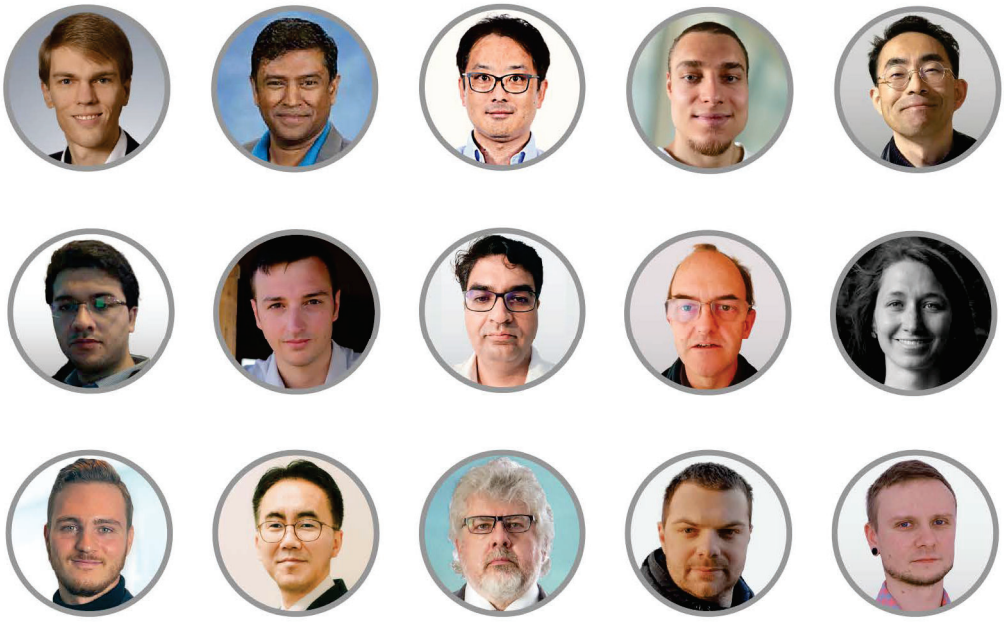
Use Case Collections

Edited by
ELISABETH **RICHTER**

Best of KNIME

The COTM Collection

August 2022 - July 2023



Open for Innovation
KNIME

KNIME Press
Talacker 50
8001 Zurich
Switzerland

knimepress@knime.com

www.knime.com

Best of KNIME goes into the 2nd Round

In November 2022, we published the first "Best of KNIME" booklet with the purpose to collect and highlight the top contributions of our KNIME Contributors of the Month from Season 1 & 2 (08/2020-07/2022). Why? Because we believe, if you want to learn about data science and KNIME, you should learn from the best!

Now, another year has passed and here we are with another set of KNIME COTMs. Season 3 has come to an end. Enough reasons for us to create a second volume of the "Best of KNIME" booklet series. The purpose of this booklet is unchanged: we again collected the top stories of our COTMs, this time from August 2022 until July 2023.

As usual for the KNIME COTM program, each month we select one (or sometimes multiple) members of the KNIME community that stood out amongst the crowd and awarded them for their excellent technical skills and for their contribution to the upskilling of the community in data science and KNIME.

During Season 3 we had 15 KNIME COTMs: One for each month, and in December 2022 we had 4 awardees at once - our *Just KNIME It!* top KNinjas. This season, we again feature all sorts of professions like Data Analysts, Data Architects, Consultants, Researchers & Scientists, Educators, Professors, and more…

In these past Season 3 of the KNIME COTM program we have seen how Arne Beckhaus' *Continental Nodes for KNIME* extension enhances data analytics, learned about 11 Digital Healthcare use cases from Dayanjan (Shanaka) Wijesinghe, followed Kazutaka Watari's *#100DaysOfData* journey, learned tips and tricks for spreadsheet handling from Daniel Weikert, let the four *Just KNIME It! – Season 1* KNinjas (aka Yasue Katsutaka, Emiliano Amendola, Raffaello Barri, and Anil Kumar Sharma) demonstrate how to solve *Just KNIME It!* Challenges like a pro, had Stephen Roughley shedding light on the historical perspective of the *Vernalis KNIME Nodes* extension, dove into the world of CADD with Dominique Sydow and into the world of Financial Investment Analysis with Cristian Rastasanu, got to know James Kim, the face behind Zalesia's upskilling journey, and learned various practical things like how to create a population pyramid using Python Plotly from David Plummer, how to process paginated API responses from Arjen Peters, and how to solve a multi-class classification problem using the *Conformal Prediction* extension from Artem Ryasik.

Let's not indulge longer into this introduction. Let's hear more from the KNIME best.

Elisabeth, Scott, Corey, and Rosaria

Table of Contents

Welcome to the KNIME Contributor of the Month Program

Each time we award a KNIME community member as the month's top contributor, we share such award cards like the one shown below on social media. So, you might have seen this card before. If you have never seen it or if you're still wondering what's behind the KNIME Contributor of the Month (COTM) program, no worries. Before starting with the actual content of the books – which is praising our KNIME COTMs of Season 3 – let's first make sure that everyone is on the same page and knows what a KNIME COTM is and how to become a COTM awardee.

The award card for our KNIME Contributor of the Month for February 2023 – Dominique Sydow. Each month we award one community member for their outstanding commitment, and each COTM receives a badge.

What is a KNIME COTM?

The KNIME COTM award is assigned to KNIME users who have shown excellent technical skills and have contributed to a better learning experience as educators, to faster and more exhaustive technical support, to knowledge sharing via articles, blogs, and YouTube videos, to a richer repository of community nodes and components, or to a stronger KNIME presence on social media.

In a nutshell, the COTM award is assigned in recognition of technical skills and contribution to the advancement of the KNIME community. Indeed, there are only a handful of such top experts around the world: this level of expertise and dedication is hard to find!

The program has started in August 2020 and is going strong ever since! With the nomination of Artem Ryasik in July 2023 we finished season 3 of the KNIME COTM

award program. Until then, we have awarded 40 KNIME users in 36 months. Why do we have more awardees than months you ask? Well, that's because sometimes we awarded multiple community members at once, for example, in January 2021 or in December 2022. But we're not stopping here: At the time of publication of this booklet, we are right in the middle of season 4.

How to become a COTM Awardee?

First of all, somebody – a fan, a colleague, a family member, yourself – must nominate you or any other KNIME user for a COTM award via the COTM Candidate Proposal form. In the description field of the form, the reasons why the nominee deserves to receive the COTM award must be stated.

Once a month all COTM nominations are evaluated. Based on their recent activities, their contribution to the KNIME community, and their technical skills, the best nominee is selected and awarded the title of COTM for the next month.

If, while reading, the name of a deserving KNIME user springs to mind, do not hesitate and nominate them for the COTM award program in the COTM Proposal form.

The list of all past COTMs for all seasons can be found in the KNIME COTM Hall of Fame.

KNIME in the Data Science Lab

In this section we have assembled all contributions that focus on KNIME Analytics Platform and its capabilities. The articles presented here reveal what KNIME has to offer and uncover the power of KNIME Analytics Platform. The category "KNIME in the Data Science Lab" features our eager Data Science experts:

- **Dr. Arne Beckhaus**

  – Head of Data Architecture Automotive *@Continental*

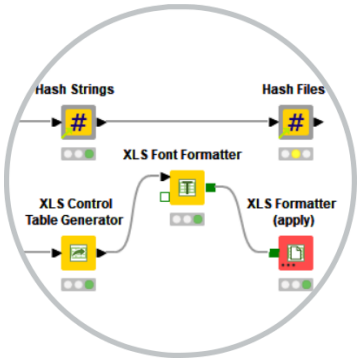
- **Cristian Rastasanu**

  – Data Analyst *@Mydral*

Dr. Arne Beckhaus was nominated KNIME Contributor of the Month for August 2022. He was awarded for building his Hash Files and Hash Strings components and for leading the development of the Continental Nodes for KNIME extension. Both components and the nodes of the extension add unique functionalities to KNIME, as they provide data processing and reporting capabilities intended for business users. Learn more at the *Community Components – Summer 2022 Collection* on and find example workflows in the *Continental Nodes Example Space* on the KNIME Community Hub.

Arne is an experienced data executive who has more than 10 years of experience in the Automotive industry and is currently Head of Data Architecture Automotive at Continental. In 2010, Arne obtained a doctorate from the University of Freiburg, and in 2017 he won the Digital Leader Award in the category of Empower People.

Visit Arne's space on the KNIME Community Hub or his profile page in the KNIME Forum (Hub/Forum handle: `arbe`).



Exploring KNIME's Continental Nodes

Enhancing Data Analytics with KNIME and Excel

Author: Victor Palacios

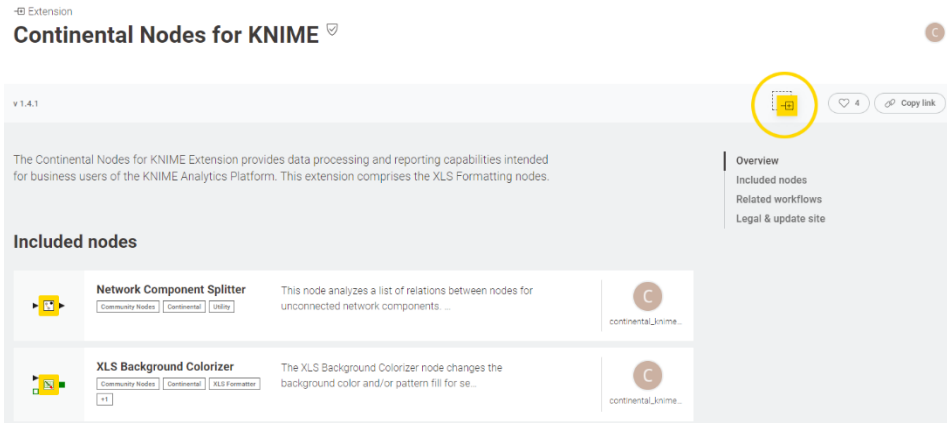
The Continental Nodes for KNIME extension, developed by Continental, provides data processing and reporting capabilities intended for business users of KNIME Analytics Platform. A large part of the extension's nodes are nodes for XLS formatting which come in handy when you want to style your table before exporting it from a KNIME table to an Excel file. For example, you can quickly and easily modify the spreadsheet's background color, add or remove borders, or format the columns' sizes. In this article, we will delve into the capabilities of KNIME's Continental nodes which are made for all your Excel needs.

How to get KNIME's Continental Nodes

The Continental Nodes for KNIME extension is one of our community extensions, developed by one of KNIME's partner companies – Continental. Our COTM, Arne Beckhaus, is leading the development of the extension.

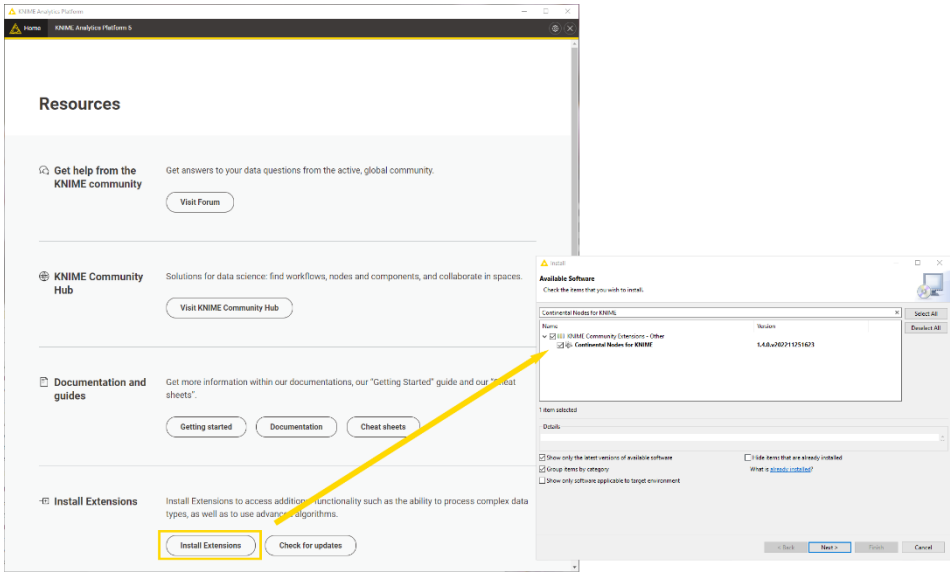
There are two ways how to install this extension:

1. Drag & Drop from KNIME Community Hub



Search the Continental Nodes for KNIME extension on the KNIME Community Hub and drag & drop the extension into your KNIME workbench. Then follow the installation steps shown.

2. Install from within KNIME Analytics Platform



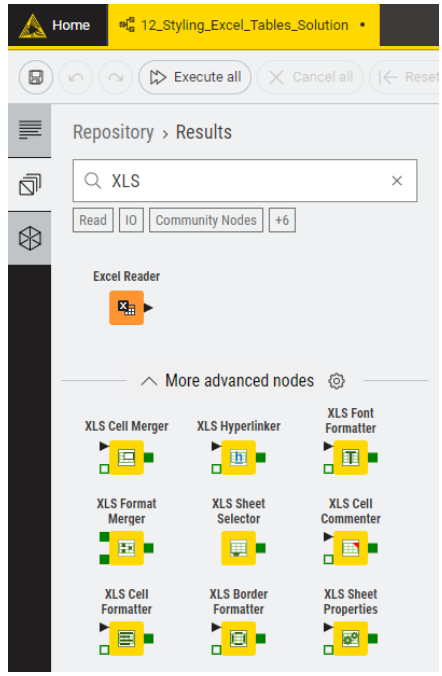
Inside the KNIME Analytics Platform go to the info page by clicking on the button in the top right corner. Scroll down to the "Install Extensions" panel and click "Install Extensions. Then search for the Continental Nodes for KNIME extension and follow the installation guide.

Once the extension has been installed, the nodes will be available in the Node Repository under "More advanced nodes".

These nodes facilitate Excel data import, export, and manipulation, empowering data lovers to seamlessly work with spreadsheets while taking advantage of KNIME's analytical power.

Styling Your Excel Table

So, what can the nodes be used for? Currently, there are over 15 Excel-related nodes in the Continental extension that allow users to change the font, color, and size of text, the background color, and the border of cells. Users can also add hyperlinks, comments, and formulas to cells. Let's have a look at a representative handful and how they can be used.



Continental nodes found in the KNIME Repository after extension installation.

Explaining some of the Nodes

Let's first list a couple of the nodes used in the following examples and explain, what they are doing.

XLS Background Colorizer

This node changes the background color and/or pattern fill for selected cells in an Excel file. Use this node to change the background color of cells to highlight important information.

Before

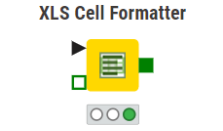
	A	B	C	D	E
1	Year	Quarter	Store - no CC	Store - with CC	OnlineStore
2	2019	1	36862.74	66775.81	114196.84
3	2019	2	38059.65	70483.79	113399.81
4	2019	3	48149.06	76791.58	96116.79
5	2019	4	47220.13	61563.41	105625.31

After

	A	B	C	D	E
1	Year	Quarter	Store - no CC	Store - with CC	OnlineStore
2	2019	1	36862.74	66775.81	114196.84
3	2019	2	38059.65	70483.79	113399.81
4	2019	3	48149.06	76791.58	96116.79
5	2019	4	47220.13	61563.41	105625.31

An Excel sheet without ("Before") and with ("After") color formatting using the XLS Background Colorizer node.

XLS Border Formatter

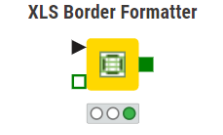


This node adds borders to selected cells in an Excel file. Use this node to add borders to cells to make them stand out.

XLS Cell Commenter

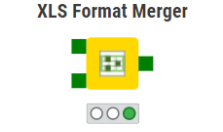
This node adds comments to selected cells in an Excel file.

XLS Cell Formatter



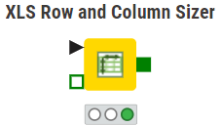
This node changes the formatting of selected cells in an Excel file, such as the font, font size, and alignment.

XLS Format Merger



This node merges the formatting of multiple Excel files. You can use the XLS Format Merger node to create a more compact layout.

XLS Row and Column Sizer



This node changes the row heights and column widths of an Excel file. You can use the XLS Row and Column Sizer node to improve the readability of the file.

Before

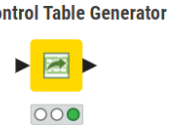
	A	B	C	D	E
1	Year	Quarter	Store - no C	Store - with	OnlineStore
2	2019	1	36862.74	66775.81	114196.84
3	2019	2	38059.65	70483.79	113399.81
4	2019	3	48149.06	76791.58	96116.79
5	2019	4	47220.13	61563.41	105625.31

After

	A	B	C	D	E
1	Year	Quarter	Store - no CC	Store - with CC	OnlineStore
2	2019	1	36862.74	66775.81	114196.84
3	2019	2	38059.65	70483.79	113399.81
4	2019	3	48149.06	76791.58	96116.79
5	2019	4	47220.13	61563.41	105625.31

An Excel sheet without ("Before") and with ("After") color formatting using the XLS Row and Column Sizer node.

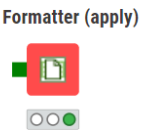
XLS Control Table Generator



This node takes an input data table and transforms the column names to A, B, C, ... and the row IDs to 1, 2, 3, ... just like they would appear when opening an Excel file in a spreadsheet application.

> ***Note.*** A control table, created by the XLS Control Table Generator node, is required to use all other Continental nodes that modify an Excel sheet if your tag table does not have appropriate column and row labels. We'll discuss the tag table later in this article.

XLS Formatter (apply)



The XLS Formatter (apply) node applies the commands of all Continental nodes to an unformatted xlsx file. The XLS Formatter also allows Chaining as well. Chaining refers to using various Continental nodes to make multiple changes in an Excel sheet. Chaining will be explained further in the next section.

> ***Note.*** An XLS Formatter (apply) node is required to make modifications to an Excel sheet. All other Excel Continental nodes give instructions while the XLS Formatter node executes those instructions.

Chaining: Using multiple Continental Nodes together

Now that we understand how each node can be used individually, let's chain a few together and observe the results. First, let's see what we are going to build and then explain the details in a before and after figure.

Before

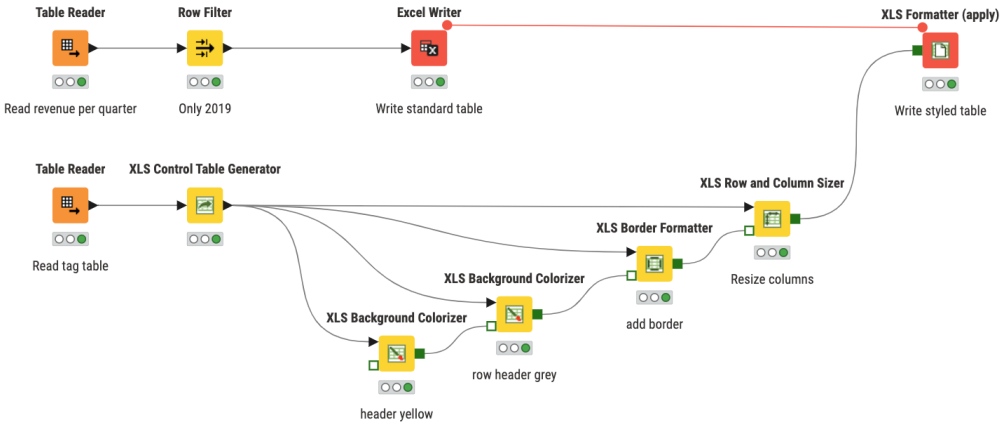
	A	B	C	D	E
1		Quarter	Store - no C	Store - with	OnlineStore
2	2019	1	36862.74	66775.81	114196.84
3	2019	2	38059.65	70483.79	113399.81
4	2019	3	48149.06	76791.58	96116.79
5	2019	4	47220.13	61563.41	105625.31

After

	A	B	C	D	E
1	Year	Quarter	Store - no CC	Store - with CC	OnlineStore
2	2019	1	36862.74	66775.81	114196.84
3	2019	2	38059.65	70483.79	113399.81
4	2019	3	48149.06	76791.58	96116.79
5	2019	4	47220.13	61563.41	105625.31

An Excel sheet without ("Before") and with ("After") various formatting changes using the KNIME Continental extension nodes.

We will go from the above before-and-after image using the following workflow below.



Workflow showing how to modify color, borders, and column size of an Excel sheet using the KNIME Continental extension nodes.

For the workflow shown above, we first use the *Table Reader* node to read in data using a KNIME native format called a table. Then we perform a basic data processing step, namely filtering the data to the year 2019 using the *Row Filter* node. Afterwards we write out the data in an Excel format using the *Excel Writer* node. This process generates the ***Before*** image above. Looking at the output, one may wish to add a little color or add borders or even make the text readable immediately upon opening the Excel file. Let's add each of these wish-list items to our output using some of the Continental nodes.

The Tag Table

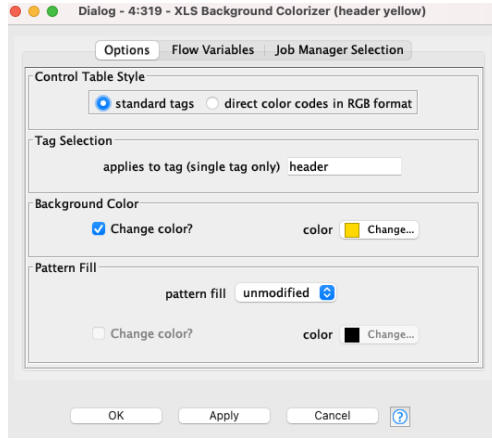
To begin, we need a table that has each and every cell labeled which can be done manually or programmatically. This table is called a *tag table* and is necessary for the Continental nodes to understand which cells you would like to target for coloring, expanding, and so forth. The tag table can appear as shown in the following image.

RowID	A *String*	B *String*	C *String*	D *String*	E *String*
1	header	header, right	header	header	header
2	row_header	row_header, right	value	value	value
3	row_header	row_header, right	value	value	value
4	row_header	row_header, right	value	value	value
5	row_header	row_header, right	value	value	value

A tag table indicating what each cell type is.

Notice that this tag table has RowIDs and column labels as we would expect to find in an Excel sheet. If your tag table does not have these row and column labels, use the *XLS Control Table Generator* node to adjust your tag table's row and column labels.

Next, let's change the background color of "header" cells, as labeled in the tag table. To do this, we drag and drop the *XLS Background Colorizer* node into our workflow, double click the node, and we are presented with a dialog that appears as shown in the figure on the right.

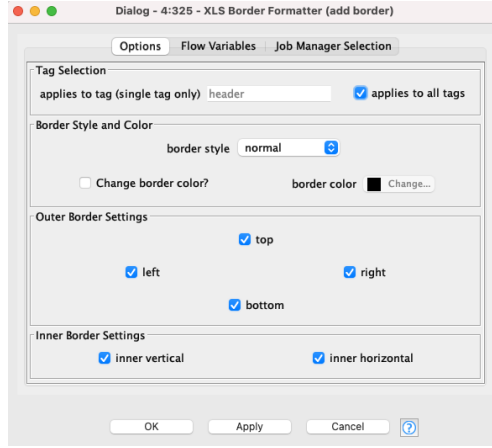


The XLS Background Colorizer node's dialog options.

Notice in the figure that there are several options to choose from. We will use the standard tags, apply them to the `header` tag, and change the background color to yellow. Likewise, we will do the same for the `row_header` tag, but instead we will change the color to gray.

Next, let's add bold borders to each cell for more appealing aesthetics. To do so, let's use the *XLS Border Formatter* node. Once we have the node in our workflow, we will open the dialog as shown in the figure below.

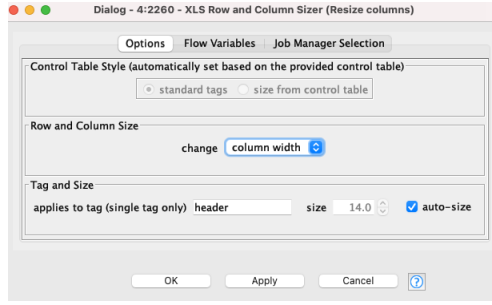
Instead of applying this node to certain tags from the tag table, we will apply these borders to all tags by ticking "applies to all tags" in the upper right corner. Then we can adjust the outer and inner borders as seen above. Pretty simple, right?



The XLS Border Formatter node's dialog options.

Finally, we will change the column and row sizes so that they are readable upon opening. Notice in the ***Before*** image above, that the cells contents are not immediately visible. Let's remedy that. As usual, navigate to the dialog box of the target node, in this case the *XLS Row and Column Sizer* node, as shown in the bottom right figure.

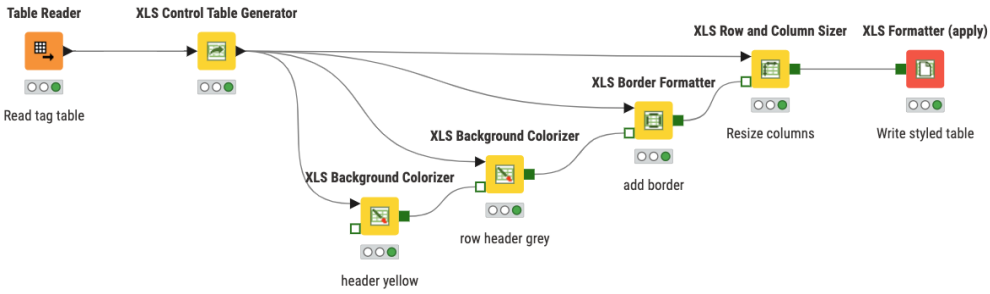
We will change the column width and we will not specify a distinct size, instead we will tick the auto-size option resulting in readable column labels when we open our Excel sheet.



The XLS Row and Column Sizer node's dialog options.

And with that, we have all the instructions we need for our chain. However, we must execute our chain using the *XLS Formatter*

(apply) node. Remember that all other nodes before the *XLS Formatter (apply)* node do not take effect without this node. The fruit of our effort is shown in the simplified figure below.



Chaining continental nodes together and executing the chain with the XLS Formatter (apply) node.

For even more advanced usage of the Continental nodes and chaining, check out the Logistics workflow found in the Continental Node's examples KNIME Community Hub page. This workflow shows how powerful KNIME and Excel can be with cell-level formatting, background color, and borders all occurring under different conditions.

Bonus: The Components

Our COTM of August 2022, Arne Beckhaus, also built the Hash Strings and Hash Files components! These nodes compute cryptographic hashes for strings and files, respectively. This can be useful for identifying content-wise duplicates, for example! Give them a try too if you fall in love with the Continental nodes like we did.

Summary

KNIME's Continental Nodes provide a powerful synergy between the KNIME Analytics Platform and Excel. With its comprehensive set of features, users can import, export, and manipulate data effortlessly, while taking advantage of KNIME's analytic capabilities. By embracing the Continental Nodes, analysts can unlock new dimensions of data exploration and make more informed decisions, driving innovation and success in their endeavors. If you are an Excel user, check out the Continental Nodes extension today.

Cristian Rastasanu was nominated KNIME Contributor of the Month for March 2023. He was awarded for building five verified components for financial analytics. The components offer data analysts popular functions in the finance industry: Net Present Value (NPV), Extended Net Present Value (XNPV), Internal Rate of Return (IRR), Extended Internal Rate of Return (XIRR), and Modified Internal Rate of Return (MIRR).

Cristian is a Data Analyst in data consulting, he currently works at Mydral, with a specialization in data visualization and data preparation. He enjoys exploring different ways of extracting insights from data and creating storytelling experiences in the form of dashboards. He also dedicates part of his time to data science and experimenting with KNIME (besides creating components, he also engages in the KNIME community for example by teaching how to automate financial calculations with KNIME). Cristian's background is in big data, computer science, and economics. He describes himself as being passionate for data and devoted to creating impact and making the world a better place.

Visit Cristian's space on the KNIME Hub or his profile page in the KNIME Forum (Hub/Forum handle: `crastasanu`).

Automating Financial Calculations with KNIME Components

Introducing NPV, XNPV, IRR, XIRR, and MIRR

Author: Cristian Rastasanu

The Need of Analytics in Finance & Financial Tools in Data

In the field of financial investments, making informed decisions is crucial. The problem is that these calculations must be done regularly, and they take a lot of time. As a result, most of the time (let's say 80%) is spent preparing the data and calculating the metrics, and only a small part of it is spent analyzing them and making decisions.

In order to be more efficient, we need to change this relationship and focus more on the business, which means dedicating valuable time and energy to what matters most and what can bring actual value to the job and self-fulfillment to yourself.

This is what our solution helps us to do: to spend less time on preparation and more on analysis.

On the other hand, there is perhaps a lack of financial tools at hand, and it can get hard sometimes for people to get into more advanced analytical tools like KNIME. As with any change, it is hard in the beginning to drop the comfortable manual spreadsheets, but if we make it easy for the financial world to implement these measures using only one component, perhaps they can discover how much easier it is to use KNIME in their everyday financial tasks.

From Messy Excel to Organized and Automatized Workflows

The traditional Excel calculations for metrics like *Net Present Value* (NPV) and *Internal Rate of Return* (IRR) can be time-consuming, tedious, and complex, especially when dealing with multiple projects. Imagine you have a project with the planned investments and cashflows on a spreadsheet. To measure its return, you would have to write a formula in a cell next to the data and select column by column the variables needed for the formula. Now imagine you want to do that for 5 other projects. What about 200 or 1000 projects? Well… It has become a bit more challenging now. It can work for one time if there are few projects, but to do actual analysis of portfolios and long-term investment projects at bigger scale, or even at smaller scale, it is way easier using KNIME, and extremely easy and fast using a component. It literally takes the same time to calculate the measure for one portfolio as it is for one thousand portfolios.

Dedicated Components for Financial Investments Analysis

Given what we know about what is needed, we wanted to develop something that will allow a smooth experience for everyone wanting to measure financial KPIs for big amounts of data. By doing so, we also wanted to show the power of components and by that bring the financial investment world closer to the public and getting out of the messiness of multiple data spreadsheets.

The five components replicate traditional Excel calculations. This Financial Analysis Category is designed for measuring financial KPIs for big amounts of data and getting out of the messiness of multiple data spreadsheets.

But first, what are these investment measures that have been developed?

There are two ways to find out whether an investment is worth it or not. Either by measuring the *Net Present Value* or by measuring the *Rate of Return*. There are various ways to determine these two values. Let's first define our five metrics.

The ***Net Present Value (NPV)*** describes the difference between the investment costs and its present value. It is used to determine whether an investment will be profitable or not. The formula for calculating the NPV is:

$$NPV = \sum \frac{C_t}{(1+r)^t} - C_0, \text{ where}$$

C_t *is the cash flow (return) in t*

r *is the interest rate in t*

t *is the time period*

C_0 *is the initial investment*

If the NPV returns a value larger than zero, the project will be profitable, if the NPV returns a value smaller than zero, the project won't make profit.

> ***Note.*** The NPV and variants of it can be approached in two different ways: either including the initial investment as a negative cash flow, or excluding it and treating it separately.

The ***Extended Net Present Value (XNPV)*** is used to calculate the NPV of an investment where the cash flows do not occur at regular intervals. For example, instead of once a year the cash flows occur at shorter, irregular intervals, like 31/01/2023, 28/02/2023,

31/03/2023, 28/04/2023, etc. To account for that, we add the date as a factor to the XNPV formula, such that:

$$XNPV = \sum \frac{C_t}{(1+r)^{(\frac{d_t - d_1}{365})}} - C_0, \text{ where}$$

C_t is the cash flow (return) in t

r is the interest rate in t

d_t is the date in t

d_1 is the date of the first cash flow

C_0 is the initial investment

The ***Internal Rate of Return*** is the rate at which the NPV of an investment equals zero. It is used to evaluate the potential profitability of a project. For example, if the IRR is greater than the required rate of return (RRR) or greater than the IRR of other potential investments, this project might be favored. Calculating the IRR is a bit more complex than calculating the NPV. In our NPV formula, r is the IRR. Hence, to calculate it, we would set the NPV equal zero and solve for r:

$$NPV = \sum \frac{C_t}{(1+r)^t} - C_0 = 0$$

Similar as to the XNPV, the ***Extended Internal Rate of Return*** (XIRR) is used to calculate the IRR for investments where the cash flows occur irregularly.

Lastly, the ***Modified Internal Rate of Return*** (MIRR) addresses some of the limitations of the classic IRR calculation and hence provides a more realistic measure of a project's profitability. The IRR assumes that cash flows are reinvested at the calculated IRR, however, this is not realistic as it sometimes might not always be feasible. In reality, cash flows are often reinvested at the cost of capital, which is not the same rate at which they were generated in the first place. To account for that, the MIRR incorporates a Reinvestment Rate (the firm's cost of capital of this investment) and a Finance Rate (the financial cost).

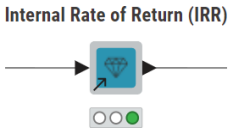
How to Create a Component for Financial Metrics?

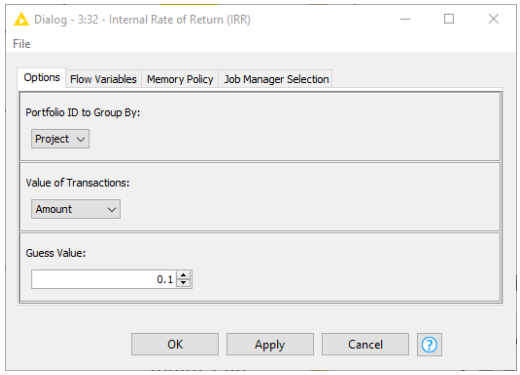
Since each formula is quite different from each other, there was the need to implement different approaches, some are more complex than others.

Regarding the configuration of the component on the other side, what I wanted to have as a final result was something that would be an easy plug-and-play solution where you just choose which columns to use for the portfolios, which are the cashflows, and any other information depending on the measure, so that you would end up with a result which is in form of a list of portfolios and the measure you want to calculate.

IRR

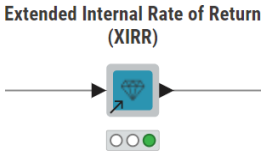
For the IRR we found out that the easiest way is to just do some coding in Java and implement an open-source library called *Apache POI* that calculates the IRR directly. You can actually open the component and check it out in detail.

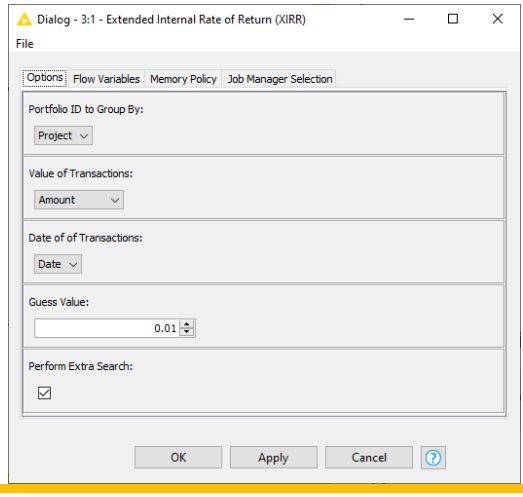




XIRR

XIRR, however, was more complicated to do. The problem was that it is a formula that starts from a guess value and some iteration to approximate the result, which was different when using different methods. The one way that ended up working was using another Java library developed by *De Campo*. But even like that, if the cashflow contained some abnormal values it would not converge to a result, this is why we added a "Perform extra search" check box which would activate a part of the workflow that loops, for the portfolios that can't converge initially, through different guessing values until it converges to a result.

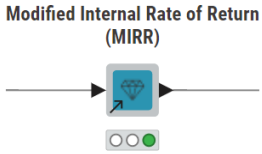


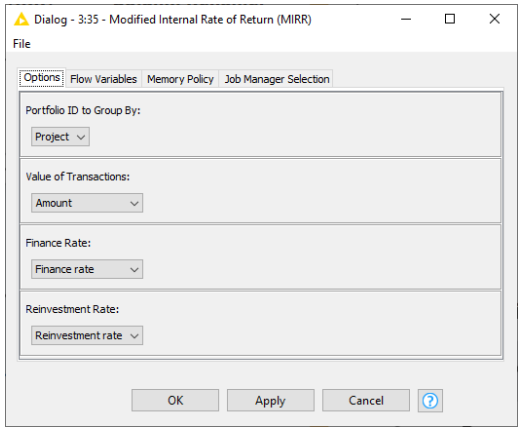


Note. When downloading the XIRR component, the Java library is also automatically downloaded to your machine.

MIRR

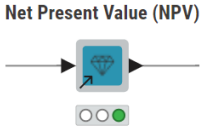
From MIRR on the challenge became easier. Since there was no iteration or approximation to do, here we just implemented the formula using native KNIME nodes. This is also since MIRR is using already enough information to get a result pretty easily.

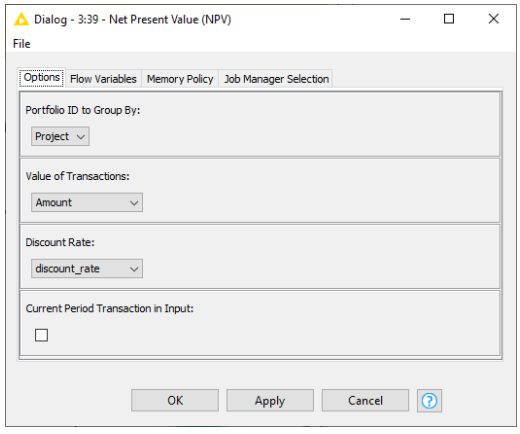




NPV

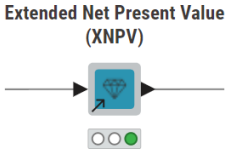
To what concerns the net present value, the first case is when we do not have any dates, which means that we need a column containing all the portfolios and one with the cashflows. Then, to calculate the NPV we also need the discount rate that can be provided in a column. As in Excel, we are given the choice of including the current period transaction in the input or not. Since the target is Excel users, I wanted the experience to be somehow similar.

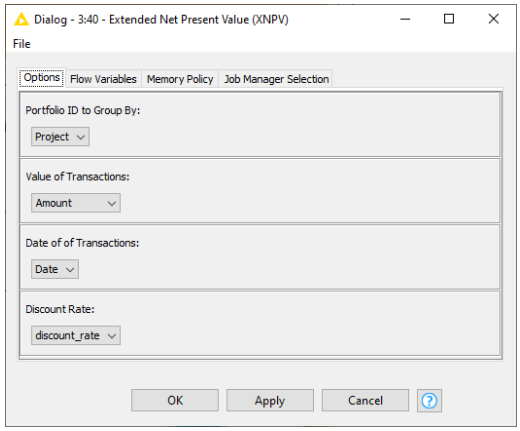




XNPV

Similarly to the NPV, and with the same logic as XIRR, the XNPV calculates the NPV taking into account the dates as well. This way it does not assume anymore that the cashflows happen at exactly the same interval but takes into account time and thus makes a better approximation of what could the net present value be.





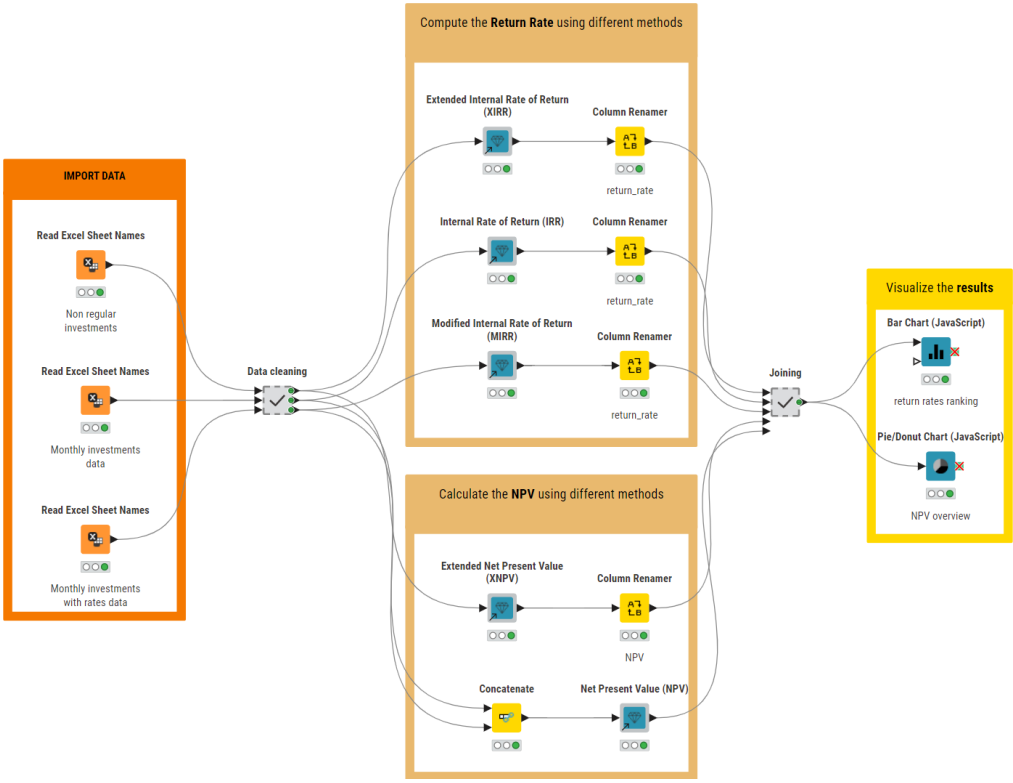
Using the Components

Putting them into practice is as easy as it can get. In the end, this is the reason that they were developed in the first place. You can find an example in the KNIME Community Hub of a demo workflow putting in practice all the components used in a fake life scenario. From my experience, which is actually the one that got the development started in the first place, big investment banking might have large datasets of portfolios that can be identified by an ID, put into one single column and have their XIRR calculated. After which it can be visualized in a dashboard and used to make investment decisions.

The example that I used is not one of a big bank, rather an unlikely personal project in which I would decide what kind of personal investments to do to reach financial sustainability (see figure below). The example workflow, Financial Metrics: Investments' return rate & value, is available for download from the KNIME Community Hub.

What Data is Needed?

The data is the same that you would need for calculating the same measures in Excel. What is essential to have in the first place is a column with the portfolio/project IDs or names, which is going to be used to group the different investment projects. Then a column of cashflows is needed, and it should always begin with a negative value as a first investment. Given these two columns you can already begin calculating some measures like the IRR. If you add a column with the discount rate or one with the date of the transactions then you can also compute the NPV, XIRR, or XNPV, and so on.



The example workflow, [Financial Metrics Example: Investments' return rate & value workflow](), is available for download from the KNIME Community Hub.

Conclusion

Data analysis should be simple and intuitive, and if we can spend more time on creating value and fulfilment for ourselves from the work, we enjoy doing we can change the world for the better. This is the main reason for developing these nodes, to make time in someone's day to do other things rather than manual and tedious data analysis and bring the financial field closer to everyone interested.

Good luck!

Data Science Use Cases

In this section we have collected all the articles that show how multifaceted KNIME Analytics Platform can be applied. Some of our COTMs created thrilling, fascinating, and creative use cases that demonstrate that KNIME is a tool for (almost) everything! KNIME can even be used to dive into the magical world of dwarves and dragons... The category “Data Science Use Cases” features our enthusiastic creators:

- **Dayanjan Wijesinghe**

  – Associate Professor *@Virginia Commonwealth University, School of Pharmacy*

- **David Plummer**

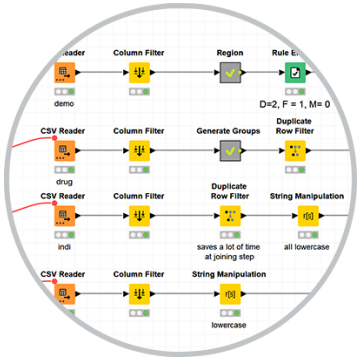
  – Senior Consultant *@Frazer-Nash Consultancy*

- **Artem Ryasik**

  – Advanced Analytics Engineer *@Redfield AB*

Dayanjan (Shanaka) Wijesinghe was nominated KNIME Contributor of the Month for September 2022. He was awarded for his effortless work to integrate KNIME into the Digital Healthcare field, including contributions on the KNIME Community Hub such as Hypothesis Testing on FAERS Data or Total Parental Nutrition Calculation Dashboard . Shanaka also (co-)authored many articles on the KNIME Blog, for example, Automating Pharmacokinetics Calculation in KNIME or Understand Polycystic Ovary Syndrome with KNIME, held a joint talk for the American Association of Pharmaceutical Scientists (AAPS), and is currently working on many other initiatives.

Shanaka is a biomedical research scientist with a primary focus on studying lipid signaling events in the onset and solution of inflammation. He holds a PhD in Biochemistry from the Virginia Commonwealth University, where he also currently works as Associate Professor.



Visit Shanaka's space on the KNIME Community Hub or his profile page in the KNIME Forum (Hub/Forum handle: `dayanjan`).

Data Science Solutions for Digital Healthcare

Authors: Dayanjan Wijesinghe & Martyna Pawletta

Healthcare, like medicine, has been in constant evolution since Hippocrates' times. The latest evolution step has been digitalization. Digitization of healthcare data has unified what has been isolated and siloed for a long time. It has opened the doors to a large number of potential applications with significant clinical impact!

Digital healthcare ranges from accessing and analyzing patient records to monitoring bio-signals, from reading and absorbing new concepts in specialist literature to innovative drug design, identification of novel combination treatments, and so much more.

The data analysis procedures are often run by non-domain experts such as data scientists. However, it's the physicians, surgeons, nurses, and pharmacists who understand the need for data driven solutions. Their domain knowledge makes them the best at drawing meaningful and actionable insights from such analysis. Training in computer programming and data science are not an integral part of their education. Ergo: Those who best understand the problems in healthcare and are keen to apply data driven solutions are the least likely to create an impact from the digitized data available at their fingertips.

Low-Code Enables Self-sufficient Domain Experts

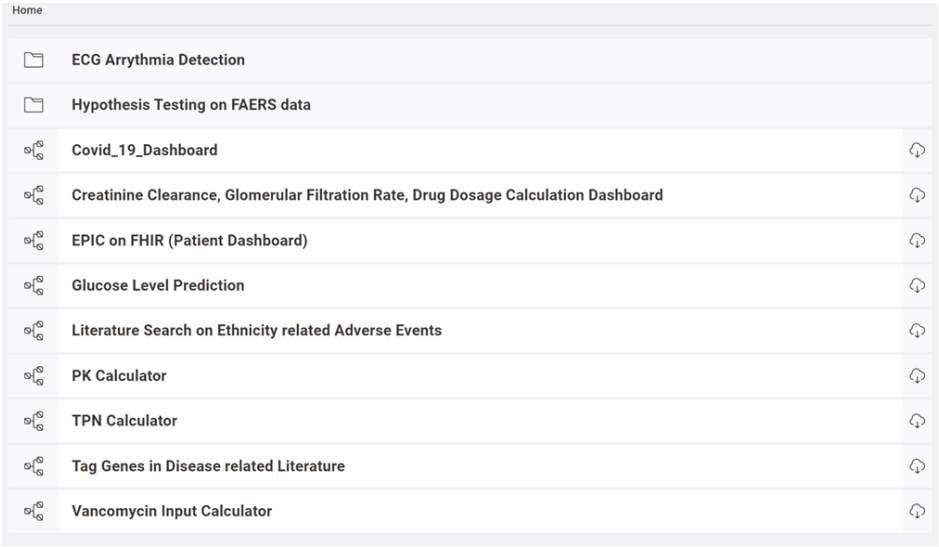
KNIME Analytics Platform, as a no-code/low-code data science tool removes these barriers and enables the healthcare domain experts to easily start creating and testing data driven solutions for their area of specialization.

Here, in this project we highlight eleven common digital healthcare use cases that can be solved using KNIME Analytics Platform. All these solutions are collected in the Digital Healthcare space on the KNIME Community Hub. They are free to download and adapt to your own requirements. In this article we describe the use cases, demonstrate the relevance of each workflow and the insights they bring.

We have created example workflows for:

1. Vancomycin dosing in obesity
2. Automating TPN calculation
3. Kidney Health Monitoring
4. Automating pharmacokinetics calculations
5. Accessing patient digital records via EPIC on FHIR

6. Visualizing disease tracking data using COVID-19 as an example
7. Predicting patient glucose levels with signal processing
8. Detecting ECG Arrhythmia with signal classification
9. Hypothesis testing based on the FAERS dataset
10. Disease tagging in literature articles using NLP techniques
11. Reviewing scientific literature reviews using ontologies, dictionaries and synonyms



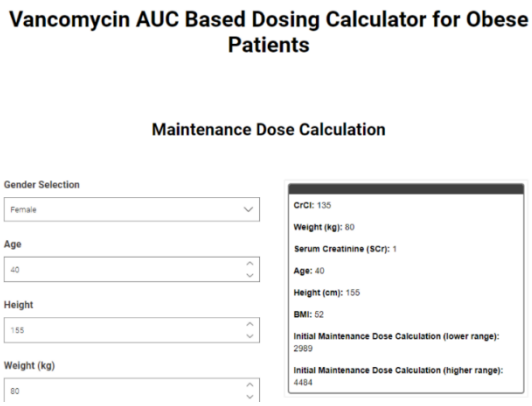
The public workflow repository for digital healthcare solutions on the KNIME Hub.

Note. This solution repository has been designed, implemented, and maintained by a mixed team of KNIME users from the KNIME offices in Germany and healthcare and data science research experts from the Virginia Commonwealth University, School of Pharmacy (VCU-SOP) in Richmond, Virginia (USA).

1. Calculate Vancomycin Dosing based on Latest Scientific Findings

The dosing of Vancomycin, an antibiotic used to treat serious bacterial infections, is a complex calculation that often can be time consuming in a clinical setting. Therefore, an automated solution, deployed as a data app and accessible via the web browser brings many advantages covering time constraints, dosing inaccuracies and also special population considerations.

With this example Danielle Holdren shows how KNIME Analytics Platform can be used to create a Vancomycin dosing calculator, based on latest scientific publications, keeping it fully flexible for changes and needs for future users.

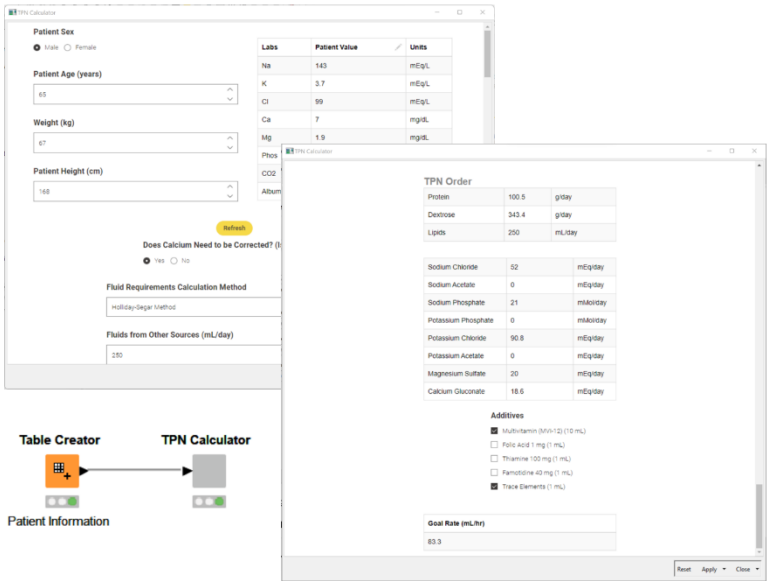


Interactive calculator data app for vancomycin dosing calculation.

Read the full article in the KNIME Blog: [How a Data App Improves Vancomycin Dosing in Obesity](#).

2. Automate Calculation of Total Parenteral Nutrition (TPN) for Results in Seconds

Total Parenteral Nutrition (TPN) is an important tool that ensures patients are provided with their daily nutritional needs. However, calculation of TPN isn't an easy task and contains many constraints that can lead to errors during the calculation. With this example workflow Courtney Ciarroca shows how KNIME can be used to develop an easy-to-use calculator that is capable of quickly calculating the TPN. It's an intuitive dashboard that allows the clinician to change values as they change during patients' treatment and get results within seconds.

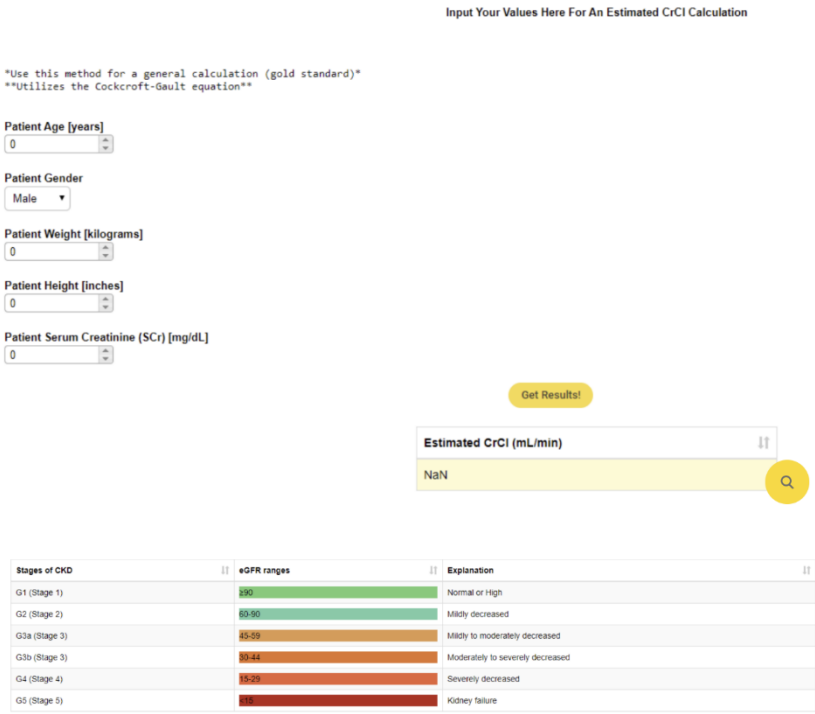


The TPN Calculator data app.

Read the full article in the KNIME Blog: [Automating TPN Calculation in KNIME for Quality Care](#).

3. Monitor Kidney Health with a Data App for Easier Calculation of Creatine Clearance

There are several values that are often used as a measurement to monitor how well our kidneys are working. Not all measurements are easy to calculate and are influenced by many different factors. With this example workflow Malik Graves shows how KNIME can be used to simplify the calculation of Creatinine Clearance via a data application that can be used in a clinical setting.



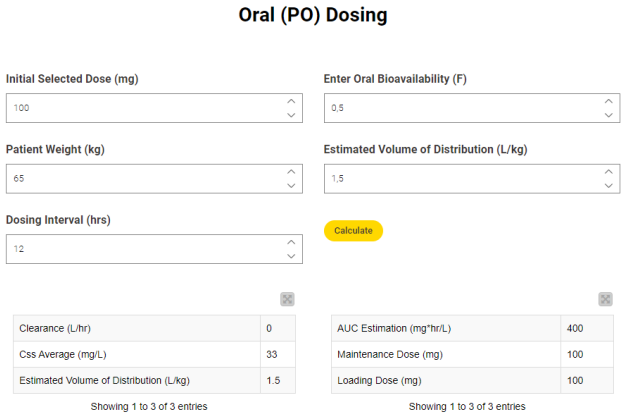
Data app for easier calculation of creatine clearance.

Read the full article in the KNIME Blog: Monitor Kidney Health in KNIME.

4. Automate Pharmacokinetics Calculation to save Time and Computing Power

Pharmacokinetics is a branch of pharmacology that provides insight on how the body responds to a drug. It encompasses processes such as **A**bsorption, **D**istribution, **M**etabolism, and **E**xcretion (ADME), which can be used to determine how efficient and how safe a drug is. Clinicians use pharmacokinetic parameters and calculations to visualize and interpret each of the phases of ADME as they monitor a drug's action in vivo. However, one of the challenges in pharmacokinetics (PK) is the time and computational power these calculations require.

Automating PL calculation in clinical practice can provide a more efficient process of drug monitoring with pharmacokinetic parameters. Danielle Holdren built a workflow for pharmacokinetic calculation and added an interactive dashboard which enables clinicians to quickly gain insight on how patients are responding to a drug.



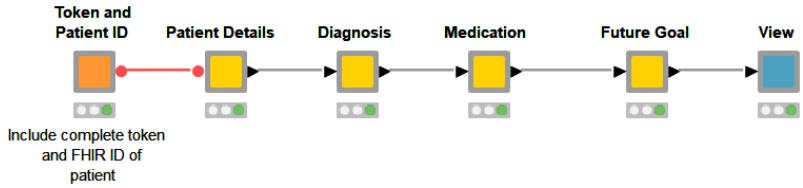
A section of the interactive dashboard in the pharmacokinetic calculator.

Read the full article in the KNIME Blog: Automating Pharmacokinetics Calculation in KNIME.

5. Access Patient Digital Records on FHIR

A majority of the US healthcare systems are moving to Epic as their primary electronic medical record platform. The ability to integrate a code free data analysis solution to such an EMR platform has potential for significant impact by allowing healthcare domain experts to test out new treatment approaches without having to learn how to code, using KNIME.

In a prior publication, Mateen *et al.*, demonstrated how to populate patient data to a FHIR server using the KNIME Analytics Platform and the POST method. Here we demonstrate how to extract FHIR patient data coming from Epic on FHIR using a GET Request. With our workflow we show what data extraction could look like, using the REST-based web service. Once the data is there, we display it in a very simple visualization.

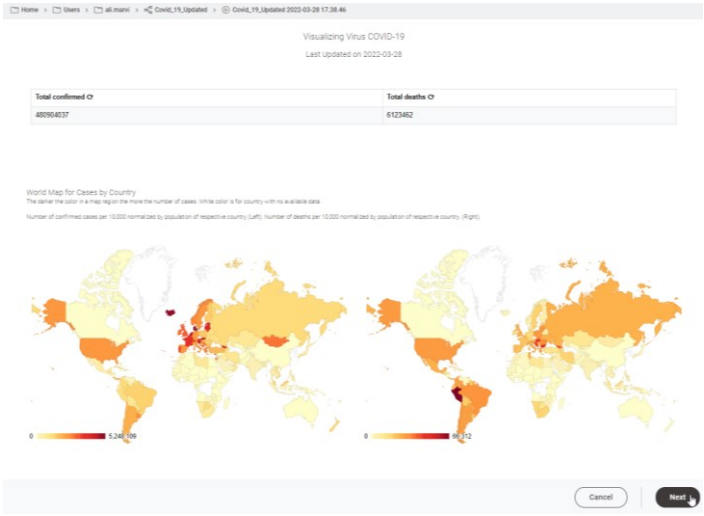


This example workflow consumes 4 different Epic on FHIR sources and displays the data in a view.

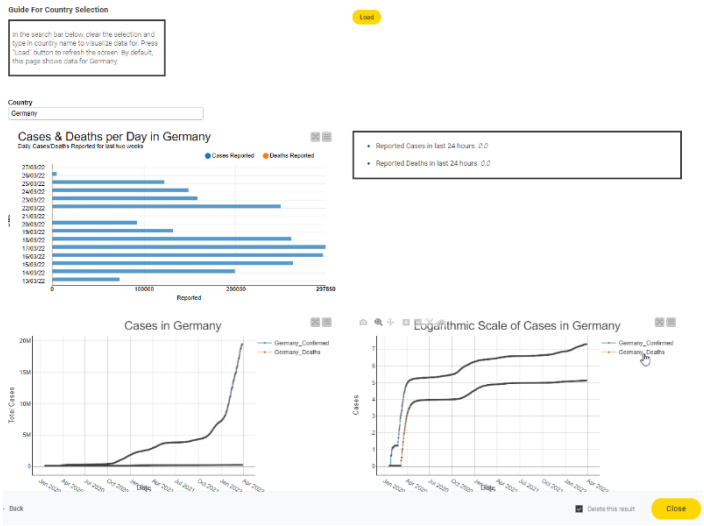
Read the full article in the KNIME Blog: Interact with EPIC on FHIR to Visualize Patient Data.

6. Visualize Disease Tracking Data in a Dashboard

The John Hopkins University tracks Covid19 data from sources from across the world. It introduced a coronavirus-tracker web service to query data from the entire world or from any specific country. With this short project we now show how to gather data from the web service they provide and also demonstrate how that data can be displayed as a dashboard.



Page 1 of the dashboard to track new cases of COVID19 globally around the world.



Page 1 of the dashboard to track new cases of COVID19 globally around the world.

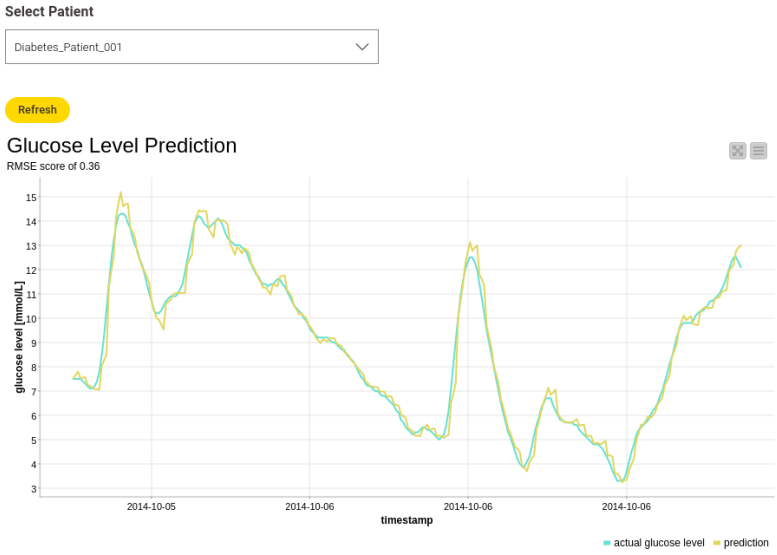
Read the full article in the KNIME Blog: [Track Disease with KNIME on COVID-19 Dashboard](#).

7. Predict Blood Sugar Levels from Glucose Monitoring

Continuous glucose monitoring (CGM) has been demonstrated to provide highly actionable insights for healthcare professionals to better manage diabetes among their patients with both type I and type II diabetes.

In this project we demonstrate how machine learning, Long Short-Term Network (LSTM) in particular, can be used to predict future blood glucose levels. Devices called continuous glucose monitors record blood sugar (glucose) levels at certain time intervals, transmitting the data to an app or software. This way, a user can see their current and past blood glucose levels. To predict future blood glucose levels an LSTM network has been trained on the sensor data. This makes it possible to recognize early on how blood sugar levels could change in response to changes such as medications, lifestyle or diet.

The fact that the person's own prior data is being used to make future predictions allows a highly personalized approach towards managing diabetes. The same approach can be applied to the management of other chronic conditions where near continuous monitoring is available. Examples include the improved management of blood pressure via LSTM predictive modeling of readings from continuous blood pressure monitoring.

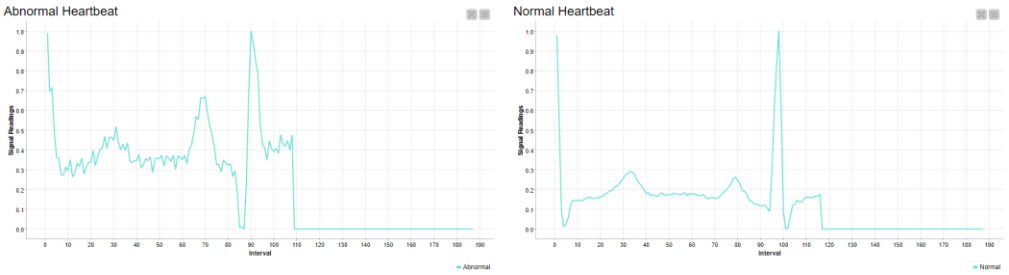


Line Plot showing the decrease in glucose level in a diabetic patient.

Read the full article in the KNIME Blog: Predict Blood Glucose with an LSTM on CGM data.

8. Detect Arrhythmia with Signal Classification on Electrocardiogram (ECG) Data

In this project, ECG data from Kaggle is used to train a deep learning model and classify ECG signals in KNIME. There are two parts of the project; the first part classifies ECG data into normal or abnormal readings and the second part is the multiclass classification for detecting Arrhythmia from a set of five different groups of readings.



Sample abnormal heartbeat (left) and normal heartbeat (right).

Read the full articles in the KNIME Blog: How to Perform ECG Categorization & Detect Arrhythmia & How to Classify ECG Signals with DL in KNIME.

9. Hypothesis Testing in Medicine with FAERS Data

Traditionally testing medical interventions always required clinical trials. However, with the large volumes of healthcare data becoming more publicly accessible, early hypotheses can be verified using a data driven approach prior to moving into clinical trials for validation.

Often accessing and analyzing these vast healthcare data repositories require a significant amount of coding knowledge. We asked the question, can such early data driven hypotheses be tested in large healthcare datasets using KNIME? Should this be the case, it will empower healthcare professionals to quickly test out possible outcomes of different treatments among specific patient populations and also to identify potential new treatment combinations.

KNIME based workflows are highly reproducible and once built can be deployed into multiple healthcare settings to assess the validity of such approaches. With this example workflow we show how to investigate adverse events submitted to the FDA from the FAERS database and try to answer the question if there are any medications that when administered with Irinotecan, has the potential to negate the onset of intestinal toxicity.

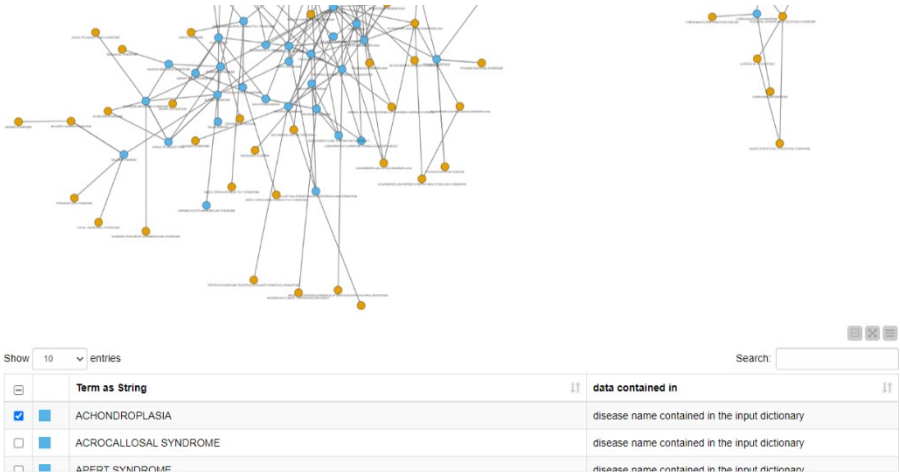
Cases of hypertension using Irinotecan without and with Loperamide.

Read the full article in the KNIME Blog: How to use FAERS for Adverse Reaction Management.

10. Disease Tagging to understand Impact of new Diseases

The pace of scientific discoveries is quickly exceeding our ability to digest those findings and apply them as innovative healthcare solutions. Computational approaches are sorely needed to find hidden insights. In this example, we demonstrate how to create a model that learns disease names in a set of documents from the biomedical literature. The authors demonstrate how to automatically extract abstracts from PubMed and use these documents to train a model with an initial list of disease names from a dictionary. Once they find new disease names, they can check their co-occurrence with other diseases in the extracted abstracts and create a co-occurrence network.

Insights gained from such information extraction have significant benefits in understanding the differential impacts of new diseases among patients with other co-occurring diseases. One current and relevant example would be to investigate how COVID-19 impacted different groups of people based on the co-occurrence of prior diseases.



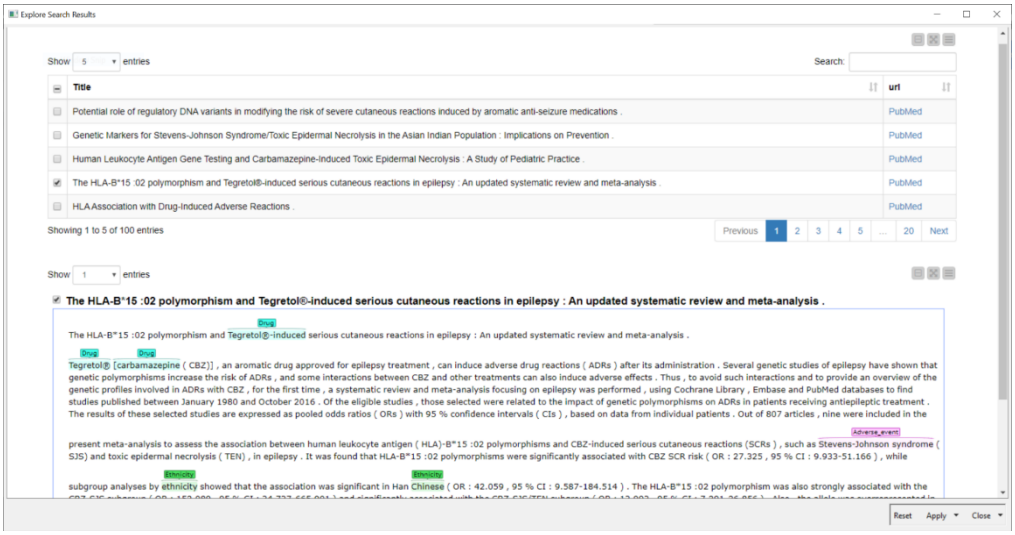
Network view of co-occuring disease names.

Read a walk-through of how to tag disease names in the KNIME Blog: Tagging Disease Names in Biomedical Literature.

11. Improve Identification of Key Information in Scientific Literature Reviews

Searching for relevant information within the large volumes of various scientific literature is not a simple task. We've built two example workflows to demonstrate how to improve the identification of relevant information via an interactive visualization. This approach helps to identify the right information with tagged terms directly in one place.

We've added two improvements: To improve the search we added terms from an ontology and synonyms to the search query. To improve coverage of the published literature we also used two different sources for biomedical literature simultaneously. We demonstrate this application of KNIME, investigating relationships between adverse events of a drug and the ethnic background of a patient to gain insights towards pharmacogenetic based precision medicine. The second example workflow maps genes identified in literature that are related to the disease being investigated.



This example workflow takes an adverse event and a drug name from the user and gets tagged publications that mention a relationship to any kind of ethnicity.

Read the full article in the KNME Blog: [Improve Literature Search & Minimize Information Overload](#).

Low-Code Tool Empowers Healthcare Experts

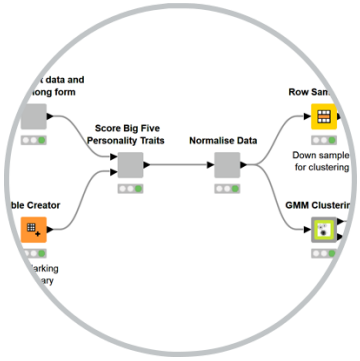
A mixed team of KNIME users from industry and academia has created, developed, and is maintaining low code solutions to access patient digital records via FHIR and EPIC, to design dashboards for disease tracking, for example for COVID19 data; to monitor glucose level or similar signals, to classify arrhythmia from an ECG signal, to perform hypothesis testing in medicine, and to tag diseases and investigate adverse events via NLP techniques.

These workflows are intended as example solutions to common problems in digital healthcare. Pharmacists, physicians, surgeons, dentists, nurses, biologists, and other domain expert figures can download them and customize them to their data and their business requirements.

This article was first published in our KNME Blog. Find the original version [here](#).

David Plummer was nominated KNIME Contributor of the Month for May 2023. He was awarded for being the prolific writer behind many excellent data stories that masterfully blend KNIME with Python for enhanced data visualization themes or cluster analysis. The early adoption of KNIME modern UI in his data stories is further testimony to this commitment to always deliver content in step with the times. He also provides great support and feedback on the KNIME Forum.

David has years of experience as a business consultant, providing analytics to the health and social care sector. His focus is on working with systems and organizations to identify key business issues, to lead workshops to explore problems and identify potential solutions, and to deliver analytics to provide insights and model future outcomes. He is currently Senior Consultant at Frazer-Nash Consultancy, a leading systems, engineering, and technology company. When he is not working with the KNIME software, David enjoys being a photographer and a writer of fiction.

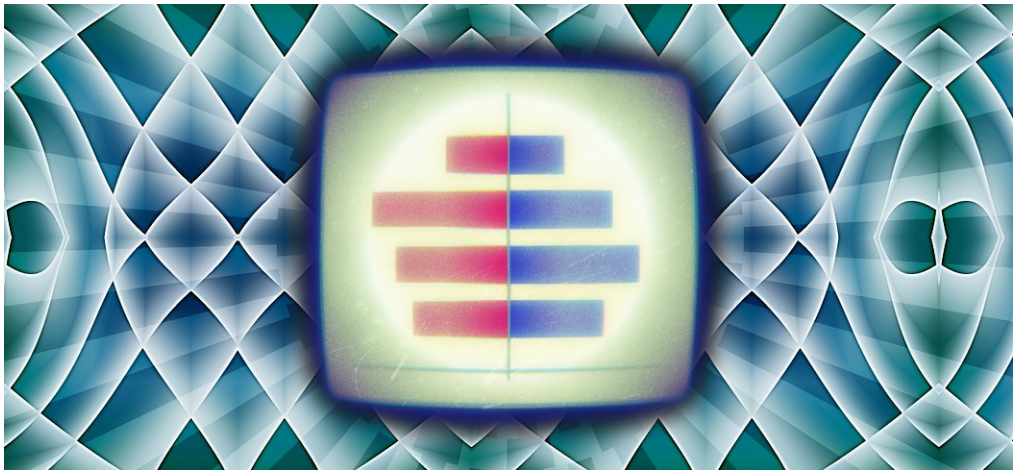


Visit David's space on the KNIME Hub or his profile page in the KNIME Forum (Hub/Forum handle: `diaazul`).

Awesome Visualizations in KNIME using Python Plotly

Author: David Plummer

How to create a configurable population pyramid visualization which can be re-distributed for use by data analysts, without having to train them in programming.



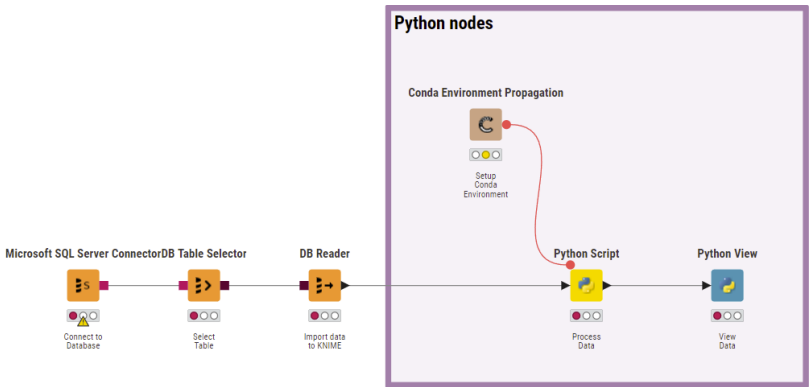
All images © David Plummer

KNIME is an easy-to-use no-code data science platform that allows non-programmers to bring together and analyze large quantities of disparate data. KNIME has, over many years, evolved to include new technologies that facilitate the integration of the latest data sources, tools and algorithms into the analyst's workflow. One of the more important developments is an emphasis on integrating Python into KNIME. This allows programmers to use Python code to create new functionality that can be redistributed for use by a much wider group of analysts, without having to train everyone to be an expert in coding.

Plotly is an open-source web based interactive visualization tool. At its core it uses JavaScript to display the visualization embedded in a webpage. Whilst the display of the visualization uses JavaScript, Plotly supports the creation of the visualizations in multiple languages, such as R, Python, Julia, and MATLAB.

This article is intended for KNIME developers and Python programmers who wish to develop Python Plotly visualization nodes. A second article will further extend the example to show how a consistent and updatable theme can be applied to multiple visualizations in the same application.

What's New?



KNIME v4.7 brings Python into the core product, including nodes to manage Python package dependencies, deploy Python scripts, and display Python generated visualizations.

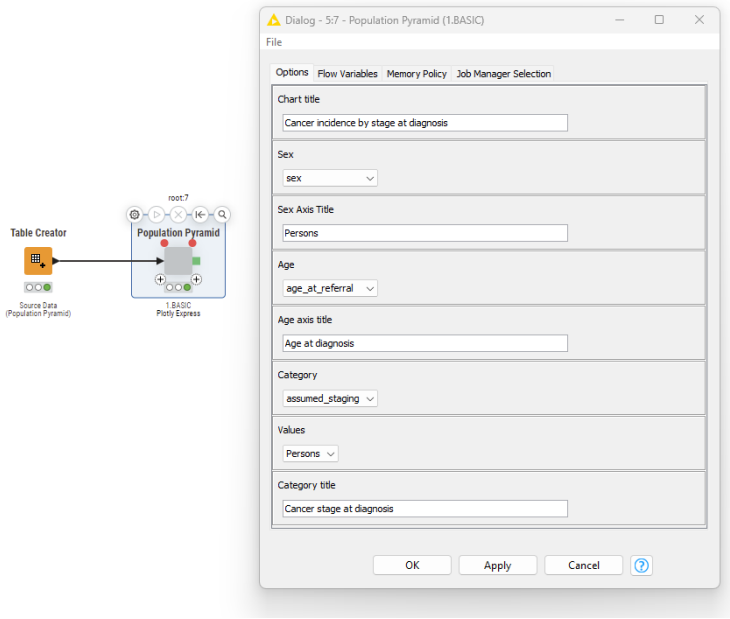
In KNIME version 4.7, Python moved from the KNIME Lab to become a core part of the product. This reflects the maturity of the Python ecosystem within KNIME, including the *Python Script* and *Python View* nodes, as well as tools to ensure package dependencies are installed when the nodes are deployed as part of a workflow. Performance of Python nodes is good with support for the same columnar back-end table storage and batch processing as the Java-based KNIME nodes. Version 4.7 also introduced a new *Python View* node which supports a variety of Python visualization packages including *Matplotlib*, *Plotly*, *PyVis* and *Shapely*, amongst others.

Population pyramid visualization generated in KNIME using Python Plotly.

KNIME Workflow

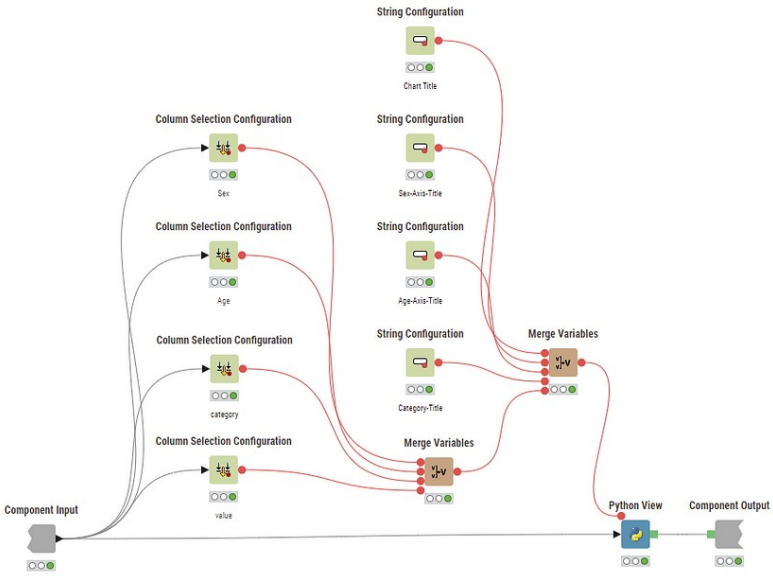
The example workflow can be downloaded [from the KNIME Community Hub] here, it requires KNIME version 4.7+.

The workflow contains top level nodes. A table, containing the example data, and the node which generates the population pyramid visualization. Double-clicking (or pressing the gear icon) on the population pyramid component opens the configuration dialogue, where the title and labels for items in the visualization can be entered and the columns containing chart data can be selected. To view the visualization, execute the workflow and open the interactive view.



Simple KNIME workflow with a configurable component to generate a custom visualization.

The workflow within the component is shown below. It comprises eight nodes which define the configuration dialogue, and which output the results of each field in the dialogue as KNIME variables. These are merged into the workflow and presented to the *Python View* node (updated in KNIME 4.7), along with the KNIME data table connected to the component input. The created visualization from the *Python View* node is connected to the component output which is configured as an Image port.

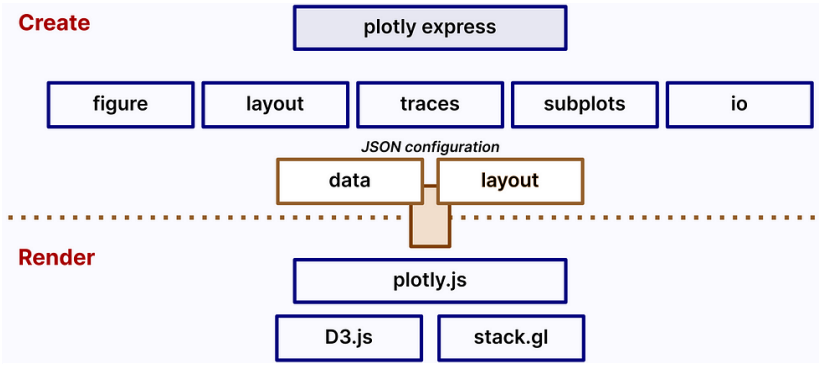


The Population Pyramid component contains KNIME nodes to configure the node and a Python View node to generate the visualization.

Python View

About Python Plotly

Before writing any code, it is worth having a basic understanding of how Plotly is organized. The rendering of visualizations in the web browser is executed by the `plotly.js` script which uses two well-known JavaScript libraries: `D3.js` and `stack.gl`. The `plotly.js` script determines what and how to produce the visualization based upon data passed to it in a JSON configuration file. This file contains two keys; the first, `data`, contains information for plotting each of the traces (data series), the second, `layout`, describes how the visualization should be displayed and formatted.



Visualizations created in Python Plotly are communicated to the Plotly JavaScript library as a JSON configuration file for rendering in a web browser.

It is not necessary to understand the JavaScript components, or the JSON configuration, as creation of visualizations is done using Python APIs. There are two APIs. A low level `graph_objects` which provides fine control over the creation of the figure (the class for the visualization), adding traces (the chart types and data series), the layout of the chart, the creation of sub-plots and various tools for importing and exporting data; and, a high level API, *Plotly Express*, which abstracts a lot of the detail so that visualizations can be created quickly and easily.

This example uses *Plotly Express* to create the population visualization. There is an example of creating the population pyramid using the `graph_objects` API in the downloadable KNIME workflow referenced above.

Reading Table Data and Configuration

Having provided some background to Python Plotly, the first piece of code is not Plotly related, but necessary to access the data in the KNIME table, and the values set in the configuration nodes. KNIME provides a `knime.scripting.io` library which provides methods for accessing both table and variable data, as well as convenience functions to convert table data into a Pandas data frame.

In the code below, the input table (on port one of the KNIME node) is read and converted to Pandas. Each of the flow variables (configuration items) are read and stored in Python variables for convenience (and to make later code more readable).

```
import knime.scripting.io as knio

# Read the input data table.
input_data = knio.input_tables[0].to_pandas()

# Read configuration from variables.
sex_column = knio.flow_variables['column-sex']
age_column = knio.flow_variables['column-age']
category_column = knio.flow_variables['column-category']
value_column = knio.flow_variables['column-values']
title = knio.flow_variables['string-chart-title']
sex_axis_title = knio.flow_variables['string-sex-axis-title']
age_axis_title = knio.flow_variables['string-age-axis-title']
category_title = knio.flow_variables['string-category-title']
```

This code snippet accesses the data in the KNIME table, and the values set in the configuration nodes. The input data is read and converted into a Pandas data frame, the flow variables are stored in Python variables for convenience.

Setting up the Visualization

Having imported all of the required data from KNIME into the Pandas script, the next task is to process the data so that it can be plotted.

A population pyramid is a bar chart where the vertical axis is in the center, with one category (female) is plotted to the left of the vertical axis, and the other category (male) is plotted on the right. The vertical axis represents age categories, the horizontal axis

is numeric (people), and the population may have several categories of data which are displayed as stacked bars (e.g., in this data set, cancer stage at diagnosis).

To create this visualization in Plotly we need to:

1. Negate the values (x-axis values, "persons") for the first sex category. These negative values will be plotted to the left of the x-axis, positive values will be plotted to the right.

   Note. The bar-chart will be plotted with a horizontal orientation, so that the x-axis will be vertical and the y-axis (values) will be horizontal. The negated values are stored in the DataFrame as `chart_values`.

2. Relabel the horizontal axis, add labels for the sex category, and make sure that the range of values (minimum and maximum) on the left and right are the same.

```
import pandas as pd
from math import log10, floor

# Invert first category values
# A population pyramid is a bar chart in which one category is plotted
# with negative values.
sex_categories = input_data[sex_column].unique().tolist()
inverted_category = sex_categories[0]

input_data["invert"] = input_data[sex_column]
                    .apply(lambda x: -1 if x == inverted_category else 1)

input_data["chart_values"] = input_data[value_column] * input_data["invert"]

# Ticks on the chart
# Re-label the x-axis to show the first category with positive
# instead of negative values.
number_ticks = 3
largest_value = input_data \
                .groupby(by=[sex_column, age_column]) \
                .sum() \
                .max()[value_column]
step = round(
        largest_value / number_ticks,
        -int(floor(log10(abs(largest_value / number_ticks))))
        )
tick_values = [tick * step
               for tick in range(-number_ticks, number_ticks + 1)
               ]
tick_labels = [
    (
        f"{int(abs(tick) * step):,}<br>"
        + ((sex_categories[0]) if (tick == -number_ticks) else '')
        + ((sex_categories[1]) if (tick == number_ticks) else '')
    )
    for tick in range(-number_ticks, number_ticks + 1)
    ]
```

This code snippet processes the data so that it can be plotted.

The `largest_value`, which defines the extent of the value-axis, is calculated by groups the sex/age categories summing across the sub-categories and then returning the maximum value in the `value_column`.

The incremental value associated with each tick is calculated by dividing the largest value by the required number of ticks on each side of the value-axis. The step value is rounded so that it is a nice-looking number (with one significant digit).

Tick values are created using list comprehension to create a list of values for the value-axis between the `-largest_value` to `+largest_value`. This scales the axis when plotting values.

As we do not want to show the left axis as negative values, we also need to create tick labels. The labels are absolute (positive only) values. For the first and last tick additional text is added to the label to indicate whether that extent is the first or second sex category.

Note. Plotly allows text to contain html formatting tags. To show the sex category label below the value a line break `<br>` is added to the text string.

Creating and Formatting the Visualization

We now have all the information required to create the visualization: The values of the first sex category have been negated and the tick-values and tick-labels created so that we have a symmetric chart with informative labels.

The convention is to import the *Plotly Express* library as `px`. A bar chart is then created by calling the `bar` method with the following arguments:

- The first argument, a positional argument, is the DataFrame containing the data for the visualization.
- `y` is the name of the column in the DataFrame that contains the y-values. In this case we are using the value of the age_column variable, which comes from the column selection configuration dialogue in the KNIME component.
- `x` is the name of the column containing the negated values calculated in the previous section.
- `color` is the name of the category columns, again from the configuration information. Each sub-category (cancer staging) in the chart will be plotted with a different color.
- `orientation`. The x-axis in a population pyramid (the age column) is plotted vertically, therefore, a horizontal `h` bar chart is required.
- `labels`. The default is to label the axis with the column name in the Pandas DataFrame. However, we want to use the labels that the user enters in the component's configuration. Using the `labels` argument, we can create a dictionary where the key is the column name in the Pandas DataFrame and the value is the text we want to show. For instance, the column in the Pandas

DataFrame is the label with the value in age_column and the text we want to show instead is contained in age_axis_title.

This is sufficient to create the basic Plotly figure, however, we still need to update the layout to change the tick labels and a few other defaults to complete the visualization.

```
import plotly.express as px

# Create the basic visualisation
fig = px.bar(
    input_data,
    y=age_column,
    x="chart_values",
    color=category_column,
    orientation='h',
    labels = {
        age_column: age_axis_title,
        "chart_values": sex_axis_title,
        category_column: category_title
    }
)

# Update the layout with the correct xaxis
fig.update_layout(
    title = title,
    barmode = 'relative',
    bargap = 0.0,
    bargroupgap = 0,
    xaxis = dict(
        tickvals = tick_values,
        ticktext = tick_labels,
        range = [-largest_value * 1.1, largest_value * 1.1],
        separatethousands = True,
        showgrid = True,
        zeroline = True,
        zerolinewidth = 3,
        ),
)
```

This code snippet creates the basic Plotly figure with `px.bar` and updates the layout with `update_layout`.

The layout and format of a Plotly visualization is determined by the layout values in the JSON configuration file passed to the `plotly.js` component. If the configuration file does not include a value for a particular visualization attribute, the Plotly uses a default value. Therefore, only changes from the defaults need to be defined when creating the visualization. This is done with a call to the `update_layout()` method on the figure.

The following changes are made to the layout defaults:

- `title` is the title of the chart, as entered by the user in the configuration.
- `Barmode` is set to “relative” which stacks bars above zero for positive values and below zero for negative values.
- `bargap`, `bargroupgap` removes the spacing from between the bars.

The x-axis attribute has sub-attributes which are defined in a dictionary:

- `tickvals` set to the tick_values calculated previously. This determines at which value the tick labels will appear.

- `ticktext`. The label that will be shown at the tick values specified above, tick_labels was calculated previously.
- `range` is the range of values shown when the visualization is first drawn. In this case the visualization is framed at 10% larger than the largest value.
- `separatethousands`. Tick values are shown with a separator between the thousands.
- `showgrid` shows grid lines for the x-axis.
- `zeroline` shows the zero line for the x-axis.
- `zerolinewidth` sets a thicker zero line for the x-axis.

These are just the relevant options used to generate this visualization, there are many more documented within the Plotly reference documentation.

Showing and Exporting the Visualization

The *Python View* node can present the visualization either in an interactive view window, or as a static image on the output port. Plotly charts are best shown within an interactive window where the user has the benefit of tooltips, pan & zoom, and downloading the visualization as an image file.

The interactive view of the visualization is created using the KNIME function `knio.view(fig)` which is then assigned to the interactive view window `knio.output_view`.

To export the visualization as an image on the *Python View* node output port `knio.output_images[0]`, the visualization needs to be converted to an image file using the Plotly function `pio.to_image` with the following parameters.

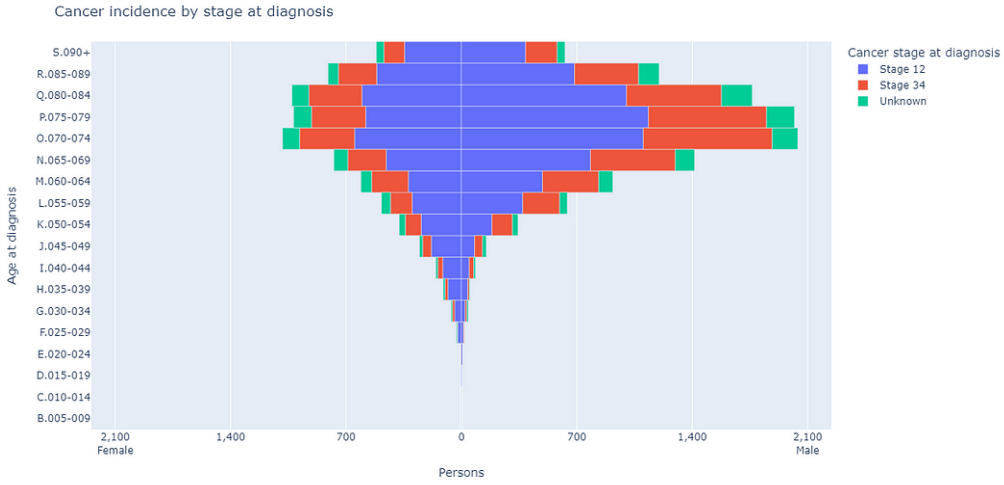
- `fig`. The Plotly visualization to export.
- `format`. The format of the exported image. Supported formats include png, jpg, webp, svg, and pdf.
- `width`, `height`. The width and height of the exported image.

```
# Assign the figure to the output_view variable
knio.output_view = knio.view(fig)


# Update image dimensions for output port
knio.output_images[0] = pio.to_image(
                          fig,
                          format="png",
                          width = 1200,
                          height = 600
                        )
```

This code snippet displays the figure (`knio.view(fig)`) and exports it as an image (`pio.to_image`).

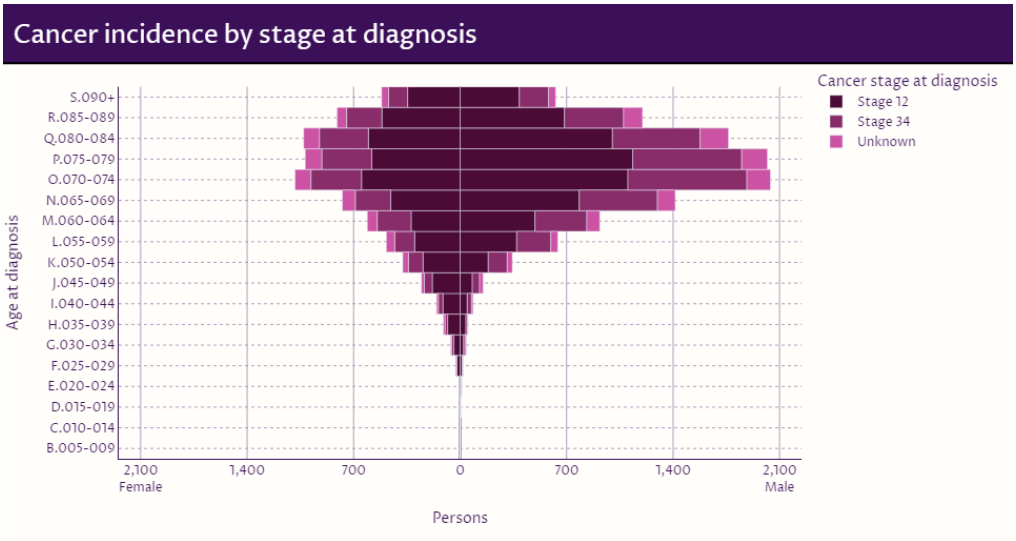
That completes the Python script which generates the following visualization at the output of the KNIME View node.



Population pyramid visualizations output from the Python View node.

Make Visualizations even more Awesome

The next article, "Great Themes For KNIME Visualizations Using Python Plotly", will discuss how to create Plotly themes and a configurable KNIME node so that theme changes can be propagated to multiple visuals. The visualization below uses the same data and *Plotly Express* to create the basic chart but uses a theme, shared between multiple *Python View* nodes, to customize the visual appearance.



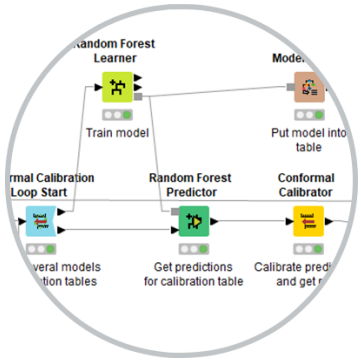
Population pyramid implemented in KNIME using Python Plotly and with a custom theme applied.

This article was originally published by David Plummer on Medium. Find the original version here.

The corresponding "Basic Plotly Component 2022-11-29 PA1" workflow can be found on the KNIME Community Hub.

Artem Ryasik was nominated KNIME Contributor of the Month for July 2023. He was awarded for his many excellent data stories that range from low code data anonymization to clustering techniques for analyzing knowledge graphs. His latest contribution shed light on conformal prediction, providing both theoretical insights and practical applications for classification and regression tasks. Recently, Artem also became a KNIME Certified Trainer, upskilling data professionals both inside and outside the organization. Read Artem's data stories here.

Artem has years of experience working as Researcher and Engineer and is currently employed as Advanced Analytics Engineer at Redfield AB, a company providing Business Intelligence consultancy and no-code solutions to help clients make data-driven decisions. Artem's background is in Physics: he holds a Master of Science in Theoretical Physics and a PhD in Biophysics.



Visit Artem's space on the KNIME Hub or his profile page in the KNIME Forum (Hub/Forum handle: `artem`).

Conformal Prediction for Classification

A Hands-On Codeless Example with KNIME

Author: Artem Ryasik

Editor's Note:

Artem wrote a preceding article to this one introducing the theory behind conformal prediction. Read the article "Conformal prediction theory explained" in our Low Code for Advanced Data Science Journal on Medium.

A fancy image to draw attention produced by Midjourney that tried to visualize the conformal prediction.

In this article, we are going to jump into a KNIME workflow to solve a multi-class classification problem using the nodes of the *Conformal Prediction* extension. We are also going to discuss how to estimate the uncertainty of predictions. Finally, we will wrap up the pipeline with the nodes of the *Integrated Deployment* extension, so the workflow will be ready for production with a couple of clicks.

Note. Read about "Conformal prediction for regression" in our Low Code for Advanced Data Science Journal on Medium.

The Data

The data we are going to work with is a data set describing different physical parameters of beans (available on Kaggle). The data set contains 13k records with 16 numerical columns, describing 7 classes, which are not well-balanced — this is actually a good test for conformal prediction.

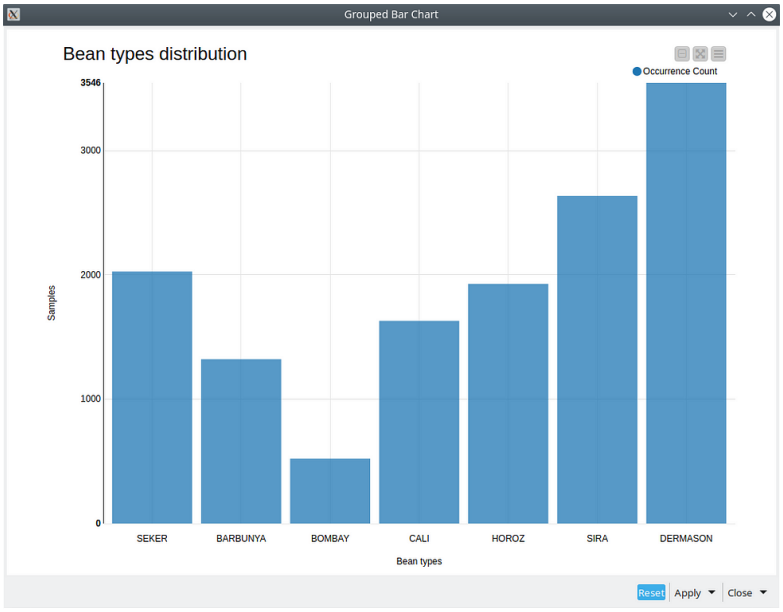
The Workflow

Here we shall start with the description of an "advanced" method that includes training multiple models and getting calibration tables for them.

Those who are impatient can directly jump to the KNIME Community Hub and download the Conformal prediction classification workflow. The implementation ("advanced") described hereafter corresponds to the upper branch of the workflow.

Data Preprocessing

As with any ML routine, we need to first read the data, do preliminary analysis, clean it, and only after this, we can move to training the models and interpreting the results. I have already converted the data into KNIME native table format and included it inside the workflow. After reading the data with the *Table Reader* node, let's have a look at the class distribution (see figure below) which is produced with the *Bar Chart* node. We can see that the classes are not balanced. However, the situation is not that dramatic, so we do not need to resort to technique for class-imbalance handling (e.g., under-sampling, over-sampling, etc.).

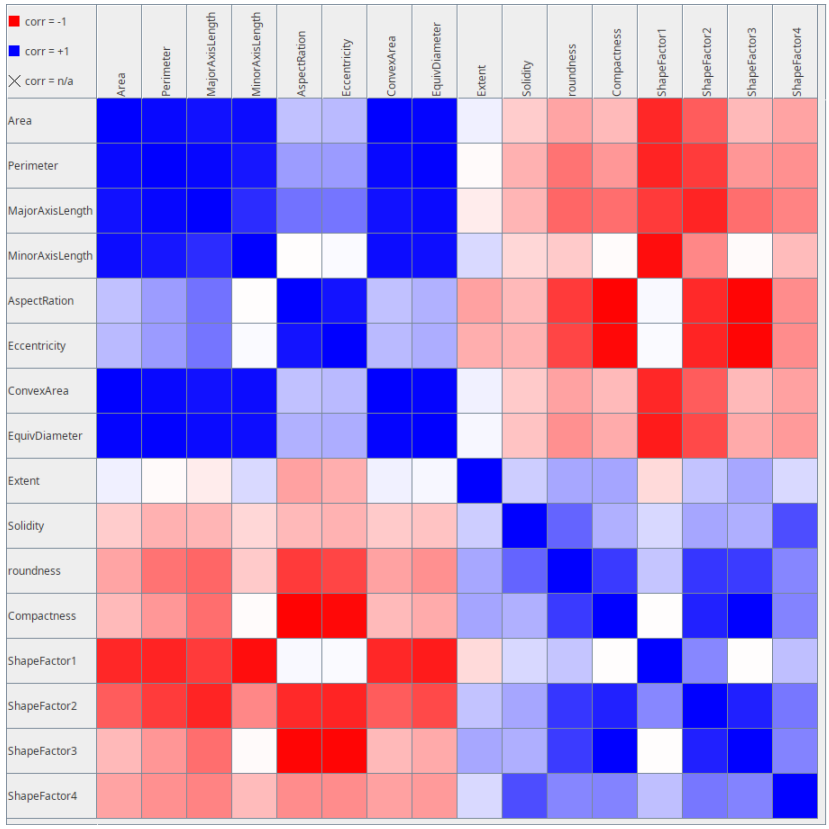


Bean types distribution.

Now, let's split the data set into two parts: the control data (10%) and the rest (90%), which is going to be used for training, calibration, and intermediate testing. The first data set is only going to be used for prediction and estimation once we finish with the whole conformal classification pipeline. This is done to simulate production data.

The next quite common step is data normalization, and in this case, it is definitely needed as the values we have are of a different scale. So, we are going to apply simple min-max normalization in the range [0, 1] since we do not have any negative values, and this is done with the *Normalizer* node.

Now, it is time to start capturing parts of the workflow using the nodes of the *KNIME Integrated Deployment* extension. These nodes will help us naturally build the final pipeline for production (you can learn more about Integrated Deployment here). The first step is to capture preprocessing operations, such as the application of the normalization model that we created before capturing, and we will later provide as one of the inputs of the *Workflow Executor* nodes.



Heatmap of features correlation matrix.

We also include a correlation filter that will reduce the number of highly correlated features. Indeed, we have a lot of them — see figure above! I set up the threshold for the correlation coefficient to 0.8, and it helped me reduce the number of features from 16 to 6, which should be both beneficial in terms of future predictions robustness and

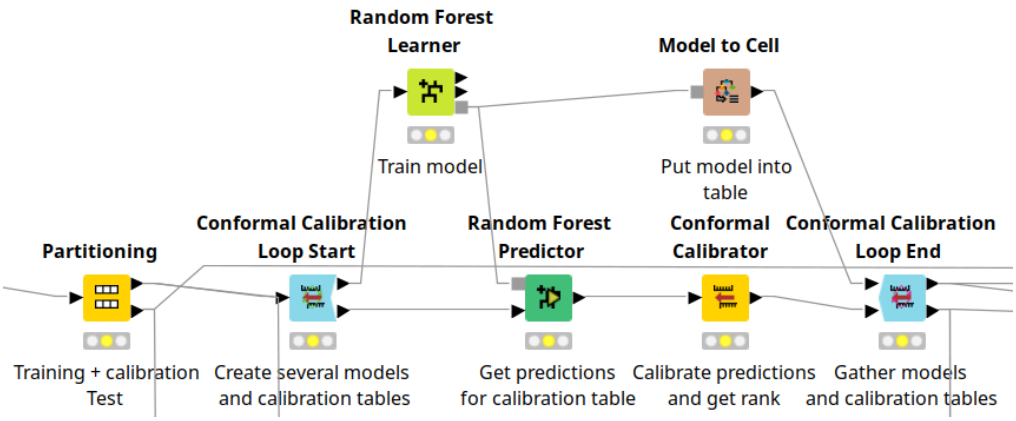
training time reduction. Here, we are closing the first capture part — this is the only preprocessing we are going to use.

Training and Calibration

We are now ready to proceed with model training and calibration. In this example, we are going to use the Random Forest algorithm, but readers are free to use any other classification algorithm. We first split the data into training + calibration and test data sets. Then, we start the calibration loop using the *Conformal Calibration Loop Start* node that helps us split the first data set into training and calibration data sets. The settings are almost identical to the *Partitioning* node, with the addition that we need to specify how many model and calibration table pairs we would like to get by tweaking the parameter "Number of iterations". In both cases, we need to use stratified sampling over the class column.

The construct within the loop block is the standard Learner + Predictor but now we are adding the calibration step that includes the creation of calibration tables based on the class probabilities and known class values. The *Conformal Calibrator* node takes as input the model predictions and produces an extra column – Rank – that will be used for calculating p-values at the next step. Within this loop, we also need to serialize the model as a cell object and put it into a KNIME table with the *Model to Cell* node. This is needed to propagate the model to the next step, but it also helps to be agnostic about the model we use. With the same procedure you can use any other model available in KNIME, or you can even serialize a model in Python – for example, from scikit-learn or *TensorFlow* libraries.

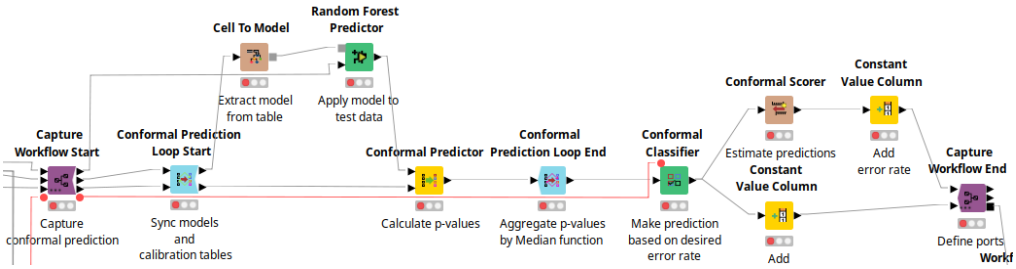
Finally, the *Conformal Calibration Loop End* node just synchronizes the models and calibration tables together by adding the iteration column for each of the tables, so they share the same iteration number. The described conformal calibration loop block can be seen in the figure below.



The conformal calibration loop block – creating the pairs of models and calibration tables.

Conformal Prediction

Finally, the next step is conformal prediction, and here we need another type of loop. This loop is intended to be used in production, so we are going to nest it between Capture nodes too (see figure below).



The conformal prediction loop block nested between Capture nodes. Here we use model predictions on the test set, then these results are fed to the Conformal Predictor node to calculate p-values. At the end of the loop, the p-values are aggregated by computing the median. Finally, the user provides the error rate in the Conformal Classifier node to get the prediction sets.

The *Conformal Prediction Loop Start* node starts iterating over consequent pairs of models and calibration tables. We feed the trained model and test set (or unlabeled data set) to a *Predictor* node in the same way we usually do in our ML routines. Then, as an additional step, we feed in the model predictions to the *Conformal Predictor* node to obtain the p-values. This process is repeated for all model + calibration table pairs we have gathered previously. Eventually, the p-values are aggregated by computing the median in the *Conformal Prediction Loop End* node. This operation helps make the predictions more accurate and robust because we are not relying just on a single model and calibration table, but on many. After the loop, we can finally get **conformal** predictions with the *Conformal Classifier* node. In the settings of this node, the user should provide the desired error rate and select the preferred format of prediction representation – array or string, so the output of the node will contain the column with prediction sets.

Conformal Prediction Quality Metrics

The main metrics one can use to estimate quality of conformal prediction are:

- **Efficiency** — the ratio of single label classification (right or wrong). It is calculated as $\frac{Single\ class\ predictions}{Total\ amount\ of\ predictions}$. In plain English this means: how many predictions we got that have just a single prediction in the set. Efficiency shows how simple it was for the model to classify a particular class or if we take the total — for all the classes.
- **Validity** — counts the fraction of correct predictions. If a record belongs to a mixed class containing the correct value, it is considered to be correct. It is calculated as $\frac{Total\_match}{Total\ amount\ of\ predictions}$. This shows whether the model did good at

getting the right class even if the prediction set contains more than one prediction. Validity basically shows the ratio of correct predictions even if they are not so clear.

There are some more metrics available and you can read about them in the description of the *Conformal Scorer* node or in this paper. If you tick the box "Additional efficiency metrics" in the settings of the Conformal Scorer node, you can get them as well. In this article, we are going to operate with efficiency and validity.

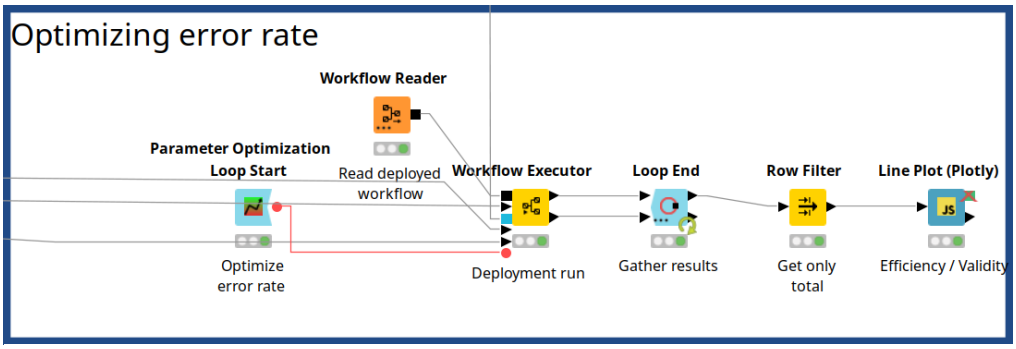
At this point, the selection of the error rate can be reduced to a fairly simple optimization problem. It is computationally cheap since we do not need to train the models. We simply bin the samples according to their p-values for the given error rate value. This is pretty much it.

Finally, we need to combine all the workflow parts together and save the production workflow with the *Workflow Combiner* and *Workflow Writer* nodes. Then, the workflow can be executed with the Workflow Executor node.

Optimizing Error Rate

Once the workflow is ready and automatically deployed, let's take a look at how it can be utilized in KNIME. First of all, we need to decide what is the best error rate value to use in order to get the best predictions. To do this, we need to read the freshly deployed workflow with the *Workflow Reader* node, and then provide the *Workflow Executor* node with all the workflow parts that we were gathering and developing as inputs:

- The unlabeled or test data set;
- Normalization model;
- The table with trained models;
- The calibration tables;
- The error rate value as a flow variable.

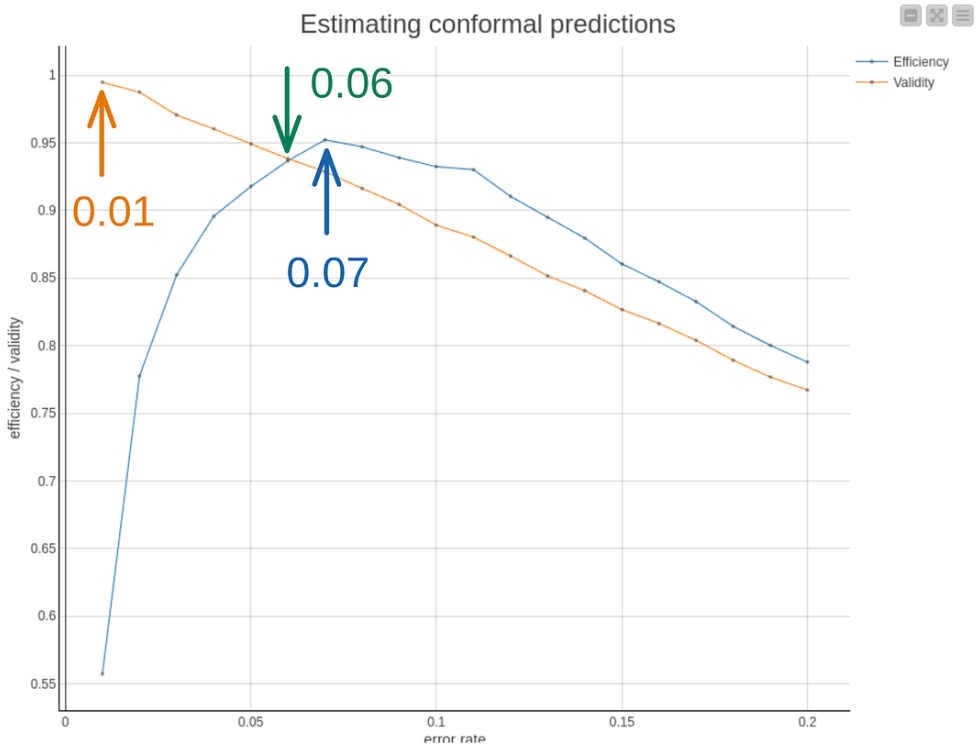


Running the deployed workflow to find the best error rate value. Read the workflow, and provide a new data set, the normalization model, the tables with trained models and calibration tables and error rate value.

Since we are going to optimize the error rate, we will put the *Workflow Executor* node inside an optimization loop, which is set to iterate over the error rate values in the interval [0.01; 0.2] with a step = 0.01 (see figure above).

After the loop is executed, we are interested in the total prediction, so we filter out the metrics for individual classes and finally plot the efficiency and validity values (see figure below).

This plot will help us select the best error rate, but users should ask themselves: "what predictions are we looking for?". In case one doesn't want to miss any correct prediction even if it is a part of a set with other values, then smaller error rate values are preferable since the validity will be the highest. In our case, it corresponds to the error rate = 0.01 (shown on the plot in orange). On the other hand, if one looks for the biggest amount of single predictions then it is better to stick to the values where the efficiency curve has the highest value — in our case, the error rate = 0.07 (shown on the plot in blue). Another criterion could be an optimum between validity and efficiency, so the predictions will be balanced in terms of both metrics. The last case refers to the error rate = 0.06 (shown on the plot in green).



The line plot shows the efficiency (blue) and validity (orange) values as a function of the error rate. Highlighted values on the plot describe the error rate values that are selected and proposed as the best values for different cases.

Note. Please consider that when you run the workflow, you might get slightly different results for predictions, optimal values of error rate, etc. In general, however, results should be reasonably close to those presented in this article.

Conformal Predictions Analysis

I prefer to select the error rate value that balances both metrics (error rate = 0.06). Once the error rate is selected, we can take the control data set we saved in the beginning and feed it into the *Workflow Executor* node. The upper output port of the *Workflow Executor* node returns the output of the *Conformal Scorer* node, and the lower output port returns the table with predictions. Let's take a look at the *Conformal Scorer* node output (see figure below). In this table, we see the validity and efficiency values for each class. Another useful metrics are the number of matches:

- Exact match – shows the number of single class predictions that are correct;
- Soft match – shows the number of the predictions that contain a correct class, but it is mixed with the other classes;
- Error – the number of predictions that have a set that does not contain the correct class;
- Null predictions – the number of empty prediction sets; nulls are always considered as errors.

Row ID	Target	Efficiency	Validity	Exact match	Soft match	Total match	Error	Total	Single class predictions	Null predictions	error_rate
Row0	SIRA	0.943	0.947	235	15	250	14	264	249	0	0.06
Row1	CALI	0.982	0.969	156	2	158	5	163	160	0	0.06
Row2	SEKER	0.98	0.931	187	2	189	14	203	199	1	0.06
Row3	BOMBAY	0.942	0.942	49	0	49	3	52	49	3	0.06
Row4	BARBUNYA	0.97	0.894	116	2	118	14	132	128	1	0.06
Row5	DERMASON	0.915	0.941	305	29	334	21	355	325	1	0.06
Row6	HOROZ	0.974	0.969	182	5	187	6	193	188	0	0.06
Row7	<Total>	0.953	0.943	1230	55	1285	77	1362	1298	6	0.06

The output of the Conformal Scorer node. The table shows the metrics for conformal predictions per class and total.

Now let's pick some insights from this table:

- Class Bombay has 3 errors and all of them belong to an empty set prediction or Null predictions, despite being a minority class. It seems that this type of bean is quite distinct, so it is not mixed with other classes. This may also explain why we got Null predictions, for these 3 samples had anomalous feature values for this type of bean.
- The largest amount of Soft Match values we got was for Dermason (even if we take the ratio of Soft Match compared to Total). This indicates that this class was mostly mixed with others. Probably, as long as this is the majority class, it is more likely to have some kind of average parameters of the bean. Hence, it can be easily misclassified.

- The Cali bean seems to be the easiest to classify since it has the highest exact match rate and the smallest error ratio.

These three insights can be useful in many ways. For example, to understand how the predictions can be interpreted, how the feature importance can be estimated, which bean classes are usually mixed together, etc. Elaborating further on how these insights can be useful is, however, out of the scope of this article.

"Simple" Case

The workflow also contains a second branch that basically does the same and also includes the nodes for Integrated Deployment. Therefore, once you run both of them, you can compare the workflows. The reason why the second branch is called "simple" is that this workflow does not contain any loops, which means that only one model is trained and only one calibration table is produced. Basically, in this case, the *Conformal Classification* node combines the functionality of the *Conformal Calibrator* node and *Conformal Predictor* node. The inputs of the *Conformal Classification* node are a calibration table without rank (just plain predictions for the calibration data set), and the unlabeled or test data set with model predictions. Its output is a table with sets of classes.

Perhaps this case is better for beginners, for those who would like to quickly implement the solution, or in case you are trying multiple models with conformal prediction, so you do not spend time on training multiple models of the same type. The results should still be quite satisfactory but usually having multiple rounds of conformal prediction (as in the "advanced" case) might give you more robust results. Hence, I encourage you to give it a try and make your own tests and comparison between the "advanced" and "simple" implementations.

Conclusion

In this article, I showed how to use the conformal prediction nodes in KNIME for a multi-class classification problem. I described how to build an "advanced" and "simple" case for multiple models and calibration tables, and a single pair, respectively. Both approaches were combined with the nodes for Integrated Deployments, so the workflow produces and automatically deploys both the "advanced" and "simple" case. We discussed which metrics can be used for estimating conformal predictions and how one can optimize the error rate.

References

- https://medium.com/low-code-for-advanced-data-science/conformal-prediction-theory-explained-14a35226df80 — the first part of the series where I described the theory of conformal prediction.

- https://www.youtube.com/watch?v=_ZVuEWEfwuw&ab_channel=KNIMETV — webinar where the first version of the nodes was presented.
- https://hub.knime.com/redfield/spaces/Public/latest/Conformal%20prediction%20workshop%20by%20Redfield~BaWdZnceB7O4sqjA — the workflow that has been presented in the webinar.
- https://hub.knime.com/-/spaces/-/latest/~gjQXYhI68F8ff-ef/ — the workflow presented in this article.

This blog post is written as my private initiative and not anyhow related to my current employer.

This article was first published in our Low Code for Advanced Data Science Journal on Medium. Find the original version here.

The corresponding "Conformal prediction classification" workflow can be found on the KNIME Community Hub.

Education and Research

In this section, we compiled all the articles that contain a teaching purpose. This includes academic contributions focusing on Drug Discovery or Gene Ontology, but also other educational articles like tutorials on how certain concepts can be implemented in KNIME Analytics Platform. The category “Education and Research” features our teachers, scientists, and academics:

- **Daniel Weikert**

  – Data Analyst *@DB Systel GmbH*

- **Arjen Peters**

  – Aircraft Data Consultant *@EXSYN Aviation Solutions*

- **Dominique Sydow**

  – Cheminformatics Research Scientist *@Exscientia*

Daniel Weikert was nominated KNIME Contributor of the Month for November 2022. He was awarded for his Udemy courses on KNIME, teaching more than 6000 users everything from how to get started with KNIME to how to convert from Excel to KNIME. In September 2022, Daniel presented the KNIME Webinar "Excel to KNIME – the Why and How to get Started" and he also became a KNIME Certified Trainer afterwards.

Daniel has multiple years of experience working as a Consultant and is currently a Data Analyst at DB Systel GmbH. He holds a Master of Science in Finance and Insurance from Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany. Daniel describes himself as being passionate about data. His experiences are wide-ranging, including not only KNIME but also other tools like Tableau and Power BI, and he also knows his way around with Python. He is teaching some of the concepts on his own YouTube channel (@DanielWeikert).

Visit Daniels' space on the KNIME Hub or his profile page in the KNIME Forum (Hub/Forum handle: `daniel_weikert`).

Successfully handling KNIME and Excel

Learning Tips and Tricks from Daniel Weikert

Author: Daniel Weikert

Excel is a widely used tool for data analysis, with many users relying on its familiar interface and basic features. However, Excel has significant limitations and working with it can be frustrating. The question arises whether there are better alternatives available. And the answer is: yes – indeed we have!

If you're an Excel user and you're considering transitioning to KNIME, you might be wondering where and how to get started. The good news is that KNIME is designed to be user-friendly and easy to learn, even if you've never worked with it before. This is due to KNIME's low-code environment and its intuitive interface, easing getting started with KNIME Analytics Platform.

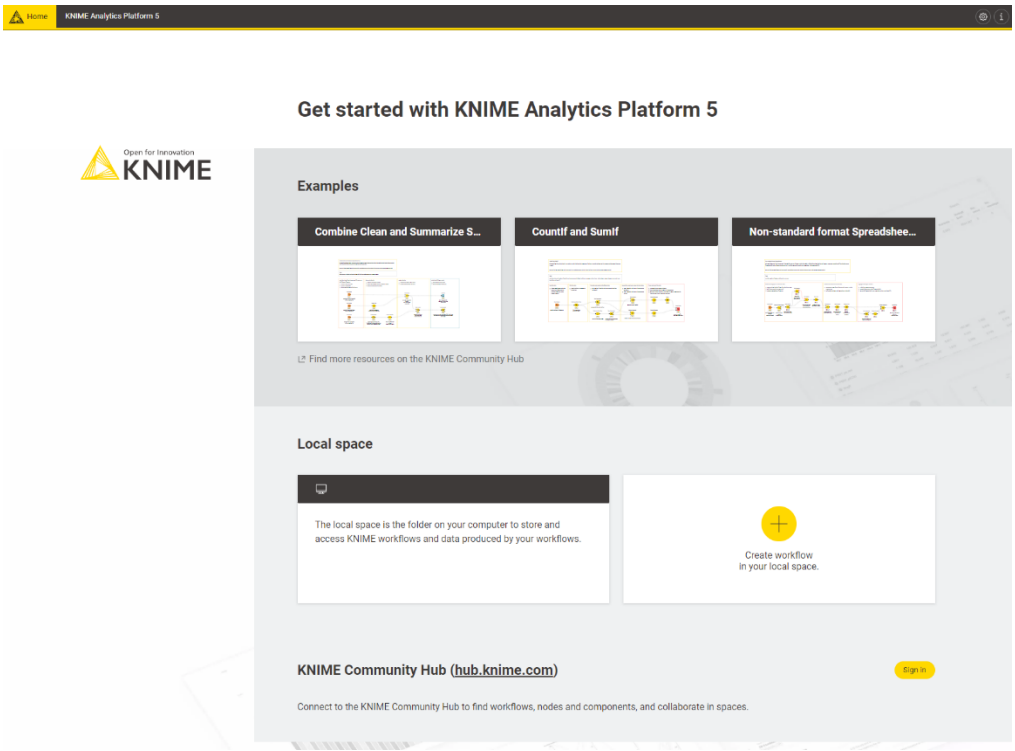
In the following article, Daniel will explain step by step how to transition from Excel to KNIME, by showing how to get started with KNIME and demonstrating 3 common Excel use cases: 1) how to do VLOOKUP, 2) how to find and remove duplicates, and 3) how to create a PivotTable. For this tutorial, he is using the following data:

- Sales_Rep.xlsx (data containing info about the sales area, the product name and the sales manager)
- RegionDicationary.xlsx (data mapping the regions' abbreviations to their full names)
- Transactions_Oct22.csv (contains transactions of energy drinks)

Importing Data into KNIME Analytics Platform

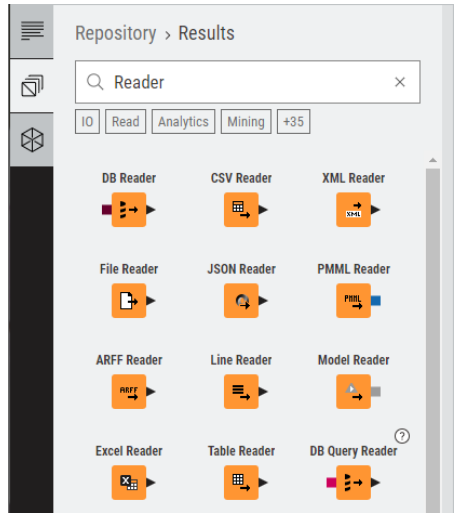
When you start KNIME Analytics Platform, the first page that you will see is the so-called *entry page* (see figure below). On here, you can explore example workflows, connect to the KNIME Community Hub, and create new workflows – either on your local space or in your Community Hub space. You can also browse your local space to access existing workflows.

Once you have created your workflow, you can start adding nodes to it via drag and drop from the *Node Repository*. Each node performs a specific task, such as reading in data, manipulating data, or visualizing data. The nodes are connected to each other, and by chaining multiple nodes, we step by step create a workflow that performs a specific analysis.



The entry page is the first thing you will see when starting KNIME Analytics Platform. Here, you can access example workflows to get started, connect to the KNIME Community Hub, and create new workflows or access your existing workflows.

One of the first steps in transitioning from Excel to KNIME is importing your data. KNIME makes this easy by allowing you to drag and drop files into the KNIME workbench or by using one of the various *Reader* nodes. KNIME supports a vast number of different data formats (see figure on the right), including simple text CSV files, Excel tables, JSON or XML files, up to unstructured data types like documents, images, or audio files.



Some of the most common Reader nodes in KNIME Analytics Platform.

In our case, we are dealing with Excel and CSV files, hence the two nodes required are the *Excel Reader* node and the *CSV Reader* node. There are two ways how to add the node to your workflow:

- We can drag and drop the file into the workflow. That way, the respective *Reader* node is automatically detected and inserted with the file path already

defined in the configuration window. For example, if we drop an Excel file, the *Excel Reader* node is added.

- Alternatively, we can first add the *Excel Reader* node to our workflow via drag and drop from the *Node Repository*, and then open the configuration window and define the file path manually.

Note. If you type in "Reader" in the search bar of the *Node Repository*, all available *Reader* nodes are shown. Click on "More advanced nodes" to see nodes not part of the *Starter Perspective* or adjust the option in the settings to show all nodes.

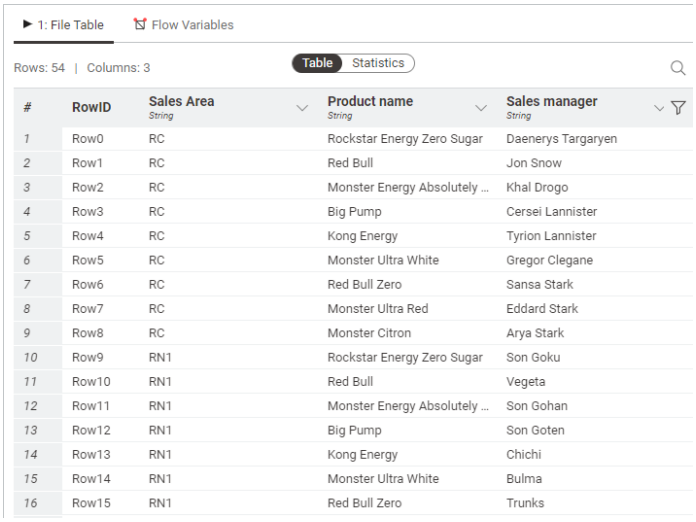
Once our data is imported, we can start building the workflow.

At this point, you might ask yourself whether you can still use the standard Excel functions that you already know by heart like VLOOKUP, SUMIF, and others. The answer is simple: yes, you can still perform all those tasks (and even more!) by using the KNIME nodes. Often times, you will even have more than just one solution. Let me introduce a few of them in the following.

How to do a VLOOKUP

Using the *Value Lookup* Node

Let's take a look at the *Sales_Rep.xlsx* file (figure below). This Excel table contains information about the sales area, the product name and the sales manager. The *Sales Area* column, however, only displays the area code. This is not very intuitive hence we would like to replace the abbreviation with the full area name.



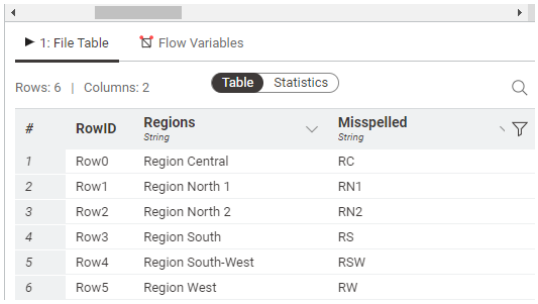
1: File Table | Flow Variables

Rows: 54 | Columns: 3 — Table | Statistics

#	RowID	Sales Area (String)	Product name (String)	Sales manager (String)
1	Row0	RC	Rockstar Energy Zero Sugar	Daenerys Targaryen
2	Row1	RC	Red Bull	Jon Snow
3	Row2	RC	Monster Energy Absolutely ...	Khal Drogo
4	Row3	RC	Big Pump	Cersei Lannister
5	Row4	RC	Kong Energy	Tyrion Lannister
6	Row5	RC	Monster Ultra White	Gregor Clegane
7	Row6	RC	Red Bull Zero	Sansa Stark
8	Row7	RC	Monster Ultra Red	Eddard Stark
9	Row8	RC	Monster Citron	Arya Stark
10	Row9	RN1	Rockstar Energy Zero Sugar	Son Goku
11	Row10	RN1	Red Bull	Vegeta
12	Row11	RN1	Monster Energy Absolutely ...	Son Gohan
13	Row12	RN1	Big Pump	Son Goten
14	Row13	RN1	Kong Energy	Chichi
15	Row14	RN1	Monster Ultra White	Bulma
16	Row15	RN1	Red Bull Zero	Trunks

The Sales_Rep.xlsx file. The table contains information about the sales area, the product name, and the sales manager.

To do so, we need to consider the second Excel file, *RegionDictionary.xlsx* (see below). This table maps the regions' abbreviations to their full names. The "Regions" column contains the full name, the "Misspelled" column contains the abbreviations.



1: File Table | Flow Variables

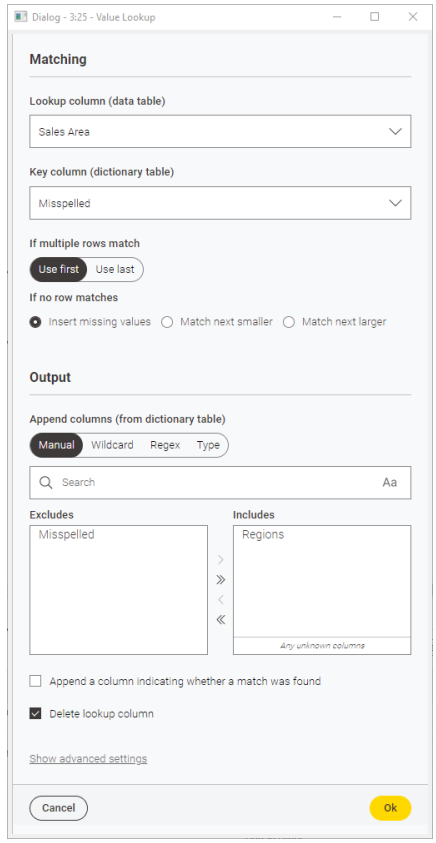
Rows: 6 | Columns: 2 — Table | Statistics

#	RowID	Regions (String)	Misspelled (String)
1	Row0	Region Central	RC
2	Row1	Region North 1	RN1
3	Row2	Region North 2	RN2
4	Row3	Region South	RS
5	Row4	Region South-West	RSW
6	Row5	Region West	RW

The RegionDictionary.xlsx file. This table maps the regions' abbreviations to their full names.

Now, to replace each sales area code with its full name in the upper table (*Sales_Rep.xlsx*) we can use the Value Lookup node. This node has two inputs: the data table we want to adjust the values in (top input port, in our case *Sales_Rep.xlsx*) and the dictionary table (bottom input port, in our case *RegionDictionary.xlsx*). In the configuration window of the *Value Lookup* node (see figure on the right), we now need to set the following settings:

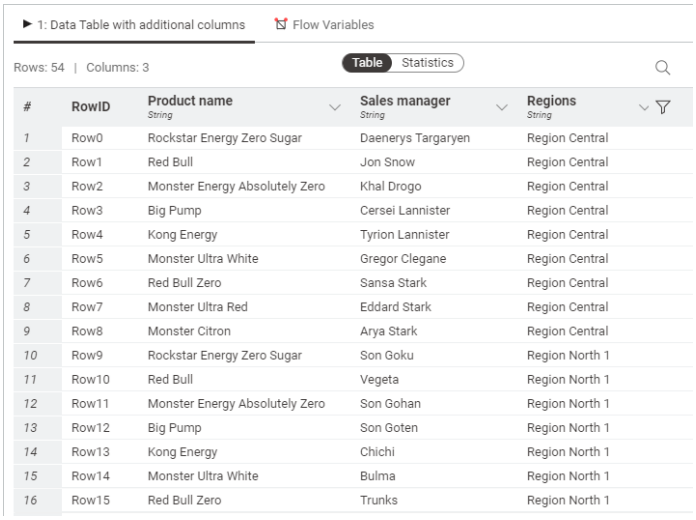
- Lookup column, i.e., the column in the data table containing the value we want to replace: *Sales Area*
- Key column, i.e., the column in the dictionary table containing the value we want to replace: *Misspelled*
- Select the relevant columns of the dictionary table to be included: *Regions*
- Tick the checkbox "Delete lookup column" to remove the column from the data table that contains the region abbreviations



The configuration window of the Value Lookup node.

There are additional settings that are not relevant for our use case here but might be important when using different data. For example, we can decide how to handle multiple matching rows, what to do if no match was found, and whether we want to add the information when a match was found.

The resulting table then looks as follows:



1: Data Table with additional columns | Flow Variables

Rows: 54 | Columns: 3 — Table | Statistics

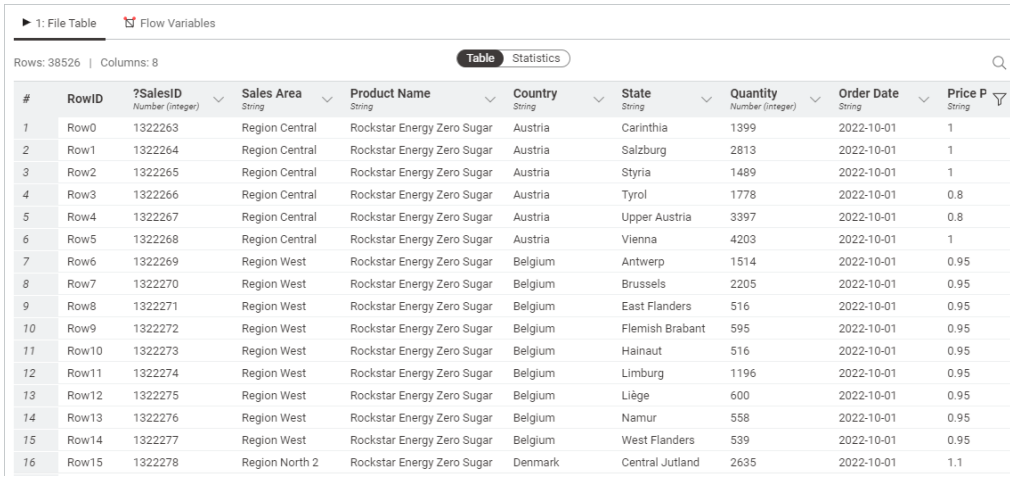
#	RowID	Product name (String)	Sales manager (String)	Regions (String)
1	Row0	Rockstar Energy Zero Sugar	Daenerys Targaryen	Region Central
2	Row1	Red Bull	Jon Snow	Region Central
3	Row2	Monster Energy Absolutely Zero	Khal Drogo	Region Central
4	Row3	Big Pump	Cersei Lannister	Region Central
5	Row4	Kong Energy	Tyrion Lannister	Region Central
6	Row5	Monster Ultra White	Gregor Clegane	Region Central
7	Row6	Red Bull Zero	Sansa Stark	Region Central
8	Row7	Monster Ultra Red	Eddard Stark	Region Central
9	Row8	Monster Citron	Arya Stark	Region Central
10	Row9	Rockstar Energy Zero Sugar	Son Goku	Region North 1
11	Row10	Red Bull	Vegeta	Region North 1
12	Row11	Monster Energy Absolutely Zero	Son Gohan	Region North 1
13	Row12	Big Pump	Son Goten	Region North 1
14	Row13	Kong Energy	Chichi	Region North 1
15	Row14	Monster Ultra White	Bulma	Region North 1
16	Row15	Red Bull Zero	Trunks	Region North 1

The resulting table when performing a VLOOKUP using the Value Lookup node to replace the regions' abbreviations with their full names.

Using the *Joiner* Node

Now that I've shown you one way how to replicate a VLOOKUP in KNIME, let me show you a second variant. Instead of using the *Value Lookup* node, we can also use the *Joiner* node. This is required for more sophisticated matching criteria, which will be explained below.

Let's first have a look at the third data table, *Transactions_Oct22.csv*. This table contains information about transactions of energy drinks, for example, the sales ID the sales area, country, and state, the product name, the quantity, etc.



1: File Table | Flow Variables

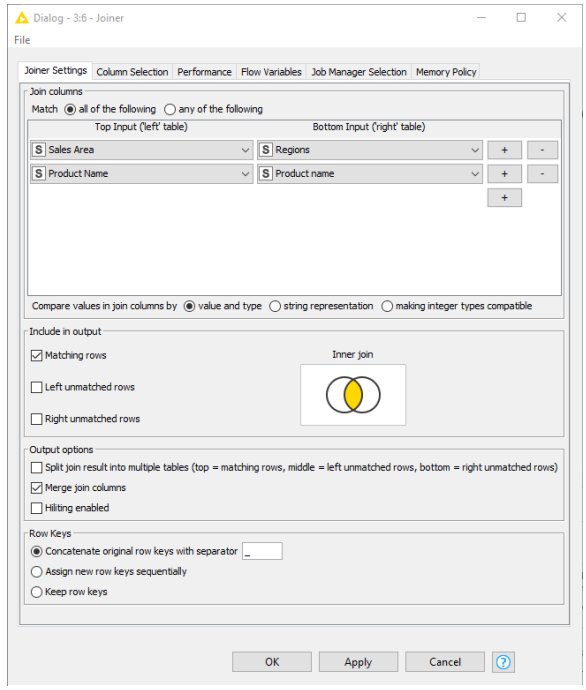
Rows: 38526 | Columns: 8 — Table | Statistics

#	RowID	?SalesID (Number (integer))	Sales Area (String)	Product Name (String)	Country (String)	State (String)	Quantity (Number (integer))	Order Date (String)	Price P (String)
1	Row0	1322263	Region Central	Rockstar Energy Zero Sugar	Austria	Carinthia	1399	2022-10-01	1
2	Row1	1322264	Region Central	Rockstar Energy Zero Sugar	Austria	Salzburg	2813	2022-10-01	1
3	Row2	1322265	Region Central	Rockstar Energy Zero Sugar	Austria	Styria	1489	2022-10-01	1
4	Row3	1322266	Region Central	Rockstar Energy Zero Sugar	Austria	Tyrol	1778	2022-10-01	0.8
5	Row4	1322267	Region Central	Rockstar Energy Zero Sugar	Austria	Upper Austria	3397	2022-10-01	0.8
6	Row5	1322268	Region Central	Rockstar Energy Zero Sugar	Austria	Vienna	4203	2022-10-01	1
7	Row6	1322269	Region West	Rockstar Energy Zero Sugar	Belgium	Antwerp	1514	2022-10-01	0.95
8	Row7	1322270	Region West	Rockstar Energy Zero Sugar	Belgium	Brussels	2205	2022-10-01	0.95
9	Row8	1322271	Region West	Rockstar Energy Zero Sugar	Belgium	East Flanders	516	2022-10-01	0.95
10	Row9	1322272	Region West	Rockstar Energy Zero Sugar	Belgium	Flemish Brabant	595	2022-10-01	0.95
11	Row10	1322273	Region West	Rockstar Energy Zero Sugar	Belgium	Hainaut	516	2022-10-01	0.95
12	Row11	1322274	Region West	Rockstar Energy Zero Sugar	Belgium	Limburg	1196	2022-10-01	0.95
13	Row12	1322275	Region West	Rockstar Energy Zero Sugar	Belgium	Liège	600	2022-10-01	0.95
14	Row13	1322276	Region West	Rockstar Energy Zero Sugar	Belgium	Namur	558	2022-10-01	0.95
15	Row14	1322277	Region West	Rockstar Energy Zero Sugar	Belgium	West Flanders	539	2022-10-01	0.95
16	Row15	1322278	Region North 2	Rockstar Energy Zero Sugar	Denmark	Central Jutland	2635	2022-10-01	1.1

The Transactions_Oct22.csv file. The table contains transaction information about energy drinks.

We now want to add the information we gathered in the first step to this transactions table. This is, for each transaction we want to add the respective sales manager which is different for each combination of product and region. Doing this with the *Value Lookup* node would become complicated, as we can only define one lookup column (see configuration window of *Value Lookup* node above). In this case, however, we need to be able to define two columns as the lookup criterion, as we need to make sure that the *Product Name* and the *Sales Area* match in order to determine the respective *Sales Manager*.

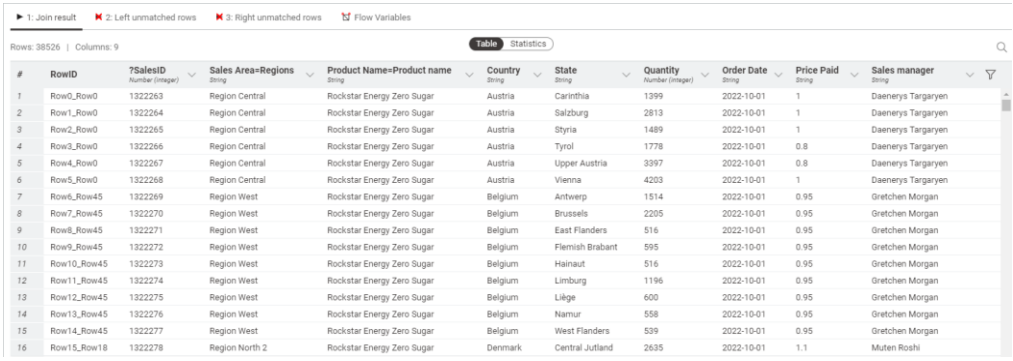
To do so, we add the *Joiner* node to our workflow. This node combines two tables in such a way that each row from the top input port is combined with each row from the bottom input port whenever the values in the selected column are identical. Depending on the setting of the configuration window, unmatched rows can also be included in the output table.



The configuration window of the Joiner node.

In the configuration window of the *Joiner* node, we define the join criterion. Here, each time where the values in the column *Sales Area* and *Regions* are identical as well as the values of the *Product Name* columns of both tables, the rows are combined. Furthermore, we check the box “Matching rows”, which replicated an inner join operation, and the box “Merge join columns” to avoid having the *Sales Area* column and the *Regions* column in the output table.

The resulting table then looks as follows:

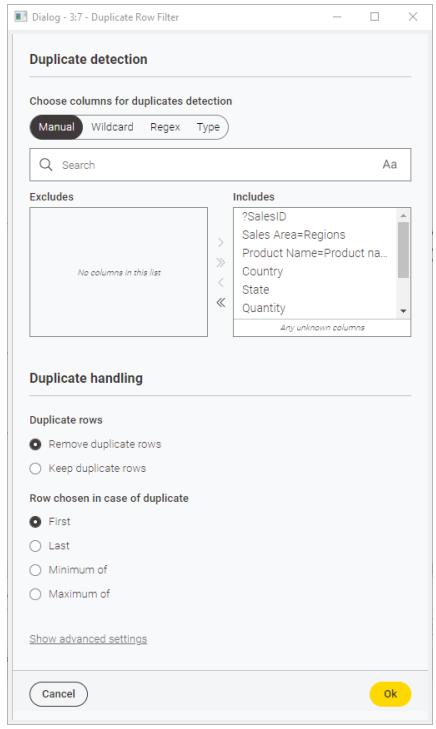


#	RowID	?SalesID	Sales Area=Regions	Product Name=Product name	Country	State	Quantity	Order Date	Price Paid	Sales manager
1	Row0_Row0	1322263	Region Central	Rockstar Energy Zero Sugar	Austria	Carinthia	1399	2022-10-01	1	Daenerys Targaryen
2	Row1_Row0	1322264	Region Central	Rockstar Energy Zero Sugar	Austria	Salzburg	2813	2022-10-01	1	Daenerys Targaryen
3	Row2_Row0	1322265	Region Central	Rockstar Energy Zero Sugar	Austria	Styria	1489	2022-10-01	1	Daenerys Targaryen
4	Row3_Row0	1322266	Region Central	Rockstar Energy Zero Sugar	Austria	Tyrol	1778	2022-10-01	0.8	Daenerys Targaryen
5	Row4_Row0	1322267	Region Central	Rockstar Energy Zero Sugar	Austria	Upper Austria	3397	2022-10-01	0.8	Daenerys Targaryen
6	Row5_Row0	1322268	Region Central	Rockstar Energy Zero Sugar	Austria	Vienna	4203	2022-10-01	1	Daenerys Targaryen
7	Row6_Row45	1322269	Region West	Rockstar Energy Zero Sugar	Belgium	Antwerp	1514	2022-10-01	0.95	Gretchen Morgan
8	Row7_Row45	1322270	Region West	Rockstar Energy Zero Sugar	Belgium	Brussels	2205	2022-10-01	0.95	Gretchen Morgan
9	Row8_Row45	1322271	Region West	Rockstar Energy Zero Sugar	Belgium	East Flanders	516	2022-10-01	0.95	Gretchen Morgan
10	Row9_Row45	1322272	Region West	Rockstar Energy Zero Sugar	Belgium	Flemish Brabant	595	2022-10-01	0.95	Gretchen Morgan
11	Row10_Row45	1322273	Region West	Rockstar Energy Zero Sugar	Belgium	Hainaut	516	2022-10-01	0.95	Gretchen Morgan
12	Row11_Row45	1322274	Region West	Rockstar Energy Zero Sugar	Belgium	Limburg	1196	2022-10-01	0.95	Gretchen Morgan
13	Row12_Row45	1322275	Region West	Rockstar Energy Zero Sugar	Belgium	Liège	600	2022-10-01	0.95	Gretchen Morgan
14	Row13_Row45	1322276	Region West	Rockstar Energy Zero Sugar	Belgium	Namur	558	2022-10-01	0.95	Gretchen Morgan
15	Row14_Row45	1322277	Region West	Rockstar Energy Zero Sugar	Belgium	West Flanders	539	2022-10-01	0.95	Gretchen Morgan
16	Row15_Row18	1322278	Region North 2	Rockstar Energy Zero Sugar	Denmark	Central Jutland	2635	2022-10-01	1.1	Muten Roshi

The resulting table when performing a VLOOKUP using the Joiner node to enhance the data table with additional information about the sales manager.

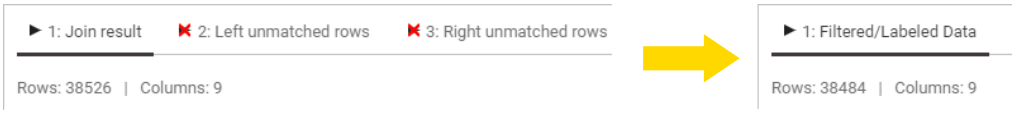
How to Find and Remove Duplicates

Another common function in Excel is duplicate handling. In KNIME Analytics Platform, we can use the *Duplicate Row Filter* node for that. In the configuration window (see figure on the right), you can define the criterion when a data row is identified as a duplicate. For example, you can choose the columns in the “Include” panel in such a way, that the row is considered a duplicate when the values in all columns are identical. Alternatively, you can remove columns from the “Input” panel, which makes the duplicate definition less strict. For example, you can select only the *SalesID* column in the “Input” panel which would mean, that the row is considered a duplicate as soon as the values in the *SalesID* column are identical independent of the values in the other columns. Additionally, you can define how the duplicates should be handled. You can either remove all duplicate rows from the input table and keep only unique and chosen rows or mark the rows with additional information about their duplication status.



The configuration window of the Duplicate Row Filter node.

If we now compare the output table of the *Joiner* node and the output table of the *Duplicate Row Filter* node, we will see that our data table contained indeed a few duplicate rows (see figure below).



The result after removing duplicate rows from the data table: roughly 50 rows were removed.

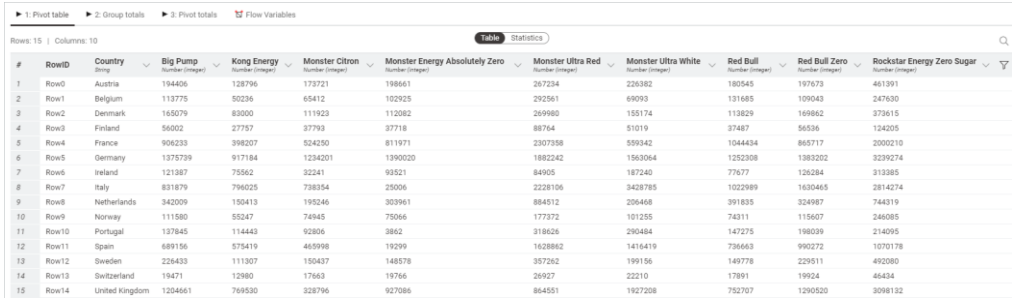
How to Create a PivotTable

To create a PivotTable in KNIME Analytics Platform, we can use the *Pivot* node. This node performs a pivoting on the given input table using a selected number of columns for grouping and pivoting.

There are multiple settings required in the configuration window (see figure below).

- "Groups" tab: Select the row(s) according to which the group rows are created. We select the "Country" column as group column
- "Pivots" tab: Select the row(s) according to which the pivot columns are created. We select the "Product Name" column as pivot column
- "Manual Aggregation" tab: Select the column(s) for aggregation and the aggregation method. We use the "Quantity" column to calculate the sum for each products sold in each country.

The resulting PivotTable is shown below. It shows how many energy drinks were sold in each country.


1: Pivot table | 2: Group totals | 3: Pivot totals | Flow Variables

Rows: 15 | Columns: 10 — Table | Statistics

#	RowID	Country	Big Pump	Kong Energy	Monster Citron	Monster Energy Absolutely Zero	Monster Ultra Red	Monster Ultra White	Red Bull	Red Bull Zero	Rockstar Energy Zero Sugar
1	Row0	Austria	194406	128796	173721	198661	267234	226382	180545	197673	461391
2	Row1	Belgium	113775	50236	65412	102925	292561	69093	131685	109043	247630
3	Row2	Denmark	165079	83000	111923	112082	269980	155174	113829	169862	373615
4	Row3	Finland	56002	27757	37793	37718	88764	51019	37487	56536	124205
5	Row4	France	906233	398207	524250	811971	2307358	559342	1044434	865717	2000210
6	Row5	Germany	1375739	917184	1234201	1390020	1882242	1563064	1252308	1383202	3239274
7	Row6	Ireland	121387	75562	32241	93521	84905	187240	77677	126284	313385
8	Row7	Italy	831879	796025	738354	25006	2228106	3428785	1022989	1630465	2814274
9	Row8	Netherlands	342009	150413	195246	303961	884512	206468	391835	324987	744319
10	Row9	Norway	111580	55247	74945	75066	177372	101255	74311	115607	246085
11	Row10	Portugal	137845	114443	92806	3862	318626	290484	147275	198039	214095
12	Row11	Spain	689156	575419	465998	19299	1628862	1416419	736663	990272	1070178
13	Row12	Sweden	226433	111307	150437	148578	357262	199156	149778	229511	492080
14	Row13	Switzerland	19471	12980	17663	19766	26927	22210	17891	19924	46434
15	Row14	United Kingdom	1204661	769530	328796	927086	864551	1927208	752707	1290520	3098132

The resulting table when creating a PivotTable using the Pivot node. The table shows how many energy drinks were sold in each country.

Summary

While creating workflows in KNIME Analytics Platform does require some learning, there are many resources available to help Excel users get started. The KNIME website provides a range of tutorials and documentation, and there are many active users in the KNIME community who are happy to help beginners (you will find me there as well 😉).

One of the advantages of transitioning to KNIME from Excel is that KNIME provides many features that are missing in Excel, such as the ability to handle large datasets, perform advanced data cleaning and preprocessing, and automate repetitive tasks. This means that while there may be a learning curve, investing time in learning KNIME can pay off in terms of efficiency and accuracy in your data analysis.

Another advantage of KNIME is that it provides a range of pre-built workflows and nodes that can be easily customized for your specific analysis. This means that even if you're not comfortable building your own workflows from scratch, you can still use KNIME to perform advanced data analysis.

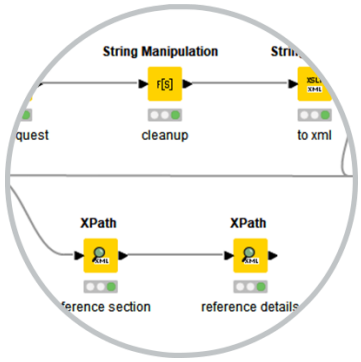
In summary, transitioning from Excel to KNIME may seem daunting at first, but KNIME is designed to be user-friendly and easy to learn. With a little bit of time and effort, Excel users can start taking advantage of the advanced data analysis capabilities provided by KNIME.

I have even created a complete course "KNIME for Microsoft Excel Users" to make the transition as easy as possible. In the course, we cover a little case study with various Excel tasks and their counterparts using KNIME Analytics Platform.

The corresponding "3 KNIME Excel CaseStudy solution" workflow partially shown in this article can be found on the KNIME Community Hub.

Arjen Peters was nominated KNIME Contributor of the Month for June 2023. He was awarded for his tireless effort as a support KNinja in the KNIME Community. In the past year, he distilled as a key leader for solutions proposed, likes received, and replies given. Whether he's answering questions about data manipulation, troubleshooting models, or suggesting a few lines of RegEx, Arjen can always be found freely giving his time to make the KNIME Forum a great resource for everyone.

Arjen holds a Bachelor in Aviation Operations & Logistics and currently works as Aircraft Data Consultant at EXSYN Aviation Solution based in Amsterdam. Arjen has a keen interest in data and his primary focus are data migration, data analysis, establishing interfaces, and deploying Natural Language Processing models. For this, he uses KNIME extensively on a daily basis with a great focus on workflow standardization and the UX of their in-house data apps.



Visit Arjen's space on the KNIME Hub or his profile in the KNIME Forum (Hub/Forum handle: `arjenex`).

Processing Paginated API Responses in KNIME

Retrieving all YouTube Comments from the 10 Hours Relaxing Fireplace Video

Author: Arjen Peters

Introduction

When working with APIs in KNIME, one of the challenges that you potentially have to overcome is pagination. Pagination in APIs is a technique used to manage large sets of data returned by the endpoint. When an API has a large number of items or records, its developers could opt to apply a limit to the number of records that it returns on each call/GET Request.

As such, pagination allows the server to divide the data into smaller, manageable chunks called “pages” and return one page at a time to the client. The key purpose of pagination is to reduce the response size and improve the API's performance. It also helps to avoid overloading the client and server with excessive data, especially when dealing with limited bandwidth or processing resources.

In a REST API for example, pagination is usually achieved by including query parameters in the API request to specify which page of data is being requested. This may vary depending on the API, making it necessary to first study the technical documentation of the API before moving to KNIME.

Extract Comments on a YouTube Video

For the sake of this tutorial, we refer to this post on the KNIME Community Forum, where a user requested assistance with retrieving comments that were left on a video on YouTube. As the user noted, Google implemented pagination on its API endpoint for the website, and he asked for help regarding this. As per its documentation, the pagination is controlled by the parameters *maxResults* and *nextPageToken* whereby the former specifies the maximum number of items that should be returned in the result set and the latter identifies the next page of the result that can be retrieved.

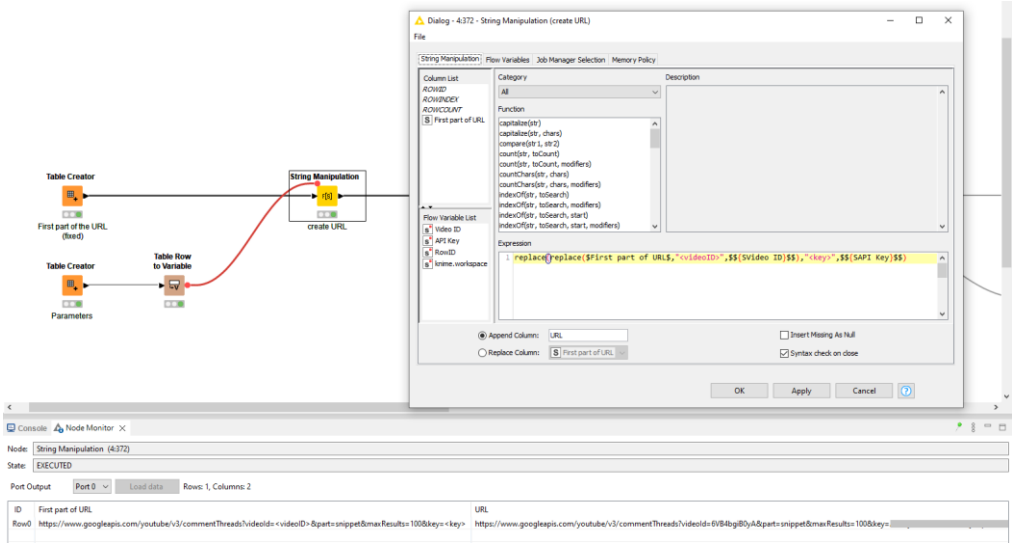
For illustration purposes, the video analyzed here is a 10-hour loop of a “Relaxing Fireplace”. It currently has about a thousand comments. Since the API only allows up to 100 results per request, it is certain that multiple requests are required to retrieve the comments.

Step 1: Making the Initial Request

Moving on to KNIME, the first major step is to make the initial call. For this, the correct URL needs to be created. To do this, one can start with the base URL:

```
https://www.googleapis.com/youtube/v3/commentThreads?videoId=<videoID>&part
=snippet&maxResults=100&key=<key>
```

Noticeably, the *videoID* and *key* need to be filled in and the *maxResults* is set to 100. For convenience, one can create two tables whereby the second instance holds these parameters. Once they have been converted to a flow variable, the final URL can be created by using a nested *replace()* function. The final URL can then be used as input for the *GET Request* node.



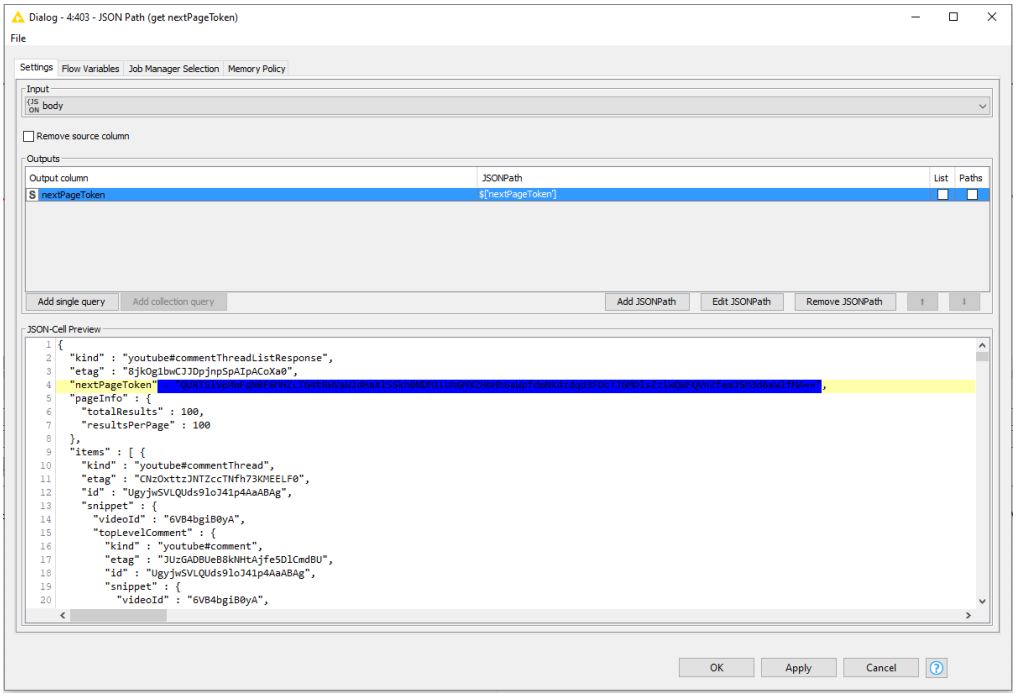
To make the initial call, first the correct URL for making the GET Request needs to be created. This is done with the String Manipulation node using the replace() function.

Upon executing, the API returns a JSON file that holds all the comments on the mentioned video. More importantly, it is noticeable that the *nextPageToken* is mentioned in its header (see figure below).

It is important to analyze this before moving on as it serves two purposes:

1. If it is not present it means that the number of comments on the video are lower than *maxResults* and pagination is subsequently not required.
2. If it is present, we know that the section of the KNIME workflow that eventually will handle it needs to be “activated”.

The API returns a JSON file, which is one cell in a KNIME table that holds all the comments made on the video. The response also contains the nextPageToken parameter.

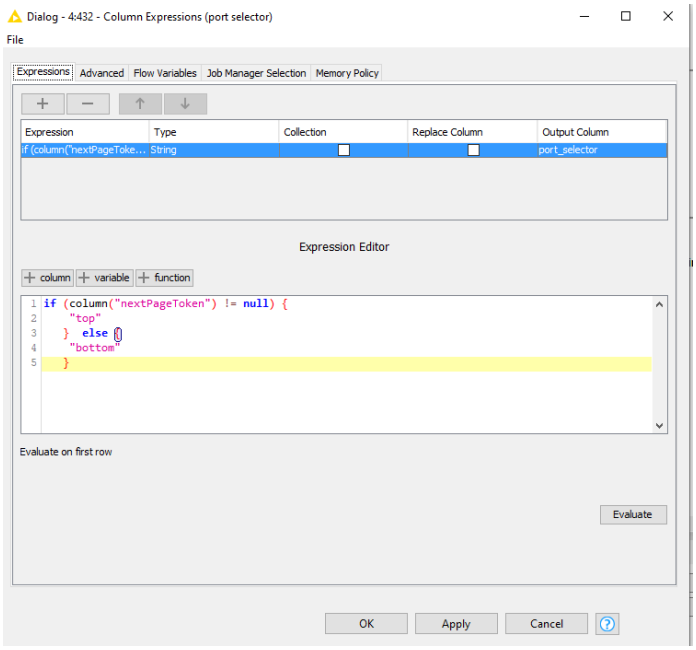


The configuration window of the JSON Path node. The node returns the value of the nextPageToken parameter which either contains a value, if a next page is available, or contains no value, if no next page is available.

Step 2: Determine if *nextPageToken* Parameter Contains any Value

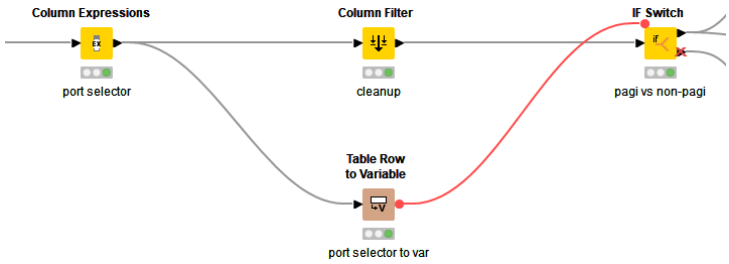
To retrieve the *nextPageToken*, use a *JSON Path* node and query `$['nextPageToken']`. This also applies whenever it is not present.

As such, the next step is to dynamically determine if the *nextPageToken* contains any value. This can be determined with any node that can evaluate rules such as the *Column Expression* node or *Rule Engine* node. As mentioned, we are looking at two options: pagination and non-pagination. This is translated into a value of *top* or *bottom*. We name this the *port_selector* value.



The configuration window of the Column Expression node. The node is used to determine, whether the nextPageToken parameter contains any value or not.

It is fed as variable into the *IF Switch* node as it passes the data either to the top, bottom, or both output ports, depending on the configuration. As the token is present in our example, the top port is activated, and the bottom port is deactivated.



The workflow snippet that is responsible for determining, whether pagination is necessary or not.

Step 3: Processing the Contents of the Response JSON

This now leads us to the main engine of the workflow (see figure below), processing the contents of the JSON. For this, 3 sections are used:

- Section 1: Process the first GET Request – Paginated

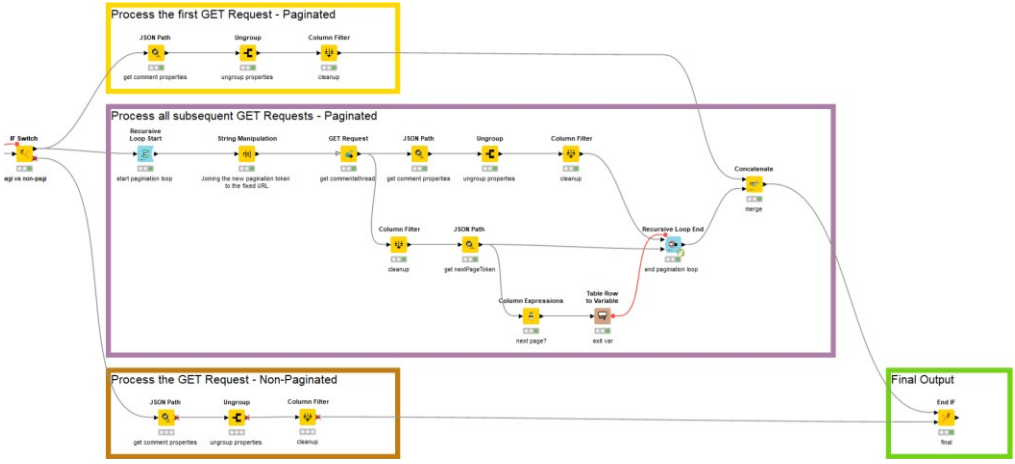
  This section processes the initial call to the API that contains the first 100 results that contains pagination.

- Section 2: Process all subsequent GET Requests – Paginated

  This section is to iterate through all the next pages.

- Section 3: Process the GET Request – Non-Paginated

  This section is to process the non-paginated results (i.e., if the *nextPageToken* parameter has no value and therefore the bottom port of the *IF Switch* node is activated).



The main engine of the workflow that is responsible for processing the contents of the JSON response.

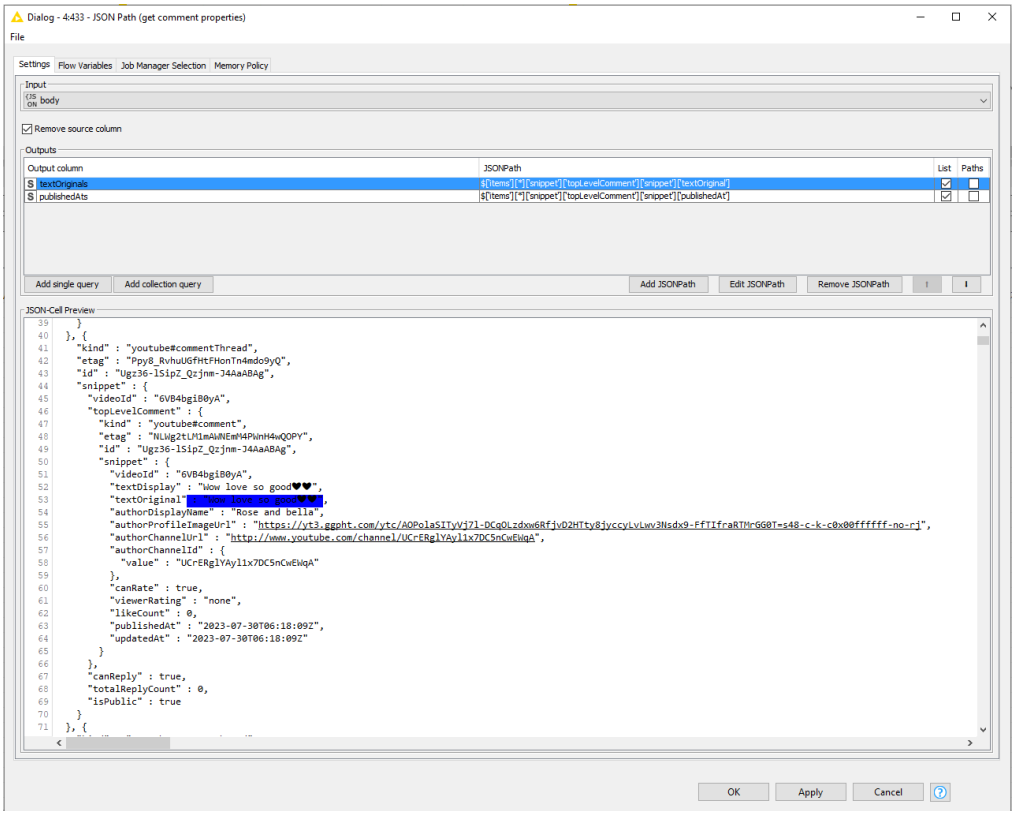
Section 1: Process the first GET Request – Paginated

To begin with, let's look at the first section which processes the initial call to the API. While the API returns many attributes, the focus will be on two: *textOriginals* which contains the text of the posted comment and *publishedAts* which designates when it was posted. For this, a *JSON Path* node is used again. The following queries apply:

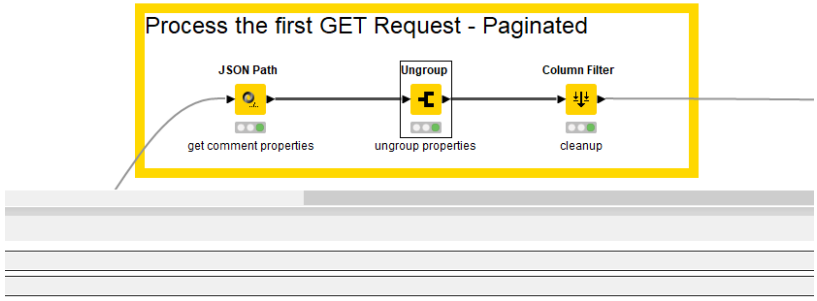
```
$['items'][*]['snippet']['topLevelComment']['snippet']['textOriginal']

$['items'][*]['snippet']['topLevelComment']['snippet']['publishedAt']
```

It is important to note the asterisk in the second position of the query (see configuration window below). This represents a wildcard that will ensure that all rows will be captured, and its contents retrieved. To ensure this, the checkbox to create a list also has to be set in the configuration.



The configuration window of the JSON Path node. The node returns the value of the textOriginals and publishedAts parameters – the content we are interested in.



textOriginals	publishedAts
마음이 따스해진다!	2023-08-06T03:02:05Z
Wow love so good♡♡	2023-07-30T06:18:09Z
nice video	2023-07-28T13:16:41Z
balck	2023-07-23T19:09:15Z
I like the part where it sounds like fire♡♡	2023-07-21T22:03:59Z
or a rocket fuel furnace	2023-07-19T13:23:40Z
yachts aye fire yachts aye i remember those back in the day to bad you cant do a space launch fireplace	2023-07-19T13:21:27Z
Anormal derecede hızlı . Ateşin acelesi var gibi	2023-07-19T06:47:19Z
these vids are perfect to fall asleep to!	2023-07-16T00:07:50Z
3h 38m of sleep, Thankyou!	2023-07-03T07:03:59Z
Im soo proud of you! ♡ i love u even if no one else does. u are Gods beautiful creation and he lovess you sooo much! Godbless w...	2023-06-16T08:24:13Z
In 144	2023-06-15T01:32:52Z
Yooooo this song is fire!	2023-06-13T15:20:34Z
Shit	2023-06-12T18:57:48Z

The workflow snippet that is responsible for converting the list with all the values into

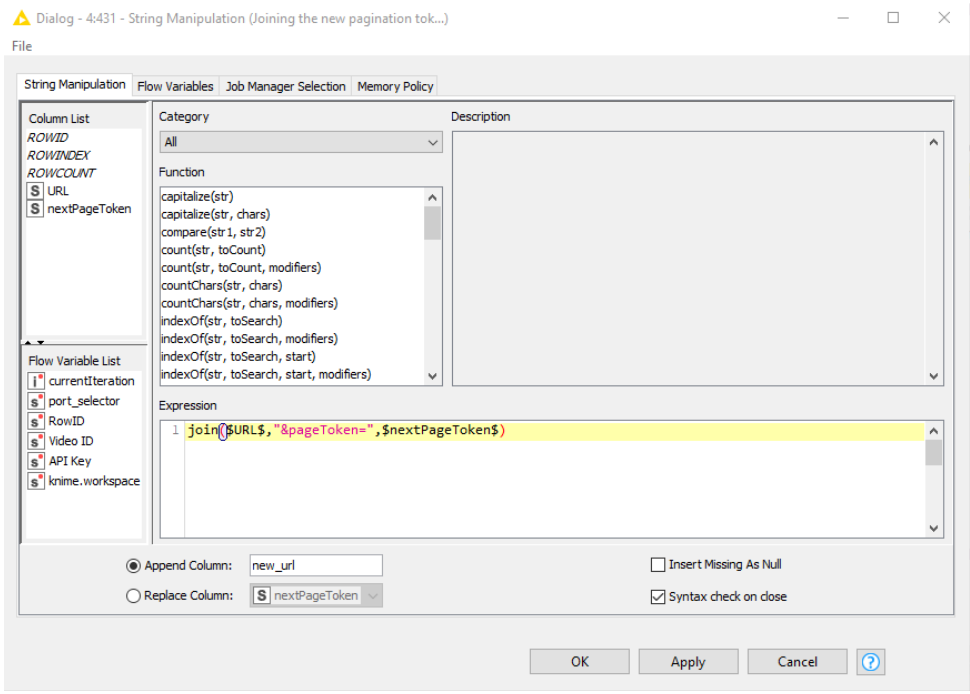
Next, the list with all the values received from the *JSON Path* node is converted to rows. For this, the *Ungroup* node is used (see figure above). In the configuration window, include the two lists created earlier. The default settings of the node are sufficient to process the data accordingly.

After this, a small clean-up of the data is done by means of a *Column Filter* node to only retain the relevant columns. We now have transformed the JSON and extracted the plain text comments with its associated publishing date.

Section 2: Process all subsequent GET Requests – Paginated

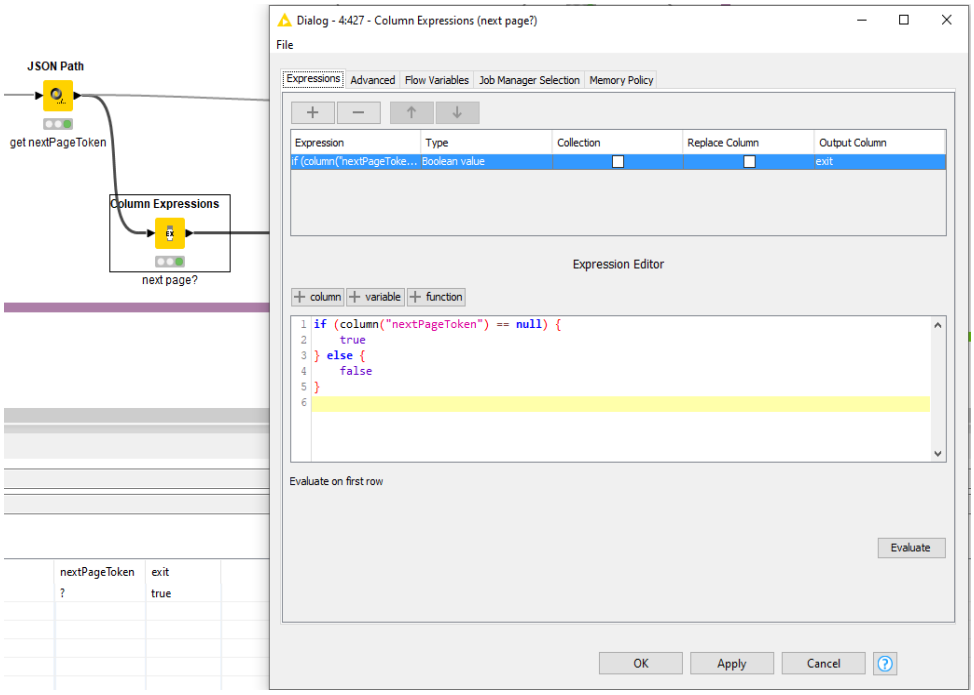
The second section is built around the *Recursive Loop* nodes. The most important feature of this loop type is that the tables received by the *Recursive Loop End* node are passed back to the *Recursive Loop Start* node until a certain condition to exit the loop is met. In our case, we need the *nextPageToken* from iteration 0, modify the URL to make the GET Request and pass it along for iteration 1. This step is then recursively repeated until the *exit_condition* is met.

To arrange this, an updated URL is created through a *String Manipulation* node that includes the *nextPageToken* and joins this with the base URL (see figure below).



The configuration window of the String Manipulation node. This node is placed within in a recursive loop to recursively generate the specific URLs for each GET Requests, including each nextPageToken parameter in the URLs.

After this, the processing steps are familiar: querying the *textOriginals* and *publishedAts* and converting this into rows of data.



The configuration window of the Column Expressions node. This node is used to determine, whether there is a next page to query or whether the loop should be exited.

Since it is not known how many iterations the loop has to perform in advance, it is key to also check for the presence of the *nextPageToken* within the loop. Ultimately, this is to determine to what is being referred to as the *exit_condition*. A feature of the *Recursive Loop End* node is to pass a variable that will cause the loop to exit. In our case, whenever there is no more *nextPageToken* present, the loop should exit. A rule evaluation is put in place to determine this and create a Boolean value accordingly (see figure above).

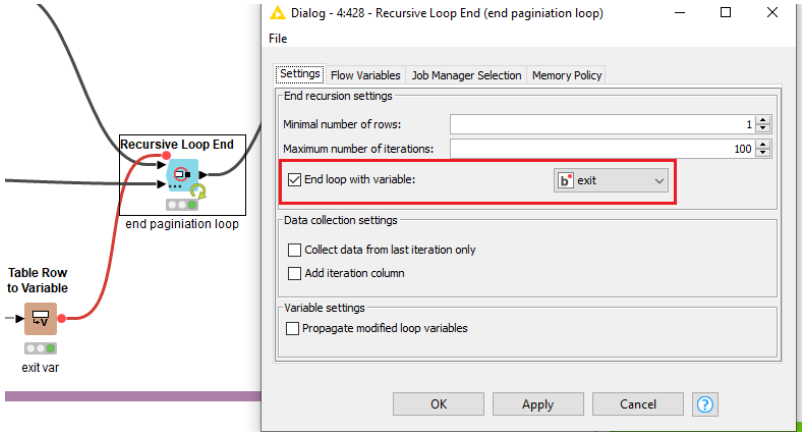
With a *Table Row to Variable* node, the created Boolean value is converted into a flow variable and passed along to the *Loop End* node (see figure below). In the configuration window of the *Recursive Loop End* node, one should activate the option to end the loop with a variable and select the correct option from the dropdown list.

Now to the most important part: setting up the *Recursive Loop End* node correctly. It is significant to understand what the requires:

- Port 0 represents the data to be collected for the output.
- Port 1 represents data to be passed back to loop start.

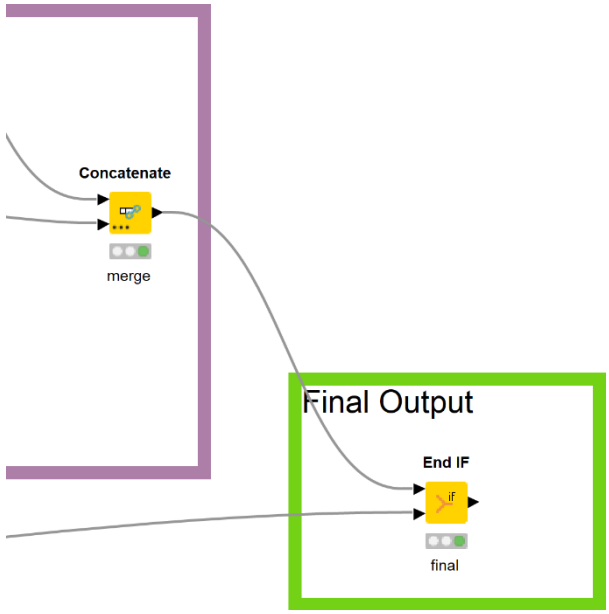
As mentioned before, the tables received by the *Recursive Loop End* node are passed back to the *Recursive Loop Start* node. Subsequently, the comments need to be put into port 0 and the *nextPageToken* into port 1. This value is then used in the $n + 1$ loop

to make another GET Request after which the whole process repeats itself until the exit condition is met.



The Table Row to Variable Loop node converts the Boolean value created by the Column Expressions node into a flow variable. This flow variable – "exit" – is then used to determine the loop exit_condition.

Since the very first GET Request was processed separately (top branch in of the workflow), a *Concatenate* node is used to merge the two data streams together (see figure below).



The Concatenate node is required to merge the very first GET Request and the remaining recursive GET Requests. The End IF node is required at the end of the workflow to properly merge the two main branches and allow further processing (which is not considered here).

Section 3: Process the GET Requests – Non-Paginated

In case the *nextPageToken* parameter has no value, the bottom port of the IF Switch node is activated. In that case, only the bottom branch of the workflow is executed, hence the GET Request if only performed once.

Step 4: Merging the two Main Branches

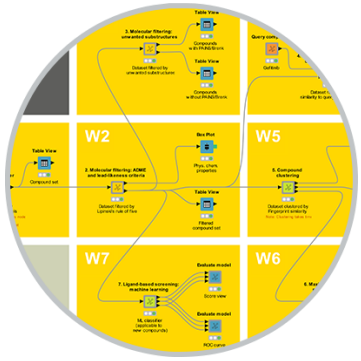
Going back to the start, we used the *IF Switch* node to determine whether pagination was required or not, creating two branches in the workflow: the upper branch being active if the response JSON includes pagination, the lower branch being active otherwise. To properly merge the two branches after processing the content of the GET Requests, a *End IF* node is required. The node is placed at the very end of the workflow and thus allows further processing of the data (which is not considered here).

Note. For this workflow to work, it is required to have an API key. Instructions on how to obtain this are available here: Obtaining authorization credentials.

The corresponding "Processing paginated API responses in KNIME" workflow *can be found on the KNIME Community Hub.*

Dominique Sydow was nominated KNIME Contributor of the Month for February 2023. She was awarded for the paper "*TeachOpenCADD-KNIME: A Teaching Platform for Computer-Aided Drug Design Using KNIME Workflows*", where she is the first author. The paper describes a KNIME workflow for different computer-aided drug design (CADD) tasks: a valuable resource for teaching students and life sciences low-coders. The workflow, publicly available on the KNIME Community Hub, is divided into eight components ready for execution on some example data. There is a different component for each topic, such as how to fetch, filter and analyze compound data.

Dominique is currently a Cheminformatics Research Scientist at Exscientia, a global pharmatech company. She describes herself as being interested in mining chemical, structural, and pharmacological data to support decisions in drug discovery. Her main focus lies on studying binding site similarity and protein-ligand interactions. Dominique also repeatedly contributes to TeachOpenCADD, a teaching platform for open-source Cheminformatics and structural Bioinformatics.



Visit Dominique's space on the KNIME Community Hub or her profile in the KNIME Forum (Hub/Forum handle: `dominiquesydow`). She is also contributing to the Hub space of Volkamer Lab (Hub handle: `volkamerlab`).

TeachOpenCADD-KNIME: A Teaching Platform for Computer-Aided Drug Design Using KNIME Workflows

*Authors: Dominique Sydow, Michele Wichmann, Jaime Rodríguez-Guerra, Daria Goldmann, Gregory Landrum, and Andrea Volkamer**

Editor's Note:

This article corresponds to the publication by Dominique Sydow, Michele Wichmann, Jaime Rodríguez-Guerra, Daria Goldmann, Gregory Landrum, and Andrea Volkamer: TeachOpenCADD-KNIME: A Teaching Platform for Computer-Aided Drug Design Using KNIME Workflows. Journal of Chemical Information and Modeling ***2019*** *59 (10), 4083-4086. http://dx.doi.org/10.1021/acs.jcim.9b00662*

Introduction

In computer-aided drug design (CADD), computational tools are used to process and rationalize large and heterogeneous data sets involving small molecules and macromolecules. For this endeavor, open-access resources have gained momentum, especially for setting up complex workflows, since they enable modular, reproducible, and reusable research. We recently reported the TeachOpenCADD[1] teaching platform (https://github.com/volkamerlab/teachopencadd) that provides learning material for CADD using open-source data and Python libraries. Central topics in CADD are covered in the form of interactive Jupyter Notebooks that contain both theory and code for each topic. An alternative to code-based pipelines are workflow managers that allow the design of protocols via an intuitive drag-and-drop style graphical interface without the need for coding. KNIME[2,3] is a popular workflow manager for data science with several open-source modules for CADD,[4] while its usage ranges from small in-house applications such as compound library preparation to more complex workflow applications integrating chemical, pharmacological, and structural information. An

[1] *Sydow, D.; Morger, A.; Driller, M.; Volkamer, A. TeachOpenCADD: A Teaching Platform for Computer-Aided Drug Design Using Open Source Packages and Data. J. Cheminf. 2019, 11, 29.*

[2] *Berthold, M. R.; Cebron, N.; Dill, F.; Gabriel, T. R.; Kötter, T.; Meinl, T.; Ohl, P.; Sieb, C.; Thiel, K.; Wiswedel, B. KNIME: The Konstanz Information Miner. In Data Analysis, Machine Learning and Applications; Preisach, C., Burkhardt, H., Schmidt-Thieme, L., Decker, R., Eds.; Springer: Berlin, 2008; pp 319–326.*

[3] *Fillbrunn, A.; Dietz, C.; Pfeuffer, J.; Rahn, R.; Landrum, G. A.; Berthold, M. R. KNIME for Reproducible Cross-Domain Analysis of Life Science Data. J. Biotechnol. 2017, 261, 149–156.*

[4] *Mazanetz, M. P.; Goode, C. H. F.; Chudyk, E. I. Ligand- and Structure-Based Drug Design and Optimization Using KNIME. Curr. Med. Chem. 2019, DOI: 10.2174/0929867326666190409141016.*

example of the latter is 3D-e-Chem,[5,6] which allows, e.g., structure-based bioactivity data mapping of kinase inhibitors or structure-based GPCR–kinase cross-reactivity prediction. Here we address users who aim to learn how to use KNIME for CADD applications as well as users who desire to study central CADD topics without necessarily learning how to code. We report the conversion of the TeachOpenCADD Python pipeline (talktorials T1–T8) to a KNIME workflow pipeline (workflows W1–W8). The KNIME pipeline is publicly available on the KNIME Hub: https://hub.knime.com/volkamerlab/space/TeachOpenCADD (current release: https://doi.org/10.5281/zenodo.3475086).

Methods

KNIME (the Konstanz Information Miner) provides an open-source data analysis, reporting, and integration platform. KNIME enables users to create data workflows, execute selected analysis steps, and check intermediate and final results, models, and interactive views via a graphical user interface. Coding is not required, since the workflows are built up by stringing together small preimplemented code units (nodes) with defined, tested, and thus standardized functionalities, which can be configured with individual settings. In addition, KNIME offers functionalities to design complex workflows in a well-structured way via metanodes that encapsulate parts of a workflow. This work was developed using KNIME version 4.0.0 and uses nodes from the KNIME Analytics Platform, KNIME extensions, and Community extensions by RDKit[3,7] and Vernalis[8] (RSCB PDB Tools).

Results

The TeachOpenCADD KNIME pipeline consists of eight interconnected workflows (W1–W8) in the form of metanodes, each containing one CADD topic. The pipeline is illustrated using the epidermal growth factor receptor (EGFR)[9,10] but can easily be applied to other targets of interest. Topics include how to fetch, filter, and analyze

---

[5] *McGuire, R.; Verhoeven, S.; Vass, M.; Vriend, G.; de Esch, I. J.P.; Lusher, S. J.; Leurs, R.; Ridder, L.; Kooistra, A. J.; Ritschel, T.; de Graaf, C. 3D-e-Chem-VM: Structural Cheminformatics Research Infrastructure in a Freely Available Virtual Machine. J. Chem. Inf. Model. 2017, 57, 115–121.*

[6] *Kooistra, A. J.; Vass, M.; McGuire, R.; Leurs, R.; de Esch, I. J. P.; Vriend, G.; Verhoeven, S.; de Graaf, C. 3D-e-Chem: Structural Cheminformatics Workflows for Computer-Aided Drug Discovery. ChemMedChem 2018, 13, 614–626.*

[7] *RDKit Nodes for KNIME. https://www.knime.com/rdkit (accessed May 15, 2019).*

[8] *Roughley, S. Five Years of the KNIME Vernalis Cheminformatics Community Contribution. Curr. Med. Chem. 2018, DOI: 10.2174/0929867325666180904113616.*

[9] *UniProt Entry for EGFR. https://www.uniprot.org/uniprot/P00533 (accessed May 16, 2019).*

[10] *Chen, J.; Zeng, F.; Forrester, S. J.; Eguchi, S.; Zhang, M.-Z.; Harris, R. C. Expression and Function of the Epidermal GrowthFactor Receptor in Physiology and Disease. Physiol. Rev. 2016, 96, 1025–1069.*

compound data associated with a query target and are briefly described in the following. For a detailed description, we refer the reader to the initial TeachOpen-CADD publication.[1]

The TeachOpenCADD KNIME pipeline offers eight KNIME workflows covering central topics in CADD while using open-source data and KNIME nodes. This figure shows the graphical interface of KNIME, demonstrating the software's visual potential.

First, compound data for the query target EGFR are acquired from the ChEMBL web services[11] (W1)[12] and subsequently filtered for drug-likeness using Lipinski's rule of five (W2). This filtered data set forms the basis for the remaining workflows. Unwanted substructures that potentially cause toxicity or nonspecific assay interactions are detected (W3), and a similarity search for a ligand-based screen with the EGFR inhibitor gefitinib as the query [13] is conducted (W4). Compounds are grouped using a hierarchical clustering algorithm (W5), [14] whereupon the maximum common

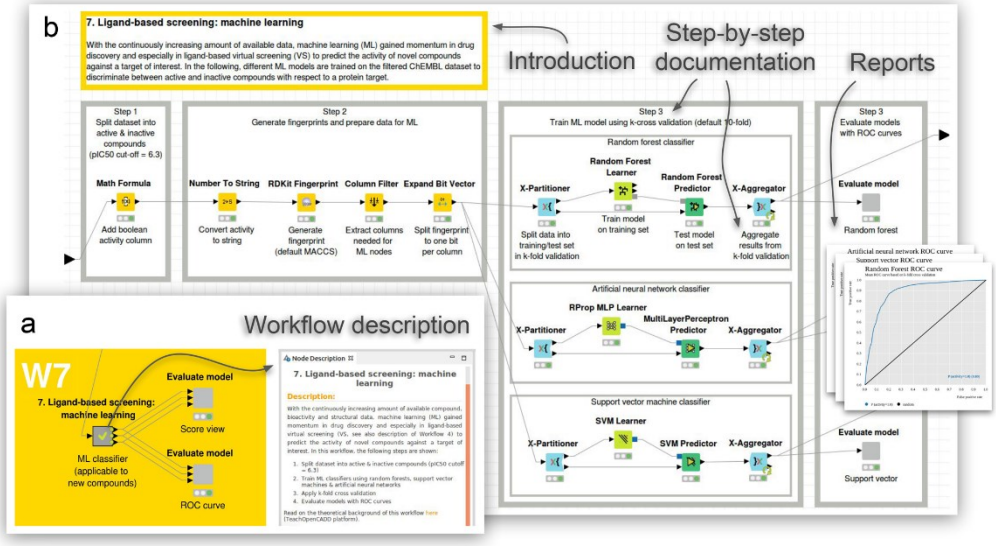
[11] *Gaulton, A.; Bellis, L. J.; Bento, A. P.; Chambers, J.; Davies, M.; Hersey, A.; Light, Y.; McGlinchey, S.; Michalovich, D.; Al-Lazikani, B.; Overington, J. P. ChEMBL: A LargeScale Bioactivity Database for Drug Discovery. Nucleic Acids Res. 2012, 40, D1100–7.*

[12] *Adapting KNIME Workflow Example to Extract Bioactivities for a Target ID. KNIME EXAMPLES Server under 50_Applications/30_RESTful_ChEMBL/03_ChEMBL_Bioactivity_Search (accessed May 18, 2019).*

[13] *DrugBank Entry for Gefitinib. https://www.drugbank.ca/drugs/DB00317 (accessed May 16, 2019).*

[14] *Adapting KNIME Workflow Example to Cluster Molecules Using RDKit Nodes. KNIME EXAMPLES Server Under 99_Community/03_RDKit/01_Clustering (accessed May 24, 2019).*

substructure is detected and visualized for the largest cluster (W6).[15] Additionally, machine learning approaches are employed to build models for active compound prediction (W7).[16] Lastly, ligand–EGFR complexes are fetched from the PDB web services[17] and filtered by criteria such as structure resolution (W8).[18] The last two previously reported talktorials, T9 and T10, were not translated to workflows because of their extensive use of PyMOL, which is currently not supported in KNIME. The workflows can be examined and executed independently from each other or as a pipeline. As shown in the figure below for W7, each workflow is introduced with a brief topic motivation and grouped into multiple steps using gray boxes that contain a step description and all step-associated nodes labeled with task descriptions. Results from intermediate steps (e.g., filtered compound tables) or from final plotting nodes can be viewed interactively and configured easily using the nodes' graphical interface.



Workflow composition shown for workflow W7 (ligand-based screening: machine learning). (a) Each workflow metanode is labeled with a brief topic description and the main workflow steps. (b) The interior of each workflow metanode consists of an introduction, nodes organized inboxes per step, node documentation, and output reports.

[15] *Adapting KNIME Workflow Example Created by Daria Goldmann. KNIME Introduction and Training Session on 2019-01-21 at Volkamer Lab in Berlin: 0×1_Maximum_Common_Substructure (accessed Jan 21, 2019).*

[16] *Adapting KNIME Workflow Example Created by Daria Goldmann. KNIME Introduction and Training Session on 2019-01-21 at Volkamer Lab in Berlin: 0×2_Machine_Learning (accessed Jan 21, 2019).*

[17] *Berman, H. M.; Westbrook, J.; Feng, Z.; Gilliland, G.; Bhat, T.N.; Weissig, H.; Shindyalov, I. N.; Bourne, P. E. The Protein Data Bank. Nucleic Acids Res.2000,28, 235–42.*

[18] *Adapting KNIME Workflow Example to Download and Save PDB Queries using Vernalis Nodes. KNIME EXAMPLES Server Under 99_Community/04_Vernalis/01_PDB_Query_Downloder_and_Save_Locally (accessed May 24, 2019).*

Conclusion

The TeachOpenCADD platform offers learning material on central topics of cheminformatics and structural bioinformatics. In the present work, teaching material was translated from code-based Jupyter Notebooks to KNIME workflows, which have several advantages. KNIME workflows (i) are knitted together from preimplemented nodes with standardized functionalities, (ii) are easy to understand because of the visual representation of their architecture, and (iii) permit a low-threshold entry for nonprogrammers to build customized pipelines. The TeachOpenCADD KNIME pipeline is suitable for self-study training and classroom teaching but can also serve as a starting point for workflows in research projects. TeachOpenCADD is open for contributions and ideas from the community with regard to both Jupyter Notebooks and KNIME workflows.

Author Information

Corresponding Author: *E-mail: andrea.volkamer@charite.de.

ORCID: Dominique Sydow: 0000-0003-4205-8705
Michele Wichmann: 0000-0002-7441-1561
Jaime Rodríguez-Guerra: 0000-0001-8974-1566
Daria Goldmann: 0000-0002-4793-8579
Gregory Landrum: 0000-0001-6279-4481
Andrea Volkamer: 0000-0002-3760-580X

Author Contributions: D.S. and M.W. share first authorship.

Notes: The authors declare no competing financial interest.

Acknowledgements

A.V. and D.S. received funding from the Deutsche Forschungsgemeinschaft (Grant VO 2353/1-1). A.V. received funding from the Bundesministerium für Bildung und Forschung (Grant 031A262C). J.R.-G. received funding from the Stiftung Charité (Einstein BIH Visiting Fellow Project). M.W. received funding from the "SUPPORT für die Lehre" Program (Förderung innovativer Lehrvorhaben) of Freie Universität Berlin.

KNIME Support

This category combines all the helpful contributions made by our COTMs. This could be on the KNIME Forum helping out other KNIME users by proposing solutions to questions, or by sharing thoughts and suggestions for specific use cases. Luckily, our army of helpers is not only present on the KNIME Forum but other places like Facebook or Twitter. If you ever end up needing advice, it's likely that one of our support KNinjas will be on the spot. The category "KNIME Support" features:

- **Stephen Roughley**

  – Principal Scientist *@Vernalis*

- **James Kim**

  – Managing Director *@Zalesia*

- **Kazutaka Watari**

  – n/a

- **Yasue Katsutaka**

  – Senior Research Scientist *@JT CPRI*

- **Emiliano Amendola**

  – Financial Data Analyst *@United Nations*

- **Raffaello Barri**

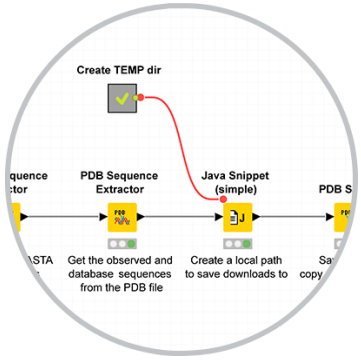
  – Consultant *@BIP*

- **Anil Kumar Sharma**

  – Deputy General Manager Purchase *@Dabur India Limited*

Stephen Roughley was nominated KNIME Contributor of the Month for January 2023. Stephen has been a longstanding member of the KNIME Community (since 2010!), and as the developer of the Vernalis KNIME Nodes extension, he is the author of over 250 nodes used for Cheminformatic, remote data searching, and general utility functions. Apart from that, he also serves on the board for KNIME Data Connect: UK and has been a speaker at several different KNIME events going back to 2013.

Stephen holds a PhD in Chemistry (Organic Synthesis) from the University of Cambridge, UK. He first encountered KNIME in 2010 in his role as a Medicinal Chemist at Vernalis, where he still works as of today, meanwhile in the role of a Principal Scientist. He taught writing nodes himself shortly after, and since then he has become the developer behind the Vernalis community contribution as well as being responsible for the internal nodes and KNIME Server at Vernalis. When he is not using KNIME, he divides his time between playing cornet or trumpet in various brass groups and wind bands around Cambridge, UK, and by night can often be found pursuing his hobby as an amateur lepidopterist.



Visit Stephen's space on the KNIME Community Hub or his profile page in the KNIME Forum (Hub/Forum handle: `s.roughley`). Stephen is also very active behind the Vernalis KNIME account, so make sure to also visit their space on the KNIME Community Hub or their profile page in the KNIME Forum (Hub/Forum handle: `vernalis`).

A Historical Perspective on the Vernalis Community Contribution

The Vernalis KNIME Nodes Extension

Author: Stephen D. Roughley

Editor's Note:

The Vernalis KNIME Nodes extension, developed by the software development team at Vernalis Research Ltd., provides more than 200 nodes serving various purposes. The nodes of this extension are very versatile, and while mostly used for Cheminformatics, e.g., for matched molecular pairs analysis, it also offers nodes for popular resources like PubMed and PDB and general utility functions. Stephen Roughly is the lead KNIME developer at Vernalis and in the following article he will share the history of the Vernalis community contribution, his own career evolution, as well as some useful tips for all striving node developers out there.

Introduction

In this article I describe the history of the Vernalis community contribution from a single node released via our own website to its current state with over 200 nodes. I will also discuss the parallel evolution of my own career from a laboratory-based medicinal chemist to the lead KNIME developer at Vernalis. This is not intended to be a detailed overview of the entire contribution – that has been done previously[19]. Instead, much of the article is based on questions I have been asked over the last 10 years, along with some which formed part of the discussion in the "Fireside Chat" at the KNIME Spring Summit in Berlin in April 2023[20].

The Beginnings of using KNIME at Vernalis & the Origins of the Vernalis Community Contribution

In late 2010 I was working to compile the data for a paper I was writing in which we analyzed the range of chemical reactions used in the published literature by medicinal chemists[21]. As part of that analysis, I had some discussions with a colleague, James

[19] *Roughley, S. D., "Five Years of the KNIME Vernalis Cheminformatics Community Contribution", Curr. Med. Chem., 2020, 27(38), 6495-6522 (DOI: 10.2174/0929867325666180904113616).*

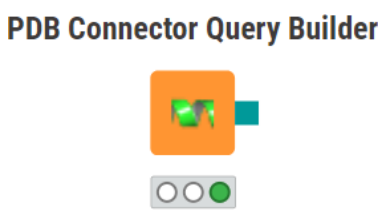
[20] *"KNIME Spring Summit 2023 Presentations", https://www.knime.com/knime-spring-summit-2023-presentations (Accessed 03/Aug/2023).*

[21] *Roughley, S. D. and Jordan, A. M., "The Medicinal Chemist's Toolbox: An Analysis of Reactions Used in the Pursuit of Drug Candidates", J. Med. Chem., 2011, 54(10), 3451-3479 (DOI: 10.1021/jm200187y).*

Davidson, who introduced me to the RDKit[22] cheminformatics toolkit and Python. As part of the analysis, I used RDKit in Python to count the occurrence of various functional groups (arrangements of a small number of atoms within a larger molecule) in the published data. The result was large numbers of Excel workbooks and CSV files containing totals for various functional groups and reaction types for each paper in the dataset we were considering. At this point, James suggested that rather than carrying on in Python, it might be worth me looking at a tool he had recently being evaluating called KNIME. The rest, as the saying goes, is history!

I recently recovered on an old hard disk an early KNIME installation – KNIME 2.7, from September 2010 – almost certainly one of my earliest copies. At that time there was relatively little Cheminformatics capability in KNIME, hence much of the earlier stages of the analysis having been performed in Python/RDKit.

As a Structure-Based Drug Discovery company, Vernalis wanted to access the RCSB PDB Advanced Query[23] web services in order to be able to use KNIME to automate searches of the database for newly published structures relating to our internal drug discovery programs. At the time, the web service API was available as both a SOAP API with WSDL description file and as a RESTful interface. KNIME has a web service node, the *Generic Web Service Client* node, which can use a WSDL definition file[24], however it was not compatible with the WSDL definition from RCSB.org. Eventually, we contracted Dr. David Morley (Enspiral Discovery) to write a node for us, the original *PDB Connector* node which replicated all the available advanced query options available on the RCSB PDB website at the time, and also the related "Custom Reporting" options. David was known to us as he had been one of the original authors of rDock[25] at RiboTargets, one of the predecessor companies which became Vernalis. The plan was to release the node back to the open-source community via a KNIME update site hosted on our own website, which we duly did in December 2012.



The original PDB Connector node is no longer functional due to the shutdown of the remote webservices. Instead, you can use the PDB Connector Query Builder node.

[22] *"RDKit", https://www.rdkit.org/ (Accessed 03/Aug/2023).*

[23] *"Web Services Overview", https://www.rcsb.org/docs/programmatic-access/web-services-overview (Accessed 03/Aug/2023).*

[24] *"Generic Web Service Client - KNIME Community Hub", https://hub.knime.com/knime/extensions/org.knime.features.ext.webservice.client/latest/org.knime.ext.cxf.webservice.client.node.CxfClientNodeFactory/ (Accessed 03/Aug/2023).*

[25] *"rDock: A Fast, Versatile and Open Source Program for Docking Ligands to Proteins and Nucleic Acids", https://rdock.sourceforge.net/ (Accessed 03/Aug/2023).*

From Laboratory Medicinal Chemist to KNIME Developer

After using KNIME for a while, I found several times I was doing the same thing using what is now the "KNIME Jython Scripting (Legacy)" extension[26] – usually taking a local file path or remote URL in an input table and downloading the entire content of the file to a single KNIME data cell. There were two problems with this approach:

1. As a chemist with a relatively limited knowledge of Python I had to look up the required scripting snippet each time I wanted to do this in a new workflow.
2. As became apparent over time, with larger tables or larger text files this became very slow as a result of the need for the node internally to export a KNIME table to a temporary CSV file, read and process it to another temporary CSV file in Python, and reimport back to KNIME each time.

At this point, I decided to try the *Java Snippet* node (this node lives on in KNIME as the *Java Snippet (simple)* node[27]) as I realized this at least did not suffer the conversion overhead. However, there was one significant problem – namely that I had never written a single line of Java. I have been asked multiple times whether it is true that I learnt to write Java in the KNIME *Java Snippet* node. The short answer to this is 'Yes'! This was a particularly steep learning curve, particularly as the node has none of the syntax highlighting or autocompletion features of the newer version (the *Java Snippet* node[28]) to help a learner along, however, I learnt how to read a text file into a String in Java. I was unaware of it then, but this was in Java 1.6 – there are now many easier methods in Java to do this, although they would have required understanding more advanced Java language features which might have put me off altogether. Whilst this approach was faster than the Jython version, it did not solve the other problem – I still had to look up how to do this every time. I'm not sure exactly when the thought occurred to me:

> *"If I had the working pieces of the code in Java then it couldn't be that big a step, to write my own node to do this, and never have to remember how to do it manually ever again."*

---

[26] *"KNIME Jython Scripting (Legacy) - KNIME Community Hub", https://hub.knime.com/knime/extensions/org.knime.features.ext.jython/latest/ (Accessed 03/Aug/2023). In 2010, this was the only Python integration available in KNIME.*

[27] *"Java Snippet (Simple) - KNIME Community Hub", https://hub.knime.com/knime/extensions/org.knime.features.javasnippet/latest/org.knime.ext.sun.nodes.script.JavaScriptingNodeFactory/ (Accessed 03/Aug/2023).*

[28] *"Java Snippet - KNIME Community Hub", https://hub.knime.com/knime/extensions/org.knime.features.javasnippet/latest/org.knime.base.node.jsnippet.JavaSnippetNodeFactory/ (Accessed 03/Aug/2023).*

Programming Background

Although I had no experience of working in Java, I was not entirely new to programming. I had a ZX Spectrum as a child, and an uncle taught me the beginnings of Spectrum BASIC. I soon got bored of the various games and decided that writing programs was far more interesting. The ZX Spectrum was a great machine to learn on, as it was difficult to type something that wasn't syntactically correct due to the system whereby the keyboard would map each key to a couple of keywords when that was the expected input. Then I began to explore the effect of changing the values held in various RAM addresses (you could change what was shown on the display in this way, along with various other things). When this became boring, I bought myself a copy of "The Complete Spectrum ROM Disassembly"[29] and by reading it taught myself Z80 assembly language and learnt a lot about the inner workings of the ZX Spectrum.

At university there was a six-week FORTRAN77 course for scientists, for which I have a certificate from Dr. Frank King. Many of the undergraduate theoretical chemistry practical classes at the time relied heavily on being able to write FORTRAN77, and so I used this regularly for a number of years. I then learnt some Excel VBA to automate some tasks over several years at Ribotargets (subsequently Vernalis) and also wrote and maintained few internal web pages using HTML and JavaScript – in a text editor.

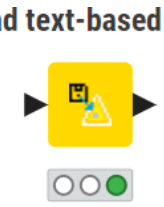
I am often asked whether my non-computer science background is a disadvantage. Clearly there are a lot of things that I had to learn as I went along that I might have understood before I started, and a lot of mistakes that I made that I would hopefully not have made if I had come from a computer science/software engineering background, however, domain knowledge is also important. I had plenty of that and knew what problems I wanted to solve. As I became more familiar with KNIME, I started to realize that the list of problems I could tackle was potentially far bigger than I first realized. If I had never worked in a chemistry laboratory and tried to grapple with designing or organizing and analyzing the data from a chemical library with several hundred members for example, I would not have understood the problems involved.

My First KNIME Node

I wrote my first node by following the online developer guide on the KNIME website[30]. The first time I tried, I failed – completely! I simply did not have even the slightest

[29] *Ian Logan and Frank O'Hara, "The Complete Spectrum ROM Disassembly", Melbourne House (Publishers) Ltd, Richmond (UK), April 2003.*

[30] *"Create a New Java based KNIME Extension", https://docs.knime.com/latest/analytics_platform_new_node_quickstart_guide/index.html (Accessed 03/Aug/2023). In 2012, much less of the KNIME source code was available online, and the original documentation when into considerably more detail than the current version.*

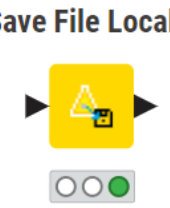


The Load text-based files node. The first node which became part of our initial re-release of our community contribution.

beginnings of an understanding, with my very basic Java knowledge, of what I was doing, and so I initially decided that it was too complicated and carried on as before, splitting my time between laboratory work and handling data – usually relating to my laboratory work – in KNIME. After a while I decided I would give it another go, and this time I would try to "walk" (i.e., create a node that did the same number binning exercise as in the documentation) before I tried to "run" (i.e., create a node that did something that I wanted it to do) and this time I got there – I still clearly remember the mixture of surprise and excitement when my node actually appeared in the *Node Repository* for the first time. This first node was the *Load text-based files* node[31] which became part of our initial re-release of our community contribution.

The Next Nodes

For a while, not very much happened after I wrote my first node – I had solved my problem. Eventually I realized that as well as loading text to a KNIME cell, I also needed sometimes to reverse the process, and write text out from a table cell to a file, and so this became my second node, the which also eventually became part of our initial release. In its initial form, this node was relatively simple – it took one string column containing the paths to save the files to, and a second string column with the content to save, did the saving, and added a column to indicate whether each file was written successfully or not. This node was more complicated than the corresponding reader node – all that had to do was try to read the file and put the content into the cell if it succeeded, whereas this node had to figure if the parent folder existed, and if not create it, and if it succeeded in doing so, then try to write the file and finally indicate whether that succeeded. Later I added more options until the node arrived in its current form. At the time there was no file handling framework in KNIME[32], and even the `knime://` URL protocol was not widely available[33]. Both of these nodes have been



The Save File Locally node. My second node which also eventually became part of our initial release.

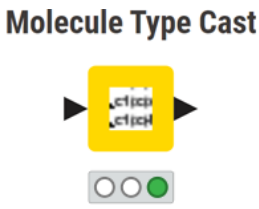
[31] *"Load text-based files - KNIME Community Hub", https://hub.knime.com/vernalis/extensions/com.vernalis.knime.feature/latest/com.vernalis.nodes.io.txt.LoadTxtNodeFactory/ (Accessed 03/Aug/2023).*

[32] *"KNIME File Handling Guide", https://docs.knime.com/latest/analytics_platform_file_handling_guide/index.html (Accessed 03/Aug/2023).*

[33] *Support for the knime:// URL protocol was first introduced in KNIME 2.6, as determined by the introduction of the ResolverUtil class, which is responsible for handling such URLs – see*

updated to handle this latter feature, but do not yet have the capability to use the newer File Handling ports and associated features.

Soon, I realized that KNIME had a PDB cell type. The MOE nodes (MOE Extension for KNIME) provided by CCG (Chemical Computing Group)[34] offered a way to take the unique 4-character structure ID which our *PDB Connector* node returned from the RCSB PDB database and load it to a PDB Cell which could then be used by other CCG nodes. However, this required a MOE license token, and it seemed like waste to have to wait for a token to become available just to read the file – particularly as I could already read the file using my own nodes, but just could not convert them to the required PDB Cell type.



The Molecule Type Cast node converts all cells of a chosen string column into several one of several molecule types, such as Mol2, PDB, SDF, CML, HELM, SLN, Smiles, Smarts, or Rxn.

By now, I had begun to appreciate the benefit of reading the KNIME source code, and so when I thought that I could possibly write adapted versions of my existing nodes to do this for me if I could find out how to create a PDB Cell I decided the obvious starting point was to see how the *Molecule Type Cast* node worked[35]. This actually was more help than I had expected – what I discovered was that the node actually had all the code to "type cast" to the PDB cell type as well as the visible molecule types, but that code had all been commented out. At this point, I emailed Thorsten Meinl to ask why this was the case, and "could it be made available as it would be quite useful?". Time has lost the answer to the first part, but the answer to the second part was that of course it could but came with the caveat that we would have to wait for the next release. Another lesson I learnt about KNIME – if in doubt ask! In the meantime, I needed the functionality now, and so the *PDB Loader*[36] and *PDB*

---

https://github.com/knime/knime-core/blob/master/org.knime.core/src/eclipse/org/knime/core/util/pathresolve/ResolverUtil.java (Accessed 05/Aug/2023). However, the first public release announcement mentioning it is at "Changelog v2.8.0 – KNIME", https://www.knime.com/changelog-v280 (Accessed 03/Aug/2023), and it is unclear how many nodes used it, if any, in KNIME 2.6, and when it became a public feature.

[34] *"MOE Extensions for KNIME - KNIME Community Hub", https://hub.knime.com/guido_kirsten/extensions/com.chemcomp/latest/ (Accessed 03/Aug/2023).*

[35] *"Molecule Type Cast - KNIME Community Hub", https://hub.knime.com/knime/extensions/org.knime.features.chem.types/latest/org.knime.chem.base.node.converter.parser.MolParserNodeFactory/ (Accessed 03/Aug/2023).*

[36] *"PDB Loader - KNIME Community Hub", https://hub.knime.com/vernalis/extensions/com.vernalis.knime.feature/latest/com.vernalis.nodes.io.pdb.loadlocal.LocalPDBLoadNodeFactory2 (Accessed 03/Aug/2023).*

Saver[37] nodes were developed, based on the existing nodes. Now I had another problem: how to actually be able to use the PDB Cell type in my code? At which point my Java and Eclipse knowledge started to expand to plugin dependencies, with some help from Thorsten who pointed me in the right direction again. Those nodes handle

The PDB Loader and PDB Saver nodes. The PDB Load node loads local or remote PDB files specified by a valid full file path or URL into a table. The PDB Saver node saves a copy of the PDB files held in a table column to filepaths held in a second column.

local PDB files using a full file path, as opposed to those hosted remotely on the RCSB PDB servers, which was also something not available to us by any other route, but then I also added a version to download from the RCSB PDB server using just the 4-character structure ID (The *PDB Downloader*[38] and *PDB Downloader (Source)*[39] nodes).

Becoming a Trusted Community Contribution

As already mentioned, we had released our original node via an update site hosted on our own corporate website, however, we had already realized that this was difficult to update when a minor patch needed to be applied. We also had no way of knowing whether anyone had ever found or used it! Early in 2013 we started talking to KNIME about becoming an official community contribution, which had the advantages of increased visibility, code hosting on KNIME's own SVN server, automated building when updated, and support from the KNIME team when we had questions. Our eventual re-release as an official community contribution was timed to coincide with a presentation I gave at the London KNIME User Day UK 2013[40].

[37] *"PDB Saver - KNIME Community Hub", https://hub.knime.com/vernalis/extensions/com.vernalis.knime.feature/latest/com.vernalis.nodes.io.pdb.savelocal.SavePDBLocalNodeFactory/ (Accessed 03/Aug/2023).*

[38] *"PDB Downloader - KNIME Community Hub", https://hub.knime.com/vernalis/extensions/com.vernalis.knime.feature/latest/com.vernalis.rcsb.io.nodes.manip.RCSBmultiDownload2NodeFactory/ (Accessed 03/Aug/2023).*

[39] *"PDB Downloader (Source) - KNIME Community Hub", https://hub.knime.com/vernalis/extensions/com.vernalis.knime.feature/latest/com.vernalis.rcsb.io.nodes.source.RCSBsDownload2NodeFactory/ (Accessed 03/Aug/2023).*

[40] *"KNIME User Day UK 2013 News", https://www.knime.com/knime-user-day-uk-2013-news (Accessed 03/Aug/2023).*

The trusted status comes with various requirements[41] – good test case coverage and SONAR code quality score, guarantees around ongoing maintenance and communication (i.e., typically a response within two working days to contact) and a requirement to follow the "KNIME Noding Guidelines" [42] (which make various guarantees around not breaking backwards compatibility when new or updated versions of nodes are released) all of which combined make the code suitable for use in a production environment. More recently the need to pass a security vulnerability scan has also been introduced. These are all good software engineering practices which I knew nothing about when I first started trying to write my own nodes, so one obvious advantage is that I have had to learn about them. In turn this means that I write better and more reliable code as a result. Another advantage is access to the KNIME community contributions team who have answered many questions over the last 10 years.

Of course, one disadvantage is that communicating with users when they have bug reports or questions about our nodes requires some ongoing resource commitment, but generally that has been quite a small part of the process and does come with the balancing advantage that if it leads to a bug being fixed or a new or enhanced feature being added to a node, then we also benefit as a result.

Training

Possibly the most important training I have done was to attend an on-site KNIME Developer Training course in February 2014 at the former KNIME office in Zurich Technopark. Learning directly from the developers of KNIME themselves was an extremely valuable experience. I still refer to my course handbook regularly. During the course I also learnt many features of the Eclipse SDK which also make the task of writing KNIME nodes (and any other Java code in Eclipse) much easier.

I also learnt about CI/CD – Continuous Integration/Continuous Deployment – the process whereby code is built automatically on a server and deployed to an update site seamlessly once tests have been passed. Initially, this was through seeing how the community build works on a Jenkins server. I then emulated that process internally at Vernalis. Previously the nodes were built manually, and the resulting jar file had to be copied into the "dropins" folder in any KNIME installation that was to use them, which was not very sustainable or reliable as a process. More recently, Buckminster (the tool that used to be used by the automated build process to gather together the

[41] *"Community Extensions (Trusted)", https://www.knime.com/trusted-community-contributions (Accessed 03/Aug/2023).*

[42] *"KNIME Noding Guidelines (v2.5), November 2015", https://www.knime.com/sites/default/files/inline-images/noding_guidelines.pdf (Accessed 03/Aug/2023).*

dependencies and then build the update site) became obsolete[43], and the build process had to be migrated to use a different tool – Apache Maven™[44]. Gabriel Einsdorf from KNIME gave some assistance with the migration process, based on an existing template[45] for a new KNIME project using Maven™, and in turn I was able to help by providing step-by-step documentation of the migration process as I applied it[46]. That process was then applied to our internal nodes and works much more smoothly than the Jenkins/Buckminster approach.

Java itself is now a regularly but rapidly evolving universe with regular updates bringing new language features, but when I started developing KNIME nodes we were using Java 1.6 and evolution was slow and irregular[47]. Java 1.7 followed in July 2011, with some useful new features (KNIME adopted Java 7 from version 2.7[48]), and then Java 1.8 ('Java 8'; March 2014) brought many new and useful features. KNIME is now using the current LTS (Long Term Support) release, Java 17 (Released September 2021), but the next LTS, Java 21 is due for release in September 2023.

Releasing Open-Source Code in a Pharmaceutical Company Environment

We were already members of an informal pre-competitive group called the "UK KNIME Industrial User Group" which met for an afternoon prior to the UK QSAR meetings which happen twice yearly[49]. Eli Lilly, the creators of the Erlwood community contribution[50] were also members, and so there was some precedent for what we wanted to do. Once we had management approval in principle, we consulted our legal department and

---

[43] *"Buckminster", https://projects.eclipse.org/build-technology/buckminster (Accessed 03/Aug/2023). See also "Eclipse Buckminster Component Assembly – projects.eclipse.org", https://projects.eclipse.org/projects/tools.buckminster/governance (Accessed 05/Aug/2023). The last project release was version 1.7.0 (Released 16/Jul/2015) which was only capable of building projects using Java 1.7 or earlier. The project was finally Archived 20/Feb/2019.*

[44] *"Welcome to Apache Maven", https://maven.apache.org/ (Accessed 03/Aug/2023).*

[45] *"Template for community extension repositories", https://github.com/knime-community/community-repository-template (Accessed 03/Aug/2023).*

[46] *"Converting an existing multi-feature/multi-plugin build from Buckminster", https://github.com/knime-community/community-repository-template/blob/master/Converting_from_Buckminster.md (Accessed 03/Aug/2023).*

[47] *"Java version history - Wikipedia", https://en.wikipedia.org/wiki/Java_version_history (Accessed 03/Aug/2023).*

[48] *"What's New in KNIME 2.7", https://www.knime.com/whats-new-in-knime-27 (Accessed 03/Aug/2023).*

[49] *"Meetings - UK QSAR and Cheminformatics Group", https://ukqsar.org/index.php/meetings/ (Accessed 03/Aug/2023).*

[50] *"Erlwood Cheminformatics Nodes for KNIME (trusteded extension)", https://www.knime.com/community/erlwood (Accessed 03/Aug/2023).*

explained we were using the same license (GPL 3.0) as other community contributions, including the Erlwood contribution. Following that, releases are treated as any other publication. Having release KNIME nodes, we have subsequently released other source code and laboratory equipment designs through GitHub[51] and Thingiverse[52].

Our philosophy has always been to release nodes when we believed that they might be of use to others in the community. That means many of the nodes are actually of very general use, and I am aware of users from a wide variety of fields using some of our nodes[53]. Sometimes we have been surprised too – for example, I wrote a very rough-and-ready pair of nodes to try to get to the bottom of a problem with an internal node – the *Benchmark Start* and *Benchmark End* nodes – and followed them up with variants with 2- and 3-ports, and a further variant which also include monitoring of the memory usage by the KNIME Analytics Platform. I had no intention of ever releasing those nodes publicly, but very shortly after I wrote them, I saw a post on the KNIME community forum where someone was looking for exactly the feature I had written. We were able to release the nodes initially within a few days[54].

The Benchmark Start and Benchmark End nodes. The Benchmark Start node provides the start of a timing block for monitoring performance and needs to be paired with the corresponding Benchmark End node.

[51] *"Vernalis Research (GitHub page)", https://github.com/vernalis (Accessed 03/Aug/2023).*

[52] *"Vernalis Limited (Thingiverse page)", https://www.thingiverse.com/vernalis_limited/designs (Accessed 03/Aug/2023).*

[53] *Some of the diversity of users can be seen in the research categories for references citing Ref. 1 (See "Dimensions - Five Years of the KNIME Vernalis Cheminformatics Community Contribution", https://badge.dimensions.ai/details/id/pub.1106677964/categories (accessed 03/Aug/2023)). Around 72% of citations are from the chemical, biomedical or clinical sciences, with another 17% from Information and Computer Science or Engineering, while the remaining includes "Built Environment and design" and "Commerce, Management and Tourism services".*

[54] *"Timing Workflows - KNIME Extensions - KNIME Community Forum", https://forum.knime.com/t/timing-workflows/7808 (Accessed 03-Aug-2023). The initial topic was posted on 6th May 2015, and we had released the nodes to our nightly build on 12th May 2015.*

A Vernalis Cheminformatics Toolkit?

Short answer: no! There are plenty of well-developed and maintained cheminformatics toolkits available in KNIME (CDK[55], RDKit[22,56], Chemaxon[57,58]) already, so when we need a toolkit for one of our nodes, we use one of the existing ones. Perhaps the nearest we have come to writing our own is ironically the "Speedy SMILES" family of nodes[59]. These nodes were designed with large datasets in mind to allow some pre-processing based on analysis of the raw SMILES[60] strings prior to the computationally costly conversion to a toolkit's internal representation. Initially this was limited to two nodes. The first calculated the Heavy Atom Count of a molecule (the number of "heavy", i.e., non-hydrogen atoms), which used functionality we already had as part of our Matched Molecular Pairs (MMPs) nodes. The second was desalting, which was something we needed for the MMP node inputs and were previously doing either using RDKit or with a string split operation within a Java Snippet. Another use which followed soon afterwards was to remove structures containing metal ions or other "non-organic subset" atoms, as many PDB structures contain ligands which are metal or other non-organic ions which either perform a structural roll or are present as part of the crystallization mixture but are not of interest to medicinal chemists. Some of the nodes now perform quite complex tasks so the boundary may have become a little blurred, but they always operate on the raw string without any other internal representation.

Future Directions

It's always difficult to predict the future, and public releases need to be fitted in around other internal work. We have recently added some further internal nodes relating to Binary Objects which may well join the existing public ones, and we have also overhauled the internal version of our Benchmarking nodes. The new version has many nice new features: more monitoring options in addition to memory usage, a single start

---

[55] *"CDK Nodes for KNIME (trusted extension)", https://www.knime.com/community/cdk (Accessed 03/Aug/2023).*

[56] *"RDKit Nodes for KNIME (trusted extension)", https://www.knime.com/rdkit (Accessed 03/Aug/2023).*

[57] *"ChemAxon/Infocom JChem Extensions Feature - KNIME Community Hub", https://hub.knime.com/infocom/extensions/jp.co.infocom.cheminfo.jchem.feature/latest/ (Accessed 03/Aug/2023).*

[58] *"ChemAxon/Infocom Marvin Extensions Feature - KNIME Community Hub", https://hub.knime.com/infocom/extensions/jp.co.infocom.cheminfo.marvin.feature/latest/ (Accessed 03/Aug/2023).*

[59] *"Nodes: "speedy smiles" - Vernalis - KNIME Community Hub", https://hub.knime.com/search?q=%22speedy%20smiles%22&type=Node&tag=Vernalis&sort=best (Accessed 03/Aug/2023).*

[60] *"Daylight Theory: SMILES", https://www.daylight.com/dayhtml/doc/theory/theory.smiles.html (Accessed 03/Aug/2023). SMILES (Simplified Molecular Input Line Entry System) is a compact text-based representation of molecular structure.*

node and a single end node, each of which can have one or more ports of any type (i.e., they are configurable from the menu on the bottom left of the node), and there is also the option to write out the monitoring table directly to a csv file in addition to making it available at an output port on the loop end. This feature is particularly useful when trying to debug what state KNIME was in if a workflow causes the entire analytics platform or server executor to crash. In order to do that, we now have our first node with an optional file system connection port. No doubt in time some of our other file-handling nodes will also get an upgrade to use path variables and columns, particularly where this eases our own internal use of the nodes.

Obviously, we have some nodes internally which we could never release – mostly because they would be no use to anyone outside of Vernalis, as they access our own internal systems or wrap 3rd party commercial tools which we are not at liberty to redistribute. We have some nodes which embed proprietary algorithms or tools which we would not release. However, in most cases it is what we do with the nodes which is proprietary rather than the actual nodes.

What Advice would you Give to Someone Wanting to Write their own Nodes?

Firstly, and this may sound obvious, check that no one else has already written it! Then I would recommend reading the node development instructions[30] and the noding guidelines[42]. If you can, attend a KNIME developer training course. As the KNIME core[61] and the community extensions[62,63,64] are all open source, it is worth familiarizing yourself with the code in those repositories to see how others have written their nodes. The KNIME community is generally a friendly place, so do not be afraid to ask questions on the forum in the "Node Development" category[65].

There are two books which I have found to be particularly helpful: The first is "Effective Java" by Joshua Bloch[66]. This is an excellent book which has helped me to improve my Java code considerably. Another useful book is "Eclipse Plug-in Development:

---

[61] *"KNIME (GitHub page)", https://github.com/knime/ (Accessed 03/Aug/2023). The source code for many parts of the KNIME Analytics Platform can be found in various repositories here.*

[62] *The source code for the Vernalis community extension can be found at "Vernalis KNIME Nodes (GitHub page)", https://github.com/vernalis/vernalis-knime-nodes (Accessed 03/Aug/2023).*

[63] *The source code for the RDKit community extension can be found at "KNIME RDKit (GitHub page)", https://github.com/rdkit/knime-rdkit/ (Accessed 03/Aug/2023).*

[64] *The source code for the CDK community extension can be found at "CDK Nodes for KNIME (GitHub page)", https://github.com/cdk/nodes4knime (Accessed 03/Aug/2023).*

[65] *"Latest Node Development Topics - KNIME Community Forum", https://forum.knime.com/c/node-development/21 (Accessed 03/Aug/2023).*

[66] *Joshua Bloch, "Effective Java", 3rd Edition, Addison-Wesley Professional, December 2017.*

Beginners Guide" by Alex Blewitt[67]. This book has some very useful chapters if you are not very familiar with Eclipse and the features/plugins framework (KNIME extensions are comprised of Eclipse "features", which in turn bundle one or more "plugins") and in particular chapters on creating Eclipse extensions using Apache Maven™.

Finally, I would suggest installing a code quality tool into your Eclipse IDE, such as the powerful and freely available SonarLint from Sonar[68]. This tool, and others like it, will highlight many potential issues with code in the editor in real time, providing explanations as to what the issue is and often suggestions as to how to resolve it, similarly to how a spelling and grammar checker works in word processing software.

When would you Recommend Someone considers Writing their own Nodes?

It's a good question, to which there is no definitive answer. Firstly, you need to consider the barriers to doing so. If you already have internal KNIME nodes and developer(s) then the barrier is relatively low compared to setting everything up from scratch. Then I would consider how difficult it is what you want to achieve without writing a node, and how often are you likely to want to do it?

Unlike in 2010, there are now several easy ways to re-use code in KNIME. For example, many of the scripting integrations allow the saving and re-use of script templates and there is also the ability to share (with yourself or publicly) metanodes and components (which could be considered a pseudo-node) via the KNIME Community Hub. Are those routes sufficient for your needs?

When I spoke at the KNIME User Day UK in London in June 2013[40] I concluded my talk with a slide which listed the following suggestions:

- If you often spend time asking "Which workflow was the Java/Python snippet node that did <<insert task here>> in?", then write a new node. In this case, the algorithm is already written, and it is normally relatively straightforward to convert it into a new node.
- Highly complex metanodes, loops (particularly nested loops) and multiple scripting nodes performing slow-to-run operations are good candidates for conversion to a new node.

[67] *Alex Blewitt, "Eclipse Plug-in Development: Beginner's Guide", 2nd Edition, Packt Publishing, August 2016.*

[68] *"Eclipse Linter for your IDE with SonarLint - Sonar", https://www.sonarsource.com/products/sonarlint/features/eclipse/ (Accessed 03/Aug/2023).*

- "Shoe-horning" a solution to a task into several nodes doing things they were not really designed for (an example of this is our *List Folders* node[69], which was written to replace many instances of a *File Reader* node pointed at a directory, followed by a *Java Snippet Row Filter* node to only keep sub-directories, and a further *Java Snippet* node to find the last modified date, directory name, or parent directory).
- Finally, when all your data is in KNIME, and you find yourself having to write it out to a file to either perform some manual steps or to import and manipulate it in another piece of software. This is potentially the most difficult case, and may not always be possible, depending on the nature of the steps or other software being used.

Although the KNIME ecosystem and my knowledge have both developed considerably in the intervening 10 years, I still think that those are a good basis.

Acknowledgements

There are many people who have helped along this journey, and it would be impossible to list them all here. However, a few special mentions are merited: Firstly, none of this would have happened if James Davidson had not introduced me to KNIME. David Morley wrote the original Vernalis KNIME node and gave advice on its maintenance. Many colleagues, past and present, have provided feedback on features of new nodes or missing functionality which have resulted in new nodes. A number of people at KNIME have also provided a lot of support, in particular Thorsten Meinl and Bernd Wiswedel in the early years, and more recently Gabriel Einsdorf and Steffen Fissler. Finally, you – the KNIME community has provided much useful feedback in the form of comments, questions, bug reports, feature requests and encouraging comments.

[69] *"List Folders - KNIME Community Hub", https://hub.knime.com/vernalis/extensions/com.vernalis.knime.feature/latest/com.vernalis.nodes.io.listdirs2.ListDirs2NodeFactory/ (Accessed 03/Aug/2023).*

James Kim was nominated KNIME Contributor of the Month for April 2023. He was awarded for being a true ambassador for the Korean-speaking analytics community! He serves on the board for the KNIME Data Connect: Korea, and along with his team members he has translated many resources into Korean, maintains a blog and a YouTube channel, and even published a book in Korean!

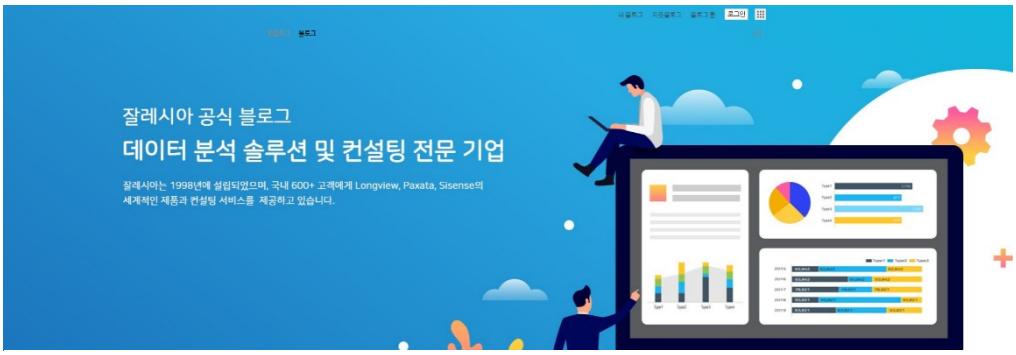
James has 25 years of experience in the field of Data Analytics solutions and consulting and is currently Managing Director at Zalesia. In the past years, he has traversed various roles, beginning as a programmer and later moving to sales, organizational management, partner relationship, and new business building. James holds a Bachelor in Computer Science and a Master in Business Administration.

Visit James' space on the KNIME Community Hub or his profile page in the KNIME Forum (Hub/Forum handle: `webdb`).

How to Upskill a Team in Record Speed

Zalesia's Story of becoming KNIME Emerging Partner of the Year 2023

Author: KNIME Team



Zalesia's Official Blog – a company specializing in data analysis solutions and consulting.

Our KNIME Partners are a critical part of the KNIME ecosystem. They are true ambassadors of the KNIME brand and trusted advisors to our joint customers. They have a deep understanding not only of the KNIME Software but also of Data Science, Machine Learning, or Artificial Intelligence, and provide expertise within their respective markets. Hence, through the partnership network, KNIME can bring analytics and AI/ML capabilities to virtually any line of business or technical department.

Today, the KNIME Partner network features partners across 60+ countries with hundreds of consulting, cloud, systems integrator, and resell partners, technology partners, local professional services companies, and many others. An overview of all our current partners is available at https://www.knime.com/partners/finder.

To honor our valuable KNIME Partner we have come up with the KNIME Partner Awards which we confer each year at the KNIME Spring Summit. The organizations recognized with these awards represent a deep commitment to leveraging KNIME to develop innovative advanced data science projects.

In this article, we would like to honor one KNIME Partner in particular, Zalesia, and the man initiating the massive upskilling efforts of the Zalesia Team, James Kim.

Zalesia was awarded Emerging Partner of the Year 2023. Within just a couple of months, the whole Zalesia team got KNIME certified and became KNIME Ready Partner, so we thought, such dedication must be awarded.

> ***Note.*** Watch the Partner of the Year Ceremony 2023 on our KNIMETV YouTube Channel.

Zalesia is a Data Analytics Solution and Consulting company founded in 1998 and based in Seoul, Korea. They provide everything you need to get a Data Analytics system – software, implementation service, consulting and education. KNIME is just one of many world-class products Zalesia has already supplied to their 600+ customers.

> ***Note.*** Visit Zalesia's website for more information.

James Kim and Neslihan Dönder at the Partner of the Year Ceremony 2023 at the KNIME Spring Summit.

But what is the KNIME Partner Award Program and how to become a KNIME Partner to begin with? Let's start from scratch and later hear a bit more from James Kim and Zalesia's upskilling journey.

The KNIME Partner Award Program

The KNIME Partner Award Program recognizes KNIME Partners based on overall revenue performance, growth with KNIME's enterprise products, excellent customer service, and delivering quality advanced data science projects. Rewarded partners distinguish themselves by a strong network of KNIME customers, strengthening the KNIME community, conducting upskilling programs with existing and potential KNIME customers, and driving KNIME awareness in their region.

The winners of the KNIME Partner Award receive a unique KNIME trophy, materials to support a press release which are published on the KNIME website and social media, a digital badge/ribbon for your website and a seat on the KNIME Partner Board.

In 2023, there were three awards: The *Partner of the Year Award*, the *Emerging Partner of the Year Award*, and the *Partner Innovation Award*. The awards are handed out each year at KNIME Spring Summit. Let's explain what each award is and how it differs from the others.

Partner of the Year Award

The *Partner of the Year Award* recognizes outstanding performance by partners across business growth, commercial activity, and delivery of quality projects. During the nomination process, activities related to strengthening KNIME awareness in the focus region, empowering customers with KNIME Software, driving customer success, and contributions to the KNIME community will be considered.

Emerging Partner of the Year Award

The *Emerging Partner of the Year Award* is dedicated to partners who have recently joined the KNIME Partner Program, demonstrated consistent expertise with KNIME's products, and commitment to outstanding business growth and performance. It recognizes contributions to KNIME's product innovation and impact on KNIME's community.

Innovation Partner of the Year Award

This *Innovation Partner of the Year Award* recognizes partners who deliver innovative projects and share best practices and information by publishing blog articles, KNIME Innovation Notes, Success Stories, and KNIME Verified Components (templates provided by KNIME). These resources provide benefits to the KNIME, data science, and machine learning communities by showing how cutting-edge KNIME technology can reveal actionable business insights.

Note. For more information about the KNIME Partner Award Program and FAQ visit https://www.knime.com/partners/partner-award-program.

See existing Success Stories and Innovation Notes at https://www.knime.com/solutions.

Becoming a KNIME Partner

One thing that is crucial for being considered for a KNIME Partner Award is to be a KNIME Partner. Let's quickly talk about how to become a KNIME Partner and why it's beneficial.

The KNIME partner program is open to businesses of all sizes, from boutique consulting companies to large system integrators. The program is built to provide

broad industry domain knowledge and expertise to understand customers' specific business challenges. Members of the partner program expand their portfolio with KNIME Software to provide the analytics at scale their customers need.

All you need to do to become a KNIME Partner is to visit https://www.knime.com/partners/new and fill out the application form. Someone from our team will then reach out to you as soon as we have reviewed and qualified your application.

Benefits of KNIME Partners

Applying to become a KNIME Partner is fairly easy, but what are the benefits? The program offers a variety of benefits to our partners, including:

- Exclusive access to materials via the Partner Portal, including education and technical enablement materials, marketing tools and sales materials, and customer success materials
- Free KNIME Business Hub license for easy planning of demos plus internal, non-commercial use
- Lead/customer referral of KNIME leads for Business Hub and consulting services
- Lead registration rewards: including commissions credit, sales and tech support, sales process ownership
- Dedicated sales support including sales enablement resources, and deal shadowing
- Market development activities, joint PR, self-service demand generation resources
- Enhanced tiered commission plan to reward sourcing new business, assisting to generate opportunities, providing technical support, and securing customer retention
- A tiered program with badges showing achievements as partners move up through the program

Note. As KNIME continues to grow and foster its partner network, it will continue to improve and update its partner program to meet the evolving needs of its partners, customers, and the market.

How James has Driven Upskilling at Zalesia

Zalesia first reached out to our Partner Team in October 2021 and became an official KNIME Partner shortly after. Roughly two months later, Zalesia already became a KNIME Ready Partner. Being KNIME Ready means that the partner has proven to KNIME that they have a team with adequate technical knowledge of our offerings. This

team will need to consist of at least two KNIME Certified Users and at least one KNIME Certified Server Administrator. It took the Zalesia team only 6-8 weeks to get the entire data team with 70+ people KNIME certified. That's beyond impressive!

The driving force behind all this is James Kim, Managing Director of Zalesia. James is responsible for finding new solutions and making go-to strategies. In this regard, he is responsible for building a new consulting team specializing in solutions and supporting the sales and marketing team for new businesses. His main focus is to find solutions to Data Analytics and support and monitor them until they are profitable.

For 25 years, Zalesia has specialized in Data Analytics solutions and consulting. We have asked James how he encountered KNIME. His response was that he is always searching for new Data Analytics solutions and one day – 2 years ago approx. – he discovered KNIME. He added:

> *"I have reviewed many products and I have come to the conclusion that KNIME is the best solution."*

He then reached out to KNIME's Partner Team and within just a couple of weeks Zalesia did a massive upskilling of the entire data team. That's very impressive so we asked James, how did they do this, and he replied:

> *"With such a well-prepared expert as KNIME is, we were already able to build a KNIME consulting team during the partner negotiations with KNIME. After becoming a KNIME partner, I immediately got everyone to complete the KNIME online trainings, started documentation to support the sales team, and started marketing through social networking services such as YouTube, blogs, or Facebook."*

James also mentions the "well-organized" online curriculum which helped in quick and easy upskilling of the team and KNIME's large online community.

Besides being *Emerging Partner of the Year*, James and his team at Zalesia are also huge ambassadors in the Korean community. They organize local events, translate resources into Korean, maintain a blog (see figure above) and a YouTube channel, and they have even written a book. "Data Preprocessing and Integrated Analysis with KNIME" is the first Korean book about KNIME that we are aware of. It's available for purchase online.

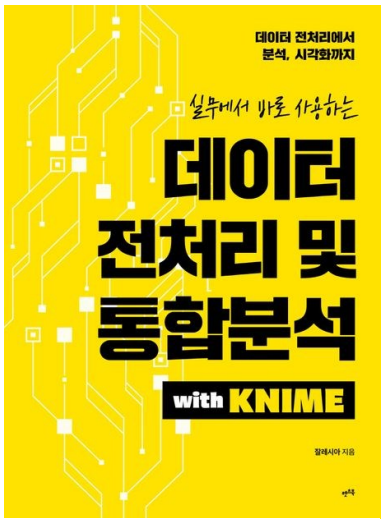
We have asked James to describe what the book is about:

> *"It is a practical guide to Data Science with a variety of information to elevate data analysis using KNIME from a basic level to a more professionally constructed and complex structure. But the main target of this book is beginner to intermediate users."*

Lastly, we have asked James whether he would like to share any tips and tricks for future authors-to-be on how to successfully publish a book. He says:

> *"I think it's better to first decide for which user level the book is intended. As long as you then create the table of contents for the target user, I think you can then get the information needed from both the KNIME documentation and the KNIME community afterwards."*

As the saying goes – where's a will, there's a way – Zalesia has impressively demonstrated that large-scale upskilling in a short amount of time is feasible. It's another great example of what you can achieve with an easy to learn and adopt tool. Nonetheless, it's also thanks to James Kim's vision and determination that has helped pave the way for his team to become expert KNIME users. Keep going, Zalesia!



The "Data Preprocessing and Integrated Analysis with KNIME" book, written by Zalesia.

Kazutaka Watari was nominated KNIME Contributor of the Month for October 2022. He was awarded for being a social media influencer for the KNIME community, ever since he started with KNIME. He is posting daily about his learning progress on social media (Twitter handle: *@watarinko*), sharing what he learned with others. He also provides his solution workflows on the KNIME Community Hub being a true motivational inspiration for others.

Kazutaka is a KNIME user based in Singapore, and his KNIME journey is beyond impressing: From zero to hero in 100 days! How did he accomplish that? He has basically completed all the KNIME challenges: #66daysofdata Challenge, L1 & L2 courses, L1 and L2 Certification Exams, *Just KNIME It!* Challenges, #100DaysOfData Challenge, and the L4-ML course on Machine Learning algorithms. He shares his progress on social media and lets others participate in his learning successes. The reason why? Because he says he is passionate about KNIME and likes to share the feeling with his friends and colleagues, and others via social media.

Visit Kazutaka's space on the KNIME Hub or his profile in the KNIME Forum (Hub/Forum handle: `kwatari`).

Learning Data Science in 100 Days

Follow Kazatuka Watari's #100DaysOfData Journey with KNIME

Author: Roberto Cadili

Kazutaka Watari, best known by many users in the KNIME community by his Twitter handle (@watarinko), is one of the most enthusiastic, vocal, and committed presences on social media. With hundreds of posts, Kazutaka has been exploring, learning and sharing content on a wide range of KNIME capabilities: from simple wrangling operations to machine learning, data apps, and cloud storage connectivity. His KNIME-inspired social media journey started, albeit rather quietly, when he joined in late May 2022 the #66DaysOfData initiative organized by KNIME. He shared daily updates on his work in Japanese, initially targeting his local community.

Kazutaka Watari is sharing his journey of learning KNIME on social media. His Twitter handle is @watarinko.

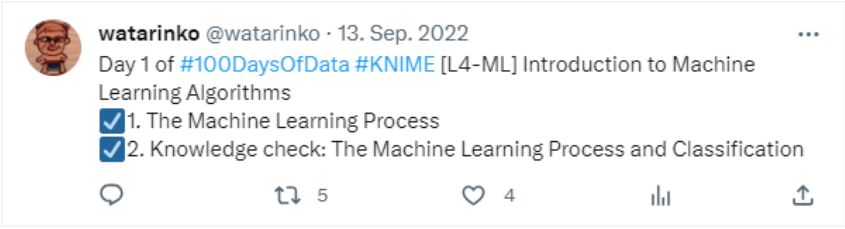
As his KNIME knowledge expanded and workflow building skills consolidated, so did Kazutaka's presence and content on Twitter. He embarked on a successful #100DaysOfData with KNIME project where he documented his learning process one node/workflow at a time. He became increasingly vocal, sharing on a daily basis hacks,

tips and impressions. He also switched from Japanese to English in order to engage with the global KNIME community.

Curious to know more about the tireless commitment of this KNIME social media influencer? Here is a collection of the five most representative milestones in Kazutaka's #100DaysOfData learning journey from zero to data science hero.

Machine Learning for All

Machine learning-powered applications are everywhere. From the autocorrect on smartphones, to movie recommendation engines and artificial assistants, they have become an essential part of our lives and contribute substantially to make our daily tasks or routines easier. To such an extent that we often take them for granted and stop wondering how they even work.



Kazutaka's first post of his journey to complete #100DaysOfData with KNIME. He started by doing the L4-ML Introduction to Machine Learning Algorithms self-paced course.

Day 9 of Kazutaka's journey to complete #100DaysOfData with KNIME.

The truth is that there wouldn't be any of the above without a crystal clear understanding of machine learning algorithms and learning paradigms. Kazutaka's journey started right here. Following the structure of the KNIME L4-ML self-paced course, he differentiated between supervised vs. unsupervised learning, and zoomed in on the Decision Tree, Logistic Regression, ensemble models, recommendation engines, and clustering algorithms. For over 30 days, his posts summarized the key takeaways of each lesson, shared solutions to the hands-on exercises, and pointed the community to valuable resources – be it videos, blog posts, etc. – to support learning and facilitate delving into the topic.

If you followed Kazutaka, you surely learned one thing or two about data science, ML algorithms and how to implement them in KNIME.

Note. Check the original posts by @*watarinko*: Day 1 and Day 9 on Twitter.

Data Analytics via Interactive Data Apps

Not everything in data analytics has to do with training and applying ML models. Sometimes we need to communicate to the Management or the Finance Department the results of an ETL process in a compelling way.

For example, we could create a customizable and user-friendly data app to perform scalable and shareable data operations, such as visualization, data import, export, and preview. With developing a data app, the workflow developer has complete control over the interactivity that will be available to the end-user and the complexity of the underlying workflow.

Components are the key building blocks of any data apps, as they can encapsulate and abstract functionality, can have their own configuration dialog, and custom interactive composite views.

While building a simple data app takes a few minutes, mastering the art of creating a complex, bulletproof, and highly interactive one requires a bit of practice. After learning about Machine Learning algorithms, Kazutaka's journey took him to explore the possibilities of data apps. From a step-by-step guide to 11 best practices for component builders, he gathered a collection of valuable resources that helped him to successfully build his, including Widget nodes for interactive filtering, selection, file upload/download, and re-execution.

For example, do you know how to include user-friendly error messages if your component fails? If not, then you should definitely catch up with Kazutaka's Twitter posts starting at Day 40.

Note. Check the original post by @*watarinko*: Day 40 on Twitter.

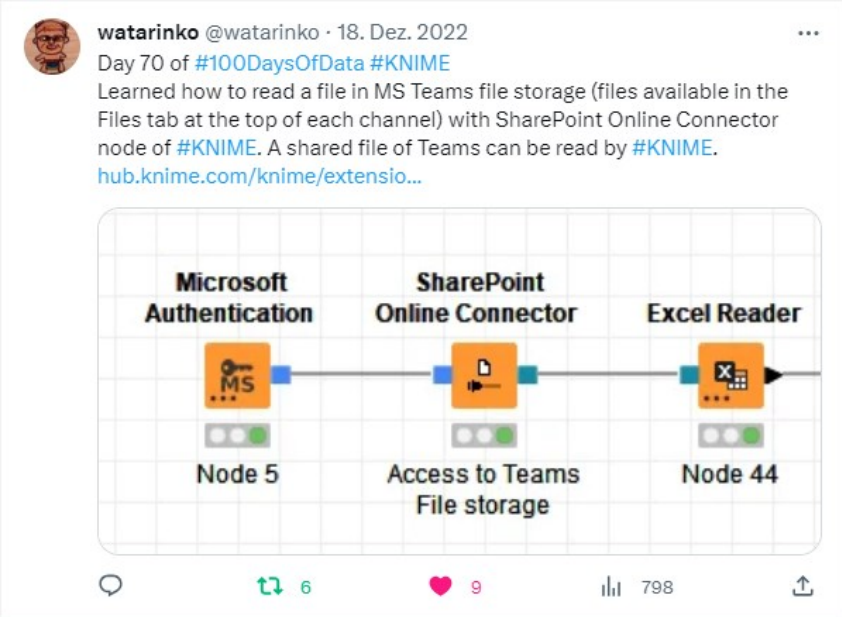
Day 40 of Kazutaka's journey to complete #100DaysOfData with KNIME. He started working on the Just KNIME It! – Challenges.

On Cloud Nine with Connectors

Data can come in different forms and shapes; it can be divided into several types and be stored locally or in cloud services. Data blending is the process of accessing and integrating seamlessly multiformat data scattered over different sources to enrich the dataset and augment its dimensionality. In this process, the diversity of the players involved is the major obstacle to overcome. Whether we work with traditional spreadsheets, images, texts, audio files, geometries or databases, a no-code/low-code tool can take us a long way.

Moving on in his journey, Kazutaka explored some of the KNIME connectors that effortlessly allow access to data stored in cloud services, such as Google Sheets and SharePoint. The paradigm is straightforward and consistent: an authenticator node (*Microsoft Authentication* node), a cloud service connector node (*SharePoint Online Connector* node), and a reader node (*Excel Reader* node). With just three nodes and a couple of clicks, Kazutaka was able to import files stored on the cloud in KNIME Analytics Platform and further process them.

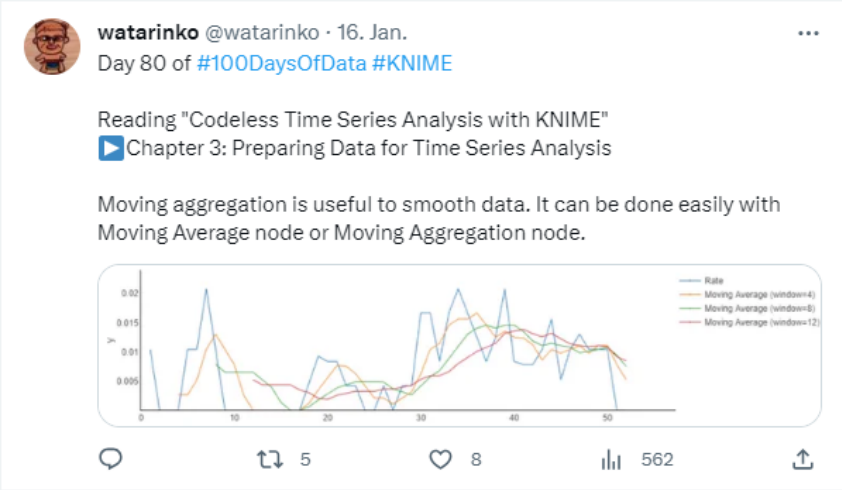
Easier done than said!



Day 70 of Kazutaka's journey to complete #100DaysOfData with KNIME. He explored some of the KNIME connectors that allow access to data stored in cloud services.

Note. Check the original post by @*watarinko*: Day 70 on Twitter.

A Practical Guide to Time Series Analysis



Day 80 of Kazutaka's journey to complete #100DaysOfData with KNIME. In his learning journey, he couldn't miss to cover Time Series Analysis.

There is never enough discussion, educational content, or data flows about one of hottest subfields in data science. It's fascinating to observe how introducing time to

analyze a sequence of data points requires ad-hoc methods and algorithms to extract meaningful insights or make predictions.

In his learning journey, Kazutaka couldn't miss to cover Time Series Analysis. A journey within the journey using the "Codeless Time Series Analysis with KNIME" book (Packt, 2022). From time granularity and alignment to basic time series plotting, Fourier Transforms and neural networks, each chapter introduces a fundamental building block to design an effective forecasting pipeline. Kazutaka shared his daily progresses, providing key insights into the chapter and attaching visuals that better illustrate the topic.

If you were new to time series analysis, Kazutaka's daily updates were hopefully the motivation you had been looking for to get started.

Note. Check the original post by @*watarinko*: Day 80 on Twitter.

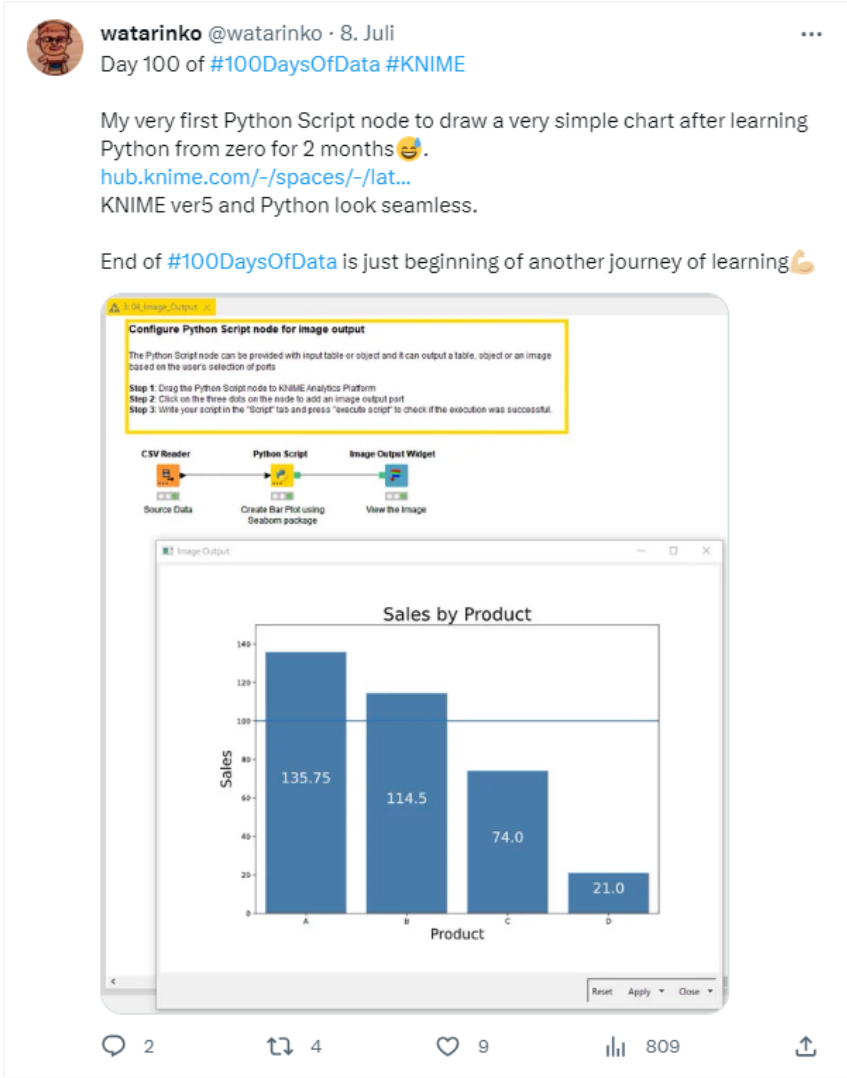
No Code meets Low-Code

No-code/low-code tools like KNIME Analytics Platform are called like that because they don't require any coding skills to work with data, but they support the option to include code snippets if needed. Namely, in R, Python, Java, and JavaScript. Introducing code snippets might be useful to integrate the latest ML algorithms, visualization library, or features. On the other hand, it may hinder the visual understanding of the data flow, for what happens in the script is not exposed. Ultimately, the choice lies with workflow developers based on their skills and target end-user.

Kazutaka's #100DaysOfData journey concluded with the creation of a bar chart that displays bar values centered on each bar. While a similar result can be also obtained with the new *Bar Chart* node, Kazutaka's commitment to embark on a Python coding journey at the same time and apply the new skills in his go-to analytics tool proved how smoothly KNIME's open-source ecosystem can blend technologies.

Not only that. It was refreshing to see that coding and no-coding were not portrayed as antagonists in a race for the best one, but rather as complementary ways to work with data. Kazutaka shared in his post a simple workflow example, helping other KNIME-Python newbies understand how to write code snippets that output an image using the *Python Script* node.

Note. Check the original post by @*watarinko*: Day 100 on Twitter.

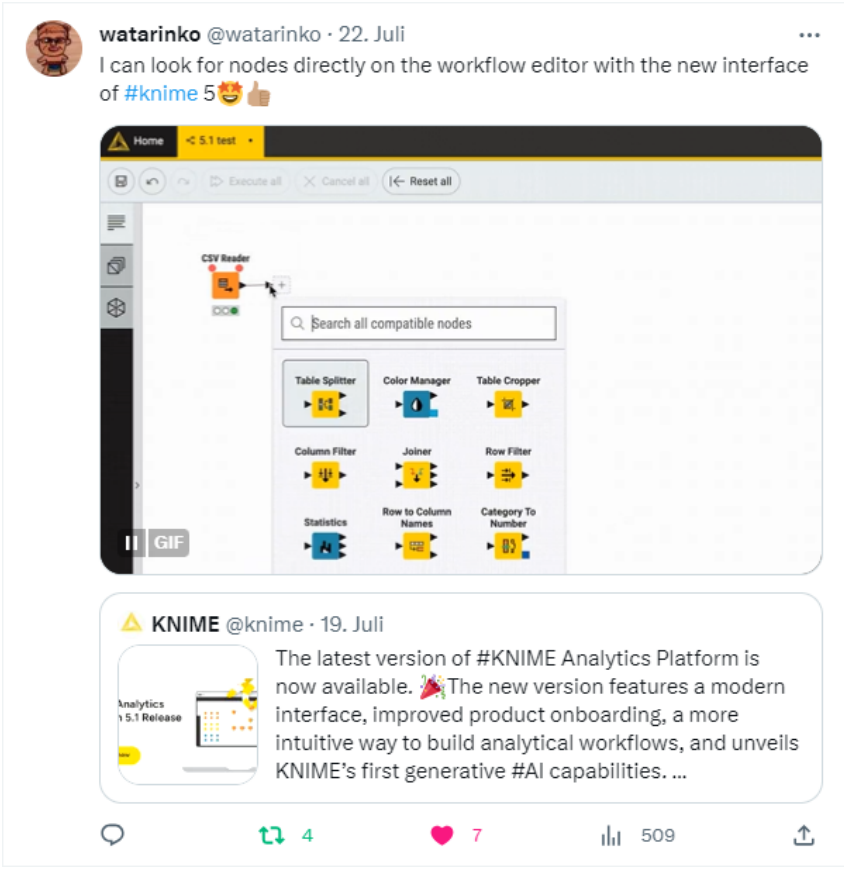


Kazutaka's last day of journey to complete #100DaysOfData with KNIME. On day 100, he concluded the challenge by creating a bar chart using a Python Script node which means, he even learned some coding during those 100 days.

Always on Top of the Game

Kazutaka's learning journey with KNIME did not end on Day 100 of his #100DaysOfData journey, nor with the COTM award in October 2022. Being a social media influencer for the KNIME community means that he needs to stay informed and relevant. And he does. He lives up to the expectations of his followers, catching the latest news around the software and spreading them.

Him sharing the solutions to weekly Just KNIME It! challenges both on the KNIME Forum and social media (indeed, some of his #100DaysOfData posts revolved around them), or the swift adoption of KNIME Analytics Platform v5.1 with its modern UI, improved features and integrated AI functionalities are a testament to his long-lasting commitment to stay always on top of the game.



Even after completing #100DaysOfData with KNIME, and being awarded as KNIME Contributor of the Month, he still continues sharing his achievements and what he's learned on social media.

Note. Check the original post by *@watarinko* on Twitter.

Yasue Katsutaka was nominated KNIME Contributor of the Month for December 2022. Together with Emiliano Amendola, Raffaello Barri, and Anil Kumar Sharma, he was nominated for being the best of the best in the "Just KNIME It!" Challenges of Season 1. The four winners were chosen based on the number of submitted solutions, as well as their creativity, efficiency, and educational value. All four KNinjas go above and beyond in making blogs and amazing dashboards, helping the community on the KNIME Forum, and being a big part of our social media events!

Yasue is based in Nara, Japan, and has years of experience working as a researcher in a pharmaceutical company: he has 10+ years of experience working as Medicinal Chemist, and 5 years of experience working as Information System Manager. His focus areas include Cheminformatics and IT Systems, including JChem, KNIME, and Oracle Database. His experience covers management of databases, machine learning, and data wrangling.

Visit Yasue's space on the KNIME Hub or his profile in the KNIME Forum (Hub/Forum handle: `knimest`).

Emiliano Amendalo was nominated KNIME Contributor of the Month for December 2022. Together with Yasue Katsutaka, Raffaello Barri, and Anil Kumar Sharma, he was nominated for being the best of the best in the "Just KNIME It!" Challenges of Season 1. The four winners were chosen based on the number of submitted solutions, as well as their creativity, efficiency, and educational value. All four KNinjas go above and beyond in making blogs and amazing dashboards, helping the community on the KNIME Forum, and being a big part of our social media events!

Emiliano is a Computer Science Engineer and currently works as Financial Data Analyst at the United Nations. In his career so far, he has mostly worked with banking and financial data, engaging in different data operations from cleaning and transformation to the application of machine learning models and the creation of dashboards. His interests in data analysis is connected to his senior years of his university studies, where he was particularly taken by all things data science.

Visit Emiliano's space on the KNIME Hub or his profile in the KNIME Forum (Hub/Forum handle: `eamendola`).

Raffaello Barri was nominated KNIME Contributor of the Month for December 2022. Together with Yasue Katsutaka, Emiliano Amendola, and Anil Kumar Sharma, he was nominated for being the best of the best in the "Just KNIME It!" Challenges of Season 1. The four winners were chosen based on the number of submitted solutions, as well as their creativity, efficiency, and educational value. All four KNinjas go above and beyond in making blogs and amazing dashboards, helping the community on the KNIME Forum, and being a big part of our social media events!

Raffaello graduated in Engineering Management, specializing in Supply Chain Management, from Polytechnic University of Milan, Italy in 2020. He currently works as Consultant at BIP. Raffaello discovered his interest in data-driven applications while working on his master's thesis where he developed a logistics network. He encountered KNIME two years ago and quickly became passionate about the tool, using it not only for work but also for hobby projects.

Visit Raffaello's space on the KNIME Hub or his profile in the KNIME Forum (Hub/Forum handle: `lelloba`).

Anil Kumar Sharma was nominated KNIME Contributor of the Month for December 2022. Together with Yasue Katsutaka, Emiliano Amendola, and Raffaello Barri, he was nominated for being the best of the best in the "Just KNIME It!" Challenges of Season 1. The four winners were chosen based on the number of submitted solutions, as well as their creativity, efficiency, and educational value. All four KNinjas go above and beyond in making blogs and amazing dashboards, helping the community on the KNIME Forum, and being a big part of our social media events!

Anil currently works as Deputy General Manager Purchase at Dabur India Limited, a leading FMCG company. In his job, he is handling soft commodity sourcing for the CPG industry. His interests in data analytics mainly sparked within the scope of his job, where he usually deals with large amounts of diverse structured and unstructured data and ensures the analytics and automation of various supply management processes. Dynamic scenarios and ever-changing data keep the buyers (sourcing professionals) on their toes, as both macro and micro detail count and require data-driven decisions.

Visit Anil's space on the KNIME Hub or his profile in the KNIME Forum (Hub/Forum handle: `anilks`).

Meet the Top 4 KNinjas of "Just KNIME It!" – Season 1

Learn to Solve Just KNIME It! Challenges like a Pro

Authors: Aline Bessa & Victor Palacios

Just KNIME It! is a series of weekly challenges with the goal of helping KNIME users prove their KNIME knowledge and practice their workflow building skills. Each week, we publish a challenge, levels range from easy to difficult, and give one week for our community to solve it before we publish our own solution.

Solutions are uploaded onto a KNIME Community Hub space with a specific tag (i.e., `justknimeit-xx` for season 1 and `JKISeason2-xx` for season 2 respectively) where `xx` is the number of the corresponding challenge. For each challenge we create a discussion space on the KNIME Forum, and also a leaderboard ranking the top 20 participants by the number of submitted solutions. Although we are right in the middle of *Just KNIME It! – Season 2*, this article focuses on our KNinjas from Season 1. Hence, all challenges we're talking about in this article are challenges from *Just KNIME It! – Season 1*.

The first season of *Just KNIME It!* took place in 2022, precisely from January 25 to October 26, 2022. In this first season, we issued a total of 40 challenges with topics ranging from visualization, web scraping, statistics, natural language processing, and much more. We had challenges from many domains, and we also had many challengers attempt these mini KNIME puzzles.

> ***Note.*** If you want to reiterate all challenges from *Just KNIME It! – Season 1*, you can download the booklet "Just KNIME It! - Learning KNIME One Challenge at a Time" (Season 1) from www.knime.com/knimepress/JKI-season1.

At the end of *Just KNIME It! – Season 1*, we picked four KNinjas who stood out amongst the crowd. The four winners are:

- Yasue Katsutaka (`KNIMEST`)
- Emiliano Amendola (`eamendola`)
- Raffaello Barri (`lelloba`)
- Anil Kumar Sharma (`AnilKS`).

How did we choose those winners? We had four criteria:

1. the number of submitted solutions
2. creativity
3. efficiency
4. teaching value to others

All members completed at least 37 challenges, and each shared something with the community – whether it be a personal website, amazing visualizations, helped users on the KNIME Forum, or made really efficient solutions.

But enough with the introductions – let's get to know the winners. Read on to find out more about them, get pointers to their space on KNIME Community Hub, see selected challenges – the ones we feel stand out from other competitors – as well as highlights on how they have contributed to the KNIME community.

Yasue Katsutaka

Yasue completed all 40 *Just KNIME It!* challenges. You can find all of his *Just KNIME It!* solutions in his public space on the KNIME Community Hub.

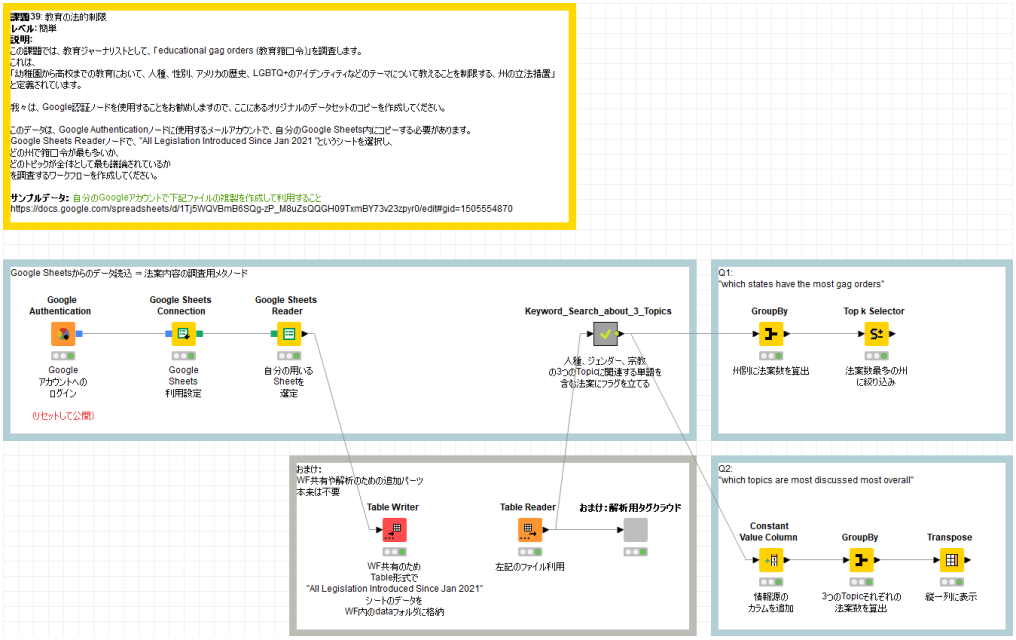
His solutions are always meticulous, with each node containing a clear description of what it will do, and all components named distinctly and meaningfully. Similarly, the dashboards contain all required explanations to understand what is depicted in them.

Other members of the KNIME team have noted that Yasue always does a great job and writes about each and every solution – a tall order to do on a weekly basis (psst… we will talk about his blog later in this article as well).

We would also like to praise Yasue's incredible work because it can be used as educational material for new KNIME users as well. Overall, we believe Yasue's fantastic workflows specialize in clean and helpful documentation, which is why he is one of our winners.

Selected Challenge

One solution we would like to feature is his workflow for Challenge 39: Legislative Limits on Teaching. In this challenge, KNIME users played the role of educational journalists investigating "educational gag orders" and were required to create a workflow that examined which states have the most gag orders, and which topics are most discussed overall. Yasue's solution is shown in the figure below. Note the clear description and the comments under every node, as in all his other solutions.



Yasue's annotated solution to Just KNIME It! challenge 39. It can be downloaded from Yasue's Community Hub Space.

Extra Community Contribution

Yasue's solution can be found not only on the KNIME Community Hub, but also on his blog. This blog – written in Japanese – has become a reference point for the KNIME newbies in Japan. Indeed, each article describes the chain of actions to bring the original data to the final target.

A picture of Yasue's blog about all the Just KNIME It! Challenges

We were especially moved by his step-by-step instructions and guidance for each and every of the 40 challenges we presented. He also translated all the challenges so that even our Japanese-only-speaking community members can participate with ease. His work embodies the open-source spirit – taking work and making it available and accessible to an even broader audience in real-time.

Thank you, Yasue, for this wonderful contribution!

Emiliano Amendola

Emiliano completed 37 *Just KNIME It!* challenges. You can find his solutions in his dedicated space on the KNIME Community Hub. Emiliano constantly blows us away with his amazing use of components to create beautiful works of art. Within the KNIME team, we each comment that his dashboards are some of the prettiest we've ever seen, the most elegant, and well documented. His workflows certainly specialize in beautiful dashboard design.

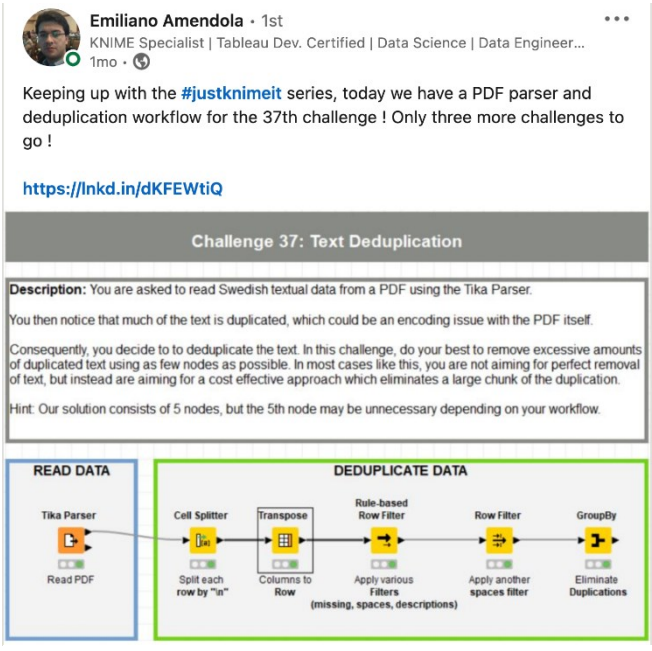
Selected Challenge

In representation of Emiliano's submitted solutions, we have selected the workflow for Challenge 7: Women in Government. In this challenge participants were asked to create one or more visualizations to inspect the percent share of seats held by women in local government, as reported by the United Nations Development Program: Human Development Reports. The purpose of this report was to "ensure women's full and effective participation and equal opportunities for leadership at all levels of decision-making in political, economic and public life". Emiliano's solution stood out for the unique use of layout, interactivity, and text size control – getting the right balance of all three is quite a feat, so he is clearly aesthetically gifted (see figure below).

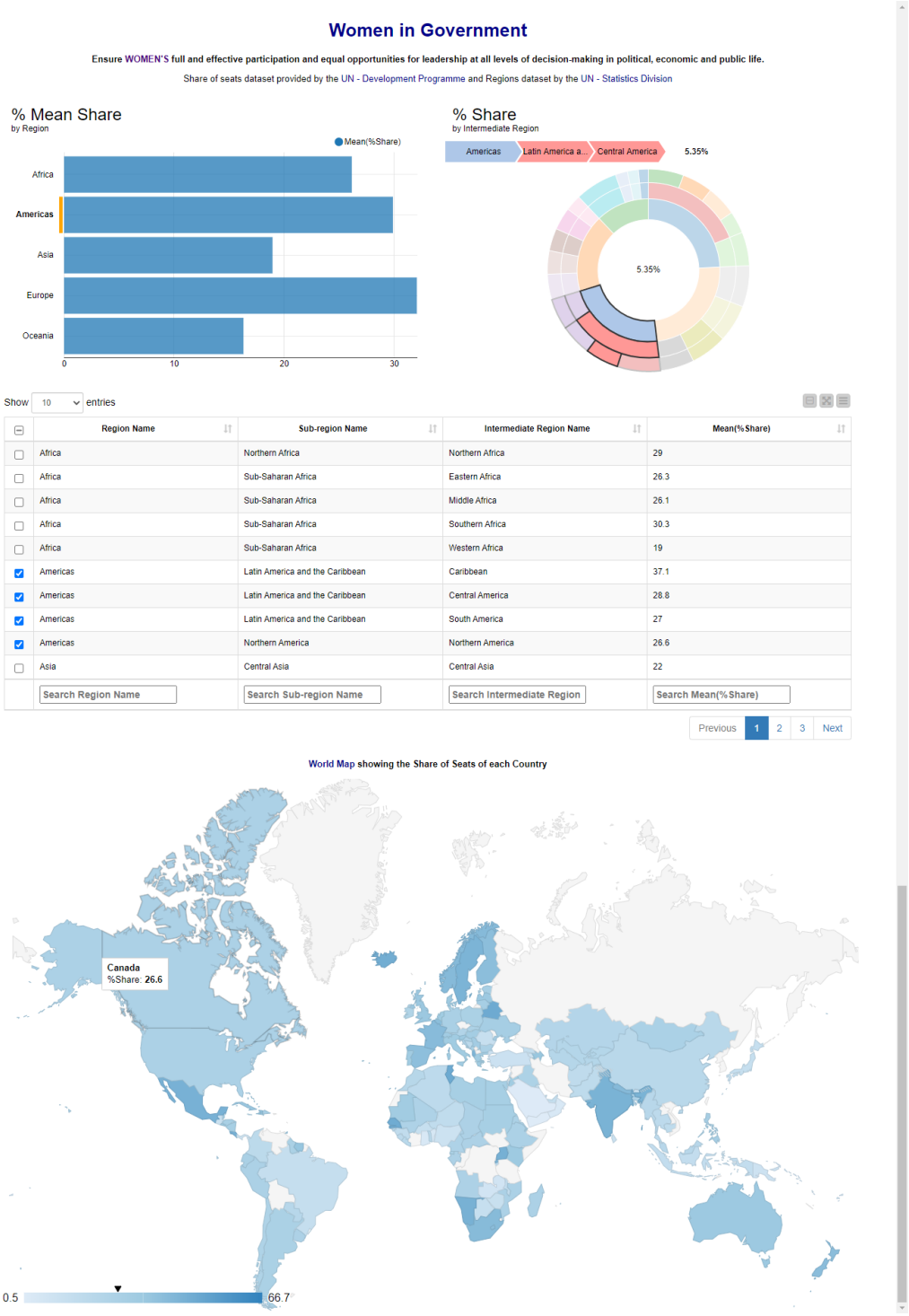
Extra Community Contribution

Not only for this challenge, but for almost any challenge with visual requirements, Emiliano constantly went above and beyond to offer excellent eye candy with his dashboards. And what we also love about this is that he brings his creations to LinkedIn and shows them off to the world.

Thanks for sharing your work with all of us in LinkedIn land, Emiliano!



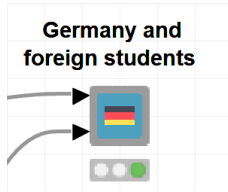
Emiliano's LinkedIn post with an eye-pleasing solution to challenge 37 of Just KNIME It!



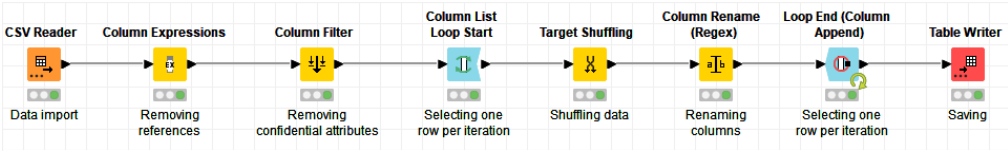
Emiliano's well-designed dashboard, made for Just KNIME It! challenge 7. It can be downloaded from [Emiliano's Community Hub Space](#).

Raffaello Barri

Raffaello completed all 40 *Just KNIME It!* challenges. You can find his solutions in his space on the KNIME Community Hub. His solutions often make use of components to encapsulate and recycle useful functionalities. Raffaello is a master in component building, often even going to great lengths to find the most appropriate icon for them. And he also makes sure his workflows are clean and crisp, annotating each appropriately and making sure they are aesthetically pleasing.



One of the many components Raffaello created.



Raffaello's well-documented and visually appealing solution to Just KNIME It! challenge 9.

Selected Challenge

In representation of Raffaello's submitted solutions, we have selected the workflow for Challenge 31: Extracting Keywords from a Website. Here, participants were tasked with extracting text from a recipe web page, performing a keyword search, and then creating a tag cloud of the keywords.

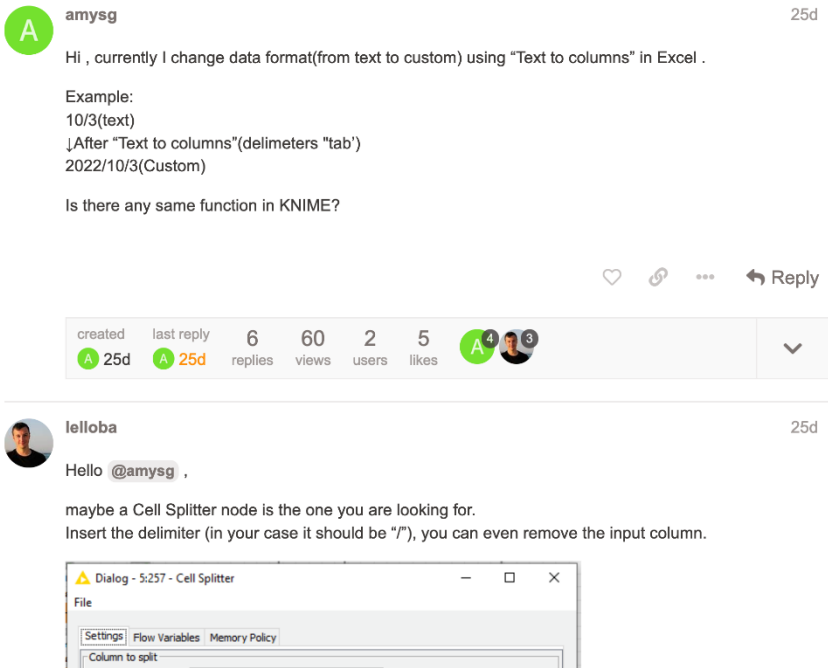
The solution should have shown the tag cloud of the keywords and produced a guess for the recipe! Looking at Raffaello's solution via the tag cloud below, can you guess what the ingredients will ultimately produce?

Raffaello's tag cloud solution to Just KNIME It! challenge 31. It can be downloaded from Raffaello's Community Hub Space.

Extra Community Contribution

Apart from being a prime competitor in the *Just KNIME It!* challenges, Raffaello also completed the *#66daysofdata* challenge, but away from the social media spotlight. He also spends time on the KNIME Forum helping others. His posts on the Forum are clearly successful, having received over 200 likes! His dedication to KNIME and to KNIME users in need is truly something we love to see and reward!

Thank you for helping us make the KNIME community even better, Raffaello!



Raffaello helping another KNIME user on the KNIME Forum.

Anil Kumar Sharma

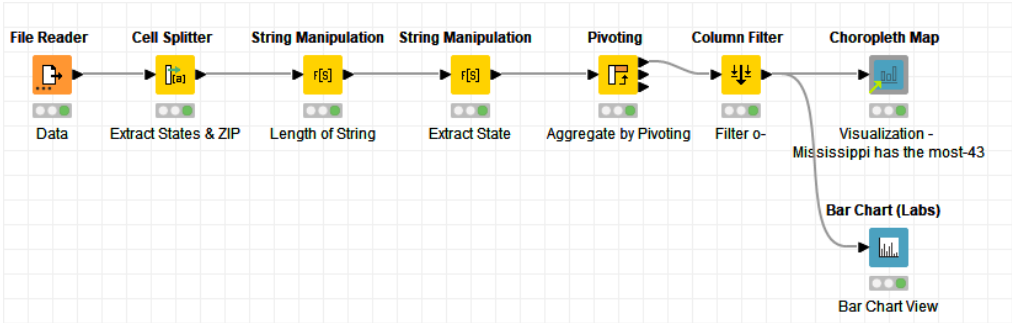
Anil also completed all 40 *Just KNIME It!* challenges. You can find his solutions in his dedicated space on the KNIME Community Hub. His solutions are efficient, to the point, and sometimes the very first solution we'd receive for a challenge! We really noticed that he'd often use the least number of nodes to reach the goal of the challenge – and in this case less is far more. His workflows specialize in compactness making them approachable and readable even for the newest of KNIME users.

Selected Challenge

In representation of Anil's submitted solutions, we have selected the workflow for Challenge 27: Rare Blood Types. Here the goal was to help a group of researchers find the number of citizens with O blood type per US state within some messy data. Anil was amongst the first challengers to submit a very practical and compact solution that

quickly gets to the point and generates the needed graphics. Notice the beauty of just 7 nodes and one component!

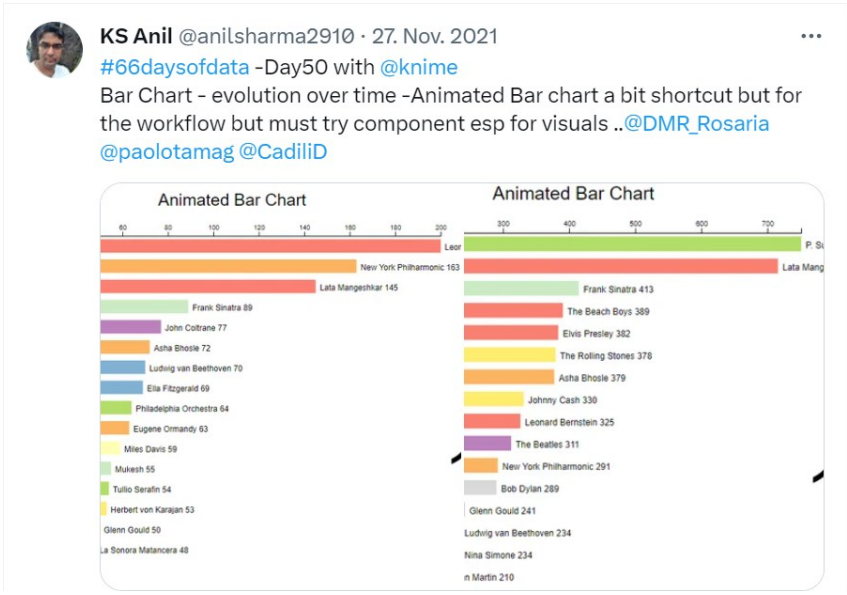
He also approached the problem a little differently. Whereas in our solution we used the Regex Split node to extract the state from the address line into a separate column, Anil more strategically used the Cell Splitter node instead, making an approach that those without regex knowledge can easily follow.



Anil's elegant solution to challenge 27 of Just KNIME It! It can be downloaded from [Anil's Community Hub Space](#).

Extra Community Contribution

Anil was also one of the participants of the first edition of the social media challenge #66daysofdata with KNIME. He was the first user to complete the challenge, documenting his progress daily on Twitter. His tweets kept him and others well motivated to keep learning KNIME for 66+ days! Great job, Anil!



Anil's animated bar chart tweet on Day 50 of the social media challenge "66 days of data with KNIME".

Congratulations to our Just KNIME It! – Season 1 KNinjas!

Double congratulations indeed for having completed most of, if not all, the challenges and for having highlighted, each one of them, a different aspect of the versatility of the KNIME Analytics Platform. We at KNIME appreciate all your hard work and dedication, so thank you, top 4! You are the elites in our book, and we look forward to seeing how you'll make use of KNIME going forward!

Note. If you also want to become a *Just KNIME It!* KNinja, make sure to participate in the weekly challenges and upload them with the correct tag (`JKISeason2-xx`) to your Hub space.

This article was first published in our KNME Blog. Find the original version here.

Node & Topic Index

J

K

L

M

N

P

R

S

T

U

V

W

X

Best of KNIME
The COTM Collection

In the second volume of Best of KNIME we collected the top stories of our KNIME COTMs from August 2022 to July 2023.This booklet contains 12 stories that teach you more about data science and KNIME. Let's learn from the best!

Elisabeth Richter holds a master's degree in Social and Economic Data Science. During her studies, she developed a keen interest in Machine Learning, Deep Learning, and various NLP-related techniques. Her research focused on understanding media bias and examining user behavior on social media. She is part of the Evangelism team at KNIME and works as a Data Science Publisher with a particular focus on the books published under KNIME Press

COTM : August 2022 - July 2023

Arne Beckhaus
Dayanjan (Shanaka) Wijesinghe
Kazutaka Watari
Daniel Weikert
Yasue Katsutaka
Emiliano Amendola
Raffaello Barri
Anil Kumar Sharma
Stephen Roughley
Dominique Sydow
Cristian Rastasanu
James Kim
David Plummer
Arjen Peters
Artem Ryasik