
Anomaly Detection in Predictive Maintenance

Time Alignment and Visualization

Phil Winters
Rosaria Silipo

Phil.Winters@knime.com
Rosaria.Silipo@knime.com

Table of Contents

Anomaly Detection in Predictive Maintenance <i>Time Alignment and Visualization</i>	1
Summary	3
Anomaly Detection: Ideal Prediction Foresees Failures Before They Occur	3
The Data	4
Data Pre-processing	5
Reading Data	5
Frequency Binning	6
Time Alignment	6
Visualization	7
Time Series Line Plots	8
Scatter Matrix	9
Correlation Maps	10
Auto-Correlation Maps	11
Heatmap	12
Conclusions	13

Summary

In this whitepaper, we present the start of a new project about the analysis of time series data, originating from a network of sensors around a working rotor. The rotor has one breakdown episode which we would like to predict, not only to avoid major damage of course, but also to keep extend the service life of the rotor and amortize its costs.

This is a classic problem of anomaly detection: predicting a major damage episode with no other examples than its available history of normal functioning. Anomaly detection is a very challenging data analytics use case. Here, we cannot use the classical supervised classification algorithms, because no known similar damage episodes are available at any point of the historical data. The challenge then is to classify something we have never seen before: it is more or less like having a crystal ball!

There are a few established techniques in the area of anomaly detection. However, if and when they fall short, creative skills are required.

This is the first of a series of whitepapers to deal with anomaly detection from different perspectives. In particular, here we describe the first project steps: pre-processing and visualization of the sensor based time series data. Before we proceed with more complex data analytics, we need to clean and suitably quantify the data and visually explore their evolution.

This whitepaper prepares the FFT processed time series data from the rotor sensors for further analytics by averaging spectral amplitudes by date and frequency bin and performing the time alignment across all of the time series. Time alignment and frequency binning are not uncommon procedures in the analysis of sensor data, like for example Internet of Things data sets. Next, the time series evolution is explored visually by means of five different visualization techniques: line plots, scatter matrices, auto-correlation maps, correlation maps, and heat maps.

The workflow used in this whitepaper is available on the KNIME EXAMPLES server under 050_Applications/050017_AnomalyDetection. This workflow group also contains a reduced data set in folder data, to run the workflow. The full original data set can be downloaded from <http://www.knime.org/files/AnomalyDetectionFullDataSet.zip>

Anomaly Detection: Ideal Prediction Foresees Failures Before They Occur

We are all witnessing the current explosion of data: social media data, clinical data, system data, CRM data, web data, and lately tons of sensor data! With the advent of the Internet of Things, systems and monitoring applications are producing humongous amounts of data which undergo evaluation to optimize costs and benefits, predict future events, classify behaviors, implement quality control, and more. All these use cases are relatively well established by now: a goal is defined, a target class is selected, a model is trained to recognize/predict the target, and the same model is applied to new never-seen-before productive data.

The newest challenge now lies in predicting the “unknown”. The “unknown” is an event that is not part of the system past, an event that cannot be found in the system historical data. In the case of network data the “unknown” event can be an intrusion, in medicine a sudden pathological status, in sales or credit card businesses a fraudulent payment, and finally, in machinery, a mechanical piece breakdown. A high value, in terms of money, life expectancy, and/or time, is usually associated with the early discovery, warning, prediction, and/or prevention of the “unknown” and, most likely, undesirable event.

Specifically, prediction of “unknown” disruptive events in the field of mechanical maintenance takes the name of “anomaly detection”.

If you look in the accredited literature or surf on the web, you will find that the term “anomaly detection” is actually used to indicate very different problems relying on different data analytics approaches. We could summarize the contexts in which the term “anomaly detection” is used with three possible situations:

- **Supervised Anomaly Detection**

A group of records is labeled as “anomaly” and the goal is to classify the records as such. This is a typical problem, for example, in medicine, where a recurring or chronic anomaly needs to be properly classified for a correct diagnosis. This, however, is just a classification problem, where one of the classes happens to be labeled as “anomaly”.

- **Static Unsupervised Anomaly Detection**

A number of labeled classes are available in the historical data, but suddenly a weird unrecognized outlier pattern shows up and does not fit any of the available class typologies. Like a random unknown heartbeat in the middle of a series of standard normal heartbeats during an ECG session. Either benign or worrisome, an alarm must be triggered.

- **Dynamic Unsupervised Anomaly Detection**

Here some measures change slowly over time, indicating a slow deviation from the past standards until a catastrophic event occurs. For example, while a motor is slowly deteriorating, one of the measurements will change gradually until eventually the motor breaks. On the one hand we want to keep the motor running as long as possible – mechanical pieces are expensive! – on the other, we want to avoid the motor breaking down completely, producing even more damage.

The problem we are going to deal with in this whitepaper belongs to the second and third groups: unsupervised anomaly detection. We have no examples of the catastrophic event in our historical data – luckily – however we still want to predict the breakdown early enough to prevent the catastrophe from striking.

There is a lot of data that lends itself to unsupervised anomaly detection use cases: turbines, rotors, chemical reactions, medical signals, spectroscopy, and so on. In this whitepaper we deal with rotor data.

The Data

The main problem in putting together a public workflow for anomaly detection is actually the lack of publicly available data, at least for unsupervised anomaly detection problems. After quite an extenuating search for publicly available data, we ended up using a data set from an anonymous donor, monitoring a rotor through a twenty-eight sensor matrix, focusing on eight parts (see the table below) of the mechanical component, on a time frame spanning January 1, 2007 through to April 20, 2009.

In total we have 28 time series from 28 sensors attached to 8 different parts of the mechanical rotor. The signals reach us after the application of the Fast Fourier Transform (FFT), spread across 28 files, in the form of (Fig. 1):

```
[date, time, FFT frequency, FFT amplitude]
```

Spectral frequency and amplitude are generated by the FFT.

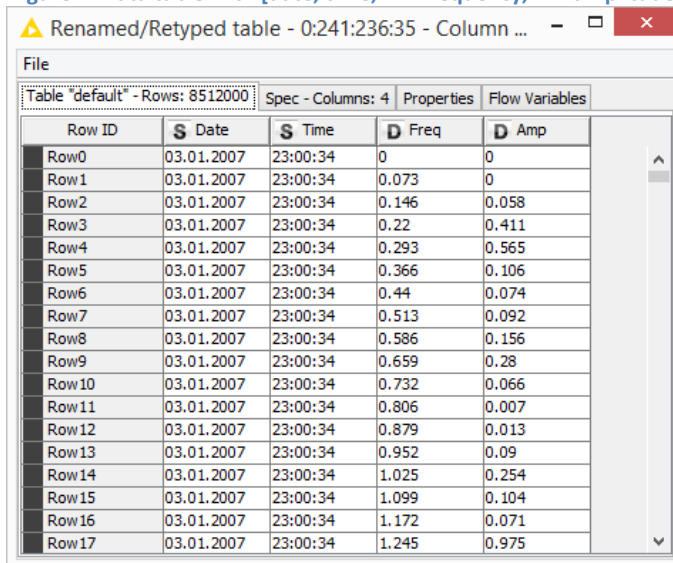
The eight parts of the rotor monitored through groups of sensors

A1	input shaft vertical
A2	second shaft horizontal upper bearing
A3	third shaft horizontal lower bearing
A4	internal gear 275 degrees
A5	internal gear 190,5 degree
A6	input shaft bearing 150
A7	input shaft bearing 151
M1	torque KnM

It is worth noticing that the signals are not synchronous in terms of date and time, meaning that they have been recorded at different times during the day and at different dates.

The whole data set shows only one breakdown episode on July 21, 2008. The breakdown is visible only from some sensors but particularly in some frequency bands. After the breakdown, the rotor was replaced, with much cleaner signals being recorded afterwards.

Figure 1. Data table with [date, time, FFT frequency, FFT amplitude]



Row ID	S Date	S Time	D Freq	D Amp
Row0	03.01.2007	23:00:34	0	0
Row1	03.01.2007	23:00:34	0.073	0
Row2	03.01.2007	23:00:34	0.146	0.058
Row3	03.01.2007	23:00:34	0.22	0.411
Row4	03.01.2007	23:00:34	0.293	0.565
Row5	03.01.2007	23:00:34	0.366	0.106
Row6	03.01.2007	23:00:34	0.44	0.074
Row7	03.01.2007	23:00:34	0.513	0.092
Row8	03.01.2007	23:00:34	0.586	0.156
Row9	03.01.2007	23:00:34	0.659	0.28
Row10	03.01.2007	23:00:34	0.732	0.066
Row11	03.01.2007	23:00:34	0.806	0.007
Row12	03.01.2007	23:00:34	0.879	0.013
Row13	03.01.2007	23:00:34	0.952	0.09
Row14	03.01.2007	23:00:34	1.025	0.254
Row15	03.01.2007	23:00:34	1.099	0.104
Row16	03.01.2007	23:00:34	1.172	0.071
Row17	03.01.2007	23:00:34	1.245	0.975

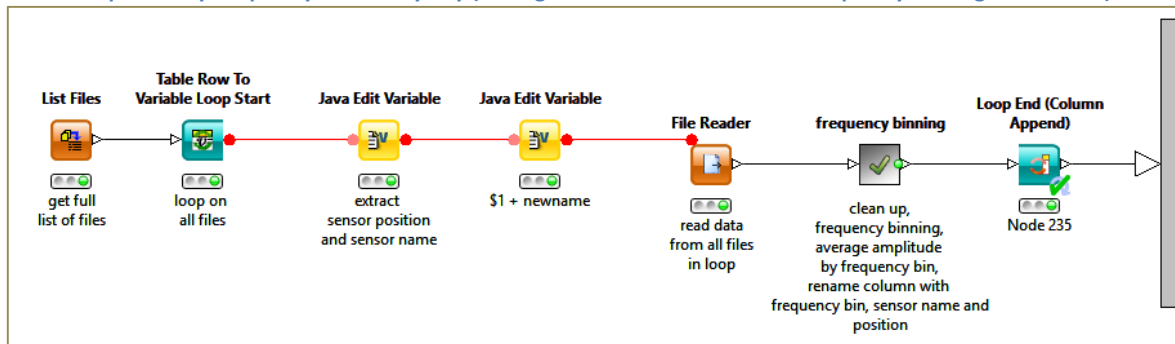
Data Pre-processing

Reading Data

The “Read all Data” metanode:

- loops over all 28 files,
- reads the content in the format described in the previous section,
- bins the frequencies into 100 Hz large bins, spanning the range from 0 to 1200 Hz,
- calculates the average spectral amplitude for each frequency bin and for each day,
- renames the resulting columns as <frequency band + current file name>,
- appends the newly created columns for the current file name to the set of already produced data columns.

Figure 2. Content of "Read All Data" metanode looping on all files, reading the data, and calculating the average spectral amplitude by frequency bin and by day (see Fig. 3 below for contents of "frequency binning" metanode)



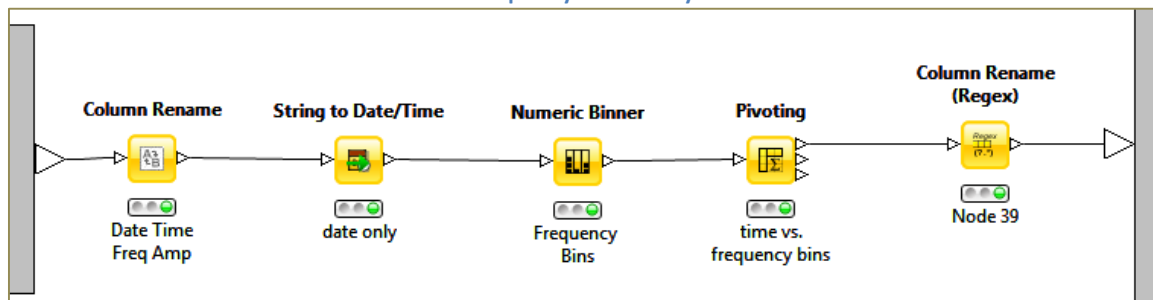
Frequency Binning

The part with the frequency binning, the calculation of the average spectral amplitude, and the renaming of the data columns is performed by the "frequency binning" metanode.

First, all datetime objects are transformed into the corresponding date by means of the String to Date/Time node; then 100Hz large frequency bins are created using the Numeric Binner node; the Pivoting node bins the frequency values into the corresponding frequency bins and the datetime values into the corresponding date bins, and calculates the average spectral amplitude value for each date and each frequency bin; finally, the Column Rename (Regex) node renames each column with the corresponding frequency band and the original file name.

The output data table contains 313 time series of average spectral amplitudes for each frequency bin (band) and for each day.

Figure 3. Metanode "frequency binning" inside "Read All Data" metanode calculating the average spectral amplitude by frequency bin and day



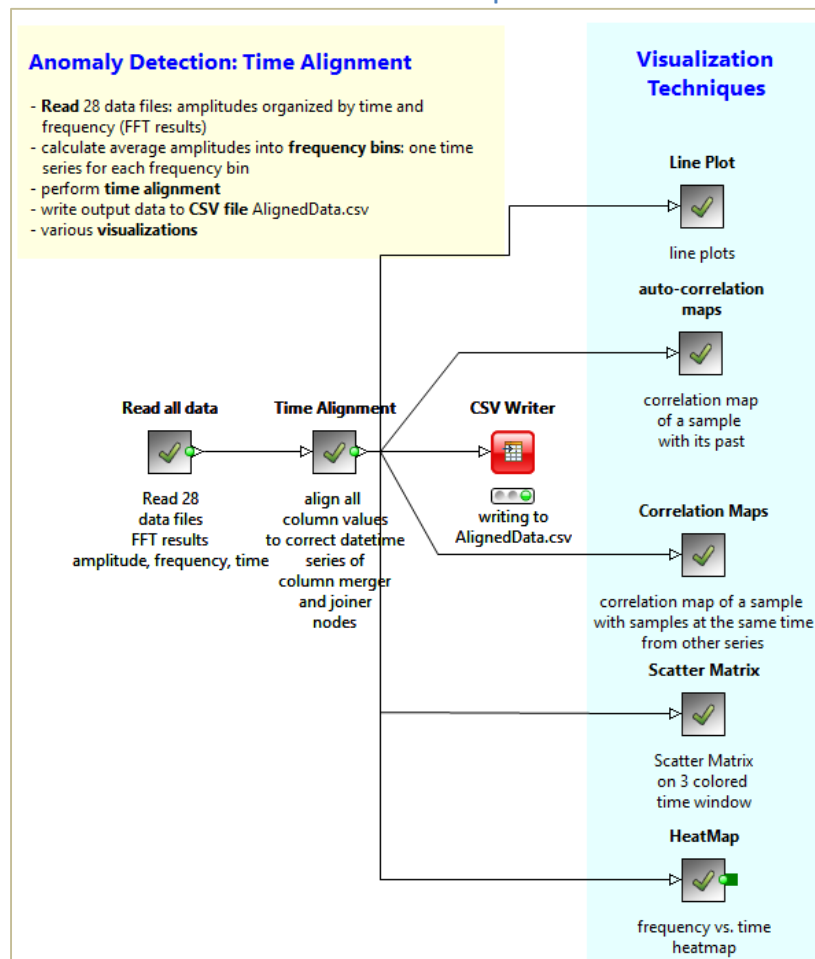
Time Alignment

The original files, however, do not contain measures all for the same dates. Some start earlier, some end earlier, and some are missing a few measures here and there. Basically, we now have a time alignment problem.

As we kept the date values for each imported file, we can now run an outer join, using a Joiner node, to join two sets of measures on their date values. The resulting data table, though, presents a number of missing values in the date column as well as in the measure columns (Fig. 4). While the missing values in the measure columns are meaningful (i.e. no signal recording for that day), there should be only one date column covering the whole time span without missing values. The Column Merger node merges two data columns, filling up the missing values in the primary column with the values in the secondary column.

<http://www.knime.org/files/AnomalyDetectionFullDataSet.zip>

Figure 6. Workflow “Time Alignment & Visualization” pre-processing, time aligning and visualizing the time series with different visualization techniques.



Time Series Line Plots

A line plot of a time series can be obtained with a Line Plot node. The interactive view of the Line Plot node allows data column to be selected as well as the color that should be displayed.

Not much pre-processing is necessary, besides overwriting the RowIDs with the date values. Indeed the Line Plot node shows the RowID values on the x-axis. Thus, if we want to see the dates on the x-axis we need to transform them into RowIDs.

In the two plots reported in Fig. 7 the rotor breakdown on July 21 is clearly visible in the [500-600Hz] band of the A1-SV3 signal and not really that much in the [0-100Hz] band of the same signal. In the second plot, representing the signal in the [500-600Hz] band, the context before and after the breakdown is easily recognizable: the “before” time window shows higher spectral amplitudes for the signal produced by the old rotor piece; the “after” time window shows a signal with lower spectral amplitudes produced by a very new rotor.

Figure 7. Line Plots of A1-SV3 Data, respectively in the [0-100Hz] and in the [500-600Hz] frequency bands



Scatter Matrix

The Scatter Matrix node generates a view of an $n \times n$ scatter matrix with n^2 scatter plots for n selected data columns.

In a 3x3 scatter matrix, 9 permutations of pairs of 3 data sets are displayed in 9 scatter plots. The three selected time series to be represented in a 3x3 scatter matrix (Fig. 8) are generated by sensor A1-SV3 on the [0-100Hz], [200-300Hz], and [500-600Hz] frequency band respectively.

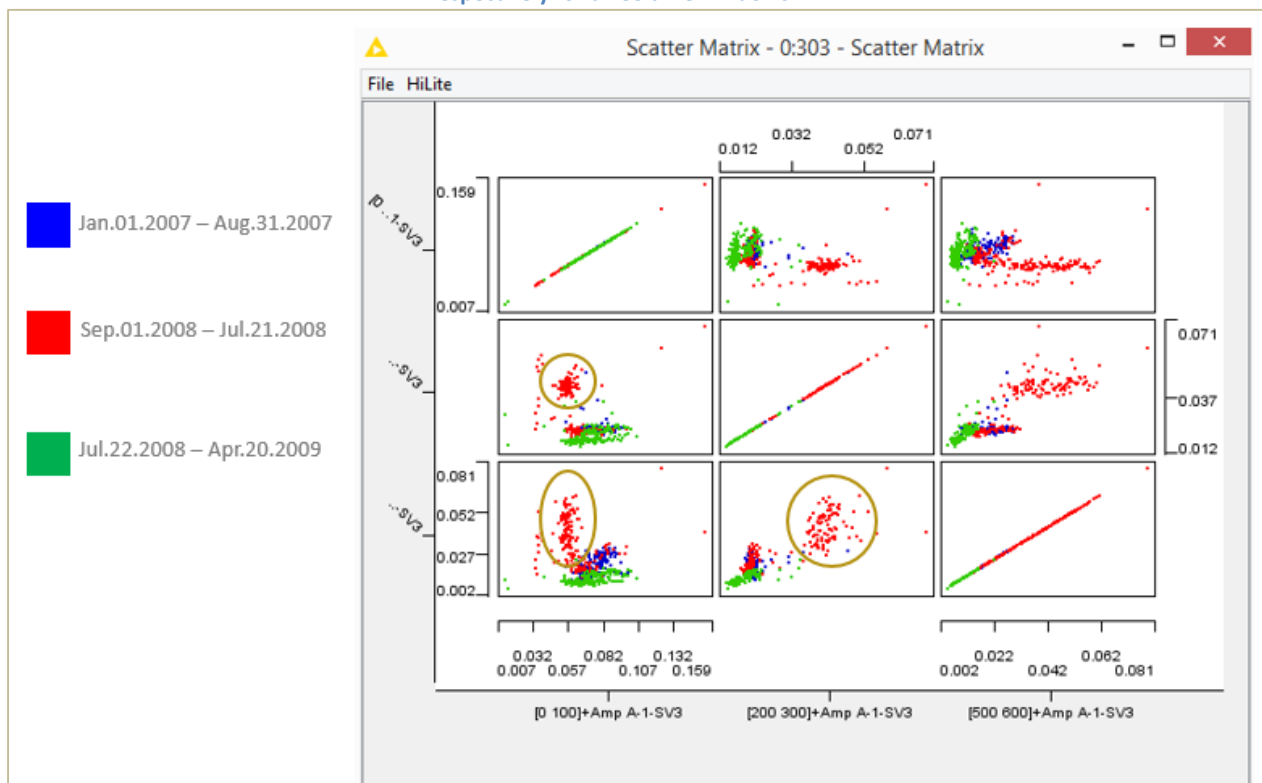
Notice that the scatter plots on the matrix diagonal represent a data set vs. itself and therefore show a diagonal line.

Notice also that the scatter plot generated by data set 1 vs. data set 2 is found again as data set 2 vs. data set 1 at the symmetrical location in the scatter matrix.

Here we also examined three different time windows:

- from January to August 2007 in blue (training window);
- from September 2007 to July 21, 2008 (breakdown date) in red (maintenance window);
- after July 22, 2008 in green (new rotor window).

Figure 8. Scatter Matrix of time series from A1-SV3 sensor in [0-100Hz], [200-300Hz], and [500-600Hz] frequency band respectively for three time windows



The interactive Scatter Matrix view allows you to choose the data columns to be displayed but not their color. Data point coloring is obtained with a Color Manager node. In the Color Manager node, a color map is created according to the values in a selected data column. If the values are nominal, a color is associated with each nominal value. If the values are numerical a heat map is defined over the data column numerical range. We associated different colors with different time windows: training (blue), maintenance (red), and new rotor (green) window.

The time preceding the breakdown in red is clearly seen wandering off from the majority of the data points in all scatter plots.

Correlation Maps

In a correlation map, the correlation is calculated between all values in a data column and all values in another data column, as the [Pearson's Product Moment Coefficient](#) if the two data columns are numeric, or as the [Pearson's Chi Square Test](#) if the two data columns are nominal. No correlation coefficient is defined for numerical vs. nominal data columns.

The configuration window of the Linear Correlation node just requires the data column set for the calculation of the correlation matrix. The correlation matrix is then produced at the output port and in the node view as a red-white-blue heatmap. Blue values show a high correlation, white values show correlation zero, and red values show an inverse correlation between two data columns.

Correlation measures for numerical columns are range dependent. Normalization is required before correlation calculation for data columns to fall in the same numerical range. Supposedly our data files reached us already normalized.

We calculated the correlation matrix over the training time window (till August 2007) and over the maintenance time window (from September 2007 till July 2008). The resulting correlation matrices are

different: signals seem to be more correlated across frequency bands over the maintenance window, right before the rotor breakdown, rather than over the training time window earlier on, when the rotor is still working normally (Fig. 9 and 10). Could this be something that we can leverage to predict the imminent breakdown?

Figure 9. Correlation matrix across frequency bands before August 31, 2007 for time series originating from sensor A1-SV3

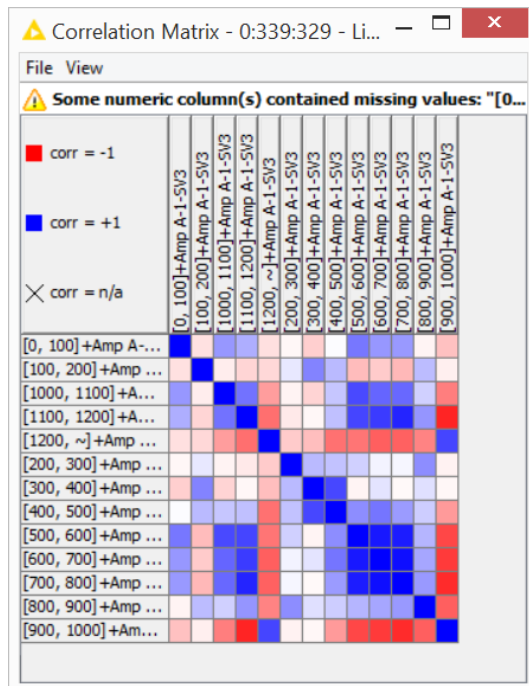
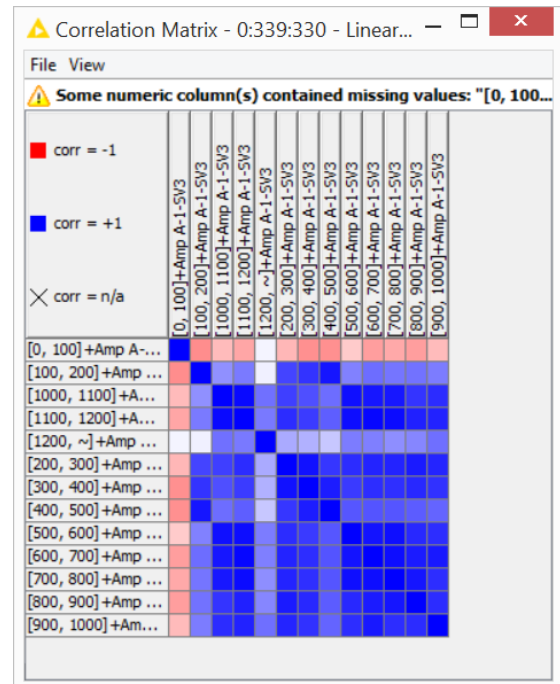


Figure 10. Correlation matrix across frequency bands after August 31, 2007 and before July 21, 2008 for time series originating from sensor A1-SV3



Auto-Correlation Maps

Correlation matrices can be calculated across frequency bands, as in the previous section, as well as over time. Using the Lag Column node we can put the current sample value and its past N values in the same data row. Therefore, feeding the Linear Correlation node with the current sample values and their past values produces an auto-correlation matrix. In an auto-correlation matrix, the correlation values are calculated across the samples and their past values.

Correlation measures for numerical columns are range dependent. However, here normalization is not necessary since the numerical range of a column of current samples is the same as the numerical range of a column of past samples of the same signal.

The resulting auto-correlation matrices in Fig. 11 and 12 show that the almost non-existing correlation of the time series values with their past in the [300-400Hz] frequency band increases noticeably moving from the training time window into the maintenance time window. Again, is this something we can leverage to predict the imminent rotor breakdown?

Figure 11. Auto-correlation matrix before August 31, 2007 for time series originating from sensor A1-SV3

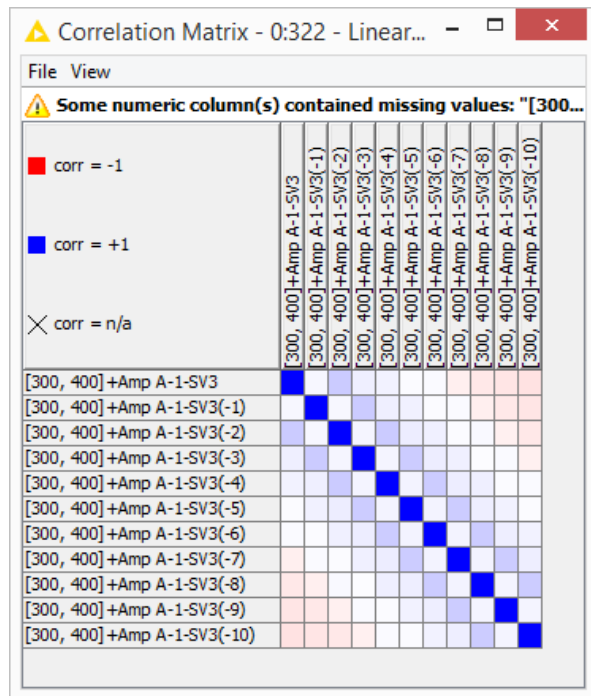
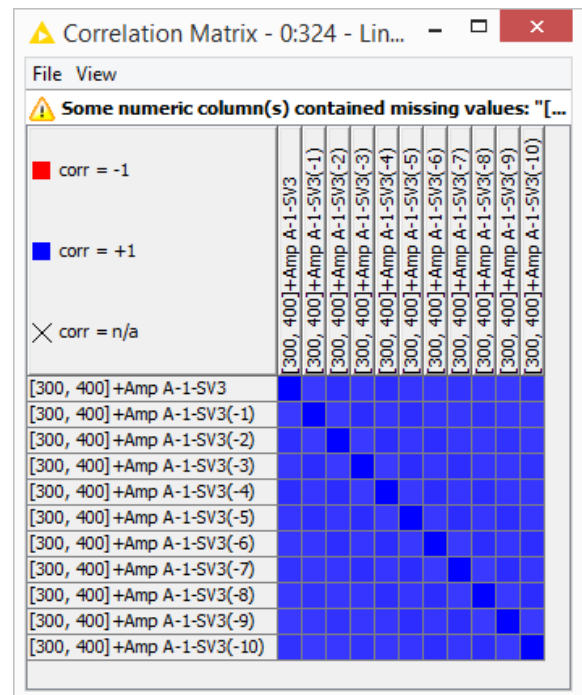


Figure 12. Auto-correlation matrix after August 31, 2007 and before July 21, 2008 for time series originating from sensor A1-SV3



Heatmap

The last visualization technique we applied to this dataset, in order to get a better idea of the breakdown and the signal evolution over time and across frequencies, is the heatmap.

A heatmap visualizes a number of data columns, effectively producing a matrix with an invisible RowID sequence on the y-axis, the column names on the x-axis, and the column numerical values as cell values. The cell values are represented graphically with a color progression from a selected color representing the lowest value to a selected color representing the highest value.

The Heatmap (JFreeChart) node, in the configuration window, selects some data columns from the input data table to convert into a heatmap image at the output port and to visualize in the View command of the node context menu. The heatmap range colors are also defined in the node configuration window.

If we want to hide the column range information, we need to normalize the data columns to fall into the same range prior feeding them into the Heatmap (JFreeChart) node. For this project we decided to keep the column range as they are, to show such information in the heat map as well.

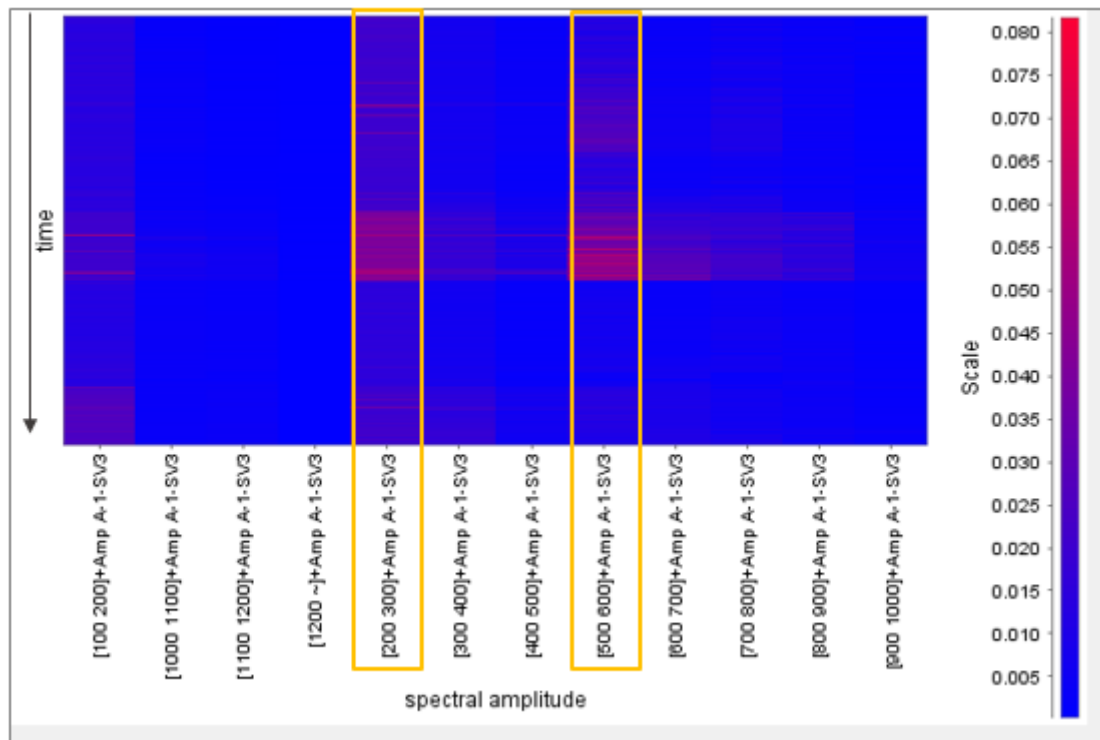
In our case (Fig. 13), we have the time on the y-axis, the different frequency bands for the selected signal on the x-axis, and the spectral amplitudes as cell values. The heatmap ranges from deep blue for the lowest numerical values to bright red for the highest numerical values.

Fig. 13 shows the heatmap of one of the sensor signals for the whole time available. There we can see the breakdown in bright red clearly over two frequency bands: [200-300Hz] and [500-600Hz].

Conclusions

In this whitepaper, we started analyzing time series data from a network of sensors monitoring a working rotor, which features a breakdown episode on July 21. It would be interesting to be able to predict the breakdown episode from no other examples than the available data history of normal functioning. This is what anomaly detection is all about. Before we proceed with complex data analytics, however, we need to clean and quantify the data suitably and visually explore their evolution over time and across frequency bands.

Figure 13. Heatmap across frequency bands and time of the signal originating from the A1-SV3 sensor



This whitepaper prepares the FFT processed time series coming from the rotor sensors for further analytics, by averaging spectral amplitudes by date and frequency bin and performing some time alignment. After that, it explores the time series evolution visually using five different visualization techniques: line plots, scatter matrix, auto-correlation maps, correlation maps, and heat maps. From the resulting views, you can see the breakdown episode clearly in some of the frequency bands.

In addition, time alignment and frequency binning are not uncommon procedures in the analysis of sensor data. The procedures described here could easily be reused for other similar applications, for example Internet of Things applications.

The final workflow, with pre-processing, time alignment and visualization nodes, named “Time Alignment & Visualization” is available on the EXAMPLES server under 050_Applications/050017_AnomalyDetection/Pre-processing.

050_Applications/050017_AnomalyDetection/data contains a reduced data set to run the workflow.

The original full data set is downloadable from

<http://www.knime.org/files/AnomalyDetectionFullDataSet.zip>