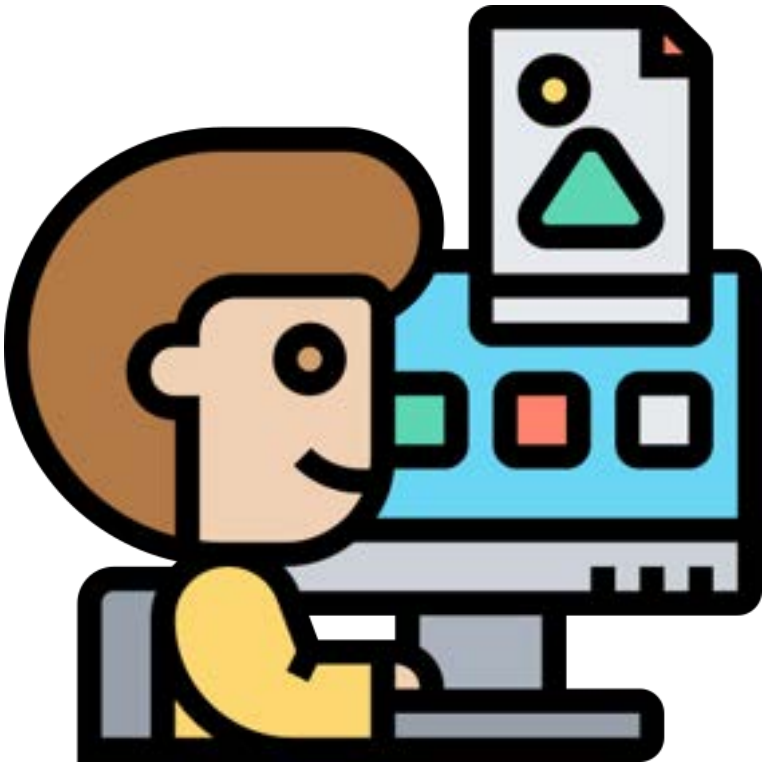
Reporting with KNIME : Literate Programming, BIRT, & Guided Analytics

Annika Hamachers
German Police University, Münster
annika.hamachers@dhpol.de



Abstract

This paper compiles a short overview of three different ways of integrating data analysis with reporting features in KNIME: I will introduce the Jupyter notebook integration, BIRT reports and the possibility to build workflows for Guided Analytics.

1 Introduction

Tools that enable you to showcase your analyses and share the results with the world are important features of every data scientist's toolkit. KNIME offers several

such opportunities. However, some of them might not be quite as sleek as people familiar with programming languages such as R or Python might be used to.

In this text, I will give a quick introduction into the three most prominent reporting tools that can be used from KNIME: the Jupyter notebook integration, the BIRT report editor, and workflow integrations for so-called 'Guided Analytics'.

2 Jupyter Notebook Integration

The term 'literate programming' refers to a paradigm in computer science which can be loosly understood as writing programs that are readable by humans. Usually, this means that the code and the documentation of the code are contained in the same file.

Probably the most popular form of literate coding that many data scientists using R or Python make use of those days are so called 'notebooks' which take code snippets and their documentation and render them into a file format that is easy to share (such as an HTML document) and in which the code is still executable. Notebooks are therefore a sleek way to share your analysis and walk others through them at the same time and are thus, probably the most common format to exchange work between data scientists.

KNIME adopts the notebook approach by offering an integration for so-called jupyter notebooks which are executed in a python environment.

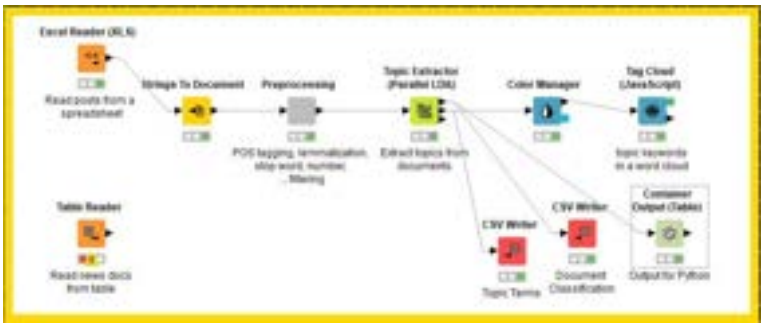


Figure 1: A workflow containing a Container Output node

Via the jupyter integration you cannot only access code snippets stored in such notebooks from within your workflows, but also easily call KNIME workflows from within a jupyter notebook without having to set up a jupyter server.

In order to do this, you need to prepare yourself in two steps:

1. You have to add output nodes to your workflow that can interact with Python (in our case this means incorporating a textbf'Container Output' node for table data that hands the data from the first output port of the topic extractor, the results from the document classification, down to Python).

2. You need to install the KNIME python package from PyPI by typing 'pip install knime' into your (Anaconda) console.

Figure 2: Installing KNIME support for Python

When you are all set up, type 'jupyter notebook' into your console.

Figure 3: Running a Jupyter notebook

This launches the jupyter dashboard in your local browser which shows already existing notebooks on your machine. If you want to create a new notebook for your KNIME workflow, click 'New' and select 'Python'.

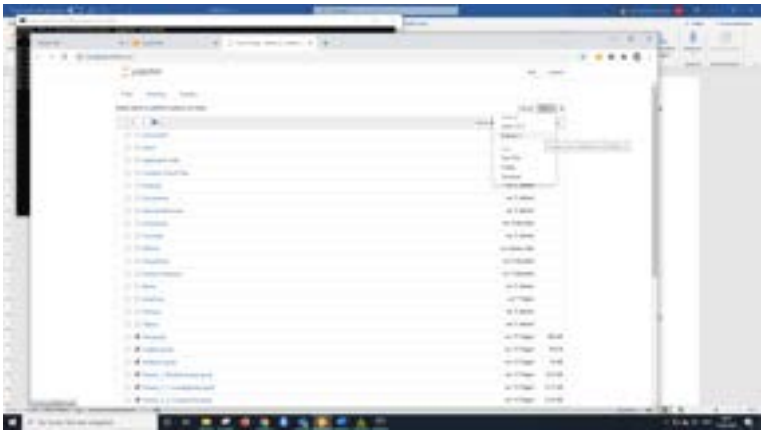


Figure 4: Creating a new Jupyter notebook

By default, new notebooks get the title 'Untitled...'. Thus, the first thing you will likely want to do is change this. Therefore, just click onto the title and rename it:

Figure 5: Renaming the notebook

Now, you can populate your notebook with content. To insert new content cells, click the '+' icon from the editing panel. By default, new cells are code cells. If

you want to create plain markdown text or a heading to structure your notebook, select another formatting from the editing panel.

We start our notebook with a title and an informative subheading and then import all the Python packages we will need:

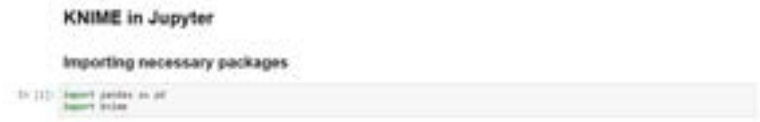


Figure 6: The first lines

Next, we make our workflow accessible in jupyter and print it into the notebook. Therefore, we need to provide 1) the entire path to our local installation of KNIME (to knime.exe), 2) the filesystem path to the KNIME workspace that contains our workflow, and) the name of our workflow itself. We save all three elements into variables which we can then provide as arguments for the `knime.workflow()` function:

Figure 7: Code for calling the workspace

This inserts a scalable vector graphic of our entire KNIME workflow into the jupyter notebook.

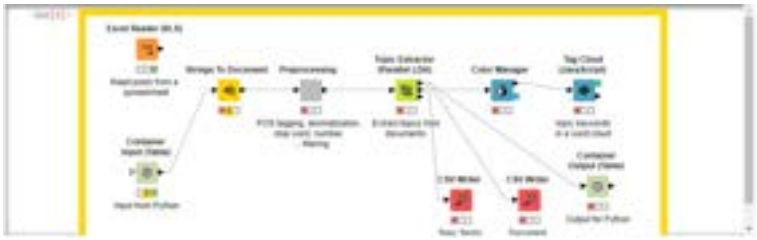


Figure 8: Inserted SVG

If no output is created and you get an error message instead, it is likely that your KNIME installation is missing the necessary extension to create SVG files. In this case, head back to KNIME itself and add it:

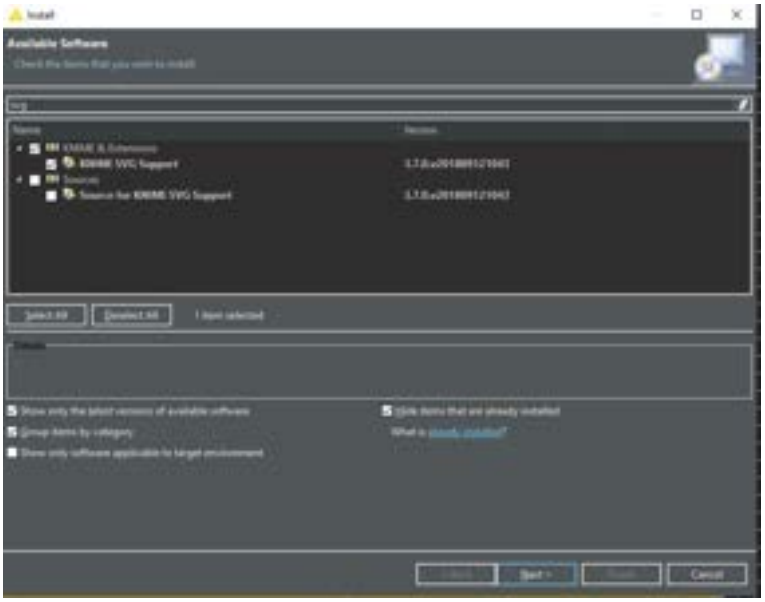


Figure 9: Installing SVG Support

Next, we load the data we want the analysis steps saved in the workflow to be applied to into the notebook. We therefore read a new excel file in (this time containing posts from the now forbidden right-wing part NPD) and turn it into a pandas dataframe called `df`. Calling `df` shows us that everything went well:

Figure 10: Data import

We can call the workflow on these data by using the `knime.workflow()` function with the same arguments as above, but this time saving the workflow into an object (we called ours `wf`) instead of printing it. Next, we set the `data_table_inputs` of this object to be our data frame (we have to use the index '[0]' since we have only one dataframe to offer but the function usually handles entire lists of data frames) and call its `execute()` function. This causes the KNIME Analytics Platform to run in the back and hand make the results available in Python as the `data_table_outputs` attribute of our wf object:

Figure 11: Executing the workflow

Now, we could use these topic classifications for further analysis in the notebook.

This looks quite cute and handy at first because you certainly feel more like a 'real' data scientist when you share something notebook-style. Yet, to me this seems a bit gimmicky and not actually useful because you need Python to run KNIME and the person you want to collaborate with also needs Python and the KNIME extension for Python set-up and running. Moreover, notebooks (no matter if jupyter, R notebooks etc.) were invented as a means of managing the source code for an analysis and the associated human-readable documentation in the same file. Since KNIME workflows themselves can be annotated nicely (and you can do Python analysis within them) and they mostly do not feature actual source code, they are 'literate', anyway. Thus, it's usually a better idea to annotate them thoroughly and share them right away instead of making things more complicated by wrapping everything with a notebook.

3 BIRT-Reports

The so-called BIRT reports, KNIME offers are much more useful in my opinion. They take the idea of data sharing one step further, in that they make the addressee of the report completely independent from KNIME: The KNIME Report Designer extension lets you take the results you produce with your workflow and incorporate them in a report template you can edit and style to your liking and export into multiple different file formats almost anyone can open (such as PDF or HTML).

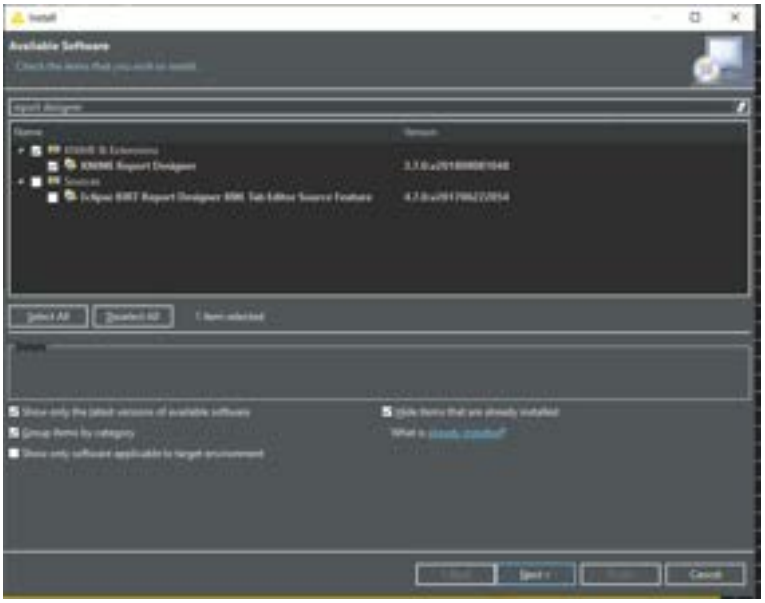


Figure 12: Installing KNIME Report Designer

Just like for the jupyter integration, we will have to add specific nodes to the workflow that make the output of our analyses available in the reporting environment. Particularly, we will have to incorporate **'Data to Report'** nodes for all tables we want to access later and **'Image to Report'** nodes for all graphics we want to appear in the report:

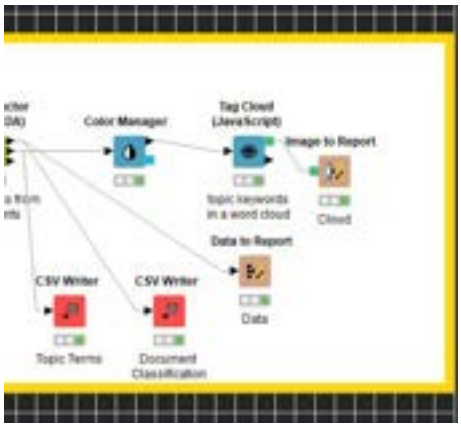


Figure 13: Workflow with reporting nodes

Once we have prepared our workflow accordingly, we can create a new report by selecting 'New' from the 'File' menu and choosing the 'Report' wizard:

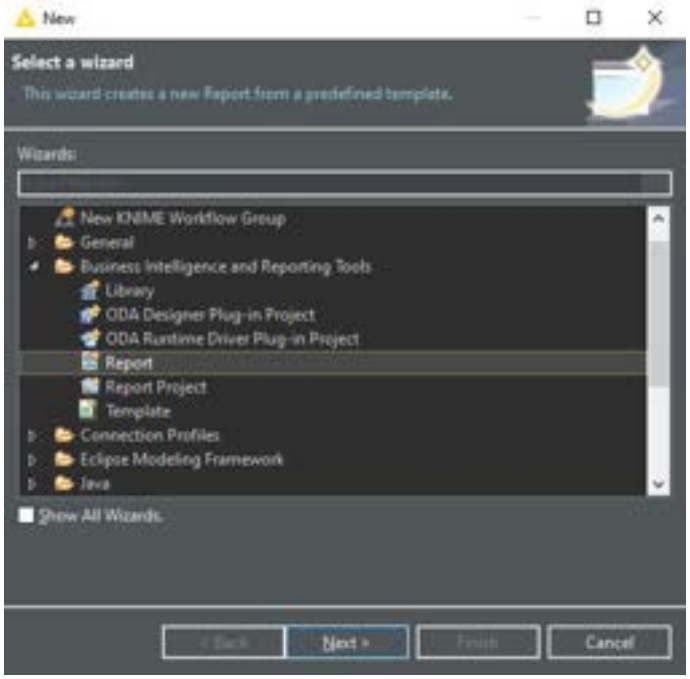


Figure 14: Creating a new report

This opens a completely new environment that looks quite different from the default KNIME GUI you are used to work with for your workflow creation: The only thing that remains the same is the KNIME explorer to navigate your file system. In place of your workflow canvas, you now have a **Property editor** as the main window that lets you style the content of your report along with a **preview** of this report on the right. On the bottom left you now have the **Palette** with quick tools and the selection of different content items to insert into your report. And right above, you have the **Data set view** that lets you access the data and graph objects you created with the reporting nodes you integrated into your original workflow.

Figure 15: The BIRT report GUI

If you are familiar with web design, editing your content for a BIRT report will feels quite natural to you. If not, the reporting may come with quite a bit of a learning curve because styling report elements is basically done with HTML tags

in KNIME. We will not deep-dive into this here, if you are hooked and want to find out more about content creation and styling for BIRT reports, please check out the associated webpage.

Figure 16: The HTML editor

When all content is created and all styling is done, you let KNIME build the report in the output format you desire. Therefore, you can either search for the little grey icon with the play button in the top panel or go to 'Run' and then 'View Report'.

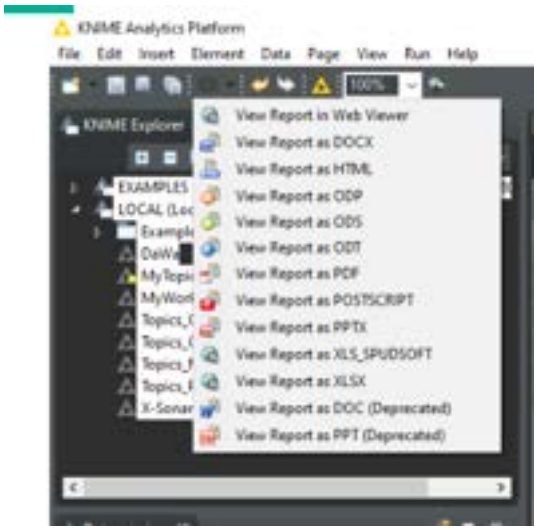


Figure 17: Selecting a report output format

This is much more convenient than the jupyter integration. Yet, I fear that as a 'pro' you will deeply miss an option to integrate BIRT reports with LaTeX. Also, I would guess that the user group of this extension is quite small: If you want to produce a webpage and are familiar with markup and scripting languages such as HTML, CSS, and JavaScript, you are likely to already have access to a more powerful and comprehensive IDE for web development and want to use this. And

if you want to create just an ordinary .pdf or .docx report and are not familiar with HTML etc. you will probably not want to spend the additional time to learn it and just copy'n'paste data tables and graphics into your documents as usual. So, BIRT reports are also not a feature that would make KNIME extraordinarily sexy for the presentation of results and collaborative projects.

4 Guided Analytics

However, KNIME indeed does have one feature that is super useful for collaboration and presentation alike and that is extremely well put together: KNIME offer support for so-called Guided Analytics. The term 'guided analytics' refers to the development of interactive visual interfaces for business intelligence and scientific applications. Those applications let other people (such as customers or collaborators) explore your data, extract information from it or even create their own reports from it in just the way you allow it /or assist them with: You guide them though a meaningful analysis of your data step by step via a web interface that you create:

Figure 18: An exemplar web interface offering Guided Analytics; souce: https://www.youtube.com/watch?v=ELqZ1NjQffg&t=12s

Those web interfaces are very similar to what you might already know from Shiny or Dash which can be applied to R or Python. The particularly nice thing for guided analytics with KNIME, however, is that, again, no programming skills are necessary for the creation of such like apps: You create them in a pretty straightforward manner by just adding another layer (and 'auditing' layer) to your analysis workflow that restricts and manages access to your data and data processing operations:

Figure 19: Exemplar workflow integrating Guided Analytics, source: https://www.youtube.com/watch?v=v2-f5eiM0mc

However, there is one big (not to say 'huge') disadvantage that comes with this great idea and is also why we will not explore guided analytics further: Guided analytics apps need to be deployed to the KNIME WebPortal which in turn is a feature of the commercial KNIME Server and thus extremely extensive (at least from a non-enterprise, private person's perspective).