Single-Cell Gene Expression Analysis

Working notes for the course on Functional Genomics and Proteomics, University of Ljubljana

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

These notes were written by Blaž Zupan with a huge help from the members of the Bioinformatics Lab in Ljubljana that develop and maintain orange. In part, we have reused lecture notes for Orange and orange workshops as designed by the same group.

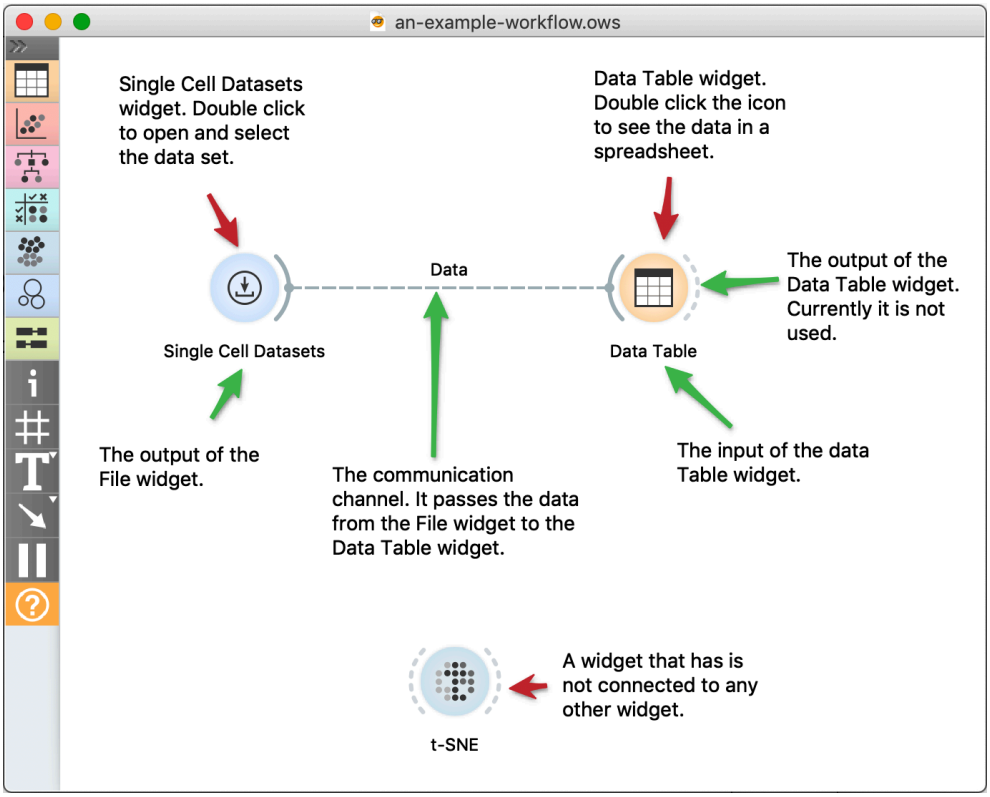
Welcome! We have designed this workshop for molecular biologists interested in interactive single-cell gene expression data analysis. You will see how you can accomplish single-cell data mining tasks without programming. We will use Orange, its bioinformatics and single-cell add-ons to construct reproducible, shareable, and visual data mining workflows.

If you haven't already installed Orange, please download the installation package from https://orange.biolab.si, install Orange, and then install the add-ons for Bioinformatics and Single-Cell Data Analysis.

Lesson 1: Workflows in Orange

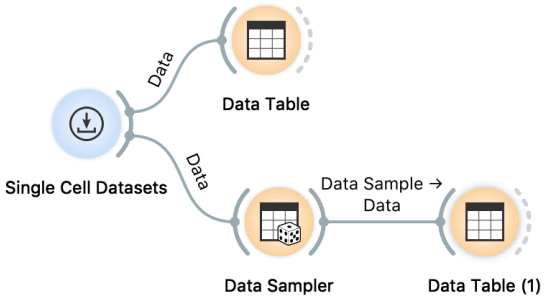
Before we dive into single-cell analysis, let us get familiar with the tool that we will be using. Orange is a machine learning and interactive data visualization platform, where data analysis pipeline are designed as workflows. Orange workflows consist of components that read, process and visualize the data. We call the components "widgets". Widgets are placed on a drawing board (the "canvas"). Widgets communicate by sending information along a communication channel. Output from one widget is used as input to another.

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



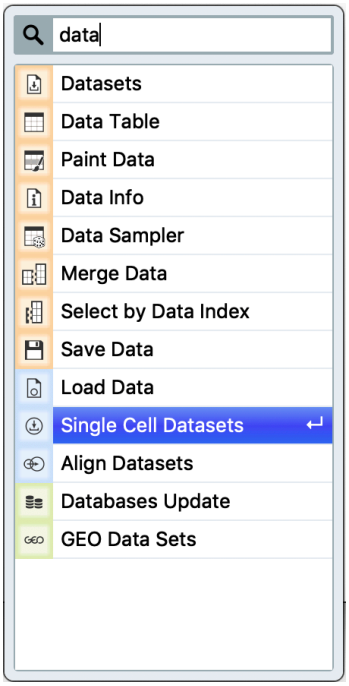
We construct workflows by placing widgets onto the canvas and connecting them by drawing a line from the transmitting widget to the receiving widget. The widget's outputs are on the right and the inputs on the left. In the workflow above, the File widget sends data to the Data Table widget.

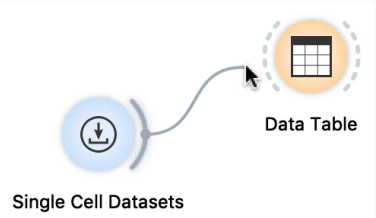
Let us start by constructing a workflow that consists of a *Single Cell Datasets* widget, two *Data Tables*, and a *Data Sampler* widget:



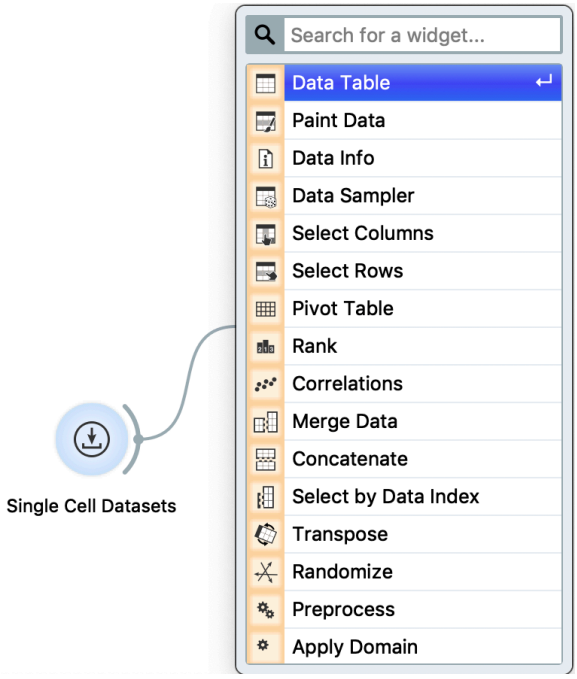
Workflow with a Single Cell Datasets widget that reads the data from orange's single cell data repository. The Data Table renders the data in a spreadsheet, while the Data Sampler samples the data. Sampled data is sent to another Data Table widget.

There are several ways to add widgets to the canvas. Try adding a file widget with a right click on the empty part of the canvas, start writing the name of the widget (like "data") in a widget list and then select Single Cell Datasets by pressing return or clicking on appropriate line. We can add the data table in the same way: right click on the empty canvas, write "data" in the search box, press down arrow key until we hit a line with the Data Table, and then press return. Once the two widgets are on the canvas, we can connect them by clicking on the right "ear" of the *File* widget and releasing a mouse on the line that says Data Table.

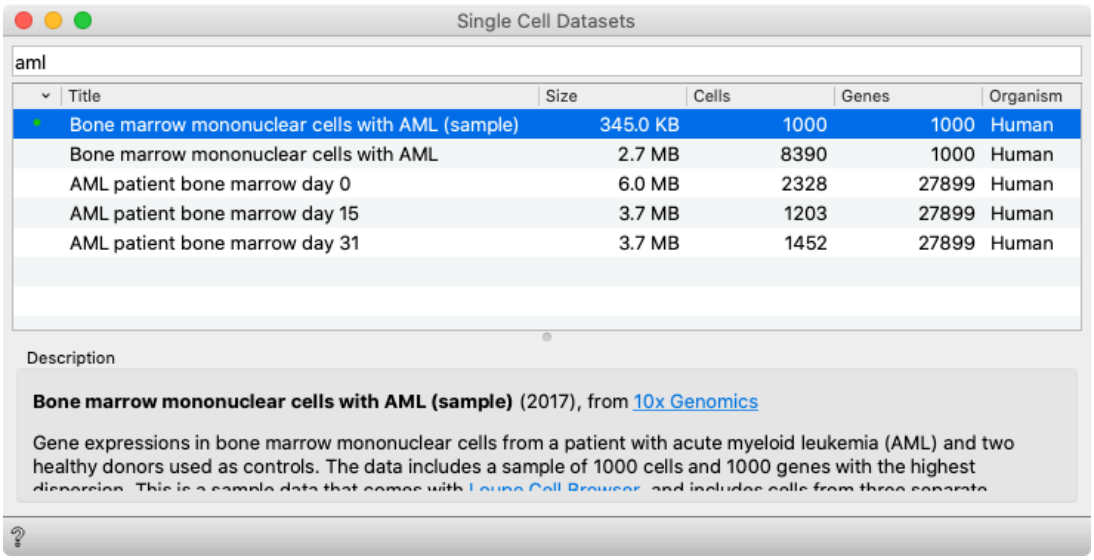




Another quick way to add a widget is to simply drag a connection from the sending widget, release the mouse somewhere on the empty canvas, and then in the menu with a list of widgets selecting the receiving widget.



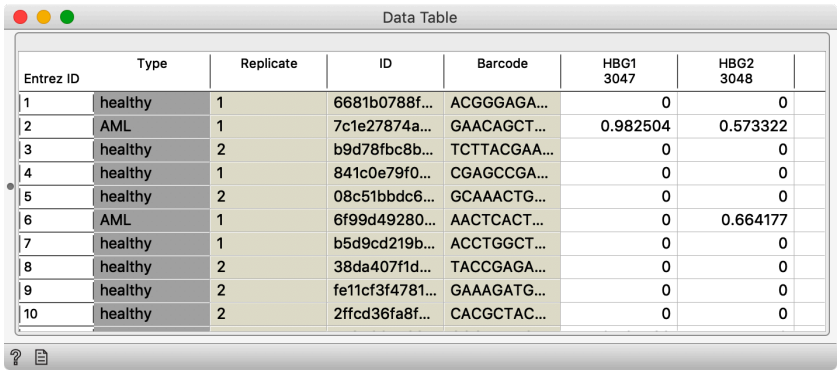
The *Single Cell Datasets* reads the data from the server. Open the *Single Cell Datasets* widget by double-clicking its icon. The window shows a list of available data sets. Let us choose a smaller dataset with bone marrow mononuclear cells from, a sample of the data from Zhang et al. (2017). After selecting the line with this data set, press "Send Data" to instruct the widget to send the data to its output.

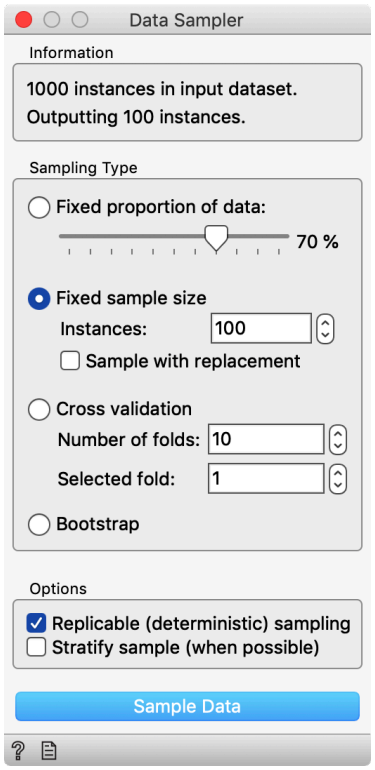


After loading the data, open the *Data Table* to see the data we have just loaded in the spreadsheet. Single cell data sets includes many dropouts, and many of the corresponding data entries have a value of zero: the gene was either not expressed or, more likely, its expression was not detected.

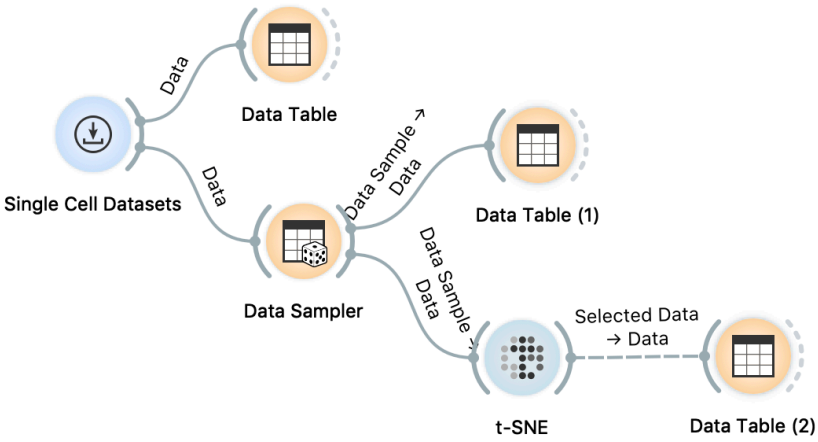
Orange is machine learning tool, and stores data items in rows. In single cell genomics, our objects of interest are cells. The cell expression profiles are therefore stored in rows. Every column refers to a gene.

Several columns contain meta information on the cells. Our example data includes type, replicate id and some other information.

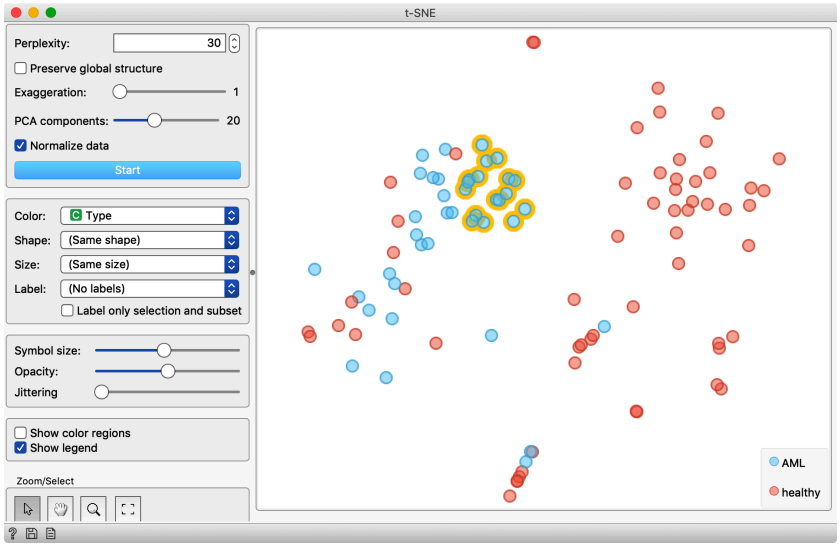




Let us now sample a 100 cells from the data. We will use *Data Sampler* widget for this task. Now, augment our workflow to visualize the data in a small plot using *t-SNE* visualization (more about this widget a bit later). We will also add another Data Table at the output of *t-SNE*.



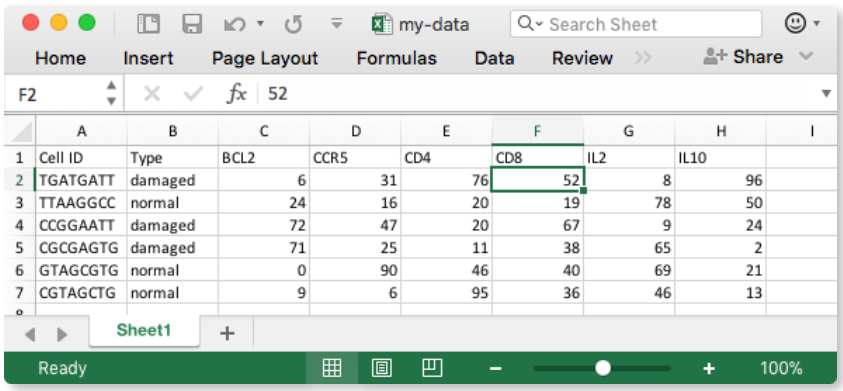
Open the *t-SNE* widget and select few data points by drawing the rectangle around them. Now open the *Data Table* (2) to observe how the data on selected cells are passed to the output of *t-SNE*. In Orange, most of the widgets are interactive, and send out the data upon any change in selection or any change of parameters of the widget.



Lesson 2: Loading Your Own Dataset

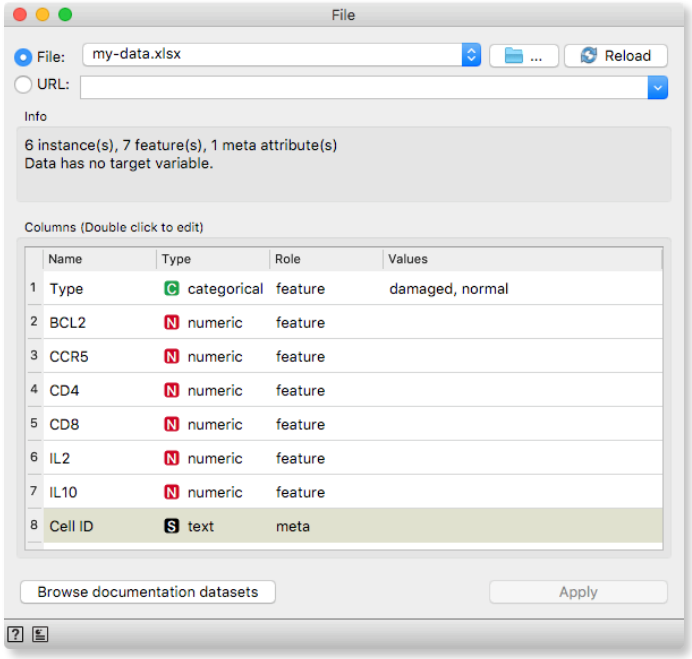
Orange reads data from Excel, comma- and tab-separated files and urls. Try constructing a spreadsheet in Excel or in Google Sheets. If using Google Sheets, copy a shareable link and paste it into the File widget. Press Enter to load the data.

The datasets we have worked with in the previous lesson come from the server. Orange can read the data from spreadsheet file formats which include tab and comma separated and Excel files. Let us prepare a toy dataset in Excel and save it on a local disk.



	A	B	C	D	E	F	G	H
1	Cell ID	Type	BCL2	CCR5	CD4	CD8	IL2	IL10
2	TGATGATT	damaged	6	31	76	52	8	96
3	TTAAGGCC	normal	24	16	20	19	78	50
4	CCGGAATT	damaged	72	47	20	67	9	24
5	CGCGAGTG	damaged	71	25	11	38	65	2
6	GTAGCGTG	normal	0	90	46	40	69	21
7	CGTAGCTG	normal	9	6	95	36	46	13

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

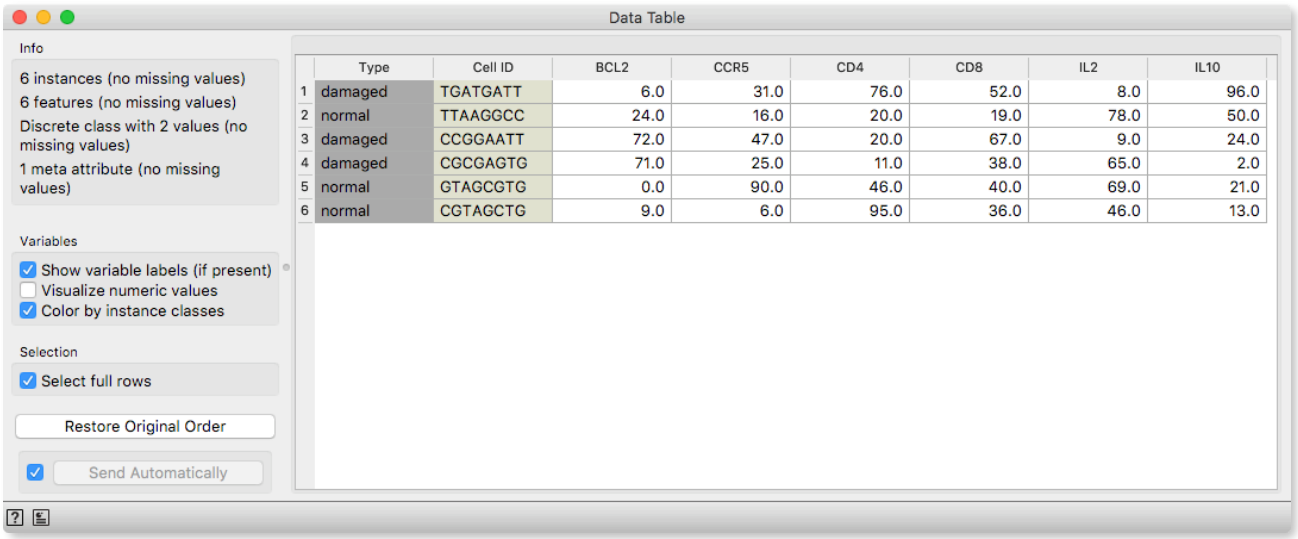


Looks good. Orange has correctly guessed that cell IDs are character strings and that this column in the dataset is special, meant to provide additional information and not to be used for any kind of modeling. All other columns are numeric features except

for the type, which is a categorical feature. This is also the feature we would not like to include in the profile of the cell and should rather consider it as a cell's class. Double-click on the "feature" in the *Role* column and change the role of the feature type to "target". Then click the *Apply* button.

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

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



	Type	Cell ID	BCL2	CCR5	CD4	CD8	IL2	IL10
1	damaged	TGATGATT	6.0	31.0	76.0	52.0	8.0	96.0
2	normal	TTAAGGCC	24.0	16.0	20.0	19.0	78.0	50.0
3	damaged	CCGGAATT	72.0	47.0	20.0	67.0	9.0	24.0
4	damaged	CGCGAGTG	71.0	25.0	11.0	38.0	65.0	2.0
5	normal	GTAGCGTG	0.0	90.0	46.0	40.0	69.0	21.0
6	normal	CGTAGCTG	9.0	6.0	95.0	36.0	46.0	13.0

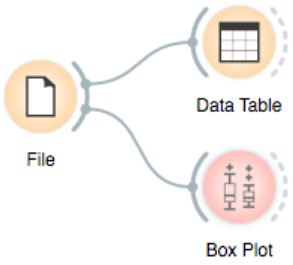
Nice, everything is here.

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

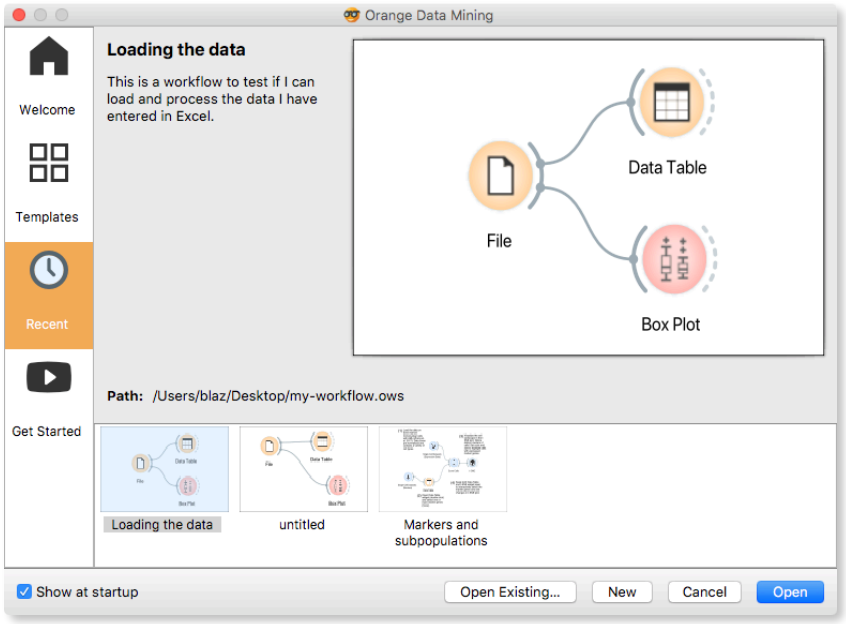
Instead of using Excel, we could also use Google Sheets, a free online spreadsheet alternative. Then, instead of finding the file on the local disk, we would enter its URL address to the File widget 's URL entry box.

Lesson 3: Saving and Sharing Your Work

Add a box plot to your workflow from the previous lesson. Your workflow should now look like this:



Orange can save your workflow to a file. Use Save or Save As from the File menu and save your work to your desktop. Before saving, you can describe your workflow on the information page using "i" icon for the toolbar. We have named the file as *my-workflow* and Orange would add a suffix to it. Now exit Orange and rerun it. Select *Recent* in the welcome screen and choose from one of the recently saved workflows.

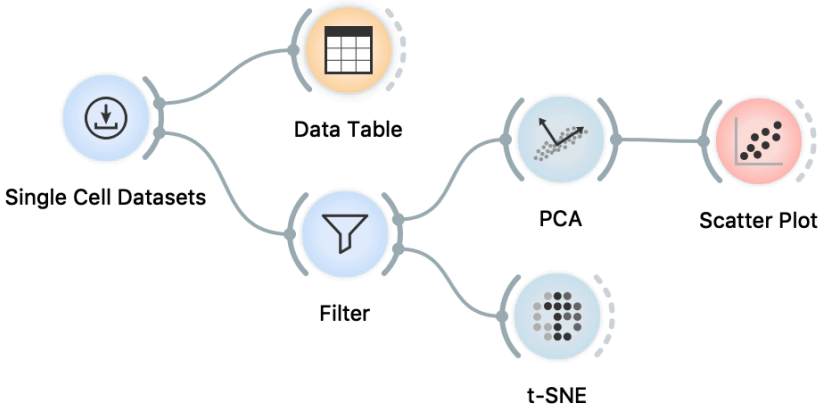


You can email workflow files or share them with your colleagues. Note though that the data, unless stored on the web, has to be sent separately.

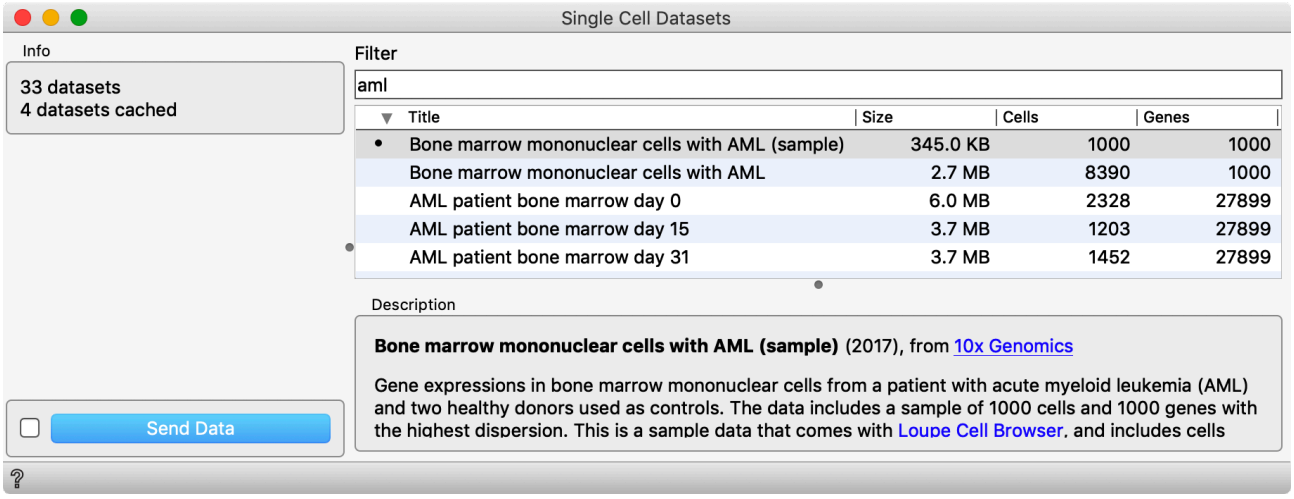
Lesson 4: Single-Cell Landscapes

Let us load some single-cell gene expression data and organize the cells in two-dimensional visualizations. We will use the following workflow, and within it, compare two popular data visualization approaches, principal component analysis and t-distributed stochastic neighbor embedding.

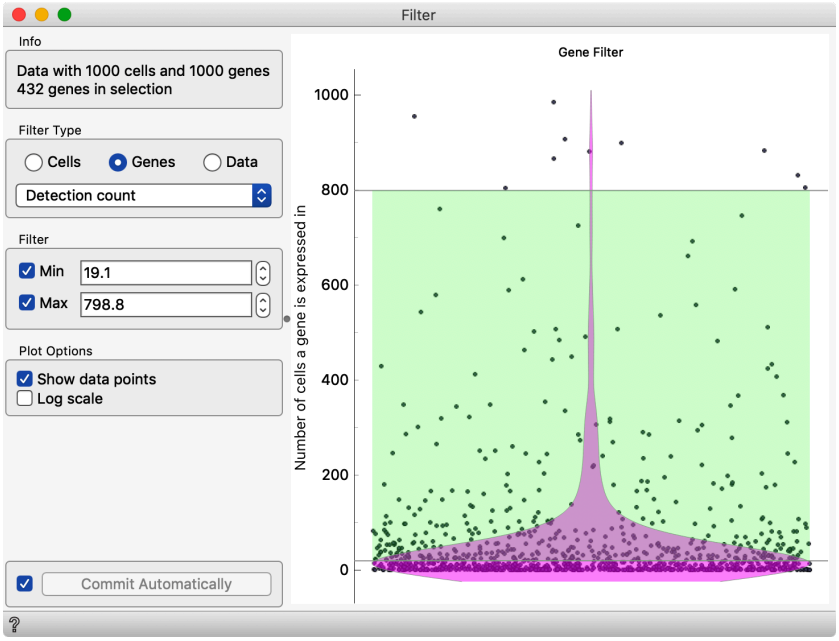
Single Cell Datasets connects to orange's data server that contains examples of datasets. You have to be connected to a network for this widget to work correctly.



From a list of examples in *Single Cell Datasets*, let us choose the data on mononuclear cells from bone marrow (Zheng et al., Nat Comm 2017). This data sets has already been preprocessed (to some degree) and comes with a selection of 1,000 genes.

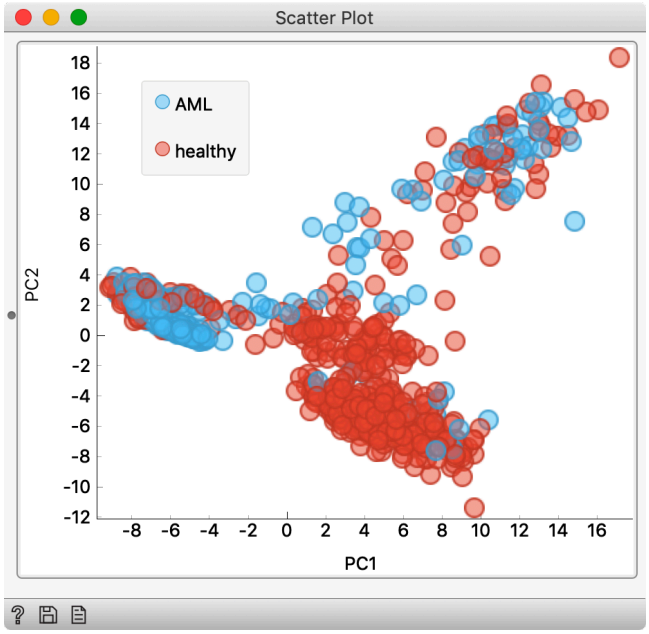


Inspection of this data set in the *Data Table* reveals that for specific cell many of the genes have not been expressed, that is, their expression is zero. We can use the Filter widget to observe the number of cells in which the genes have been expressed.

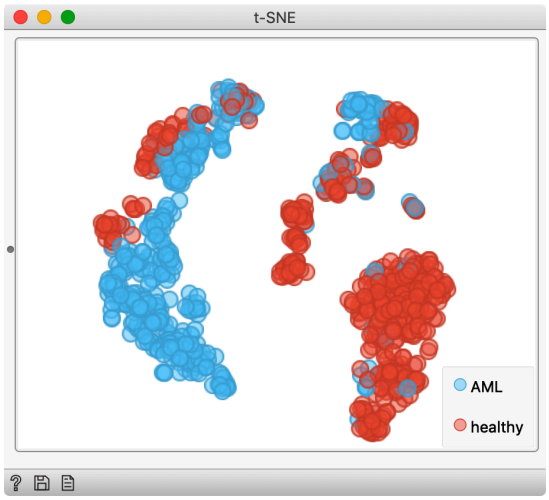


In the *Filter* widget, we have decided to filter genes based on detection count. We can drag lines that define the selection in green to filter out genes that have been expressed in too few or too many cells. Check the t-SNE widget does not include axis. In fact, axis in t-SNE make no sense. Why?

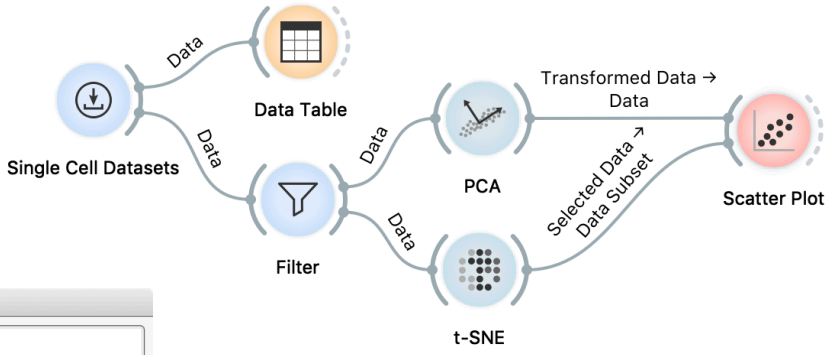
With a Filter widget, we can select the genes that have been detected at the "appropriate" number of cells, filtering out the genes that have been expressed in very few cells. We pass the data to *PCA* with the scree diagram, a chart that shows how much of the variance is explained with a first few components. PCA transforms our data to a new coordinate system defined by principal components, where the components are orthogonal to each other and where the transformation is constructed so that the first component explains most of the variance, then second-most of the remaining variance, and so on.

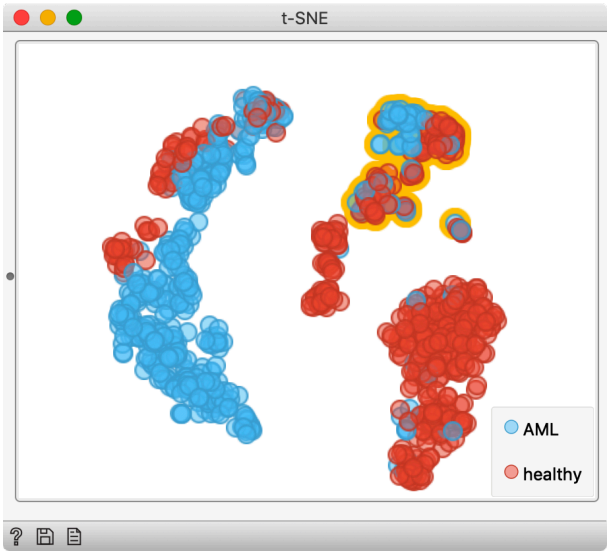


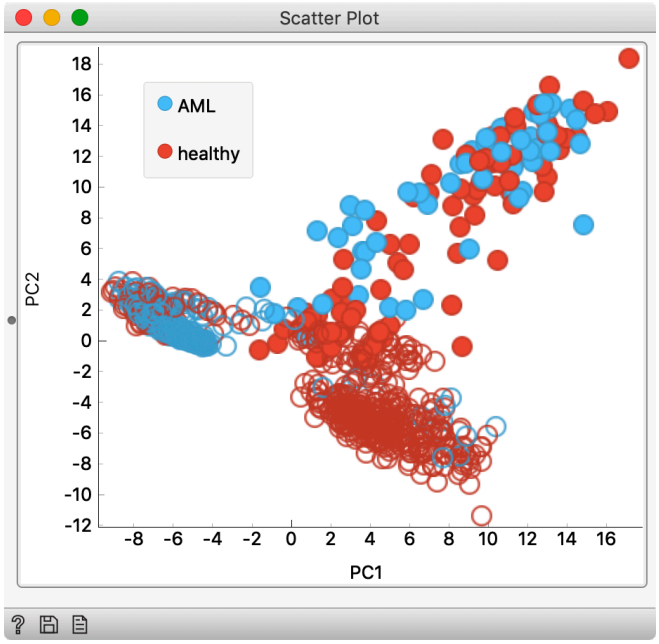
A conceptually very different technique to PCA is t-SNE, which embeds the data into two-dimensions so that cells with similar expression stay together.



PCA and t-SNE are two popular visualizations of single-cell gene expression data. Their visual depictions are often very different. PCA is a linear transformation that aims to be "more faithful" to the original data, while t-SNE aims to expose the clustering structure and focuses on preserving local similarities. We can compare the layout of the two visualizations by adding a connection from *t-SNE* widget to the *Scatter Plot* showing the PCA projection. With it, a subset of cells selected in the *t-SNE* will be exposed in the PCA plot.







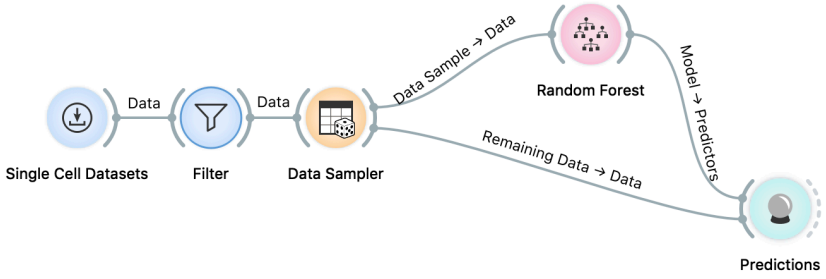
To explore the differences between t-SNE and PCA, have both associated windows open, select the data in t-SNE, and observe the changes in Scatter Plot showing PCA projection. If Orange canvas window is getting in your way, use "Bring Widgets to the Front" command from the View menu.

Lesson 5: Classification, A Detour

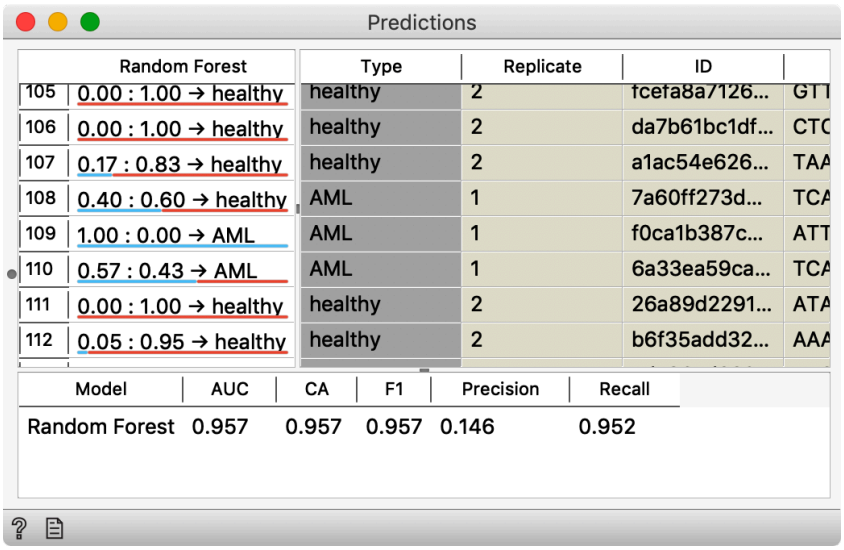
This lesson is a detour. Predicting the phenotype of the cell is indeed interesting, but our higher interest (in the following lecture) will be on cell type discovery and prediction. But since the cells that we are working on already have labels, we could not resist to invoke some machine learning, the techniques we will also use later in the workshop.

You most likely noticed that the data we are using comes from several donors. The cells are from two healthy individuals and one with acute myeloid leukemia. Visualizations from the last lesson somehow separate the diseased from the normal cells, but not entirely. Can we build a model to predict the origin (healthy, AML) of the cells? And if, how accurate this model is?

We are actually addressing the problem of prediction. We would like to build a model on a set of phenotype-labeled cells, and the use it to predict the phenotypes of the "new" cells. We here use quotes as we will not deal with an additional data set, but rather split the data that we have and use a sample to train the model, and out-of-sample cells for prediction. Here goes the workflow (we left the first two widgets from the workflow in the previous lesson, and remove everything else).



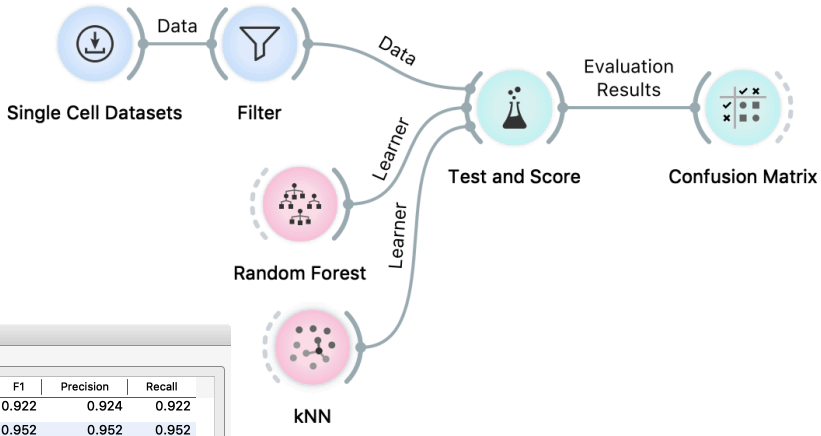
Comparing predictions of the random forest and the true type of the cell shows that the model did well, with classification accuracy of 0.957.

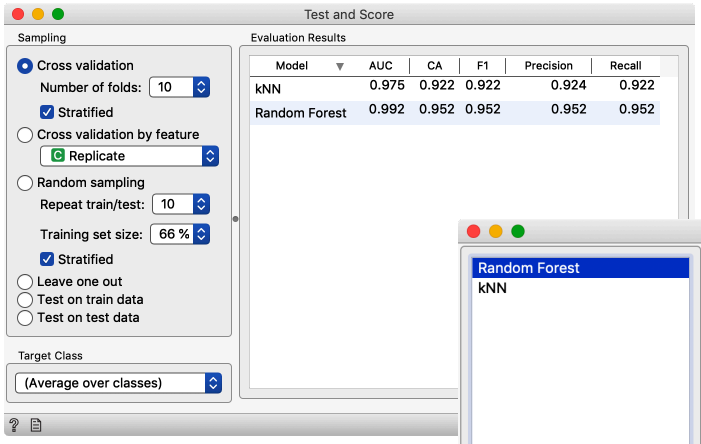


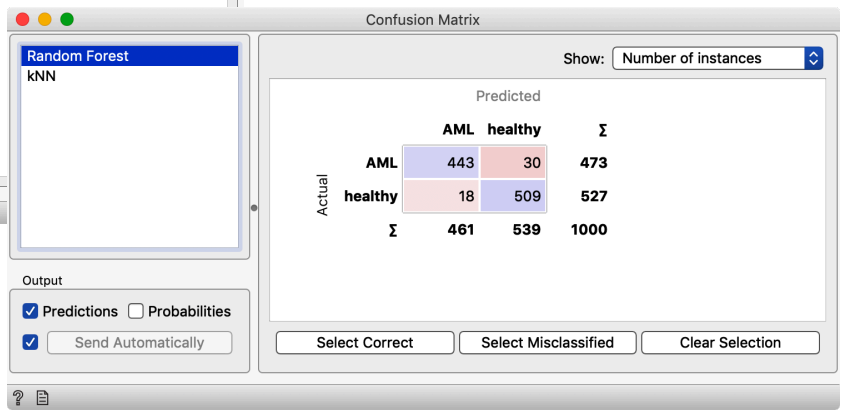
Accuracy, as reported above, depends on the particular split of the data to training and testing. Try opening *Data Sample* and pressing "Sample Data" button to see this. In machine learning, the accuracies of modeling methods are estimated through repeating sampling, training, testing procedures and reporting the average accuracy across many such iterations. Random sampling is often replaced by a procedure called cross-validation, that split the data to k samples of approximately similar size, and in each iteration uses on fold for testing and all other for training.

The widget that implements cross-validation is *Test & Score*. Cross-validated accuracy of random forest is about 0.95 (high). In the workflow, we compared it to the accuracy of k-nearest neighbor, another machine learning technique that classifies cells according to the class of its most similar cells in the training set.

Test & Score reports numerical estimates of classification accuracy. To actually see if and what went wrong in classification, a more telling is a *Confusion Matrix*. For instance, on our data, random forest models misclassified 30 cells from a donor with AML to a healthy donor.





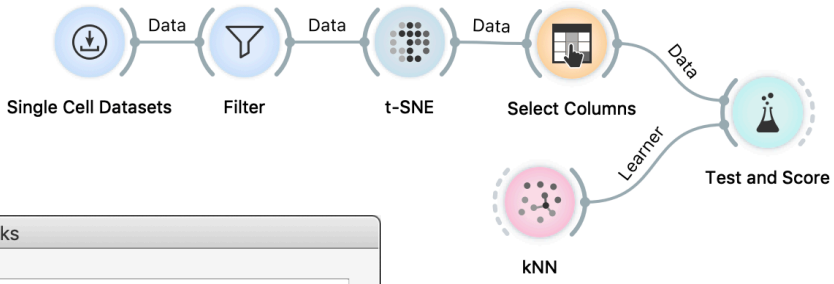


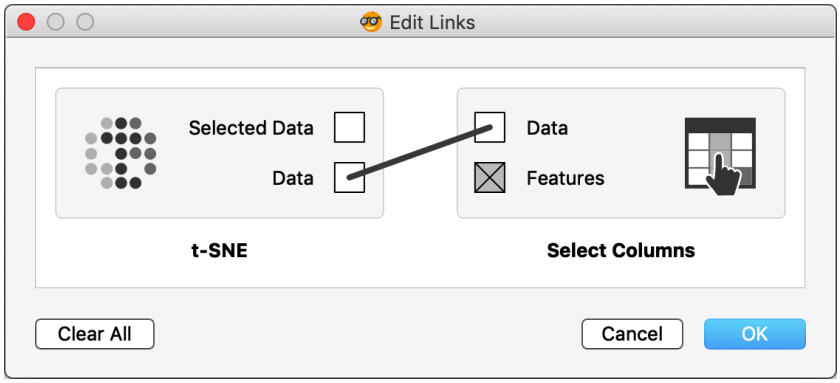
Try using a combination of *Confusion Matrix* and *t-SNE* to figure out where in the t-SNE projection are the misclassified cells. For input to *t-SNE*, use the data that comes out of *Filter*, and connect the output of *Confusion Matrix* to a subset data input of *t-SNE* widget.

Lesson 6: Classification Accuracy in Embedding Space

If the cells are labeled, as is the case with the data we have been analyzing so far, we could use cross-validation and k-nearest neighbors to assess the quality of visualization. In this case, we need to ignore expression data and cross-validate the model only on the data of positions of cells in the visualization, that is, position in the embedding (for t-SNE) or projection (for PCA) space.

We could use a similar workflow to evaluate the PCA projection, or even to compare PCA projection to t-SNE embedding. Note, though, that this comparison is concerned only with separation of cells with different labels.

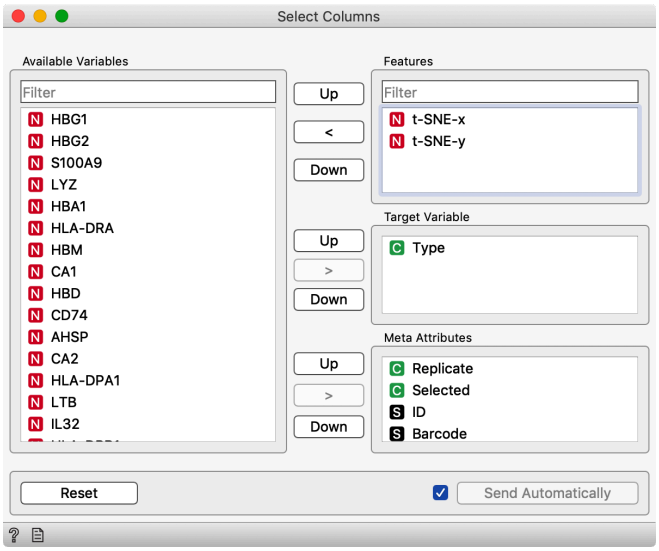




Our t-SNE widget can output coordinates of cells in the embedding space. We need these for entire data set, so we need to rewire the outgoing connection from t-SNE to include entire data.

The accuracy of kNN on embedding positions of cells is about 0.94. We can experiment with different threads set in the Filter widget to see if the quality of t-SNE position degrades with more strict filtering in terms of phenotype classification.

The composition of the data is set in Select Columns widget. We removed the gene expression features and included only embedding positions as predictive features.

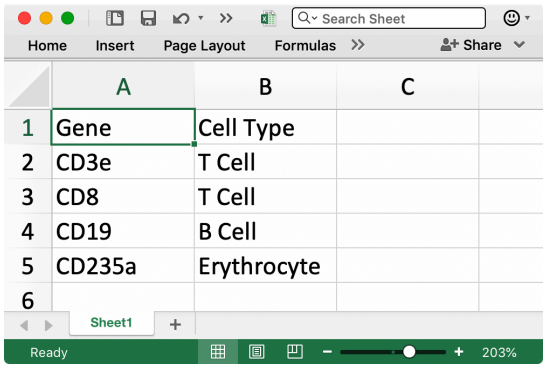


Lesson 7: Cell Types and Marker Genes

Score Cells widget has two inputs: the data and a set of marker genes. Since both inputs are data tables, how does it know which one is which? Double-click on any of the edges connecting the widgets to see the actual wiring. You can rewire the connections there.

Gene markers are one of the standard methods for discovering cell populations, determining cell states (e.g. cell cycle) and more. Here, we will continue to use a sample of the data on bone marrow mononuclear cells from previous lessons. We will score the cells according to gene markers, and observe the scored cells in the t-SNE visualization. We will assume, as it is often the case, that marker genes are keyed-in from some publication. We will only later resort to a list of marker genes from a public database..

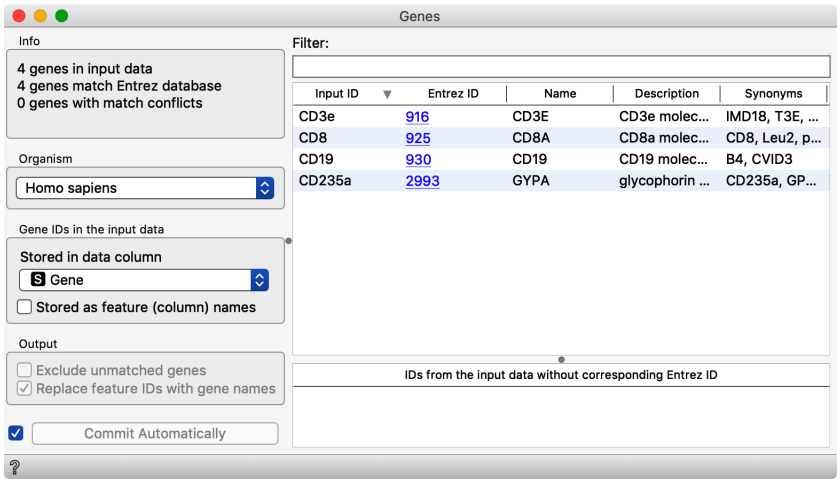
So, we start with the marker genes. We picked a few that are contained in our data. Here they are, keyed-in Excel.

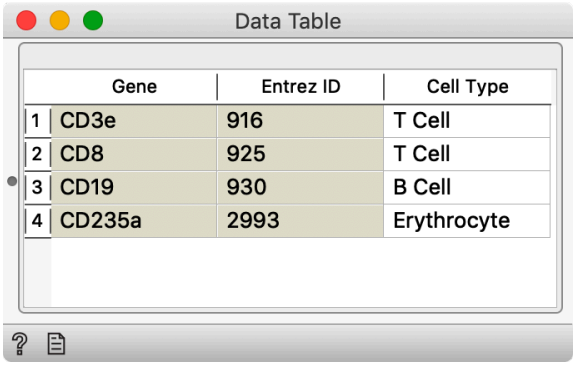


For Excel, we can save this list in, say, genes.xlsx, and load it with File widget.

Notice that we have used the only names of the genes, and not any official codes. Orange deals with genes through NCBI's IDs, and we add them using Genes widget. This widget will also tell us if the names of the genes have been resolved correctly.

Genes widget needs to be set correctly. Genes in our marker file were not stored as features, but instead in a data column Gene (there will be time when this widget will do this automatically).

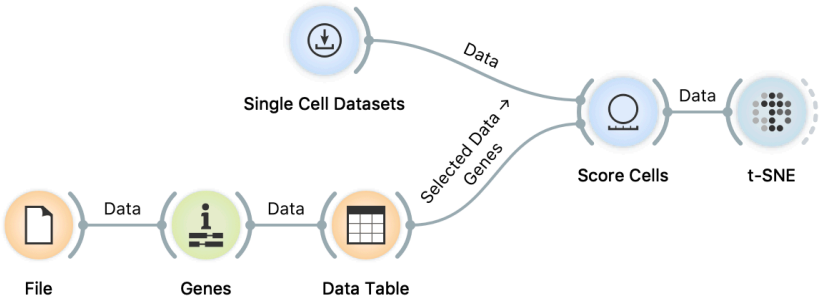


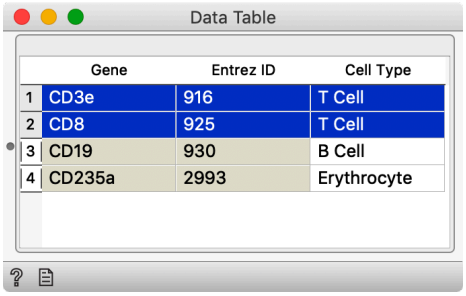


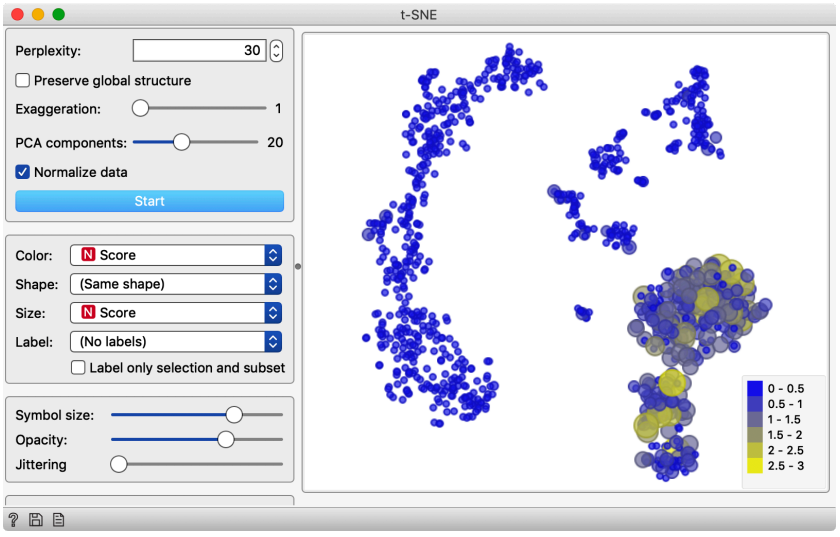
The output of the widget is a table that includes a gene name and cell type, both as specified in the input file, and Entrez ID.

The idea is now that we would select the gene(s) from the data table, and then score the cells according to the mean expression of selected genes. Widget *Score Cells* assigns a numerical score to each cell that is proportional to an average expression of the marker genes at the input of the widget. The score is added as a meta attribute to the cell data on the output of *Score Cells*. Check this using the *Data Table*! We can now feed this data into *t-SNE* and set the color and size of the points to the cell score.

We should be curious how *Score Cells* actually works and what is on its output. Use *Data Table* to check this out: connect it to the output of *Score Cells* and find the column Score.





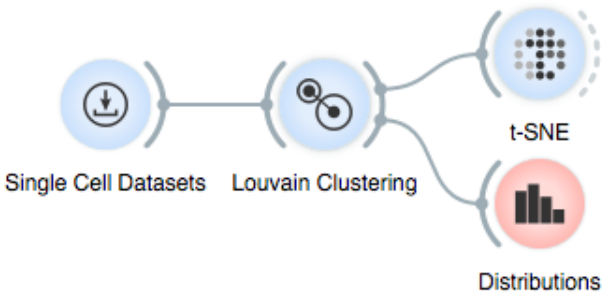


Notice that with any change in the selection of marker genes, we find a group of cells in t-SNE plot where these genes are expressed. Looks like T cells are in the bottom right cluster, B cells somewhere in the middle, and erythrocytes in the left cluster. Did we say cluster? Oh, we are not there yet...

Lesson 8: Cell Clustering

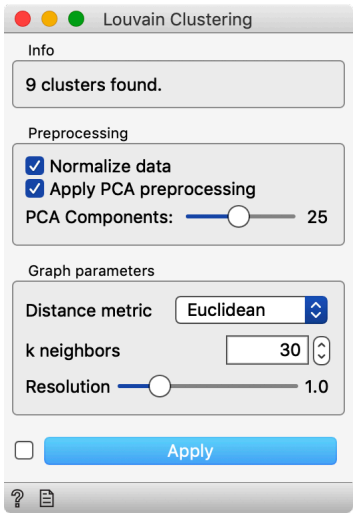
A typical task with a single cell data is to find clusters of cells. An advanced method that does this is Louvain clustering. Given the expression matrix, Louvain clustering creates a network of cells based on pairwise distances. Then, it searches for local communities — the parts of the network that are more strongly interconnected than expected by chance (think of friendships on social networks). Each local community is a cluster, and genes are assigned cluster labels accordingly.

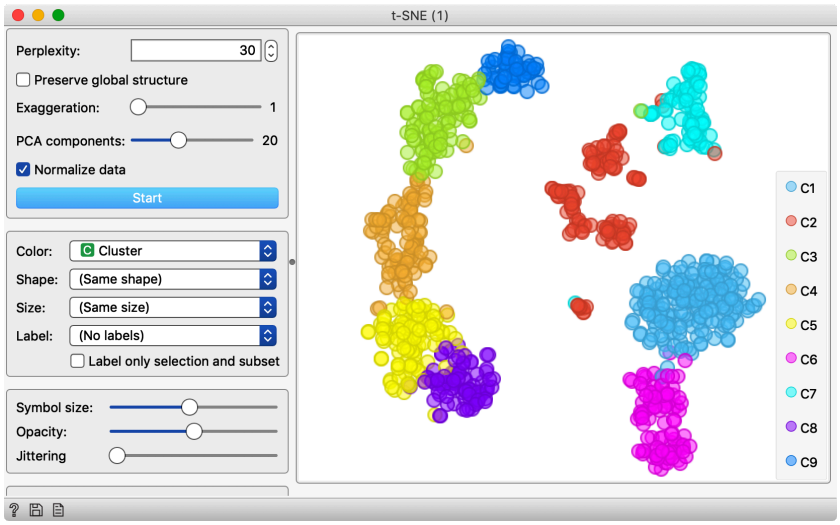
We will again use a sample from Bone marrow mononuclear cells with AML with 1000 cells that contain 1000 most variable genes.



In Orange, *Louvain clustering* is in its own widget that appends a column of cluster labels to the cell data. Quite neatly, the number of clusters is determined automatically. Let us construct a workflow that displays the results of the clustering in the t-SNE plot and that examines the frequency of the cells in each of the clusters. Let us observe the t-SNE plot first.

Louvain Clustering has a number of parameters. Here, we will stay with defaults, but you can experiment, change them and see the effect in t-SNE visualization.





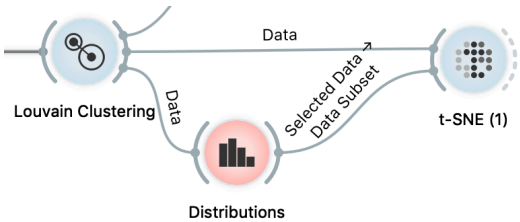
We have colored the points (cells) in t-SNE according to the cluster membership. Notice a nice separation of the clusters in the t-SNE plot. It looks like the cells are also well-separated in the original space of features (genes). A common mistake would be to compute the data projection first and then cluster the projected points. Obviously, then, the clusters would be separated perfectly and there would be no overlap.

We can now use *Distributions* widget to observe the frequency of the cells within each cluster.

Nice, the clusters are well represented and there is no need for any filtering at this stage. Notice also that some of the clusters mix healthy and diseased cells, and while this could be interesting, we will refrain to explore this aspect in this workshop.

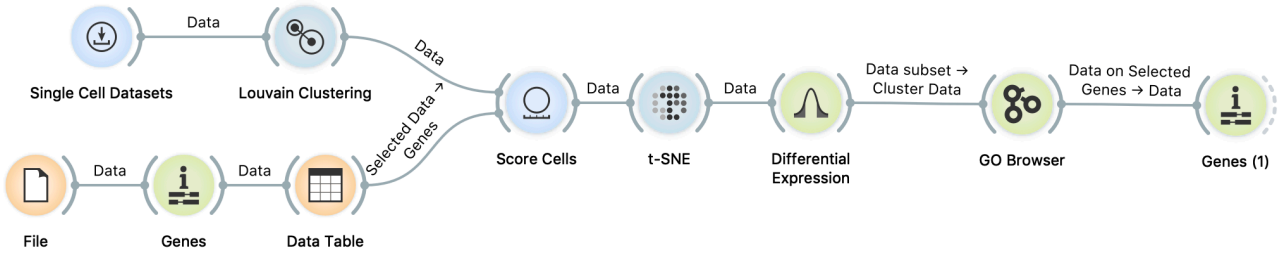
We now need to find out what are these clusters, that is, what types of cells are specific to the clusters. There are many ways to do this, and time permitting, we will explore the one that uses our marker genes and t-SNE visualization.

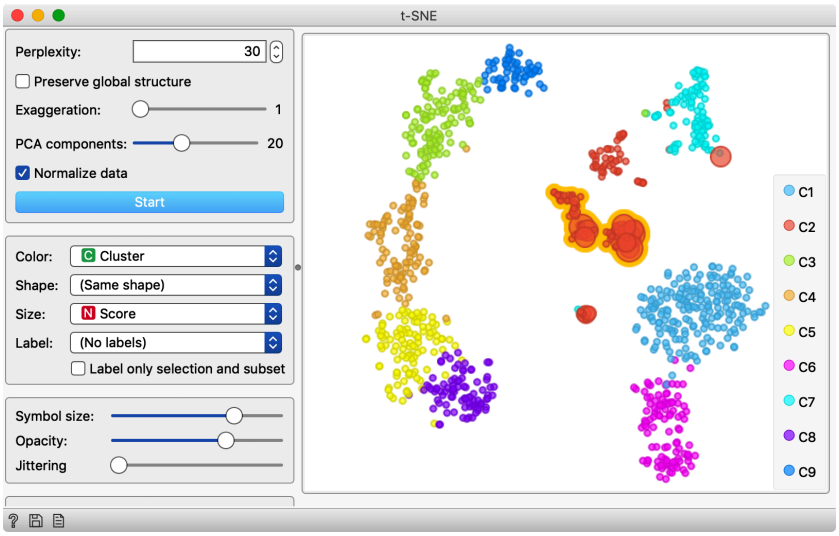
Most visualizations in Orange are interactive. In Distributions, you can click on the on the bar to select the associated data. Try connecting Distribution to t-SNE widget to explore where are the regions in the embedding space of each cluster.



Lesson 9: Cluster Exploration & Discovery of Markers

Here, we aim to discover "new" marker genes for B cells. We use quotes, of course, because it is likely that all markers for these cell types are already known. All we can do here is to rediscover some of them. But even that is great (at least for a computer scientist who is writing these notes). The workflow we will use is our most complex one so far.

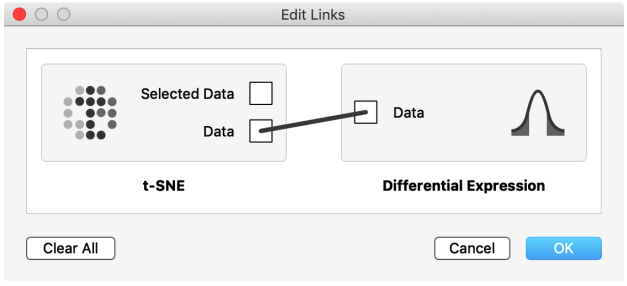




We are already familiar with its part until t-SNE. In *Data Table*, we select one marker we have for B cells, CD19. Then, in t-SNE, we select a subset of cells of the red cluster in the center of the graph.

We want to find genes that are expressed in the selected cells, but not expressed in all the other cells. We thus need to get all the data out from t-SNE, not just the selection and have a column that tells us if the cells were included in the selection. Whereas the default output of t-SNE is "Selected Data," the output called Data of the *t-SNE* widget has all the data required, and we should rewire the connection.

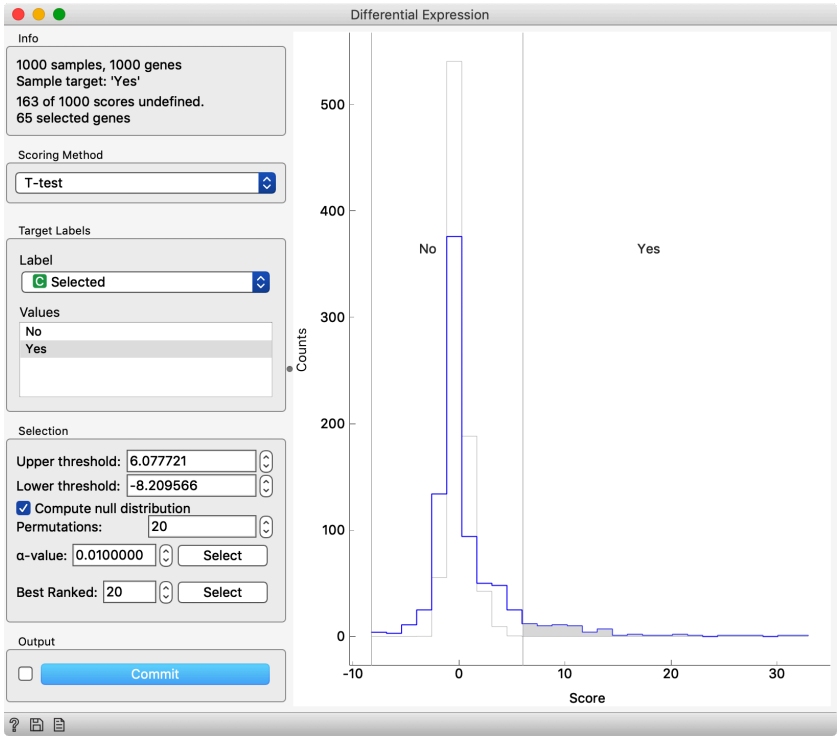
Double click on the connection between *t-SNE* and *Differential Expression* widget and instead of Selected Data, connect the Data channel of the t-SNE. You are



Differential expression shows the distribution of the differences of gene expression in selected and all other cells. We have to set this widget properly.

We have set the scoring method to *T-test,* set the *Label* to *Selected*, and marked that *Yes* is our target value. *Differential Expression* can also compares the observed distribution of changes to the null-distribution – the thin grey line – where data cells in each row are randomly permuted. Click on *Compute null distribution* to switch on the visualization of null-distribution.

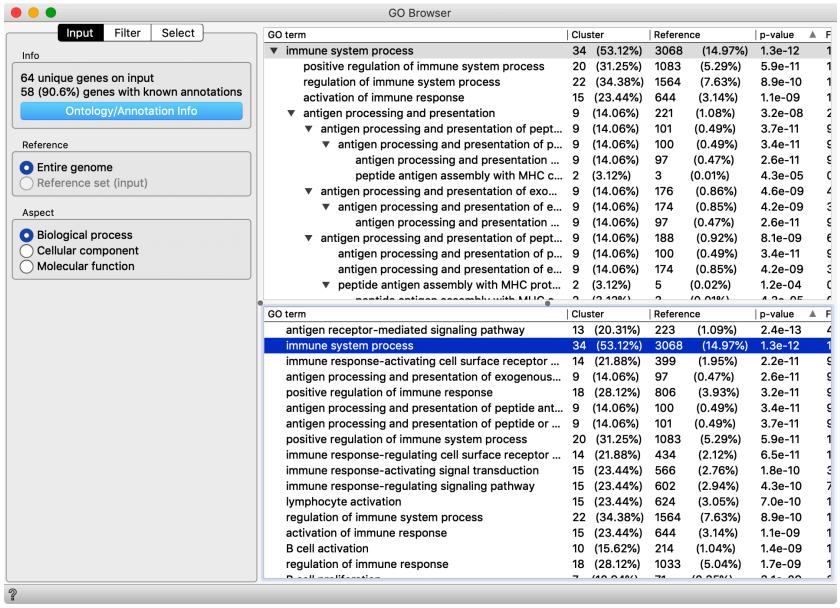
The distribution marked with grey line shows the *null distribution*, the distribution of genes scores under the arbitrary selection of target genes.



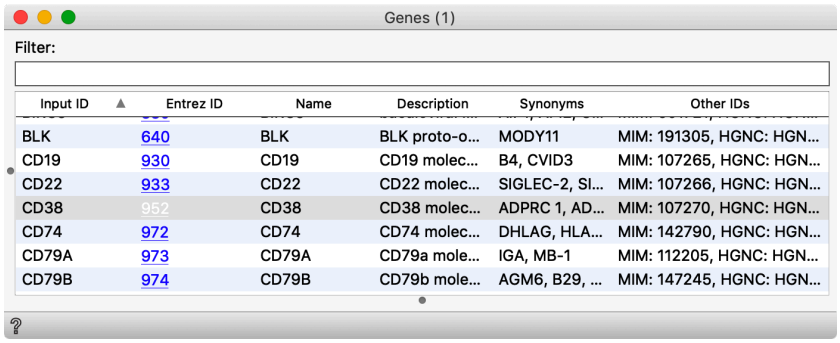
Differential Expression widget outputs the data with genes that are in extremes of the distribution. That is those, for which the difference in selected and non-selected cells is the largest. Genes that are most differentially expressed, lie on the left and on the right side of the two vertical splitters and their score value belongs to the shaded part of the distribution. Move the two vertical splitters such that there are only about 60 selected genes which are highly expressed in the selected group.

So, where are the genes that are selected in the *Differential Expression* widget? In the output of the widget. We can observe the output dataset and analyze the set of selected genes with widgets that we connected to *Differential Expression*. Observe the data in the *Data Table*, a list of selected genes in *Gene Info* and the results of analysis of Gene Ontology term enrichment in *GO Browser*. From all these choices, our workflow shows only the *GO Browser*, but you are welcome to explore other widgets as well.

In GO Browser, we find that our differentially expressed genes are characteristically present in several Gene Ontology terms. That is, several GO terms are enriched with our selection of genes.



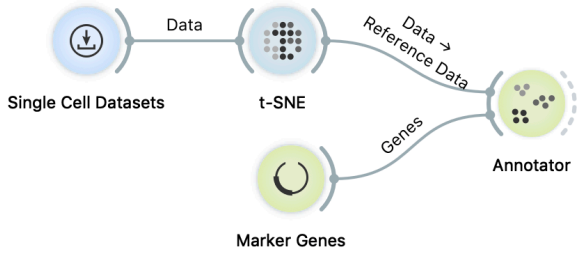
Interestingly, one of the most enriched terms is immune system process, with 34 genes from our differential expression set. Selecting this term, we get the data on these genes on the output of *GO Browser*, and we can check them out in the *Genes* widget.



Among the list of genes, there are also CD22 and CD38. “CD” stands for cluster of differentiation. Googling it, we find that two CD22 and CD38 are markers for B-cells. Oh, what a rediscovery!

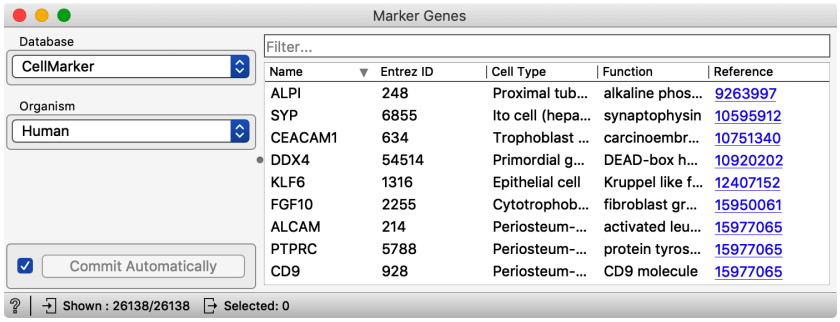
Lesson 10: Visual Annotation of Cell Types

There is a neat widget in orange that can analyze two-dimensional cell maps and can annotate the clusters in such visualizations. The widget is called *Annotator*, and it requires on the input the data set with two features that define the positions of the cells in two-dimensional embedding, and a set of marker genes with associated cell type. We use the following workflow to get all these.

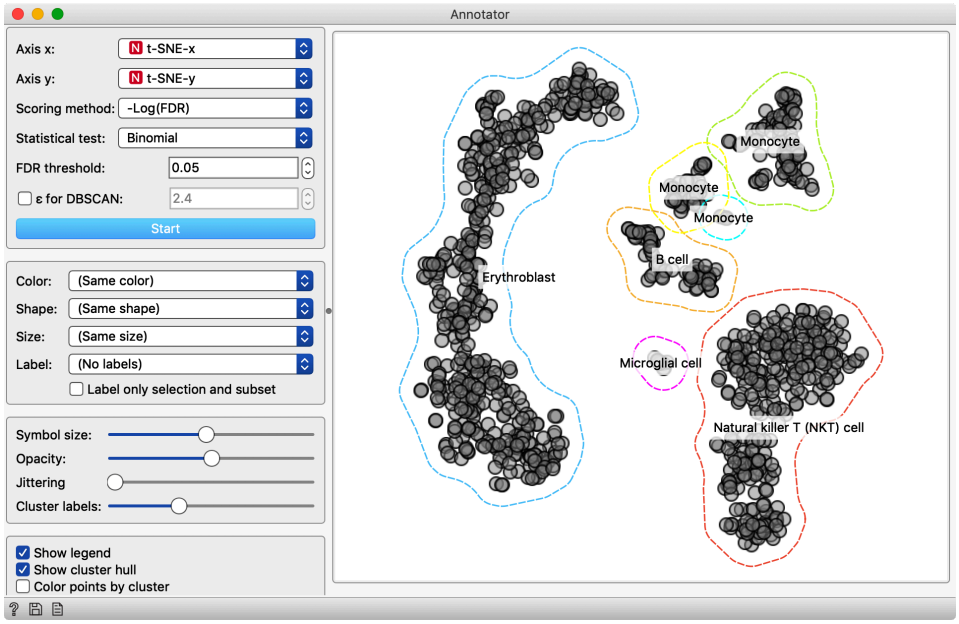


Some rewiring is required to get this workflow work right. Notice that *t-SNE*'s output channel is Data, and that *Marker Genes* output goes to Genes channel of the *Annotator*. Also, make sure that Marker Genes outputs the genes for human (not mouse, as

Marker Genes uses a large collection of markers from an open marker gene database. If nothing is selected, the widget will output all the genes from the list, just what we need at this stage.



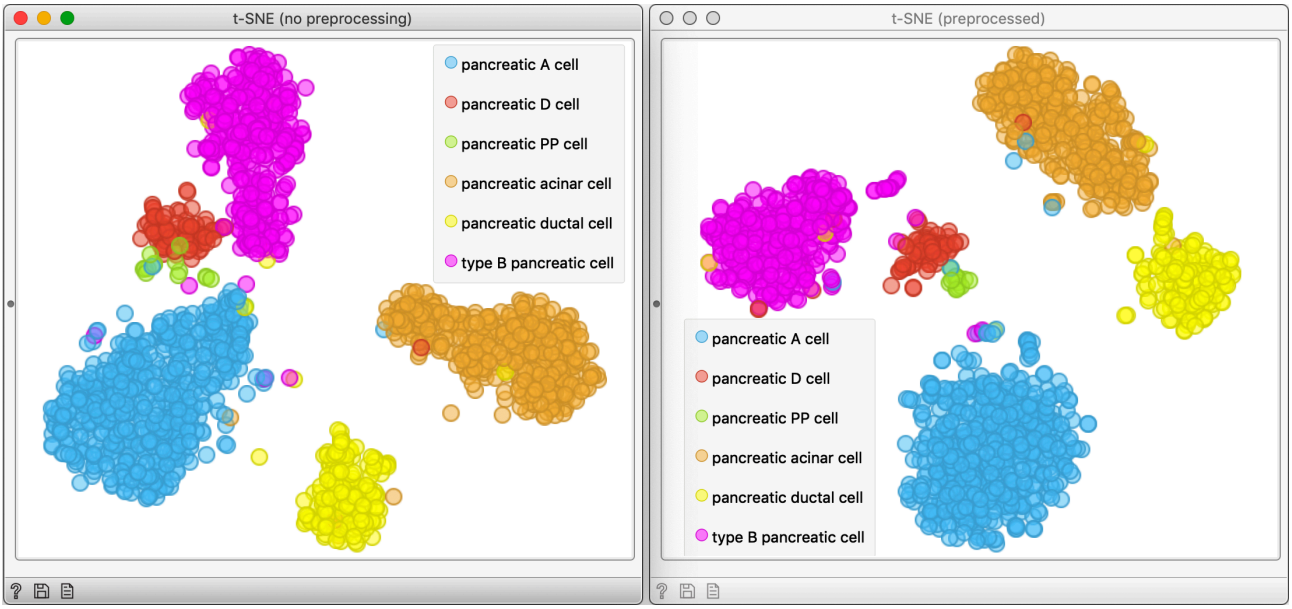
The Annotator has to be set to use t-SNE-x and t-SNE-y to position each cell, but once this is set, the display is cute and perhaps quite relevant.



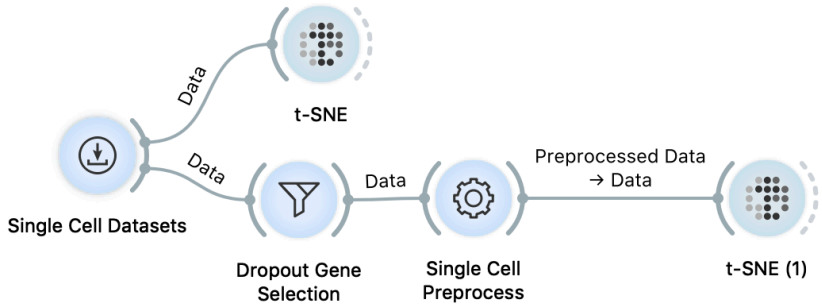
Lesson 11: Data Preprocessing

We here change the data set and use "Pancreas cells in human (sample)" from the *Single Cell Datasets* widget.

Consider the two t-SNE plots below. Both show a sample of human pancreas cells (Baron *et al.*, Cell Systems, 2016), but for the left t-SNE the input were raw count data, whereas for the right the data was preprocessed. The main difference between the two plots is that in the visualization of preprocessed data, pancreatic polypeptide cells got its own cluster. If you agree with us that this is substantial, read on. (If you disagree, read on just the same: we need the preprocessed in the coming lessons).



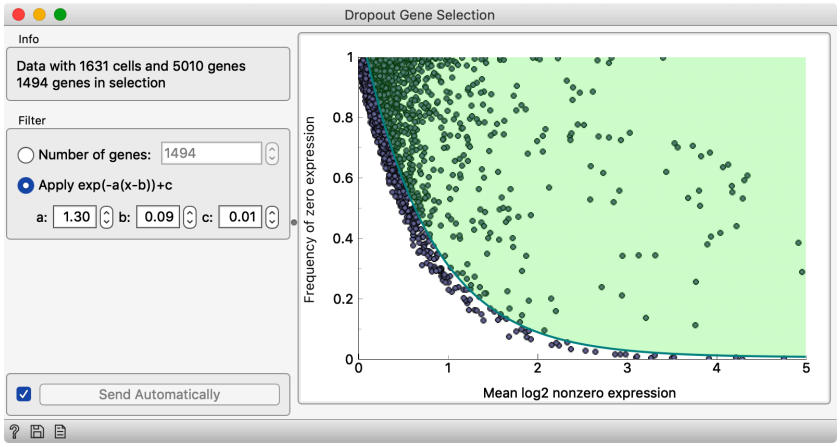
We will use two preprocessing steps. In the first one, we will select genes according to their dropout rates, and in the second we will use a quite standard pipeline for normalization and standardization of the count data.



Dropout gene selection approach was recently proposed by Kobak and Berens (biorxiv, 2018) and neatly select genes based on the

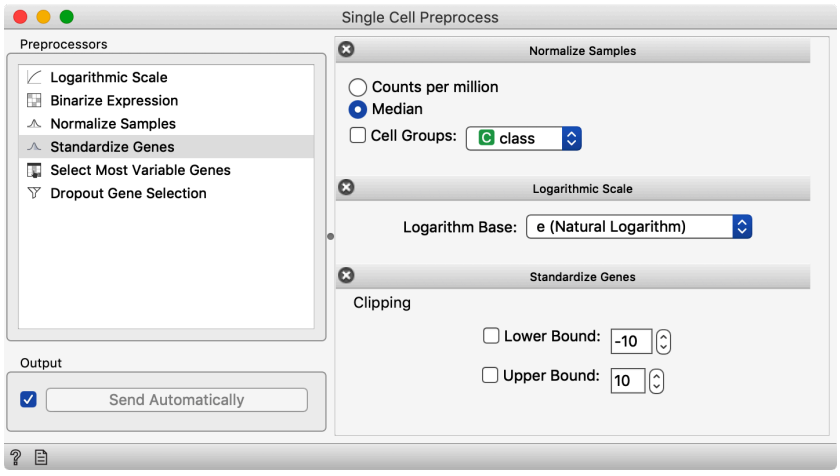
interplay of mean expression across the cells and and frequency of dropouts, that is, proportion of cells where gene was not expressed.

We have slightly changed the default parameters of Dropout Gene Selection to slightly alter the selection boundary.



Notice that in our data there are many genes with large dropout, where we get rid of those with low mean expression in cells where the genes were expressed.

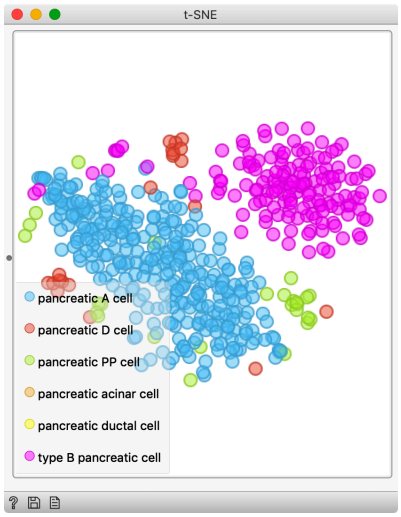
Single Cell Preprocess offers a set of standard preprocessing steps for the count data. We have configured it as shown.



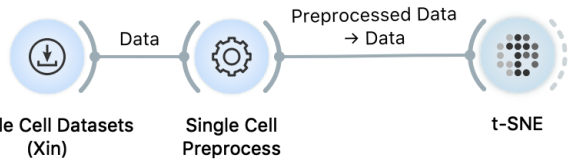
Notice that we first normalize the cell data with the median expression, transform the counts with the logarithm function, and standard the resulting expression to zero mean and standard deviation of 1.

Lesson 12: Batch Effects

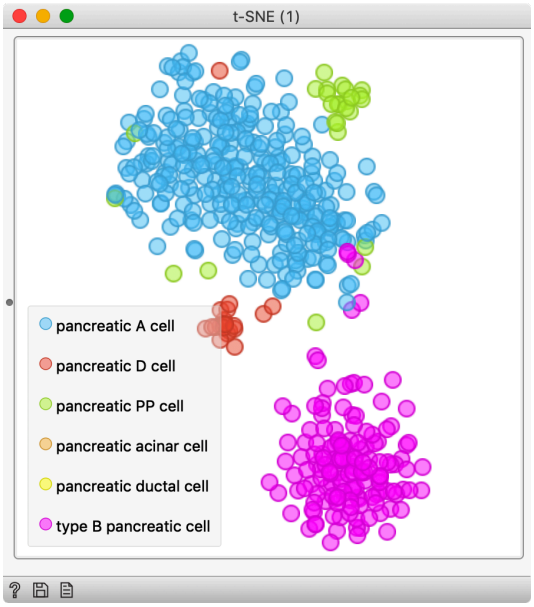
All our analysis so far considered the data from a single provider, or even single experiment. But in single cell analysis, we often deal with the data from several different source. With present batch effects, we often find that such data are not compatible and need to be aligned.

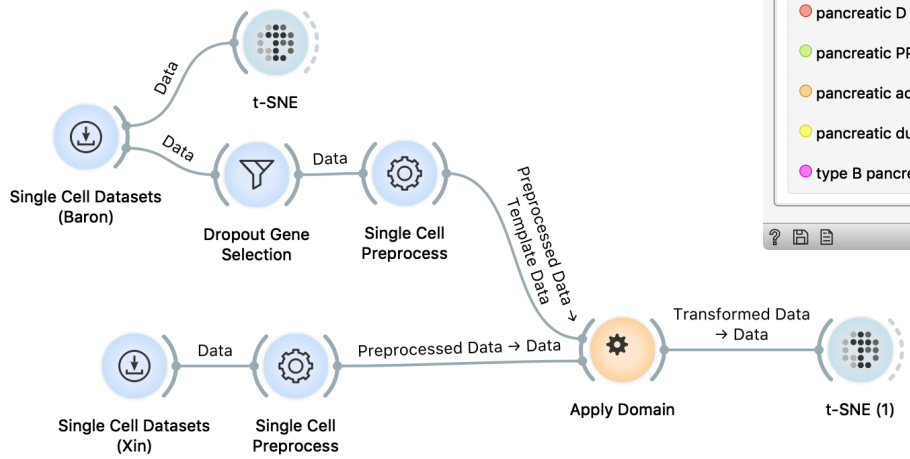


We will consider another data set from human pancreas cells (Xin *et al.*, Cell Metabolism, 2016), from where we have selected cells that share labels from the Baron et al. data from the previous lessons. Let us examine Xin et al. data alone first.

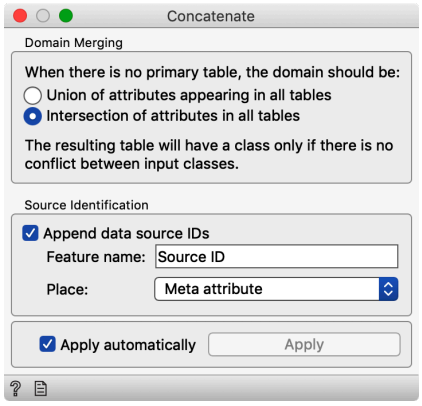


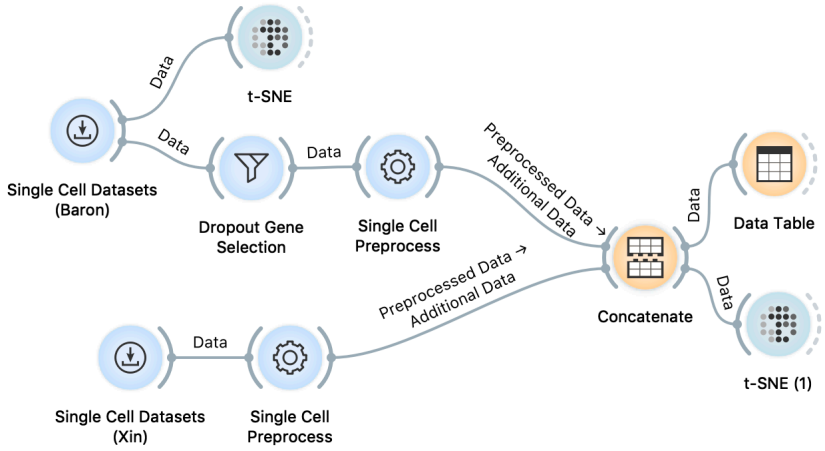
We have used the same preprocessing as introduces in the previous lesson. The t-SNE display is not as great, and the reason is that the gene selection is missing. With a widget called *Apply Domain* we can request Orange to select the genes from the post-processed Baron et al. data, and then get a much better t-SNE.



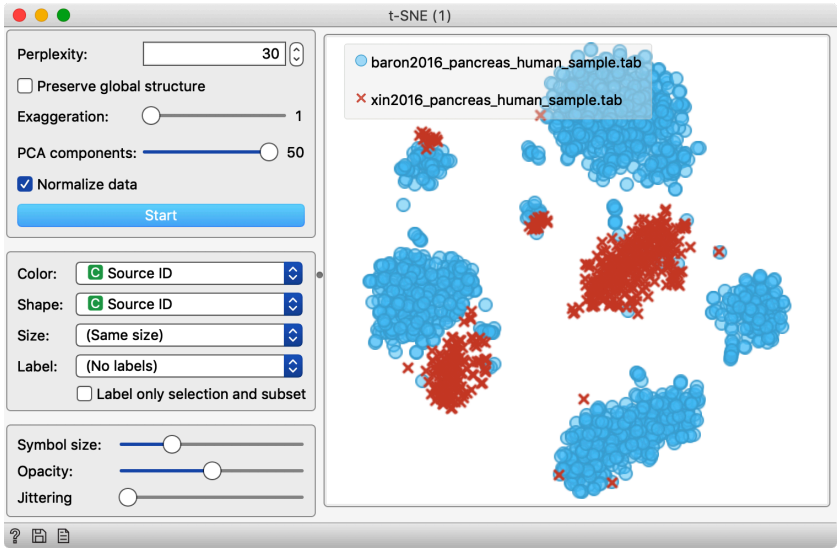


Now for the real thing. We will concatenate the two data sets, that is, the data from Baron at al. and the data from Xin et al., and plot a t-SNE. The workflow uses Concatenate widget, so that the Baron et al. data is used as a primary source, therefore removing all genes from Xin et al. that are not present in Baron et al. We also augment the data to include meta feature with the data set name.





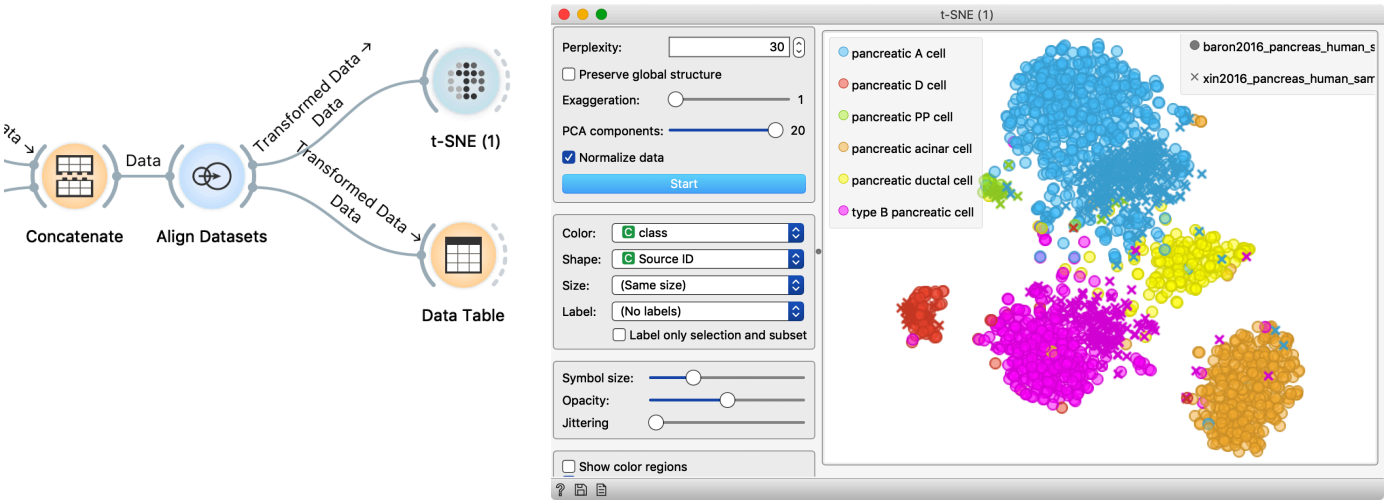
And the resulting t-SNE plot? Bad. The cells from one and the other data source don't mix and there is very little overlap. If we were to assign cell types of Xin et al. using t-SNE and join visualization with Baron et al., this procedure would fail.



Lesson 13: Dataset Alignment

There are different approaches to align data sets and remove batch effects. orange currently implements two, one through t-SNE visualization (see Policar et al., biorxiv 2019), and the other, more standard, through canonical correlation. This approach seeks the transformation of data so that the correlations of expression vector across the two data sets are maximized.

We will continue with the workflow from the previous lesson, use Align Datasets widget that implements canonical correlation analysis, and then plot the resulting data in the t-SNE.

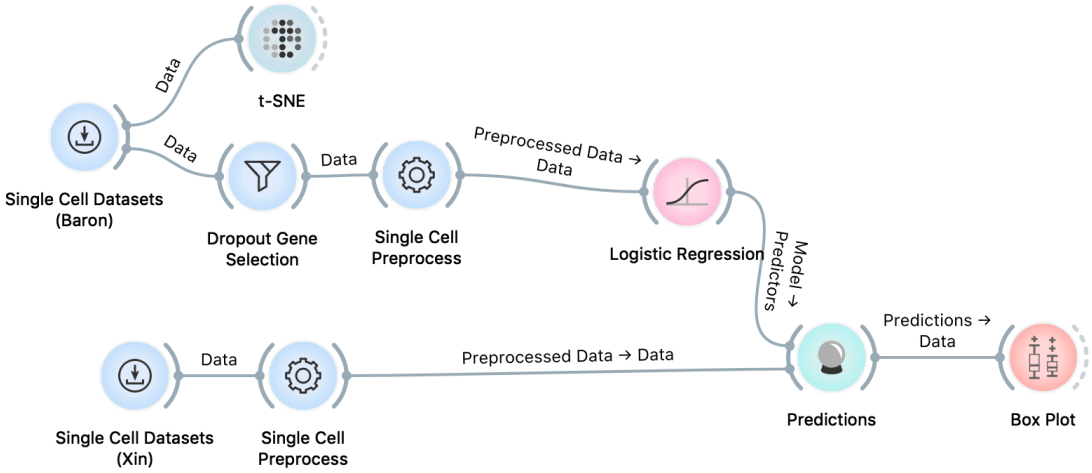


The cells from the two data sources are better aligned, and even more importantly, the cells of the same type but from a different data source cluster together.

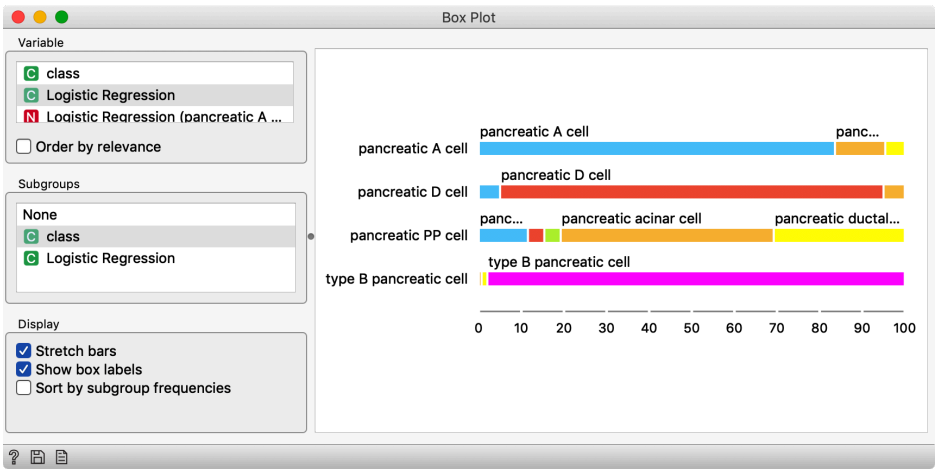
Lesson 14: Cell Type Classification

Our aim here is to use pancreas cell data from Baron et al., train a cell type classification model, and then for testing its accuracy apply it to data from Xin et al. We can perform such classification with and without data alignment, expecting, of course, that the removal of batch effects with alignment should lead to better classification accuracy.

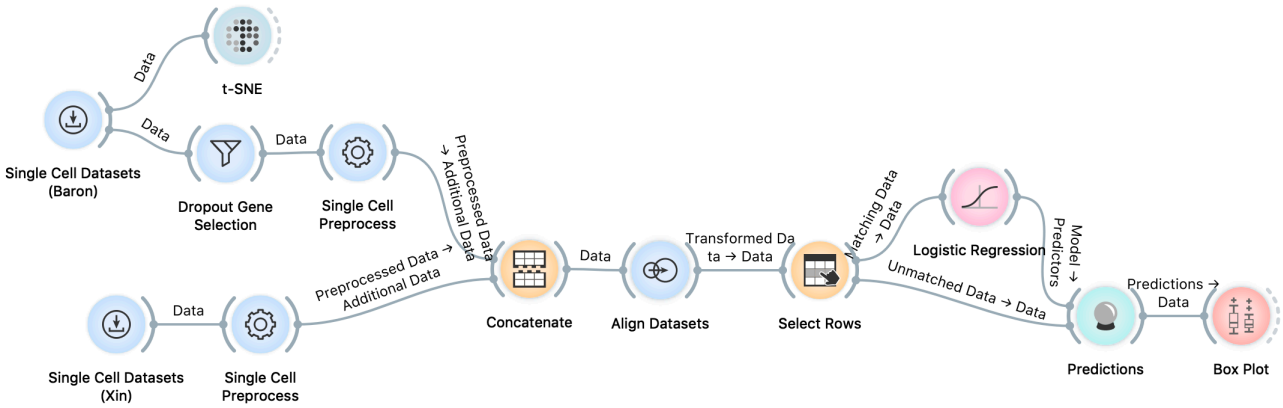
We start with a brute force approach with no alignment.

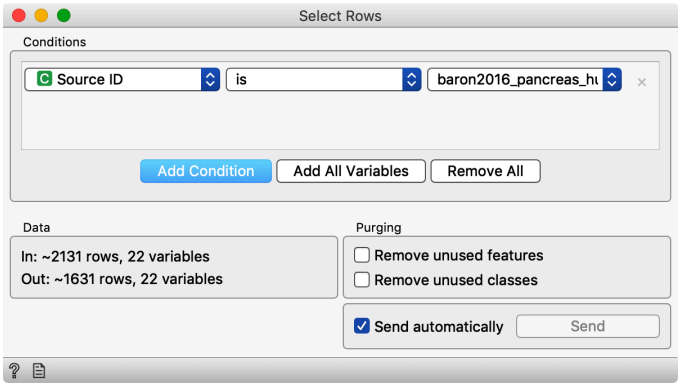


We use logistic regression here, as it performs well on this data set. The result is, despite low expectation, encouraging. Accuracy is high (F1=0.84, classification accuracy at 98.8%), but Box Plot with misclassifications showing misclassification of all pancreatic polypeptide cells.



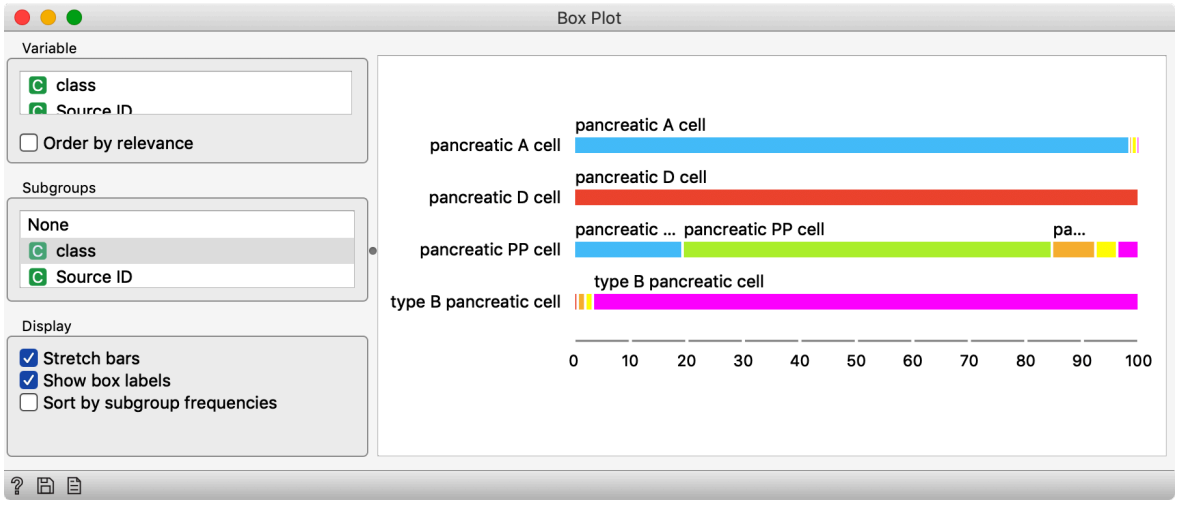
The workflow with alignment before classification is slightly more complex.





We needed to concatenate the data sets before alignment, so we need to split up the concatenated data set according to the source ID before training and testing. We use *Select Rows* widget for this task.

After the *Select Rows*, the selected data (Baron et al.) as sent to *Logistic Regression*, and the remaining data (Xin et al.) to *Predictions* to assess classification accuracy of constructed model. Box plot with misclassifications reveals substantial improvement of this classifier. Notice also good performance on pancreatic polypeptide cells. Data set alignment helped.



For the End

Our single-cell Orange Data Mining workshop ends here. We covered quite many topics, but just scratched the surface of a large field of single-cell analytics. This three-hour workshop, of course, was only an introduction. There are many advanced topics that we left out. These include imputation, other batch effect removal techniques, inference of cell development trajectories and pseudo-time, and cell tissue structure reconstruction. Our goal was to expose you to various aspects of data science and to show you how to do data analysis through interactive visualizations and construction of workflows. We hope you have enjoyed the course.