Processing of proteomics data in KNIME

Practical part manual

Kristína Gömöryová, David Potěšil

Brno 2019-11-20

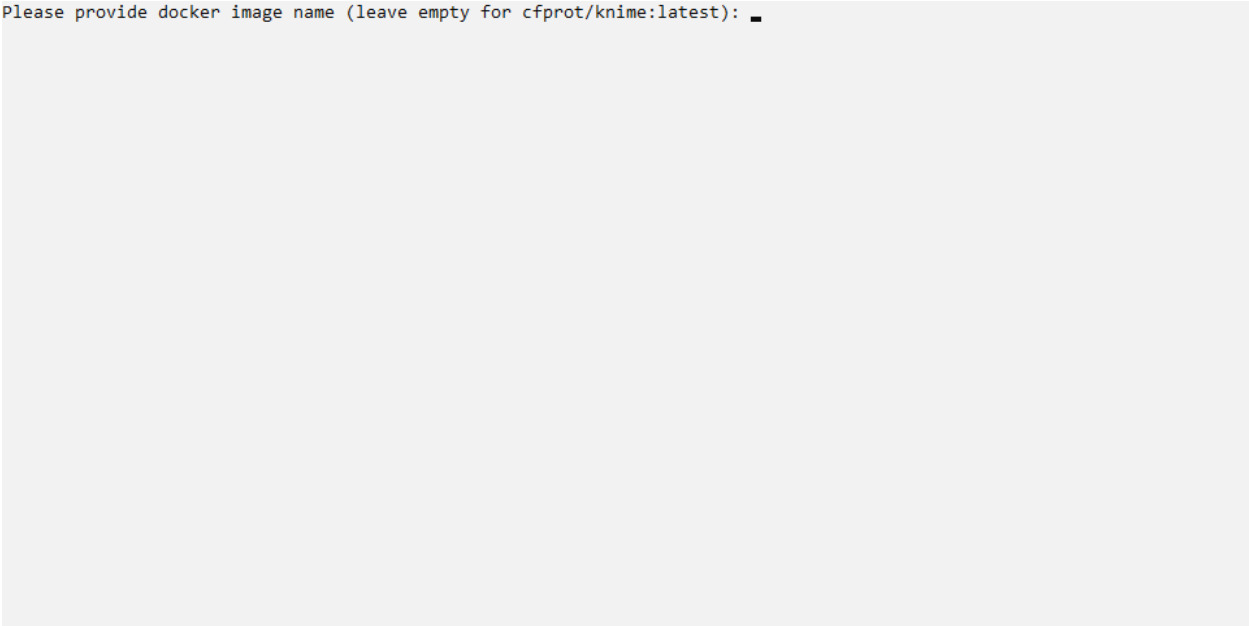
Contents

1 Environment setup

1.1 Start the docker container

Let's start the workshop with the docker container running KNIME. Please note that here we expect you to already have the script to start the container on your system adjusted to your settings, and you have prepared folder to be used as a KNIME workspace in the following steps. Alternatively, we are running the container for you and in that case, you may just try to check the situation on your colleague's screen.

Using the start_container script for your platform (Linux, Windows or Mac) start the docker container on your PC/Mac by running the script without any parameter (e.g. by double-clicking on it). This will show you a new window that will look like this:



We are going to use the latest version of the docker image so just confirm this by pressing Enter. You should get this on your screen:

```
Please provide docker image name (leave empty for cfprot/knime:latest):
Which port should the container run on? (e.g. 5901; leave empty for "5901"):
```

We are going to use again the default port to get inside the running container so leave this empty and confirm by Enter. You should get this on your screen:

```
Please provide docker image name (leave empty for cfprot/knime:latest):
Which port should the container run on? (e.g. 5901; leave empty for "5901"):
Name of the workspace to use - leave blank for "knime-workspace":
```

Now the script needs you to specify folder inside the KNIME workspaces folder specified in the script (e.g. “D:\knime-workspaces\” on Windows or "/Users/user_name/knime-workspaces/" on Mac). Enter only its name, e.g. “test” (without quotes). The screen should update to something like this:

```
Please provide docker image name (leave empty for cfprot/knime:latest):
Which port should the container run on? (e.g. 5901; leave empty for "5901"):
Name of the workspace to use - leave blank for "knime-workspace": test
Password:
```

Now the running container needs a password we will use later on in VNC viewer application. Please provide the password twice to verify it as well and you should get window like this:

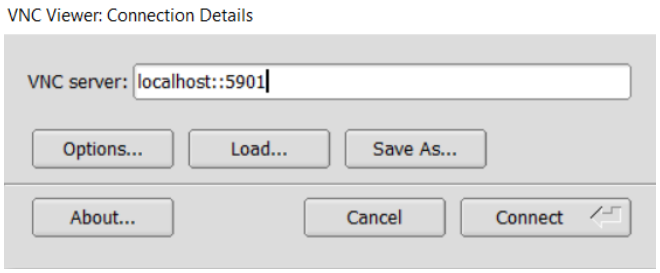
```
Please provide docker image name (leave empty for cfprot/knime:latest):
Which port should the container run on? (e.g. 5901; leave empty for "5901"):
Name of the workspace to use - leave blank for "knime-workspace": test
Password:
Verify:
Would you like to enter a view-only password (y/n)?
```

It is possible to set also view only password, but we will not need it now, so just hit N or n and confirm to skip this.

The window will now show some line from the container being started. You can either close this window or leave it opened. You should have now running docker container with KNIME and other applications inside on your own PC/Mac.

1.2 Accessing the KNIME workspace

Download the VNC viewer (https://bintray.com/tigervnc/stable/tigervnc) and open it:

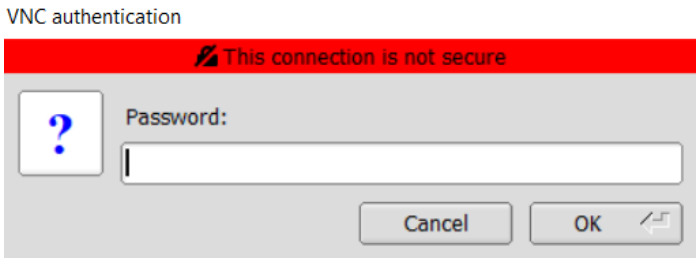


VNC server address that needs to be specified is as follows

- "IP address of the local/remote machine running the container"::"VNC server port number"

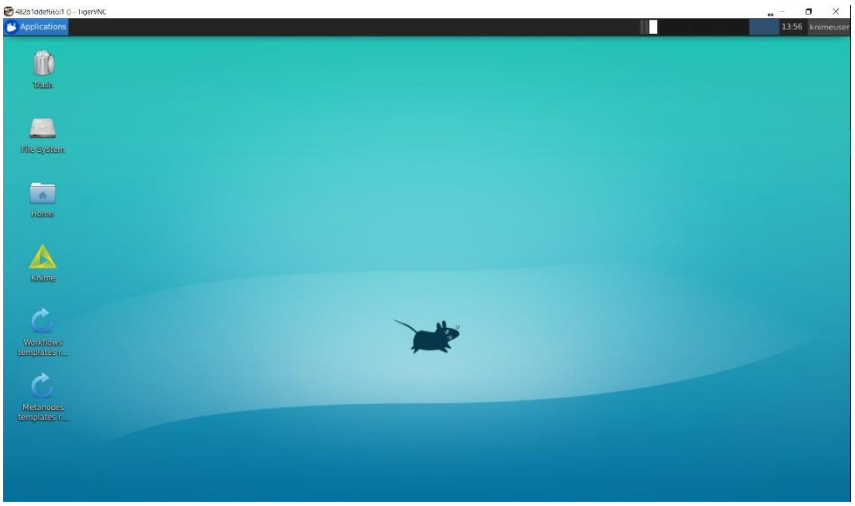
Example addresses are:

a) If you are running the local version, the address will be in format e.g. localhost::5901
b) If you are running the server version, the address will be in format e.g. 147.251.21.0::5901

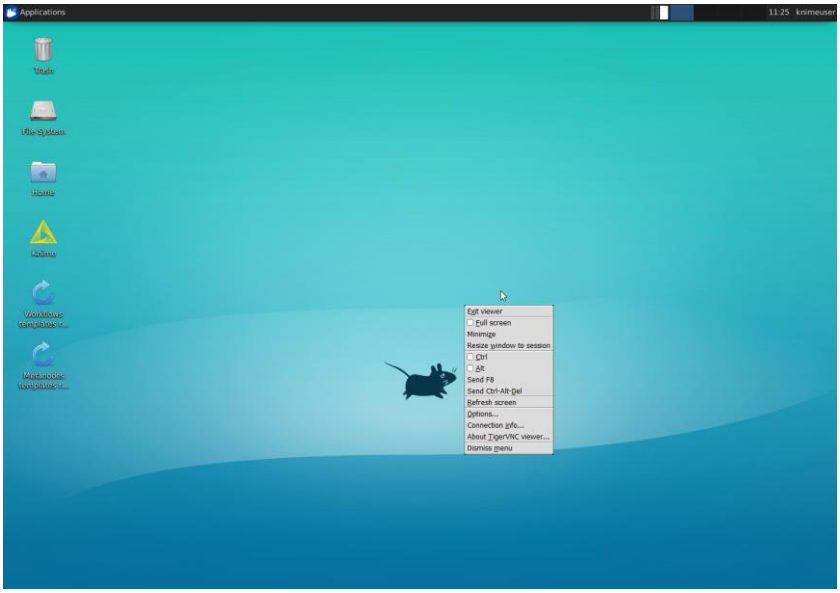


Provide the password you set when setting up the VNC connection. The information about insecure connection is not so critical – the current VNC viewer and server protocol is prone to man-in-the-middle types of attacks.

In case you provided correct VNC server address and password, you will successfully get to the KNIME workspace.

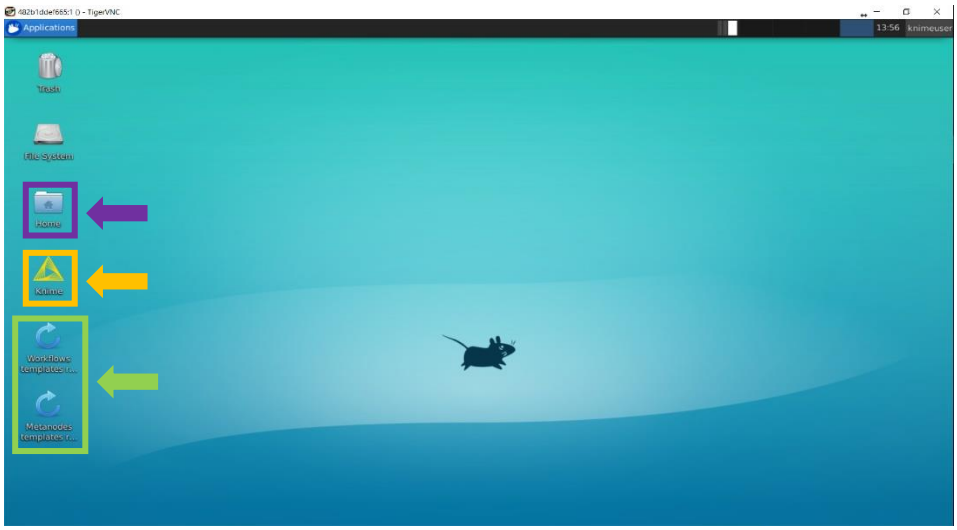


The container screen size should adjust based on the size of your window in your system. We recommend to maximize the VNC viewer window so you will see whole screen of the container. You can also run the VNC viewer window in full screen mode or on multiple screens. Press F8 to bring up the menu that will enable you to do so (option “Full screen” and also other things if needed).

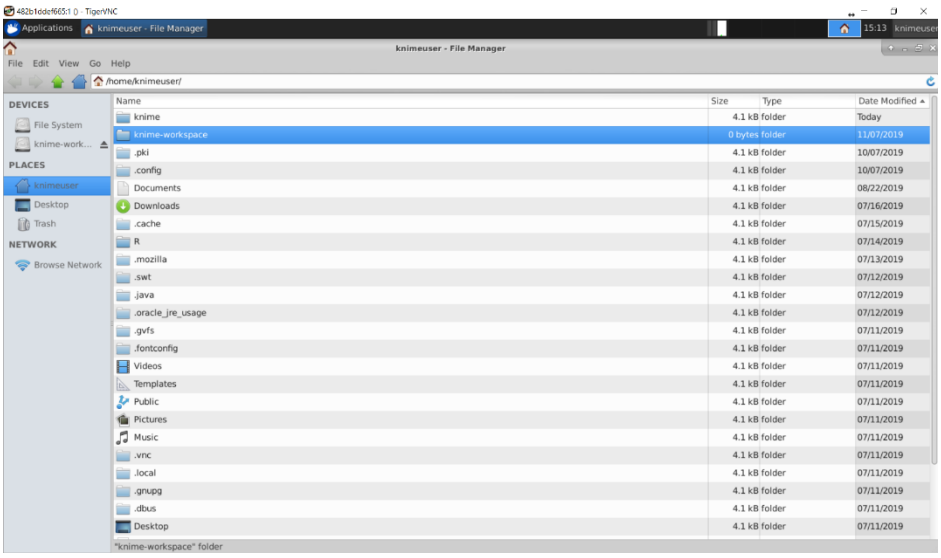


KNIME application itself is marked by the yellow triangle, on double-click you will be able to open it. Use **Workflows templates reset** and **Metanodes templates reset** icons for getting the most actual versions of workflows and metanodes from github (https://github.com/OmicsWorkflows/KNIME_metanodes and https://github.com/OmicsWorkflows/KNIME_workflows). Double-click the respective icons and wait till

the message that the templates reset was finished (for more detail explanation refer please to the chapter 9)



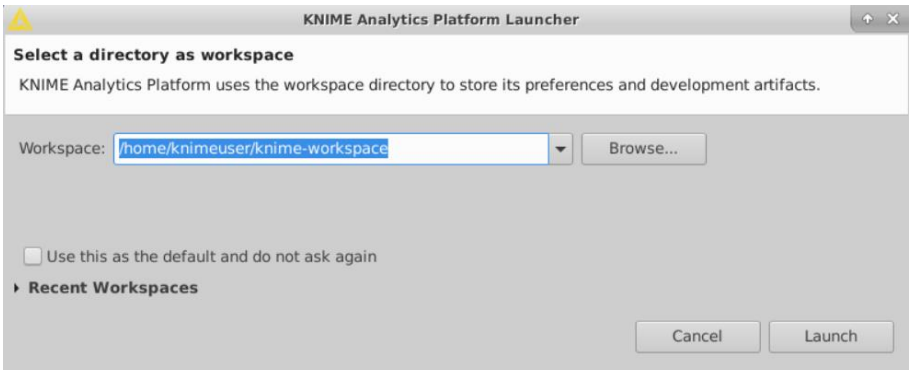
Via **Home** icon you will be able to get to the folder system. Knime-workspaces subfolder mirrors the shared drive on your computer, so use this one for the file transfer.



2 Basic KNIME features

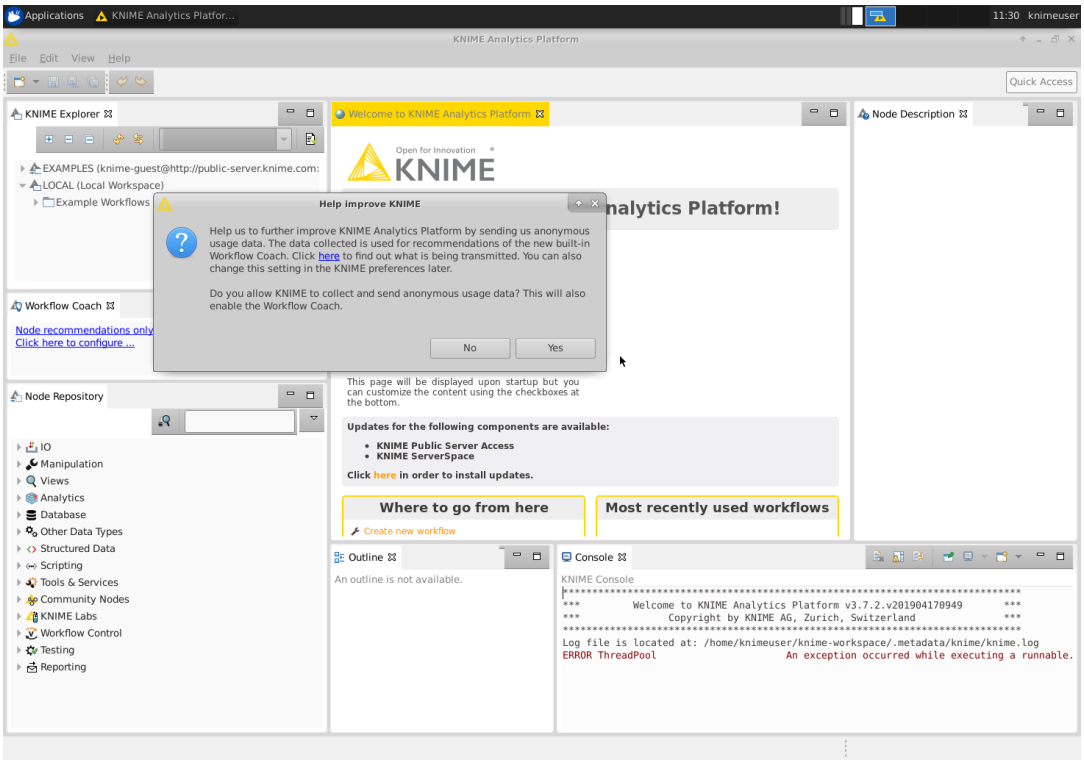
2.1 Orientation in KNIME application

Double-click the KNIME icon and when asked to select a directory as a workspace, just click “Launch”.

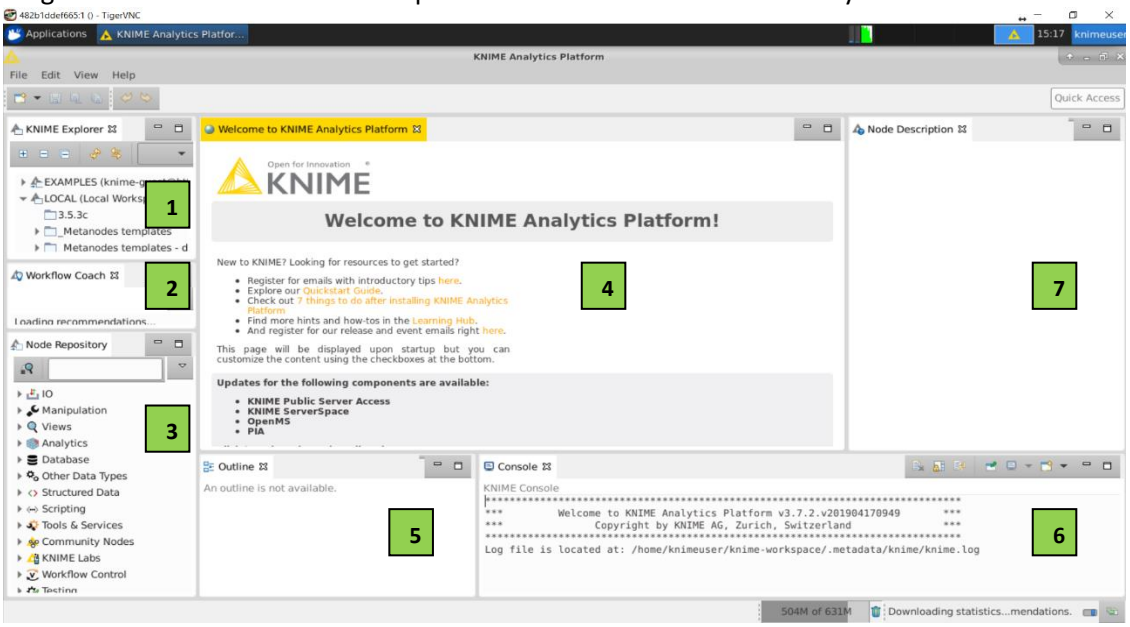


- you can check “Use this as the default and do not ask again” option not to be asked next time you lunch KNIME inside the currently running container

First time you run KNIME in empty workspace (i.e. empty folder on your system you have selected during the container start) there will be two folders created (“.metadata” – hidden on some systems, and “Example Workflows”). You will be also asked whether to share your usage statistics, feel free to respond as you like. If you enable it, the KNIME application will provide you some hints on what other users are using via the Workflow coach.



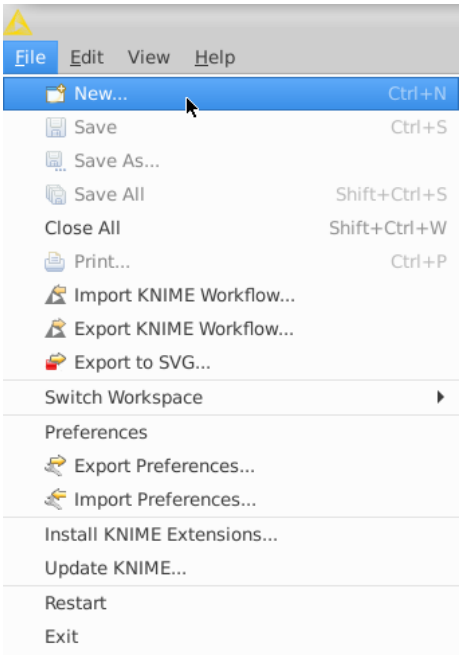
You will get to the default KNIME workspace. Let's describe now it's basic layout:

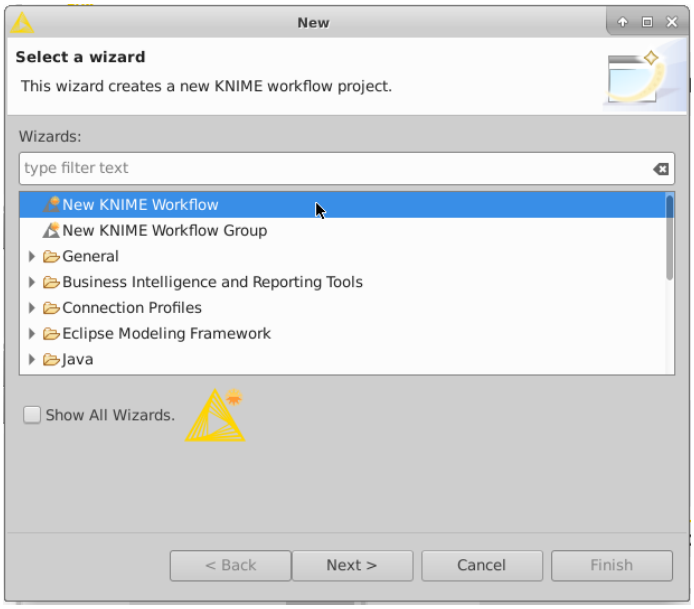


1	KNIME Explorer	*Overview of all available workflows and folders in knime-workspaces*
2	Workflow Coach	*Nodes recommendations based on KNIME users usage*
3	Node Repository	*Repository of in-built KNIME nodes (and extensions)*
4	Workflow editor	*Canvas for editing the currently active workflow*
5	Outline	*Overview of the currently active workflow*
6	Console	*Shows execution messages*
7	Node Description	*Description of particular nodes (after clicking any node in the active workflow or in the node repository)*

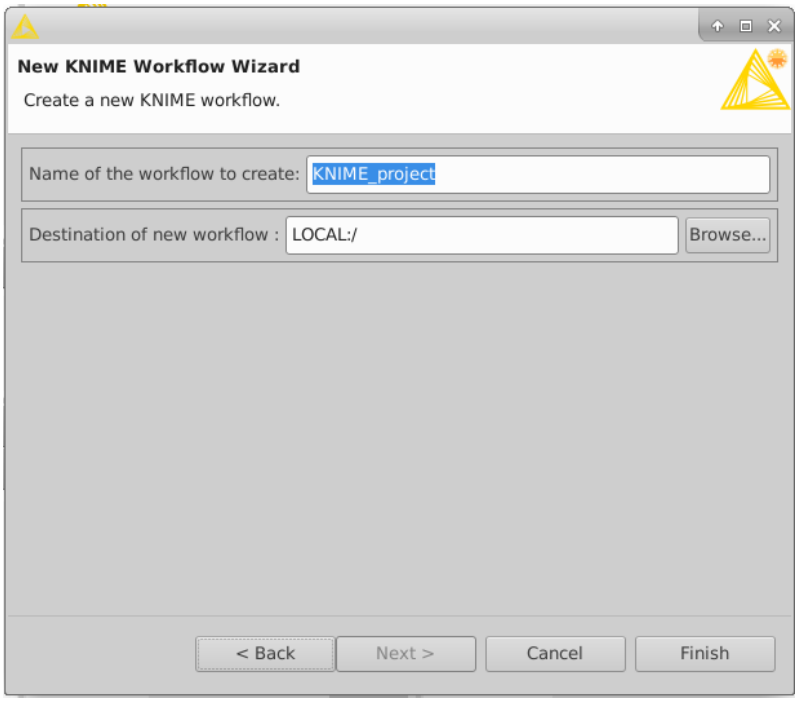
2.2 First workflow creation

Navigate yourself into: File -> New -> New KNIME workflow

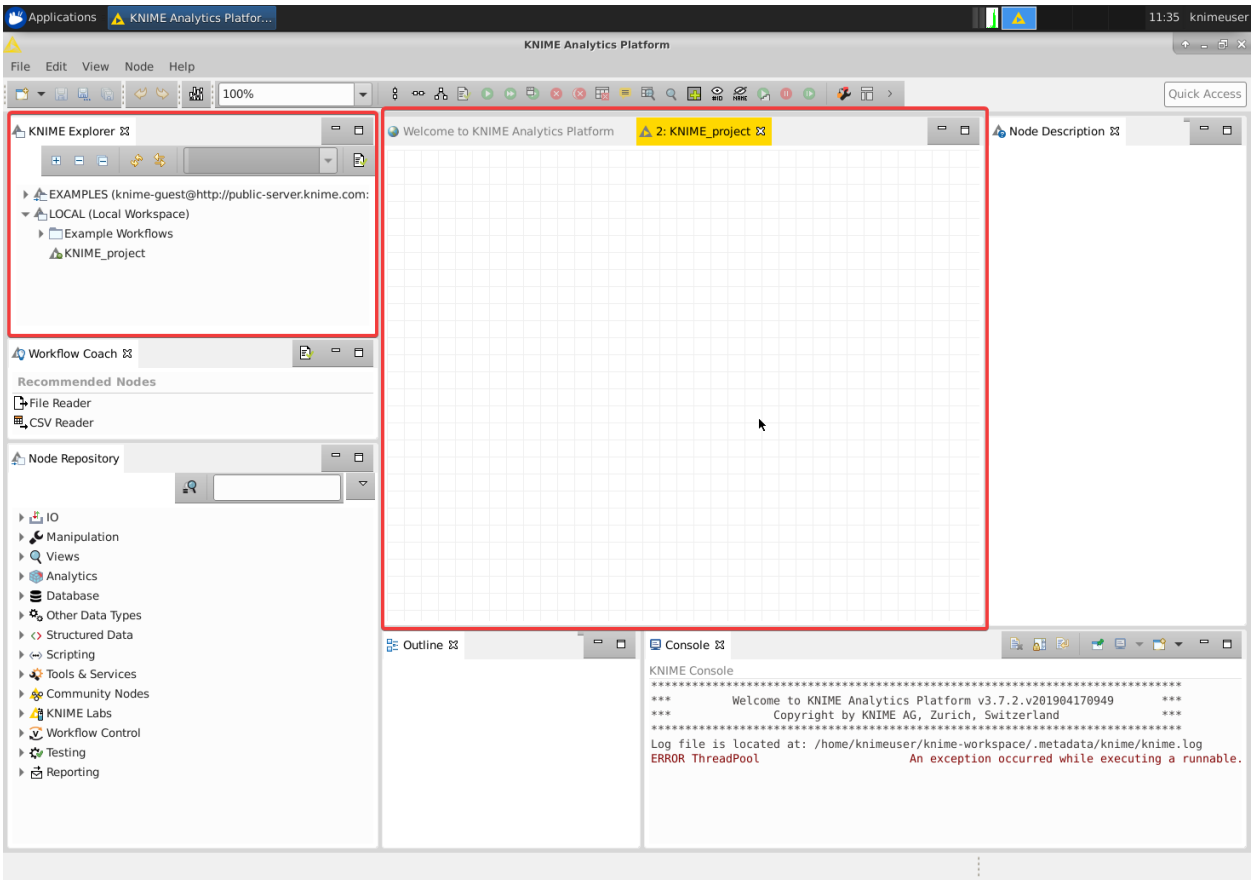




Provide the name of workflow you will be building and its destination (within the knime-workspaces folder) and click “Finish”.



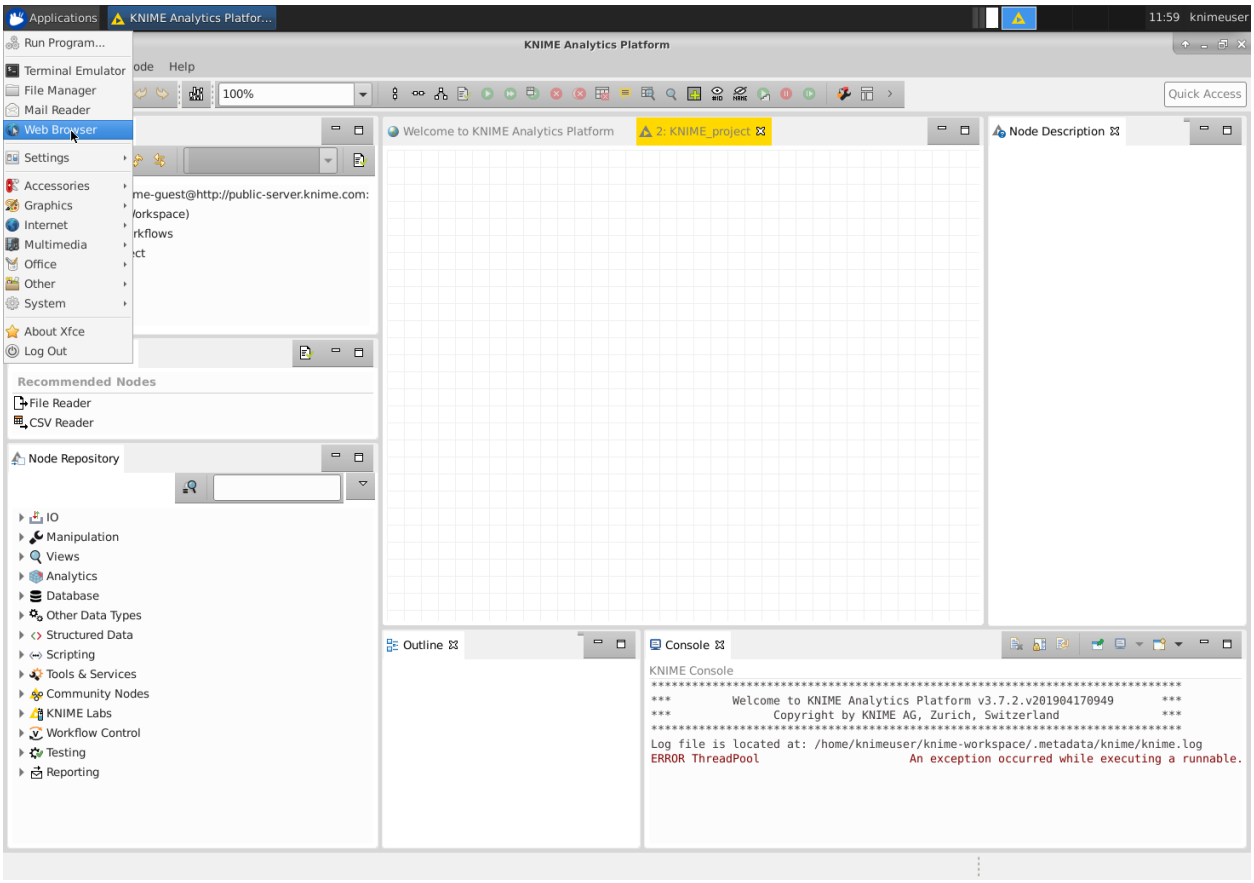
In the Workflow editor part, a new blank screen will be opened and folder for the given workflow will be created and you may notice the workflow being visible in your workspace as well.



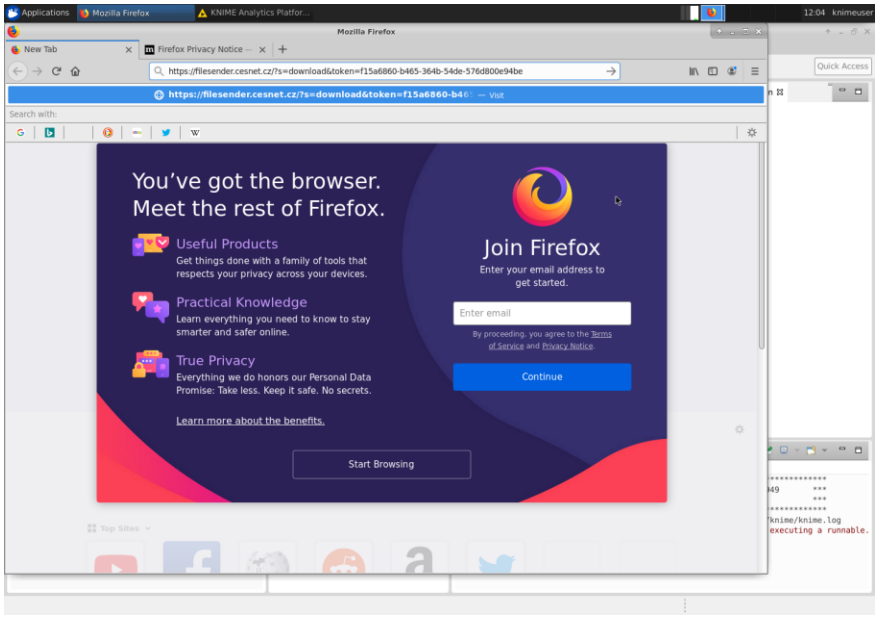
2.3 Getting example proteinGroups.txt file into KNIME

Prior we start with the workflow building, copy the proteinGroups.txt file (https://github.com/OmicsWorkflows/Workshops/tree/master/workshop_2019/files/proteinGroups.txt) into the knime-workspaces folder. This file, the output of the MaxQuant software we have modified for the purposes of the workshop, will be used for the further processing.

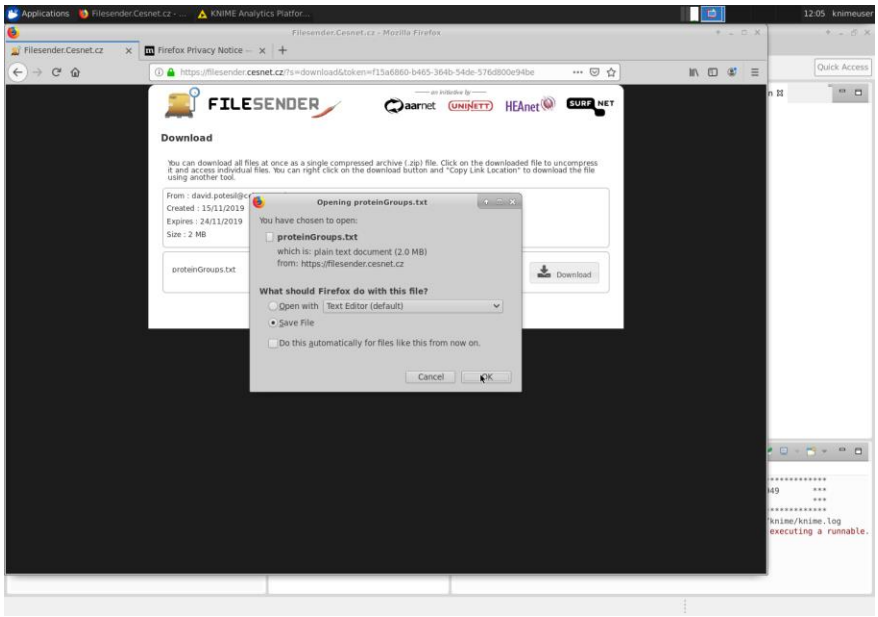
You can open the link directly in your running container. Navigate to the “Applications” button in the left top corner of the container screen and select Web browser.



Now you can copy (Ctrl+C) the link to the file into your clipboard on your own system (e.g. Windows or Mac) and paste it directly into the running container Firefox window. This clipboard sharing works both ways, so you can transfer text this way in both directions. You should see something like this:

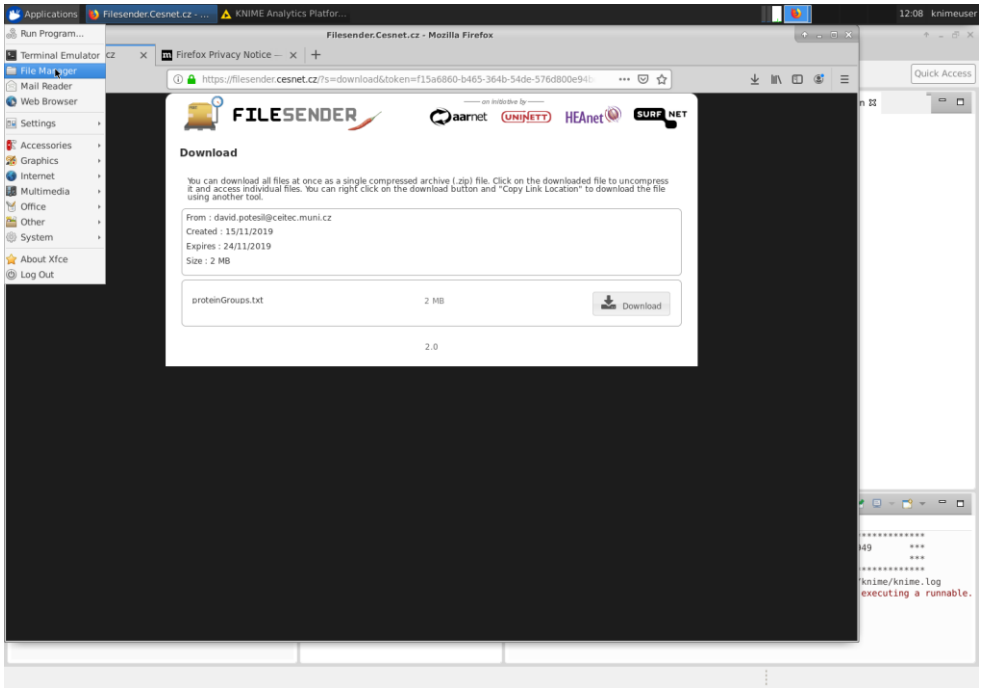


Confirm the link and download the file into your container hard drive by selecting "Save File":

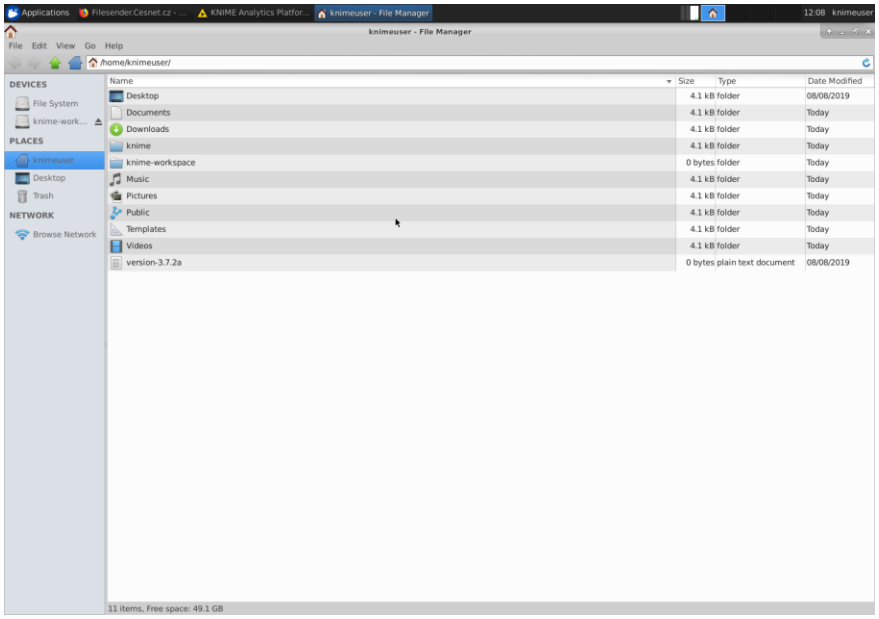


By default, the file is saved into the Downloads folder inside the container. This folder is not inside the KNIME workspace that is accessible by KNIME, so we need to move the file into the KNIME workspace inside the running container.

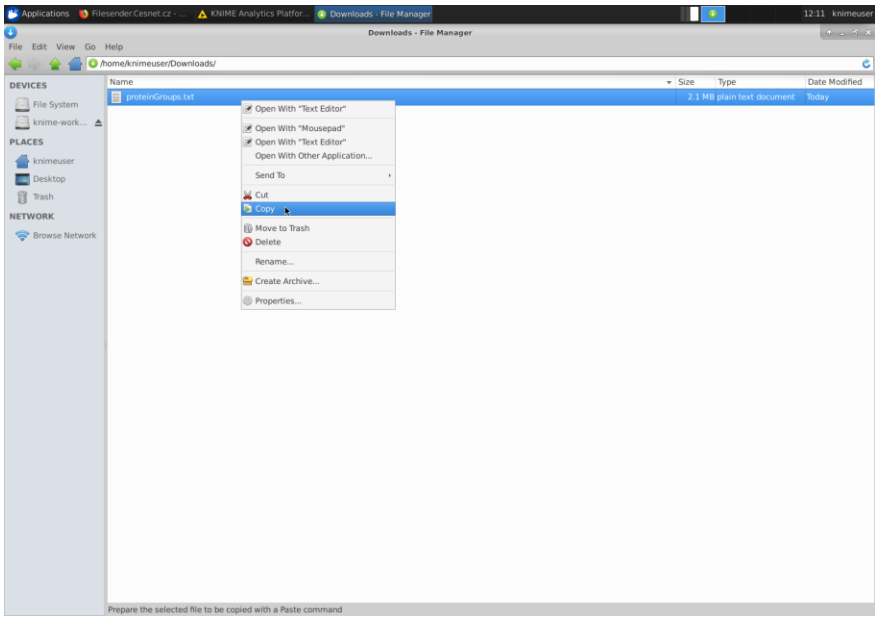
Please go to Applications icon and select File Manager this time:



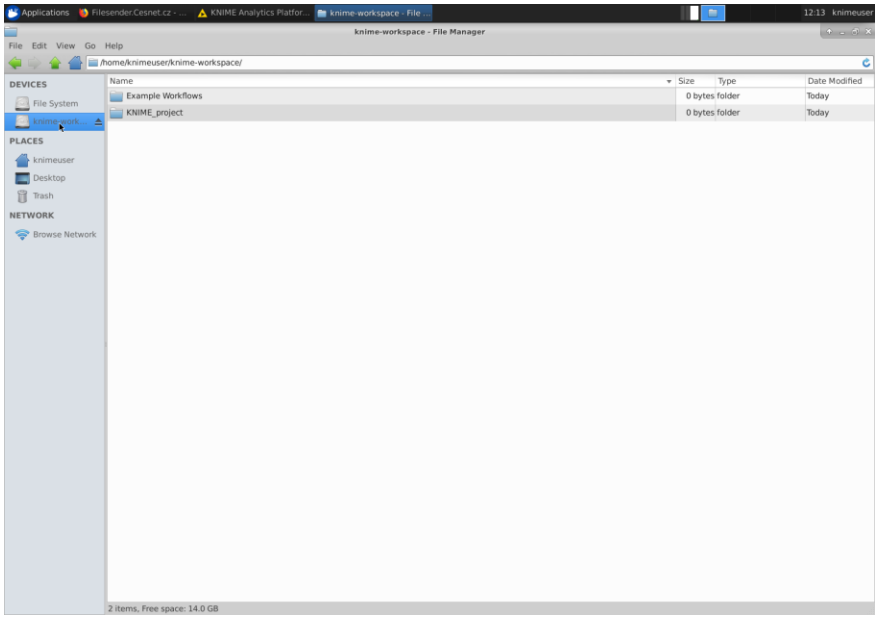
You should get the file manager window displayed.



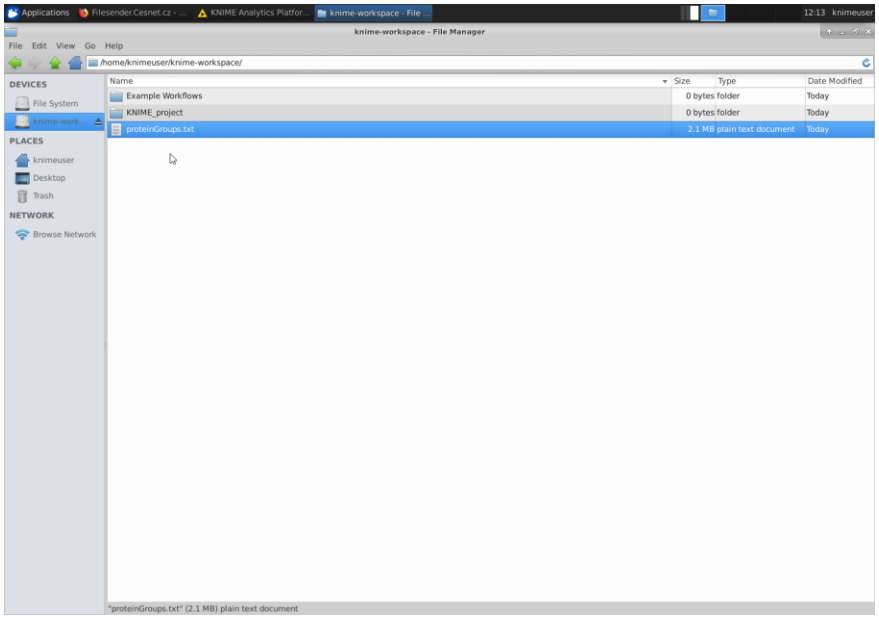
Navigate to the Downloads folder and copy (Ctrl+C) the file into your clipboard inside the container. Alternatively, you can also click on the file by your right mouse button and select “Copy”:



Navigate into your KNIME workspace folder inside the container by clicking onto “knime-work...” device in the upper left corner of the file manager window:

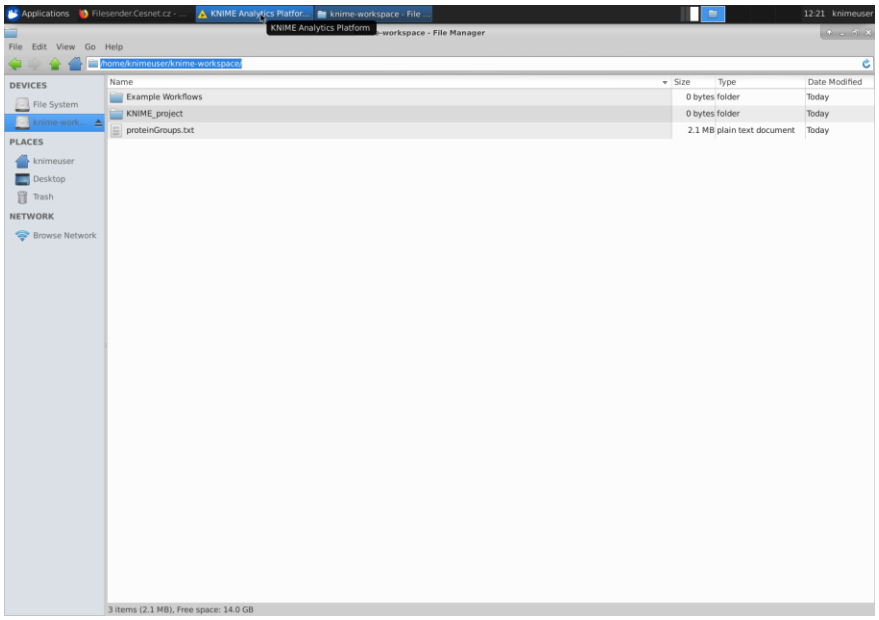


Press Ctrl+V or right-click -> Paste to get the file into this folder. You should now see the situation like this:

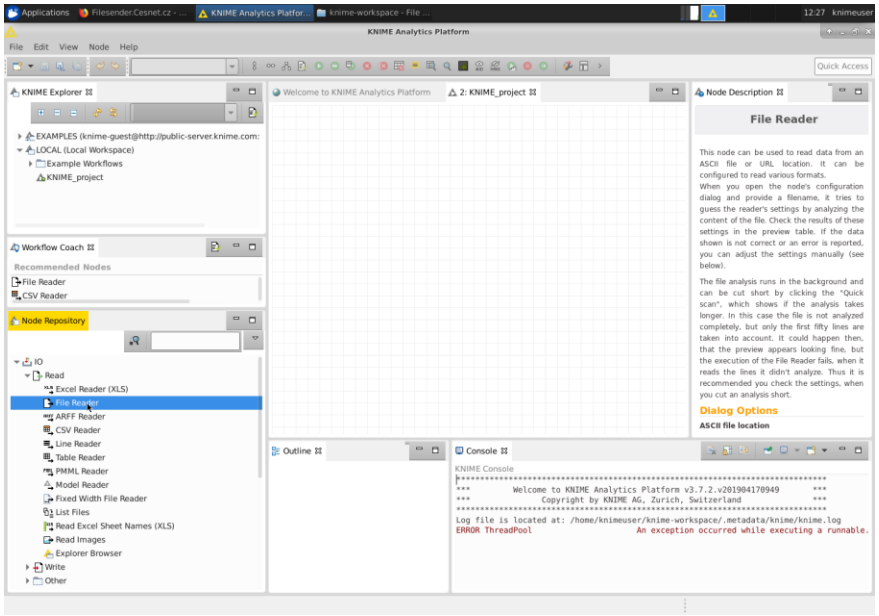


Please note that KNIME workspace folder inside the container (/home/knimeuser/knime-workspace/) is the only place where you should store any data you do not want to lose from the running container. In case the container needs to be started again (e.g. after your system restart) the container will look and behave the same way as before because it was built from scratch using the docker image specified during its start. The same applies to any other changes of the system like Firefox settings and installed applications and/or R packages. This is desired feature that makes the environment reproducible – not to change any container settings that might influence the results of the data processing in KNIME.

Now we can finally start with the workflow building itself! Navigate to the KNIME application window in the container by clicking onto it in the top panel:



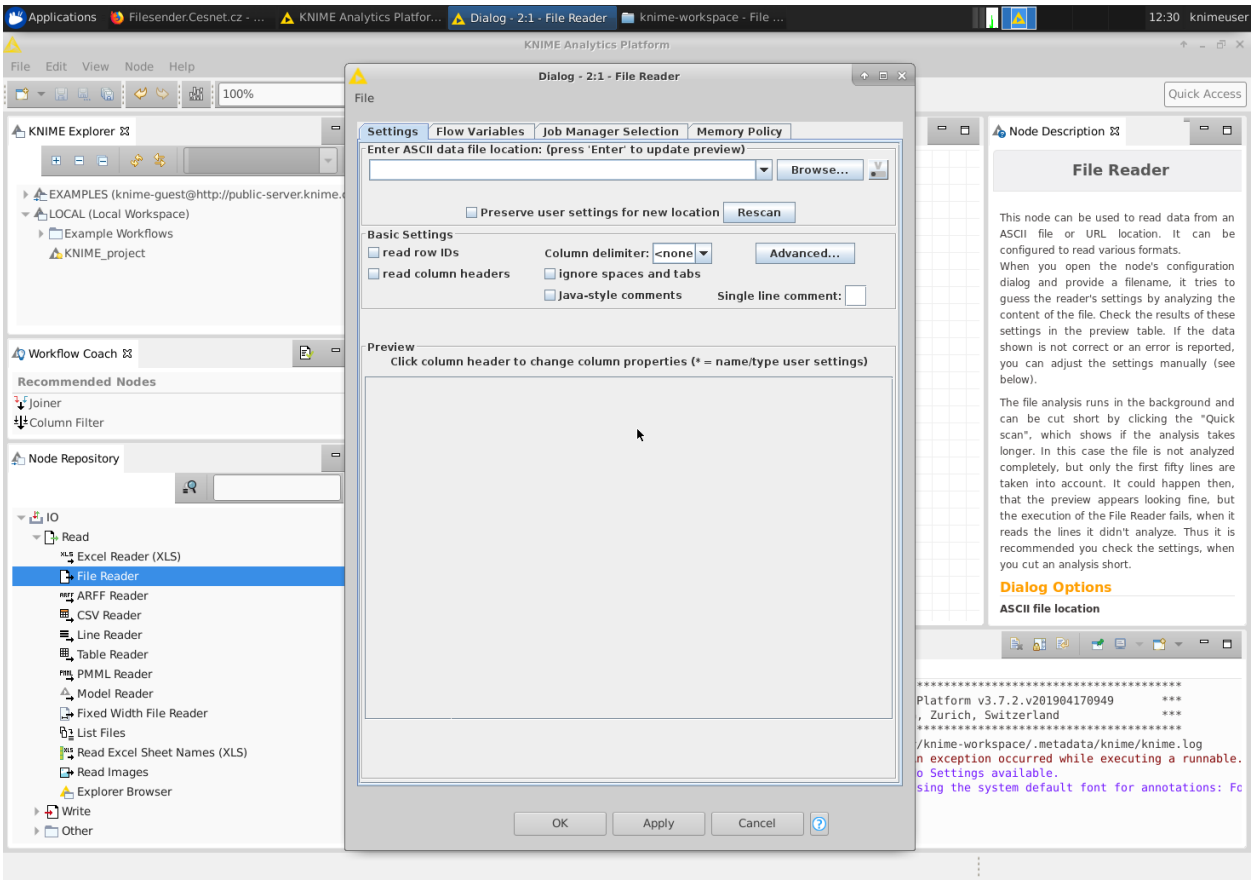
We will start with reading the **proteinGroups.txt** file. Navigate yourself into Node Repository -> IO -> Read -> **File Reader** and select the node to get window like this:



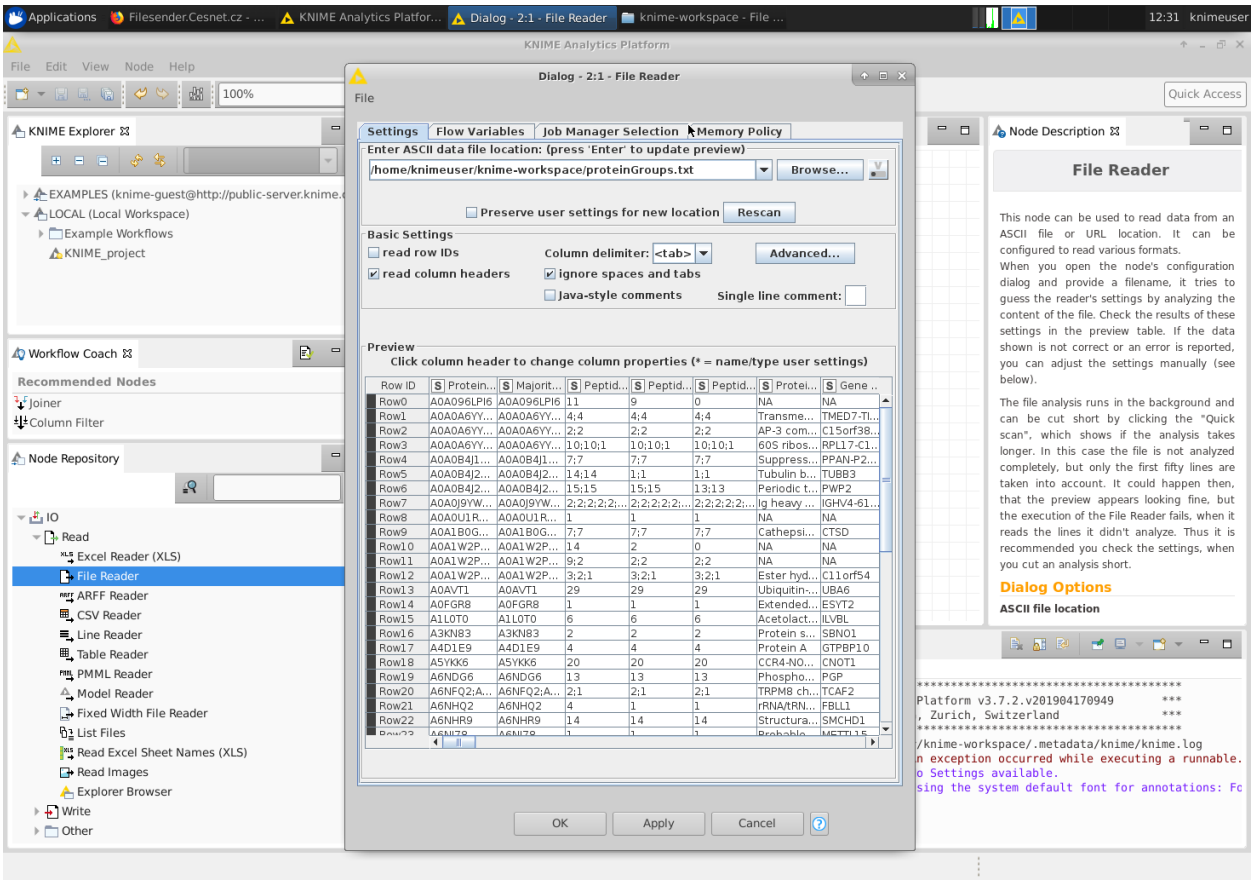
Please note the Node description part of the KNIME window that gives you help on the currently selected node.

Drag-and-drop the node into the workflow or you can also double click on the node inside the nodes repository to be automatically inserted into the workflow.

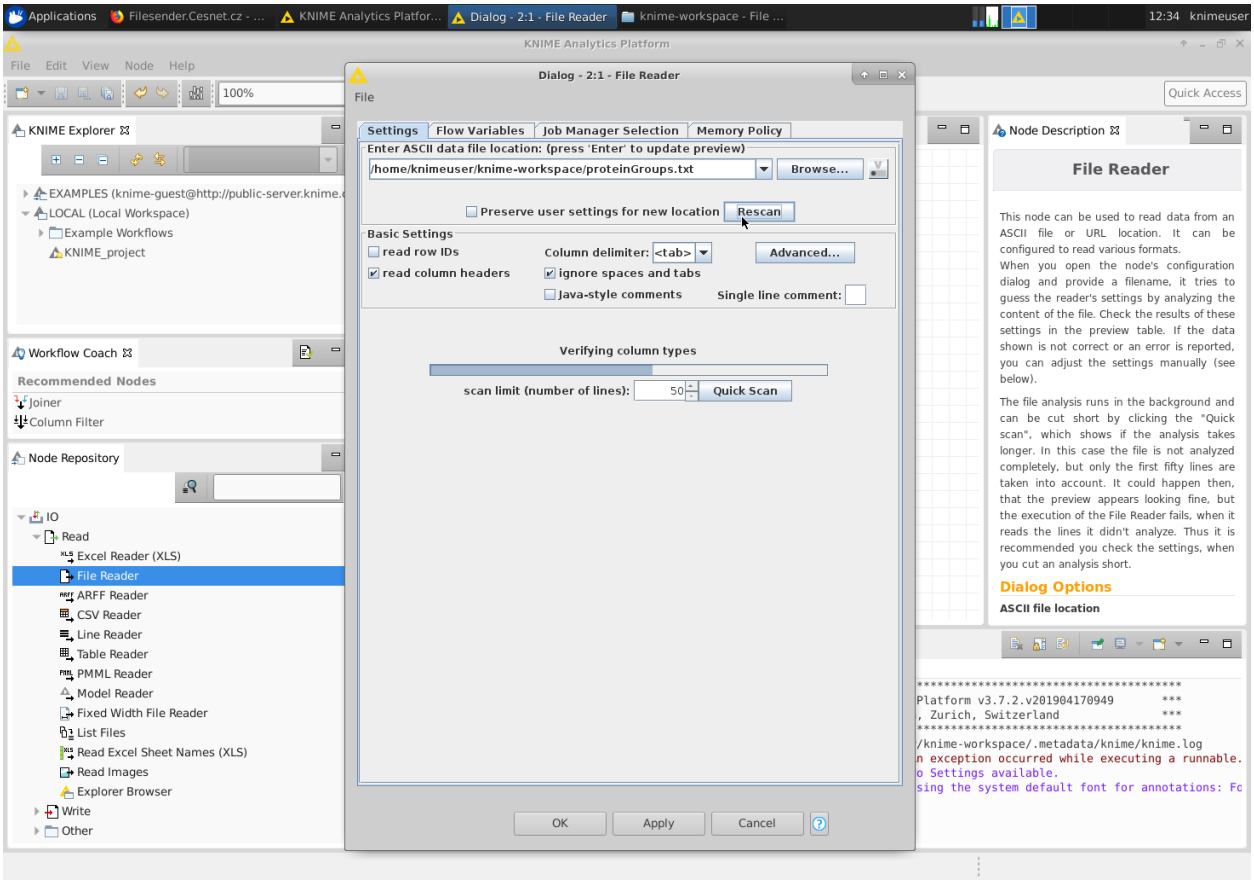
On double-clicking the node or pressing the F6 key while the node is selected, we will get into the node Settings.



Using the “Browse” button navigate yourself into the knime-workspaces and choose the proteinGroups.txt file:

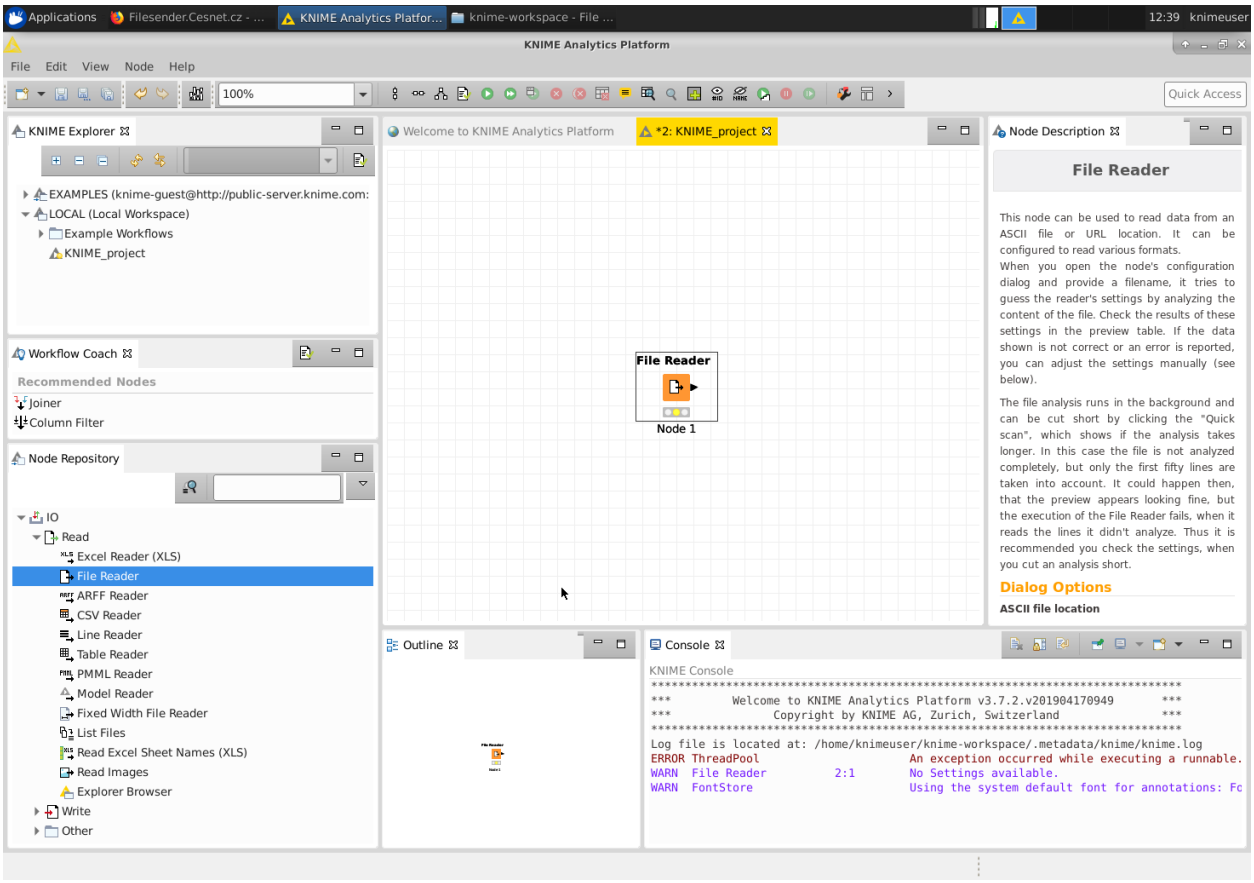


You will see the preview and you may adjust some basic settings such as whether the column headers should be read (leave ticked), what should be the column delimiter (tab in this case), etc. The file reader node tries to get all necessary settings automatically by reading the whole file or just part of it. You could have noticed this during the file selection:



In case you are trying to load big tables having e.g. hundreds of thousands of rows this process of reading the whole file to guess the settings might be long so you can select to guess the settings just using part of the file, e.g. 1st 1000 rows. To do this you can simply change number 50 seen in the figure above to 1000 and hit Quick Scan button.

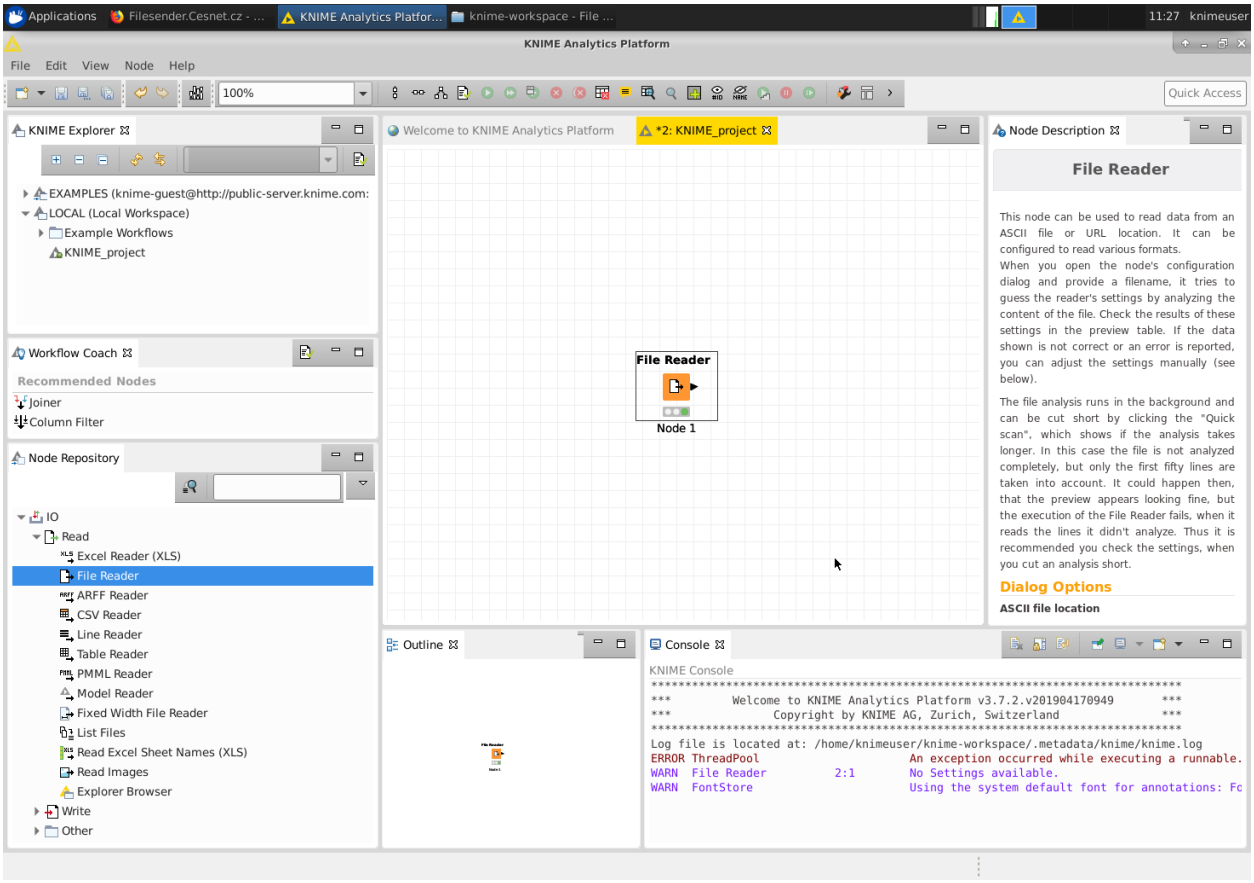
We can click OK in our case now, which will close the settings window. You can notice that the node status has changed from the red color to yellow color meaning the node is ready to be executed.



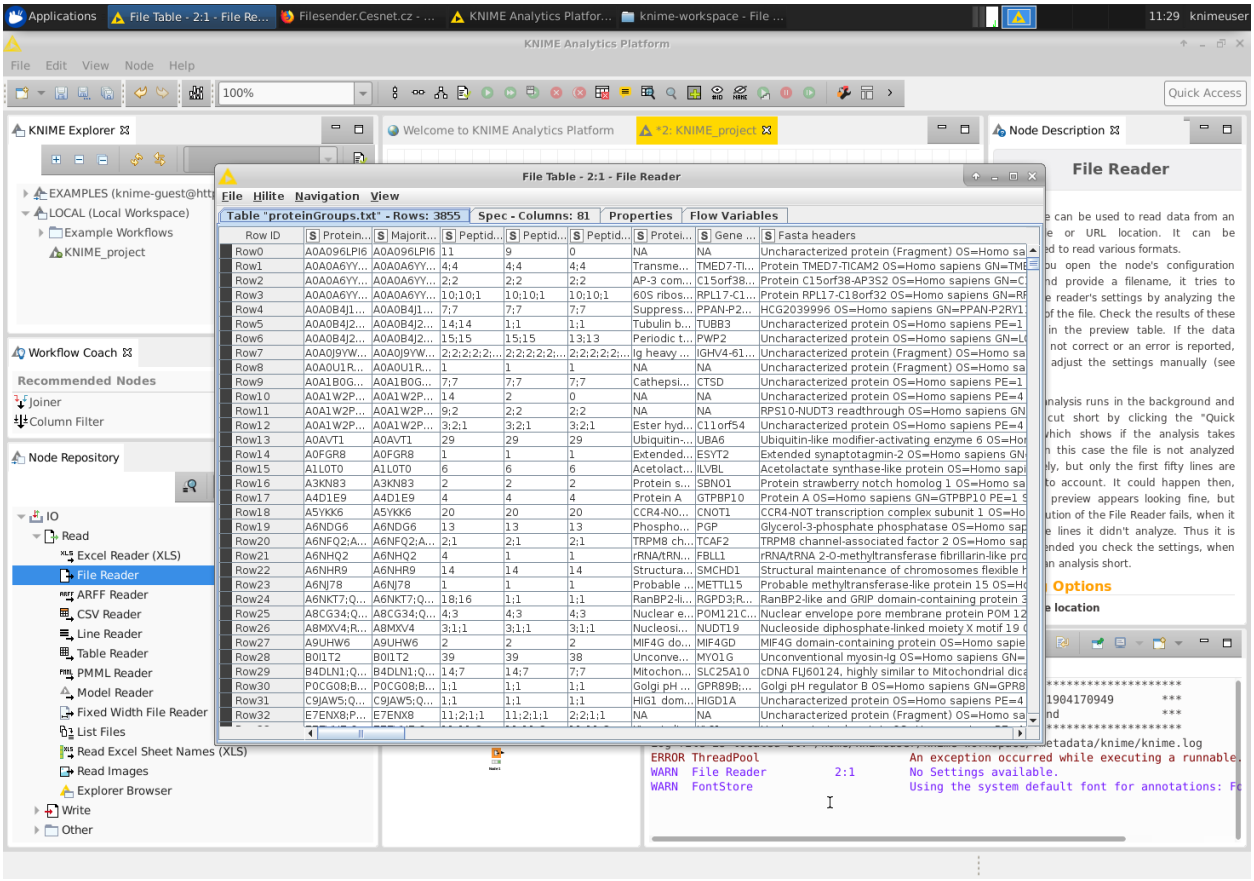
You can execute the node by three different ways:

a) Press F7 key
b) Right-click the node and choose “Execute”
c) Click on the green circle with white triangle inside in the top bar

After successful execution of the node you will notice the node status has changed to green.

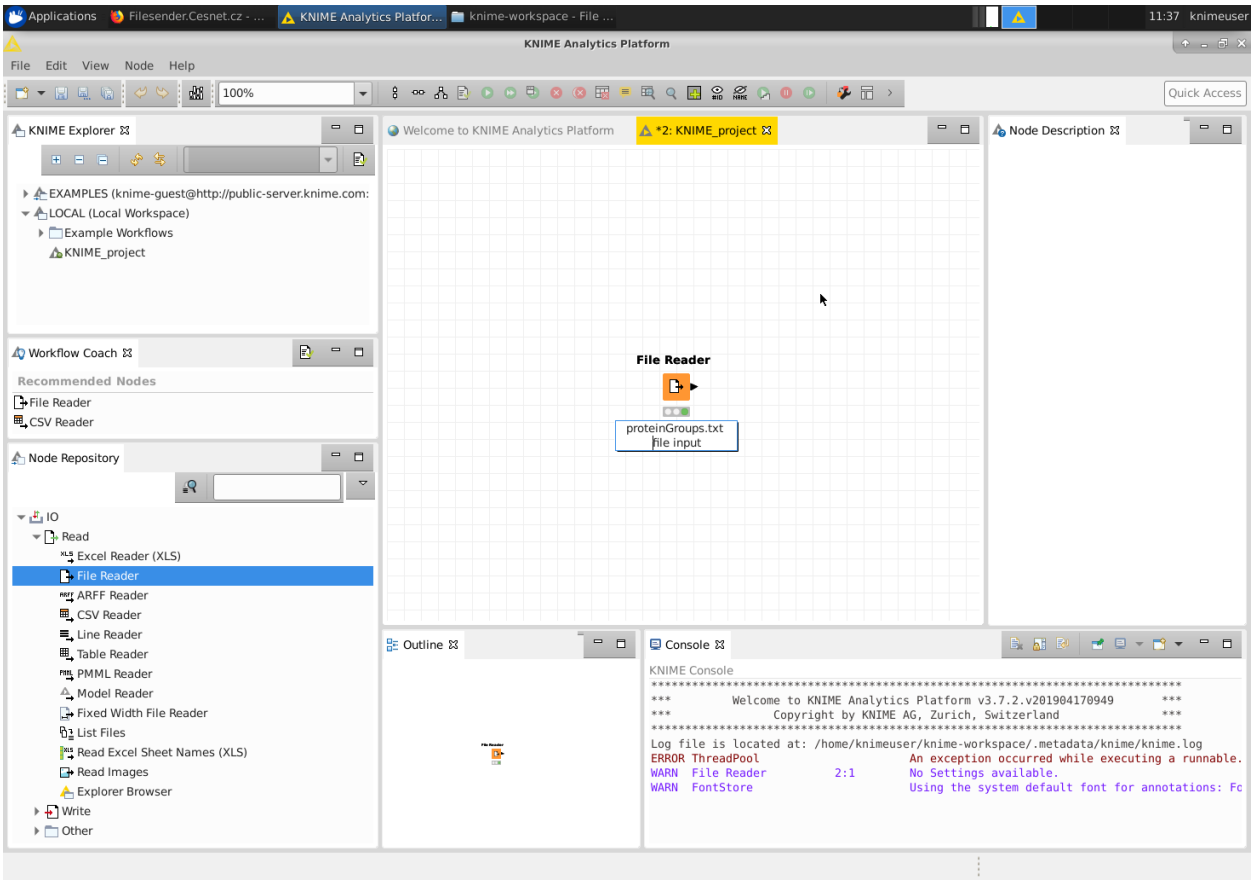


This can take long time for bigger table, therefore there is a progress bar as well telling at which point of the loading process the node is. When you now right-click the node and choose “File Table”, another tab will open where you can view the table on the node output port, i.e. our input table.



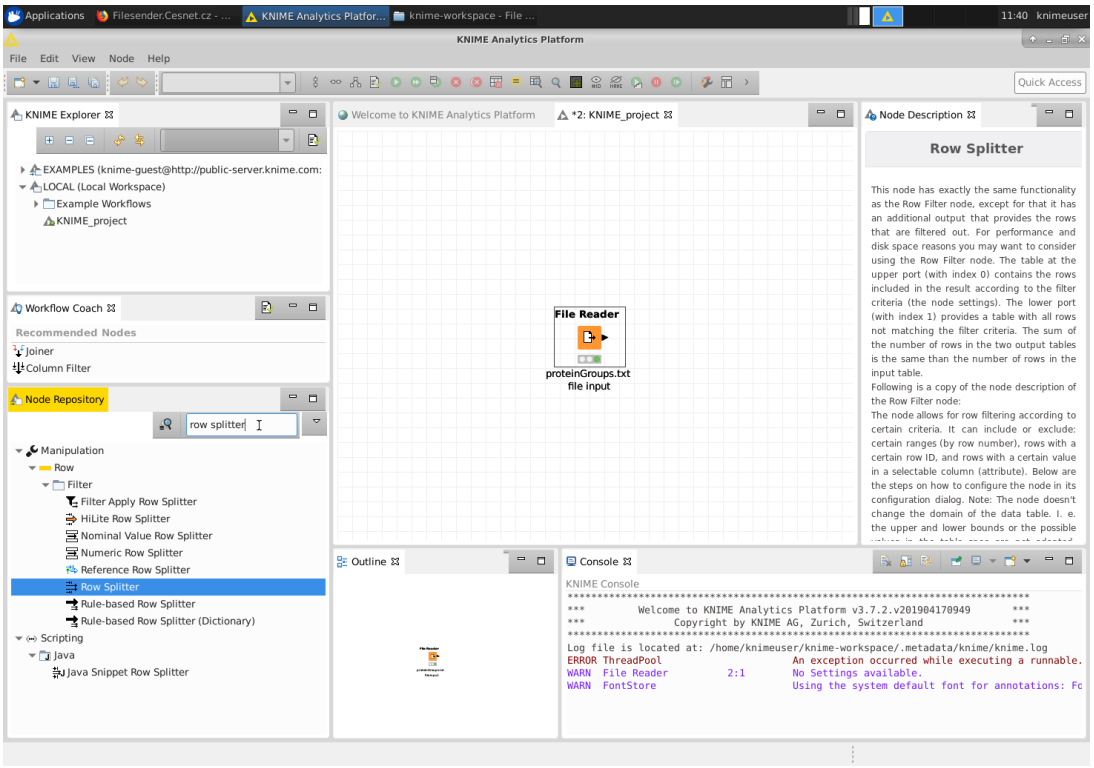
In general, when you will want to view any table/image produced by node, you can apply the same procedure – right click and the last option will get you to the view. This way you can check all intermediate results of your processing pipeline. You can also select the parts of the table and copy it into your clipboard for transfer to another node or outside the container. It is not possible to adjust any information at this point similarly to the data view in e.g. Excel. This is again in agreement with reproducibility concept – data adjustment this way may be harder or impossible to reproduce, but there are other ways to achieve that.

We can, on the other hand, adjust the node description, so double-click the “Node 1” text below the File Reader node and write your own text (e.g. proteinGroups.txt file input). Alternatively, just select the node and press F2 to get inside the node description field. You can also use Enter to make a multiline description.

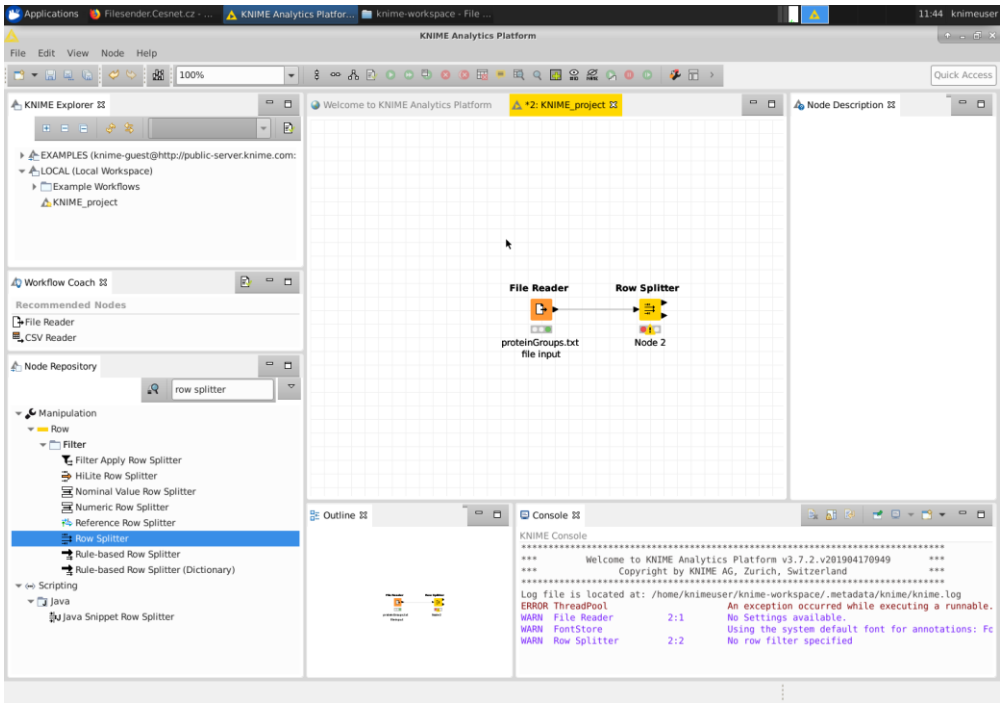


2.4 Addition of more nodes and connecting them into a simple pipeline

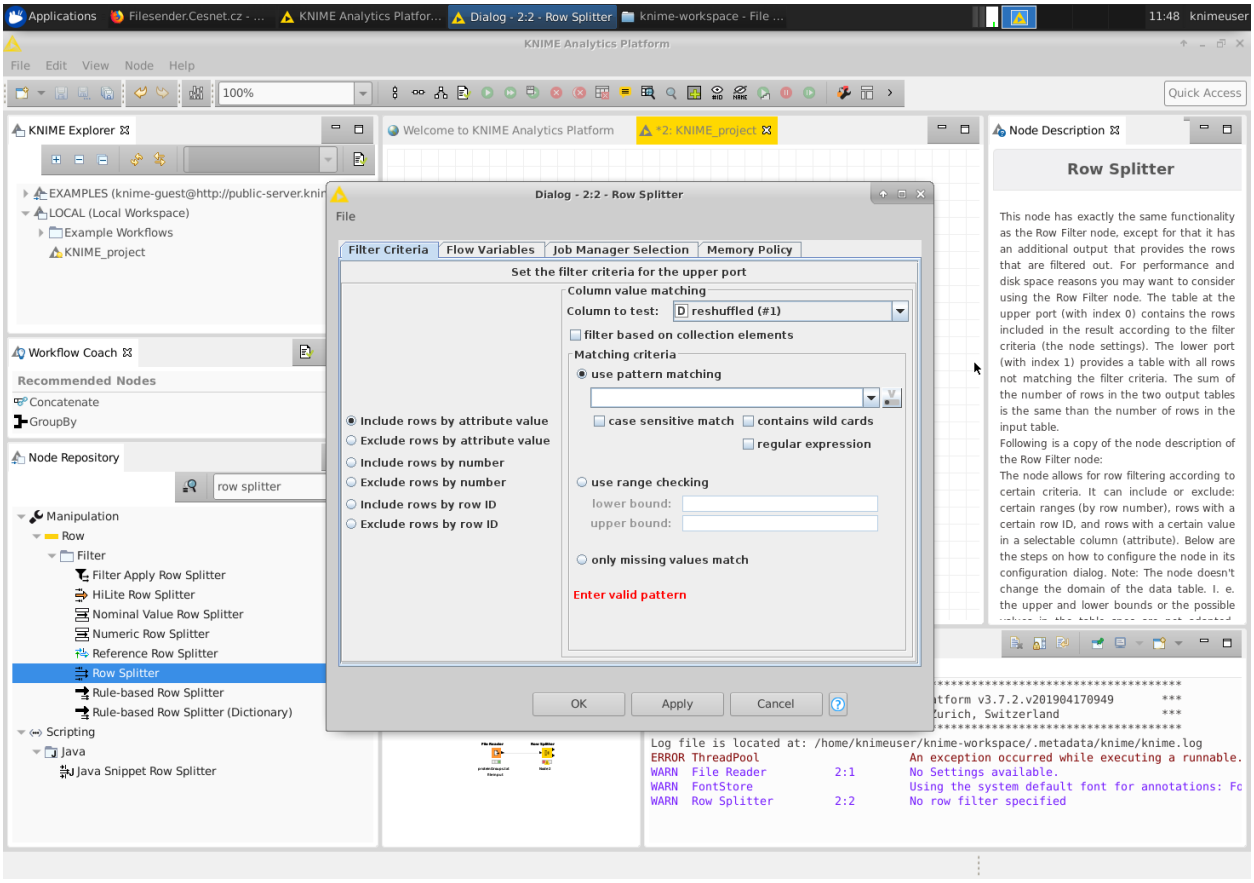
Let’s now connect the next node. In the typical proteomic workflow, we continue with the contaminants filtering and for this purpose we will use the **Row Splitter** node. Navigate yourself to Node Repository -> Manipulation -> Row -> Row Splitter and drag-and-drop the particular node into the workflow. Alternatively, you can search for the node writing its name into the search field in the Note repository KNIME subwindow:



We can connect two nodes by connecting output port of one node with the input port of the second node (click on the outport of File Reader and drag the line into the inport of Row Splitter). You can connect selected nodes also by Ctrl+L – KNIME will guess the way you want to connect the nodes based on their position and in- and outports of the selected nodes. You should end in this situation:

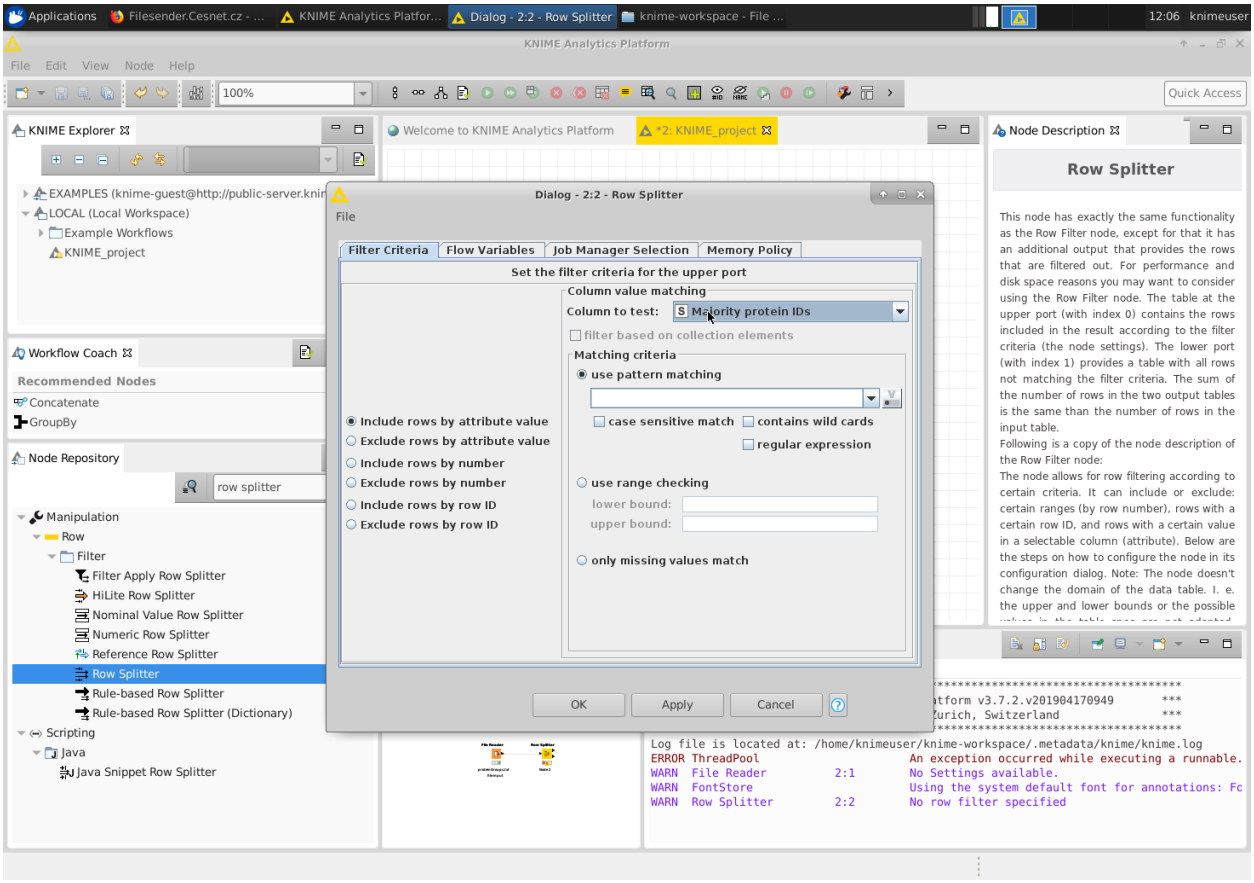


Let's configure the Row Splitter node. This node will split the input table into two based on matching the set criteria (in this case it will pass further the proteins which are not considered to be contaminants and exclude contaminants).

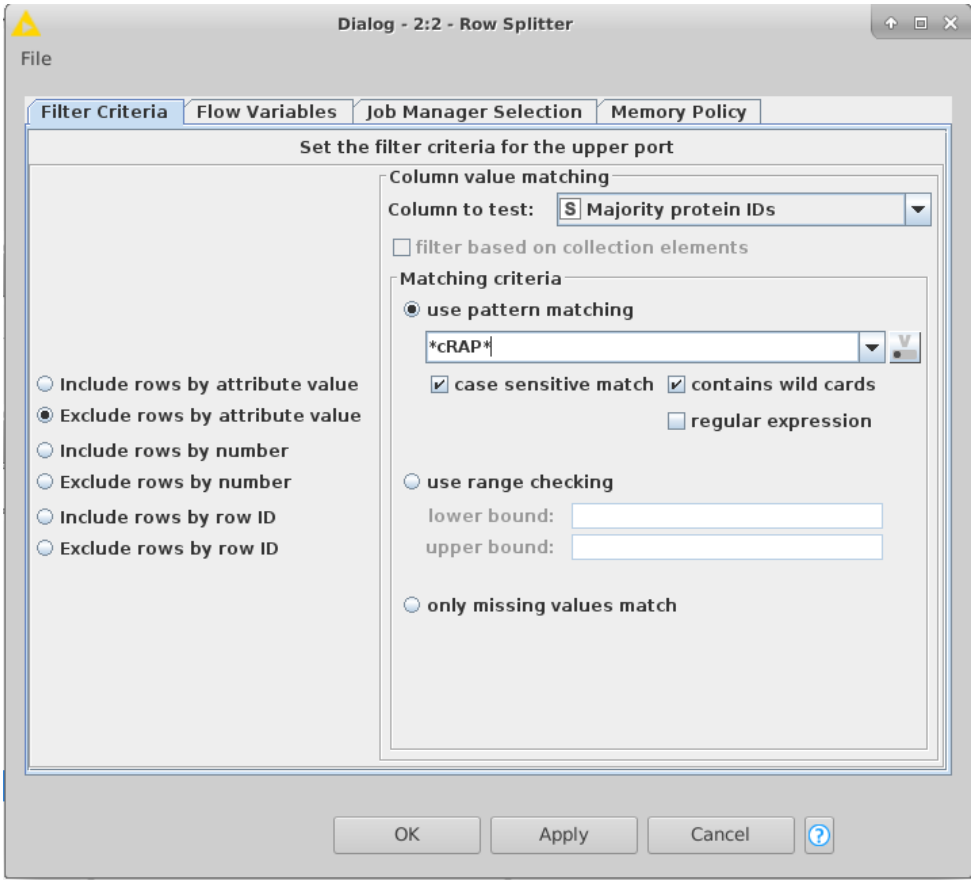


Row Splitter node needs you to select column where it should take the information based on which it will decide whether to include the whole row in the top/first node outport table or it will exclude it from the input table and pass it to the bottom/second node outport.

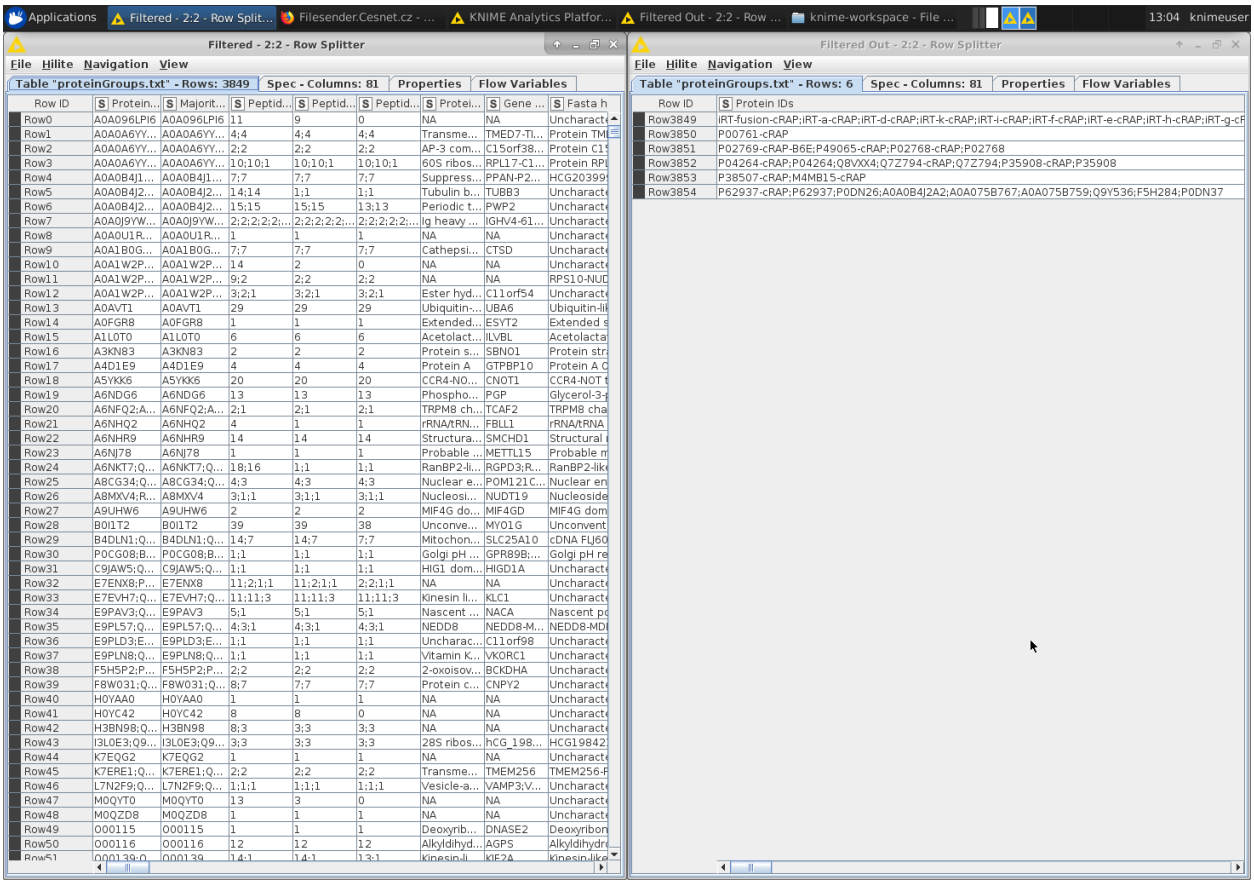
Firstly, we will filter out the proteins from cRAP database. We have used modified cRAP contamination database that contains “cRAP” keyword in inside the protein accessions and protein names. Column to test will be therefore “Majority protein IDs”. Click on the “Column to test” drop down field (now with “reshuffled (#1)” column selected) and either scroll all the way up to find the right column or start writing its name to get it automatically selected (the drop down field needs to be “selected”, so just click on the current value twice to make the drop down menu appear and hide again to enable you to select the column by writing its name):



We want to exclude the rows which will contain the “cRAP” keyword in the majority protein IDs (e.g. see row 3850: P00761-cRAP). Because we don’t know where exactly the cRAP keyword will stand in the text string of the tested column rows, we will use the **wildcards (*)** on both sides (i.e., “*cRAP*”) which means that any number of characters can precede (*string) or succeed (string*) the particular string. Also, we will use the **case sensitive match** to ensure that only the pattern “cRAP” (not e.g. CRAP, Crap, etc.) will be matched. We want to exclude these rows from further processing, so we will also set “Exclude rows by attribute value” in the left part of the node settings. The final node settings should look like this:

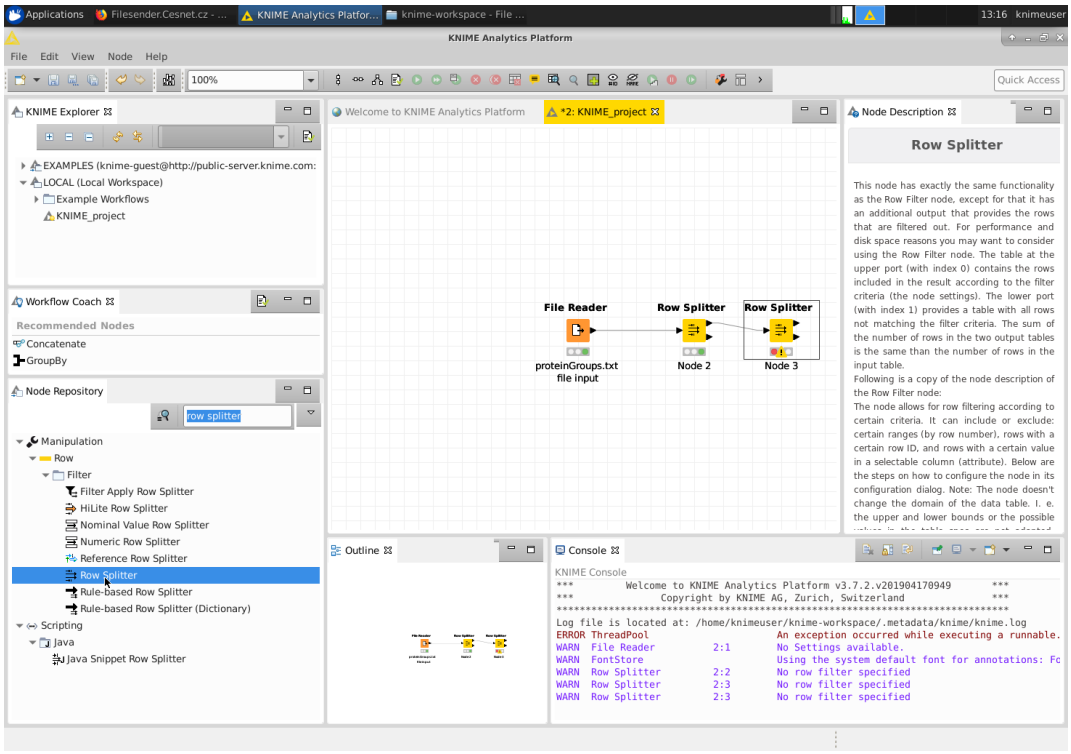


Now we are ready to execute the node, confirm the settings by OK and run the node (e.g. by pressing F7). You can now check both outports of the Row Splitter node to check it worked as expected. The first/top output (table on the left) are the proteins which did not pass the criteria (i.e. don't contain "*cRAP*" in the "Majority protein IDs" column) and on the second/bottom output (table on the right) we get the proteins which contain the cRAP string:

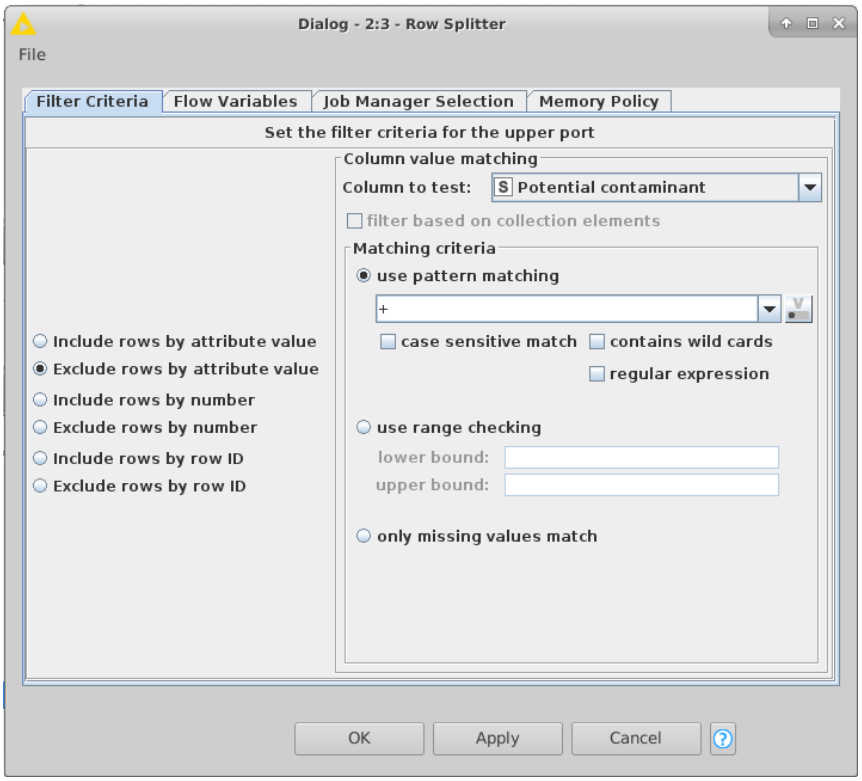


Next, we would like to filter out potential contaminants one can include during the MaxQuant search. There is specific column holding the information whether the protein group is (there is "+") or is not (empty cell) coming from the internal MaxQuant contaminants database.

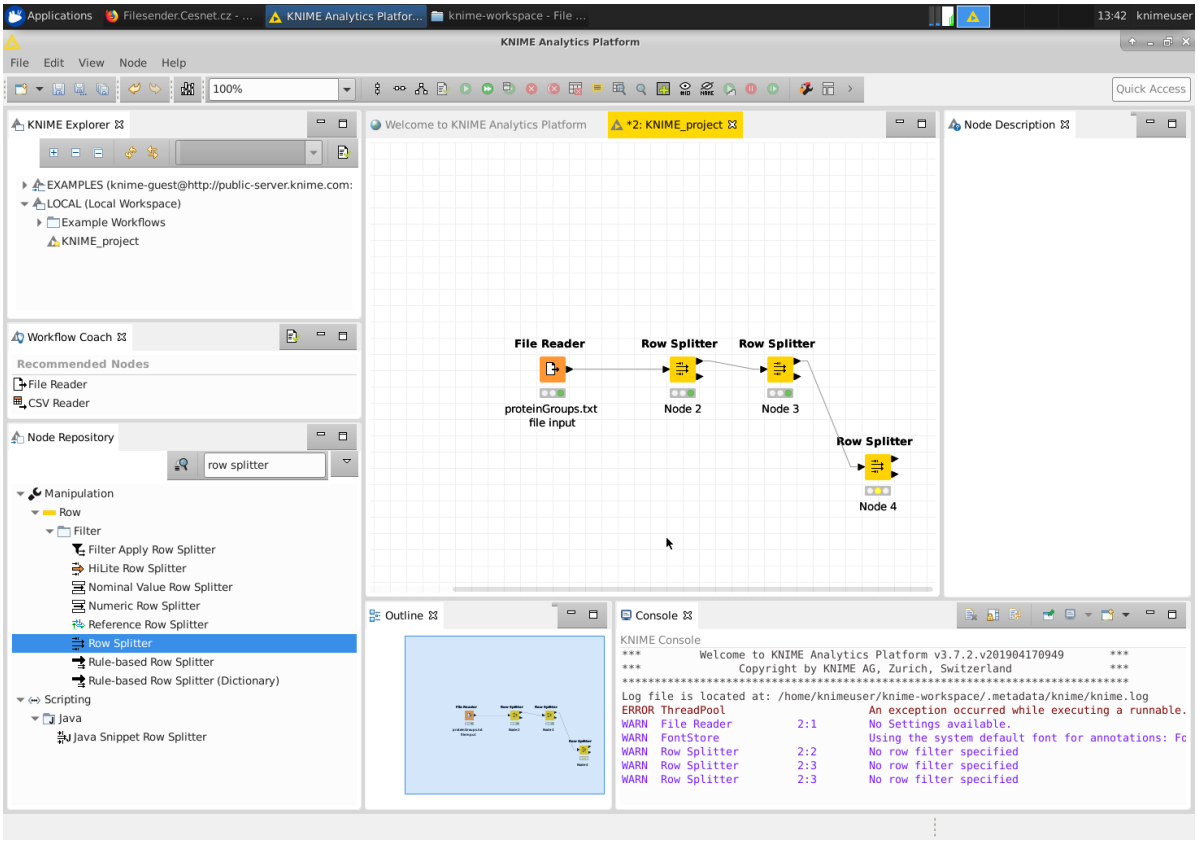
We will need 2nd Rows Splitter node to do second round of table filtration. We can add it again from the Nodes repository. This time select the already present Rows Splitter node first and double click the Rows Splitter inside the Nodes repository. The node will be automatically added next to the selected node and connected:



Configure the 2nd Rows Filter node – give it also some appropriate name – in the following way to exclude potential contaminants based on the MaxQuant internal contaminants database:



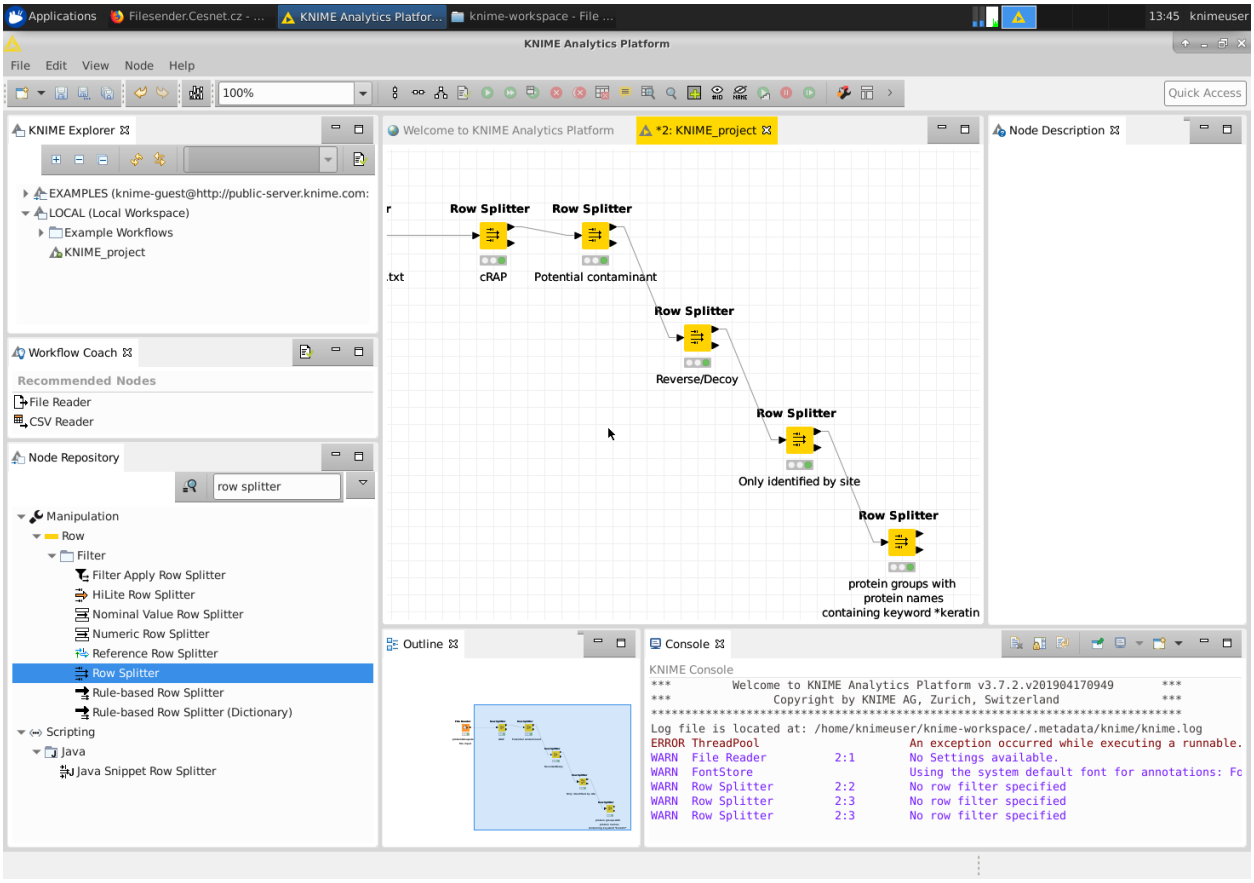
We will be adding more Rows Filter nodes and it is worth noting that you do not have to add new nodes only from the nodes repository – you can simply make copies of nodes already present in the workflow. Just select the 2nd metanode and make a copy of it by pressing Ctrl+C and Ctrl+V. Copy of the original node has also the same node settings as the original node which can be used to make additional steps faster not to set up everything from scratch. Connect the 3rd Rows Splitter node to the upper outport of the 2nd one to get situation like this:



Now, create two additional Rows Splitter nodes either by copying of already present ones or using the one from the Nodes Repository to end up with 3 nodes that needs adjustment of their settings. Use the following settings to filter out 3 other types of protein groups:

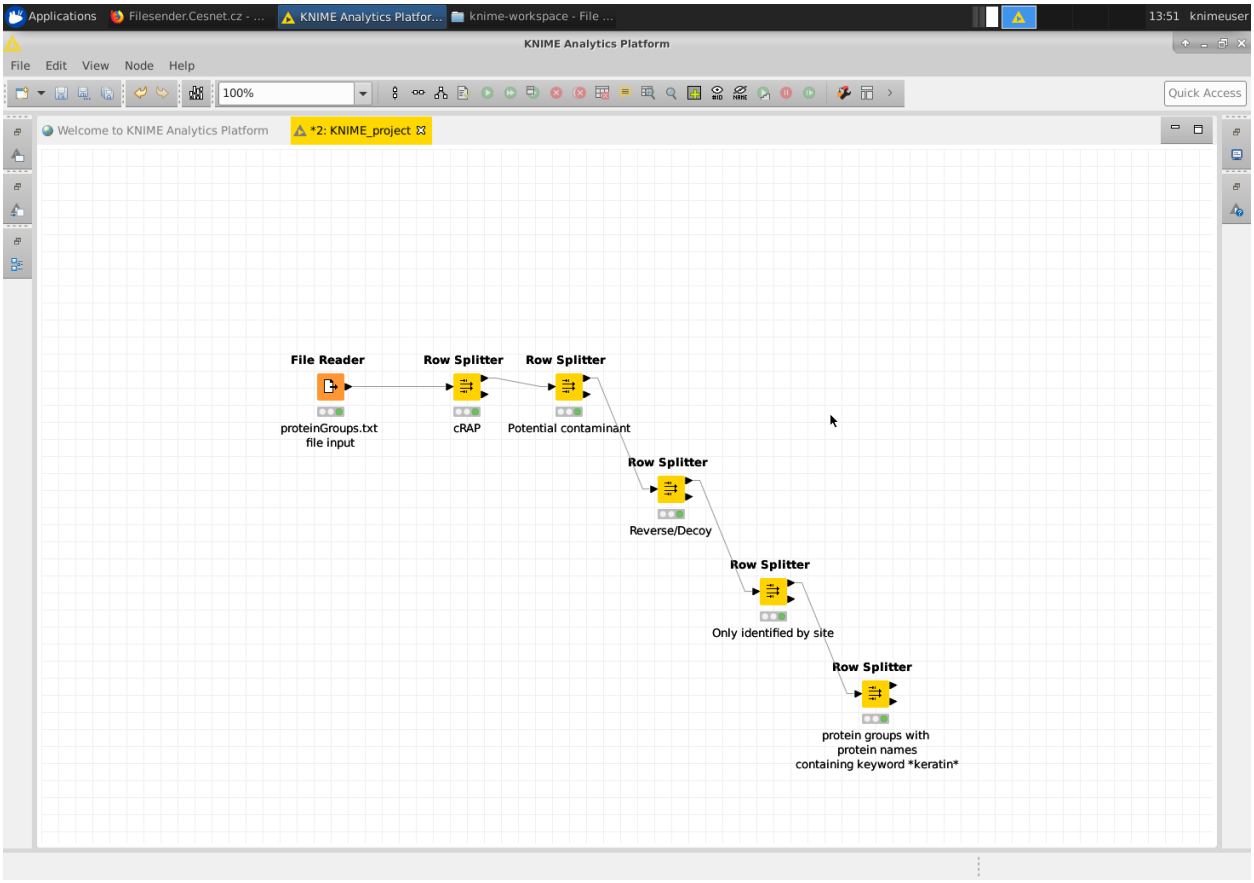
- Reverse/Decoy
  - Column to test: Reverse
  - pattern matching: +
  - no case sensitive match and no wild cards
- Only identified by site
  - Column to test: Only identified by site
  - pattern matching: +
  - no case sensitive match and no wild cards
- protein groups with protein names containing keyword “keratin”
  - Column to test: Fasta headers
  - pattern matching: *keratin*
  - no case sensitive match, but wild cards yes

As you can see, it is really good idea to add some description to the individual nodes so, let's just use the text mentioned above or use your own to get yourself oriented in the workflow. The final workflow will then look like this:

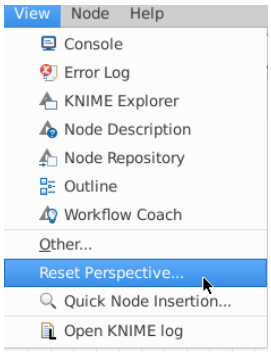


It is also not very practical how the workflow is displayed in the KNIME application window, but this can be adjusted. So, let's make more space on our screens dedicated to the workflow by hiding or closing subwindows we will not use or not that often at least. Close Workflow Coach subwindow by pressing the cross next to its name. Hiding is done by pressing horizontal line symbol in the right upper corner of each but node description subwindow. Hide all other remaining subwindows except the one showing workflow.

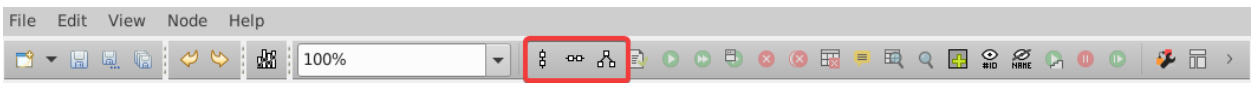
You should get situation like this:



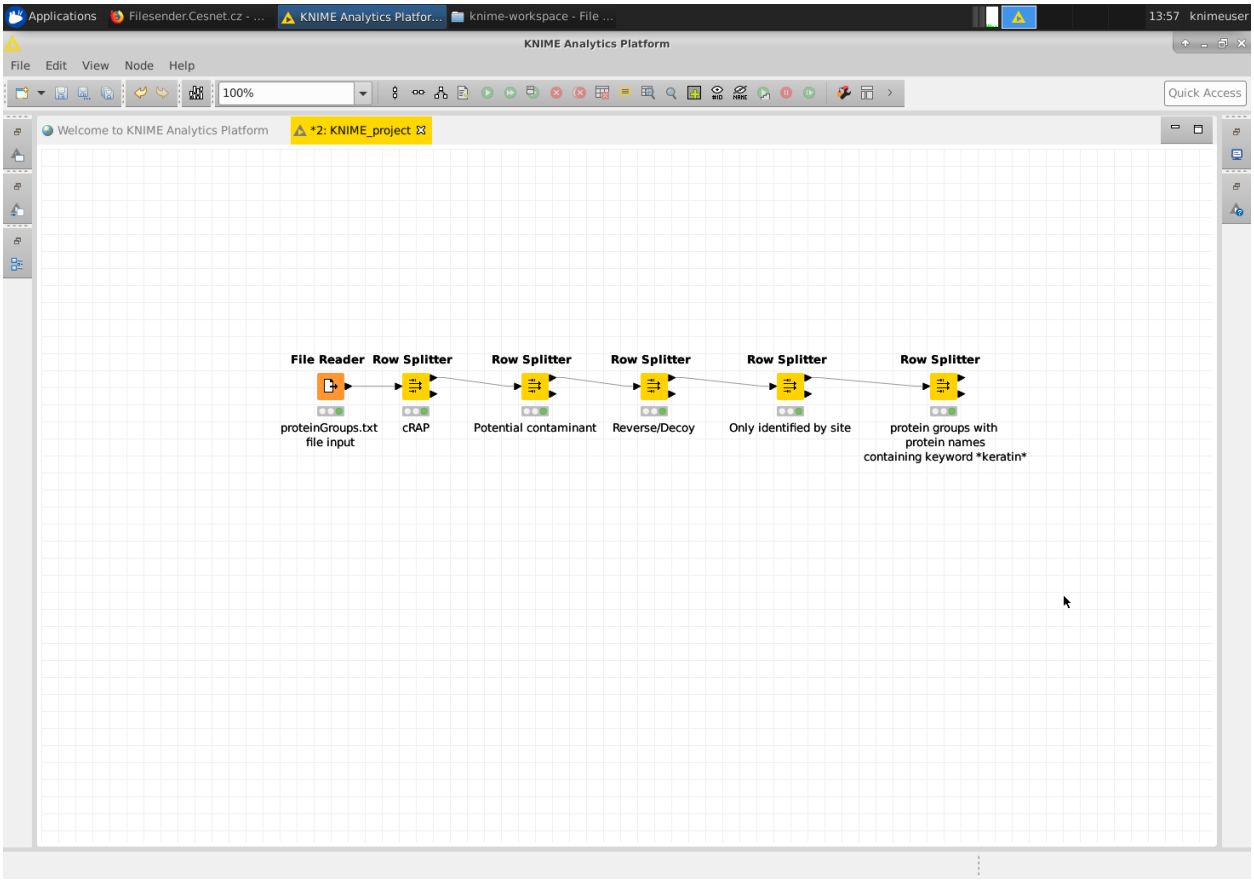
In case you would get back to the original display, there is a Reset Perspective option in the View menu that will get you to the default subwindows layout:



Individual nodes layout can be also adjusted by individual nodes moving or automatically using the dedicated icons in the top panel:

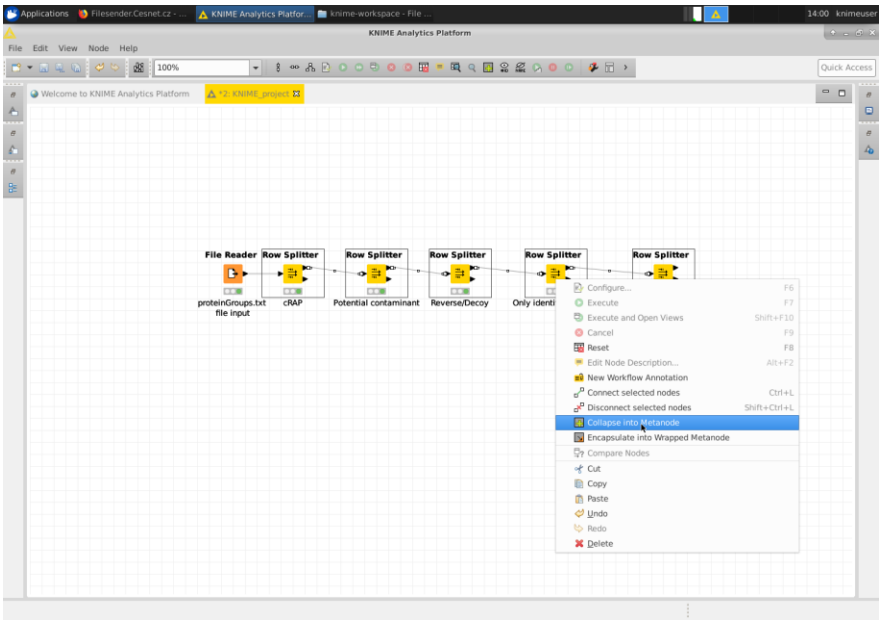


You can get workflow look like this which is still far from perfect as we are in the very beginning of the data processing, and we will need to use tens of nodes so we need better structure:

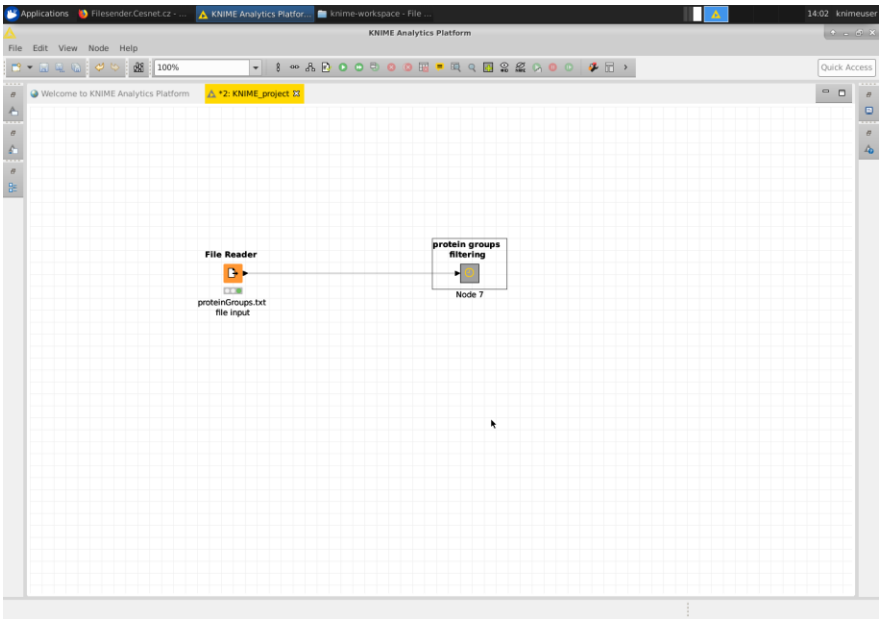


2.5 Creating a metanode

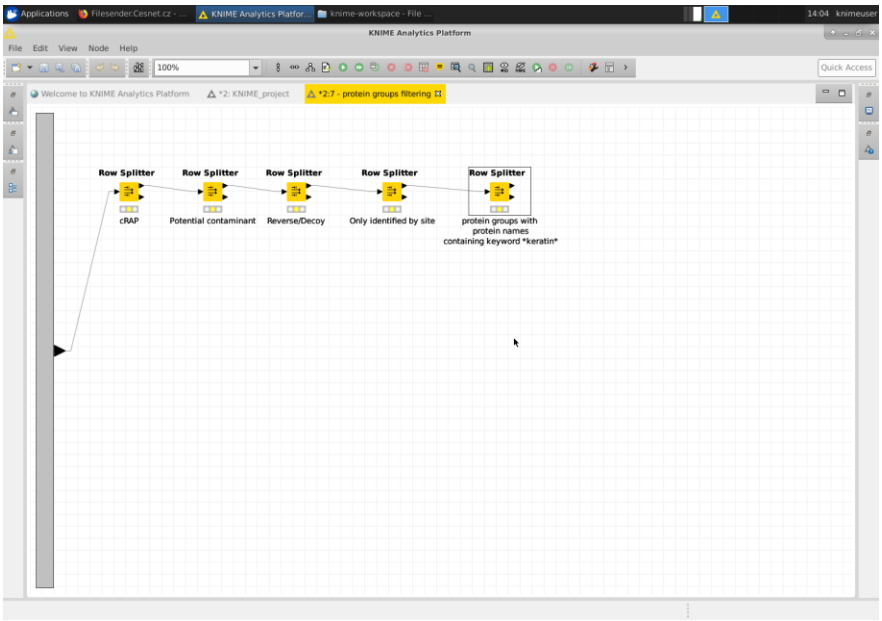
Sometimes it's useful to pack several nodes into so called metanodes, which might significantly improve the workflow readability and structurability. So, let's pack the nodes dealing with protein groups filtering into one single metanode. Select all 5 row splitter nodes, right click on any of them and choose "Collapse into Metanode".



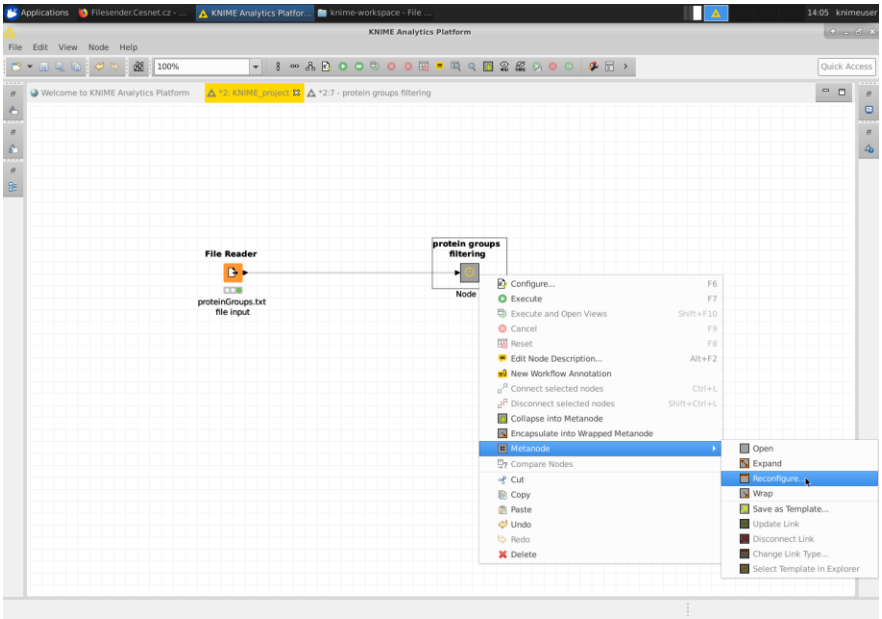
The nodes going into the metanode will have to be reset (i.e. we will have to run them once more, settings are preserved) so confirm that dialog and give the metanode a name, e.g. "protein groups filtering" to get situation like this:



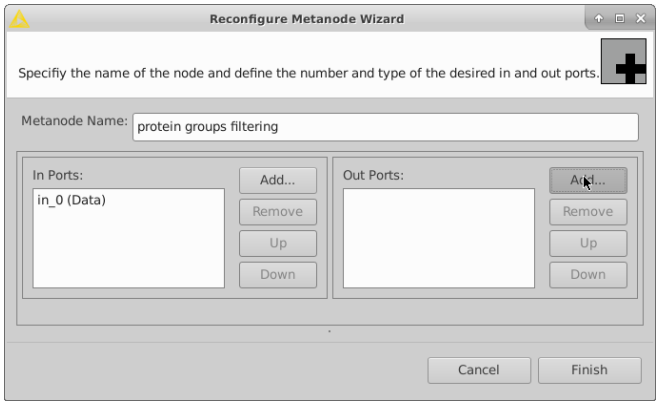
When you double-click on the created metanode, a new tab will open with the particular nodes inside the metanode shown in form of subworkflow:



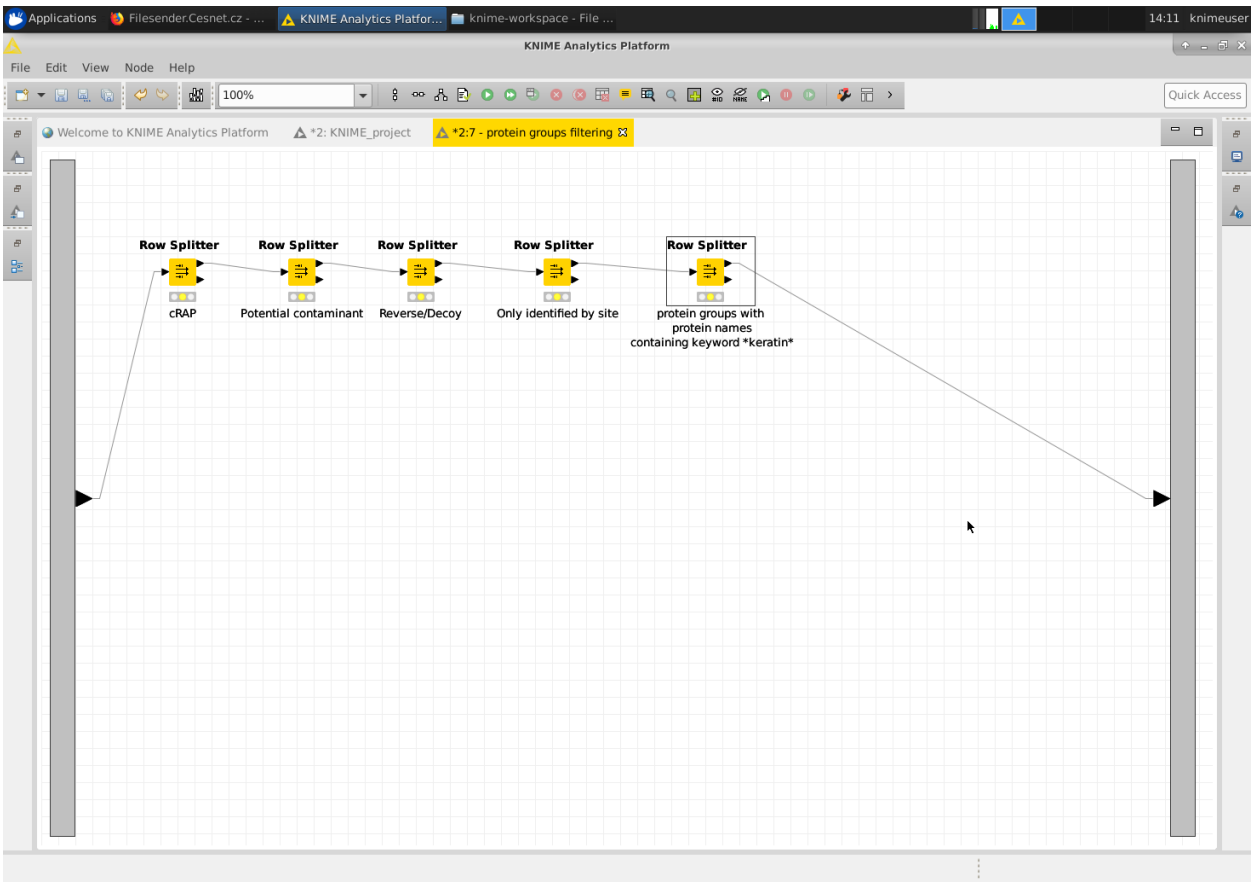
When we want to continue further in the data processing, we need an outport on the metanode, similarly to the inport of the metanode evident on the left side of the metanode subworkflow. To get one, navigate yourself back to the workflow (left click on the workflow name in the tab) and right click on the new metanode and choose Metanode -> Reconfigure.



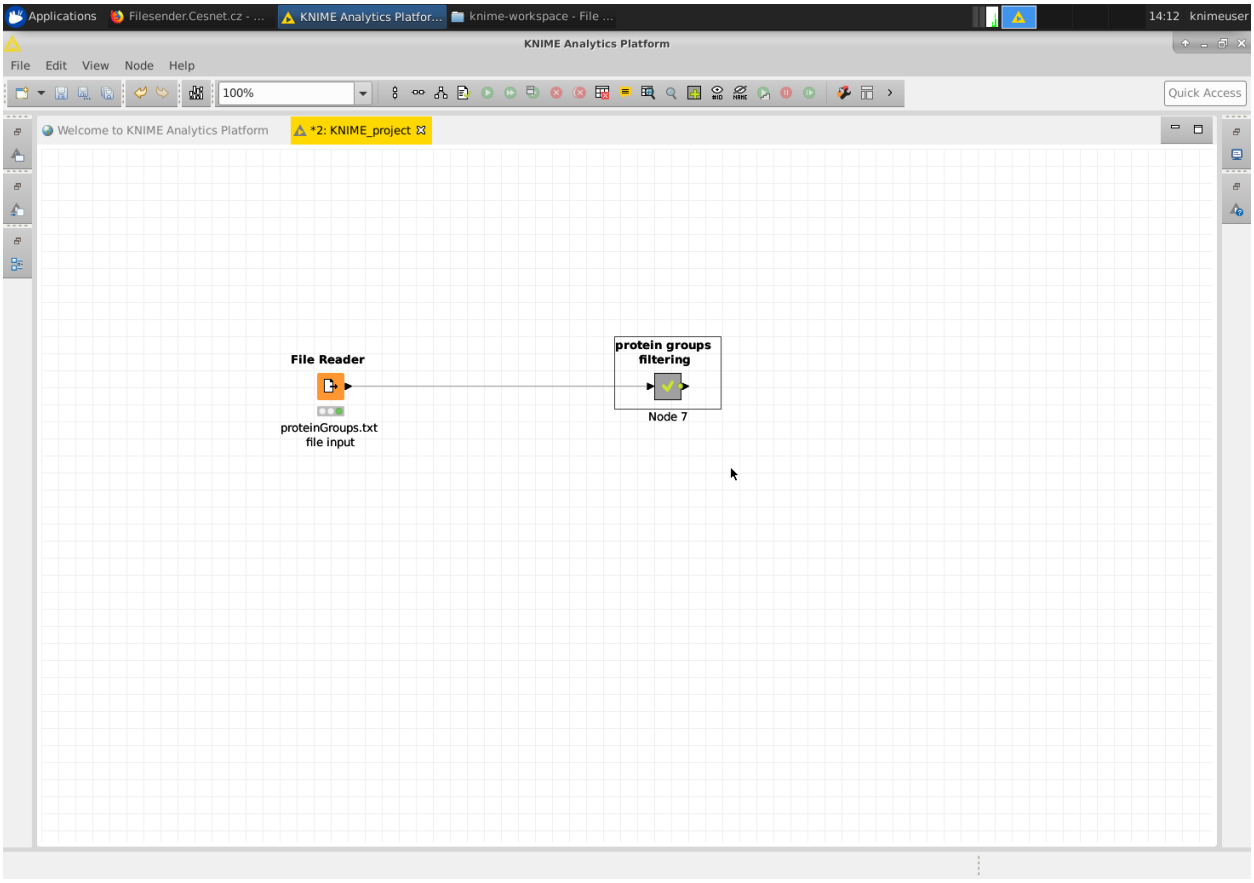
In the presented form which will open for you one can change metanode name and also add other inports and more importantly for us now, add outports. Click on the “Add” button located on the right side to add outport:



There are different port types as you may see when you click on the Port Type drop down selection menu. In our case, we will need port for table, which is named here as “Data” port type and is selected by default. When you click on Finish, you will see that the contaminants filtering metanode has now one output port. Navigate yourself back to the metanode by double-clicking on it or selecting its tab and connect now the last Row splitter with the output port of the metanode:

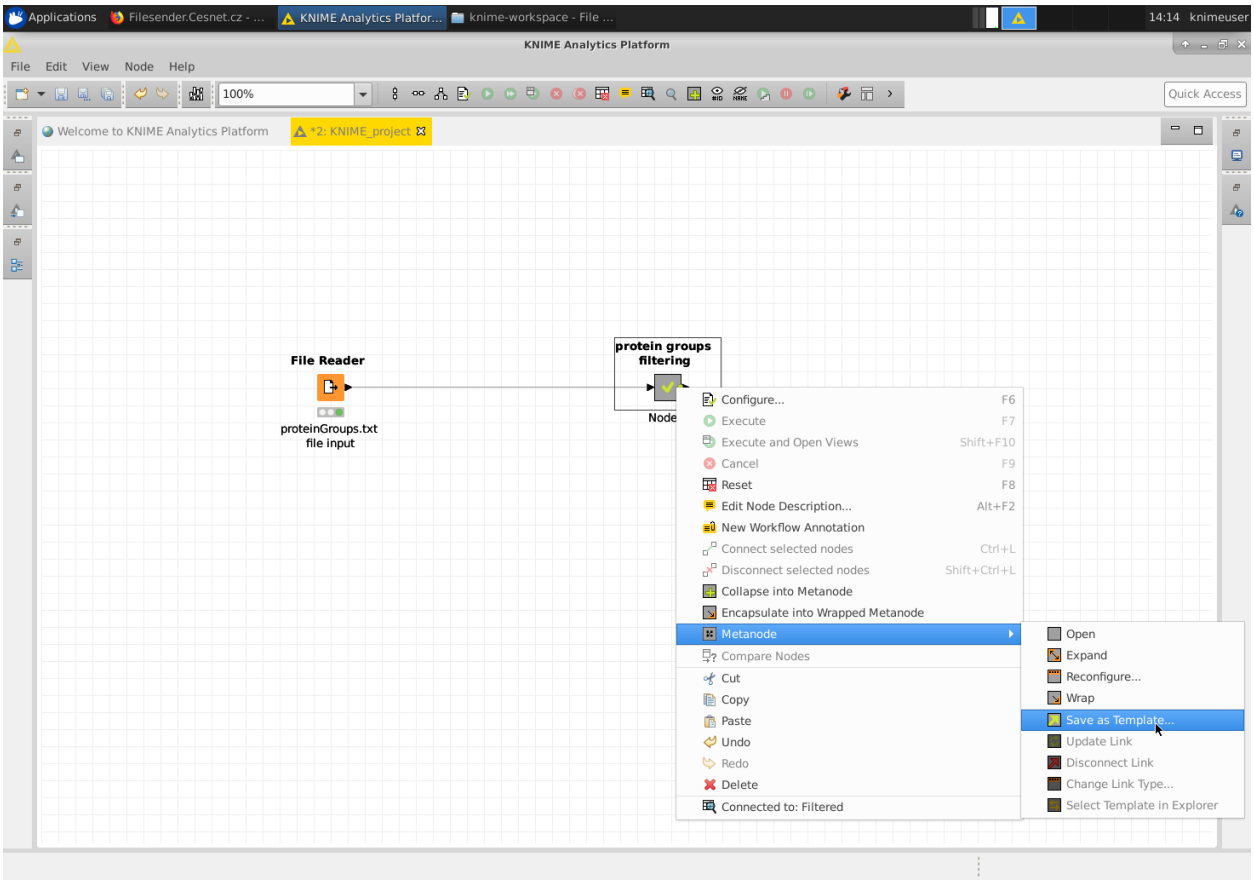


Close the metanode subworkflow and execute the metanode the same way you have executed individual nodes, e.g. by pressing F7, and after successful execution you will see the metanode appearance changed and there is green “done mark” inside:



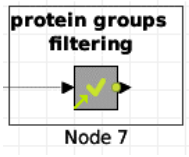
2.6 Creating a metanode template (linked metanode)

To share the metanode with our colleagues or simply re-using it in our own workflows we can save the created metanode as a so-called template. To achieve this, right-click the metanode -> Metanode -> Save as template:



Choose the destination folder (in the knime-workspaces), where your new metanode template will be saved and select “Create absolute link”.

There will be a green arrow added to the metanode icon meaning this metanode is linked to the template.

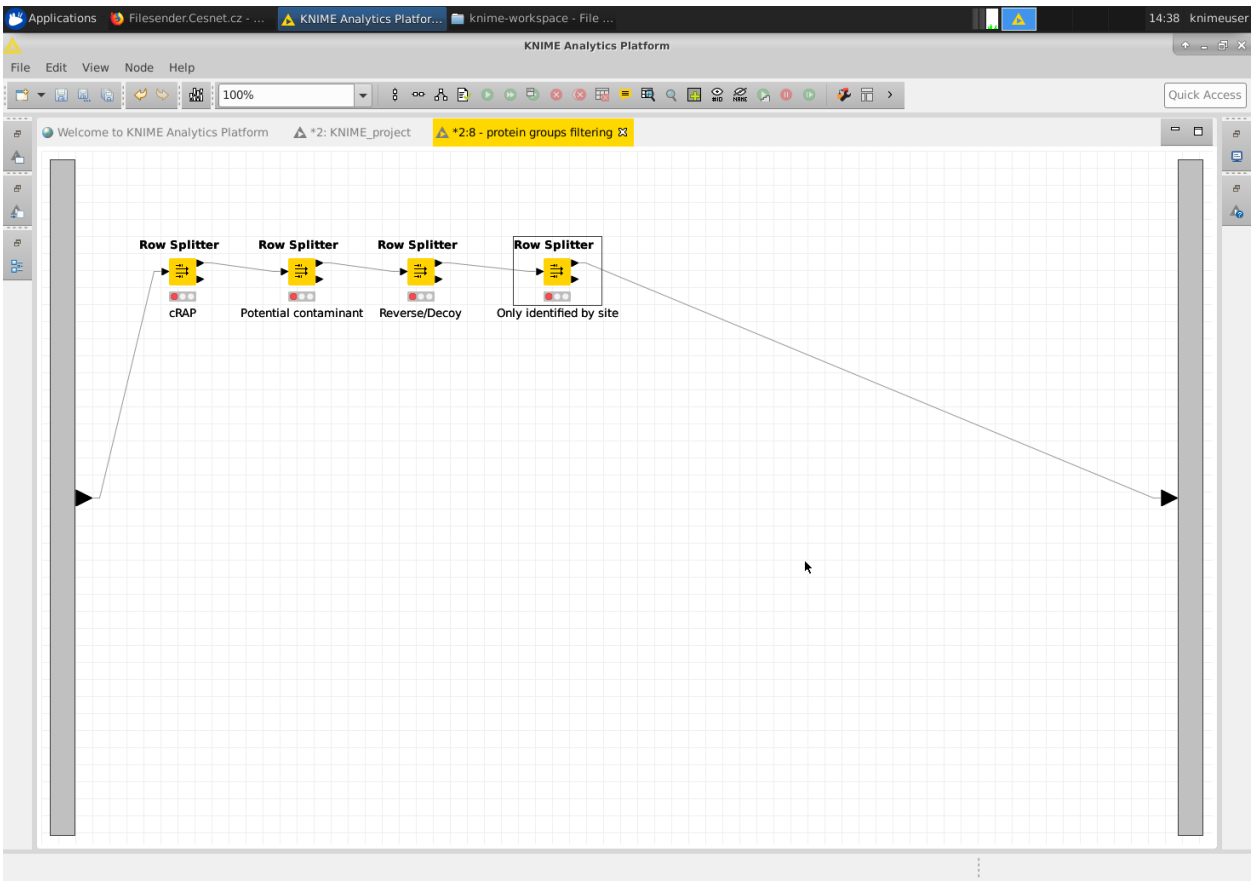


The linked metanode is able to get updated based on its template. This can be done manually either in the Node menu -> Update Metanode Links (this will check updates of all metanodes in the current workflow) or by right-clicking on the metanode (having the green arrow) -> Metanode -> Update Link. The linked metanodes have limitation that they cannot be modified – this can be also seen when inside the

metanode, the background is different. This is safety precaution as the template might get changed and you may lose changes you have done in the linked metanode meanwhile when the linked metanode would get updated. In case one would like to adjust the linked metanode, it is good practice to update it to the most recent version and disconnect the link afterwards

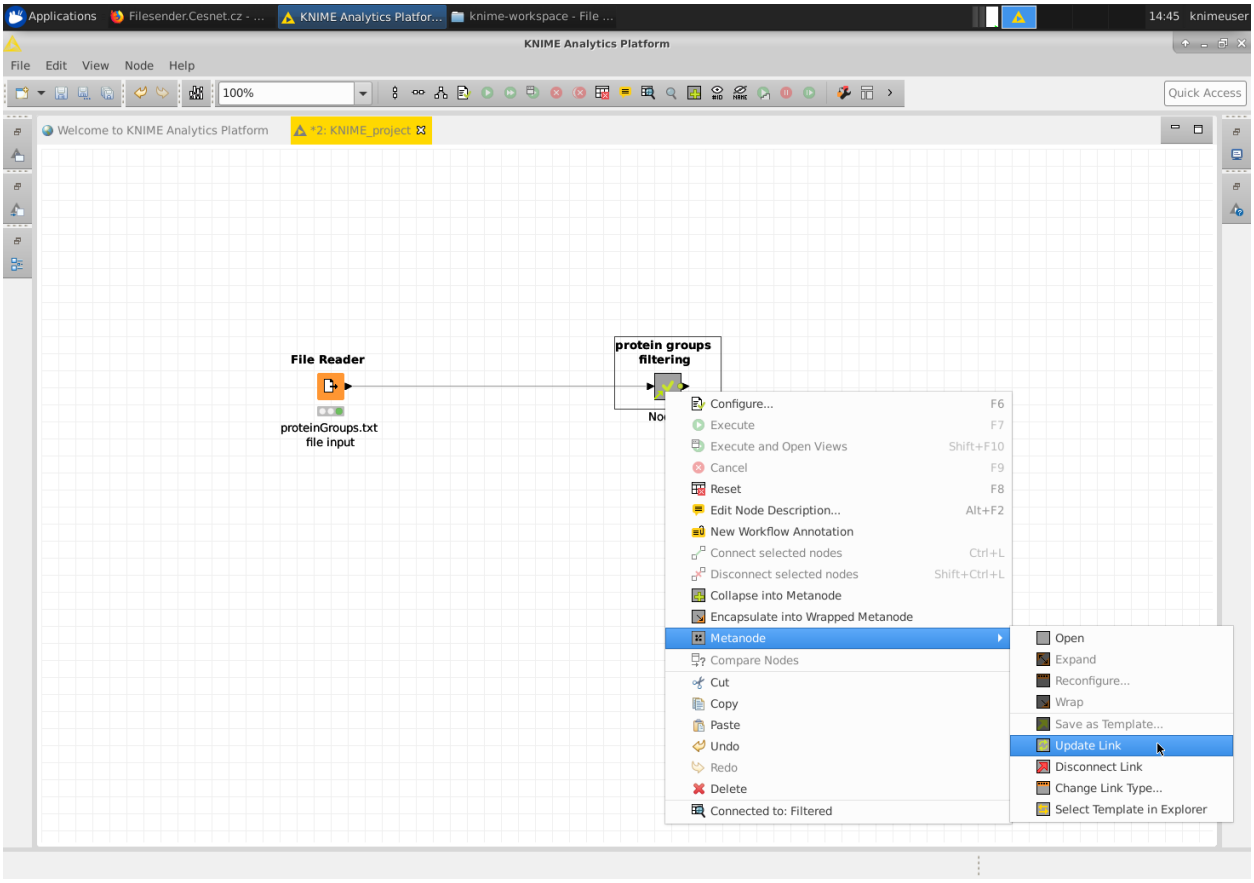
2.7 Updating the metanode template and linked metanode

Let's try how you can modify the metanode template now. Make a copy of the linked metanode template e.g. by pressing Ctrl+C and Ctrl+V or by drag-drop procedure from your workspace folder. Linked metanodes cannot be modified so let's disconnect the linked metanode copy from its template (Right click -> Metanode -> Disconnect Link). Navigate yourself into the disconnected metanode and delete e.g. the last Rows Splitter node and connect the last node to the metanode outport to get the situation like this:



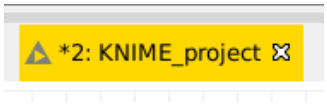
Close the metanode tab (cross on the right side of its tab) and right click on the metanode again in the workflow and save the metanode as template again (in the right click menu go to Metanode -> Save as Template...) into the same place overwriting the already present metanode template and creating absolute link type.

The metanode template modification does not change the content of the linked metanodes (e.g. the one you have still connected in the workflow) unless you ask KNIME to do so. Let's right click on the linked metanode connected to the File Reader node and select Metanode -> Update Link:



You will receive information that there is an update of the metanode template to which the metanode is linked to and you are asked if you want to update the metanode based on its template. Select you want to update it to get the metanode newer version. This will result also in the linked metanode resetting (one need to run it again). You can navigate into the metanode connected to the File Reader node to find out there is no longer the last Rows Splitter node.

2.8 Saving the workflow



You may have noticed the asterisk before the workflow name meaning the workflow has been changed since the last saving. To save the workflow you can use ctrl+S or navigate yourself into File -> Save (or "Save as" in the case you want to save the modified workflow under the different name). Saving of the workflow can take some time; and we recommend to save in short time intervals in order not to lose unsaved work in case of unforeseen complications.

3 Workflow for processing label-free quantification MS data

3.1 Getting the workflow templates and metanode templates for MS data processing

Workflow creation can take a lot of time and it makes no sense to define it again and again if you are doing similar steps frequently. Metanode templates are one way how to reuse and share your work in KNIME. Reusing the complete workflows is also possible.

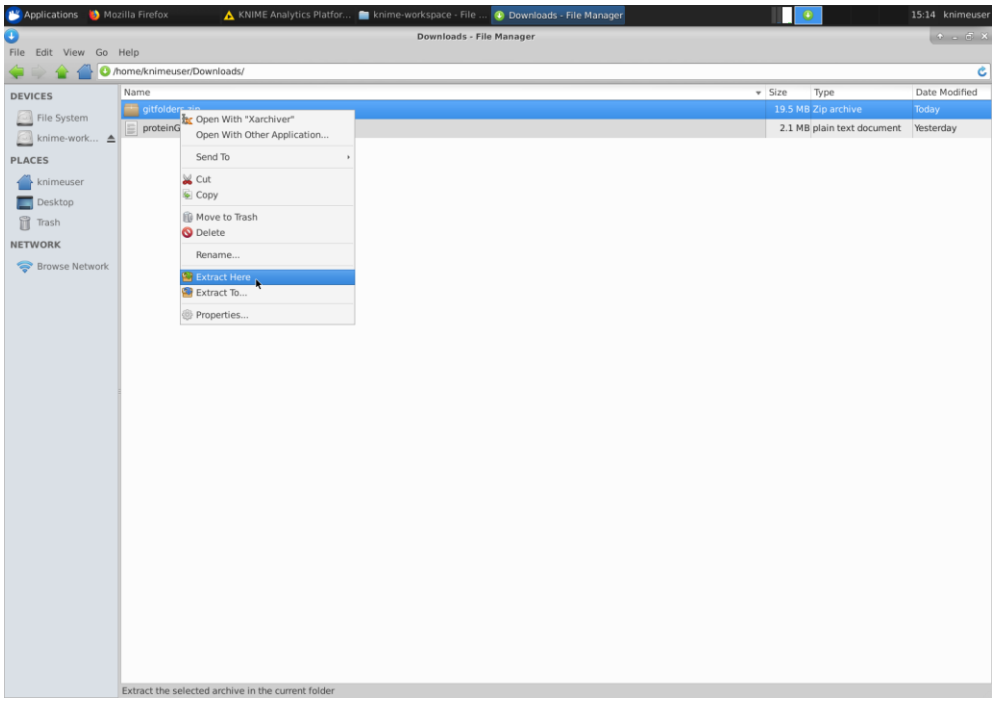
We have created GitHub repository for the KNIME workflows (https://github.com/OmicsWorkflows/KNIME_workflows) we would like to share with the community. There is up to now only a single workflow for processing of protein groups from label-free experiment, concretely protein groups from MaxQuant. The workflow can be downloaded and pasted into the KNIME workspace. We have also prepared scripts to reset dedicated folder inside the KNIME workspace inside the running container (i.e. also folder present on the shared folder) with the GitHub KNIME workflows repository.

The workflow builds on our KNIME metanodes available also on the GitHub repository (https://github.com/OmicsWorkflows/KNIME_metanodes).

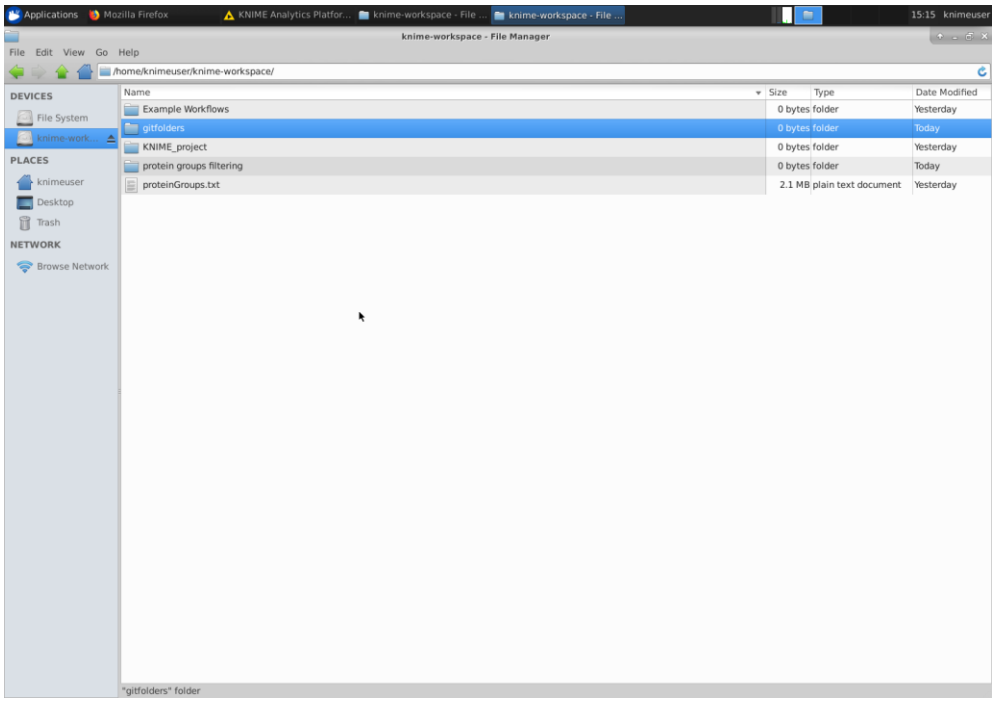
There are two icons on the container desktop (right below the KNIME icon) for the purpose of workflows and metanodes templates reset to the situation on the GitHub – “Workflows templates reset” and “Metanodes templates reset”.

The icons run scripts that needs dedicated folder inside your KNIME workspace – gitfolders. There is a zip file containing the folder with all necessary settings prepared on the GitHub repository: https://github.com/OmicsWorkflows/KNIME_docker_vnc/raw/master/gitfolders.zip.

Download the zip file using Firefox inside the container and extract it inside the downloads folder to get the gitfolders folder:

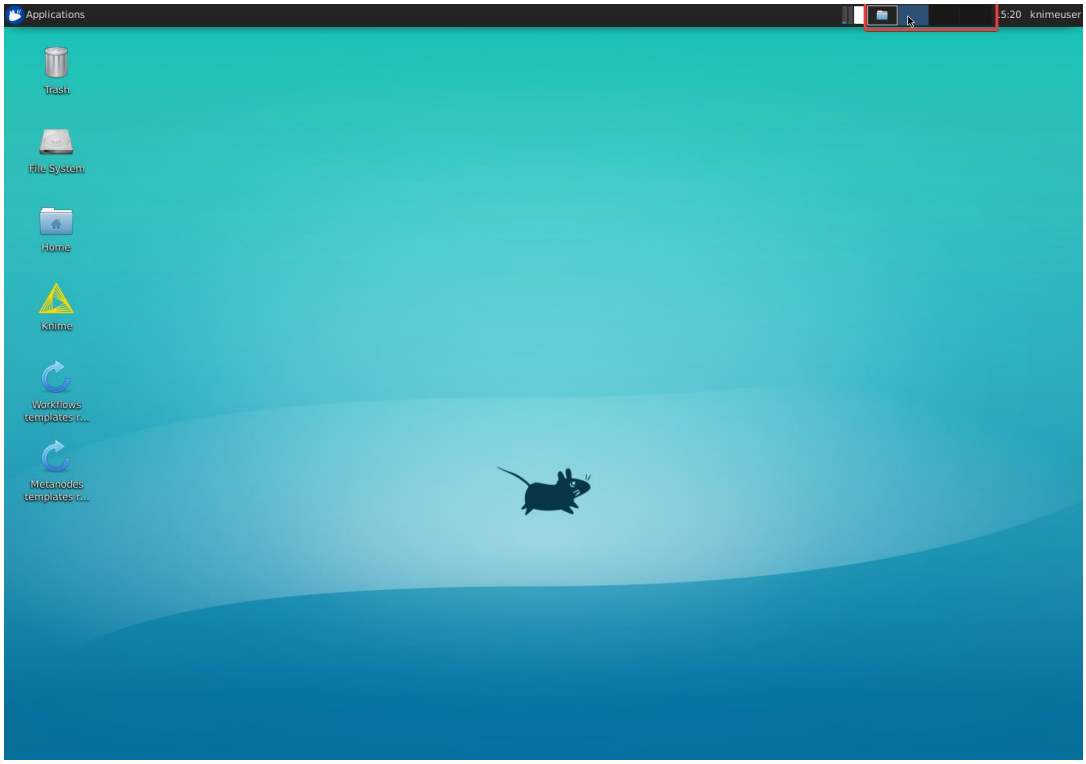


Move or copy the gitfolders folder directly into your workspace (click on the “knime-work...” device in the File Manager window) to get this content of your workspace now:

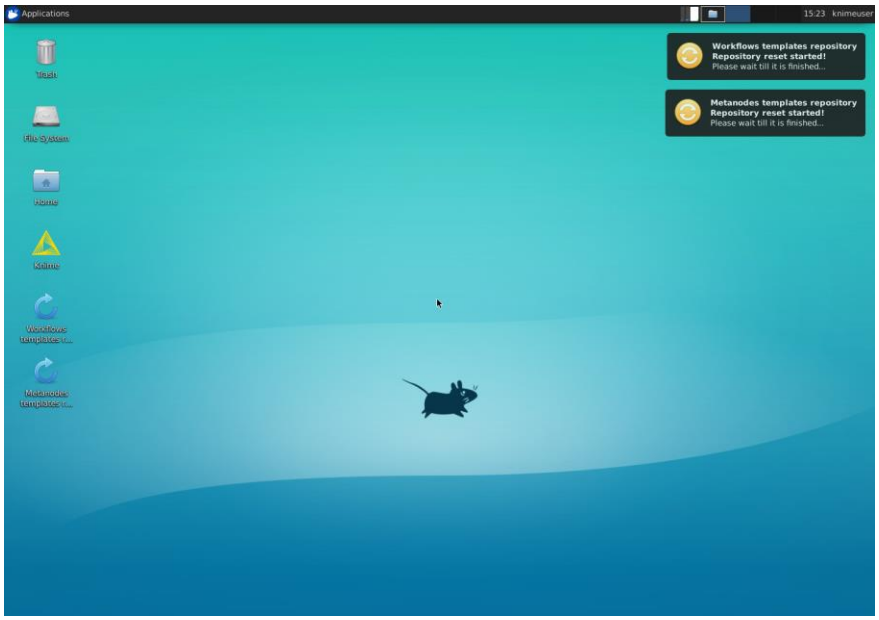


Now you have set up your workspace to work with the two icons to reset the workflows and metanodes templates according to the GitHub repositories.

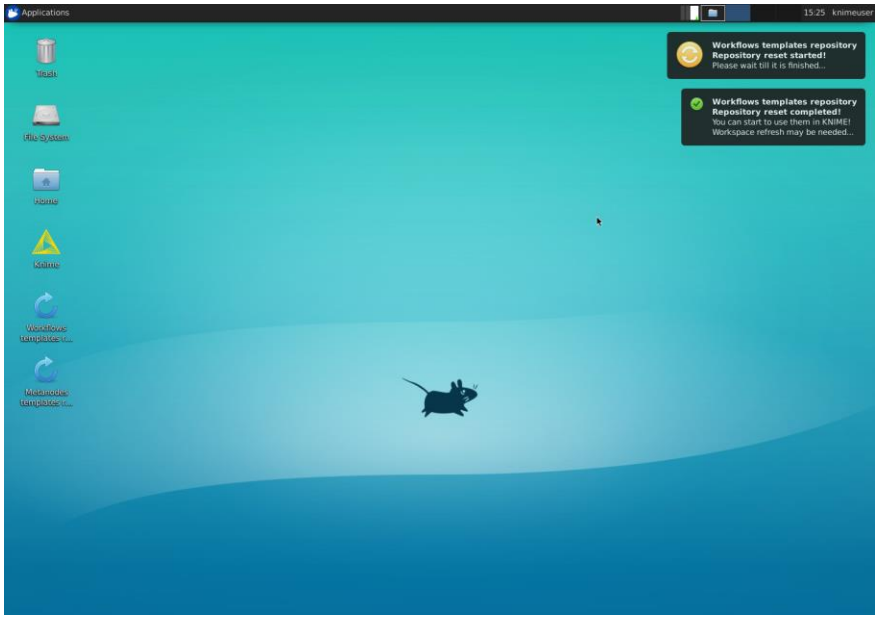
Navigate yourself to the container desktop – there are 4 separates desktops (red rectangle part of the window on the screen shot below) you can use for different purposes later. Click now onto the 2nd desktop (the 2nd one may have the folder icon on it or icon of any other application maximized on that desktop) of your container to get:



Now, double-click on Workflows templates reset icon and afterward also on the Metanodes templates reset icon. There will be an information that the reset process has just started:

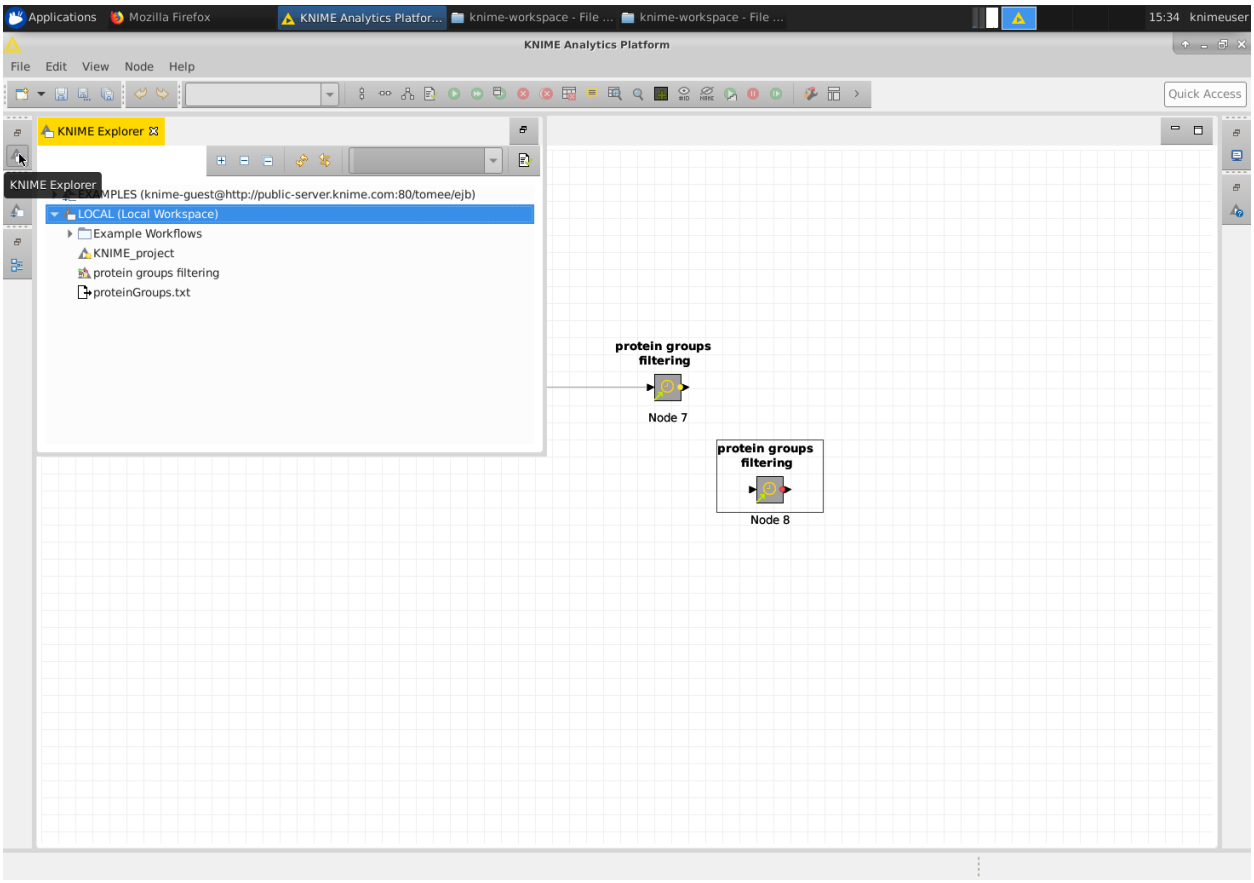


This may take a while based on your network and hard drive speed. There should be another notification message after the successful reset of the given folder – here how it looks like for Workflows templates reset process (top and bottom message is to indicate the start and end of the reset process, respectively):

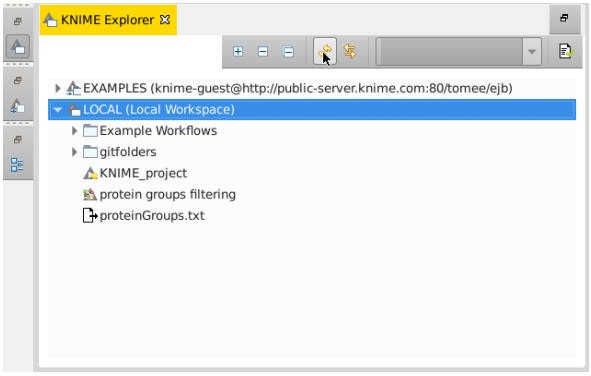


Please note that the gitfolders content (e.g. "KNIME_workflows" directory) is reset to the one on the GitHub page, therefore any changes made inside these folders will get lost if you will initiate the reset procedure! Never save anything into the gitfolders folder because of that!

Navigate yourself back to the first desktop and to the KNIME application. Click on the KNIME Explorer icon to show now minimized KNIME subwindow showing you content of your KNIME workspace:

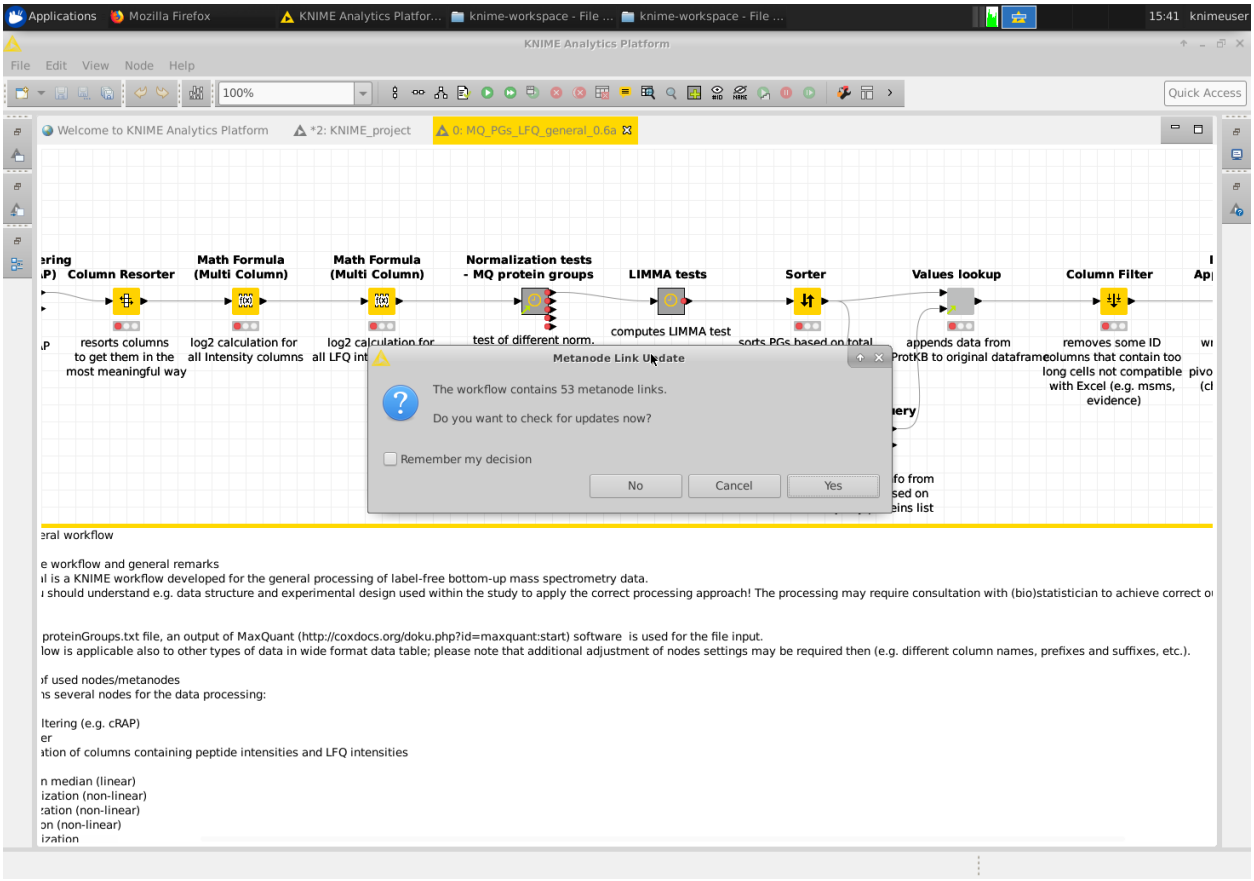


You may notice that there is no gitfolders folder visible in KNIME. This is because of not update view of the folder. Select the “LOCAL (Local Workspace)” line and press Refresh the view button which will update the view and you will be able to see the actual content of your workspace (or alternatively, you can right-click and hold the LOCAL folder -> Refresh):

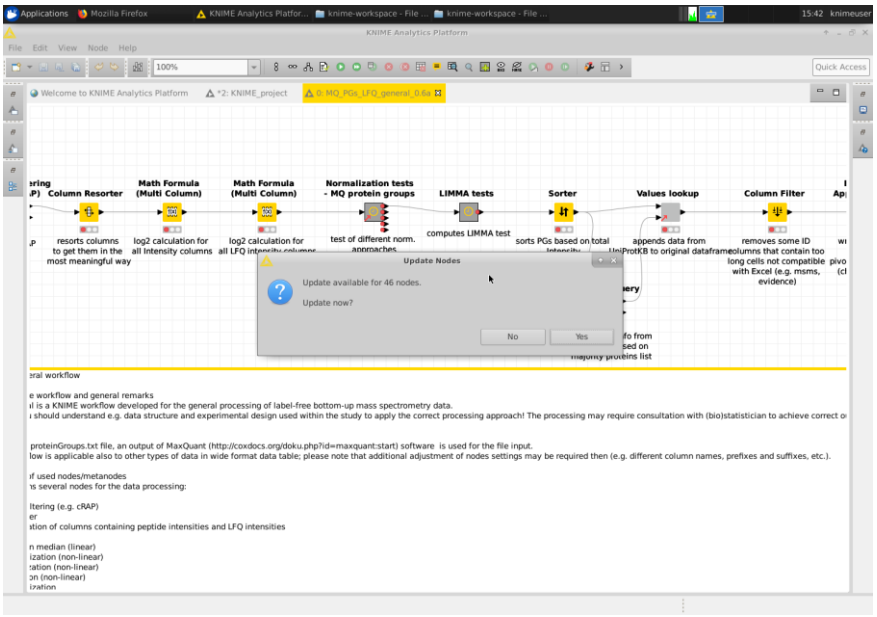


3.2 Opening and preparing our workflow for LFQ data processing

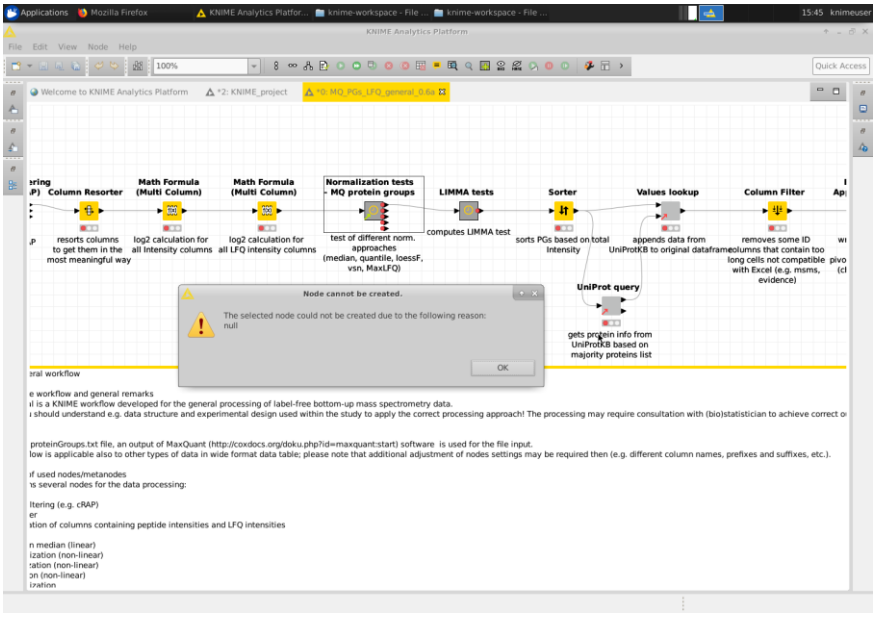
Navigate yourself into KNIME Explorer -> gitfolders -> KNIME_workflows -> Workflow_templates -> MQ_PGs_LFQ_general_0.6a. Double click on the workflow and it will open for you. Depending on your KNIME settings stored inside the KNIME workspace you may be asked (default behavior) whether to update the metanodes present in the workflow, select Yes (do not check the Remember my decision as this is not always wanted and to be asked or to initiate the update manually is safe option):



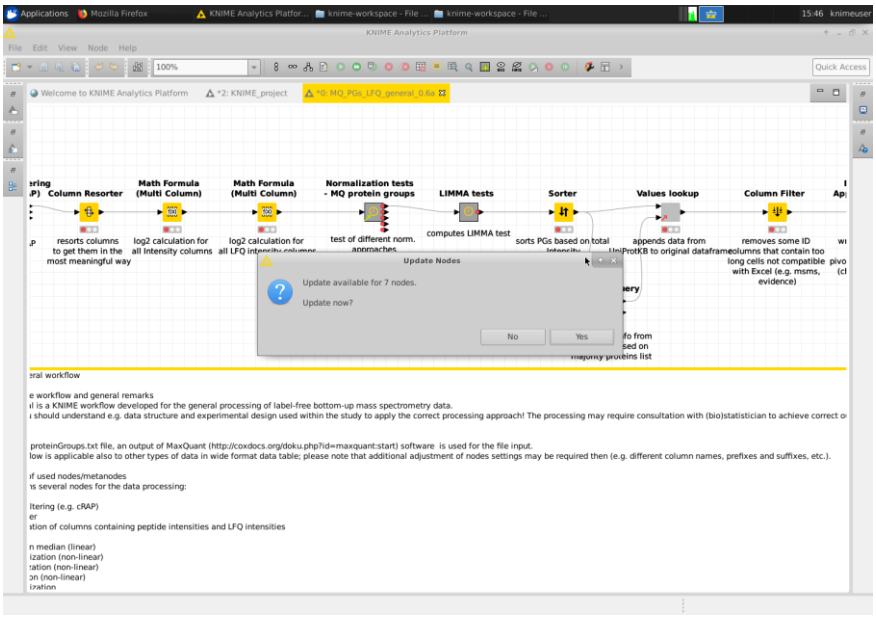
You will be provided with the info on how many metanodes can be updated and whether to update them now, select Yes to update them now:



Sometimes there is an issue with some linked metanodes update, especially when there is some bigger one like the normalization tests metanode. The linked metanodes where the update has failed have red arrow instead of the green one (notice this in case of UniProt query and Values lookup metanode on the following figure):



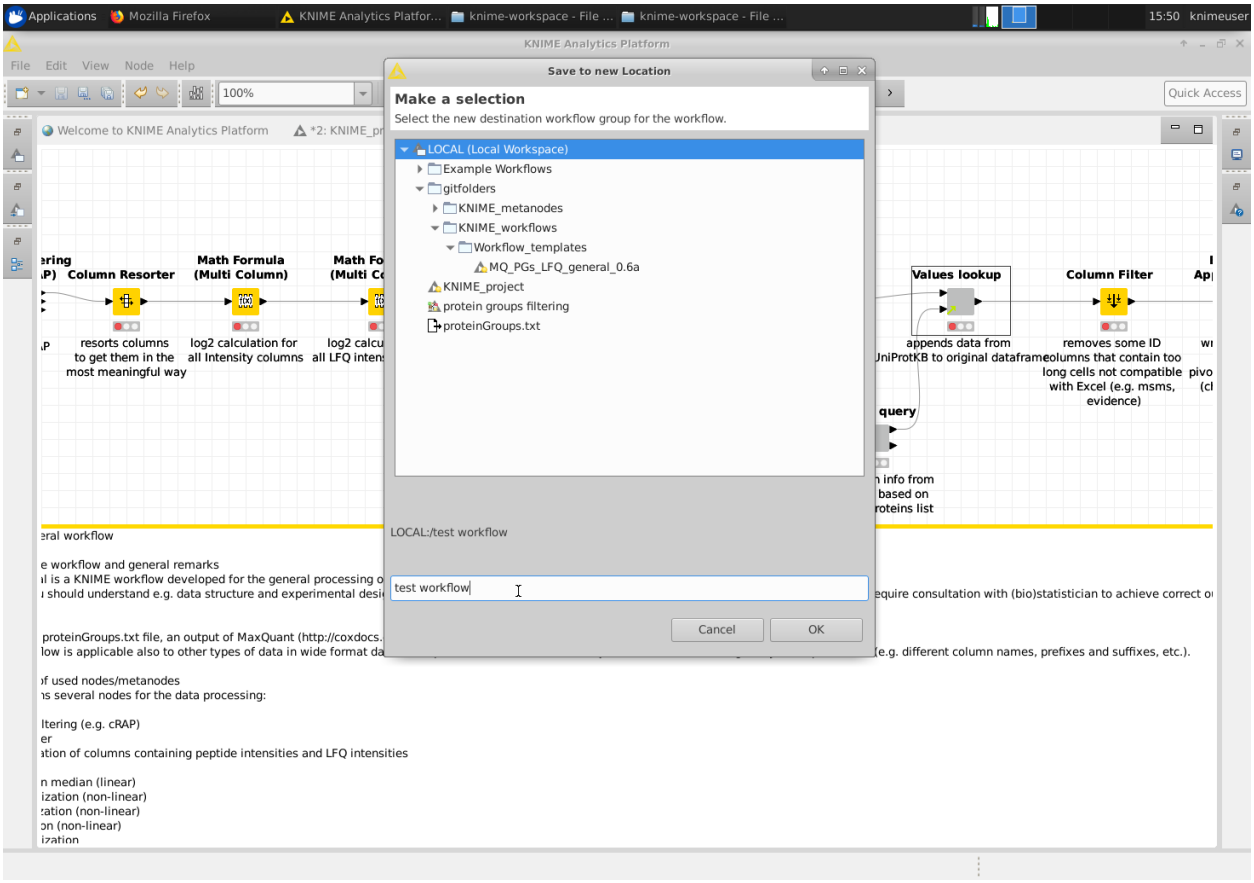
This can be sometimes resolved by updating the metanodes once more (menu Node -> Update metanode links) or individually (right click on the node/metanode -> Metanode -> Update Link):



In this case the second round of update should work properly and should update you all the linked metanodes inside the workflow and you should get info that there are no further updates.

The opened workflow with now updated linked metanodes is still opened from the gitfolders folder so it would get overwritten upon workflows reset, so save it directly into your workspace – File -> Save as...

Select your local workspace and give the new workflow new name, e.g. “test workflow” and hit OK:

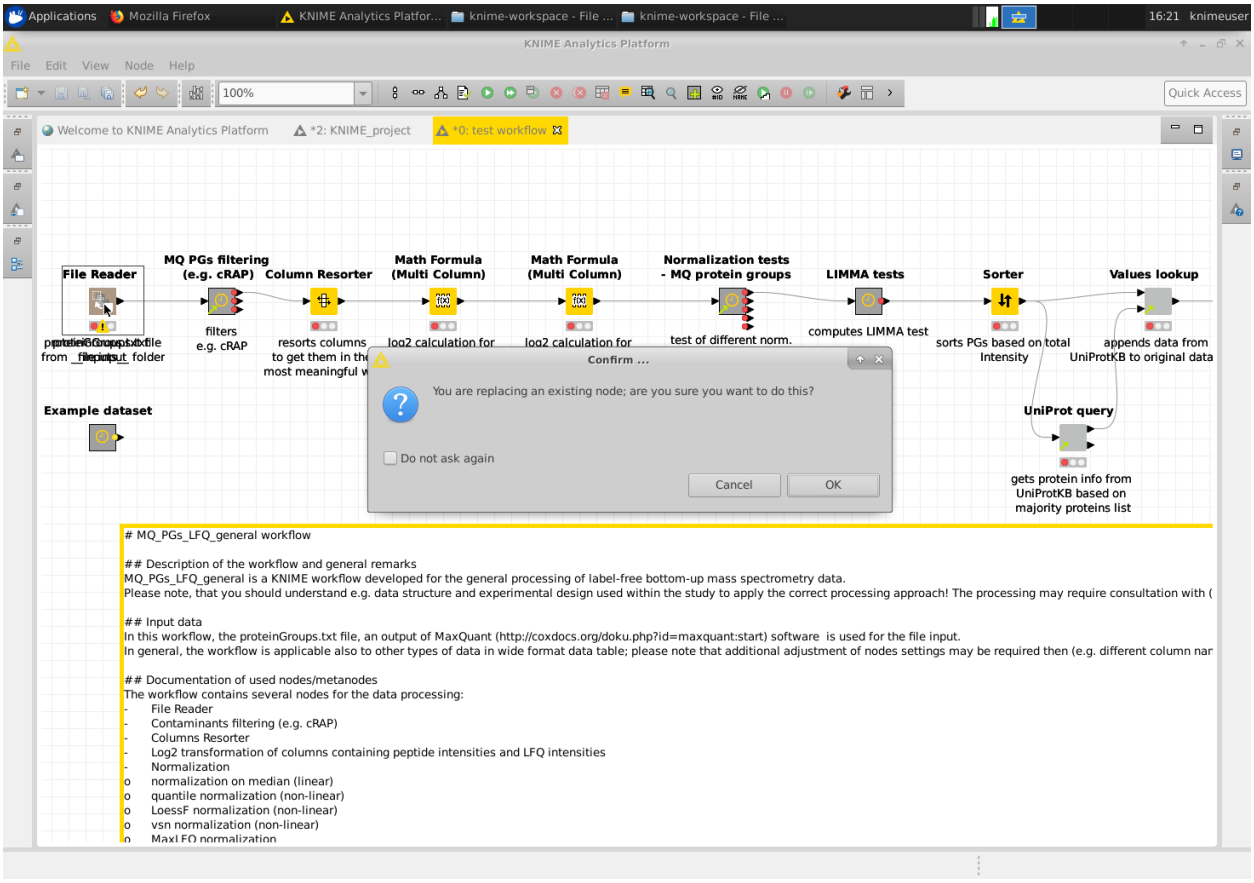


The workflow opening and saving process can be slow for bigger workflows or when you are having your shared workspace folder on network drive or slow hard drive (not SSD).

3.3 Processing the proteinGroups.txt table using the updated workflow – initial orientation and steps

We can now start to process the table we have processed before using this workflow. You can use the File Reader node and set it to read the file from your workspace. But we have that node already set, even though it is in different workflow. Nodes can be copied also among opened workflows so just navigate to our 1st workflow, select the File Reader node and copy it into our new workflow.

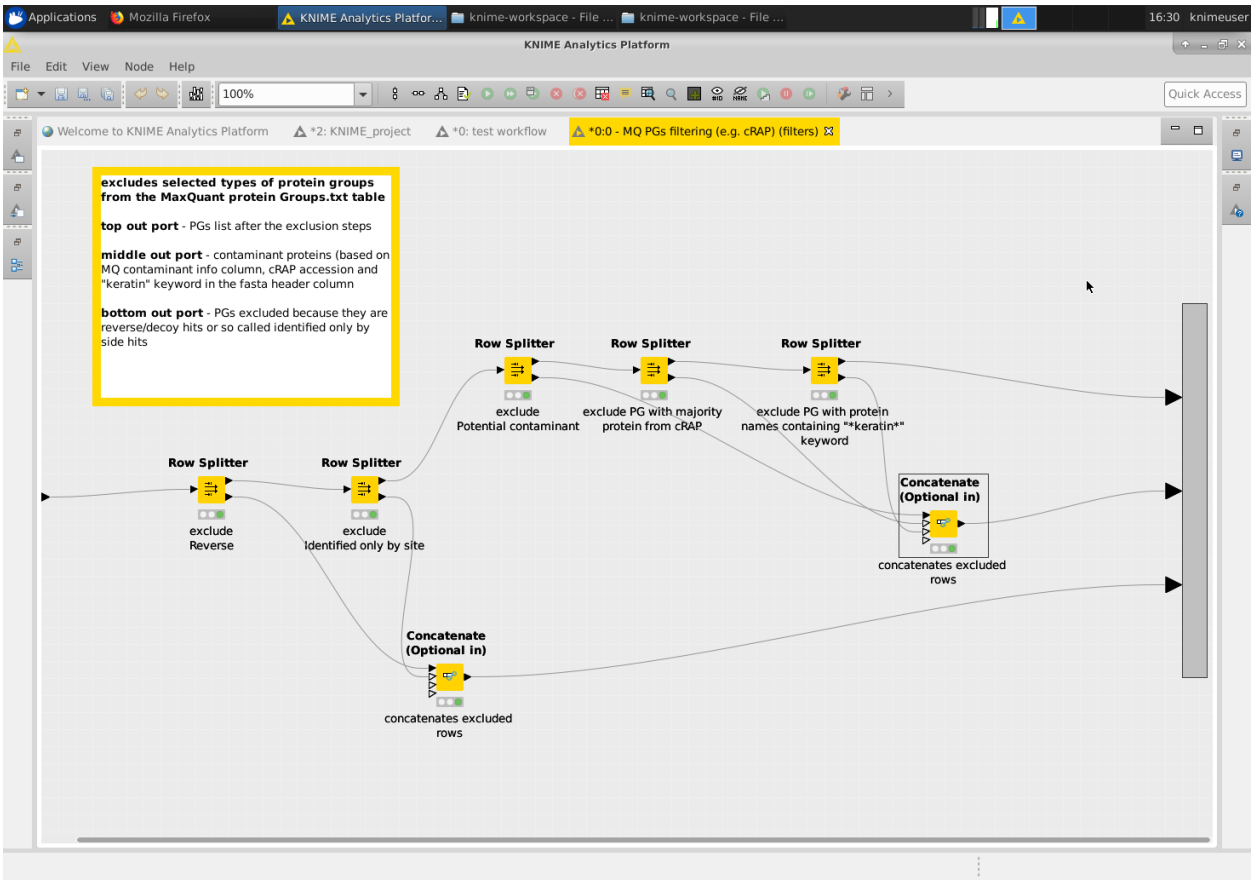
If you want to replace already existing node leaving the connections simply drag and drop new node/metanode onto the node/metanode you want to be replaced which icon changes and you are asked whether you want to replace the current node, confirm this action now:



Now let's get into the actual data processing pipeline.

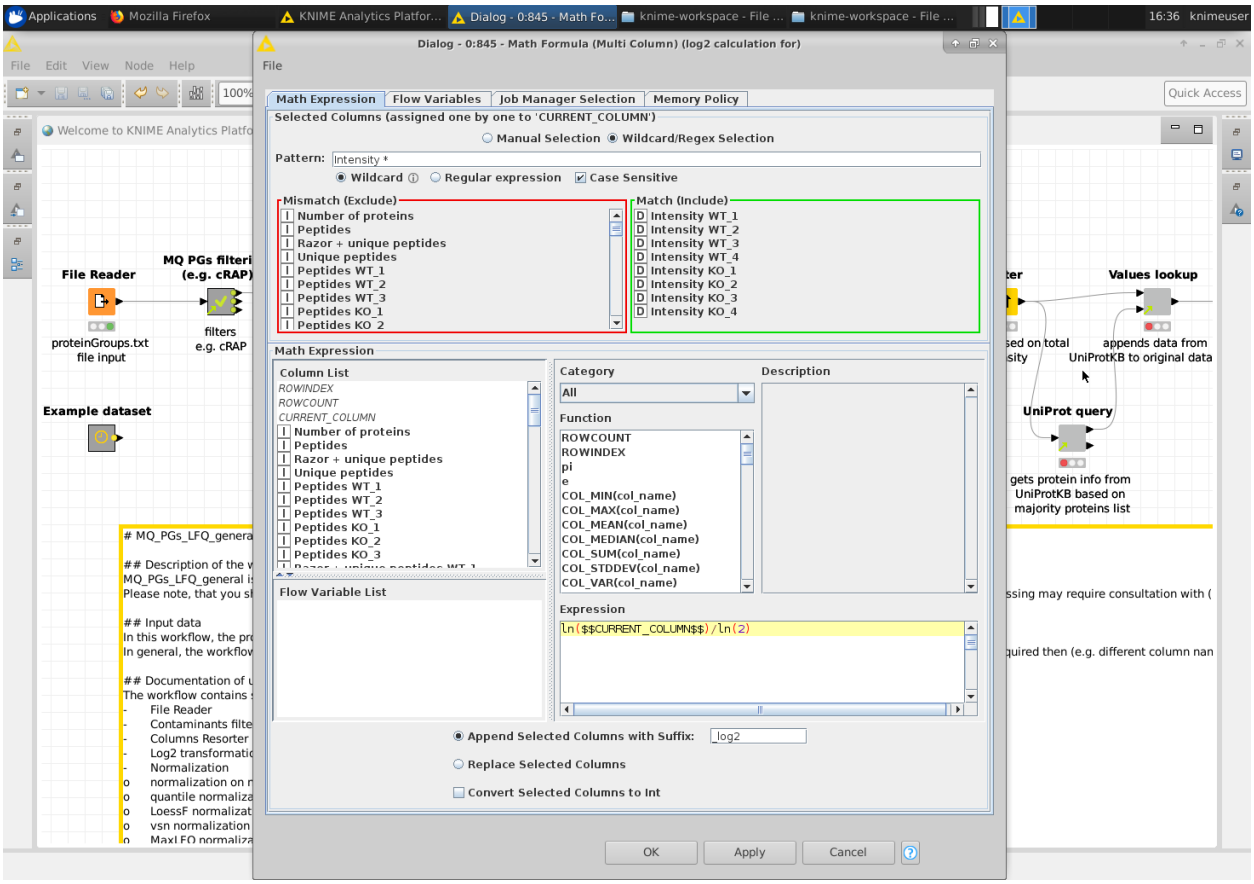
Load the data and process also the metanode to filter out contaminants and other protein groups from the input table. You can do that in one step by running the protein groups filtering metanode – it will initiate any preceding node if needed.

Get inside the protein groups filtration metanode and see it contains similar nodes from our first workflow and on top of that it also collects filtered out rows in two extra nodes and provides you with 3 outports in total: top – filtered table; middle – protein contaminants (cRAP, MaxQuant contaminants and keratins); bottom – reverse and identified only by site protein groups. Notice also the description inside the metanode used to make the meaning of the metanode template clearer to new users:



Next part of the processing pipeline – Column Resorter node – just enables you to resort the columns to make them appear later in the graphics in more meaningful way to make interpretation of the graphics easier. We will ignore it and just go through it without leaving the table untouched by it.

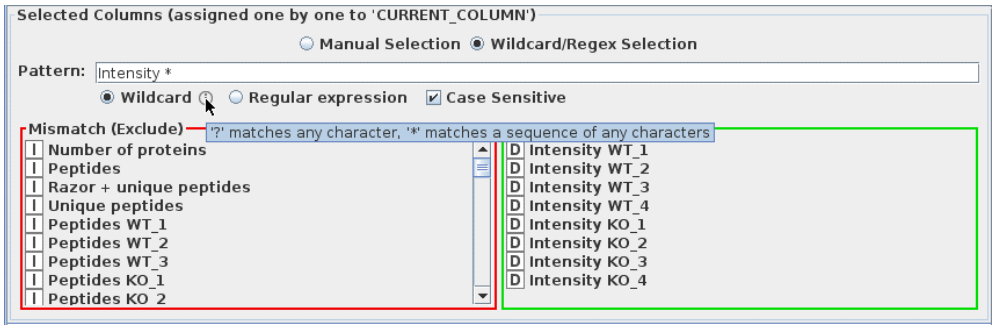
The two next nodes are more interesting for us. You may know that the protein Intensities and LFQ intensities in the MaxQuant proteinGroups.txt table are not normally distributed. Therefore, they need to be transformed e.g. by log2 transformation step. Double click the settings of the 1st log2 transformation node to see its settings saved in the workflow:



As you can see it is already set to automatically select all Intensity columns and do some specified mathematical operation on the selected columns and adding them to the input table with specified column names suffix.

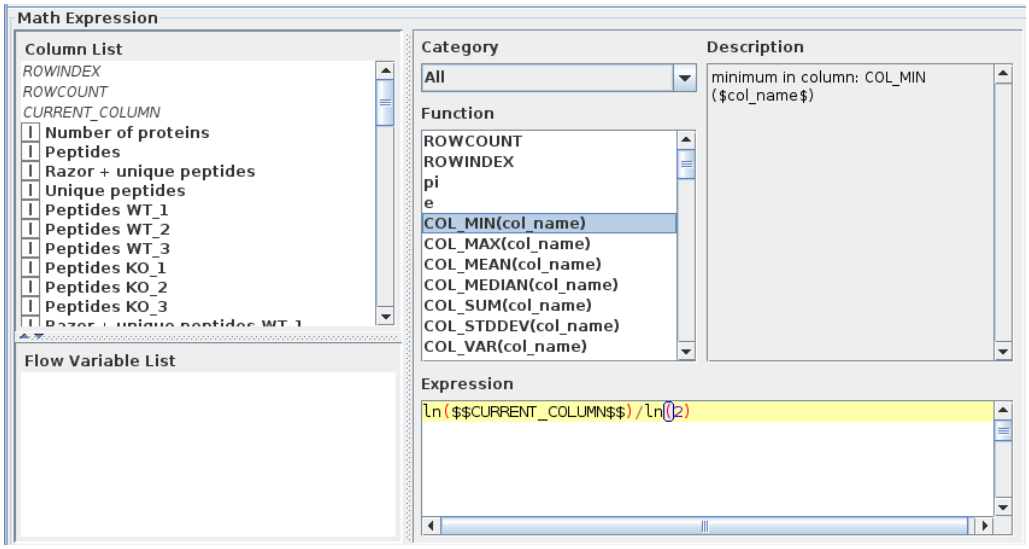
Columns to be processed selection is done using the specified pattern where asterisk is used instead of any number of letters or numbers in the column names. You can select the columns manually, but this way it is more useful and majority of the tools provides you with the column names compatible with this selection, including e.g. Proteome Discoverer output tables.

Let's see how that selection works in more detail. Let's call some help to assist us with columns selection by moving our mouse over the question mark next to the "Wildcard" keyword:

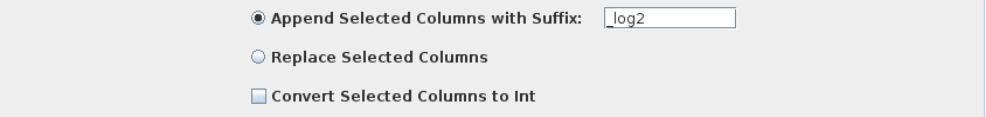


Next to the asterisk we can use also question mark sign as a replacement for single character. This may be handy later. There is also an option to use regular expressions for the columns selection for those who are familiar with them.

The actual mathematical operation is done based on the information written in the "Expression" field of this form. One can use functions specified in the "Function" window for which you will also see get to the help. You can write the function manually or double click on it to paste it into the Expression field. This node enables you to process multiple columns at a time using the same formula. The value from the given column being processed is mentioned in the Expression field by $$CURRENT_COLUMN$$ variable. One can see it also on the left side part of the form inside the Columns list. There you can find other columns which values you can use to define the output of the Expression. You can notice that there are only columns of numerical format unfortunately, so you would have to use some preceding node to get numerical value based on e.g. some text and to use that numerical output here in the form of this Math formula node.



The last part of the form is dedicated to the newly created columns, whether they should be appended with a specified suffix or replace the original ones. Added columns are appended to the very end of the table:



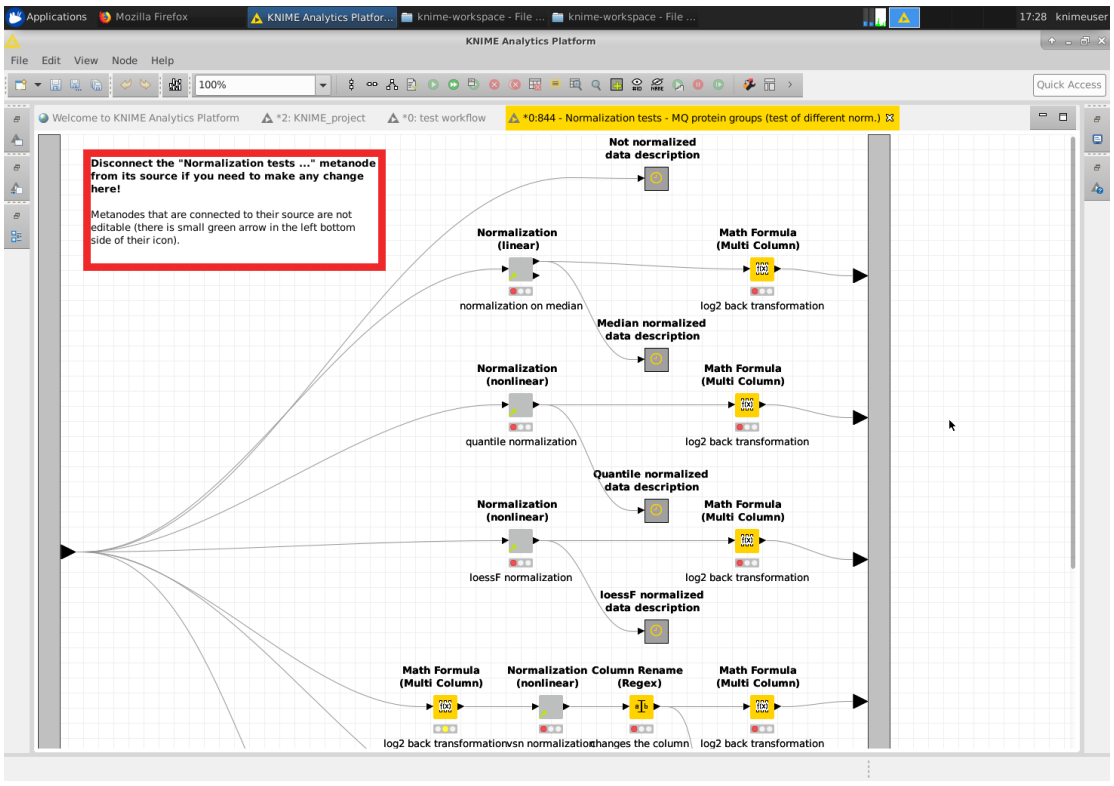
Let's finish with description of this node and process the data using it.

The next node does the same operation, only on LFQ intensity columns. These columns were removed from the table during the dataset preparation not to enable direct comparison of normal and LFQ intensities. We should have the data ready for the test of different normalization procedures done within the "Normalization tests ..." metanode. Save the workflow now not to lose any progress we have made so far.

3.4 Normalization strategies test metanode – overview

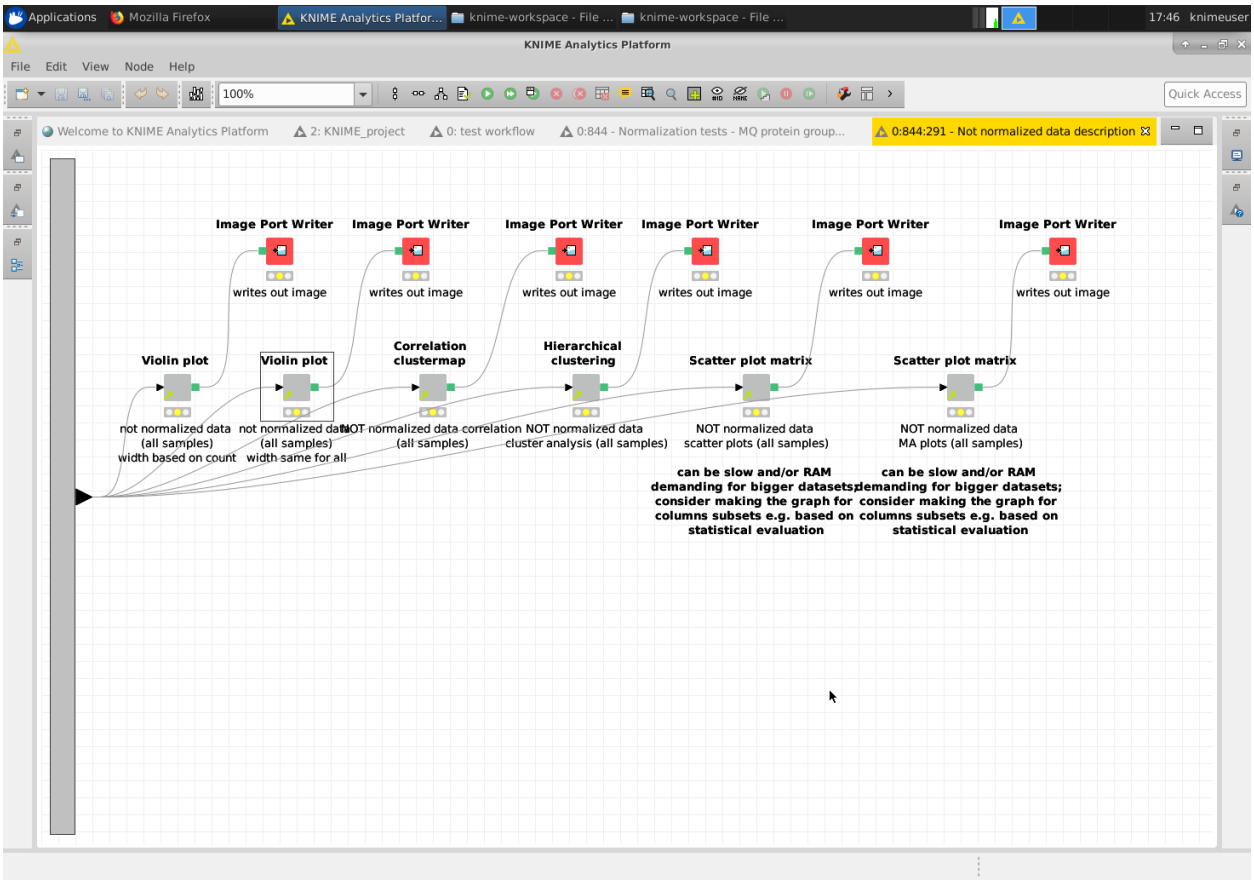
We are going to modify the normalization tests metanode, so disable its link to the metanode template (right click -> Metanode -> Disconnect link).

Let's get into it to see its structure:



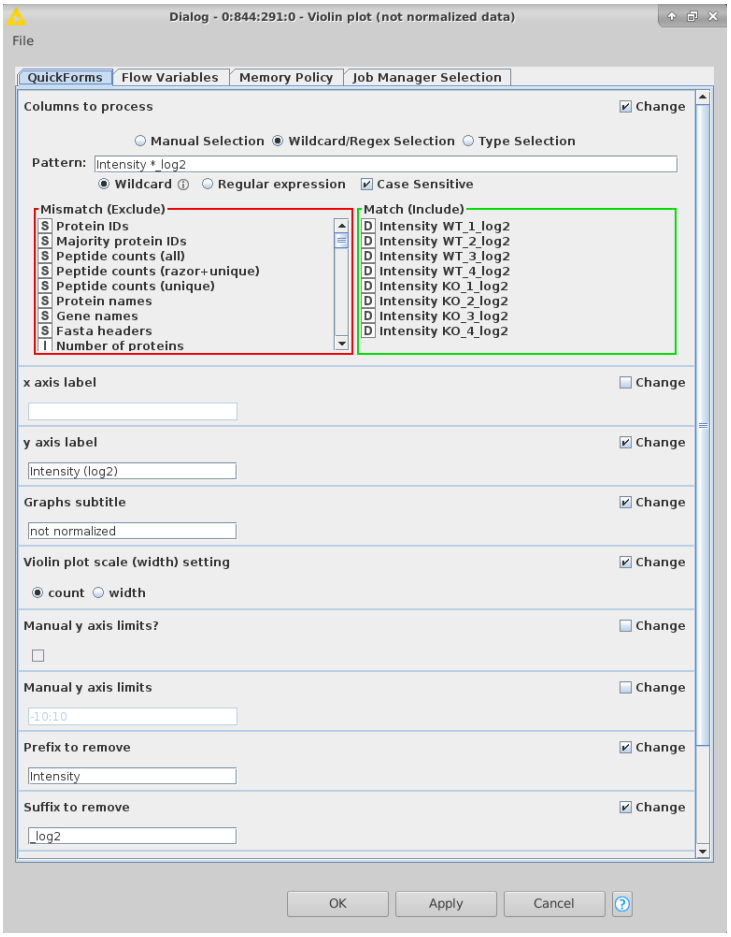
The input port is used multiple times for several normalization strategies and also for description of not normalized and LFQ normalized (done in MaxQuant independently to KNIME processing if enabled) intensities.

When we are processing any data, firstly we want to know their global behavior, so we plot some figures in order to get to know the data matrix more. Navigate yourself further into the “Not normalized data description” metanode on top of the currently opened metanode. New tab will open for you with the content of the inner metanode:

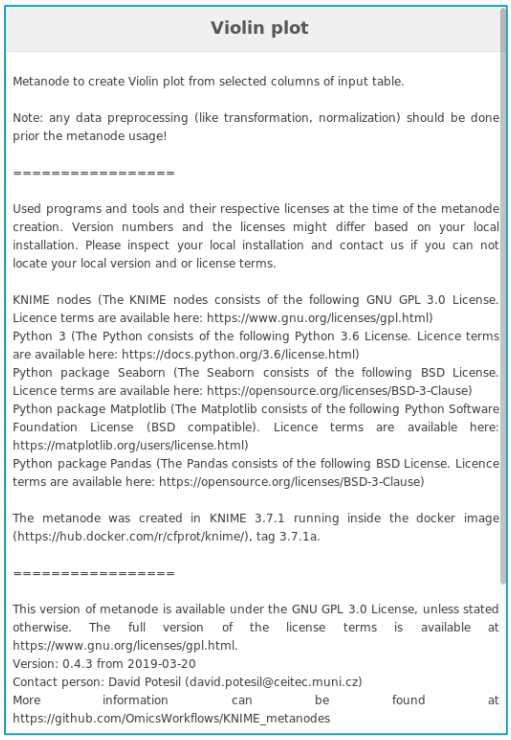


3.5 Wrapped metanodes introduction

Nodes in the bottom line on the figure above are so called “wrapped metanodes” and they are special type of metanodes. They provide you with the form if you double click them (unlike classical metanodes):



We are also trying to make them easy to use and well documented so you can use them on your own and without any scripting knowledge needed on your side. There are used tools and packages/libraries we have used to create the metanode and/or inside the scripts specified in each wrapped metanode description. There is also information on when and where it has been made, its version and where you can find more information about our KNIME metanodes. Each form field has its description as well. All the information is available upon clicking onto the question mark icon next to the Cancel button in the wrapped metanode form. Here is the description of the Violin plot metanode for example:



Violin plot

Metanode to create Violin plot from selected columns of input table.

Note: any data preprocessing (like transformation, normalization) should be done prior the metanode usage!

==========

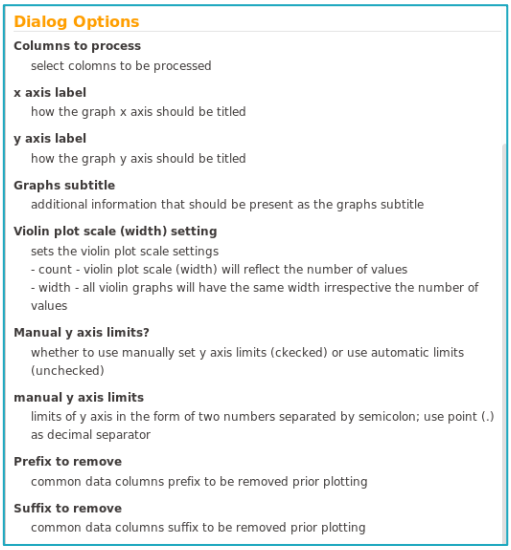
Used programs and tools and their respective licenses at the time of the metanode creation. Version numbers and the licenses might differ based on your local installation. Please inspect your local installation and contact us if you can not locate your local version and or license terms.

KNIME nodes (The KNIME nodes consists of the following GNU GPL 3.0 License. Licence terms are available here: https://www.gnu.org/licenses/gpl.html)
Python 3 (The Python consists of the following Python 3.6 License. Licence terms are available here: https://docs.python.org/3.6/license.html)
Python package Seaborn (The Seaborn consists of the following BSD License. Licence terms are available here: https://opensource.org/licenses/BSD-3-Clause)
Python package Matplotlib (The Matplotlib consists of the following Python Software Foundation License (BSD compatible). Licence terms are available here: https://matplotlib.org/users/license.html)
Python package Pandas (The Pandas consists of the following BSD License. Licence terms are available here: https://opensource.org/licenses/BSD-3-Clause)

The metanode was created in KNIME 3.7.1 running inside the docker image (https://hub.docker.com/r/cfprot/knime/), tag 3.7.1a.

==========

This version of metanode is available under the GNU GPL 3.0 License, unless stated otherwise. The full version of the license terms is available at https://www.gnu.org/licenses/gpl.html.
Version: 0.4.3 from 2019-03-20
Contact person: David Potesil (david.potesil@ceitec.muni.cz)
More information can be found at https://github.com/OmicsWorkflows/KNIME_metanodes



Dialog Options

Columns to process
select colomns to be processed

x axis label
how the graph x axis should be titled

y axis label
how the graph y axis should be titled

Graphs subtitle
additional information that should be present as the graphs subtitle

Violin plot scale (width) setting
sets the violin plot scale settings
- count - violin plot scale (width) will reflect the number of values
- width - all violin graphs will have the same width irrespective the number of values

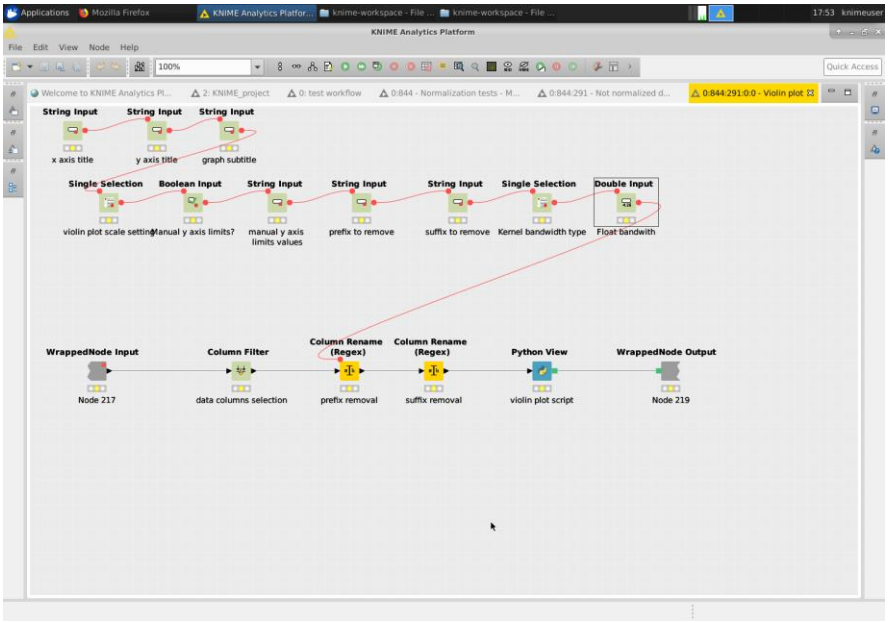
Manual y axis limits?
whether to use manually set y axis limits (ckecked) or use automatic limits (unchecked)

manual y axis limits
limits of y axis in the form of two numbers separated by semicolon; use point (.) as decimal separator

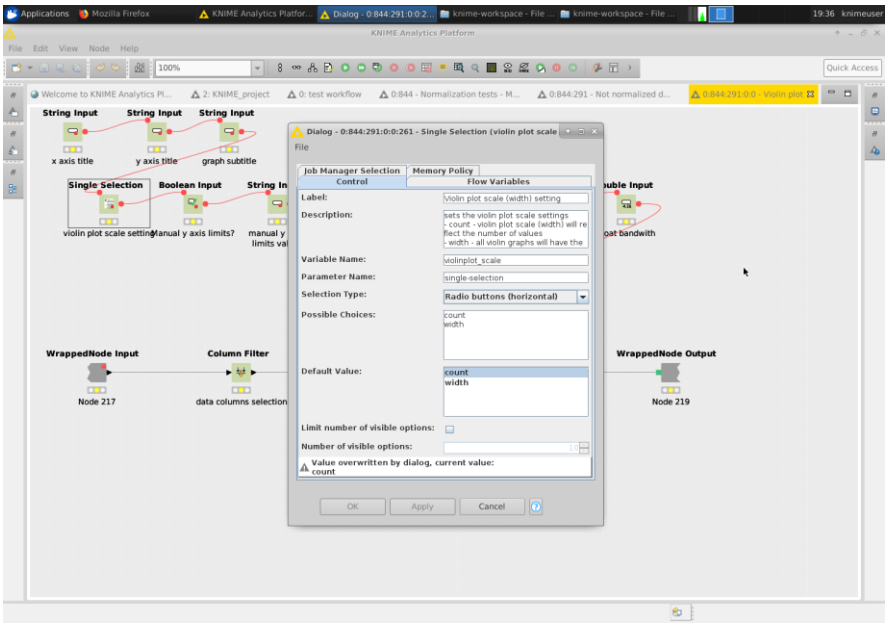
Prefix to remove
common data columns prefix to be removed prior plotting

Suffix to remove
common data columns suffix to be removed prior plotting

Wrapped metanodes form and all what it does is available for you to be checked and modified if needed. The Violin plot wrapped metanode inner structure looks like this (accessible via double click while holding Ctrl or right click -> Wrapped metanode -> Open):



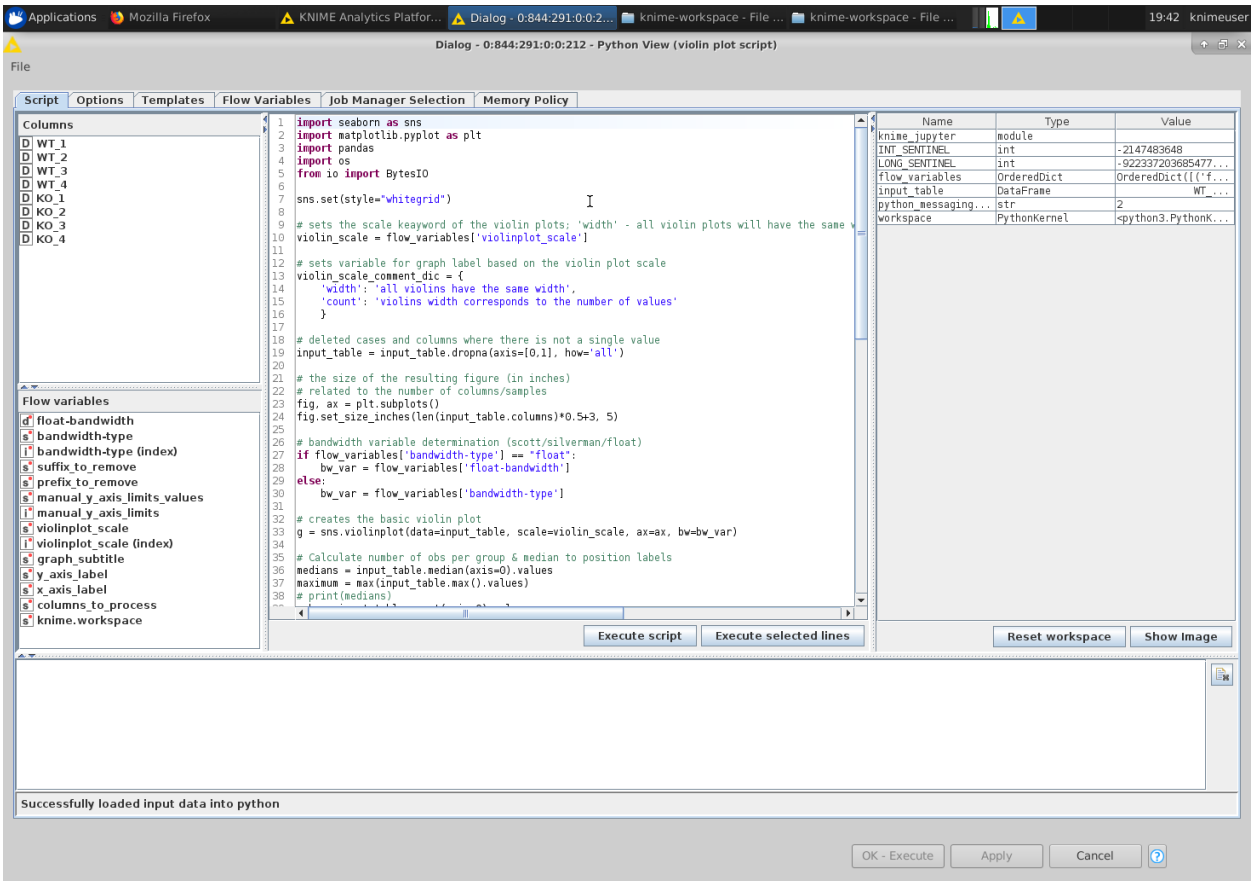
There are quickform nodes (green color) that user of the wrapped metanode can see in the actual form. Their settings are accessible in the same way as the settings of any other node, by double clicking on them. Here what the settings of the quickform node looks like for the violin plot scale settings:



There are the texts that wrapped metanode user can see in the wrapped metanode form (label, description and possible options with the default value) and how the settings should be shown in the

form. There is also variable name that is possible to be used in the following script. This way one can visually “program” the form the way they like and find suitable for the given purpose.

In case you would be interested or would like to modify the wrapped metanode behavior you can access the script as well. In the case of Violin plot wrapped metanode, the script is inside the Python view node, again accessible upon double click:



The central part contains the actual python script that does the visualization, there is also list of flowvariables (left middle part) and columns available for the script at the given moment (left upper part). On the right side there is a list of variables used and set by the script. If you press Execute script button, you will be able to show the result of the script via Show image button. This can be useful in case you would like to modify or troubleshoot the script if needed.

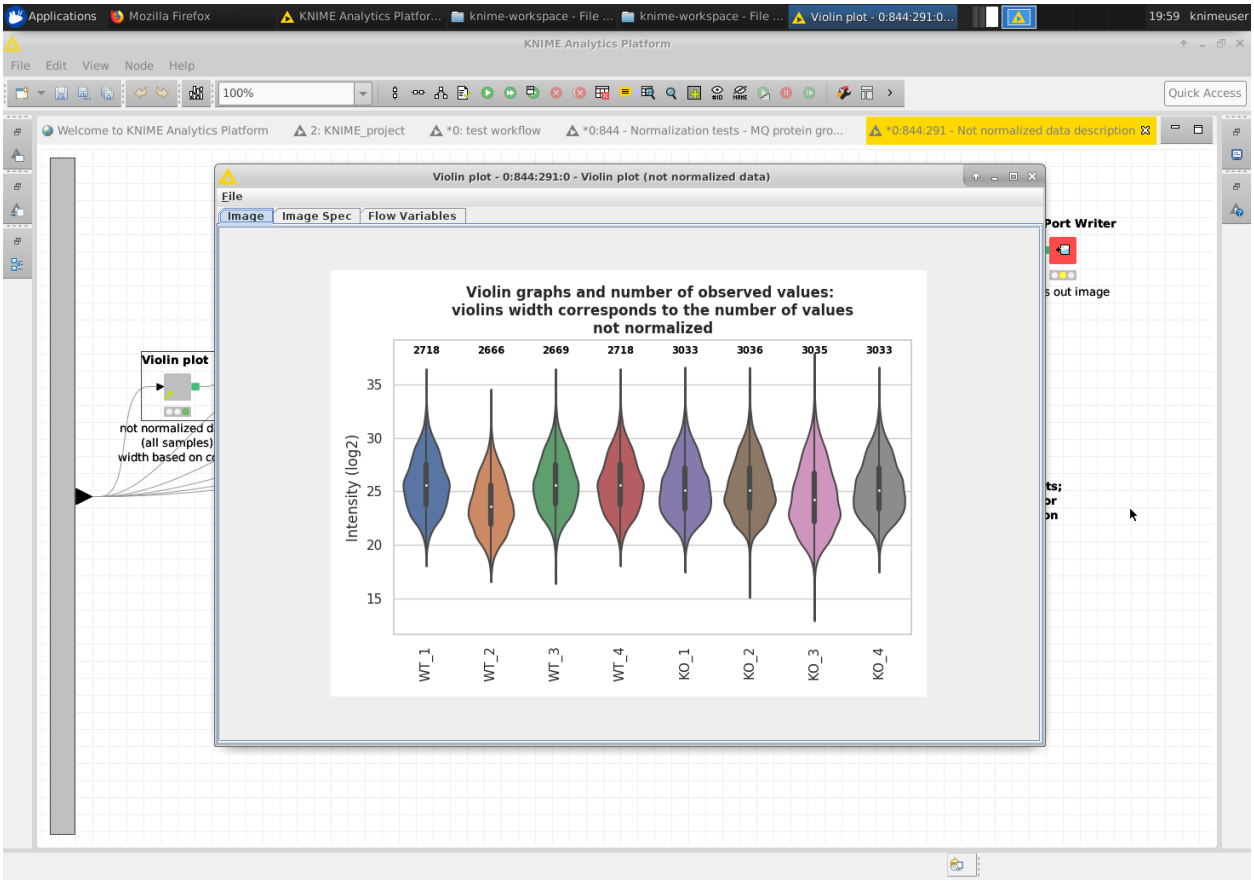
This way you can check all wrapped metanodes in the GitHub repository, so if you really need to know what the given metanode does and how, or whether it does it the way you understand or you want to make some modification, you should have all you need to achieve that. Keep in mind that you will be not able to modify the given wrapped metanode unless you disconnect it from its template as for any other linked metanode.

The form of the metanodes works the same way as of any other in-built node in KNIME.

3.6 Normalization strategies test metanode – not normalized data description

Let's get back to the normalization step. It is good to know your data prior the selection/verification of the right normalization strategy. Go into the normalization tests metanode and Not normalized data description inner metanode. Run the 1st four metanodes from the left, i.e. two violin plots and two cluster analysis metanodes.

Violin graphs are good way to check the overall distribution. The graph shows also the number of values in each column so you can get easily information about the data distribution and total counts. To see the violin plot produced by the most left metanode right click on it and select Violin plot option, new window will be shown to you displaying the violin plot:



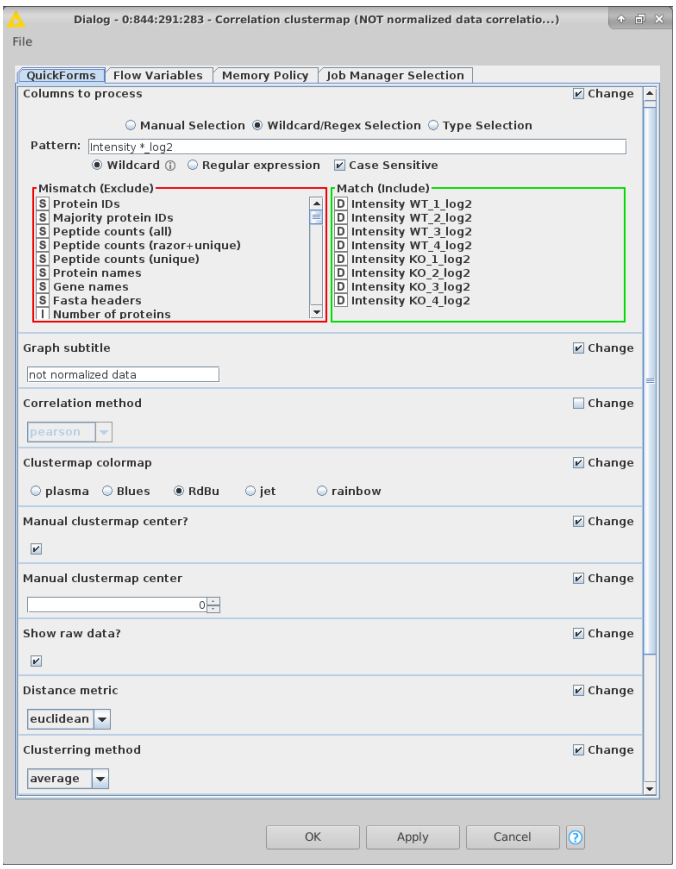
Here we can notice several things already:

- the number of protein groups is on similar level, but WT sample has lower number of them in general
- the violin plots are at the same time comparable with their median value position, WT sample replicates are a bit more intense taking median value into account
- there are two strange samples – WT_2 and KO_3 that have different distribution to other replicates of the same sample

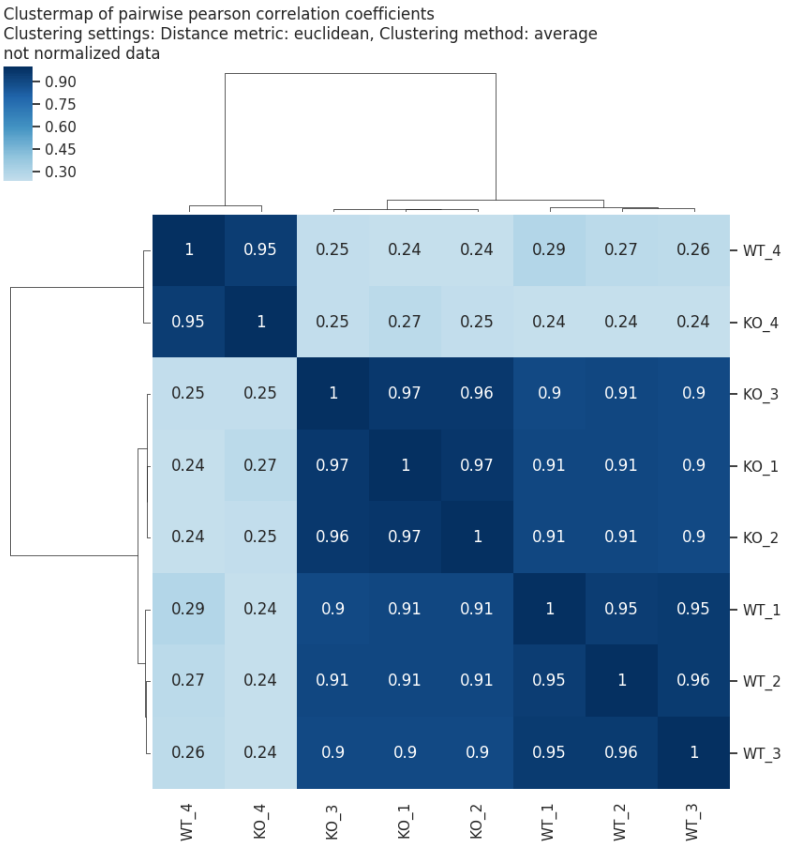
- the distribution of values within individual replicates is roughly comparable among replicates and among different samples.

Here the violin plots width is based on actual number of protein groups in the given sample replicate. In case of hugely different protein group numbers it may be good to draw violin graphs having the same width irrespective to the number of protein groups to see the changes in the distribution – that is the reason for the second violin plot metanode. In this case the differences are not so big so the differences between the two violin plots are not so evident.

Another type of graph that can help us to check overall data quality is the Correlation clustermap – 3rd metanode from the left. It does cluster analysis using correlation coefficients. The settings of the metanode is as follows:



The metanode computes pairwise correlation coefficients and uses them for the following cluster analysis. Correlation method as well as cluster analysis settings can be specified. Correlation clustermap enables us to again compare samples based on their pairwise correlation coefficients – which samples tend to form clusters – are similar in their pairwise correlation coefficients? How much are the samples similar to each other (correlation coefficient)? Right click on the metanode and select the last option to get the figure:

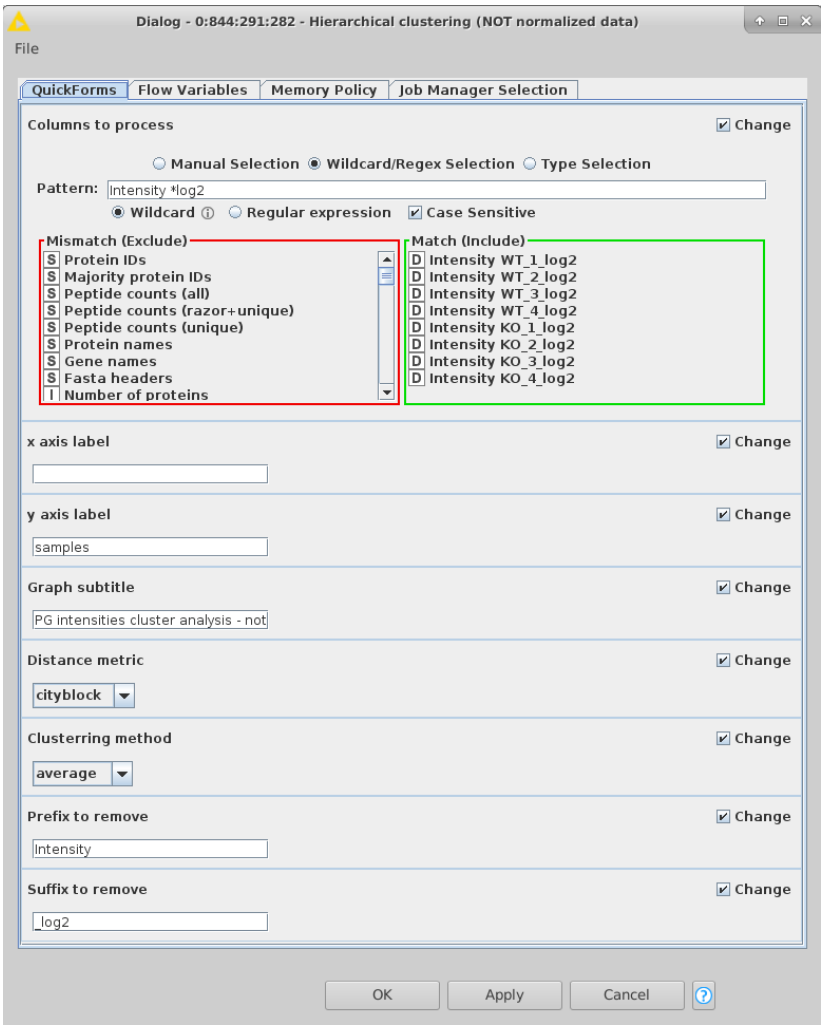


One can clearly see that two sample replicates are off from all other ones. And, a bit surprisingly, these are 4th replicates in each sample type where we have not been able to see any difference on the violin plots above. The correlation coefficients (Pearson) are relatively high for all other individual sample replicates and also correlation among different sample types (replicates 1-3) is relatively high (~0.9). So, the differences observed in the violin plots for the 2nd and the 3rd replicate of WT and KO sample, respectively, do not have to be so critical.

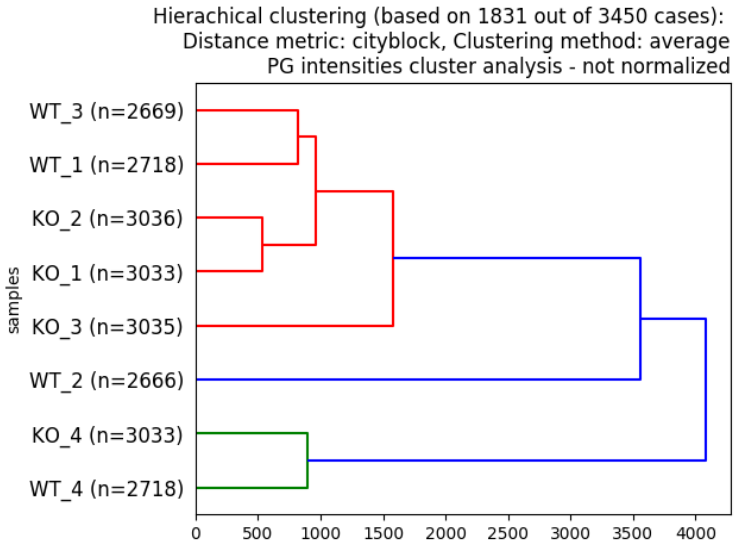
It is worth notice, that by default, the workflow has the metanode cluster analysis settings set in a way it increases the differences in the correlation coefficients (distance metric Euclidian) as there is only one number used for the cluster analysis. This way higher differences in the correlation coefficients are more pronounced in the cluster analysis results (distance).

Another type of cluster analysis can be done using the Hierarchical clustering metanode. Here the raw log2 transformed intensities are used directly for the cluster analysis, but because of the cluster analysis

procedure and not complete data matrix, all rows containing NA values only, i.e. protein groups, not having intensity value in all 8 columns in our case, are excluded prior the cluster analysis. This is somewhat limiting, but should give more appropriate cluster analysis results – missing values can be also tackled so they should not pose any problem here if needed (imputation topic will be discussed further). The form of the metanode looks like this:



And the corresponding cluster analysis result is as follows:

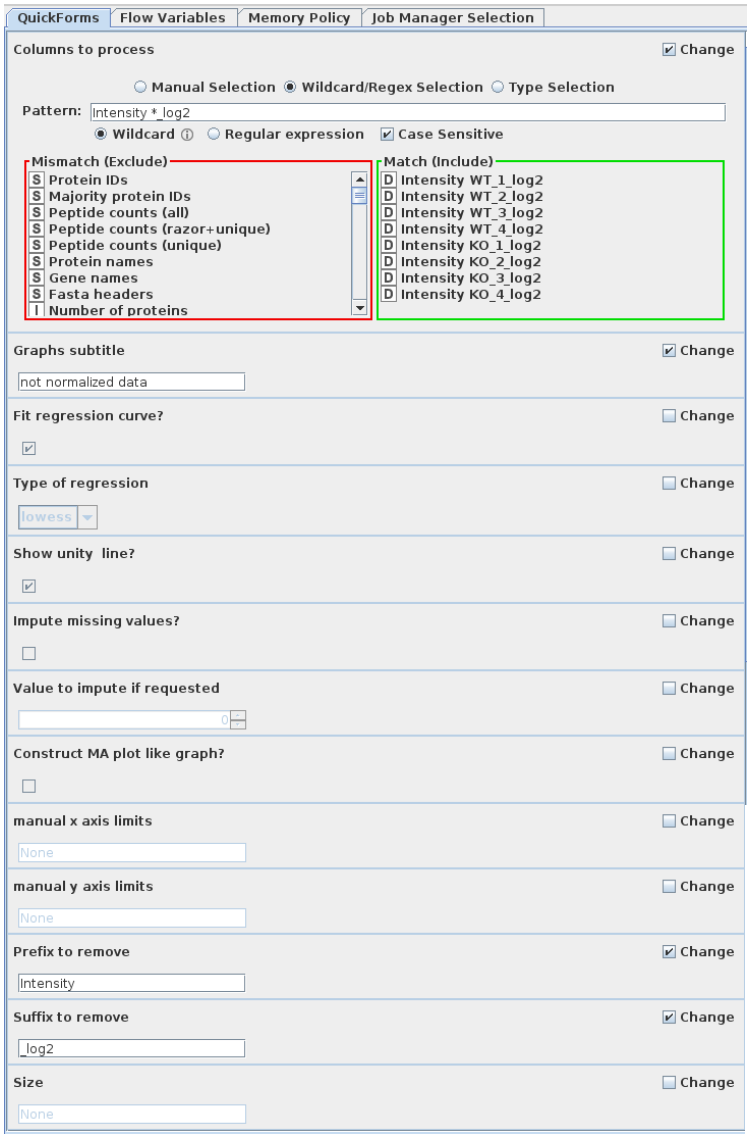


The figure legend tells what is the number of cases/rows/protein groups used for the cluster analysis (1831 out of 3450 in total) and there are also actual number for individual sample replicates. We can notice here the same trend as in the first cluster analysis – the 4th replicates tend to cluster together and off from other replicates. The result for the two sample replicates being a bit off on the violin plots suggest they differ in terms of individual protein group intensities comparison.

It is worth notice, that, by default, the workflow has the metanode cluster analysis settings set in a way it does simple summing of the differences (cityblock distance metric), not raising them to a higher power (Euclidian) not to influence the results of the cluster analysis completely by few outlying protein groups, but rather equal contribution to the distance by each of them.

There are few replicates that might be off based on the figures we have seen so far. At this point it is good to check how the correlation graphs actually look like for individual pairs. That is why there are two other metanodes in the row – Scatter plot matrix. These are making e.g. log2 transformed protein group intensity correlation for all possible sample pairs with potential imputation of missing values by the specified constant if selected. It shows also the expected ideal behavior in case one would not see any difference, i.e. x = y or so-called unity line. On top of that, there can be also regression curve displayed (linear or lowess). This should enable us to say how similar the two samples are from individual scatter plots and how the correlations compare to correlations with other samples.

The form of the Statter plot matrix metanode looks like this:

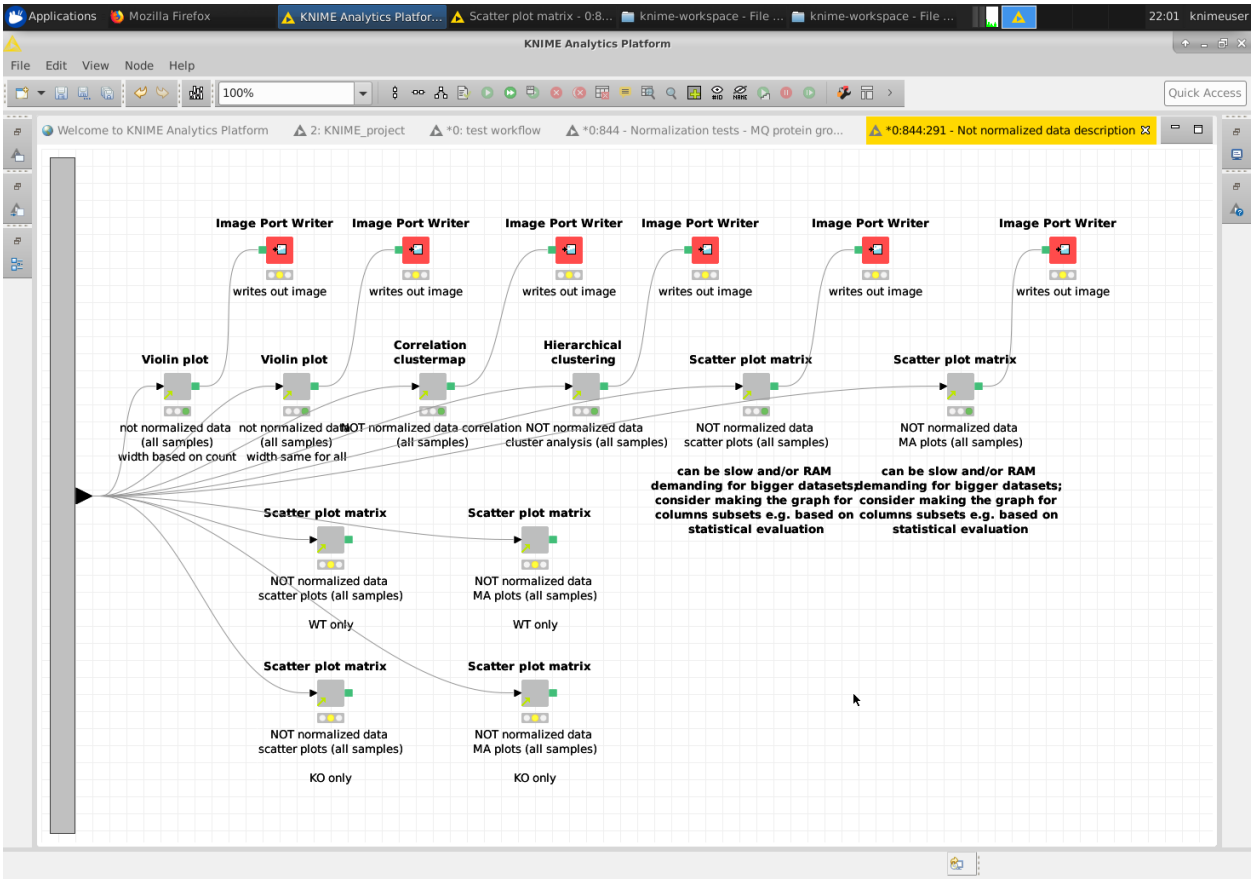


One important setting is “Construct MA plot like graph?” check box. If checked, MA plot instead of standard dependence of “y” on “x” will be produced. MA plot like graph displays dependence of “x – y” (i.e. difference between the two sample replicates) on “(x + y)/2” (i.e. average value from the two sample replicates).

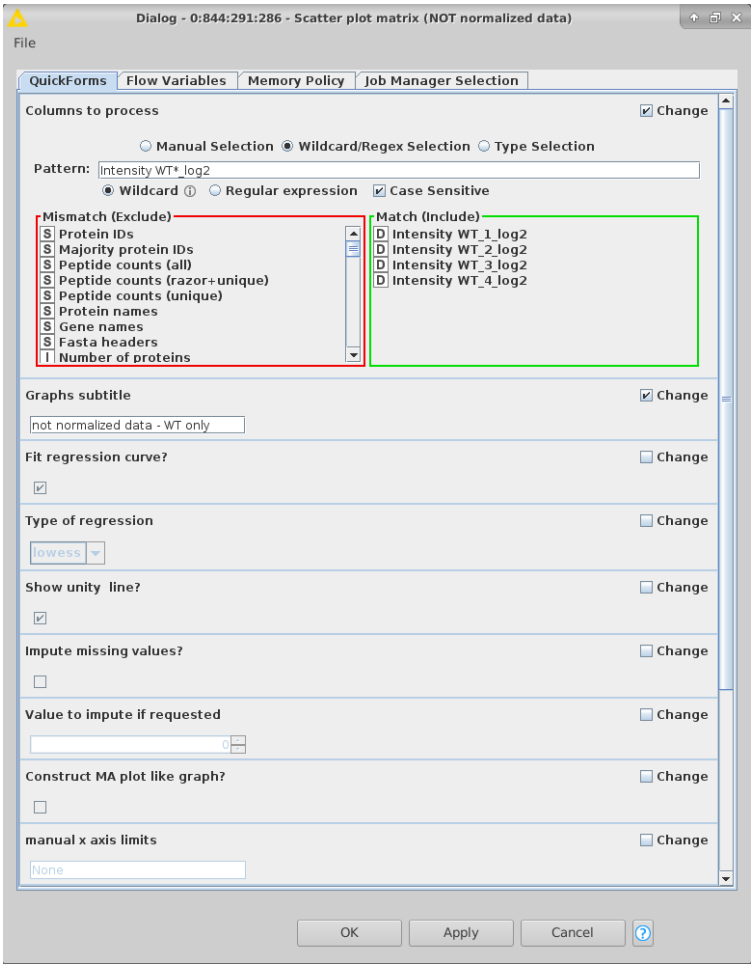
These scatter plot graphs can be quite big and in case you have high number of protein groups and higher number of columns to display they can take few minutes to be computed. It is good practice not to start higher number of these metanodes than the number of available processing cores.

So, let’s at first run the two nodes with predefined settings. Meanwhile, let’s do also graphs for individual sample types, i.e. for WT and KO separately. Select the two now running metanodes and make two additional copies of them directly into the same place. Move the copies a bit aside and connect them with

the input port of the metanode. Adjust the description of the 1st copy to tell it is only for WT sample replicates and similarly the 2nd copy for KO sample replicates. You should see the inner metanode structure similar to this:

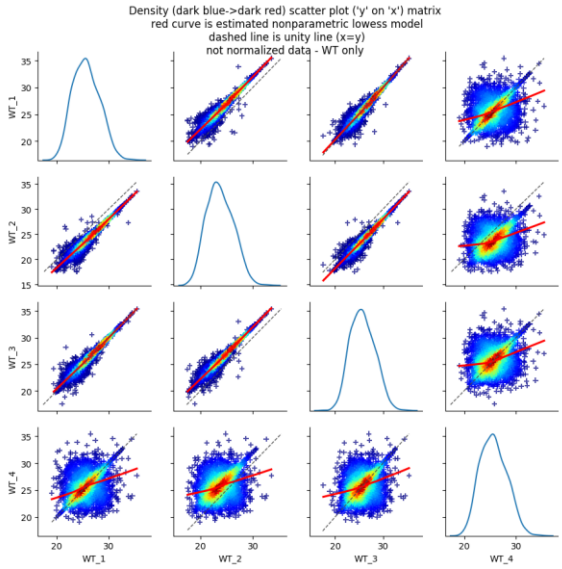


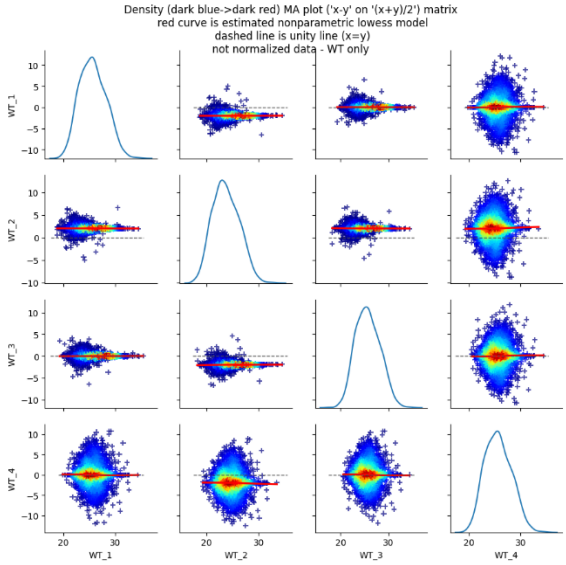
Now setup the two metanodes for both, scatter plots and MA plots, for WT and KO samples only. Modify the columns selection wildcard string to contain either WT or KO prior the asterisk symbol (Intensity WT*_log2 or Intensity KO*_log2 respectively):



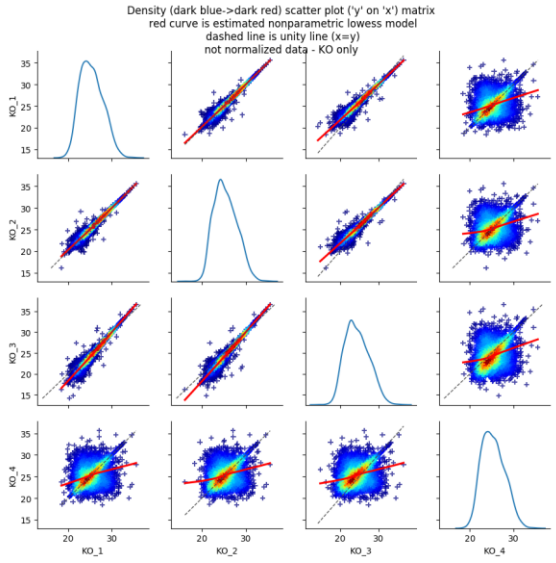
Once the two originally present metanodes will be executed, initiate the graphs creation for individual sample types, separately for WT and KO samples.

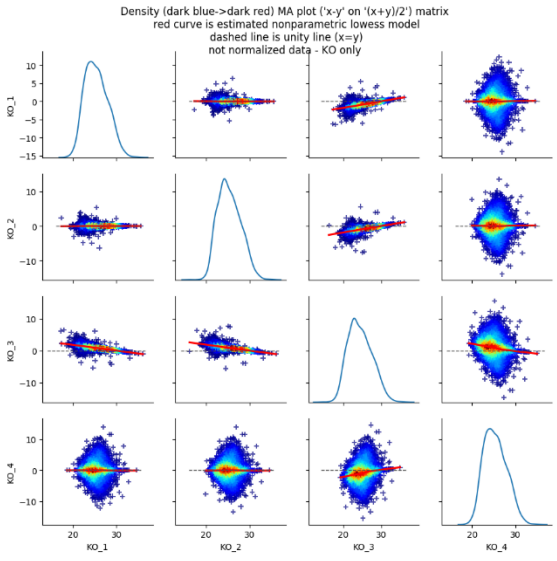
Let's now check how the plots look like e.g. for WT only scenario in standard scatterplot and MA plot format:





And similarly, for the KO sample replicates:

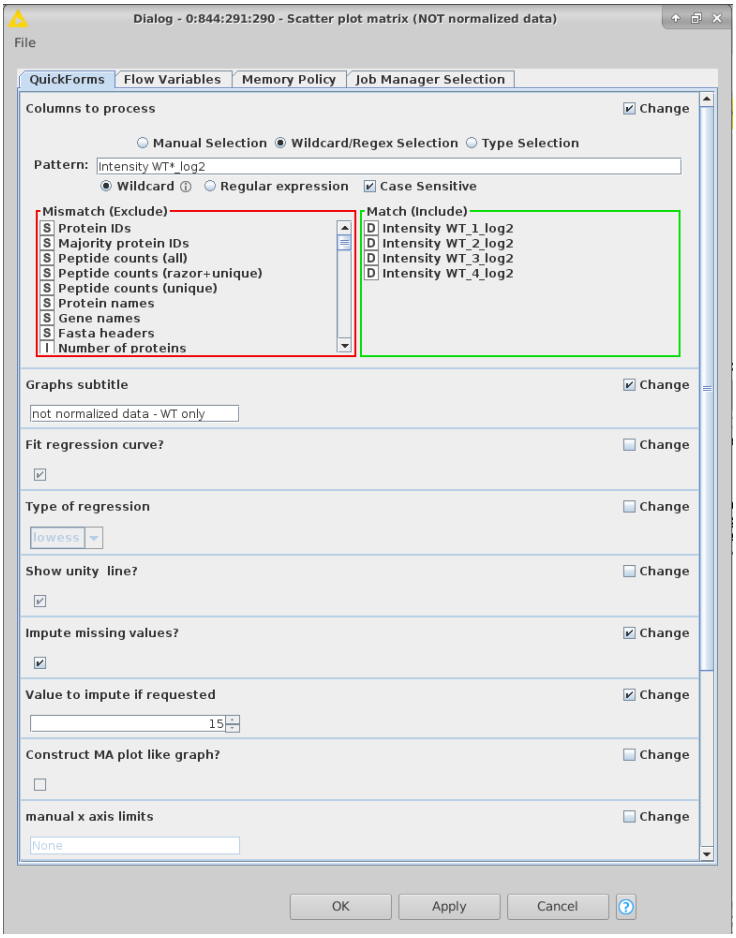




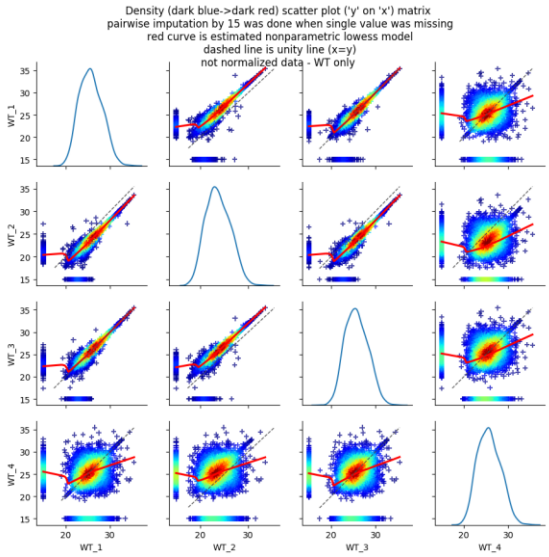
Colors provide the density information (i.e. blue - low density; red – high density).

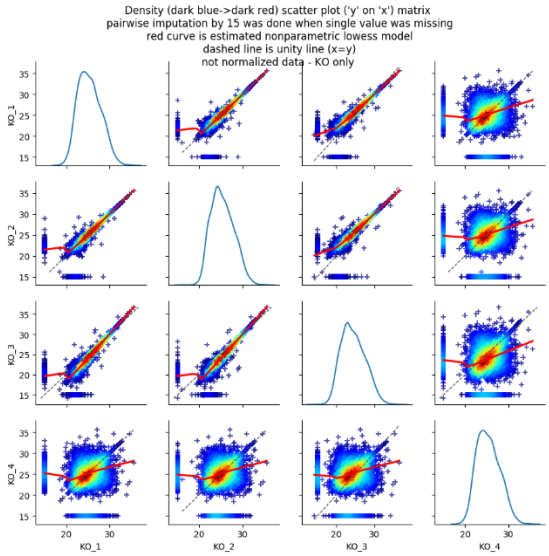
The last replicate seems to be off to all other replicates as evident from the cluster analysis. Here we can see it clearly on individual protein group intensities correlation. MA plots are often used to check on the normalization procedure efficiency as they use space more efficiently and one can notice linear/nonlinear effects better on MA plot compared to the classical y-x scatter plot.

It is also good practice to check on qualitative changes present in the dataset. We have seen some difference in the number of protein groups in individual sample replicates on the violin plots, so we can check where are potentially qualitatively changing protein groups located. Make a 3rd copy of the KO and WT only correlation scatter plots (not MA plots) and move them to the right. We can use "impute missing values" feature of the scatter plot matrix metanode. Use value of 15 to be used for imputation that seems to be adequate based on the scatter plots above (we don't want to overlap the imputed values with the existing scatterplot, so we usually choose the value as ~(lowest intensity – 2). The settings for WT sample replicates should look like this:



And this is how the graphs with imputed values look like:



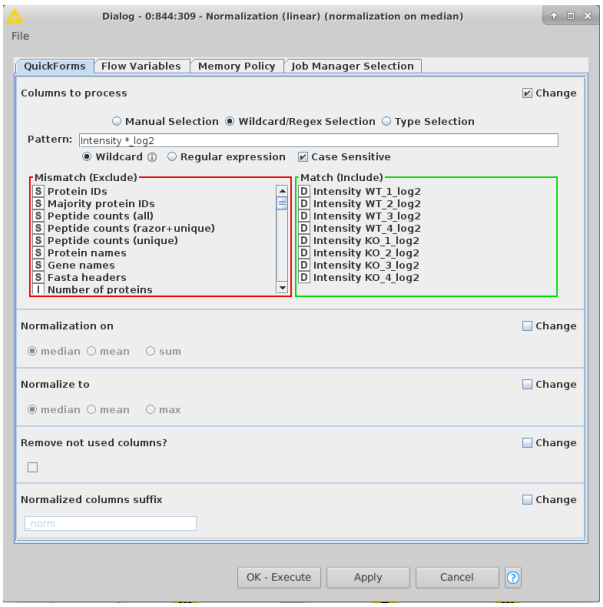


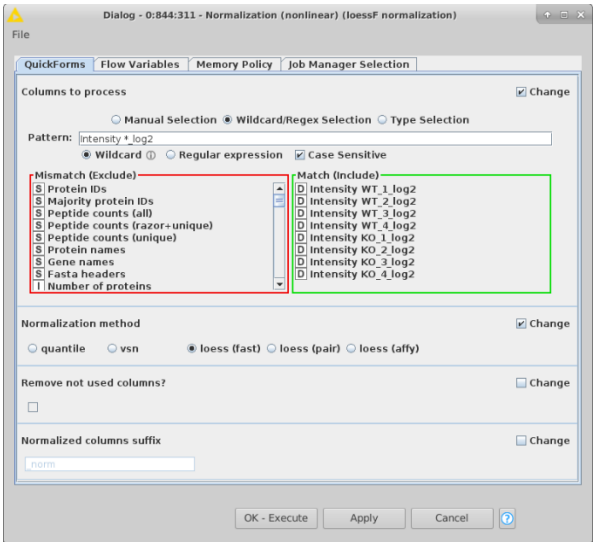
Again, more things we can notice about the data:

- correlations of all but 4th replicate are comparable among replicates
- the 2nd replicate of WT sample is generally lower intense, but there seems to be some linear effect present only like lower injection volume used or something similar
- the 3rd replicate of KO sample is off as well, but here seems to be some more complex effect present
- missing values among 1st to 3rd replicates of both sample types are on lower intensity scale, including the WT_2 and KO_3 samples, so these are really probably ok
- missing values in the 4th replicate are along the whole intensity range of other replicates

3.7 Normalization strategies test metanode – normalization strategies testing

Let's use now the acquired information to test the influence of different normalization strategies on the data. Navigate yourself to the normalization tests metanode and see the settings of linear and non-linear normalization metanodes:





Linear normalization, e.g. on median value, removes only linear technical processing artefacts from data in each column. Violin plots are just moved up or down to have the median value equal along all columns selected for normalization, no distribution change is taking place.

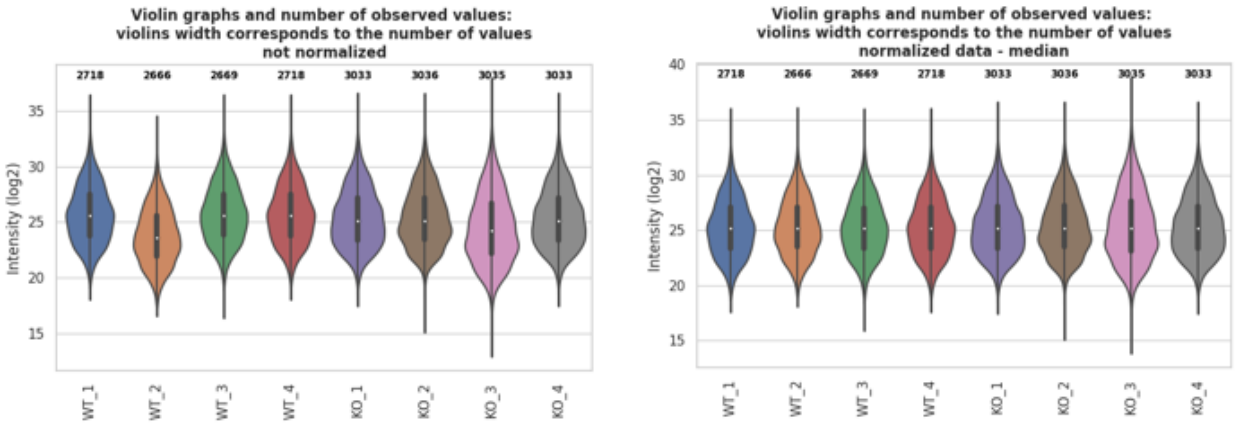
Nonlinear techniques like vsn, loess and quantile do more complex data processing to achieve data normalization. Description of each technique is beyond the scope of this document; we will test for the purposes of the workshop loess (fast) method for nonlinear data normalization and compare it with the normalization on the median value.

To check how the selected normalization technique influences the results, run the metanode normalizing the data on median value (the upper normalization metanode) and metanode doing loessF normalization step (3rd normalization metanode from top). Both should be already predefined (selecting the Intensity *_log2 columns). At this moment, normalize all 8 data columns at once.

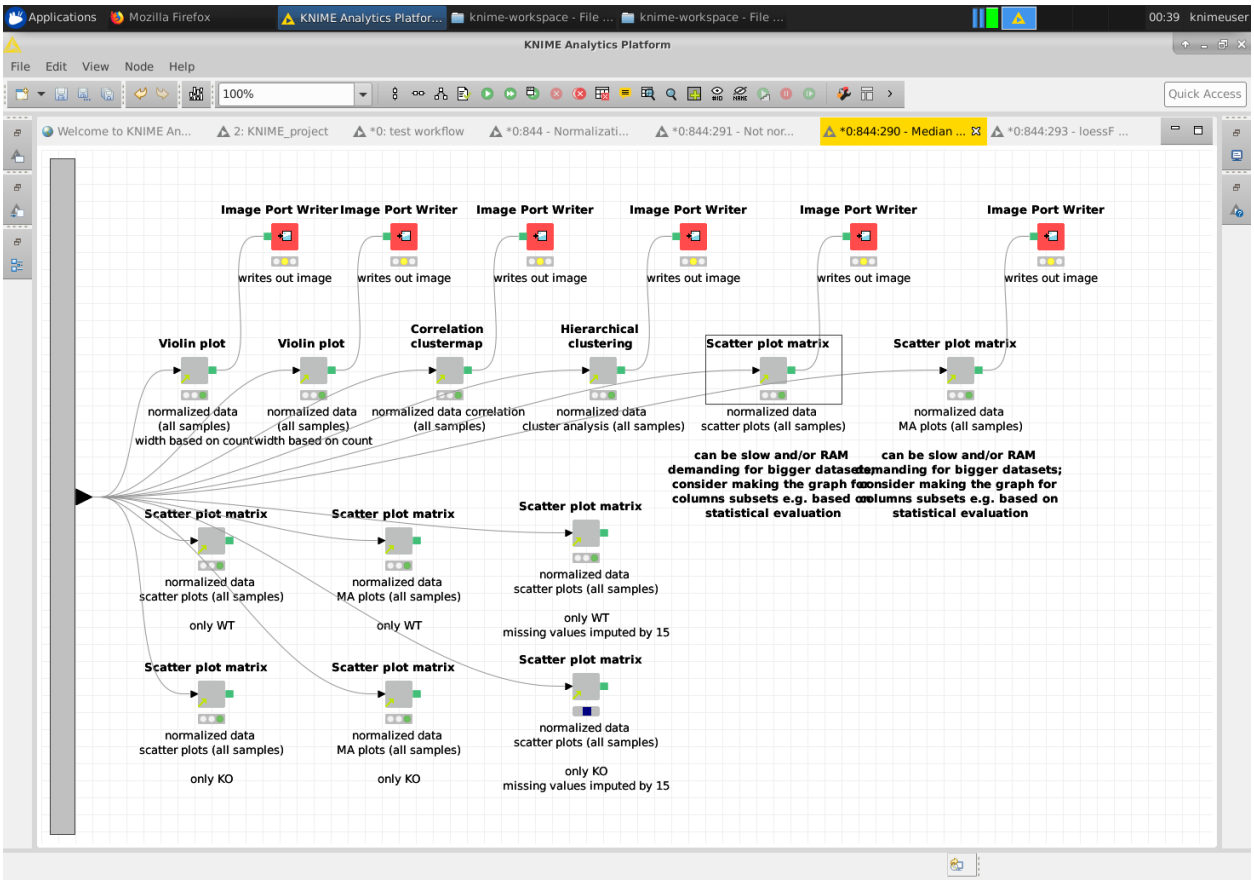
Inspection of the normalization step effect can be done using the same graphs we have used to check the situation prior the normalization. Now we are going to consider the data should be ready (after the normalization step) to be used directly in the statistical evaluation, unless we will find any major issue, so we should not tolerate any major issue observed.

Let's check at first how the situation has changed after the normalization on median value. Go into the metanode for description of median normalized values and initiate 1st four metanodes – two violin plots and two cluster analyses. When these 4 will be done, run also the remaining two graphs plotting.

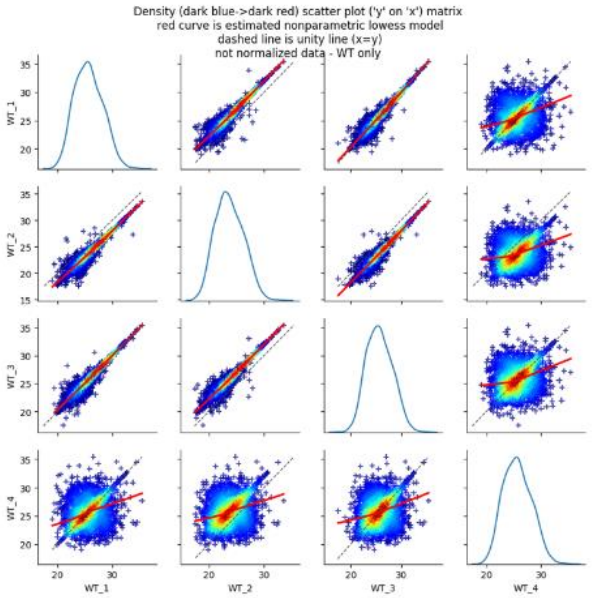
Here is the comparison of violin plots for not normalized and normalized data:

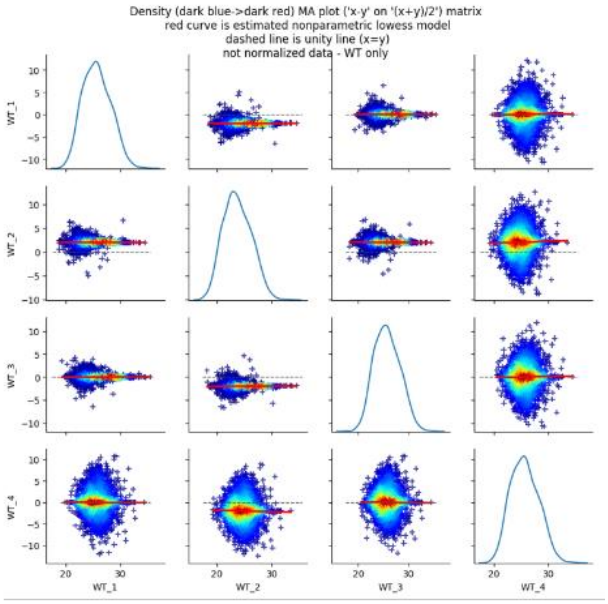


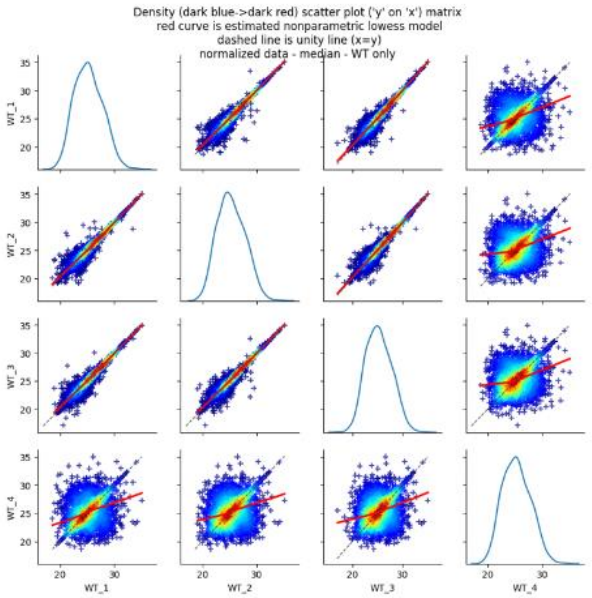
One can notice the KO_3 is still not in line with other replicates and if we would produce similar scatter plots to what we have done for not normalized values, we would see the reason clearly. To prepare the scatter plots for individual sample types you can either copy the scatter plot metanodes in the given inner metanode for median normalized values or we can also copy metanodes from not normalized data description metanode. We will have to make some adjustments in both cases, but we should end with the situation similar to this:

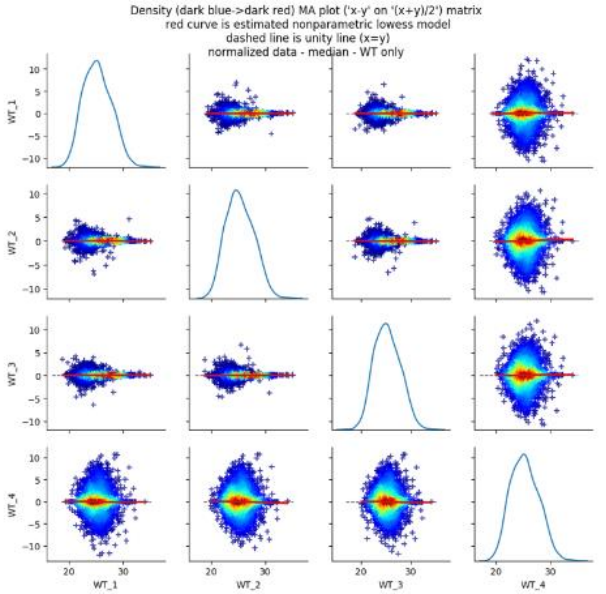


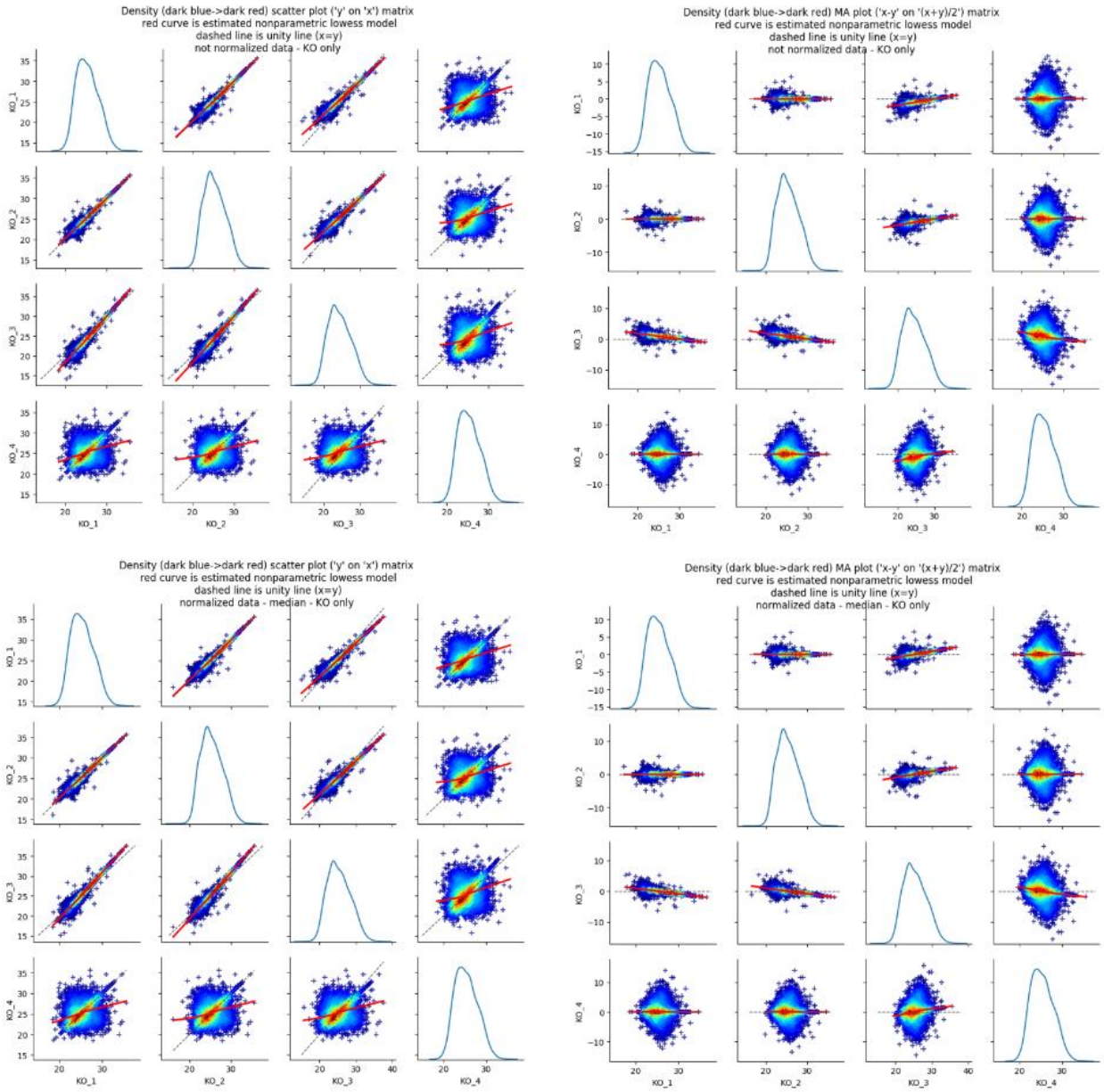
We can now compare the scatter plots before and after the normalization on median. Just open the graphs from the corresponding metanodes and compare either standard x-y scatters or MA plots for both WT and KO sample replicates:











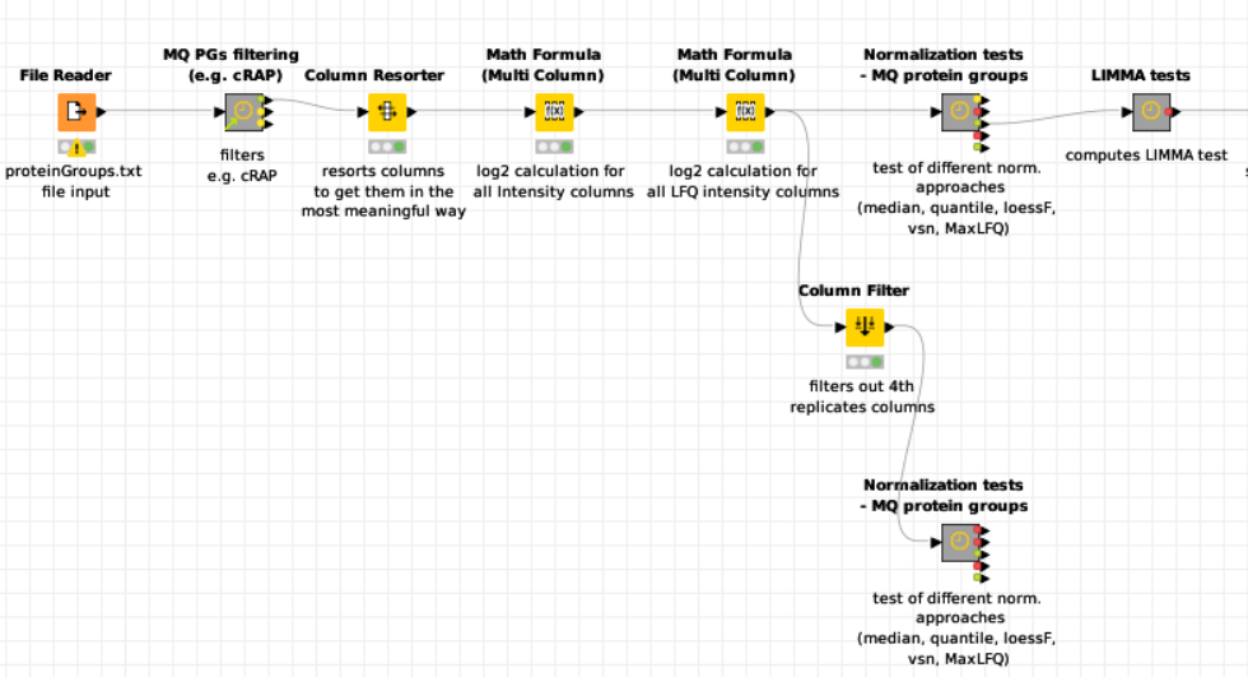
It is evident that median normalization did a good job in case of WT_2 sample, where the regression curve is now in line with the unity line. The KO_3 sample replicate is still off as there is some more complex effect observed, not only simple constant shift in protein group intensities along the whole intensity range.

3.8 Exclusion of the 4th replicate from the dataset

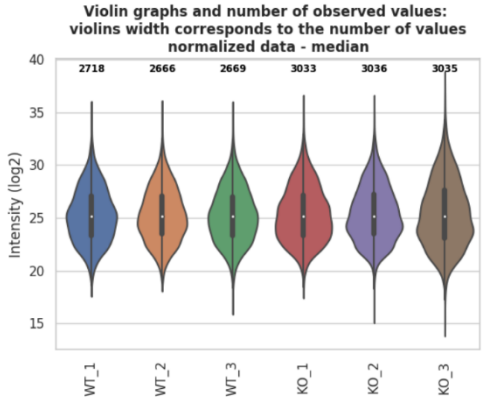
It was shown above that the 4th replicate is rather off and should be probably excluded from the data processing. In case we would like to process the data and get the final results, it would be better to re-

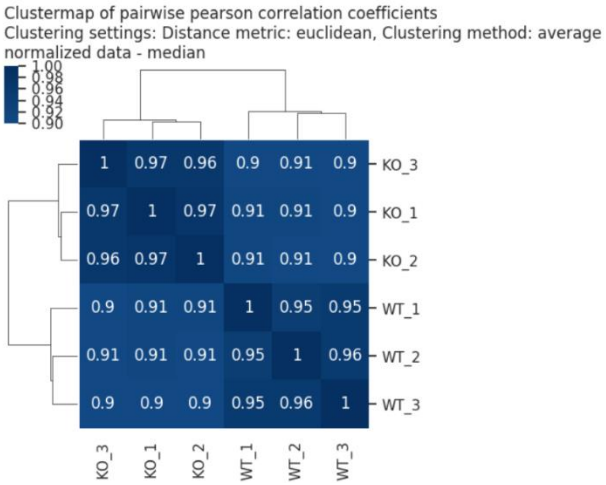
search the data in MaxQuant without 4th replicate of both samples, i.e. using only 6 samples. Here we will only remove columns of the 4th replicate and will continue further without them.

To remove them, search for the Column filter node in the nodes repository and try to set it up in a way it will pass all but the 4th replicates data column – notice, you can search for the columns by their names so you can use the fact they are both having "_4" tag in their names and search all columns having it and remove them from the table. Let's remove the data prior the normalization step and let's make a separate branch for the filtered data table scenario, i.e. copy normalization tests metanode and initiate the normalization once more, using 6 intensity columns this time:



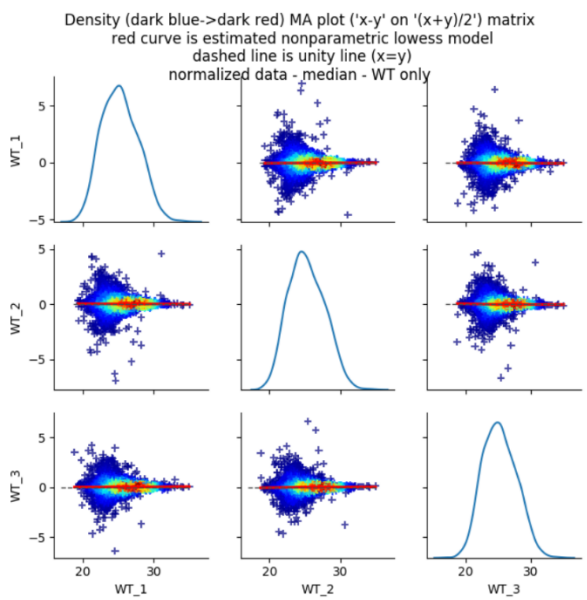
Let's firstly check how the median normalization worked for 6 samples only. Open the Median normalized data description and run the violin plot graph and correlation clustermap for all 6 samples:

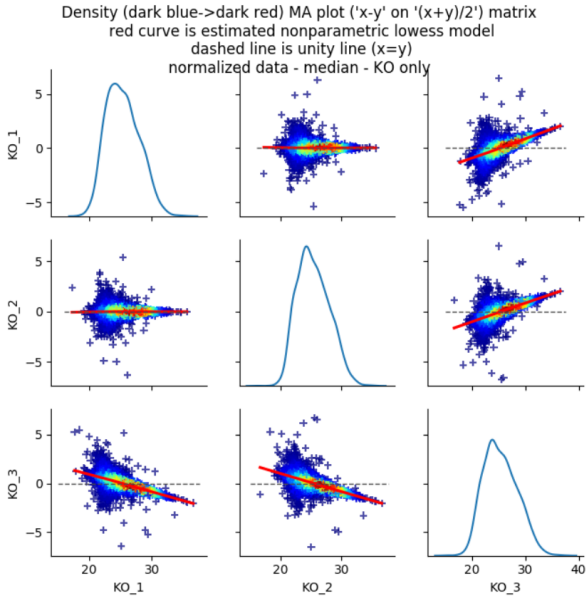




Both graphs look according to our expectations, in case of correlation clustermaps we are now getting two clusters according to the sample type.

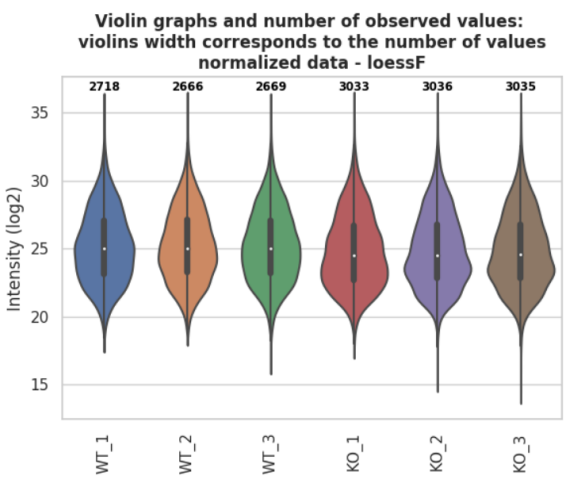
Let's check it on the scatterplots now. Run the scatterplots and/or MA plots for the particular sample types, i.e. separately for the WT and KO samples to make it easier to visualize.

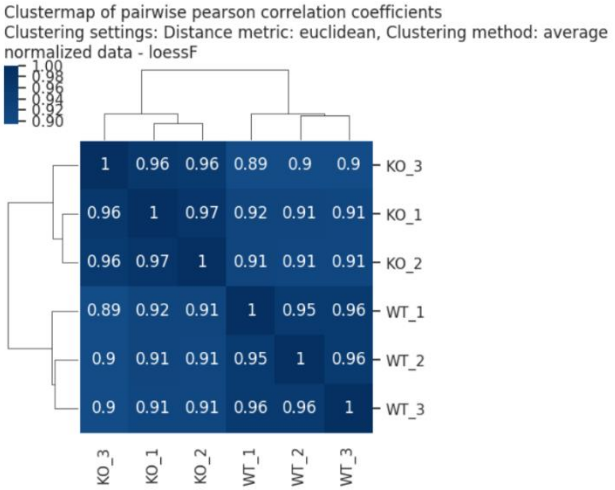




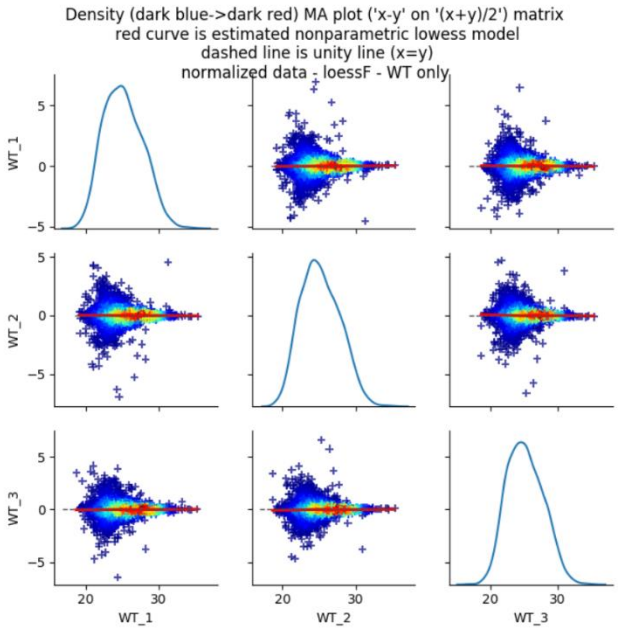
In case of WT samples, we can see that the median normalization worked nicely as before, because the unity line overlaps with the dashed line. However, in the case of KO samples we can see that the third replicate is still off, so some non-linear normalization method would be more suitable.

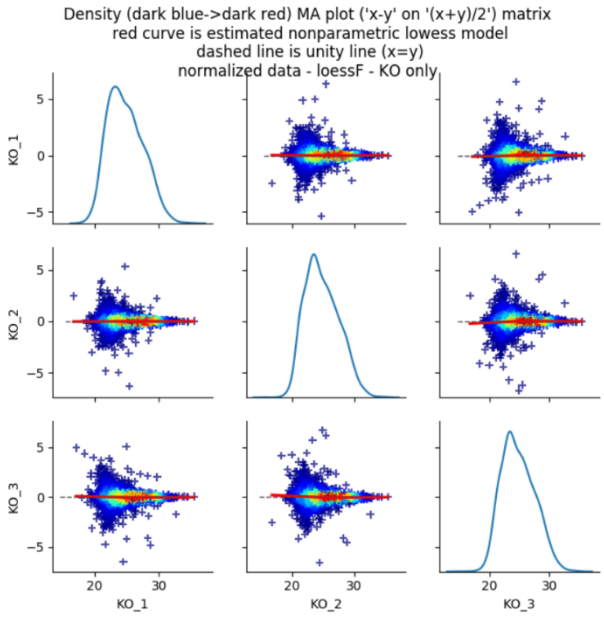
Therefore, let's try the LoessF normalization. Open then the LoessF normalized data description and let's look again at the violin plots and correlation clustermaps first:





Again, both the graphs look according to our expectations, in these two graphs it initially looks like the normalization worked well in this case. So, let's run again the scatterplots and/or MA plots separately for the WT and KO samples again after LoessF normalization:



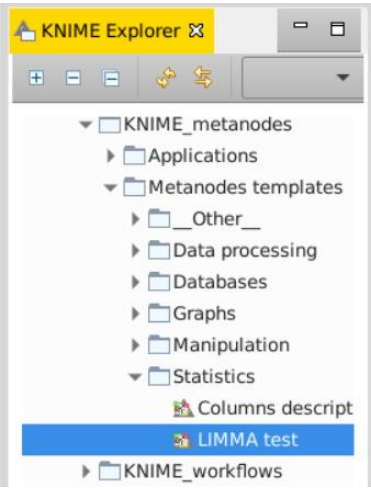


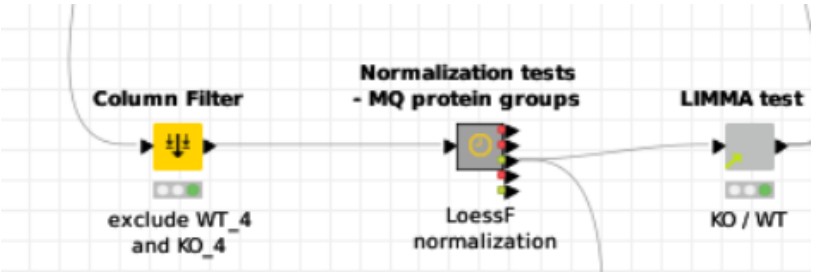
Here we can see that in the WT normalization again the dashed line overlaps with the unity line, and majority of the values lie on this line. In the case of KO samples, where after median normalization the 3rd replicate was still off, the LoessF normalization seems to work much better, the unity line and the dashed line are overlapping – not entirely, but the central regions look acceptable. This means, that for the further processing the LoessF normalization seems to be a good choice.

3.9 Differential expression using LIMMA test

In the next step, we will continue with the statistical testing. In our workflow we take advantage of the R package LIMMA (http://bioconductor.org/packages/release/bioc/html/limma.html), and we use the LIMMA test for the statistical evaluation.

Navigate yourself into KNIME Explorer -> gitfolders -> KNIME_metanodes -> Metanodes templates -> Statistics -> LIMMA test. By drag and drop move the metanode into the workflow and connect it to the appropriate port of the normalization metanode (ports are in the order of particular normalization techniques), i.e. in our case, connect it to the third port from the top, meaning the normalization by the LoessF technique.





Open the wrapped metanode form by double-clicking or F6. In the settings, we can adjust the test name (which will be projected into the columns name, in our case we can leave it as it is). Then we choose the columns which will be processed and compared. Usually we use the LIMMA test to compare some conditions to the control, in our case we will compare the KO condition to the WT, so we choose the log2 transformed and loessF normalized protein intensities: using the wildcards, specify "*norm", which will take all columns having in column names any length of characters followed by the "norm" pattern.

In the LIMMA design we specify the design of an experiment, where the order of the design corresponds to the order of the columns in the column selection. You can either write the design condition separated by comma into one row or you can separate the conditions by enter.

In our case the first three columns represent the WT condition and the second three columns represent the KO condition, therefore the design will be in a format “WT,WT,WT,KO,KO,KO”. The number of the samples in the conditions does not have to necessarily match (e.g. we could have 3 samples for the WT condition and 5 samples for the KO condition). You can use basically any strings (not numbers!) for specifying the design, this is only used for the R script to correctly assign particular columns to conditions (i.e. you could use also “A,A,A,B,B,B” as your design).

In case you have a paired design or batch effect, there is a possibility to specify sample pairs and sample blocks, refer please to the node documentation for further information.

Very importantly we need to specify the contrasts (comparisons). Because we are working on the log2 transformed data, we define the contrast not by division, but by substraction: in our case we compare the KO condition to WT, so the contrast will be defined as KO-WT. You can define as many contrasts as you wish at one time, separated by comma.

The option “treat comparisons separately” is by default checked, so all the specified comparisons will be tested and reported separately. If you want to test them all together as one 'factor', uncheck this option and you will get single table with statistics results like in case of one-way (one-factor) ANOVA.

Because we are doing multiple testing, we should do the p-values correction for multiple testing. We use by default the Benjamini-Hochberg method for controlling the false discovery rate, but you can choose also “none”, “Benjamini-Yekutieli” or “Bonferroni”.

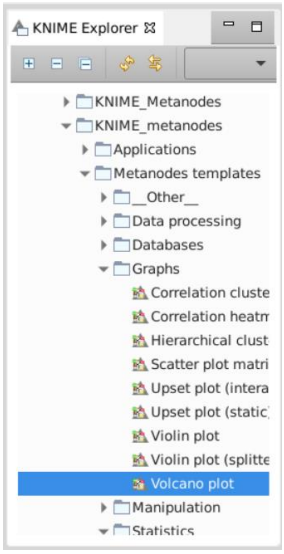
Lastly, you have an option to remove the not used columns, so you would end up with the results of LIMMA test only, the rest of the table would be deleted.

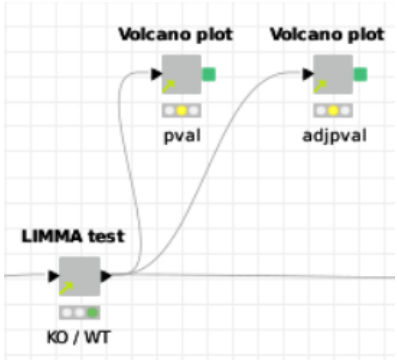
Now having the node configured, we can execute it (F7). Let's look on the results: right-click the node, choose Output dataframe, and in the most right of the table you will notice newly processed columns starting with the test name you specified in the form. There is in total 6 newly added columns per one comparison, most importantly:

- **logFC:** mean values of input (log transformed) data subtracted based on the concrete comparison - e.g. average of three KO column values minus average of three WT column values for each row in case of KO-WT comparison; i.e. log fold change of not log transformed data using log of the same base
- **P.value:** not adjusted raw p-value of moderated t-test statistics; this is not preferred p-value due to multiple hypothesis testing (each row represents one test) that needs to be addressed not to give to high false positive discovery rates
- **adj.P.val:** adjusted p-value using the selected method

3.10 Visualization of the statistical results using volcano plots

We can plot the results of the LIMMA test by **volcano plot** graph. Navigate yourself into KNIME Explorer -> gitfolders -> KNIME_metanodes -> Metanodes templates -> Graphs -> Volcano plot. Drag-and-drop the node into the workflow and connect it to the outport of the LIMMA metanode. We will plot both raw p-values and adjusted p-values to see the comparison, so copy the volcano plot one more time and adjust the description of the node.



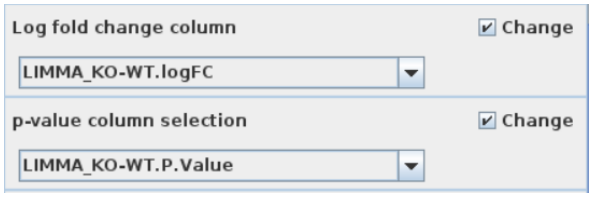


Now let's configure the volcano plot node: double-click it or press F6 to get to the form. Volcano plot plots by default the log fold change on the x-axis and -log10(p-value) on the y axis. Based on the selected thresholds for logFC (default: 1) and p-value (default: 0.05) it sorts the values into four groups:

- non-significant (blue) – neither passing logFC threshold, nor p-value threshold
- significant (yellow) – passing p-value threshold, but not logFC threshold
- upregulated (red) – passing p-value threshold and logFC threshold is >1

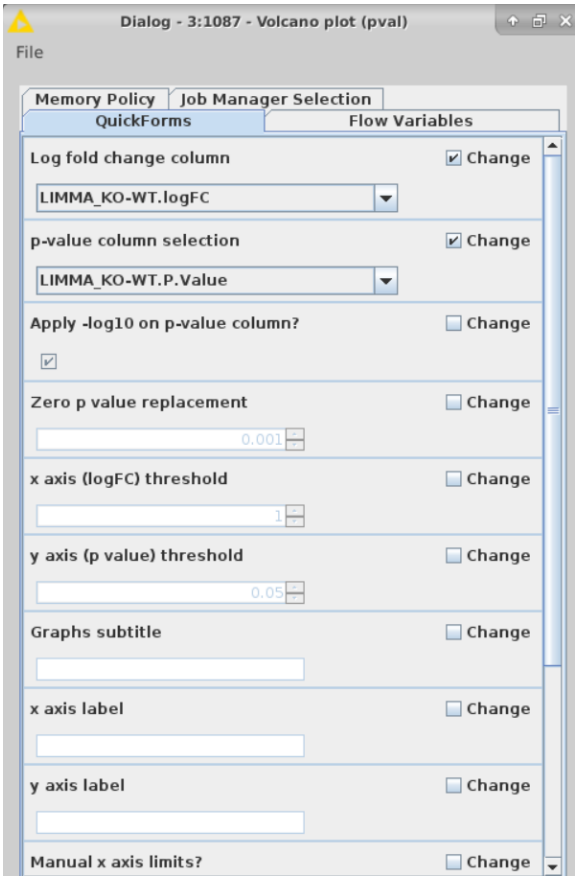
- downregulated (green) – passing p-value threshold and logFC threshold is < -1

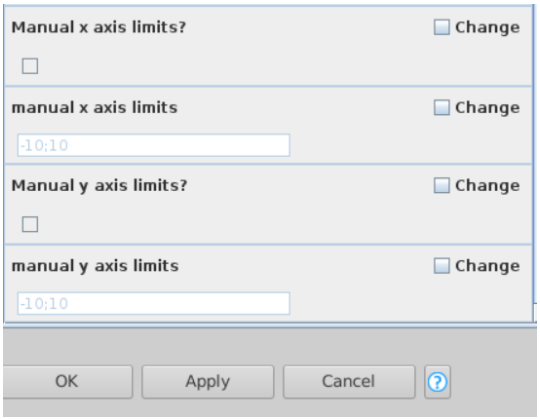
In our form, we firstly specify the columns for the logFC and p-value:



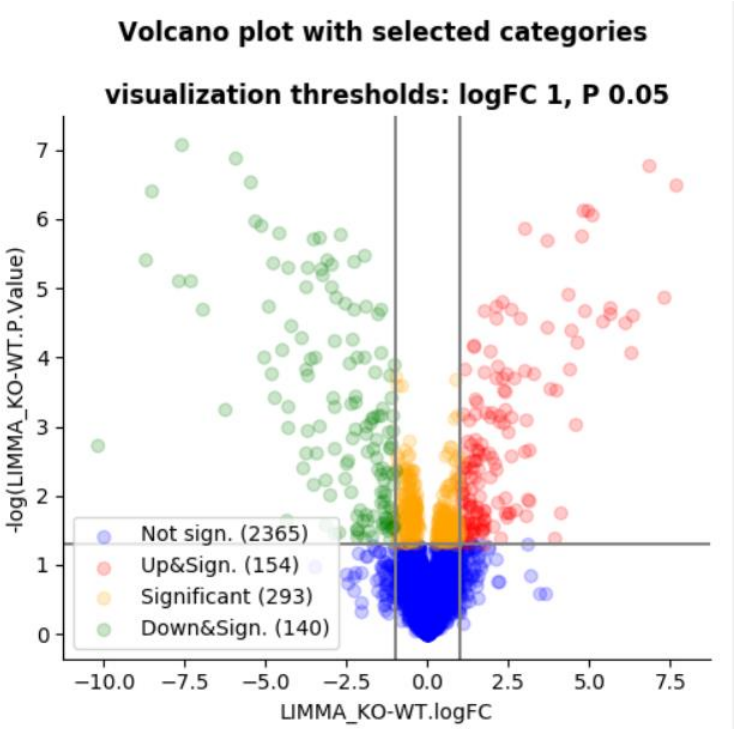
Next, we tick whether to apply -log10 on the y axis, which we by default have ticked. Thanks to this option the graph will have a typical shape of volcano plot instead of ordinary scatter plot.

In the next option we can replace the zero values by some other value (by default 0.001) to get these into the graph. As already mentioned, the thresholds for x-axis (i.e. logFC) we set to 1 and for y-axis (i.e. p-value) to 0.05. Changing the thresholds will result in either less stringency (e.g. having lower logFC and/or higher p-value threshold) or higher stringency (higher logFC, lower p-value). We can further provide some Graph subtitle and modify the axis labels or ranges. The form will in the end look like this:



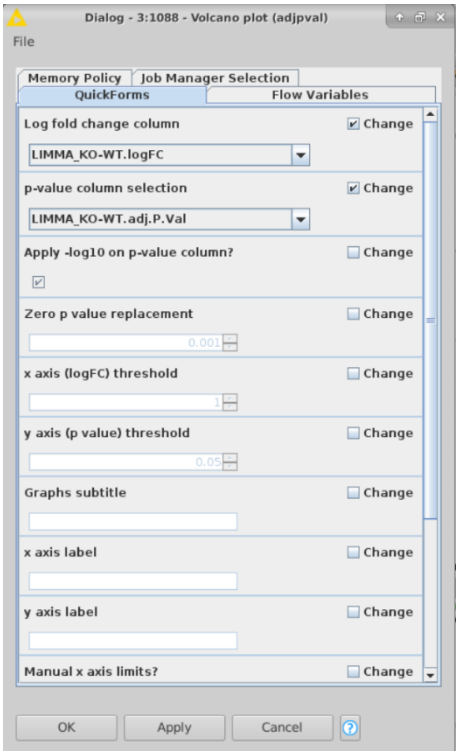


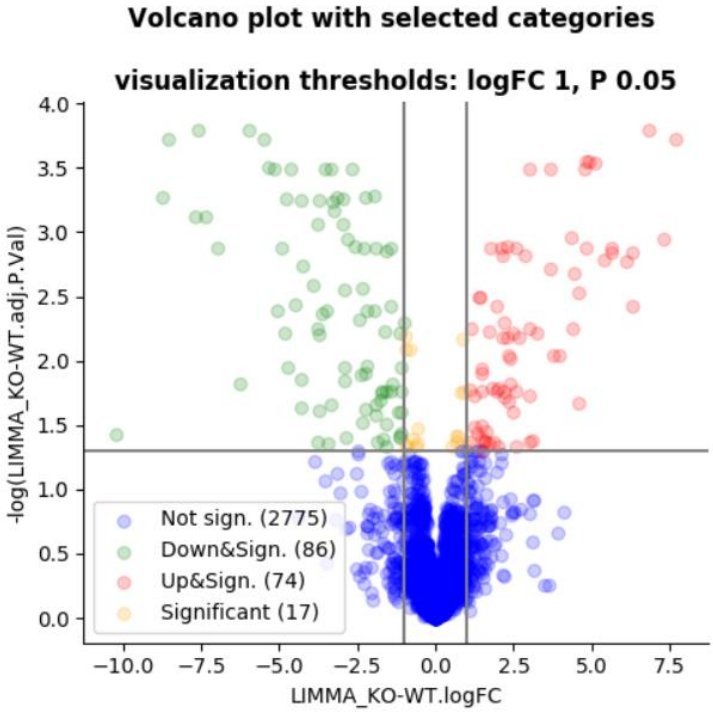
We can execute the metanode (F7) and look at the output:



We can see, that we got 2365 proteins which are non-significant, 154 proteins which are significantly upregulated, 140 which are significantly downregulated and 293 which are significant.

Let's now run the second volcano plot, so apply the same settings, but use the adjusted p-value instead of raw p-value:

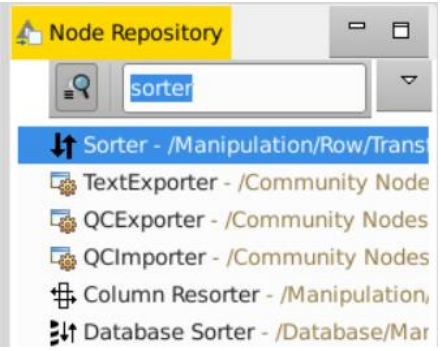


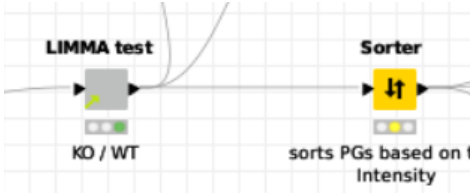


We can see that indeed after having more stringent p-values after correction for multiple testing resulted into lower number of significant proteins. In our workflows we consider the adjusted p-value to be more trustable and therefore we use mainly this one for any further purposes.

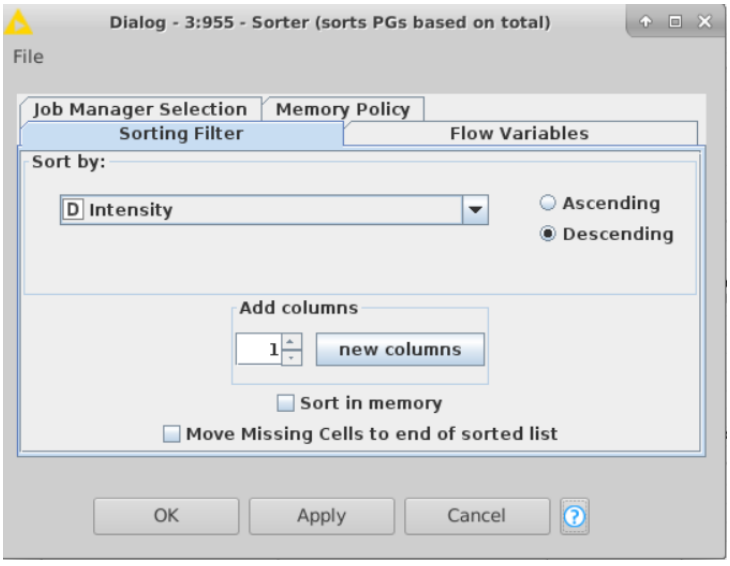
3.11 Resorting the table if needed

In the next node, **Sorter**, we can sort the rows of the table e.g. according to the descending value in the Intensity column. To get the Sorter node, write simply "Sorter" into Node Repository (or Node Repository -> Manipulation -> Row -> Transform -> Sorter) and then drag-and-drop the node to the workflow, connecting it to the outport of LIMMA metanode:



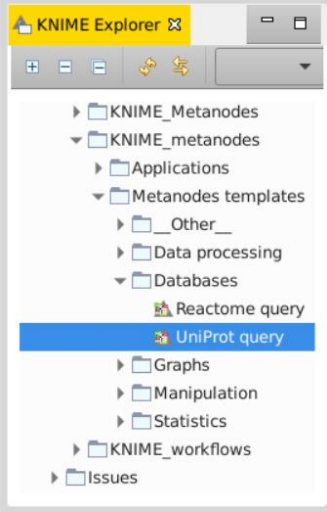


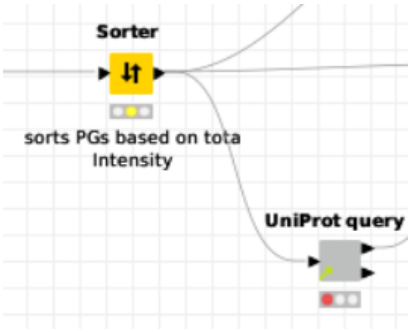
On double-click or F6 open the form. Here we can set which column to sort according to and also whether the sorting should be ascending or descending. We'll set sort by "Intensity" and "Descending". Execute the node by F7.



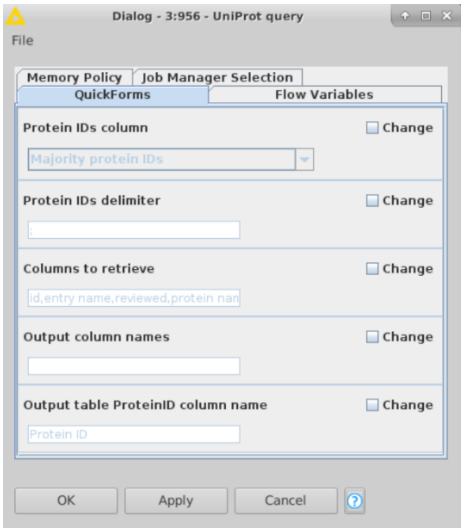
Further we will use the **Uniprot Query** metanode, which will retrieve information from the UniProt database. Navigate yourself into KNIME Explorer -> gitfolders -> KNIME_metanodes -> Metanodes

templates -> Databases -> UniProt query; drag-and-drop the metanode into the workflow and connect it to the output of Sorter node.





Double-click the metanode or press F6 to get to the metanode form. In here, we can specify the Protein IDs column, so the column which will be used to retrieve data from UniProt database. In our case we use the Majority protein IDs. Next, you select the delimiter by which are the data from the column above delimited, in our case it's ";". Next, you can specify which columns from the UniProt will be actually retrieved (in here http://www.uniprot.org/help/uniprotkb_column_names you can find full list of columns which might be retrieved, change the form according to your wishes in this aspect). You may then change the Output column names. If you leave it empty, column names from uniprot will be used, other way you can specify the names separated by comma (e.g. genenames,interactors,entry name). Finally, we can set the Output table ProteinID column name, so the column name to set for proteinID column after processing input column with protein IDs. The final form will look like this:



Execute the node (F7); if many protein columns are to be retrieved, the execution may take some time. When we look now at the node output, we can notice that there are two outports: the top is related to the IDs which were successfully found in the UniProt, in the bottom outport you will then see a list of proteins, which empty query result was retrieved (e.g. due to missing accession).

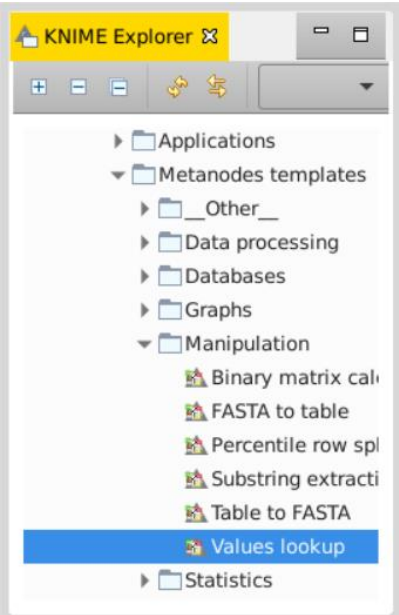
Row ID	Protein ID	Entry	Entry name	Status	Protein names
Row0	Q9Y4P1	Q9Y4P1	ATG4B_HUMAN	reviewed	Cysteine protease ATG4B (EC 3.4.22.-) (AUT-like 1 cysteine endopeptidase) (Autophagin-1) (Autophagy-related cysteine endopeptidase 1) (Autophagy-related prote
Row1	A0A1B0G...	A0A1B0G...	A0A1B0GU03...	unreviewed	Peptidase A1 domain-containing protein
Row2	Q9BVW5	Q9BVW5	TIPIN_HUMAN	reviewed	TIMELESS-interacting protein
Row3	Q96EH3	Q96EH3	MASU1_HUMAN	reviewed	Mitochondrial assembly of ribosomal large subunit protein 1
Row4	Q9BWJ5	Q9BWJ5	SF3B5_HUMAN	reviewed	Splicing factor 3B subunit 5 (SF3b5) (Pre-mRNA-splicing factor SF3b 10 kDa subunit)
Row5	Q13619	Q13619	CUL4A_HUMAN	reviewed	Cullin-4A (CUL-4A)

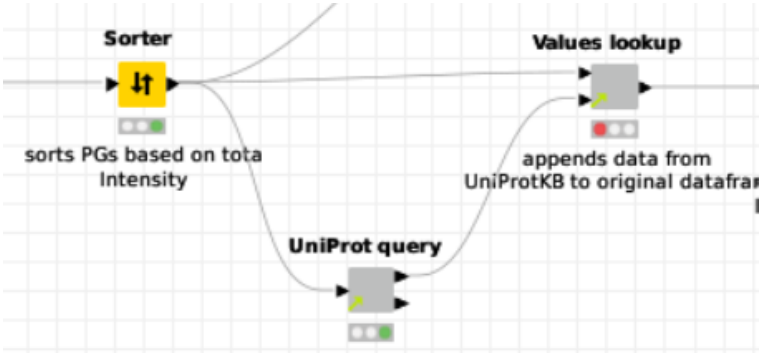
Subcellular location [CC]	Protein families	Interacts with	Sequence
SUBCELLULAR LOCATION: Cytoplasm {ECO:0000250}.	Peptidase C54 family	O95166; Q9H0R8; P60520; Q9H492; Q9GZQ8; Q9BXW4	MDAATLTYDTLRFAEFEDFPETS
	Peptidase A1 family		MQPSSLLPLALCLLAAPASALVRI
SUBCELLULAR LOCATION: Cytoplasm {ECO:0000269|...	CSM3 family	P15927; Q8IYF3	MLEPQENGVIDLPDYEHVEDETI
SUBCELLULAR LOCATION: Mitochondrion matrix {EC...	Iojap/RsfS family	P32321	MGPGGRVARLLAPLMWRRAVSS
SUBCELLULAR LOCATION: Nucleus {ECO:0000269|P...	SF3B5 family		MTDRYTIHSQLEHLQSKYIGTGHA
	Cullin family	Q86VP6; Q16531; Q92466; Q15291; O94888; P55072	MADEAPRKGSFSALVGRTNGLT
SUBCELLULAR LOCATION: Cytoplasm {ECO:0000269|...		Q9WMX2; P26196; Q9H4E7; Q86UW9; Q5JVL4; P0673...	MDRRSMGETESGDAFLDLKKPP
SUBCELLULAR LOCATION: Cell membrane {ECO:000...	Selectin/LECAM family		MIFPWKCQSTQRDLWNIFKLWG
SUBCELLULAR LOCATION: Cytoplasm, cytoskeleton {...	WD repeat EMAP fa...		MDGFAGSLDDSISAASTSDVQD

These are only two examples from the resulting table, but of course, many more columns were retrieved.

Next, we will use the **Values lookup** metanode to append the information from UniProt query to the original table.

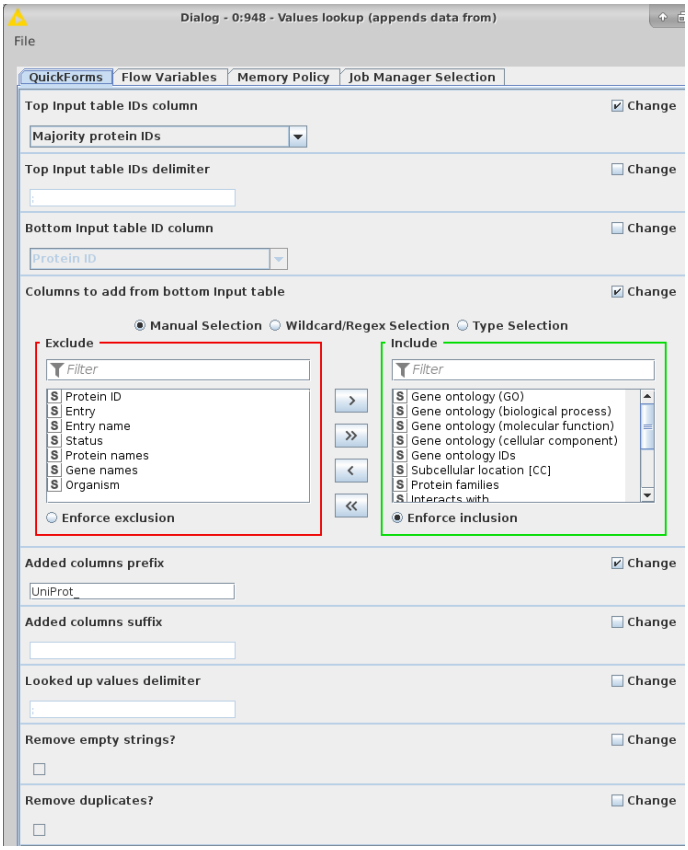
Navigate yourself into KNIME Explorer -> gitfolders -> KNIME_metanodes -> Metanodes templates -> Manipulation -> Values lookup. Drag-and-drop to the workflow, and connect the Sorter output with the top port of the Values lookup metanode, and the top outport of the UniProt query with the bottom inport of the Values lookup metanode:





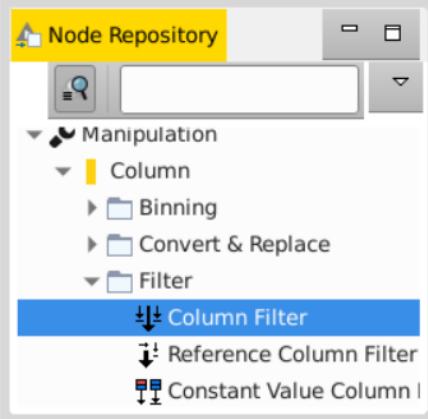
Double-click on the Values lookup metanode or F6 to get the settings. In here, we can specify which column to use for the IDs in the top table. So, basically, to which IDs we will be mapping in the bottom table. Choose the Majority protein IDs column and in the next form, selecting the delimiter, choose ";". This means, that to every accession of the Majority protein IDs column will be mapped the corresponding row from the bottom table in case the accession will be the same. Next, specify the Bottom Input Table IDs column, so the accessions for the bottom table, which will be compared to the accessions from the top table (Majority protein IDs in our case). Then, we can specify which columns from the bottom table we want to map to the top table (so the original table). In here, we can choose the ones not already being in the table, so all Gene ontology related columns, Subcellular location, Protein Families, Interacts with, Sequence and all Cross-references. We can also specify whether the appended columns will have any prefix or suffix (none in our case). Further we specify, how will be the looked up values delimited. It might happen, that there will be e.g. two accession numbers, and for each of them, an information will be retrieved, in that case, the information will be separated by selected delimiter (";"). We can again specify whether to remove empty strings, duplicates and not used columns (in our case, do not tick any of the options).

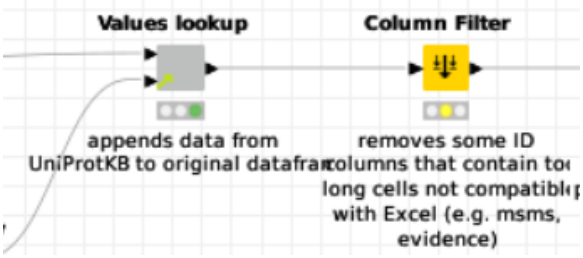
This is the resulting form:



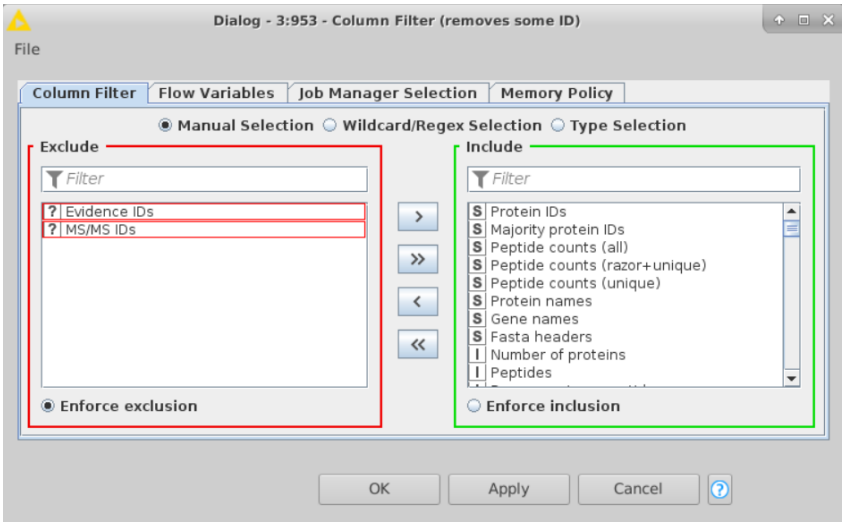
Execute the metanode (F7), again, this might be slower if appended table is large. You can notice, that the columns from the UniProt query were now appended to the original table.

The next node we will use is the **Column Filter**. Navigate yourself into Node Repository -> Manipulation -> Column -> Filter -> Column Filter. Drag-and-drop the node to the workflow and connect to the outport of the Value lookup metanode.

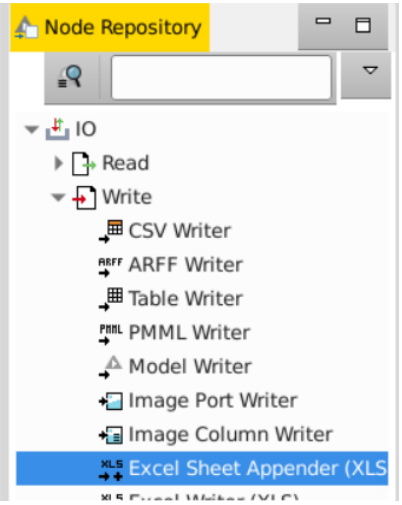


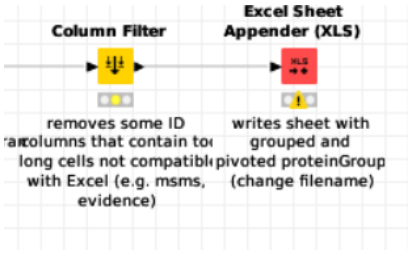


The column filter is present in the workflow due to the Excel Sheet Appender in the next step and Excel limitation of maximum 32767 characters per one cell, which we are unfortunately able to exceed in some cases. In those cases, we sometimes are required to filter out the Evidence IDs and MS/MS IDs which cause the trouble. It might be useful to run the node with all columns included and in case of the following Excel Sheet Appended error to go step back, exclude the columns causing troubles and re-run the Excel Sheet Appender. For the purposes of the workshop, we prepared a dataset not containing the Evidence IDs and MS/MS IDs, so you can just execute the node (F7).

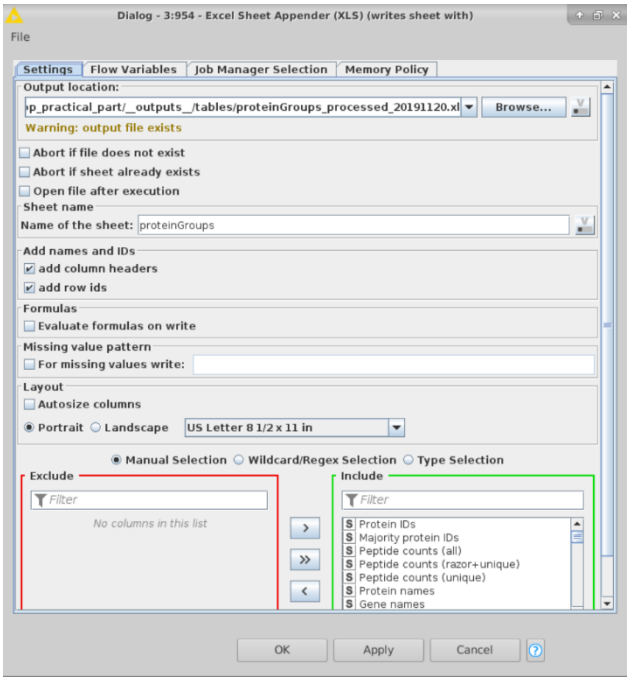


Finally, we write the resulting the table by the **Excel Sheet Appender**. Navigate yourself into Node Repository -> IO -> Write -> Excel Sheet Appender and connect it to the Column filter node.





Double-click the node or press F6 to get the settings form. Here we can adjust the output location, and also to specify whether to abort the node if the file does not exist or if the sheet already exists. We can also set whether to open the file after execution. In our case we will leave all of the boxes unticked. We can then specify the sheet name (e.g. proteinGroups). We tick that we will add both column and row IDs. We can also specify whether to write anything for the missing values. Finally, we can select which columns we will include, in our case all of them. Execute the node (F7). You can find the resulting .xlsx file in the output location you specified. As mentioned before, in case the node is failing it might be due to the number of characters limit exceeded, in that case return to the column filter and filter out the columns causing the troubles.

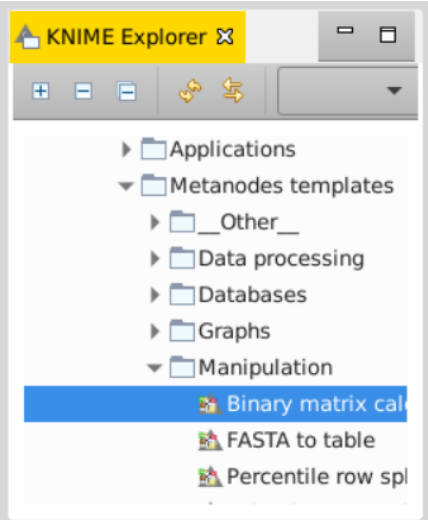


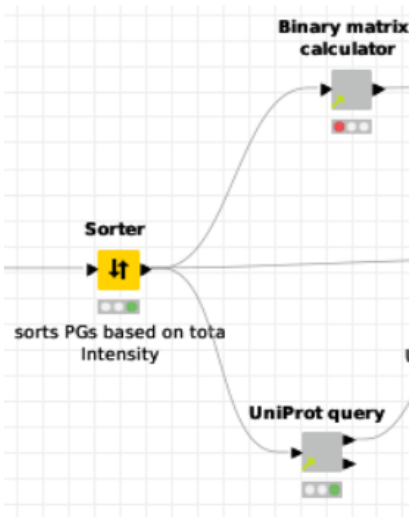
3.12 Filtering and Reactome metanode

Next, we will concentrate on a bit specific use cases, not included in the standard MS data analysis pipeline.

Firstly, often you encounter some type of filtering, either in the quantitative (on the Intensities level) or qualitative (on the Peptides level) way.

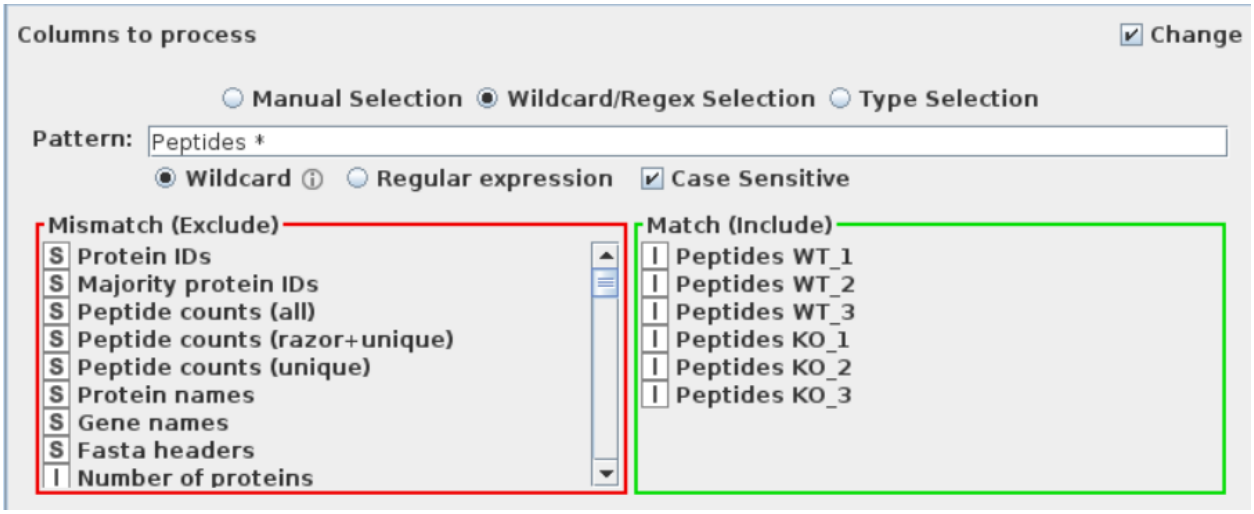
Let's show how we will filter the proteins, which were identified on at least 2 peptides in at least 2 replicates out of 3. For this purpose, we will use the **Binary matrix calculator** node. Navigate yourself into KNIME Explorer -> gitfolders -> KNIME_metanodes -> Metanodes templates -> Binary matrix calculator. Drag and drop the metanode into the workflow and connect it in our case to the Sorter we used a while ago.



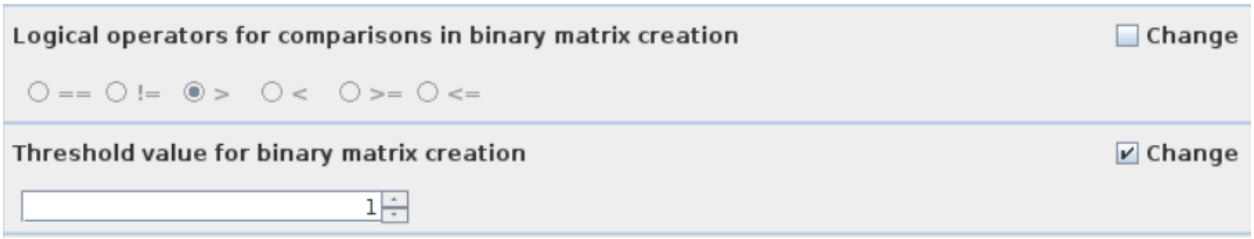


This node can calculate the binary matrix (so table containing only 0 and 1) according to some criteria. The proteins passing the criteria will be assigned 1, whereas the proteins not passing the criteria will be assigned 0.

Open the node settings (F6). We can choose all columns containing peptides, so choose Wildcard selection in the Columns to process and the pattern we will be searching form is “Peptides *”, so the columns starting with Peptides – space – any number of characters.

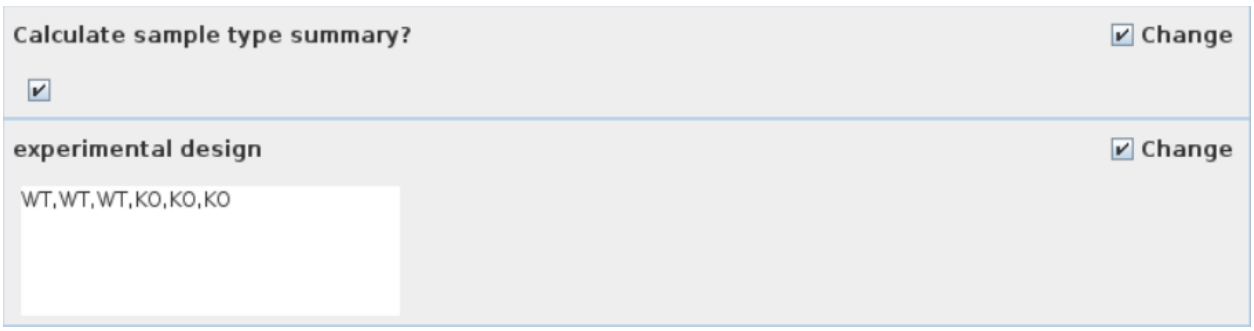


Next, we will specify the logical operator and threshold for creating of the binary matrix. In our case we want proteins identified on at least 2 peptides, so either choose >=2 or >1.

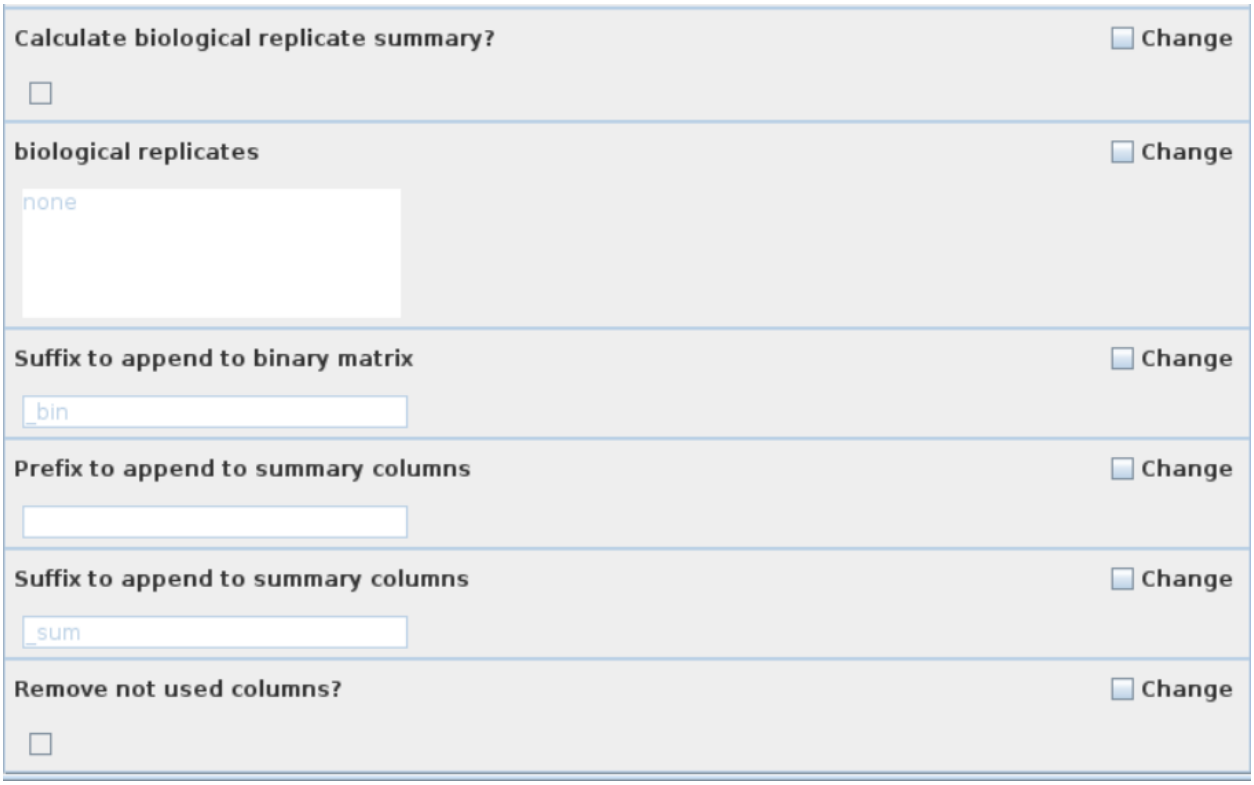


Next, we can specify whether to compute the sample type or biological replicate summary. In our case, we will use the sample type summary, we want to know in how many replicates the protein was identified on >=2 peptides.

So, tick the “Calculate sample type summary?” option and provide the design: “WT,WT,WT,KO,KO,KO” – the design similarly to LIMMA test corresponds to the order of “Columns to process” selection.



In similar way you could specify also the biological replicates if applicable. We will then specify, that we want the processed columns containing binary matrix will have suffix _bin and that columns containing sum have suffix _sum.



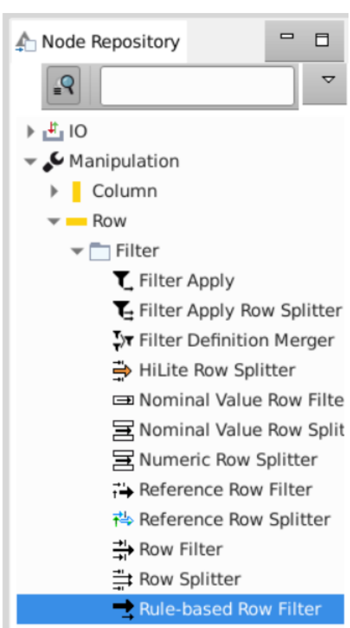
With these settings we will run the metanode (F7). You can notice then, that in the right end of the table new columns containing (a) the binary matrix and (b) summary per sample type were appended:

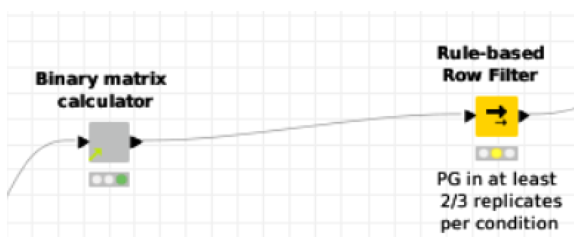
D Peptides WT_1_bin	D Peptides WT_2_bin	D Peptides WT_3_bin	D Peptides KO_1_bin	D Peptides KO_2_bin	D Peptides KO_3_bin	D KO_sum	D WT_sum
1	1	1	1	1	1	3	3
1	1	1	1	1	1	3	3
1	1	1	1	1	1	3	3
1	1	1	1	1	1	3	3
1	1	1	1	1	1	3	3
1	1	1	1	1	1	3	3
1	1	1	1	1	1	3	3
1	1	1	1	1	1	3	3
1	1	1	1	1	1	3	3
1	1	1	1	1	1	3	3
1	1	1	1	1	1	3	3
1	1	1	1	1	1	3	3
1	1	1	1	1	1	3	3
1	1	1	1	1	1	3	3
1	1	1	1	1	1	3	3
1	1	1	1	1	1	3	3
1	1	1	1	1	1	3	3

Scroll down to see that actually there are also proteins containing zeroes.

In the next step we will use the **Rule-based Row Filter** to include further only proteins which were identified on >=2 peptides in at least 2/3 replicates per condition.

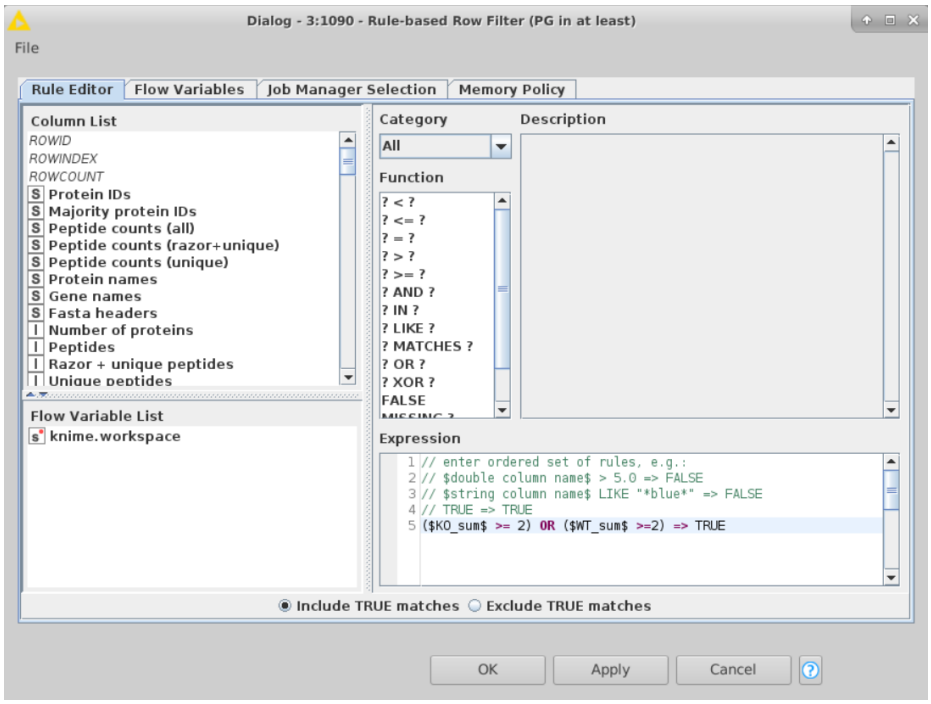
Navigate yourself into Node Repository -> Manipulation -> Row -> Filter -> Rule-based Row Filter, drag the node to the workflow and connect it to the binary matrix calculator.





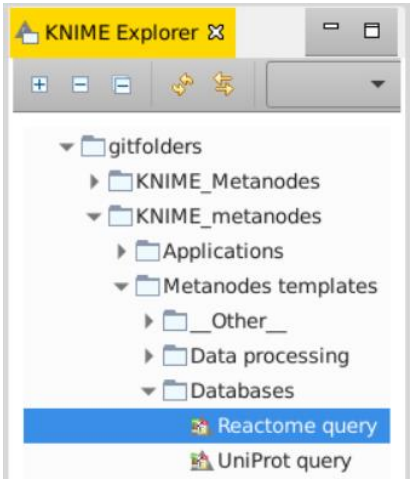
Open the Rule-based row filter. In this node we have a Column list, from which we can choose the columns to process by clicking on particular columns. Then there are various mathematical operators which we can use and finally the space for the expression itself. In our case, we will filter the proteins where sum of the replicates is in at least one condition >=2: (KO_sum >= 2) OR (WT_sum >=2) => TRUE

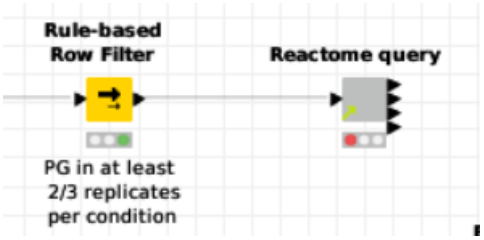
So the syntax is as follows: the conditions using logical operators and then => TRUE, meaning, that those rows passing the criteria will be considered to be TRUE. In the bottom part of the form you can then specify whether to include or exclude the True matches (in our case Include):



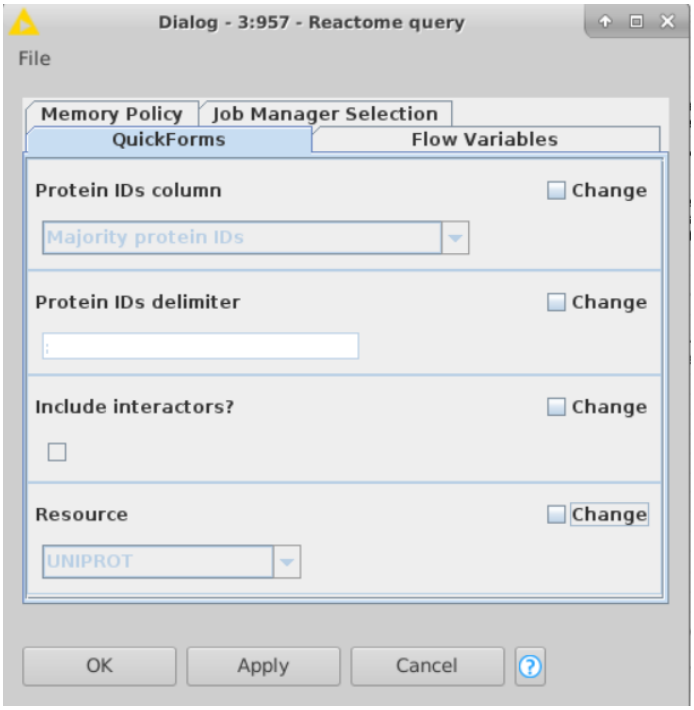
We can execute the node (F7).

Finally, we can run **Reactome,** a pathway analysis tool, on the filtered table. Navigate yourself into KNIME Explorer -> gitfolders -> KNIME_metanodes -> Metanodes templates -> Databases -> Reactome query, and insert the node to the workflow, connect to the preceding row-filter.





Open the form (F6). We will firstly set the Protein IDs column, so which IDs will be searched for in the Reactome, in our case Majority protein IDs, and similarly to the Uniprot Query, we will set the IDs delimiter (";"). Next, we can specify whether we want to include also the interactors, for the simplicity we will leave this now unticked. Lastly, we can specify which resource to use, in our case Uniprot, but you can choose from a list of possibilities including Ensembl, Mirbase, NCBI_PROTEIN, CHEBI, etc.



Execute the metanode (F7). On the output we will get 4 tables:

1) pathways table

table of pathways returned by the analysis (similarly to downloading the pathway analysis results in csv format via web interface)

2) not mapped identifiers

identifiers that were not found/mapped during the Reactome analysis

3) summary table

summary information about the analysis including date of running, database version and more

4) complete analysis results

complete analysis results in json format for further data mining if needed.

In real-life scenario you would be more interested in e.g. up- or down-regulated proteins in which case you would use just different (e.g. logFC column) criteria in the **Rule-based Row Filter/Splitter** node.

3.13 Missing values imputation

In the last part we will concentrate on the imputation of missing values.

One of the challenges of proteomics is large portion of missing values in MS experiments, in case of LC-MS/MS approaches it can be between 10-50% (Lazar et al., 2016). These missing values can be of different nature: (a) the peptide is truly present, but under the detection limit of an instrument, (b) the peptide is truly present, but is not detected or is incorrectly identified and (c) the peptide is not present. Therefore, there should be different mechanisms for dealing with the particular cause of missingness. In the case of (a) and (c) the values are termed "censored" and ignoring or imputing such data based on the observed results might lead to peptide abundances overestimation and thus biased results (Karpievitch et al., 2012). In proteomic data we are mostly dealing with so-called "left-censored values", meaning that the distribution is truncated on the left side where low-intensity proteins occur.

The problem with missing values is problem especially for the differential expression, because these, so-called qualitative changes (i.e. the protein is present in one condition and completely missing in the second one) are not reflected in the statistical testing, so we lose quite significant portion of potential hits.

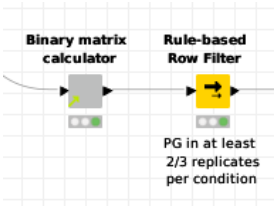
Currently there are several approaches how to deal with the missing values, e.g. knn, bpca, QRILC, MLE, MinDet, MinProb, min, zero (and others, described e.g. in MSnbase R package). In our KNIME environment, we apply three different strategies for the differential expression accounting for missing values:

- Imputation by the global minimum of a dataset, followed by LIMMA test
- Imp4p R package (https://cran.r-project.org/web/packages/imp4p/imp4p.pdf), followed by LIMMA test
- proDA (https://www.biorxiv.org/content/10.1101/661496v1) performing differential expression without an imputation

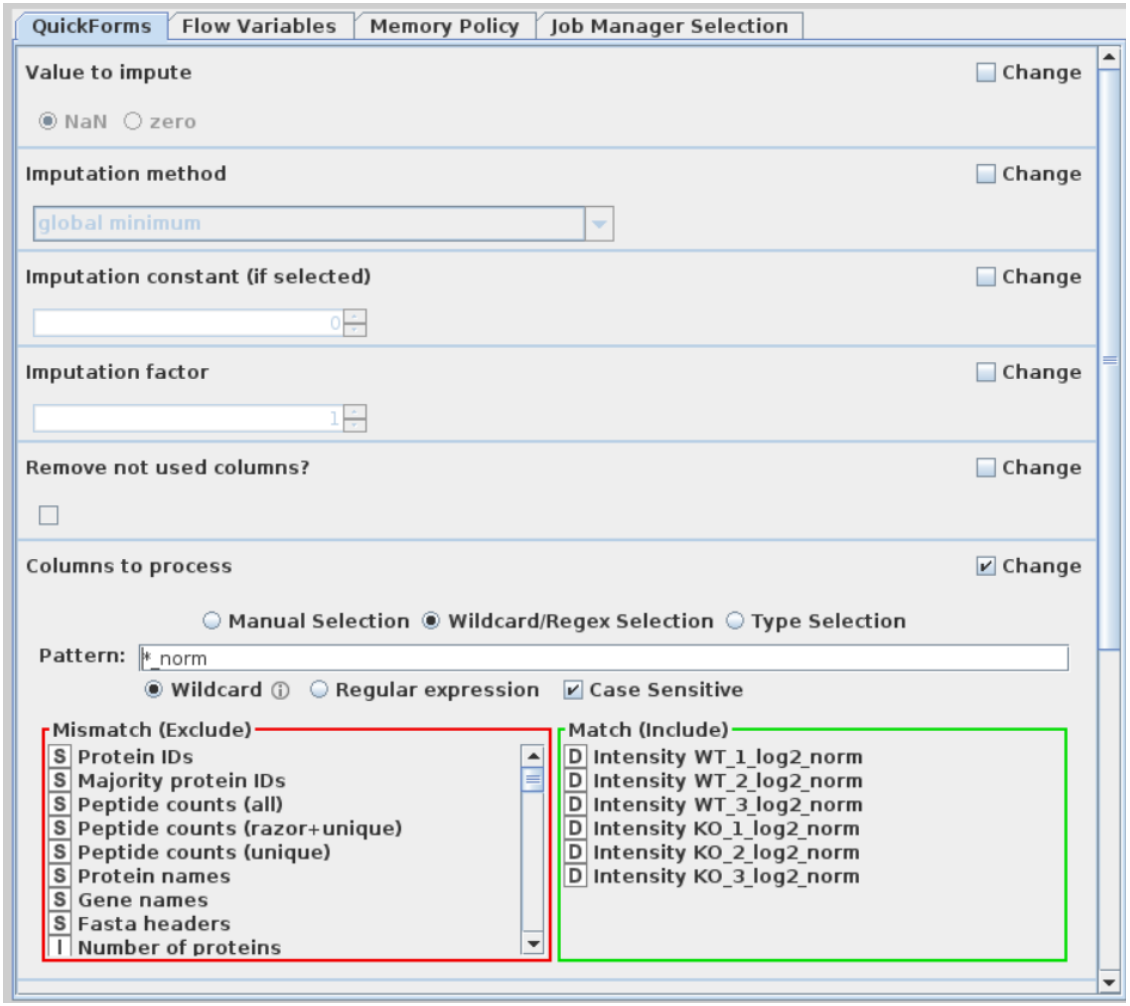
So, in the last part of the workflow we will impute the missing data in the protein intensities by all three methods, and we will compare and visualize the results.

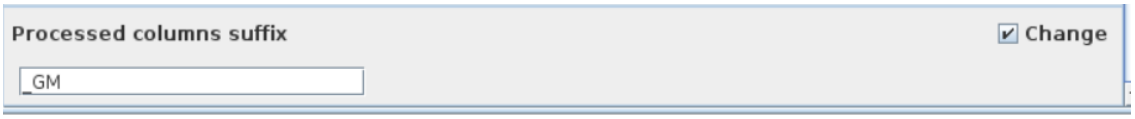
We typically do the imputation on the log2 transformed and normalized protein intensities (this might be an issue for the imp4p package, which is designed primarily for the imputation on the peptide level, which we are tackling by using modified parameters to the original ones as will be shown further).

It is also advisable to perform imputation on filtered data matrix – we will remove protein groups that have less than 2 values in 3 replicates of both WT and KO sample – i.e. protein group needs to have at least 2 intensity values in either WT or KO sample. To achieve this, we will use again Binary matrix calculator combined with Rule Based Rows Splitter similarly as before the Reactome metanode. This time the "Pattern" should be "Intensity *_norm" to choose all normalized intensity columns; and the intensity should be >0, so the logical operator will be ">" and the threshold will be "0". Settings of the Rule-based row splitter will remain unchanged. This step is important at this step because of the 4th replicate removal as well – we might have quite a few protein groups having all 6 remaining Intensity values zero/missing and we might end up in the situation of statistical significance being calculated on imputed data only!



Now let's start with the imputation by the **global minimum**, which is done via the Values imputation node. Open the node to see the settings (F6). We will impute the NaN values and the Imputation method will be "global minimum". The Columns to process are the normalized intensities, so using the Wildcard selection (Intensity *_norm or *norm) we will select the columns to be processed. Finally, we will select the processed column suffix to be _GM.



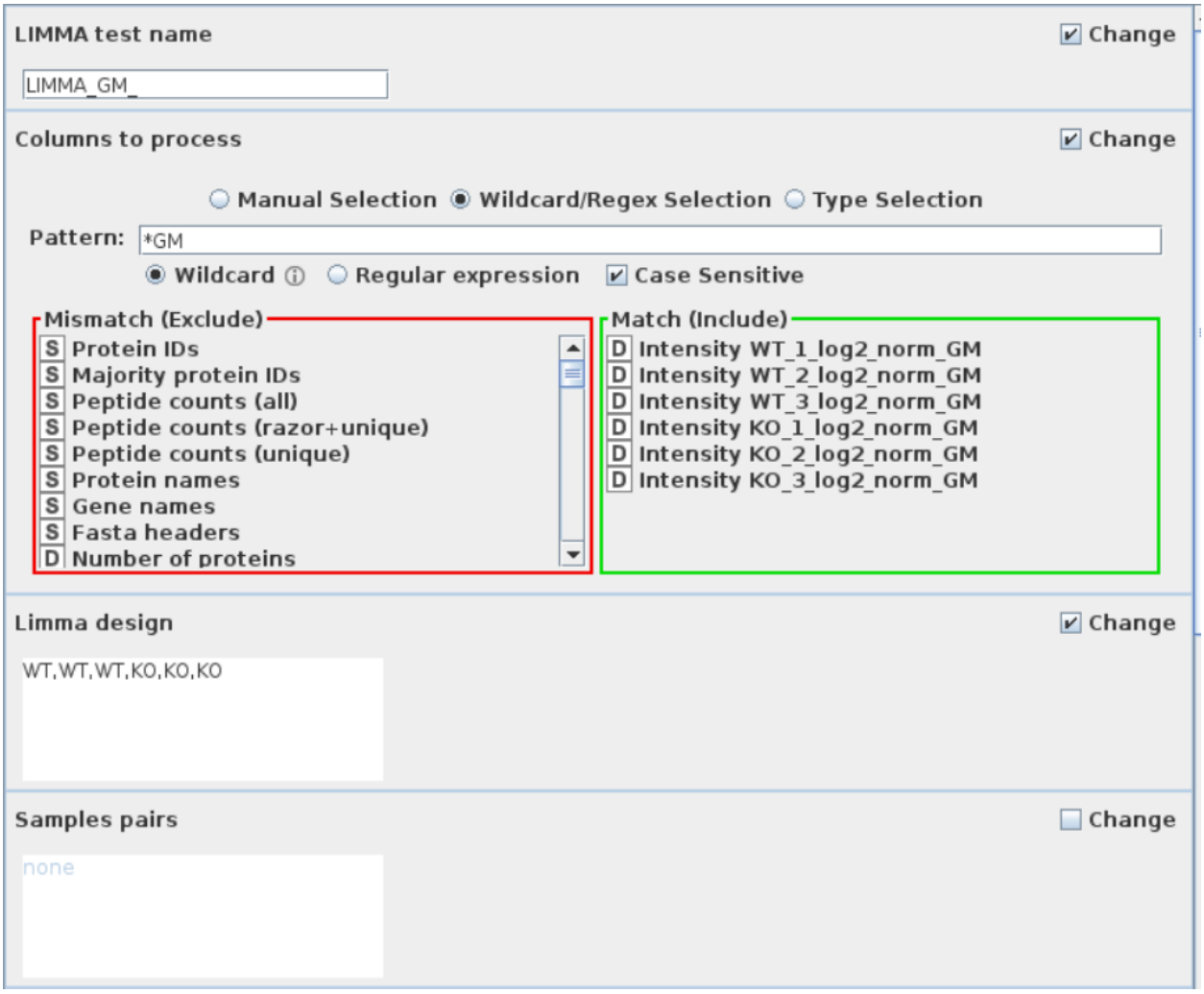


Although the node offers more options to be defined, we will not use them, if you would like to have more information, please, refer to the metanode documentation.

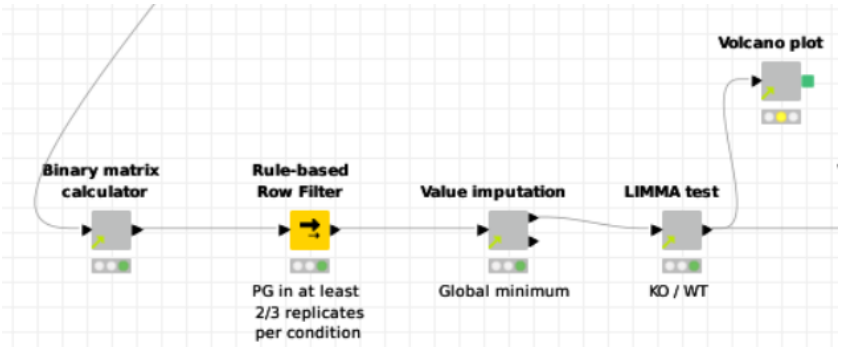
In this setup, we can execute the node (F7). It will find the global minimum across all the chosen columns and substitute the missing values by the global minimum value. On the output of the node, we will have two tables: the top table contains the original table with imputed columns appended; the bottom output then contains the table with the true/false matrix based on the imputation (i.e. imputed values = 'True', original values = 'False').

Let's now run the LIMMA test and see how the volcano plot changes after the missing values were imputed:

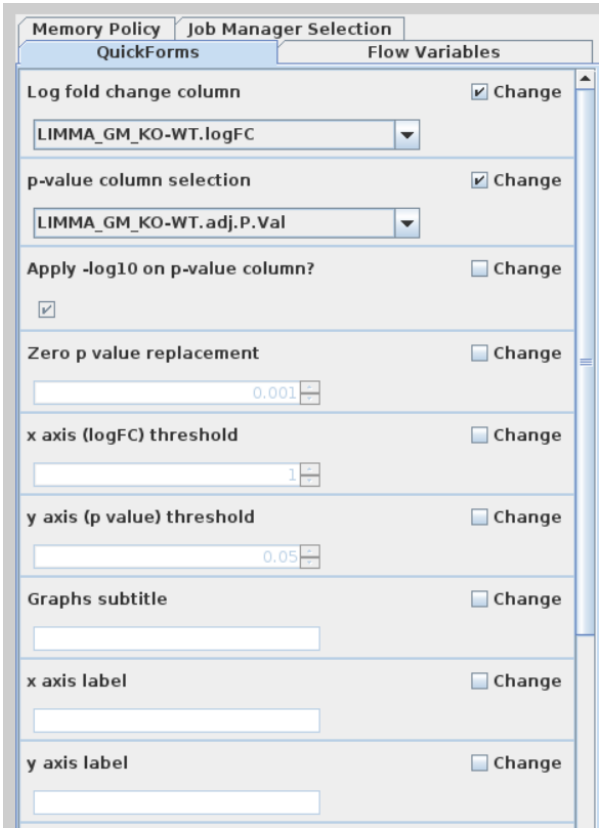
In LIMMA we will specify the test name as LIMMA_GM_ this time and we will choose the columns containing _GM suffix. Again, we will specify the design corresponding to the column to process, so "WT,WT,WT,KO,KO,KO" and the we are testing the changes of KO compared to WT.



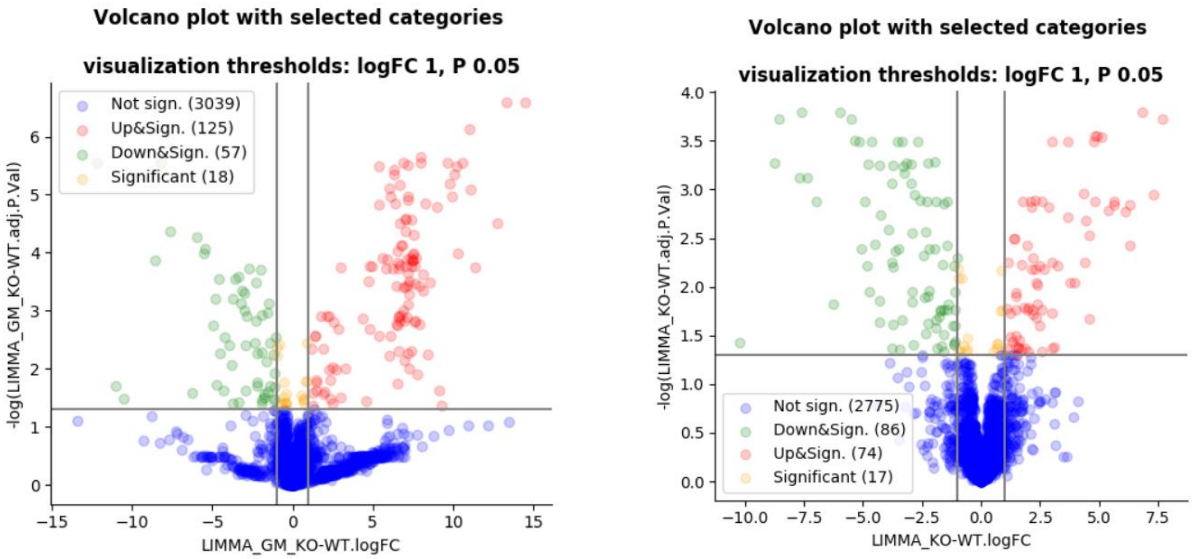
Execute the node (F7). We will now also run the Volcano plot, so the resulting pipeline looks as following:



In the volcano plot only choose the corresponding columns containing the results of LIMMA test (logFC and adj.pvalue):



Execute the node and we will look at the output image (left). We can now also compare to the volcano plot we have seen without any imputation (right):



We can see, that we got much more proteins after the imputation compared to the situation without any imputation.

We in general recommend to always look into the raw data (normalized intensities) whether the imputation worked correctly. In our case, let's look on our Protein A (row 17):

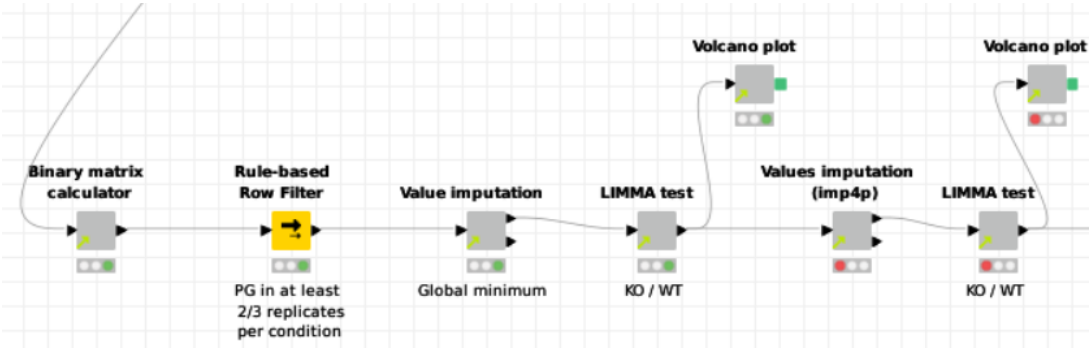
Row ID	Protein IDs	Protein names	Gene names	Intensity WT_1_log2_n...	Intensity WT_2_log2_n...	Intensity WT_3_log2_n...	Intensity KO_1_log2_n...	Intensity KO_2_log2_n...	Intensity KO_3_log2_n...	Intensity WT_1_log2_no...	Intensity WT_2_log2_no...	Intensity WT_3_log2_no...	Intensity KO_1_log2_no...	Intensity KO_2_log2_no...	Intensity KO_3_log2_no...	LIMMA_GM_KO-WT.log...	LIMMA_GM_KO-WT.P.Va...	LIMMA_GM_KO-WT.adj....
Row17	A4D1E9	Protein A	GTPBP10	?	22.294	22.698	?	?	?	14.687	22.294	22.698	14.687	14.687	14.687	-5.206	0.086	0.323

We can see that this protein has in the WT condition one missing value (in the 1st replicate), in the KO condition it is completely missing [columns 5-10]. After the imputation by the global minimum we get in all the cells where there was previously "?" the value of global minimum, in this case 14.687 [columns 11-16]. When we run the LIMMA test, this protein has logFC = -5.206 and the adjusted p-value = 0.323, so it would be in the non-significant section.

Imputation should be used wisely and the optimal technique differs from case to case, so it is always needed to check whether the results are meaningful. As mentioned previously, we also recommend to do any filtering (qualitative or/and quantitative) prior to imputation in order to decrease the number of false positive hits.

Next, we will apply the imputation by the **imp4p** R package. This package can distinguish between different mechanism of missingness and impute the values accordingly. We are using the impute.mi function of the package, applying the multiple imputation strategy.

The metanode can be also found in KNIME Explorer -> KNIME_metanodes -> Metanodes templates -> Data processing (Values imputation (imp4p)). As in the case of imputation by global minimum, we will use the imputation by imp4p followed by the LIMMA test and subsequent volcano plot visualization:



Let's configure the imp4p node: Columns to process are similarly to the Global minimum imputation the log2 transformed normalized protein intensities, so select Wildcard selection and "*norm". Further, we specify the Conditions (or experimental design), again, the order of the conditions should correspond to the order of columns to process: "WT,WT,WT,KO,KO,KO". We will not set any biological or technical replicates. Due to the imputation on protein level, not peptide level, we will slightly adjust the q.min and eps parameters, so set the q.min to 0.025 and eps to 0, other options leave unchanged. These values are related to left censored missing data imputation and suggested values should give you relatively high imputed values (up to 2.5 percentile of the observed values). Qualitative changes (i.e. protein groups quantified only in one sample type) close to minimal intensities will therefore be likely not significant in contrast to global minimum imputation technique. Here you can see the explanation for the particular parameters:

q.min
A quantile value of the observed values allowing defining the maximal value which can be generated. Default is 0 (the maximal value is the minimum of observed values minus eps)

q.norm
A quantile value of a normal distribution allowing defining the minimal value which can be generated. Default is 3 (the minimal value is the maximal value minus qn*median(sd(observed values)) where sd is the standard deviation of a row in a condition)

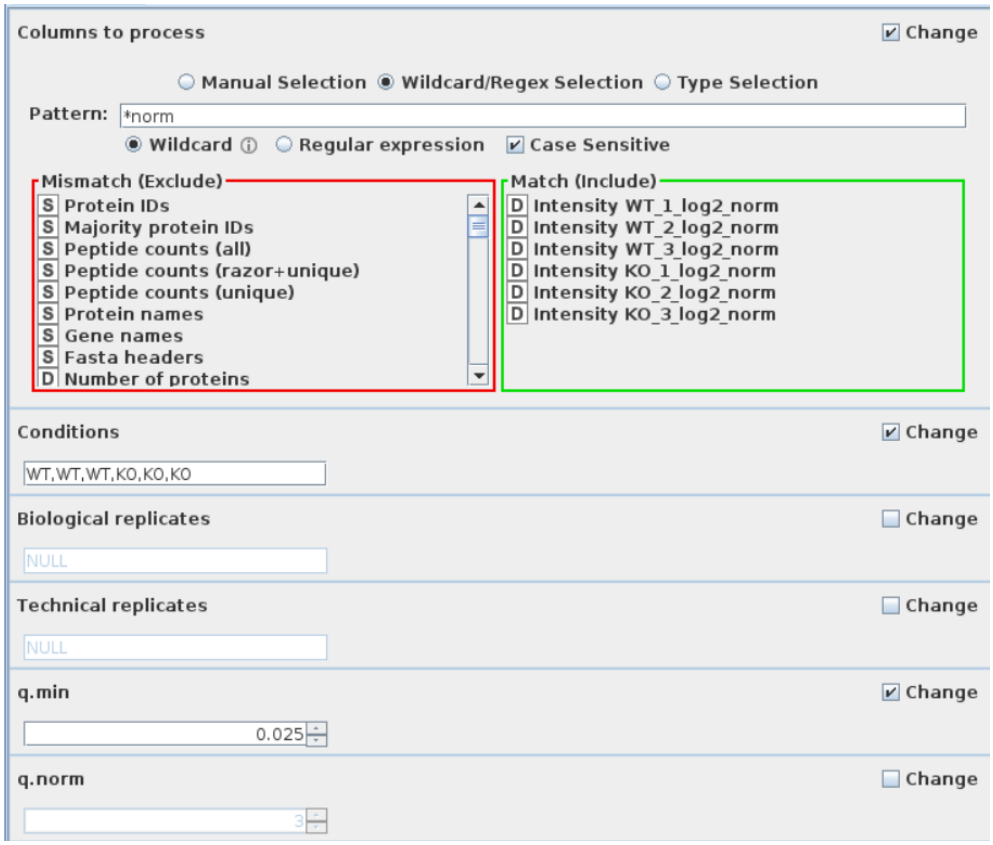
eps
A quantile value of the observed values allowing defining the maximal value which can be generated. Default is 0 (the maximal value is the minimum of observed values minus eps)

Number of iterations (nb.iter)
The number of iterations used for the multiple imputation method

Number of nearest neighbours (nknn)
The number of nearest neighbours used in the SLSA algorithm

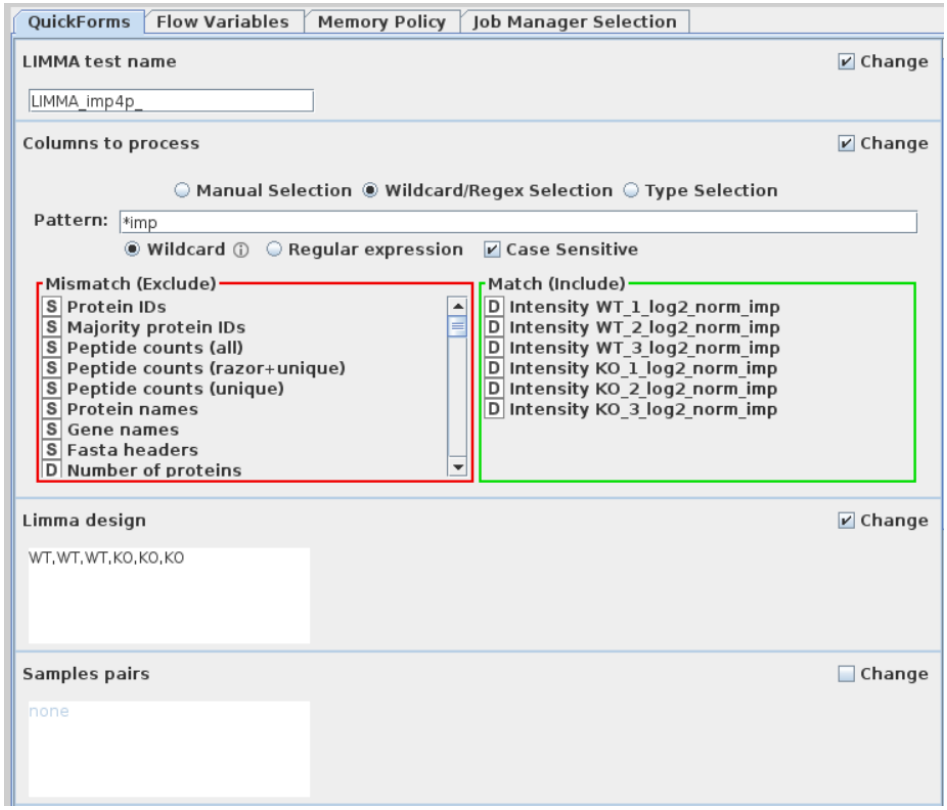
For the further information, please refer the package documentation.

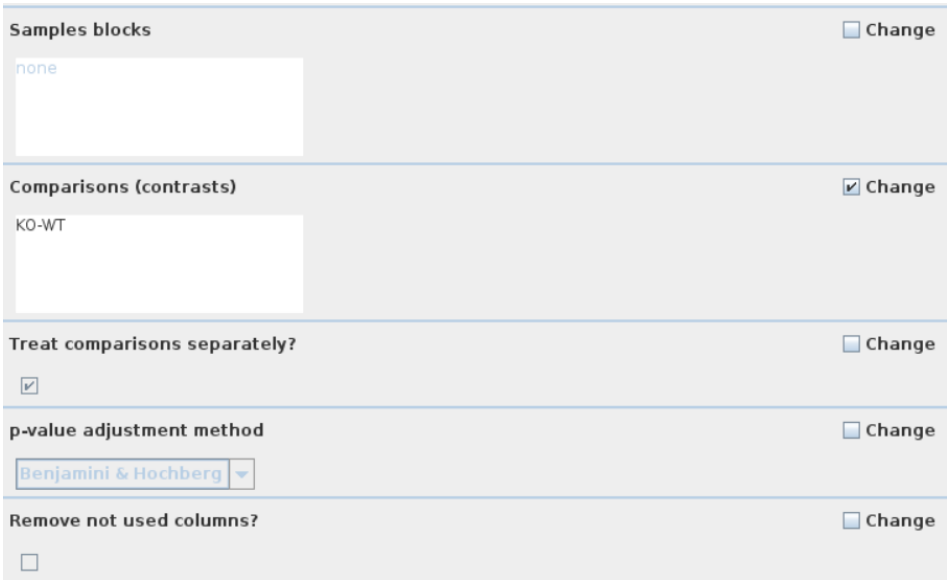
The resulting settings will look like following:



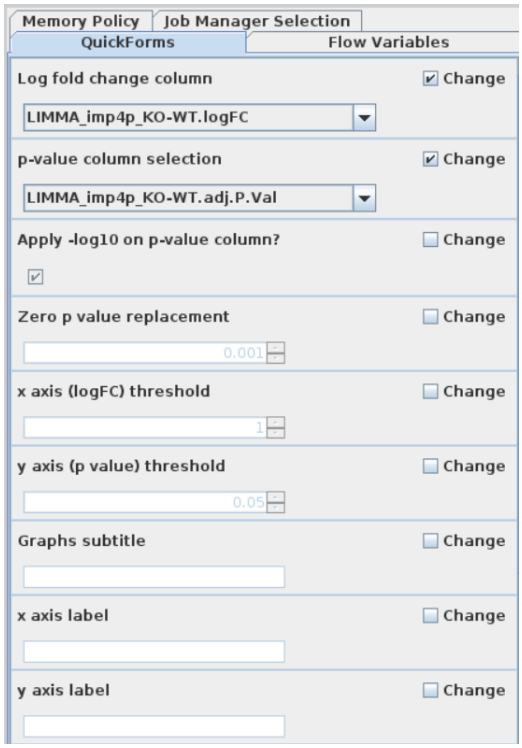
Execute the metanode (F7).

The LIMMA test will be very similar to the global minimum, just we will change the test name to LIMMA_imp4p_ and the column to process will have to finish with the _imp suffix as defined in the form above.

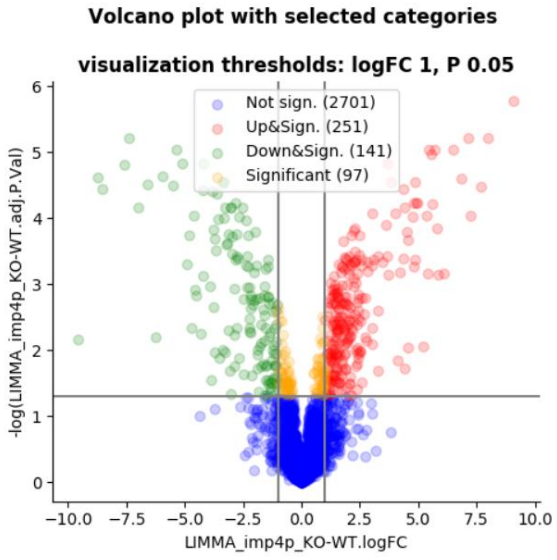


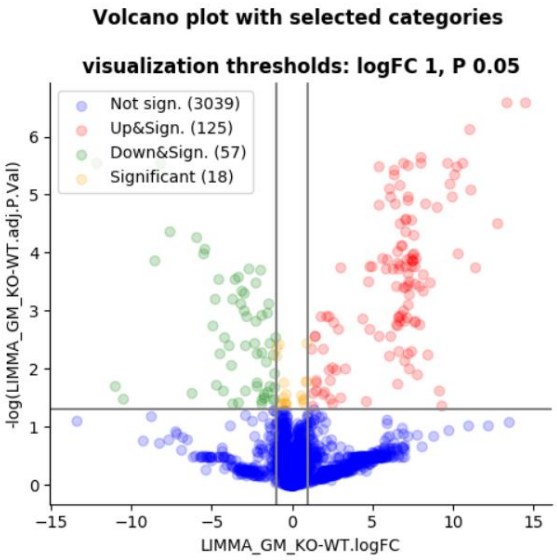


Execute the node and we will move to the visualization by the volcano plot. In here, again, majority of the settings will stay the same, just change the logFC and adj.pvalue columns to be the ones carrying results for the LIMMA of imp4p imputed values.



We can now see how the volcano plot looks like after imp4p imputation (left), let's compare it with the imputation by the global minimum (right). You should be able to notice, that the pattern in the plots is different and by the imp4p we are getting more proteins in the upregulated and downregulated parts:





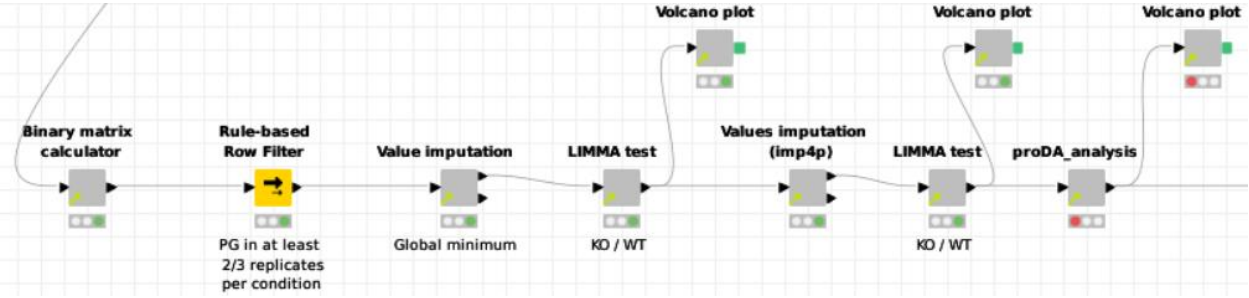
Let's also check how the imputation looks in the raw data for our Protein A:

Row ID	Protein IDs	Protein names	Gene names	Intensity WT_1_log2_n...	Intensity WT_2_log2_n...	Intensity WT_3_log2_n...	Intensity KO_1_log2_n...	Intensity KO_2_log2_n...	Intensity KO_3_log2_n...	Intensity WT_1_log2_n...	Intensity WT_2_log2_n...	Intensity WT_3_log2_n...	Intensity KO_1_log2_no...	Intensity KO_2_log2_no...	Intensity KO_3_log2_no...	LIMMA_imp4p_KO-WT.l...	LIMMA_imp4p_KO-WT....	LIMMA_imp4p_KO-WT....
Row17	A4D1E9	Protein A	GTPBP10	?	22.294	22.698	?	?	?	22.366	22.294	22.698	20.411	20.384	19.926	-2.212	0	0.001

In this case, for the 1st replicate of the WT condition the value was imputed with a value close to the rest two of the same condition (so 22.366 imputed; 22.294 and 22.698 present). For the KO condition, where all the values were missing, were imputed lower values around 20. After the LIMMA test, the protein was classified to be down-regulated as it has logFC = -2.212 and adjusted p-value 0.001. In this case we can see that the imputation worked the expected way.

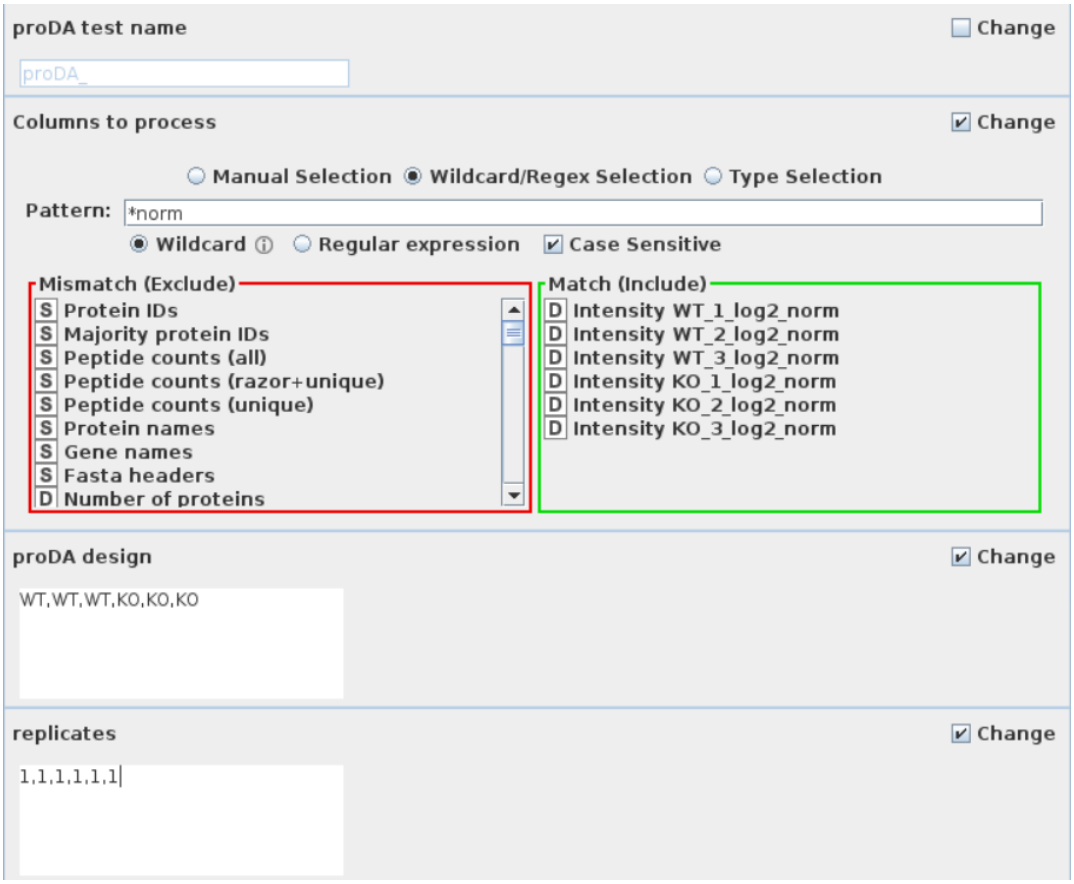
It should be noted though, that missing value in the 1st replicate could have been also left censored and the imputation using imp4p ended up in wrong estimate of protein group intensity in the given replicate. The probability of missing value type depends also on the given experiment design and replicates nature. In case the replicates are e.g. different animals, the replicates variability can be expected higher and one may prefer to use e.g. global minimum imputation to get more robust results.

Lastly, let's use the **proDA** metanode for the differential expression without the imputation based on the article of Constantin Ahlmann-Eltze*: "We present proDA, a method to perform statistical tests for differential abundance of proteins. It models missing values in an intensity-dependent probabilistic manner. proDA is based on linear models and thus suitable for complex experimental designs, and boosts statistical power for small sample sizes by using variance moderation."* In this case, we will use only the proDA metanode and volcano plot for the visualization, the statistics is done inside the proDA metanode, so no LIMMA is needed (proDA uses by default the Wald's statistical test):



proDA in contrast to LIMMA requires a bit different settings and also provides slightly different output. We firstly specify the test name, proDA_; columns to process we can set again using the Wildcards to "*norm" taking all normalized columns. proDA design applies the same as LIMMA, so provide the conditions corresponding with the order of columns to process: "WT,WT,WT,KO,KO,KO". In contrast to LIMMA, proDA always requires replicates specification. In our case, we don't consider any biological or technical replicates, so we will set it as "1,1,1,1,1,1". There always should be a reference sample specified, so the one against all the contrasts will be made – in our case, WT. Then we can specify one or more contrasts, separated by comma, in our case KO. The resulting comparison will be the same as in LIMMA: KO vs WT. Again, we can choose the p-value adjustment method, so Bonferroni-Hochberg for our case.

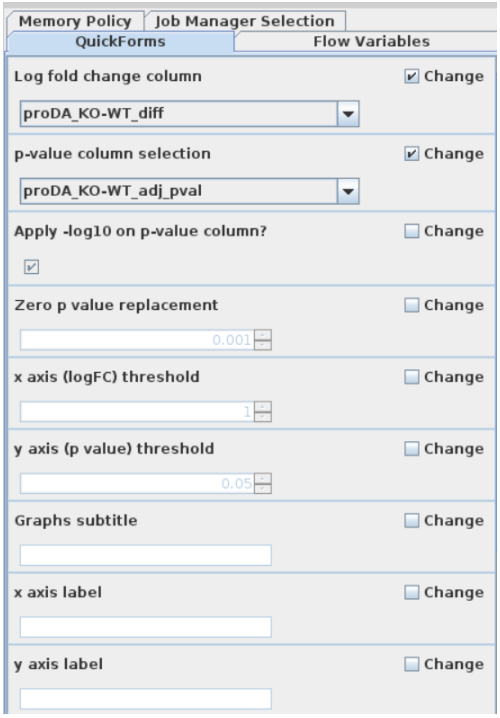
Now you can execute the metanode. Please notice, that both proDA and imp4p might take longer time, especially if there are many samples and complicated design.



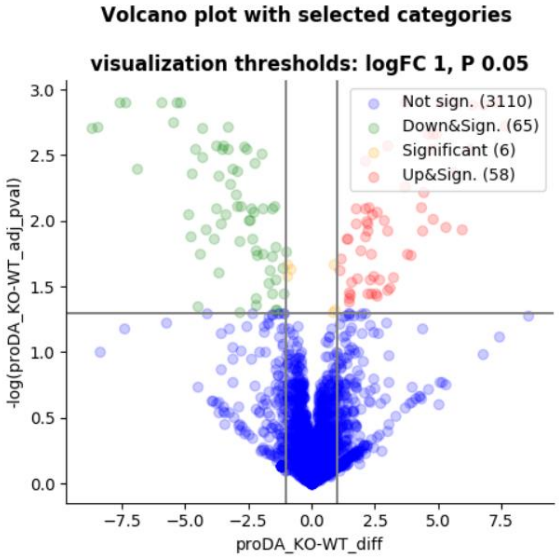
proDA will provide you on the output more columns than ordinary LIMMA:

proDA_KO-WT_pval	proDA_KO-WT_adj_pval	proDA_KO-WT_diff	proDA_KO-WT_t_statistic	proDA_KO-WT_se	proDA_KO-WT_df	proDA_KO-WT_avg_abu...	proDA_KO-WT_n_approx	proDA_KO-WT_n_obs
0.233	0.564	-0.569	-1.404	0.405	4	25.38	6	6
0.031	0.226	4.111	3.275	1.255	4	22.687	3.015	3
0.177	0.511	-0.232	-1.637	0.142	4	21.257	4	4
0.973	0.985	-0.009	-0.036	0.244	4	29.784	6	6
0.379	0.683	0.698	0.989	0.706	4	23.718	6	6
0.112	0.421	2.677	2.033	1.317	4	21.984	1.768	2
0.721	0.862	0.173	0.383	0.451	4	24.936	6	6
0.435	0.72	0.269	0.867	0.31	4	24.905	6	6
0.303	0.623	-0.46	-1.182	0.389	4	23.266	6	6
0.589	0.797	0.736	0.586	1.255	4	20.997	1.995	2
0.129	0.448	-2.362	-1.909	1.237	4	20.592	0.996	1
0.486	0.735	-0.222	-0.766	0.29	4	22.108	6	6
0.305	0.623	0.205	1.174	0.174	4	25.4	6	6
0.011	0.137	1.786	4.521	0.395	4	22.576	3.981	4
0.062	0.311	-3.078	-2.569	1.198	4	20.955	1.982	2
0.031	0.226	-0.574	-3.257	0.176	4	24.676	6	6
0.526	0.761	0.097	0.693	0.14	4	25.766	6	6
0.327	0.64	0.404	1.117	0.362	4	19.443	4.021	4

For us, the most interesting are again pvalue (pval), adjusted pvalue (adj_pval) and logFC (diff). We can now visualize the results in the volcano plot (change again only the columns for logFC and adjusted pvalue):



In comparison to previous two imputation approaches, proDA considers majority of the proteins to be not significant and lower portion is considered to be significant, up or down regulated.



Let's look again on our Protein A in the raw data:

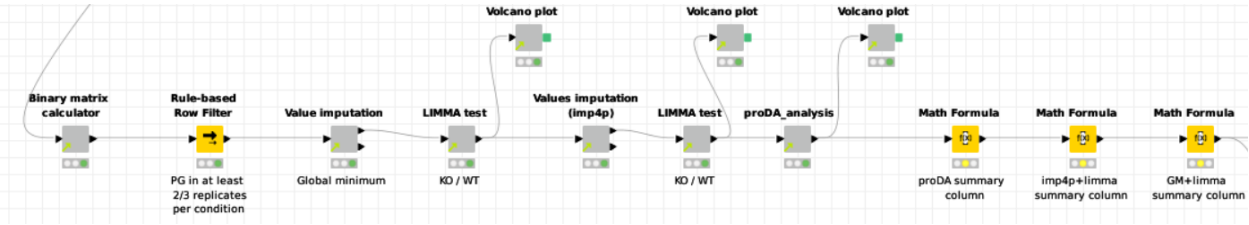
Row ID	Protein IDs	Protein names	Gene names	Intensity WT_1_log2_n...	Intensity WT_2_log2_n...	Intensity WT_3_log2_n...	Intensity KO_1_log2_n...	Intensity KO_2_log2_n...	Intensity KO_3_log2_n...	proDA_KO-WT_pval	proDA_KO-WT_adj_pval	proDA_KO-WT_diff
Row17	A4D1E9	Protein A	GTPBP10	?	22.294	22.698	?	?	?	0.062	0.311	-3.078

Similarly to global minimum imputation, proDA considers it to be non-significant as the diff (=logFC) = -3.078, but the adjusted p-value is 0.311.

In our case, looking only at the chosen Protein A we could say that the imp4p technique worked the best in case we would have the protein independently verified as qualitatively changing e.g. by western blot.

In general, the pattern which we observe quite often is, that imp4p usually identifies highest numbers of differentially expressed proteins, but in case the pre-filtering of the dataset is omitted, it can result in the high false positivity rates. Global minimum, together with proDA results tend to be more conservative, leading to lower number of false positives and higher number of false negatives. We therefore recommend to always try several imputation techniques, look into the raw data whether the provided results are meaningful and imputation technique fits the expected reason of missing values presence and potentially known candidates' behavior.

Finally, let's visualize how many upregulated proteins were identified by all three imputation techniques. For this purpose, we will use the **UpSet plot**. Prior the plotting we need to firstly filter the upregulated proteins, so the ones passing the criteria of having logFC >1 and adjusted pvalue <0.05. The easiest way is to use the Math formula nodes:

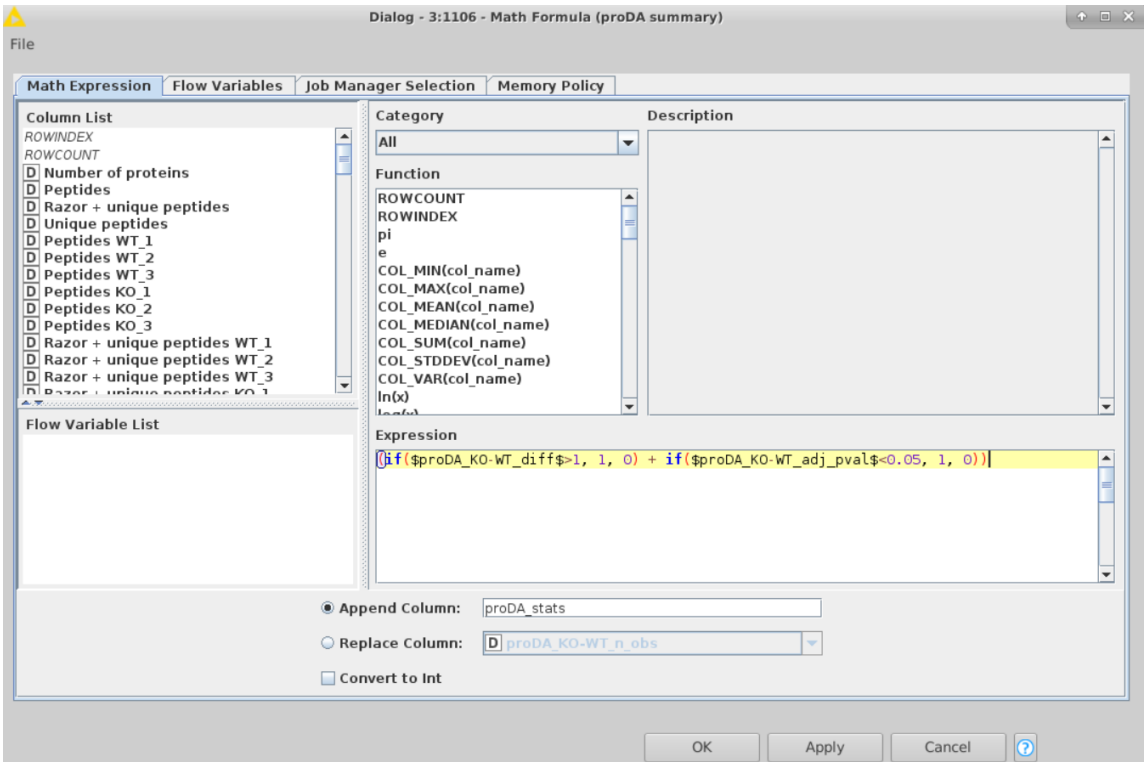


Let's configure them. Within the Expression form we will specify the thresholds in sort of Excel way:

(if($proDA_KO-WT_diff$>1, 1, 0) + if($proDA_KO-WT_adj_pval$<0.05, 1, 0))

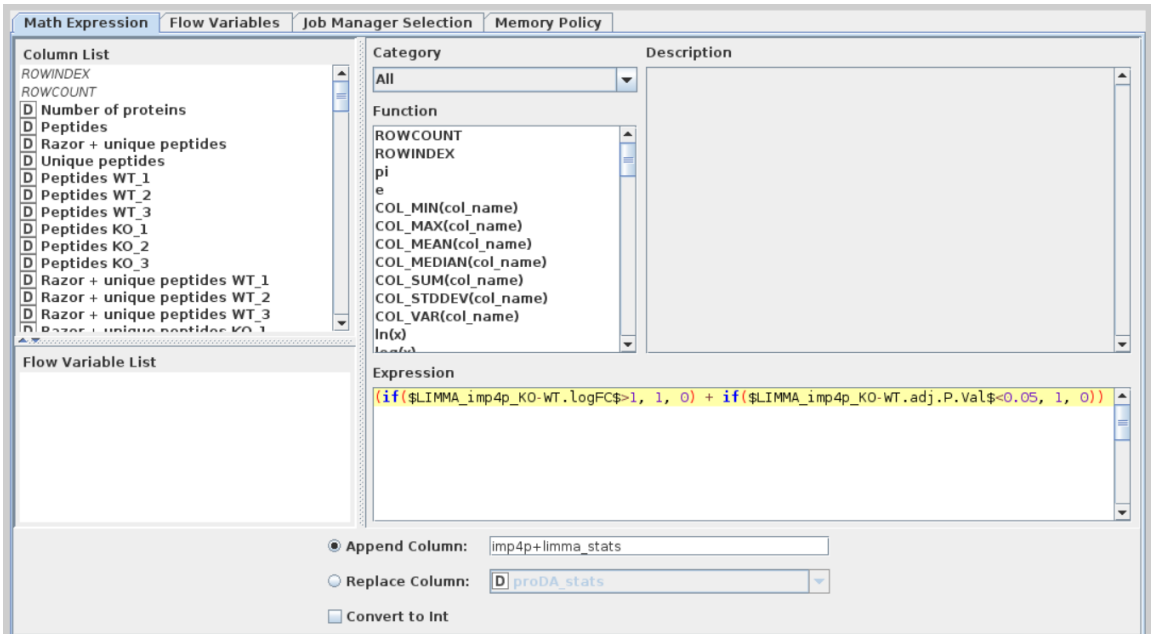
We have here the IF expression: IF(condition,TRUE,FALSE). And because both conditions should fulfilled at the same time, we will use "+" to count them.

The node will append also the new column called "proDA_stats" specifying whether the protein passed zero conditions (0), one of the conditions (1) or both of them (2).

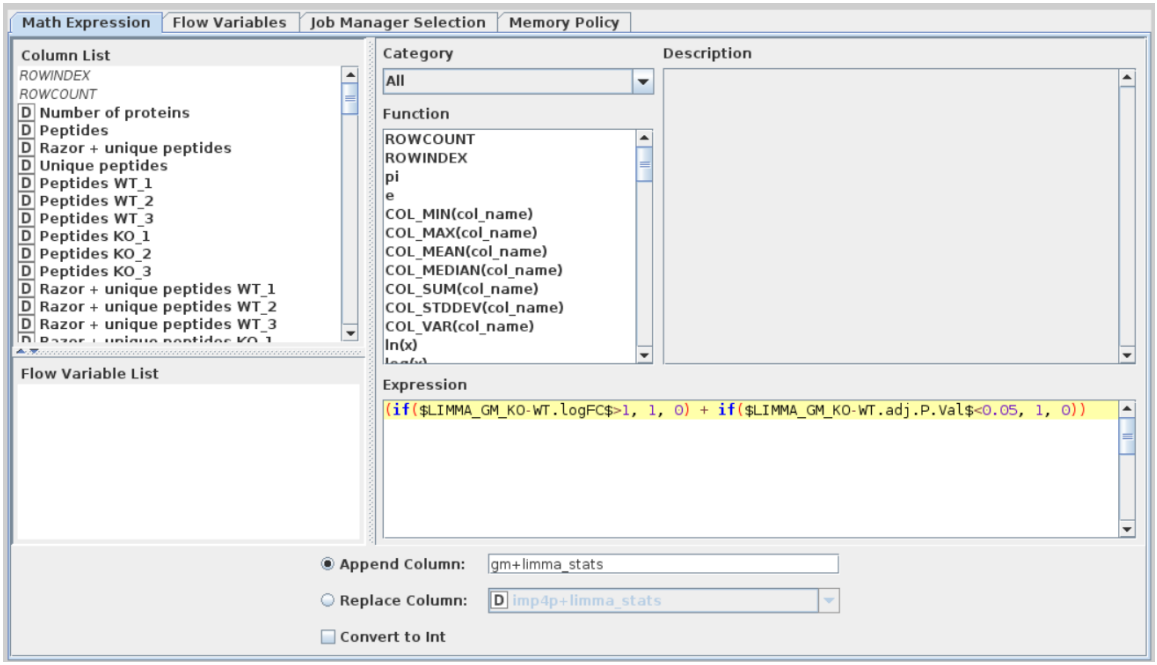


Similarly, we will configure also the next two Math formula nodes:

Imp4p+limma_stats:



GM+limma_stats:

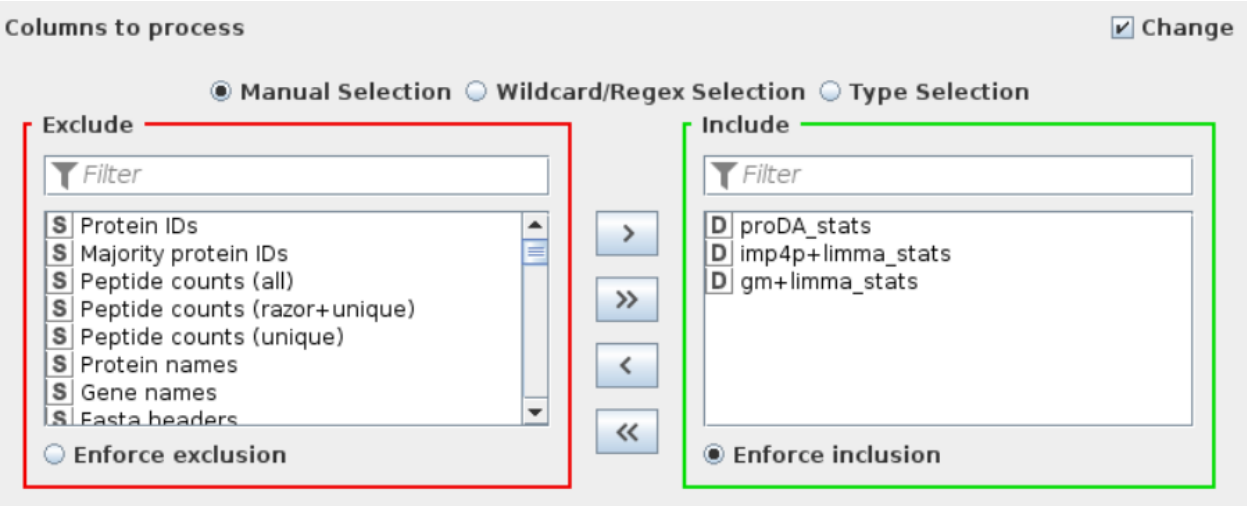


On the output there will be now three columns connected containing the matrix:

D proDA_stats	D imp4p+limma_stats	D gm+limma_stats
0	0	0
1	2	2
0	0	1
0	0	0
0	0	0
1	2	1
0	0	0
0	0	0
0	0	0
0	2	1
0	1	0
0	0	0
0	0	0
1	2	1
0	1	0
0	0	0
0	0	0
0	0	1
0	0	0

This matrix can be subjected to the UpSet plot. UpSet plot is an alternative to Venn diagram, which becomes hard to read if more than 3 samples are compared. Similarly to Venn diagram, UpSet plot allows us to display the intersections between particular datasets, so we can tell how many proteins are common for the particular combination of samples.

Let's configure the node: Columns to process are the last three columns we got by the Math Formula nodes, containing the information whether the protein was upregulated or not:

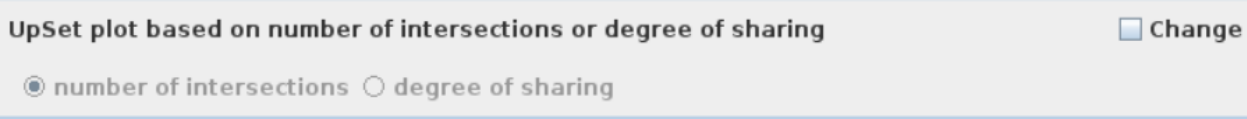


We can further select the ordering of the intersections. Options include number of intersections (i.e. the order is given by the number of intersections in datasets, descending order) and degree of sharing (i.e. the order is given by the number of datasets the intersection is present in). In our case, we will leave it on "number of intersections".

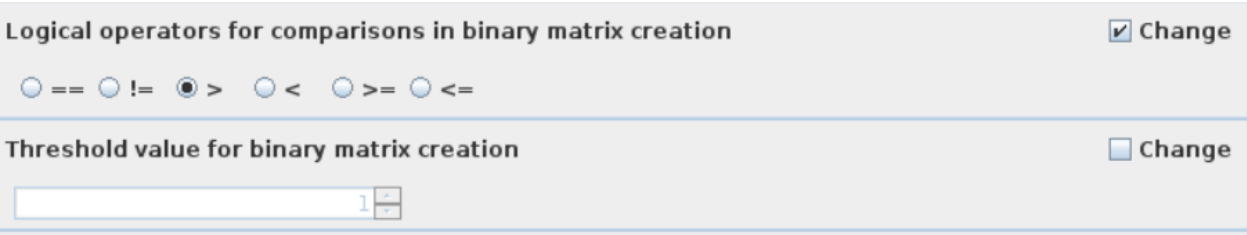
To illustrate the options, let's assume we have three datasets: A,B,C.

If the ordering is based on the number of intersections, the order of bars will be given by the number of proteins shared in particular datasets, so e.g. A+B+C (100 proteins), A+C (50 proteins), B (20 proteins), A (10 proteins), etc.

If the ordering is based on the degree of sharing, the order of bars will be given by the number of datasets the intersection is present in, so e.g. 3 datasets, so 3 degrees of sharing -> A+B+C, then 2 degrees of sharing (A+B, B+C, A+C) and then 1 degree (A, B, C).



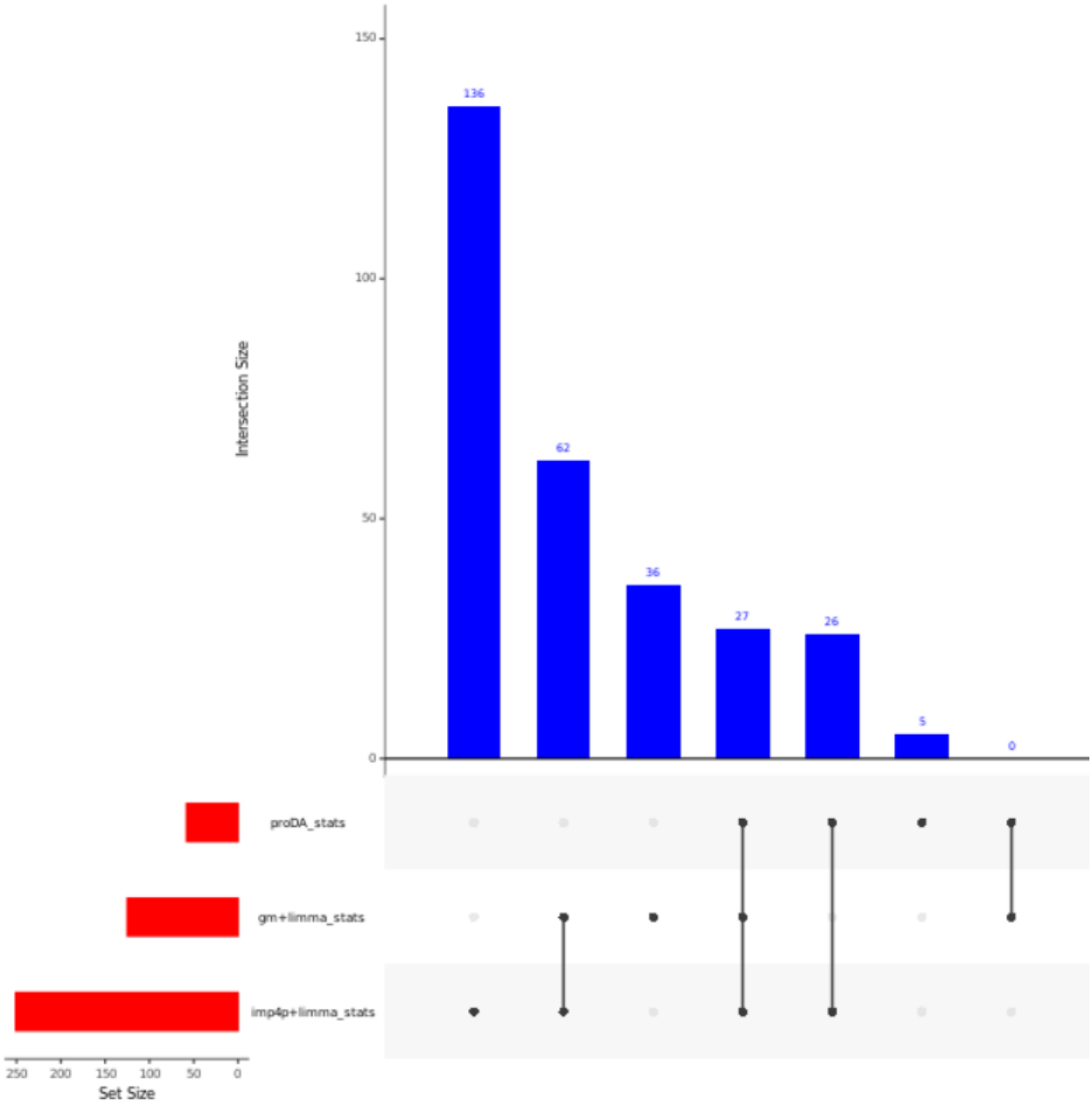
UpSet plot will be creating a binary matrix inside based on logical operator and threshold. In our case, we will be concentrating on the proteins which are upregulated and therefore are marked by the number 2 in the columns we're processing.



We can also select the lower threshold for binary matrix row sum, so if the sum is lower than the specified threshold, it will not be included in the data displayed in the UpSet plot application; 0 means all rows will be displayed. We will not tick this in our case.

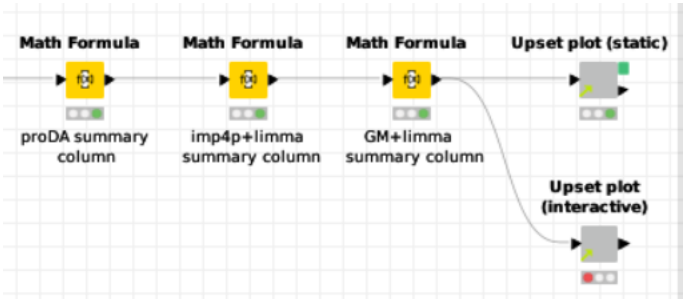
We can further specify the number of sets and intersections to be displayed or whether to show also the empty intersections. Leave all the rest of the settings in the defaults state and execute the node.

On the output we will get the resulting plot in the top port, and the binary matrix created in the bottom port:

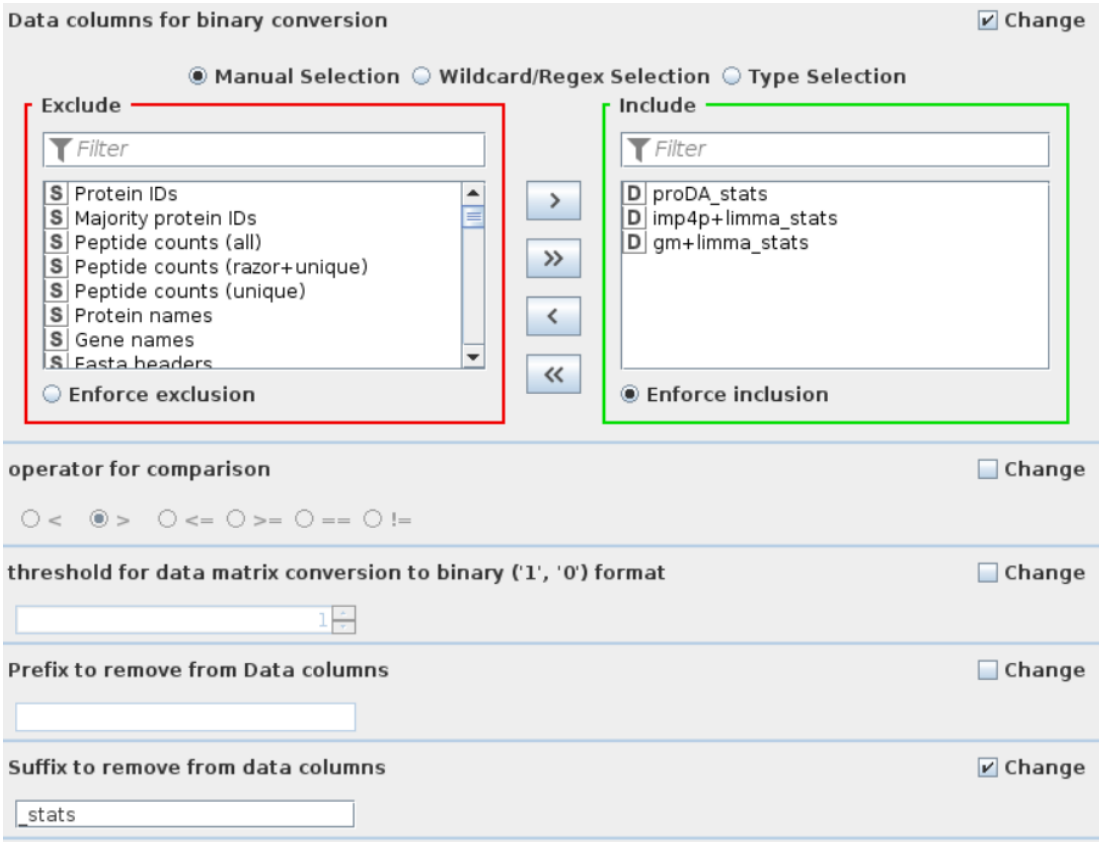


The vertical columns in a graph (blue ones) represent number of intersections, horizontal columns (red ones) represent the size of particular dataset. The cells provide an information whether the particular dataset is a part of intersection (black color) or not (light grey color). In our case, 136 proteins are unique for imp4p imputation, 62 are common for global minimum and imp4p, etc. As you can see, this can be very useful for comparing more samples.

We can use also the interactive version of the upset plot, which was based on the following work: https://caleydo.org/tools/upset/ where you can find also detailed help.



Let's now set the node: exactly as in the case of static upset plot, we will choose the stats columns produced by Math formula nodes and specify the logical operator and the threshold for binary matrix creation. We can also remove a prefix or suffix from the column names:

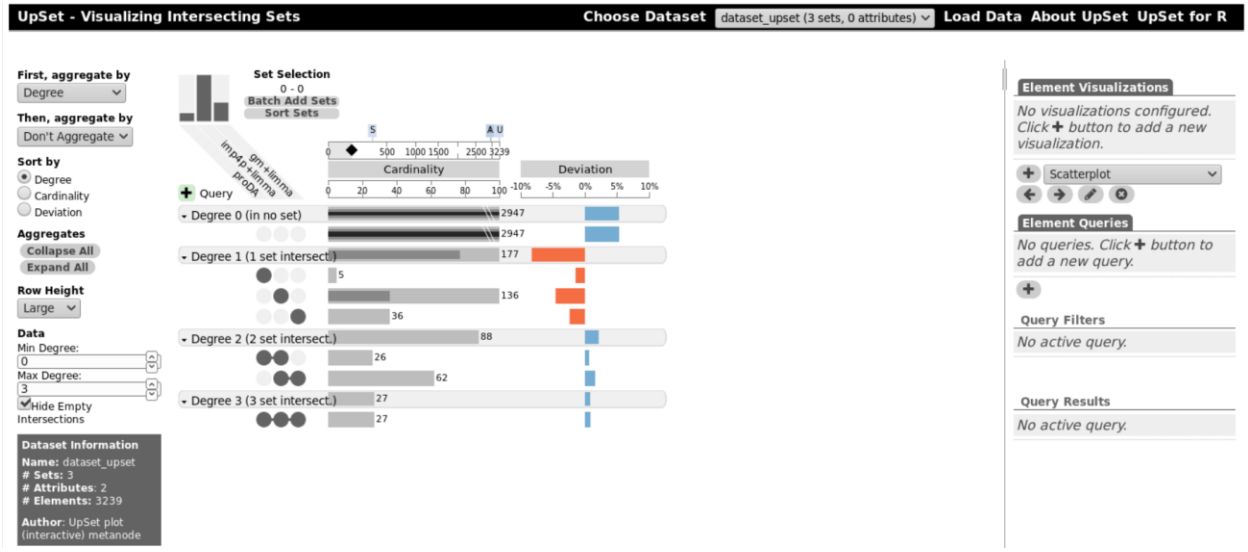


Then there is option to specify any metadata of string (text; e.g. Protein names, Gene names, etc.), integer (whole numbers; e.g. number of peptides) or float (numbers containing decimal places; e.g. logFC) type. These metadata will be then displayed in the resulting graph/application, but we really recommend to use only necessary metadata as addition of each new component significantly slows down the application and might eventually cause its crash and necessity of running the node again.

So, for our case, we will now not choose any metadata to display.

The UpSet plot also comes with “Applications” folder you may have already noticed in the KNIME_metanodes, so don’t please change the path (or change it accordingly to the place where you store the UpSet application). The metanode can be found in gitfolders -> KNIME_metanodes -> Metanodes templates -> Graphs -> UpSet plot (interactive).

Now let’s execute the node. A browser with the UpSet application will be opened and you will notice to be still in the state of executing. This is ok, don’t worry, it means that the webserver is running from that concrete metanode. You can cancel the metanode then or close the workflow with the running metanode to stop the webserver.



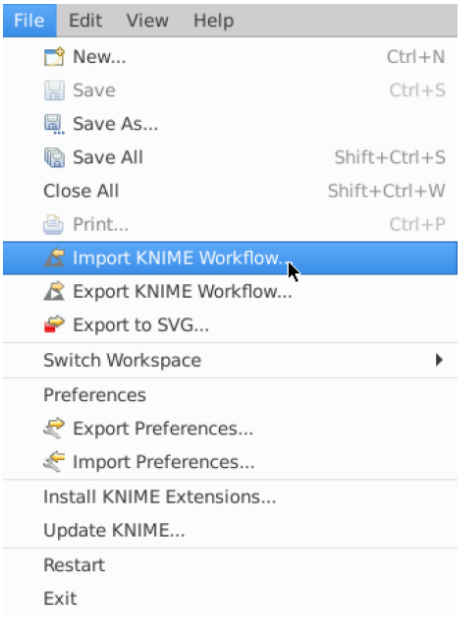
This interactive UpSet plot provides several options to aggregate or sort the results, also what is very useful is the possibility to create queries (so to specify in which particular samples intersections are you interested in and in case you have e.g. Protein names as metadata, you will see the list of the proteins in the particular intersection in the Query Results).

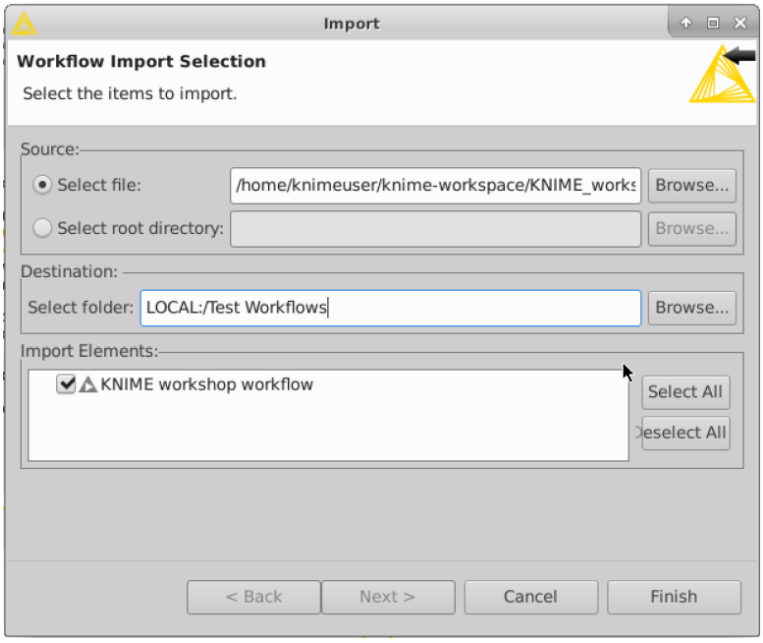
3.14 Complete KNIME workflow availability

Completely processed workflow is available on GitHub pages - https://github.com/OmicsWorkflows/Workshops/tree/master/workshop_2019.

Download the compressed workflow file (.7z extension) and extract the exported workflow (file with .knwf extension) into your KNIME workspace. Once extracted, import the knwf file into your KNIME instance using the menu File -> Import Workflow.

On the new window (right side figure below) select the downloaded file as the file to import (using "Browse..." button next to the "Select file" field) and adjust the folder into which the workflow should be imported (using "Browse..." button next to the "Select folder:" field). The workflow import will be initiated after confirming the form settings using the Finish button:





The workflow extraction and opening will take few minutes depending on your system hardware setup. In case there is low amount of RAM avaiable for KNIME application (< ~4GB) it may also fail to open the workflow.