

Visualization Methods for Periodic Time Series Data

Deniz Neufeld¹

¹*Cognitive Systems Group, Otto-Friedrich University Bamberg, An der Weberei 5, 96047 Bamberg, Germany*

Abstract

This work presents visualization methods for the analysis of periodic or seasonal time series data sets in the univariate and multivariate case. Humans often find large data sets difficult to parse when they are plotted over a large time period. This poses a problem not only when labelling data for Machine Learning, but also when trying to explain an algorithm's prediction based on an input: In the case of periodic time series, in order to judge the results, a human would have to visually compare several data points at the same position in a period over many pattern repetitions. For this reason, we propose to split time series along period borders and remove time shifts and trend differences while enabling filtering by on-click highlighting of line segments to make outliers visible at first glance. We demonstrate our method using examples from three different domains. This method is optimized towards periodic data sets and is robust with respect to changes in trend and minor irregularities in periodicity. The method can trivially be implemented in most visualization frameworks, and is adaptable towards the specific use case and domain of new problem statements.

Keywords

sequential data, seasonal time series, periodic time series, interactive visual data analysis, anomaly detection, visualization

1. Introduction

Given a new problem statement, visual exploration and the labelling of given data are necessary before implementing and training an Artificial Intelligence (AI) Algorithm. Afterwards at prediction time, human users should be able to interpret a model's predictions efficiently and precisely, which can also be aided by providing accessible visualization. This work focuses on periodic time series, which are sequential data sets of repeating, similar patterns. Many of today's time series data sets fall into this category, for example IT infrastructure access times, medical data, traffic counts and climate data. Possible data science problems include anomaly detection, state classification, or change point detection. One challenge of this data type are the large amounts of data, which stem from high sampling rates and/or long recording times. Figure 1a shows a common plot type of a periodic data for univariate anomaly detection. In order to evaluate the data points that were classified as anomalous by an AI, a human has to evaluate the amplitude of each point relative to the points at the same time offset in other signal repetitions. Still, such data sets are often visualized only as a sequence over the complete recording time range. This process of assessing the correctness of an AI algorithm is time intensive and error prone.


LWDA'21: Lernen, Wissen, Daten, Analysen September 01–03, 2021, Munich, Germany

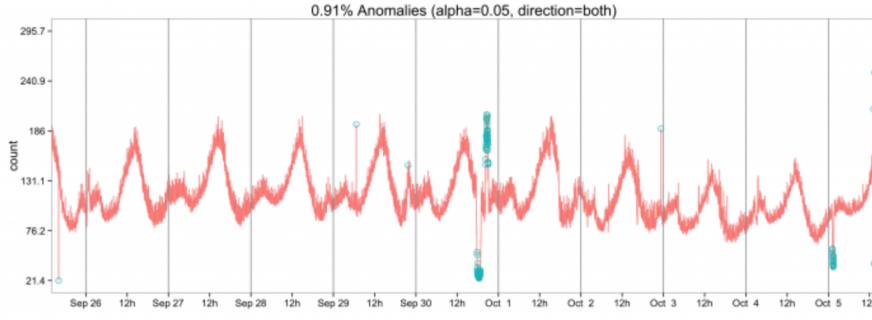
✉ deniz.neufeld@uni-bamberg.de (D. Neufeld)

ORCID [0000-0001-6533-7649](https://orcid.org/0000-0001-6533-7649) (D. Neufeld)

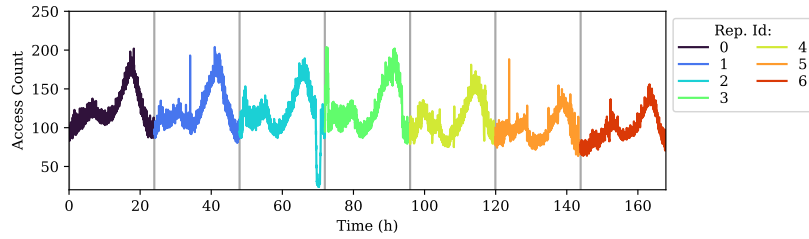


© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

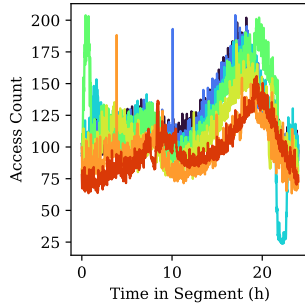
 CEUR Workshop Proceedings (CEUR-WS.org)



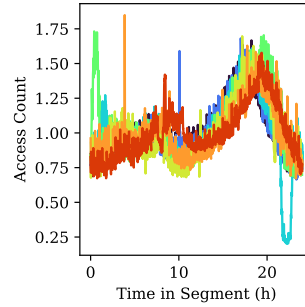
(a) Visualization of anomalies from [1] as a red line plot over the range of nine days. The blue dots show the anomalies computed by an anomaly detection algorithm.



(b) Server access count over time. Vertical lines are plotted between period repetitions (of 24 hours).



(c) Per-day data plotted above each other.



(d) Data de-trended and correlated over the time axis.

Figure 1: Plots of data that was published in Kejariwal’s article [1] about Twitter’s SESD algorithm for the anomaly detection in server access counts. Figure 1a is from the publication, Figures 1b through 1d are based on our method.

Instead, we show how periodic data can be plotted per period by splitting a data set across period lines (Figure 1b to Figure 1c) and mitigating small changes in trend and periodicity to improve visualization quality, shown in Figure 1d. This method can be implemented with current plotting frameworks and used as part of specialized tools by end users and data scientists and end users for data labelling, viewing, and for the explanation of AI algorithm results. It can even make some AI applications unnecessary when the plots show the relevant results on first glance. We demonstrate our method on three different data sets. Of course, like other data

science methods, this work only presents a method kit, and it cannot replace expert domain knowledge to judge how to use it.

First, this paper shows related work examples on anomaly detection for periodic data, visualization and interactivity. Then different steps of the presented technique are explained. Finally, the core concepts are demonstrated based on three examples: an ECG signal, server access count at Twitter and a data set for condition monitoring of a hydraulic system.

2. Related Work

Periodic time series are a common format in data science and AI. We demonstrate our visualization approach on three different data sets. First is a ECG data set from the Scipy Package [2], a problem type which was evaluated for anomaly detection amongst others by Chakraborty et al.[3]. They extracted the normal signal from multiple signal periods and used it as a benchmark to compare new measurements. The difference is then classified based on a pre-set threshold. Second is a data set of server access counts over time at Twitter, published by Kejariwal and later also used by Hochenbaum et al. ([1], [4]). For point- and global anomaly detection of this data set, they first extract the trend and the seasonal component of the data and then classify the residual with an Extreme Studentized Deviate (ESD) test. The third use case is a data set for condition monitoring of a hydraulic system with multiple deteriorating subcomponents by Helwig et al. (used in [5], [6] and [7]) in the domain of supervised state classification. While this data set is aligned in amplitude and time direction, it is a complex, multivariate data set. This shows that our method can be of interest for engineering use cases as well.

Interestingly, Wu and Keogh [8] have reviewed the progress in machine learning for time series anomaly detection in the last years and discovered that many of the newly published pieces work use complex models (Deep Neural Networks) for "trivial" problems that "can be solved with a single line of standard library MATLAB code" ([8], p. 2). They have shown that most of these published pieces of work have been applied to trivial problems, and that many of the data sets they show were periodic time series. Thorough visual inspection of data can help in designing more efficient machine learning models and in avoiding over-engineering.

Visualization is an important step in data science and can reveal patterns in a data set (David and Tukey [9]). In their survey, Fang et al. [10] discussed several visualization methods for multivariate time series. They show one example similar to ours called the calendar view, a plot for seasonal data. Here, each season of the data is plotted in a multi-line plot over one common time axis with the length of the season, for example the data of a year is divided into months and displayed over the range of one month. The plot is e.g. combined with a calendar widget to filter the days to show. Fang et al. cite examples by Wijk and Selow [11], Macas and Machado [12] and Buono and Balducci [13] for this. Matkovic et al. [14] developed a GUI (graphical user interface) application visually similar to ours with the focus on simulation result visualization. For this, they plot generated data based on simulation parameters chosen by the user. The resulting data was also plotted as a multi-line plot over one time axis, improving the visibility of differences in the results. Our method also suggests to present time series segments

in a multi-line plot over a common time axis, but in contrast to other work we also show the benefits of de-trending and time offset removal.

As another example, Lin et al. [15] have presented a graphical user interface (GUI) tool called VizTree, for the visualization of clusters of patterns in large time series databases based on augmenting suffix trees. This work is more focused on visualizing common sub-sequences across several time series. Our approach is focussed on the special case of periodic time series, instead.

A different data science domain, Functional Data Analysis, routinely uses plots visually similar to ours. Hyndman and Shang [16] developed rainbow plots for visualizing continuous, smooth functions over an identical data range. This results in multiple line plots where each line is coloured with a different shade, chosen from a rainbow colour map. We also propose the use of a rainbow-like colour map. The difference is that our method works with a broader range of application domains, and therefore depending on the use case, also entails different pre-processing steps (de-trending, fitting along time axis) and interactivity functionality.

Interactive visualization is an important tool, especially for end users. Yi et al. [17], for example, published an review of interactivity categories for data visualization. They identified the taxonomic units of Select, Explore, Reconfigure, Encode, Abstract/Elaborate, Filter and Connect ([17], p. 3). Adding to this, known Gestalt principles (first described by Wertheimer [18]), describe how humans visually interact with data representations and can be leveraged to improve complex visualizations. We will describe in the following Chapter, how our approach follows interactivity and Gestalt principles.

3. Method

Having shown several plotting approaches and different use cases where periodic data is relevant, our visualization method is presented next. To support the visual evaluation by humans, the visualization is designed to compress the plot of a long, periodic time series such that related points across all period repetitions are shown closer together. This is to better utilize the Gestalt principles of law of proximity (elements that are close together, appear as a unit) and law of similarity (elements that are similar, appear as a unit). Our goal is to enable the visual estimation of the data distribution per period. At the same time, anomalies can be identified faster, since they are data points that have above average distance from the rest. Our approach consists of the following steps:

1. Split the complete time series along period borders
2. (Optional) Remove the time offset using correlation (used in Section 4.1 and 4.2)
3. (Optional) Remove the trend component of the repetitions (used in Section 4.2)
4. Plot line segments over one common time axis
5. (Optional) Enable interactive highlighting (used in Section 4.3)
6. (Optional) Create additional plots, like an anomaly metric in the multivariate case (used in Section 4.3)

The computation of the periodicity of the signal is use case dependant. For example, in an ECG case, the period of the signal can be determined by counting the peaks in the signal above

a certain threshold (see Figure 2d). Splitting the signal across period lines is the first and main aspect of the presented work, and if there are no other offsets or noise present, this is the only preprocessing step before plotting. Depending on the problem domain, the time and trend offsets between the signal repetitions can be removed in order to improve visualization. The computation of the trend component used in our examples is done based on the seasonal trend decomposition (STD) of the time series, shown in Listing 1. Depending on the data set, one can choose a multiplicative or additive removal of the trend. Then, the time offset (in x-axis direction) of the segments can be removed using correlation. We used the version implemented in the Scipy library [2] `signal.correlate` as described in Listing 2. For two sequences with the length N and M , `correlate` has the time complexity of $O(NM)$. For our examples it was chosen over a Dynamic Time Warping algorithm, which in its basic implementation also runs in $O(NM)$, but in practice runs much slower. The data segments are then ready for visualization.

Our examples were implemented and plotted in Python 3.7 using Matplotlib.pyplot [19]. The colour map used is the standard "turbo" colour map, which is a continuous over a wide array of colours. In the multivariate case, each channel of the time series is plotted in a separate sub-plot. Since lines can cover each other in the plot depending on painting order, it is useful to provide the possibility to highlight and hide certain lines in the plots on mouse-click. This was implemented using the `mplcursors` library [20]. For the multivariate case, highlighting all channels of a repetition after mouse click and displaying the repetition index on click is also a useful feature, which will be shown later in this paper. Just by implementing our proposed visualization algorithm with Matplotlib (a python plotting framework), we are able to cover many interactive features described by Yi et al. [17]: Select (by highlighting of lines on mouse click), Encode (by using additional sub-plots on the data), Filter (by enabling the hiding of lines on click) and Connect (by enabling that all relevant lines of a repetition are highlighted on mouse click).

```

1 import numpy as np
2 from statsmodels.tsa._stl import STL
3
4 def remove_signal_trend(series, repetition_count):
5     stl = STL(series, period=(len(series)) // repetition_count)
6     res = stl.fit()
7     return series - res.trend

```

Listing 1: Example method for removing the trend component in time series.

```

1 def fit_signal_time(wanted_signal, other_signal):
2     sampling_time = wanted_signal.dt
3     shift_index = scipy.correlate(other_signal.y, wanted_signal.y).argmax()
4     if shift_index != 0: # shift_index is 0 if there is no correlation
5         shift_time = wanted_signal.time[-1] - other_signal.time[0] - \
6             shift_index * sampling_time
7         other_signal[signal_name].time = other_signal[signal_name].time + shift_time

```

Listing 2: Code for adjusting the time axis of one signal (`other_signal`) to fit the signal to another (`wanted_signal`) signal.

4. Examples

The presented methods are shown on three different data sets: the ECG data set from the Scipy package, the data set published by Kejariwal [1] in the presentation of Twitter’s Seasonal ESD algorithm, and the hydraulic systems condition monitoring data set by Helwig et al. [5].

4.1. ECG

For demonstration purposes, a section of the data set was chosen that has a constant periodicity and no trend (Figure 2a). Just splitting the time series into its number of periods yields Figure 2b, which is a very noisy plot due to slight offsets in periodicity in the signal. Applying Listing 2 to the split data results in the data set shown in Figure 2c. Plotting this side by side with an standard ECG graph (Source: <https://en.wikipedia.org/wiki/Electrocardiography>) in Figure 2d, it becomes obvious even to laymen that the signal frequently displays an anomaly in the ST part of the signal, and the R peak is often broader and less pointy than normal. Without our plot, a human would have to compare all different parts of the signal along all repetitions of the period. This type of plot can also be used for de-noising. If all the lines are plotted with higher transparency (an $\alpha < 1.0$), noisy parts of the data are displayed less prominently.

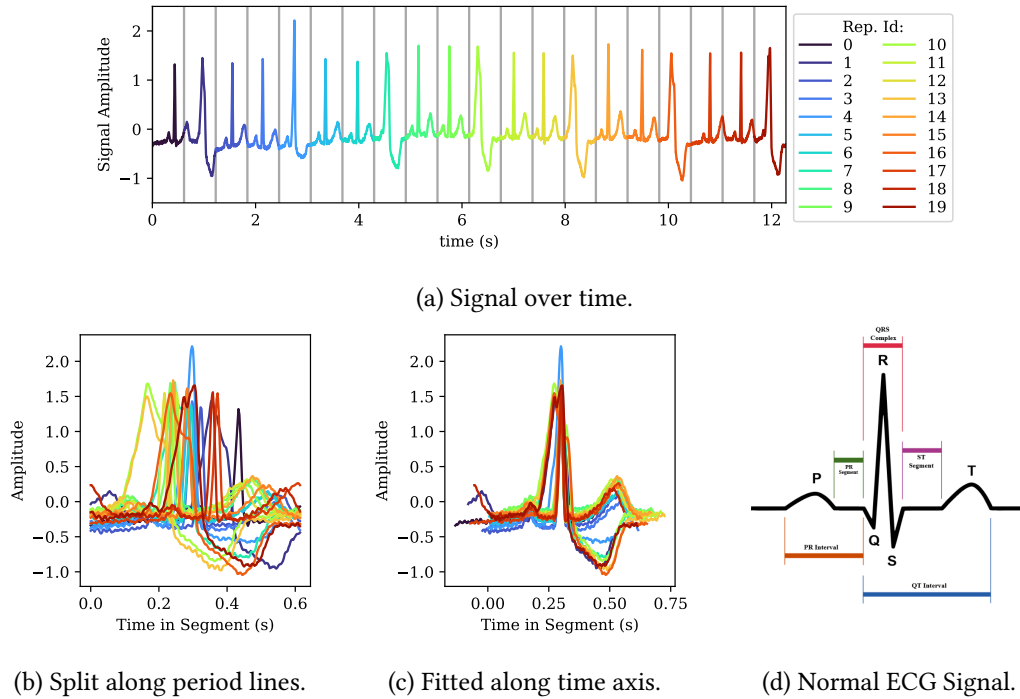


Figure 2: Visualization of part of Scipy’s ECG data set.

4.2. Twitter

In the use case of server access count anomaly detection, Kejariwal shows an anomaly detection algorithm which is based on Seasonal Trend Decomposition [1]. Figure 1a shows a visualization from the blog post. As can be seen, the data set exhibits a slight downwards trend over the span of the data set. Based on this plot it is difficult to judge the performance of the algorithm closely: focussing on the plot from the evening September 29th, an anomaly was detected, but it is difficult to interpret why this was an anomaly, when the peak in the mornings of October 3rd and 4th were not. In terms of visualization, de-trending can be applied because the use case is focused on point anomalies. Therefore, the trend removal (Listing 1) and amplitude shift removal (Listing 2) were used. The resulting plot in Figure 1d visualizes the processed data set. Anomalous peaks are now visible upon first glance. The day of the anomaly can be displayed, again, by selection and highlighting via mouse-click.

4.3. Hydraulic Anomaly Classification Data Set

The data set for condition monitoring in hydraulic systems by Helwig et al. [5] is a multivariate data set. It is designed for supervised anomaly classification of a hydraulic system that has several subcomponents which can fail. The data set is recorded on a hardware test bench, which means the measurement cycles of the systems are repeated as uniformly as possible. This is a typical use case in hardware engineering, especially in the domain of reliability analysis (see e.g. Birolini [21]), where load is applied to several structurally identical hardware systems over a long amount of time in order to estimate the systems behaviour over its product life cycle. The time series run for 6 seconds, with a maximum sampling rate of 1 kHz. Amongst the recorded signals of the hydraulic system are several pressures (P1-P6) and temperatures (TS1-TS4). For our plot in Figure 3, we chose 10 normal cycles (index 0 through 9) and two abnormal cycles (Id. 10 and 11), where the cooling subcomponent was damaged. The removal of trend and time offset is counter-productive, because the trend over time is an important sign of anomaly. All channels were plotted in separate sub-plots. Additionally, we computed an anomaly metric algorithm by taking the median of all measurements of the normal and calculating the Mean Absolute Error (MAE) of each measurement repetition towards it. This is visualized in the "MAE across channels" sub-plot: it displays the distance of each repetition per channel normalized to $[0, 1]$ in a parallel coordinate plot. This way, it is visible on first glance that 10 and 11 have unusual high deviation from the other cycles, and it is visible which channels show the most deviation. This can be used by a human to decide, which channel's sub-plot should be looked at closely. Due to its multivariate nature and complexity of data, Figure 3 shows the usage of an interactive highlighting functionality. After clicking on one line in this example the line 11 in the MAE plot, all corresponding lines are highlighted in black in all sub-plots. Since the plot was implemented in Matplotlib, it is also possible to interactively zoom in on each plot if required.

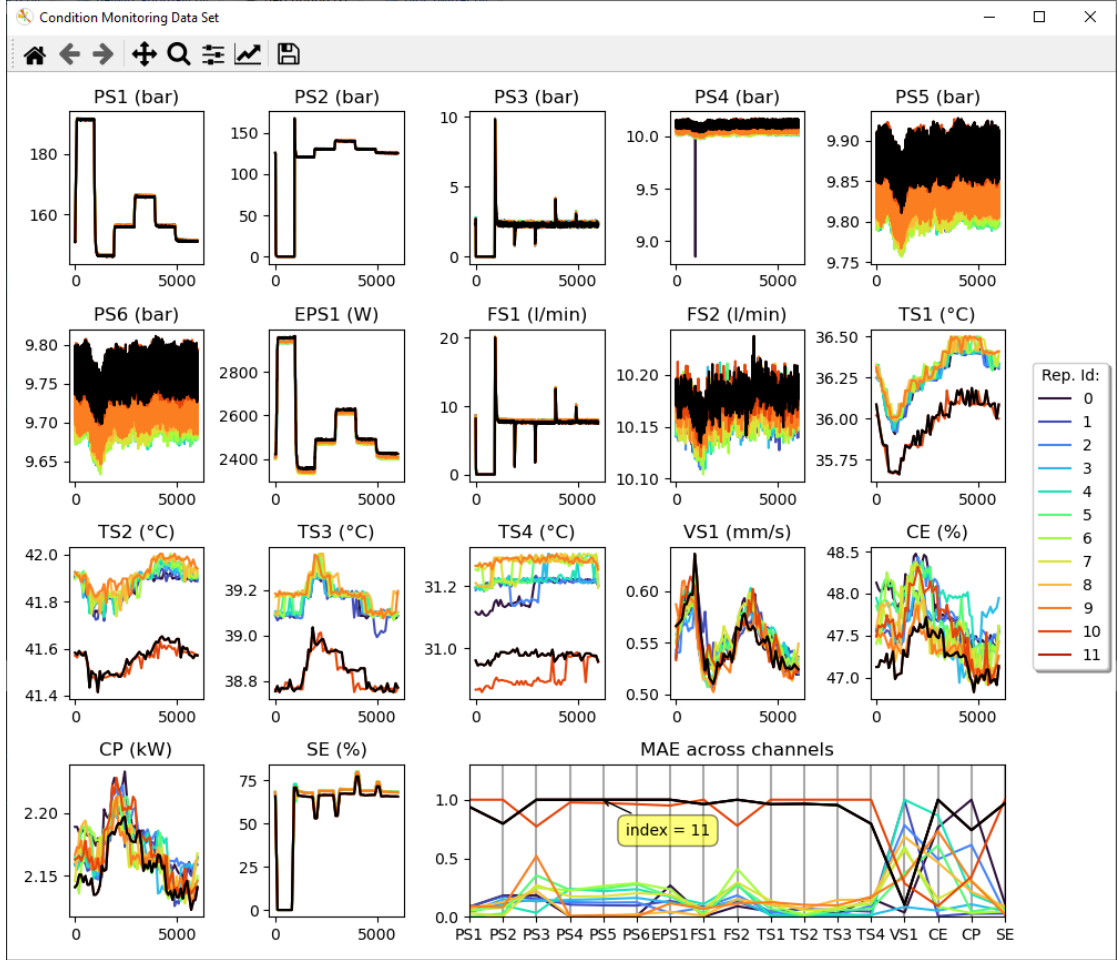


Figure 3: Screenshot of the visualization of the multivariate data set from Helwig et al. [5]. Repetitions 10 and 11 were chosen from a run with a broken subcomponent. Clicking on one line in the plot highlights all other lines belonging to the same repetition. The plot titled "MAE across channels" shows a per-channel distance metric, where each line belongs to one repetition.

5. Discussion

The shown method is very flexible and can be adapted to more domains. Still, as usual with machine learning, the type of preprocessing to choose depends on the problem domain. For the example of the hydraulic system, it is wrong to use the trend removal because trend changes indicate when the parts start to fail. Furthermore, in order to actually use the tool to label data for training, the implementation of a logging mechanism or a GUI for keeping track of the labelled data points would be helpful.

It would furthermore be of interest to conduct usability studies on how much time is actually saved using the method, and how much the accuracy of labelling improves compared to not using this approach.

Lastly, the method was presented with only three examples and can be tried for more difficult data sets to verify the usability for more domains. We estimate that the method as presented here can work well even with harsh measurements errors, for example if an ECG sensor slides off the patients skin. In this case hiding the "broken" measurement repetition using a mouse click would prevent unreadable plots, as we described before.

6. Conclusion

We have demonstrated the time and trend aligned plotting of periodic time series. It is a simple and easy to understand method of visualization, which can be adapted to be used for complex, multivariate data set. While the examples in this work were implemented using simple plotting frameworks, implementation in other areas like web based application is also possible. Since it is a general approach to data visualization, it can be adapted to more domains.

References

- [1] A. Kejariwal, Introducing practical and robust anomaly detection in a time series, 2015. URL: https://web.archive.org/web/20210506134624/https://blog.twitter.com/engineering/en_us/a/2015/introducing-practical-and-robust-anomaly-detection-in-a-time-series.html.
- [2] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, Polat, VanderPlas, Jake, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, SciPy 1.0 Contributors, SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python, *Nature Methods* 17 (2020) 261–272. doi:10.1038/s41592-019-0686-2.
- [3] G. Chakraborty, T. Kamiyama, H. Takahashi, T. Kinoshita, An Efficient Anomaly Detection in Quasi-Periodic Time Series Data—A Case Study with ECG, in: I. Rojas, H. Pomares, O. Valenzuela (Eds.), *Time Series Analysis and Forecasting, Contributions to Statistics*, Springer International Publishing, Cham, 2018, pp. 147–157. doi:10.1007/978-3-319-96944-2.
- [4] J. Hochenbaum, O. S. Vallis, A. Kejariwal, Automatic Anomaly Detection in the Cloud Via Statistical Learning, 2017. URL: <http://arxiv.org/pdf/1704.07706v1>.
- [5] N. Helwig, E. Pignanelli, A. Schütze, Condition monitoring of a complex hydraulic system using multivariate statistics, in: 2015 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings, IEEE, 11.05.2015 - 14.05.2015, pp. 210–215. URL: <https://archive.ics.uci.edu/ml/datasets/Condition+monitoring+of+hydraulic+systems>. doi:10.1109/I2MTC.2015.7151267.
- [6] N. Helwig, E. Pignanelli, A. Schütze (Eds.), D8.1 - Detecting and Compensating Sensor Faults in a Hydraulic Condition Monitoring System: AMA Service GmbH, P.O. Box 2352, 31506 Wunstorf, Germany, 2015. doi:10.5162/sensor2015/D8.1.
- [7] T. Schneider, N. Helwig, A. Schütze, Automatic feature extraction and selection for

classification of cyclical time series data, *tm - Technisches Messen* 84 (2017). doi:10.1515/teme-2016-0072.

- [8] R. Wu, E. J. Keogh, Current Time Series Anomaly Detection Benchmarks are Flawed and are Creating the Illusion of Progress, 2009. URL: <http://arxiv.org/pdf/2009.13807v3>.
- [9] F. N. David, J. W. Tukey, Exploratory Data Analysis, *Biometrics* 33 (1977) 768. doi:10.2307/2529486.
- [10] Y. Fang, H. Xu, J. Jiang, A Survey of Time Series Data Visualization Research, *IOP Conference Series: Materials Science and Engineering* 782 (2020) 022013. doi:10.1088/1757-899X/782/2/022013.
- [11] J. J. van Wijk, E. R. van Selow, Cluster and calendar based visualization of time series data, in: *Proceedings 1999 IEEE Symposium on Information Visualization (InfoVis'99)*, IEEE Comput. Soc, 24-29 Oct. 1999, pp. 4–9. doi:10.1109/INFVIS.1999.801851.
- [12] C. Macas, P. Machado, Radial Calendar of Consumption, in: *2018 22nd International Conference Information Visualisation (IV)*, IEEE, 10.07.2018 - 13.07.2018, pp. 96–102. doi:10.1109/iv.2018.00027.
- [13] P. Buono, F. Balducci, MonitorApp: a web tool to analyze and visualize pollution data detected by an electronic nose, *Multimedia Tools and Applications* 78 (2019) 33023–33040. doi:10.1007/s11042-019-7676-3.
- [14] K. Matkovic, D. Gracanin, M. Jelovic, H. Hauser, Interactive visual analysis of large simulation ensembles, in: *2015 Winter Simulation Conference (WSC)*, IEEE, 06.12.2015 - 09.12.2015, pp. 517–528. doi:10.1109/WSC.2015.7408192.
- [15] J. Lin, E. Keogh, S. Lonardi, J. Lankford, D. Nystrom, VizTreeA Tool for Visually Mining and Monitoring Massive Time Series Databases, in: *Proceedings 2004 VLDB Conference*, Elsevier, 2004, pp. 1269–1272. doi:10.1016/B978-012088469-8/50124-8.
- [16] R. J. Hyndman, H. L. Shang, Rainbow Plots, Bagplots, and Boxplots for Functional Data, *Journal of Computational and Graphical Statistics* 19 (2010) 29–45. doi:10.1198/jcgs.2009.08158.
- [17] J. S. Yi, Y. A. Kang, J. Stasko, J. Jacko, Toward a deeper understanding of the role of interaction in information visualization, *IEEE transactions on visualization and computer graphics* 13 (2007) 1224–1231. doi:10.1109/TVCG.2007.70515.
- [18] M. Wertheimer, Untersuchungen zur Lehre von der Gestalt. II, *Psychologische Forschung* 4 (1923) 301–350. doi:10.1007/BF00410640.
- [19] J. D. Hunter, Matplotlib: A 2D graphics environment, *Computing in Science & Engineering* 9 (2007) 90–95. doi:10.1109/MCSE.2007.55.
- [20] A. Lee, mplcursors, 2021. URL: <https://mplcursors.readthedocs.io/en/stable/>.
- [21] A. Birolini, Reliability Engineering, Springer, Dordrecht, 2007. URL: <http://gbv.ebib.com/patron/FullRecord.aspx?p=372652>.