

CHAPTER 5

INTRODUCTION TO DATA MINING

Learning Objectives:

1. *Understand the different categories of data mining techniques*
 2. *Distinguish between supervised and unsupervised techniques*
 3. *Understand common terminology used in data mining*
 4. *Understand the data mining process*
 5. *Perform data exploration, preparation and cleaning using KNIME*
-


Content

Lecture Notes	p. 2
- What is data mining?	p. 2
- Some Common Terminology	p. 3
- Some Common Data Mining Techniques	p. 4
- The Data Mining Process	p. 7
- Introduction to KNIME	p. 8
Tutorial 5	p. 10
Answers	p. 14
Lab 5A	p. 17
Lab 5B	p. 30

1. What is data mining?

Data mining is the process of digging through large data sets to discover hidden patterns and predict future trends. Its foundation is made up of three scientific disciplines:

- *statistics* (the numeric study of data relationships),
- *artificial intelligence* (human-like intelligence displayed by software and/or machines), and
- *machine learning* (algorithms that can learn from data to make predictions).



Why is data mining important?

So why is data mining important? You've seen the staggering numbers – the volume of data produced is doubling every two years. Unstructured data alone makes up 90 percent of the digital universe. But more information does not necessarily mean more knowledge.

Data mining allows you to:

- Sift through all the chaotic and repetitive noise in your data.
- Understand what is relevant and then make good use of that information to assess likely outcomes.
- Accelerate the pace of making informed decisions.

Source: https://www.sas.com/en_sg/insights/analytics/data-mining.html#

Not every interaction with data is considered a data mining task. If we retrieve a subset of the existing data and summarize it to answer a specific question, then we are simply performing a **data query** task. For example, constructing comparison box plots to find out which gender has higher income. On the other hand, **data mining** is much more than mere retrieval of data. To extract useful and previously unknown information from raw data, we typically need to perform modelling and apply algorithms. For example, when we try to ‘sort’ data into groups which have characteristics that are unknown to us, but surfaced naturally by data.

Example 1: In the scenarios below, identify whether each of the following is a data query or data mining task.

- Find the names of customers who have purchased tours costing less than \$1000.
- List the customers who fulfil the following criteria in the past 12 months:
 - Spent more than \$10,000
 - Bought two or more tour packages
- Develop a profile of customers who are likely to purchase tours to Finland.
- Find the probability that a certain customer will respond to an upcoming promotion for a holiday package.

2. Some Common Terminology

Because the foundation of data mining is made up of three scientific disciplines, practitioners often use multiple terms to refer to the same thing. Here is a summary of terms commonly used:

Term	Meaning
Algorithm	A specific procedure used to implement a particular data mining technique: classification tree, k-means clustering etc.
Model	An algorithm as applied to a dataset, complete with its settings (many of the algorithms have parameters that the user can adjust).
Case/ Observation/ Record/ Pattern	The unit of analysis on which the measurements are taken (a customer, a transaction, etc.). Each row typically represents a record whereas, each column, a variable (e.g. the height, weight and age of a person).
Attribute/ Predictor/ Feature/ Input variable	Usually denoted by X , or from a database perspective, a field, that represents characteristics of an object.
Dependent variable/ Response/ Outcome variable/ Output variable/ Target variable	Usually denoted by Y , a variable being predicted in supervised learning.
Variable	Any measurement on the records, including both the input (X) variables and the output (Y) variable.
Supervised learning	Refers to the process of providing an algorithm (decision tree, simple linear regression etc.) with records in which an output variable of interest is known and the algorithm “learns” how to predict this value with new records where the output is unknown.
Unsupervised learning	Refers to analysis in which one attempts to learn something about the data other than predicting an output value of interest (whether it falls into clusters for example).
Estimation/ Prediction	Prediction means the prediction of the value of a continuous output variable; also called estimation.
Test data/ Test set	Refers to that portion of the data used only at the end of the model building and selection process to assess how well the final model might perform on additional data.
Training data/ training set	Refers to that portion of data used to fit a model.
Validation data/ validation set	Refers to that portion of the data used to assess how well the model fits, to adjust some models, and to select the best model from among those that have been tried.
Score	Refers to a predicted value or class. Scoring new data means to use a model developed with training data to predict output values in new data.
Success class	The class of interest in a binary outcome (e.g. purchasers in the outcome purchase/no purchase).

3. Some Common Data Mining Techniques

Data mining techniques are broadly classified as supervised or unsupervised.

In **supervised learning** techniques, we have a specific target variable which we would like to predict. Classification and regression are examples of supervised techniques.

Unsupervised learning techniques, on the other hand, does not focus on any target variable, but try to find hidden structures and relations among the data. An example is clustering.

Some data mining techniques covered in this module include classification, estimation (eg. by regression), and clustering.

3.1 Classification

Classification is the task of assigning records to one of several predefined categories. It is a supervised learning technique. The input data for a classification task is a collection of records (x, y) , where x is the set of attributes and y is the **target** (or **response**) variable called the **class** (or **class label**). Note that in a classification task the target variable is *categorical*.

Examples of applications include:

- Categorize bank loan applications as safe or risky.
- Predict whether newly admitted patients in a hospital are low-risk or high-risk patients.
- Use texture information in images to classify CT scan brain images as inflammatory, tumour or stroke.

Example 2: Records of a population are collected in a health study. An application of data mining is to learn what combination of attributes characterises patients at risk of diabetes and hence predict whether a patient is at risk of diabetes. A partial dataset is given in *Table 1* on the right.

		Attributes			
Records		BMI	Family History	Age	Diabetes
	→	33.6	Yes	Above	Yes
	→	28.1	Yes	Below	No
	→	27.1	No	Above	Yes
	→	37.1	Yes	Above	Yes
	→	36.0	Yes	Above	No
	→	32.9	Yes	Below	Yes
	→	19.4	Yes	Below	No
	→	30.8	No	Below	No
	→	25.0	No	Below	No

Table 1

- How many records are there in the dataset shown in *Table 1*?
- What are the attributes and their data types?
- Which is the target variable and what is its data type?
- Is the task of predicting whether a patient is at risk of diabetes a data query or data mining task?
If it is a data mining task, is it a supervised or unsupervised learning technique?

In this example, each patient is represented as (\mathbf{x}, y) , where:

- $\mathbf{x} = (x_1, x_2, x_3) = (BMI, Family\ History, Age)$ is the attribute set, and
- the last column $y = Diabetes$ is the class variable.

3.2 Estimation

Estimation (eg. by **regression**) is the task of predicting the value of a *numeric* target (or response) variable. It is a supervised learning technique. The input data is again a collection of records (\mathbf{x}, y) , where \mathbf{x} is the set of attributes and y the target variable. Unlike classification, the target variable y in regression is *continuous*.

Examples of application include:

- Predict the time-to-failure of equipment based on external environment conditions.
- Predict sales from cross selling of products using historical data.
- Predict children's height based on age, weight, and other factors.

Example 3: Getting an accurate measurement of the percentage body fat in humans is inconvenient and costly so it is desirable to have easy methods of estimating it. *Table 2* below gives some anthropometric and demographic variables together with the percentage body fat (*%Fat*) in a group of participants in a study.

	Age (years)	Weight (lbs)	Height (in)	Neck (cm)	Chest (cm)	Abdomen (cm)	Hip (cm)	Thigh (cm)	Knee (cm)	Ankle (cm)	Biceps (cm)	Forearm (cm)	Wrist (cm)	%Fat
1	23	154.25	67.75	36.2	93.1	85.2	94.5	59.0	37.3	21.9	32.0	27.4	17.1	12.3
2	22	173.25	72.25	38.5	93.6	83.0	98.7	58.7	37.3	23.4	30.5	28.9	18.2	6.1
3	22	154.00	66.25	34.0	95.8	87.9	99.2	59.6	38.9	24.0	28.8	25.2	16.6	25.3
4	26	184.75	72.25	37.4	101.8	86.4	101.2	60.1	37.3	22.8	32.4	29.4	18.2	10.4
5	24	184.25	71.25	34.4	97.3	100.0	101.9	63.2	42.2	24.0	32.2	27.7	17.7	28.7
6	24	210.25	74.75	39.0	104.5	94.4	107.8	66.0	42.0	25.6	35.7	30.6	18.8	20.9
7	26	181.00	69.75	36.4	105.1	90.7	100.3	58.4	38.3	22.9	31.9	27.8	17.7	19.2
8	25	176.00	72.50	37.8	99.6	88.5	97.1	60.0	39.4	23.2	30.5	29.0	18.8	12.4
9	25	191.00	74.00	38.1	100.9	82.5	99.9	62.9	38.3	23.8	35.9	31.1	18.2	4.1
10	23	198.25	73.50	42.1	99.6	88.6	104.1	63.1	41.7	25.0	35.6	30.0	19.2	11.7

Table 2

- How many records are shown in the table below?
- What are the attributes and their data types?
- Which is the target variable and what is its data type?
- Is the task of estimating the percentage body fat a data query or data mining task?
If it is a data mining task, is it a supervised or unsupervised learning technique?

3.3 Clustering

Clustering is the task of grouping a set of records in such a way that the records in the same group (or cluster) are more similar (using some measures) to each other than records in other groups (clusters). Unlike classification, class labels are *unknown* and it is up to the clustering algorithm to discover acceptable classes. It is an unsupervised learning technique.

Examples of application include:

- Design web recommender systems to recommend new items based on a user's taste, predicted using the preferences of other users in the same cluster.
- Identify groups of students with similar online behaviour.
- Segment customers in a business into a small number of groups for marketing activities.

Example 4: Recall the Prestige Mall dataset where feedback from 200 customers are collected through a survey. The marketer would like to identify and segment the customers into groups that are similar in specific ways relevant to marketing, and come up with different marketing programs.

No.	Job sector	Age	No. of visits per month	Gender	Household income (in \$)	Amount spent per month (in \$)
1	Bus/Fin	18	6	Female	8852.32	441.73
2	Bus/Fin	19	3	Female	7889.24	663.63
3	Bus/Fin	22	1	Female	6901.79	489.26
4	Bus/Fin	22	6	Female	8566.96	412.19
5	Bus/Fin	23	6	Female	7144.45	188.59
6	Bus/Fin	24	6	Female	8092.08	253.59
⋮						
⋮						

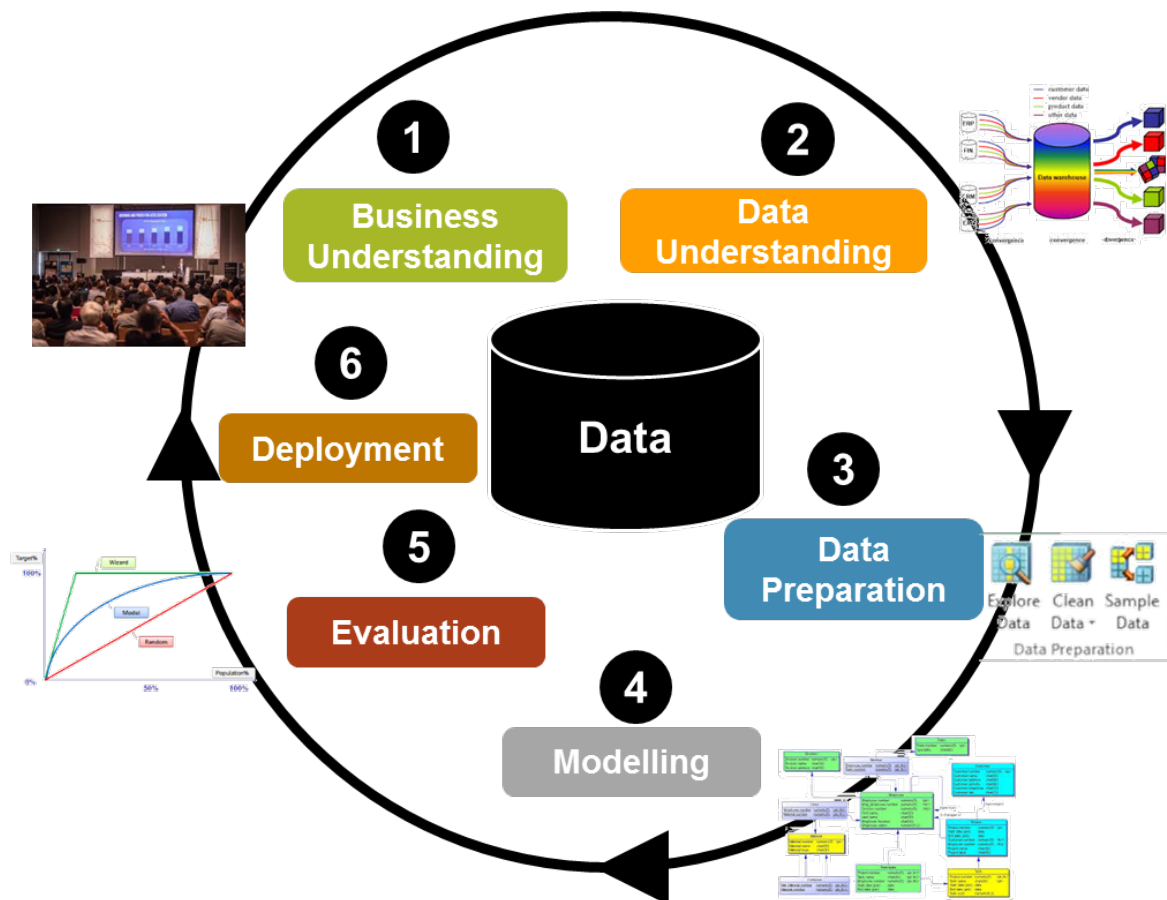
Table 3: Incomplete table for the Prestige Mall dataset

- How many records are there in the dataset?
- Is there a target variable?
- Is the task of customer segmentation a data query or data mining task?
If it is a data mining task, is it a supervised or unsupervised learning technique?

4. The Data Mining Process

A common approach used in the industry for data mining is **CRISP-DM** (Cross-Industry Standard Process for Data Mining). This approach consists of six steps:

1. **Business understanding.** This includes understanding business objectives, establish data mining goals, and develop a project plan.
2. **Data understanding.** This includes data collection, data description, data exploration, and the verification of data quality.
3. **Data preparation.** Once collected, data cleaning and transformation in preparation for data modelling occur in this step.
4. **Modelling.** Software tools can be used to generate detailed models using appropriate data mining techniques. The partition of data into training, validation and test sets is also implemented for modelling.
5. **Evaluation.** Model results are evaluated in the context of the business objectives.
6. **Deployment.** Models found to be appropriate and sound can be applied to business operations.

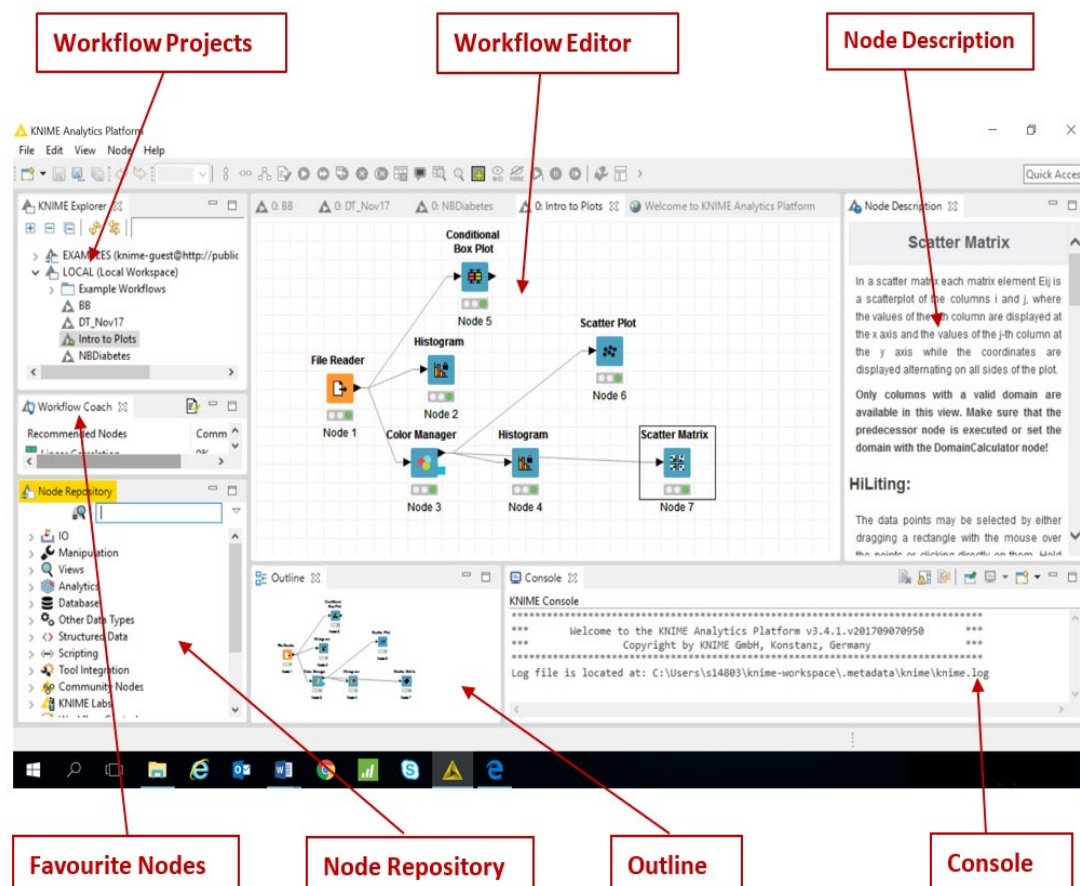


5. Introduction to KNIME

Due to large amount of computing required with possibly large data sets, software is required to perform data mining tasks. The data mining software adopted in this module is KNIME.

KNIME (Konstanz Information Miner) is an open source software developed at the University of Konstanz in Germany used for data analytics, data mining and machine learning.

A KNIME **workbench** looks like this:

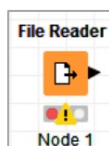


The workbench consists of the following panels:

- **Node Repository**

A node is a single processing unit of a workflow (explained below).

An example is a **File Reader** node:

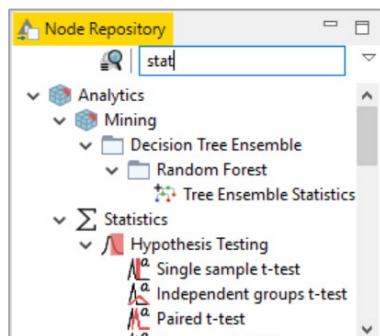


A node can be in one of three states:

- **Red:** node is inactive and not yet configured
- **Yellow:** node is configured but not yet executed
- **Green:** executed successfully

If a node is executed unsuccessfully, its status is yellow.

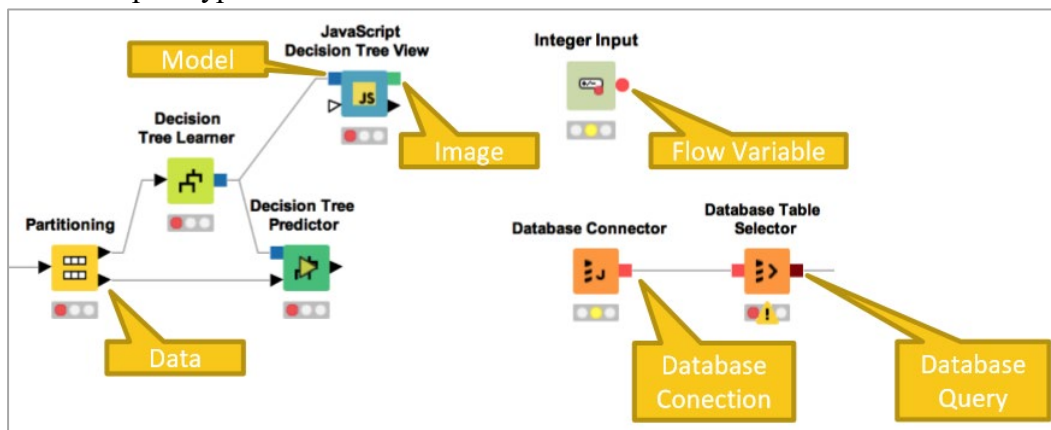
There is also a *Search* box in this panel. We can type a keyword in the search box, hit *Enter* and find the node we want. To see all the nodes again press *Esc*.



- **Workflow Editor**
This is the panel we work in most of the time. A node can be selected from the *Node Repository* and dragged and dropped here. Many nodes can be assembled to form a **workflow**.
- **Node Description**
This panel displays a summary description of a selected node.
- **Workflow Project**
This gives the list of workflow projects in the selected workspace.
- **Favourite Nodes**
This displays nodes which are most recent, or which are used most frequently.
- **Console**
This displays error and warning messages. It also shows the location of the log file.
- **Outline**
This displays a small overview of the contents of the *Workflow Editor*.

Inserting and connecting nodes in a workflow:

- Insert nodes into *Workflow Editor* by dragging them from *Node Repository* or by double-clicking in *Node Repository*.
- Connect nodes by left-clicking output port of Node A and dragging the cursor to (matching) input port of Node B
- Common port types in nodes:



TUTORIAL 5

1. Assuming that data mining techniques are to be used in each of the following scenarios, identify whether the task required *supervised learning* or *unsupervised learning* techniques.
 - (a) Identify whether a network data packet is dangerous (virus, hacker attack), based on past similar records with known threat status.
 - (b) Predict whether a company will go bankrupt, based on comparing its financial data to similar bankrupt and non-bankrupt companies.
 - (c) Identify segments of similar customers in an insurance company.

2. In each of the following scenarios, identify if *classification*, *regression*, or *clustering* is an appropriate data mining task. Also, determine the target variable in cases where a classification or regression task is appropriate.
 - (a) A stock market analyst has been asked to predict the future stock price of a company using past historical data.
 - (b) An IT department in a company would like to predict whether its incoming email messages are spam, by using attributes such as the percentage of characters in a collection of emails that match a particular character, the average length of uninterrupted sequences of capital letters, the length of the longest uninterrupted sequence of capital letter, the total number of capital letters in the email, etc.
 - (c) The health authority in a city would like to predict which patients will have diabetic retinopathy, glaucoma, or age-related macular degeneration in the future, based on half a million of retinal images from people of different ethnicities.
 - (d) An educator is interested to use web log data to discover groups of similar access pattern to the school's learning management system among her students.
 - (e) A large retailer makes use of information on store footfall, advertising campaign and other records to predict sales.

3. [📄] The iris flower dataset *iris.csv* is a multivariate dataset first introduced by the British statistician and biologist Ronald Fisher in 1936. The dataset consists of 50 samples for each of the three species of Iris, namely *Iris Setosa*, *Iris Versicolor* and *Iris Virginica*. Four attributes were measured for each sample: *sepal length*, *sepal width*, *petal length*, and *petal width* (all in cm).

Download *iris.csv* from Blackboard or the UCI Machine Learning Repository (<https://archive.ics.uci.edu/ml/datasets/iris>) and answer each question that follows.

- (a) Find the minimum, maximum, mean and standard deviation for each of the four attributes *sepal length (cm)*, *sepal width (cm)*, *petal length (cm)*, and *petal width (cm)* and fill in the table below.

Attribute	Minimum	Maximum	Mean (\bar{x})	Std. Dev. (s)
<i>sepal length (cm)</i>				
<i>sepal width (cm)</i>				
<i>petal length (cm)</i>				
<i>petal width (cm)</i>				

- (b) For each species, find the mean (\bar{x}) and standard deviation (s) for each of the four attributes and fill in the tables below.

Species	\bar{x} (sepal length)	\bar{x} (sepal width)	\bar{x} (petal length)	\bar{x} (petal width)
<i>Iris-setosa</i>				
<i>Iris-versicolor</i>				
<i>Iris-virginica</i>				

Species	s (sepal length)	s (sepal width)	s (petal length)	s (petal width)
<i>Iris-setosa</i>				
<i>Iris-versicolor</i>				
<i>Iris-virginica</i>				

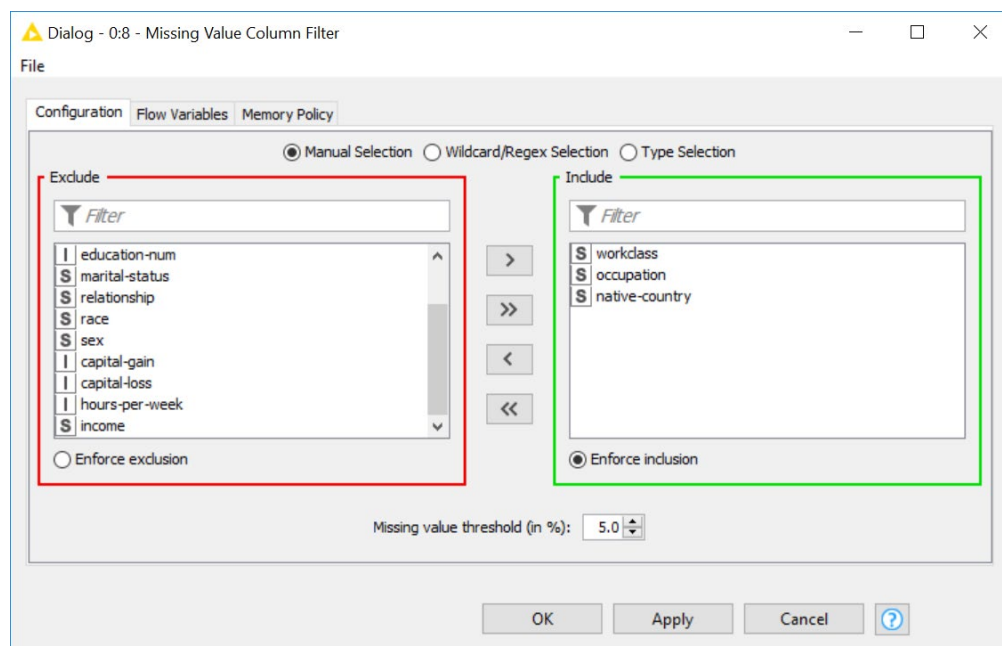
- (c) Construct a pie chart (or bar graph) for *species*. Comment on the overall proportions.
- (d) Construct a histogram for each of the attribute *sepal length (cm)* and *sepal width (cm)*. Comment on the distribution of each of the attribute and compare.
- (e) Construct the boxplots for the attributes *petal length (cm)*, and *petal width (cm)* for each species. Compare across each species and comment on the separability of each species.
- (f) A **scatterplot matrix** allows us to examine the relationship between pairs of variables simultaneously.
Construct a scatterplot matrix for the attributes *sepal length (cm)*, *sepal width (cm)*, *petal length (cm)*, and *petal width (cm)*. Comment on the relationship and discuss on the separability of each species.
- (g) Compute pairwise correlations for the variables. Comment on your results. Are they consistent with your observations in part (f)?
- (h) A **parallel coordinates plot** is a visualization technique used to plot each observation (or row) across many variables. Each variable corresponds to a vertical axis and each observation is displayed as a series of connected lines along the axes.

Construct a parallel coordinates plot for the attributes *sepal length (cm)*, *sepal width (cm)*, *petal length (cm)*, and *petal width (cm)*. Discuss on the separability of each species.

4. [📄] The census bureau dataset is available at the UCI Machine Learning Repository (<https://archive.ics.uci.edu/ml/datasets/adult>). This data was extracted from the census bureau database found at <http://www.census.gov/ftp/pub/DES/www/welcome.html>. An individual's annual income is a result of various factors. Intuitively, it is influenced by the individual's education level, age, gender, occupation, etc. The dataset contains 15 columns with the *Income* as the target attribute. *Income* is divided into two classes: $\leq 50K$ and $> 50K$. There are 14 attributes which are mainly demographics and features to describe a person.

Download *adult.csv* from Blackboard or the UCI Machine Learning Repository (<https://archive.ics.uci.edu/ml/datasets/adult>) and answer each question below.

- How many records are there in the dataset?
- One of the analysis of the data collected is to predict individual's annual income by their demographics. Is this task a data query task or a data mining task?
- Are there missing values in any of the attributes? If there are, identify the attributes and state the number of missing values in each of the attributes.
- Suggest one appropriate way to handle the missing values for *native-country*.
- If we use *Missing Value Column Filter* node to filter the original dataset with the configuration setting as follows, is there any column(s) that will be filtered? Briefly explain why.



- (f) What is the number of records that contain no missing values in all the attributes of female individuals whose “marital-status” contain ‘married’ (not case sensitive) in the dataset? (Try using the various filter nodes in KNIME.)
- (g) From the filtered data in part (f), replace the numeric column *age* with an attribute column as shown here:

```
young : ] -∞ ... 20.0 [
middle-low : [ 20.0 ... 35.0 [
middle-high : [ 35.0 ... 50.0 [
old : [ 50.0 ... ∞ [
```

Hence, what is the proportion of middle-low *age* group in the filtered data thus far?

- (h) If we want to add the following 2 rows of records to the existing dataset, which node in KNIME should we use?

I age	S workclass	I fnlwgt	S education	I educati...	S marital-status	S occupa...	S relation...	S race	S sex	I capital-...	I capital-...	I hours-p...	S native-...	S income
30	Private	111415	HS-grad	9	Married-civ-spouse	Other-service	Husband	White	Male	0	0	55	Germany	<=50K
44	Private	456236	Masters	14	Married-civ-spouse	Adm-clerical	Husband	White	Male	0	0	40	United-States	<=50K

ANSWERS

1.

- (a) Supervised learning (classification)
- (b) Supervised learning (classification)
- (c) Unsupervised learning (eg clustering)

2.

- (a) Target variable is stock price, which is continuous. Estimation (e.g. by regression) may be appropriate.
- (b) Target variable is spam (Yes/No), which is categorical. Classification is appropriate.
- (c) Target variable is type of eye disease (diabetic retinopathy/glaucoma/age-related macular degeneration), which is categorical. Classification is appropriate.
- (d) No target variable here; unsupervised learning techniques (e.g. clustering) may be appropriate.
- (e) Target variable is sales, which is continuous. Estimation (e.g. by regression) may be appropriate.

3.

(a)

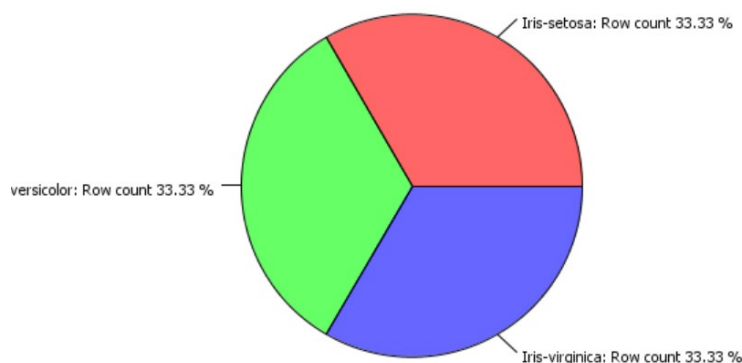
Column	Min	Mean	Max	Std. Dev.
sepal length (cm)	4.3	5.8433	7.9	0.8281
sepal width (cm)	2	3.054	4.4	0.4336
petal length (cm)	1	3.7587	6.9	1.7644
petal width (cm)	0.1	1.1987	2.5	0.7632

(b)

Row ID	S species	D Mean(s...	D Mean(s...	D Mean(p...	D Mean(p...
Row0	Iris-setosa	5.006	3.418	1.464	0.244
Row1	Iris-versicolor	5.936	2.77	4.26	1.326
Row2	Iris-virginica	6.588	2.974	5.552	2.026

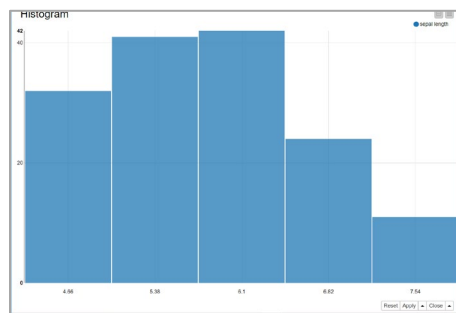
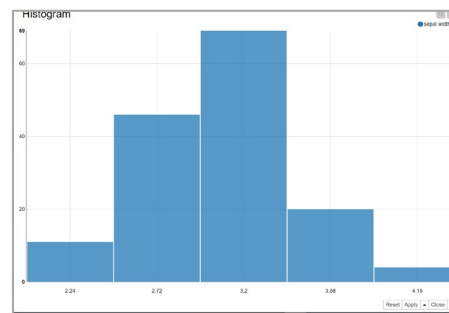
Row ID	S species	D Standa...	D Standa...	D Standa...	D Standa...
Row0	Iris-setosa	0.352	0.381	0.174	0.107
Row1	Iris-versicolor	0.516	0.314	0.47	0.198
Row2	Iris-virginica	0.636	0.322	0.552	0.275

(c)



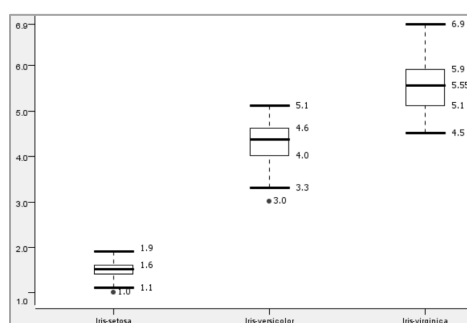
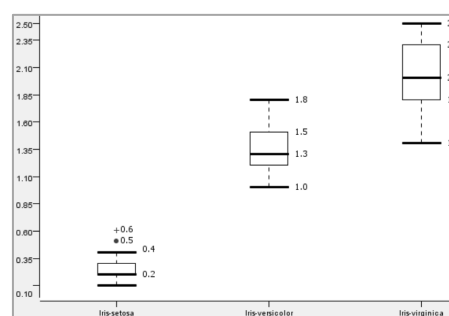
All three species have the same frequency and relative frequency.

(d)

Histogram of *sepal length* (cm)Histogram of *sepal width* (cm)

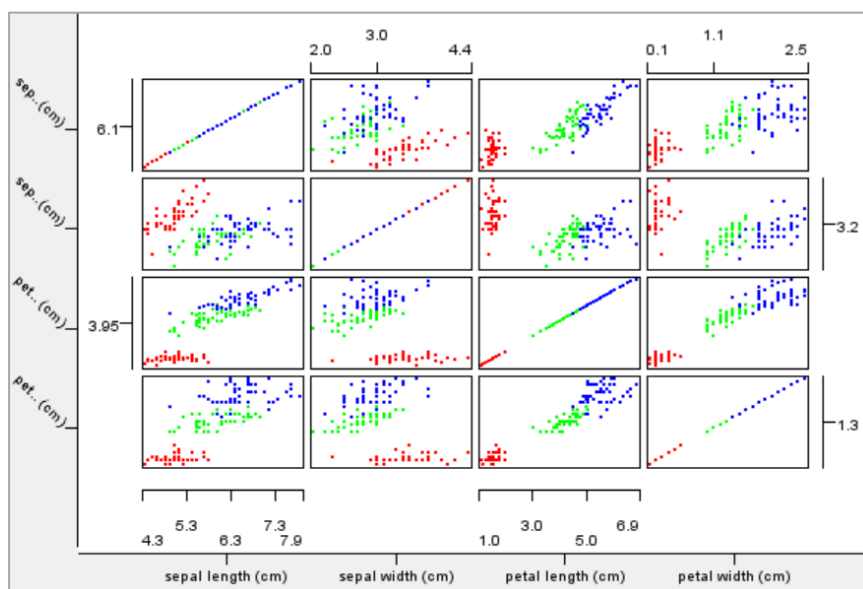
Both histograms are skewed slightly to the right, with sepal length (peak about 6cm) appearing to be larger than the sepal width (peak at 3.2cm).

(e)

Boxplot of *petal length* (cm)Boxplot of *petal width* (cm)

Petal length and petal width for Iris-setosa appear to be much smaller than those from the other two species. There is no overlap in the range of petal length and petal width for Iris-setosa with the other two species.

(f)



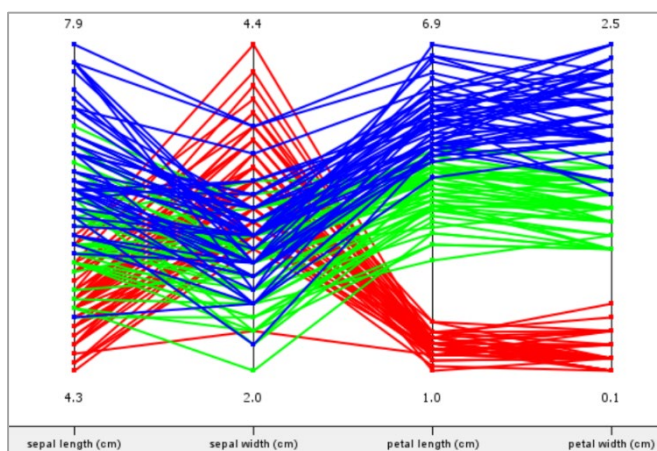
From the scatter matrix, Iris-setosa is well-separated from the other species in all the pairs sepal length & petal length, sepal length & petal width, and petal length & petal width. Some pairs have strong positive associations (sepal length-petal length, sepal length-petal width, petal length-petal width).

(g)

Row ID	D sepal le...	D sepal w...	D petal le...	D petal wi...
sepal length (...)	1	-0.109	0.872	0.818
sepal width (cm)	-0.109	1	-0.421	-0.357
petal length (...)	0.872	-0.421	1	0.963
petal width (cm)	0.818	-0.357	0.963	1

- The pairs sepal length & petal length, sepal length & petal width, and petal length & petal width have strong positive linear relationship.
- Petal length and petal width are highly linearly correlated. This relationship appear to hold among all species.
- Petal length (or petal width) could be a variable which distinguishes Iris-setosa from the other species.

(h)



Consistent with observations above, the species are reasonably well separated for petal length and petal width, but less well separated for sepal length and sepal width.

4.

- 32561
- data mining
- Yes, *workclass* (1836 missing values), *occupation* (1843 missing values) and *native-country* (583 missing values).
- Remove rows that contains missing value because the number of records with missing value for *native-country* is small ($583/32561 = 1.8\%$).
- Yes, *workclass* and *occupation* columns are filtered. This is because the percentage of missing value for *workclass* is $1836/32561 = 5.6\%$ and for *occupation* is $1843/32561 = 5.7\%$, both exceeding the missing value threshold of 5% indicated in the configuration setting.
- 5993
- 55.6%
- Concatenate* node

LAB 5A : Data Exploration using KNIME

Learning Objectives:

1. Create a local workspace and a workflow in KNIME.
2. Import a dataset in KNIME.
3. Explore the dataset by graphical and numerical summaries in KNIME.

Task: The Pima Indians Diabetes Dataset

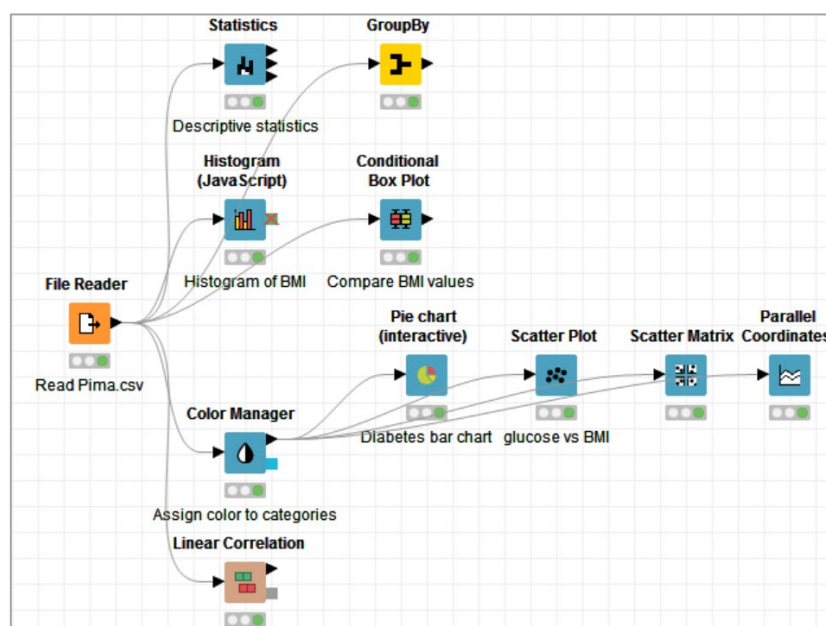
This dataset consists of female diabetic patients who are at least 21 years of age with Pima Indian heritage, residing near Arizona, USA. For now, we want to explore the dataset to understand the data and possibly identify preliminary patterns and insights. Ultimately, the objective is to predict the onset of diabetes based on some diagnostic measurements.

The dataset is available at the UCI Machine Learning Repository (<https://archive.ics.uci.edu/ml/datasets/pima+indians+diabetes>). It is originally from the National Institute of Diabetes and Digestive and Kidney Diseases in the US.

The variables and their descriptions are as follows:

Variable	Description
<i>pregnancy</i>	Number of times pregnant
<i>glucose</i>	Plasma glucose concentration at 2 hours in an oral glucose tolerance test
<i>BP</i>	Diastolic blood pressure (mmHg)
<i>triceps</i>	Triceps skin fold thickness (mm)
<i>insulin</i>	2-hour serum insulin ($\mu\text{U/ml}$)
<i>BMI</i>	Body mass index (weight in kg/(height in m) ²)
<i>DPF</i>	Diabetes pedigree function, a measure of genetic influence on the hereditary risk of diabetes
<i>age</i>	Age (years)
<i>Diabetes</i>	Class variable (0 or 1) where 1 = patient at risk of having diabetes, 0 = patient not at risk of having diabetes

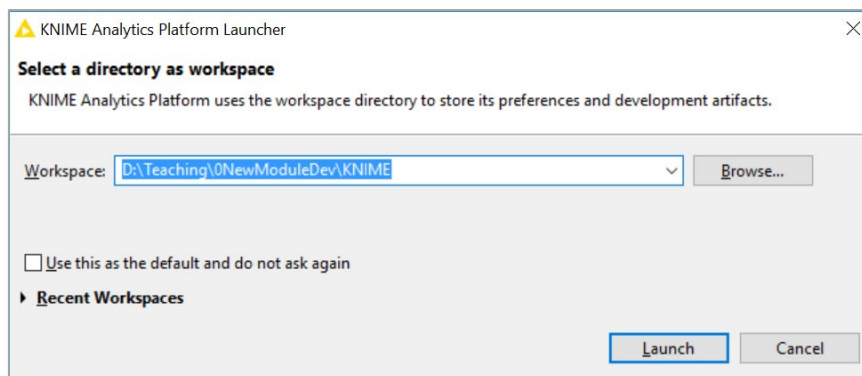
The completed workflow in KNIME should look like this:



A. Create a KNIME Local Workspace

The workspace is the folder/directory in which workflows (and potentially data files) are stored for the current KNIME session.

1. Launch KNIME for the first time.
2. If you do not want to use the default directory indicated, select your desired directory as the workspace by clicking on **Browse**. Then, click **Launch**.



B. Create a KNIME Workflow

1. Launch KNIME.
2. In the upper menu bar, choose **File > New....**
3. Choose **New KNIME Workflow** and click **Next**.
4. Give the workflow a name, for example, “*Pima_explore*”.
5. Use the default destination **LOCAL** (the local workspace that you have selected in *A. Create a KNIME Local Workspace*) to save the workflow and click **Finish**.
(Note: KNIME workflow files are saved as *.knwf)

C. Import the Dataset

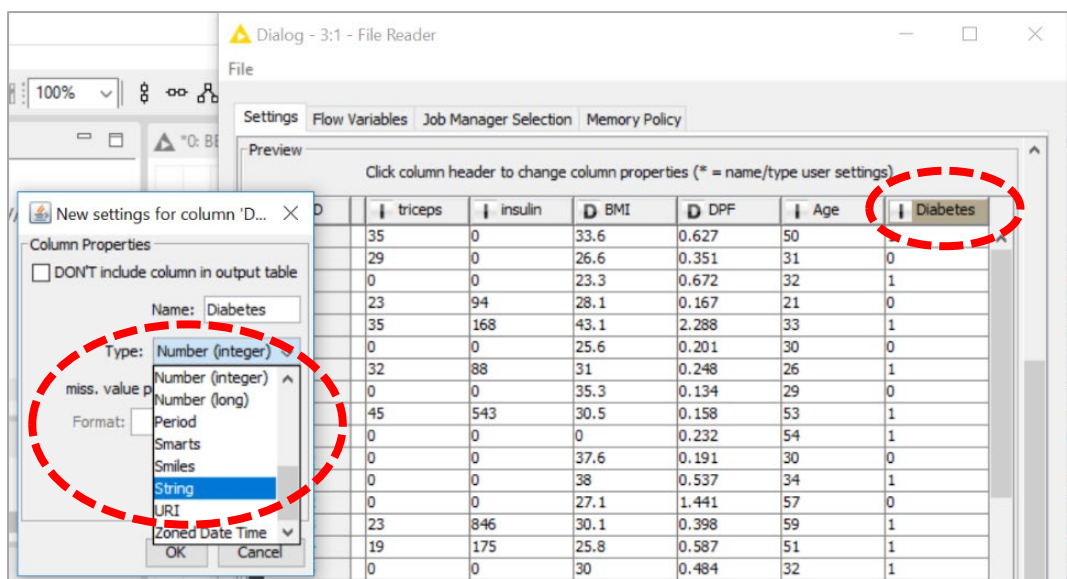
To read data from a CSV (comma-separated values) data file, we use **File Reader** node. The following steps are taken:

1. Drag and drop the **File Reader** node into the *Workflow Editor*. You can search for this node by typing “*File Reader*” in the search box in the *Node Repository*. Alternatively, select **IO** in the *Node Repository* followed by **Read**.
2. In the *Workflow Editor*, right-click on the node and select **Configure**. Alternatively, double-click on the node to configure.
3. Click **Browse**, locate the data file “*Pima.csv*”, and click **Open**. The data will appear in the **Preview** table within the dialog box.

Before loading the data, it is important to check that the data types of the variables are correct. In KNIME, available data types are Double (*D*), Integer (*I*), String (*S*), Date&Time (calendar + clock icon) and Unknown (?).

Notice that the data type of variable *Diabetes* is currently Integer. However, *Diabetes* is categorical, thus we want to change the data type to String. The following steps are taken:

4. In the **Preview** table, click on the *Diabetes* column header and a dialog box will pop up (see screenshot). Under **Type**, select “String” in the drop-down options and click **OK**.



5. Check the data types of other variables and click **OK**.
6. Right-click the node and select **Execute**. The state of the node should now be **Green**.
7. Right-click the node and select **File Table**. You will see the imported data in a tabular format (see screenshot). The number of rows (or records) is reported at the top of the table and the data type of Diabetes is changed to *S* (i.e. String).

File Hilite Navigation View

Table "pima_SAE.csv" - Rows: 392 Spec - Columns: 9 Properties Flow Variables

Row ID	triceps	glucose	BP	triceps	insulin	D BMI	D DPF	I Age	S Diabetes
Row3	1	89	66	23	94	28.1	0.167	21	0
Row4	0	137	40	35	168	43.1	2.288	33	1
Row6	3	78	50	32	88	31	0.248	26	1
Row8	2	197	70	45	543	30.5	0.158	53	1
Row13	1	189	60	23	846	30.1	0.398	59	1
Row14	5	166	72	19	175	25.8	0.587	51	1

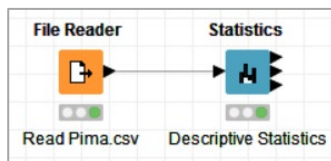
Note that there is one output port for the **File Reader** node.

CHECKPOINT

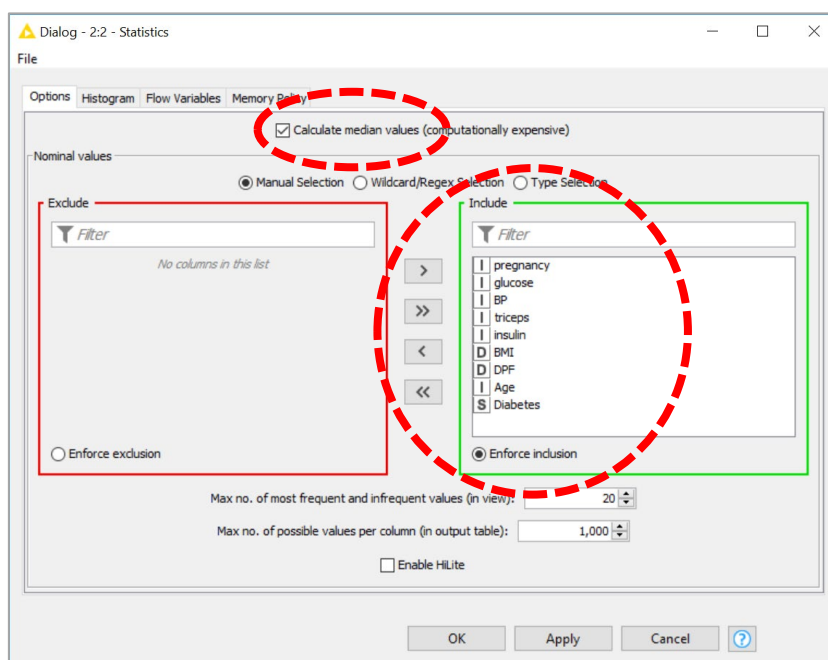
- #1. What is the file extension of the Pima dataset that prompts us to use the *File Reader* node for import?
- #2. How many records are there in the Pima dataset?
- #3. Which is the variable that requires change of data type? Briefly explain why.

D. Generate Numerical Summaries

We use **Statistics** node to generate numerical summaries for the variables in *Pima* dataset. The connection in our workflow should look like this:



1. Locate the **Statistics** node, drag it to the *Workflow Editor* and connect it to the **File Reader** node.
2. Right-click on the **Statistics** node and select **Configure**. Add all attributes to the **Include** panel on the right (see screenshot). Ensure that the checkbox **Calculate median values** is enabled and click **OK**.



3. Right-click on the **Statistics** node and select **Execute and Open Views**. Part of your output should look like this:

Numeric Nominal Top/bottom										
Column	Min	Mean	Median	Max	Std. Dev.	Skewness	Kurtosis	No. Missing	No. +∞	No. -∞
pregnancy	0.0	3.301	2	17	3.2114	1.3356	1.4863	0	0	0
glucose	56	122.6276	119	198	30.8608	0.5178	-0.4832	0	0	0

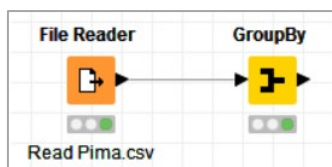
Investigative Task

The **Statistics** node has 3 output ports. Find out what each output table reports.

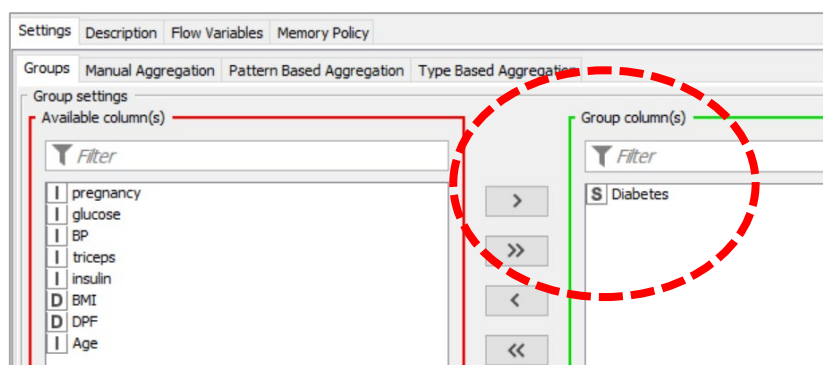
- (a) **Statistics Table**
- (b) **Nominal Histogram Table**
- (c) **Occurrences Table**

E. Generate Numerical Summaries using Groupby

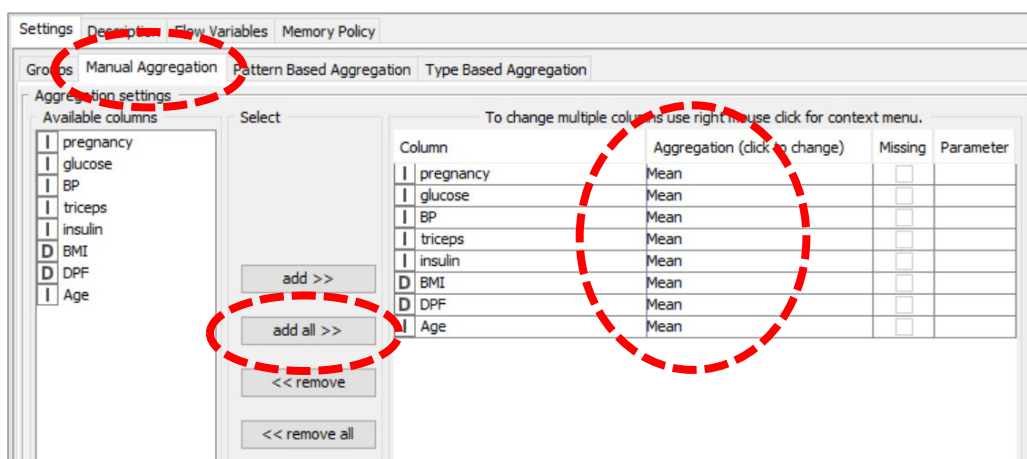
Instead of finding numerical summaries of the whole Pima dataset, we might want to obtain numerical summaries separately for the group at risk of having diabetes (*Diabetes* = 1) and the group with no risk of having diabetes (*Diabetes* = 0). To do so, we generate numerical summaries for the variables in Pima dataset by different groups using **GroupBy** node. The connection in our workflow should look like this:



1. Connect the **GroupBy** node to the **File Reader** node.
2. Right-click on the **GroupBy** node and select **Configure**.
3. Select *Diabetes* and add it to the **Group column(s)** box on the right (see screenshot).



4. Click on the tab **Manual Aggregation** (see screenshot). Add all eight variables to the box on the right. Note that by default **Mean** is selected in the **Aggregation** column for each variable. Click **OK**.

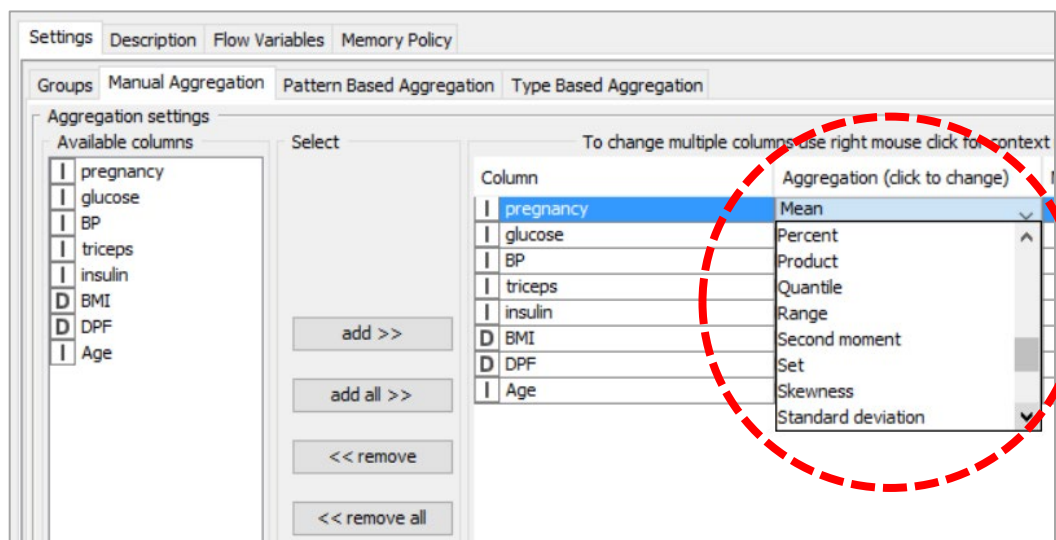


5. Right-click on the **GroupBy** node and select **Execute**.
6. Right-click on the **GroupBy** node and select **Group table**. Your output should look like this:

Row ID	S Diabetes	D Mean(p...	D Mean(g...	D Mean(BP)	D Mean(t...	D Mean(i...	D Mean(B...	D Mean(D...	D Mean(A...
Row0	0	2.721	111.431	68.969	27.252	130.855	31.751	0.472	28.347
Row1	1	4.469	145.192	74.077	32.962	206.846	35.778	0.626	35.938

Suppose now we want to see standard deviation instead of mean. The following steps are taken:

7. Right-click on the **GroupBy** node and select **Configure**.
8. In the tab **Manual Aggregation**, in the **Aggregation** column, click on in each row (variable) to choose **Standard deviation** (see screenshot) from drop-down options. Click **OK**.



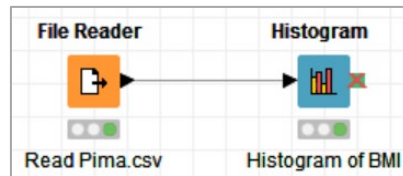
9. **Execute** the node. Then, right-click on the node to select **Group table** to see the table of values.

Investigative Task

Find the minimum and maximum values of BMI and glucose for the group of patients with risk of having diabetes and for the group of patients with no risk of having diabetes.

F. Construct a Histogram

Consider the variable *BMI* in the Pima dataset; it is a continuous variable. Thus, we can use a histogram to visualise its distribution. The connection in our workflow should look like this:

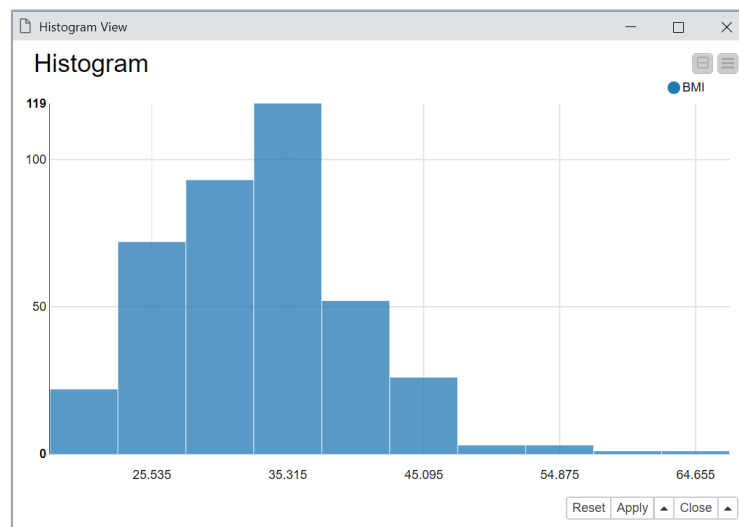


1. Search for **Histogram (JavaScript)** node in the *Node Repository*. Alternatively, look for it in **Views>JavaScript** category. Drag it to the *Workflow Editor*.
2. Connect the **Histogram** node to the **File Reader** node.
3. Right-click on the **Histogram** node and select **Configure**.
4. At **Histogram Column**, select *BMI*. At **Aggregation Method**, select *Occurrence Count* so that height of each bin will correspond to frequency.
5. Click on the tab **Binning**, at **Number of bins**, increase to 10. Click **OK**.
6. Right-click on the node and select **Execute and Open Views**.

Alternatively, after executing the node, right-click on the node and select **Interactive View: Histogram View**.

(Note: As it is a JavaScript viewer, it may take a while for the viewer to appear.)

Your output should look like this:



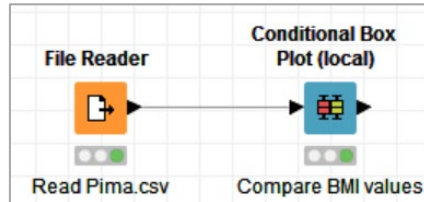
The x-axis gives the bins representing the *BMI* values, and the y-axis gives the counts/frequencies in each bin. Mouse over any of the bars to see more details, such as the occurrence count and the BMI values of the bin.

CHECKPOINT

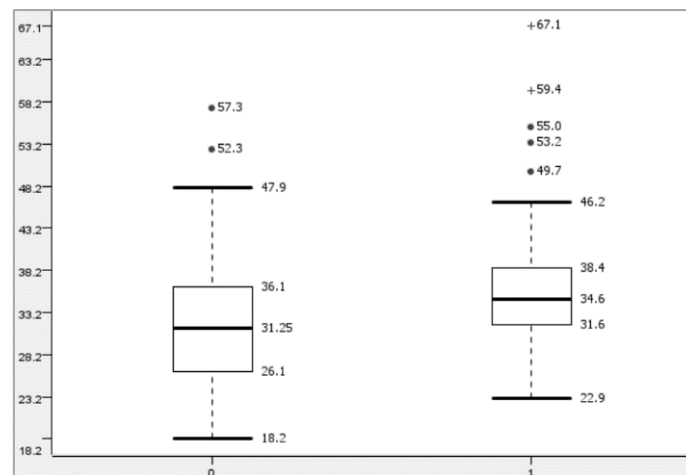
- #1. Recall from the chapter on Descriptive Statistics, describe the shape of distribution of BMI from the histogram plot.
- #2. Why are we not able to generate a histogram for the variable *Diabetes*?

G. Construct a Conditional Box Plot

Suppose we want to compare the *BMI* values (continuous variable) between the two *Diabetes* groups (categorical variable), we can use a side-by-side boxplot, called **conditional box plot** in KNIME. The connection in our workflow should look like this:



1. Search for **Conditional Box Plot (local)** node in *Node Repository*. Alternatively, look for it in **Views>Local (Swing)** category. Drag it to the *Workflow Editor*.
2. Connect the **Conditional Box Plot** node to the **File Reader** node.
3. Right-click on the **Conditional Box Plot** node and click **Configure**.
4. At **Nominal column**, select *Diabetes*. At **Numeric column**, select *BMI*. Click **OK**.
5. Right-click on the node and select **Execute and Open Views**.
Alternatively, after executing the node, right-click on the node and select **View: Conditional Box Plot**. Your output should look like this:

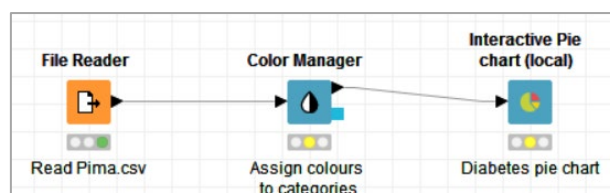


CHECKPOINT

- #1. Recall from the chapter on Descriptive Statistics, a boxplot is effective for comparing multiple groups of data using five-number summary. Hence, extract the five-number summary of the *BMI* values for the two *Diabetes* groups.
- #2. Identify outliers, if any.

H. Construct a Pie Chart

Consider the variable *Diabetes*, which is categorical. We can use a pie chart to visualise the proportions of the categories. To do this in KNIME, we use the **Interactive Pie chart (local)** node. Optionally, to make the pie chart more readable or aesthetically pleasant, we might want to use **Color Manager** node to assign different colours to different categories. The connections in our workflow should look like this:

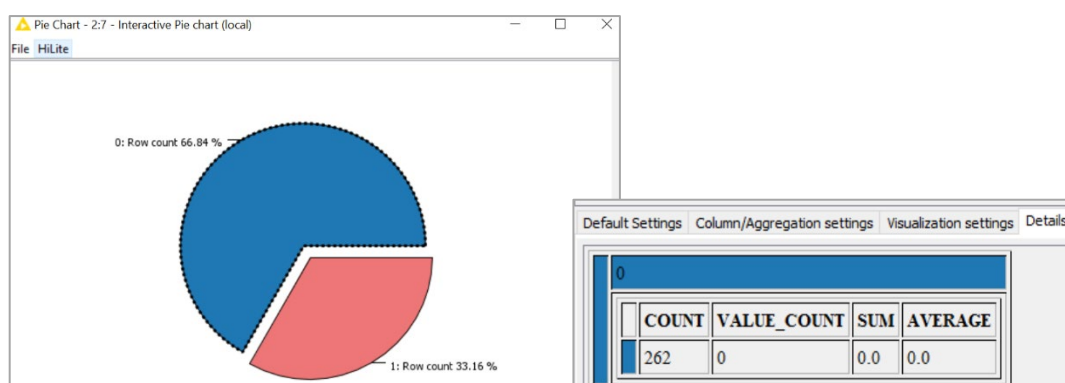


Let's first use **Color Manager** node to assign **red** to the group at risk of diabetes and **blue** to the group not at risk. The following steps are taken:

1. Search for the **Color Manager** node in *Node Repository*. Alternatively, look for it in **Views>Property**. Drag and drop the **Color Manager** node to the *Workflow Editor*.
2. Connect the **Color Manager** node to the **File Reader** node.
3. **Configure** the **Color Manager** node.
4. At **Select one Column**, select the categorical variable *Diabetes*. Then, in the panel just below, assign red colour to value 1 and blue colour to value 0. Click **OK**.
5. **Execute** the **Color Manager** node.

Next, we construct the pie chart. The following steps are taken:

6. Search, drag and drop the **Interactive Pie chart (local)** node to the *Workflow Editor*.
7. Connect the **Color Manager** node to the **Interactive Pie chart (local)** node.
8. **Configure** the **Interactive Pie chart (local)** node.
9. At **Pie column**, select *Diabetes*. At **Aggregation column**, select *<none>*. Click **OK**.
10. Right-click on the **Interactive Pie chart (local)** node and select **Execute and Open Views**. Alternatively, **Execute** the node, then right-click and select **View: Pie Chart**. Click on any part of the pie to see more details (e.g. occurrence count). Your pie chart should look like this:

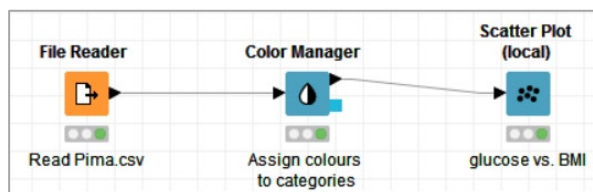


🔍 CHECKPOINT 🔍

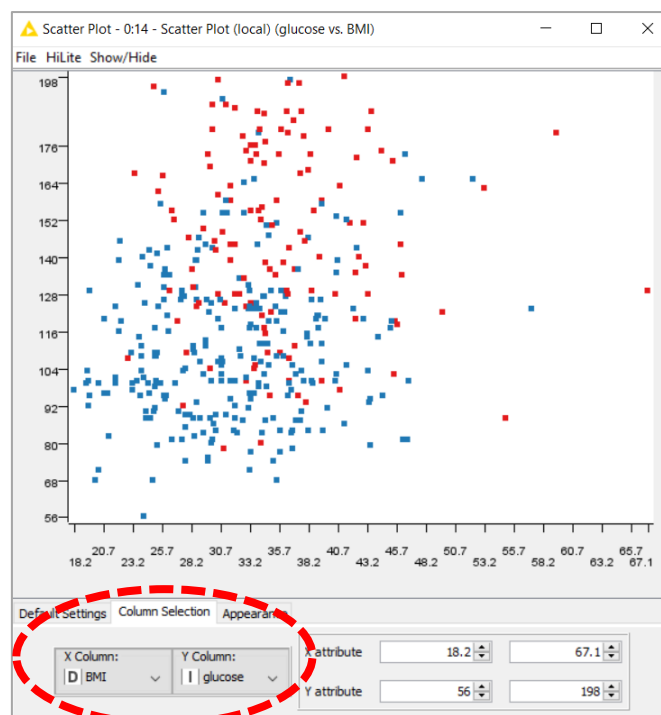
Recall from the chapter on Descriptive Statistics, a pie chart gives a visual overview of proportions belonging to each category. Hence, describe the overview of proportions of patients with risk of having diabetes using the pie chart generated.

I. Construct a Scatterplot

We use a scatterplot to see if there is any linear relationship between quantitative variables *BMI* and *glucose*. Since KNIME allows customization of visual properties like size, shape and colour (via **Size Manager** node, **Shape Manager** node and **Color Manager** node respectively), so we can actually visualize up to five different columns of values on a 2D scatterplot. For our scenario, we shall use only the colour property via the **Color Manager** node. The connections in our workflow should look like this:



1. Search for the **Scatter Plot (local)** node in *Node Repository*, drag it to the *Workflow Editor* and connect it to the **Color Manager** node.
2. Right-click on the **Scatter Plot (local)** node and select **Execute and Open Views**. Alternatively, **Execute** the node, then right-click and select **View: Scatter Plot**.
3. Click on the tab **Column Selection**, and choose *BMI* as the **X Column** and *glucose* as the **Y Column**. Your output should look like this:



CHECKPOINT

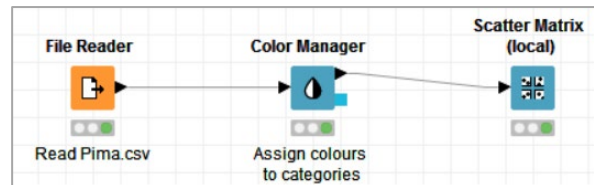
Recall from the chapter on Descriptive Statistics, a scatterplot shows the nature of a relationship between two quantitative variables. When looking at scatterplot, we should look out for:

- Overall pattern – how strong it is and its direction
- Deviations from pattern
- Outliers

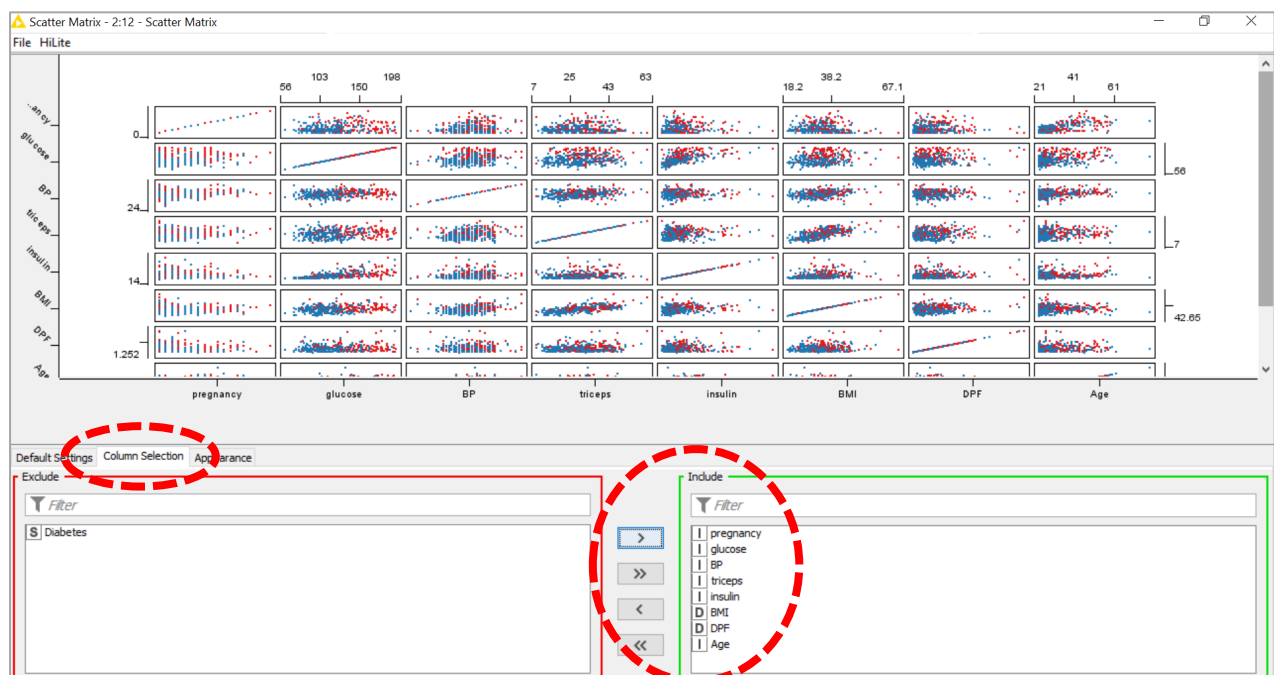
Comment on the relationship between BMI and glucose.

J. Construct a Scatter Matrix

We use a scatter matrix to see if there is any linear relationship between every pair of variables. It is just a matrix of scatterplots! In a scatter matrix, element E_{ij} is a scatterplot of data columns i and j , where the values of the i -th column are displayed on the x -axis and the values of the j -th column on the y -axis, while the coordinates are displayed alternating on all sides of the plot. The connections in our workflow should look like this:



1. Search for the **Scatter Matrix (local)** node in *Node Repository*, drag it to the *Workflow Editor* and connect it to the **Color Manager** node.
2. Right-click on the **Scatter Matrix (local)** node and select **Execute and Open Views**. Alternatively, **Execute** the node, then right-click and select **View: Scatter Matrix**.
3. Click on the tab **Column Selection**, and add all variables, except *Diabetes*, to the **Include** panel. Your output should look like this:

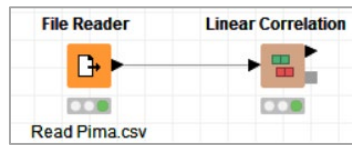


🔍 CHECKPOINT 🔍

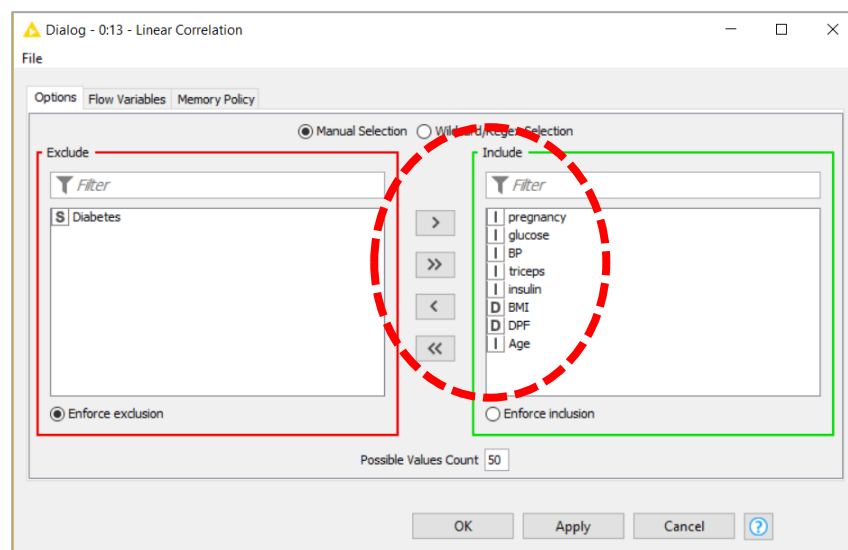
- #1. Why do you think we excluded the variable *Diabetes* to display in the scatter matrix?
- #2. From the upper triangular matrix plots, which pair(s) of variables may have positive association?

K. Generate Correlation Coefficient

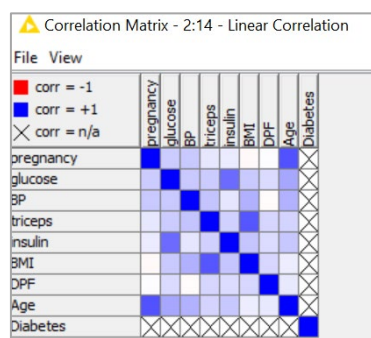
To quantify strength and direction of the linear relationship between two quantitative variables, we compute the correlation coefficient. In KNIME, we can use **Linear Correlation** node to measure the correlation between any pair of quantitative variables. The connection in our workflow should look like this:



1. Search for the **Linear Correlation** node in *Node Repository*, drag it to the *Workflow Editor* and connect it to the **File Reader** node.
2. **Configure** the **Linear Correlation** node.
3. Add all quantitative variables to the **Include** panel (see screenshot).



4. Right-click on the node and select **Execute and Open Views**. Alternatively, **Execute** the node, then right-click and select **View: Correlation Matrix**. Your output should look like this:



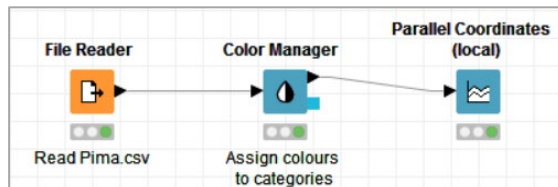
It gives an overview of the correlation values with colour codes. Mouse over the elements in the matrix to see the correlation value.

CHECKPOINT

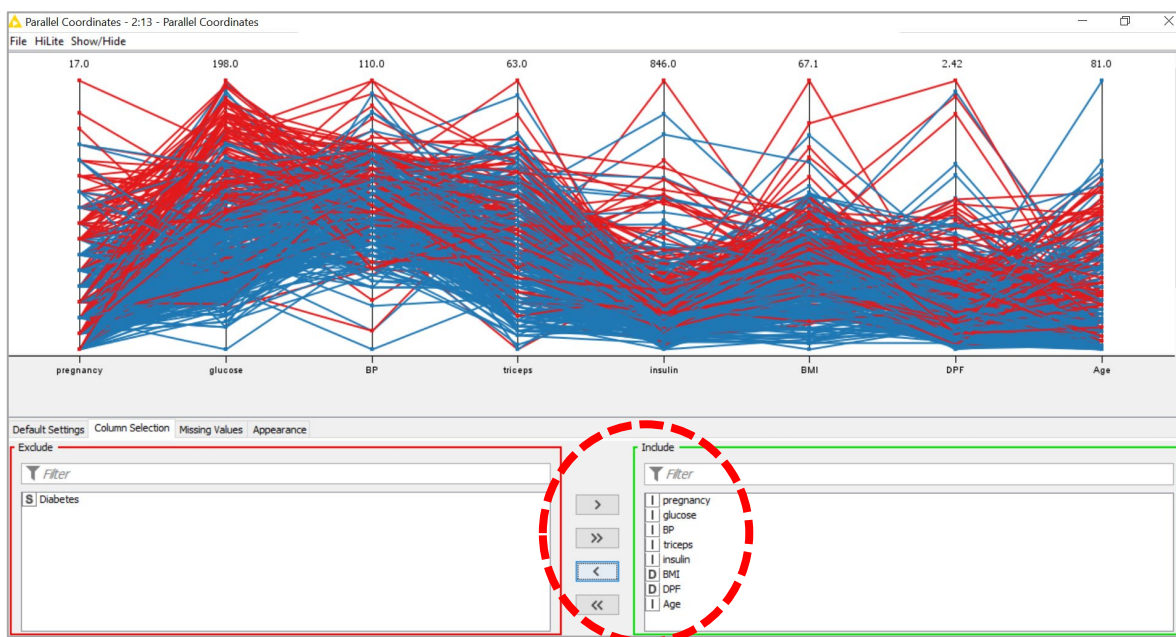
What can correlation coefficient tells us about the relationship between two variables?

L. Construct Parallel Coordinates

We use a parallel coordinates plot to get an idea of pattern-groups across columns. A parallel coordinates plot unfolds the column names along the x -axis and displays each column value on a separated y -axis. As a result, a data point (a row) is mapped as a line connecting values across variables. The connections in our workflow should look like this:



1. Search for the **Parallel Coordinates (local)** node in *Node Repository*, drag it to the *Workflow Editor* and connect it to the **Color Manager** node.
2. Right-click on the **Parallel Coordinates (local)** node and select **Execute and Open Views**. Alternatively, **Execute** the node, then right-click and select **View: Parallel Coordinates**.
3. Click on the tab **Column Selection**, and add all variables, except *Diabetes*, to the **Include** panel. Your output should look like this:



Investigative Task

Let's explore the parallel coordinates plot more so that we can interpret it. What does each line in the parallel coordinates plot means?

1. Go to *File Reader* node, right-click and select *File Table* to display the Pima data table. Click on *Row4*, go to menu, click *HiLite* and choose *HiLite Selected*. You will see *Row4* highlighted.
2. Go to *Parallel Coordinates (local)* node, right-click and select *View: Parallel Coordinates* to display the plot. Go to menu, click *Show/Hide* and select *Show hilited only*.
3. Mouse over the dots along the line and compare the variable values with *Row4* record in the data table. For example, what is the glucose and BP values shown in the parallel coordinates plot for the *hilited* line? Are the values the same as the *File Table* for record *Row4*?

LAB 5B :

Data Preparation and Cleaning using KNIME

Learning Objectives:

1. Use different filter nodes.
2. Rename column and replace numeric column with attribute column.
3. Create table and handle missing values.

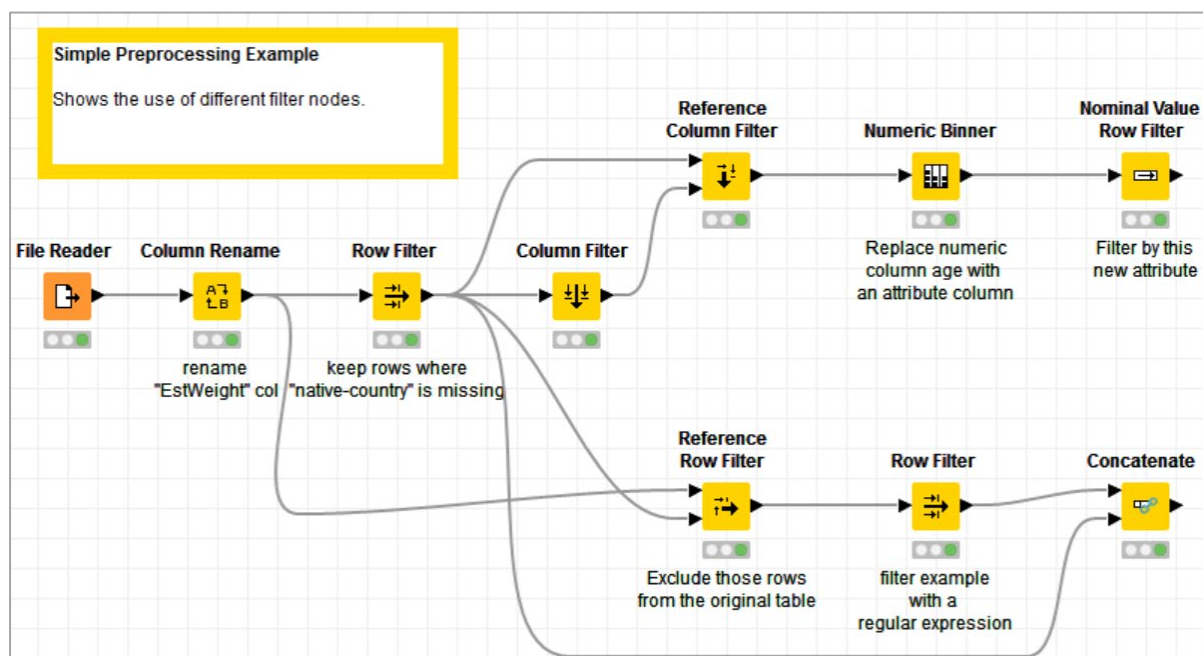
Task 1: The Census Bureau (adult) Dataset

An individual's annual income is a result of various factors. Intuitively, it is influenced by the individual's education level, age, gender, occupation, etc. The dataset contains 15 columns with the *Income* as the target attribute. *Income* is divided into two classes: $\leq 50K$ and $> 50K$. There are 14 attributes which are mainly demographics and features to describe a person.

The dataset is available at the UCI Machine Learning Repository (<https://archive.ics.uci.edu/ml/datasets/adult>). This data was extracted from the census bureau database found at <http://www.census.gov/ftp/pub/DES/www/welcome.html>. (The training data set is used here.)

To perform data mining task, the dataset needs to be prepped first. We will explore the use of different filter nodes and basic manipulation nodes available in KNIME for simple data preparation and cleaning.

The completed workflow in KNIME should look like this:



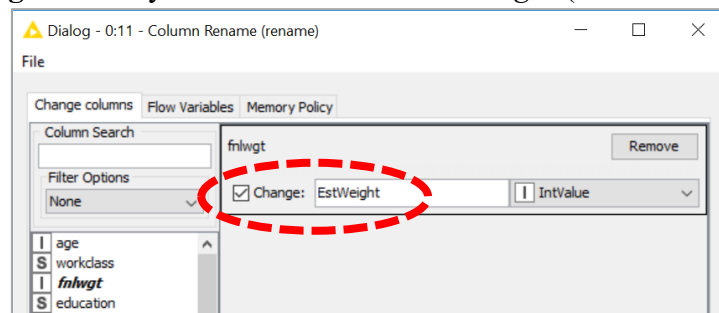
A. Rename Column

We want to rename the variable *fnlwgt* to better reflect its description as an estimated weight, *EstWeight*. This variable is actually a term estimate. The term estimate refers to population totals derived from multiple files by creating ‘weighted tallies’ of any specified socio-economic characteristics of the population.

1. Drag and drop the **File Reader** node into the *Workflow Editor*.
2. **Configure** the **File Reader** node. Import the dataset “*adult.csv*” into the workflow.
3. Search for the **Column Rename** node, drag it to the *Workflow Editor* and connect it to the **File Reader** node.
4. **Configure** the **Column Rename** node.

We want to rename the *fnlwgt* column to *EstWeight*. The following steps are taken:

5. On the left panel, double-click on *fnlwgt* variable.
6. Check the **Change** box. Key in the new name as *EstWeight* (see screenshot). Click **OK**.

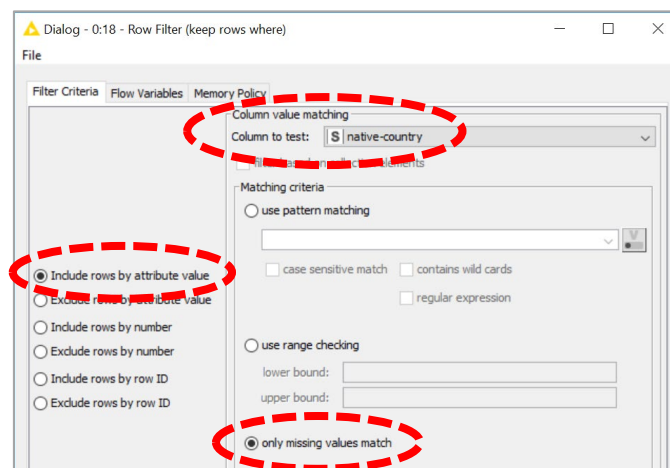


7. **Execute** the node. Then, right-click on the node to select the **Renamed/Retyped table**, which will reflect the new column name.

B. Filter Rows by Row Filter Node

We observed that the variable *native-country* has missing values and we want to study these records separately. Let’s filter the rows using **Row Filter** node to keep/include all rows with missing values for *native-country*.

1. Connect the **Row Filter** node to the **Column Rename** node. **Configure** the node:



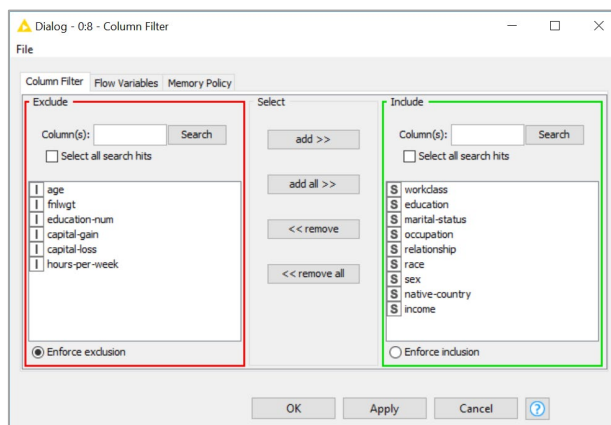
2. **Execute** the node, and right-click to select **Filtered** table.

C. Filter Columns by Column Filter and Reference Column Filter Nodes

Next, we want to create two separate data tables from the missing values records – one table that comprises all the numeric variables (columns) and the other table that comprises the non-numeric variables (columns).

Filter columns by **Column Filter** node:

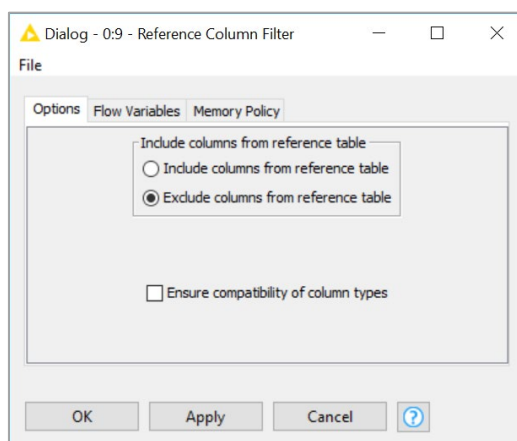
1. Search for the **Column Filter** node, and connect it to the previous **Row Filter** node.
2. **Configure** the **Column Filter** node. Under the **Exclude** panel on the left, select variables of Integer data type to be excluded (see screenshot). Under the **Include** panel on the right, select variables of String data type to be included. Click **OK**.



3. **Execute** the node, and right-click to select **Filtered table**. This data table will consist of rows where *native-country* is missing and only non-numeric columns.

Filter columns by **Reference Column Filter** node:

4. Search for the **Reference Column Filter** node and drag it to the *Workflow Editor*. This node has two input ports – *Table to be filtered* and *Reference table*.
5. Connect the first input port (*Table to be filtered*) to the previous **Row Filter** node. This is the data source that we want to filter.
6. Connect the second input port (*Reference table*) to the previous **Column Filter** node. This is the reference data that we want to reference for filtering.
7. **Configure** the **Reference Column Filter** node. Check the radio button **Exclude columns from reference table** (see screenshot). Click **OK**.

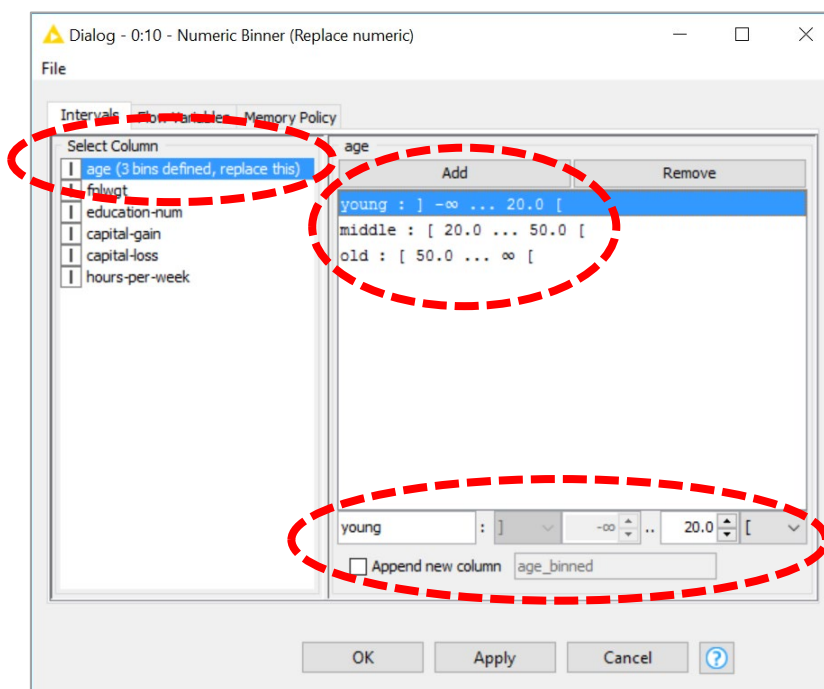


8. **Execute** the node, and right-click to select **Filtered table**. This data table will consist of rows where *native-country* is missing and only numeric columns.

D. Replace Numeric Column with Binned, String-type Columns (Intervals)

We want to replace the numeric data column *age* with binned data column, where *young* refers to anyone <20 years old, *middle* refers to anyone between 20 and 50 (exclusive), and *old* refers to anyone >= 50.

1. Search for the **Numeric Binner** node, and connect it to the previous **Reference Column Filter** node (whose output consists of numeric columns only).
2. **Configure** the **Numeric Binner** node (see screenshot). At **Select Column**, select *age*.
3. Click **Add** 3 times to add 3 bins.



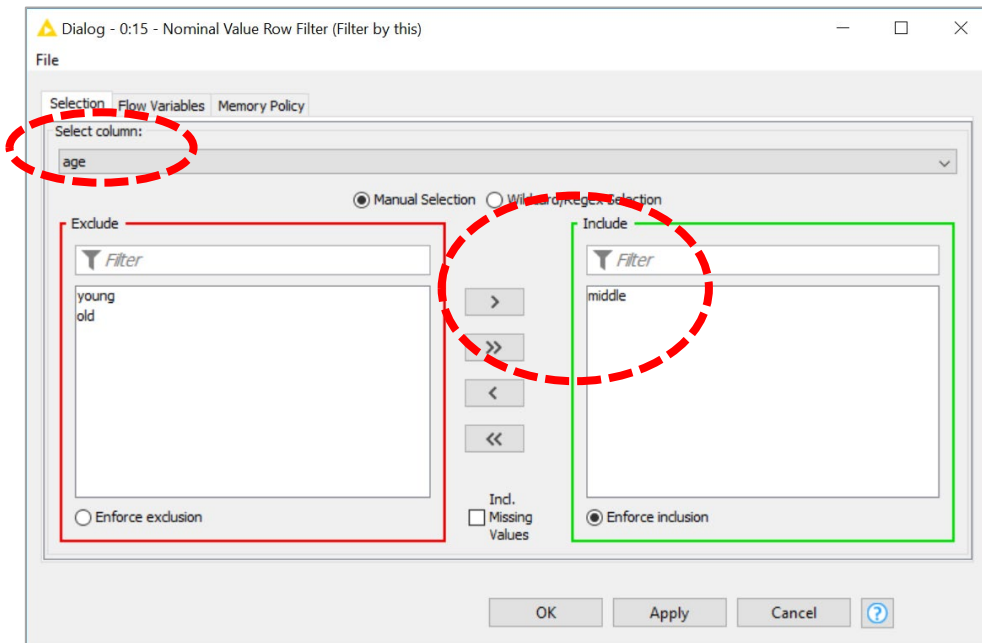
4. For the first bin, name it *young* and key in 20 as the end value of this bin.
5. For the second bin, name it *middle*, and key in 20 as the start value and 50 as the end value of this bin.
6. For the third bin, name it *old* and key in 50 as the start value of this bin. Click **OK**.
7. **Execute** the node, and right-click to select the **Binned Data**. This data table will reflect the change of *age* column from numeric to String data type (*young/middle/ old*). Part of your output should look like this:

Row ID	age	EstWeight	educati...	capital-...	capital-...	hours-p...
Row14	middle	121772	11	0	0	40
Row38	middle	84154	10	0	0	38
Row51	young	226956	9	0	0	30
Row61	middle	293936	4	0	0	40
Row93	middle	117747	9	0	1573	35
Row245	old	203580	9	0	0	35
Row249	middle	153141	9	0	0	40
Row297	middle	157443	14	3464	0	40
Row393	middle	98101	13	7688	0	45
Row453	middle	197583	12	0	0	40
Row557	middle	323069	9	0	0	20

E. Filter Rows by Nominal Value Row Filter Node

Now that we have replaced numeric values of *age* with categorical values, we want to further filter the data such that only records belonging to the *middle* age group are kept/included.

1. Search for the **Nominal Value Row Filter** node, and connect it to the previous **Numeric Binner** node.
2. **Configure** the **Nominal Value Row Filter** node (see screenshot). At **Select column**, select *age*. Under the **Include** panel on the right, select the category *middle* to be included. Click **OK**.



3. **Execute** the node, and right-click to select the **Included** data. This data table will consist of rows where *native-country* is missing, numeric columns only, and rows where *age* group is *middle*.

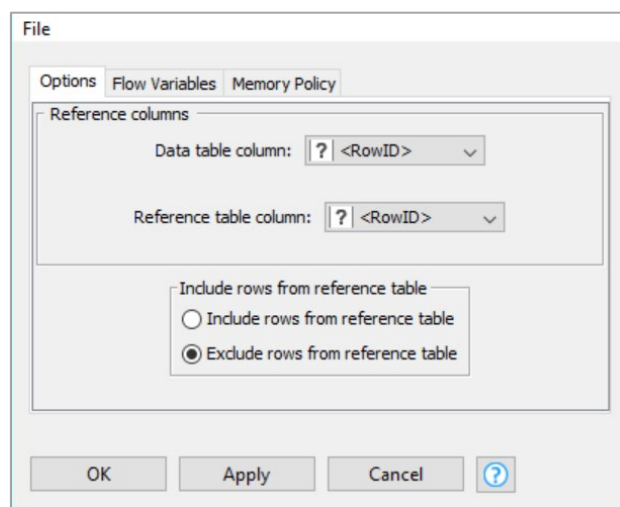
CHECKPOINT

- #1. How many records are there after the 'row filtering' action (with missing values of *native-country* included)?
- #2. Do we expect the same number of records from the output table of 'column filtering' and the output table of 'reference column filtering'? Why?
- #3. What is the difference between **Column Filter** node and **Reference Column Filter** node in terms of their functionality?

F. Filter Rows by Reference Row Filter and Row Filter Nodes

Starting again, suppose now we want to make use of the **Reference Row Filter** node to filter the records (rows) where *native-country* is not missing. The following steps are taken:

1. Search for the **Reference Row Filter** node, drag it to the *Workflow Editor*. This node has two input ports – *Table to be filtered* and *Reference table*.
2. Connect the first input port (*Table to be filtered*) to the previous **Column Rename** node. This is the data source that we want to filter.
3. Connect the second input port (*Reference table*) to the previous **Row Filter** node. This is the reference data that we want to reference for filtering.
4. **Configure** the **Reference Row Filter** node. Check the radio button **Exclude rows from reference table** (see screenshot). Click **OK**.



5. **Execute** the node, and right-click to select **Filtered table**. This data table will consist of rows where *native-country* is non-missing.

Next, we want to continue to filter the records where *education* follows the pattern of *th where * is a wildcard (e.g. *fourth*, *fifth*, *sixth*, etc). This can be done using **Row Filter** node, in these steps:

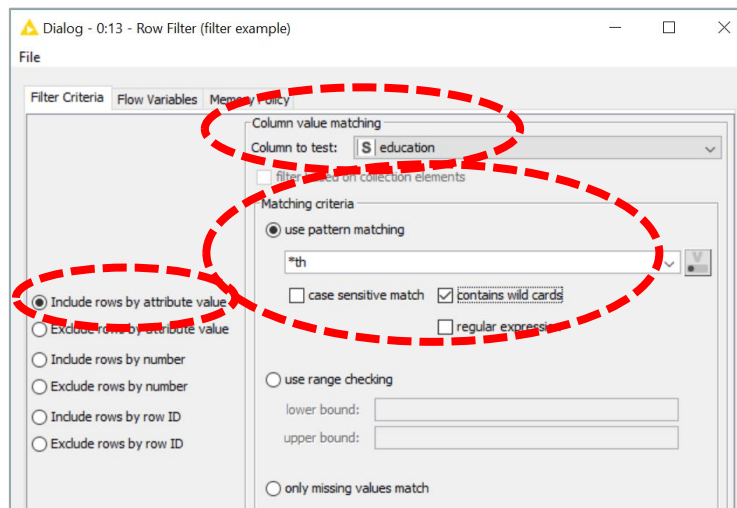
6. Search for the **Row Filter** node, and connect it to the **Reference Row Filter** node.

7. **Configure the Row Filter node (see screenshot).**

Check the radio button **Include rows by attribute value**.

At **Column to test**, select *education*.

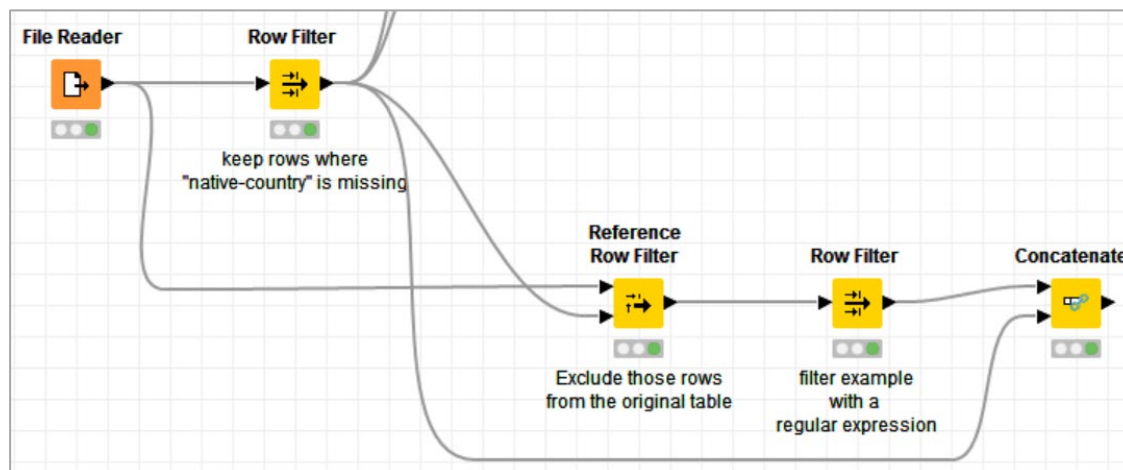
At **Matching criteria**, check the radio button **use pattern matching**, key in **th*, and enable **contains wild cards**. Click **OK**.



8. **Execute the node, and right-click to select the **Filtered** data.** This data table will consist of rows where *native-country* is non-missing and *education* matches the pattern **th*.

Investigative Task

Let's explore *Concatenate* and *Joiner* nodes.



(a) If we connect the *Concatenate* node in the workflow above, what does it do? Explain in context.

(b) What is the difference between the *Concatenate* node and the *Joiner* node?

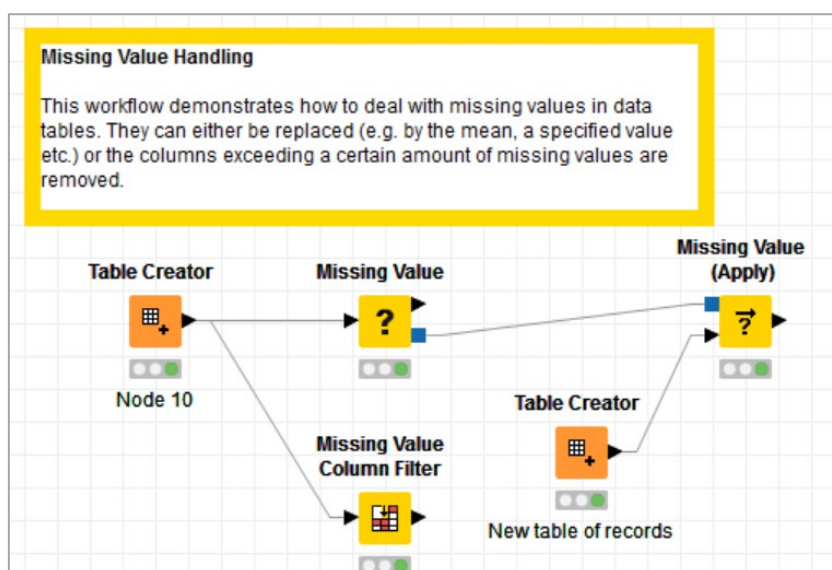
Task 2: Handling Missing Values

How should we handle missing values in our dataset?

The most convenient way is to discard records that are incomplete (i.e. missing values). However, we could possibly throw away precious information present in data. If the number of missing values is small, there is less issue with discarding. But if the number of missing values is substantial, we might want to replace or impute the missing values with a *typical* data value. For missing values belonging to a **categorical variable, we can impute with the mode of the variable**. As for missing values belonging to a **numeric variable, we can impute with the mean or median of the variable**.

In this task, we will create data tables, and take a closer look at how to handle missing values in KNIME using **Missing Value** and **Missing Value Column Filter** nodes.

The completed workflow in KNIME should look like this:



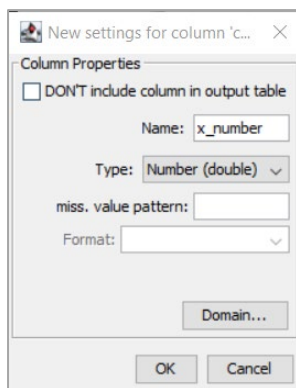
A. Create a Data Table

Before handling missing values, we shall use the **Table Creator** node to create a simple dataset in a data table. The small dataset will have 10 records/rows and 3 variables/columns.

- First column → numbers with some missing values
- Second column → letters with some missing values
- Third column → letters with no missing value (i.e. complete)

The following steps are taken:

1. Search for the **Table Creator** node, and drag it to the *Workflow Editor*.
2. **Configure** the **Table Creator** node.
3. Right-click on the header of the first column and select **Column Properties...**, or simply double-click on the header; a dialog box will appear (see screenshot).
At **Name**, key in *x_number*. At **Type**, select **Number (double)**. Click **OK**.
Repeat for second column with **Name** *y_letter* and **Type** *String*, then third column with **Name** *z_complete* and **Type** *String*.



4. Key in data into the cells in the table according to the following:

Row ID	x_number	y_letter	z_complete
Row0	10	a	a
Row1	6	b	b
Row2			c
Row3	4		d
Row4	11	e	e
Row5			f
Row6	9	g	g
Row7	12	a	h
Row8	10	a	i
Row9	6	b	j

5. Click **OK**. **Execute** the node, and right-click to select **Manually created table**. Your output should look like this:

Row ID	x_number	y_letter	z_complete
Row0	10	a	a
Row1	6	b	b
Row2	?	c	
Row3	4	?	d
Row4	11	e	e
Row5	?	f	
Row6	9	g	g
Row7	12	a	h
Row8	10	a	i
Row9	6	b	j

'?' in the cells indicate missing value.

B. Impute Missing Values

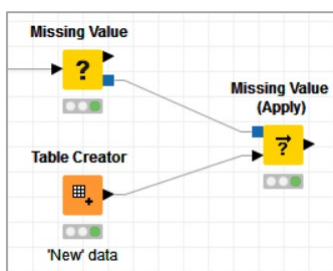
We use the **Missing Value** node to impute missing values. Common practice is to impute missing categorical values with mode and missing numerical values with mean (or median).

The following steps are taken:

1. Search for the **Missing Value** node, and connect it to the previous **Table Creator** node.
2. **Configure** the **Missing Value** node.
3. For *Number (double)* attributes, select **Mean** from the drop-down options. This computes the mean value of all non-missing cells in a column and replaces the missing values with this mean value.
4. For *String* attributes, select **Most Frequent Value** from the drop-down options. This finds the mode (most frequent value) in a column and replaces the missing values with this mode. Click **OK**.
5. **Execute** the node, and right-click to select **Output table**. Check that the missing values have been imputed accordingly.

Sometimes new data will come along and we want to apply the same rules of handling missing values as before. To do so, we use the **Missing Value (Apply)** node.

But first, observe that the **Missing Value** node has 2 output ports – a data port (black triangle) and a model port (blue square). Conversely, the **Missing Value (Apply)** node has 2 input ports – a model port (blue square) and a data port (black triangle). The connection in our workflow should look like this:



The following steps are taken:

6. Use another **Table Creator** node to create the following table of new records:

x_number	y_letter
7	a
	b
9	

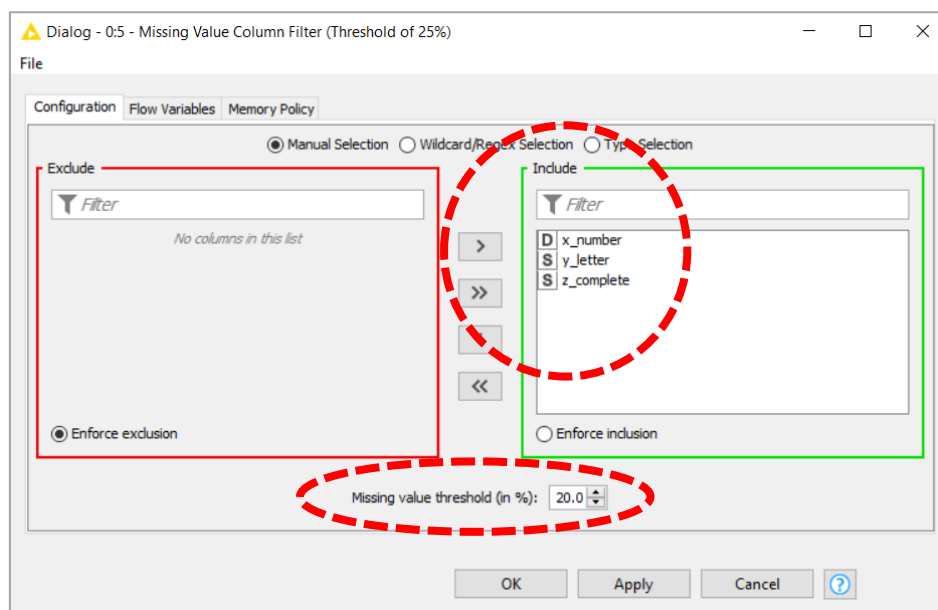
7. Search for the **Missing Value (Apply)** node, and connect the data input port to the previous **Table Creator** node from step 6 (black triangle to black triangle).
8. Connect the model input port to the model output port of the previous **Missing Value** node (blue box to blue box). This is to reuse the rules to handle the missing values in the data table created in step 6.
9. **Execute** the **Missing Value (Apply)** node, and right-click to select **Output table**. Your output should look like this:

Table "default" - Rows: 3 Spec - Columns: 2 Properties Flow Variables		
Row ID	D x_number	S y_letter
Row0	7	a
Row1	8.5	b
Row2	9	a

C. Filter to Remove Missing Values

Suppose we want to remove columns which have substantial missing values. We can use the **Missing Value Column Filter** node where a threshold can be set for a column with missing values to be removed. The following steps are taken:

1. Search for the **Missing Value Column Filter** node, and connect it to the first **Table Creator** node.
2. **Configure** the **Missing Value Column Filter** node (see screenshot).
Under the **Include** panel on the right, select all 3 variables be included.
At **Missing value threshold (in %)**, key in 20. Click **OK**.



3. **Execute** the node, and right-click to select **Filtered table**. This data table contains only *z_complete* column. The *x_number* and *y_letter* columns are removed because they have 2 (20%) and 3 (30%) missing values respectively, which is \geq to the specified threshold of 20%.

CHECKPOINT

- #1. What is the difference in functionality between **Missing Value** and **Missing Value (Apply)** nodes?
- #2. Which column(s) will remain if we set the following thresholds in **Missing Value Column Filter** node?
 - Threshold = 10% : _____
 - Threshold = 30% : _____
 - Threshold = 31% : _____