Visual Informatics 6 (2022) 51–66

Contents lists available at ScienceDirect

Visual Informatics

journal homepage: www.elsevier.com/locate/visinf

Research article

TBSSvis: Visual analytics for Temporal Blind Source Separation

Nikolaus Piccolotto [a,*], Markus Bögl [a], Theresia Gschwandtner [b], Christoph Muehlmann [c], Klaus Nordhausen [d], Peter Filzmoser [c], Silvia Miksch [a]

[a] TU Wien, Institute of Visual Computing & Human-Centered Technology, Favoritenstrasse 9–11, A-1040 Vienna, Austria
[b] Erste Group Bank AG, Am Belvedere 1, A-1100 Vienna, Austria
[c] TU Wien, Institute of Statistics and Mathematical Methods in Economics, Wiedner Hauptstrasse 8–10, A-1040 Vienna, Austria
[d] University of Jyväskylä, Department of Mathematics and Statistics, FI, 40014 Jyväskylä, Finland

ARTICLE INFO

Article history:
Received 11 October 2021
Received in revised form 23 June 2022
Accepted 8 October 2022
Available online 25 October 2022

Keywords:
Blind source separation
Ensemble visualization
Visual analytics
Parameter space exploration

ABSTRACT

Temporal Blind Source Separation (TBSS) is used to obtain the true underlying processes from noisy temporal multivariate data, such as electrocardiograms. TBSS has similarities to Principal Component Analysis (PCA) as it separates the input data into univariate components and is applicable to suitable datasets from various domains, such as medicine, finance, or civil engineering. Despite TBSS's broad applicability, the involved tasks are not well supported in current tools, which offer only text-based interactions and single static images. Analysts are limited in analyzing and comparing obtained results, which consist of diverse data such as matrices and sets of time series. Additionally, parameter settings have a big impact on separation performance, but as a consequence of improper tooling, analysts currently do not consider the whole parameter space. We propose to solve these problems by applying visual analytics (VA) principles. Our primary contribution is a design study for TBSS, which so far has not been explored by the visualization community. We developed a task abstraction and visualization design in a user-centered design process. Task-specific assembling of well-established visualization techniques and algorithms to gain insights in the TBSS processes is our secondary contribution. We present TBSSvis, an interactive web-based VA prototype, which we evaluated extensively in two interviews with five TBSS experts. Feedback and observations from these interviews show that TBSSvis supports the actual workflow and combination of interactive visualizations that facilitate the tasks involved in analyzing TBSS results.

1. Introduction

Multivariate measurements of a phenomenon are common in many domains. Medical doctors place electrodes on a patient's body to analyze processes such as brain activity, eye movements, or heart rhythm. Civil engineers measure vibrations on different parts of a structure, such as a bridge, to detect possible faults. Financial managers invest money in stocks, which are in a way sensors of economic processes, to gain wealth. Common to all these examples is the time-oriented data and the assumption that data from different sensors are in some way correlated and/or influenced by noise. However, analysts are usually only interested in the "true" underlying processes.

To obtain these processes, analysts turn to Blind Source Separation (BSS). BSS comprises established methods for signal separation that were applied, among others, in the mentioned domains of medicine (Comon and Jutten, 2010; de Lathauwer et al., 2000; Van Thanh et al., 2017), civil engineering (Amezquita-Sanchez and Adeli, 2016) and finance (Oja et al., 2000). Temporal Blind Source Separation (TBSS) refers to a subset of BSS methods that specifically account for temporal correlation. TBSS is similar to Principal Component Analysis (PCA) in the sense that (i) TBSS methods work on any multivariate dataset with quantitative variables, (ii) they work on measured data only (hence "blind") and (iii) separate it into a linear combination of uncorrelated components, like PCA. Unlike PCA, TBSS accounts for temporal correlation and often requires complex tuning parameters. As both TBSS and PCA can be considered forms of dimension reduction, analysts use TBSS and PCA for similar reasons, like data analysis or modeling/prediction.

During these activities, it is at some point necessary to inspect components visually. Like with PCA, components are hidden until the separation algorithm is executed, but TBSS's complex parameter space severely complicates the issue: It is known that parameter settings greatly influence the result, but not in which way a change in parameters translates to change in components. Experts regard automated analysis by extensive sampling (Sedlmair et al., 2014) not a feasible option and there is little guidance

* Corresponding author.
E-mail address: nikolaus.piccolotto@tuwien.ac.at (N. Piccolotto).

https://doi.org/10.1016/j.visinf.2022.10.002

from the literature, which parameters to pick. Because a ground truth is rarely available, TBSS analysis is inherently open-ended and exploratory as there are no known insights to confirm. The workflow of TBSS analysts can broadly be described as (i) pick a parameter setting, (ii) see if obtained components are useful or interesting and if not, go to (i).

Some challenges make TBSS difficult to use in practice. Despite the important role of visualization in their workflow, the current tool used by the analysts does not support them well in this regard. Analysts need to manually program static visualizations, which requires time they could otherwise spend on data analysis. Another challenge is the amount of components. Each parametrization on a p-variate dataset yields a set of p components that need inspection and comparison to previous sets. Analysts are, for example, interested in commonly found components, but very quickly confronted with hundreds of components to consider. This is a common task in ensemble visualization (Wang et al., 2019), but made more difficult by components appearing in sets instead of one by one. Also, when comparing multiple results, analysts will eventually find competing options for their final choice. As there is usually no ground truth available to compare the result to, analysts need detailed ways to compare individual results to make an informed decision.

Visual analytics (VA (Thomas and Cook, 2005)) as defined by Keim et al. (2008) "combines automated analysis techniques with interactive visualizations for an effective understanding, reasoning and decision making on the basis of very large and complex data sets." Considering the strong focus of BSS analysis on visual inspection on multiple levels of detail, in combination with mentioned challenges, we propose applying VA principles to overcome these. We designed TBSSvis according to Munzner's Nested Model (Munzner, 2009) for the TBSS method "generalized Second Order Blind Identification" (gSOBI) (Miettinen et al., 2020). We chose gSOBI because it is recent and well suited to real-world datasets due to its flexibility (see Section 3). The source code of TBSSvis is available at https://github.com/npiccolotto/tbss-vis.

Our primary research contribution is a design study (Sedlmair, 2016) for TBSS, which improves the visualization community's knowledge about an area that it did not explore so far. Specifically we provide:

- A task abstraction for TBSS which we obtained through a user-centered visualization design process with TBSS experts (Section 5).
- A VA design for gSOBI, a TBSS method, that supports the abstracted tasks by combining visualizations, interactions, and guidance methods (Section 6).
- Confirmation of the effectiveness of our design in two interviews with five TBSS experts (Section 8).

As part of this design study we put well-established visualization techniques together to support the identified tasks. They include a multivariate autocorrelation function plot and the application of a slope graph to sets of time series. These, together with a set-aware clustering scheme (Section 6.3), are our secondary contribution.

2. Related work

In the following, we elaborate on different approaches to visualize and compare time series, ensembles, and models.

2.1. Time series visualization

Temporal data is ubiquitous in many domains such as finance, health, or biology, and has been visualized for centuries since the first line graph was introduced by Playfair (Tufte, 2001). Various other visual encodings have been proposed afterwards, such as tile maps, sparklines, or horizon graphs (Aigner et al., 2011). They use different visual variables (Mackinlay, 1986) such as position, color, or slope, and therefore exhibit different perceptual properties, which makes them suitable for different analysis tasks. E.g., Gogolou et al. (2019) investigated the relation between different time series visualization idioms and perceived similarity. They recommend to use horizon graphs when local variations in temporal position or speed is important, while others (line graph, color band) are better suited for notions of similarity where amplitude is less important. As this is the case with TBSS, where analysts look for patterns independent of amplitude, we show time series as the familiar line graph.

When multiple time series are at hand, their respective visualizations need to be composed. Two popular approaches to do so are superposition and juxtaposition. Superimposed encodings trade decreased usage of display space for legibility, as they do not scale well after a couple of variables due to occlusion. An example besides the well known superimposed line graph is the braided graph (Javed et al., 2010), which superimposes multiple area-based marks. Because of the varying data dimensionality in TBSS, superimposition is generally not a promising strategy. Multiple time series can also be composed with juxtaposition, as is the case in LiveRAC (McLachlan et al., 2008). Various system measures (columns) are displayed per machine (rows) in a space-filling table design, using semantic zooming to change the level of detail between color bars, sparklines, and labeled line graphs. When not using all available space, one could use small multiples (Tufte, 2001) in different arrangements. For instance, Stitz et al. (2016) arrange small multiples of stocks by price and price change in a user-selected time frame. Liu et al. (2018), on the other hand, lay them out with a modified Multidimensional Scaling (MDS) algorithm such that similar items are near each other. These approaches proved to be very useful for individual time series, but cannot be applied as such to TBSS, where sets of time series are involved. In the experience of our collaborators only some time series in TBSS will carry a signal and be interesting for closer inspection. Our approach employs various strategies to account for both facts, e.g., grouping time series by similarity and sorting representatives by a user-selected degree-of-interest function, or juxtaposing time series sets in a table-like design.

To keep features of long time series visible, designers often turn to focus-and-context techniques, such as lenses (Tominski et al., 2017). In the simplest case, a lens mainly enlarges an area of interest, such as in SignalLens (Kincaid, 2010). But more complex interactions are possible, such as in ChronoLenses (Zhao et al., 2011), where users can combine and stack multiple lenses. As we designed TBSSvis for analysts who are accustomed to text-based interfaces, we took care to avoid complex interactions. Time series may be enlarged up to a certain level of detail in discrete steps and filtered to a contiguous subset of the currently visible time interval with simple direct manipulation interactions. We describe them in Section 6.1. We did not employ data reduction methods, neither in a data-driven (Shurkhovetskyy et al., 2018) nor visualization-driven way, e.g., by line simplification (Rosen et al., 2020), as one risks that important features are removed.

2.2. Ensemble visualization

The goal in ensemble visualization is to make sense of a set of similar complex data items, such as trajectories, often

produced by a simulation with perturbed parameter settings. Component sets obtained from different TBSS parametrizations constitute such an ensemble, where each ensemble member is a set of time series. Ensemble visualization has its origin in meteorology (Potter et al., 2009), but since expanded to more domains (Wang et al., 2019). Analytic tasks for ensemble data (Wang et al., 2019) indicate popular strategies, such as comparing members or grouping them by similarity, to support the stated goal. Existing works (Hao et al., 2016; Ferstl et al., 2017) often use popular clustering techniques (with domain-specific distance functions) to support the latter task. This is not straightforward in TBSS as one has to take care to not mix members of different sets into the same cluster. We discuss our approach, a custom clustering algorithm that respects this constraint, in Section 6.3.

Time is a common part of ensemble data, but not a requirement (Matkovic et al., 2009; Piringer et al., 2012; Matković et al., 2018; Xu et al., 2019). One possible case is when ensemble members are univariate time series, such as for Köthur et al. (2015), who encoded the correlation between members in a heatmap to support comparison of two ensembles. More commonly, other data types have an associated time dimension such as multivariate data (Obermaier et al., 2016), particle data (Hao et al., 2016), network security data (Hao et al., 2015), or spatial data (Buchmüller et al., 2019). However, ensembles of sets of time series, as in our case, are not thoroughly explored so far and our paper presents a first step in that direction.

2.3. VA for model construction

VA supported the construction and validation of various kinds of models, such as linear regression (Mühlbacher and Piringer, 2013; Zhao et al., 2014), logistic regression (Dingen et al., 2019), dimension reduction (Anand et al., 2012), classification (Choo et al., 2010), or artificial neural networks (Zhang et al., 2019; Wexler et al., 2020). Most works in the literature focus on non-temporal data, Bögl et al. (2013) (univariate time series modeling) and Sun et al. (2020) (univariate time series forecasting) provided two exceptions. TBSSvis, supporting construction and comparison of TBSS model alternatives, extends the state of the art as TBSS works on multivariate time series. During the construction step, questions of analysts pertain to which variables should be included, how many parameters should the model have, and which subgroups should be modeled. The latter question is closely related to model validation, where analysts, e.g., verify that a model works for diverse data cases, or how multiple models agree/differ on outputs, such as predicted class labels. Established diagnostic plots or data exist for several of these procedures, e.g., residual plots in time series modeling (Bögl et al., 2013), or confusion matrices in classification (Wexler et al., 2020). In contrast, the quality of a TBSS model is solely defined by the presence of domain-specific interesting features in the output, thus TBSSvis focuses on comparing multiple alternatives in terms of similarity of their output and parameter settings. A complicating factor that we tackle is that TBSS outputs are *sets* of time series.

3. Temporal Blind Source Separation

The statistical analysis of multiple measurements taken at different times is a challenging task. Often, such multivariate time series are analyzed by transforming the data in certain simple ways to uncover latent processes which generated the data. Probably the most used method for such a task is the classical PCA, which uses linear transformations of the data that result in components which have highest variance and are uncorrelated. Uncorrelated components imply that the covariance between the found linear combinations is zero. The linear combinations are given by diagonalizing the covariance matrix. Furthermore, as the nature of the transformation is linear, interpretations of the results can be carried out by the simple and well studied loadings-scores scheme. However, PCA might not be the best choice when the data at hand shows dependencies in time, as the main source of information is in that case not covariance, but rather serial dependence. Serial dependence is characterized by autocovariance, i.e., covariance between measurements separated in time by a given lag. In analogy to PCA it would be desirable to find linear combinations of the multivariate time series data which are not only uncorrelated marginally (zero covariance between variables at each time step), but also uncorrelated in time (zero autocovariance between variables for any lag). TBSS is a field of multivariate statistics that studies methods delivering the former desired properties. Generally, BSS is a well established model-based framework. It assumes that the observed data are a linear mixture of latent components, which are considered usually easier to model and/or more meaningful for interpretation than multivariate models. The goal of BSS is to recover these components based on the observed data alone. BSS is formulated and used for many types of data, as outlined in recent reviews (Comon and Jutten, 2010; Nordhausen and Oja, 2018; Pan et al.; Nordhausen and Ruiz-Gazen, 2022). In the following, we outline the concept of TBSS.

The model of TBSS considered here is $\boldsymbol{x}_t = \boldsymbol{A}\boldsymbol{c}_t$, where $\boldsymbol{x}_t$ denotes the observed p-variate time series, $\boldsymbol{A}$ is the full-rank $p \times p$ mixing matrix and $\boldsymbol{c}_t = (c_{1,t}, \ldots, c_{p,t})^\top$ is the set of p latent components, which should be estimated. Thus the goal is to find a $p \times p$ unmixing matrix $\boldsymbol{W} = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_p)^\top$, such that $\boldsymbol{c}_t = \boldsymbol{W}\boldsymbol{x}_t$ up to sign and order of the components in $\boldsymbol{c}_t$. To facilitate the recovery, the assumption is made that the components in $\boldsymbol{c}_t$ have $\mathrm{Cov}(\boldsymbol{c}_t) = \boldsymbol{I}_p$ and are uncorrelated (or independent) with mutually distinct serial dependence. This means, for example, that all cross-moment matrices, such as autocovariance matrices, of $\boldsymbol{c}_t$ are diagonal matrices.

A very first approach for TBSS is the Second-Order Blind Identification (SOBI) algorithm (Belouchrani et al., 1997; Miettinen et al., 2014, 2016). It finds the linear combinations of the data which make autocovariance matrices for several lags as diagonal as possible. Hence, found components are uncorrelated marginally and uncorrelated in time. It is well known in the statistical analysis of time series data, that time series emerging from different scientific fields have different key characteristics. For example, financial time series are not well characterized by autocovariance matrices, but instead higher-order moments carry the most information. This is denoted as stochastic volatility and in the TBSS literature it is shown that SOBI fails for such time series (Matilainen et al., 2017). Higher-order moments relate often to skewness and kurtosis and, for example in our context, to the covariance of the squared data and are meant to detect more unusual observations (heavy tails). An intuitive notion of higher-order moments is that they translate to quickly changing effects, such as stock prices that increase/decrease by large margins within short time frames (high volatility). In such cases, higher-order moments describe volatility better than second-order moments. To overcome this issue, a new TBSS method, denoted as a variant of SOBI (vSOBI) (Matilainen et al., 2017), was introduced. Similar to SOBI, vSOBI finds the latent time series by diagonalizing matrices of lagged fourth moments. Uncovered latent components are uncorrelated marginally and additionally have zero fourth-order dependence.

Generally, time series might carry information both in the autocovariance and in the higher-order time dependence, thus a combination of SOBI and vSOBI might deliver the best results. Indeed, Miettinen et al. (2020) proposed such a method, referred to as generalized SOBI (gSOBI), which we focus on in this

manuscript. It diagonalizes several autocovariance matrices (SOBI part) and several matrices of lagged fourth moments (vSOBI part). This method has three rather involved tuning parameters. The first one $b \in [0, 1]$ weighs SOBI versus vSOBI, where SOBI ($b = 1$) and vSOBI ($b = 0$) are the extreme cases. The second ($\boldsymbol{k}_1$) and third ($\boldsymbol{k}_2$) tuning parameters provide the sets of lags used for the SOBI and vSOBI part, respectively. A lag is a time interval given by a number of time steps, the size of which is determined by the resolution of the underlying time series. For instance, a lag of 6 in an hourly observed thermometer refers to an interval of 6 h. Common default values for gSOBI are $b = 0.9$, $\boldsymbol{k}_1 = \{1, \ldots, 12\}$ and $\boldsymbol{k}_2 = \{1, 2, 3\}$ (Miettinen et al., 2020), but Miettinen et al. also show that the selection of lag sets and weight has a huge impact on the performance (Miettinen et al., 2020). Vague guidelines for these tuning parameters exist in the community, such as lag sets should not be too small or too large, and the lags should be chosen so that the corresponding (cross-)moment matrices for the *latent* components have diagonal values far apart. Thus, parameter selection in the context of SOBI is a highly complex problem with no practical solution yet (Tang et al., 2005; Taskinen et al., 2016). First steps for an informed trial & error routine can be determined from those guidelines and by looking at the data. As an example, if the time series at hand show substantial volatility, then the b parameter (weight of second- vs. fourth-order moments) would initially be chosen closer to 0. Otherwise, volatility observed in the dataset might not be visible in latent components. Lag sets generally can be chosen by observing how long any visible patterns, like volatility, last. If they are rather short, then short lags are more suitable than longer lags, and vice versa. The interdependence between parameter choice and the variation in the output depends a lot on the dataset at hand, so much so that general statements about it would be misleading. Instead we propose an advanced VA approach, which allows defining alternative parameter choices and comparing the respective outputs effectively, to discover such relations.

The R implementation of gSOBI used in the following is available in the package tsBSS (Nordhausen et al., 2021). We call one execution of gSOBI a *run*. As outlined before it yields a set of p univariate time series, which we call *components*. The outcomes of multiple runs with varying parameter settings form an ensemble, where each member corresponds to a single run. A member has the used parameters $\boldsymbol{k}_1$, $\boldsymbol{k}_2$ and b associated, as well as the output of gSOBI. The latter is either the component set $\boldsymbol{c}_t$ and the estimated unmixing matrix $\hat{\boldsymbol{W}}$, or nothing, in case the (cross-)moment matrices could not be diagonalized in a predefined number of iterations. We call a run *succeeding* or *failing*, depending on the outcome.

4. Datasets

In this section we introduce two datasets, one from the financial domain (Fig. 1(a)) and one from the medical domain (Fig. 1(b)), along with reasons why TBSS analysis of them can be desired. Analysis of both datasets shares similar tasks. For instance, analysts are interested in relevant parameter subspaces, common components and alternatives to them, as well as the stability of obtained results. We formalize typical tasks and questions involved in TBSS analysis in Section 5.

4.1. Financial data

Goods, currencies, and company stocks are traded every day at high frequencies. In simple terms, investors make money by buying something at a price X and selling it later at a price Y larger than X. To maximize $Y - X$ in a short time frame the idea here is to find a volatile collection of currencies or stocks (a portfolio), i.e., one that is subject to sudden and extreme changes in value. To do so, we look at the daily exchange rate of 23 currencies to Euro between the years 2000–2012 (23 variables, 3139 time steps). We preprocess the data to get logarithmic returns, a common measure in quantitative finance when the temporal behavior of return is of interest. The first three variables are shown in Fig. 1(a).

4.2. Medical data

An electrocardiogram (ECG) is a recording of the heart's electrical activity. To obtain it, electrodes are placed on the patient's skin. These electrodes detect small electrical changes which occur due to muscle de- and repolarization. ECGs are important for medical analysis as many cardiac abnormalities show deviations to the normal ECG pattern. Analysis of fetal ECGs may detect problems during fetal development, such as fetal distress. While invasive methods exist to measure the fetal ECG directly, a non-invasive method is often preferred as it does not harm neither mother nor fetus. The fetal ECG is visible in the mother's ECG, but it is weak and mixed with, e.g., respiratory noise or frequency interference (compare first three rows in Fig. 1(b)). Using TBSS on ten seconds of the ECG of a pregnant woman (8 dimensions, 2500 time steps), we try to extract the fetal ECG following previous work (de Lathauwer et al., 2000).

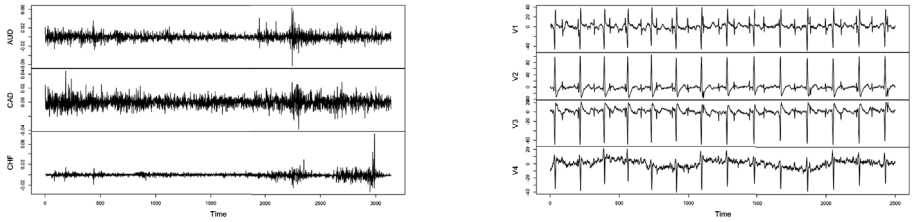
5. Task abstraction

In this section we present a task abstraction for TBSS. We structure it according to the data-users-tasks triangle by Miksch and Aigner (2014) and use the terminology by Brehmer and Munzner (2013) for tasks. We developed the abstraction together with the visualizations in an iterative design process following Munzner's Nested Model (Munzner, 2009) with three collaborators, who are co-authors of this paper and experts in BSS. In this user-centered design process model, we first conducted unstructured interviews in order to understand their problems and made ourselves familiar with literature they provided. After that, we discussed our assumptions and ideas regularly with them over a course of nine months. We discussed iteratively developed prototypes ranging from hand-drawn sketches, to static digital images, to an interactive application which is described in Section 6. During these sessions, we also questioned our current understanding of their tasks either implicitly through visualization designs or explicitly through discussions. In the end, we interviewed five TBSS experts, who did not collaborate with us on the design, to further validate our abstracted tasks (Section 8). The presented task abstraction is a reflection on this process.

We touched upon the involved data with TBSS in Section 3 and Section 4 already. These are a multivariate time series (input data), one real and two sets of integers (TBSS parameters b, $\boldsymbol{k}_1$ and $\boldsymbol{k}_2$) and a set of univariate time series (latent components). The temporal dimension is discrete and linear.

5.1. Users

Our users are data analysts or data scientists with formal education in statistics/math and basic knowledge of BSS. They may also be experts in a specific application domain, like medicine or finance. They work mostly with R (Team, 2020), a language and environment for statistical computation in which most BSS researchers publish their implementations. The preferred work environment is RStudio, a popular text-based development environment for R. Currently, they use built-in plotting functionality, and sometimes they use, for example, ggplot2 to build customized visualizations. The output of either option is a static



(a) First three currencies of the *exrates* daily currency exchange rate dataset.

(b) First four variables in the ECG dataset.

Fig. 1. Datasets in this paper.

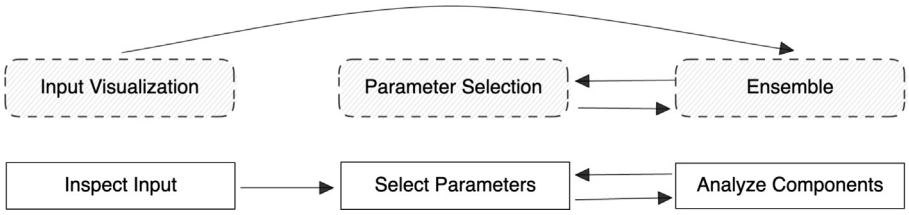


Fig. 2. The existing analysis workflow (bottom) and the corresponding screens in TBSSvis (top). The new workflow automatically obtains initial results and analysts can start exploring immediately.

visualization, of which RStudio by default displays only one at a time. Because of this, our users are accustomed to well known static statistics visualizations such as histograms, line graphs, box plots, etc.

5.2. Tasks

During this user-centered design process we identified the following tasks, which we describe using the abstraction terminology by Brehmer and Munzner (2013).

The high-level workflow can be separated into three phases, which are depicted in Fig. 2: Analysts first inspect the raw input data, continue to find parameter settings, and then analyze obtained components. Given the exploratory nature of their analysis process, analysts switch between the latter two phases until they feel they exhausted the parameter space or obtained a useful result.

Generally, analysts want to *discover* observations or *derive* a modified dataset with reduced dimensionality. There are two main *targets* of analysis. Components, which are mostly analyzed as sets, are one target. Still, analysts want to *discover* and *explore* interesting components, whatever interesting means in the data domain. Parameters are the other analysis target. Analysts look for a "stable" result, i.e., one that can be obtained with rather diverse parameter settings. The assumption is that its components are then more likely to represent real processes. To this end, they need to *compare* components and parameters of different runs. As an additional obstacle, when lacking intuition and/or domain knowledge, analysts struggle to select (*lookup*, *locate*) parameters and need guidance support (Ceneda et al., 2017) so they can *browse* and *explore* parameter settings in an informed way. All these observations lead us to some *low-level query actions*:

I1: Identify used parameters. Analysts want to see values of existing parametrizations. In case of lag sets they inspect the distribution of chosen lags and if one lag set contains more lags than the other.

I2: Identify unmixing matrix. Analysts turn to the unmixing matrix to interpret components and to understand how they were formed. They look for large absolute values per component.

I3: Identify cross-moment diagonality. Analysts want to inspect runs on a technical level that is currently inconvenient to obtain and difficult to quantify. If the TBSS model holds, then all cross-moment matrices are exactly diagonalized by the unmixing matrix estimate. For real data, this is, however, rarely the case and thus analysts are interested in the impact of the parameters on the diagonality of the different cross-moment matrices.

I4: Identify components. In a single component, analysts look for interesting features like outliers or uncommon changes in shape. They thereby also check the absence of features (noisiness). Analysts are further interested in the stability of a component, i.e., in how many ensemble members the component is present.

C1: Compare success. First of all, prior to any comparisons, analysts must know what can be compared. If a run did not succeed, only parameters could be compared as no components were found.

C2: Compare parameters. To carry out sensitivity analysis, analysts need to compare parameters between runs. They mainly look at differences in lag distribution and amount.

C3: Compare unmixing matrices. Before inspecting factors of individual components, analysts investigate the similarity of unmixing matrixes by means of a custom metric (MD-Index (Ilmonen et al., 2010)).

C4: Compare component sets. This task mostly relates to membership, which, however, is difficult to assert with complex objects, such as time series, where one usually speaks of similarity instead of equality. Analysts compare components only between sets and want to know which component exists in multiple sets, and if so at which ranks, and if not which is the most similar component, plus in which time frames components disagree.

C5: Compare possible parameters. When choosing a new parametrization, analysts need guidance through the parameter space and the ability to compare possible parameters in some meaningful way to find promising settings.

6. Visualization design & justification

In this section, we present the visualization design we obtained based on the task abstraction (Section 5) and implemented in a web-based prototype for gSOBI. A design goal was to make

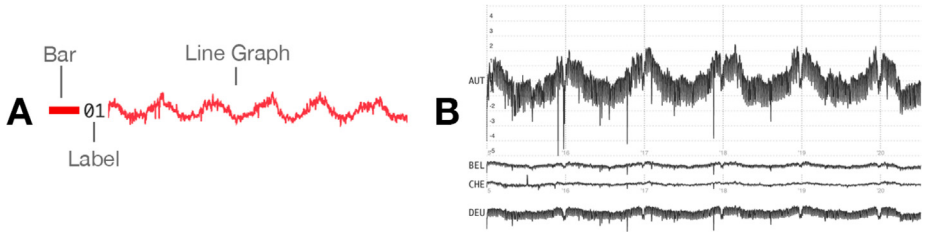


Fig. 3. Display of and interaction with time series. (A) A time series is displayed with a line graph, an optional label, and an optional bar to its left. The bar encodes a DOI function of a time series. (B) Typical vertical arrangement of multiple time series in TBSSvis. The user enlarged the first and last time series to different sizes and hovered over the third, thus its X axis labels are shown.

TBSSvis generic enough to allow its use in many application domains, because, like PCA, TBSS is a domain-independent method. After a cursory literature search of TBSS applications and considering the available amount of pixels on common screen sizes, we designed TBSSvis for inputs with the length of up to 5000 time steps, and up to 50 dimensions in mind. These limits do not accommodate extreme cases we found, like EEG data (128 dimensions, 1.2 million time steps (Tang et al., 2005)), but are suitable to financial (40 dimensions, 140 time steps (Oja et al., 2000)) or civil engineering (3 dimensions, 9000 time steps (Liu et al., 2019)) use cases we found. Our design guidelines denote soft limits of an interactive TBSS application: The quality of visualizations will gradually decline with data of higher complexity and the execution time of the used TBSS algorithm will increase.

While we implemented visualizations for all abstracted tasks, for brevity we will focus on an illustrative subset of those. Specifically, we will discuss visualizations for tasks that pertain to

- identifying and comparing components (or sets thereof),
- identifying and comparing used parameters, and
- comparing possible parameter settings.

TBSSvis consists of three screens, which are depicted with their connection to analysis phases in Fig. 2. The *Input Visualization* screen shows the raw input data, a feature requested by our collaborators. The *Ensemble* screen allows exploration of parameter settings and components. Finally, the *Parameter Selection* screen is used to select new parameter settings. We will focus on the latter two. How presented visualizations work together is illustrated in the usage scenarios (Section 7).

6.1. Time series visualization and interactions

Time series are plotted vertically aligned to facilitate comparison and ordered by variable name (for input variables) or by an interestingness function (for latent components, see below). The display of and interaction with all time series in TBSSvis is handled by the same logic as shown in Fig. 3. Due to the length and amount of time series, we employ semantic zooming and at first save display space by drastically shrinking their Y axis and omitting any labels by default. This can be changed with interaction: On hover, we display axis labels for the hovered time series. The Y axis of an individual time series can be enlarged in discrete steps by another interaction (pressing hotkey on keyboard and clicking). If an analyst is interested in a contiguous subset of the time series, it is possible to zoom in with brushing, which will affect all time series in the application. Both the semantic and temporal zoom can be reset with interactions recommended by Schwab et al. (2019).

As described in Section 3, the order of components is not defined. In practice, this means that analysts use measures which are sign-independent to compare components, such as absolute Pearson correlation, and impose an order by sorting components according to a function. We will call this a *degree-of-interestingness function* (DOI), and require it to be any function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ that maps a time series of length n to a scalar. Because TBSS is a domain-independent method, many DOI functions could be useful (Fu, 2011) depending on what the domain's interesting features are. E.g., for detailed cardiac analysis, different widths and types of ECG wave patterns could be mined. Based on discussions with our collaborators we use the absolute third (skewness) and fourth moment (kurtosis) in TBSSvis. These are useful to find the most skewed components and those with the most outlying values, as the first two moments (mean and variance) of all components are identical. We added a measure for periodicity (Vlachos et al., 2005) after our user studies. The DOI function can be changed in the toolbar, and all views that show component-related data will update as the set members are sorted in descending order based on the new DOI function values.

6.2. Color

According to Mackinlay (1986), color is the most effective visual variable for nominal data after position, and, therefore, often used to encode different data classes. In multiple views, the same classes should be encoded with the same palette (Qu and Hullman, 2018). Because humans can only reasonably distinguish a few different colors, we cannot statically assign colors to all ensemble members. We, therefore, use a user-controlled dynamic assignment of colors of a qualitative palette to encode data related to user-selected members. The available colors are displayed in the toolbar and can be reordered with drag & drop. When hovering over an unselected member, the next free color (left to right) is (a) used to highlight its related data in all views and (b) associated to the member when selected. As one color is always needed for highlighting, the last free color cannot be used for selection. The color order determines the plotting order in all comparison views.

6.3. Tasks I4/C4: Identify/compare components (sets)

We precompute initial parameter settings automatically, to allow immediate exploration of the output space: Variations of gSOBI's R package's defaults (3 settings), a recommendation from the literature (Tang et al., 2005) (1 setting), and an additional user-defined number of random settings. This overcomes initial hesitation towards parameter setting choice and may give an estimate of what the relevant parameter subspace is.

Each successful run (Section 3) produces a set of time series. Already at the start of the analysis after precomputation, the amount of components to consider might be in the hundreds. Clustering is an established approach to counteract this, where data cases are grouped by similarity. This allows an analyst to focus on representative elements of the clusters. Many clustering techniques exist (Xu and Wunsch, 2005; Xu and Tian, 2015),

but using them with all components from all sets has a major drawback: The clustering scheme will put components from the same set into the same group, which our collaborators found undesirable. The grouping should respect the set structure in the data and group components only between sets, not within them. Additional requirements we gathered for the clustering scheme are that it should not depend on a distance metric (unlike, e.g., k-means) and produce an existing data case as cluster representative (again unlike, e.g., k-means). The former is related to the similarity measure for components suggested by our collaborators, the difference in absolute Pearson correlation $dist_{cor} = 1 - |cor(c_i, c_j)|$. Since we do not know if it supports the triangle inequality, we should not rely on it. The latter requirement stems from the design principle to show actual data over visual abstractions.

6.3.1. Clustering algorithm

We developed a custom clustering scheme to achieve our requirements. Starting from the realization that we basically want k-medoids, as it does not need a distance and produces existing representatives (medoids), we looked for a way to constrain the clustering process to obey the set structure. Constrained versions exist for k-means (Wagstaff et al., 2001), but we did not find one for k-medoids. However, it was possible to adapt it using a k-means-like formulation of k-medoids (Park and Jun, 2009). Constraints in our case are of the type *cannot-link*, i.e., they express which data cases must not be grouped into the same cluster. We add one cannot-link constraint per pair of elements that belong to the same set. For m sets, each containing p data cases, this amounts to $mp(p-1)/2$ constraints in total.

Algorithm 1 shows pseudocode of our custom clustering scheme. The initial medoids are obtained by an unconstrained k-medoids algorithm (Schubert and Rousseeuw, 2019) on line 2. Following (Park and Jun, 2009) we use the sum of distances from all data cases to their medoids as cost function (line 3) and compute it for the initial clustering. Then, while constraints are violated, i.e., there is a cannot-link constraint for any two data cases assigned to the same medoid, we update the clustering (line 5). If there are no violated constraints, there is nothing to do as the initial clustering is a valid solution. Otherwise, we first select the most central of data cases assigned to same medoid (i.e., with smallest sum of distance to other data cases in that cluster) as a new medoid (line 6). Data cases are then reassigned to the nearest medoid that does not already contain another data case for which there exists a cannot-link constraint (line 7). These are steps 2 and 3 in the k-means-like formulation for k-medoids (Park and Jun, 2009). We update the cost for the current clustering (lines 8–9) and repeat this loop (lines 5–12) until no constraints are violated anymore. Small necessary checks, e.g., whether or not there are still violations after the loop, were left out for brevity.

```
   Data: Dissimilarity Matrix D, constraints C, medoids M,
         assignments of data cases to medoids A, number of
         partitions k
1  Function constrainedPAM(D, C, k) is
2      M, A ← FastPAM(D, k)
3      cost ← getCost(A, D)
4      cost′ ← cost
5      while violatesConstraint(A, C) or
        cost − cost′ > ϵ do
6          M′ ← findNewMedoids(A, D, k)
7          A′ ←
            assignToNearestPossibleMedoid(M′, C, D)
8          cost ← cost′
9          cost′ ← getCost(A′, D)
10         A ← A′
11         M ← M′
12     end
13     return (M, A)
14 end
```

Algorithm 1: Pseudocode of constrained k-medoids, with which we obtain a clustering on sets of time series (Section 6.3.1).

6.3.2. Clustering quality and number of partitions

The constrained k-medoids clustering takes one user-provided parameter, which is the desired number of clusters. We use a scented widget (Willett et al., 2007) to allow setting this parameter in an informed way (Fig. 4A). The bar chart in the widget shows the average cluster separation as a clustering quality measure for a given number of clusters. Therefore, values with high bars suggest the number of meaningfully different components in all currently available sets.

6.3.3. Component overview

The cluster medoids are shown underneath the Clustering Quality visualization, vertically aligned in a list, sorted by the DOI rank of the medoid (Fig. 4B). To further support Task C4, we show a histogram to the left of the medoid. The histogram shows the rank distribution of the contained components in their respective sets. Additionally, we encode $dist_{cor}$ to the cluster medoid with opacity. This way, stable (stacked bars with high opacity) and unstable (scattered bars with low opacity) components have distinct histogram shapes.

Analysts can inspect components in a cluster by clicking the "eye" icon, after which the list item expands and lists contained components in the same fashion as cluster medoids. Clicking a bar in the histogram or a time series label selects the associated ensemble member.

6.3.4. Slope graph

Components of selected sets are visible in a separate view, again vertically aligned and sorted by DOI (Fig. 4C). Each selected set has a unique assigned color and all associated data is shown in this color. Multiple selections are juxtaposed horizontally in columns, which can be rearranged by the analyst. Analysts can inspect components visually as they are, or they can also display a slope graph between columns. Lines of the slope graph connect similar components, and thickness encodes similarity from high correlation (thick) to low. This way, it is easy to see stable (thick, single, mostly straight lines) and unstable components (no or thin, multiple, tilted lines), their rank changes and set similarity at a glance.

6.4. Tasks I1/C2: Identify/compare used parameters

Parameter space analysis (Sedlmair et al., 2014) is another important task for BSS experts, where they are mainly interested in *sensitivity analysis* and *partitioning*. We facilitate these tasks with tailored visualizations (Fig. 5).

6.4.1. Similarity views

Similarity of so far obtained component sets, as well as selected parameters, are shown in three separate dimensionally-reduced views. Marks that are close to each other suggest similar components and $\boldsymbol{k}_1/\boldsymbol{k}_2$ parameters. Multidimensional Scaling (MDS) is an appropriate dimension reduction technique for global cluster analysis according to recent publications (Nonato and Aupetit, 2019; Xia et al., 2022). We use non-metric MDS (Venables and Ripley, 2010) as we do not always have a distance metric. As

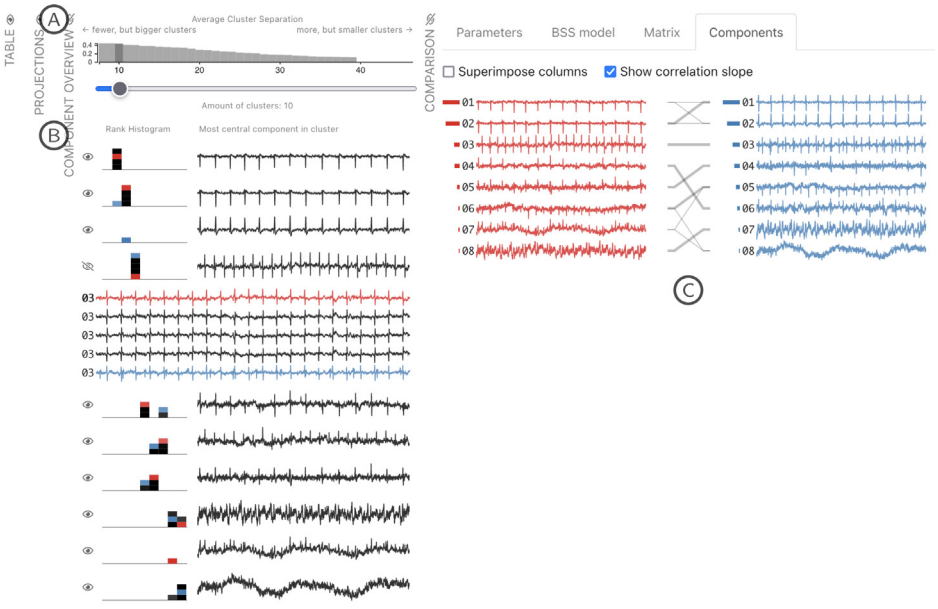


Fig. 4. *Ensemble* screen of TBSSvis (medical data) configured to facilitate comparison and inspection of components and sets thereof. Left part shows the Clustering Quality (A), which suggests an optimal clustering with 8–10 partitions. Medoids of the 10 clusters are listed underneath (B). The fourth list item readily shows the fetal heart signal and was expanded to show cluster members. Two component sets (red and blue) were selected for detailed comparison (right half). The Slope Graph (C) highlights similar components, their rank changes and set similarity overall.

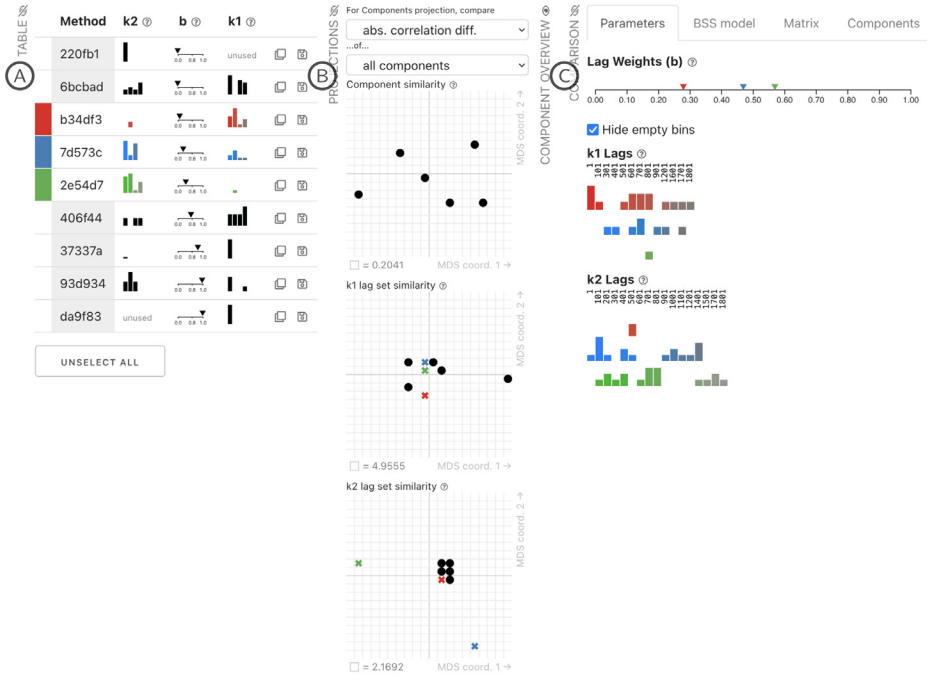


Fig. 5. *Ensemble* screen of TBSSvis (medical data) configured to facilitate comparison and inspection of parameters. Three failing runs were selected. Left column shows a tabular overview (A). Middle column shows DR projections of component and parameter similarities (B). Right column shows detailed comparison views to facilitate parameter comparison (C). It is apparent that failing runs had a weight parameter b of 0.25–0.6 and $\boldsymbol{k}_1/\boldsymbol{k}_2$ lag sets that span the whole range, which suggests that this parameter subspace should be avoided.

MDS will project elements with same values in high-dimensional space to the same low-dimensional points, we would soon run into an occlusion problem—consider an analyst who keeps lag sets the same, but varies only the weight. There are a couple of ways to deal with occlusion, most notably lenses (Tominski et al., 2017). However, our users are not used to complex interactions, so we changed the tradeoff between position accuracy and occlusion. As an implementation of CorrelatedMultiples (Liu et al., 2018) was not available, we only rasterize the MDS plot and move overlapping points to the next free cell. When hovering over a point, the other points will change their size proportionally to the original dissimilarity, thereby allowing analysts to investigate projection errors.

6.4.2. Parameter comparison

To compare weights of different parametrizations, we encode triangle marks on a shared axis. Triangles are stacked if they would otherwise completely occlude each other. To compare lag

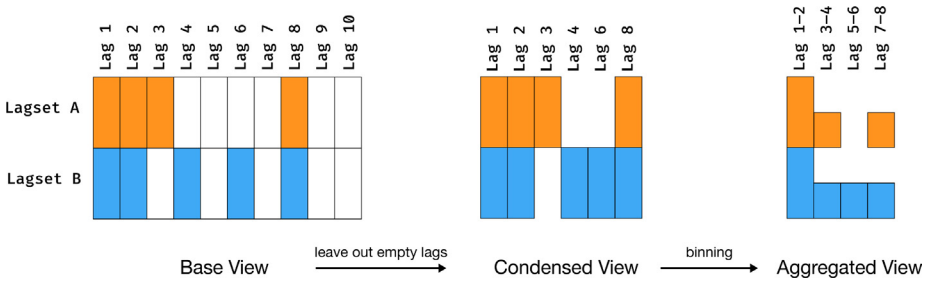


Fig. 6. Interweaved histograms encoding two lag sets A and B facilitate comparison. The base version encodes the presence of a lag by filling the corresponding rectangle with color hue. The condensed version, which is the default, omits lags that are in no lag set. Finally, when increasing the bin size (pictured: to 2), the analyst sees the aggregated view, where rectangles are transformed in small bar charts.

sets, we use interweaved histograms where the color saturation of a bar encodes the lag size to give an additional visual hint of the lag distribution, and to be consistent with the encoding in the lag selection (Section 6.5.1). Fig. 6 shows how they are generated. First, individual bars are positioned in a grid such that bars of the same lag set are in the same row, and bars representing the same bin are in the same column (base view). To save display space, empty columns are hidden by default (condensed view), but can be shown after user interaction. Increasing the bin size leads to familiar histogram shapes (aggregated view). Interweaved histograms show distinct images for same (bars align vertically and have similar height) and different lag sets (bars appear interweaved).

6.5. Task C5: Compare possible parameters

To obtain a new result, analysts need to select parameters. They consist in the case of gSOBI of two lag sets and one weight (Section 3).

To facilitate this selection process, we used the guidance design framework (Ceneda et al., 2020) to design appropriate guidance (Ceneda et al., 2017). Analysts do not know which lags to select and are generally aware of this *knowledge gap*. As discussed in Section 5.2, the *analysis goal* is to obtain a new/interesting result. Issues occur in the phase of lag selection, because the space of possible lag sets is huge. Analysts currently do not use additional information about lags, mostly due to time constraints. The *knowledge gap* lies in the execution and relates to the input data. We opt for *orienting* guidance, because analysts select lags also based on past experience and domain knowledge, so stronger guidance could be detrimental, and because our guidance *input* is not (cannot be) the "true" data: We compute it from the input data, which are per BSS model a linear combination of the components we are interested in. Based on the input data, we calculate guidance *output* per lag that help relate them to each other:

Guidance Output (GO) 1: Calendar relation. We compute which lag fits best to intervals in bigger calendar granules. The benefit of this is two-fold. First, lags are abstract and do not consider the calendar used in the data, so thinking in terms of days, weeks, etc., is a more intuitive alternative for someone familiar with the data. Second, it allows us to organize lags by filtering to those which correspond to a difference in a given calendar granule, thereby reducing the amount of lags to reason about.

GO2: Largest autocorrelation in input time series. White noise is a serially uncorrelated process, i.e., does not exhibit autocorrelation, so this measure indicates a latent component might not be white noise.

GO3: Eigenvalue difference in autocovariance matrices. The analysts use it to learn more about the input data and it can inform the parameter selection as lags should be chosen such that this eigenvalue difference is big (see Section 3).

GO4: Cross-moment matrix diagonality. This can only be computed when a parametrization of a successful run is refined, i.e., an unmixing matrix estimate exists. It shows the analyst which selected lags had an impact on the diagonality of autocovariance and fourth cross-cumulant matrices. It can be understood as feedback into the guidance system.

6.5.1. Lag selection

We support selection of a single lag set with multiple coordinated views (see Fig. 7). The lag size is encoded with color saturation, to make long, medium, and short lags distinguishable in all views, which is roughly how analysts reason about lag sets.

A parallel coordinates plot (PCP) displays all lags corresponding to a selected calendar granule (Fig. 7A), which can be configured by the user. Its dimensions are GO1–4, and values of selected lags are displayed as triangle marks next to the axes. The PCP supports common interactions such as inverting dimensions, reordering dimensions, and brushing. It is used to reduce the parameter space to a manageable subset. This subset is then visualized in a multivariate autocorrelation function plot (MACF), so analysts can view the temporal structure of all variables (Fig. 7B). The MACF shows all univariate autocorrelation function plots, composited through nesting: One box contains the autocorrelations of all variables at a given lag. Autocorrelations are encoded as bars, as in the univariate version, and can be sorted by variable name or by value. The latter is the default because it shows the distribution of autocorrelation values. Hovering over a box highlights the lag, which affects the next view below it. Clicking a box adds or removes the lag from/to the selection, which is shown in the right column in the same fashion as interweaved histograms (Fig. 6).

Underneath the MACF we display a user-selected input time series as line graph, and a scatterplot of the time series' values vs. the values lagged by the currently highlighted lag (Fig. 7C). This allows the analyst to find correlation patterns which are not surfaced by the MACF or the time series itself. A line on top of the time series shows the extent of the currently highlighted lag in context.

These views allow the analyst to interactively explore possible lags. Should they exactly know what they want to select, or rather not use an interactive system because they are used to static tools, they can enter the desired lags in the input box in the right column (Fig. 7D) in a format similar to R's `seq` shorthand syntax and proceed.

6.6. Task C3/I2: Compare unmixing matrices

We support this task (I2) by showing the factors as a heatmap where a univariate color scale encodes the absolute value in a row with white (low value) to black. When analysts see interesting patterns, they can select cells, and the respective input data and components will be shown underneath the matrices (Fig. 8H).

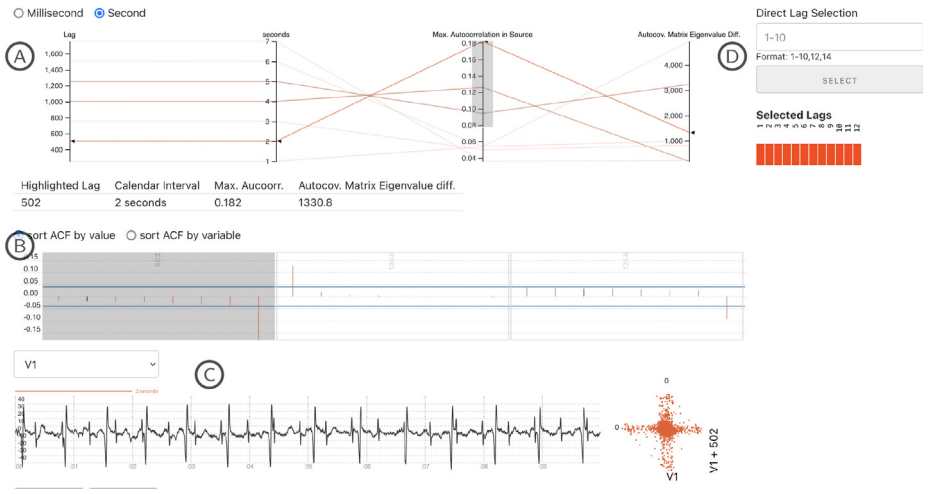


Fig. 7. The Lag Selection view (ECG data): Lag size is encoded with color saturation. Lags are filtered to those corresponding to a temporal difference of multiple seconds in the underlying calendar. The PCP (A) further narrows them down to those with high autocorrelation. A MACF plot (B) shows the autocorrelation of input data at brushed lags. Lags can be selected by clicking and highlighted by hovering in the MACF plot. A user-selected input time series is shown underneath (C) next to a scatterplot of the datapoints of the series at the currently highlighted lag. The right-most column (D) allows analysts to skip the interaction, and shows the current selection.

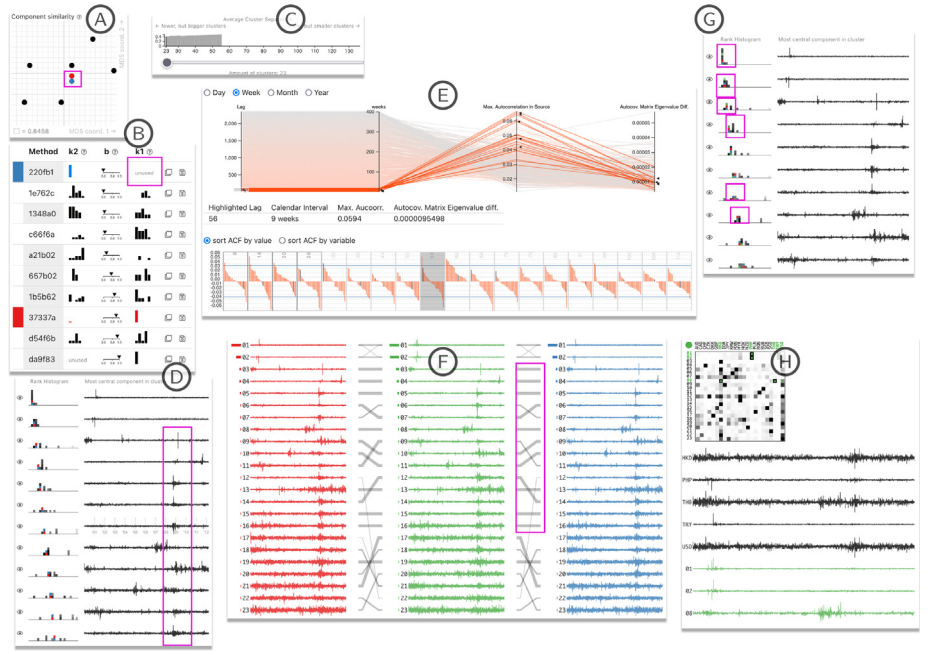


Fig. 8. Usage scenario on financial dataset, see Section 7.1 for details.

This allows to investigate the relationship between inputs and components. Task C3 is also supported, for which we encode a BSS-specific similarity measure (Ilmonen et al., 2010) in a heatmap with a univariate color scale.

7. Usage scenarios

In this section we describe how the designed visualizations (Section 6) provide insights into the presented datasets (Section 4). The financial dataset was used in our user studies (Section 8), while we added the medical dataset ourselves to provide broader context to the reader. The usage scenarios we describe are based on what we learned during aforementioned user studies and also during discussions with our collaborators.

7.1. Financial data

We load the financial dataset (Section 4.1) of 23 currency exchange rates to Euro into TBSSvis and start with 10 parameter settings. From the *Component Similarity View* (Fig. 8A, Section 6.4.1) we can immediately see that two component sets are very similar as they are very close to each other (purple highlight) and hovering does not change their relative sizes. Selecting them reveals that one of them did not use the $\boldsymbol{k}_1$ lag set at all (Fig. 8B), suggesting that this parameter's influence is small and we should focus on $\boldsymbol{k}_2$ when selecting parameters. This is in line with our expectations of financial data (Section 3). Looking at the clustering visualizations, no clear picture emerges. The *Clustering Quality* (Section 6.3.2) increases slowly with the number of clusters, but there is no distinctive peak (Fig. 8C). Thus, we expect that all components are somewhat similar to each other. Inspection of

the *Component Overview* (Section 6.3.3) confirms that, as many components share a similar pattern: They are very noisy with more extreme values during the years 2008–2009 (Fig. 8D, purple highlight marks years 2008–2009). This was the time of the global financial crisis. We try to obtain an alternative result and go to the *Parameter Selection*. We set the weight b to zero and do not use SOBI part ($\boldsymbol{k}_1$ parameter) at all, following our initial hypothesis. In the *Lag Selection* (Fig. 8E, Section 6.5.1) for $\boldsymbol{k}_2$ we quickly select lags that correspond to 1–3 days, 1–4 weeks, 1–3 months and 1 year intervals in the underlying calendar. We do it this way because available guidance outputs do not seem informative due to the amount of noise in the dataset. E.g., the autocorrelation (GO2) of weekly lags is very low (at most 0.06). The newly computed result is colored green in TBSSvis and automatically selected. We look at its components and compare it to the two identical results. The *Slope Graph* (Fig. 8F, Section 6.3.4) shows many thick lines that connect identical components (purple highlight). As we want to find currencies to invest in, we turn to the *Component Overview* again. The histograms show stacked and saturated bars, thus suggesting that the first couple of components are stable and common in all results (Fig. 8G). We, therefore, pick three that have volatile segments outside of 2008–2009 to rule out a global financial crisis as the cause for volatility. The *Unmixing Matrix* visualization (Section 6.6) shows which currencies are associated with these components (Fig. 8H, black time series). We will ask our financial advisor about investing in Thai bhat, US dollars, Turkish lira, or Philippine pesos.

7.2. Medical data

We load the ECG dataset (Section 4.2) from a pregnant woman into TBSSvis. Looking at the raw inputs in the *Input Visualization* we can confirm that the fetal heart signal is visible in the mother's ECG (Fig. 9A, purple highlight). We start with 10 precomputed parameter settings, 7 of which succeed. The *Clustering Quality* (Fig. 9B, Section 6.3.2) suggests that 8–11 meaningfully different components were obtained, as the height of bars steadily declines afterwards. We set the clustering to 10 partitions. A healthy fetus has a heart rate of 110–160 beats/minute on average, which is higher than that of an adult (60–100). A candidate component for the fetal heart signal, which shows peaks of increased frequency, is readily visible as 4th (sorted by kurtosis) in the *Component Overview* (Fig. 9C, Section 6.3.3). The rank histogram next to the cluster medoid shows that components in the cluster are very similar, as all boxes are quite black, and it can be confirmed by looking at components directly (Fig. 9C). We select a couple of results containing this component to compare their parameters. We see that the parameters vary wildly (Fig. 9D), and the fetal heart signal was found using long and short lags for either lag set with different weights. This, along with the absence of other candidate components, suggests that we found the correct signal. A medical doctor would be able to inspect the obtained fetal ECG wave patterns in detail and determine whether or not it is healthy.

Looking at the values of the three parameter settings that did not produce results, we can also form an initial hypothesis about the useful parameter subspace (Fig. 9E). The weight b alone did not seem to play too much of a role as values span a wide range (0.11–0.94) and we do have several successful results within that range. But an apparent difference to those parameter settings is that both lag sets in failing results had lags that were distributed over the whole range instead of sticking to either the short or long end. This can be seen from Fig. 9D and E, as lags of the former appear blocked, whereas they are more interweaved in the latter. Thus, when trying to find new parameters for this dataset, we would steer clear of such lag sets.

8. Evaluation

To assess the usefulness of our visualization design, we conducted two interviews with five TBSS experts external to the project. Our research questions were:

RQ1 What are advantages and disadvantages of TBSSvis in comparison to their current tools?
RQ2 Does TBSSvis in fact support the analysis tasks?
RQ3 What are possible improvements to TBSSvis?

We decided for an *Expert Review* (Elmqvist and Yi, 2015) using interviews, as no comparable tool for a quantitative evaluation exists and qualitative data allows much deeper insights. Two interview cycles were conducted: In the first we gathered initial external feedback and supporting evidence for our task abstraction, and in the second we verified that this feedback was integrated accordingly. They lasted 2.5 h and 1 h, respectively.

8.1. Participants

Participants were the same for both interviews and previous collaborators of our co-authors. They participated voluntarily without promised benefits, financial or other. All are adults and not dependent on any author, be it financially, professionally or personally.

Our five experts (E1–E5) all hold a Ph.D. degree in mathematics or statistics. Four obtained their Ph.D. with research in BSS somewhat recently, while the other researches BSS already for 10 years. Therefore, they more than fit the *basic knowledge of BSS* and *formal education in math/statistics* requirements from our user characterization (Section 5.1). Since participants applied (T)BSS on diverse datasets and collaborated with various domain experts both in the context of (T)BSS (e.g., genome biology, cancer research) as well as outside of it (e.g., ecology, neurology), we think they are very well suited to answer our research questions. Although they cannot provide us with deep data-related insights as they are not application domain experts, they are our primary intended users and bring sufficient experience and a broad perspective to our research questions around TBSS analysis and involved tasks. This helps us to keep TBSSvis generic, yet effective, as was our design goal (Section 6).

No participant uses visual-interactive tools regularly. Their self-assessed experience in visualization is "basic", as E4 put it: "I only use what R has to offer, like ggplot and the base graphics (...) scatterplots, time series plots, (...) box plots. I tend to stick with these basic kinds of plots (...)".

8.2. Methodology

The interviews were conducted and recorded via Zoom with explicit consent by participants.[1] Two researchers were involved in each interview, one tasked with moderation and one took notes. Participants used TBSSvis on their own machines and shared their screen during usage. We used Zoom annotations to point out relevant parts of TBSSvis when necessary.

Both sessions were structured the same. We compiled a text explanation with images of TBSSvis, so that participants can familiarize themselves with it beforehand. The tutorial document was sent to participants together with the consent form ahead of the interview. Steps during the interview were as follows:

[1] As of manuscript submission, the TU Wien has a Pilot Research Ethics Committee. Approaching it for peer review of research with human participants is not required by the TU Wien, and its response is non-binding. Therefore we do not provide an official ethics approval. Nonetheless, we believe we conducted our research adhering to sufficient ethical standards.

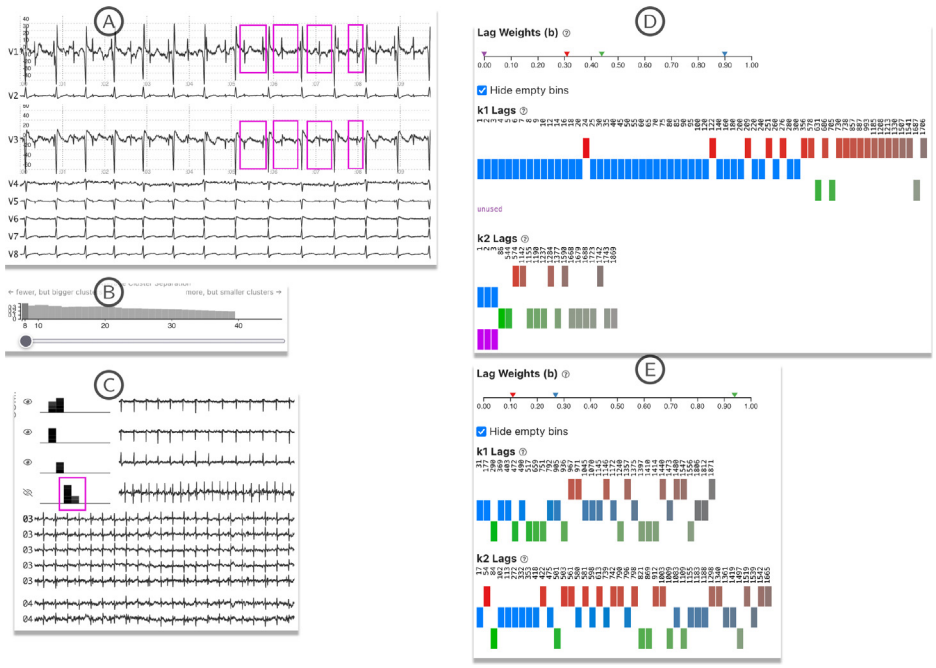


Fig. 9. Usage scenario on medical dataset, see Section 7.2 for details. Note that (D) and (E) show different parameter settings.

1. (Only in first session.) We conducted a structured interview about their background and experiences with (T)BSS (15–30 min).
2. We gave participants a structured introduction to interactions and visualizations in TBSSvis (up to 1–hour). The dataset used was synthetic and unfamiliar to them. We asked participants to solve small tasks to practice what we explained. We skipped these tasks when we either saw that they understood it, or when we were short on time.
3. (Optional.) Participants were allowed to further use TBSSvis for some minutes on their own.
4. We asked participants to conduct an open analysis on the dataset used in Section 4.1, which most have worked on in the past, and articulate their thoughts and plans ("think aloud"). We pointed out parts of TBSSvis they did not use or consider so far. This step took around 30–minutes.
5. We discussed tasks, visualizations, interactions and possible further improvements in an unstructured fashion (15–30 min). Before we finished the session, we encouraged participants to use TBSSvis more without our supervision.

To answer RQ2 we found it sufficient to check whether or not participants can interpret our visualizations, and if visualizations show the necessary data in the right moment to support their tasks. To do so, we analyzed the recorded video and notes after each session. We looked for articulated suggestions, discussions, and situations where users interacted with visualizations. These instances were transcribed and grouped by tasks (Section 5.2). Feedback and possible issues of participants were noted, deduplicated, and presented to our collaborators. Subsequent discussions then informed changes to the first design, which we confirmed in the second interview.

The interview guide, tutorial documents, datasets, and our transcripts of the interviews are available as supplementary material.

8.3. Expert feedback

We describe evidence for our research questions in this section.

8.3.1. RQ1: Advantages and disadvantages

Our participants agreed that TBSSvis has clear advantages compared to current tools used and greatly improves the analysis process. E5 even said that TBSSvis is "an absolute time saver" and "very useful for applied work." The majority of them mentioned that it is easier than in RStudio to compare components, matrices, and parameters. The same outcomes can be achieved faster in TBSSvis and it provides useful new visualizations they could not have in RStudio, such as the component overview/comparison. All participants mentioned to enjoy playing with TBSSvis. Even in our limited time we saw indications how TBSSvis can change the way they work. We observed E4 in the open session to pursue an analysis process resembling binary search, toggling individual lags on and off. Asked about it later, E4 mentioned to be "not sure if I'd have thought about [this approach] just with RStudio." E5 was very eager to get hold of TBSSvis, as the intention was to recommend it to their students. E1 stated that better supported comparison tasks give more structure to the analysis process, so all this suggests that TBSSvis allows new or more streamlined analyses.

As for disadvantages, there is one very basic: RStudio allows more flexible and specialized computations than TBSSvis. However, this was not explicitly mentioned by participants. Some said it took time to put everything together, but all our participants managed to do so quickly. A few plots were difficult to understand at first, but after explanations it was relatively easy to use for all participants. In addition, we observed some participants having trouble with idioms that are common and popular in the visualization community, such as PCPs and multiple linked views, which could be overcome by visual literacy efforts.

8.3.2. RQ2: Supported tasks

In this section, we discuss how TBSSvis supports analysis tasks (Section 5). We provide quotes from participants to let them speak for themselves, but their sentiment is shared by the majority and not an isolated opinion.

Identify used parameters (I1): The tabular overview (Fig. 5A) was considered "really useful" (E2) and participants thought it "makes a lot of sense" (E4).

Identify unmixing matrix (I2): Participants could easily identify similar matrices and dominant factors of components. Viewing involved time series (Fig. 8H) was considered useful.

Identify cross-moment diagonality (I3): It is "something I don't usually have the time and energy to compute" (E5) and "very interesting" (E1), but also something they do not regularly use for their analysis today.

Identify components (I4): Our participants found the added interactivity compared to RStudio very useful.

Compare success (C1): They had no trouble with visual encodings, but participants sometimes forgot that failure is an option.

Compare parameters (C2): While the interweaved lag histograms were easy to interpret, it took some time for participants to realize that it is a regular histogram with hidden bins (Fig. 6). Similarity projections of parameters (Fig. 5B) were rarely used by our participants. A possible explanation is because histograms show more data and participants worked with only 5–7 parametrizations, they could use their working memory. We believe their benefits would have become apparent with more parametrizations.

Compare unmixing matrices (C3): Some (E3–E5) mentioned that interpreting the MD-Index for other than extreme values is not easy as it depends on the data dimensionality. While both visualizations were used by all, some participants seemed to prefer the MD-Index (E3, E4) over the factors (E2) to compare matrices.

Compare component sets (C4): Participants understood the slope graph (Fig. 4) easily and immediately saw its benefits. E3 mentioned that using it is "easier than looking at a correlation matrix." The projection view was in fact used to see how similar ensemble members are. For this purpose participants also appreciated the component overview ("you can very fast get an idea of how similar different methods are", E3), although most did not change the initial clustering parameter.

Compare possible parameters (C5): After we introduced participants to individual views and interactions, they learned quickly how to use it and found it useful and convenient. They understood how and why to filter visualized lags, but were not sure about the data-driven calendar-based approach, presumably because they currently analyze data detached from any calendar. Participants appreciated the PCP with its dimensions, even though they sometimes did not know right away how to interpret all of them: For example, E2 asked what the eigenvalue metric means, what the optimal choice is, and if lower or higher is better. Participants were also sometimes irritated by the number of dimensions, as they depend on the outcome of the refined run.

8.3.3. RQ3: Possible improvements

When asked about improvements to TBSSvis, we got responses mainly pertaining to the parameter selection. E4 would prefer if the syntax to directly select lags matched commands available in R. E2–E4 often ended up with an empty selection in the PCP because they expected brushes to be combined with union instead of intersection. They also want to select all filtered lags and remove all selected lags at once. Aside from the lag selection improvements, more DOI functions would be appreciated. We added one measure for periodicity (Vlachos et al., 2005) following the suggestion of one participant. E5 suggested to support loading precomputed results, possibly from other TBSS methods. E2 asked for more legends, explanations, and a stronger guidance degree in general. E1 suggested the ability to freely reorder components everywhere, and providing alternative color palettes. With E1 we also discussed the option of showing correlations between input data in the *Input Visualization* screen as another sanity check.

9. Reflection and discussion

Reflecting on our findings and lessons learned during our design study with experts in BSS, we claim that TBSSvis supports tasks involved with TBSS analysis (Section 5) and encourages usage of TBSS in various application domains. Despite differences in what an application domain considers interesting in latent dimensions (e.g., doctors might search for specific wave patterns, while investors look for sudden and extreme changes), many tasks are the same. We showed this transferability to financial and medical datasets in Section 7. We developed and evaluated TBSSvis with TBSS experts, who are our primary intended users. They worked with many domain experts in the past to apply TBSS in their respective fields. Their practical experience with different use cases for TBSS informed our visualization design (Section 6). Therefore, based on the mostly positive feedback by our interview participants, we expect that TBSSvis can be useful in many application domains.

In line with the design study methodology (Sedlmair et al., 2012), we used well known visualization idioms and data mining algorithms, applied them in a new context and extended them as necessary. As a consequence, individual parts of TBSSvis will be useful to other visualization researchers and designers. For instance, a slope graph usually shows categorical data cases and their change of rank by line slope. We adapted it to time series by encoding similarity in line thickness. In our user studies it was considered an easy-to-understand visualization to visually compare sets of time series. The clustering scheme (Section 6.3) is useful whenever members of sets should be clustered and set membership must be taken into account. It works with any dissimilarity measure because it is based on k-medoids. Set-typed data is prevalent (Alsallakh et al., 2016), so we expect this to be useful to others.

9.1. Design process

Following the recommendations of the data-users-tasks design triangle (Miksch and Aigner, 2014) our proposed visualizations are close to what TBSS experts are used to and therefore quite simple. We also did not include more advanced interactions than highlighting, filtering, hovering, or brushing because TBSS experts come from a text-based software where even these do not exist. Looking back, we think this was a good decision, as in our interviews some participants had initially trouble using, e.g., the PCP.

What was difficult for us visualization researchers during the design is the domain-independence of TBSS. Our goal, therefore, was to make TBSSvis applicable in a wide range of domain-specific contexts, e.g., in medicine or finance. But both size and complexity of the data vary considerably among the domains, as do the definitions of "interesting" features and the location and role of TBSS in the data processing pipeline (von Landesberger et al., 2017). Therefore, we opted in the end for simple interactions and generic/extendable approaches, such as the use of DOI functions, to avoid a "lock-in" to any specific application domain.

9.2. Limitations and future work

We discuss some limitations in our paper. Most study participants used the financial dataset (Section 4.1) at some point in the past to test varying TBSS methods. Although participants fit well to our user description (Section 5.1), they were not as intimately familiar with the dataset as it is often the case in visualization-related evaluations. Had this been the case, we may have found additional analysis goals and insights. Nevertheless,

we maintain that our study methodology and participant selection was sufficient and appropriate to investigate how TBSSvis impacts involved tasks (Section 5.2). Participants used TBSSvis in both interviews for in total around 45 minutes on their own terms. More time using it may have surfaced additional necessary analysis tasks or improvement suggestions.

As part of our future work, we would like to integrate the suggested improvements by our experts, support larger datasets and allow provision of custom DOI functions.

10. Summary and conclusion

We presented TBSSvis, a VA solution for TBSS. TBSS is in a way similar to PCA, in that it can be used to analyze suitable datasets from any application domain, such as biomedical analysis, finance, or civil engineering. Unlike PCA, TBSS properly accounts for temporal correlation and requires complex tuning parameters. Because of these parameter settings, TBSS analysis is inherently open-ended and exploratory as there are no known insights to confirm. TBSSvis is based on a task abstraction and visualization design that we developed together in a user-centered design process with TBSS experts. We evaluated the final interactive prototype with five other TBSS experts, who did not participate in the design process, by conducting two interviews. Feedback from these shows that TBSSvis supports the actual workflow and combination of interactive visualizations that facilitate the tasks involved in analyzing TBSS results—this process was previously a laborious back-and-forth for which analysts had to manually program static visualizations and data mining algorithms. TBSSvis also provides guidance to facilitate the analysis of the data at hand and informed parameter selection, which was previously mostly a guessing game.

CRediT authorship contribution statement

Nikolaus Piccolotto: Conceptualization, Software, Validation, Investigation, Writing – original draft, Writing – review & editing, Visualization, Data curation. **Markus Bögl:** Conceptualization, Validation, Investigation, Writing – review & editing. **Theresia Gschwandtner:** Conceptualization, Writing – review & editing. **Christoph Muehlmann:** Conceptualization, Writing – original draft, Writing – review & editing. **Klaus Nordhausen:** Conceptualization, Resources, Writing – original draft, Writing – review & editing, Supervision, Project administration, Funding acquisition. **Peter Filzmoser:** Conceptualization, Writing – review & editing, Supervision, Funding acquisition. **Silvia Miksch:** Conceptualization, Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Ethical Approval

All procedures followed were in accordance with the ethical standards of the institutional and/or national research committee and with the 1964 Helsinki Declaration and its later amendments or comparable ethical standards.

Acknowledgments

We would like to thank our experts for spending so much of their valuable time on discussions and evaluations with us. This work was supported by the Austrian Science Fund (FWF) under grant P31881-N32.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.visinf.2022.10.002.

References

Aigner, W., Miksch, S., Schumann, H., Tominski, C., 2011. Visualization of Time-Oriented Data. Springer, Berlin, Heidelberg.

Alsallakh, B., Micallef, L., Aigner, W., Hauser, H., Miksch, S., Rodgers, P., 2016. The State-of-the-Art of Set Visualization. Comput. Graph. Forum 35 (1), 234–260. http://dx.doi.org/10.1111/cgf.12722.

Amezquita-Sanchez, J.P., Adeli, H., 2016. Signal Processing Techniques for Vibration-Based Health Monitoring of Smart Structures. Arch. Comput. Methods Eng. 23 (1), 1–15. http://dx.doi.org/10.1007/s11831-014-9135-7.

Anand, A., Wilkinson, L., Dang, T.N., 2012. Visual pattern discovery using random projections. In: Proceedings VAST. IEEE, New York, NY, USA, pp. 43–52. http://dx.doi.org/10.1109/VAST.2012.6400490.

Belouchrani, A., Abed-Meraim, K., Cardoso, J.-F., Moulines, E., 1997. A blind source separation technique using second-order statistics. IEEE Trans. Signal Process. 45 (2), 434–444. http://dx.doi.org/10.1109/78.554307.

Bögl, M., Aigner, W., Filzmoser, P., Lammarsch, T., Miksch, S., Rind, A., 2013. Visual Analytics for Model Selection in Time Series Analysis. IEEE Trans. Vis. Comput. Graphics 19 (12), 2237–2246. http://dx.doi.org/10.1109/TVCG.2013.222.

Brehmer, M., Munzner, T., 2013. A multi-level typology of abstract visualization tasks. IEEE Trans. Vis. Comput. Graphics 19 (12), 2376–2385. http://dx.doi.org/10.1109/TVCG.2013.124.

Buchmüller, J., Jäckle, D., Cakmak, E., Brandes, U., Keim, D.A., 2019. MotionRugs: Visualizing Collective Trends in Space and Time. IEEE Trans. Vis. Comput. Graphics 25 (1), 76–86. http://dx.doi.org/10.1109/TVCG.2018.2865049.

Ceneda, D., Andrienko, N., Andrienko, G., Gschwandtner, T., Miksch, S., Piccolotto, N., Schreck, T., Streit, M., Suschnigg, J., Tominski, C., 2020. Guide Me in Analysis: A Framework for Guidance Designers. Comput. Graph. Forum 39 (6), 269–288. http://dx.doi.org/10.1111/cgf.14017.

Ceneda, D., Gschwandtner, T., May, T., Miksch, S., Schulz, H., Streit, M., Tominski, C., 2017. Characterizing Guidance in Visual Analytics. IEEE Trans. Vis. Comput. Graphics 23 (1), 111–120. http://dx.doi.org/10.1109/TVCG.2016.2598468.

Choo, J., Lee, H., Kihm, J., Park, H., 2010. iVisClassifier: An interactive visual analytics system for classification based on supervised dimension reduction. In: Proceedings VAST. IEEE, New York, NY, USA, pp. 27–34. http://dx.doi.org/10.1109/VAST.2010.5652443.

Comon, P., Jutten, C., 2010. Handbook of Blind Source Separation: Independent Component Analysis and Applications. Academic Press.

de Lathauwer, L., de Moor, B., Vandewalle, J., 2000. Fetal electrocardiogram extraction by blind source subspace separation. IEEE Trans. Biomed. Eng. 47 (5), 567–572. http://dx.doi.org/10.1109/10.841326.

Dingen, D., van't Veer, M., Houthuizen, P., Mestrom, E.H.J., Korsten, E.H., Bouwman, A.R., van Wijk, J., 2019. RegressionExplorer: Interactive Exploration of Logistic Regression Models with Subgroup Analysis. IEEE Trans. Vis. Comput. Graphics 25 (1), 246–255. http://dx.doi.org/10.1109/TVCG.2018.2865043.

Elmqvist, N., Yi, J.S., 2015. Patterns for visualization evaluation. Inform. Visual. 14 (3), 250–269. http://dx.doi.org/10.1177/1473871613513228.

Ferstl, F., Kanzler, M., Rautenhaus, M., Westermann, R., 2017. Time-Hierarchical Clustering and Visualization of Weather Forecast Ensembles. IEEE Trans. Vis. Comput. Graphics 23 (1), 831–840. http://dx.doi.org/10.1109/TVCG.2016.2598868.

Fu, T.-c., 2011. A review on time series data mining. Eng. Appl. Artif. Intell. 24 (1), 164–181. http://dx.doi.org/10.1016/j.engappai.2010.09.007.

Gogolou, A., Tsandilas, T., Palpanas, T., Bezerianos, A., 2019. Comparing Similarity Perception in Time Series Visualizations. IEEE Trans. Vis. Comput. Graphics 25 (1), 523–533. http://dx.doi.org/10.1109/TVCG.2018.2865077.

Hao, L., Healey, C.G., Bass, S.A., 2016. Effective Visualization of Temporal Ensembles. IEEE Trans. Vis. Comput. Graphics 22 (1), 787–796. http://dx.doi.org/10.1109/TVCG.2015.2468093.

Hao, L., Healey, C.G., Hutchinson, S.E., 2015. Ensemble visualization for cyber situation awareness of network security data. In: Proceedings VizSec. IEEE, New York, NY, USA, pp. 1–8. http://dx.doi.org/10.1109/VIZSEC.2015.7312766.

Ilmonen, P., Nordhausen, K., Oja, H., Ollila, E., 2010. A New Performance Index for ICA: Properties, Computation and Asymptotic Analysis. In: Vigneron, V., Zarzoso, V., Moreau, E., Gribonval, R., Vincent, E. (Eds.), Latent Variable Analysis and Signal Separation. In: Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, pp. 229–236. http://dx.doi.org/10.1007/978-3-642-15995-4_29.

Javed, W., McDonnel, B., Elmqvist, N., 2010. Graphical Perception of Multiple Time Series. IEEE Trans. Vis. Comput. Graphics 16 (6), 927–934. http://dx.doi.org/10.1109/TVCG.2010.162.

Keim, D., Andrienko, G., Fekete, J.-D., Görg, C., Kohlhammer, J., Melançon, G., 2008. Visual Analytics: Definition, Process, and Challenges. In: Kerren, A., Stasko, J.T., Fekete, J.-D., North, C. (Eds.), Information Visualization. In: Lecture Notes in Computer Science no. 4950, Springer, Berlin, Heidelberg, pp. 154–175. http://dx.doi.org/10.1007/978-3-540-70956-5_7.

Kincaid, R., 2010. SignalLens: Focus+Context Applied to Electronic Time Series. IEEE Trans. Vis. Comput. Graphics 16 (6), 900–907. http://dx.doi.org/10.1109/TVCG.2010.193.

Köthur, P., Witt, C., Sips, M., Marwan, N., Schinkel, S., Dransch, D., 2015. Visual Analytics for Correlation-Based Comparison of Time Series Ensembles. Comput. Graph. Forum 34 (3), 411–420. http://dx.doi.org/10.1111/cgf.12653.

Liu, X., Hu, Y., North, S., Shen, H.-W., 2018. CorrelatedMultiples: Spatially Coherent Small Multiples With Constrained Multi-Dimensional Scaling. Comput. Graph. Forum 37 (1), 7–18. http://dx.doi.org/10.1111/cgf.12526.

Liu, X., Wang, H., Huang, M., Yang, W., 2019. An Improved Second-Order Blind Identification (SOBI) Signal De-Noising Method for Dynamic Deflection Measurements of Bridges Using Ground-Based Synthetic Aperture Radar (GBSAR). Appl. Sci. 9 (17), 3561. http://dx.doi.org/10.3390/app9173561, URL https://www.mdpi.com/2076-3417/9/17/3561.

Mackinlay, J., 1986. Automating the design of graphical presentations of relational information. ACM Trans. Graph. 5 (2), 110–141. http://dx.doi.org/10.1145/22949.22950.

Matilainen, M., Miettinen, J., Nordhausen, K., Oja, H., Taskinen, S., 2017. On Independent Component Analysis with Stochastic Volatility Models. Austrian J. Stat. 46 (3–4), 57–66. http://dx.doi.org/10.17713/ajs.v46i3-4.671.

Matković, K., Gračanin, D., Hauser, H., 2018. Visual Analytics for Simulation Ensembles. In: Proceedings WSC. IEEE, New York, NY, USA, pp. 321–335. http://dx.doi.org/10.1109/WSC.2018.8632312.

Matkovic, K., Gracanin, D., Klarin, B., Hauser, H., 2009. Interactive Visual Analysis of Complex Scientific Data as Families of Data Surfaces. IEEE Trans. Vis. Comput. Graphics 15 (6), 1351–1358. http://dx.doi.org/10.1109/TVCG.2009.155.

McLachlan, P., Munzner, T., Koutsofios, E., North, S., 2008. LiveRAC: Interactive Visual Exploration of System Management Time-series Data. In: Proceedings CHI. ACM, New York, NY, USA, pp. 1483–1492. http://dx.doi.org/10.1145/1357054.1357286.

Miettinen, J., Illner, K., Nordhausen, K., Oja, H., Taskinen, S., Theis, F., 2016. Separation of uncorrelated stationary time series using autocovariance matrices. J. Time Series Anal. 37 (3), 337–354.

Miettinen, J., Matilainen, M., Nordhausen, K., Taskinen, S., 2020. Extracting Conditionally Heteroskedastic Components using Independent Component Analysis. J. Time Series Anal. 41 (2), 293–311. http://dx.doi.org/10.1111/jtsa.12505.

Miettinen, J., Nordhausen, K., Oja, H., Taskinen, S., 2014. Deflation-based separation of uncorrelated stationary time series. J. Multivariate Anal. 123, 214–227. http://dx.doi.org/10.1016/j.jmva.2013.09.009.

Miksch, S., Aigner, W., 2014. A Matter of Time: Applying a Data-users-tasks Design Triangle to Visual Analytics of Time-oriented Data. Comput. Graph. 38, 286–290. http://dx.doi.org/10.1016/j.cag.2013.11.002.

Mühlbacher, T., Piringer, H., 2013. A Partition-Based Framework for Building and Validating Regression Models. IEEE Trans. Vis. Comput. Graphics 19 (12), 1962–1971. http://dx.doi.org/10.1109/TVCG.2013.125.

Munzner, T., 2009. A Nested Model for Visualization Design and Validation. IEEE Trans. Vis. Comput. Graphics 15 (6), 921–928. http://dx.doi.org/10.1109/TVCG.2009.111.

Nonato, L.G., Aupetit, M., 2019. Multidimensional Projection for Visual Analytics: Linking Techniques with Distortions, Tasks, and Layout Enrichment. IEEE Trans. Vis. Comput. Graphics 25 (8), 2650–2673. http://dx.doi.org/10.1109/TVCG.2018.2846735.

Nordhausen, K., Matilainen, M., Miettinen, J., Virta, J., Taskinen, S., 2021. Dimension reduction for time series in a blind source separation context using R. J. Stat. Softw. Articles 98 (15), 1–30. http://dx.doi.org/10.18637/jss.v098.i15.

Nordhausen, K., Oja, H., 2018. Independent component analysis: A statistical perspective. WIREs Comput. Stat. 10 (5), e1440. http://dx.doi.org/10.1002/wics.1440.

Nordhausen, K., Ruiz-Gazen, A., 2022. On the usage of joint diagonalization in multivariate statistics. J. Multivariate Anal. 188, 104844. http://dx.doi.org/10.1016/j.jmva.2021.104844.

Obermaier, H., Bensema, K., Joy, K.I., 2016. Visual Trends Analysis in Time-Varying Ensembles. IEEE Trans. Vis. Comput. Graphics 22 (10), 2331–2342. http://dx.doi.org/10.1109/TVCG.2015.2507592.

Oja, E., Kiviluoto, K., Malaroiu, S., 2000. Independent component analysis for financial time series. In: Proceedings AS-SPCC. IEEE, New York, NY, USA, pp. 111–116. http://dx.doi.org/10.1109/ASSPCC.2000.882456.

Pan, Y., Matilainen, M., Taskinen, S., Nordhausen, K., A review of second-order blind identification methods. WIREs Comput. Stat. http://dx.doi.org/10.1002/wics.1550, First published Feb. 2021, to appear.

Park, H.-S., Jun, C.-H., 2009. A simple and fast algorithm for K-medoids clustering. Expert Syst. Appl. 36 (2), 3336–3341. http://dx.doi.org/10.1016/j.eswa.2008.01.039.

Piringer, H., Pajer, S., Berger, W., Teichmann, H., 2012. Comparative Visual Analysis of 2D Function Ensembles. Comput. Graph. Forum 31 (3pt3), 1195–1204. http://dx.doi.org/10.1111/j.1467-8659.2012.03112.x.

Potter, K., Wilson, A., Bremer, P.-T., Williams, D., Doutriaux, C., Pas, V., Johnson, C.R., 2009. Ensemble-Vis: A Framework for the Statistical Visualization of Ensemble Data. In: Proceedings ICDM Workshops. IEEE, New York, NY, USA, pp. 233–240. http://dx.doi.org/10.1109/ICDMW.2009.55.

Qu, Z., Hullman, J., 2018. Keeping Multiple Views Consistent: Constraints, Validations, and Exceptions in Visualization Authoring. IEEE Trans. Vis. Comput. Graphics 24 (1), 468–477. http://dx.doi.org/10.1109/TVCG.2017.2744198.

Rosen, P., Suh, A., Salgado, C., Hajij, M., 2020. TopoLines: Topological Smoothing for Line Charts. In: Proceedings EuroVis Short Papers. Eurographics Association, Geneve, Switzerland, pp. 85–89. http://dx.doi.org/10.2312/evs.20201053.

Schubert, E., Rousseeuw, P.J., 2019. Faster K-Medoids Clustering: Improving the PAM, CLARA, and CLARANS Algorithms. In: Proceedings SISAP. In: Lecture Notes in Computer Science, vol. 11807, Springer, Berlin, Heidelberg, pp. 171–187. http://dx.doi.org/10.1007/978-3-030-32047-8_16.

Schwab, M., Hao, S., Vitek, O., Tompkin, J., Huang, J., Borkin, M.A., 2019. Evaluating Pan and Zoom Timelines and Sliders. In: Proceedings CHI. ACM, New York, NY, USA, pp. 1–12. http://dx.doi.org/10.1145/3290605.3300786.

Sedlmair, M., 2016. Design Study Contributions Come in Different Guises: Seven Guiding Scenarios. In: Proceedings BELIV. ACM, New York, NY, USA, pp. 152–161. http://dx.doi.org/10.1145/2993901.2993913.

Sedlmair, M., Heinzl, C., Bruckner, S., Piringer, H., Möller, T., 2014. Visual Parameter Space Analysis: A Conceptual Framework. IEEE Trans. Vis. Comput. Graphics 20 (12), 2161–2170. http://dx.doi.org/10.1109/TVCG.2014.2346321.

Sedlmair, M., Meyer, M., Munzner, T., 2012. Design Study Methodology: Reflections from the Trenches and the Stacks. IEEE Trans. Vis. Comput. Graphics 18 (12), 2431–2440. http://dx.doi.org/10.1109/TVCG.2012.213.

Shurkhovetskyy, G., Andrienko, N., Andrienko, G., Fuchs, G., 2018. Data Abstraction for Visualizing Large Time Series. Comput. Graph. Forum 37 (1), 125–144. http://dx.doi.org/10.1111/cgf.13237.

Stitz, H., Gratzl, S., Aigner, W., Streit, M., 2016. ThermalPlot: Visualizing Multi-Attribute Time-Series Data Using a Thermal Metaphor. IEEE Trans. Vis. Comput. Graphics 22 (12), 2594–2607. http://dx.doi.org/10.1109/TVCG.2015.2513389.

Sun, D., Feng, Z., Chen, Y., Wang, Y., Zeng, J., Yuan, M., Pong, T.-C., Qu, H., 2020. DFSeer: A Visual Analytics Approach to Facilitate Model Selection for Demand Forecasting. In: Proceedings CHI. ACM, New York, NY, USA, pp. 1–13. http://dx.doi.org/10.1145/3313831.3376866.

Tang, A.C., Liu, J.-Y., Sutherland, M.T., 2005. Recovery of correlated neuronal sources from EEG: The good and bad ways of using SOBI. NeuroImage 28 (2), 507–519. http://dx.doi.org/10.1016/j.neuroimage.2005.06.062.

Taskinen, S., Miettinen, J., Nordhausen, K., 2016. A more efficient second order blind identification method for separation of uncorrelated stationary time series. Statist. Probab. Lett. 116, 21–26. http://dx.doi.org/10.1016/j.spl.2016.04.007.

Team, R.C., 2020. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria, URL https://www.R-project.org/.

Thomas, J.J., Cook, K.A., 2005. Illuminating the Path: An R&D Agenda for Visual Analytics. National Visualization and Analytics Center, Department of Homeland Security, IEEE Computer Society.

Tominski, C., Gladisch, S., Kister, U., Dachselt, R., Schumann, H., 2017. Interactive Lenses for Visualization: An Extended Survey. Comput. Graph. Forum 36 (6), 173–200. http://dx.doi.org/10.1111/cgf.12871.

Tufte, E.R., 2001. The Visual Display of Quantitative Information, second ed. Graphics Press.

Van Thanh, P., Nhu Dinh, D., Duc Anh, N., Tien Anh, N., Duc Hoang, C., Duc-Tan, T., 2017. Automatic removal of EOG artifacts using SOBI algorithm combined with intelligent source identification technique. In: Proceedings ATC. IEEE, New York, NY, USA, pp. 260–264. http://dx.doi.org/10.1109/ATC.2017.8167629.

Venables, W.N., Ripley, B.D., 2010. Modern Applied Statistics with S, fourth ed. In: Statistics and Computing, Springer, New York.

Vlachos, M., Yu, P., Castelli, V., 2005. On Periodicity Detection and Structural Periodic Similarity. In: Proceedings of the 2005 SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics, pp. 449–460. http://dx.doi.org/10.1137/1.9781611972757.40.

von Landesberger, T., Fellner, D.W., Ruddle, R.A., 2017. Visualization System Requirements for Data Processing Pipeline Design and Optimization. IEEE Trans. Vis. Comput. Graphics 23 (8), 2028–2041. http://dx.doi.org/10.1109/TVCG.2016.2603178.

Wagstaff, K., Cardie, C., Rogers, S., Schroedl, S., 2001. Constrained K-means Clustering with Background Knowledge. In: Proceedings ICML. Morgan Kaufmann, San Francisco, CA, USA, pp. 577–584.

Wang, J., Hazarika, S., Li, C., Shen, H.-W., 2019. Visualization and Visual Analysis of Ensemble Data: A Survey. IEEE Trans. Vis. Comput. Graphics 25 (9), 2853–2872. http://dx.doi.org/10.1109/TVCG.2018.2853721.

Wexler, J., Pushkarna, M., Bolukbasi, T., Wattenberg, M., Viegas, F., Wilson, J., 2020. The What-If Tool: Interactive Probing of Machine Learning Models. IEEE Trans. Vis. Comput. Graphics 26 (1), 56–65. http://dx.doi.org/10.1109/TVCG.2019.2934619, arXiv:1907.04135.

Willett, W., Heer, J., Agrawala, M., 2007. Scented Widgets: Improving Navigation Cues with Embedded Visualizations. IEEE Trans. Vis. Comput. Graphics 13 (6), 1129–1136. http://dx.doi.org/10.1109/TVCG.2007.70589.

Xia, J., Zhang, Y., Song, J., Chen, Y., Wang, Y., Liu, S., 2022. Revisiting Dimensionality Reduction Techniques for Visual Cluster Analysis: An Empirical Study. IEEE Trans. Vis. Comput. Graphics 28 (1), 529–539. http://dx.doi.org/10.1109/TVCG.2021.3114694.

Xu, D., Tian, Y., 2015. A Comprehensive Survey of Clustering Algorithms. Ann. Data Sci. 2 (2), 165–193. http://dx.doi.org/10.1007/s40745-015-0040-1.

Xu, R., Wunsch, D., 2005. Survey of clustering algorithms. IEEE Trans. Neural Netw. 16 (3), 645–678. http://dx.doi.org/10.1109/TNN.2005.845141.

Xu, K., Xia, M., Mu, X., Wang, Y., Cao, N., 2019. EnsembleLens: Ensemble-based Visual Exploration of Anomaly Detection Algorithms with Multidimensional Data. IEEE Trans. Vis. Comput. Graphics 25 (1), 109–119. http://dx.doi.org/10.1109/TVCG.2018.2864825.

Zhang, J., Wang, Y., Molino, P., Li, L., Ebert, D.S., 2019. Manifold: A Model-Agnostic Framework for Interpretation and Diagnosis of Machine Learning Models. IEEE Trans. Vis. Comput. Graphics 25 (1), 364–373. http://dx.doi.org/10.1109/TVCG.2018.2864499.

Zhao, J., Chevalier, F., Pietriga, E., Balakrishnan, R., 2011. Exploratory Analysis of Time-Series with ChronoLenses. IEEE Trans. Vis. Comput. Graphics 17 (12), 2422–2431. http://dx.doi.org/10.1109/TVCG.2011.195.

Zhao, K., Ward, M.O., Rundensteiner, E.A., Higgins, H.N., 2014. LoVis: Local Pattern Visualization for Model Refinement. Comput. Graph. Forum 33 (3), 331–340. http://dx.doi.org/10.1111/cgf.12389.