

Outputs:

- **Relation**

  Relationships between two groups of objects.

- **Fitted Fusion Graph**

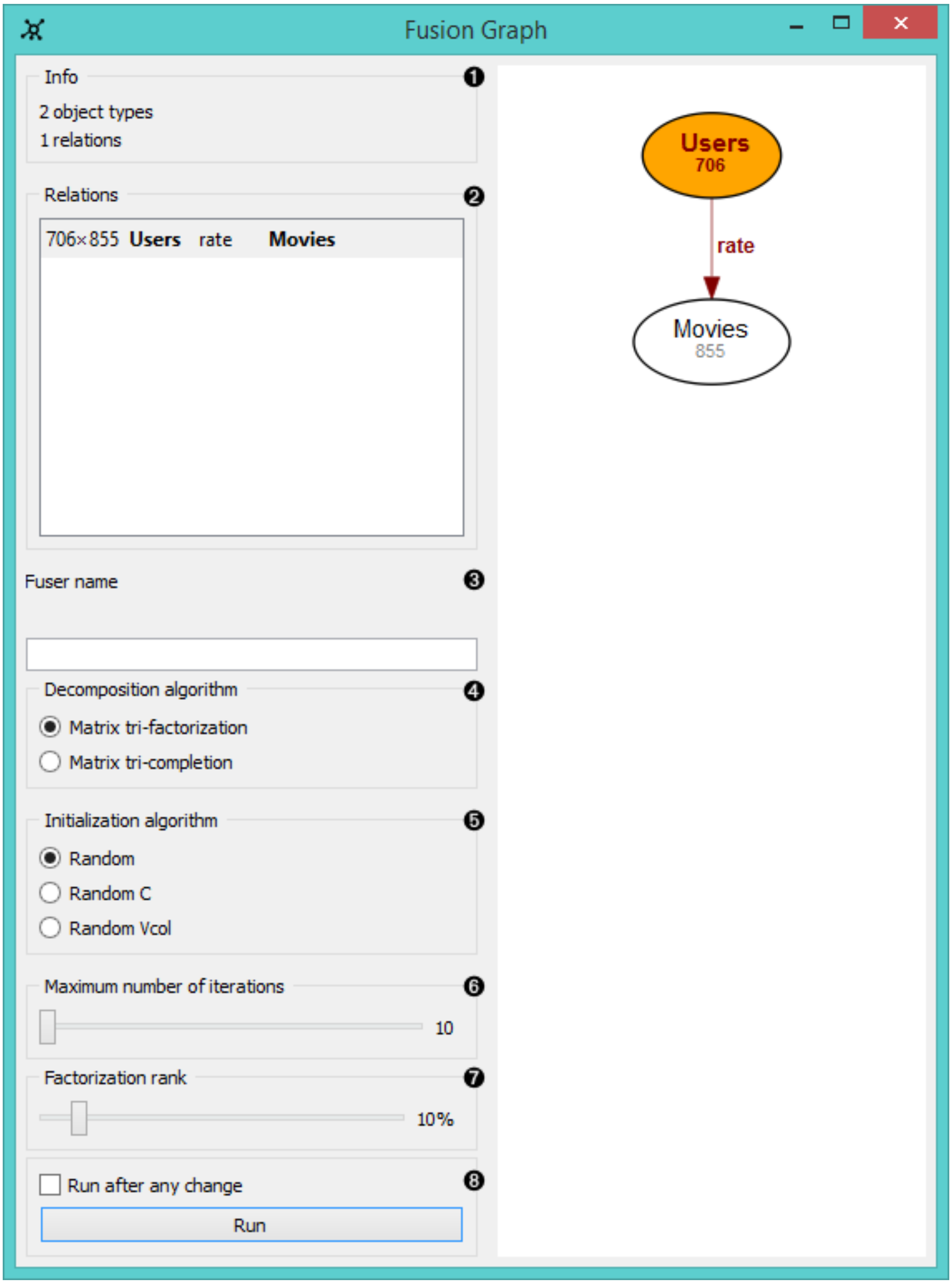
  Fitted collective latent data model.

- **Fusion Graph**

  Input data system.

Description

Fusion Graph widget performs data fusion by collective matrix factorization. It fuses multiple related data sets into one comprehensive structure. The widget returns a relational structure of the entire data system estimated by a collective latent factor approach.


Info
2 object types
1 relations

Relations
706×855 **Users** rate **Movies**

Fuser name

Decomposition algorithm
(●) Matrix tri-factorization
( ) Matrix tri-completion

Initialization algorithm
(●) Random
( ) Random C
( ) Random Vcol

Maximum number of iterations
10

Factorization rank
10%

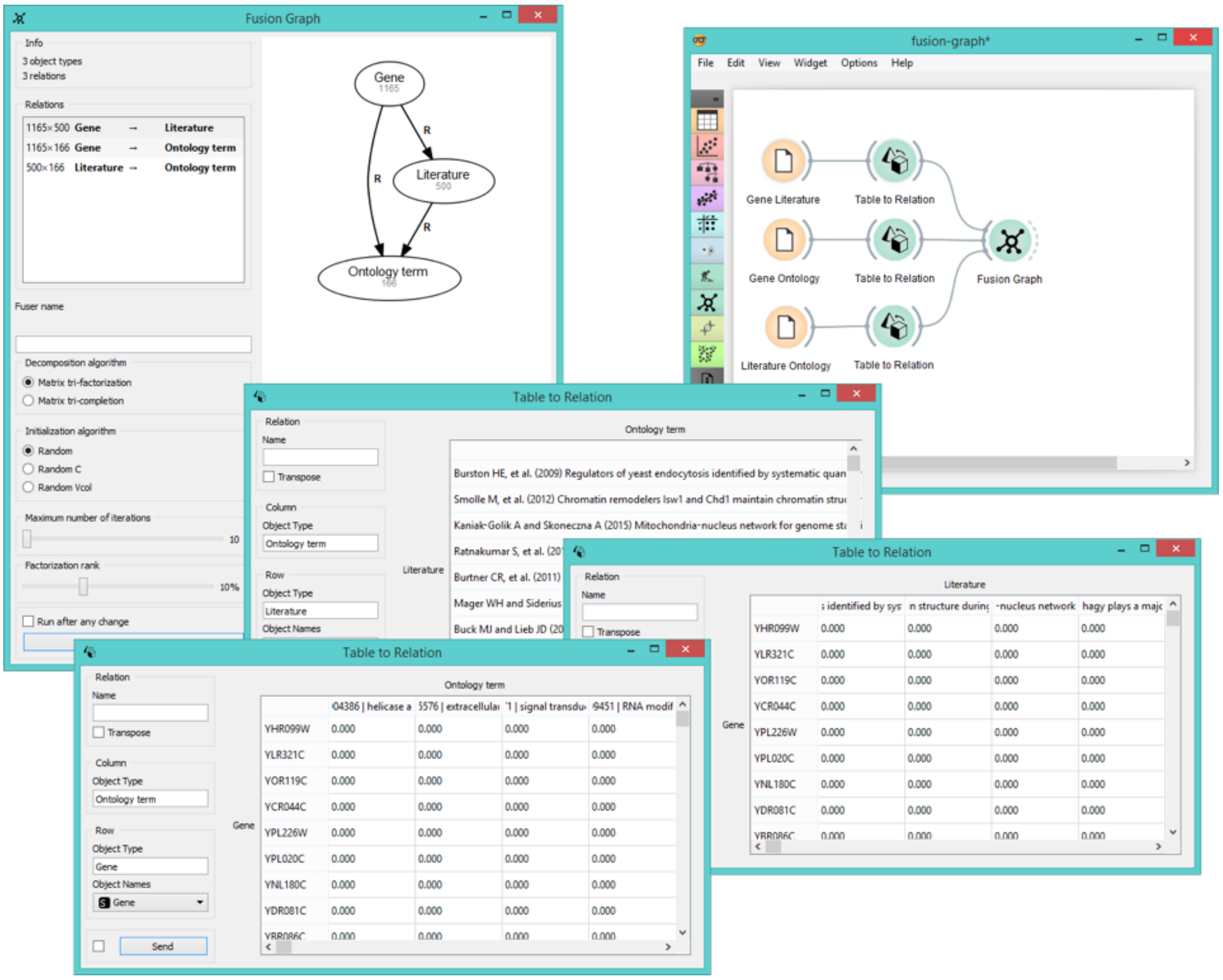
☐ Run after any change

1. Information on the input (object types are nodes, relations are links between the nodes).
2. List of identified relations. Click on the relation to output it.
3. Specify a descriptive name for your fusion system.
4. Select the algorithm for factorization:
   - **matrix tri-factorization** decomposes each relation matrix into three latent matrices and shares the latent matrices between related data sets. Unknown values are imputed prior to collective factorization.
   - **matrix tri-completion** works the same as matrix tri-factorization, but does not require re-

lation matrices to be fully observed.
5. Select the *initialization algorithm* for matrix factorization.
6. Set the *maximum number of iterations* used for factorization. Default is 10.
7. Set the *factorization rank* (the ratio of data compression based on the input data). Default is 10%.
8. If *Run after every change* is ticked, the widget will automatically commit changes. Alternatively press *Run*. For large data sets we recommend to commit the changes manually.

Example

The example below shows how to fuse several data sets together. Say we have the data on ontology terms for many genes, literature on ontology terms and literature on genes. To fuse these data together we first use **Table to Relation** widget, where we manually set the object type and relation names. **Fusion Graph** will compile the fusion graph of our three data sets with connections between object types based on previously defined data relations, display the connections and run matrix decomposition algorithm.



Fusion Graph

Info
3 object types
3 relations

Relations

1165×500	**Gene**	→	**Literature**
1165×166	**Gene**	→	**Ontology term**
500×166	**Literature**	→	**Ontology term**

Fuser name

Decomposition algorithm
[✓ Matrix tri-factorization]
Matrix tri-completion

Initialization algorithm
[✓ Random]
Random C
Random Vcol

Maximum number of iterations 10

Factorization rank 10%

Run after any change

Table to Relation

Relation
Name
Transpose

Column
Object Type: Ontology term

Row
Object Type: Literature
Object Names

Ontology term

Literature

Burston HE, et al. (2009) Regulators of yeast endocytosis identified by systematic quan
Smolle M, et al. (2012) Chromatin remodelers Isw1 and Chd1 maintain chromatin struc
Kaniak-Golik A and Skoneczna A (2015) Mitochondria-nucleus network for genome sta
Ratnakumar S, et al. (20
Burtner CR, et al. (2011)
Mager WH and Siderius
Buck MJ and Lieb JD (20

Table to Relation

Relation
Name
Transpose

Literature

Gene

	; identified by sys	n structure during	-nucleus network	hagy plays a majc
YHR099W	0.000	0.000	0.000	0.000
YLR321C	0.000	0.000	0.000	0.000
YOR119C	0.000	0.000	0.000	0.000
YCR044C	0.000	0.000	0.000	0.000
YPL226W	0.000	0.000	0.000	0.000
YPL020C	0.000	0.000	0.000	0.000
YNL180C	0.000	0.000	0.000	0.000
YDR081C	0.000	0.000	0.000	0.000
YBR086C	0.000	0.000	0.000	0.000

Table to Relation

Relation
Name
Transpose

Column
Object Type: Ontology term

Row
Object Type: Gene
Object Names: Gene

Send

Ontology term

Gene

	04386 | helicase a	5576 | extracellular	'1 | signal transdu	9451 | RNA modif
YHR099W	0.000	0.000	0.000	0.000
YLR321C	0.000	0.000	0.000	0.000
YOR119C	0.000	0.000	0.000	0.000
YCR044C	0.000	0.000	0.000	0.000
YPL226W	0.000	0.000	0.000	0.000
YPL020C	0.000	0.000	0.000	0.000
YNL180C	0.000	0.000	0.000	0.000
YDR081C	0.000	0.000	0.000	0.000
YBR086C	0.000	0.000	0.000	0.000

IMDb Actors

Constructs a movies-by-actors or actors-by-actors relation matrix.

Signals

Inputs:

- **Filter**

  Data filter.

Outputs:

- **Movie Actors**

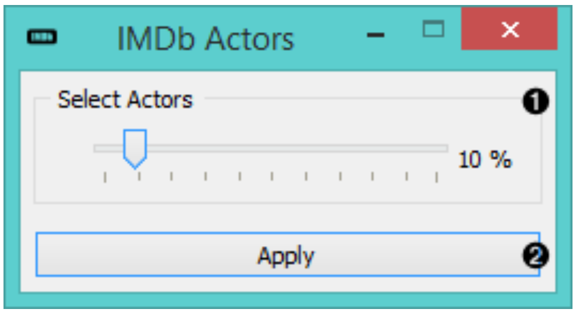
  A movies-by-actors relation matrix.

- **Costarring Actors**

  An actors-by-actors relation matrix.

Description

This widget gives you the access to the IMDb data sets on actors and movies. It outputs either a movies-by-actors relation matrix, an actors-by-actors relation matrix or both.

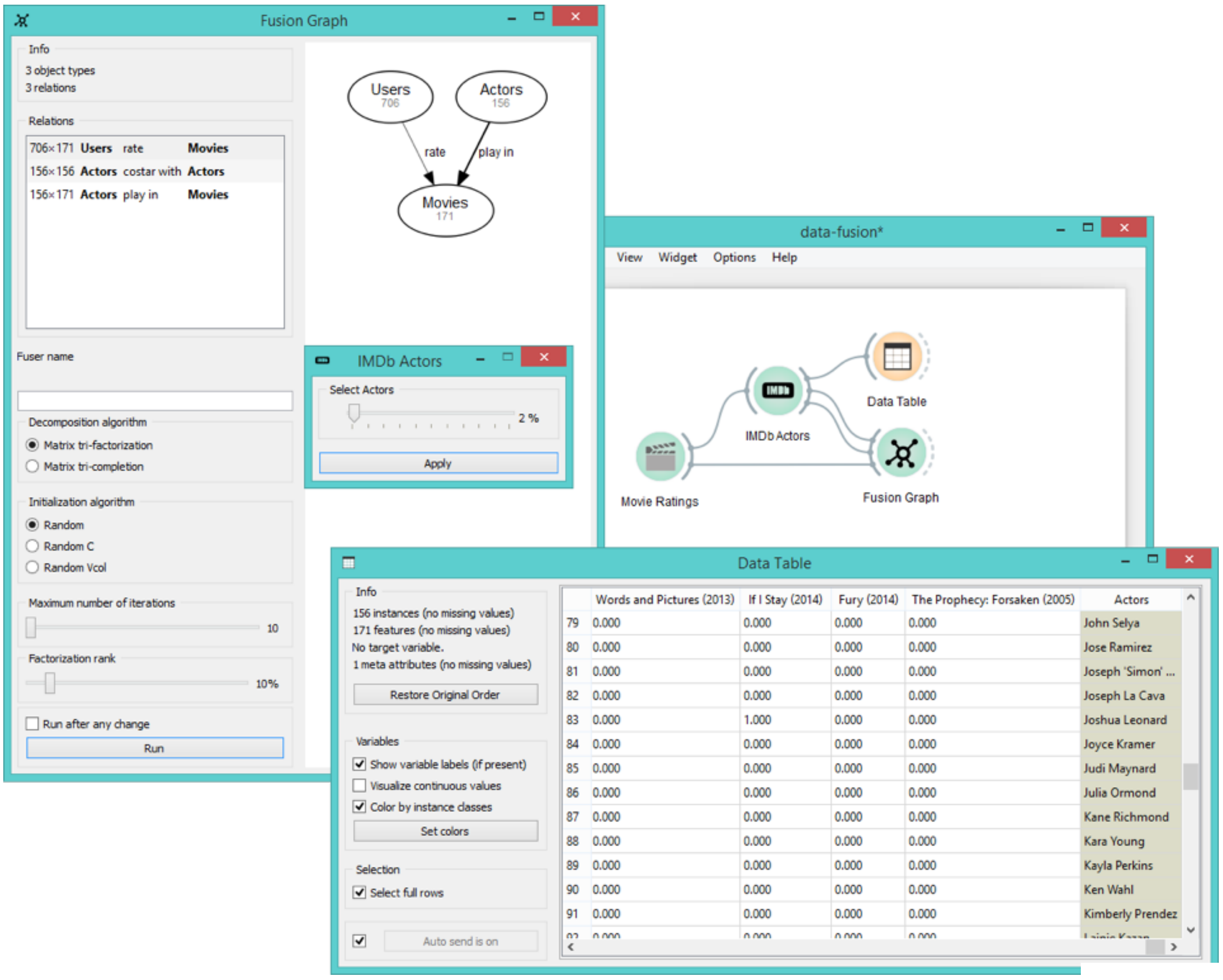


1. Select how many actors from the IMDb database would you like to consider.
2. Click *Apply* to commit your data.

Example

This simple widget is great for learning how data fusion works since it enables immediate access to the

IMDb database. To use it, you need to connect it to **Movie Ratings** widget in the input and with **Fusion Graph** in the output. This will add the information on actors in relation to movies. You can view this new data in the **Data Table** widget.



Info
3 object types
3 relations

Relations

706×171	**Users**	rate	**Movies**
156×156	**Actors**	costar with	**Actors**
156×171	**Actors**	play in	**Movies**

Fuser name

Decomposition algorithm
[✓ Matrix tri-factorization] Matrix tri-completion

Initialization algorithm
[✓ Random] Random C Random Vcol

Maximum number of iterations 10

Factorization rank 10%

☐ Run after any change

Run

Info
156 instances (no missing values)
171 features (no missing values)
No target variable.
1 meta attributes (no missing values)

Variables
☑ Show variable labels (if present)
☐ Visualize continuous values
☑ Color by instance classes

Selection
☑ Select full rows

	Words and Pictures (2013)	If I Stay (2014)	Fury (2014)	The Prophecy: Forsaken (2005)	Actors
79	0.000	0.000	0.000	0.000	John Selya
80	0.000	0.000	0.000	0.000	Jose Ramirez
81	0.000	0.000	0.000	0.000	Joseph 'Simon' ...
82	0.000	0.000	0.000	0.000	Joseph La Cava
83	0.000	1.000	0.000	0.000	Joshua Leonard
84	0.000	0.000	0.000	0.000	Joyce Kramer
85	0.000	0.000	0.000	0.000	Judi Maynard
86	0.000	0.000	0.000	0.000	Julia Ormond
87	0.000	0.000	0.000	0.000	Kane Richmond
88	0.000	0.000	0.000	0.000	Kara Young
89	0.000	0.000	0.000	0.000	Kayla Perkins
90	0.000	0.000	0.000	0.000	Ken Wahl
91	0.000	0.000	0.000	0.000	Kimberly Prendez

Latent Factors

Draws data fusion graph with the estimated latent factors overlaid. Outputs latent factors for further analysis.

Signals

Inputs:

- **Fitted fusion graph**

  Fitted collective latent data model.

Outputs:

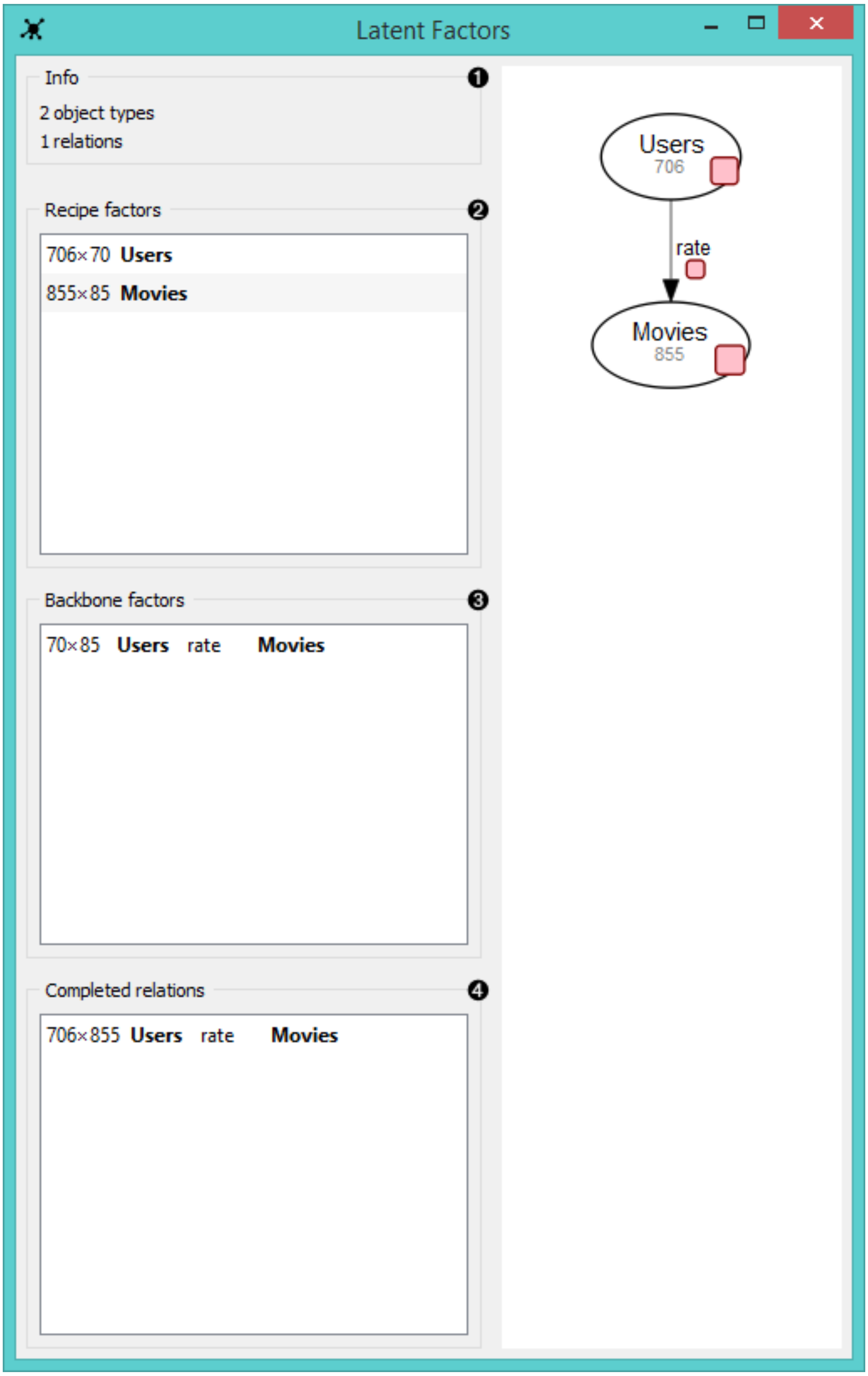
- **Relation**

  Selected latent data matrix or a completed relation.

Description

Latent Factors widget displays the fusion graph together with the backbone and recipe matrices estimated by collective matrix factorization.

Fused data from the widget input are decomposed into latent factors, which serve as components for subsequent matrix reconstruction. You would normally draw this widget from **Fusion Graph** and feed its output (a backbone matrix, a recipe matrix or a completed relation) into widgets for downstream data analysis, such as **Hierarchial Clustering** or **Heat Map**.


Latent Factors

Info
2 object types
1 relations

Recipe factors
706×70 **Users**
855×85 **Movies**

Backbone factors
70×85 **Users** rate **Movies**

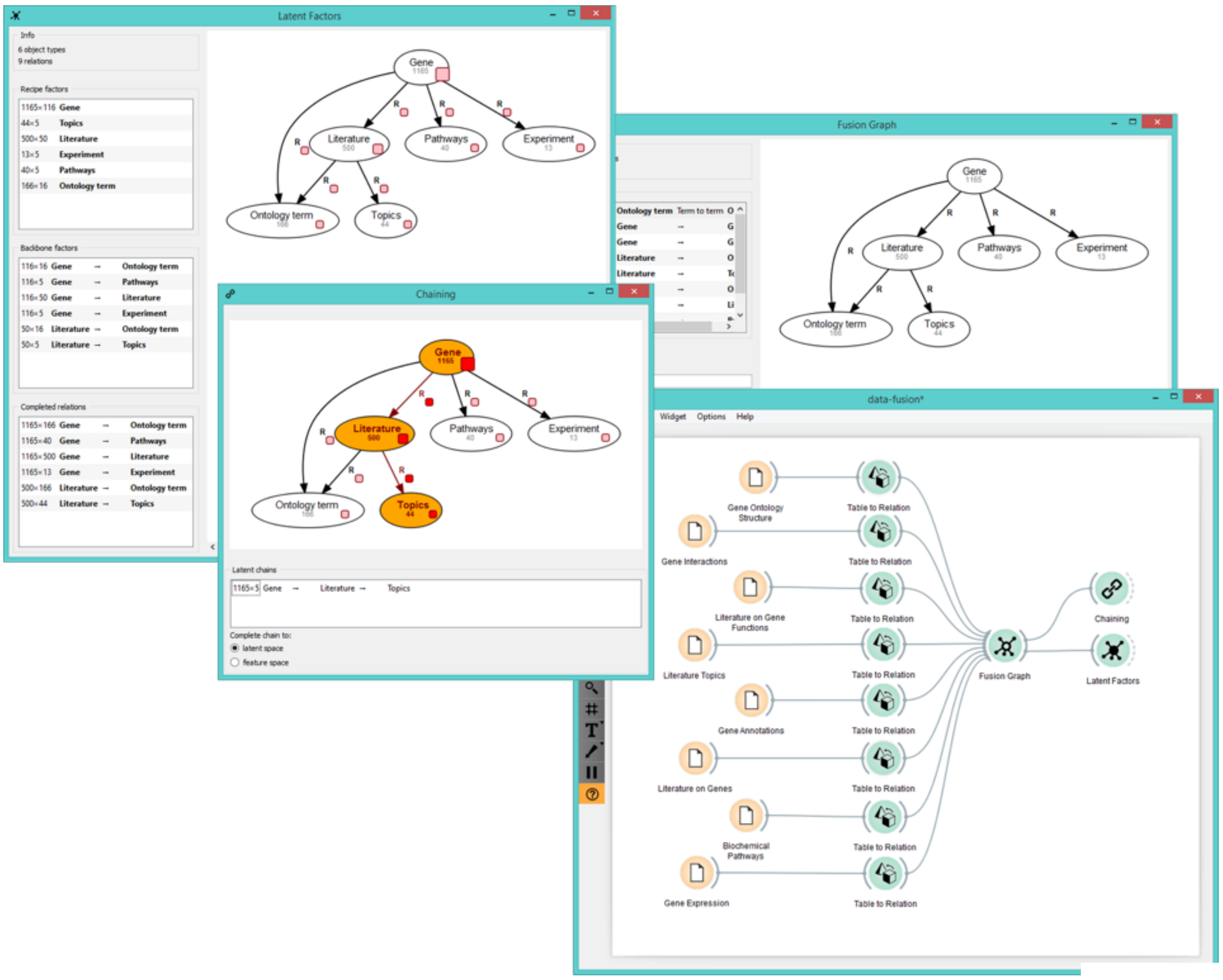
Completed relations
706×855 **Users** rate **Movies**

1. Information on the input (object types are nodes, data relations are links between the nodes).
2. A list of **recipe factors** (latent matrices containing compressed representation of object types). Recipe factors encode latent components of respective object types.
3. A list of **backbone factors** (latent matrices containing compressed representation of data relations). Backbone factors encode interactions between the latent components.
4. A list of **completed relations** (completed relation matrices obtained by multiplying the corre-

sponding latent matrices).

Example

In the example below we demonstrate how 8 separate yeast data sets are fused together in **Fusion Graph** and then decomposed into latent factors with **Latent Factors** widget.



Info
6 object types
9 relations

Recipe factors

1165×116	Gene
44×5	Topics
500×50	Literature
13×5	Experiment
40×5	Pathways
166×16	Ontology term

Backbone factors

116×16	Gene	→	Ontology term
116×5	Gene	→	Pathways
116×50	Gene	→	Literature
116×5	Gene	→	Experiment
50×16	Literature	→	Ontology term
50×5	Literature	→	Topics

Completed relations

1165×166	Gene	→	Ontology term
1165×40	Gene	→	Pathways
1165×500	Gene	→	Literature
1165×13	Gene	→	Experiment
500×166	Literature	→	Ontology term
500×44	Literature	→	Topics

Latent chains

1165×5 Gene → Literature → Topics

Complete chain to:
[✓ latent space]
feature space

v: latest

Matrix Sampler

Samples a relation matrix.

Signals

Inputs:

- **Data**

Data set.

Outputs:

- **In-sample Data**

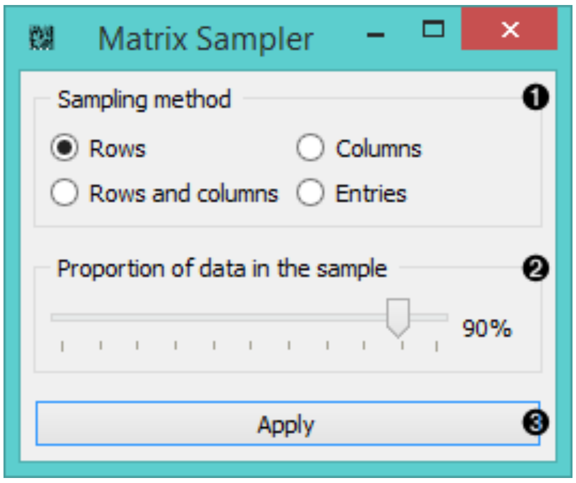
  Selected data.

- **Out-of-the-sample Data**

  Remaining data.

Description

This widget samples the input data and sends both the sampled and the remaining data to the output. It is useful for evaluating the performance of recommendation systems.


Matrix Sampler

Sampling method ❶
(● Rows) ○ Columns
○ Rows and columns ○ Entries

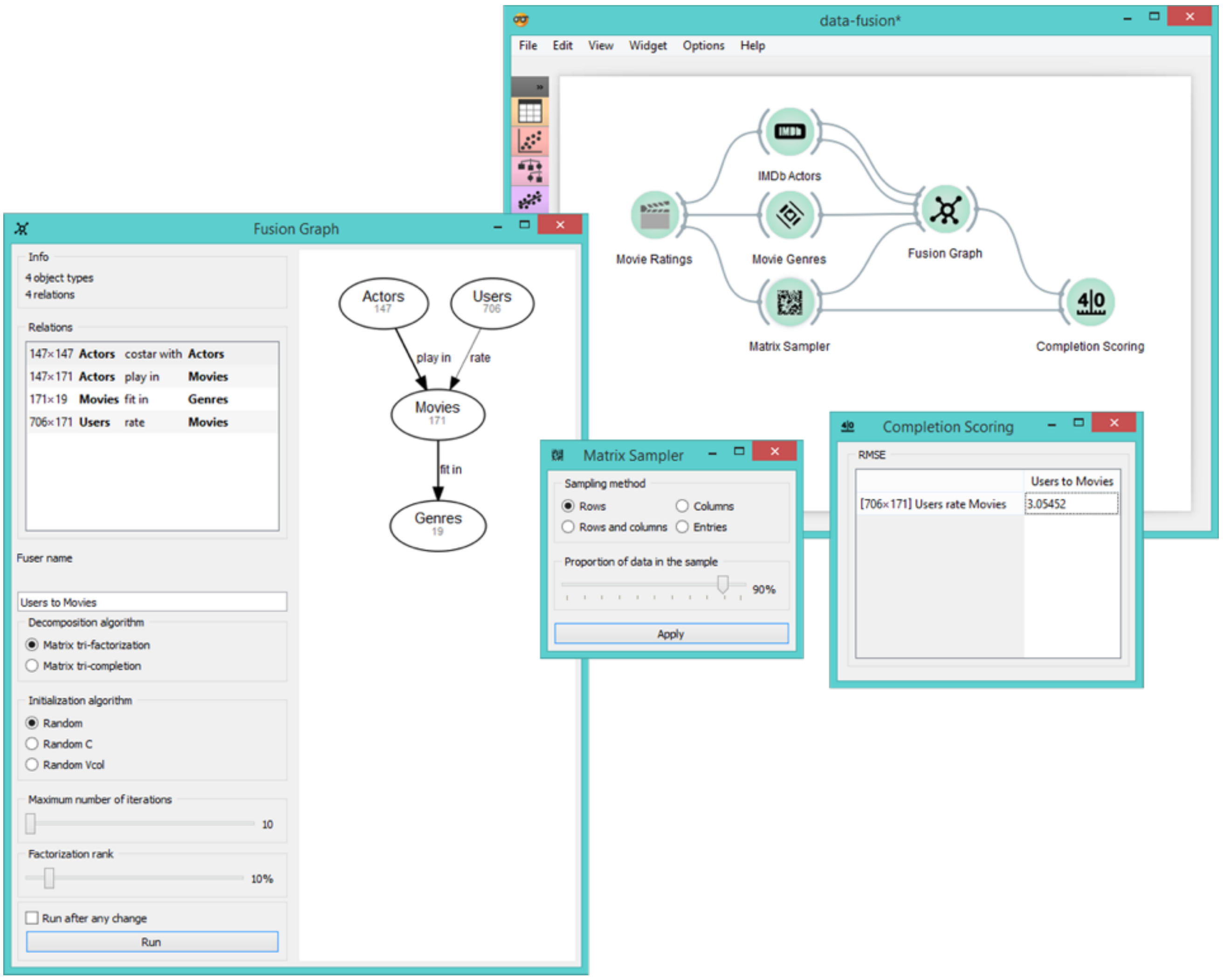
Proportion of data in the sample ❷
90%

1. Select the desired *sampling method*:
   - **rows** (randomly samples entire matrix rows)
   - **columns** (randomly samples entire matrix columns)

    - **rows and columns** (samples from the entire matrix)
    - **entries** (randomly samples individual matrix elements)
2. Select the proportion of the data you want at the output.
3. Press **Apply** to commit the changes.

Example

Matrix Sampler widget samples data into two subsets: in-sample and out-of-the-sample data. This is useful if you want to check the accuracy of matrix reconstruction with **Completion Scoring**. Feed in-sample data into the **Fusion Graph** to reconstruct the matrix and then compare the results with out-of-the-sample data.



Fusion Graph

Info
4 object types
4 relations

Relations

147×147	**Actors**	costar with	**Actors**
147×171	**Actors**	play in	**Movies**
171×19	**Movies**	fit in	**Genres**
706×171	**Users**	rate	**Movies**

Fuser name: Users to Movies

Decomposition algorithm: [✓ Matrix tri-factorization] Matrix tri-completion

Initialization algorithm: [✓ Random] Random C Random Vcol

Maximum number of iterations: 10

Factorization rank: 10%

Run after any change

Run

Matrix Sampler

Sampling method: [✓ Rows] Columns Rows and columns Entries

Proportion of data in the sample: 90%

Apply

Completion Scoring

RMSE

	Users to Movies
[706×171] Users rate Movies	3.05452

Mean Fuser

Constructs relation matrices based on the average values of matrix elements.

Signals

Inputs:

- **Fusion Graph**

  A relational scheme of a data compendium.

- **Relation**

  Relationships between two groups of objects.

Outputs:

- **Mean-fitted fusion graph**

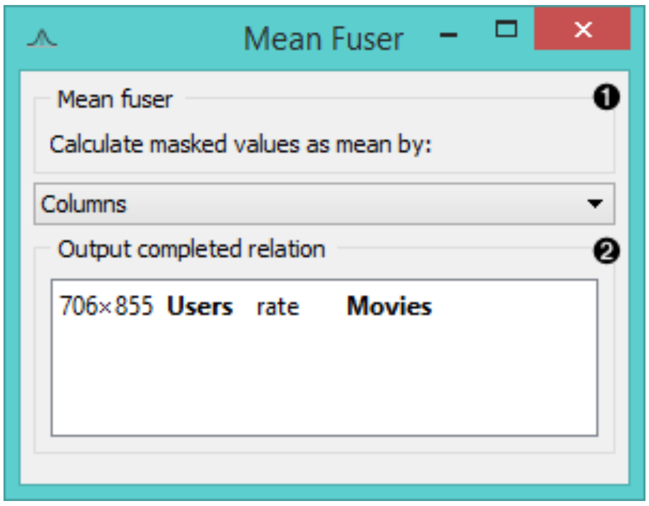
  Mean fuser.

- **Relation**

  Relationships between two groups of objects.

Description

The widget completes each relation matrix at the input based on the available data in the matrix. Unknown values in the matrix can be replaced with the values obtained by averaging matrix rows, matrix columns or the entire data matrix.



Mean fuser ❶

Calculate masked values as mean by:

Columns

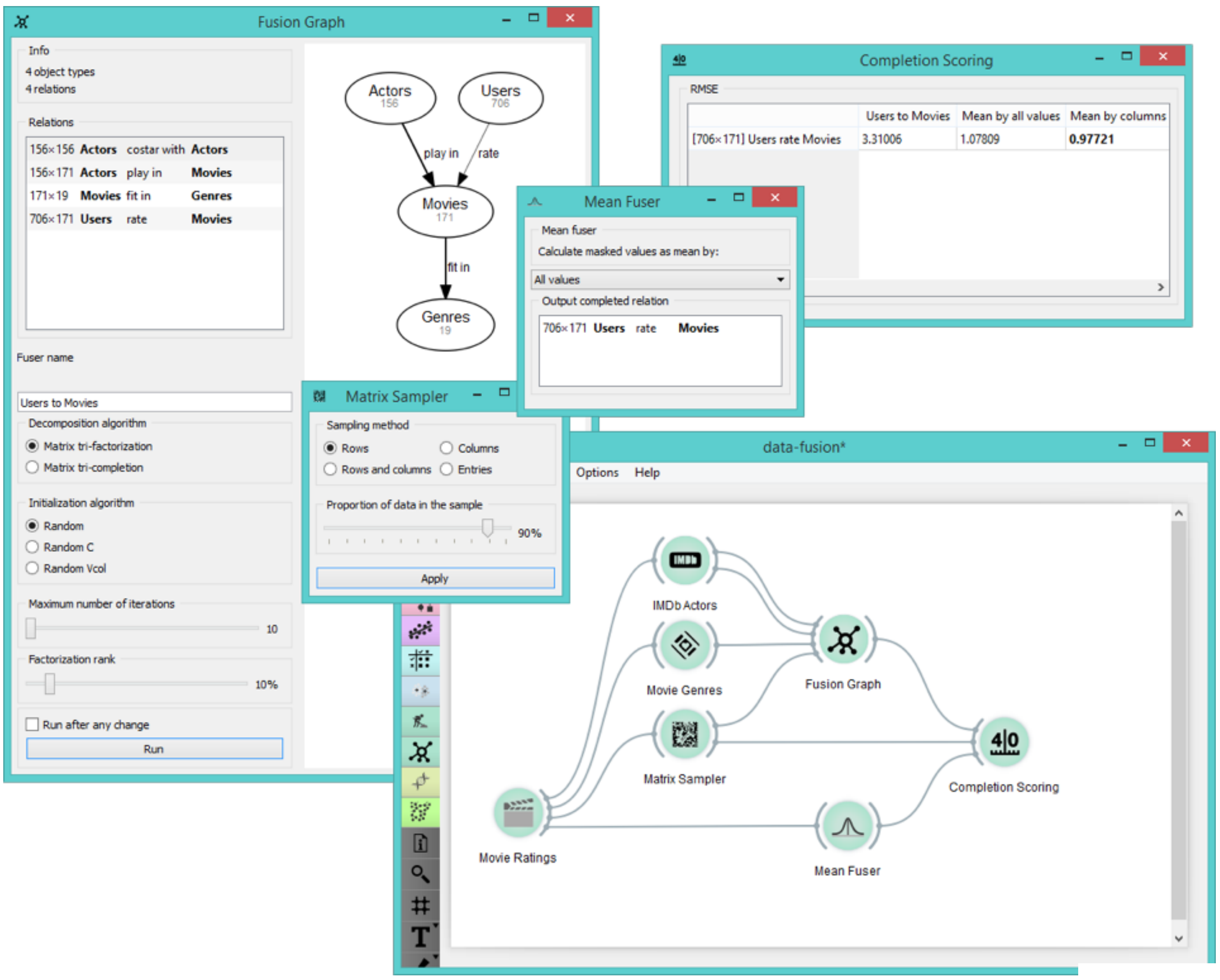
Output completed relation ❷

706×855 **Users** rate **Movies**

1. Select the axis for mean value calculation:
   - **rows**
   - **columns**
   - **all**
2. Output selected relation matrix, where unknown matrix elements are replaced with mean values.

Example

Mean Fuser widget is useful for comparing RMSE values in **Completion Scoring** widget for the input data set. In the example below we have sampled movie ratings, fed the in-sample movie ratings data into **Fusion Graph** and from there into **Completion Scoring** for evaluation. We also fed the out-of-sample data from **Matrix Sampler** into **Completion Scoring** widget as out-of-sample movie ratings data is needed to assess how well the predicted values correspond to the true data. Finally, we compare prediction to those obtained by **Mean Fuser**.



Info
4 object types
4 relations

Relations

156×156	Actors	costar with	Actors
156×171	Actors	play in	Movies
171×19	Movies	fit in	Genres
706×171	Users	rate	Movies

Fuser name: Users to Movies

Decomposition algorithm: [✓ Matrix tri-factorization] Matrix tri-completion

Initialization algorithm: [✓ Random] Random C Random Vcol

Maximum number of iterations: 10

Factorization rank: 10%

Run after any change

Run

Matrix Sampler

Sampling method: [✓ Rows] Columns Rows and columns Entries

Proportion of data in the sample: 90%

Apply

Mean Fuser

Mean fuser
Calculate masked values as mean by: All values

Output completed relation
706×171 Users rate Movies

Completion Scoring

RMSE

	Users to Movies	Mean by all values	Mean by columns
[706×171] Users rate Movies	3.31006	1.07809	**0.97721**

Movie Genres

Constructs a movies-by-genres or actors-by-genres relation matrix.

Signals

Inputs:

- **Row type**

  Instances from the input data.

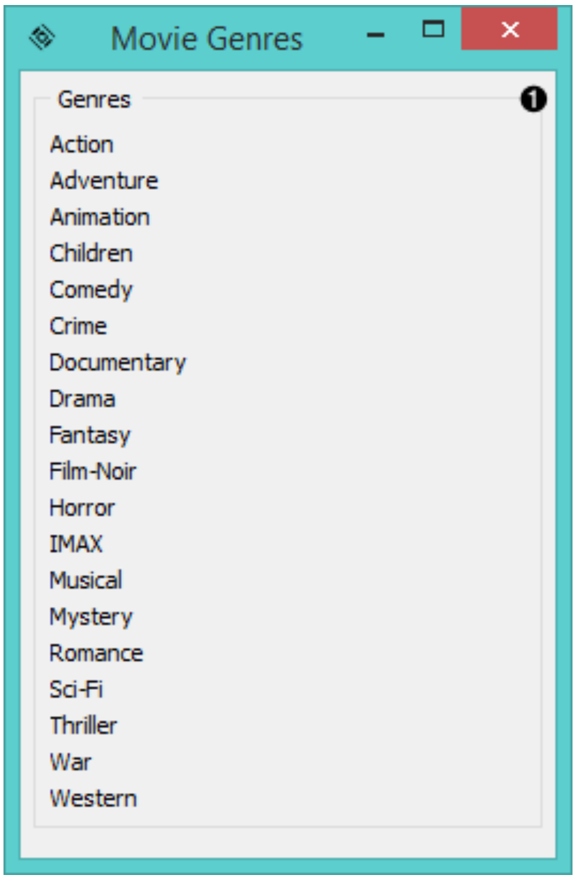
Outputs:

- **Genres**

  Data-by-genres relation matrix.

Description

This widget matches movies or actors to movie genres and forms a relation matrix. It is used to obtain information about the genres to which movies in the input belong or about genres that are associated with actors given in the input.

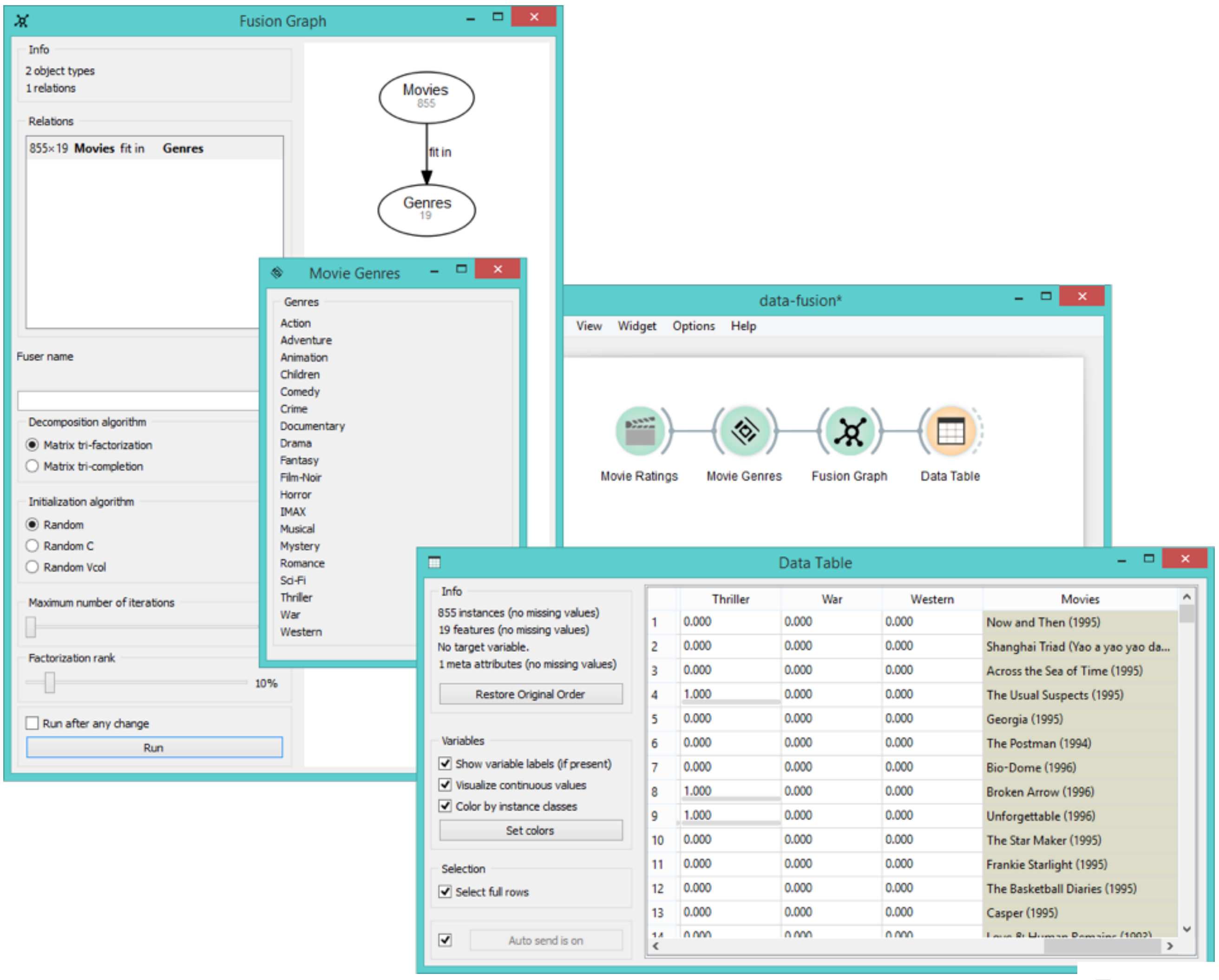


Action
Adventure
Animation
Children
Comedy
Crime
Documentary
Drama
Fantasy
Film-Noir
Horror
IMAX
Musical
Mystery
Romance
Sci-Fi
Thriller
War
Western

1. A list of movie genres included in the MovieLens database.

Example

Below we constructed a movies-by-genres relation matrix using the **Movie Genres** widget. You can see in the **Data Table** that all movies are matched by their genres.



Fusion Graph

Info
2 object types
1 relations

Relations
855×19 **Movies** fit in **Genres**

Fuser name

Decomposition algorithm
[✓ Matrix tri-factorization]
Matrix tri-completion

Initialization algorithm
[✓ Random]
Random C
Random Vcol

Maximum number of iterations

Factorization rank
10%

☐ Run after any change
Run

Movie Genres

Genres
Action
Adventure
Animation
Children
Comedy
Crime
Documentary
Drama
Fantasy
Film-Noir
Horror
IMAX
Musical
Mystery
Romance
Sci-Fi
Thriller
War
Western

data-fusion*

Data Table

Info
855 instances (no missing values)
19 features (no missing values)
No target variable.
1 meta attributes (no missing values)
Restore Original Order

Variables
✓ Show variable labels (if present)
✓ Visualize continuous values
✓ Color by instance classes
Set colors

Selection
✓ Select full rows

✓ Auto send is on

	Thriller	War	Western	Movies
1	0.000	0.000	0.000	Now and Then (1995)
2	0.000	0.000	0.000	Shanghai Triad (Yao a yao yao da...
3	0.000	0.000	0.000	Across the Sea of Time (1995)
4	1.000	0.000	0.000	The Usual Suspects (1995)
5	0.000	0.000	0.000	Georgia (1995)
6	0.000	0.000	0.000	The Postman (1994)
7	0.000	0.000	0.000	Bio-Dome (1996)
8	1.000	0.000	0.000	Broken Arrow (1996)
9	1.000	0.000	0.000	Unforgettable (1996)
10	0.000	0.000	0.000	The Star Maker (1995)
11	0.000	0.000	0.000	Frankie Starlight (1995)
12	0.000	0.000	0.000	The Basketball Diaries (1995)
13	0.000	0.000	0.000	Casper (1995)

v: latest

Movie Ratings

Constructs a relation matrix of user ratings for movies.

Signals

Inputs:

- (None)

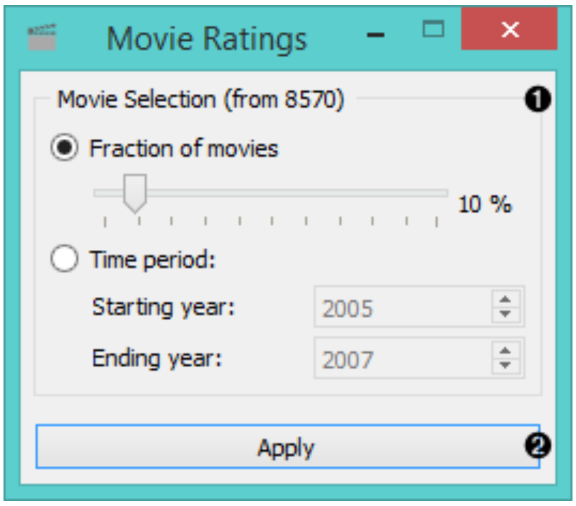
Outputs:

- **Ratings**

  Movie ratings relation matrix.

Description

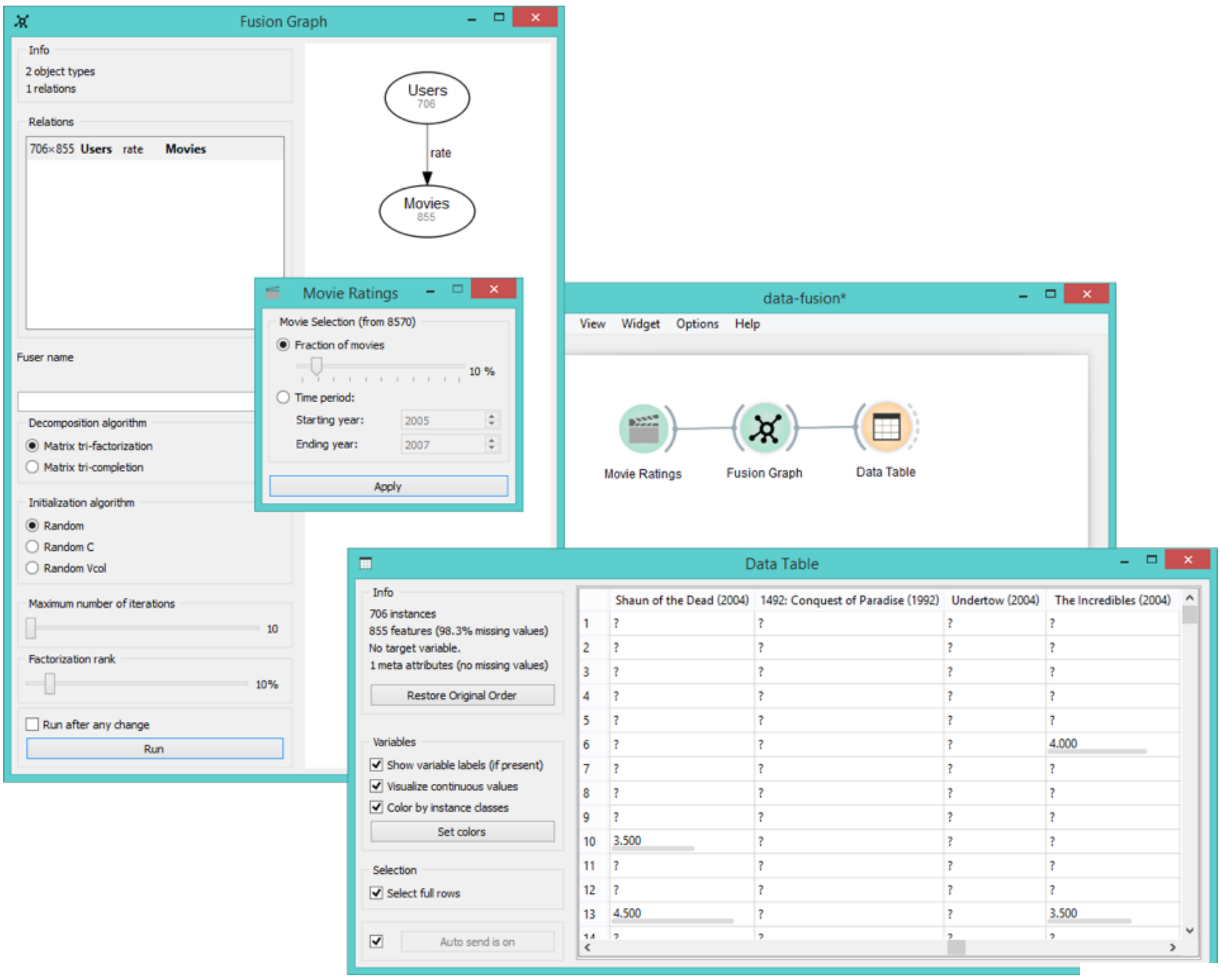
Movie Ratings widget gives you access to data on user ratings for more than 8500 movies from the Movielens database. The data set contains 1 to 5-star ratings representing user-movie preferences. This is a good widget to try out data fusion as it gives you instant access to the data.



1. Select a subset of movies for which you would like to obtain user ratings:
   - **fraction of movies** will output a specified fraction of movies selected uniformly at random from the entire database.
   - **time period** will output all the movies released in a specified time period
2. Click *Apply* to commit the changes.

Example

Movie Ratings will output users-by-movies data matrix for further analysis. Feed it into the **Fusion Graph** to decompose data matrix into the product of smaller latent data matrices or view the data in a **Data Table**.



Fusion Graph

Info
2 object types
1 relations

Relations
706×855 **Users** rate **Movies**

Fuser name

Decomposition algorithm
(●) Matrix tri-factorization
( ) Matrix tri-completion

Initialization algorithm
(●) Random
( ) Random C
( ) Random Vcol

Maximum number of iterations: 10

Factorization rank: 10%

☐ Run after any change

Movie Ratings

Movie Selection (from 8570)
(●) Fraction of movies: 10 %
( ) Time period:
Starting year: 2005
Ending year: 2007

Data Table

Info
706 instances
855 features (98.3% missing values)
No target variable.
1 meta attributes (no missing values)

Variables
☑ Show variable labels (if present)
☑ Visualize continuous values
☑ Color by instance classes

Selection
☑ Select full rows

☑ Auto send is on

	Shaun of the Dead (2004)	1492: Conquest of Paradise (1992)	Undertow (2004)	The Incredibles (2004)
1	?	?	?	?
2	?	?	?	?
3	?	?	?	?
4	?	?	?	?
5	?	?	?	?
6	?	?	?	4.000
7	?	?	?	?
8	?	?	?	?
9	?	?	?	?
10	3.500	?	?	?
11	?	?	?	?
12	?	?	?	?
13	4.500	?	?	3.500

v: latest

Table to Relation

Converts a data table into a relation matrix. Labels objects in rows and columns of a relation matrix.

Signals

Inputs:

- **Data**

  Attribute-valued data set.

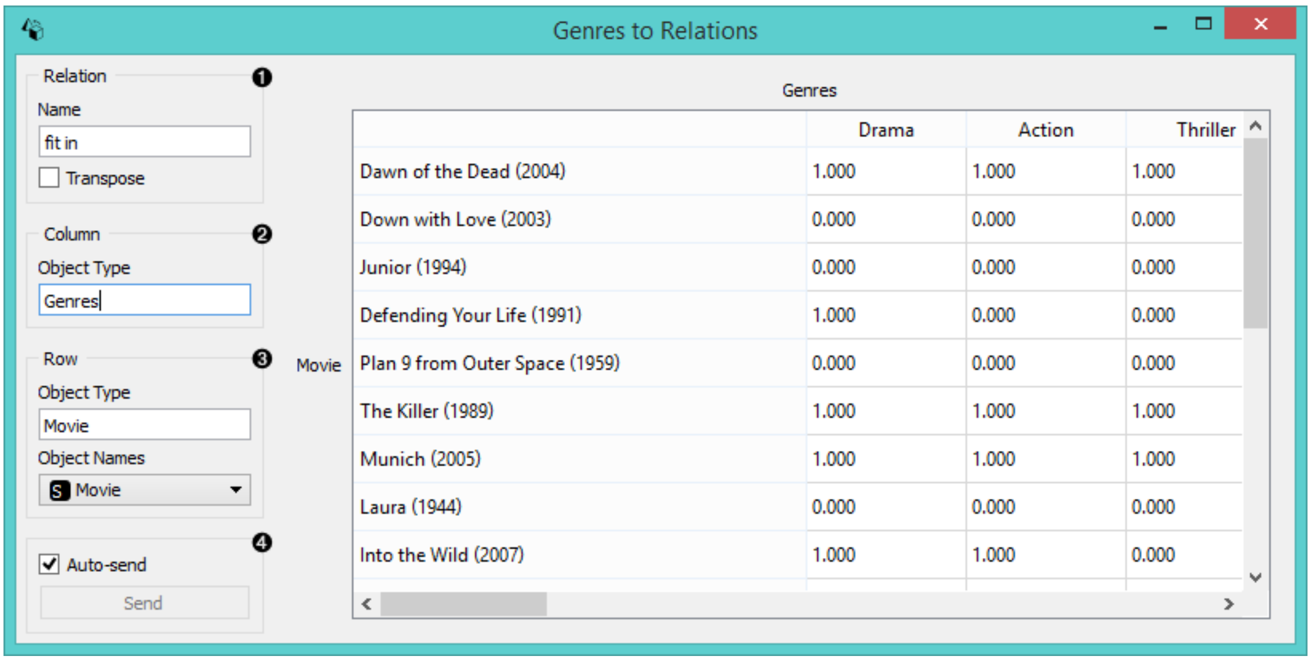
Outputs:

- **Relation**

  Relationships between two groups of objects.

Description

Table to Relation widget is probably the most often used widget in the data fusion set. It allows you to define relations just by labeling the axes. Your data set from the **File** widget will be transformed into a relation matrix, which can be later fused together with other relation matrices into a collective latent data model.


Genres to Relations

Relation ❶
Name: fit in
☐ Transpose

Column ❷
Object Type: Genres

Row ❸
Object Type: Movie
Object Names: Movie

❹
☑ Auto-send
Send

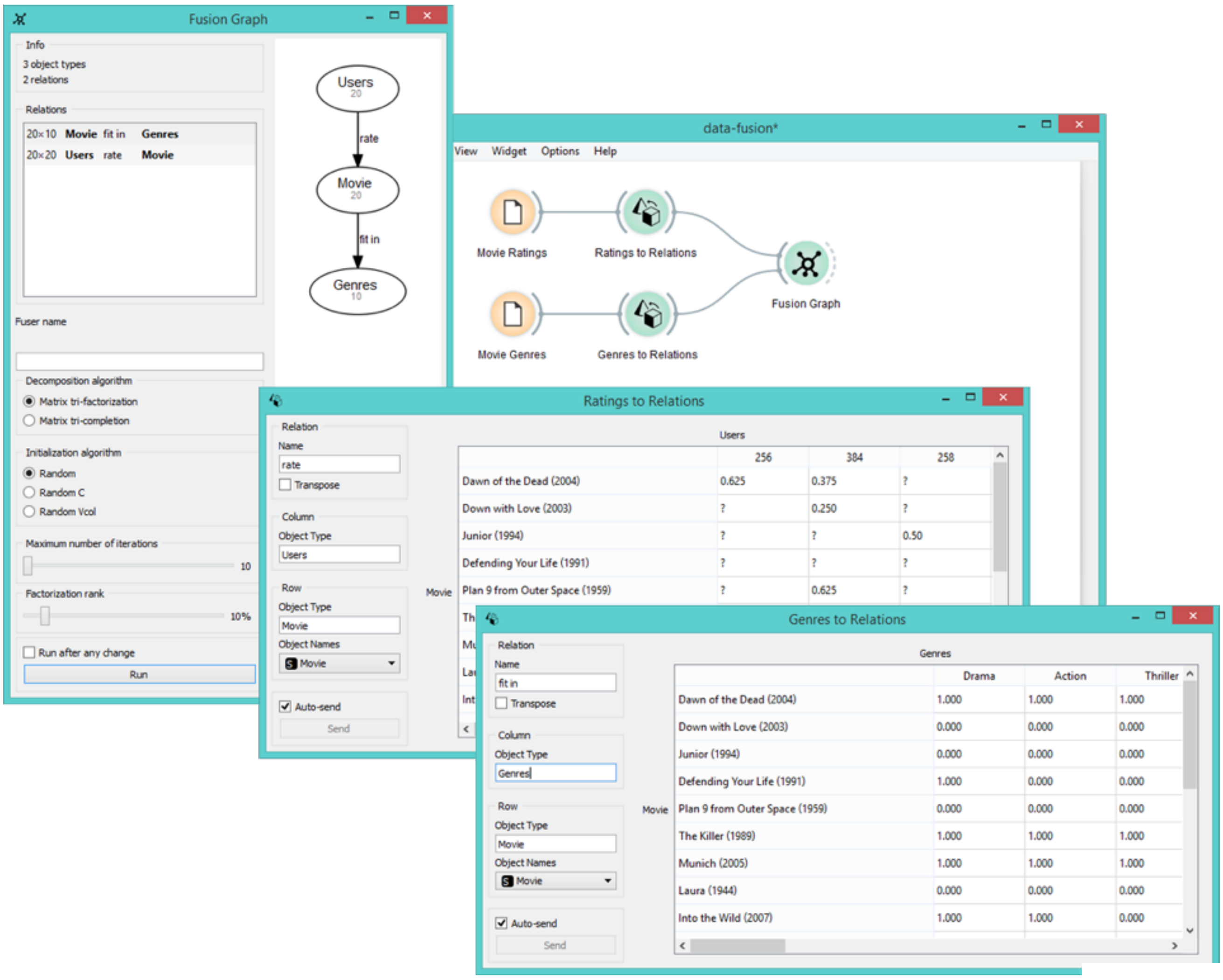
Genres (columns), Movie (rows)

	Drama	Action	Thriller
Dawn of the Dead (2004)	1.000	1.000	1.000
Down with Love (2003)	0.000	0.000	0.000
Junior (1994)	0.000	0.000	0.000
Defending Your Life (1991)	1.000	0.000	0.000
Plan 9 from Outer Space (1959)	0.000	0.000	0.000
The Killer (1989)	1.000	1.000	1.000
Munich (2005)	1.000	1.000	1.000
Laura (1944)	0.000	0.000	0.000
Into the Wild (2007)	1.000	1.000	0.000

1. Provide a descriptive name for the relation. Option *transpose* will shift the axes.
2. Label the object type in columns. Your entry will be displayed on top of the table. Note that the labels are case-sensitive.
3. Label the object type in rows. If there is a label present in the data, it will be used as default.
4. If *Auto send* is ticked, your changes will be communicated automatically. Alternatively click *Send*.

Example

In the example below we took two regular files with data on movie ratings and movie genres and fed them into separate **Table to Relation** widgets. In these widgets we specified the relations contained in the data and named the axes accordingly. See how **Fusion Graph** is then able to organize data sets into a relational graph, i.e. a data fusion graph, simply on the basis of axes names?



Fusion Graph

Info
3 object types
2 relations

Relations

20×10	**Movie**	fit in	**Genres**
20×20	**Users**	rate	**Movie**

Fuser name

Decomposition algorithm
● Matrix tri-factorization
○ Matrix tri-completion

Initialization algorithm
● Random
○ Random C
○ Random Vcol

Maximum number of iterations: 10

Factorization rank: 10%

☐ Run after any change

Ratings to Relations

Relation
Name: rate
☐ Transpose

Column
Object Type: Users

Row
Object Type: Movie
Object Names: Movie

☑ Auto-send

Users

Movie	256	384	258
Dawn of the Dead (2004)	0.625	0.375	?
Down with Love (2003)	?	0.250	?
Junior (1994)	?	?	0.50
Defending Your Life (1991)	?	?	?
Plan 9 from Outer Space (1959)	?	0.625	?

Genres to Relations

Relation
Name: fit in
☐ Transpose

Column
Object Type: Genres

Row
Object Type: Movie
Object Names: Movie

☑ Auto-send

Genres

Movie	Drama	Action	Thriller
Dawn of the Dead (2004)	1.000	1.000	1.000
Down with Love (2003)	0.000	0.000	0.000
Junior (1994)	0.000	0.000	0.000
Defending Your Life (1991)	1.000	0.000	0.000
Plan 9 from Outer Space (1959)	0.000	0.000	0.000
The Killer (1989)	1.000	1.000	1.000
Munich (2005)	1.000	1.000	1.000
Laura (1944)	0.000	0.000	0.000
Into the Wild (2007)	1.000	1.000	0.000

v: latest

Educational

Interactive k-Means

Polynomial Regression

Gradient Descent

Polynomial Classification

Gradient Descent

Educational widget that shows gradient descent algorithm on a logistic or linear regression.

Signals

Inputs:

- **Data**

Input data set.

Outputs:

- **Data**

Data with columns selected in widget.

- **Classifier**

Model produced on the current step of the algorithm.

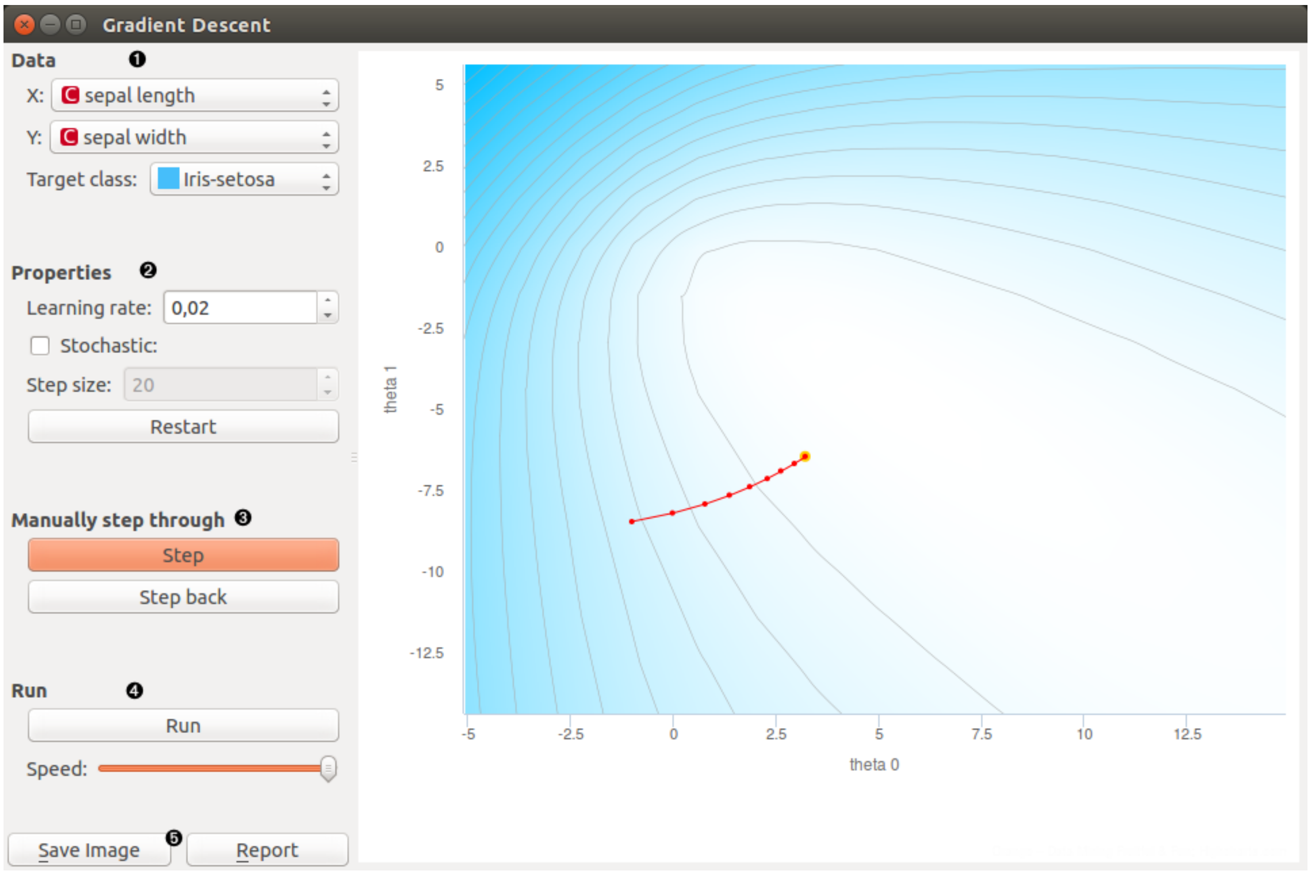
- **Coefficients**

Logistic regression coefficients on the current step of the algorithm.

Description

This widget shows steps of gradient descent for a logistic and linear regression step by step. Gradient descent is demonstrated on two attributes that are selected by user.

Gradient descent is performed on logistic regression if class in data set is discrete and linear regression if class is continuous.



1. Select two attributes (**x** and **y**) on which gradient descent algorithm is preformed. Select **target class**. It is class that is classified against all other classes.
2. **Learning rate** is step size in a gradient descent

   With **stochastic** checkbox you can select whether gradient descent is stochastic or not. If stochastic is checked you can set **step size** that is amount of steps of stochastic gradient descent performed in one press on step button.

   Restart: start algorithm from beginning
3. **Step**: perform one step of the algorithm

   Step back: make a step back in the algorithm
4. **Run**: automatically perform several steps until algorithm converge

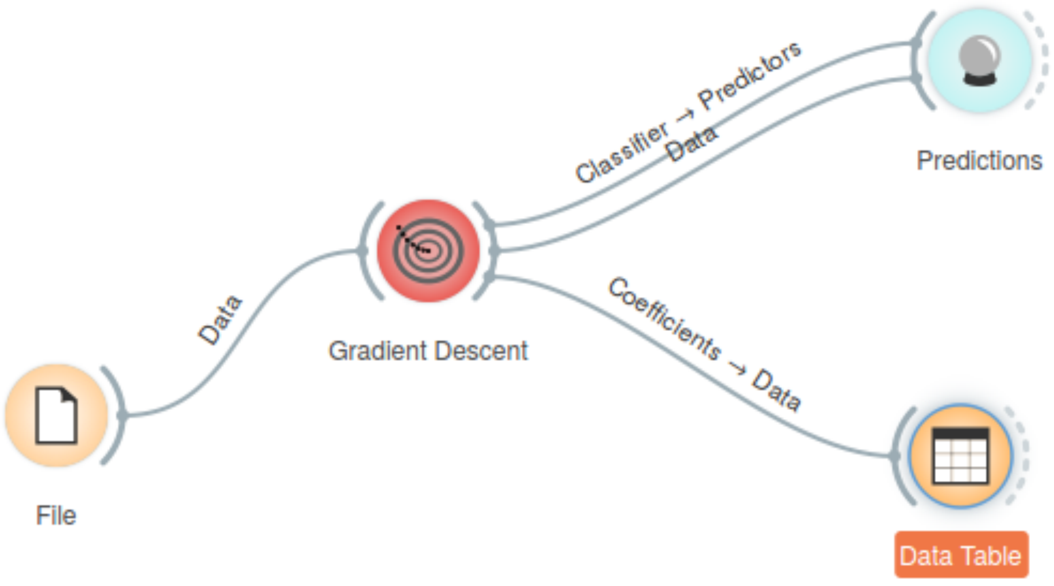
   Speed: set speed of automatic stepping
5. **Save Image** saves the image to the computer in a .svg or .png format.

   Report includes widget parameters and visualization in the report.

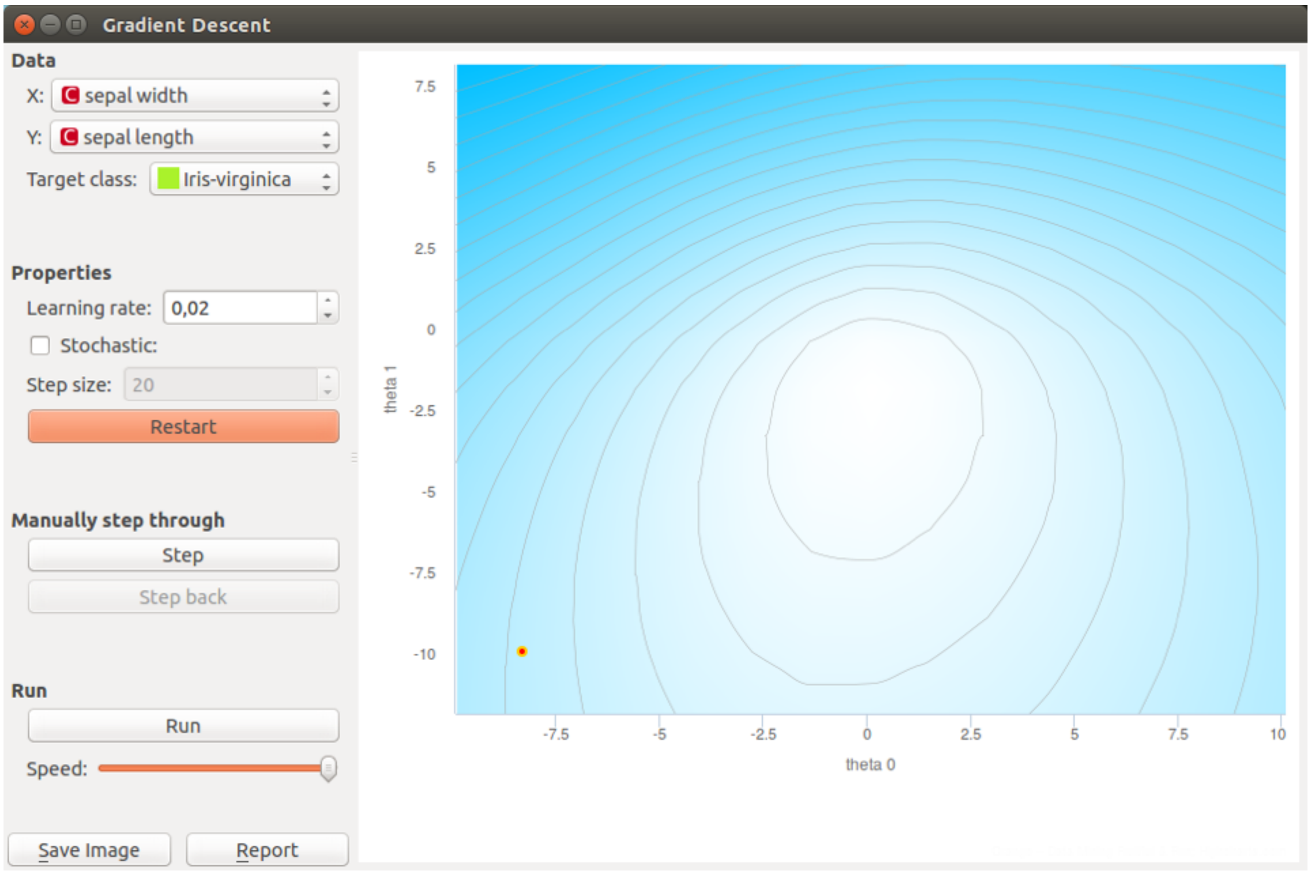
Example

In Orange we connected *File* widget with *Iris* data set to *Gradient Descent* widget. Iris data set has discrete class so *Logistic regression* will be used this time. We connected outputs of the widget to *Predic-*

tions widget to see how data are classified and *Data Table* widget where we inspect coefficients of logistic regression.



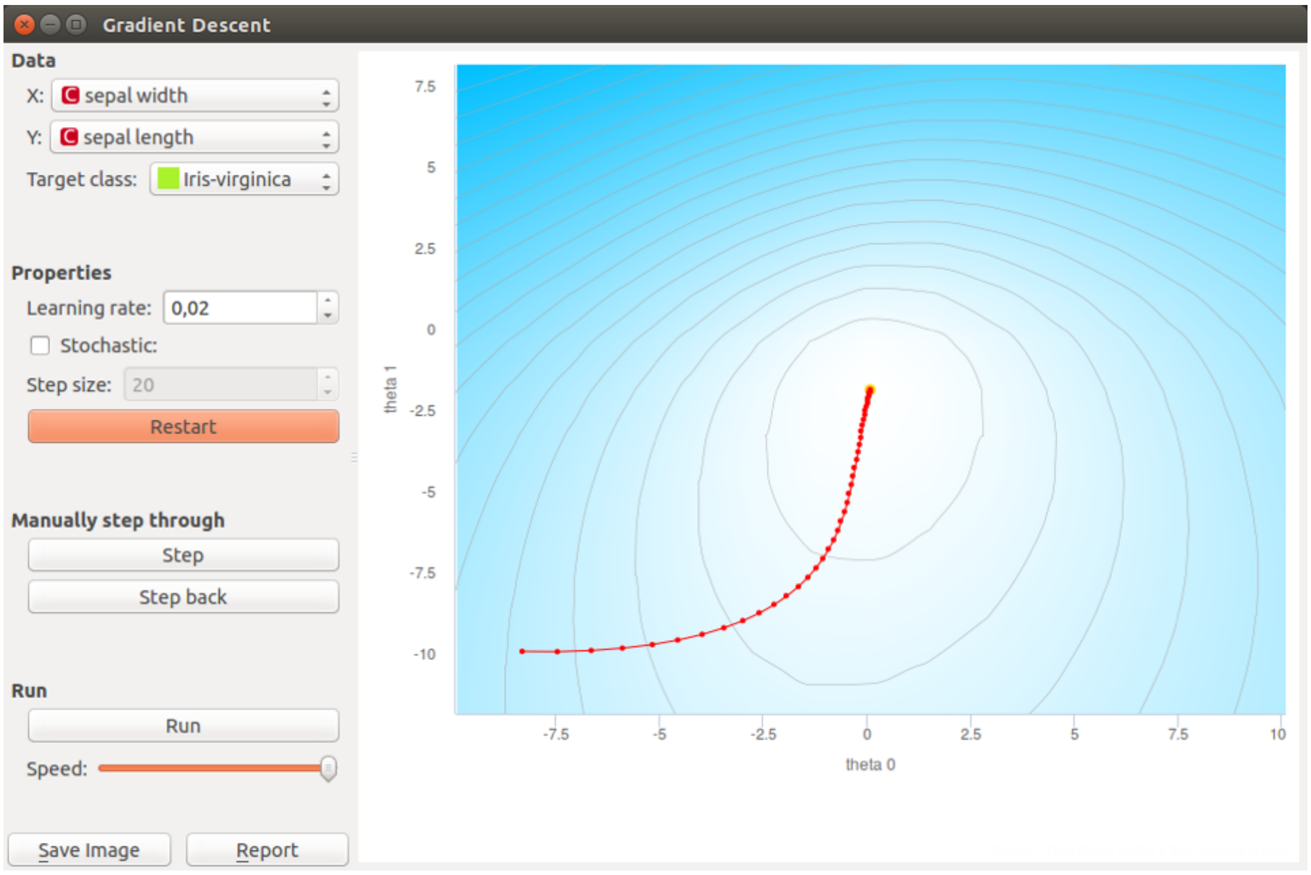
We opened *Gradient Descent* widget and set *X* to *sepal width* and *Y* to *sepal length*. Target class is set to *Iris-virginica*. We set *learning rate* to 0.02. With click in graph we set beginning coefficients (red dot).


Gradient Descent

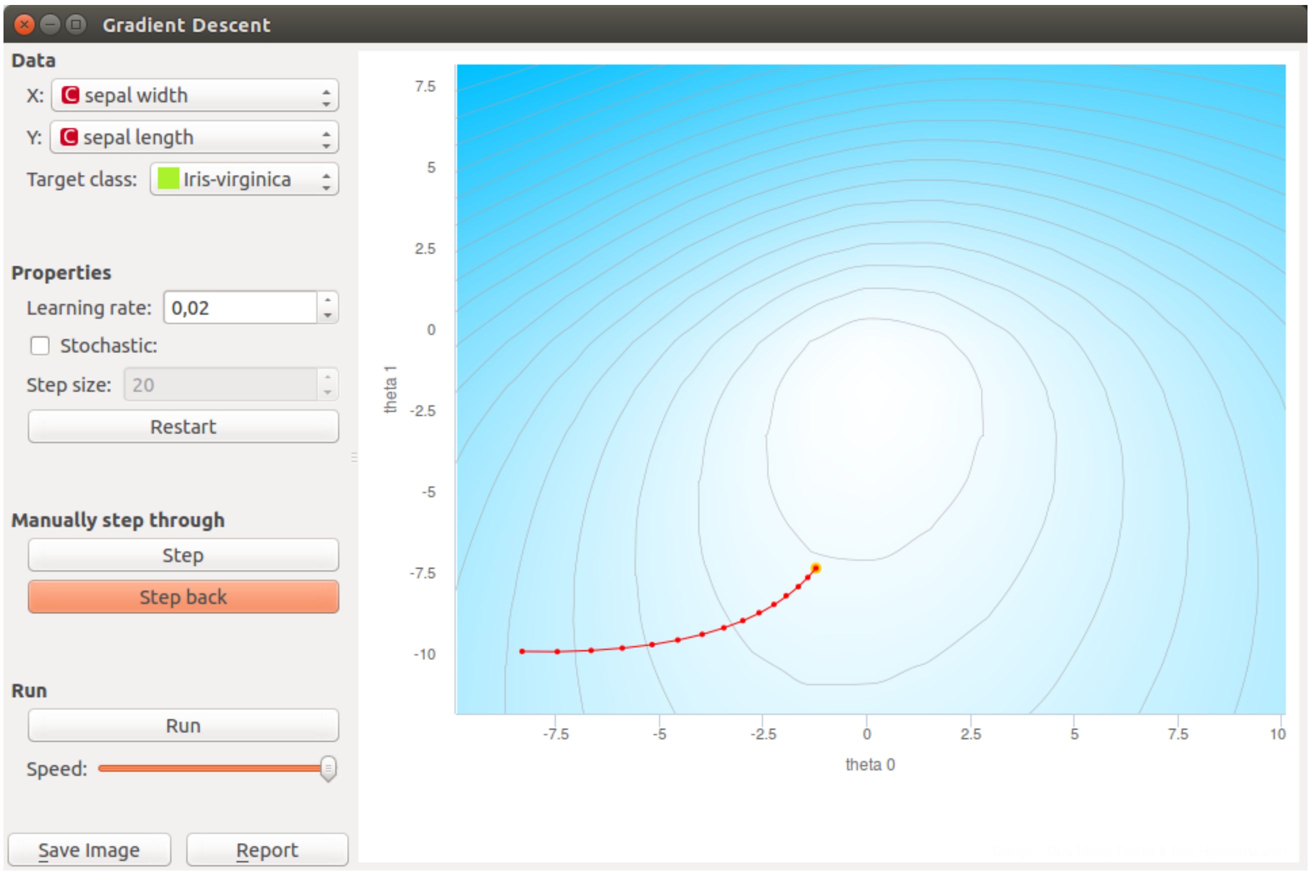
Data
X: sepal width
Y: sepal length
Target class: Iris-virginica

Properties
Learning rate: 0,02
Stochastic:
Step size: 20

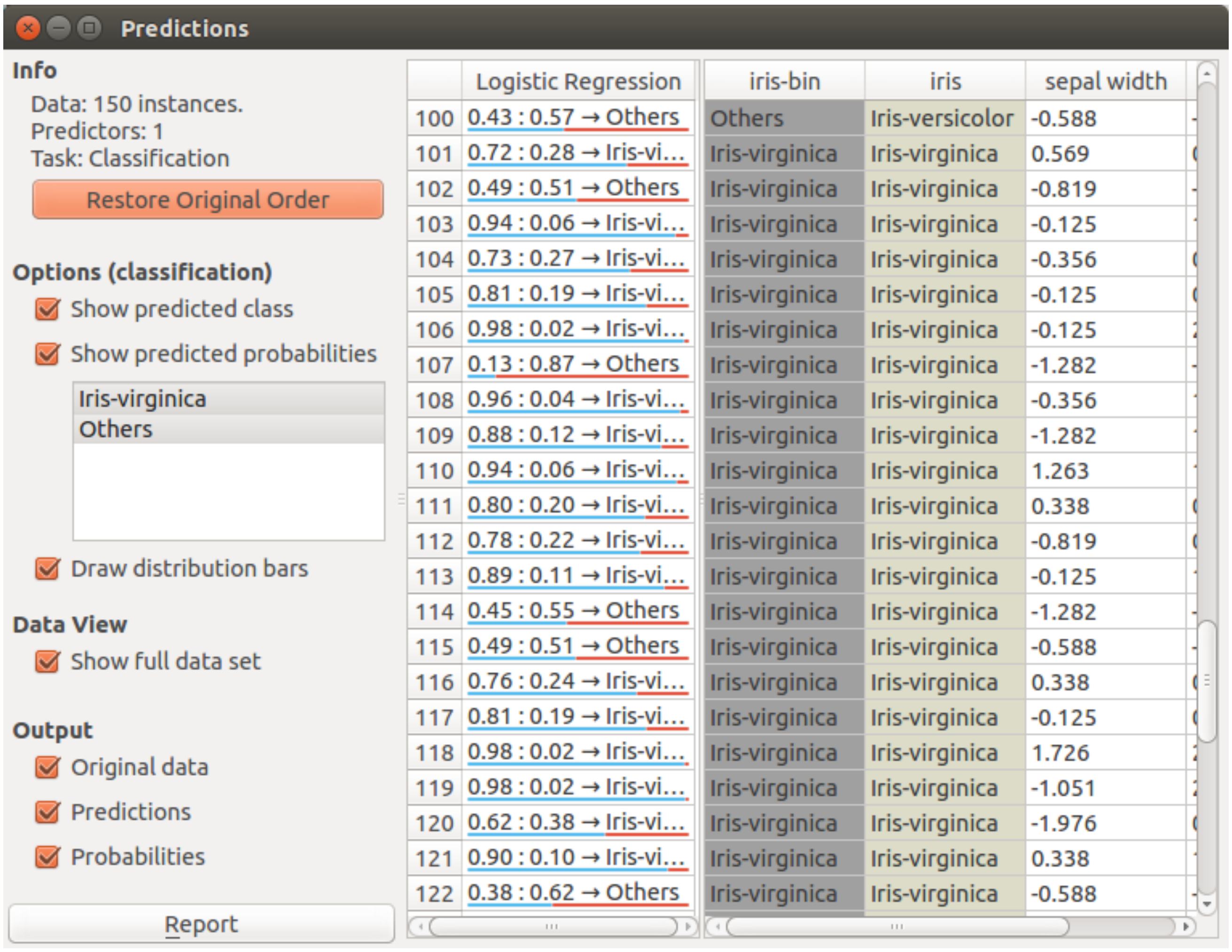
We performs step of the algorithm with pressing **Step** button. When we get bored with clicking we can finish stepping with press on **Run** button.



If we want to go back in the algorithm we can do it with pressing **Step back** button. This will also change model. Current model uses positions of last coefficients (red-yellow dot).



In the end we want to see predictions for input data so we can open *Predictions* widget. Predictions are listed in left column. We can compare this predictions to real classes.



Info

Data: 150 instances.
Predictors: 1
Task: Classification

Options (classification)

- ✓ Show predicted class
- ✓ Show predicted probabilities
  - Iris-virginica
  - Others
- ✓ Draw distribution bars

Data View

- ✓ Show full data set

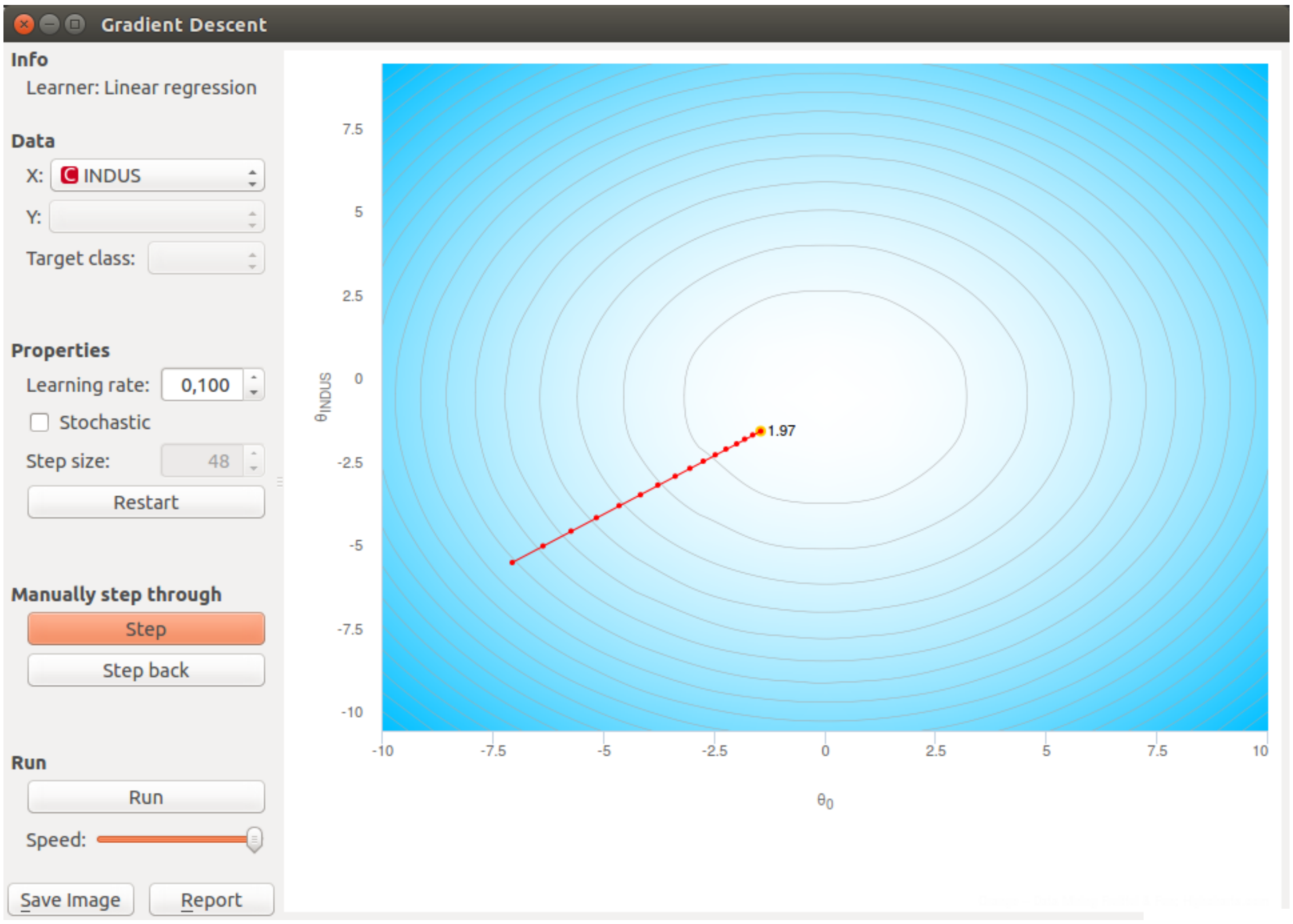
Output

- ✓ Original data
- ✓ Predictions
- ✓ Probabilities

	Logistic Regression	iris-bin	iris	sepal width
100	0.43 : 0.57 → Others	Others	Iris-versicolor	-0.588
101	0.72 : 0.28 → Iris-vi…	Iris-virginica	Iris-virginica	0.569
102	0.49 : 0.51 → Others	Iris-virginica	Iris-virginica	-0.819
103	0.94 : 0.06 → Iris-vi…	Iris-virginica	Iris-virginica	-0.125
104	0.73 : 0.27 → Iris-vi…	Iris-virginica	Iris-virginica	-0.356
105	0.81 : 0.19 → Iris-vi…	Iris-virginica	Iris-virginica	-0.125
106	0.98 : 0.02 → Iris-vi…	Iris-virginica	Iris-virginica	-0.125
107	0.13 : 0.87 → Others	Iris-virginica	Iris-virginica	-1.282
108	0.96 : 0.04 → Iris-vi…	Iris-virginica	Iris-virginica	-0.356
109	0.88 : 0.12 → Iris-vi…	Iris-virginica	Iris-virginica	-1.282
110	0.94 : 0.06 → Iris-vi…	Iris-virginica	Iris-virginica	1.263
111	0.80 : 0.20 → Iris-vi…	Iris-virginica	Iris-virginica	0.338
112	0.78 : 0.22 → Iris-vi…	Iris-virginica	Iris-virginica	-0.819
113	0.89 : 0.11 → Iris-vi…	Iris-virginica	Iris-virginica	-0.125
114	0.45 : 0.55 → Others	Iris-virginica	Iris-virginica	-1.282
115	0.49 : 0.51 → Others	Iris-virginica	Iris-virginica	-0.588
116	0.76 : 0.24 → Iris-vi…	Iris-virginica	Iris-virginica	0.338
117	0.81 : 0.19 → Iris-vi…	Iris-virginica	Iris-virginica	-0.125
118	0.98 : 0.02 → Iris-vi…	Iris-virginica	Iris-virginica	1.726
119	0.98 : 0.02 → Iris-vi…	Iris-virginica	Iris-virginica	-1.051
120	0.62 : 0.38 → Iris-vi…	Iris-virginica	Iris-virginica	-1.976
121	0.90 : 0.10 → Iris-vi…	Iris-virginica	Iris-virginica	0.338
122	0.38 : 0.62 → Others	Iris-virginica	Iris-virginica	-0.588

If we want to demonstrate *linear regression* we can change data set to *Housing*. That data set has a continuous class variable. When using linear regression we can select only one feature what means that our function is linear. The another parameter that is plotted in the graph is intercept of a linear function.

This time we selected *INDUS* as a independent variable. In widget we can make same actions as before. In the end we can also check predictions for each point with *Predictions* widget. And coefficients of linear regression in a *Data Table*.



Learner: Linear regression

X: INDUS

Learning rate: 0,100

Stochastic

Step size: 48

Interactive k-means

Educational widget that shows the working of a k-means clustering.

Signals

Inputs:

- **Data**

Input data set.

Outputs:

- **Data**

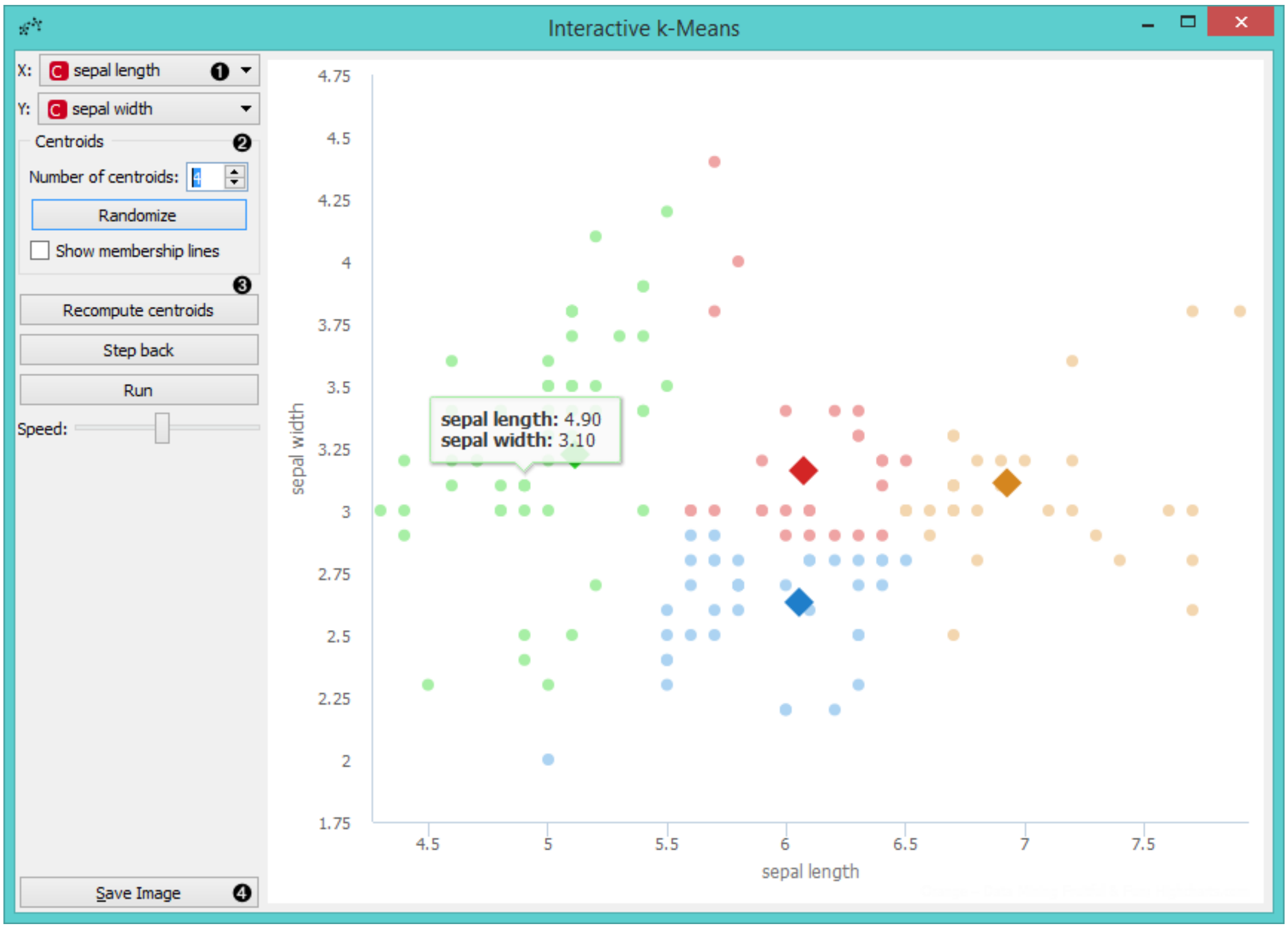
Data set with cluster annotation.

- **Centroids**

Centroids position.

Description

The aim of this widget is to show the working of a k-means clustering algorithm on two attributes from a data set. The widget applies k-means clustering to the selected two attributes step by step. Users can step through the algorithm and see how it works.



1. Select attributes for **x** and **y** axis.
2. *Number of centroids*: set the number of centroids.

   Randomize: randomly assigns position of centroids. If you want to add centroid on a particular position in the graph, click on this position. If you want to move the centroid, drag and drop it on the desired position.

   Show membership lines: if ticked, connection between data points and closest centroids are shown.
3. **Recompute centroids** or **Reassign membership**: step through different stages of the algoritm. *Recompute centroids* moves centroids to new positions, based on the most central position of the data assigned to the centroid. *Reassign membership* reassigns data points to the centroid they are the closest to.

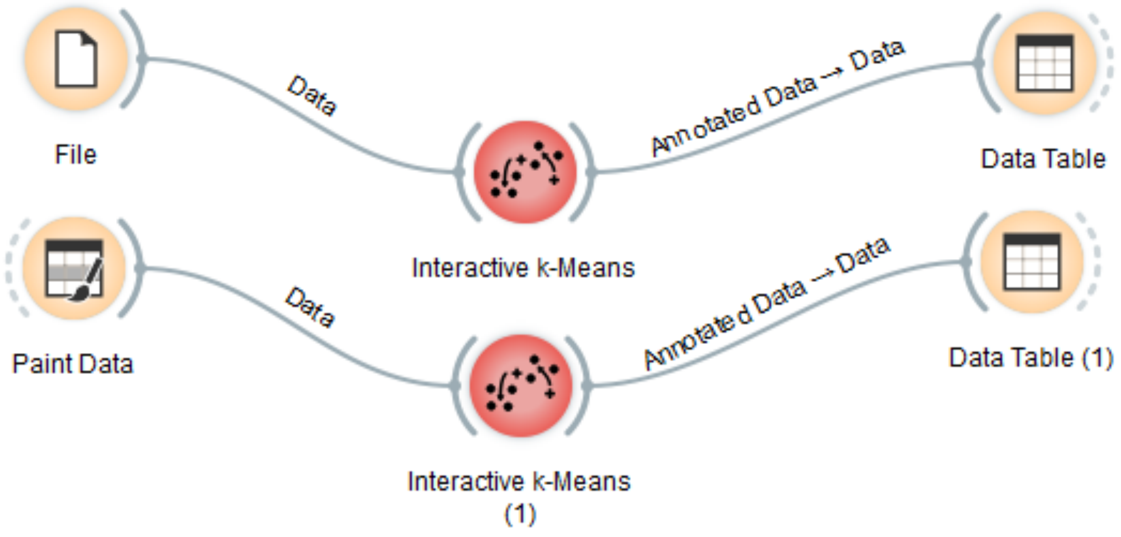
   Step back: make a step back in the algorithm.

   Run: step through the algorithm automatically.

   Speed: set the speed of automatic stepping.
4. *Save Image* saves the image to the computer in a .svg or .png format.

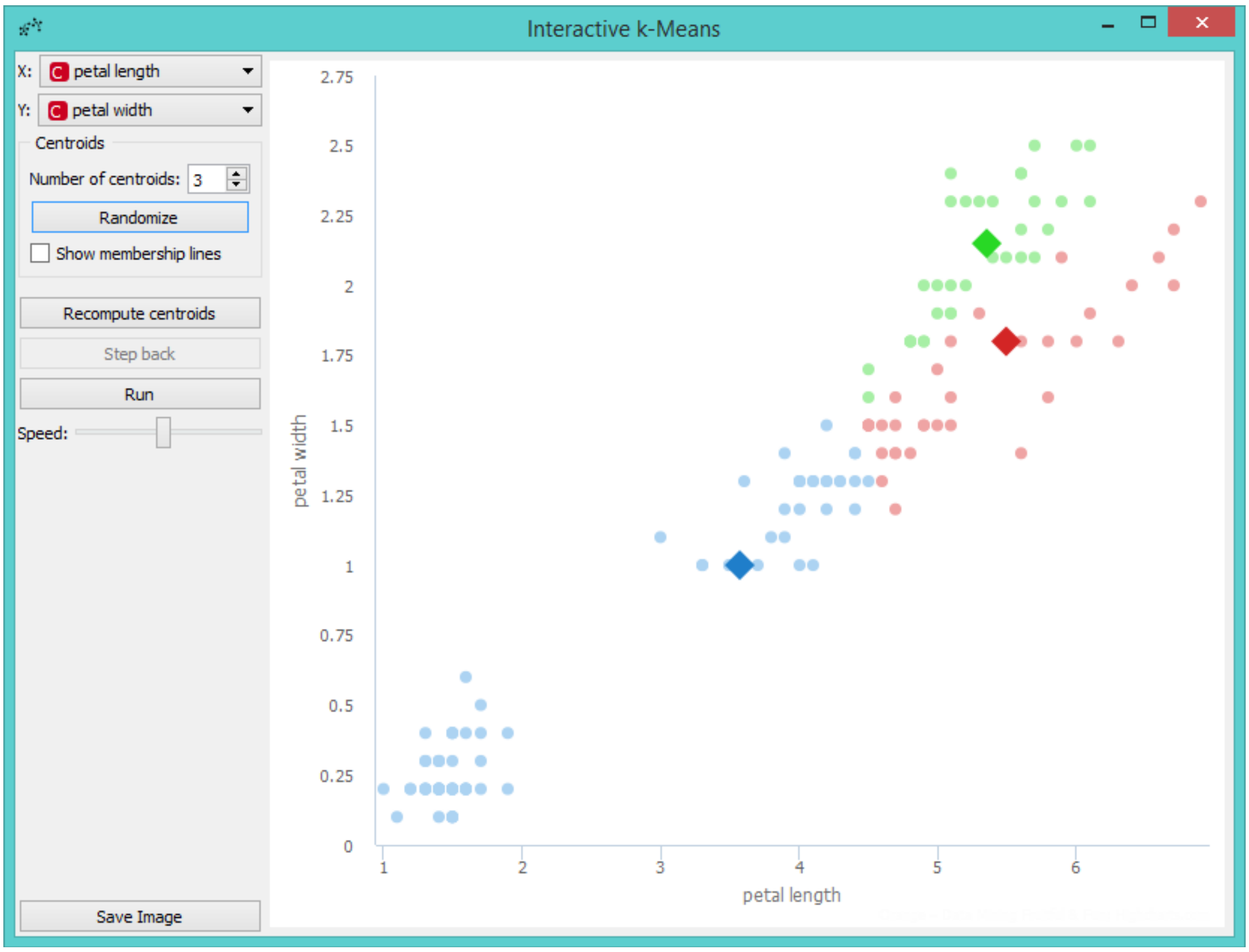
Example

Here are two possible schemas that show how the **Interactive k-Means** widget can be used. You can load the data from **File** or use any other data source, such as **Paint Data**. Interactive k-Means widget also produces a data table with results of clustering and a table with centroids positions. These data can be inspected with the **Data Table** widget.



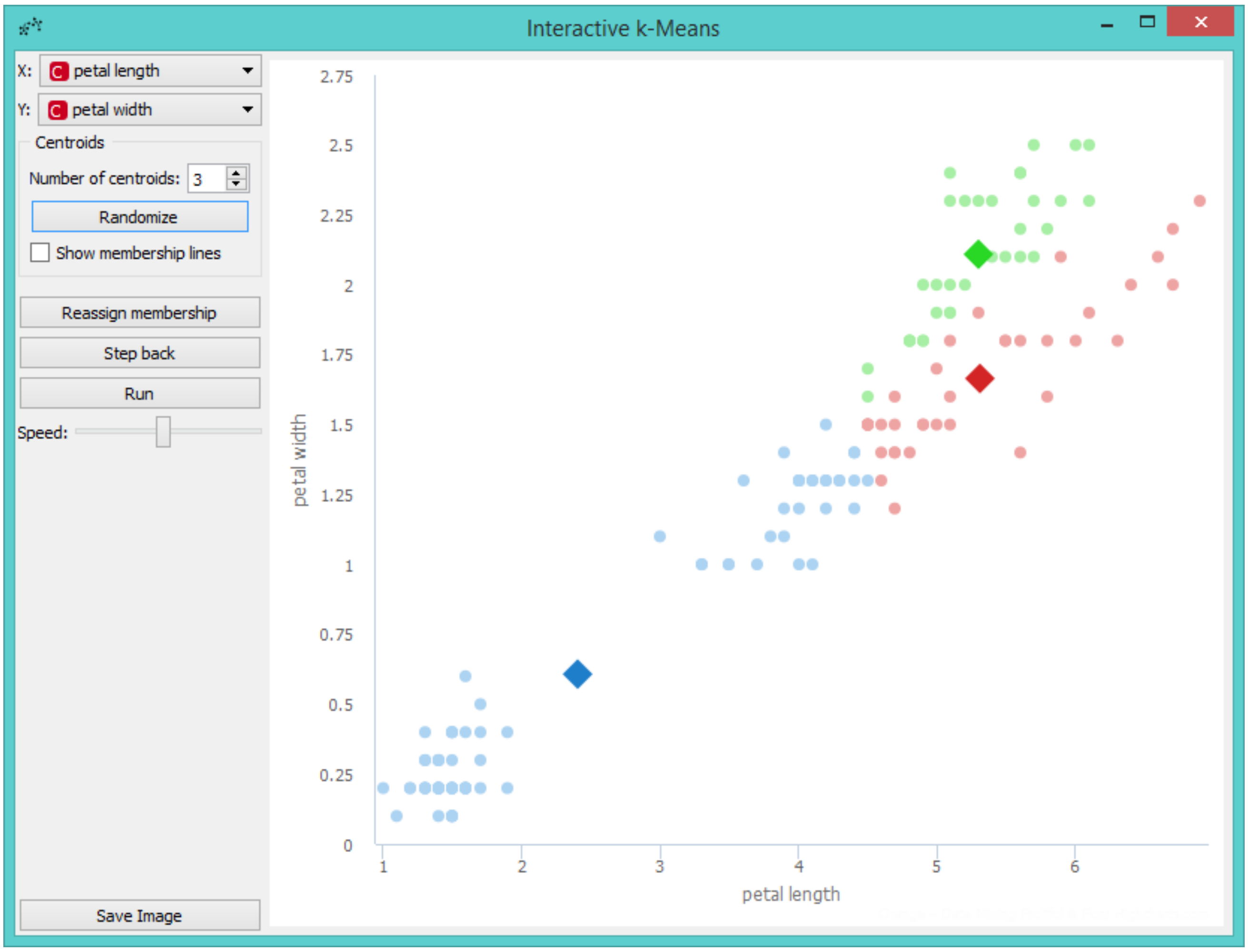
Let us demonstrate the working of the widget on *Iris* data set.

We provide the data using **File**. Then we open **Interactive k-Means**. Say, we will demonstrate k-Means on *petal length* and *petal width* attributes, so we set them as *X* and *Y* parameters. We also decided to perform clustering for 3 clusters. This is set as the *Number of centroids*.

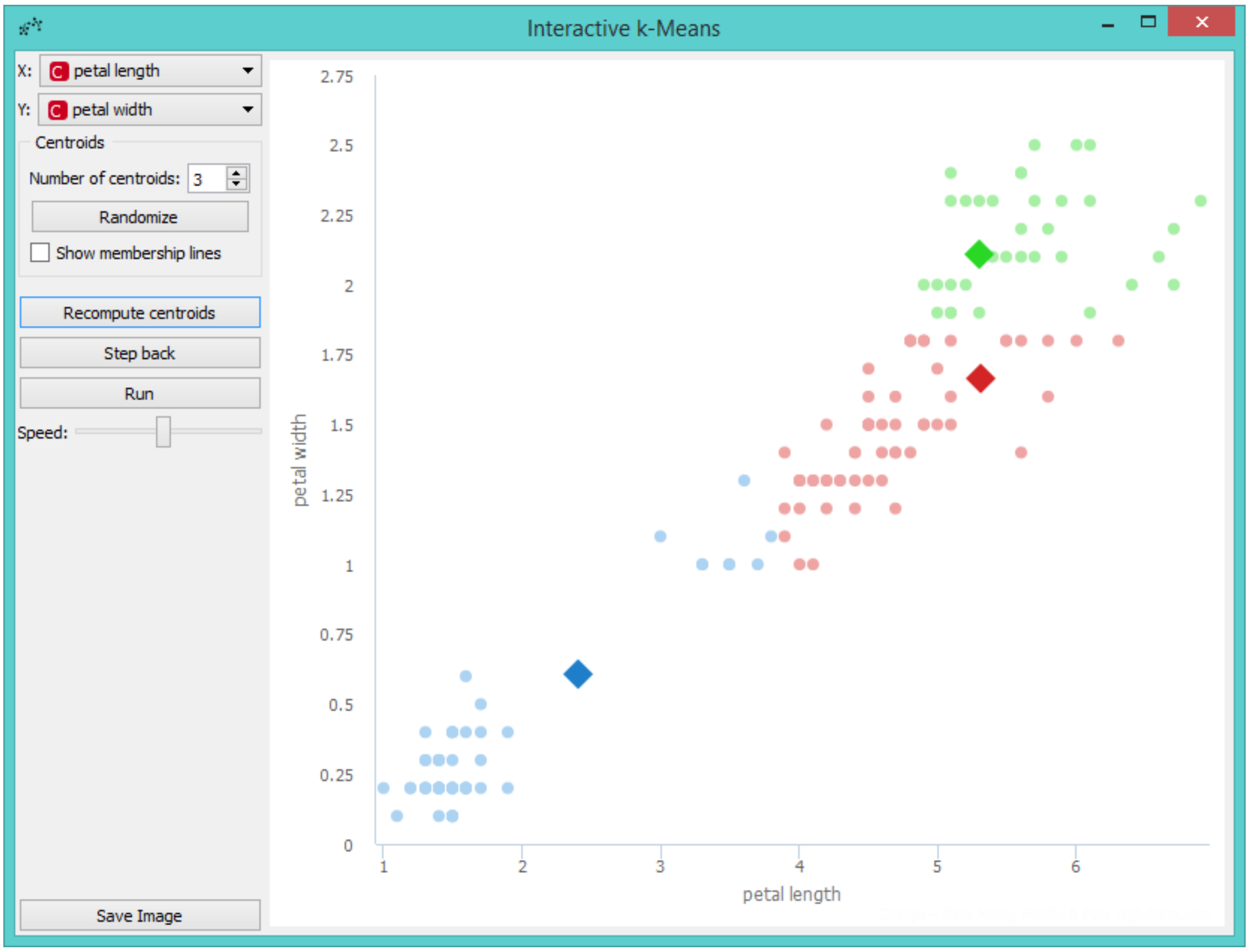


X: petal length
Y: petal width
Centroids
Number of centroids: 3

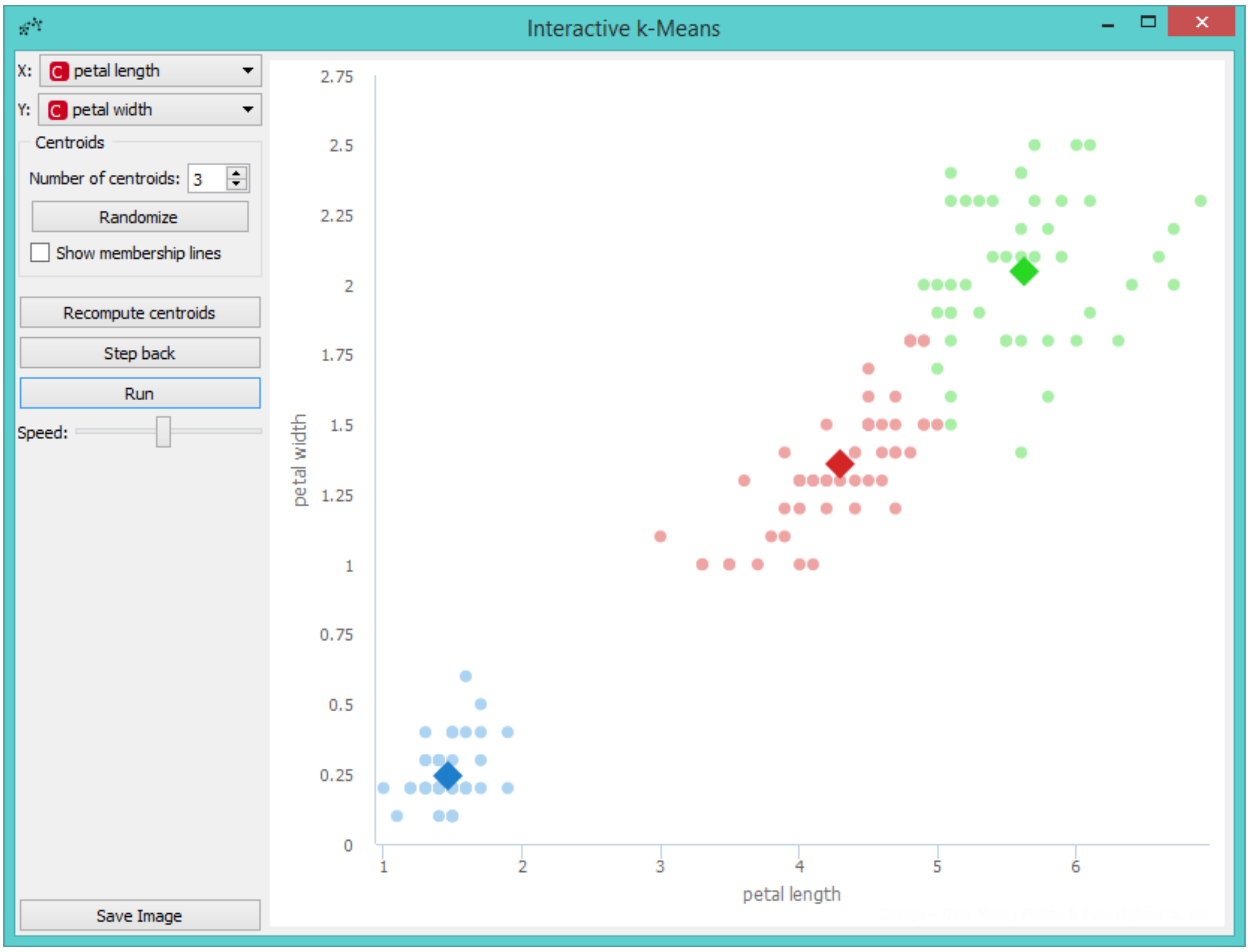
If we are not satisfied with positions of centroids we can change them with a click on the **Randomize** button. Then we perform the first recomputing of centroids with a click on the **Recompute centroids**. We get the following image.



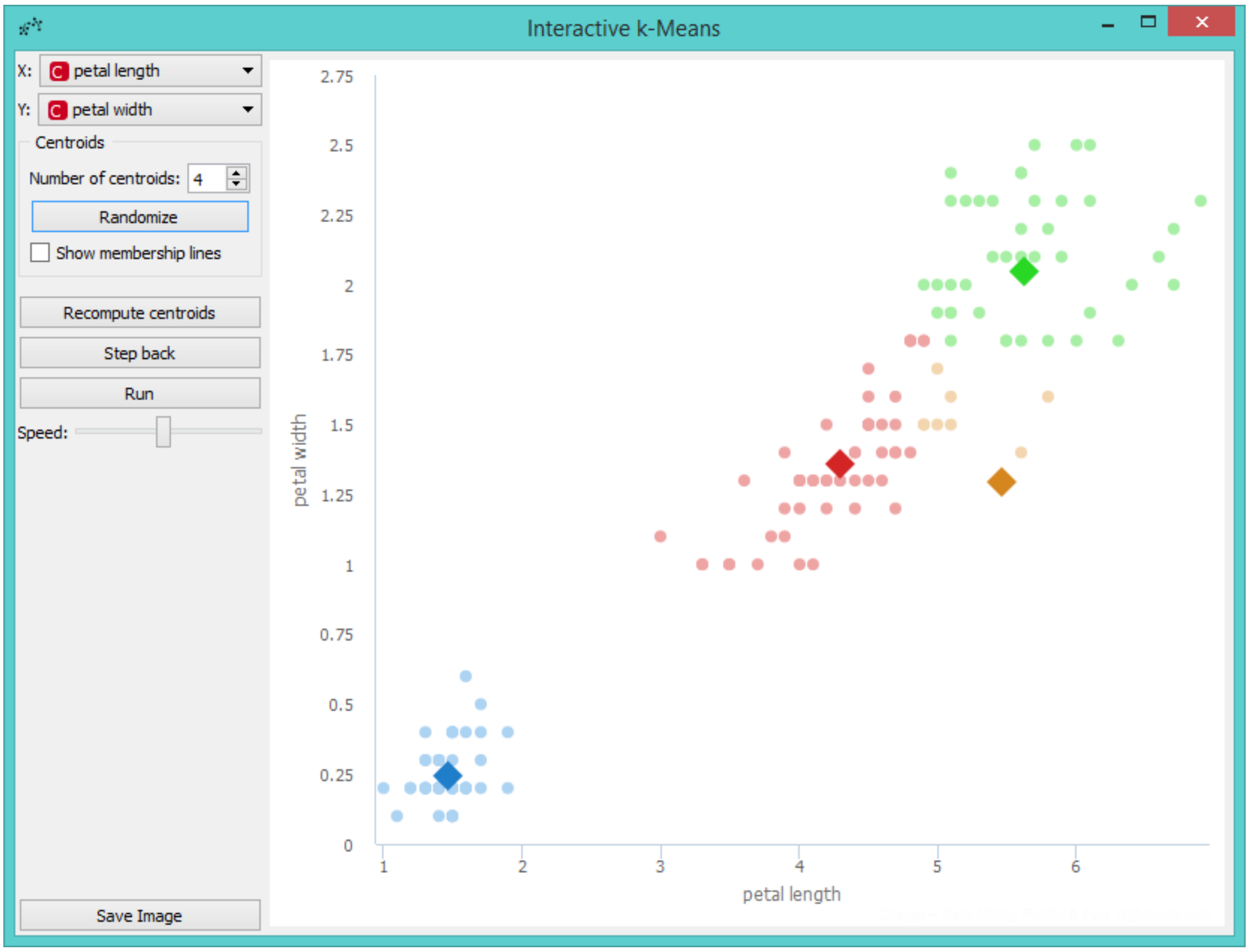
The next step is to reassign membership of all points to the closest centroid. This is performed with a click on the **Reasign membership** button.



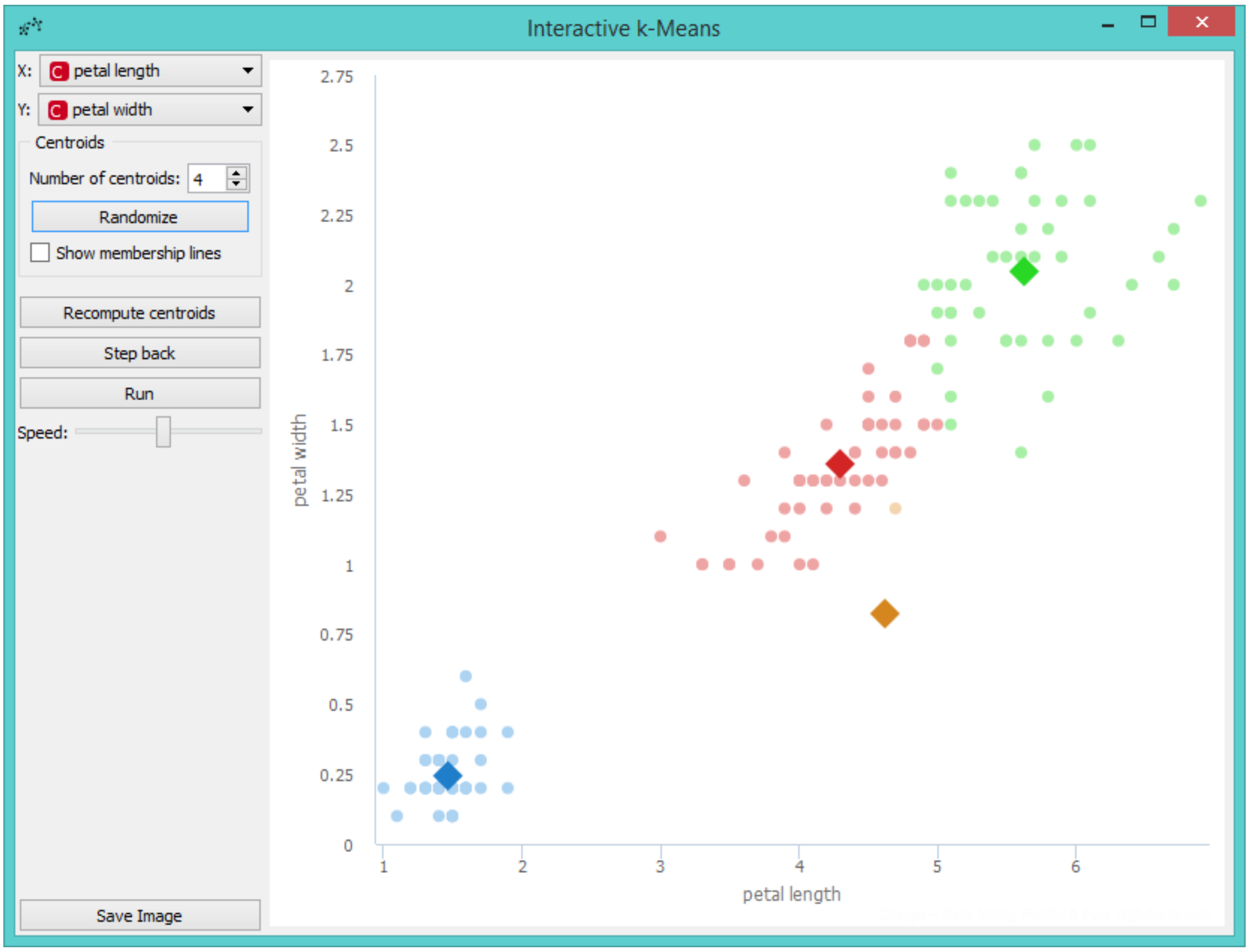
Then we repeat these two steps until the algorithm converges. This is the final result.



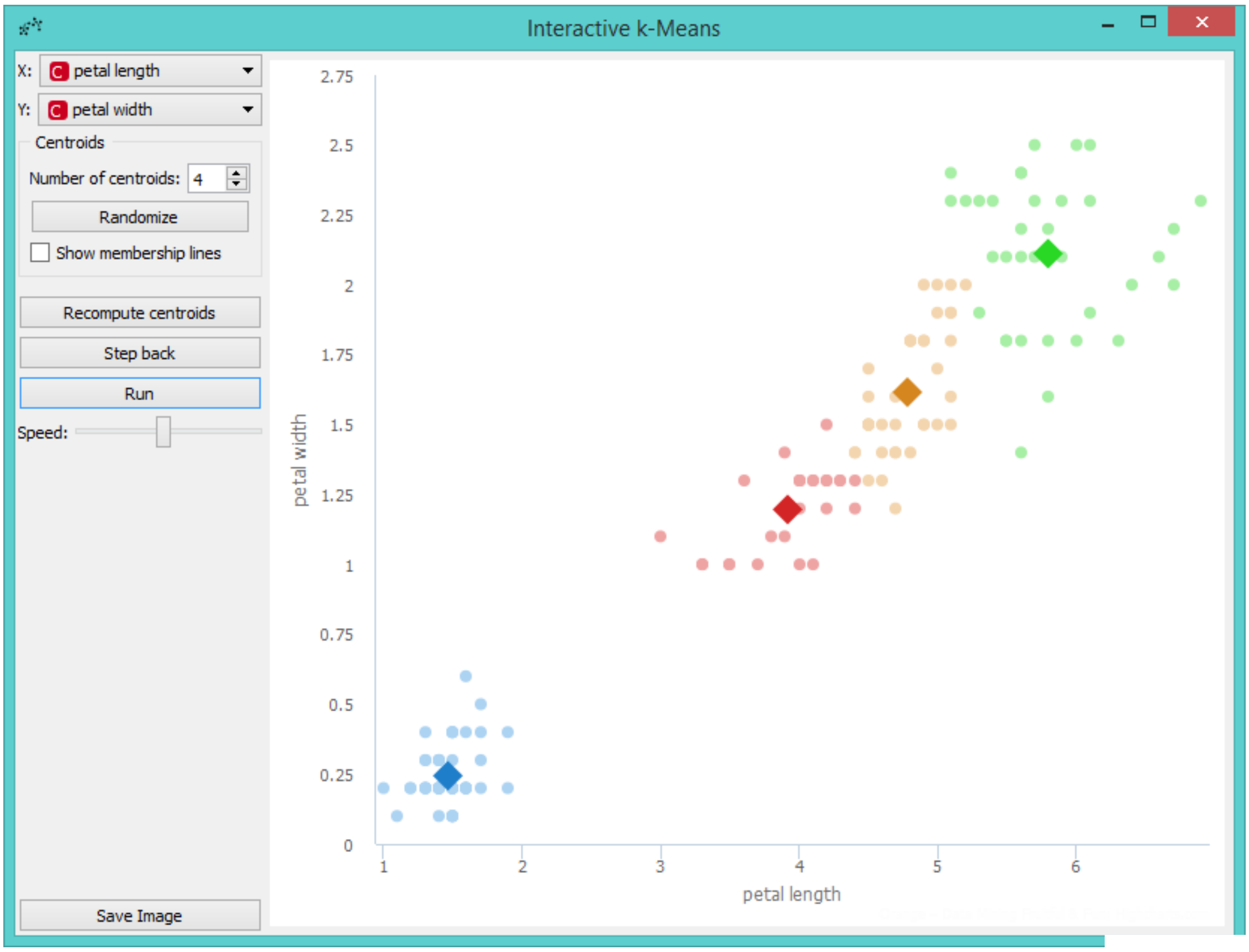
Perhaps we are not satisfied with the result because we noticed that maybe classification into 4 clusters would be better. So we decided to add a new centroid. We can do this by increasing the number of centroids in the control menu or with a click on the position in the graph where we want to place the centroid. We decided to add it with a click. The new centroid is the orange one.



Now we can repeat running the algorithm until it converges again, but before that we will move the new centroid to change the behavior of the algorithm. We grabbed the orange centroid and moved it to the desired position.



Then we press *Run* and observe the centroids while the algorithm converges again.



Interactive k-Means
X: petal length
Y: petal width
Centroids
Number of centroids: 4
Randomize
Show membership lines
Recompute centroids
Step back
Run
Speed:
Save Image
2.75
2.5
2.25
2
1.75
1.5
1.25
1
0.75
0.5
0.25
0
petal width
1
2
3
4
5
6
petal length

Polynomial Classification

Educational widget that visually demonstrates classification in two classes for any classifier.

Signals

Inputs

- **Data**

Input data set.

- **Preprocessor**

Data preprocessors.

- **Learner**

Classification algorithm used in the widget. Default set to Logistic Regression Learner.

Outputs

- **Learner**

Classification algorithm used in the widget.

- **Classifier**

Trained classifier.

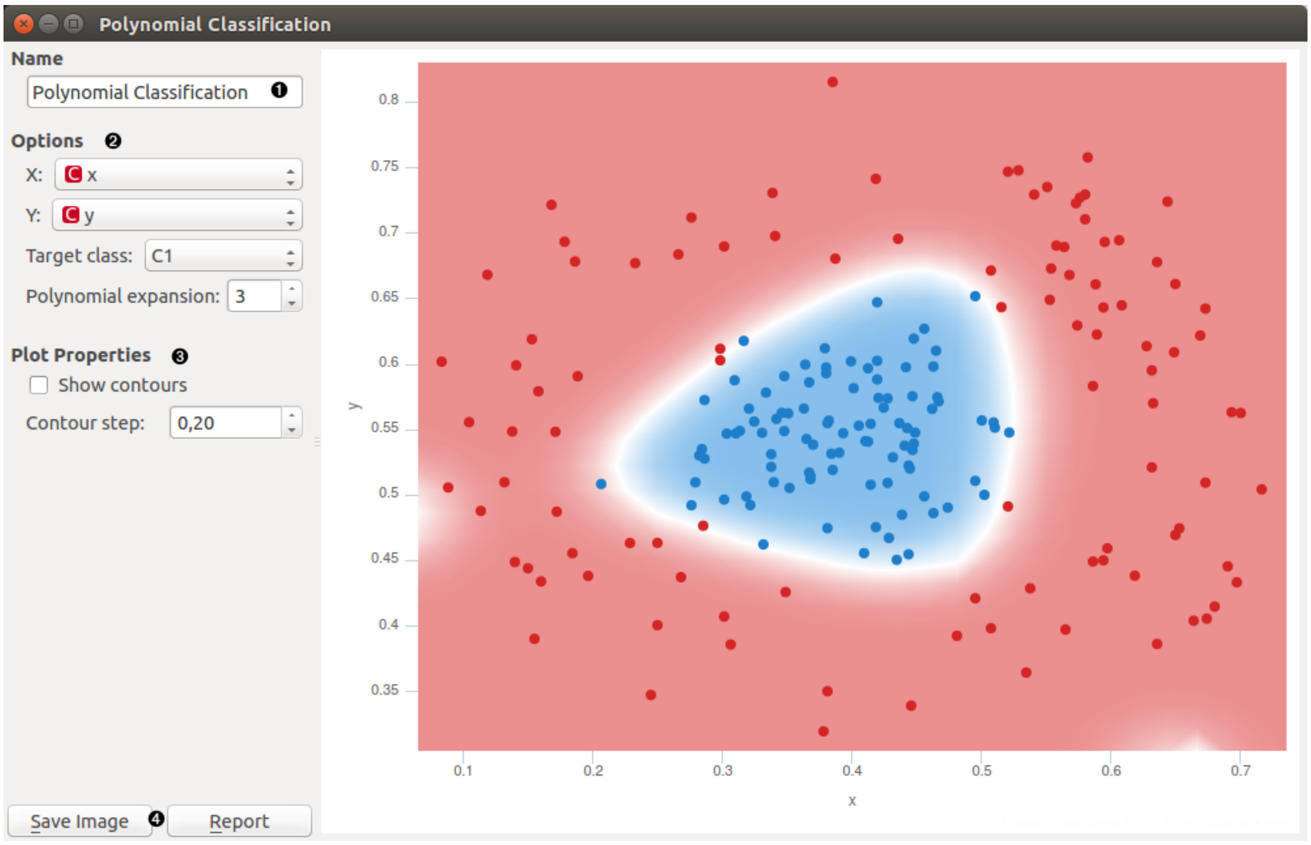
- **Coefficients**

Classifier coefficients if it has them.

Description

This widget interactively shows classification probabilities for classification in two classes using color gradient and contour lines for any classifiers form *Orange Classification* module. In the widget, polynomial expansion can be set. Polynomial expansion is a regulation of the degree of polynom that is used to transform the input data and has an effect on classification. If polynomial expansion is set to 1 it means that untransformed data are used in the regression. If polynomial expansion is set to 2 we get following additional attributes:

- first attribute on power 2
- first attribute * second attribute
- second attribute on power 2



1. Classifier name.

2. *X*: attribute on axis x.

   Y: attribute on axis y.

   Target class: Class in input data that is classified apart from others classes because widget support only two

       class classification.

   Polynomial expansion: Degree of polynom that is used to transform the input data.

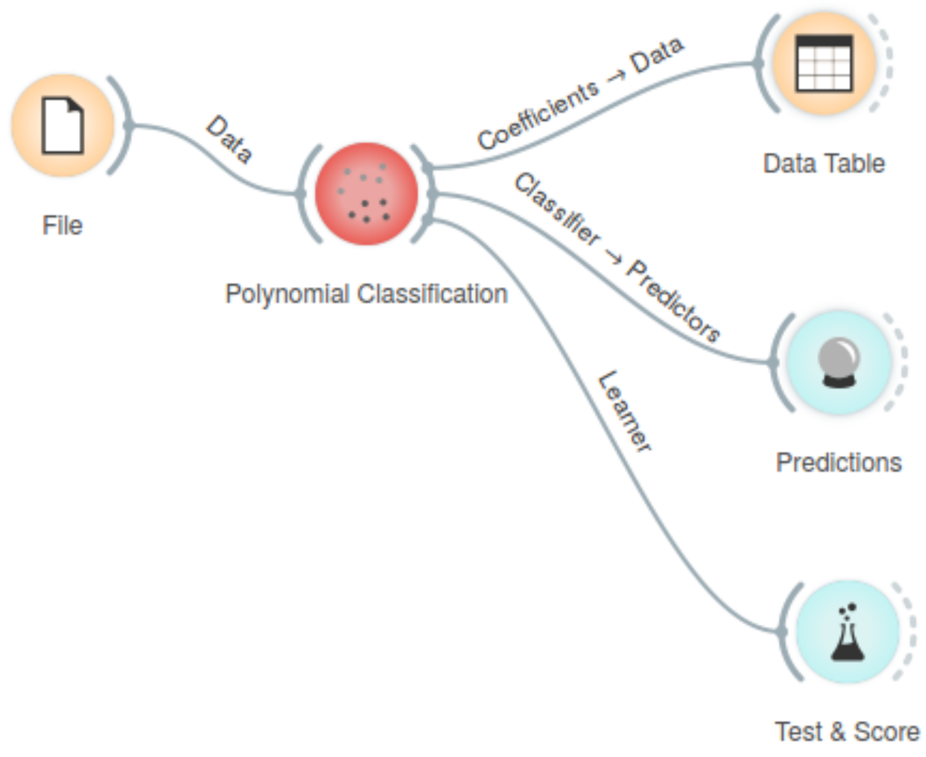
3. *Show contours*: Enable contour lines in the graph.

   Contour step: Density of contour lines.

4. *Save Image* saves the image to the computer in a .svg or .png format.

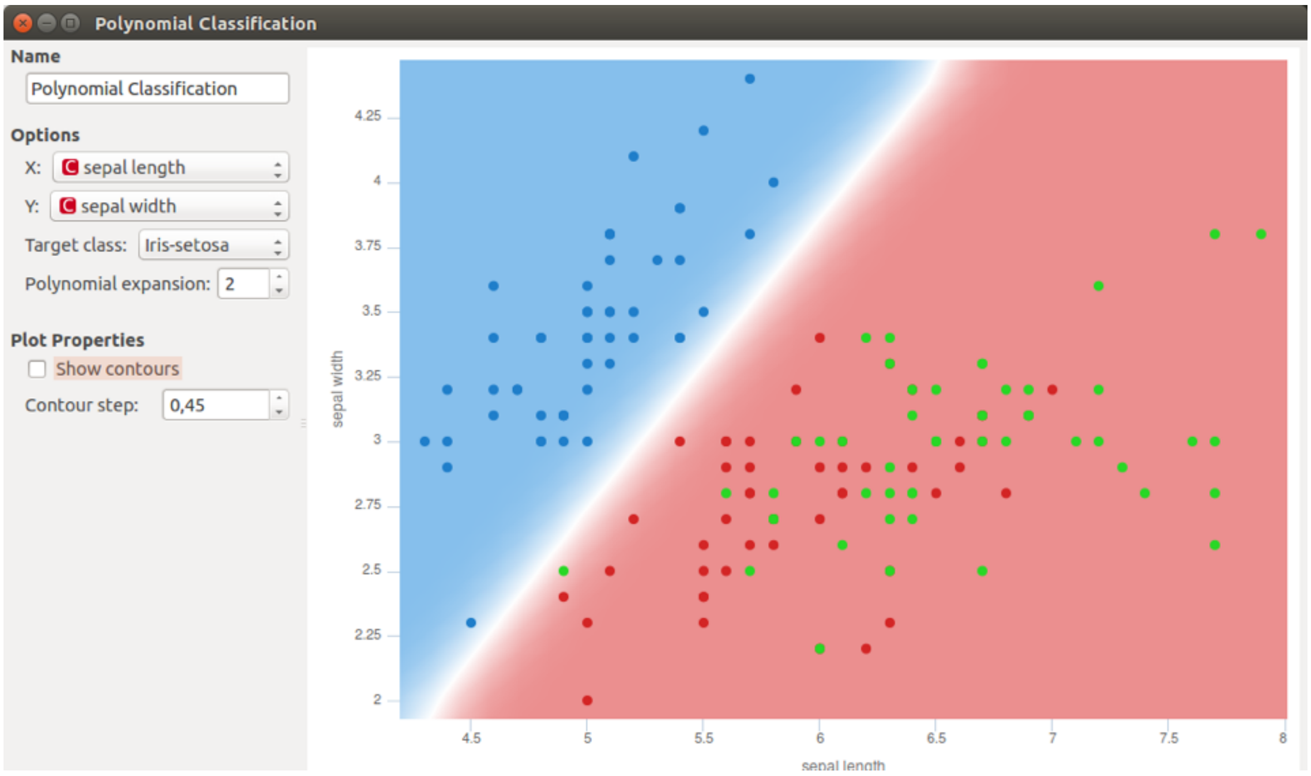
   Report includes widget parameters and visualization in the report.

Example



We loaded *iris* data set with the File widget and connected it to *Polynomial Classification* widget. To demonstrate outputs connections we connected *Coefficients* to Data Table widget where we can inspect their values. *Learner* output can be connected to *Test & Score* widget and *Classifier* to *Predictions widget*.

In the widget we selected *sepal length* as our *X* variable and *sepal width* as our *Y* variable. We set *Polynomial expansion* to 1. That performs classification on non transformed data. Result is show on the figure below. Color gradient represents the probability to classify data on its position in one of two classes. Blue color represents classification in target class and red color classification in class with all others examples.


Polynomial Classification

Name

Polynomial Classification

Options

X: sepal length

Y: sepal width

Target class: Iris-setosa

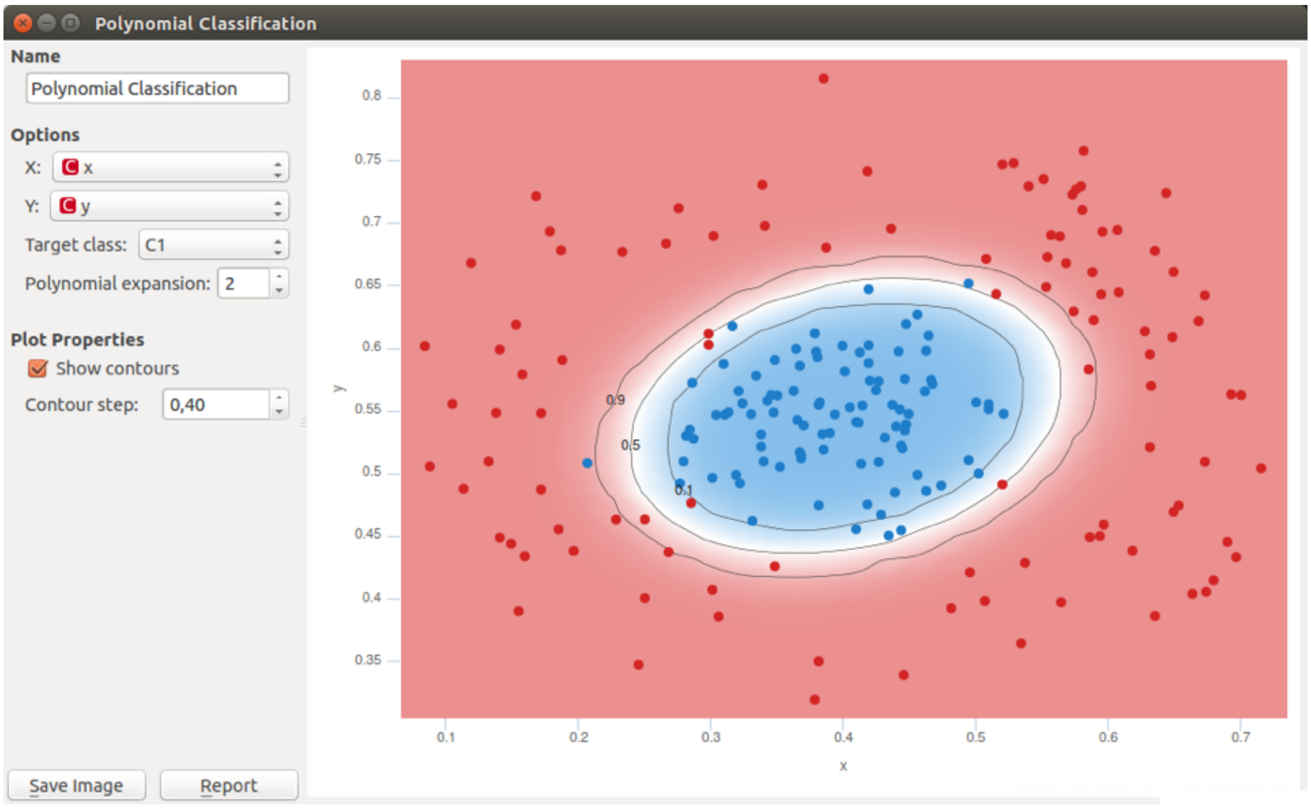
Polynomial expansion: 2

Plot Properties

Show contours

Contour step: 0,45

In next example we changed *File* widget with *Paint data* widget and plotted some custom data. Because center of data has one class and surrounding another *Polynomial expansion* degree 1 does not perform good classification. We set *Polynomial expansion* to 2 and got classification in figure below. We also selected to use contour lines.


Polynomial Classification

Name

Polynomial Classification

Options

X: x

Y: y

Target class: C1

Polynomial expansion: 2

Plot Properties

[✓ Show contours]

Contour step: 0,40

Polynomial Regression

Educational widget that interactively shows regression line for different regressors.

Signals

Inputs:

- **Data**

Input data set. It needs at least two continuous attributes.

- **Preprocessor**

Data preprocessors.

- **Learner**

Regression algorithm used in the widget. Default set to Linear Regression.

Outputs:

- **Learner**

Regression algorithm used in the widget.

- **Predictor**

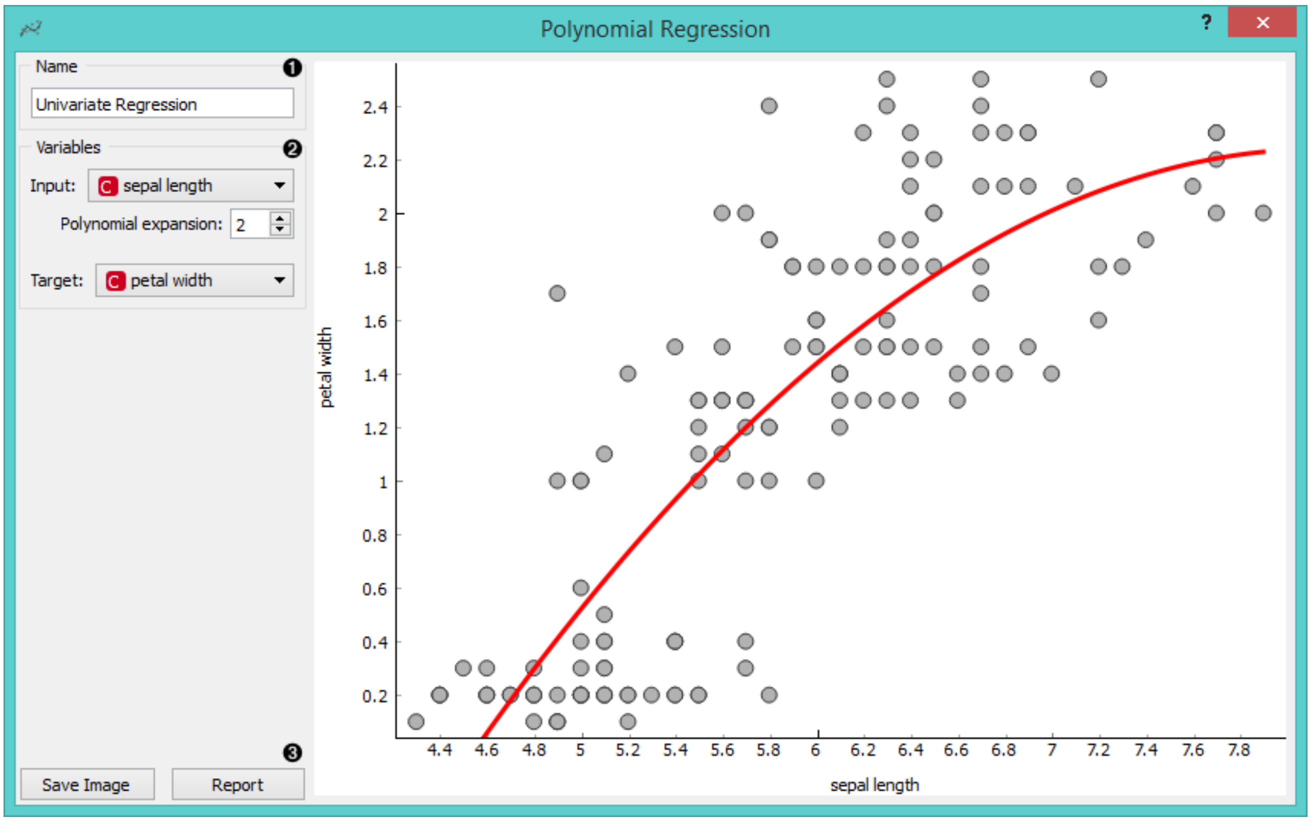
Trained regressor.

- **Coefficients**

Regressor coefficients if it has them.

Description

This widget interactively shows regression line using any of the regressors from *Orange3 Regression* module. In the widget, polynomial expansion can be set. Polynomial expansion is a regulation of the degree of polynom that is used to transform the input data and has an effect on the shape of a curve. If polynomial expansion is set to 1 it means that untransformed data are used in the regression.



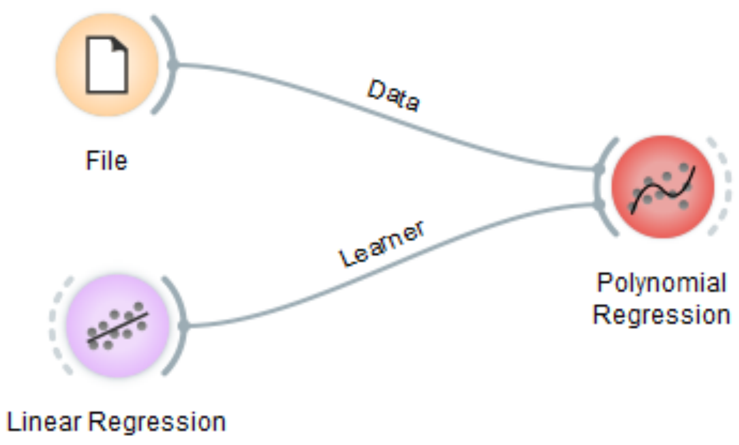
1. Regressor name.
2. *Input*: independent variable on axis x.

   Polynomial expansion: degree of polynomial expansion.

   Target: dependent variable on axis y.
3. *Save Image* saves the image to the computer in a .svg or .png format.

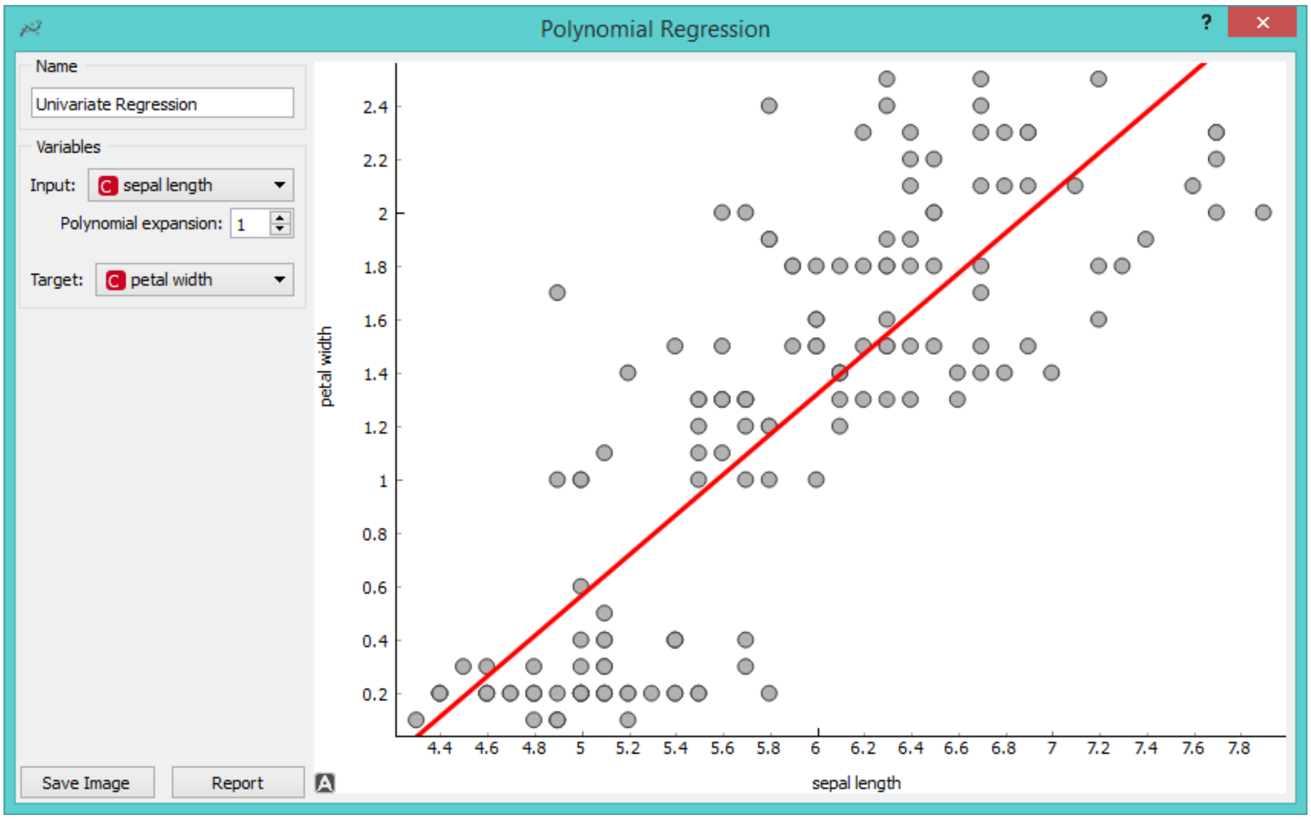
   Report includes widget parameters and visualization in the report.

Example



We loaded *iris* data set with the **File** widget. Then we connected **Linear Regression** learner to the

Polynomial Regression widget. In the widget we selected *petal length* as our *Input* variable and *petal width* as our *Target* variable. We set *Polynomial expansion* to 1 which gives us a linear regression line. The result is shown on the figure below.


Polynomial Regression

Name: Univariate Regression

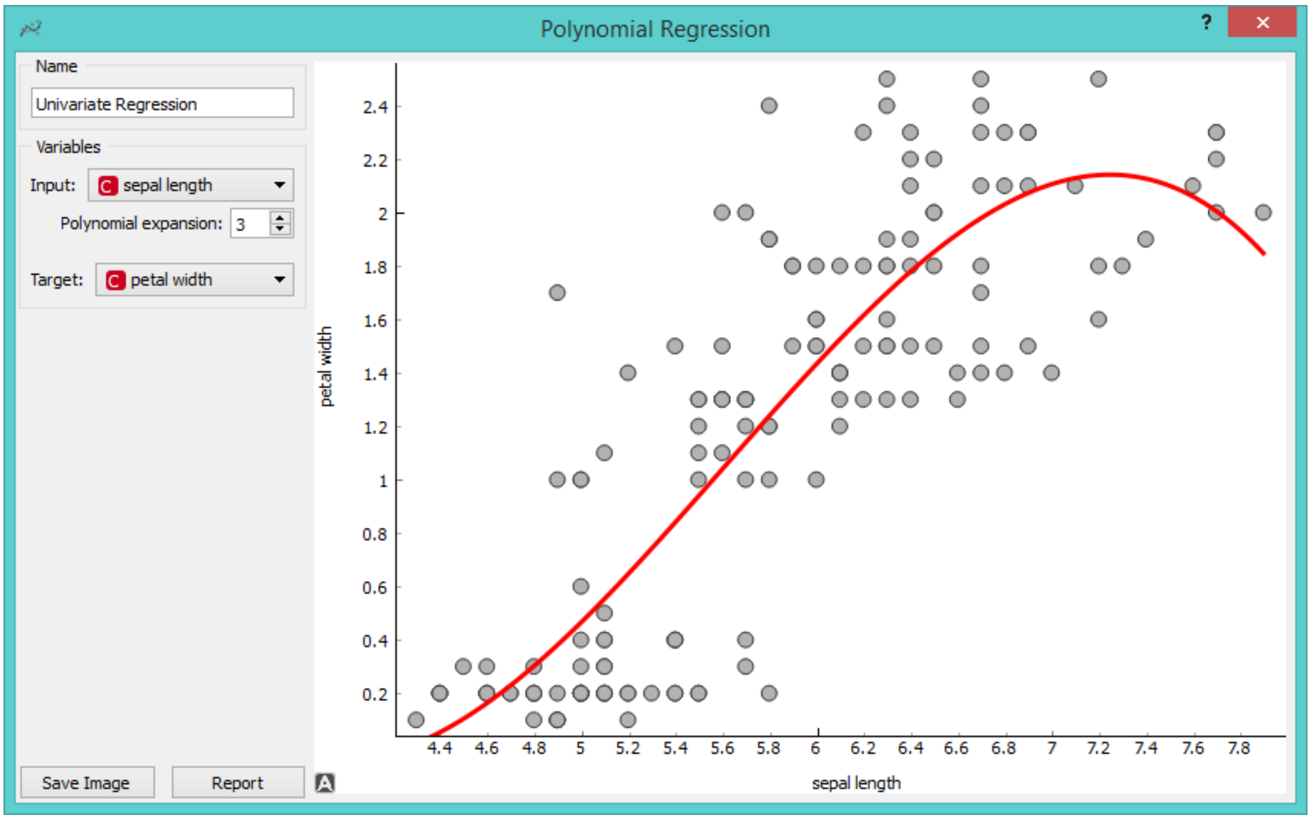
Variables

Input: sepal length

Polynomial expansion: 1

Target: petal width

The line can fit better if we increase the **Polynomial expansion** parameter. Say, we set it to 3.


Polynomial Regression

Name: Univariate Regression

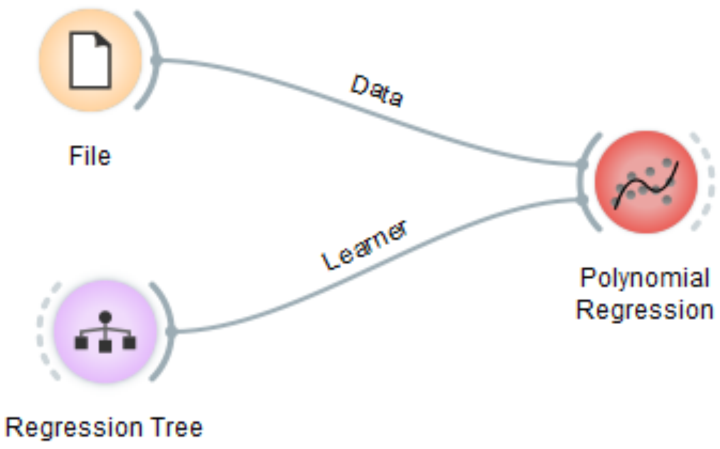
Variables

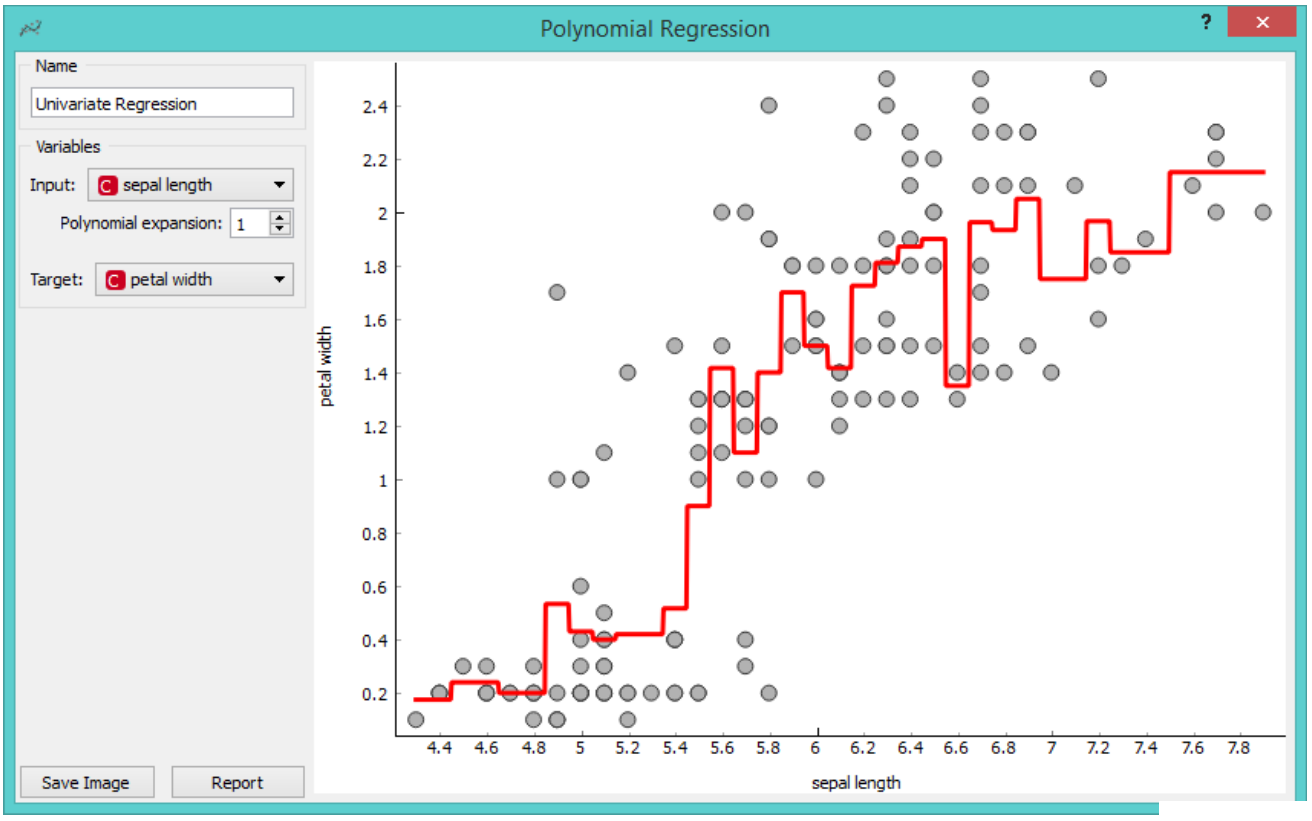
Input: sepal length

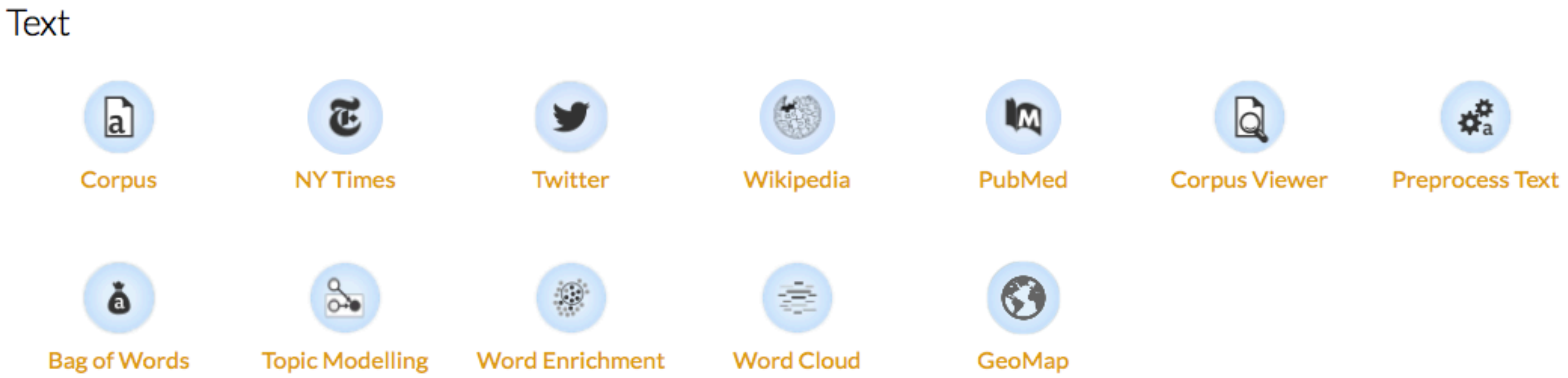
Polynomial expansion: 3

Target: petal width

To observe different results, change **Linear Regression** to any other regression learner from Orange. Example below is done with **Regression Tree** learner.


File
Data
Learner
Polynomial Regression
Regression Tree


Polynomial Regression
Name
Univariate Regression
Variables
Input: sepal length
Polynomial expansion: 1
Target: petal width
Save Image
Report
petal width
sepal length


Text
Corpus
NY Times
Twitter
Wikipedia
PubMed
Corpus Viewer
Preprocess Text
Bag of Words
Topic Modelling
Word Enrichment
Word Cloud
GeoMap

Bag of Words

Generates a bag of words from the input corpus.

Signals

Inputs:

- **Corpus**

  Corpus instance.

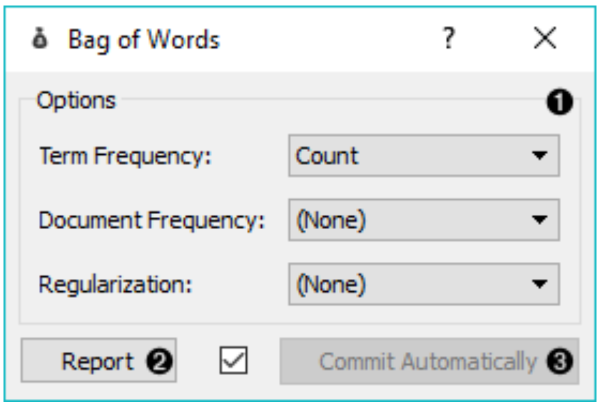
Outputs:

- **Corpus**

  Corpus with bag of words.

Description

Bag of Words model creates a corpus with word counts for each data instance (document). The count can be either absolute, binary (contains or does not contain) or sublinear (logarithm of the term frequency). Bag of words model is required in combination with Word Enrichment and could be used for predictive modelling.



Term Frequency: Count
Document Frequency: (None)
Regularization: (None)

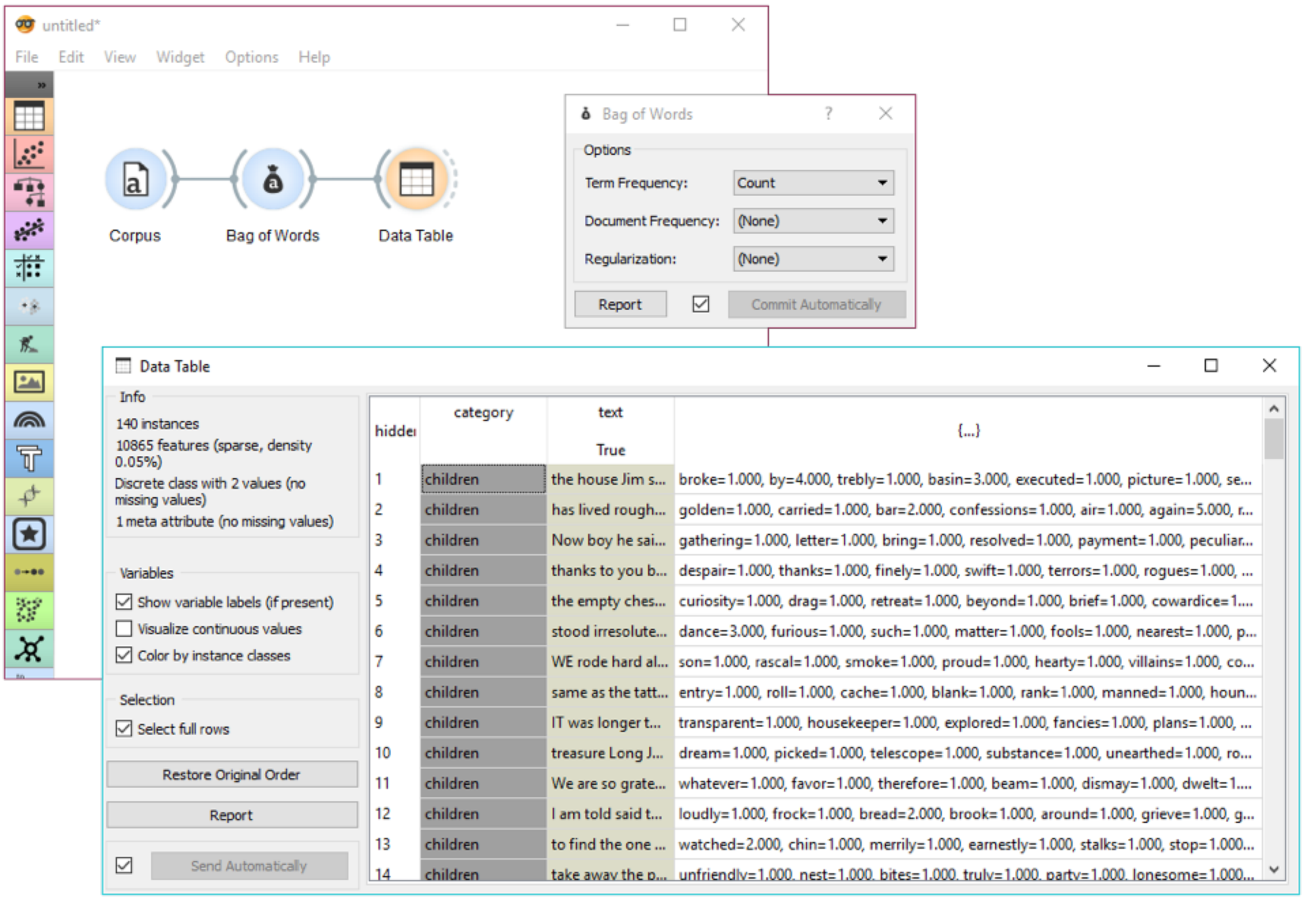
1. Parameters for bag of words model:
   - Term Frequency:
     - Count: number of occurences of a word in a document
     - Binary: word appears or does not appear in the document
     - Sublinear: logarithm of term frequency (count)

- Document Frequency:
    - (None)
    - IDF: inverse document frequency
    - Smooth IDF: adds one to document frequencies to prevent zero division.
- Regulariation:
    - (None)
    - L1 (Sum of elements): normalizes vector length to sum of elements
    - L2 (Euclidean): normalizes vector length to sum of squares

2. Produce a report.
3. If *Commit Automatically* is on, changes are communicated automatically. Alternatively press *Commit*.

Example

In the first example we will simply check how the bag of words model looks like. Load *book-excerpt-s.tab* with Corpus widget and connect it to **Bag of Words**. Here we kept the defaults - a simple count of term frequencies. Check what the **Bag of Words** outputs with **Data Table**. The final column in white represents term frequencies for each document.



Options
Term Frequency: Count
Document Frequency: (None)
Regularization: (None)

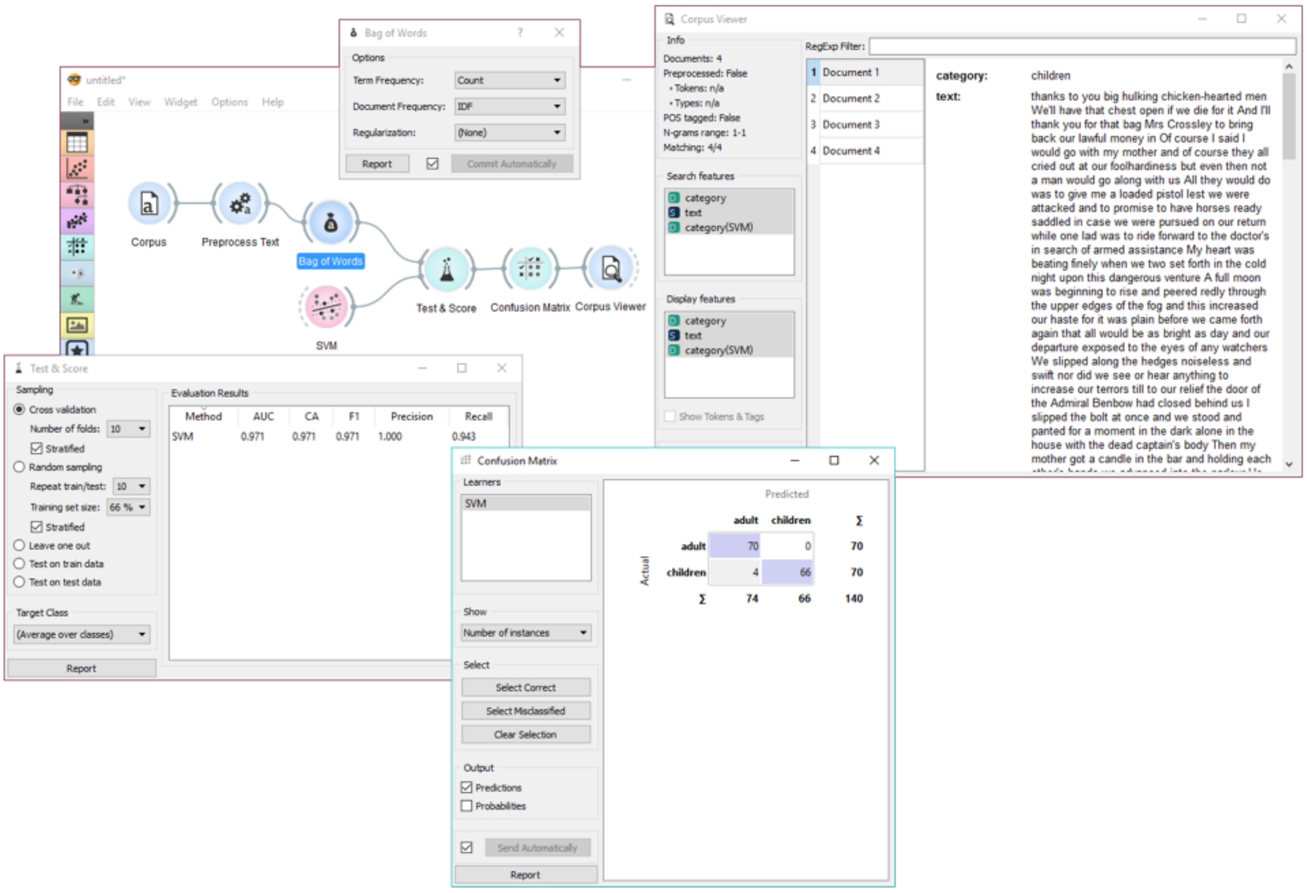
Info
140 instances
10865 features (sparse, density 0.05%)
Discrete class with 2 values (no missing values)
1 meta attribute (no missing values)

Variables
☑ Show variable labels (if present)
☐ Visualize continuous values
☑ Color by instance classes

Selection
☑ Select full rows

hidde	category	text True	{...}
1	children	the house Jim s...	broke=1.000, by=4.000, trebly=1.000, basin=3.000, executed=1.000, picture=1.000, se...
2	children	has lived rough...	golden=1.000, carried=1.000, bar=2.000, confessions=1.000, air=1.000, again=5.000, r...
3	children	Now boy he sai...	gathering=1.000, letter=1.000, bring=1.000, resolved=1.000, payment=1.000, peculiar...
4	children	thanks to you b...	despair=1.000, thanks=1.000, finely=1.000, swift=1.000, terrors=1.000, rogues=1.000, ...
5	children	the empty ches...	curiosity=1.000, drag=1.000, retreat=1.000, beyond=1.000, brief=1.000, cowardice=1....
6	children	stood irresolute...	dance=3.000, furious=1.000, such=1.000, matter=1.000, fools=1.000, nearest=1.000, p...
7	children	WE rode hard al...	son=1.000, rascal=1.000, smoke=1.000, proud=1.000, hearty=1.000, villains=1.000, co...
8	children	same as the tatt...	entry=1.000, roll=1.000, cache=1.000, blank=1.000, rank=1.000, manned=1.000, houn...
9	children	IT was longer t...	transparent=1.000, housekeeper=1.000, explored=1.000, fancies=1.000, plans=1.000, ...
10	children	treasure Long J...	dream=1.000, picked=1.000, telescope=1.000, substance=1.000, unearthed=1.000, ro...
11	children	We are so grate...	whatever=1.000, favor=1.000, therefore=1.000, beam=1.000, dismay=1.000, dwelt=1....
12	children	I am told said t...	loudly=1.000, frock=1.000, bread=2.000, brook=1.000, around=1.000, grieve=1.000, g...
13	children	to find the one ...	watched=2.000, chin=1.000, merrily=1.000, earnestly=1.000, stalks=1.000, stop=1.000...
14	children	take away the p...	unfriendly=1.000, nest=1.000, bites=1.000, truly=1.000, party=1.000, lonesome=1.000...

In the second example we will try to predict document category. We are still using the *book-excerpt-s.tab* data set, which we sent through Preprocess Text with default parameters. Then we connected **Preprocess Text** to **Bag of Words** to obtain term frequencies by which we will compute the model.



Bag of Words

Options

Term Frequency: Count

Document Frequency: IDF

Regularization: (None)

Report ☑ Commit Automatically

Corpus Viewer

Info

Documents: 4
Preprocessed: False
• Tokens: n/a
• Types: n/a
POS tagged: False
N-grams range: 1-1
Matching: 4/4

Search features: category, text, category(SVM)

Display features: category, text, category(SVM)

☐ Show Tokens & Tags

RegExp Filter:

1 Document 1
2 Document 2
3 Document 3
4 Document 4

category: children

text: thanks to you big hulking chicken-hearted men We'll have that chest open if we die for it And I'll thank you for that bag Mrs Crossley to bring back our lawful money in Of course I said I would go with my mother and of course they all cried out at our foolhardiness but even then not a man would go along with us All they would do was to give me a loaded pistol lest we were attacked and to promise to have horses ready saddled in case we were pursued on our return while one lad was to ride forward to the doctor's in search of armed assistance My heart was beating finely when we two set forth in the cold night upon this dangerous venture A full moon was beginning to rise and peered redly through the upper edges of the fog and this increased our haste for it was plain before we came forth again that all would be as bright as day and our departure exposed to the eyes of any watchers We slipped along the hedges noiseless and swift nor did we see or hear anything to increase our terrors till to our relief the door of the Admiral Benbow had closed behind us I slipped the bolt at once and we stood and panted for a moment in the dark alone in the house with the dead captain's body Then my mother got a candle in the bar and holding each

Test & Score

Sampling

(● Cross validation)
Number of folds: 10
☑ Stratified
○ Random sampling
Repeat train/test: 10
Training set size: 66 %
☑ Stratified
○ Leave one out
○ Test on train data
○ Test on test data

Target Class: (Average over classes)

Report

Evaluation Results

Method	AUC	CA	F1	Precision	Recall
SVM	0.971	0.971	0.971	1.000	0.943

Confusion Matrix

Learners: SVM

Show: Number of instances

Select: Select Correct / Select Misclassified / Clear Selection

Output
☑ Predictions
☐ Probabilities

☑ Send Automatically

Report

		Predicted		
		adult	children	Σ
Actual	adult	70	0	70
	children	4	66	70
	Σ	74	66	140

Connect **Bag of Words** to **Test & Score** for predictive modelling. Connect **SVM** or any other classifier to **Test & Score** as well (both on the left side). **Test & Score** will now compute performance scores for each learner on the input. Here we got quite impressive results with SVM. Now we can check, where the model made a mistake.

Add **Confusion Matrix** to **Test & Score**. Confusion matrix displays correctly and incorrectly classified documents. *Select Misclassified* will output misclassified documents, which we can further inspect with Corpus Viewer.

Corpus Viewer

Displays corpus content.

Signals

Inputs:

- **Corpus**

  Corpus instance.

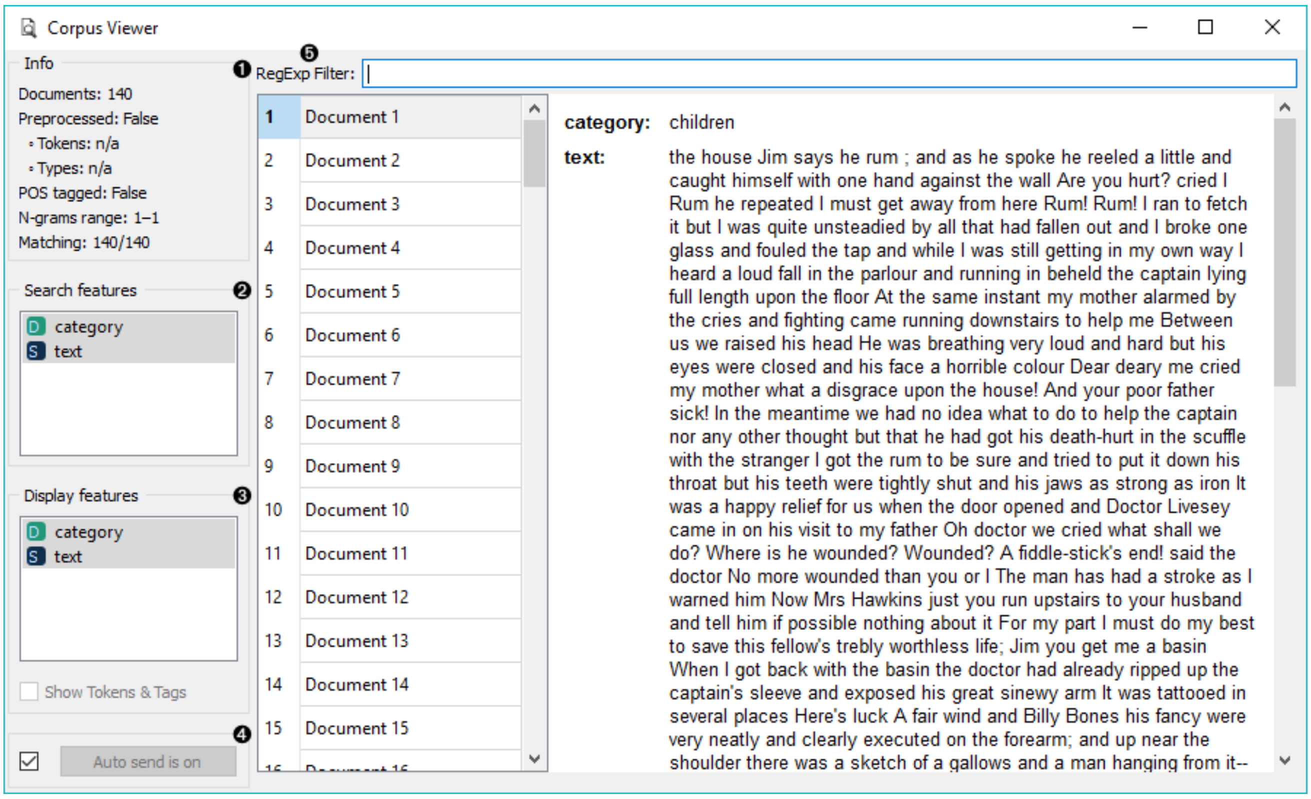
Outputs:

- **Corpus**

  A Corpus instance.

Description

Corpus Viewer is meant for viewing text files (instances of Corpus). It will always output an instance of corpus. If *RegExp* filtering is used, the widget will output only matching documents.



1	Document 1
2	Document 2
3	Document 3
4	Document 4
5	Document 5
6	Document 6
7	Document 7
8	Document 8
9	Document 9
10	Document 10
11	Document 11
12	Document 12
13	Document 13
14	Document 14
15	Document 15

category: children

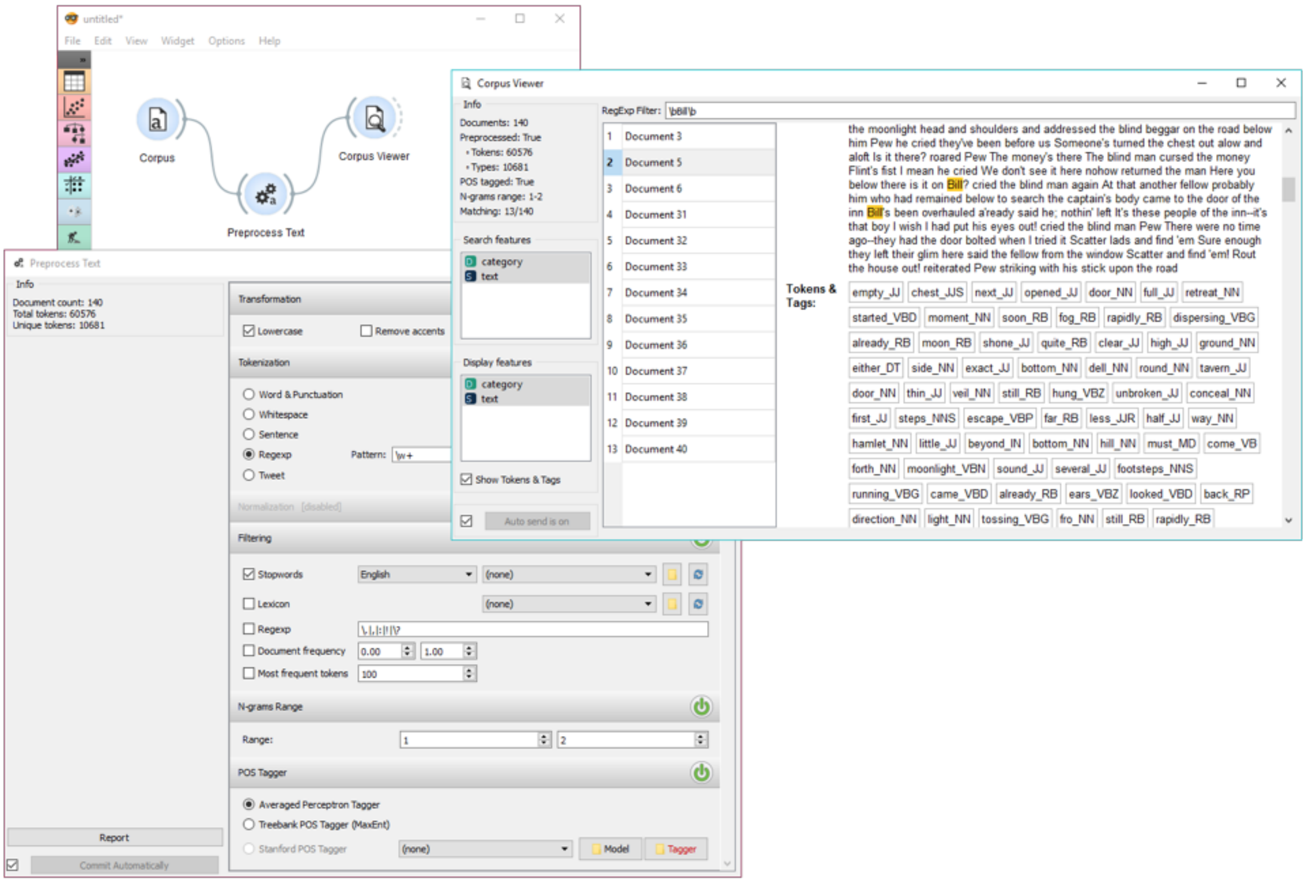
text: the house Jim says he rum ; and as he spoke he reeled a little and caught himself with one hand against the wall Are you hurt? cried I Rum he repeated I must get away from here Rum! Rum! I ran to fetch it but I was quite unsteadied by all that had fallen out and I broke one glass and fouled the tap and while I was still getting in my own way I heard a loud fall in the parlour and running in beheld the captain lying full length upon the floor At the same instant my mother alarmed by the cries and fighting came running downstairs to help me Between us we raised his head He was breathing very loud and hard but his eyes were closed and his face a horrible colour Dear deary me cried my mother what a disgrace upon the house! And your poor father sick! In the meantime we had no idea what to do to help the captain nor any other thought but that he had got his death-hurt in the scuffle with the stranger I got the rum to be sure and tried to put it down his throat but his teeth were tightly shut and his jaws as strong as iron It was a happy relief for us when the door opened and Doctor Livesey came in on his visit to my father Oh doctor we cried what shall we do? Where is he wounded? Wounded? A fiddle-stick's end! said the doctor No more wounded than you or I The man has had a stroke as I warned him Now Mrs Hawkins just you run upstairs to your husband and tell him if possible nothing about it For my part I must do my best to save this fellow's trebly worthless life; Jim you get me a basin When I got back with the basin the doctor had already ripped up the captain's sleeve and exposed his great sinewy arm It was tattooed in several places Here's luck A fair wind and Billy Bones his fancy were very neatly and clearly executed on the forearm; and up near the shoulder there was a sketch of a gallows and a man hanging from it--

1. *Information*:
   - *Documents*: number of documents on the input
   - *Preprocessed*: if preprocessor is used, the result is True, else False. Reports also on the number of tokens and types (unique tokens).
   - *POS tagged*: if POS tags are on the input, the result is True, else False.
   - *N-grams range*: if N-grams are set in Preprocess Text, results are reported, default is 1-1 (one-grams).
   - *Matching*: number of documents matching the *RegExp Filter*. All documents are output by default.
2. *RegExp Filter*: Python regular expression for filtering documents. By default no documents are filtered (entire corpus is on the output).
3. *Search Features*: features by which the RegExp Filter is filtering. Use Ctrl (Cmd) to select multiple features.
4. *Display Features*: features that are displayed in the viewer. Use Ctrl (Cmd) to select multiple features.
5. *Show Tokens & Tags*: if tokens and POS tag are present on the input, you can check this box to display them.
6. If *Auto commit is on*, changes are communicated automatically. Alternatively press *Commit*.

Example

Corpus Viewer can be used for displaying all or some documents in corpus. In this example, we will first load *book-excerpts.tab*, that already comes with the add-on, into Corpus widget. Then we will preprocess the text into words, filter out the stopwords, create bi-grams and add POS tags (more on preprocessing in Preprocess Text). Now we want to see the results of preprocessing. In *Corpus Viewer* we can see, how many unique tokens we got and what they are (tick *Show Tokens & Tags*). Since we used also POS tagger to show part-of-speech labels, they will be displayed alongside tokens underneath the text.

Now we will filter out just the documents talking about a character Bill. We use regular expression *\bBill\b* to find the documents containing only the word Bill. You can output matching or non-matching documents, view them in another *Corpus Viewer* or further analyse them.



Preprocess Text

Info
Document count: 140
Total tokens: 60576
Unique tokens: 10681

Transformation
[✓] Lowercase
[ ] Remove accents

Tokenization
( ) Word & Punctuation
( ) Whitespace
( ) Sentence
(●) Regexp   Pattern: \w+
( ) Tweet

Normalization [disabled]

Filtering
[✓] Stopwords   English   (none)
[ ] Lexicon   (none)
[ ] Regexp   \.|,|:|!|\?
[ ] Document frequency   0.00   1.00
[ ] Most frequent tokens   100

N-grams Range
Range: 1   2

POS Tagger
(●) Averaged Perceptron Tagger
( ) Treebank POS Tagger (MaxEnt)
( ) Stanford POS Tagger   (none)   Model   Tagger

Report
[✓] Commit Automatically

Corpus Viewer

Info
Documents: 140
Preprocessed: True
• Tokens: 60576
• Types: 10681
POS tagged: True
N-grams range: 1-2
Matching: 13/140

Search features
category
text

Display features
category
text

[✓] Show Tokens & Tags

[✓] Auto send is on

RegExp Filter: \bBill\b

1	Document 3
2	Document 5
3	Document 6
4	Document 31
5	Document 32
6	Document 33
7	Document 34
8	Document 35
9	Document 36
10	Document 37
11	Document 38
12	Document 39
13	Document 40

the moonlight head and shoulders and addressed the blind beggar on the road below him Pew he cried they've been before us Someone's turned the chest out alow and aloft Is it there? roared Pew The money's there The blind man cursed the money Flint's fist I mean he cried We don't see it here nohow returned the man Here you below there is it on Bill? cried the blind man again At that another fellow probably him who had remained below to search the captain's body came to the door of the inn Bill's been overhauled a'ready said he; nothin' left It's these people of the inn--it's that boy I wish I had put his eyes out! cried the blind man Pew There were no time ago--they had the door bolted when I tried it Scatter lads and find 'em Sure enough they left their glim here said the fellow from the window Scatter and find 'em! Rout the house out! reiterated Pew striking with his stick upon the road

Tokens & Tags:
empty_JJ chest_JJS next_JJ opened_JJ door_NN full_JJ retreat_NN
started_VBD moment_NN soon_RB fog_RB rapidly_RB dispersing_VBG
already_RB moon_RB shone_JJ quite_RB clear_JJ high_JJ ground_NN
either_DT side_NN exact_JJ bottom_NN dell_NN round_NN tavern_JJ
door_NN thin_JJ veil_NN still_RB hung_VBZ unbroken_JJ conceal_NN
first_JJ steps_NNS escape_VBP far_RB less_JJR half_JJ way_NN
hamlet_NN little_JJ beyond_IN bottom_NN hill_NN must_MD come_VB
forth_NN moonlight_VBN sound_JJ several_JJ footsteps_NNS
running_VBG came_VBD already_RB ears_VBZ looked_VBD back_RP
direction_NN light_NN tossing_VBG fro_NN still_RB rapidly_RB

v: latest

Corpus

Load a corpus of text documents, (optionally) tagged with categories.

Signals

Inputs:

- (None)

Outputs:

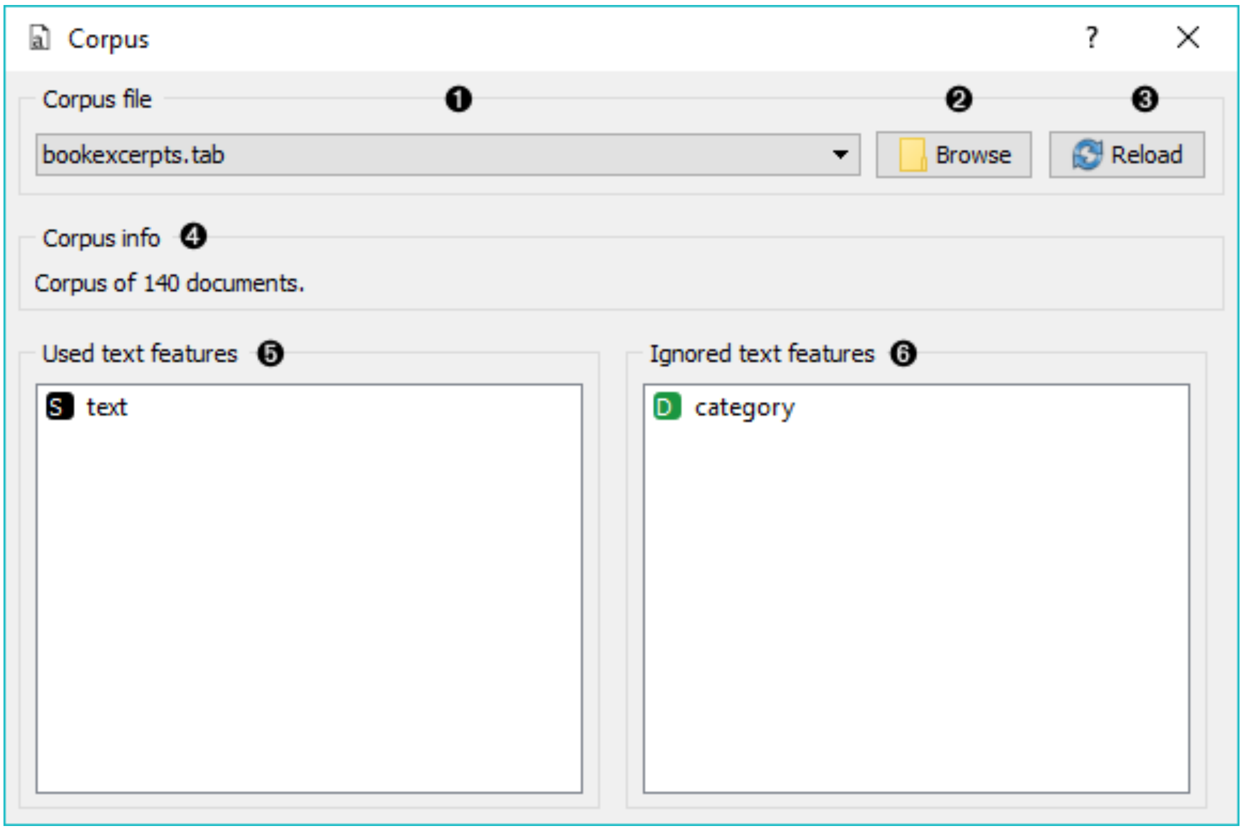
- **Corpus**

A Corpus instance.

Description

Corpus widget reads text corpora from files and sends a corpus instance to its output channel. History of the most recently opened files is maintained in the widget. The widget also includes a directory with sample corpora that come pre-installed with the add-on.

The widget reads data from Excel (**.xlsx**), comma-separated (**.csv**) and native tab-delimited (**.tab**) files.

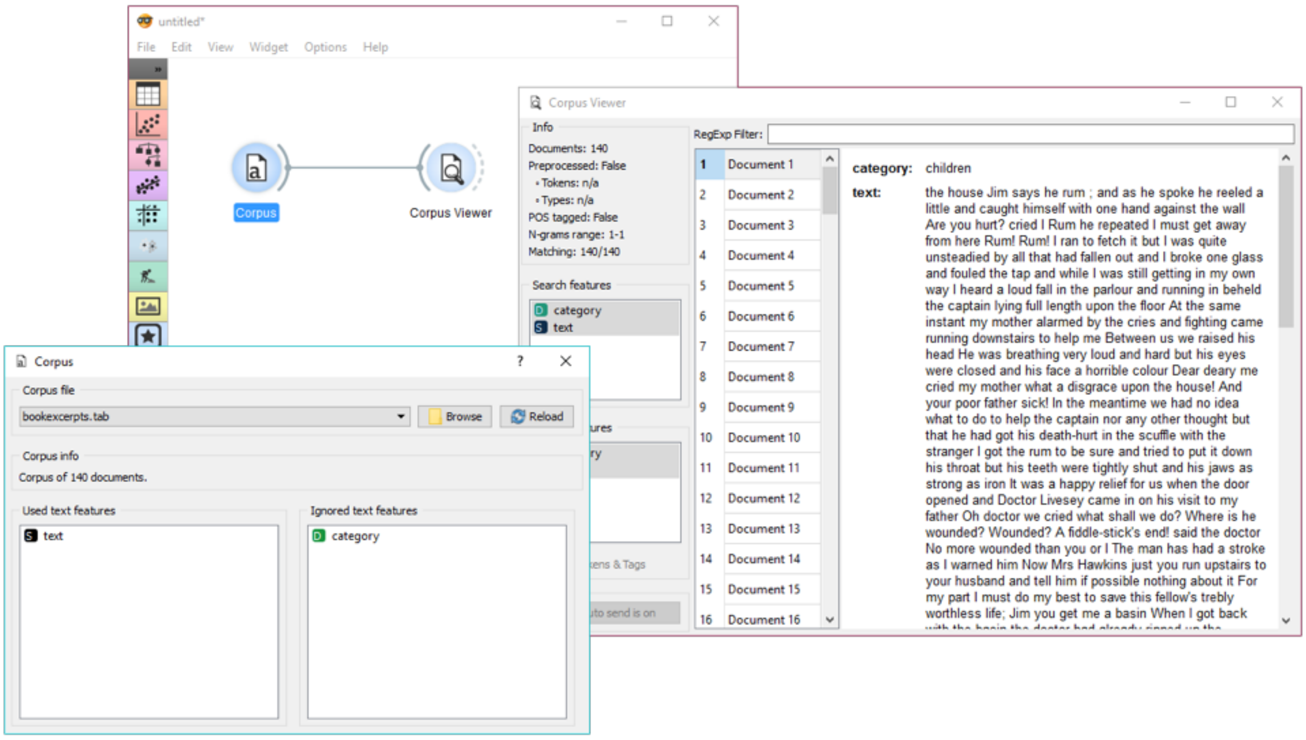


1. Browse through previously opened data files, or load any of the sample ones.
2. Browse for a data file.
3. Reloads currently selected data file.
4. Information on the loaded data set.
5. Features that will be used in text analysis.
6. Features that won't be used in text analysis and serve as labels or class.

You can drag and drop features between the two boxes and also change the order in which they appear.

Example

The first example shows a very simple use of **Corpus** widget. Place **Corpus** onto canvas and connect it to Corpus Viewer. We've used *booxexcerpts.tab* data set, which comes with the add-on, and inspected it in **Corpus Viewer**.



Corpus Viewer

Info
Documents: 140
Preprocessed: False
• Tokens: n/a
• Types: n/a
POS tagged: False
N-grams range: 1-1
Matching: 140/140

Search features
category
text

RegExp Filter:

1	Document 1
2	Document 2
3	Document 3
4	Document 4
5	Document 5
6	Document 6
7	Document 7
8	Document 8
9	Document 9
10	Document 10
11	Document 11
12	Document 12
13	Document 13
14	Document 14
15	Document 15
16	Document 16

category: children

text: the house Jim says he rum ; and as he spoke he reeled a little and caught himself with one hand against the wall Are you hurt? cried I Rum he repeated I must get away from here Rum! Rum! I ran to fetch it but I was quite unsteadied by all that had fallen out and I broke one glass and fouled the tap and while I was still getting in my own way I heard a loud fall in the parlour and running in beheld the captain lying full length upon the floor At the same instant my mother alarmed by the cries and fighting came running downstairs to help me Between us we raised his head He was breathing very loud and hard but his eyes were closed and his face a horrible colour Dear deary me cried my mother what a disgrace upon the house! And your poor father sick! In the meantime we had no idea what to do to help the captain nor any other thought but that he had got his death-hurt in the scuffle with the stranger I got the rum to be sure and tried to put it down his throat but his teeth were tightly shut and his jaws as strong as iron It was a happy relief for us when the door opened and Doctor Livesey came in on his visit to my father Oh doctor we cried what shall we do? Where is he wounded? Wounded? A fiddle-stick's end! said the doctor No more wounded than you or I The man has had a stroke as I warned him Now Mrs Hawkins just you run upstairs to your husband and tell him if possible nothing about it For my part I must do my best to save this fellow's trebly worthless life; Jim you get me a basin When I got back

Corpus

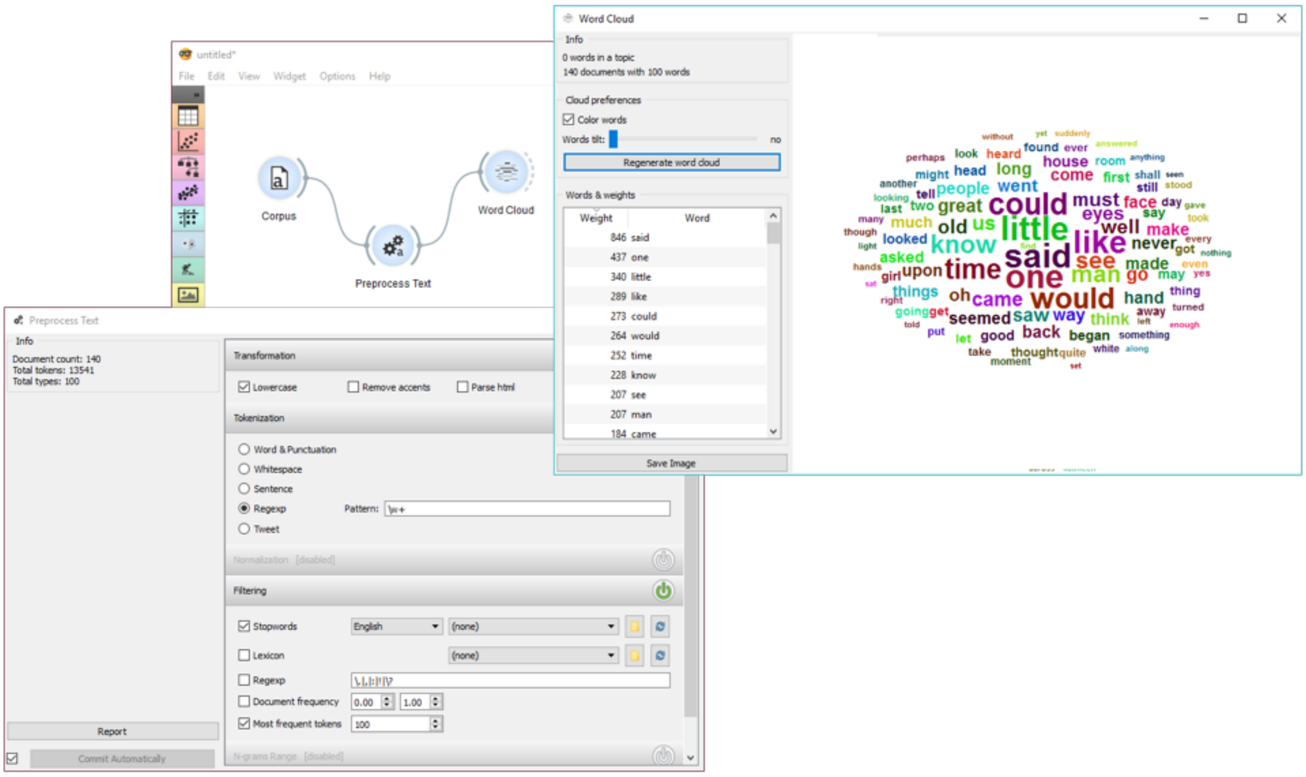
Corpus file
bookexcerpts.tab
Browse Reload

Corpus info
Corpus of 140 documents.

Used text features
text

Ignored text features
category

The second example demonstrates how to quickly visualize your corpus with Word Cloud. We could connect **Word Cloud** directly to **Corpus**, but instead we decided to apply some preprocessing with Preprocess Text. We are again working with *book-excerpts.tab*. We've put all text to lowercase, tokenized (split) the text to words only, filtered out English stopwords and selected a 100 most frequent tokens.


untitled*
File Edit View Widget Options Help
Corpus
Preprocess Text
Word Cloud
without yet suddenly perhaps look heard found ever answered anything might head long house room another come first shall seen tell people went still stood looking last two great could must face day gave many much eyes say took though old us little well make looked know like never every light asked said got nothing hands girl upon time see made even sat one man go may yes right things oh came would hand thing going get seemed saw way think away turned told put let good back began something take thought quite white along moment set

Word Cloud

Info
0 words in a topic
140 documents with 100 words

Cloud preferences
☑ Color words
Words tilt: no
Regenerate word cloud

Words & weights

Weight	Word
846	said
437	one
340	little
289	like
273	could
264	would
252	time
228	know
207	see
207	man
184	came

Save Image

Preprocess Text

Info
Document count: 140
Total tokens: 13541
Total types: 100

Transformation
☑ Lowercase ☐ Remove accents ☐ Parse html

Tokenization
○ Word & Punctuation
○ Whitespace
○ Sentence
● Regexp Pattern: \w+
○ Tweet

Normalization [disabled]

Filtering
☑ Stopwords English (none)
☐ Lexicon (none)
☐ Regexp \.|,|:|!|\?
☐ Document frequency 0.00 1.00
☑ Most frequent tokens 100

N-grams Range [disabled]

Report
☑ Commit Automatically

GeoMap

Displays geographic distribution of data.

Signals

Inputs:

- **Data**

  Data set.

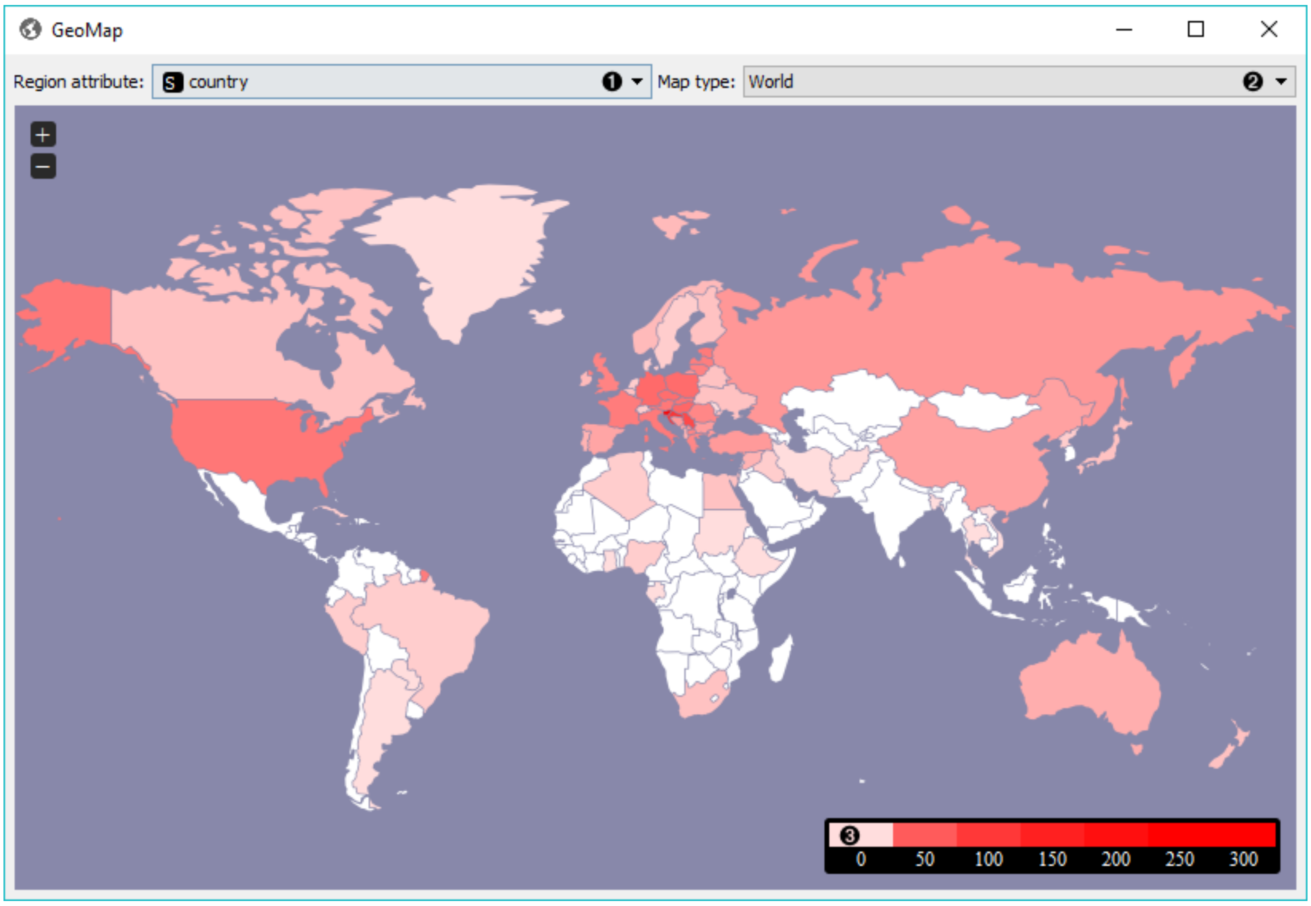
Outputs:

- **Corpus**

  A Corpus instance.

Description

GeoMap widget shows geolocations from textual (string) data. It finds mentions of geographic names (countries and capitals) and displays distributions (frequency of mentiones) of these names on a map. It works with any Orange widget that outputs a data table and that contains at least one string attribute. The widget outputs selected data instances, that is all documents containing mentions of a selected country (or countries).

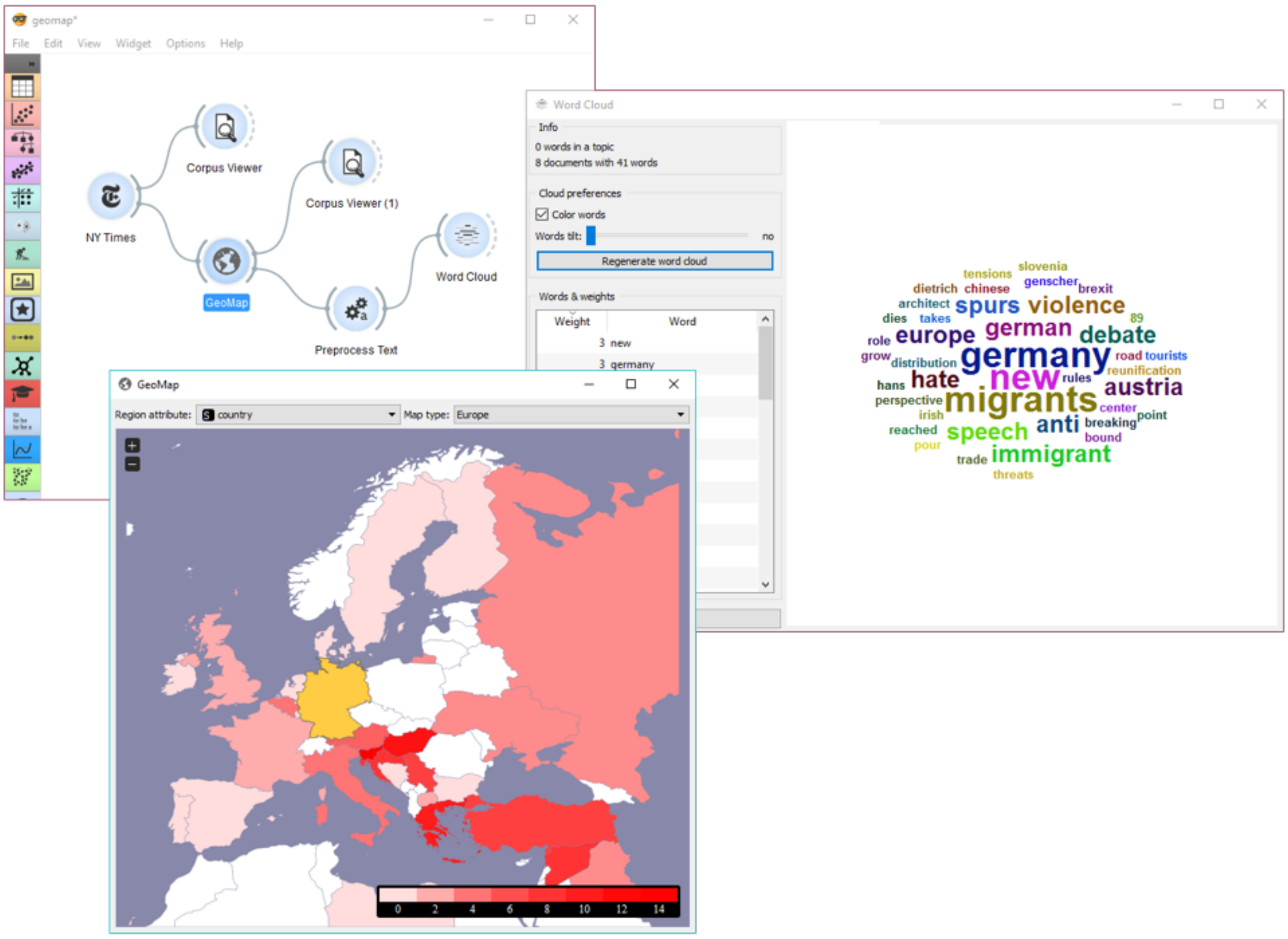


1. Select the meta attribute you want to search geolocations by. The widget will find all mentions of geolocations in a text and display distributions on a map.
2. Select the type of map you wish to display. The options are *World*, *Europe* and *USA*. You can zoom in and out of the map by pressing + and - buttons on a map or by mouse scroll.
3. The legend for the geographic distribution of data. Countries with the boldest color are most often mentioned in the selected region attribute (highest frequency).

To select documents mentioning a specific country, click on a country and the widget will output matching documents. To select more than one country hold Ctrl/Cmd upon selection.

Example

GeoMap widget can be used for simply visualizing distributions of geolocations or for a more complex interactive data analysis. Here, we've queried NY Times for articles on Slovenia for the time period of the last year (2015-2016). First we checked the results with Corpus Viewer.


geomap*
File Edit View Widget Options Help
NY Times
Corpus Viewer
Corpus Viewer (1)
GeoMap
Preprocess Text
Word Cloud
tensions slovenia dietrich chinese genscher brexit architect spurs violence dies takes 89 role europe german debate grow distribution germany road tourists reunification hans hate new rules austria perspective migrants center irish breaking point reached speech anti bound pour trade immigrant threats
0 2 4 6 8 10 12 14

Word Cloud

Info
0 words in a topic
8 documents with 41 words

Cloud preferences
☑ Color words
Words tilt: no
Regenerate word cloud

Words & weights

Weight	Word
3	new
3	germany

GeoMap

Region attribute: country Map type: Europe

Then we sent the data to **GeoMap** to see distributiosn of geolocations by *country* attribute. The attribute already contains country tags for each article, which is why **NY Times** is great in combinations with **GeoMap**. We selected Germany, which sends all the documents tagged with Germany to the output. Remember, we queried **NY Times** for articles on Slovenia.

We can again inspect the output with **Corpus Viewer**. But there's a more interesting way of visualizing the data. We've sent selected documents to Preprocess Text, where we've tokenized text to words and removed stopwords.

Finally, we can inspect the top words appearing in last year's documents on Slovenia and mentioning also Germany with Word Cloud.

NY Times

Loads data from the New York Times' Article Search API.

Signals

Inputs:

- (None)

Outputs:

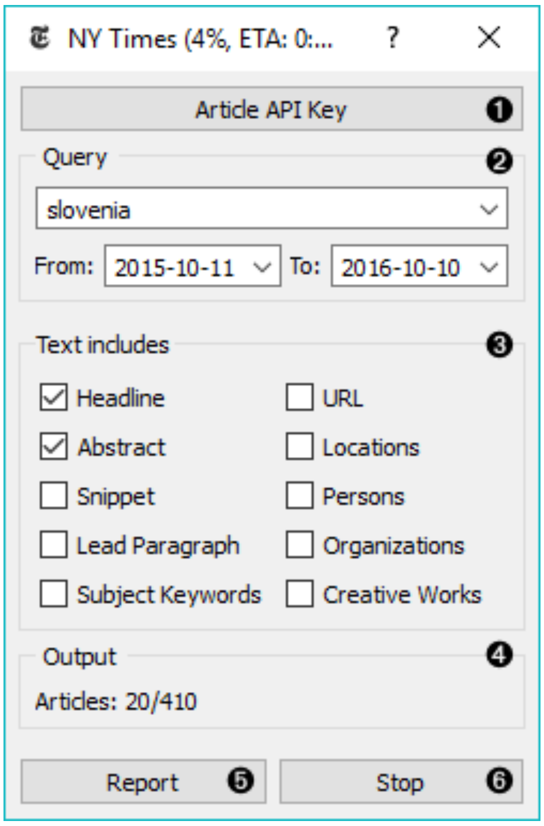
- **Corpus**

  A Corpus instance.

Description

NYTimes widget loads data from New York Times' Article Search API. You can query NYTimes articles from September 18, 1851 to today, but the API limit is set to allow retrieving only a 1000 documents per query. Define which features to use for text mining, *Headline* and *Abstract* being selected by default.

To use the widget, you must enter your own API key.


NY Times (4%, ETA: 0:...

Article API Key ❶

Query ❷

slovenia

From: 2015-10-11 To: 2016-10-10

Text includes ❸

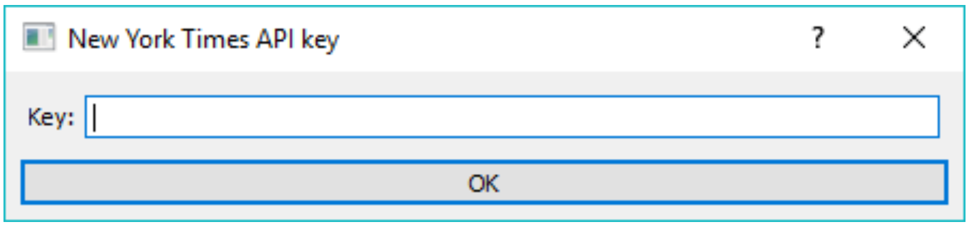
- [✓] Headline
- [✓] Abstract
- [ ] Snippet
- [ ] Lead Paragraph
- [ ] Subject Keywords
- [ ] URL
- [ ] Locations
- [ ] Persons
- [ ] Organizations
- [ ] Creative Works

Output ❹

Articles: 20/410

Report ❺ Stop ❻

1. To begin your query, insert NY Times' Article Search API key. The key is securely saved in your system keyring service (like Credential Vault, Keychain, KWallet, etc.) and won't be deleted when clearing widget settings.


New York Times API key

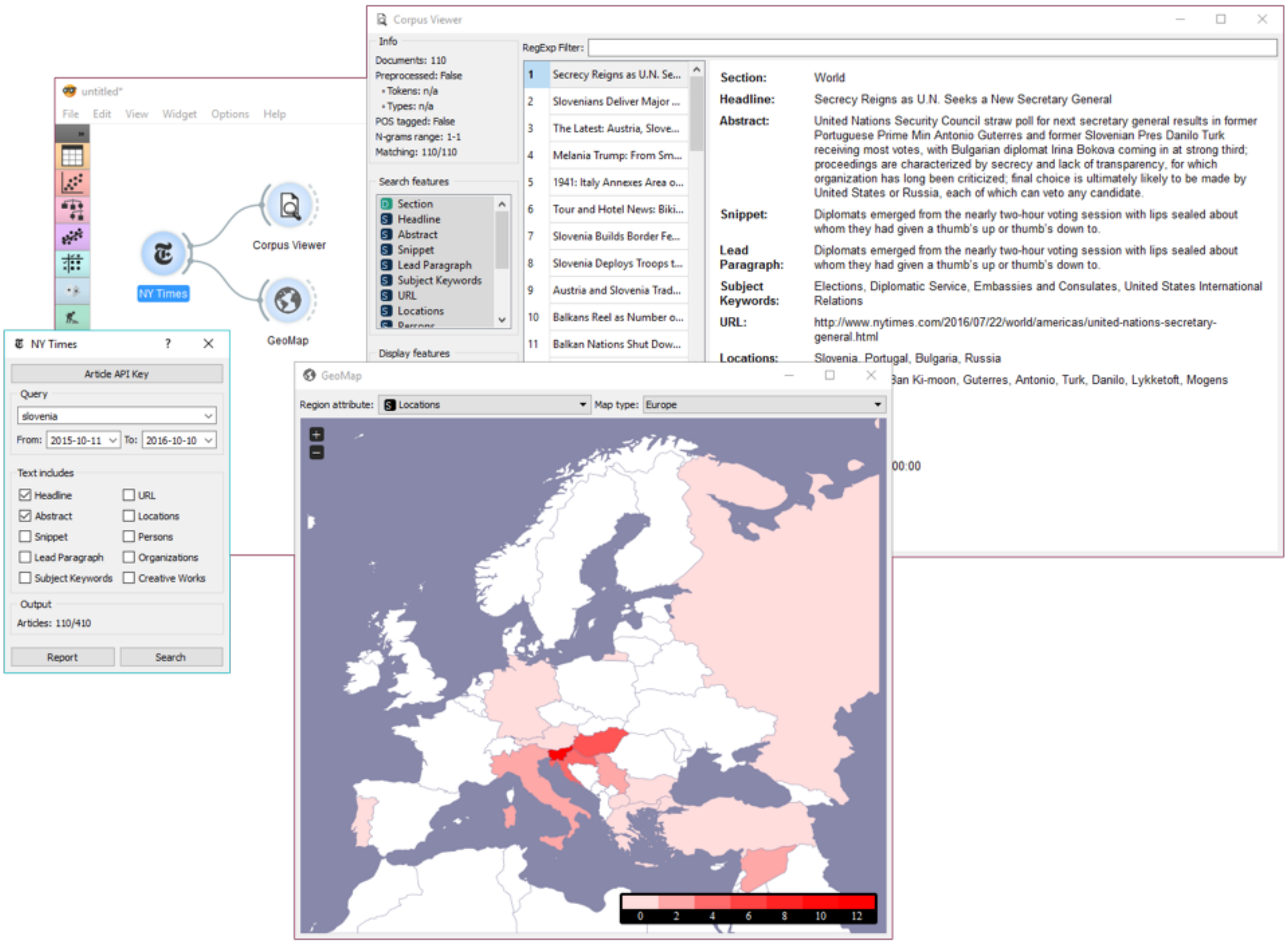
Key:

OK

2. Set query parameters:
    - *Query*
    - Query time frame. The widget allows querying articles from September 18, 1851 onwards. Default is set to 1 year back from the current date.
3. Define which features to include as text features.
4. Information on the output.
5. Produce report.
6. Run or stop the query.

Example

NYTimes is a data retrieving widget, similar to Twitter and Wikipedia. As it can retrieve geolocations, that is geographical locations the article mentions, it is great in combination with GeoMap widget.



NY Times

Article API Key

Query: slovenia

From: 2015-10-11 To: 2016-10-10

Text includes

- [x] Headline
- [x] Abstract
- [ ] Snippet
- [ ] Lead Paragraph
- [ ] Subject Keywords
- [ ] URL
- [ ] Locations
- [ ] Persons
- [ ] Organizations
- [ ] Creative Works

Output

Articles: 110/410

Report | Search

Corpus Viewer

Info

Documents: 110
Preprocessed: False
• Tokens: n/a
• Types: n/a
POS tagged: False
N-grams range: 1-1
Matching: 110/110

Search features

Section
Headline
Abstract
Snippet
Lead Paragraph
Subject Keywords
URL
Locations
Persons

Display features

RegExp Filter:

1 Secrecy Reigns as U.N. Se...
2 Slovenians Deliver Major ...
3 The Latest: Austria, Slove...
4 Melania Trump: From Sm...
5 1941: Italy Annexes Area o...
6 Tour and Hotel News: Biki...
7 Slovenia Builds Border Fe...
8 Slovenia Deploys Troops t...
9 Austria and Slovenia Trad...
10 Balkans Reel as Number o...
11 Balkan Nations Shut Dow...

Section:	World
Headline:	Secrecy Reigns as U.N. Seeks a New Secretary General
Abstract:	United Nations Security Council straw poll for next secretary general results in former Portuguese Prime Min Antonio Guterres and former Slovenian Pres Danilo Turk receiving most votes, with Bulgarian diplomat Irina Bokova coming in at strong third; proceedings are characterized by secrecy and lack of transparency, for which organization has long been criticized; final choice is ultimately likely to be made by United States or Russia, each of which can veto any candidate.
Snippet:	Diplomats emerged from the nearly two-hour voting session with lips sealed about whom they had given a thumb's up or thumb's down to.
Lead Paragraph:	Diplomats emerged from the nearly two-hour voting session with lips sealed about whom they had given a thumb's up or thumb's down to.
Subject Keywords:	Elections, Diplomatic Service, Embassies and Consulates, United States International Relations
URL:	http://www.nytimes.com/2016/07/22/world/americas/united-nations-secretary-general.html
Locations:	Slovenia, Portugal, Bulgaria, Russia
	Ban Ki-moon, Guterres, Antonio, Turk, Danilo, Lykketoft, Mogens
	00:00

GeoMap

Region attribute: Locations Map type: Europe

First, let's query **NYTimes** for all articles on Slovenia. We can retrieve the articles found and view the results in Corpus Viewer. The widget displays all the retrieved features, but includes on selected features as text mining features.

Now, let's inspect the distribution of geolocations from the articles mentioning Slovenia. We can do this with GeoMap. Unsuprisignly, Croatia and Hungary appear the most often in articles on Slovenia (discounting Slovenia itself), with the rest of Europe being mentioned very often as well.

Preprocess Text

Preprocesses corpus with selected methods.

Signals

Inputs:

- **Corpus**

  Corpus instance.

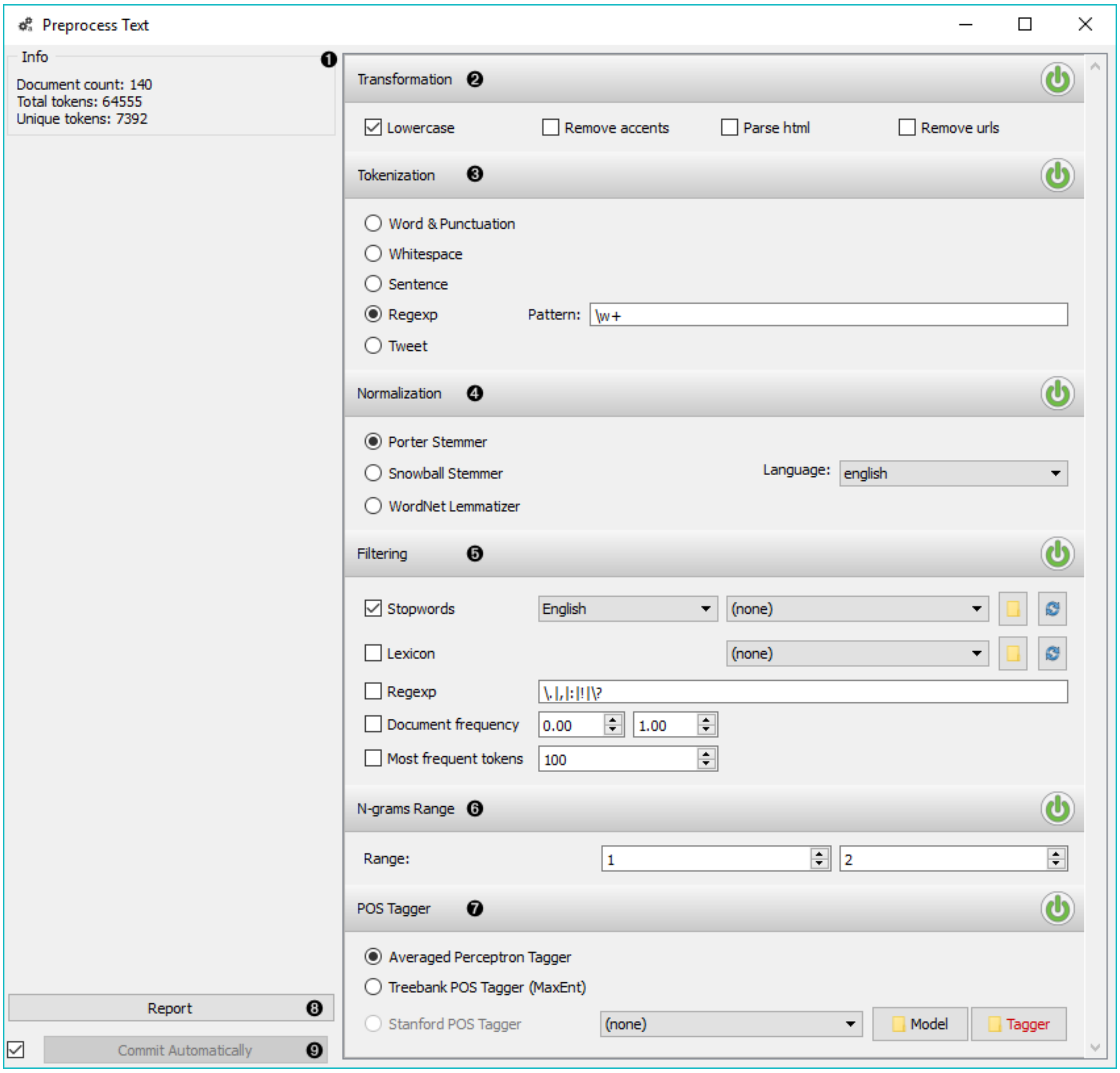
Outputs:

- **Corpus**

  Preprocessed corpus.

Description

Preprocess Text splits your text into smaller units (tokens), filters them, runs normalization (stemming, lemmatization), creates n-grams and tags tokens with part-of-speech labels. Steps in the analysis are applied sequentially and can be turned on or off.


Preprocess Text

Info ❶
Document count: 140
Total tokens: 64555
Unique tokens: 7392

Transformation ❷
[✓ Lowercase] ☐ Remove accents ☐ Parse html ☐ Remove urls

Tokenization ❸
○ Word & Punctuation
○ Whitespace
○ Sentence
● Regexp Pattern: \w+
○ Tweet

Normalization ❹
● Porter Stemmer
○ Snowball Stemmer Language: english
○ WordNet Lemmatizer

Filtering ❺
[✓ Stopwords] English (none)
☐ Lexicon (none)
☐ Regexp \.|,|:|!|\?
☐ Document frequency 0.00 1.00
☐ Most frequent tokens 100

N-grams Range ❻
Range: 1 2

POS Tagger ❼
● Averaged Perceptron Tagger
○ Treebank POS Tagger (MaxEnt)
○ Stanford POS Tagger (none) Model Tagger

Report ❽
[✓] Commit Automatically ❾

1. **Information on preprocessed data**. *Document count* reports on the number of documents on the input. *Total tokens* counts all the tokens in corpus. *Unique tokens* excludes duplicate tokens and reports only on unique tokens in the corpus.

2. **Transformation** transforms input data. It applies lowercase transformation by default.
   - *Lowercase* will turn all text to lowercase.
   - *Remove accents* will remove all diacritics/accents in text.

     naïve → naive

   - *Parse html* will detect html tags and parse out text only.

     <a href…>Some text</a> → Some text

   - *Remove urls* will remove urls from text.

     This is a http://orange.biolab.si/ url. → This is a url.

3. Tokenization is the method of breaking the text into smaller components (words, sentences, bigrams).

- *Word & Punctuation* will split the text by words and keep punctuation symbols.

  This example. → (This), (example), (.)

- *Whitespace* will split the text by whitespace only.

  This example. → (This), (example.)

- *Sentence* will split the text by fullstop, retaining only full sentences.

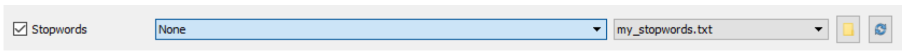
  This example. Another example. → (This example.), (Another example.)

- Regexp will split the text by provided regex. It splits by words only by default (omits punctuation).
- *Tweet* will split the text by pre-trained Twitter model, which keeps hashtags, emoticons and other special symbols.

  This example. :-) #simple → (This), (example), (.), (:-)), (#simple)

4. **Normalization** applies stemming and lemmatization to words. (I've always loved cats. → I have alway love cat.) For languages other than English use Snowball Stemmer (offers languages available in its NLTK implementation).
   - Porter Stemmer applies the original Porter stemmer.
   - Snowball Stemmer applies an improved version of Porter stemmer (Porter2). Set the language for normalization, default is English.
   - WordNet Lemmatizer applies a networks of cognitive synonyms to tokens based on a large lexical database of English.

5. **Filtering** removes or keeps a selection of words.
   - *Stopwords* removes stopwords from text (e.g. removes 'and', 'or', 'in'…). Select the language to filter by, English is set as default. You can also load your own list of stopwords provided in a simple *.txt file with one stopword per line.

     Click 'browse' icon to select the file containing stopwords. If the file was properly loaded, its name will be displayed next to pre-loaded stopwords. Change 'English' to 'None' if you wish to filter out only the provided stopwords. Click 'reload' icon to reload the list of stopwords.

   - *Lexicon* keeps only words provided in the file. Load a *.txt file with one word per line to use as lexicon. Click 'reload' icon to reload the lexicon.

   - *Regexp* removes words that match the regular expression. Default is set to remove punctuation.

   - *Document frequency* keeps tokens that appear in not less than and not more than the specified number / percentage of documents. If you provide integers as parameters, it keeps only tokens that appear in the specified number of documents. E.g. DF = (3, 5) keeps only tokens that appear in 3 or more and 5 or less documents. If you provide floats as parameters, it keeps only tokens that appear in the specified percentage of documents. E.g. DF = (0.3, 0.5) keeps only tokens that appear in 30% to 50% of documents. Default returns all tokens.

  - *Most frequent tokens* keeps only the specified number of most frequent tokens. Default is a 100 most frequent tokens.

6. **N-grams Range** creates n-grams from tokens. Numbers specify the range of n-grams. Default returns one-grams and two-grams.

7. **POS Tagger** runs part-of-speech tagging on tokens.
   - Averaged Perceptron Tagger runs POS tagging with Matthew Honnibal's averaged perceptron tagger.
   - Treebank POS Tagger (MaxEnt) runs POS tagging with a trained Penn Treebank model.
   - Stanford POS Tagger runs a log-linear part-of-speech tagger designed by Toutanova et al. Please download it from the provided website and load it in Orange.

8. Produce a report.

9. If *Commit Automatically* is on, changes are communicated automatically. Alternatively press *Commit*.

Note: **Preprocess Text** applies preprocessing steps in the order they are listed. This means it will first transform the text, then apply tokenization, POS tags, normalization, filtering and finally constructs n-grams based on given tokens. This is especially important for WordNet Lemmatizer since it requires POS tags for proper normalization.

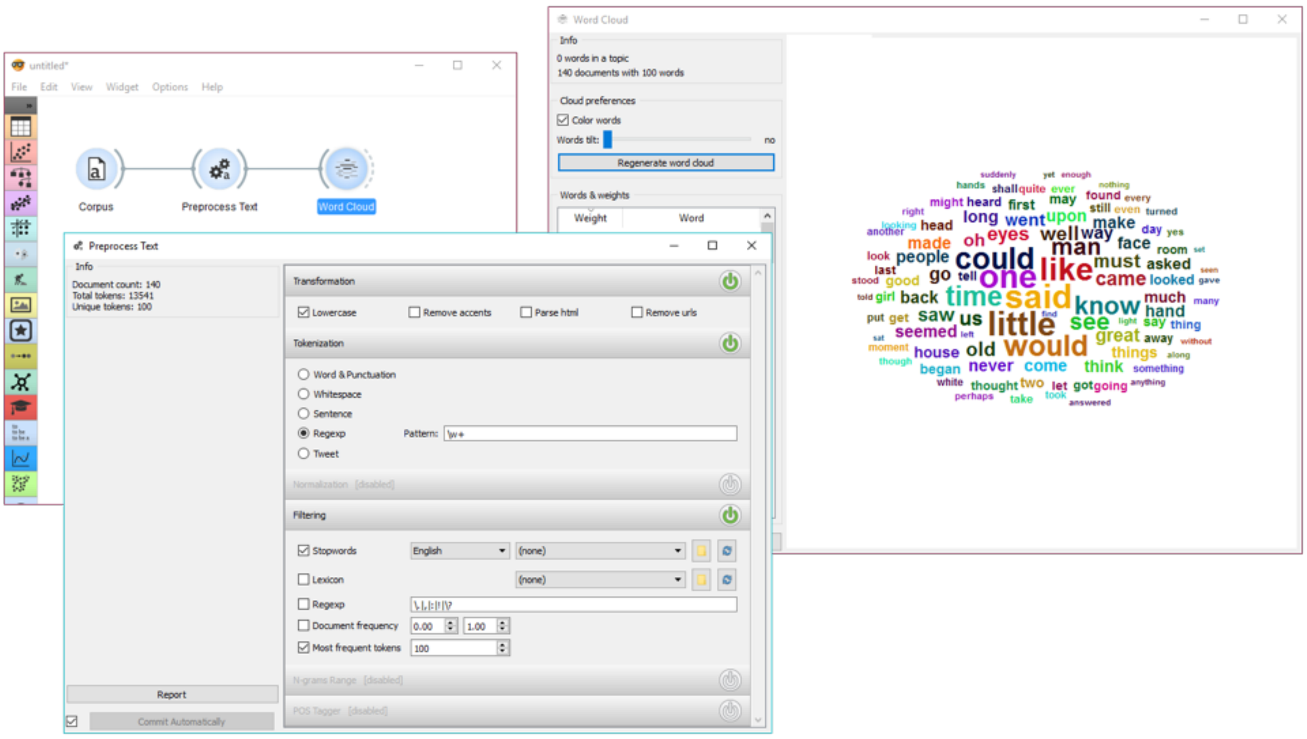
Useful Regular Expressions

Here are some useful regular expressions for quick filtering:

`\bword\b`	matches exact word
`\w+`	matches only words, no punctuation
`\b(B|b)\w+\b`	matches words beginning with the letter b
`\w{4,}`	matches words that are longer than 4 characters
`\b\w+(Y|y)\b`	matches words ending with the letter y

Examples

In the first example we will observe the effects of preprocessing on our text. We are working with *book-excerpts.tab* that we've loaded with Corpus widget. We have connected **Preprocess Text** to **Corpus** and retained default preprocessing methods (lowercase, per-word tokenization and stopword removal). The only additional parameter we've added as outputting only the first 100 most frequent tokens. Then we connected **Preprocess Text** with Word Cloud to observe words that are the most frequent in our text. Play around with different parameters, to see how they transform the output.



Info
0 words in a topic
140 documents with 100 words

Preprocess Text

Info
Document count: 140
Total tokens: 13541
Unique tokens: 100

Transformation
[✓] Lowercase [ ] Remove accents [ ] Parse html [ ] Remove urls

Tokenization
( ) Word & Punctuation
( ) Whitespace
( ) Sentence
(●) Regexp Pattern: \w+
( ) Tweet

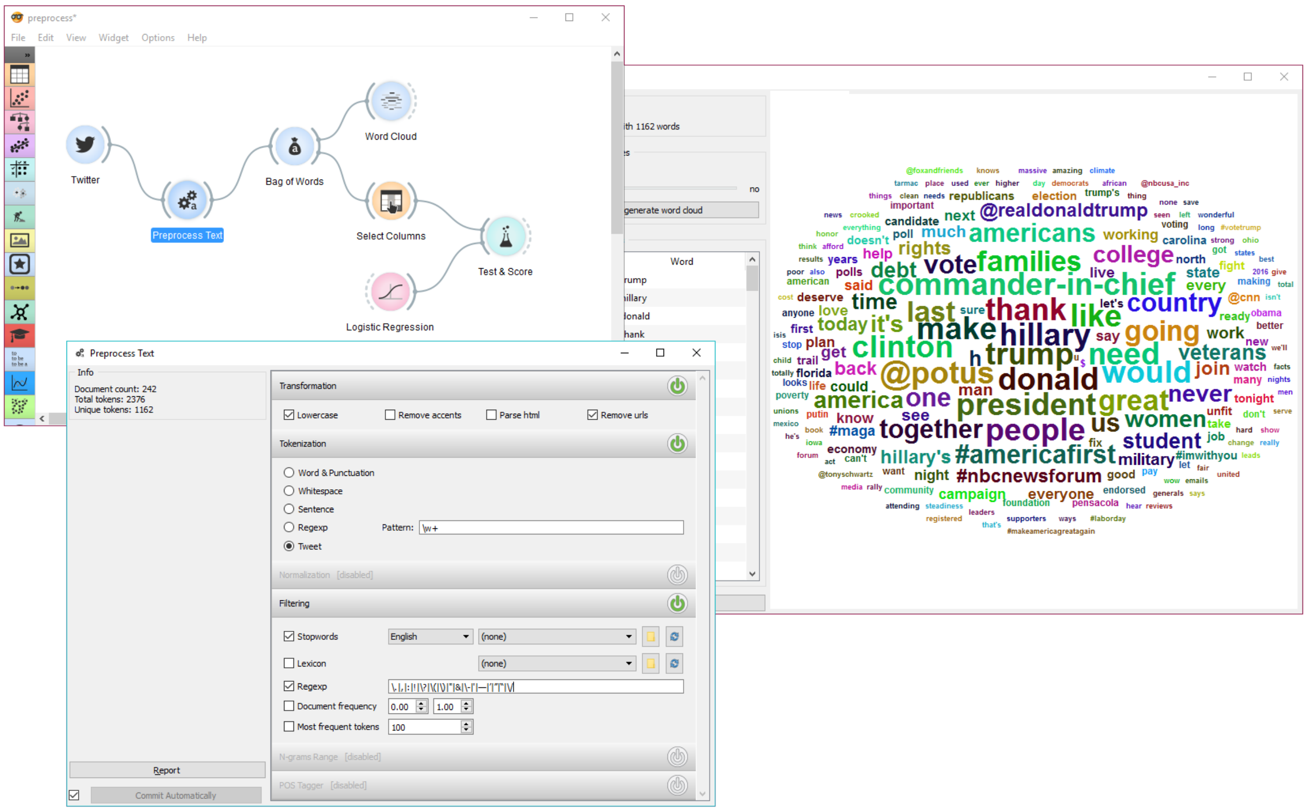
Normalization [disabled]

Filtering
[✓] Stopwords English (none)
[ ] Lexicon (none)
[ ] Regexp \.|,|:|!|\?
[ ] Document frequency 0.00 1.00
[✓] Most frequent tokens 100

N-grams Range [disabled]
POS Tagger [disabled]

Report
[✓] Commit Automatically

The second example is slightly more complex. We first acquired our data with Twitter widget. We quired the internet for tweets from users @HillaryClinton and @realDonaldTrump and got their tweets from the past two weeks, 242 in total.



th 1162 words

Preprocess Text

Info
Document count: 242
Total tokens: 2376
Unique tokens: 1162

Transformation
[✓] Lowercase [ ] Remove accents [ ] Parse html [✓] Remove urls

Tokenization
( ) Word & Punctuation
( ) Whitespace
( ) Sentence
( ) Regexp Pattern: \w+
(●) Tweet

Normalization [disabled]

Filtering
[✓] Stopwords English (none)
[ ] Lexicon (none)
[✓] Regexp \.|,|:|!|\?|\(|\)|"|&|\-|'|—|'|"|"|\/
[ ] Document frequency 0.00 1.00
[ ] Most frequent tokens 100

N-grams Range [disabled]
POS Tagger [disabled]

Report
[✓] Commit Automatically

In **Preprocess Text** there's *Tweet* tokenization available, which retains hashtags, emojis, mentions and so on. However, this tokenizer doesn't get rid of punctuation, thus we expanded the Regexp filtering with symbols that we wanted to get rid of. We ended up with word-only tokens, which we displayed in Word Cloud. Then we created a schema for predicting author based on tweet content, which is explained in more details in the documentation for Twitter widget.

Pubmed

Fetch data from PubMed journals.

Signals

Inputs:

- (None)

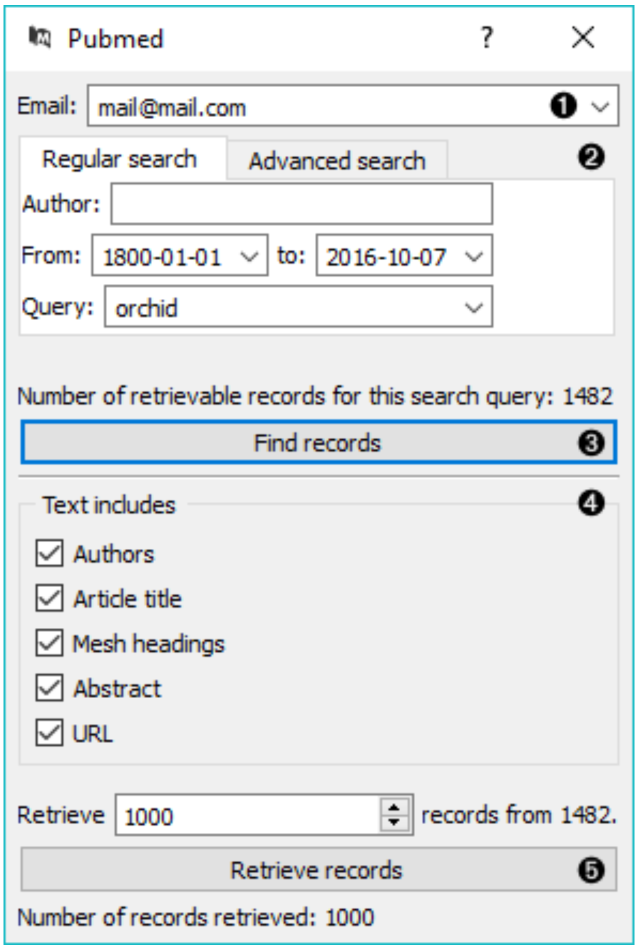
Outputs:

- **Corpus**

  A Corpus instance.

Description

PubMed comprises more than 26 million citations for biomedical literature from MEDLINE, life science journals, and online books. The widget allows you to query and retrieve these entries. You can use regular search or construct advanced queries.


Pubmed

Email: mail@mail.com

Regular search | Advanced search

Author:

From: 1800-01-01 to: 2016-10-07

Query: orchid

Number of retrievable records for this search query: 1482

Find records

Text includes

☑ Authors

☑ Article title

☑ Mesh headings

☑ Abstract

☑ URL

Retrieve 1000 records from 1482.

Retrieve records

Number of records retrieved: 1000

1. Enter a valid e-mail to retrieve queries.

2. *Regular search*:
    - *Author*: queries entries from a specific author. Leave empty to query by all authors.
    - *From*: define the time frame of publication.
    - *Query*: enter the query.

   Advanced search: enables you to construct complex queries. See PubMed's website to learn how to construct such queries. You can also copy-paste constructed queries from the website.

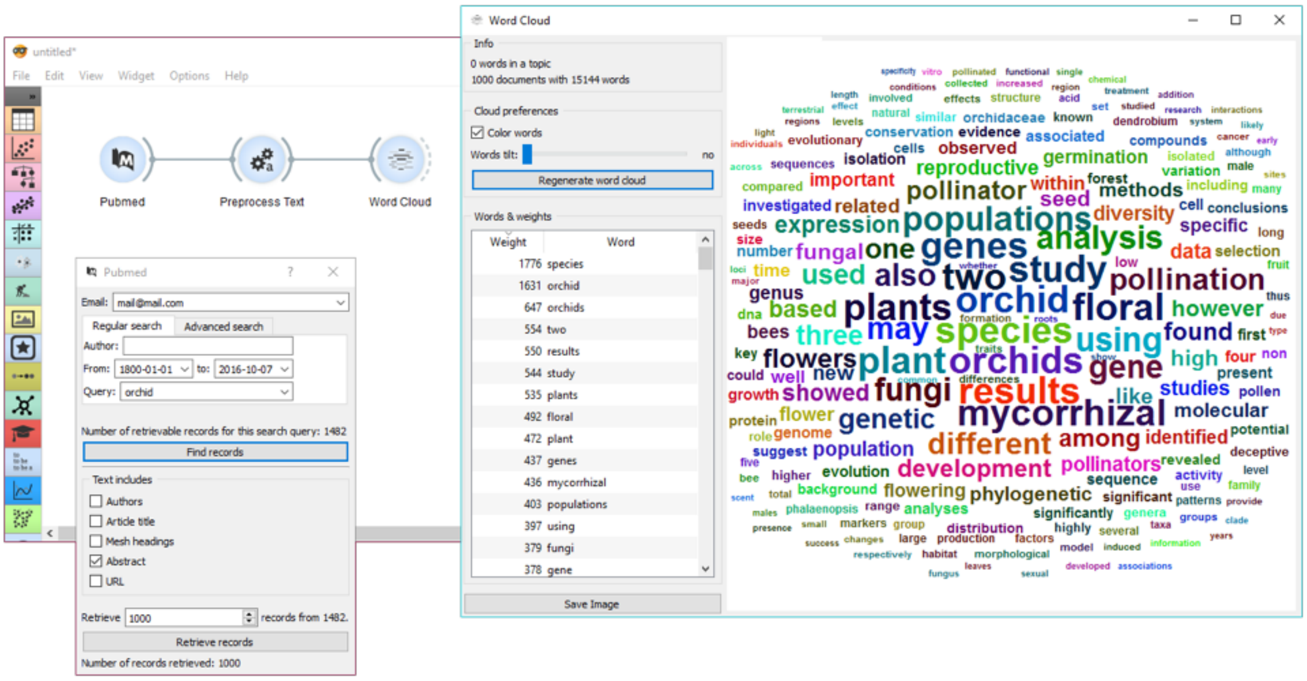
3. *Find records* finds available data from PubMed matching the query. Number of records found will be displayed above the button.

4. Define the output. All checked features will be on the output of the widget.

5. Set the number of record you wish to retrieve. Press *Retrieve records* to get results of your query on the output. Below the button is an information on the number of records on the output.

Example

PubMed can be used just like any other data widget. In this example we've queried the database for records on orchids. We retrieved 1000 records and kept only 'abstract' in our meta features to limit the construction of tokens only to this feature.



Pubmed

Email: mail@mail.com

Regular search | Advanced search

Author:

From: 1800-01-01 to: 2016-10-07

Query: orchid

Number of retrievable records for this search query: 1482

Find records

Text includes

- [ ] Authors
- [ ] Article title
- [ ] Mesh headings
- [x] Abstract
- [ ] URL

Retrieve 1000 records from 1482.

Retrieve records

Number of records retrieved: 1000

Word Cloud

Info

0 words in a topic

1000 documents with 15144 words

Cloud preferences

- [x] Color words

Words tilt: no

Regenerate word cloud

Words & weights

Weight	Word
1776	species
1631	orchid
647	orchids
554	two
550	results
544	study
535	plants
492	floral
472	plant
437	genes
436	mycorrhizal
403	populations
397	using
379	fungi
378	gene

Save Image

We used Preprocess Text to remove stopword and words shorter than 3 characters (regexp `\b\w{1,2}\b`). This will perhaps get rid of some important words denoting chemicals, so we need to be careful with what we filter out. For the sake of quick inspection we only retained longer words, which are displayed by frequency in Word Cloud.

Topic Modelling

Topic modelling with Latent Diriclet Allocation, Latent Semantic Indexing or Hierarchical Dirichlet Process.

Signals

Inputs:

- **Corpus**

  Corpus instance.

Outputs:

- **Data**

  Data with topic weights appended.

- **Topics**

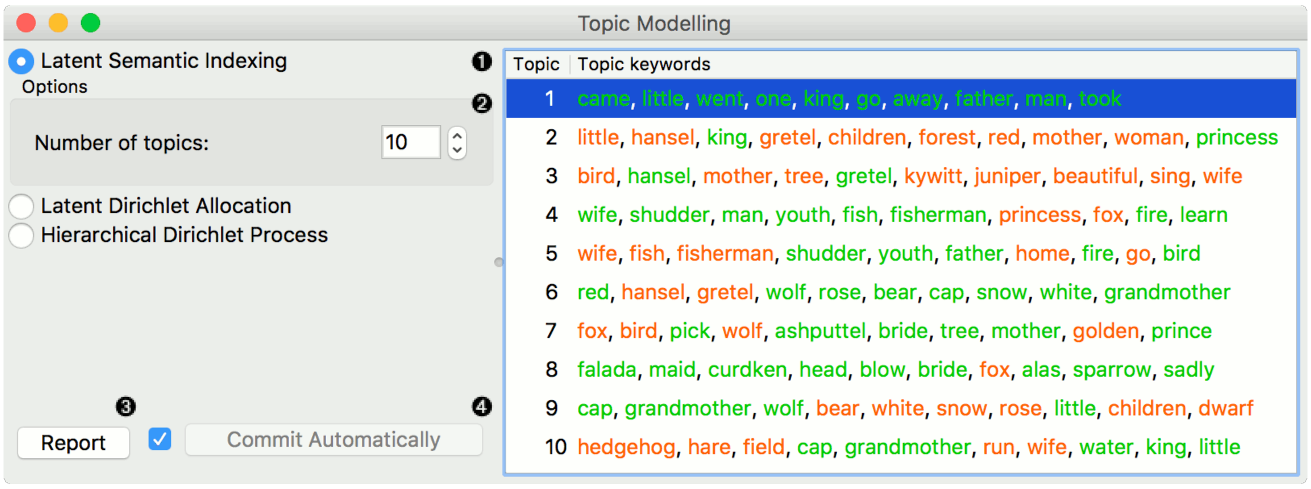
  Selected topics with word weights.

- **All Topics**

  Topic weights by tokens.

Description

Topic Modelling discovers abstract topics in a corpus based on clusters of words found in each document and their respective frequency. A document typically contains multiple topics in different proportions, thus the widget also reports on the topic weight per document.


Topic Modelling

(●) Latent Semantic Indexing ❶

Options ❷

Number of topics: 10

( ) Latent Dirichlet Allocation

( ) Hierarchical Dirichlet Process

❸ Report ❹ [✓] Commit Automatically

Topic	Topic keywords
1	came, little, went, one, king, go, away, father, man, took
2	little, hansel, king, gretel, children, forest, red, mother, woman, princess
3	bird, hansel, mother, tree, gretel, kywitt, juniper, beautiful, sing, wife
4	wife, shudder, man, youth, fish, fisherman, princess, fox, fire, learn
5	wife, fish, fisherman, shudder, youth, father, home, fire, go, bird
6	red, hansel, gretel, wolf, rose, bear, cap, snow, white, grandmother
7	fox, bird, pick, wolf, ashputtel, bride, tree, mother, golden, prince
8	falada, maid, curdken, head, blow, bride, fox, alas, sparrow, sadly
9	cap, grandmother, wolf, bear, white, snow, rose, little, children, dwarf
10	hedgehog, hare, field, cap, grandmother, run, wife, water, king, little

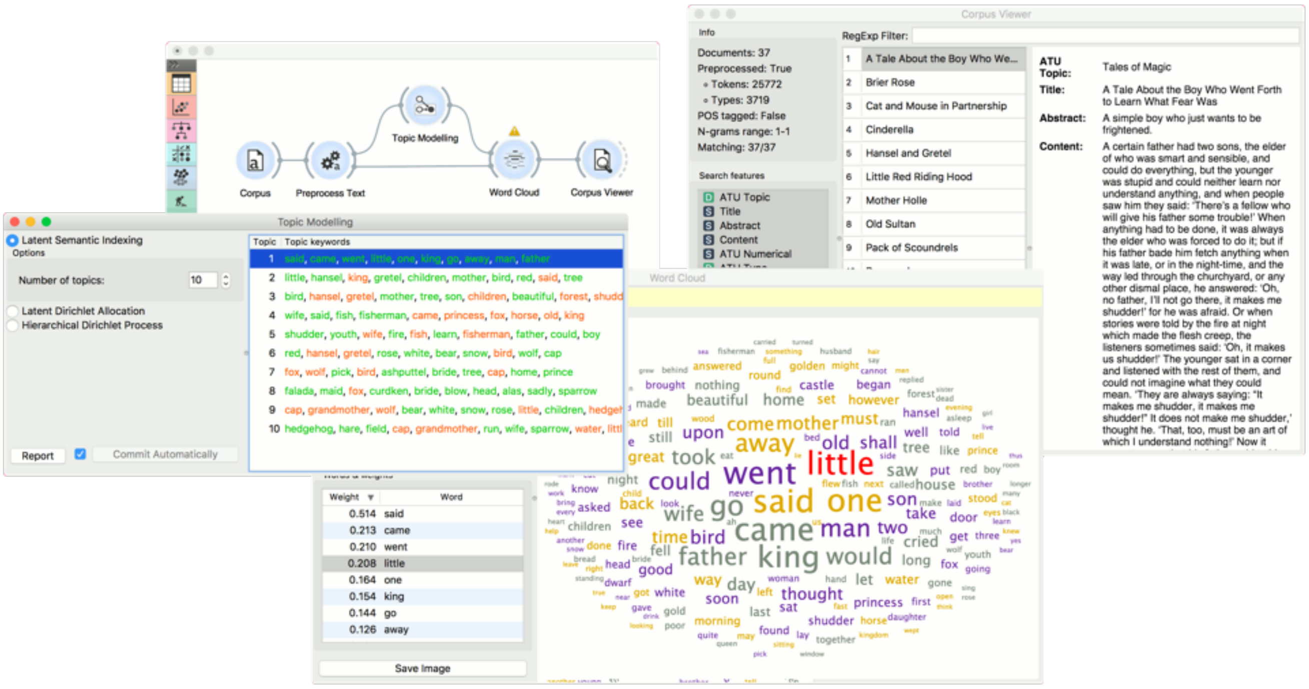
1. Topic modelling algorithm:
   - Latent Semantic Indexing
   - Latent Dirichlet Allocation
   - Hierarchical Dirichlet Process

2. Parameters for the algorithm. LSI and LDA accept only the number of topics modelled, with the default set to 10. HDP, however, has more parameters. As this algorithm is computationally very demanding, we recommend you to try it on a subset or set all the required parameters in advance and only then run the algorithm (connect the input to the widget).
   - First level concentration (γ): distribution at the first (corpus) level of Dirichlet Process
   - Second level concentration (α): distribution at the second (document) level of Dirichlet Process
   - The topic Dirichlet (α): concentration parameter used for the topic draws
   - Top level truncation (Τ): corpus-level truncation (no of topics)
   - Second level truncation (Κ): document-level truncation (no of topics)
   - Learning rate (κ): step size
   - Slow down parameter (τ)

3. Produce a report.
4. If *Commit Automatically* is on, changes are communicated automatically. Alternatively press *Commit*.

Example

In the first example, we present a simple use of the **Topic Modelling** widget. First we load *grimm-tales-selected.tab* data set and use Preprocess Text to tokenize by words only and remove stopwords. Then we connect **Preprocess Text** to **Topic Modelling**, where we use a simple *Latent Semantic Indexing* to find 10 topics in the text.



Topic Modelling

[✓ Latent Semantic Indexing]
Options
Number of topics: 10
Latent Dirichlet Allocation
Hierarchical Dirichlet Process

Topic	Topic keywords
1	said, came, went, little, one, king, go, away, man, father
2	little, hansel, king, gretel, children, mother, bird, red, said, tree
3	bird, hansel, gretel, mother, tree, son, children, beautiful, forest, shudd
4	wife, said, fish, fisherman, came, princess, fox, horse, old, king
5	shudder, youth, wife, fire, fish, learn, fisherman, father, could, boy
6	red, hansel, gretel, rose, white, bear, snow, bird, wolf, cap
7	fox, wolf, pick, bird, ashputtel, bride, tree, cap, home, prince
8	falada, maid, fox, curdken, bride, blow, head, alas, sadly, sparrow
9	cap, grandmother, wolf, bear, white, snow, rose, little, children, hedgeh
10	hedgehog, hare, field, cap, grandmother, run, wife, sparrow, water, littl

Weight	Word
0.514	said
0.213	came
0.210	went
0.208	little
0.164	one
0.154	king
0.144	go
0.126	away

Corpus Viewer

Info
Documents: 37
Preprocessed: True
- Tokens: 25772
- Types: 3719
POS tagged: False
N-grams range: 1-1
Matching: 37/37

Search features
- ATU Topic
- Title
- Abstract
- Content
- ATU Numerical

RegExp Filter:

1	A Tale About the Boy Who We...
2	Brier Rose
3	Cat and Mouse in Partnership
4	Cinderella
5	Hansel and Gretel
6	Little Red Riding Hood
7	Mother Holle
8	Old Sultan
9	Pack of Scoundrels

ATU Topic:	Tales of Magic
Title:	A Tale About the Boy Who Went Forth to Learn What Fear Was
Abstract:	A simple boy who just wants to be frightened.
Content:	A certain father had two sons, the elder of who was smart and sensible, and could do everything, but the younger was stupid and could neither learn nor understand anything, and when people saw him they said: 'There's a fellow who will give his father some trouble!' When anything had to be done, it was always the elder who was forced to do it; but if his father bade him fetch anything when it was late, or in the night-time, and the way led through the churchyard, or any other dismal place, he answered: 'Oh, no father, I'll not go there, it makes me shudder!' for he was afraid. Or when stories were told by the fire at night which made the flesh creep, the listeners sometimes said: 'Oh, it makes us shudder!' The younger sat in a corner and listened with the rest of them, and could not imagine what they could mean. 'They are always saying: "It makes me shudder, it makes me shudder!" It does not make me shudder,' thought he. 'That, too, must be an art of which I understand nothing!' Now it

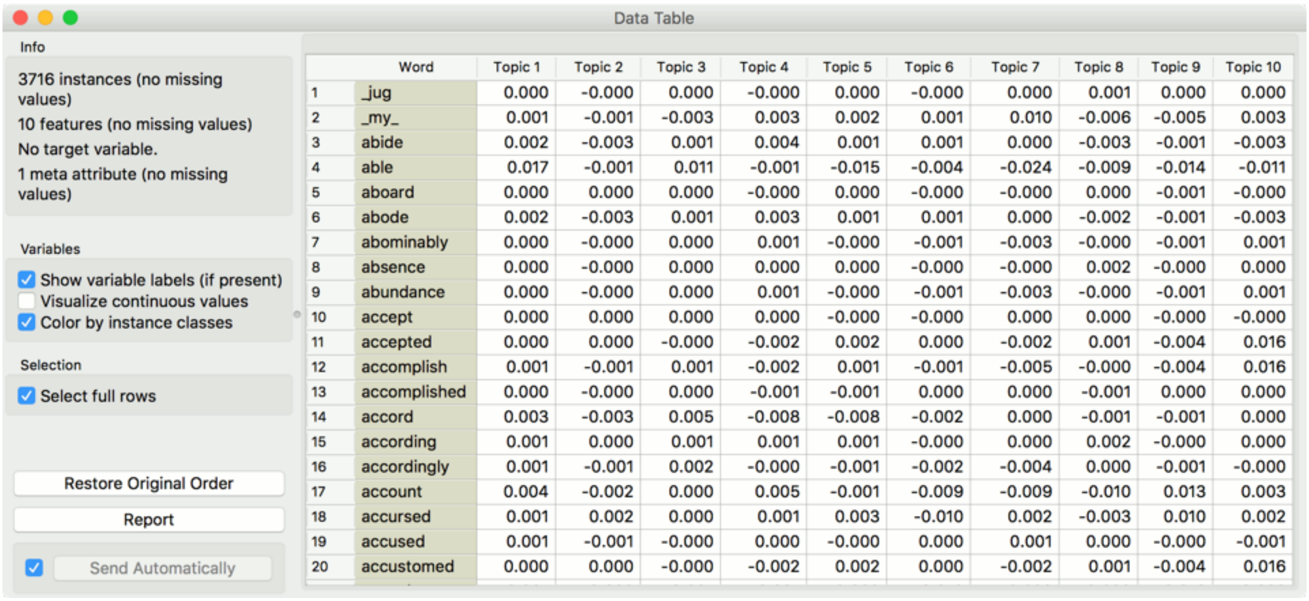
LSI provides both positive and negative weights per topic. A positive weight means the word is highly representative of a topic, while a negative weight means the word is highly unrepresentative of a topic (the less it occurs in a text, the more likely the topic). Positive words are colored green and negative words are colored red.

We then select the first topic and display the most frequent words in the topic in Word Cloud. We also connected **Preprocess Text** to **Word Cloud** in order to be able to output selected documents. Now we can select a specific word in the word cloud, say *little*. It will be colored red and also highlighted in the word list on the left.

Now we can observe all the documents containing the word *little* in Corpus Viewer.

In the second example, we will look at the correlation between topics and words/documents. Connect **Topic Modelling** to **Heat Map**. Ensure the link is set to *All Topics* - *Data*. **Topic Modelling** will output a matrix of topic weights by words from text (more precisely, tokens).

We can observe the output in a **Data Table**. Tokens are in rows and retrieved topics in colums. Values represent how much a word is represented in a topic.



Data Table

Info

3716 instances (no missing values)
10 features (no missing values).
No target variable.
1 meta attribute (no missing values)

Variables

✓ Show variable labels (if present)
Visualize continuous values
✓ Color by instance classes

Selection

✓ Select full rows

Restore Original Order
Report
✓ Send Automatically

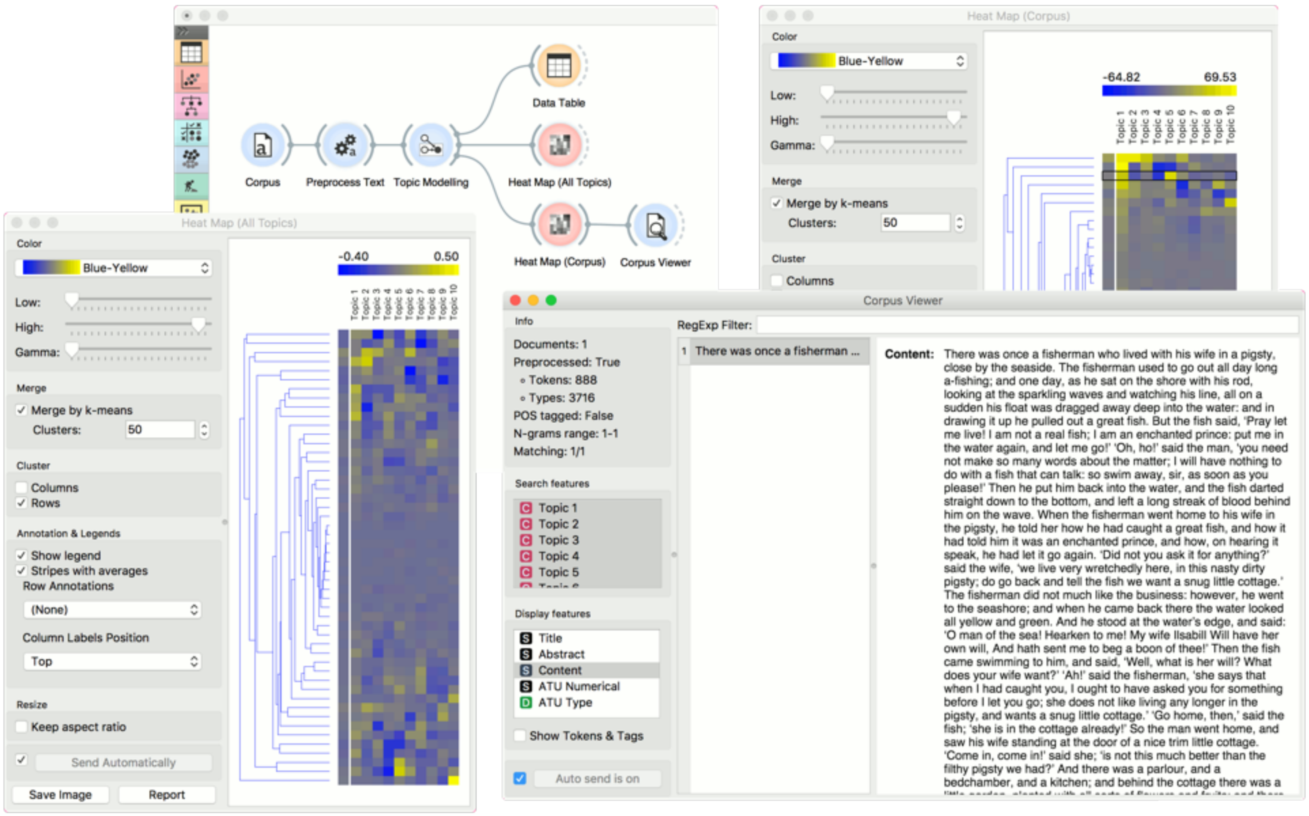
	Word	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9	Topic 10
1	_jug	0.000	-0.000	0.000	-0.000	0.000	-0.000	0.000	0.001	0.000	0.000
2	_my_	0.001	-0.001	-0.003	0.003	0.002	0.001	0.010	-0.006	-0.005	0.003
3	abide	0.002	-0.003	0.001	0.004	0.001	0.001	0.000	-0.003	-0.001	-0.003
4	able	0.017	-0.001	0.011	-0.001	-0.015	-0.004	-0.024	-0.009	-0.014	-0.011
5	aboard	0.000	0.000	0.000	-0.000	0.000	-0.000	-0.000	0.000	-0.001	-0.000
6	abode	0.002	-0.003	0.001	0.003	0.001	0.001	0.000	-0.002	-0.001	-0.003
7	abominably	0.000	-0.000	0.000	0.001	-0.000	-0.001	-0.003	-0.000	-0.001	0.001
8	absence	0.000	-0.000	0.000	0.000	0.000	-0.000	-0.000	0.002	-0.000	0.000
9	abundance	0.000	-0.000	0.000	0.001	-0.000	-0.001	-0.003	-0.000	-0.001	0.001
10	accept	0.000	0.000	0.000	0.000	-0.000	-0.000	0.000	0.000	-0.000	-0.000
11	accepted	0.000	0.000	-0.000	-0.002	0.002	0.000	-0.002	0.001	-0.004	0.016
12	accomplish	0.001	-0.001	0.001	-0.002	0.001	-0.001	-0.005	-0.000	-0.004	0.016
13	accomplished	0.000	-0.000	0.000	-0.001	-0.001	0.000	0.000	-0.001	0.000	0.000
14	accord	0.003	-0.003	0.005	-0.008	-0.008	-0.002	0.000	-0.001	-0.001	0.000
15	according	0.001	0.000	0.001	0.001	0.001	-0.000	0.000	0.002	-0.000	0.000
16	accordingly	0.001	-0.001	0.002	-0.000	-0.001	-0.002	-0.004	0.000	-0.001	-0.000
17	account	0.004	-0.002	0.000	0.005	-0.001	-0.009	-0.009	-0.010	0.013	0.003
18	accursed	0.001	0.002	0.000	0.001	0.003	-0.010	0.002	-0.003	0.010	0.002
19	accused	0.001	-0.001	-0.000	0.000	-0.000	0.000	0.001	0.000	-0.000	-0.001
20	accustomed	0.000	0.000	-0.000	-0.002	0.002	0.000	-0.002	0.001	-0.004	0.016

To visualize this matrix, open **Heat Map**. Select *Merge by k-means* and *Cluster* - *Rows* to merge similar rows into one and sort them by similarity, which makes the visualization more compact.

In the upper part of the visualization, we have words that highly define topics 1-3 and in the lower part those that define topics 5 and 10.

We can similarly observe topic representation across documents. We connect another **Heat Map** to **Topic Modelling** and set link to *Corpus* - *Data*. We set *Merge* and *Cluster* as above.

In this visualization we see how much is a topic represented in a document. Looks like Topic 1 is represented almost across the entire corpus, while other topics are more specific. To observe a specific set of document, select either a clustering node or a row in the visualization. Then pass the data to Corpus Viewer.



Heat Map (All Topics)

Color: Blue-Yellow
Low:
High:
Gamma:

Merge
✓ Merge by k-means
Clusters: 50

Cluster
Columns
✓ Rows

Annotation & Legends
✓ Show legend
✓ Stripes with averages
Row Annotations: (None)
Column Labels Position: Top

Resize
Keep aspect ratio

Heat Map (Corpus)

Color: Blue-Yellow
Low:
High:
Gamma:

Merge
✓ Merge by k-means
Clusters: 50

Cluster
Columns

Corpus Viewer

Info
Documents: 1
Preprocessed: True
∘ Tokens: 888
∘ Types: 3716
POS tagged: False
N-grams range: 1-1
Matching: 1/1

Search features
Topic 1
Topic 2
Topic 3
Topic 4
Topic 5

Display features
Title
Abstract
Content
ATU Numerical
ATU Type

Show Tokens & Tags

✓ Auto send is on

RegExp Filter:

1 There was once a fisherman ...

Content: There was once a fisherman who lived with his wife in a pigsty, close by the seaside. The fisherman used to go out all day long a-fishing; and one day, as he sat on the shore with his rod, looking at the sparkling waves and watching his line, all on a sudden his float was dragged away deep into the water: and in drawing it up he pulled out a great fish. But the fish said, 'Pray let me live! I am not a real fish; I am an enchanted prince: put me in the water again, and let me go!' 'Oh, ho!' said the man, 'you need not make so many words about the matter; I will have nothing to do with a fish that can talk: so swim away, sir, as soon as you please!' Then he put him back into the water, and the fish darted straight down to the bottom, and left a long streak of blood behind him on the wave. When the fisherman went home to his wife in the pigsty, he told her how he had caught a great fish, and how it had told him it was an enchanted prince, and how, on hearing it speak, he had let it go again. 'Did not you ask it for anything?' said the wife, 'we live very wretchedly here, in this nasty dirty pigsty; do go back and tell the fish we want a snug little cottage.' The fisherman did not much like the business: however, he went to the seashore; and when he came back there the water looked all yellow and green. And he stood at the water's edge, and said: 'O man of the sea! Hearken to me! My wife Ilsabill Will have her own will, And hath sent me to beg a boon of thee!' Then the fish came swimming to him, and said, 'Well, what is her will? What does your wife want?' 'Ah!' said the fisherman, 'she says that when I had caught you, I ought to have asked you for something before I let you go; she does not like living any longer in the pigsty, and wants a snug little cottage.' 'Go home, then,' said the fish; 'she is in the cottage already!' So the man went home, and saw his wife standing at the door of a nice trim little cottage. 'Come in, come in!' said she; 'is not this much better than the filthy pigsty we had?' And there was a parlour, and a bedchamber, and a kitchen; and behind the cottage there was a

Twitter

Fetching data from The Twitter Search API.

Signals

Inputs:

- (None)

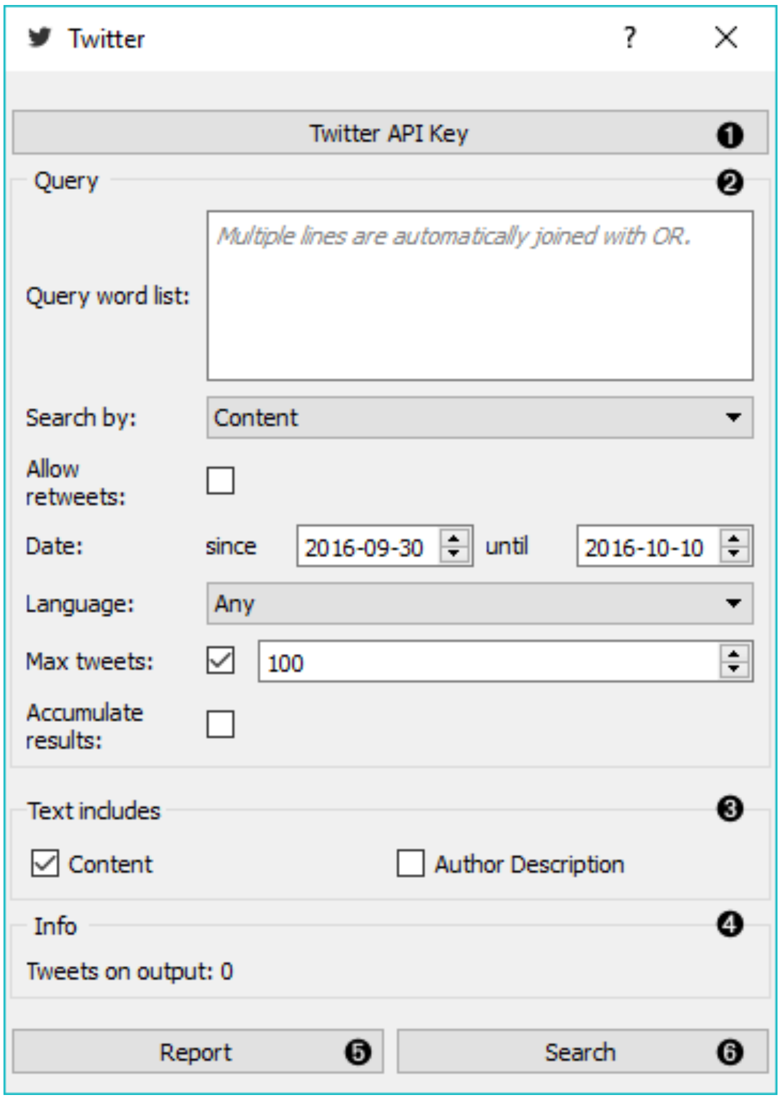
Outputs:

- **Corpus**

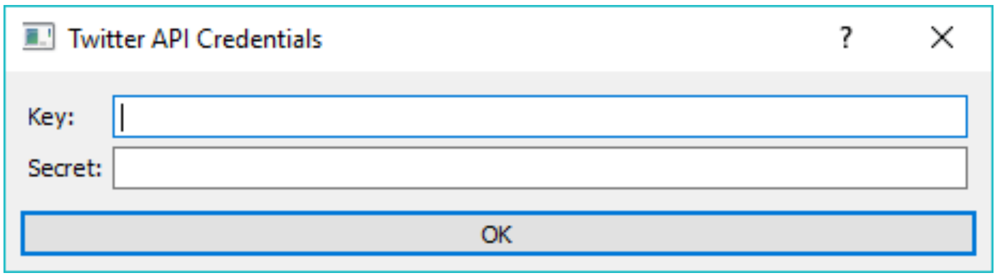
  A Corpus instance.

Description

Twitter widget enables querying tweets through Twitter API. You can query by content, author or both and accummulate results should you wish to create a larger data set. The widget only supports REST API and allows queries for up to two weeks back.



1. To begin your queries, insert Twitter key and secret. They are securely saved in your system keyring service (like Credential Vault, Keychain, KWallet, etc.) and won't be deleted when clearing widget settings. You must first create a Twitter app to get API keys.



2. Set query parameters:
   - *Query word list*: list desired queries, one per line. Queries are automatically joined by OR.
   - *Search by*: specify whether you want to search by content, author or both. If searching by author, you must enter proper Twitter handle (without @) in the query list.
   - *Allow retweets*: if 'Allow retweets' is checked, retweeted tweets will also appear on the output. This might duplicate some results.
   - *Date*: set the query time frame. Twitter only allows retrieving tweets from up to two weeks back.
   - *Language*: set the language of retrieved tweets. Any will retrieve tweets in any

language.
  - *Max tweets*: set the top limit of retrieved tweets. If box is not ticked, no upper bound will be set - widget will retrieve all available tweets.
  - *Accumulate results*: if 'Accumulate results' is ticked, widget will append new queries to the previous ones. Enter new queries, run *Search* and new results will be appended to the previous ones.

3. Define which features to include as text features.

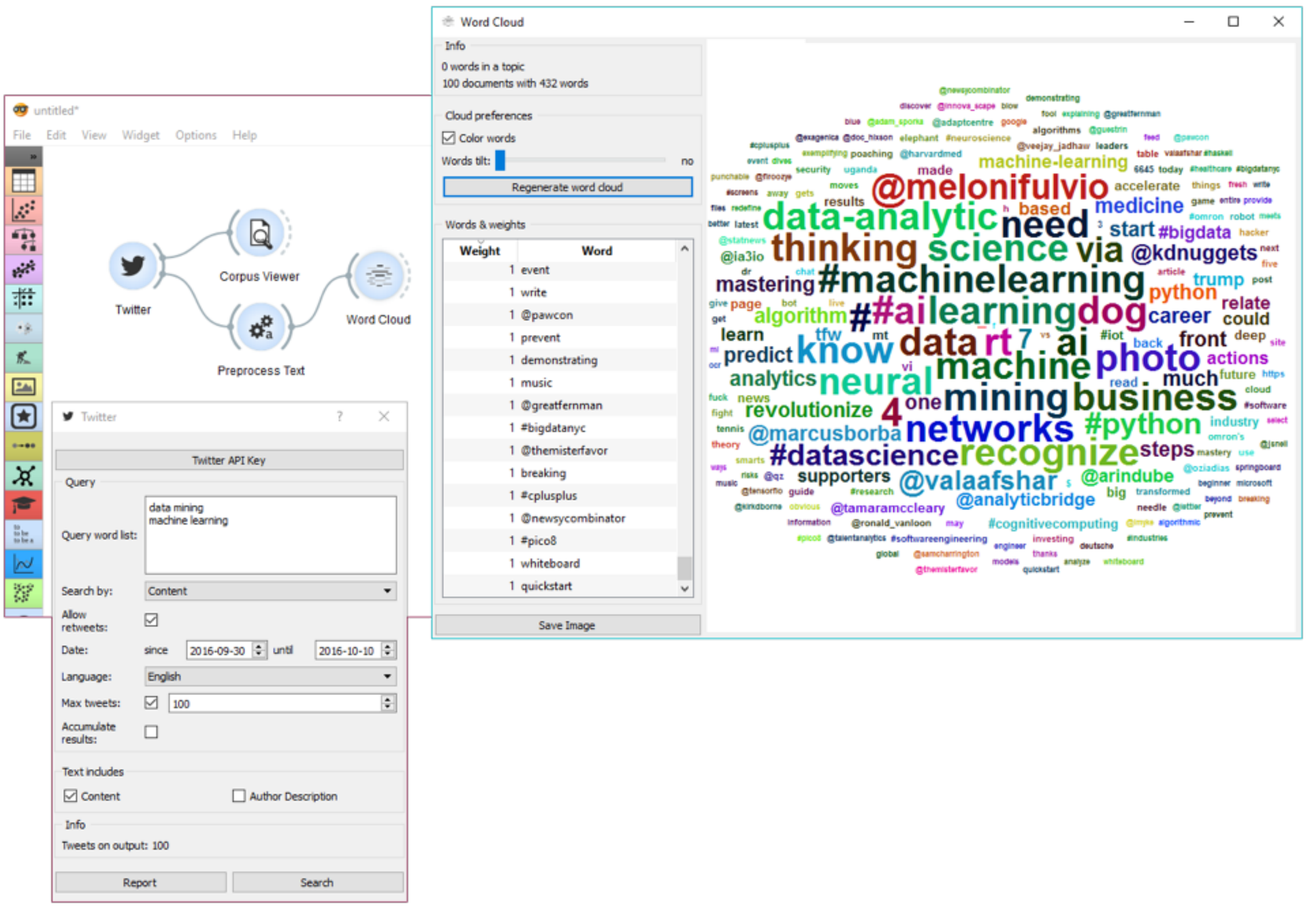
4. Information on the number of tweets on the output.

5. Produce report.

6. Run query.

Examples

First, let's try a simple query. We will search for tweets containing either 'data mining' or 'machine learning' in the content and allow retweets. We will further limit our search to only a 100 tweets in English.



Twitter

Twitter API Key

Query

Query word list: data mining
machine learning

Search by: Content

Allow retweets: ☑

Date: since 2016-09-30 until 2016-10-10

Language: English

Max tweets: ☑ 100

Accumulate results: ☐

Text includes

☑ Content ☐ Author Description

Info

Tweets on output: 100

Report Search

Word Cloud

Info

0 words in a topic
100 documents with 432 words

Cloud preferences

☑ Color words

Words tilt: no

Regenerate word cloud

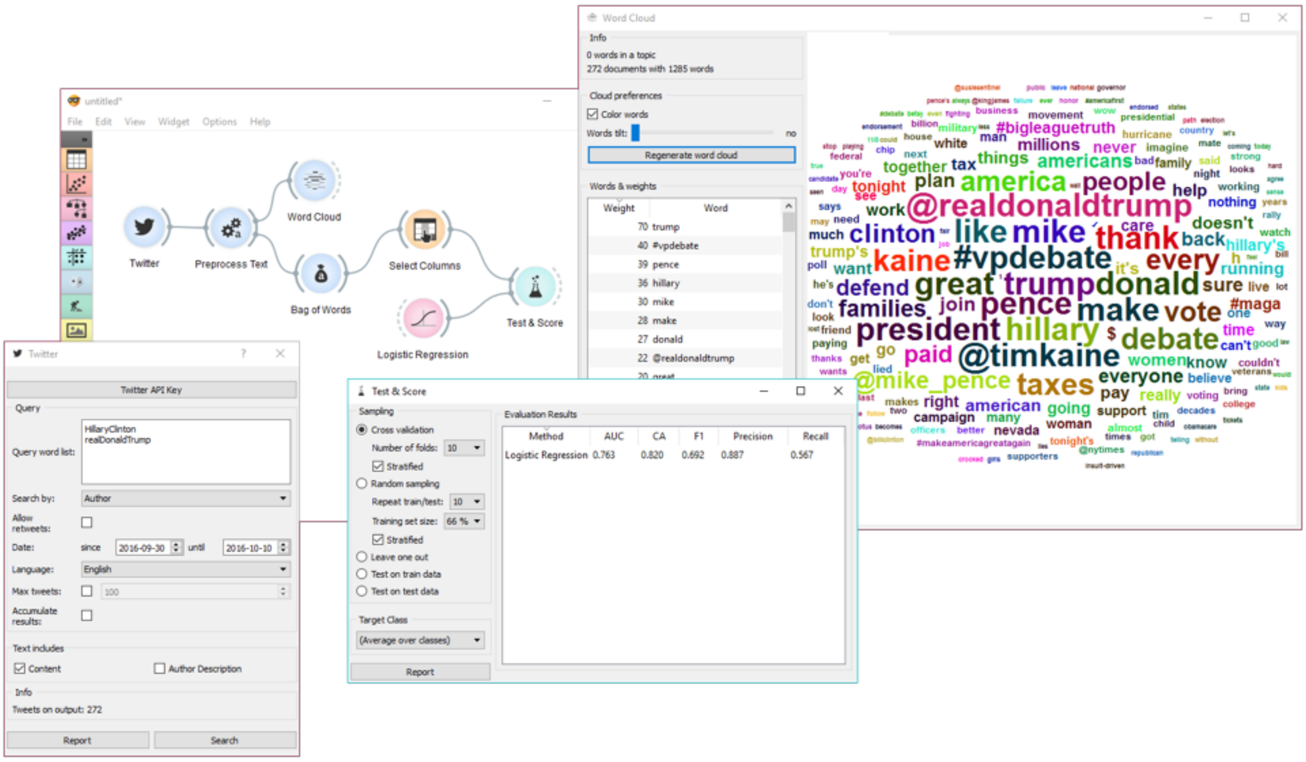
Words & weights

Weight	Word
1	event
1	write
1	@pawcon
1	prevent
1	demonstrating
1	music
1	@greatfernman
1	#bigdatanyc
1	@themisterfavor
1	breaking
1	#cplusplus
1	@newsycombinator
1	#pico8
1	whiteboard
1	quickstart

Save Image

First, we're checking the output in Corpus Viewer to get the initial idea about our results. Then we're preprocessing the tweets with lowercase, url removal, tweet tokenizer and removal of stopword and punctuation. The best way to see the results is with Word Cloud. This will display the most popular words in field of data mining and machine learning in the past two weeks.

Our next example is a bit more complex. We're querying tweets from Hillary Clinton and Donald Trump from the presidential campaign 2016.



Twitter

Twitter API Key

Query

Query word list: HillaryClinton
realDonaldTrump

Search by: Author

Allow retweets: ☐

Date: since 2016-09-30 until 2016-10-10

Language: English

Max tweets: ☐ 100

Accumulate results: ☐

Text includes

☑ Content ☐ Author Description

Info

Tweets on output: 272

Report Search

Word Cloud

Info

0 words in a topic
272 documents with 1285 words

Cloud preferences

☑ Color words

Words tilt: no

Regenerate word cloud

Words & weights

Weight	Word
70	trump
40	#vpdebate
39	pence
36	hillary
30	mike
28	make
27	donald
22	@realdonaldtrump
20	great

Test & Score

Sampling

(●) Cross validation
Number of folds: 10
☑ Stratified
( ) Random sampling
Repeat train/test: 10
Training set size: 66 %
☑ Stratified
( ) Leave one out
( ) Test on train data
( ) Test on test data

Target Class

(Average over classes)

Report

Evaluation Results

Method	AUC	CA	F1	Precision	Recall
Logistic Regression	0.763	0.820	0.692	0.887	0.567

Then we've used Preprocess Text to get suitable tokens on our output. We've connected **Preprocess Text** to Bag of Words in order to create a table with words as features and their counts as values. A quick check in **Word Cloud** gives us an idea about the results.

Now we would like to predict the author of the tweet. With **Select Columns** we're setting 'Author' as our target variable. Then we connect **Select Columns** to **Test & Score**. We'll be using **Logistic Regression** as our learner, which we also connect to **Test & Score**.

We can observe the results of our author predictions directly in the widget. AUC score is quite ok. Seems like we can to some extent predict who is the author of the tweet based on the tweet content.

Wikipedia

Fetching data from MediaWiki RESTful web service API.

Signals

Inputs:

- (None)

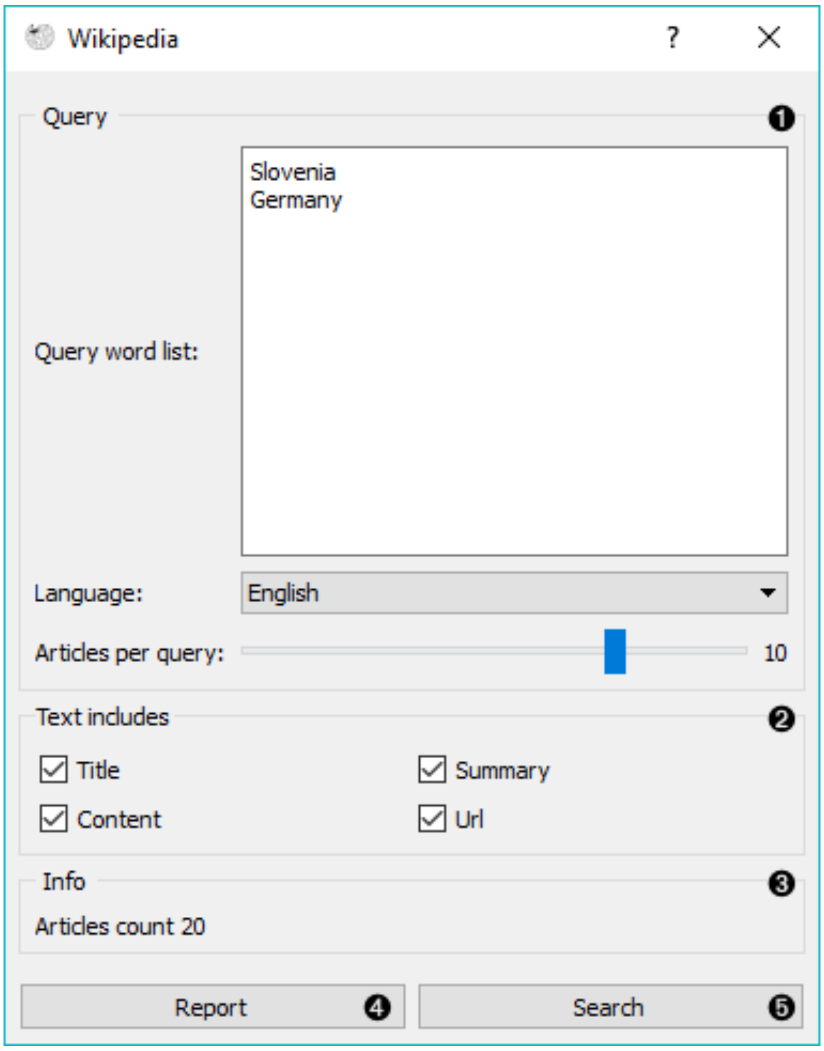
Outputs:

- **Corpus**

  A Corpus instance.

Description

Wikipedia widget is used to retrieve texts from Wikipedia API and it is useful mostly for teaching and demonstration.


Wikipedia

Query

Query word list:

Slovenia
Germany

Language: English

Articles per query: 10

Text includes

- [x] Title
- [x] Summary
- [x] Content
- [x] Url

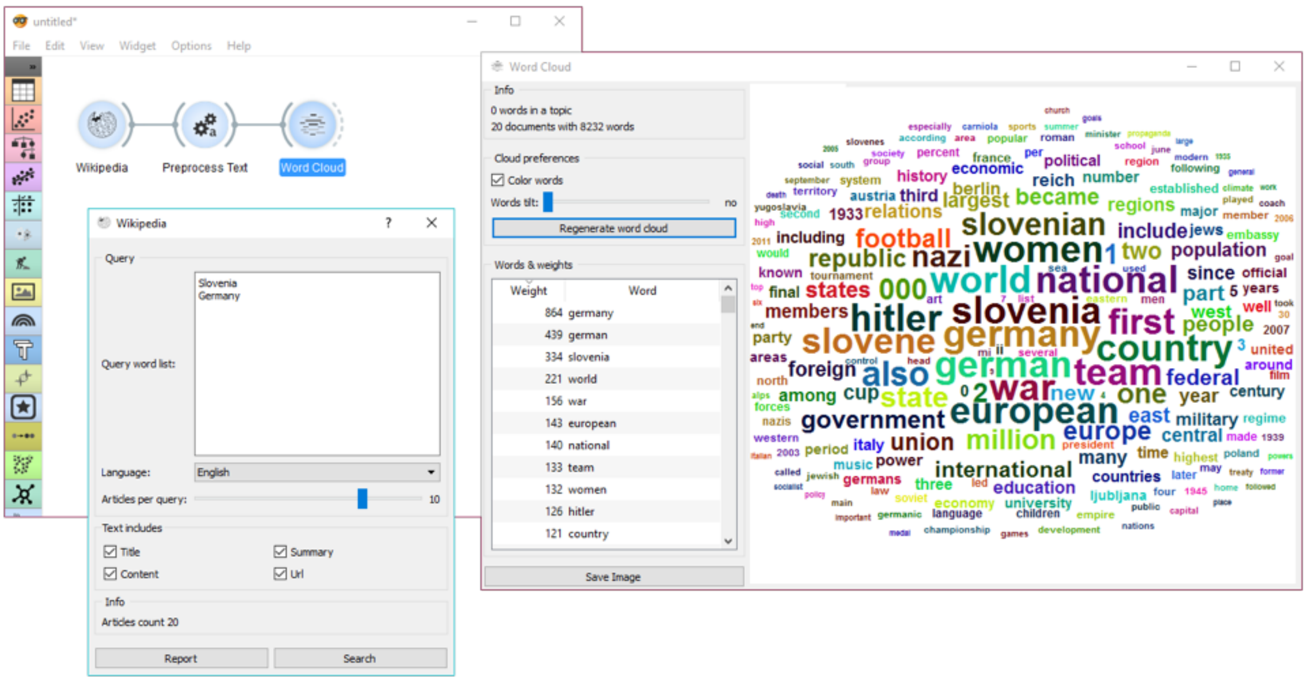
Info

Articles count 20

1. Query parameters:
   - Query word list, where each query is listed in a new line.
   - Language of the query. English is set by default.
   - Number of articles to retrieve per query (range 1-25). Please note that querying is done recursively and that disambiguations are also retrieved, sometimes resulting in a larger number of queries than set on the slider.
2. Select which features to include as text features.
3. Information on the output.
4. Produce a report.
5. Run query.

Example

This is a simple example, where we use **Wikipedia** and retrieve the articles on ‘Slovenia’ and ‘Germany’. Then we simply apply default preprocessing with Preprocess Text and observe the most frequent words in those articles with Word Cloud.



Wikipedia

Query

Query word list:
Slovenia
Germany

Language: English

Articles per query: 10

Text includes

☑ Title ☑ Summary

☑ Content ☑ Url

Info

Articles count 20

Report Search

Word Cloud

Info

0 words in a topic

20 documents with 8232 words

Cloud preferences

☑ Color words

Words tilt: no

Regenerate word cloud

Words & weights

Weight	Word
864	germany
439	german
334	slovenia
221	world
156	war
143	european
140	national
133	team
132	women
126	hitler
121	country

Save Image

Wikipedia works just like any other corpus widget (NY Times, Twitter) and can be used accordingly.

v: latest

Word Cloud

Generates a word cloud from corpus.

Signals

Inputs:

- **Topic**

  Selected topic.

- **Corpus**

  A Corpus instance.

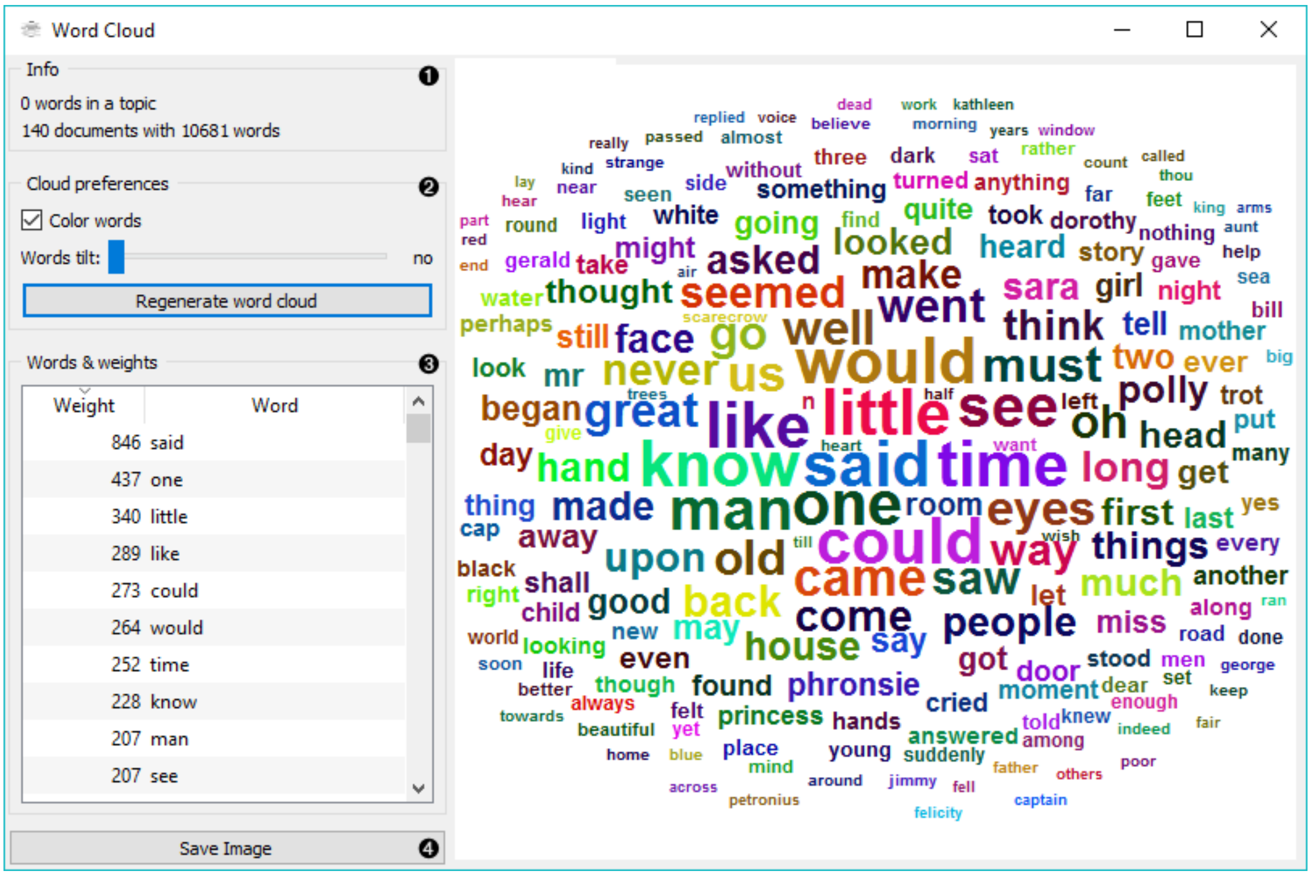
Outputs:

- **Corpus**

  Documents that match the selection.

Description

Word Cloud displays tokens in the corpus, their size denoting the frequency of the word in corpus. Words are listed by their frequency (weight) in the widget. The widget outputs documents, containing selected tokens from the word cloud.



Word Cloud

Info ❶

0 words in a topic
140 documents with 10681 words

Cloud preferences ❷

☑ Color words

Words tilt: no

Regenerate word cloud

Words & weights ❸

Weight	Word
846	said
437	one
340	little
289	like
273	could
264	would
252	time
228	know
207	man
207	see

Save Image ❹

1. Information on the input.
   - number of words (tokens) in a topic
   - number of documents and tokens in the corpus

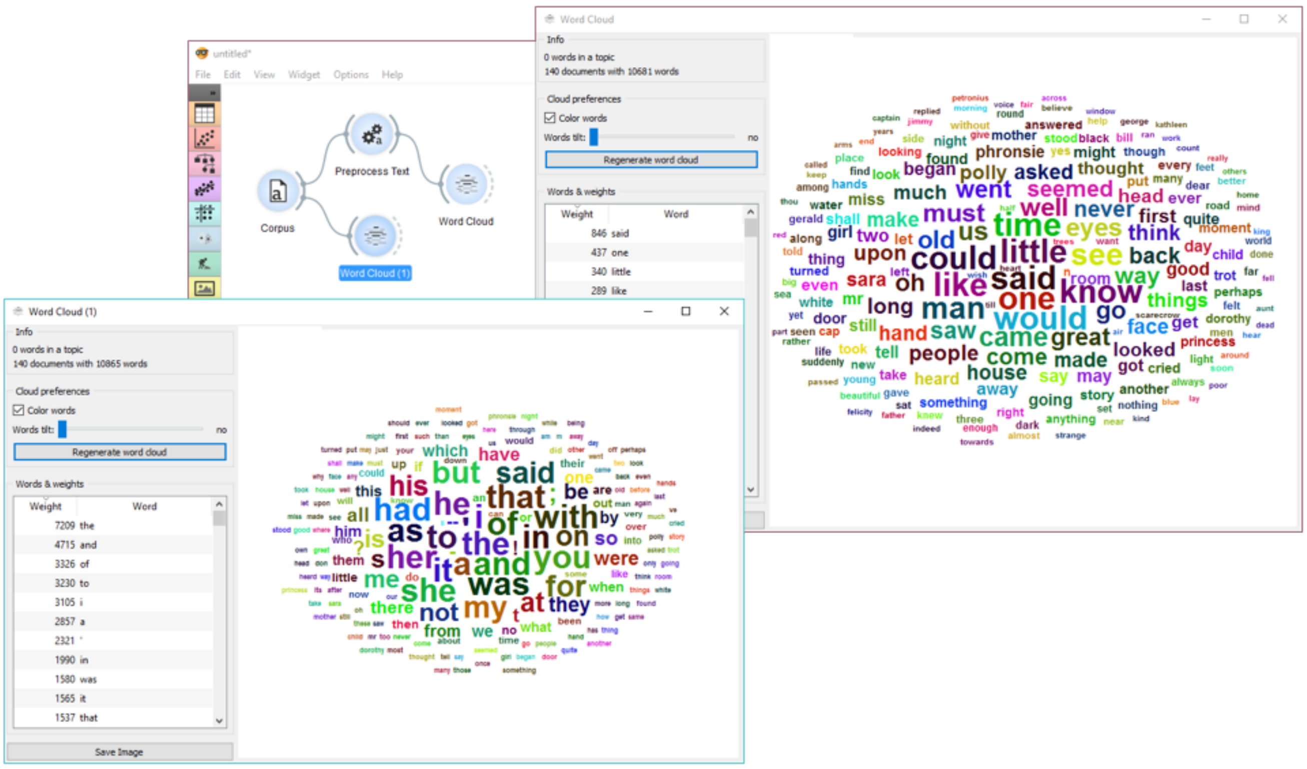
2. Adjust the plot.
   - If *Color words* is ticked, words will be assigned a random color. If unchecked, the words will be black.
   - *Word tilt* adjust the tilt of words. The current state of tilt is displayed next to the slider (‘no’ is the default).
   - *Regenerate word cloud* plot the cloud anew.

3. Words & weights displays a sorted list of words (tokens) by their frequency in the corpus or topic. Clicking on a word will select that same word in the cloud and output matching documents. Use *Ctrl* to select more than one word. Documents matching ANY of the selected words will be on the output (logical OR).
4. *Save Image* saves the image to your computer in a .svg or .png format.

Example

Word Cloud is an excellent widget for displaying the current state of the corpus and for monitoring the effects of preprocessing.

Use Corpus to load the data. Connect Preprocess Text to it and set your parameters. We’ve used defaults here, just to see the difference between the default preprocessing in the **Word Cloud** widget and the **Preprocess Text** widget.



Word Cloud

Info
0 words in a topic
140 documents with 10681 words

Cloud preferences
☑ Color words
Words tilt: no
Regenerate word cloud

Words & weights

Weight	Word
846	said
437	one
340	little
289	like

Word Cloud (1)

Info
0 words in a topic
140 documents with 10865 words

Cloud preferences
☑ Color words
Words tilt: no
Regenerate word cloud

Words & weights

Weight	Word
7209	the
4715	and
3326	of
3230	to
3105	i
2857	a
2321	'
1990	in
1580	was
1565	it
1537	that

Save Image

We can see from the two widgets, that **Preprocess Text** displays only words, while default preprocessing in the **Word Cloud** tokenizes by word and punctuation.

Word Enrichment

Word enrichment analysis for selected documents.

Signals

Inputs:

- **Data**

  Corpus instance.

- **Selected Data**

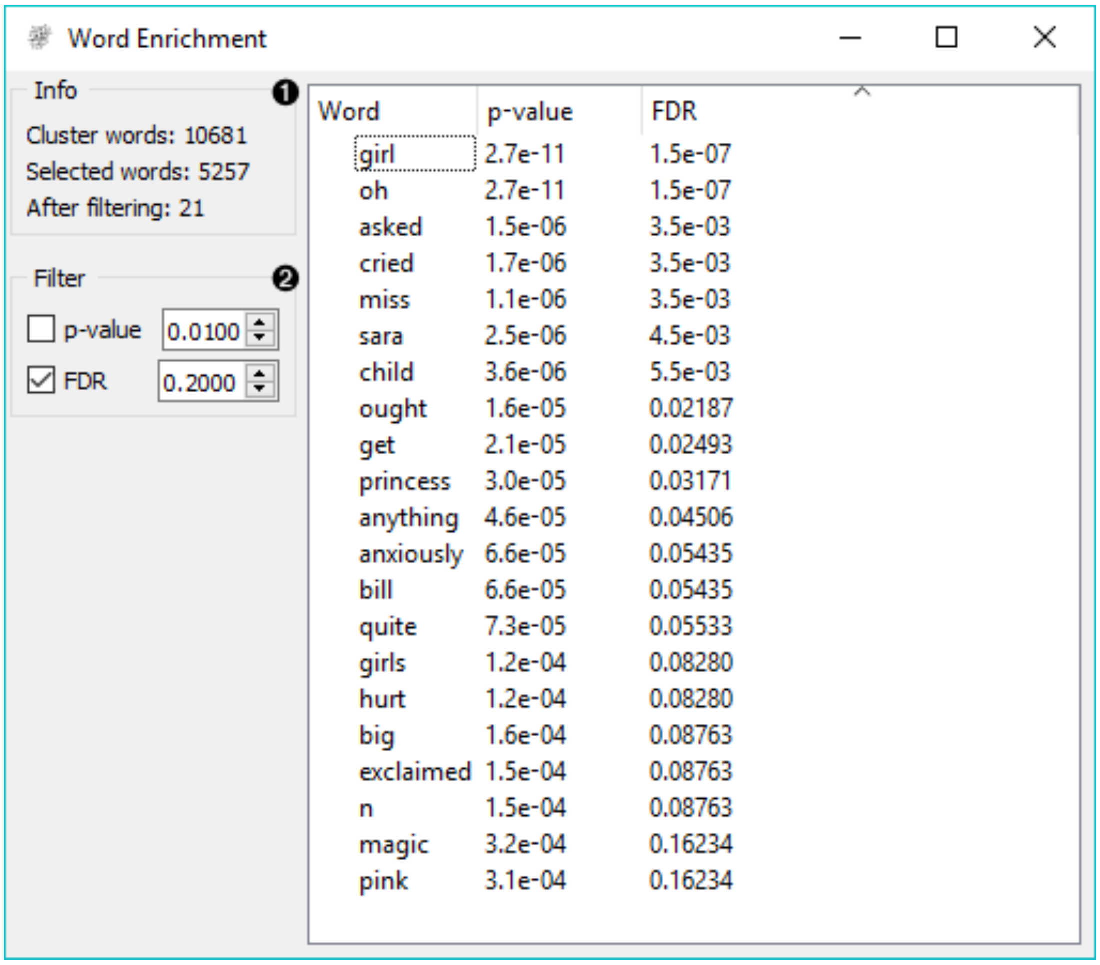
  Selected instances from corpus.

Outputs:

- (None)

Description

Word Enrichment displays a list of words with lower p-values (higher significance) for a selected subset compared to the entire corpus. Lower p-value indicates a higher likelihood that the word is significant for the selected subset (not randomly occurring in a text). FDR (False Discovery Rate) is linked to p-value and reports on the expected percent of false predictions in the set of predictions, meaning it account for false positives in list of low p-values.



Cluster words: 10681
Selected words: 5257
After filtering: 21

☐ p-value 0.0100
☑ FDR 0.2000

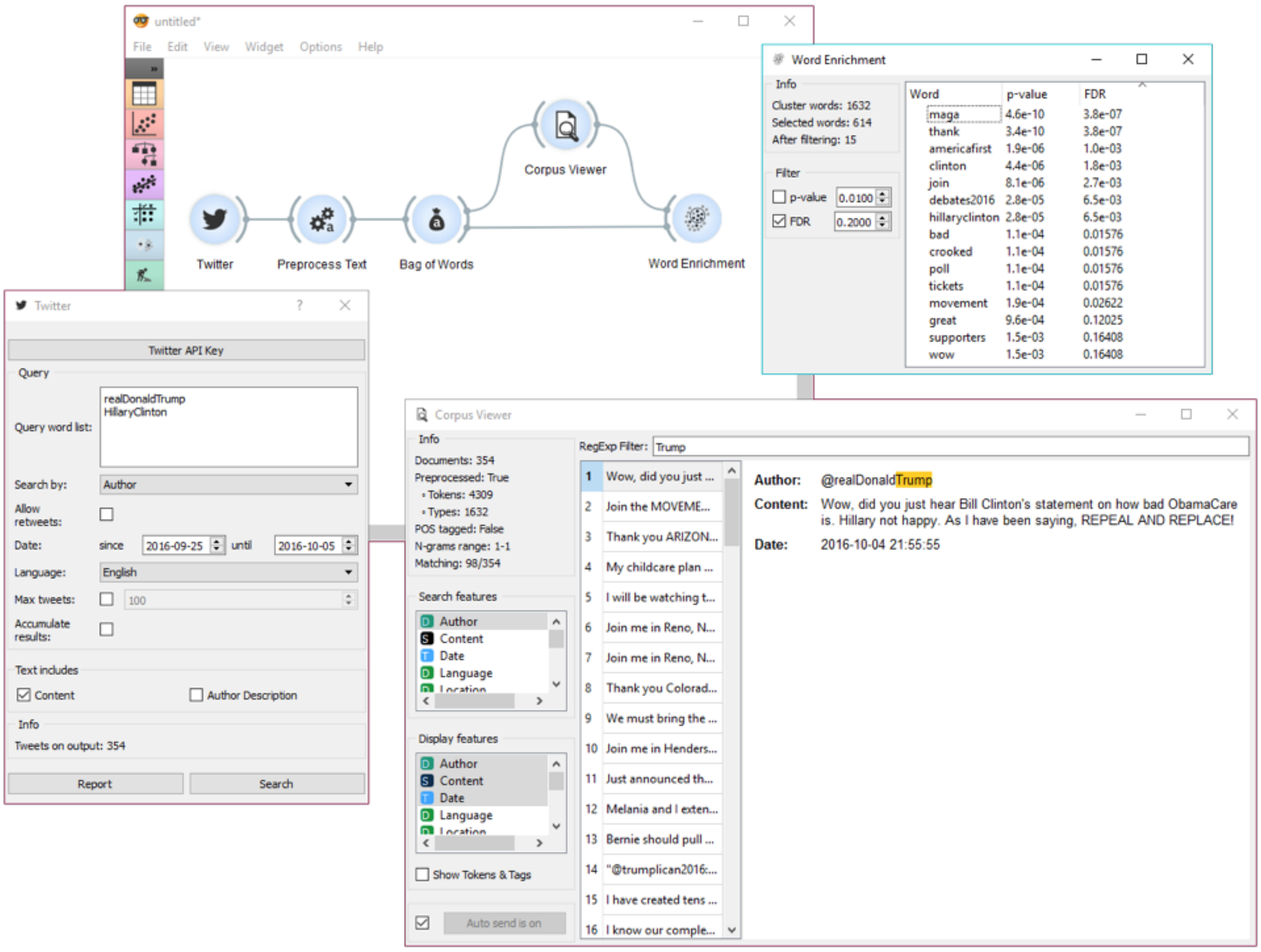
Word	p-value	FDR
girl	2.7e-11	1.5e-07
oh	2.7e-11	1.5e-07
asked	1.5e-06	3.5e-03
cried	1.7e-06	3.5e-03
miss	1.1e-06	3.5e-03
sara	2.5e-06	4.5e-03
child	3.6e-06	5.5e-03
ought	1.6e-05	0.02187
get	2.1e-05	0.02493
princess	3.0e-05	0.03171
anything	4.6e-05	0.04506
anxiously	6.6e-05	0.05435
bill	6.6e-05	0.05435
quite	7.3e-05	0.05533
girls	1.2e-04	0.08280
hurt	1.2e-04	0.08280
big	1.6e-04	0.08763
exclaimed	1.5e-04	0.08763
n	1.5e-04	0.08763
magic	3.2e-04	0.16234
pink	3.1e-04	0.16234

1. Information on the input.
    - Cluster words are all the tokens from the corpus.
    - Selected words are all the tokens from the selected subset.
    - After filtering reports on the enriched words found in the subset.

2. Filter enables you to filter by:
    - p-value
    - false discovery rate (FDR)

Example

In the example below, we're retrieved recent tweets from the 2016 presidential candidates, Donald Trump and Hillary Clinton. Then we've preprocessed the tweets to get only words as tokens and to remove the stopwords. We've connected the preprocessed corpus to Bag of Words to get a table with word counts for our corpus.



Twitter

Twitter API Key

Query

Query word list: realDonaldTrump
HillaryClinton

Search by: Author

Allow retweets: ☐

Date: since 2016-09-25 until 2016-10-05

Language: English

Max tweets: ☐ 100

Accumulate results: ☐

Text includes

☑ Content ☐ Author Description

Info

Tweets on output: 354

Report Search

Word Enrichment

Info

Cluster words: 1632
Selected words: 614
After filtering: 15

Filter

☐ p-value 0.0100
☑ FDR 0.2000

Word	p-value	FDR
maga	4.6e-10	3.8e-07
thank	3.4e-10	3.8e-07
americafirst	1.9e-06	1.0e-03
clinton	4.4e-06	1.8e-03
join	8.1e-06	2.7e-03
debates2016	2.8e-05	6.5e-03
hillaryclinton	2.8e-05	6.5e-03
bad	1.1e-04	0.01576
crooked	1.1e-04	0.01576
poll	1.1e-04	0.01576
tickets	1.1e-04	0.01576
movement	1.9e-04	0.02622
great	9.6e-04	0.12025
supporters	1.5e-03	0.16408
wow	1.5e-03	0.16408

Corpus Viewer

Info

Documents: 354
Preprocessed: True
- Tokens: 4309
- Types: 1632
POS tagged: False
N-grams range: 1-1
Matching: 98/354

Search features

Author
Content
Date
Language
Location

Display features

Author
Content
Date
Language
Location

☐ Show Tokens & Tags

☑ Auto send is on

RegExp Filter: Trump

1 Wow, did you just ...
2 Join the MOVEME...
3 Thank you ARIZON...
4 My childcare plan ...
5 I will be watching t...
6 Join me in Reno, N...
7 Join me in Reno, N...
8 Thank you Colorad...
9 We must bring the ...
10 Join me in Henders...
11 Just announced th...
12 Melania and I exten...
13 Bernie should pull ...
14 "@trumplican2016:...
15 I have created tens ...
16 I know our comple...

Author: @realDonaldTrump
Content: Wow, did you just hear Bill Clinton's statement on how bad ObamaCare is. Hillary not happy. As I have been saying, REPEAL AND REPLACE!
Date: 2016-10-04 21:55:55

Then we've connected Corpus Viewer to **Bag of Words** and selected only those tweets that were published by Donald Trump. See how we marked only the *Author* as our *Search feature* to retrieve those tweets.

Word Enrichment accepts two inputs - the entire corpus to serve as a reference and a selected subset from the corpus to do the enrichment on. First connect **Corpus Viewer** to **Word Enrichment** (input Matching Docs → Selected Data) and then connect **Bag of Words** to it (input Corpus → Data). In the **Word Enrichment** widget we can see the list of words that are more significant for Donald Trump than they are for Hillary Clinton.

Network

Network File

Network Explorer

Network Generator

Network Analysis

Network Clustering

Network from Distances

Network Analysis

Statistical analysis of network data.

Signals

Inputs:

- **Network**

  An instance of Network Graph.

- **Items**

  Properties of a network file.

Outputs:

- **Network**

  An instance of Network Graph with appended information.

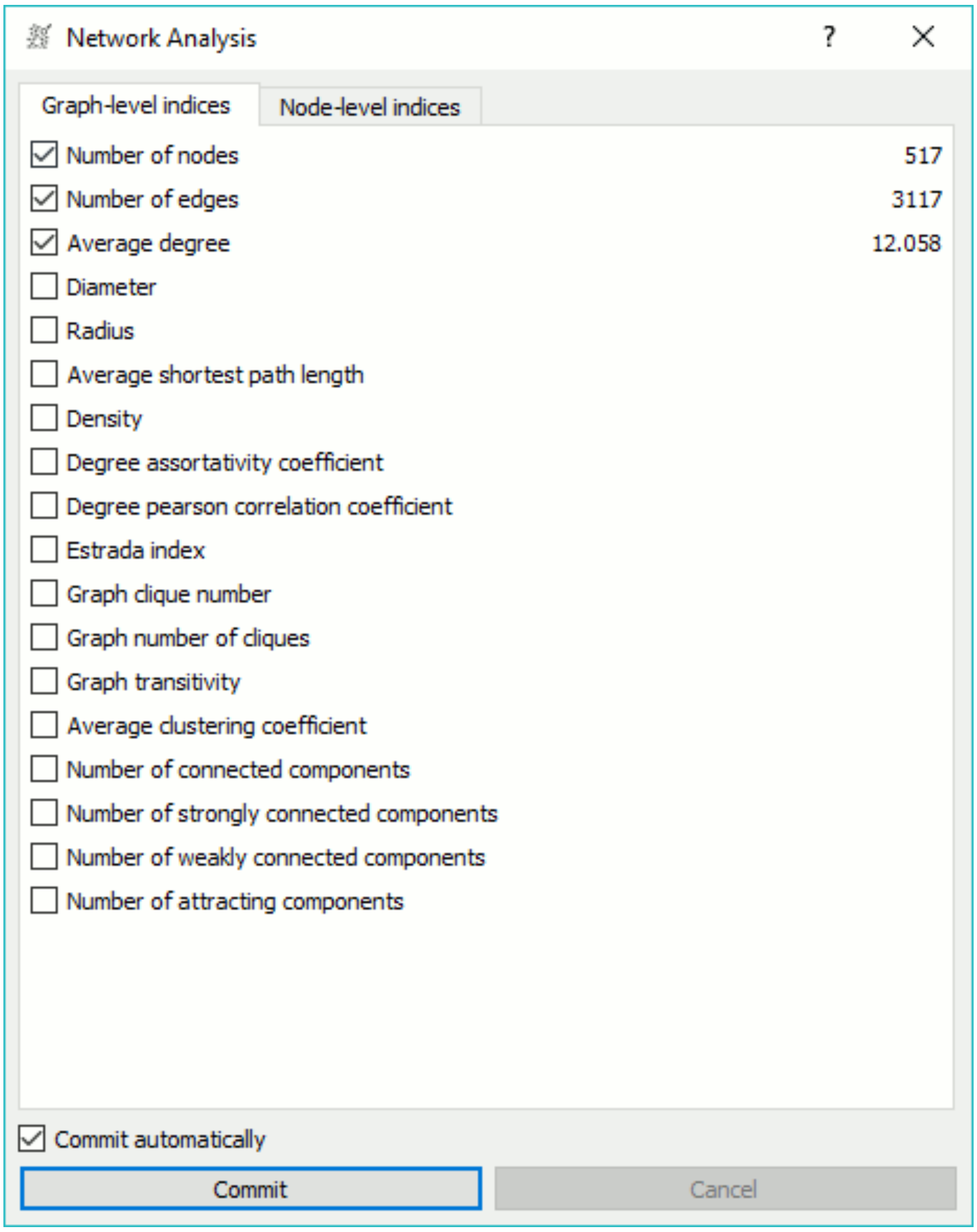
- **Items**

  New properties of a network file.

Description

Network Analysis widget computes node-level and graph-level summary statistics for the network. It can output a network with the new computed statistics appended or an extended item data table.

Graph level


Network Analysis

Graph-level indices | Node-level indices

- [x] Number of nodes 517
- [x] Number of edges 3117
- [x] Average degree 12.058
- [ ] Diameter
- [ ] Radius
- [ ] Average shortest path length
- [ ] Density
- [ ] Degree assortativity coefficient
- [ ] Degree pearson correlation coefficient
- [ ] Estrada index
- [ ] Graph clique number
- [ ] Graph number of cliques
- [ ] Graph transitivity
- [ ] Average clustering coefficient
- [ ] Number of connected components
- [ ] Number of strongly connected components
- [ ] Number of weakly connected components
- [ ] Number of attracting components

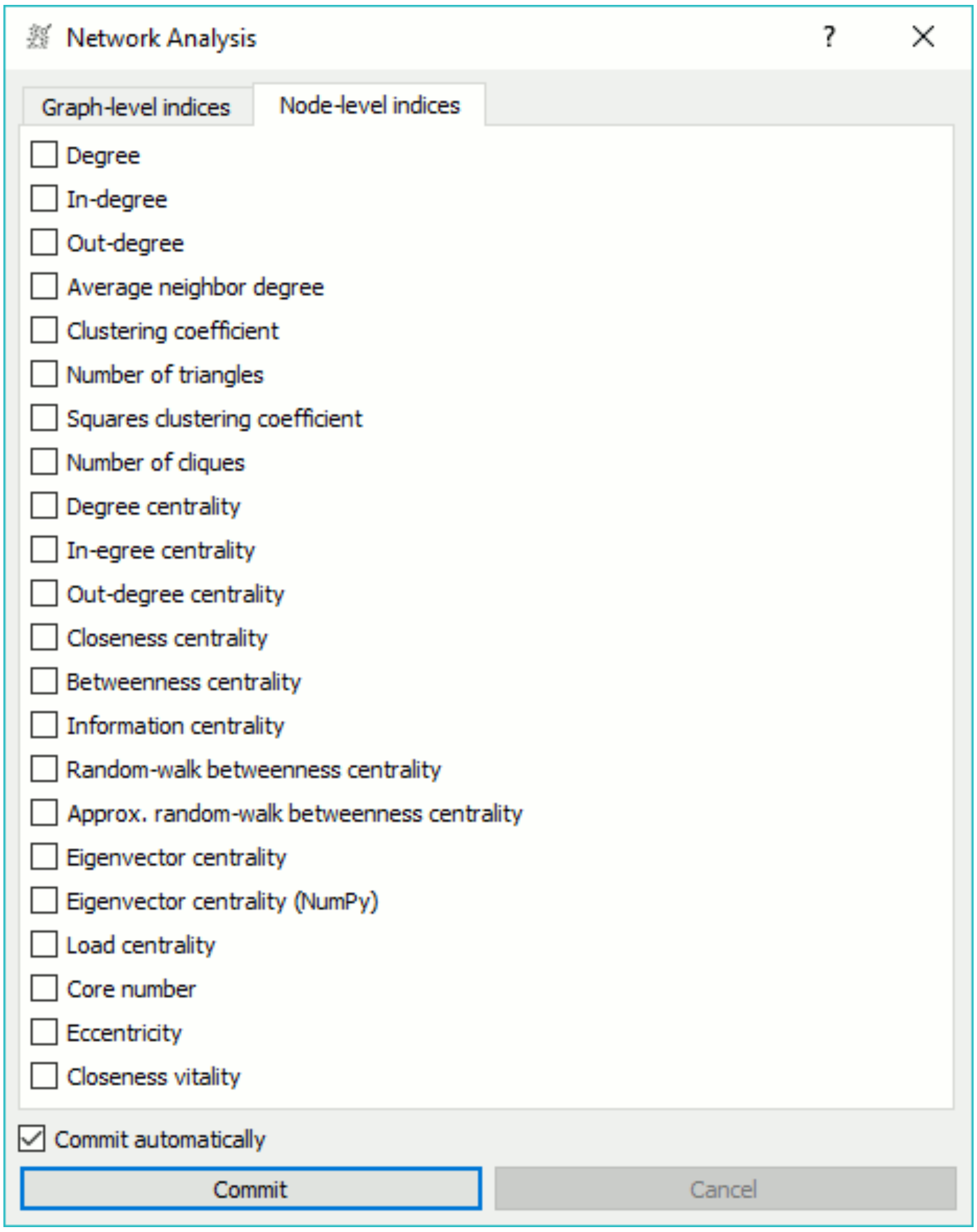
- [x] Commit automatically

Commit | Cancel

- Number of nodes: number of vertices in a network.
- Number of edges: number of connections in a network.
- Average degree: average number of connections per node.
- Diameter: maximum eccentricity of the graph.
- Radius: minimum eccentricity of the graph.
- Average shortest path length: expected distance between two nodes in the graph.
- Density: ratio between actual number of edges and maximum number of edges (fully connected graph).
- Degree assortativity coefficient: correlations between nodes of similar degree.
- Degree pearson correlation coefficient: same as degree assortativity coefficient but with a scipy.stats.pearsonr function.
- Estrada index: Estrada index of the graph.
- Graph clique number: number of nodes in the largest clique (size of a clique).
- Graph number of cliques: number of cliques (subsets of nodes, where every two nodes are connected).
- Graph transitivity: ratio of all possible triangles in the network (if node A connects to B and C, how often are B and C connected in the graph).

- Average clustering coefficient: average of the local clustering coefficients of all the vertices.
- Number of connected components: number of separate networks in a graph
- Number of strongly connected components: parts of network where every vertex is reachable from every other vertex (for directed graphs only).
- Number of weakly connected components: parts of network where replacing all of its directed edges with undirected edges produces a connected (undirected) graph (for directed graphs only).
- Number of attracting components: node in a direct graph that a random walker in a graph cannot leave (for directed graphs only).

Node level


Network Analysis

Graph-level indices | Node-level indices

- [ ] Degree
- [ ] In-degree
- [ ] Out-degree
- [ ] Average neighbor degree
- [ ] Clustering coefficient
- [ ] Number of triangles
- [ ] Squares clustering coefficient
- [ ] Number of cliques
- [ ] Degree centrality
- [ ] In-egree centrality
- [ ] Out-degree centrality
- [ ] Closeness centrality
- [ ] Betweenness centrality
- [ ] Information centrality
- [ ] Random-walk betweenness centrality
- [ ] Approx. random-walk betweenness centrality
- [ ] Eigenvector centrality
- [ ] Eigenvector centrality (NumPy)
- [ ] Load centrality
- [ ] Core number
- [ ] Eccentricity
- [ ] Closeness vitality

- [x] Commit automatically

Commit | Cancel

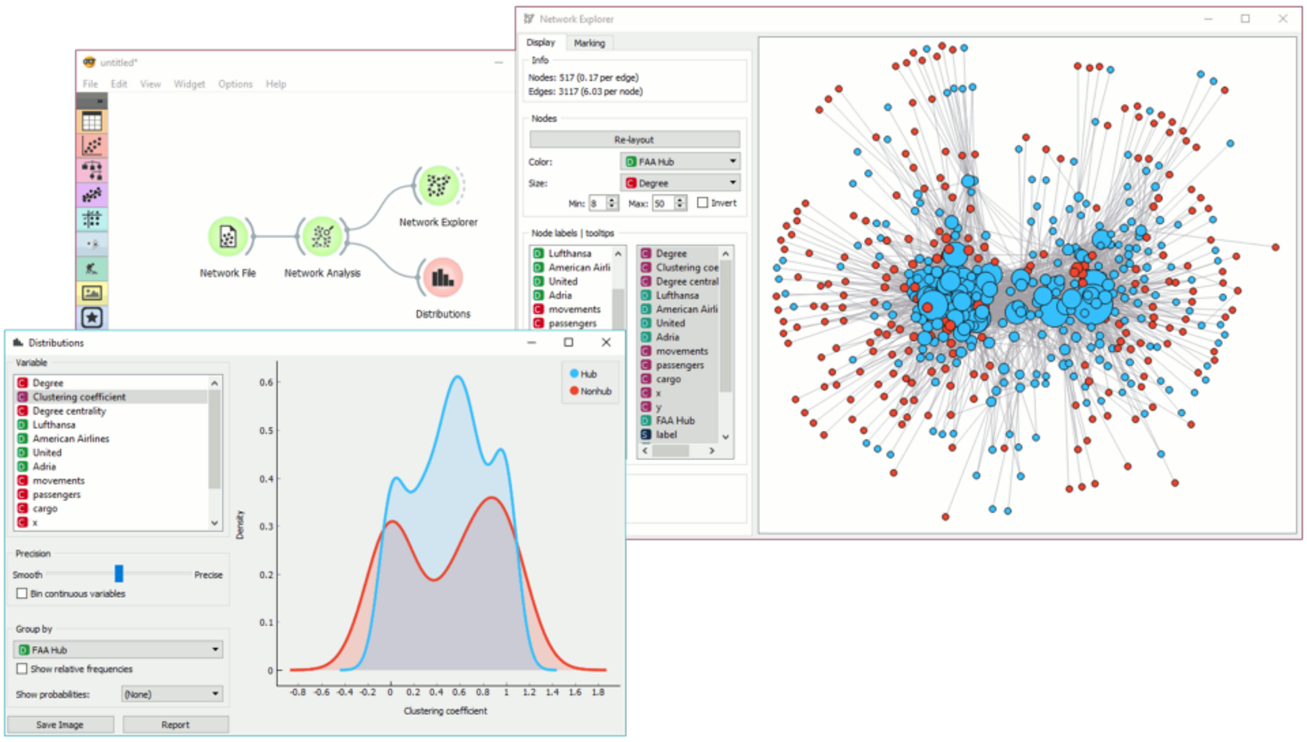
- Degree: number of edges per node.
- In-degree: number of incoming edges in a directed graph.
- Out-degree: number of outgoing edges in a directed graph.
- Average neighbor degree: average degree of neighboring nodes.

- Clustering coefficient: ratio of triangles in a node neighborhood to all possible triangles.
- Number of triangles: number of triangles that include a node as one vertex.
- Squares clustering coefficient: ratio of possible squares that exist for a node.
- Number of cliques: number of complete (fully connected) subgraphs in a network.
- Degree centrality: ratio of other nodes connected to the node.
- In-degree centrality: ratio of incoming edges to a node in a directed graph.
- Out-degree centrality: ratio of outgoing edges from a node in directed graph.
- Closeness centrality: distance to all other nodes.
- Betweenness centrality: measure of control a node exerts over the interaction of other nodes in the network.
- Information centrality: proportion of total information flow that is controlled by each node.
- Random-walk betweenness centrality: number of times a node would be on the path between two nodes if employing a random walk.
- Approx. random-walk betweenness centrality: approximate current-flow betweenness centrality.
- Eigenvector centrality: score nodes by their connections to high-scoring nodes (measure of centrality of a node based on its connection to other central nodes).
- Eigenvector centrality (NumPy): eigenvector centrality with NumPy eigenvalue solver.
- Load centrality: ratio of all shortest paths that lead through the node.
- Core number: largest value k of a k-core containing that node.
- Eccentricity: maximum distance between the node and every other node in the network.
- Closeness vitality: change in the sum of distances for all node pairs when excluding that node.

If *Commit automatically* is on, new information will be commited automatically. Alternatively, press *Commit*.

Example

This simple example shows how **Network Analysis** can enrich the workflow. We have used *airtraffic.net* as our input network from *Network File* and sent it to **Network Analysis**. We've decided to compute *density*, *number of cliques* and *graph transitivity* at graph level and *degree*, *clustering coefficient* and *degree centrality* at node level. The widget instantly computes score for graph-level methods and displays them in the widget. It also computes scores for node-level methods, appends them as additional columns and outputs them as *Items*.



We can use node-level scores with **Distributions** widget to observe, say, clustering coefficient distribution or set the size of nodes in *Network Explorer* to *Degree*.

Network Clustering

Detect clusters in a network.

Signals

Inputs:

- **Network**

  An instance of Network Graph.

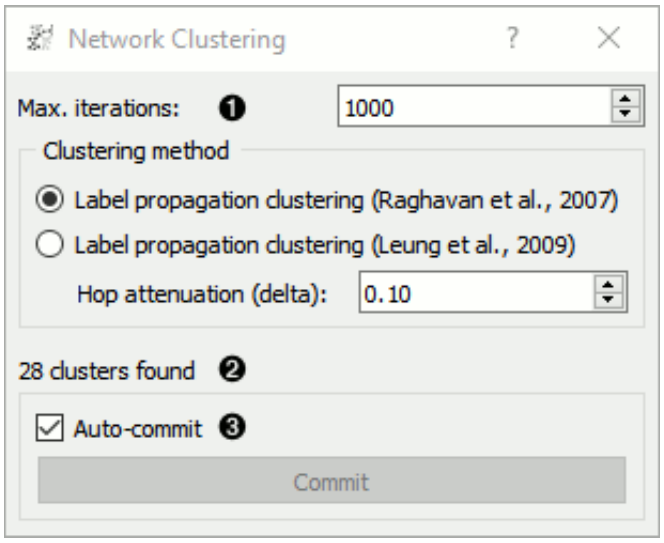
Outputs:

- **Network**

  An instance of Network Graph with clustering information appended.

Description

Network Clustering widget finds clusters in a network. Clustering works with two algorithms, one from Raghavan et al. (2007), which uses label propagation to find appropriate clusters, and one from Leung et al. (2009), which builds upon the work from Raghavan and adds hop attenuation as a parameters for cluster formation.


Network Clustering

Max. iterations: ❶ 1000

Clustering method

[✓ Label propagation clustering (Raghavan et al., 2007)]

Label propagation clustering (Leung et al., 2009)

Hop attenuation (delta): 0.10

28 clusters found ❷

[✓] Auto-commit ❸

1. Clustering parameters: - Max. iterations: maximum number of iteration allowed for the algorithm to run (can converge before reaching the maximum). - Clustering method:

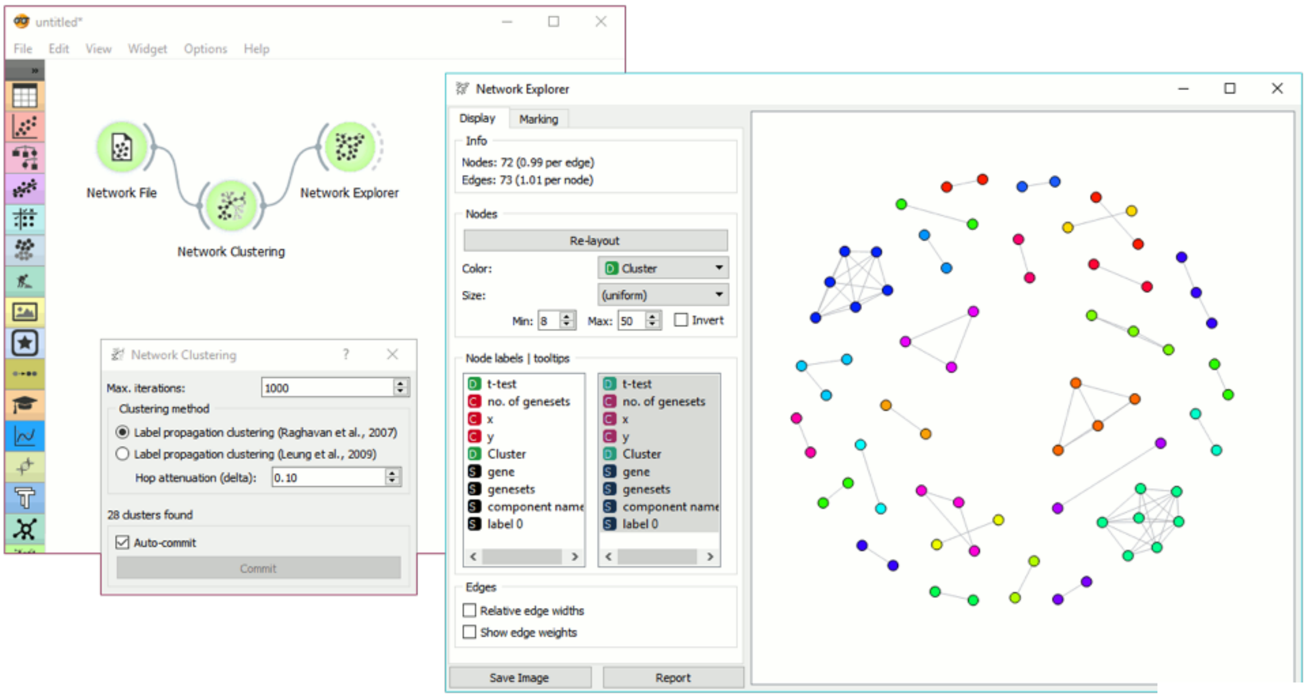
   - Label propagation clustering (Raghavan et al., 2007)
   - Label propagation clustering (Leung et al., 2009) with hop attenuation.

2. Information on the number of clusters found.

3. If *Auto-commit* is ticked, results will be automatically sent to the output. Alternatively, press *Commit*.

Example

Network Clustering can help you uncover cliques and highly connected groups in a network. First, we will use *Network File* to load *leu_by_genesets.net* data set. Then we will pass the network through **Network Clustering**. The widget found 28 clusters in a network. To visualize the results, use *Network Explorer* and set *Color* attribute to *Cluster*. This will color network nodes with the corresponding cluster color - this is a great way to visualize highly connected groups in dense networks.



Network Clustering

Max. iterations: 1000

Clustering method

(●) Label propagation clustering (Raghavan et al., 2007)

( ) Label propagation clustering (Leung et al., 2009)

Hop attenuation (delta): 0.10

28 clusters found

☑ Auto-commit

Commit

Network Explorer

Info

Nodes: 72 (0.99 per edge)

Edges: 73 (1.01 per node)

Nodes

Re-layout

Color: Cluster

Size: (uniform)

Min: 8 Max: 50 ☐ Invert

Node labels | tooltips

t-test	t-test
no. of genesets	no. of genesets
x	x
y	y
Cluster	Cluster
gene	gene
genesets	genesets
component name	component name
label 0	label 0

Edges

☐ Relative edge widths

☐ Show edge weights

Save Image Report

Network Explorer

Visually explore the network and its properties.

Signals

Inputs:

- **Network**

  An instance of Network Graph.

- **Node Subset**

  A subset of vertices.

- **Node Data**

  Information on vertices.

- **Node Distances**

  Data on distances between nodes.

Outputs:

- **Selected sub-network**

  A network of selected nodes.

- **Distance Matrix**

  Distance matrix.

- **Selected Items**

  Information on selected vertices.

- **Highlighted Items**

  Information on highlighted vertices.

- **Remaining Items**

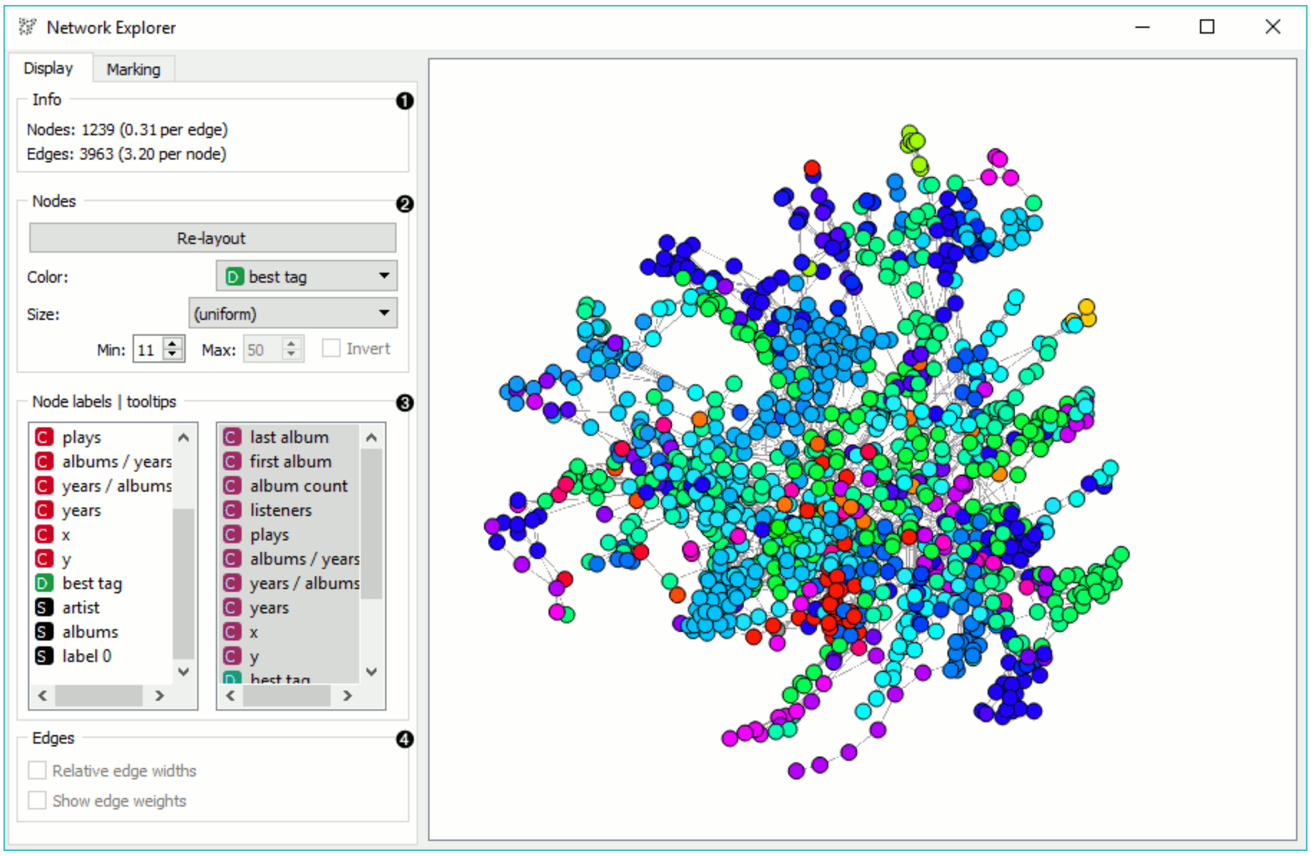
  Information on remaining items (not selected or highlighted).

Description

Network Explorer is the primary widget for visualizing network graphs. It displays a graph with Fruchterman-Reingold layout optimization and enables setting the color, size and label of nodes. One can also highlight nodes of specific properties and output them.

Nodes can be moved around freely as their position in space is not fixed (only optimized). To select a subset of nodes, draw a rectangle around the subset. To highlight the nodes, set the criterium in *Marking* tab and press Enter to turn highlighted nodes (orange) into selected nodes (red). To use pan and move the network around, use the right click. Scroll in for zoom.

Display



Network Explorer

Display | Marking

Info ❶

Nodes: 1239 (0.31 per edge)
Edges: 3963 (3.20 per node)

Nodes ❷

Re-layout

Color: best tag

Size: (uniform)

Min: 11 Max: 50 Invert

Node labels | tooltips ❸

plays	last album
albums / years	first album
years / albums	album count
years	listeners
x	plays
y	albums / years
best tag	years / albums
artist	years
albums	x
label 0	y
	best tag

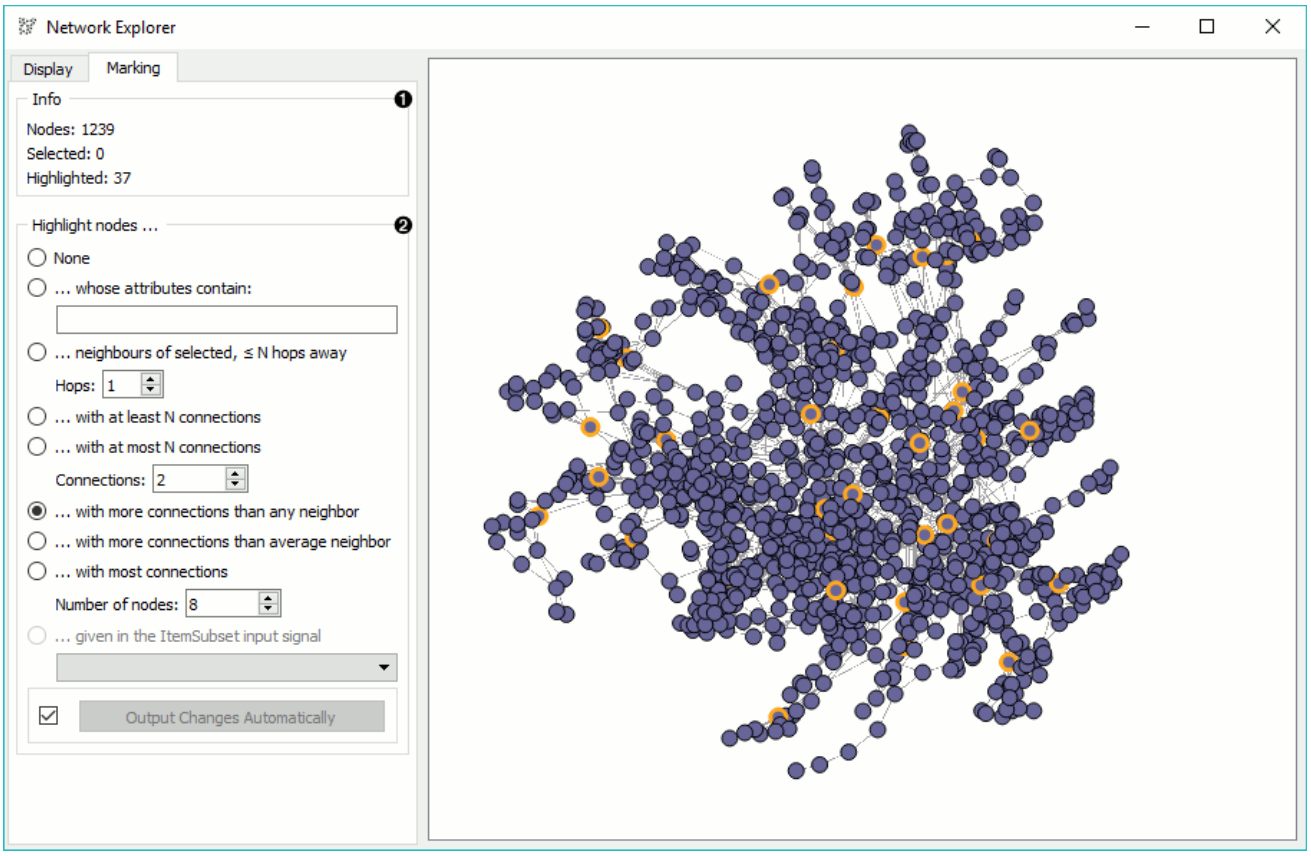
Edges ❹

Relative edge widths

Show edge weights

1. Information on the network. Reports on the number (and proportion) of nodes and edges.
2. Nodes: re-layout nodes with Fruchterman-Reingold optimization. Color and set the size of nodes by attribute. Set the maximum and minimum size of nodes and/or invert their sizing.
3. Node labels | tooltips: set node labels from the menu on the left and node tooltips from the menu on the right.
4. Edges: - If *Relative edge widths* is ticked, edges will have a thickness proportionate to their weight. Weights must be provided on the input for the option to be available. - If *Show edge weights* is ticked, weight will be displayed above the edges.

Marking

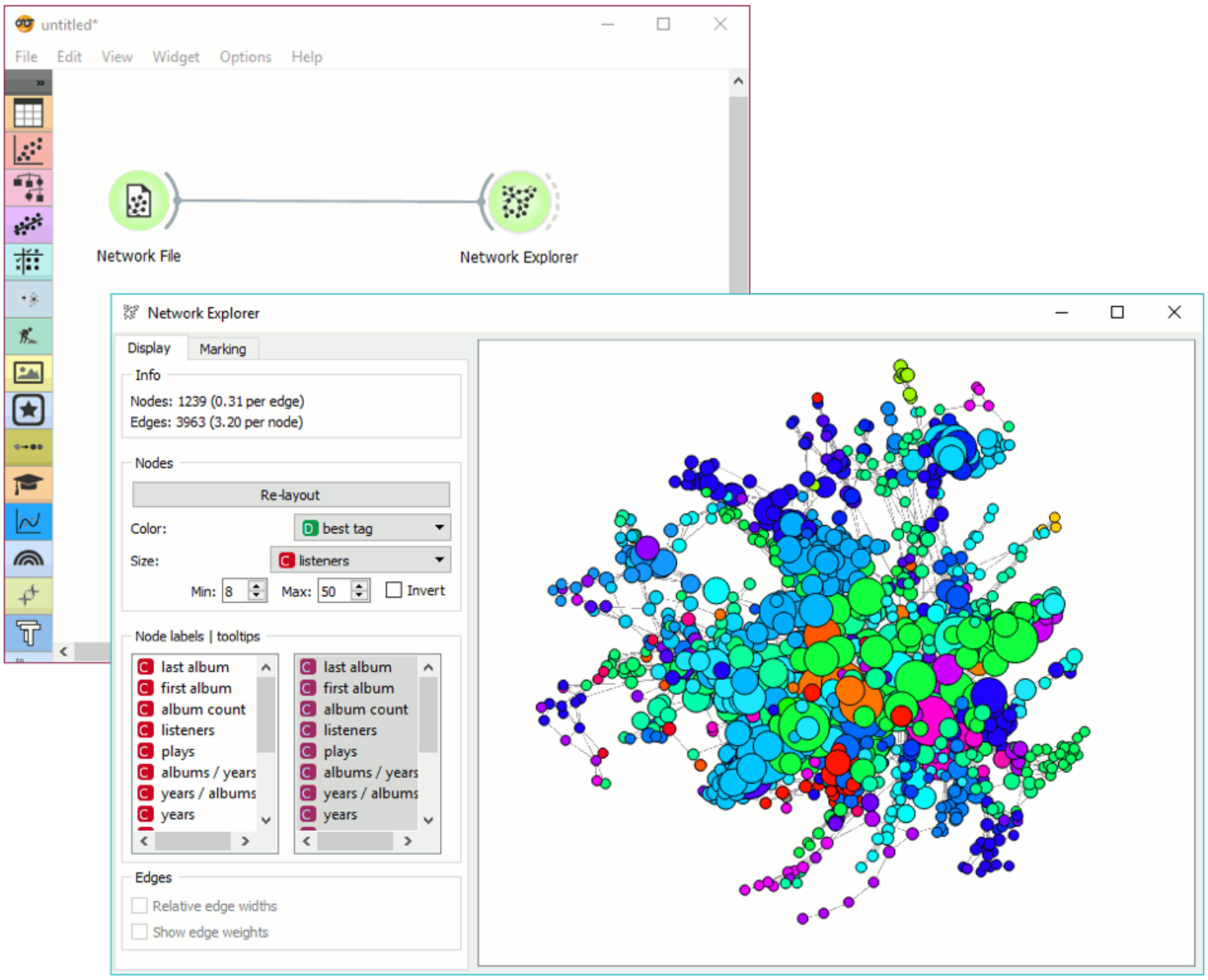


Network Explorer
Display
Marking
Info
Nodes: 1239
Selected: 0
Highlighted: 37
Highlight nodes ...
None
... whose attributes contain:
... neighbours of selected, ≤ N hops away
Hops: 1
... with at least N connections
... with at most N connections
Connections: 2
... with more connections than any neighbor
... with more connections than average neighbor
... with most connections
Number of nodes: 8
... given in the ItemSubset input signal
Output Changes Automatically

1. Information on the output. Reports on the number of nodes in the graph, selected nodes (red color), and highlighted nodes (orange color).
2. Highlight nodes: - None. Nodes are highlighted. - ...whose attributes contain. Nodes that satisfy a stated condition will be highlighted. - ...neighbors of selected, ≤ N hops away. Highlights nodes of selected points extending a specified number of hops away. - ...with at least N connections. With equal or more connections than specified in ‘Connections’. - ...with at most N connections. With less or equal connections than specified in ‘Connections’. - ...with more connections than any neighbor. Highlights well connected nodes (hubs). - ...with more connections than average neighbor. Highlights relatively well connected nodes. - ...with most connections. Highligts a specified number of well connected nodes. - ...given in the ItemSubset input signal. Highlights nodes matching the provided subset criteria (ID or other attribute). If ‘Output Changes Automatically’ is ticked, changes will be communicated automatically. Alternatively, press ‘Output Changes’.

Examples

In the first example we will simply display a network. We loaded *lastfm.net* data in *Network File* and send the data to **Network Explorer**. The widget shows an optimized projection of artist similarity data. We colored the nodes by ‘best tag’ attribute, showing different genres artists belong to, and set the size to the number of listeners per artist.



Network Explorer

Display | Marking

Info
Nodes: 1239 (0.31 per edge)
Edges: 3963 (3.20 per node)

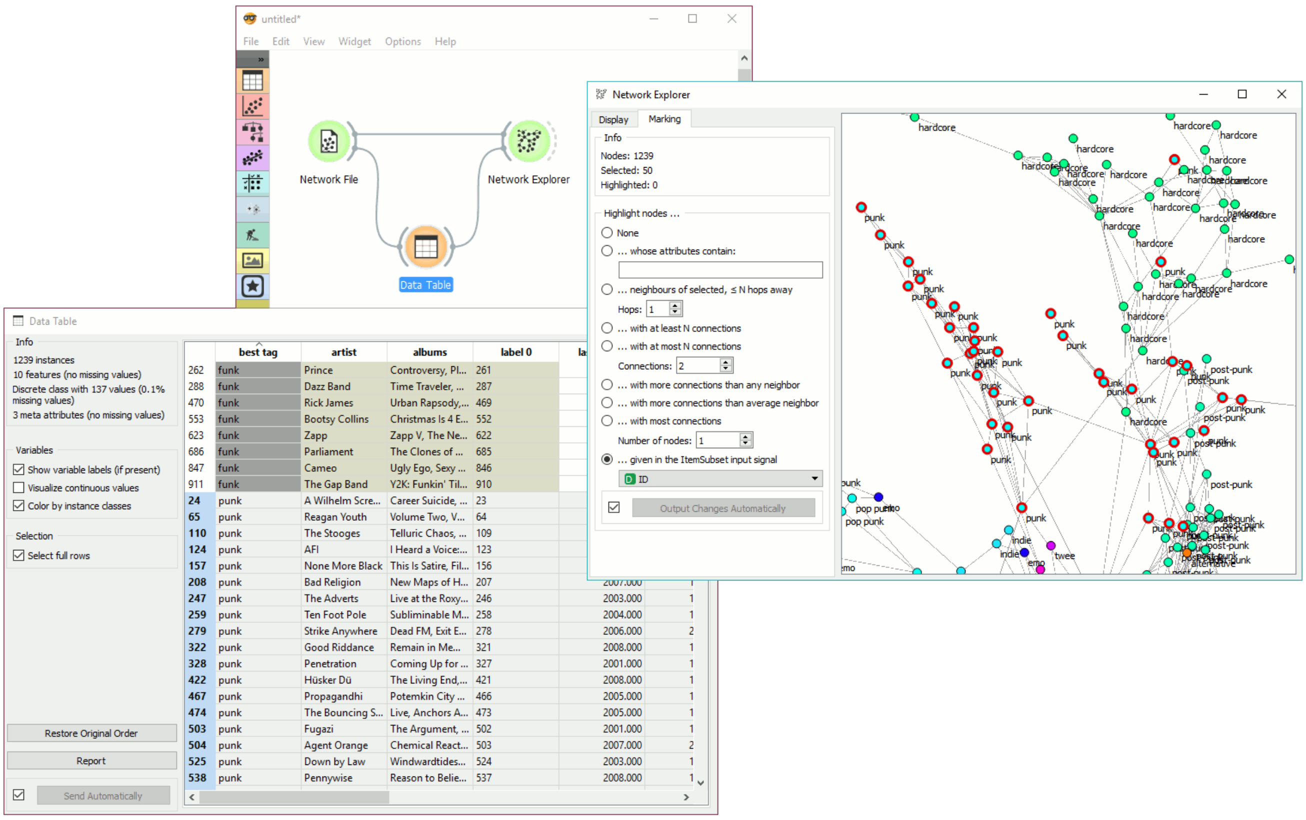
Nodes
Re-layout
Color: best tag
Size: listeners
Min: 8 Max: 50 ☐ Invert

Node labels | tooltips

Node labels	Tooltips
last album	last album
first album	first album
album count	album count
listeners	listeners
plays	plays
albums / years	albums / years
years / albums	years / albums
years	years

Edges
☐ Relative edge widths
☐ Show edge weights

The second example shows how to highlight a specific subset in the graph. We continue to use *lastfm.net* data from the *Network File*. We also retained connection to the **Network Explorer**.



Data Table

Info
1239 instances
10 features (no missing values)
Discrete class with 137 values (0.1% missing values)
3 meta attributes (no missing values)

Variables
☑ Show variable labels (if present)
☐ Visualize continuous values
☑ Color by instance classes

Selection
☑ Select full rows

Restore Original Order
Report
☑ Send Automatically

	best tag	artist	albums	label 0	la	
262	funk	Prince	Controversy, Pl...	261		
288	funk	Dazz Band	Time Traveler, ...	287		
470	funk	Rick James	Urban Rapsody,...	469		
553	funk	Bootsy Collins	Christmas Is 4 E...	552		
623	funk	Zapp	Zapp V, The Ne...	622		
686	funk	Parliament	The Clones of ...	685		
847	funk	Cameo	Ugly Ego, Sexy ...	846		
911	funk	The Gap Band	Y2K: Funkin' Til...	910		
24	punk	A Wilhelm Scre...	Career Suicide, ...	23		
65	punk	Reagan Youth	Volume Two, V...	64		
110	punk	The Stooges	Telluric Chaos, ...	109		
124	punk	AFI	I Heard a Voice:...	123		
157	punk	None More Black	This Is Satire, Fil...	156		
208	punk	Bad Religion	New Maps of H...	207	2007.000	1
247	punk	The Adverts	Live at the Roxy...	246	2003.000	1
259	punk	Ten Foot Pole	Subliminable M...	258	2004.000	1
279	punk	Strike Anywhere	Dead FM, Exit E...	278	2006.000	2
322	punk	Good Riddance	Remain in Me...	321	2008.000	1
328	punk	Penetration	Coming Up for ...	327	2001.000	1
422	punk	Hüsker Dü	The Living End,...	421	2008.000	1
467	punk	Propagandhi	Potemkin City ...	466	2005.000	1
474	punk	The Bouncing S...	Live, Anchors A...	473	2005.000	1
503	punk	Fugazi	The Argument, ...	502	2001.000	1
504	punk	Agent Orange	Chemical React...	503	2007.000	2
525	punk	Down by Law	Windwardtides...	524	2003.000	1
538	punk	Pennywise	Reason to Belie...	537	2008.000	1

Network Explorer

Display | Marking

Info
Nodes: 1239
Selected: 50
Highlighted: 0

Highlight nodes ...
○ None
○ ... whose attributes contain:
○ ... neighbours of selected, ≤ N hops away
Hops: 1
○ ... with at least N connections
○ ... with at most N connections
Connections: 2
○ ... with more connections than any neighbor
○ ... with more connections than average neighbor
○ ... with most connections
Number of nodes: 1
● ... given in the ItemSubset input signal
ID
☑ Output Changes Automatically

Then we created a second link to **Data Table** widget, where we selected all the artists from the punk genre. We sent these data to **Network Explorer** where we set *Highlight nodes* to *...given in the ItemSubset input signal*. Attribute ID was automatically considered for matching nodes. We can see nodes we selected in the subset highlighted in the graph. To mark them as a selected subset, press Enter.

Network File

Read network graph file in Pajek or GML format.

Signals

Inputs:

- (None)

Outputs:

- **Network**

  An instance of Network Graph.

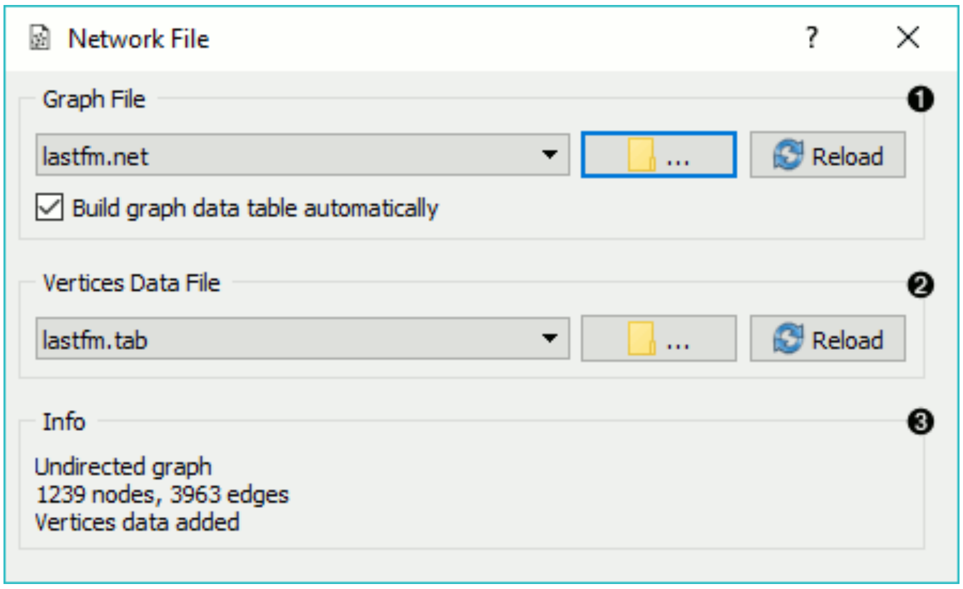
- **Items**

  Properties of a network file.

Description

Network File widget reads network files and sends the input data to its output channel. History of the most recently opened files in maintained in the widget. The widget also includes a directory with sample data sets that come pre-installed with the add-on.

The widget reads data in .net, .gml, .gpickle, .gz, and .edgelist formats. A complimentary .tab or .csv data set can be provided for node information. Orange by default matches a file with the same name as .net file. If (None) is selected, the widget will generate the data from the graph.


Network File

Graph File

lastfm.net

Reload

☑ Build graph data table automatically

Vertices Data File

lastfm.tab

Reload

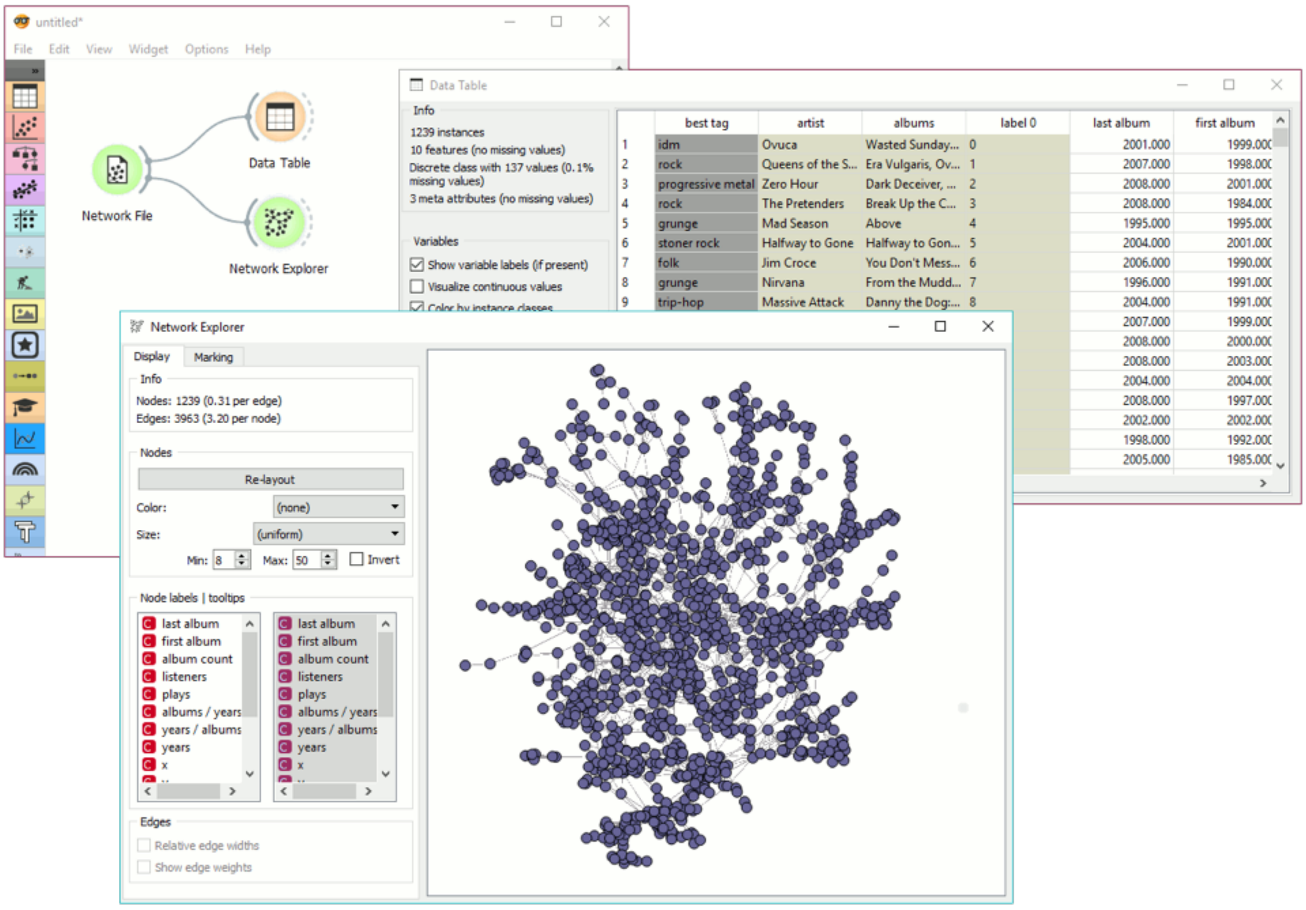
Info

Undirected graph
1239 nodes, 3963 edges
Vertices data added

1. Graph File. Loads network file and (optionally) constructs a data table from the graph. A dropdown menu provides access to documentation data sets with *Browse documentation networks...*. The folder icon provides access to local data files. If *Build graph data table automatically* is checked, the widget will not output an inferred data table (no *Items* output will be available).
2. Vertices Data File. Information on the network nodes. Reads standard Orange data files. he folder icon provides access to local data files.
3. Information on the constructed network. Reports on the type of graph, number of nodes and edges and the provided vertices data file.

Examples

We loaded *lastfm.net* from documentation data set (dropdown → Browse documentation networks) and connected **Data Table** and *Network Explorer* to the widget. **Network File** widget automatically matched the corresponding vertices data file. It outputs *Network* to **Network Explorer** where we can visualize the constructed network and *Items* to **Data Table**, where we can check the attributes of vertices.



Data Table

Info

1239 instances
10 features (no missing values)
Discrete class with 137 values (0.1% missing values)
3 meta attributes (no missing values)

Variables

☑ Show variable labels (if present)
☐ Visualize continuous values
☑ Color by instance classes

	best tag	artist	albums	label 0	last album	first album
1	idm	Ovuca	Wasted Sunday...	0	2001.000	1999.000
2	rock	Queens of the S...	Era Vulgaris, Ov...	1	2007.000	1998.000
3	progressive metal	Zero Hour	Dark Deceiver, ...	2	2008.000	2001.000
4	rock	The Pretenders	Break Up the C...	3	2008.000	1984.000
5	grunge	Mad Season	Above	4	1995.000	1995.000
6	stoner rock	Halfway to Gone	Halfway to Gon...	5	2004.000	2001.000
7	folk	Jim Croce	You Don't Mess...	6	2006.000	1990.000
8	grunge	Nirvana	From the Mudd...	7	1996.000	1991.000
9	trip-hop	Massive Attack	Danny the Dog:...	8	2004.000	1991.000
					2007.000	1999.000
					2008.000	2000.000
					2008.000	2003.000
					2004.000	2004.000
					2008.000	1997.000
					2002.000	2002.000
					1998.000	1992.000
					2005.000	1985.000

Network Explorer

Display | Marking

Info
Nodes: 1239 (0.31 per edge)
Edges: 3963 (3.20 per node)

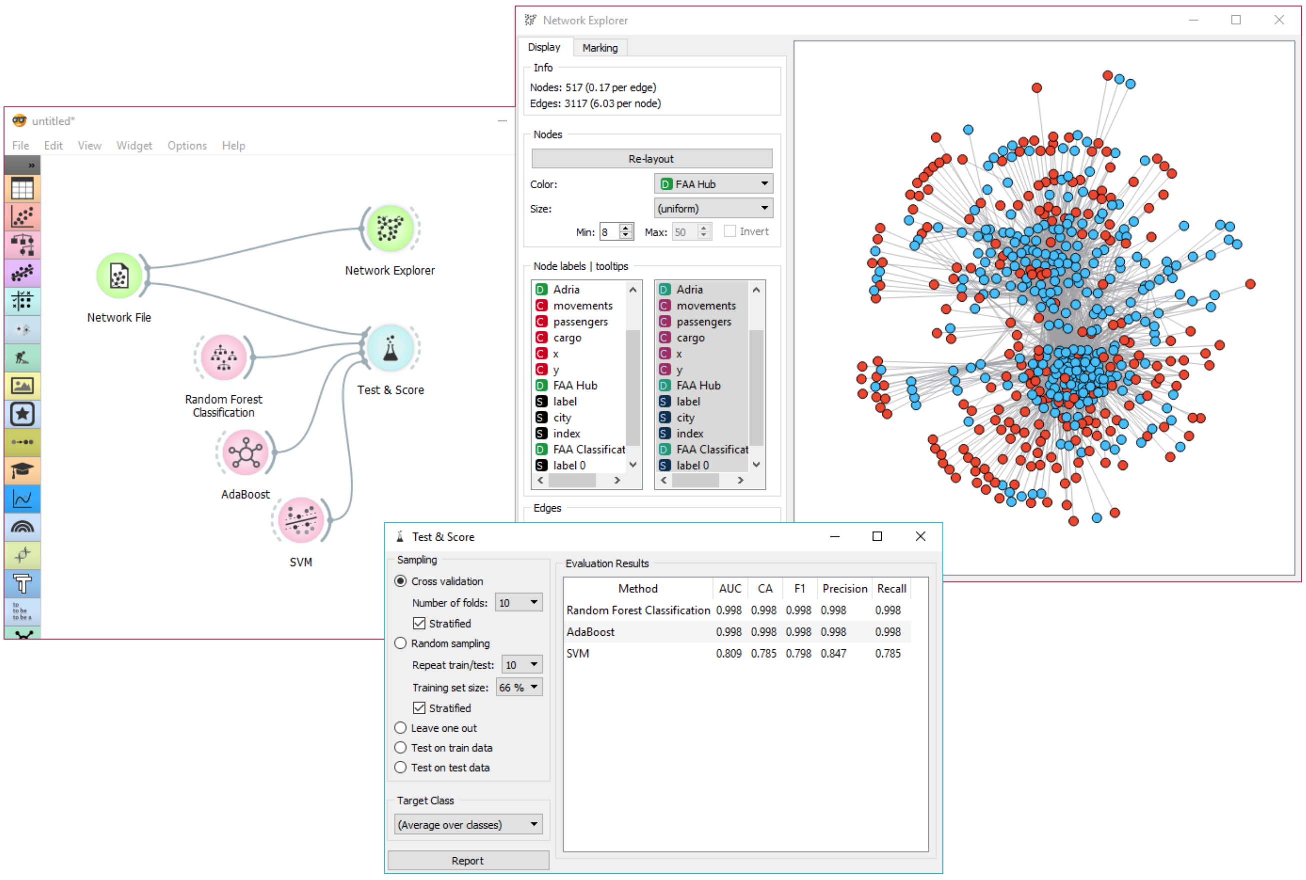
Nodes
Re-layout
Color: (none)
Size: (uniform)
Min: 8 Max: 50 ☐ Invert

Node labels | tooltips

last album	last album
first album	first album
album count	album count
listeners	listeners
plays	plays
albums / years	albums / years
years / albums	years / albums
years	years
x	x

Edges
☐ Relative edge widths
☐ Show edge weights

The second example shows how to use the Network add-on for predictive modelling. We used *airtraffic.net* data and visualized the network in *Network Explorer*. We colored the nodes by FAA Hub attribute (is the airport a hub or not).


untitled*
File Edit View Widget Options Help
Network File
Network Explorer
Random Forest Classification
AdaBoost
SVM
Test & Score

Network Explorer

Display | Marking

Info
Nodes: 517 (0.17 per edge)
Edges: 3117 (6.03 per node)

Nodes
Re-layout
Color: FAA Hub
Size: (uniform)
Min: 8 Max: 50 ☐ Invert

Node labels | tooltips

Adria	Adria
movements	movements
passengers	passengers
cargo	cargo
x	x
y	y
FAA Hub	FAA Hub
label	label
city	city
index	index
FAA Classificat	FAA Classificat
label 0	label 0

Edges

Test & Score

Sampling
(●) Cross validation
Number of folds: 10
☑ Stratified
( ) Random sampling
Repeat train/test: 10
Training set size: 66 %
☑ Stratified
( ) Leave one out
( ) Test on train data
( ) Test on test data

Target Class
(Average over classes)

Report

Evaluation Results

Method	AUC	CA	F1	Precision	Recall
Random Forest Classification	0.998	0.998	0.998	0.998	0.998
AdaBoost	0.998	0.998	0.998	0.998	0.998
SVM	0.809	0.785	0.798	0.847	0.785

Then we tried to predict this value using **Test&Score** and a few classifiers (Random Forest, AdaBoost, SVM) from the core Orange. We can also connect the output of **Test&Score** to **Network Explorer** using the Predictions → Node Data link and then coloring the nodes by predictions in the visualization.

Network From Distances

Constructs a network from distances between instances.

Signals

Inputs:

- **Distances**

  A distance matrix.

Outputs:

- **Network**

  An instance of Network Graph.

- **Data**

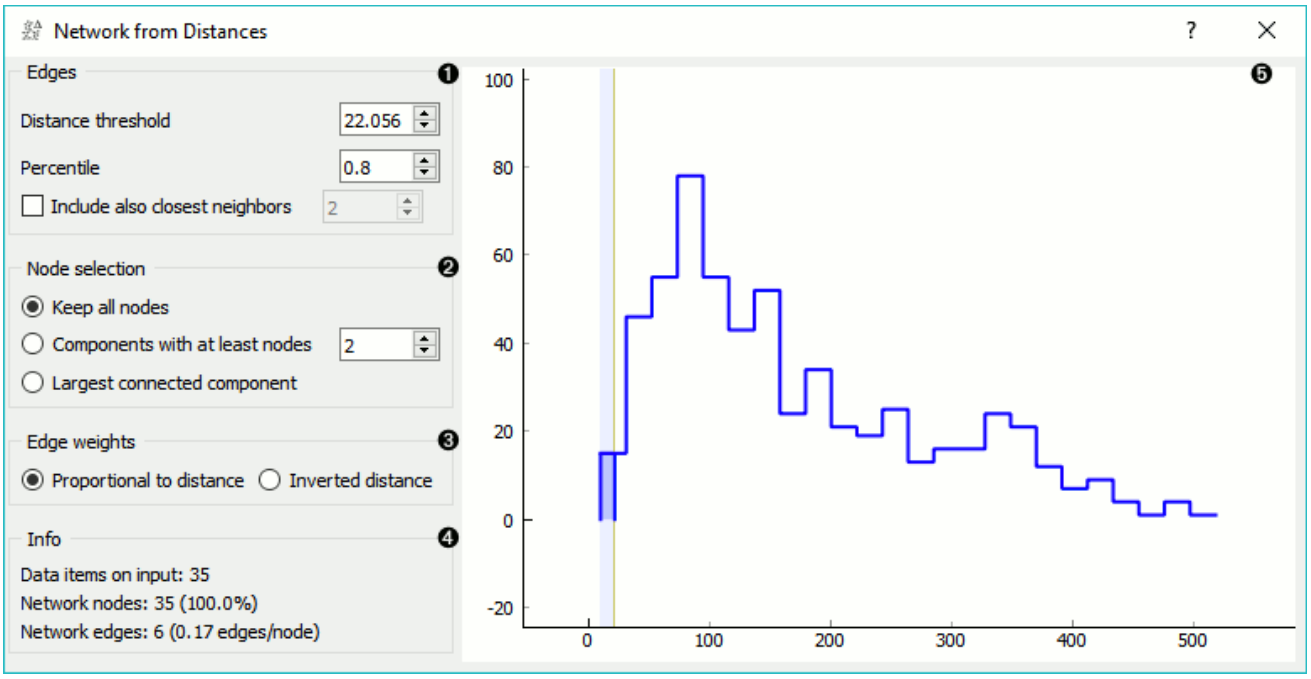
  Attribute-valued data set.

- **Distances**

  A distance matrix.

Description

Network from Distances constructs a network graph from a given distance matrix. Graph is constructed by connecting nodes from data table where the distance between nodes is between the given threshold. In other words, all instances with a distance lower than the selected threshold, will be connected.


Network from Distances

Edges

Distance threshold 22.056

Percentile 0.8

☐ Include also closest neighbors 2

Node selection

● Keep all nodes

○ Components with at least nodes 2

○ Largest connected component

Edge weights

● Proportional to distance ○ Inverted distance

Info

Data items on input: 35

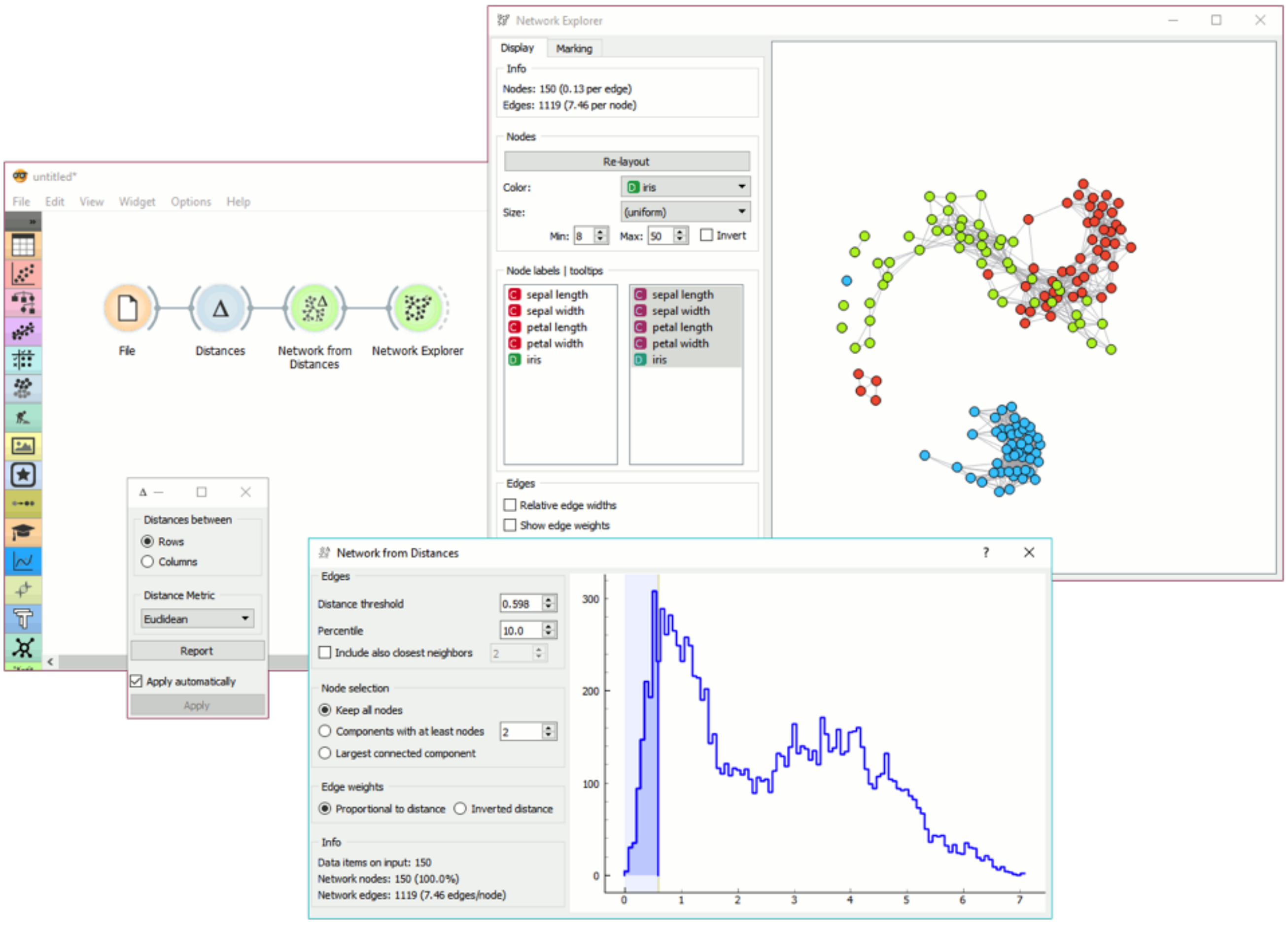
Network nodes: 35 (100.0%)

Network edges: 6 (0.17 edges/node)

1. Edges: - Distance threshold: a closeness threshold for the formation of edges. - Percentile: the percentile of data instances to be connected. - *Include also closest neighbors*: includes a number of closest neighbor to the selected instances.
2. Node selection: - Keep all nodes: entire network is on the ouput. - Components with at least X nodes: filters out nodes with less than the set number of nodes. - Largest connected component: keep only the largest cluster.
3. Edge weights: - Proportional to distance: weights are set to reflect the distance (closeness). - Inverted distance: weights are set to reflect the inverted distance.
4. Information on the constructed network: - Data items on input: number of instances on the input. - Network nodes: number of nodes in the network (and the percentage of the original data). - Network edges: number of constructed edges/connections (and the average number of connections per node).
5. Distance graph. Manually select the distance threshold from the graph by dragging the vertical line left or right.

Example

Network from Distances creates networks from distance matrices. It can transform continuous-valued data sets from a data table via distance matrix into a network graph. This widget is great for visualizing instance similarity as a graph of connected instances.



Info
Nodes: 150 (0.13 per edge)
Edges: 1119 (7.46 per node)

Nodes
Color: iris
Size: (uniform)
Min: 8 Max: 50 ☐ Invert

Node labels | tooltips
sepal length / sepal width / petal length / petal width / iris
sepal length / sepal width / petal length / petal width / iris

Edges
☐ Relative edge widths
☐ Show edge weights

Distances between
[✓ Rows] Columns

Distance Metric
Euclidean

☑ Apply automatically

Network from Distances

Edges
Distance threshold 0.598
Percentile 10.0
☐ Include also closest neighbors 2

Node selection
[✓ Keep all nodes]
Components with at least nodes 2
Largest connected component

Edge weights
[✓ Proportional to distance] Inverted distance

Info
Data items on input: 150
Network nodes: 150 (100.0%)
Network edges: 1119 (7.46 edges/node)

We took *iris.tab* to visualize instance similarity in a graph. We sent the output of **File** widget to **Distances**, where we computed Euclidean distances between rows (instances). Then we sent the output of **Distances** to **Network from Distances**, where we set the distance threshold (how similar the instances have to be to draw an edge between them) to 0.598. We kept all nodes and set edge weights to *proportional to distance*.

Then we observed the constructed network in a *Network Explorer*. We colored the nodes by *iris* attribute.

Network Generator

Signals

Inputs:

- (None)

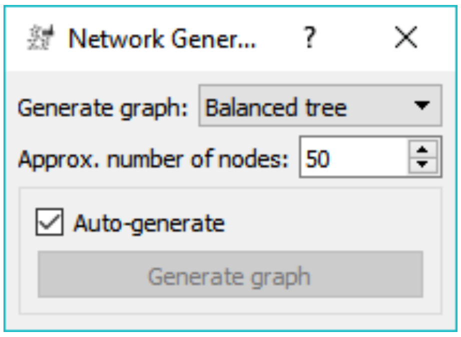
Outputs:

- **Generated Network**

  An instance of Network Graph.

Description

Network Generator constructs exemplary networks. It is mostly intended for teaching/learning about networks.

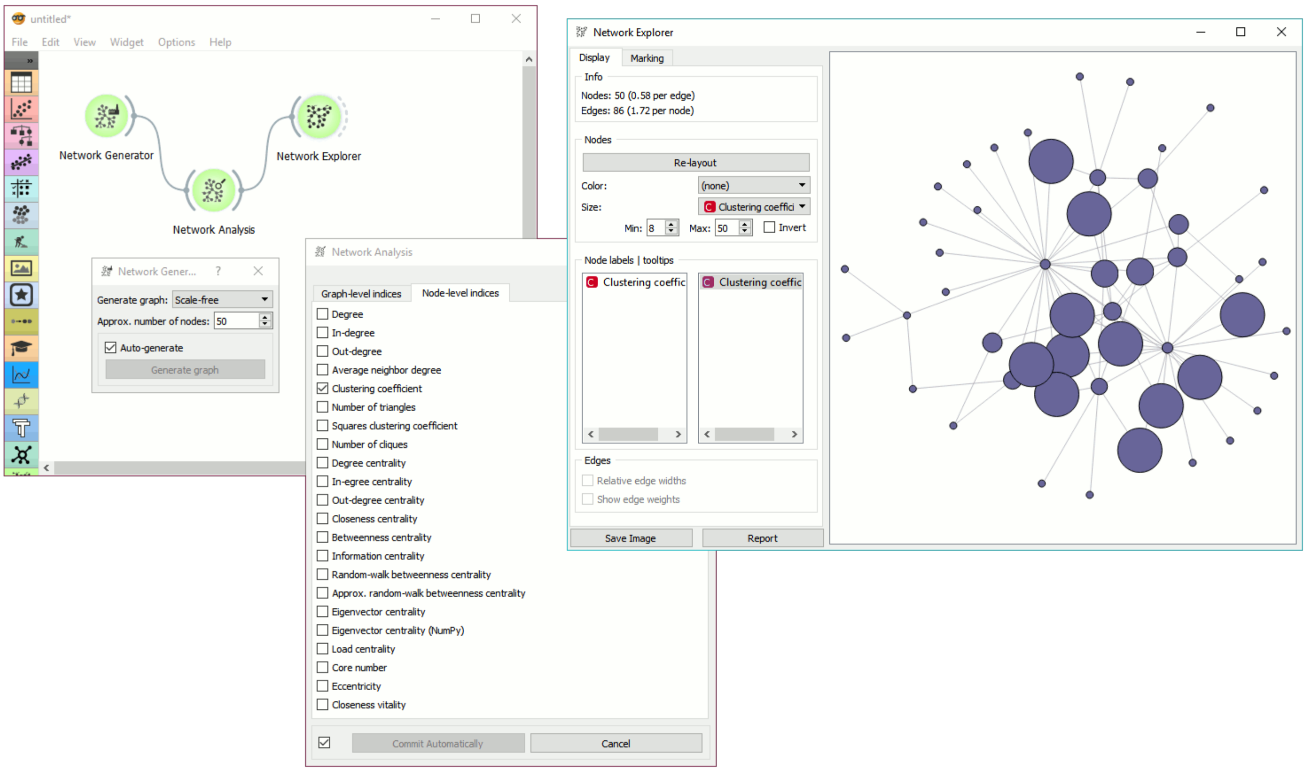


Generate graph: Balanced tree
Approx. number of nodes: 50
☑ Auto-generate

1. Generate graph: - Balanced tree - Barbell - Circular ladder - Complete - Complete bipartite - Cycle - Grid - Hypercube - Ladder - Lobster - Lollipop - Path - Regular - Scale-free - Shell - Star - Waxman - Wheel
2. Approx. number of nodes: nodes that should roughly be in the network (some networks cannot exactly satisfy this condition, hence an approximation).
3. If *Auto-generate* is on, the widget will automatically send the constructed graph to the output. Alternatively, press *Generate graph*.

Example

Network Generator is a nice tool to explore typical graph structures.



Network Gener...

Generate graph: Scale-free

Approx. number of nodes: 50

☑ Auto-generate

Generate graph

Network Analysis

Graph-level indices | Node-level indices

- ☐ Degree
- ☐ In-degree
- ☐ Out-degree
- ☐ Average neighbor degree
- ☑ Clustering coefficient
- ☐ Number of triangles
- ☐ Squares clustering coefficient
- ☐ Number of cliques
- ☐ Degree centrality
- ☐ In-egree centrality
- ☐ Out-degree centrality
- ☐ Closeness centrality
- ☐ Betweenness centrality
- ☐ Information centrality
- ☐ Random-walk betweenness centrality
- ☐ Approx. random-walk betweenness centrality
- ☐ Eigenvector centrality
- ☐ Eigenvector centrality (NumPy)
- ☐ Load centrality
- ☐ Core number
- ☐ Eccentricity
- ☐ Closeness vitality

☑ Commit Automatically | Cancel

Network Explorer

Display | Marking

Info

Nodes: 50 (0.58 per edge)

Edges: 86 (1.72 per node)

Nodes

Re-layout

Color: (none)

Size: Clustering coeffici

Min: 8 Max: 50 ☐ Invert

Node labels | tooltips

Clustering coeffic | Clustering coeffic

Edges

☐ Relative edge widths

☐ Show edge weights

Save Image | Report

Here, we generated a *Scale-free* graph with approximately 50 vertices and sent it to *Network Analysis*. We computed the clustering coefficient and sent the data to *Network Explorer*. Finally, we observed the generated graph in the visualization and set the size of the vertices to *Clustering coefficient*. This is a nice tool to observe and explain the properties of networks.

Bioinformatics

BioMart	Databases Update	Data Profiles	Dicty Express	Differential Expression	Expression Profile Distances	Gene Info
GenExpress	GEO Data Sets	GO Browser	KEGG Pathways	MA Plot	PIPAx	Quality Control
Select Genes	Set Enrichment	Volcano Plot				

BioMart

Gives access to **BioMart** databases.

Signals

Inputs:

- None.

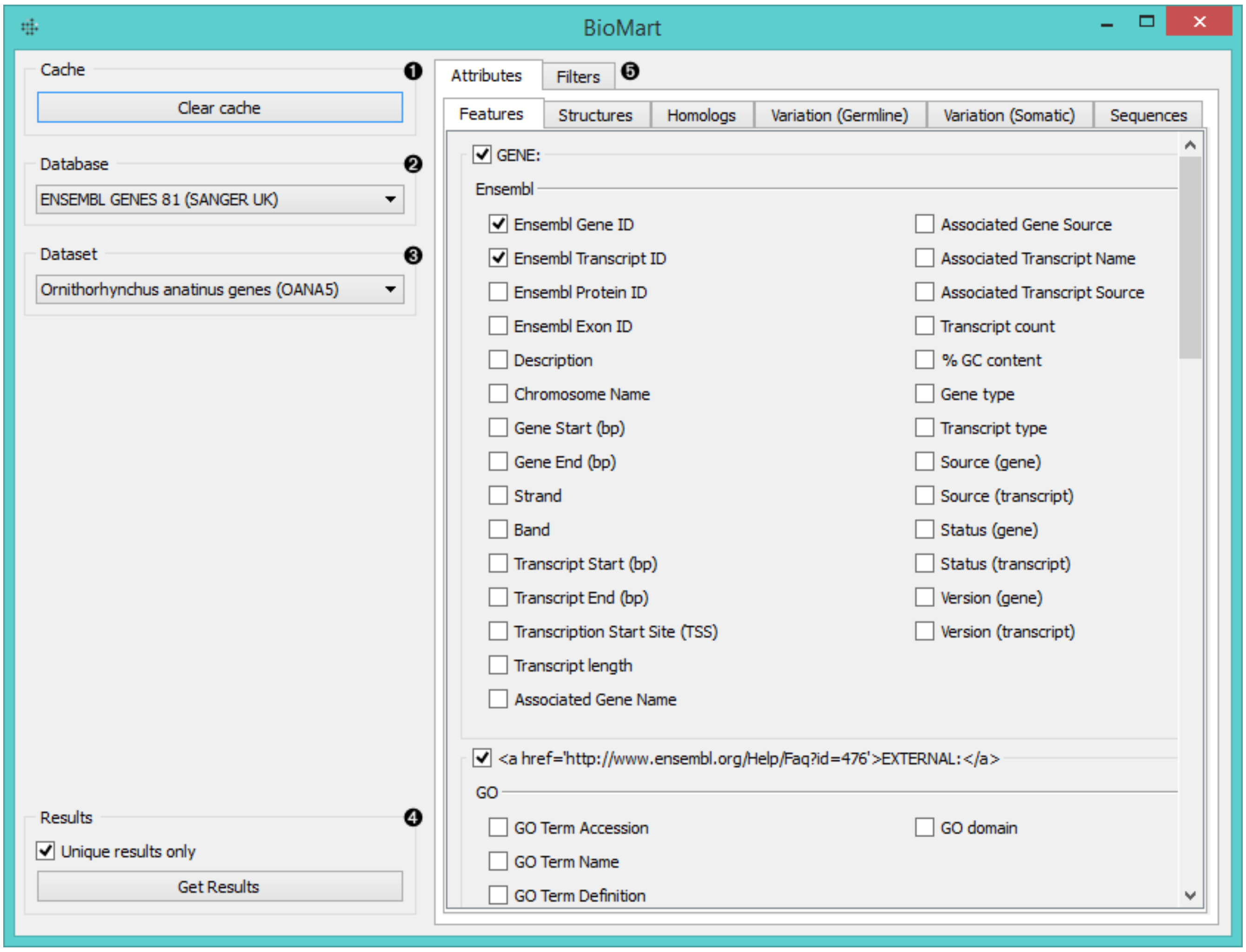
Outputs:

- **Data**

  Data set.

Description

BioMart is a widget for direct access to **BioMart** databases. It sources data from BioMart, filters it by categories (gene, region, phenotype, gene ontology, etc.) and appends selected attributes in the output (IDs, sources, strains, etc.). Read more on the BioMart database library here.



Database: ENSEMBL GENES 81 (SANGER UK)

Dataset: Ornithorhynchus anatinus genes (OANA5)

[✓] Unique results only

[✓] GENE:

[✓] Ensembl Gene ID

[✓] Ensembl Transcript ID

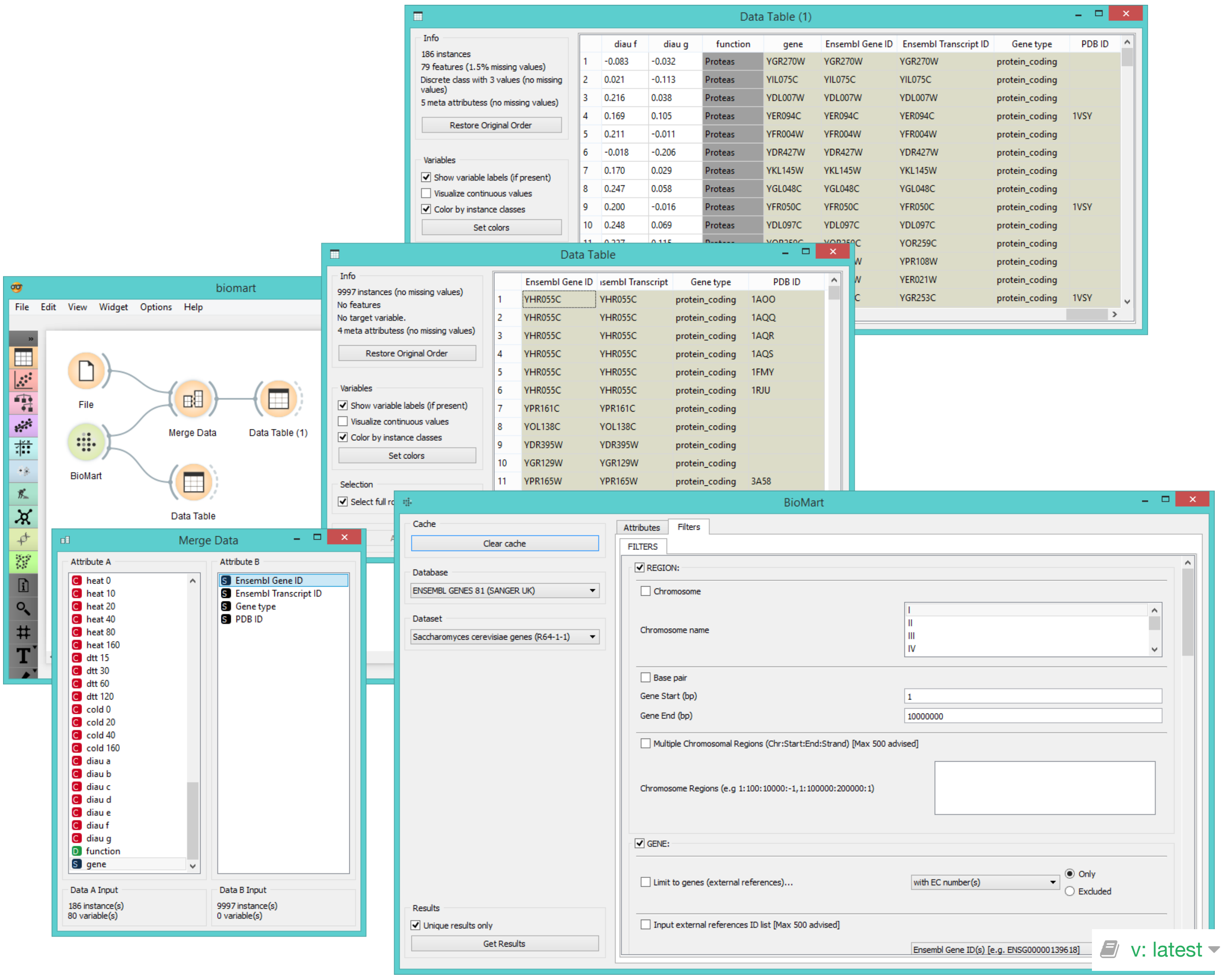
[✓] <a href='http://www.ensembl.org/Help/Faq?id=476'>EXTERNAL:</a>

1. Clear cached data.
2. Select the database to source your data from.
3. Select the dataset (organism) to source your genes from.

4. If *Unique results only* is ticked, the widget will prevent data duplication. Click *Get results* to output the data.
5. Set the output:
   - in **Attributes** you set the meta data you wish to output (e.g. IDs, sources, strains…).
   - in **Filter** you filter the data by gene, phenotype, ontology, protein domains, etc.

Example

BioMart is a great widget for appending additional information to your data. We used *brown-selected* data in the **File** widget. Then we selected *Ensembl genes 81 (Sanger UK)* database to source our additional data from. We decided to append *Ensembl Gene ID*, *Ensembl Transcript ID*, *gene type* and *PDB ID*. We also filtered the data to output only those genes that can be found on chromosome I. We got 9997 instances with 4 meta attributes. Then we used **Merge Data** widget to append these metas to our data. We matched the data by gene/Ensemble gene ID and in the end we got a merged data table with 5 meta attributes.



Data Table (1)

Info
186 instances
79 features (1.5% missing values)
Discrete class with 3 values (no missing values)
5 meta attributess (no missing values)
Restore Original Order

Variables
☑ Show variable labels (if present)
☐ Visualize continuous values
☑ Color by instance classes
Set colors

	diau f	diau g	function	gene	Ensembl Gene ID	Ensembl Transcript ID	Gene type	PDB ID
1	-0.083	-0.032	Proteas	YGR270W	YGR270W	YGR270W	protein_coding	
2	0.021	-0.113	Proteas	YIL075C	YIL075C	YIL075C	protein_coding	
3	0.216	0.038	Proteas	YDL007W	YDL007W	YDL007W	protein_coding	
4	0.169	0.105	Proteas	YER094C	YER094C	YER094C	protein_coding	1VSY
5	0.211	-0.011	Proteas	YFR004W	YFR004W	YFR004W	protein_coding	
6	-0.018	-0.206	Proteas	YDR427W	YDR427W	YDR427W	protein_coding	
7	0.170	0.029	Proteas	YKL145W	YKL145W	YKL145W	protein_coding	
8	0.247	0.058	Proteas	YGL048C	YGL048C	YGL048C	protein_coding	
9	0.200	-0.016	Proteas	YFR050C	YFR050C	YFR050C	protein_coding	1VSY
10	0.248	0.069	Proteas	YDL097C	YDL097C	YDL097C	protein_coding	
11						YOR259C	protein_coding	
						YPR108W	protein_coding	
						YER021W	protein_coding	
						YGR253C	protein_coding	1VSY

Data Table

Info
9997 instances (no missing values)
No features
No target variable.
4 meta attributess (no missing values)
Restore Original Order

Variables
☑ Show variable labels (if present)
☐ Visualize continuous values
☑ Color by instance classes
Set colors

Selection
☑ Select full ro

	Ensembl Gene ID	ısembl Transcript	Gene type	PDB ID
1	YHR055C	YHR055C	protein_coding	1AOO
2	YHR055C	YHR055C	protein_coding	1AQQ
3	YHR055C	YHR055C	protein_coding	1AQR
4	YHR055C	YHR055C	protein_coding	1AQS
5	YHR055C	YHR055C	protein_coding	1FMY
6	YHR055C	YHR055C	protein_coding	1RJU
7	YPR161C	YPR161C	protein_coding	
8	YOL138C	YOL138C	protein_coding	
9	YDR395W	YDR395W	protein_coding	
10	YGR129W	YGR129W	protein_coding	
11	YPR165W	YPR165W	protein_coding	3A58

Merge Data

Attribute A: heat 0, heat 10, heat 20, heat 40, heat 80, heat 160, dtt 15, dtt 30, dtt 60, dtt 120, cold 0, cold 20, cold 40, cold 160, diau a, diau b, diau c, diau d, diau e, diau f, diau g, function, gene

Attribute B: Ensembl Gene ID, Ensembl Transcript ID, Gene type, PDB ID

Data A Input
186 instance(s)
80 variable(s)

Data B Input
9997 instance(s)
0 variable(s)

BioMart

Cache
Clear cache

Database
ENSEMBL GENES 81 (SANGER UK)

Dataset
Saccharomyces cerevisiae genes (R64-1-1)

Results
☑ Unique results only
Get Results

Attributes | Filters

FILTERS

☑ REGION:

☐ Chromosome

Chromosome name: I, II, III, IV

☐ Base pair
Gene Start (bp): 1
Gene End (bp): 10000000

☐ Multiple Chromosomal Regions (Chr:Start:End:Strand) [Max 500 advised]

Chromosome Regions (e.g 1:100:10000:-1,1:100000:200000:1)

☑ GENE:

☐ Limit to genes (external references)… with EC number(s) [● Only] ○ Excluded

☐ Input external references ID list [Max 500 advised]

Ensembl Gene ID(s) [e.g. ENSG00000139618]

v: latest

Data Profiles

Plots gene expression levels by attribute in a graph.

Signals

Inputs:

- **Data**

  Data set.

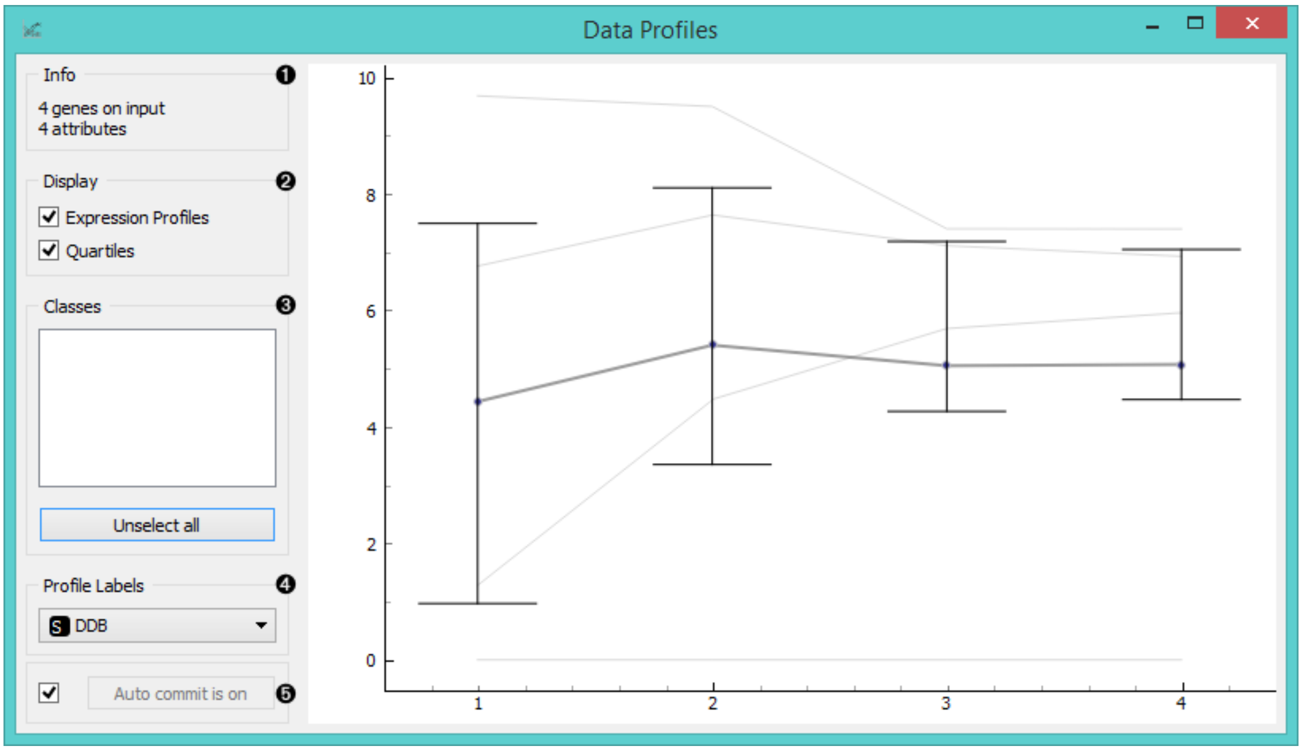
Outputs:

- **Selected Data**

  Instances that the user has manually selected from the plot.

Description

Data Profiles plots gene expression levels for each attribute in a graph. The default graph displays the mean expression level for the input data set. The x-axis represents attributes and the y-axis gene expression values. By hovering over the line you can see which gene it represents and by click on the line you will select the gene and output it.



1. Information on the input data.
2. Select display options:
   - **Expression Profiles** will display expression levels for individual data instances.
   - **Quartiles** will show quartile cut-off points.
3. If the data has classes, you can select which class to display by clicking on it. Such data will also be colored by class. *Unselect All* will show an empty plot, while *Select All* will diplay all data instances by class.
4. Select which attribute you wish to use as a profile label.
5. If *Auto commit is on*, the widget will automatically apply changes to the output. Alternatively click *Commit*.

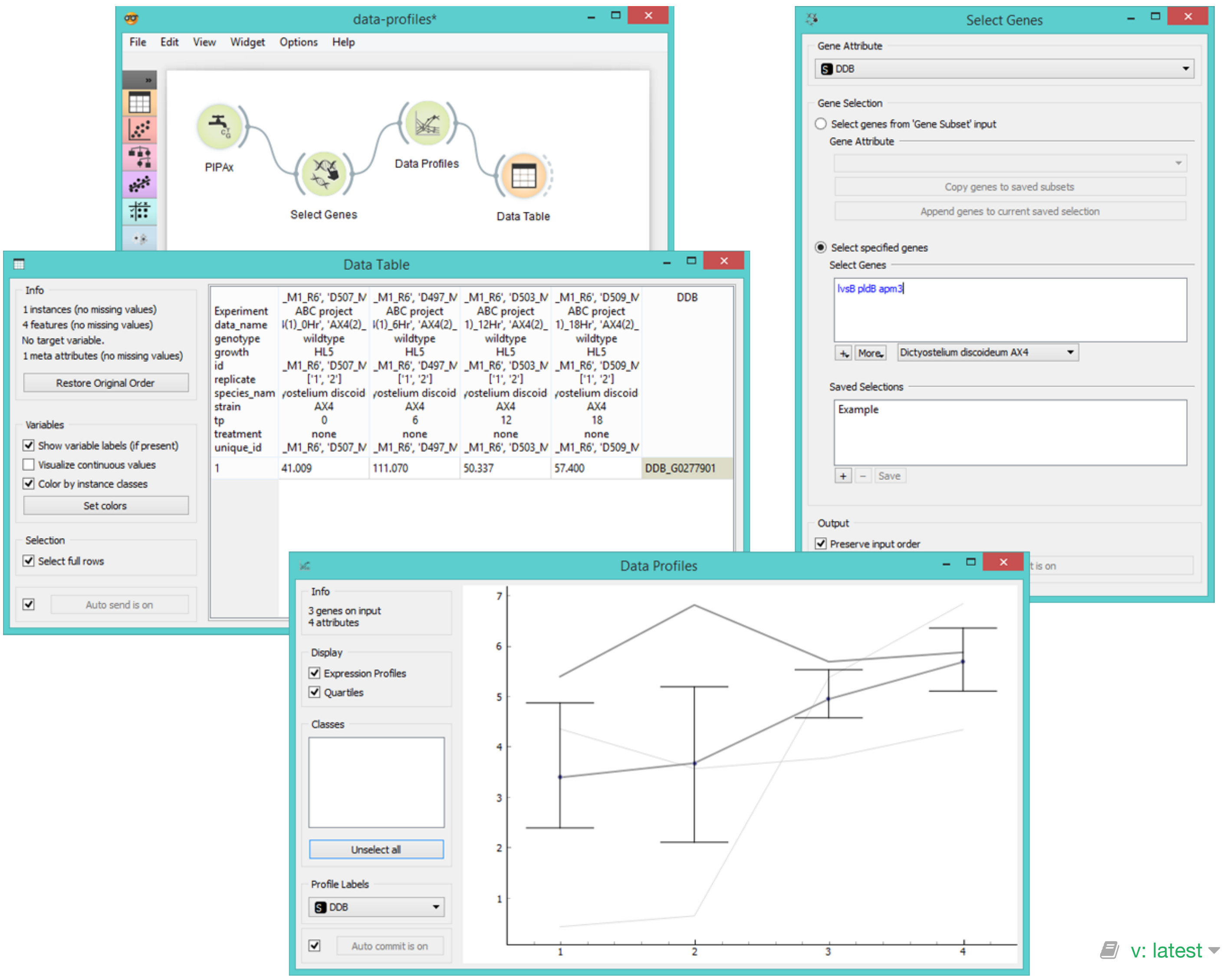
Example

Data Profiles is a great widget for visualizing significant gene expression levels, especially if the data has been sourced at different timepoints. This allows the user to see differences in expression levels in time for each instance in the data set and the overall mean.

Below we used the **PIPAx** widget, where we selected 8 *AX4 Dictyostelium* experiments, all having been sourced at diffferent timepoints and belonging to one of the two replicates. We decided to average replicates (to get one instance for both replicates) and to apply logarithmic transformation to adjust expression levels.

In **Select Genes** we decided to observe only the three genes from the data set that are a part of the *increased exocytosis* process (lsvB, pldB, amp3), which we selected in the *Import gene set names* option. This allows us to specify which biological process we're interested in and to observe only the specified genes.

Then we observe expression levels in **Data Profiles** widget, where we see all three *Expression Profiles* plotted, together with *Quartiles* and mean expression level. Finally, we selected the gene with the highest overall expression level and output it to **Data Table**.



lvsB pldB apm3

1 instances (no missing values)
4 features (no missing values)
No target variable.
1 meta attributes (no missing values)

3 genes on input
4 attributes

	_M1_R6', 'D507_M	_M1_R6', 'D497_M	_M1_R6', 'D503_M	_M1_R6', 'D509_M	DDB
Experiment	ABC project	ABC project	ABC project	ABC project	
data_name	l(1)_0Hr', 'AX4(2)_	l(1)_6Hr', 'AX4(2)_	1)_12Hr', 'AX4(2)_	1)_18Hr', 'AX4(2)_	
genotype	wildtype	wildtype	wildtype	wildtype	
growth	HL5	HL5	HL5	HL5	
id	_M1_R6', 'D507_M	_M1_R6', 'D497_M	_M1_R6', 'D503_M	_M1_R6', 'D509_M	
replicate	['1', '2']	['1', '2']	['1', '2']	['1', '2']	
species_nam	yostelium discoid	yostelium discoid	yostelium discoid	yostelium discoid	
strain	AX4	AX4	AX4	AX4	
tp	0	6	12	18	
treatment	none	none	none	none	
unique_id	_M1_R6', 'D507_M	_M1_R6', 'D497_M	_M1_R6', 'D503_M	_M1_R6', 'D509_M	
1	41.009	111.070	50.337	57.400	DDB_G0277901

v: latest

Databases Update

Updates local systems biology databases, like gene ontologies, annotations, gene names, protein interaction networks, and similar.

Signals

Inputs:

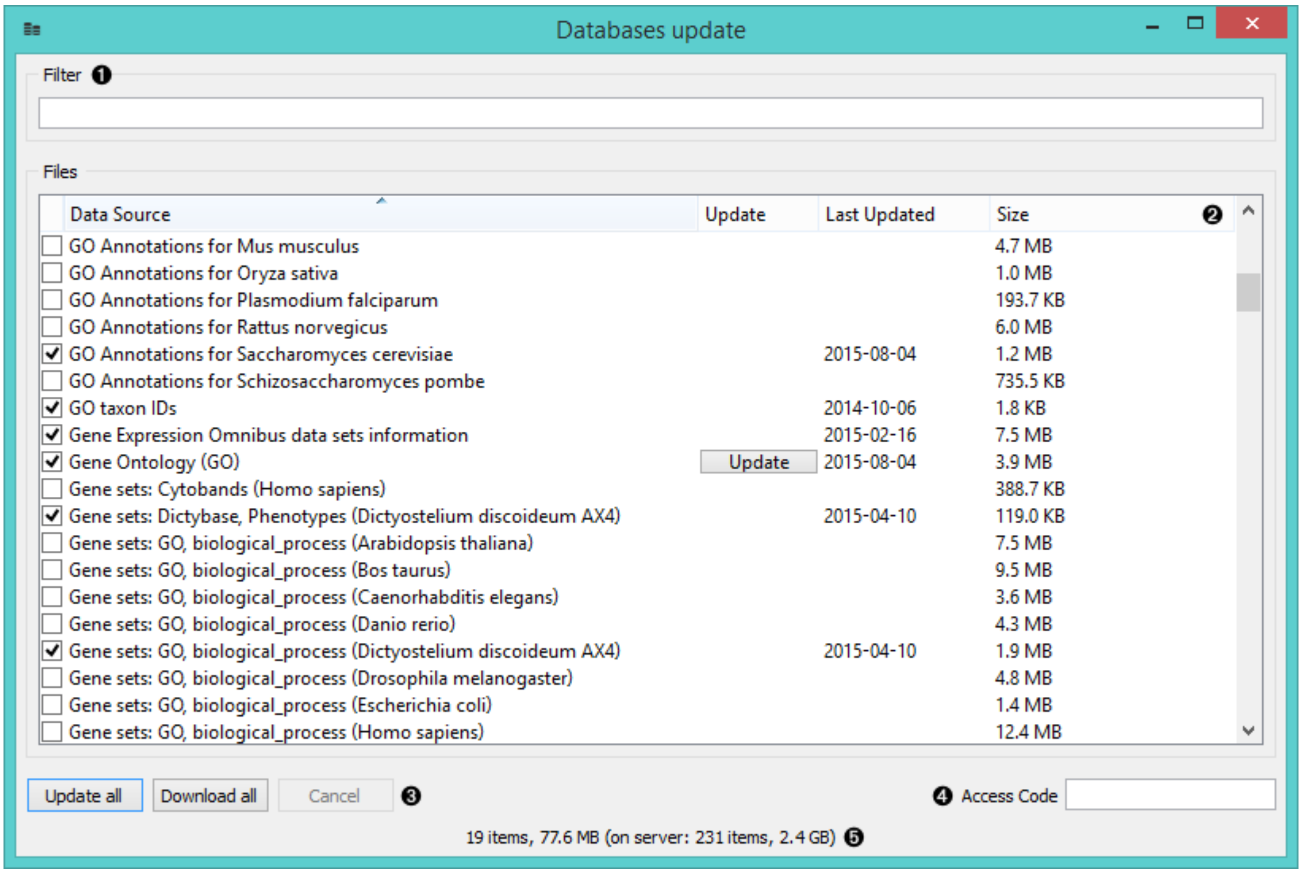
- None

Outputs:

- None

Description

With the bioinformatics add-on you can access several databases directly from Orange. The widget can also be used to update and manage locally stored databases.

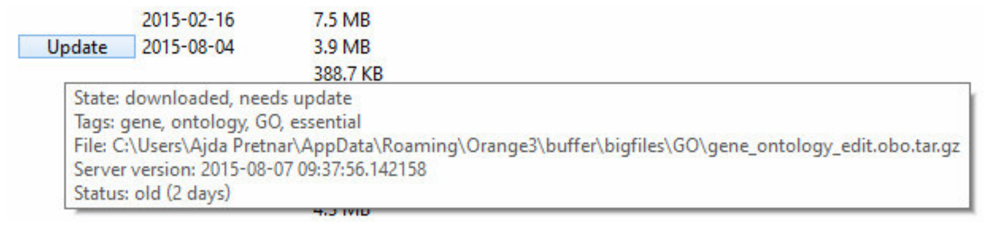


	Data Source	Update	Last Updated	Size
	GO Annotations for Mus musculus			4.7 MB
	GO Annotations for Oryza sativa			1.0 MB
	GO Annotations for Plasmodium falciparum			193.7 KB
	GO Annotations for Rattus norvegicus			6.0 MB
✓	GO Annotations for Saccharomyces cerevisiae		2015-08-04	1.2 MB
	GO Annotations for Schizosaccharomyces pombe			735.5 KB
✓	GO taxon IDs		2014-10-06	1.8 KB
✓	Gene Expression Omnibus data sets information		2015-02-16	7.5 MB
✓	Gene Ontology (GO)	Update	2015-08-04	3.9 MB
	Gene sets: Cytobands (Homo sapiens)			388.7 KB
✓	Gene sets: Dictybase, Phenotypes (Dictyostelium discoideum AX4)		2015-04-10	119.0 KB
	Gene sets: GO, biological_process (Arabidopsis thaliana)			7.5 MB
	Gene sets: GO, biological_process (Bos taurus)			9.5 MB
	Gene sets: GO, biological_process (Caenorhabditis elegans)			3.6 MB
	Gene sets: GO, biological_process (Danio rerio)			4.3 MB
✓	Gene sets: GO, biological_process (Dictyostelium discoideum AX4)		2015-04-10	1.9 MB
	Gene sets: GO, biological_process (Drosophila melanogaster)			4.8 MB
	Gene sets: GO, biological_process (Escherichia coli)			1.4 MB
	Gene sets: GO, biological_process (Homo sapiens)			12.4 MB

19 items, 77.6 MB (on server: 231 items, 2.4 GB) 5

1. Find the desired database.
2. A list of available databases described with data source, update availability, date of your last update and file size. A large **Update** button will be displayed next to the database that needs to be updated.
3. **Update All** will update and **Download All** will download all the selected databases. **Cancel** will abort the action.
4. Some data sets require the *Access code*. Type it in the provided field to access the database.
5. Information on the selected databases.

To get a more detailed information on the particular database hover on its name.


Update 2015-02-16 7.5 MB
2015-08-04 3.9 MB
388.7 KB

State: downloaded, needs update
Tags: gene, ontology, GO, essential
File: C:\Users\Ajda Pretnar\AppData\Roaming\Orange3\buffer\bigfiles\GO\gene_ontology_edit.obo.tar.gz
Server version: 2015-08-07 09:37:56.142158
Status: old (2 days)

v: latest

dictyExpress

Gives access to [dictyExpress](#) databases.

Signals

Inputs:

- (None)

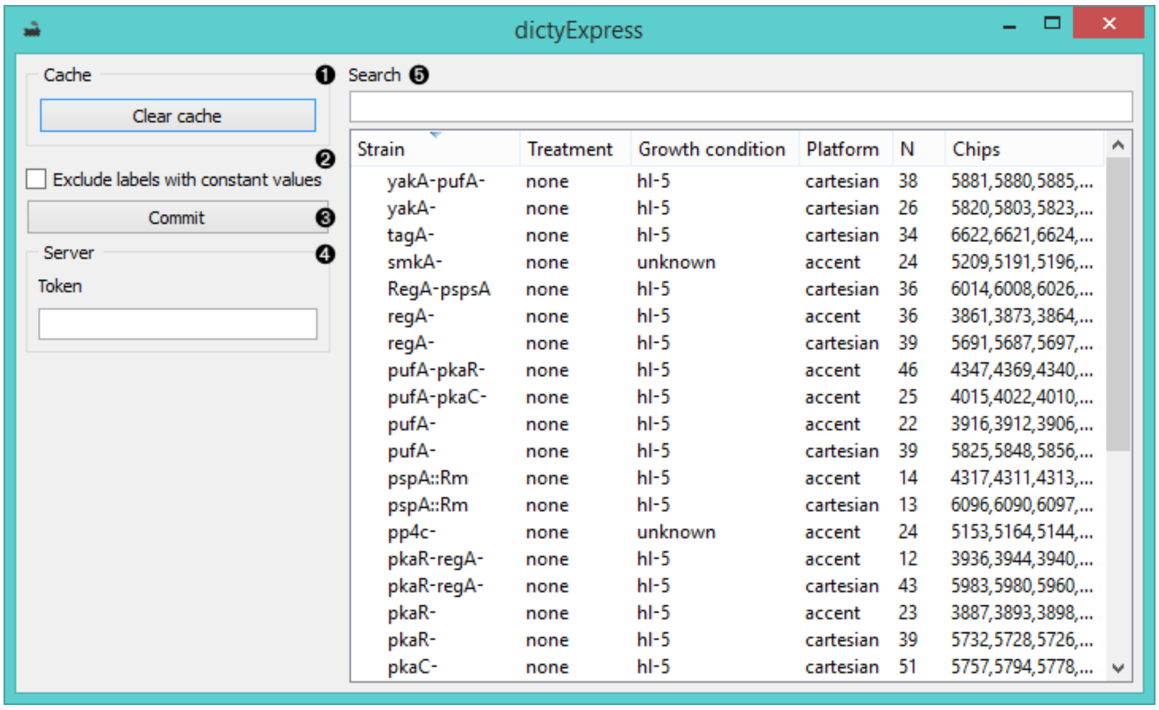
Outputs:

- **Data**

  Selected experiments. Each annotated column contains results of a single experiment or, if the corresponding option is chosen, the average of multiple replicates.

Description

dictyExpress is a widget for a direct access to **[dictyExpress](#)** database and it is very similar to the **GenExpress** and **GEO Data Sets** widgets as it allows you to dowload selected experiments.



dictyExpress

Cache ❶

Clear cache

❷ Exclude labels with constant values

Commit ❸

Server ❹

Token

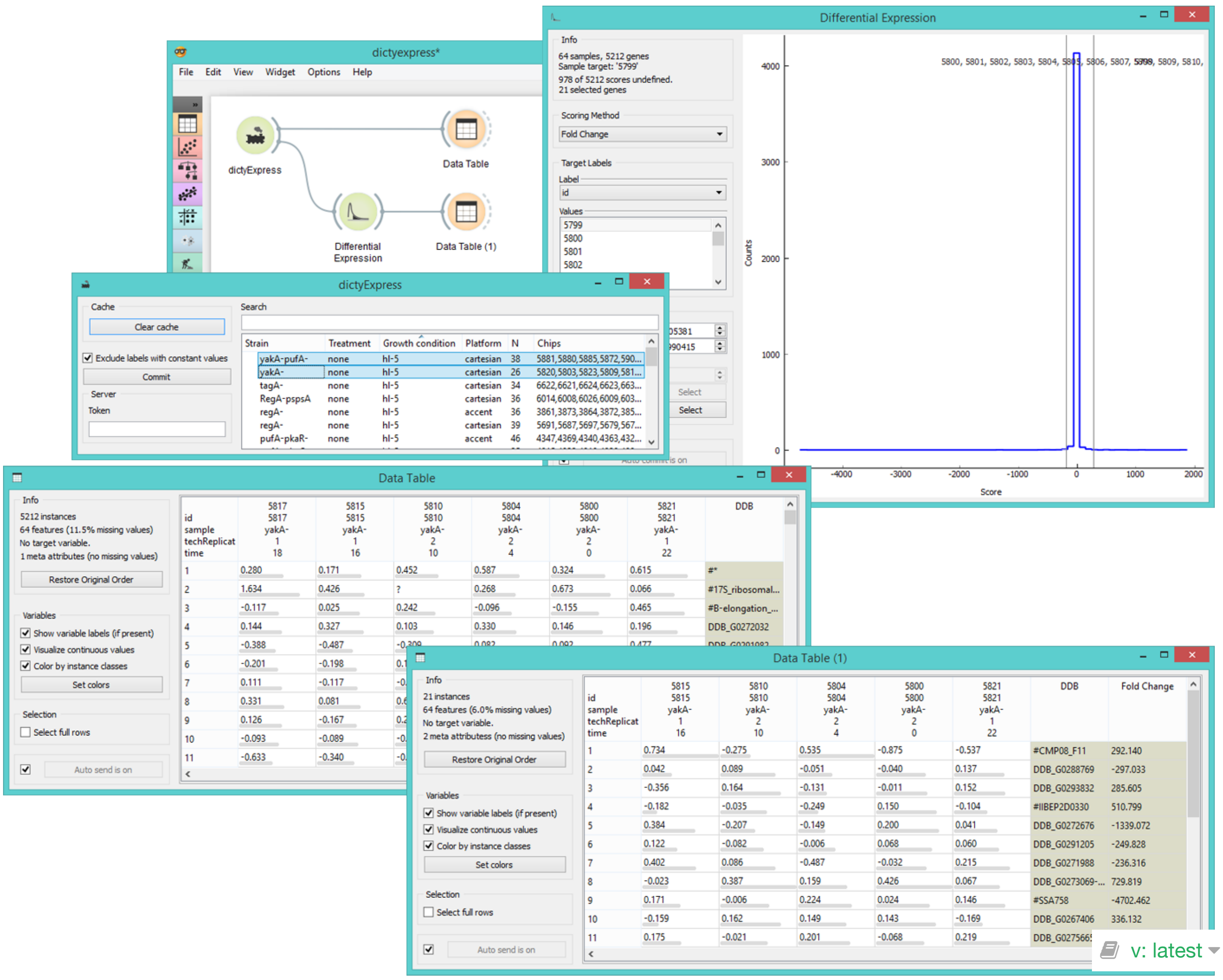
Search ❺

Strain	Treatment	Growth condition	Platform	N	Chips
yakA-pufA-	none	hl-5	cartesian	38	5881,5880,5885,...
yakA-	none	hl-5	cartesian	26	5820,5803,5823,...
tagA-	none	hl-5	cartesian	34	6622,6621,6624,...
smkA-	none	unknown	accent	24	5209,5191,5196,...
RegA-pspsA	none	hl-5	cartesian	36	6014,6008,6026,...
regA-	none	hl-5	accent	36	3861,3873,3864,...
regA-	none	hl-5	cartesian	39	5691,5687,5697,...
pufA-pkaR-	none	hl-5	accent	46	4347,4369,4340,...
pufA-pkaC-	none	hl-5	accent	25	4015,4022,4010,...
pufA-	none	hl-5	accent	22	3916,3912,3906,...
pufA-	none	hl-5	cartesian	39	5825,5848,5856,...
pspA::Rm	none	hl-5	accent	14	4317,4311,4313,...
pspA::Rm	none	hl-5	cartesian	13	6096,6090,6097,...
pp4c-	none	unknown	accent	24	5153,5164,5144,...
pkaR-regA-	none	hl-5	accent	12	3936,3944,3940,...
pkaR-regA-	none	hl-5	cartesian	43	5983,5980,5960,...
pkaR-	none	hl-5	accent	23	3887,3893,3898,...
pkaR-	none	hl-5	cartesian	39	5732,5728,5726,...
pkaC-	none	hl-5	cartesian	51	5757,5794,5778,...

1. The widget will automatically save (cache) downloaded data, which makes them available also in the offline mode. To reset the widget click *Clear cache*.
2. *Exclude labels with constant values* removes labels that are the same for all the selected experiments in the output.
3. Click *Commit* to output the data.
4. Publicly available data are accessible from the outset. Use *Token* to access password protected data.
5. Available experiments can be filtered with the *Search* box at the top.

Example

In the schema below we connected **ditcyExpress** to a **Data Table** to observe all of the selected experiments. Then we used **Differential Expression** widget to select the most relevant genes and output them to another **Data Table**.



Differential Expression

Info
64 samples, 5212 genes
Sample target: '5799'
978 of 5212 scores undefined.
21 selected genes

Scoring Method: Fold Change

Target Labels
Label: id
Values: 5799, 5800, 5801, 5802

dictyExpress

Cache: Clear cache
☑ Exclude labels with constant values
Commit
Server
Token

Search

Strain	Treatment	Growth condition	Platform	N	Chips
yakA-pufA-	none	hl-5	cartesian	38	5881,5880,5885,5872,590...
yakA-	none	hl-5	cartesian	26	5820,5803,5823,5809,581...
tagA-	none	hl-5	cartesian	34	6622,6621,6624,6623,663...
RegA-pspsA	none	hl-5	cartesian	36	6014,6008,6026,6009,603...
regA-	none	hl-5	accent	36	3861,3873,3864,3872,385...
regA-	none	hl-5	cartesian	39	5691,5687,5697,5679,567...
pufA-pkaR-	none	hl-5	accent	46	4347,4369,4340,4363,432...

Data Table

Info
5212 instances
64 features (11.5% missing values)
No target variable.
1 meta attributes (no missing values)
Restore Original Order

Variables
☑ Show variable labels (if present)
☑ Visualize continuous values
☑ Color by instance classes
Set colors

Selection
☐ Select full rows

☑ Auto send is on

id sample techReplicat time	5817 5817 yakA- 1 18	5815 5815 yakA- 1 16	5810 5810 yakA- 2 10	5804 5804 yakA- 2 4	5800 5800 yakA- 2 0	5821 5821 yakA- 1 22	DDB
1	0.280	0.171	0.452	0.587	0.324	0.615	#*
2	1.634	0.426	?	0.268	0.673	0.066	#17S_ribosomal...
3	-0.117	0.025	0.242	-0.096	-0.155	0.465	#B-elongation_...
4	0.144	0.327	0.103	0.330	0.146	0.196	DDB_G0272032
5	-0.388	-0.487	-0.309	0.082	0.092	0.477	DDB_G0291082
6	-0.201	-0.198					
7	0.111	-0.117					
8	0.331	0.081					
9	0.126	-0.167					
10	-0.093	-0.089					
11	-0.633	-0.340					

Data Table (1)

Info
21 instances
64 features (6.0% missing values)
No target variable.
2 meta attributess (no missing values)
Restore Original Order

Variables
☑ Show variable labels (if present)
☑ Visualize continuous values
☑ Color by instance classes
Set colors

Selection
☐ Select full rows

☑ Auto send is on

id sample techReplicat time	5815 5815 yakA- 1 16	5810 5810 yakA- 2 10	5804 5804 yakA- 2 4	5800 5800 yakA- 2 0	5821 5821 yakA- 1 22	DDB	Fold Change
1	0.734	-0.275	0.535	-0.875	-0.537	#CMP08_F11	292.140
2	0.042	0.089	-0.051	-0.040	0.137	DDB_G0288769	-297.033
3	-0.356	0.164	-0.131	-0.011	0.152	DDB_G0293832	285.605
4	-0.182	-0.035	-0.249	0.150	-0.104	#IIBEP2D0330	510.799
5	0.384	-0.207	-0.149	0.200	0.041	DDB_G0272676	-1339.072
6	0.122	-0.082	-0.006	0.068	0.060	DDB_G0291205	-249.828
7	0.402	0.086	-0.487	-0.032	0.215	DDB_G0271988	-236.316
8	-0.023	0.387	0.159	0.426	0.067	DDB_G0273069-...	729.819
9	0.171	-0.006	0.224	0.024	0.146	#SSA758	-4702.462
10	-0.159	0.162	0.149	0.143	-0.169	DDB_G0267406	336.132
11	0.175	-0.021	0.201	-0.068	0.219	DDB_G027566	

Differential Expression

Plots differential gene expression for selected experiments.

Signals

Inputs:

- **Data**

  Data set.

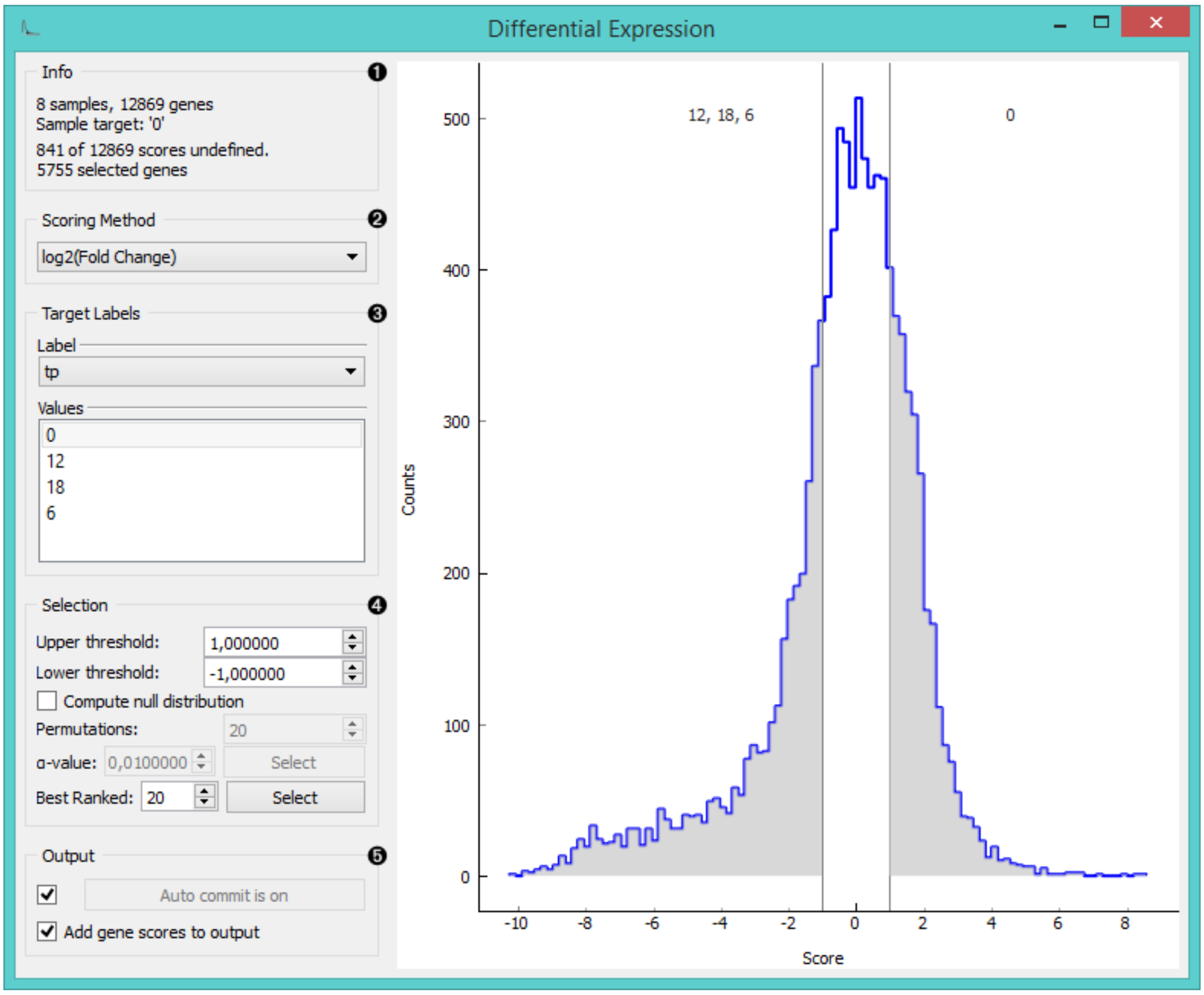
Outputs:

- **Selected data**

  Data subset.

Description

This widget plots a differential gene expression graph for a sample target. It takes gene expression data as an input (from **dictyExpress**, **PIPAx**, etc.) and outputs a selected data subset (normally the most interesting genes).



Info

8 samples, 12869 genes
Sample target: '0'
841 of 12869 scores undefined.
5755 selected genes

Scoring Method: log2(Fold Change)

Target Labels
Label: tp
Values: 0, 12, 18, 6

Selection
Upper threshold: 1,000000
Lower threshold: -1,000000
Compute null distribution: unchecked
Permutations: 20
α-value: 0,0100000
Best Ranked: 20

Output
Auto commit is on: checked
Add gene scores to output: checked

1. Information of the data input and output. The first line shows the number of samples and genes in the data set. The second line displays the selected sample target (read around which the graph is plotted). The third line shows

the number of undefined gene (missing data) and the fourth the number of genes in the output.
2. Select the plotting method in *Scoring method*:
   - **Fold change**: final to initial value ratio
   - **log2 (fold change)**: binary logarithmic transformation of fold change values
   - **T-test**: parametric test of null hypothesis
   - **T-test (P-value)**: parametric test of null hypothesis with P-value as criterium
   - **ANOVA**: variance distribution
   - **ANOVA (P-value)**: variance distribution with P-value as criterium
   - **Signal to Noise Ratio**: biological signal to noise ratio
   - **Mann-Whitney**: non-parametric test of null hypothesis with P-value as criterium
3. Select *Target Labels*. Labels depend on the attributes in the input. In *Values* you can change the sample target (default value is the first value on the list, alphabetically or numerically).
4. *Selection* box controls the output data.
   - By setting the *Lower threshold* and *Upper threshold* values you are outputting the data outside this interval (the most interesting expression levels). You can also manually place the threshold lines by dragging left or right in the plot.
   - If you click *Compute null distribution* box, the widget will calculate null distribution and display it in the plot. *Permutations* field allows you to set the precision of null distribution (the more permutations the more precise the distribution), while

     α

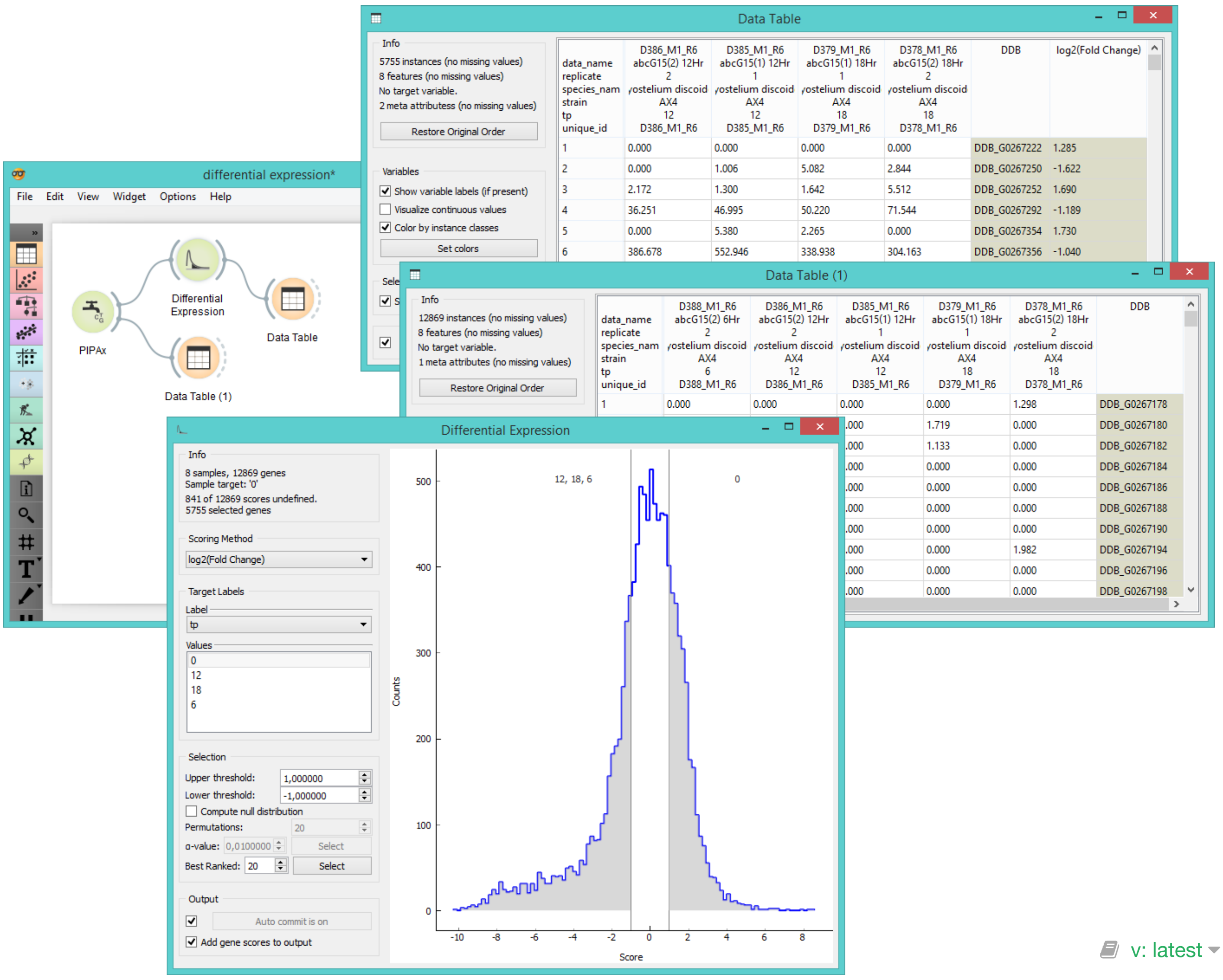
     -value will be the allowed probability of false positives. Press *Select* to output this data.

- The final option is to set the number of best ranked genes and output them with *Select*.

1. When *Auto commit is on* is ticked, the widget will automatically apply the changes. Alternatively press *Commit*. If the *Add gene scores to output* is ticked, the widget will append an additional column with gene scores to the data.

Example

In the example below we chose two experiments from the **PIPAx** widget (8 experiments measuring gene expression levels on *Dictyostelium discoideum* at different timepoints) and observed them in the **Data Table**. Then we used the **Differential Expression** widget to select the most interesting genes. We left upper and lower threshold at default (1 and -1) and output the data. Then we observed the selected data in another **Data Table**. As we have ticked the *Add gene scores to output*, the table shows an additional column with gene scores as instances.



Data Table

Info
5755 instances (no missing values)
8 features (no missing values)
No target variable.
2 meta attributes (no missing values)

Restore Original Order

Variables
☑ Show variable labels (if present)
☐ Visualize continuous values
☑ Color by instance classes

Set colors

data_name replicate species_nam strain tp unique_id	D386_M1_R6 abcG15(2) 12Hr 2 yostelium discoid AX4 12 D386_M1_R6	D385_M1_R6 abcG15(1) 12Hr 1 yostelium discoid AX4 12 D385_M1_R6	D379_M1_R6 abcG15(1) 18Hr 1 yostelium discoid AX4 18 D379_M1_R6	D378_M1_R6 abcG15(2) 18Hr 2 yostelium discoid AX4 18 D378_M1_R6	DDB	log2(Fold Change)
1	0.000	0.000	0.000	0.000	DDB_G0267222	1.285
2	0.000	1.006	5.082	2.844	DDB_G0267250	-1.622
3	2.172	1.300	1.642	5.512	DDB_G0267252	1.690
4	36.251	46.995	50.220	71.544	DDB_G0267292	-1.189
5	0.000	5.380	2.265	0.000	DDB_G0267354	1.730
6	386.678	552.946	338.938	304.163	DDB_G0267356	-1.040

Data Table (1)

Info
12869 instances (no missing values)
8 features (no missing values)
No target variable.
1 meta attributes (no missing values)

Restore Original Order

data_name replicate species_nam strain tp unique_id	D388_M1_R6 abcG15(2) 6Hr 2 yostelium discoid AX4 6 D388_M1_R6	D386_M1_R6 abcG15(2) 12Hr 2 yostelium discoid AX4 12 D386_M1_R6	D385_M1_R6 abcG15(1) 12Hr 1 yostelium discoid AX4 12 D385_M1_R6	D379_M1_R6 abcG15(1) 18Hr 1 yostelium discoid AX4 18 D379_M1_R6	D378_M1_R6 abcG15(2) 18Hr 2 yostelium discoid AX4 18 D378_M1_R6	DDB
1	0.000	0.000	0.000	0.000	1.298	DDB_G0267178
			.000	1.719	0.000	DDB_G0267180
			.000	1.133	0.000	DDB_G0267182
			.000	0.000	0.000	DDB_G0267184
			.000	0.000	0.000	DDB_G0267186
			.000	0.000	0.000	DDB_G0267188
			.000	0.000	0.000	DDB_G0267190
			.000	0.000	1.982	DDB_G0267194
			.000	0.000	0.000	DDB_G0267196
			.000	0.000	0.000	DDB_G0267198

Differential Expression

Info
8 samples, 12869 genes
Sample target: '0'
841 of 12869 scores undefined.
5755 selected genes

Scoring Method
log2(Fold Change)

Target Labels
Label
tp
Values
0
12
18
6

Selection
Upper threshold: 1,000000
Lower threshold: -1,000000
☐ Compute null distribution
Permutations: 20
α-value: 0,0100000 Select
Best Ranked: 20 Select

Output
☑ Auto commit is on
☑ Add gene scores to output

v: latest

Expression Profile Distances

Computes distances between gene expression levels.

Signals

Inputs:

- **Data**

  Data set.

Outputs:

- **Distances**

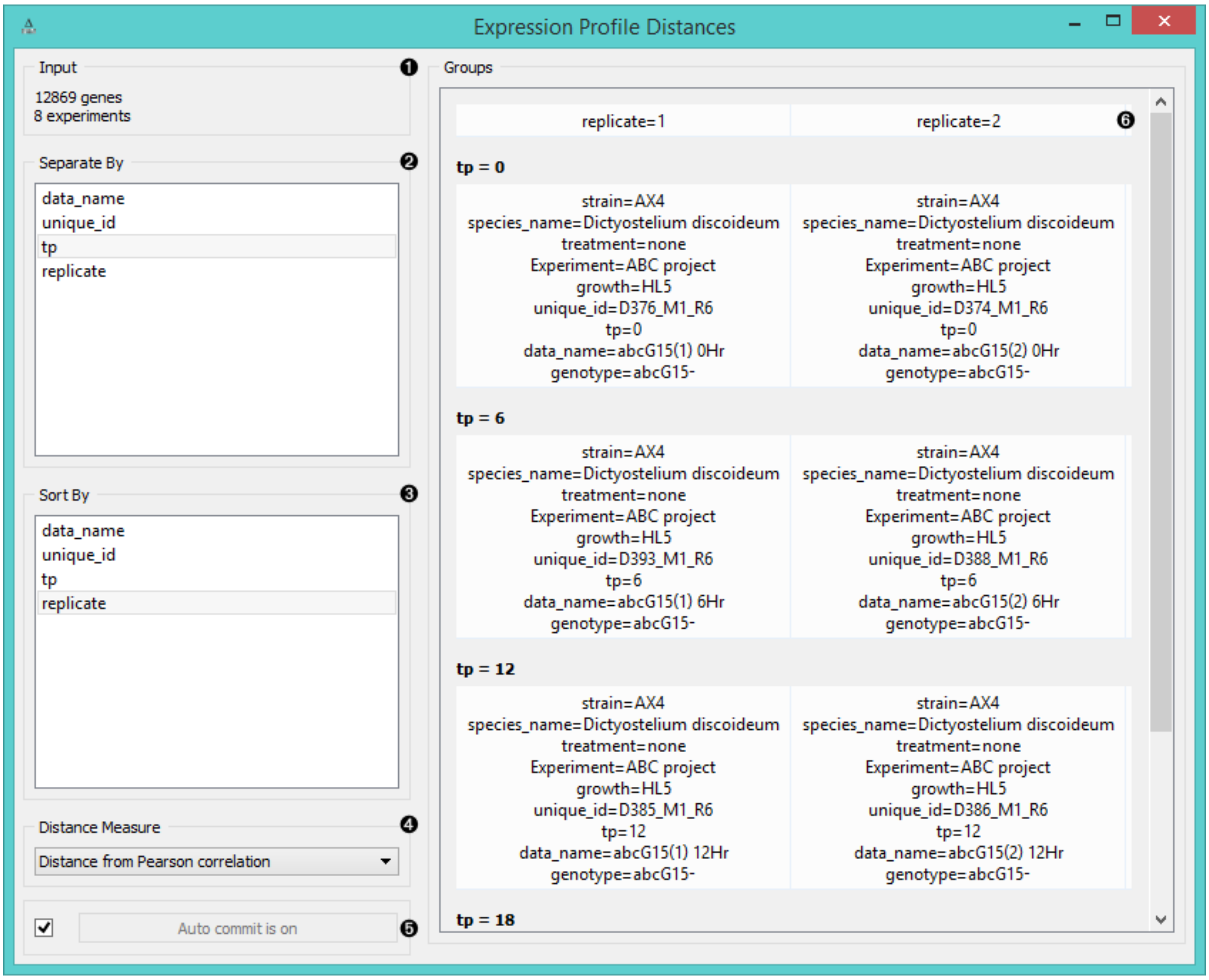
  Distance matrix.

- **Sorted Data**

  Data with groups as attributes.

Description

Widget **Expression Profile Distances** computes distances between expression levels among groups of data. Groups are data clusters set by the user through *separate by* function in the widget. Data can be separated by one or more variable labels (usually timepoint, replicates, IDs, etc.). Widget outputs distance matrix that can be fed into **Distance Map** and **Hierarchical Clustering** widgets.



Expression Profile Distances

Input
12869 genes
8 experiments

Separate By
data_name
unique_id
tp
replicate

Sort By
data_name
unique_id
tp
replicate

Distance Measure
Distance from Pearson correlation

Auto commit is on

Groups

	replicate=1	replicate=2

tp = 0	strain=AX4
	species_name=Dictyostelium discoideum
	treatment=none
	Experiment=ABC project
	growth=HL5
	unique_id=D376_M1_R6
	tp=0
	data_name=abcG15(1) 0Hr
	genotype=abcG15-
		strain=AX4
		species_name=Dictyostelium discoideum
		treatment=none
		Experiment=ABC project
		growth=HL5
		unique_id=D374_M1_R6
		tp=0
		data_name=abcG15(2) 0Hr
		genotype=abcG15-

tp = 6	strain=AX4
	species_name=Dictyostelium discoideum
	treatment=none
	Experiment=ABC project
	growth=HL5
	unique_id=D393_M1_R6
	tp=6
	data_name=abcG15(1) 6Hr
	genotype=abcG15-
		strain=AX4
		species_name=Dictyostelium discoideum
		treatment=none
		Experiment=ABC project
		growth=HL5
		unique_id=D388_M1_R6
		tp=6
		data_name=abcG15(2) 6Hr
		genotype=abcG15-

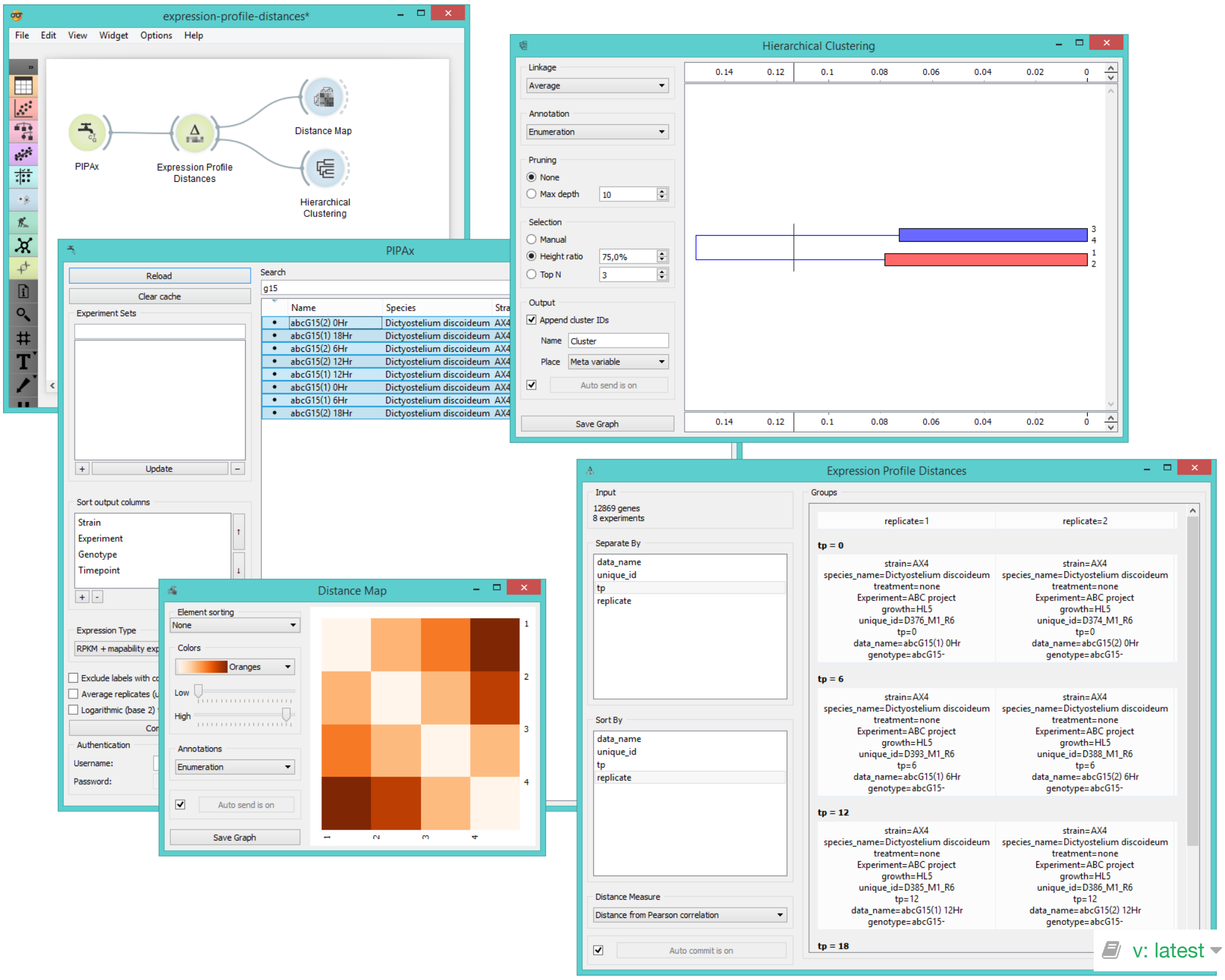
tp = 12	strain=AX4
	species_name=Dictyostelium discoideum
	treatment=none
	Experiment=ABC project
	growth=HL5
	unique_id=D385_M1_R6
	tp=12
	data_name=abcG15(1) 12Hr
	genotype=abcG15-
		strain=AX4
		species_name=Dictyostelium discoideum
		treatment=none
		Experiment=ABC project
		growth=HL5
		unique_id=D386_M1_R6
		tp=12
		data_name=abcG15(2) 12Hr
		genotype=abcG15-

tp = 18		

1. Information on the input data.
2. Separate the experiments into groups by labels (normally timepoint, replicates, data name, etc.).
3. Sort the experiments inside the group by labels.
4. Choose the *Distance Measure*:
   - **Pearson** (linear correlation between the values, remapped as a distance in a [0, 1] interval)
   - **Euclidean** ("straight line", distance between two points)
   - **Spearman** (linear correlation between the rank of the values, remapped as a distance in a [0, 1] interval)
5. If *Auto commit is on*, the widget will automatically compute the distances and output them. Alternatively click *Commit*.
6. This snapshot shows 4 groups of experiments (tp=0, tp=6, tp=12, tp=18) with 2 experiments (replicates) in each group.

Example

Expression Profile Distances widget is used to calculate distances between gene expression values sorted by labels. We chose 8 experiments measuring gene expression levels on *Dictyostelium discoideum* at different timepoints. In the **Expression Profile Distances** widget we separated the data by timepoint and sorted them by replicates. We could see the grouping immediately in the *Groups* box on the right. Then we fed the results to **Distance Map** and **Hierarchical Clustering** to visualize the distances and cluster the attributes.


expression-profile-distances*

File Edit View Widget Options Help

PIPAx

Reload

Clear cache

Experiment Sets

Update

Search: g15

Name	Species	Stra
abcG15(2) 0Hr	Dictyostelium discoideum	AX4
abcG15(1) 18Hr	Dictyostelium discoideum	AX4
abcG15(2) 6Hr	Dictyostelium discoideum	AX4
abcG15(2) 12Hr	Dictyostelium discoideum	AX4
abcG15(1) 12Hr	Dictyostelium discoideum	AX4
abcG15(1) 0Hr	Dictyostelium discoideum	AX4
abcG15(1) 6Hr	Dictyostelium discoideum	AX4
abcG15(2) 18Hr	Dictyostelium discoideum	AX4

Sort output columns: Strain, Experiment, Genotype, Timepoint

Expression Type: RPKM + mapability exp

☐ Exclude labels with c
☐ Average replicates (u
☐ Logarithmic (base 2)

Authentication

Username:

Password:

Hierarchical Clustering

Linkage: Average

Annotation: Enumeration

Pruning: ● None ○ Max depth 10

Selection: ○ Manual ● Height ratio 75,0% ○ Top N 3

Output: ☑ Append cluster IDs

Name: Cluster

Place: Meta variable

☑ Auto send is on

Save Graph

Distance Map

Element sorting: None

Colors: Oranges

Low

High

Annotations: Enumeration

☑ Auto send is on

Save Graph

Expression Profile Distances

Input

12869 genes
8 experiments

Separate By: data_name, unique_id, tp, replicate

Sort By: data_name, unique_id, tp, replicate

Distance Measure: Distance from Pearson correlation

☑ Auto commit is on

Groups

	replicate=1	replicate=2

tp = 0	strain=AX4
	species_name=Dictyostelium discoideum
	treatment=none
	Experiment=ABC project
	growth=HL5
	unique_id=D376_M1_R6
	tp=0
	data_name=abcG15(1) 0Hr
	genotype=abcG15-
		strain=AX4
		species_name=Dictyostelium discoideum
		treatment=none
		Experiment=ABC project
		growth=HL5
		unique_id=D374_M1_R6
		tp=0
		data_name=abcG15(2) 0Hr
		genotype=abcG15-

tp = 6	strain=AX4
	species_name=Dictyostelium discoideum
	treatment=none
	Experiment=ABC project
	growth=HL5
	unique_id=D393_M1_R6
	tp=6
	data_name=abcG15(1) 6Hr
	genotype=abcG15-
		strain=AX4
		species_name=Dictyostelium discoideum
		treatment=none
		Experiment=ABC project
		growth=HL5
		unique_id=D388_M1_R6
		tp=6
		data_name=abcG15(2) 6Hr
		genotype=abcG15-

tp = 12	strain=AX4
	species_name=Dictyostelium discoideum
	treatment=none
	Experiment=ABC project
	growth=HL5
	unique_id=D385_M1_R6
	tp=12
	data_name=abcG15(1) 12Hr
	genotype=abcG15-
		strain=AX4
		species_name=Dictyostelium discoideum
		treatment=none
		Experiment=ABC project
		growth=HL5
		unique_id=D386_M1_R6
		tp=12
		data_name=abcG15(2) 12Hr
		genotype=abcG15-

tp = 18		

v: latest

Gene Info

Displays information on the genes in the input.

Signals

Inputs:

- **Data**

  Data set.

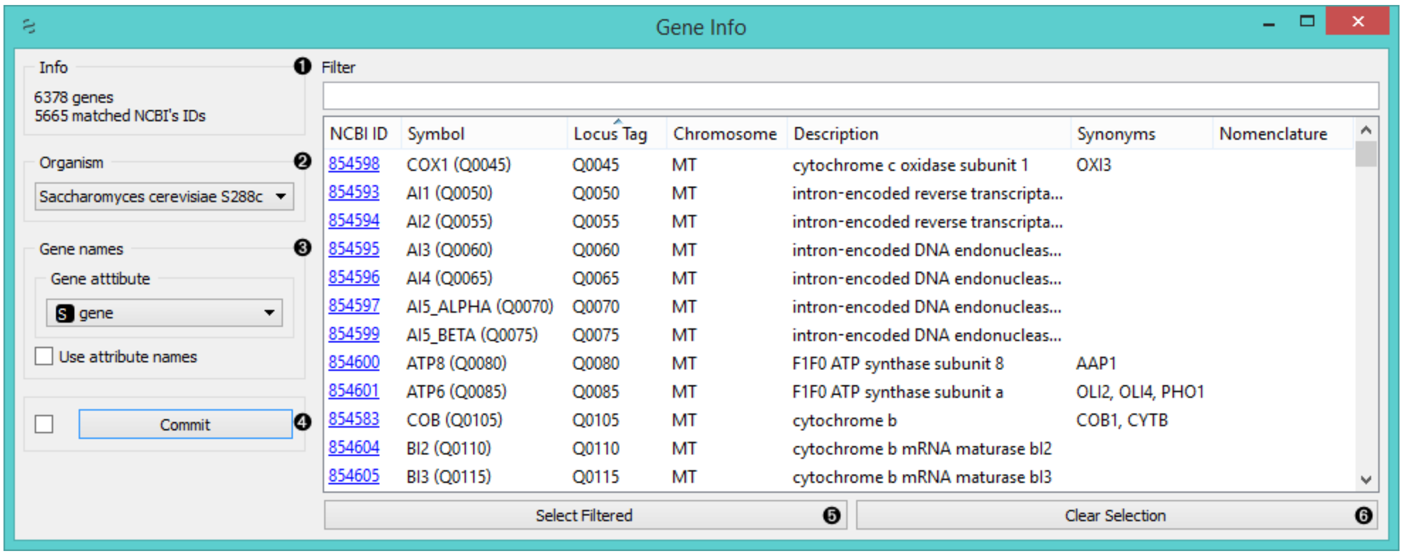
Outputs:

- **Selected Data**

  Instances with meta data that the user has manually selected in the widget.

Description

A useful widget that presents information on the genes from the NCBI database. You can also select a subset and feed it to other widgets. By clicking on the gene NCBI ID in the list, you will be taken to the NCBI site with the information on the gene.



6378 genes
5665 matched NCBI's IDs

Organism: Saccharomyces cerevisiae S288c

Gene attibute: gene

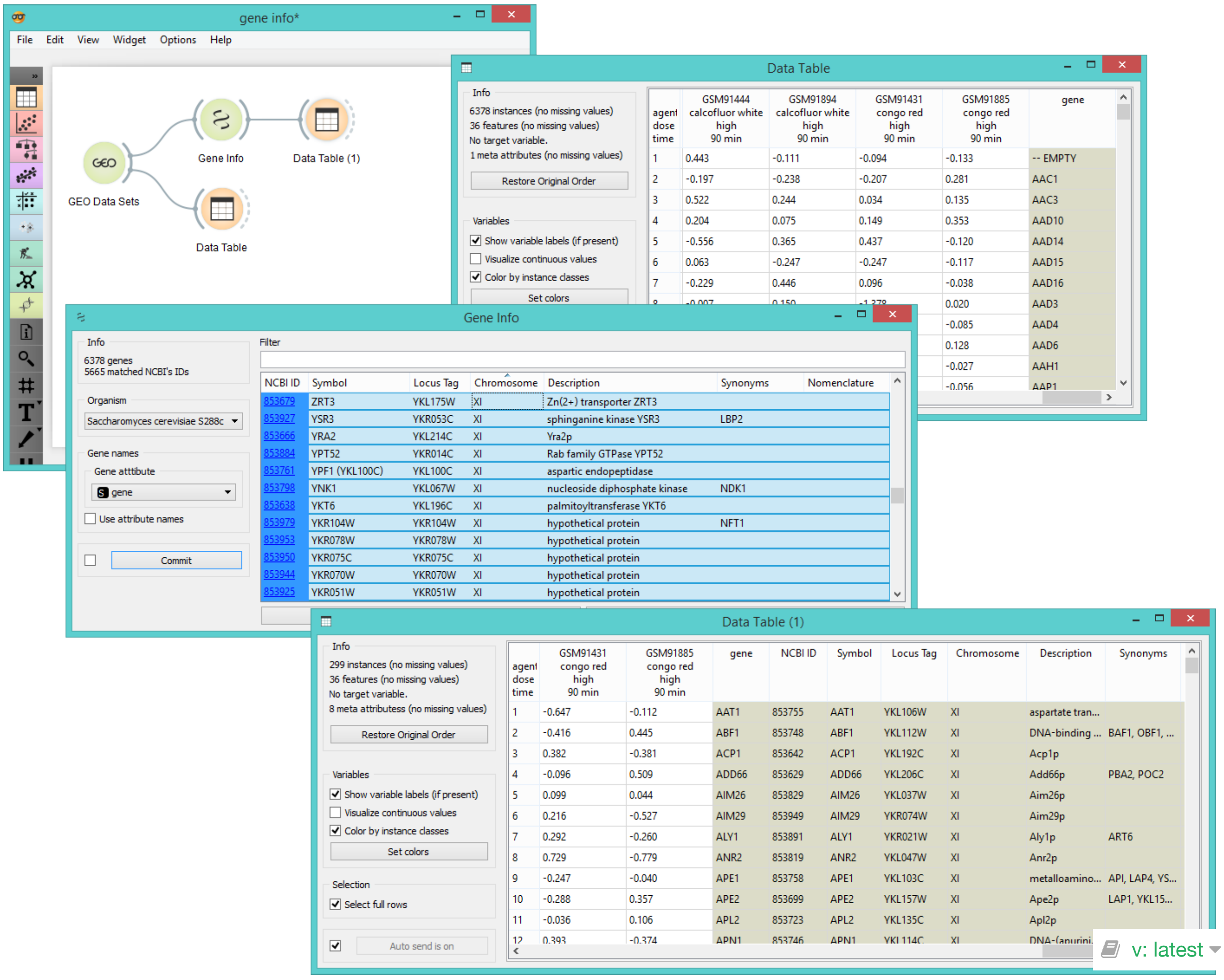
Use attribute names

NCBI ID	Symbol	Locus Tag	Chromosome	Description	Synonyms	Nomenclature
854598	COX1 (Q0045)	Q0045	MT	cytochrome c oxidase subunit 1	OXI3	
854593	AI1 (Q0050)	Q0050	MT	intron-encoded reverse transcripta...		
854594	AI2 (Q0055)	Q0055	MT	intron-encoded reverse transcripta...		
854595	AI3 (Q0060)	Q0060	MT	intron-encoded DNA endonucleas...		
854596	AI4 (Q0065)	Q0065	MT	intron-encoded DNA endonucleas...		
854597	AI5_ALPHA (Q0070)	Q0070	MT	intron-encoded DNA endonucleas...		
854599	AI5_BETA (Q0075)	Q0075	MT	intron-encoded DNA endonucleas...		
854600	ATP8 (Q0080)	Q0080	MT	F1F0 ATP synthase subunit 8	AAP1	
854601	ATP6 (Q0085)	Q0085	MT	F1F0 ATP synthase subunit a	OLI2, OLI4, PHO1	
854583	COB (Q0105)	Q0105	MT	cytochrome b	COB1, CYTB	
854604	BI2 (Q0110)	Q0110	MT	cytochrome b mRNA maturase bI2		
854605	BI3 (Q0115)	Q0115	MT	cytochrome b mRNA maturase bI3		

1. Information on data set size and genes that matched the NCBI ID's.
2. Select the organism of reference.
3. Set the source of gene names. If your gene names are placed as attributes names, select *Use attribute names*.
4. If *Auto commit is on*, changes will be communicated automatically. Alternatively click *Commit*.
5. In the row above the list you can filter the genes by search word(s). If you wish to output the filtered data, click *Select Filtered*.
6. If you wish to start from scratch, click *Clear Selection*.

Example

Below we first view the entire *Caffeine effect: time course and dose response* data set in the *Data Table* widget. Then we feed the same data into the *Gene Info*, where we select only the genes that are located on the 11th chromosome. We can observe these data in another *Data Table*, where additional information on the selected genes are appended as meta attributes.



Data Table

Info
6378 instances (no missing values)
36 features (no missing values)
No target variable.
1 meta attributes (no missing values)

Restore Original Order

Variables
- [x] Show variable labels (if present)
- [ ] Visualize continuous values
- [x] Color by instance classes

Set colors

agent dose time	GSM91444 calcofluor white high 90 min	GSM91894 calcofluor white high 90 min	GSM91431 congo red high 90 min	GSM91885 congo red high 90 min	gene
1	0.443	-0.111	-0.094	-0.133	-- EMPTY
2	-0.197	-0.238	-0.207	0.281	AAC1
3	0.522	0.244	0.034	0.135	AAC3
4	0.204	0.075	0.149	0.353	AAD10
5	-0.556	0.365	0.437	-0.120	AAD14
6	0.063	-0.247	-0.247	-0.117	AAD15
7	-0.229	0.446	0.096	-0.038	AAD16
8	-0.007	0.150	-1.378	0.020	AAD3
				-0.085	AAD4
				0.128	AAD6
				-0.027	AAH1
				-0.056	AAP1

Gene Info

Info
6378 genes
5665 matched NCBI's IDs

Organism
Saccharomyces cerevisiae S288c

Gene names
Gene atttibute
gene
- [ ] Use attribute names

- [ ] Commit

Filter

NCBI ID	Symbol	Locus Tag	Chromosome	Description	Synonyms	Nomenclature
853679	ZRT3	YKL175W	XI	Zn(2+) transporter ZRT3		
853927	YSR3	YKR053C	XI	sphinganine kinase YSR3	LBP2	
853666	YRA2	YKL214C	XI	Yra2p		
853884	YPT52	YKR014C	XI	Rab family GTPase YPT52		
853761	YPF1 (YKL100C)	YKL100C	XI	aspartic endopeptidase		
853798	YNK1	YKL067W	XI	nucleoside diphosphate kinase	NDK1	
853638	YKT6	YKL196C	XI	palmitoyltransferase YKT6		
853979	YKR104W	YKR104W	XI	hypothetical protein	NFT1	
853953	YKR078W	YKR078W	XI	hypothetical protein		
853950	YKR075C	YKR075C	XI	hypothetical protein		
853944	YKR070W	YKR070W	XI	hypothetical protein		
853925	YKR051W	YKR051W	XI	hypothetical protein		

Data Table (1)

Info
299 instances (no missing values)
36 features (no missing values)
No target variable.
8 meta attributes (no missing values)

Restore Original Order

Variables
- [x] Show variable labels (if present)
- [ ] Visualize continuous values
- [x] Color by instance classes

Set colors

Selection
- [x] Select full rows

- [x] Auto send is on

agent dose time	GSM91431 congo red high 90 min	GSM91885 congo red high 90 min	gene	NCBI ID	Symbol	Locus Tag	Chromosome	Description	Synonyms
1	-0.647	-0.112	AAT1	853755	AAT1	YKL106W	XI	aspartate tran...	
2	-0.416	0.445	ABF1	853748	ABF1	YKL112W	XI	DNA-binding ...	BAF1, OBF1, ...
3	0.382	-0.381	ACP1	853642	ACP1	YKL192C	XI	Acp1p	
4	-0.096	0.509	ADD66	853629	ADD66	YKL206C	XI	Add66p	PBA2, POC2
5	0.099	0.044	AIM26	853829	AIM26	YKL037W	XI	Aim26p	
6	0.216	-0.527	AIM29	853949	AIM29	YKR074W	XI	Aim29p	
7	0.292	-0.260	ALY1	853891	ALY1	YKR021W	XI	Aly1p	ART6
8	0.729	-0.779	ANR2	853819	ANR2	YKL047W	XI	Anr2p	
9	-0.247	-0.040	APE1	853758	APE1	YKL103C	XI	metalloamino...	API, LAP4, YS...
10	-0.288	0.357	APE2	853699	APE2	YKL157W	XI	Ape2p	LAP1, YKL15...
11	-0.036	0.106	APL2	853723	APL2	YKL135C	XI	Apl2p	
12	0.393	-0.374	APN1	853746	APN1	YKL114C	XI	DNA-(apurini...	

GenExpress

Gives access to **GenExpress** databases.

Signals

Inputs:

- (None)

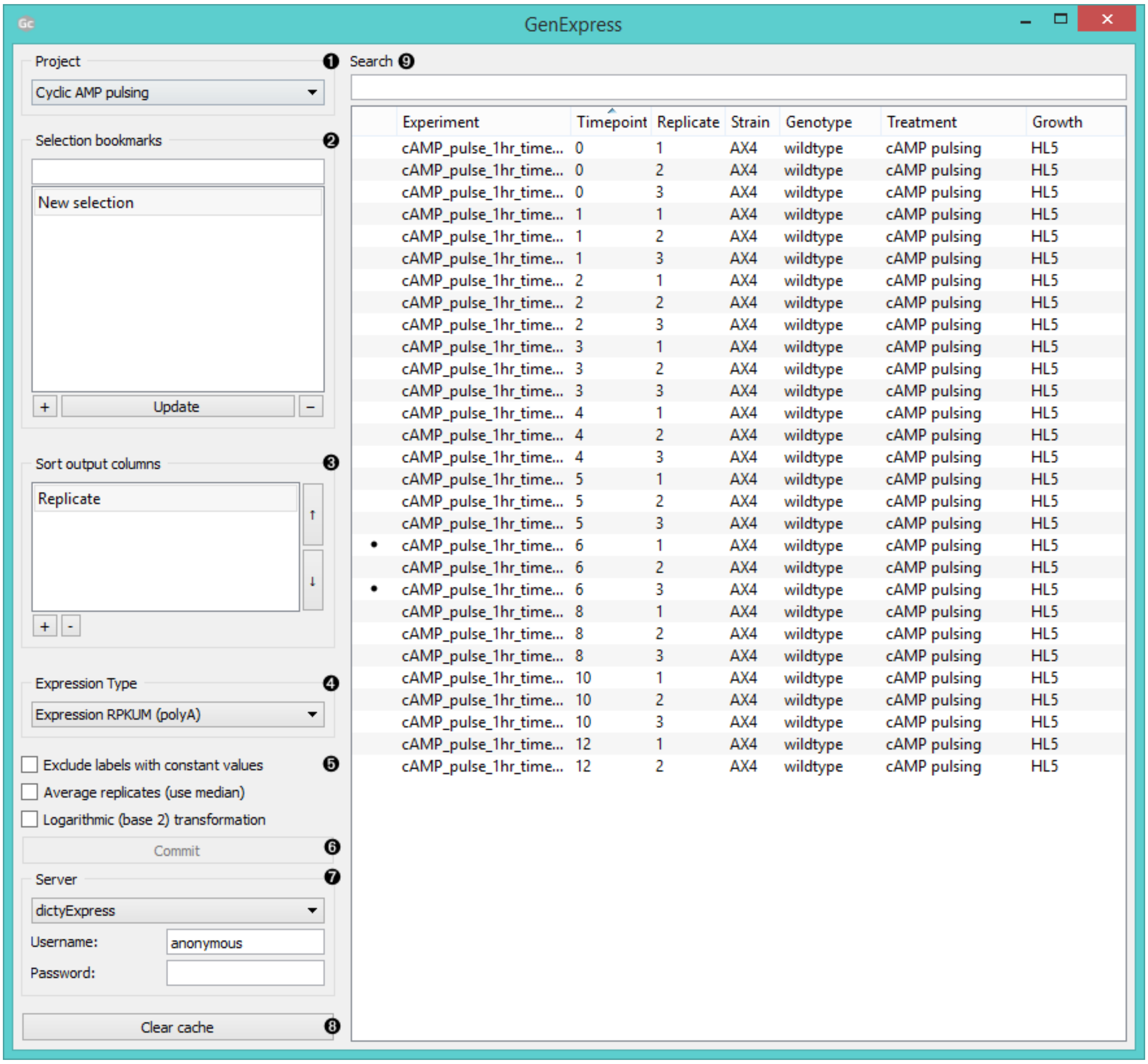
Outputs:

- **Data**

  Selected experiments. Each annotated column contains results of a single experiment or, if the corresponding option is chosen, the average of multiple replicates.

Description

GenExpress is a widget for a direct access to **GenExpress** database. It is very similar to the **PIPAx** and **GEO Data Sets** widgets as it allows you to download the data from selected experiments.



Project: Cyclic AMP pulsing

Selection bookmarks: New selection

Sort output columns: Replicate

Expression Type: Expression RPKUM (polyA)

☐ Exclude labels with constant values

☐ Average replicates (use median)

☐ Logarithmic (base 2) transformation

Server: dictyExpress

Username: anonymous

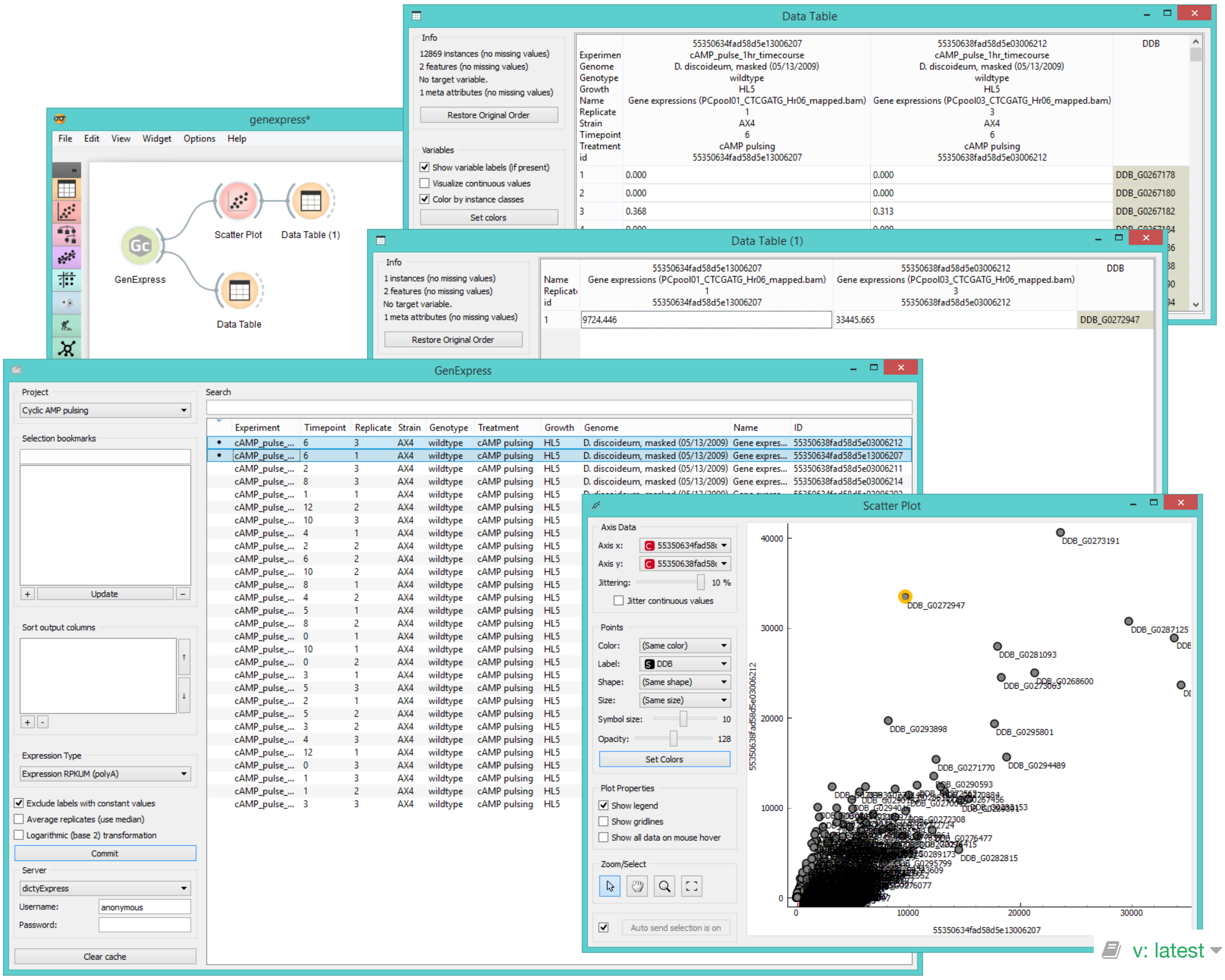
Password:

	Experiment	Timepoint	Replicate	Strain	Genotype	Treatment	Growth
	cAMP_pulse_1hr_time...	0	1	AX4	wildtype	cAMP pulsing	HL5
	cAMP_pulse_1hr_time...	0	2	AX4	wildtype	cAMP pulsing	HL5
	cAMP_pulse_1hr_time...	0	3	AX4	wildtype	cAMP pulsing	HL5
	cAMP_pulse_1hr_time...	1	1	AX4	wildtype	cAMP pulsing	HL5
	cAMP_pulse_1hr_time...	1	2	AX4	wildtype	cAMP pulsing	HL5
	cAMP_pulse_1hr_time...	1	3	AX4	wildtype	cAMP pulsing	HL5
	cAMP_pulse_1hr_time...	2	1	AX4	wildtype	cAMP pulsing	HL5
	cAMP_pulse_1hr_time...	2	2	AX4	wildtype	cAMP pulsing	HL5
	cAMP_pulse_1hr_time...	2	3	AX4	wildtype	cAMP pulsing	HL5
	cAMP_pulse_1hr_time...	3	1	AX4	wildtype	cAMP pulsing	HL5
	cAMP_pulse_1hr_time...	3	2	AX4	wildtype	cAMP pulsing	HL5
	cAMP_pulse_1hr_time...	3	3	AX4	wildtype	cAMP pulsing	HL5
	cAMP_pulse_1hr_time...	4	1	AX4	wildtype	cAMP pulsing	HL5
	cAMP_pulse_1hr_time...	4	2	AX4	wildtype	cAMP pulsing	HL5
	cAMP_pulse_1hr_time...	4	3	AX4	wildtype	cAMP pulsing	HL5
	cAMP_pulse_1hr_time...	5	1	AX4	wildtype	cAMP pulsing	HL5
	cAMP_pulse_1hr_time...	5	2	AX4	wildtype	cAMP pulsing	HL5
	cAMP_pulse_1hr_time...	5	3	AX4	wildtype	cAMP pulsing	HL5
•	cAMP_pulse_1hr_time...	6	1	AX4	wildtype	cAMP pulsing	HL5
	cAMP_pulse_1hr_time...	6	2	AX4	wildtype	cAMP pulsing	HL5
•	cAMP_pulse_1hr_time...	6	3	AX4	wildtype	cAMP pulsing	HL5
	cAMP_pulse_1hr_time...	8	1	AX4	wildtype	cAMP pulsing	HL5
	cAMP_pulse_1hr_time...	8	2	AX4	wildtype	cAMP pulsing	HL5
	cAMP_pulse_1hr_time...	8	3	AX4	wildtype	cAMP pulsing	HL5
	cAMP_pulse_1hr_time...	10	1	AX4	wildtype	cAMP pulsing	HL5
	cAMP_pulse_1hr_time...	10	2	AX4	wildtype	cAMP pulsing	HL5
	cAMP_pulse_1hr_time...	10	3	AX4	wildtype	cAMP pulsing	HL5
	cAMP_pulse_1hr_time...	12	1	AX4	wildtype	cAMP pulsing	HL5
	cAMP_pulse_1hr_time...	12	2	AX4	wildtype	cAMP pulsing	HL5

1. Choose a projects to source your data from.
2. Use *Selection bookmarks* to save a selection: select experiments, click the "+" button and name the set. To add experiments to your set, click on the set name, select additional experiments and click *Update*. To remove the set click "-".
3. In *Sort output columns* set the attributes by which the output columns are sorted. Add attributes with a "+" button and remove them with "-". Switch the sorting order with arrows on the right.
4. Set the expression type for your output data.
   - **Expression RPKM** outputs data in *reads per kilobase of transcript per million mapped reads*
   - **Expression RPKUM** outputs only RPKUM data.
   - **Read counts (raw)** outputs raw read count data.
     The polyA variants use only polyA (mRNA) mapped hits.
5. **Exclude labels with constant values** removes labels that are the same for all selected experiments.
   Average replicates (use median) averages identical experiments by using medians as values.
   Logarithmic (base 2) transformation returns $\log_2$(value+1) for each value.
6. Click *Commit* to output selected data.
7. Select the server you wish to access the data from. Log in to access private data.
8. *Clear cache* removes the uploaded data sets from internal memory.
9. Experiments can be filtered with the *Search* box. To select which attributes to display right-click on the header. To select multiple experiments click them while holding the *Control/Command* key.

Example

In the schema below we connected **GenExpress** to **Data Table** to view the gene expression reads and then to **Scatter Plot**, where we chose to view expression levels from two experiments. In the plot we select an outlier and view it in another **Data Table**.



Data Table

Info
12869 instances (no missing values)
2 features (no missing values)
No target variable.
1 meta attributes (no missing values)
Restore Original Order

Variables
[✓] Show variable labels (if present)
[ ] Visualize continuous values
[✓] Color by instance classes
Set colors

	55350634fad58d5e13006207	55350638fad58d5e03006212	DDB
Experimen	cAMP_pulse_1hr_timecourse	cAMP_pulse_1hr_timecourse	
Genome	D. discoideum, masked (05/13/2009)	D. discoideum, masked (05/13/2009)	
Genotype	wildtype	wildtype	
Growth	HL5	HL5	
Name	Gene expressions (PCpool01_CTCGATG_Hr06_mapped.bam)	Gene expressions (PCpool03_CTCGATG_Hr06_mapped.bam)	
Replicate	1	3	
Strain	AX4	AX4	
Timepoint	6	6	
Treatment	cAMP pulsing	cAMP pulsing	
id	55350634fad58d5e13006207	55350638fad58d5e03006212	
1	0.000	0.000	DDB_G0267178
2	0.000	0.000	DDB_G0267180
3	0.368	0.313	DDB_G0267182

Data Table (1)

Info
1 instances (no missing values)
2 features (no missing values)
No target variable.
1 meta attributes (no missing values)
Restore Original Order

	55350634fad58d5e13006207	55350638fad58d5e03006212	DDB
Name	Gene expressions (PCpool01_CTCGATG_Hr06_mapped.bam)	Gene expressions (PCpool03_CTCGATG_Hr06_mapped.bam)	
Replicat	1	3	
id	55350634fad58d5e13006207	55350638fad58d5e03006212	
1	9724.446	33445.665	DDB_G0272947

GenExpress

Project: Cyclic AMP pulsing
Selection bookmarks
Update
Sort output columns
Expression Type: Expression RPKUM (polyA)
[✓] Exclude labels with constant values
[ ] Average replicates (use median)
[ ] Logarithmic (base 2) transformation
Commit
Server: dictyExpress
Username: anonymous
Password:
Clear cache

Search

	Experiment	Timepoint	Replicate	Strain	Genotype	Treatment	Growth	Genome	Name	ID
•	cAMP_pulse_...	6	3	AX4	wildtype	cAMP pulsing	HL5	D. discoideum, masked (05/13/2009)	Gene expres...	55350638fad58d5e03006212
•	cAMP_pulse_...	6	1	AX4	wildtype	cAMP pulsing	HL5	D. discoideum, masked (05/13/2009)	Gene expres...	55350634fad58d5e13006207
	cAMP_pulse_...	2	3	AX4	wildtype	cAMP pulsing	HL5	D. discoideum, masked (05/13/2009)	Gene expres...	55350638fad58d5e03006211
	cAMP_pulse_...	8	3	AX4	wildtype	cAMP pulsing	HL5	D. discoideum, masked (05/13/2009)	Gene expres...	55350638fad58d5e03006214
	cAMP_pulse_...	1	1	AX4	wildtype	cAMP pulsing	HL5			
	cAMP_pulse_...	12	2	AX4	wildtype	cAMP pulsing	HL5			
	cAMP_pulse_...	10	3	AX4	wildtype	cAMP pulsing	HL5			
	cAMP_pulse_...	4	1	AX4	wildtype	cAMP pulsing	HL5			
	cAMP_pulse_...	2	2	AX4	wildtype	cAMP pulsing	HL5			
	cAMP_pulse_...	6	2	AX4	wildtype	cAMP pulsing	HL5			
	cAMP_pulse_...	10	2	AX4	wildtype	cAMP pulsing	HL5			
	cAMP_pulse_...	8	1	AX4	wildtype	cAMP pulsing	HL5			
	cAMP_pulse_...	4	2	AX4	wildtype	cAMP pulsing	HL5			
	cAMP_pulse_...	5	1	AX4	wildtype	cAMP pulsing	HL5			
	cAMP_pulse_...	8	2	AX4	wildtype	cAMP pulsing	HL5			
	cAMP_pulse_...	0	1	AX4	wildtype	cAMP pulsing	HL5			
	cAMP_pulse_...	10	1	AX4	wildtype	cAMP pulsing	HL5			
	cAMP_pulse_...	0	2	AX4	wildtype	cAMP pulsing	HL5			
	cAMP_pulse_...	3	1	AX4	wildtype	cAMP pulsing	HL5			
	cAMP_pulse_...	5	3	AX4	wildtype	cAMP pulsing	HL5			
	cAMP_pulse_...	2	1	AX4	wildtype	cAMP pulsing	HL5			
	cAMP_pulse_...	5	2	AX4	wildtype	cAMP pulsing	HL5			
	cAMP_pulse_...	3	2	AX4	wildtype	cAMP pulsing	HL5			
	cAMP_pulse_...	4	3	AX4	wildtype	cAMP pulsing	HL5			
	cAMP_pulse_...	12	1	AX4	wildtype	cAMP pulsing	HL5			
	cAMP_pulse_...	0	3	AX4	wildtype	cAMP pulsing	HL5			
	cAMP_pulse_...	1	3	AX4	wildtype	cAMP pulsing	HL5			
	cAMP_pulse_...	1	2	AX4	wildtype	cAMP pulsing	HL5			
	cAMP_pulse_...	3	3	AX4	wildtype	cAMP pulsing	HL5			

Scatter Plot

Axis Data
Axis x: 55350634fad58d
Axis y: 55350638fad58d
Jittering: 10 %
[ ] Jitter continuous values

Points
Color: (Same color)
Label: DDB
Shape: (Same shape)
Size: (Same size)
Symbol size: 10
Opacity: 128
Set Colors

Plot Properties
[✓] Show legend
[ ] Show gridlines
[ ] Show all data on mouse hover

Zoom/Select

[✓] Auto send selection is on

GEO Data Sets

Provides access to data sets from gene expression omnibus (GEO DataSets).

Signals

Inputs:

- (None)

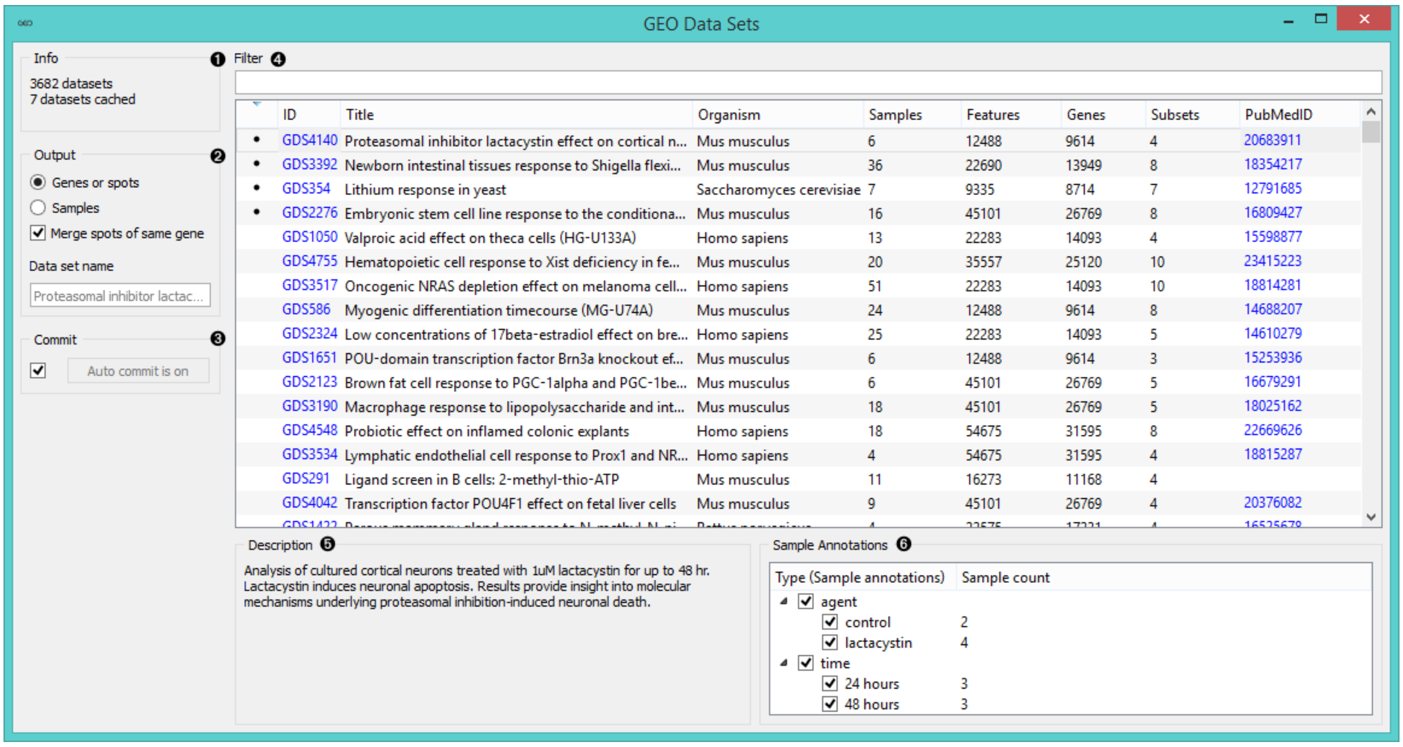
Outputs:

- **Data**

Data set selected in the widget with genes or samples in rows.

Description

GEO DataSets is a data base of gene expression curated profiles maintained by NCBI and included in the Gene Expression Omnibus. This Orange widget provides access to all its data sets and outputs a data set selected for further processing. For convenience, each dowloaded data set is stored locally.


GEO Data Sets

Info ❶
3682 datasets
7 datasets cached

Output ❷
(●) Genes or spots
( ) Samples
[✓] Merge spots of same gene
Data set name: Proteasomal inhibitor lactac...

Commit ❸
[✓] Auto commit is on

Filter ❹

	ID	Title	Organism	Samples	Features	Genes	Subsets	PubMedID
•	GDS4140	Proteasomal inhibitor lactacystin effect on cortical n...	Mus musculus	6	12488	9614	4	20683911
•	GDS3392	Newborn intestinal tissues response to Shigella flexi...	Mus musculus	36	22690	13949	8	18354217
•	GDS354	Lithium response in yeast	Saccharomyces cerevisiae	7	9335	8714	7	12791685
•	GDS2276	Embryonic stem cell line response to the conditiona...	Mus musculus	16	45101	26769	8	16809427
	GDS1050	Valproic acid effect on theca cells (HG-U133A)	Homo sapiens	13	22283	14093	4	15598877
	GDS4755	Hematopoietic cell response to Xist deficiency in fe...	Mus musculus	20	35557	25120	10	23415223
	GDS3517	Oncogenic NRAS depletion effect on melanoma cell...	Homo sapiens	51	22283	14093	10	18814281
	GDS586	Myogenic differentiation timecourse (MG-U74A)	Mus musculus	24	12488	9614	8	14688207
	GDS2324	Low concentrations of 17beta-estradiol effect on bre...	Homo sapiens	25	22283	14093	5	14610279
	GDS1651	POU-domain transcription factor Brn3a knockout ef...	Mus musculus	6	12488	9614	3	15253936
	GDS2123	Brown fat cell response to PGC-1alpha and PGC-1be...	Mus musculus	6	45101	26769	5	16679291
	GDS3190	Macrophage response to lipopolysaccharide and int...	Mus musculus	18	45101	26769	5	18025162
	GDS4548	Probiotic effect on inflamed colonic explants	Homo sapiens	18	54675	31595	8	22669626
	GDS3534	Lymphatic endothelial cell response to Prox1 and NR...	Homo sapiens	4	54675	31595	4	18815287
	GDS291	Ligand screen in B cells: 2-methyl-thio-ATP	Mus musculus	11	16273	11168	4	
	GDS4042	Transcription factor POU4F1 effect on fetal liver cells	Mus musculus	9	45101	26769	4	20376082

Description ❺
Analysis of cultured cortical neurons treated with 1uM lactacystin for up to 48 hr. Lactacystin induces neuronal apoptosis. Results provide insight into molecular mechanisms underlying proteasomal inhibition-induced neuronal death.

Sample Annotations ❻

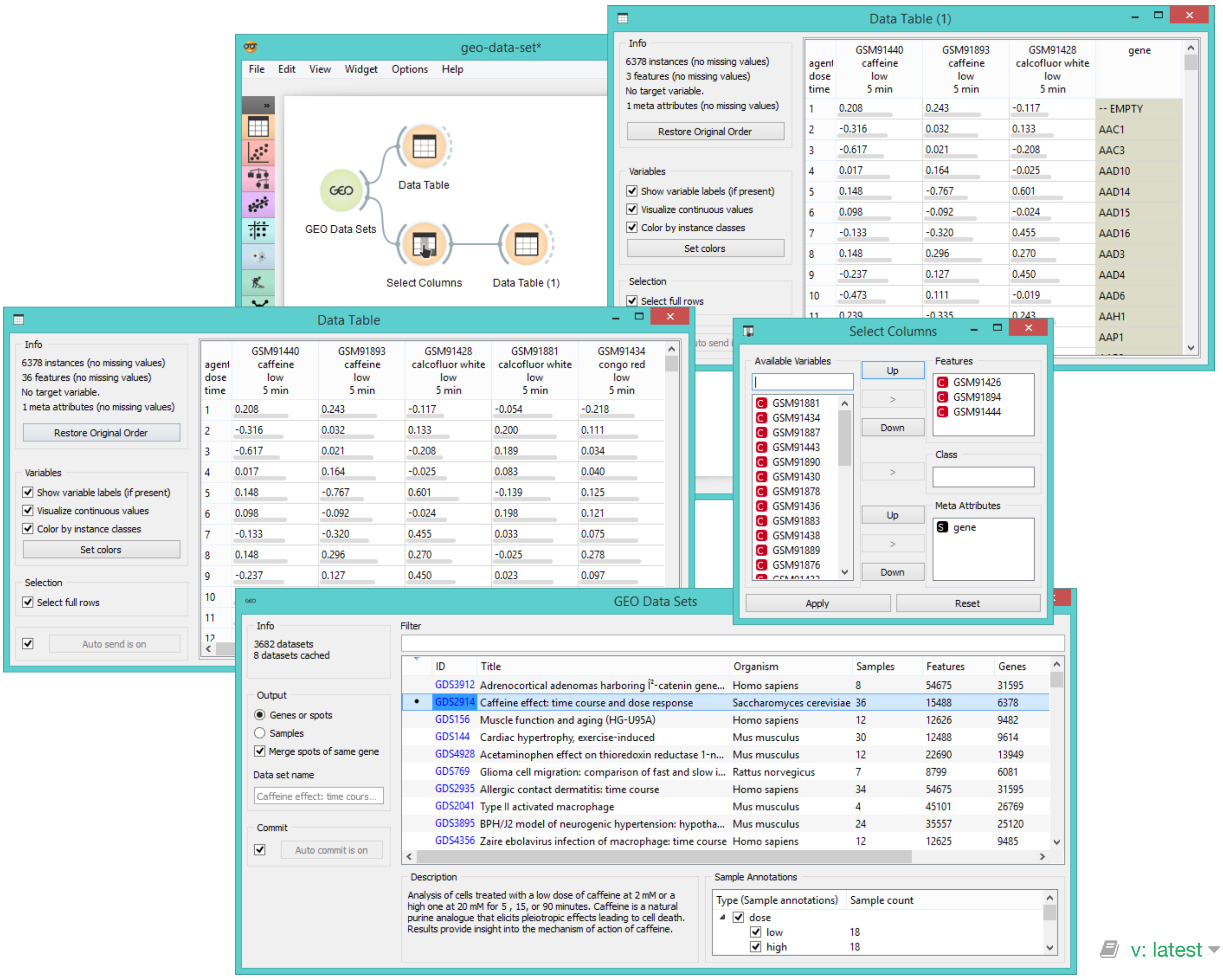
Type (Sample annotations)	Sample count
[✓] agent	
[✓] control	2
[✓] lactacystin	4
[✓] time	
[✓] 24 hours	3
[✓] 48 hours	3

1. Information on the GEO data set collection. Cached data sets are the ones currently stored on the computer.
2. Output features. If *Genes or spots* is selected, genes (or spots) will be used as attributes. Alternatively samples will be used as attributes. *Merge spots of same gene* averages measures of the same gene. Finally, in the *Data set name* you can rename the output data. GEO title will be used as a default name.
3. If *Auto commit is on*, then the selected data set will be automatically communicated to other widgets. Alternatively, click *Commit*.
4. *Filter* allows you to search for the data set. Below you see a list of GEO data sets with an ID number (link to the NCBI Data Set Browser), title, organism used in the experiment, number of samples, features, genes, subsets and a reference number for the PubMed journal (link to the article abstract).
5. Short description of the experiment from which the data set is sourced.

6. Select which *Sample Annotations* will be used in the output.

Example

GEO Data Sets is similar to the **File** widget. In the example below we selected *Caffeine effect: time dose and response* data set from the GEO data base and used *Genes or spots* as attributes. We inspected the data in *Data Table*. Then we selected 3 genes in the **Select Columns** widget for a detailed analysis in another data table.



Data Table (1)

Info
6378 instances (no missing values)
3 features (no missing values)
No target variable.
1 meta attributes (no missing values)

Restore Original Order

Variables
☑ Show variable labels (if present)
☑ Visualize continuous values
☑ Color by instance classes
Set colors

Selection
☑ Select full rows

agent dose time	GSM91440 caffeine low 5 min	GSM91893 caffeine low 5 min	GSM91428 calcofluor white low 5 min	gene
1	0.208	0.243	-0.117	-- EMPTY
2	-0.316	0.032	0.133	AAC1
3	-0.617	0.021	-0.208	AAC3
4	0.017	0.164	-0.025	AAD10
5	0.148	-0.767	0.601	AAD14
6	0.098	-0.092	-0.024	AAD15
7	-0.133	-0.320	0.455	AAD16
8	0.148	0.296	0.270	AAD3
9	-0.237	0.127	0.450	AAD4
10	-0.473	0.111	-0.019	AAD6
11	0.239	-0.335	0.243	AAH1
				AAP1

Data Table

Info
6378 instances (no missing values)
36 features (no missing values)
No target variable.
1 meta attributes (no missing values)

Restore Original Order

Variables
☑ Show variable labels (if present)
☑ Visualize continuous values
☑ Color by instance classes
Set colors

Selection
☑ Select full rows

☑ Auto send is on

agent dose time	GSM91440 caffeine low 5 min	GSM91893 caffeine low 5 min	GSM91428 calcofluor white low 5 min	GSM91881 calcofluor white low 5 min	GSM91434 congo red low 5 min
1	0.208	0.243	-0.117	-0.054	-0.218
2	-0.316	0.032	0.133	0.200	0.111
3	-0.617	0.021	-0.208	0.189	0.034
4	0.017	0.164	-0.025	0.083	0.040
5	0.148	-0.767	0.601	-0.139	0.125
6	0.098	-0.092	-0.024	0.198	0.121
7	-0.133	-0.320	0.455	0.033	0.075
8	0.148	0.296	0.270	-0.025	0.278
9	-0.237	0.127	0.450	0.023	0.097
10					
11					
12					

Select Columns

Available Variables
GSM91881
GSM91434
GSM91887
GSM91443
GSM91890
GSM91430
GSM91878
GSM91436
GSM91883
GSM91438
GSM91889
GSM91876

Features
GSM91426
GSM91894
GSM91444

Class

Meta Attributes
gene

Up > Down
Apply Reset

GEO Data Sets

Info
3682 datasets
8 datasets cached

Output
(●) Genes or spots
( ) Samples
☑ Merge spots of same gene

Data set name
Caffeine effect: time cours...

Commit
☑ Auto commit is on

Filter

ID	Title	Organism	Samples	Features	Genes
GDS3912	Adrenocortical adenomas harboring Î²-catenin gene...	Homo sapiens	8	54675	31595
• GDS2914	Caffeine effect: time course and dose response	Saccharomyces cerevisiae	36	15488	6378
GDS156	Muscle function and aging (HG-U95A)	Homo sapiens	12	12626	9482
GDS144	Cardiac hypertrophy, exercise-induced	Mus musculus	30	12488	9614
GDS4928	Acetaminophen effect on thioredoxin reductase 1-n...	Mus musculus	12	22690	13949
GDS769	Glioma cell migration: comparison of fast and slow i...	Rattus norvegicus	7	8799	6081
GDS2935	Allergic contact dermatitis: time course	Homo sapiens	34	54675	31595
GDS2041	Type II activated macrophage	Mus musculus	4	45101	26769
GDS3895	BPH/J2 model of neurogenic hypertension: hypotha...	Mus musculus	24	35557	25120
GDS4356	Zaire ebolavirus infection of macrophage: time course	Homo sapiens	12	12625	9485

Description
Analysis of cells treated with a low dose of caffeine at 2 mM or a high one at 20 mM for 5 , 15, or 90 minutes. Caffeine is a natural purine analogue that elicits pleiotropic effects leading to cell death. Results provide insight into the mechanism of action of caffeine.

Sample Annotations

Type (Sample annotations)	Sample count
☑ dose	
☑ low	18
☑ high	18

GO Browser

Provides access to Gene Ontology database.

Signals

Inputs:

- **Cluster Data**

  Data on clustered genes.

- **Reference Data**

  Data with genes for the reference set (optional).

Outputs:

- **Data on Selected Genes**

  Data on genes from the selected GO node.

- **Data on Unselected Genes**

  Data on genes from GO nodes that weren't selected.

- **Data on Unknown Genes**

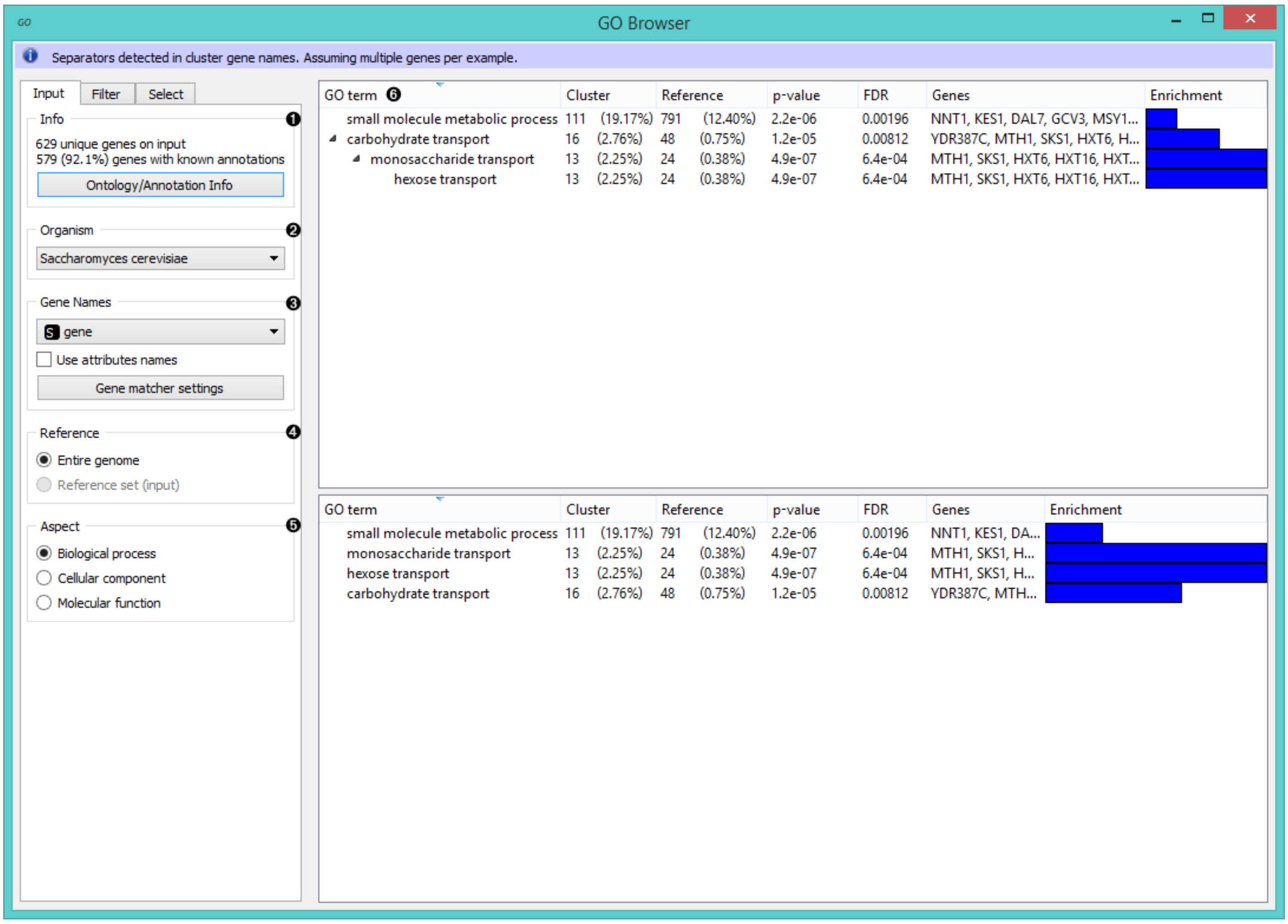
  Data on genes that are not in the GO database.

- **Enrichment Report**

  Data on GO enrichment analysis.

Description

GO Browser widget provides access to *Gene Ontology database*. Gene Ontology (GO) classifies genes and gene products to terms organized in a graph structure called an ontology. The widget takes any data on genes as an input (it is best to input statistically significant genes, for example from the output of the **Differential Expression** widget) and shows a ranked list of GO terms with p-values. This is a great tool for finding biological processes that are over- or under-represented in a particular gene set. The user can filter input data by selecting terms in a list.


GO Browser

Separators detected in cluster gene names. Assuming multiple genes per example.

Input | Filter | Select

Info ❶
629 unique genes on input
579 (92.1%) genes with known annotations
Ontology/Annotation Info

Organism ❷
Saccharomyces cerevisiae

Gene Names ❸
gene
☐ Use attributes names
Gene matcher settings

Reference ❹
◉ Entire genome
○ Reference set (input)

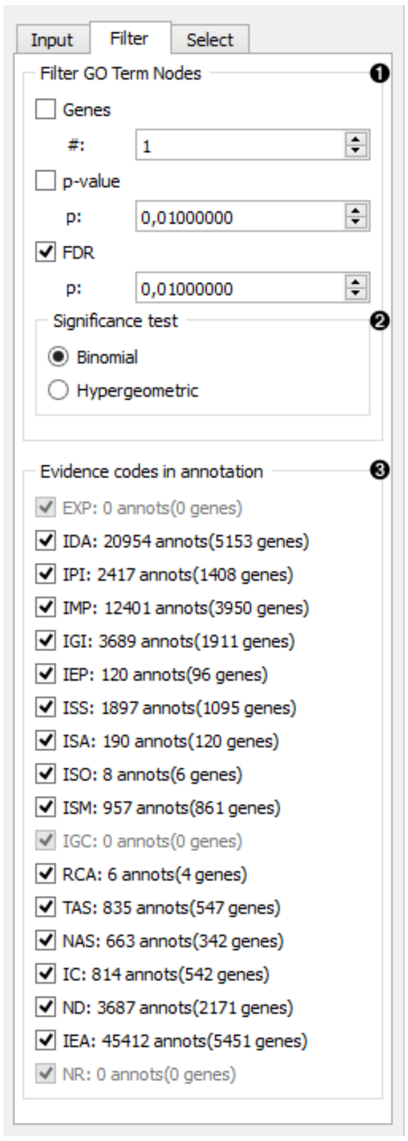
Aspect ❺
◉ Biological process
○ Cellular component
○ Molecular function

GO term ❻	Cluster		Reference		p-value	FDR	Genes	Enrichment
small molecule metabolic process	111	(19.17%)	791	(12.40%)	2.2e-06	0.00196	NNT1, KES1, DAL7, GCV3, MSY1...	🟦
◢ carbohydrate transport	16	(2.76%)	48	(0.75%)	1.2e-05	0.00812	YDR387C, MTH1, SKS1, HXT6, H...	🟦
◢ monosaccharide transport	13	(2.25%)	24	(0.38%)	4.9e-07	6.4e-04	MTH1, SKS1, HXT6, HXT16, HXT...	🟦
hexose transport	13	(2.25%)	24	(0.38%)	4.9e-07	6.4e-04	MTH1, SKS1, HXT6, HXT16, HXT...	🟦

GO term	Cluster		Reference		p-value	FDR	Genes	Enrichment
small molecule metabolic process	111	(19.17%)	791	(12.40%)	2.2e-06	0.00196	NNT1, KES1, DA...	🟦
monosaccharide transport	13	(2.25%)	24	(0.38%)	4.9e-07	6.4e-04	MTH1, SKS1, H...	🟦
hexose transport	13	(2.25%)	24	(0.38%)	4.9e-07	6.4e-04	MTH1, SKS1, H...	🟦
carbohydrate transport	16	(2.76%)	48	(0.75%)	1.2e-05	0.00812	YDR387C, MTH...	🟦

INPUT tab

1. Information on the input data set. *Ontology/Annotation Info* reports the current status of the GO database.
2. Select organism for the GO term analysis.
3. Use this attribute to extract gene names for the input data. You can use attribute names as gene names and adjust gene matching in the *Gene matcher settings* box.
4. Select the reference. You can either have the *entire genome* as reference or a *reference set* from the input.
5. Select the ontology where you want to calculate the enrichment. There are three *Aspect* options:
   - **Biological process**
   - **Cellular component**
   - **Molecular function**
6. A ranked tree (upper pane) and list (lower pane) of GO terms for the selected aspect:
   - **GO term**
   - **Cluster**: number of genes from the input that are also annotated to a particular GO term (and its proportion in all the genes from that term).
   - **Reference**: number of genes that are annotated to a particular GO term (and its proportion in the entire genome).
   - **P-value**: probability of seeing as many or more genes at random. The closer the p-value is to zero, the more significant a particular GO term is. Value is written in e notation.
   - **FDR**: false discovery rate - a multiple testing correction that means a proportion of false discoveries among all discoveries up to that FDR value.
   - **Genes**: genes in a biological process.
   - **Enrichment** level


Input | Filter | Select

Filter GO Term Nodes ❶

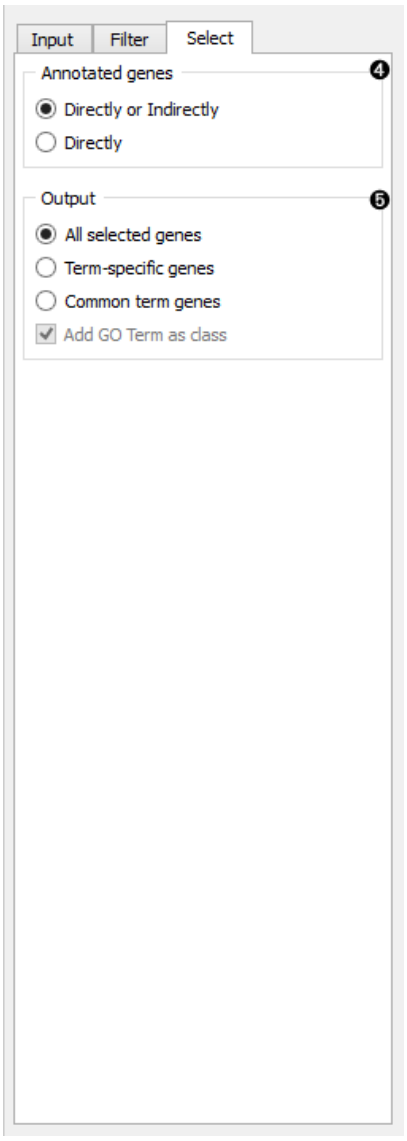
[ ] Genes
#: 1
[ ] p-value
p: 0,01000000
[✓] FDR
p: 0,01000000

Significance test ❷

(●) Binomial
( ) Hypergeometric

Evidence codes in annotation ❸

[✓] EXP: 0 annots(0 genes)
[✓] IDA: 20954 annots(5153 genes)
[✓] IPI: 2417 annots(1408 genes)
[✓] IMP: 12401 annots(3950 genes)
[✓] IGI: 3689 annots(1911 genes)
[✓] IEP: 120 annots(96 genes)
[✓] ISS: 1897 annots(1095 genes)
[✓] ISA: 190 annots(120 genes)
[✓] ISO: 8 annots(6 genes)
[✓] ISM: 957 annots(861 genes)
[✓] IGC: 0 annots(0 genes)
[✓] RCA: 6 annots(4 genes)
[✓] TAS: 835 annots(547 genes)
[✓] NAS: 663 annots(342 genes)
[✓] IC: 814 annots(542 genes)
[✓] ND: 3687 annots(2171 genes)
[✓] IEA: 45412 annots(5451 genes)
[✓] NR: 0 annots(0 genes)


Input | Filter | Select

Annotated genes ❹

(●) Directly or Indirectly
( ) Directly

Output ❺

(●) All selected genes
( ) Term-specific genes
( ) Common term genes
[✓] Add GO Term as class

FILTER tab

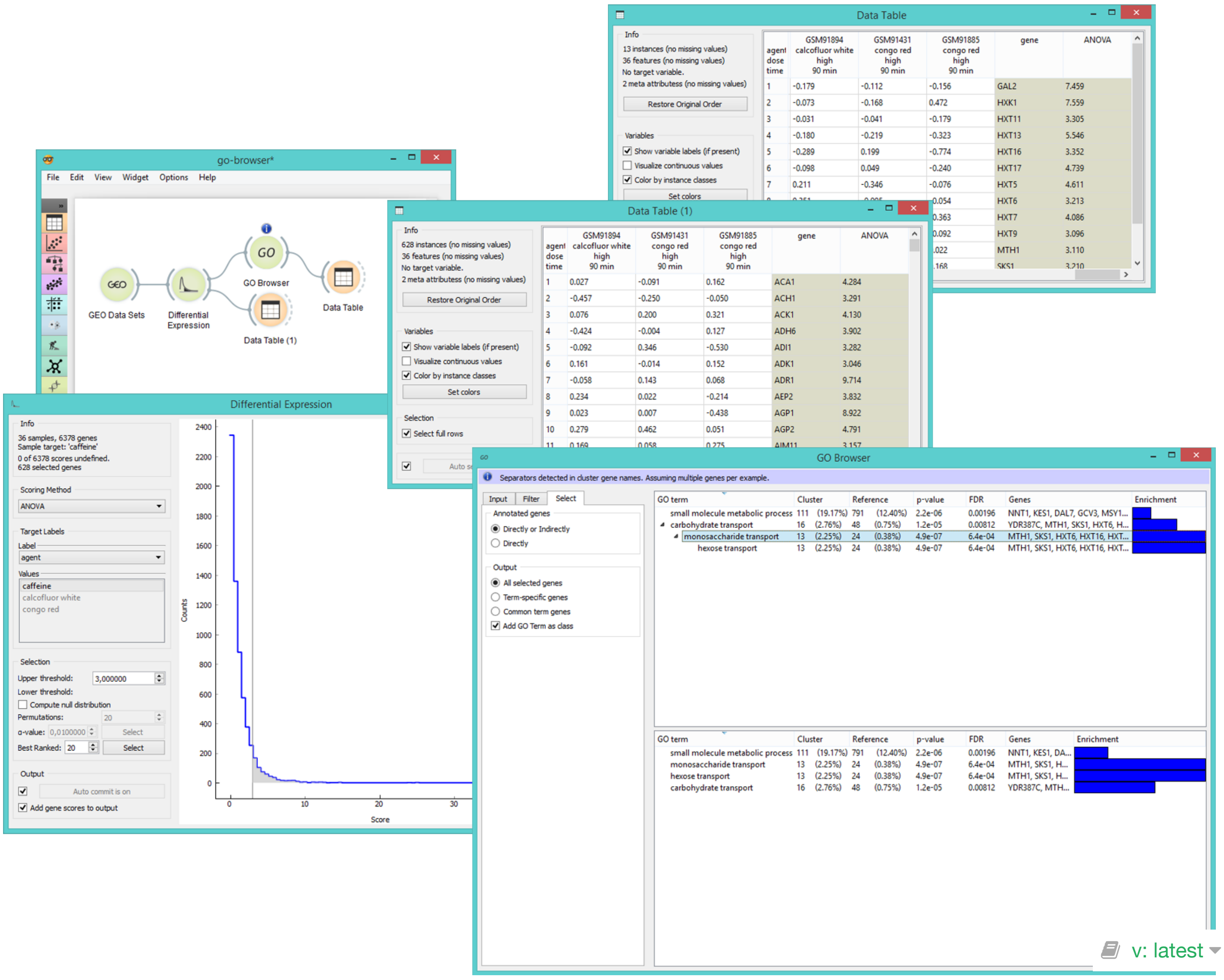
1. *Filter GO Term Nodes* by:
   - **Genes** is a minimal number of genes mapped to a term
   - **P-value** is a max term p-value
   - **FDR**: is a max term false discovery rate
2. *Significance test* specifies distribution to use for null hypothesis:
   - **Binomial**: use a binomial distribution
   - **Hypergeometric**: use a hypergeometric distribution
3. *Evidence codes in annotation* show how the annotation to a particular term is supported.

SELECT tab

1. *Annotated genes* outputs genes that are:
   - **Directly or Indirectly** annotated (direct and inherited annotations)
   - **Directly** annotated (inherited annotations won't be in the output)
2. *Output*:
   - **All selected genes**: outputs genes annotated to all selected GO terms
   - **Term-specific genes**: outputs genes that appear in only one of selected GO terms
   - **Common term genes**: outputs genes common to all selected GO terms
   - **Add GO Term as class**: adds GO terms as class attribute

Example

In the example below we have used **GEO Data Sets** widget, in which we have selected *Caffeine effects: time course and dose response* data set, and connected it to a **Differential Analysis**. Differential analysis allows us to select genes with the highest statistical relevance (we used ANOVA scoring) and feed them to **GO Browser**. This widget lists four biological processes for our selected genes. Say we are interested in finding out more about *monosaccharide transport* as this term has a high enrichment rate. To learn more about which genes are annotated to this GO term we view it in the **Data Table**, where we see all the genes participating in this process listed.



Data Table

Info
13 instances (no missing values)
36 features (no missing values)
No target variable.
2 meta attributes (no missing values)

Variables
☑ Show variable labels (if present)
☐ Visualize continuous values
☑ Color by instance classes

agent dose time	GSM91894 calcofluor white high 90 min	GSM91431 congo red high 90 min	GSM91885 congo red high 90 min	gene	ANOVA
1	-0.179	-0.112	-0.156	GAL2	7.459
2	-0.073	-0.168	0.472	HXK1	7.559
3	-0.031	-0.041	-0.179	HXT11	3.305
4	-0.180	-0.219	-0.323	HXT13	5.546
5	-0.289	0.199	-0.774	HXT16	3.352
6	-0.098	0.049	-0.240	HXT17	4.739
7	0.211	-0.346	-0.076	HXT5	4.611
			-0.054	HXT6	3.213
			0.363	HXT7	4.086
			0.092	HXT9	3.096
			.022	MTH1	3.110
			.168	SKS1	3.210

Data Table (1)

Info
628 instances (no missing values)
36 features (no missing values)
No target variable.
2 meta attributes (no missing values)

Variables
☑ Show variable labels (if present)
☐ Visualize continuous values
☑ Color by instance classes

Selection
☑ Select full rows

agent dose time	GSM91894 calcofluor white high 90 min	GSM91431 congo red high 90 min	GSM91885 congo red high 90 min	gene	ANOVA
1	0.027	-0.091	0.162	ACA1	4.284
2	-0.457	-0.250	-0.050	ACH1	3.291
3	0.076	0.200	0.321	ACK1	4.130
4	-0.424	-0.004	0.127	ADH6	3.902
5	-0.092	0.346	-0.530	ADI1	3.282
6	0.161	-0.014	0.152	ADK1	3.046
7	-0.058	0.143	0.068	ADR1	9.714
8	0.234	0.022	-0.214	AEP2	3.832
9	0.023	0.007	-0.438	AGP1	8.922
10	0.279	0.462	0.051	AGP2	4.791
11	0.169	0.058	0.275	AIM11	3.157

Differential Expression

Info
36 samples, 6378 genes
Sample target: 'caffeine'
0 of 6378 scores undefined.
628 selected genes

Scoring Method: ANOVA

Target Labels
Label: agent
Values: caffeine, calcofluor white, congo red

Selection
Upper threshold: 3,000000
Lower threshold:
☐ Compute null distribution
Permutations: 20
α-value: 0,0100000
Best Ranked: 20

Output
☑ Auto commit is on
☑ Add gene scores to output

GO Browser

Separators detected in cluster gene names. Assuming multiple genes per example.

Annotated genes
(●) Directly or Indirectly
( ) Directly

Output
(●) All selected genes
( ) Term-specific genes
( ) Common term genes
☑ Add GO Term as class

GO term	Cluster		Reference		p-value	FDR	Genes
small molecule metabolic process	111	(19.17%)	791	(12.40%)	2.2e-06	0.00196	NNT1, KES1, DAL7, GCV3, MSY1...
carbohydrate transport	16	(2.76%)	48	(0.75%)	1.2e-05	0.00812	YDR387C, MTH1, SKS1, HXT6, H...
monosaccharide transport	13	(2.25%)	24	(0.38%)	4.9e-07	6.4e-04	MTH1, SKS1, HXT6, HXT16, HXT...
hexose transport	13	(2.25%)	24	(0.38%)	4.9e-07	6.4e-04	MTH1, SKS1, HXT6, HXT16, HXT...

GO term	Cluster		Reference		p-value	FDR	Genes
small molecule metabolic process	111	(19.17%)	791	(12.40%)	2.2e-06	0.00196	NNT1, KES1, DA...
monosaccharide transport	13	(2.25%)	24	(0.38%)	4.9e-07	6.4e-04	MTH1, SKS1, H...
hexose transport	13	(2.25%)	24	(0.38%)	4.9e-07	6.4e-04	MTH1, SKS1, H...
carbohydrate transport	16	(2.76%)	48	(0.75%)	1.2e-05	0.00812	YDR387C, MTH...

v: latest

KEGG Pathways

Diagrams of molecular interactions, reactions, and relations.

Signals

Inputs:

- **Data**

  Data set.

- **Reference**

  Referential data set.

Outputs:

- **Selected Data**

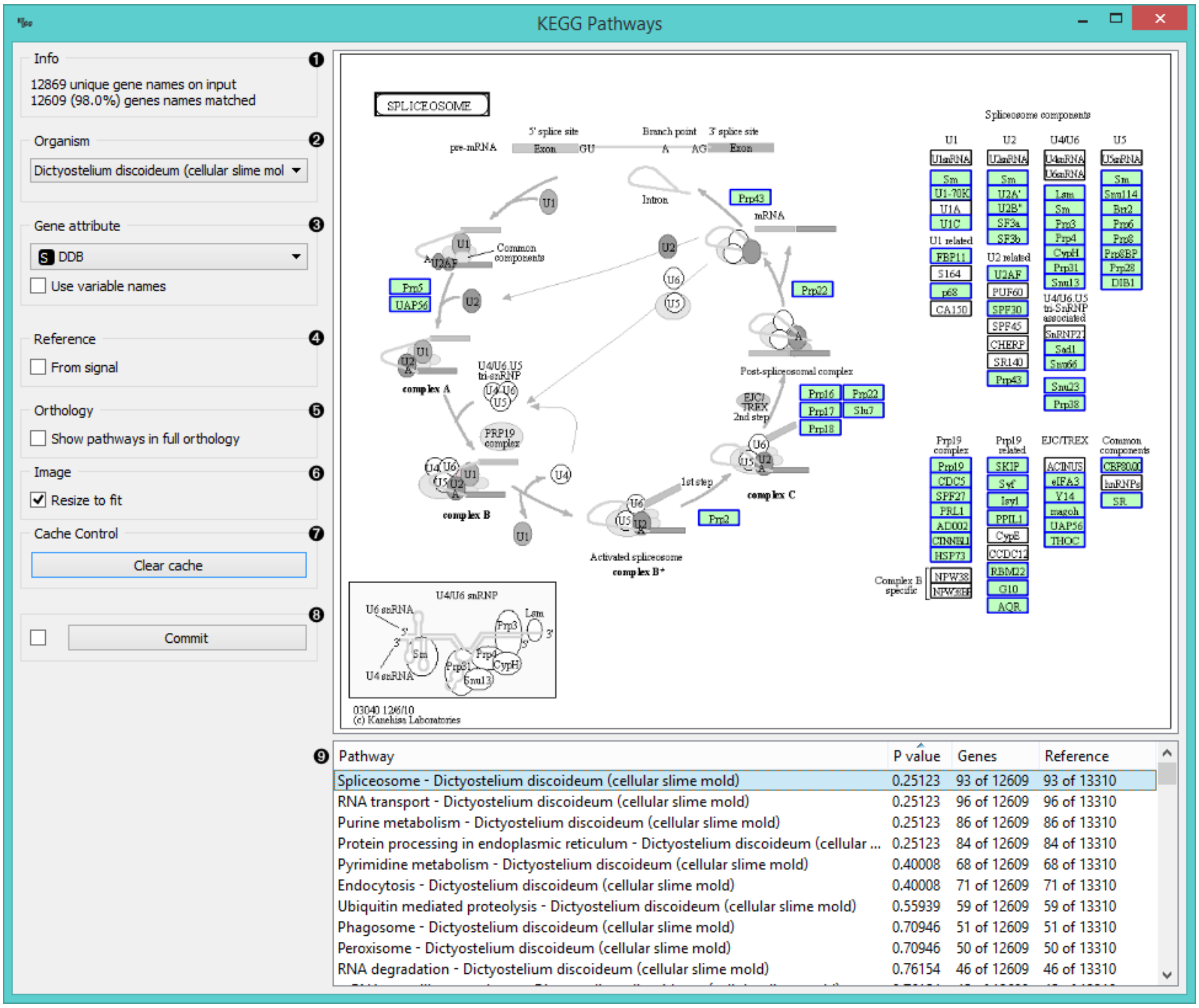
  Data subset.

- **Unselected Data**

  Remaining data.

Description

KEGG Pathways widget displays diagrams of molecular interactions, reactions and relations from the KEGG Pathways Database. It takes data on gene expression as an input, matches the genes to the biological processes and displays a list of corresponding pathways. To explore the pathway, the user can click on any process from the list or arrange them by P-value to get the most relevant processes at the top.


KEGG Pathways

Info
12869 unique gene names on input
12609 (98.0%) genes names matched

Organism: Dictyostelium discoideum (cellular slime mol

Gene attribute: DDB
☐ Use variable names

Reference
☐ From signal

Orthology
☐ Show pathways in full orthology

Image
☑ Resize to fit

Cache Control
Clear cache

☐ Commit

Pathway	P value	Genes	Reference
Spliceosome - Dictyostelium discoideum (cellular slime mold)	0.25123	93 of 12609	93 of 13310
RNA transport - Dictyostelium discoideum (cellular slime mold)	0.25123	96 of 12609	96 of 13310
Purine metabolism - Dictyostelium discoideum (cellular slime mold)	0.25123	86 of 12609	86 of 13310
Protein processing in endoplasmic reticulum - Dictyostelium discoideum (cellular ...	0.25123	84 of 12609	84 of 13310
Pyrimidine metabolism - Dictyostelium discoideum (cellular slime mold)	0.40008	68 of 12609	68 of 13310
Endocytosis - Dictyostelium discoideum (cellular slime mold)	0.40008	71 of 12609	71 of 13310
Ubiquitin mediated proteolysis - Dictyostelium discoideum (cellular slime mold)	0.55939	59 of 12609	59 of 13310
Phagosome - Dictyostelium discoideum (cellular slime mold)	0.70946	51 of 12609	51 of 13310
Peroxisome - Dictyostelium discoideum (cellular slime mold)	0.70946	50 of 12609	50 of 13310
RNA degradation - Dictyostelium discoideum (cellular slime mold)	0.76154	46 of 12609	46 of 13310

1. Information on the input and the ratio of matched genes.
2. Select the organism for term analysis. The widget automatically selects the organism from the input data.
3. Set the attribute to use for gene names. If gene names are your attribute names, tick *Use variable names*.
4. If you have a separate reference set in the input, tick *From signal* to use these data as reference.
5. To have pathways listed and displayed by vertical descent, tick *Show pathways in full orthology*.
6. To fit the image to screen, tick *Resize to fit*. Untick the box if you wish to explore the pathways.
7. To clear all locally cached KEGG data, press *Clear cache*.
8. When *Auto commit is on*, the widget will automatically apply the changes. Alternatively press *Commit*.
9. A list of pathways either as processes or in full orthology. Click on the process to display the pathway. You can sort the data by P-value to get the most relevant results at the top.

Example

MA Plot

Visualization of intensity-dependent ratios of raw microarray data.

Signals

Inputs:

- **Expression Array**

  DNA microarray.

Outputs:

- **Normalized Expression Array**

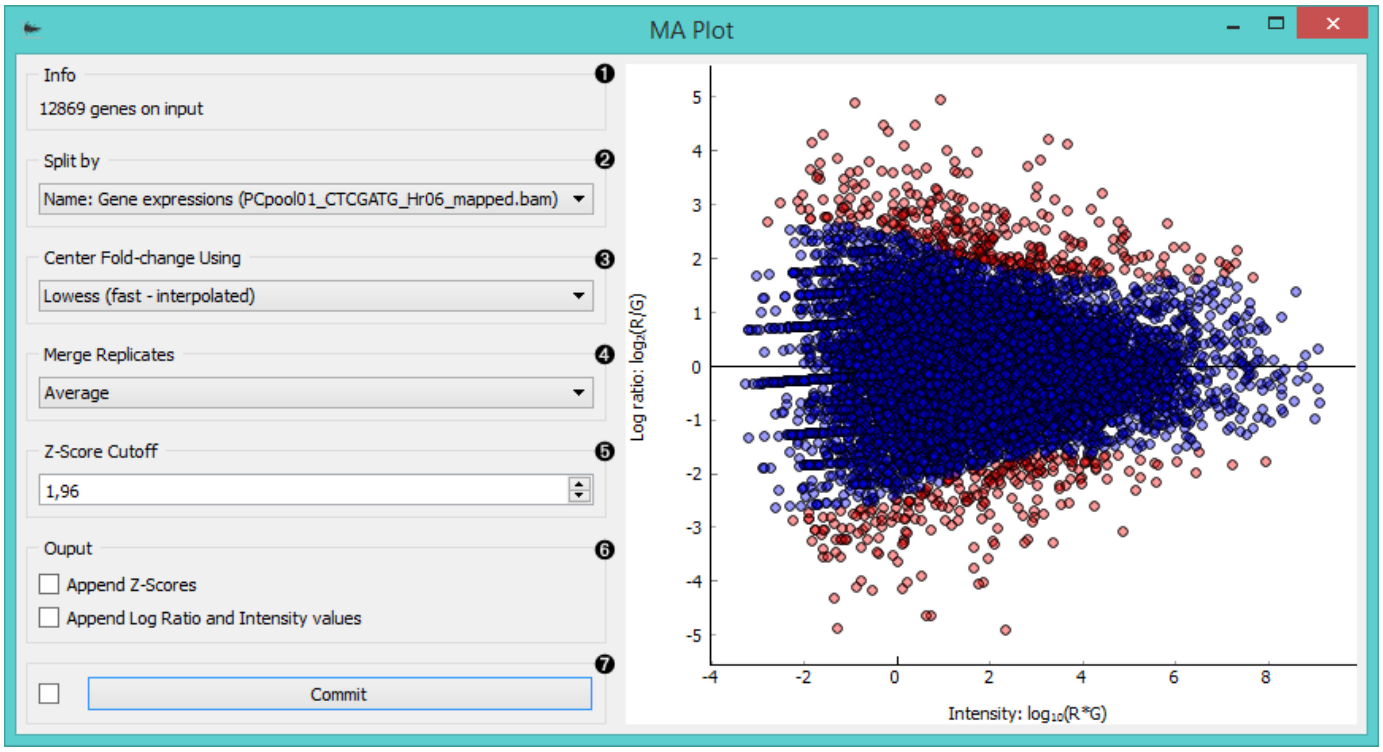
  Lowess-normalized microarray.

- **Filtered Expression Array**

  Selected instances (in the Z-score cutoff).

Description

MA Plot is a graphical method for visualizing intensity-dependent ratio of raw mircoarray data. The A represents the average log intensity of the gene expression (x-axis in the plot), while M stands for the binary log of intensity ratio (y-axis). The widget outputs either normalized data (Lowess normalization method) or instances above the Z-score cut-off line (instances with meaningful fold changes).


MA Plot

Info
12869 genes on input

Split by
Name: Gene expressions (PCpool01_CTCGATG_Hr06_mapped.bam)

Center Fold-change Using
Lowess (fast - interpolated)

Merge Replicates
Average

Z-Score Cutoff
1,96

Ouput
☐ Append Z-Scores
☐ Append Log Ratio and Intensity values

☐ Commit

1. Information on the input data.
2. Select the attribute to split the plot by.
3. Center the plot using:

    - **average**
    - **Lowess (fast-interpolated)** normalization method
    - **Lowess** normalization method

4. Merge replicated by:
    - **average**
    - **median**
    - **geometric mean**

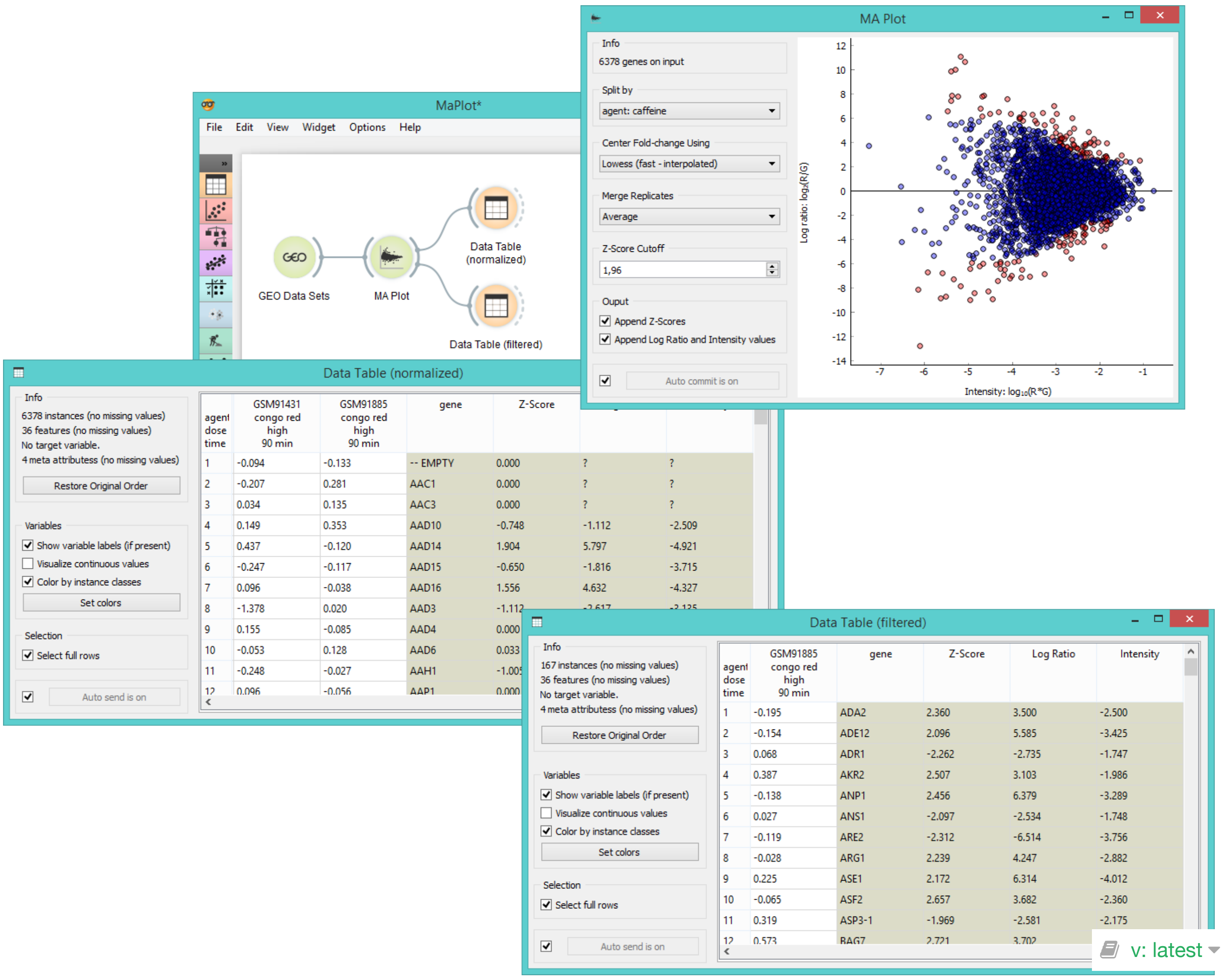
5. Set the **Z-score cutoff** threshold. Z-score is your confidence interval and it is set to 95% by default. If the widget is set to output *filtered expression array*, instances above the Z-score threshold will be in the output (red dots in the plot).
6. Ticking the *Append Z-scores* will add an additional meta attribute with Z-scores to your output data. Ticking the *Append Log ratio and Intensity values* will add two additional meta attributes with M and A values to your output data.
7. If *Auto commit is on*, the widget will automatically apply changes to the output. Alternatively click *Commit*.

Example

MA Plot is a great widget for data visualization and selection. First we select *Caffeine effect: time course and dose response* data from the **GEO Data Sets** widget and feed it to **MA Plot**. In the plot we see intensity ratios for a selected experiment variable.

We often need to normalize the experiment data to avoid systematic biases, thus we select *Lowess (fast-interpolated)* in the *Center Fold-change* box. By ticking both boxes in the *Output* subsection, we get three new meta attributes appended - Z-score, Log ratio and Intensity. We see these new attributes and normalized instances in the **Data Table** (normalized).

Another possible output for the MA plot widget is *Filtered expression array*, which will give us instances above the Z-score cutoff threshold (red dots in the plot). We observe these instances the **Data Table** (filtered).



MA Plot

Info
6378 genes on input

Split by
agent: caffeine

Center Fold-change Using
Lowess (fast - interpolated)

Merge Replicates
Average

Z-Score Cutoff
1,96

Ouput
☑ Append Z-Scores
☑ Append Log Ratio and Intensity values

☑ Auto commit is on

Data Table (normalized)

Info
6378 instances (no missing values)
36 features (no missing values)
No target variable.
4 meta attributess (no missing values)

Restore Original Order

Variables
☑ Show variable labels (if present)
☐ Visualize continuous values
☑ Color by instance classes
Set colors

Selection
☑ Select full rows

☑ Auto send is on

agent dose time	GSM91431 congo red high 90 min	GSM91885 congo red high 90 min	gene	Z-Score		
1	-0.094	-0.133	-- EMPTY	0.000	?	?
2	-0.207	0.281	AAC1	0.000	?	?
3	0.034	0.135	AAC3	0.000	?	?
4	0.149	0.353	AAD10	-0.748	-1.112	-2.509
5	0.437	-0.120	AAD14	1.904	5.797	-4.921
6	-0.247	-0.117	AAD15	-0.650	-1.816	-3.715
7	0.096	-0.038	AAD16	1.556	4.632	-4.327
8	-1.378	0.020	AAD3	-1.112	-2.617	-3.135
9	0.155	-0.085	AAD4	0.000		
10	-0.053	0.128	AAD6	0.033		
11	-0.248	-0.027	AAH1	-1.005		
12	0.096	-0.056	AAP1	0.000		

Data Table (filtered)

Info
167 instances (no missing values)
36 features (no missing values)
No target variable.
4 meta attributess (no missing values)

Restore Original Order

Variables
☑ Show variable labels (if present)
☐ Visualize continuous values
☑ Color by instance classes
Set colors

Selection
☑ Select full rows

☑ Auto send is on

agent dose time	GSM91885 congo red high 90 min	gene	Z-Score	Log Ratio	Intensity
1	-0.195	ADA2	2.360	3.500	-2.500
2	-0.154	ADE12	2.096	5.585	-3.425
3	0.068	ADR1	-2.262	-2.735	-1.747
4	0.387	AKR2	2.507	3.103	-1.986
5	-0.138	ANP1	2.456	6.379	-3.289
6	0.027	ANS1	-2.097	-2.534	-1.748
7	-0.119	ARE2	-2.312	-6.514	-3.756
8	-0.028	ARG1	2.239	4.247	-2.882
9	0.225	ASE1	2.172	6.314	-4.012
10	-0.065	ASF2	2.657	3.682	-2.360
11	0.319	ASP3-1	-1.969	-2.581	-2.175
12	0.573	BAG7	2.721	3.702	

v: latest

PIPAx

Gives access to **PIPA** databases.

Signals

Inputs:

- (None)

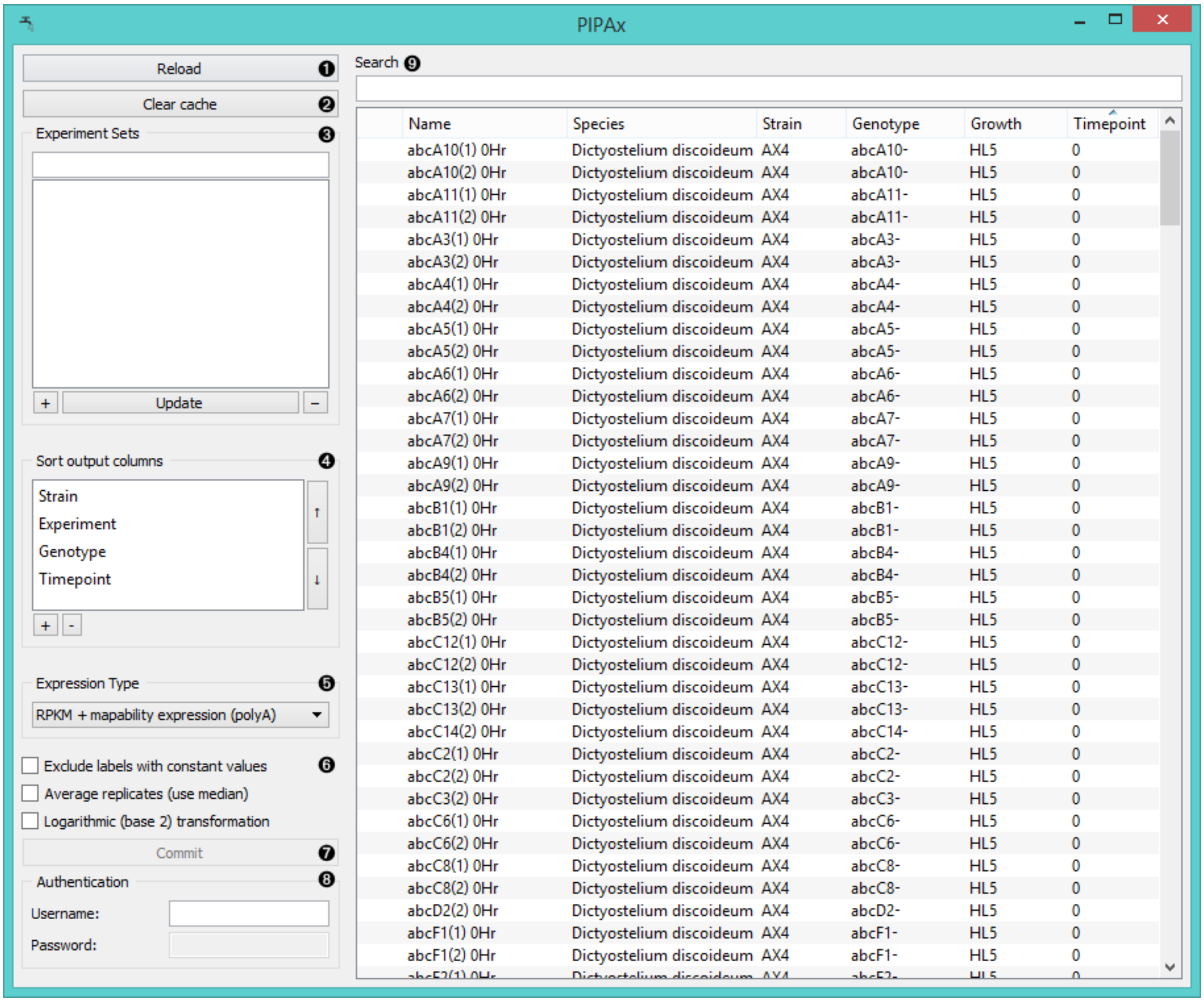
Outputs:

- **Data**

  Selected experiments. Each annotated column contains results of a single experiment or, if the corresponding option is chosen, the average of multiple replicates.

Description

PIPAx is a widget for a direct access to **PIPA** database. It is very similar to the **GenExpress** and **GEO Data Sets** widgets as it allows you to download the data from selected experiments.



Sort output columns 4: Strain, Experiment, Genotype, Timepoint

Expression Type 5: RPKM + mapability expression (polyA)

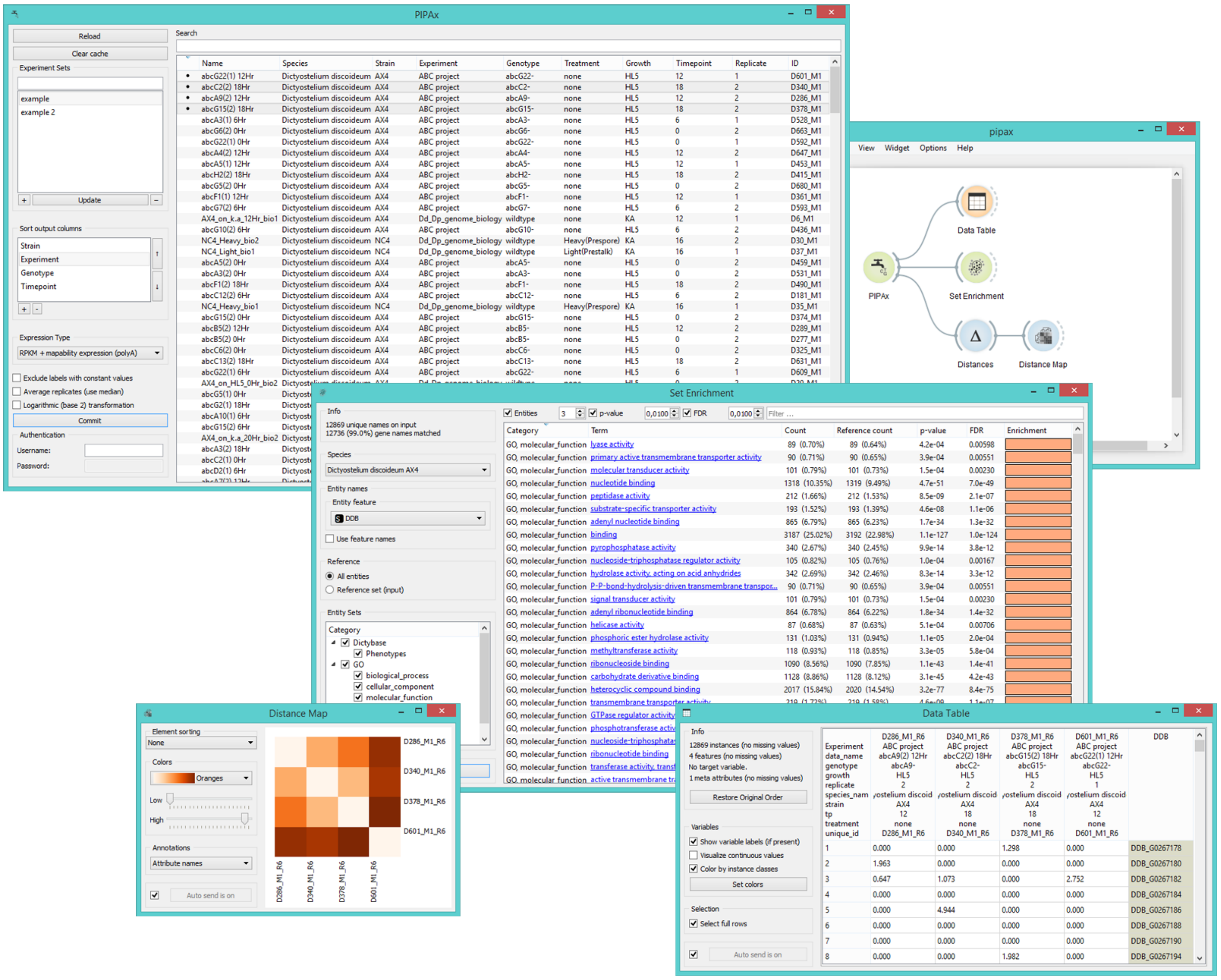
6: ☐ Exclude labels with constant values ☐ Average replicates (use median) ☐ Logarithmic (base 2) transformation

Name	Species	Strain	Genotype	Growth	Timepoint
abcA10(1) 0Hr	Dictyostelium discoideum	AX4	abcA10-	HL5	0
abcA10(2) 0Hr	Dictyostelium discoideum	AX4	abcA10-	HL5	0
abcA11(1) 0Hr	Dictyostelium discoideum	AX4	abcA11-	HL5	0
abcA11(2) 0Hr	Dictyostelium discoideum	AX4	abcA11-	HL5	0
abcA3(1) 0Hr	Dictyostelium discoideum	AX4	abcA3-	HL5	0
abcA3(2) 0Hr	Dictyostelium discoideum	AX4	abcA3-	HL5	0
abcA4(1) 0Hr	Dictyostelium discoideum	AX4	abcA4-	HL5	0
abcA4(2) 0Hr	Dictyostelium discoideum	AX4	abcA4-	HL5	0
abcA5(1) 0Hr	Dictyostelium discoideum	AX4	abcA5-	HL5	0
abcA5(2) 0Hr	Dictyostelium discoideum	AX4	abcA5-	HL5	0
abcA6(1) 0Hr	Dictyostelium discoideum	AX4	abcA6-	HL5	0
abcA6(2) 0Hr	Dictyostelium discoideum	AX4	abcA6-	HL5	0
abcA7(1) 0Hr	Dictyostelium discoideum	AX4	abcA7-	HL5	0
abcA7(2) 0Hr	Dictyostelium discoideum	AX4	abcA7-	HL5	0
abcA9(1) 0Hr	Dictyostelium discoideum	AX4	abcA9-	HL5	0
abcA9(2) 0Hr	Dictyostelium discoideum	AX4	abcA9-	HL5	0
abcB1(1) 0Hr	Dictyostelium discoideum	AX4	abcB1-	HL5	0
abcB1(2) 0Hr	Dictyostelium discoideum	AX4	abcB1-	HL5	0
abcB4(1) 0Hr	Dictyostelium discoideum	AX4	abcB4-	HL5	0
abcB4(2) 0Hr	Dictyostelium discoideum	AX4	abcB4-	HL5	0
abcB5(1) 0Hr	Dictyostelium discoideum	AX4	abcB5-	HL5	0
abcB5(2) 0Hr	Dictyostelium discoideum	AX4	abcB5-	HL5	0
abcC12(1) 0Hr	Dictyostelium discoideum	AX4	abcC12-	HL5	0
abcC12(2) 0Hr	Dictyostelium discoideum	AX4	abcC12-	HL5	0
abcC13(1) 0Hr	Dictyostelium discoideum	AX4	abcC13-	HL5	0
abcC13(2) 0Hr	Dictyostelium discoideum	AX4	abcC13-	HL5	0
abcC14(2) 0Hr	Dictyostelium discoideum	AX4	abcC14-	HL5	0
abcC2(1) 0Hr	Dictyostelium discoideum	AX4	abcC2-	HL5	0
abcC2(2) 0Hr	Dictyostelium discoideum	AX4	abcC2-	HL5	0
abcC3(2) 0Hr	Dictyostelium discoideum	AX4	abcC3-	HL5	0
abcC6(1) 0Hr	Dictyostelium discoideum	AX4	abcC6-	HL5	0
abcC6(2) 0Hr	Dictyostelium discoideum	AX4	abcC6-	HL5	0
abcC8(1) 0Hr	Dictyostelium discoideum	AX4	abcC8-	HL5	0
abcC8(2) 0Hr	Dictyostelium discoideum	AX4	abcC8-	HL5	0
abcD2(2) 0Hr	Dictyostelium discoideum	AX4	abcD2-	HL5	0
abcF1(1) 0Hr	Dictyostelium discoideum	AX4	abcF1-	HL5	0
abcF1(2) 0Hr	Dictyostelium discoideum	AX4	abcF1-	HL5	0

1. Reloads the experiment data.
2. The widget will save (cache) downloaded data, which makes them also available offline. To reset the widget click *Clear cache*.
3. Use *Experiment Sets* to save a selection: select the experiments, click the "+" button and name the set. To add experiments to the set, click on its name, select additional experiments and click *Update*.
   To remove the set click "-".
4. In *Sort output columns* set the attributes by which the output columns are sorted. Add attributes with a "+" button and remove them with "-". Switch the sorting order with arrows on the right.
5. Set the expression type for your output data.
   - **Raw expression** outputs raw experiment data
   - **RPKM expression** outputs data in *reads per kilobase of transcript per million mapped reads*
   - **RPKM expression + mapability expression** uses similar normalization, but divides with gene mapability instead of exon lengths.
     The polyA variants use only polyA (mRNA) mapped hits.
6. **Exclude labels with constant values** removes attribute labels that are the same for all selected experiments from the output data.
   Average replicates (use median) averages identical experiments by using medians as values.
   Logarithmic (base 2) transformation computes the $\log_2$(value+1) for each value.
7. Click *Commit* to output selected experiments.
8. Log in to access private data.
9. Experiments can be filtered with the *Search* box. To select which attributes to display right-click on the header. To select multiple experiments click them while holding the *Control/Command* key.

Example

In the schema below we connected **PIPAx** to **Data Table**, **Set Enrichment**, and **Distance Map** (through **Distances**) widgets.



The **Data Table** widget above contains the output from the **PIPAx** widget. Each column contains gene expressions of a single experiment. The labels are shown in the table header. The **Distance Map** widget shows distances between experiments. The distances are measured with **Distance** widget, which was set to compute *Euclidean* distances.

Quality Control

Computes and plots distances between experiments or replicates.

Signals

Inputs:

- **Data**

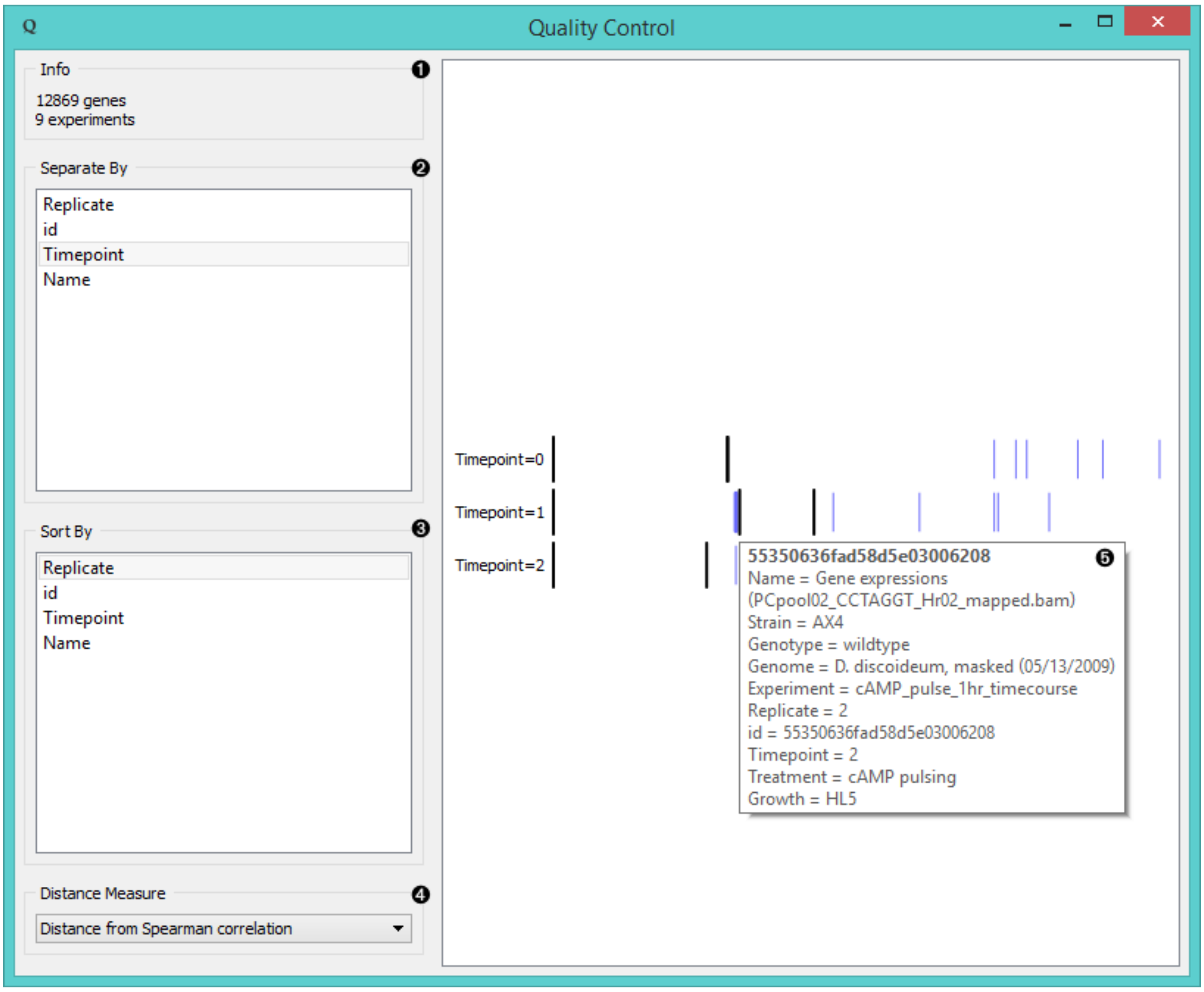
  Data set.

Outputs:

- (None)

Description

Quality Control measures distances between experiments (usually replicates) for a selected label. The widget visualizes distances by selected label. Experiments that lie the farthest from the initial black line should be inspected for anomalies.



55350636fad58d5e03006208
Name = Gene expressions (PCpool02_CCTAGGT_Hr02_mapped.bam)
Strain = AX4
Genotype = wildtype
Genome = D. discoideum, masked (05/13/2009)
Experiment = cAMP_pulse_1hr_timecourse
Replicate = 2
id = 55350636fad58d5e03006208
Timepoint = 2
Treatment = cAMP pulsing
Growth = HL5

1. Information on the input.

2. Separate experiments by label.
3. Sort experiments by label.
4. Compute distances by:
   - **Pearson correlation**
   - **Euclidean distances**
   - **Spearman correlation**
5. Hover over the vertical line to display the information on a chosen instance. Click on the black line to change the reference to that instance.

Example

Quality Control widget gives us feedback on the quality of our data and can be connected to any widget with data output. In the example above (see the image under *Description*) we fed 9 experiments of *Cyclic AMP pulsing* of *Dictyostelium discoideum* from **GenExpress** widget into **Quality Control** and separated them by timepoint label. We found replicate 2 from tp 2 among the tp 1 data, meaning we should inspect these data further.

Select Genes

Manual selection of gene subset.

Signals

Inputs:

- **Data**

  Data set.

- **Gene Subset**

  A subset of genes to be used for gene selection (optional).

Outputs:

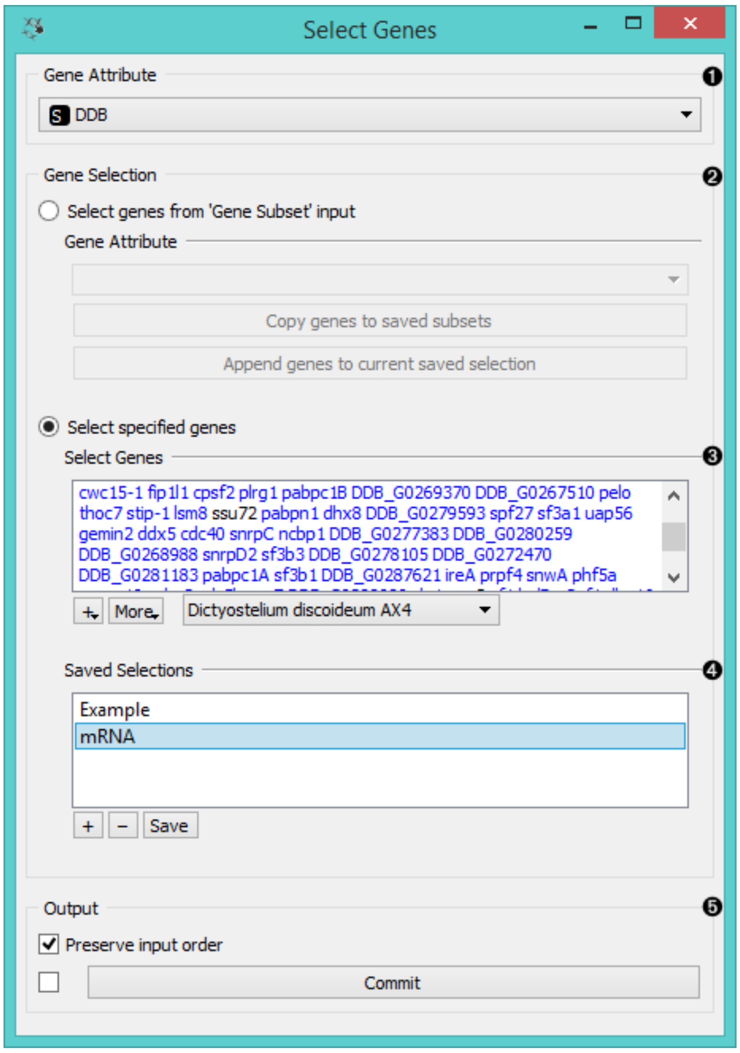
- **Selected Data**

  A subset of genes selected in the widget.

Description

Select Genes widget is used to manually create the gene subset. There are three ways to select genes:

- Manual gene selection (written input). The widget supports autocompletion for gene names.
- Selecting genes from gene sets in the “+” option.
- Selecting genes from a separate input (input can be adjusted in the widget).



Gene Attribute: DDB

Gene Selection: ( ) Select genes from 'Gene Subset' input [✓ Select specified genes]

Select Genes:

cwc15-1 fip1l1 cpsf2 plrg1 pabpc1B DDB_G0269370 DDB_G0267510 pelo
thoc7 stip-1 lsm8 ssu72 pabpn1 dhx8 DDB_G0279593 spf27 sf3a1 uap56
gemin2 ddx5 cdc40 snrpC ncbp1 DDB_G0277383 DDB_G0280259
DDB_G0268988 snrpD2 sf3b3 DDB_G0278105 DDB_G0272470
DDB_G0281183 pabpc1A sf3b1 DDB_G0287621 ireA prpf4 snwA phf5a

Dictyostelium discoideum AX4

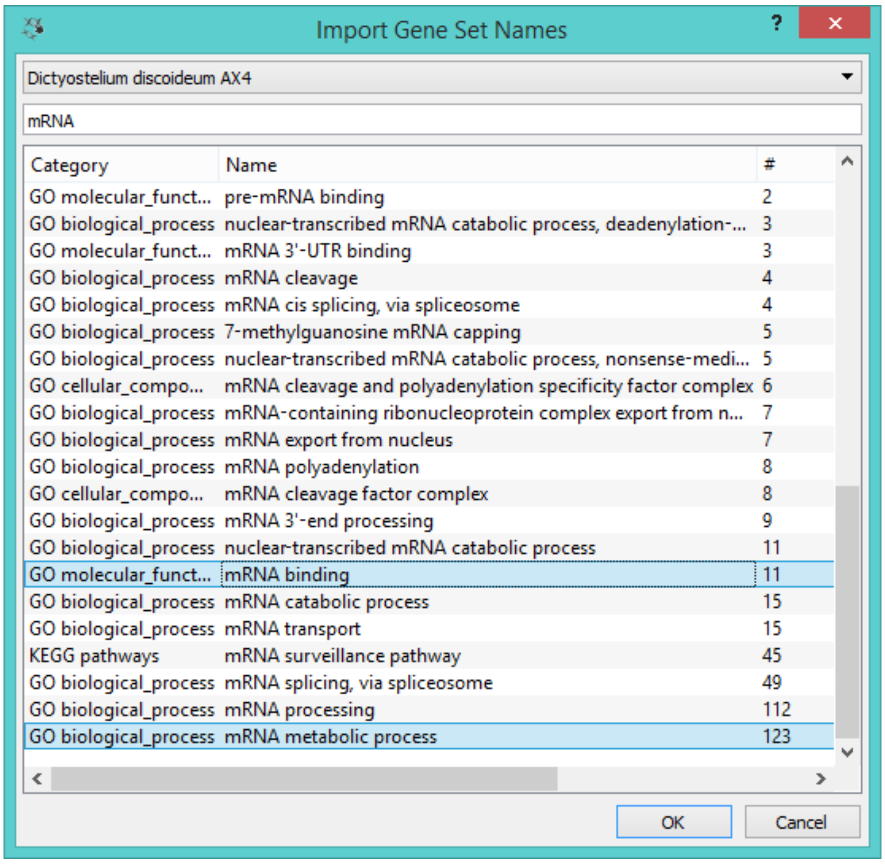
Saved Selections:

Example
mRNA

☑ Preserve input order

1. Select *Gene Attribute* if there is more than one column with gene names.
2. Specify how you want to select your genes: *Select Genes from 'Gene Subset' input* adds genes from the separate input to selected genes. To create a new saved selection, click *Copy genes to saved subsets*. The genes will be listed in *Select Genes* text area below. To add these genes to an existing selection, click *Append genes to current saved selection*.
3. In *Select specified genes* you can type the gene name and the widget will automatically suggest corresponding genes. Genes that match the genes in the input will be colored blue, while the unmatched will remain black.
4. The "+" button has a drop-down menu with two options.
   - *Import names from gene sets...* gives a list of gene sets and copies genes from selected sets into the list.
   - *Import names from text files...* imports gene names from the file.
5. *More* has two settings: *Complete on gene symbol names* (for easier gene selection) and *Translate all names to official symbol names* (for uniformity).
6. Set the organism to select the genes from (organism from the input data is chosen as default).
7. *Saved Selections* saves the most frequently used genes. "+" adds a new selection, "-" removes the existing one, and *Save* saves the current list. Double-click the selection to rename it.
8. *Output* for this widget is a data subset. If you wish to preserve the order of instances from your input data, tick the *Preserve input order* box. If *Auto commit is on*, all changes will be communicated automatically. Alternatively press *Commit*.

Below is a screenshot of the *Import Gene Set Names* option.


Import Gene Set Names

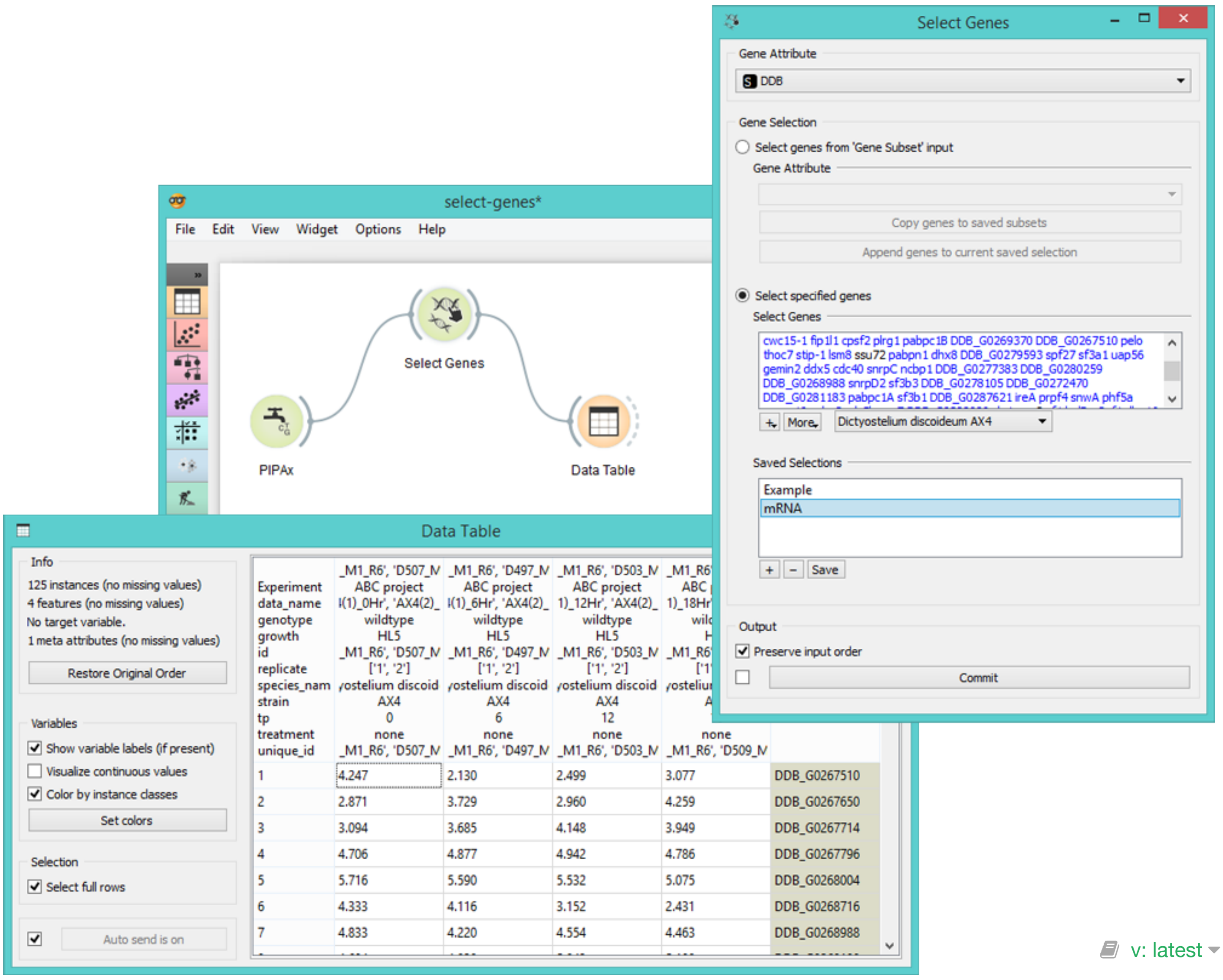
Dictyostelium discoideum AX4

mRNA

Category	Name	#
GO molecular_funct...	pre-mRNA binding	2
GO biological_process	nuclear-transcribed mRNA catabolic process, deadenylation-...	3
GO molecular_funct...	mRNA 3'-UTR binding	3
GO biological_process	mRNA cleavage	4
GO biological_process	mRNA cis splicing, via spliceosome	4
GO biological_process	7-methylguanosine mRNA capping	5
GO biological_process	nuclear-transcribed mRNA catabolic process, nonsense-medi...	5
GO cellular_compo...	mRNA cleavage and polyadenylation specificity factor complex	6
GO biological_process	mRNA-containing ribonucleoprotein complex export from n...	7
GO biological_process	mRNA export from nucleus	7
GO biological_process	mRNA polyadenylation	8
GO cellular_compo...	mRNA cleavage factor complex	8
GO biological_process	mRNA 3'-end processing	9
GO biological_process	nuclear-transcribed mRNA catabolic process	11
GO molecular_funct...	mRNA binding	11
GO biological_process	mRNA catabolic process	15
GO biological_process	mRNA transport	15
KEGG pathways	mRNA surveillance pathway	45
GO biological_process	mRNA splicing, via spliceosome	49
GO biological_process	mRNA processing	112
GO biological_process	mRNA metabolic process	123

Example

Below is a very simple workflow for this widget. We selected *AX4 Dictyostelium discoideum* data from different time points and two different replicates from **PIPAx** widget. In **Select Genes** we used the *Import names from gene sets...* option and selected two mRNA processes that gave us a list of genes you can see in the *Select Genes* box. Then we fed these data into the **Data Table**. There are 125 genes in the entire *AX4 Dictyostelium discoideum* data that are present in the selected mRNA processes.



Select Genes

Gene Attribute: DDB

Gene Selection

( ) Select genes from 'Gene Subset' input

Gene Attribute

Copy genes to saved subsets

Append genes to current saved selection

(●) Select specified genes

Select Genes

cwc15-1 fip1l1 cpsf2 plrg1 pabpc1B DDB_G0269370 DDB_G0267510 pelo thoc7 stip-1 lsm8 ssu72 pabpn1 dhx8 DDB_G0279593 spf27 sf3a1 uap56 gemin2 ddx5 cdc40 snrpC ncbp1 DDB_G0277383 DDB_G0280259 DDB_G0268988 snrpD2 sf3b3 DDB_G0278105 DDB_G0272470 DDB_G0281183 pabpc1A sf3b1 DDB_G0287621 ireA prpf4 snwA phf5a

+ More Dictyostelium discoideum AX4

Saved Selections

Example

mRNA

+ − Save

Output

☑ Preserve input order

☐ Commit

Data Table

Info

125 instances (no missing values)
4 features (no missing values)
No target variable.
1 meta attributes (no missing values)

Restore Original Order

Variables

☑ Show variable labels (if present)
☐ Visualize continuous values
☑ Color by instance classes

Set colors

Selection

☑ Select full rows

☑ Auto send is on

	_M1_R6', 'D507_M	_M1_R6', 'D497_M	_M1_R6', 'D503_M	_M1_R6'	
Experiment	ABC project	ABC project	ABC project	ABC	
data_name	l(1)_0Hr', 'AX4(2)_	l(1)_6Hr', 'AX4(2)_	1)_12Hr', 'AX4(2)_	1)_18Hr'	
genotype	wildtype	wildtype	wildtype	wilc	
growth	HL5	HL5	HL5	H	
id	_M1_R6', 'D507_M	_M1_R6', 'D497_M	_M1_R6', 'D503_M	_M1_R6'	
replicate	['1', '2']	['1', '2']	['1', '2']	['1'	
species_nam	yostelium discoid	yostelium discoid	yostelium discoid	yosteliur	
strain	AX4	AX4	AX4	A	
tp	0	6	12		
treatment	none	none	none	none	
unique_id	_M1_R6', 'D507_M	_M1_R6', 'D497_M	_M1_R6', 'D503_M	_M1_R6', 'D509_M	
1	4.247	2.130	2.499	3.077	DDB_G0267510
2	2.871	3.729	2.960	4.259	DDB_G0267650
3	3.094	3.685	4.148	3.949	DDB_G0267714
4	4.706	4.877	4.942	4.786	DDB_G0267796
5	5.716	5.590	5.532	5.075	DDB_G0268004
6	4.333	4.116	3.152	2.431	DDB_G0268716
7	4.833	4.220	4.554	4.463	DDB_G0268988

v: latest

Set Enrichment

Determines statistically significant differences in expression levels for biological processes.

Signals

Inputs:

- **Data**

  Data set.

- **Reference**

  Data with genes for the reference set (optional).

Outputs:

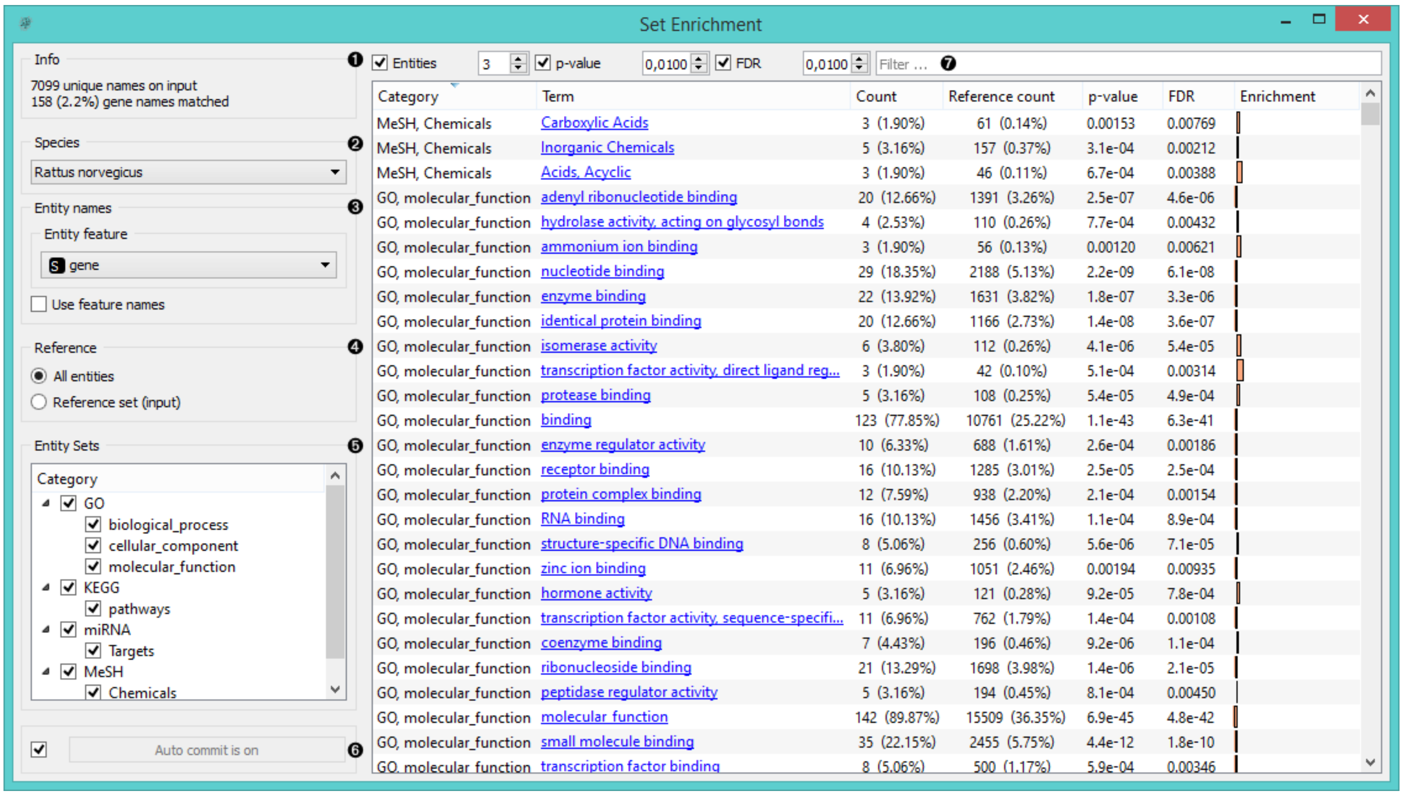
- **Selected data**

  Data subset.

Description

The widget shows a ranked list of terms with p-values, FDR and enrichment. **Set Enrichment** is a great tool for finding biological processes that are over-represented in a particular gene or chemical set.

Sets from (GO, KEGG, miRNA and MeSH) come with the Orange installation.


Set Enrichment

Info ❶
7099 unique names on input
158 (2.2%) gene names matched

Species ❷
Rattus norvegicus

Entity names ❸
Entity feature: gene
☐ Use feature names

Reference ❹
(●) All entities
( ) Reference set (input)

Entity Sets ❺
Category
- ☑ GO
  - ☑ biological_process
  - ☑ cellular_component
  - ☑ molecular_function
- ☑ KEGG
  - ☑ pathways
- ☑ miRNA
  - ☑ Targets
- ☑ MeSH
  - ☑ Chemicals

☑ Auto commit is on ❻

☑ Entities 3 ☑ p-value 0,0100 ☑ FDR 0,0100 Filter ... ❼

Category	Term	Count	Reference count	p-value	FDR	Enrichment
MeSH, Chemicals	Carboxylic Acids	3 (1.90%)	61 (0.14%)	0.00153	0.00769	
MeSH, Chemicals	Inorganic Chemicals	5 (3.16%)	157 (0.37%)	3.1e-04	0.00212	
MeSH, Chemicals	Acids, Acyclic	3 (1.90%)	46 (0.11%)	6.7e-04	0.00388	
GO, molecular_function	adenyl ribonucleotide binding	20 (12.66%)	1391 (3.26%)	2.5e-07	4.6e-06	
GO, molecular_function	hydrolase activity, acting on glycosyl bonds	4 (2.53%)	110 (0.26%)	7.7e-04	0.00432	
GO, molecular_function	ammonium ion binding	3 (1.90%)	56 (0.13%)	0.00120	0.00621	
GO, molecular_function	nucleotide binding	29 (18.35%)	2188 (5.13%)	2.2e-09	6.1e-08	
GO, molecular_function	enzyme binding	22 (13.92%)	1631 (3.82%)	1.8e-07	3.3e-06	
GO, molecular_function	identical protein binding	20 (12.66%)	1166 (2.73%)	1.4e-08	3.6e-07	
GO, molecular_function	isomerase activity	6 (3.80%)	112 (0.26%)	4.1e-06	5.4e-05	
GO, molecular_function	transcription factor activity, direct ligand reg...	3 (1.90%)	42 (0.10%)	5.1e-04	0.00314	
GO, molecular_function	protease binding	5 (3.16%)	108 (0.25%)	5.4e-05	4.9e-04	
GO, molecular_function	binding	123 (77.85%)	10761 (25.22%)	1.1e-43	6.3e-41	
GO, molecular_function	enzyme regulator activity	10 (6.33%)	688 (1.61%)	2.6e-04	0.00186	
GO, molecular_function	receptor binding	16 (10.13%)	1285 (3.01%)	2.5e-05	2.5e-04	
GO, molecular_function	protein complex binding	12 (7.59%)	938 (2.20%)	2.1e-04	0.00154	
GO, molecular_function	RNA binding	16 (10.13%)	1456 (3.41%)	1.1e-04	8.9e-04	
GO, molecular_function	structure-specific DNA binding	8 (5.06%)	256 (0.60%)	5.6e-06	7.1e-05	
GO, molecular_function	zinc ion binding	11 (6.96%)	1051 (2.46%)	0.00194	0.00935	
GO, molecular_function	hormone activity	5 (3.16%)	121 (0.28%)	9.2e-05	7.8e-04	
GO, molecular_function	transcription factor activity, sequence-specifi...	11 (6.96%)	762 (1.79%)	1.4e-04	0.00108	
GO, molecular_function	coenzyme binding	7 (4.43%)	196 (0.46%)	9.2e-06	1.1e-04	
GO, molecular_function	ribonucleoside binding	21 (13.29%)	1698 (3.98%)	1.4e-06	2.1e-05	
GO, molecular_function	peptidase regulator activity	5 (3.16%)	194 (0.45%)	8.1e-04	0.00450	
GO, molecular_function	molecular_function	142 (89.87%)	15509 (36.35%)	6.9e-45	4.8e-42	
GO, molecular_function	small molecule binding	35 (22.15%)	2455 (5.75%)	4.4e-12	1.8e-10	
GO, molecular_function	transcription factor binding	8 (5.06%)	500 (1.17%)	5.9e-04	0.00346	

1. Information on the input data set and the ratio of genes that were found in the databases.
2. Select the species.
3. *Entity names* define the features in the input data that you wish to use for term analysis. Tick *Use feature names*

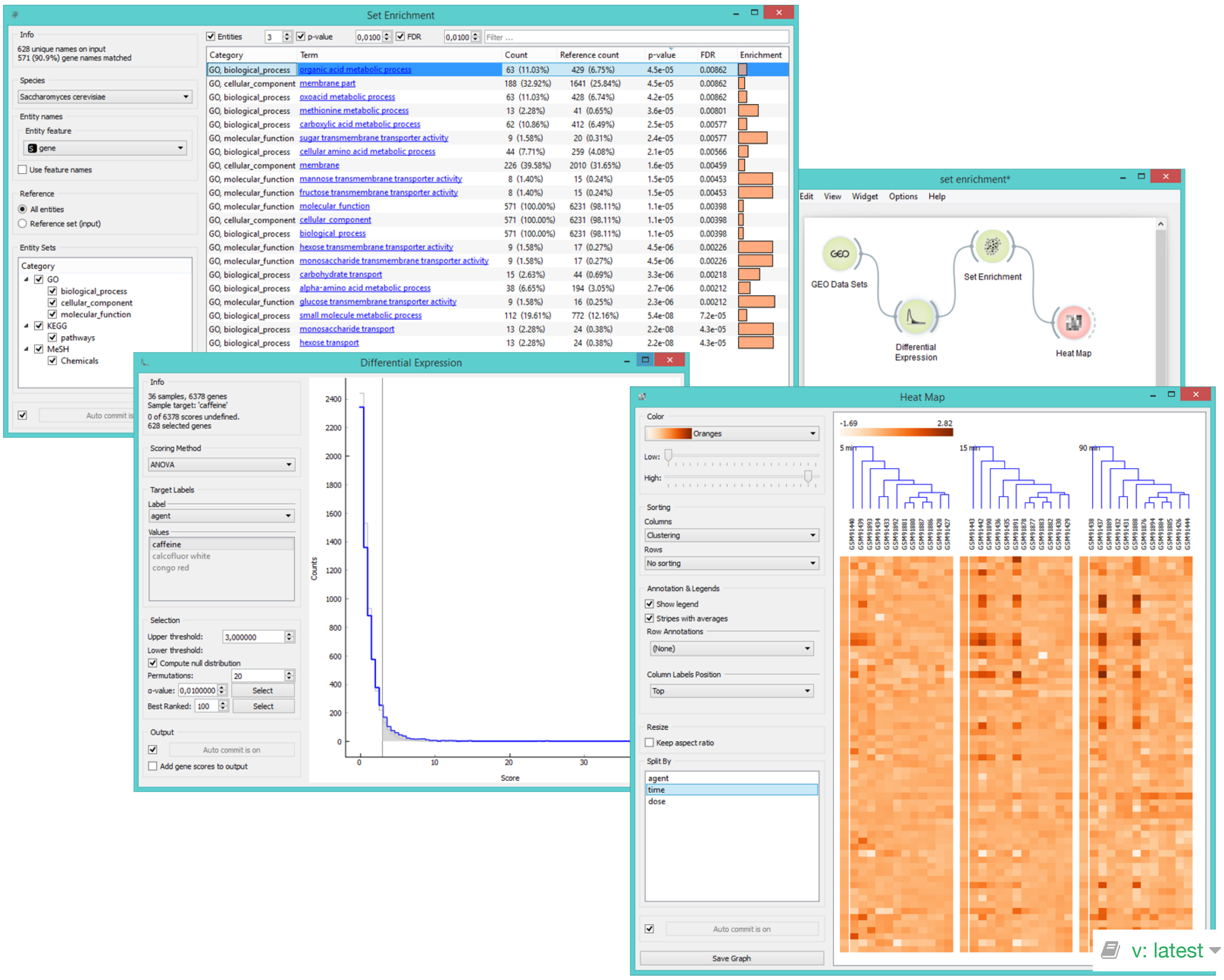
if your genes or chemicals are used as attribute names rather than as meta attributes.

4. Select the reference data. You can either have entities (usually genes from the organism - *All Entities*) as a reference or a reference set from the input.
5. Select which *Entity sets* you wish to have displayed in the list.
6. When *Auto commit is on*, the widget will automatically apply the changes. Alternatively press *Commit*.
7. Filter the list by:

    - the minimum number of **entities** included in each term
    - the minimum threshold for **p-value**
    - the maximum threshold for **false discovery rate**
    - a search word

Example

In the example below we have decided to analyse gene expression levels from *Caffeine effect: time course and dose response* data set. We used the ANOVA scoring in the **Differential Expression** widget to select the most interesting genes. Then we fed those 628 genes to **Set Enrichment** for additional analysis of the most valuable terms. We sorted the data by FDR values and selected the top-scoring term. **Heat Map** widget provides a nice visualization of the data.



Set Enrichment

Info
628 unique names on input
571 (90.9%) gene names matched

Species: Saccharomyces cerevisiae

Entity names
Entity feature: gene
☐ Use feature names

Reference
● All entities
○ Reference set (input)

Entity Sets
Category
☑ GO: ☑ biological_process, ☑ cellular_component, ☑ molecular_function
☑ KEGG: ☑ pathways
☑ MeSH: ☑ Chemicals

☑ Entities 3 ☑ p-value 0,0100 ☑ FDR 0,0100 Filter ...

Category	Term	Count	Reference count	p-value	FDR	Enrichment
GO, biological_process	organic acid metabolic process	63 (11.03%)	429 (6.75%)	4.5e-05	0.00862	
GO, cellular_component	membrane part	188 (32.92%)	1641 (25.84%)	4.5e-05	0.00862	
GO, biological_process	oxoacid metabolic process	63 (11.03%)	428 (6.74%)	4.2e-05	0.00862	
GO, biological_process	methionine metabolic process	13 (2.28%)	41 (0.65%)	3.6e-05	0.00801	
GO, biological_process	carboxylic acid metabolic process	62 (10.86%)	412 (6.49%)	2.5e-05	0.00577	
GO, molecular_function	sugar transmembrane transporter activity	9 (1.58%)	20 (0.31%)	2.4e-05	0.00577	
GO, biological_process	cellular amino acid metabolic process	44 (7.71%)	259 (4.08%)	2.1e-05	0.00566	
GO, cellular_component	membrane	226 (39.58%)	2010 (31.65%)	1.6e-05	0.00459	
GO, molecular_function	mannose transmembrane transporter activity	8 (1.40%)	15 (0.24%)	1.5e-05	0.00453	
GO, molecular_function	fructose transmembrane transporter activity	8 (1.40%)	15 (0.24%)	1.5e-05	0.00453	
GO, molecular_function	molecular_function	571 (100.00%)	6231 (98.11%)	1.1e-05	0.00398	
GO, cellular_component	cellular_component	571 (100.00%)	6231 (98.11%)	1.1e-05	0.00398	
GO, biological_process	biological_process	571 (100.00%)	6231 (98.11%)	1.1e-05	0.00398	
GO, molecular_function	hexose transmembrane transporter activity	9 (1.58%)	17 (0.27%)	4.5e-06	0.00226	
GO, molecular_function	monosaccharide transmembrane transporter activity	9 (1.58%)	17 (0.27%)	4.5e-06	0.00226	
GO, biological_process	carbohydrate transport	15 (2.63%)	44 (0.69%)	3.3e-06	0.00218	
GO, biological_process	alpha-amino acid metabolic process	38 (6.65%)	194 (3.05%)	2.7e-06	0.00212	
GO, molecular_function	glucose transmembrane transporter activity	9 (1.58%)	16 (0.25%)	2.3e-06	0.00212	
GO, biological_process	small molecule metabolic process	112 (19.61%)	772 (12.16%)	5.4e-08	7.2e-05	
GO, biological_process	monosaccharide transport	13 (2.28%)	24 (0.38%)	2.2e-08	4.3e-05	
GO, biological_process	hexose transport	13 (2.28%)	24 (0.38%)	2.2e-08	4.3e-05	

Differential Expression

Info
36 samples, 6378 genes
Sample target: 'caffeine'
0 of 6378 scores undefined.
628 selected genes

Scoring Method: ANOVA

Target Labels
Label: agent
Values: caffeine, calcofluor white, congo red

Selection
Upper threshold: 3,000000
Lower threshold:
☑ Compute null distribution
Permutations: 20
α-value: 0,0100000 Select
Best Ranked: 100 Select

Output
☑ Auto commit is on
☐ Add gene scores to output

Heat Map

Color: Oranges
Low:
High:

Sorting
Columns: Clustering
Rows: No sorting

Annotation & Legends
☑ Show legend
☑ Stripes with averages
Row Annotations: (None)

Column Labels Position: Top

Resize
☐ Keep aspect ratio

Split By: agent, time, dose

☑ Auto commit is on
Save Graph

Volcano Plot

Plots significance versus fold-change for gene expression rates.

Signals

Inputs:

- **Data**

  Input data set.

Outputs:

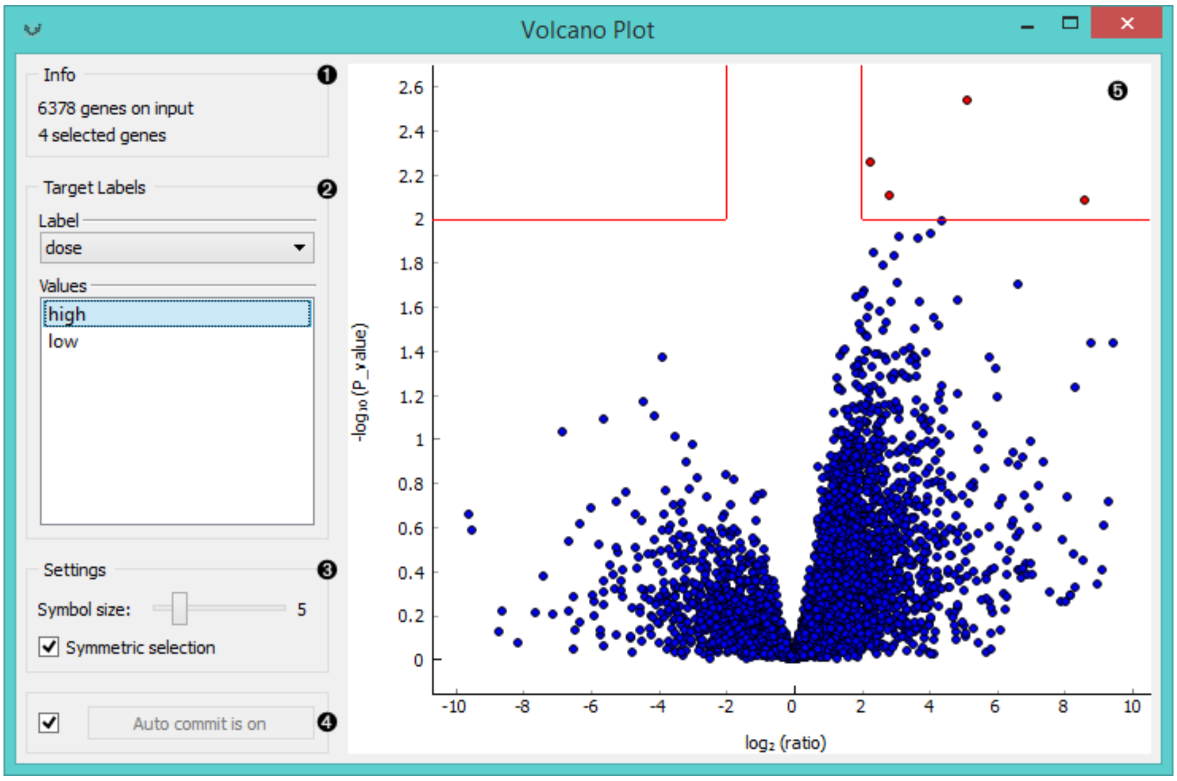
- **Selected data**

  Data subset.

Description

Volcano plot is a graphical method for visualizing changes in replicate data. The widget plots a binary logarithm of fold-change on the x-axis versus statistical significance (negative base 10 logarithm of p-value) on the y-axis.

Volcano Plot is useful for a quick visual identification of statistically significant data (genes). Genes that are highly dysregulated are farther to the left and right, while highly significant fold changes appear higher on the plot. A combination of the two are those genes that are statistically significant - the widget selects the top-ranking genes within the top right and left fields by default.



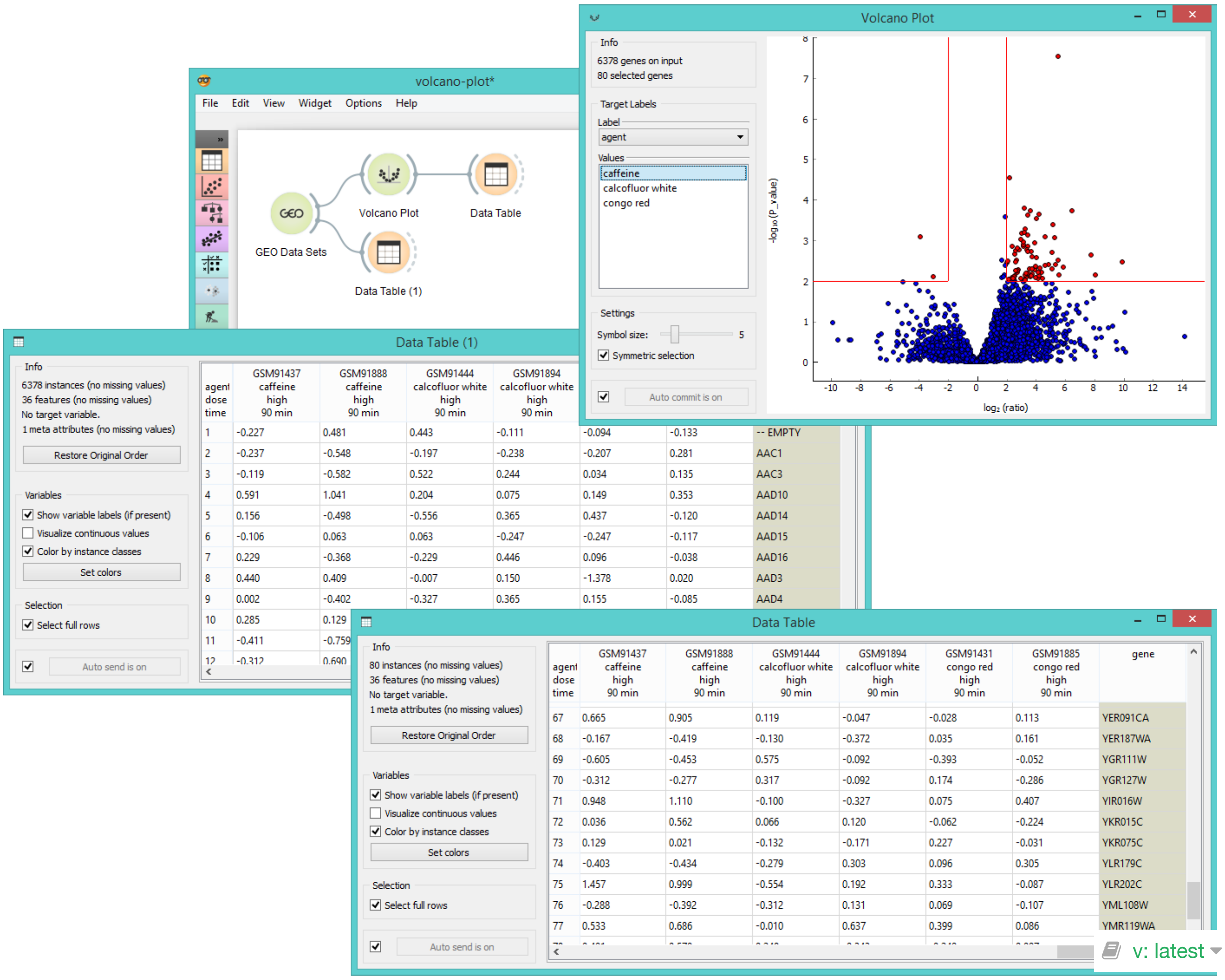
1. Information on the input and output data.
2. Select *Target Labels*. Labels depend on the attributes in the input. In *Values* you can change the sample target (default value is the first value on the list, alphabetically or numerically).
3. Change the *Settings*: adjust the symbol size and turn off symmetrical selection of the output data (the widget se-

lects statistically significant instances by default).

4. If *Auto commit is on* the widget will automatically apply the changes. Alternatively click *Commit*.
5. Visualization of the changes in gene expression. The red lines represent the area with the most statistically significant instances. Symmetrical selection is chosen by default, but you can also manually adjust the area you want in the output.

Example

Below you can see a simple workflow for **Volcano Plot**. We use *Caffeine effect: time course and dose response* data from **GEO Data Sets** widget and visualize them in a **Data Table**. We have 6378 gene in the input, so it is essential to prune the data and analyse only those genes that are statistically significant. **Volcano Plot** helps us do exactly that. Once the desired area is selected in the plot, we output the data and observe them in another **Data Table**. Now we get only 80 instances, which were those genes that had a high normalized fold change under high dose of caffeine and had a low p-value at the same time.



Volcano Plot

Info
6378 genes on input
80 selected genes

Target Labels
Label: agent
Values: caffeine, calcofluor white, congo red

Settings
Symbol size: 5
☑ Symmetric selection
☑ Auto commit is on

Data Table (1)

Info
6378 instances (no missing values)
36 features (no missing values)
No target variable.
1 meta attributes (no missing values)

Restore Original Order

Variables
☑ Show variable labels (if present)
☐ Visualize continuous values
☑ Color by instance classes
Set colors

Selection
☑ Select full rows

☑ Auto send is on

agent dose time	GSM91437 caffeine high 90 min	GSM91888 caffeine high 90 min	GSM91444 calcofluor white high 90 min	GSM91894 calcofluor white high 90 min			
1	-0.227	0.481	0.443	-0.111	-0.094	-0.133	-- EMPTY
2	-0.237	-0.548	-0.197	-0.238	-0.207	0.281	AAC1
3	-0.119	-0.582	0.522	0.244	0.034	0.135	AAC3
4	0.591	1.041	0.204	0.075	0.149	0.353	AAD10
5	0.156	-0.498	-0.556	0.365	0.437	-0.120	AAD14
6	-0.106	0.063	0.063	-0.247	-0.247	-0.117	AAD15
7	0.229	-0.368	-0.229	0.446	0.096	-0.038	AAD16
8	0.440	0.409	-0.007	0.150	-1.378	0.020	AAD3
9	0.002	-0.402	-0.327	0.365	0.155	-0.085	AAD4
10	0.285	0.129					
11	-0.411	-0.759					
12	-0.312	0.690					

Data Table

Info
80 instances (no missing values)
36 features (no missing values)
No target variable.
1 meta attributes (no missing values)

Restore Original Order

Variables
☑ Show variable labels (if present)
☐ Visualize continuous values
☑ Color by instance classes
Set colors

Selection
☑ Select full rows

☑ Auto send is on

agent dose time	GSM91437 caffeine high 90 min	GSM91888 caffeine high 90 min	GSM91444 calcofluor white high 90 min	GSM91894 calcofluor white high 90 min	GSM91431 congo red high 90 min	GSM91885 congo red high 90 min	gene
67	0.665	0.905	0.119	-0.047	-0.028	0.113	YER091CA
68	-0.167	-0.419	-0.130	-0.372	0.035	0.161	YER187WA
69	-0.605	-0.453	0.575	-0.092	-0.393	-0.052	YGR111W
70	-0.312	-0.277	0.317	-0.092	0.174	-0.286	YGR127W
71	0.948	1.110	-0.100	-0.327	0.075	0.407	YIR016W
72	0.036	0.562	0.066	0.120	-0.062	-0.224	YKR015C
73	0.129	0.021	-0.132	-0.171	0.227	-0.031	YKR075C
74	-0.403	-0.434	-0.279	0.303	0.096	0.305	YLR179C
75	1.457	0.999	-0.554	0.192	0.333	-0.087	YLR202C
76	-0.288	-0.392	-0.312	0.131	0.069	-0.107	YML108W
77	0.533	0.686	-0.010	0.637	0.399	0.086	YMR119WA

Associate

Association Rules

Frequent Itemsets

Association Rules

Induction of association rules.

Signals

Inputs

- **Data**

  Data set

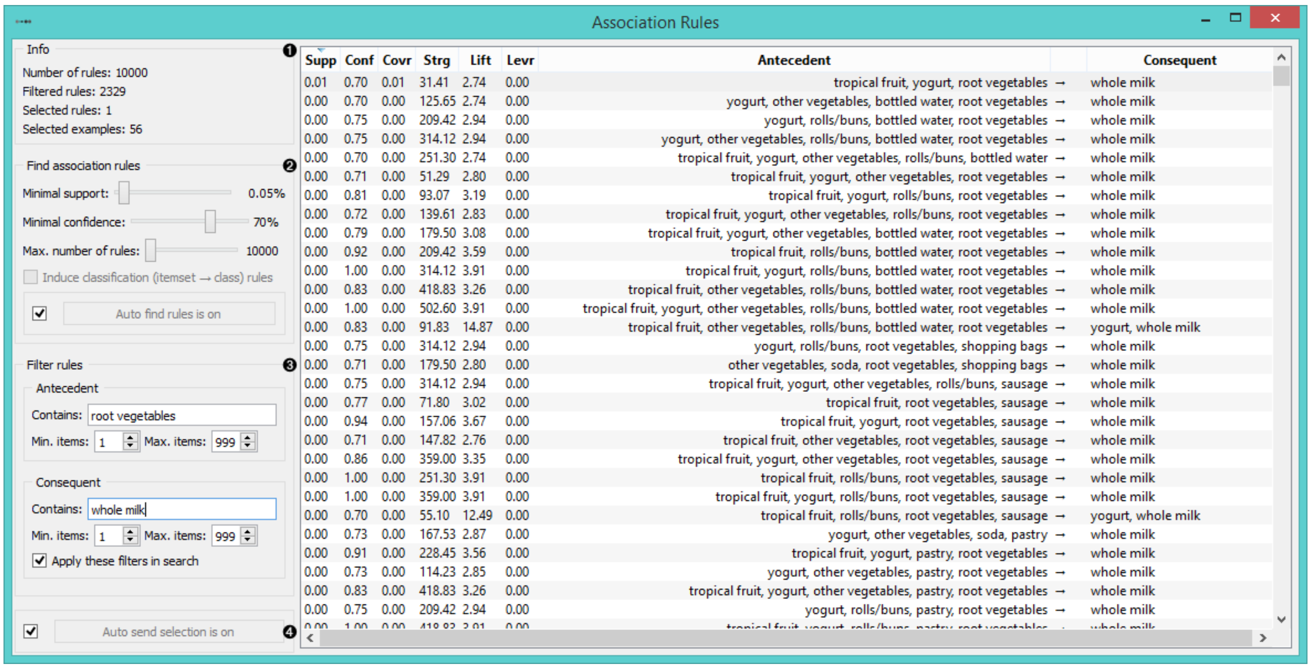
Outputs

- **Matching Data**

  Data instances matching the criteria.

Description

This widget implements FP-growth [1] frequent pattern mining algorithm with bucketing optimization [2] for conditional databases of few items. For inducing classification rules, it generates rules for the entire itemset and skips the rules where the consequent does not match one of the class' values.



Info
Number of rules: 10000
Filtered rules: 2329
Selected rules: 1
Selected examples: 56

Find association rules
Minimal support: 0.05%
Minimal confidence: 70%
Max. number of rules: 10000
☐ Induce classification (itemset → class) rules
✓ Auto find rules is on

Filter rules
Antecedent
Contains: root vegetables
Min. items: 1 Max. items: 999
Consequent
Contains: whole milk
Min. items: 1 Max. items: 999
✓ Apply these filters in search

✓ Auto send selection is on

Supp	Conf	Covr	Strg	Lift	Levr	Antecedent		Consequent
0.01	0.70	0.01	31.41	2.74	0.00	tropical fruit, yogurt, root vegetables	→	whole milk
0.00	0.70	0.00	125.65	2.74	0.00	yogurt, other vegetables, bottled water, root vegetables	→	whole milk
0.00	0.75	0.00	209.42	2.94	0.00	yogurt, rolls/buns, bottled water, root vegetables	→	whole milk
0.00	0.75	0.00	314.12	2.94	0.00	yogurt, other vegetables, rolls/buns, bottled water, root vegetables	→	whole milk
0.00	0.70	0.00	251.30	2.74	0.00	tropical fruit, yogurt, other vegetables, rolls/buns, bottled water	→	whole milk
0.00	0.71	0.00	51.29	2.80	0.00	tropical fruit, yogurt, other vegetables, root vegetables	→	whole milk
0.00	0.81	0.00	93.07	3.19	0.00	tropical fruit, yogurt, rolls/buns, root vegetables	→	whole milk
0.00	0.72	0.00	139.61	2.83	0.00	tropical fruit, yogurt, other vegetables, rolls/buns, root vegetables	→	whole milk
0.00	0.79	0.00	179.50	3.08	0.00	tropical fruit, yogurt, other vegetables, bottled water, root vegetables	→	whole milk
0.00	0.92	0.00	209.42	3.59	0.00	tropical fruit, rolls/buns, bottled water, root vegetables	→	whole milk
0.00	1.00	0.00	314.12	3.91	0.00	tropical fruit, yogurt, rolls/buns, bottled water, root vegetables	→	whole milk
0.00	0.83	0.00	418.83	3.26	0.00	tropical fruit, other vegetables, rolls/buns, bottled water, root vegetables	→	whole milk
0.00	1.00	0.00	502.60	3.91	0.00	tropical fruit, yogurt, other vegetables, rolls/buns, bottled water, root vegetables	→	whole milk
0.00	0.83	0.00	91.83	14.87	0.00	tropical fruit, other vegetables, rolls/buns, bottled water, root vegetables	→	yogurt, whole milk
0.00	0.75	0.00	314.12	2.94	0.00	yogurt, rolls/buns, root vegetables, shopping bags	→	whole milk
0.00	0.71	0.00	179.50	2.80	0.00	other vegetables, soda, root vegetables, shopping bags	→	whole milk
0.00	0.75	0.00	314.12	2.94	0.00	tropical fruit, yogurt, other vegetables, rolls/buns, sausage	→	whole milk
0.00	0.77	0.00	71.80	3.02	0.00	tropical fruit, root vegetables, sausage	→	whole milk
0.00	0.94	0.00	157.06	3.67	0.00	tropical fruit, yogurt, root vegetables, sausage	→	whole milk
0.00	0.71	0.00	147.82	2.76	0.00	tropical fruit, other vegetables, root vegetables, sausage	→	whole milk
0.00	0.86	0.00	359.00	3.35	0.00	tropical fruit, yogurt, other vegetables, root vegetables, sausage	→	whole milk
0.00	1.00	0.00	251.30	3.91	0.00	tropical fruit, rolls/buns, root vegetables, sausage	→	whole milk
0.00	1.00	0.00	359.00	3.91	0.00	tropical fruit, yogurt, rolls/buns, root vegetables, sausage	→	whole milk
0.00	0.70	0.00	55.10	12.49	0.00	tropical fruit, rolls/buns, root vegetables, sausage	→	yogurt, whole milk
0.00	0.73	0.00	167.53	2.87	0.00	yogurt, other vegetables, soda, pastry	→	whole milk
0.00	0.91	0.00	228.45	3.56	0.00	tropical fruit, yogurt, pastry, root vegetables	→	whole milk
0.00	0.73	0.00	114.23	2.85	0.00	yogurt, other vegetables, pastry, root vegetables	→	whole milk
0.00	0.83	0.00	418.83	3.26	0.00	tropical fruit, yogurt, other vegetables, pastry, root vegetables	→	whole milk
0.00	0.75	0.00	209.42	2.94	0.00	yogurt, rolls/buns, pastry, root vegetables	→	whole milk
0.00	1.00	0.00	418.83	3.91	0.00	tropical fruit, yogurt, rolls/buns, pastry, root vegetables	→	whole milk

1. Information on the data set.
2. In *Find association rules* you can set criteria for rule induction:
    - **Minimal support**: percentage of the entire data set covered by the entire rule (antecedent and consequent).
    - **Minimal confidence**: proportion of the number of examples which fit the right side (consequent) among those that fit the left side (antecedent).
    - **Max. number of rules**: limit the number of rules the algorithm generates. Too many rules can slow down the widget considerably.

    If *Induce classification (itemset → class) rules* is ticked, the widget will only generate rules that have a class value on the right-hand side (consequent) of the rule.

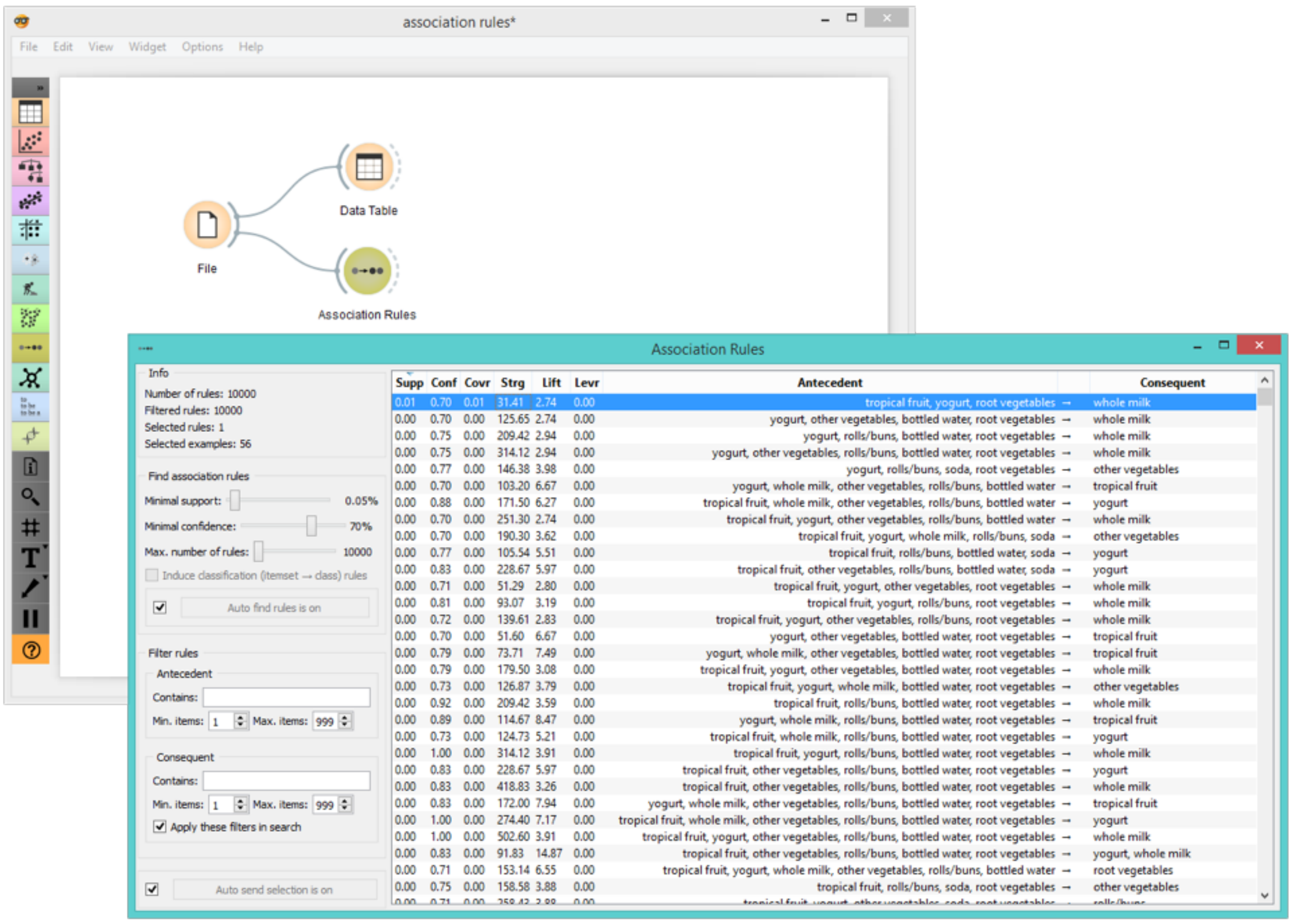
    If *Auto find rules is on*, the widget will run the search at every change of parameters. Might be slow for data sets with many attributes, so pressing *Find rules* only when the parameters are set is a good idea.
3. *Filter rules* by:
    - **Antecedent**:
        - *Contains*: will filter rules by matching space-separated regular expressions in antecedent items.
        - *Min. items*: minimum number of items that have to appear in an antecedent.
        - *Max. items*: maximum number of items that can appear in an antecedent.
    - **Consequent**:
        - *Contains*: will filter rules by matching space-separated regular expressions in consequent items.
        - *Min. items*: minimum number of items that have to appear in a consequent.
        - *Max. items*: maximum number of items that can appear in a consequent.

    If *Apply these filters in search* is ticked, the widget will limit the rule generation only to rules that match the filters. If unchecked, all rules are generated, but only the matching are shown.

4. If *Auto send selection is on*, data instances that match the selected association rules are output automatically. Alternatively press *Send selection*.

Example

Association Rules can be used directly with the File widget.



Association Rules

Info
Number of rules: 10000
Filtered rules: 10000
Selected rules: 1
Selected examples: 56

Find association rules
Minimal support: 0.05%
Minimal confidence: 70%
Max. number of rules: 10000
☐ Induce classification (itemset → class) rules
☑ Auto find rules is on

Filter rules
Antecedent
Contains:
Min. items: 1 Max. items: 999
Consequent
Contains:
Min. items: 1 Max. items: 999
☑ Apply these filters in search

☑ Auto send selection is on

Supp	Conf	Covr	Strg	Lift	Levr	Antecedent		Consequent
0.01	0.70	0.01	31.41	2.74	0.00	tropical fruit, yogurt, root vegetables	→	whole milk
0.00	0.70	0.00	125.65	2.74	0.00	yogurt, other vegetables, bottled water, root vegetables	→	whole milk
0.00	0.75	0.00	209.42	2.94	0.00	yogurt, rolls/buns, bottled water, root vegetables	→	whole milk
0.00	0.75	0.00	314.12	2.94	0.00	yogurt, other vegetables, rolls/buns, bottled water, root vegetables	→	whole milk
0.00	0.77	0.00	146.38	3.98	0.00	yogurt, rolls/buns, soda, root vegetables	→	other vegetables
0.00	0.70	0.00	103.20	6.67	0.00	yogurt, whole milk, other vegetables, rolls/buns, bottled water	→	tropical fruit
0.00	0.88	0.00	171.50	6.27	0.00	tropical fruit, whole milk, other vegetables, rolls/buns, bottled water	→	yogurt
0.00	0.70	0.00	251.30	2.74	0.00	tropical fruit, yogurt, other vegetables, rolls/buns, bottled water	→	whole milk
0.00	0.70	0.00	190.30	3.62	0.00	tropical fruit, yogurt, whole milk, rolls/buns, soda	→	other vegetables
0.00	0.77	0.00	105.54	5.51	0.00	tropical fruit, rolls/buns, bottled water, soda	→	yogurt
0.00	0.83	0.00	228.67	5.97	0.00	tropical fruit, other vegetables, rolls/buns, bottled water, soda	→	yogurt
0.00	0.71	0.00	51.29	2.80	0.00	tropical fruit, yogurt, other vegetables, root vegetables	→	whole milk
0.00	0.81	0.00	93.07	3.19	0.00	tropical fruit, yogurt, rolls/buns, root vegetables	→	whole milk
0.00	0.72	0.00	139.61	2.83	0.00	tropical fruit, yogurt, other vegetables, rolls/buns, root vegetables	→	whole milk
0.00	0.70	0.00	51.60	6.67	0.00	yogurt, other vegetables, bottled water, root vegetables	→	tropical fruit
0.00	0.79	0.00	73.71	7.49	0.00	yogurt, whole milk, other vegetables, bottled water, root vegetables	→	tropical fruit
0.00	0.79	0.00	179.50	3.08	0.00	tropical fruit, yogurt, other vegetables, bottled water, root vegetables	→	whole milk
0.00	0.73	0.00	126.87	3.79	0.00	tropical fruit, yogurt, whole milk, bottled water, root vegetables	→	other vegetables
0.00	0.92	0.00	209.42	3.59	0.00	tropical fruit, rolls/buns, bottled water, root vegetables	→	whole milk
0.00	0.89	0.00	114.67	8.47	0.00	yogurt, whole milk, rolls/buns, bottled water, root vegetables	→	tropical fruit
0.00	0.73	0.00	124.73	5.21	0.00	tropical fruit, whole milk, rolls/buns, bottled water, root vegetables	→	yogurt
0.00	1.00	0.00	314.12	3.91	0.00	tropical fruit, yogurt, rolls/buns, bottled water, root vegetables	→	whole milk
0.00	0.83	0.00	228.67	5.97	0.00	tropical fruit, other vegetables, rolls/buns, bottled water, root vegetables	→	yogurt
0.00	0.83	0.00	418.83	3.26	0.00	tropical fruit, other vegetables, rolls/buns, bottled water, root vegetables	→	whole milk
0.00	0.83	0.00	172.00	7.94	0.00	yogurt, whole milk, other vegetables, rolls/buns, bottled water, root vegetables	→	tropical fruit
0.00	1.00	0.00	274.40	7.17	0.00	tropical fruit, whole milk, other vegetables, rolls/buns, bottled water, root vegetables	→	yogurt
0.00	1.00	0.00	502.60	3.91	0.00	tropical fruit, yogurt, other vegetables, rolls/buns, bottled water, root vegetables	→	whole milk
0.00	0.83	0.00	91.83	14.87	0.00	tropical fruit, other vegetables, rolls/buns, bottled water, root vegetables	→	yogurt, whole milk
0.00	0.71	0.00	153.14	6.55	0.00	tropical fruit, yogurt, whole milk, other vegetables, rolls/buns, bottled water	→	root vegetables
0.00	0.75	0.00	158.58	3.88	0.00	tropical fruit, rolls/buns, soda, root vegetables	→	other vegetables
0.00	0.71	0.00	258.42	2.88	0.00	tropical fruit, yogurt, other vegetables, soda, root vegetables	→	rolls/buns

References and further reading

[1]: J. Han, J. Pei, Y. Yin, R. Mao. (2004) Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach.

[2]: R. Agrawal, C. Aggarwal, V. Prasad. (2000) Depth first generation of long patterns.

On how to use regular expressions.

Frequent Itemsets

Finds frequent itemsets in the data.

Signals

Inputs

- **Data**

  Data set

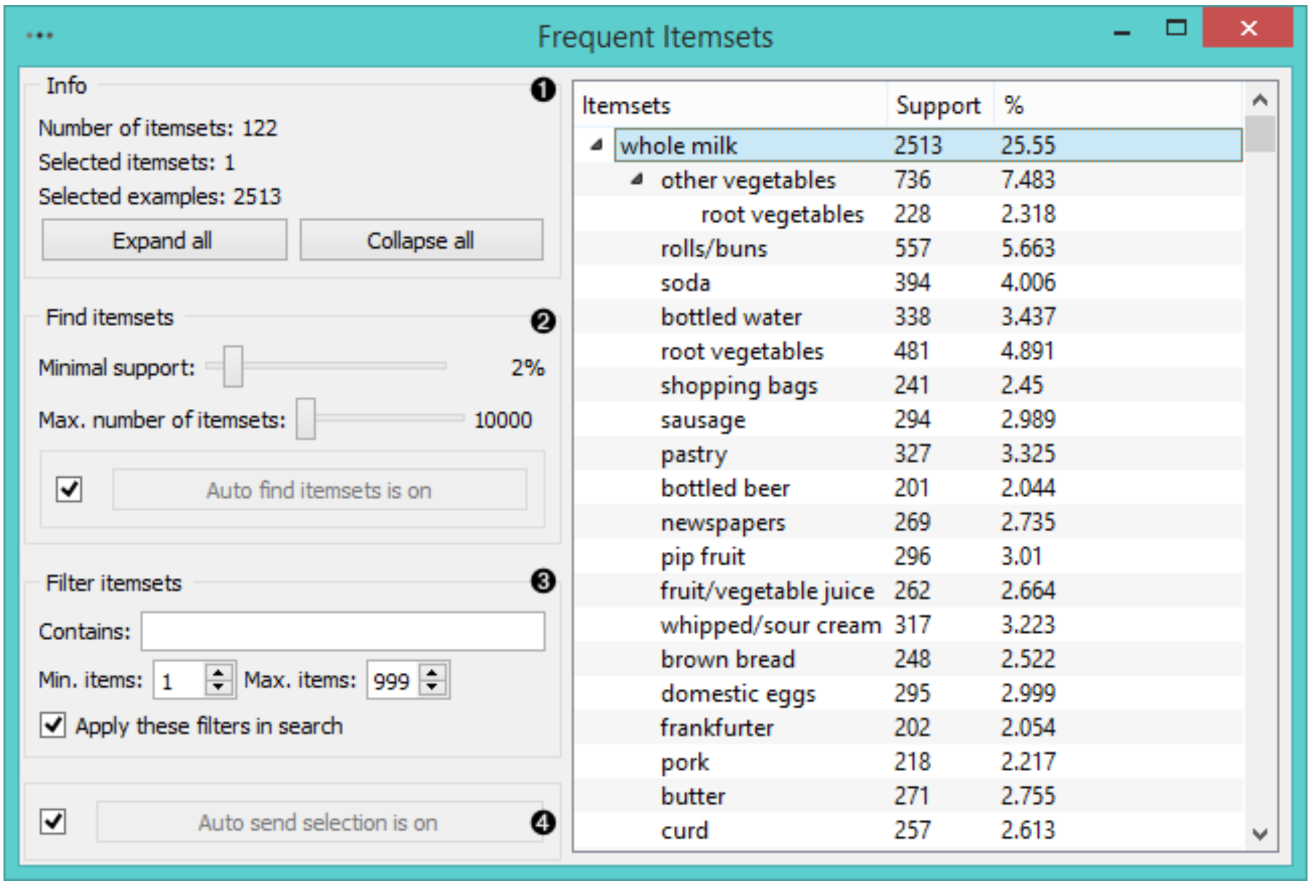
Outputs

- **Matching Data**

  Data instances matching the criteria.

Description

The widget finds frequent items in a data set based on a measure of support for the rule.



Number of itemsets: 122
Selected itemsets: 1
Selected examples: 2513

Minimal support: 2%
Max. number of itemsets: 10000
☑ Auto find itemsets is on

Contains:
Min. items: 1 Max. items: 999
☑ Apply these filters in search

☑ Auto send selection is on

Itemsets	Support	%
whole milk	2513	25.55
other vegetables	736	7.483
root vegetables	228	2.318
rolls/buns	557	5.663
soda	394	4.006
bottled water	338	3.437
root vegetables	481	4.891
shopping bags	241	2.45
sausage	294	2.989
pastry	327	3.325
bottled beer	201	2.044
newspapers	269	2.735
pip fruit	296	3.01
fruit/vegetable juice	262	2.664
whipped/sour cream	317	3.223
brown bread	248	2.522
domestic eggs	295	2.999
frankfurter	202	2.054
pork	218	2.217
butter	271	2.755
curd	257	2.613

1. Information on the data set. ‘Expand all’ expands the frequent itemsets tree, while ‘Collapse all’ collapses it.
2. In *Find itemsets by* you can set criteria for itemset search:
   - **Minimal support**: a minimal ratio of data instances that must support (contain) the itemset for it to be generated. For large data sets it is normal to set a lower minimal support (e.g. between 2%-0.01%).
   - **Max. number of itemsets**: limits the upward quantity of generated itemsets. Itemsets are generated in no particular order.

   If *Auto find itemsets is on*, the widget will run the search at every change of parameters. Might be slow for large data sets, so pressing *Find itemsets* only when the parameters are set is a good idea.
3. *Filter itemsets*:

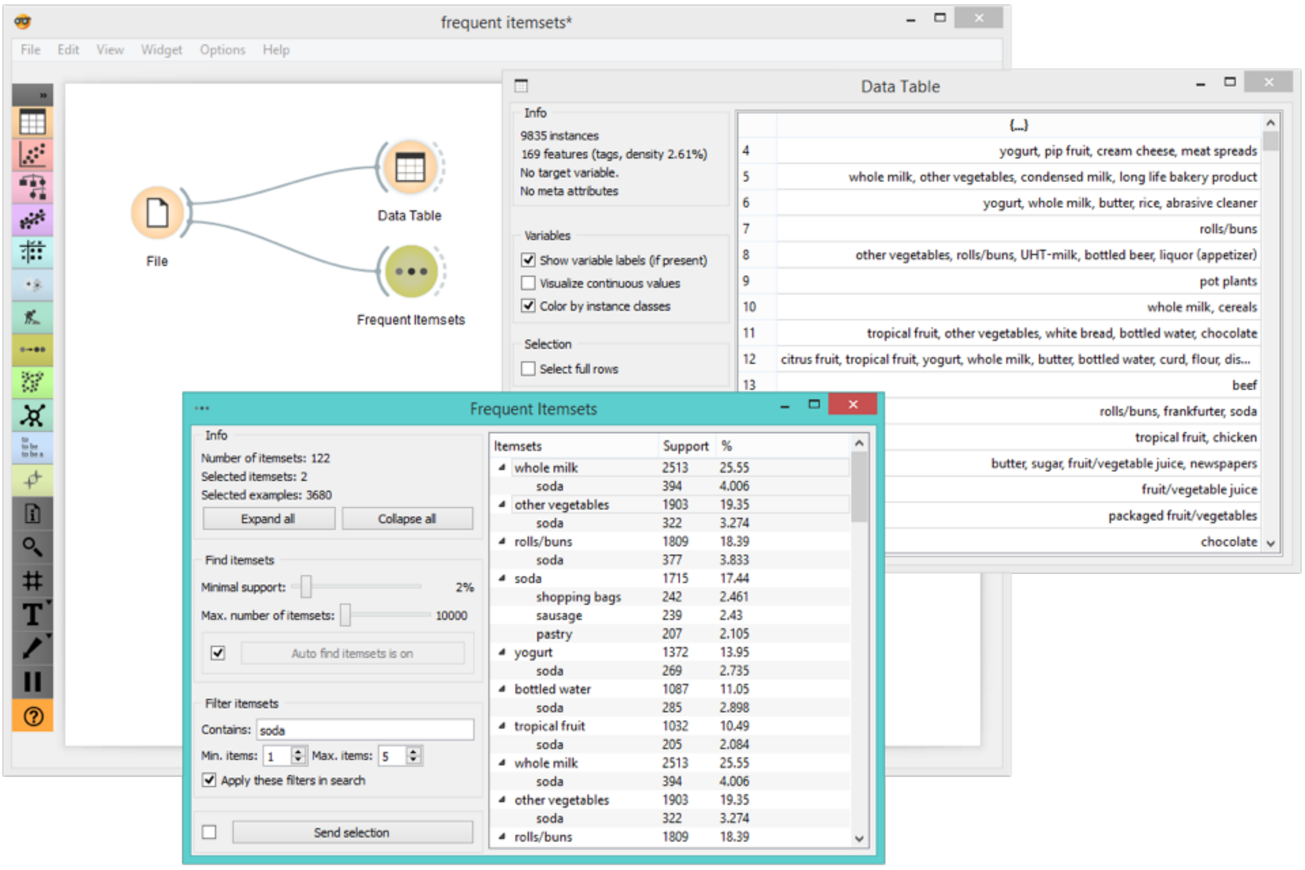
   If you’re looking for a specific item or itemsets, filter the results by regular expressions. Separate regular expressions by comma to filter by more than one word.
   - **Contains**: will filter itemsets by regular expressions.
   - **Min. items**: minimum number of items that have to appear in an itemset. If 1, all the itemsets will be displayed. Increasing it to, say, 4, will only display itemsets with four or more items.
   - **Max. items**: maximum number of items that are to appear in an itemset. If you wish to find, say, only itemsets with less than 5 items in it, you’d set this parameter to 5.

If *Apply these filters in search* is ticked, the widget will filter the results in real time. Preferably not ticked for large data sets.

4. If *Auto send selection is on*, changes are communicated automatically. Alternatively press *Send selection*.

Example

Frequent Itemsets can be used directly with the File widget.



Data Table

Info
9835 instances
169 features (tags, density 2.61%)
No target variable.
No meta attributes

Variables
☑ Show variable labels (if present)
☐ Visualize continuous values
☑ Color by instance classes

Selection
☐ Select full rows

	{...}
4	yogurt, pip fruit, cream cheese, meat spreads
5	whole milk, other vegetables, condensed milk, long life bakery product
6	yogurt, whole milk, butter, rice, abrasive cleaner
7	rolls/buns
8	other vegetables, rolls/buns, UHT-milk, bottled beer, liquor (appetizer)
9	pot plants
10	whole milk, cereals
11	tropical fruit, other vegetables, white bread, bottled water, chocolate
12	citrus fruit, tropical fruit, yogurt, whole milk, butter, bottled water, curd, flour, dis...
13	beef
	rolls/buns, frankfurter, soda
	tropical fruit, chicken
	butter, sugar, fruit/vegetable juice, newspapers
	fruit/vegetable juice
	packaged fruit/vegetables
	chocolate

Frequent Itemsets

Info
Number of itemsets: 122
Selected itemsets: 2
Selected examples: 3680
Expand all | Collapse all

Find itemsets
Minimal support: 2%
Max. number of itemsets: 10000
☑ Auto find itemsets is on

Filter itemsets
Contains: soda
Min. items: 1 Max. items: 5
☑ Apply these filters in search

☐ Send selection

Itemsets	Support	%
whole milk	2513	25.55
soda	394	4.006
other vegetables	1903	19.35
soda	322	3.274
rolls/buns	1809	18.39
soda	377	3.833
soda	1715	17.44
shopping bags	242	2.461
sausage	239	2.43
pastry	207	2.105
yogurt	1372	13.95
soda	269	2.735
bottled water	1087	11.05
soda	285	2.898
tropical fruit	1032	10.49
soda	205	2.084
whole milk	2513	25.55
soda	394	4.006
other vegetables	1903	19.35
soda	322	3.274
rolls/buns	1809	18.39