Trelliscope: A System for Detailed Visualization in the Deep Analysis of Large Complex Data

Ryan Hafen, Luke Gosink, William S. Cleveland, Jason McDermott, Karin Rodland, and Kerstin Kleese-Van Dam

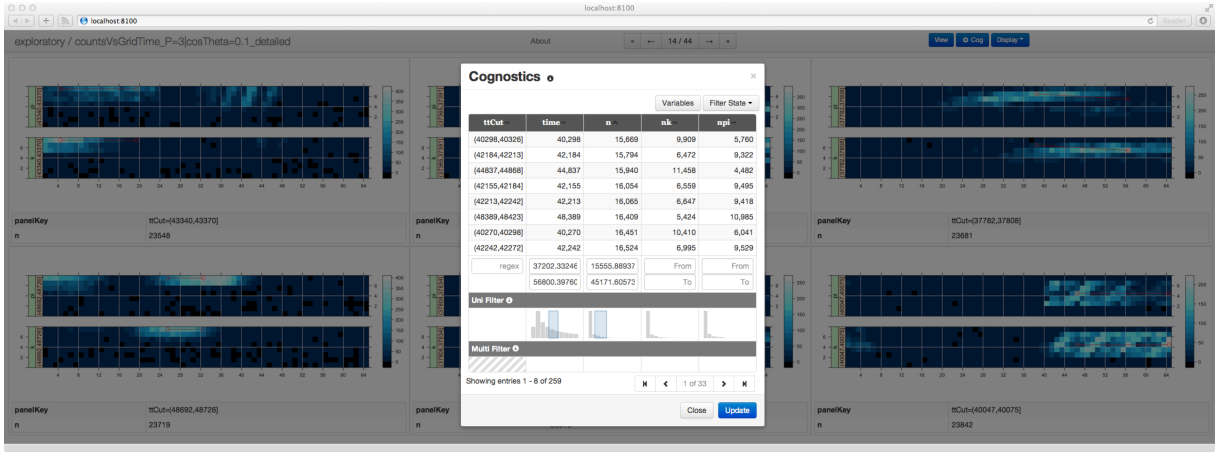

Fig. 1. Screenshot of the Trelliscope viewer showing a display created during analysis of high intensity physics data. In the configuration shown, 6 panels are displayed per screen, which are sorted and filtered according the parameters specified in the cognostics pane.

Abstract— Trelliscope emanates from the Trellis Display framework for visualization and the divide, and recombine (D&R) approach to large complex data. In Trellis, the data are broken up into subsets, a visualization method is applied to each subset, and the display result is an array of panels, one per subset. This is a powerful framework for visualization of data, both small and large. In D&R, the data are broken up into subsets, and any analytic method from statistics and machine learning is applied to each subset independently. Then the outputs are recombined. This provides not only a powerful framework for analysis, but also feasible and practical computations using distributed computational facilities. It enables deep analysis of the data: study of both data summaries as well as the detailed data at their finest granularity. This is critical to full understanding of the data. It also enables the analyst to program using the an interactive high-level language for data analysis such as R, which allows the analyst to focus more on the data and less on code. In this paper we introduce Trelliscope, a system that scales Trellis to large complex data. It provides a way to create, arrange, and interactively view panels with summaries or with detailed data. We discuss the underlying principles, design, and scalable architecture of Trelliscope, and illustrate its use on three analysis projects in proteomics, high intensity physics, and power systems engineering.

Index Terms—data analysis, data visualization, big data, statistical computing

1 Introduction

The Trellis visualization framework [2] provides an effective approach to detailed visualization of complex data. It has been widely used for data analysis as a way to meaningfully visualize both summary and detailed data. Currently, the most used implementation is the R package, Lattice [18].

In Trellis Display, the data are divided into subsets based on conditioning variables, and a plotting method is applied to each subset. The resulting plot for each subset is displayed on a "panel". The current Trellis renders the panels on a single display that takes the form of a three-dimensional array with rows, columns, and "pages". The whole process is controlled by a number of display methods that can be simply specified by the user. Implementations have provided a programming capability using the graphics language of the system that allows a user to specify almost any visualization method as the panel method. All of this enables a user to uncover the structure of data even when the structure is quite complicated. Trellis Display is discussed in Section 2.

Like all other analytic methods for data analysis, Trellis Display becomes difficult to use effectively for large complex data. There are two problems: (1) how to efficiently generate and organize a large number of panels; and (2) how to view and interact interact with many panels.

Trelliscope is an extension of Trellis that solves these problems. The extensions are described in Section 3. Problem (1) is solved by employing the Divide and Recombine (D&R) approach to large complex data, which is discussed in Section 3.1. Problem (2) is solved by making methods of sampling subsets a part of Trelliscope. There are a number of sampling methods that can be used; here we focus on "cognostics", described in Section 3.2. Then, Section 4 introduces the design and implementation of Trelliscope. Section 5 illustrates its usage in three applications.

- *Ryan Hafen, Luke Gosink, Jason McDermott, Karin Rodland, and Kerstin Kleese-Van Dam are with Pacific Northwest National Laboratory. Email: ryan.hafen@pnnl.gov*
- *William S. Cleveland is with Purdue University*

Manuscript received 31 March 2013; accepted 1 August 2013; posted online 13 October 2013; mailed on 27 September 2013.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

2 TRELLIS DISPLAY

Divide and Recombine (D&R), Trellis Display, and Trelliscope all have as a fundamental aspect a division of data into subsets, and then a recombination. We begin with an illustration of division and Trellis Display with a very small dataset.

2.1 An Example

Figure 2 is a Trellis Display of data from a 1930s experiment at the Minnesota Agricultural Experiment Station. At 6 sites, 10 varieties of barley were grown in each of 2 years. The data collected for the experiment are the yields for all combinations of the three factors — site, variety, and year — so there are 120 observations. In the figure, there are two conditioning variables: site and year. Site has 6 values and year has 2, so there are 12 combinations of site and year. The data are divided into 12 subsets by the 12 combinations. The visualization analytic method applied to each subset is a dotplot. Each panel of the display is this dotplot for one subset.

This Trellis display allows us to study the the dependence of yield on variety conditional and site and year, and how the relationship changes with the values of the conditioning variables. This turns out to be an exceptionally powerful mechanism for analysis of data, both small and large. Often, a number of Trellis displays are made with different conditioning variables; for example, we can plot yield and site given variety and year.

The barley data demonstrate the power of D&R. They were published by R. A. Fisher in his book, *Statistical Methods for Research Workers*, and through the decades became a canonical dataset used to illustrate many new numerical statistical methods. Despite the many re-analyses through time, it was not until the use of Trellis Display in their analysis 60 years later that a major error in the reported data was discovered [6]. This display gives a first clue, an anomaly at Morris.

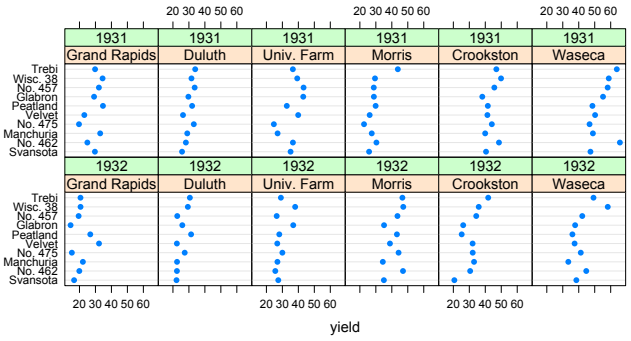

Fig. 2. A Trellis display of the barley data.

2.2 Trellis Display Methods

There are many visualization methods that are employed to enhance the effectiveness of Trellis Display. While these choices are largely up to the user to specify, a good Trellis implementation will provide mechanisms to specify and explore these choices with ease. More details on these and other considerations can be found in [11] and [2]. Here, we describe two choices to illustrate.

A major consideration when creating a Trellis Display is the scaling of an axis cross the panels, for example, the horizontal axis in Figure 2. The scales can be the *same*, as they are in the figure. This means we can compare across panels the overall level of the values on each panel, the amounts of variability of the values on each panel, and the patterns of the values on each panel independent of level and variability. This typically does not work if the panels have substantially different overall levels because the resolution for values on some or all panels can be too small. While in Figure 2 the overall level changes, it is not so large that resolution is adversely affected. In cases where it is affected, we can use *sliced*: the maximum and minimum on each scale change but we keep the number of units per cm are the same, so the maximum minus the minimum is the same across panels. Now we can judge overall level only by reading the tick-mark labels on each panel, but we can readily judge the amounts of variability and the patterns. This does not work either if the panel values across panels have substantially different amounts of variability. In this case, we can make the scales *free*: no coordination of scales across panels. Now we can only readily judge patterns independent of overall level and amount of variability.

Another critical consideration is the aspect ratio of the panels, which is the same for all. The aspect ratio, the height of a rectangle enclosing the data divided by the width, has an immense impact on our ability to judge the rate of change of one variable as function of another. A change in the aspect ratio changes our ability to visually decode slopes to judge rate of change. In Trellis, a value can be specified; in Figure 2 it is specified to be 1. Or an algorithm, banking to 45°, can be used to make the choice to enhance judgements of rate of change.

3 EXTENDING TRELLIS

Trelliscope extends Trellis Display to large complex data, enabling the efficient generation of displays with a very large number of panels that can be easily and effectively viewed. To achieve this we use the Divide and Recombine (D&R) approach to large complex data, R, Hadoop (or potentially other backends), and sampling of subsets to make the number of panels manageable.

3.1 Divide and Recombine

3.1.1 Introduction

Divide and Recombine (D&R) is a statistical approach to the analysis of large complex data [10]. The data are divided into subsets. Analytic methods are applied to each of the subsets, and the outputs of each method are recombined to form a result for the entire data. This provides computational feasibility can practicality.

There are two categories of analytic methods: statistical methods (including machine learning methods), whose output is numeric and categorical, and visualization methods, whose output is visual. For a visualization method, the recombination is a visual display that assembles the panels for viewing across subsets. For a statistical analytic method, the recombination results in numeric and categorical values. For example, suppose we carry out logistic linear regression on subsets. The outputs are the estimates of the regression coefficients, and statistical information about the estimates. The recombination can be unweighted means of the subset coefficient estimates, or means weighted by estimates of their variances.

The D&R division method and the recombination method used for an analytic method are critical to the success of the D&R result. We seek "best" division and recombination methods that serve as approximations that are as close as possible to what we could have gotten had we been able to apply the analytic method to all of the data directly. So division methods are not thought of as an arbitrary "chunking" of the data. Each subset of a division can be thought of as an experiment. We want each experiment to be as effective as possible. Doing so makes the D&R result as effective as possible.

3.1.2 Two Goals

One goal for D&R is deep analysis of large complex data: detailed, comprehensive analysis that does not lose important information in the data. Deep analysis requires visualization of the data at their finest granularity, not just visualization of summaries of the data. This was established in the 1960s by the pioneers of data visualization for model building such as John Tukey, Frank Anscombe, Cuthbert Daniel, and George Box. We can do this with small data. For large complex data, should we wave the white flag, surrendering to the size and complexity, and visualize just summaries? Would that not be a step backward? Large complex data should not get a pass.

A second goal is the analyst being able to use a well-designed interactive language for data analysis, for example R. This can reduce substantially the time an analyst spends programming with the data

compared with using a lower-level language. There is a very clear metric: the fraction of the time the analyst spends thinking about the programming versus the fraction of time spent thinking about the data. We seek to minimize the former. The language also needs to be powerful, allowing a tailoring of the analysis to the data. The language needs to provide access to the 1000s of statistical and analytic methods that exist today. Often, packages for these methods are written in C to provide fast computation, but the data analyst needs to call them them from the interactive language. R [16], does a very good job in efficiency for the analyst, programming power, and accessing methods. This is why it won the 1998 ACM Software System Award, by far the most prestigious software award with other winners like Unix, the World Wide Web, Visicalc, and Postscript.

3.1.3 The Computational Environment

What makes D&R work is that the computations for the application of an analytic method to the subsets can be run in parallel with no communication among them. D&R can exploit parallel distributed computational environments like Hadoop with its MapReduce compute engine and Hadoop distributed file system. D&R makes it feasible for a data analyst to apply almost any statistical or visualization method to large complex data.

Of course, the data analyst does not want to program the distributed computations in Hadoop's native language, Java. RHIPE, the R and Hadoop Integrated Programming Environment, enables the analyst to write Hadoop MapReduce jobs completely in R [9]. RHIPE has two parts. A core, written in R and Java, that communicates with Hadoop. The second part is a prototype domain-specific language for D&R in R, available as a package named `datadr` [13]. This software provides a simple interface for specifying division and recombination methods. The `datadr` methods provide a layer on top of RHIPE, allowing the analyst to only think in terms of meaningful division and recombination methods, hiding the details of MapReduce from the user. Also, `datadr` has been designed with the goal in mind of linking to other computing backends.

Trelliscope has been implemented in the the D&R computational environment of R and `datadr`. There are a number of potential backends. Thus far we have used RHIPE/Hadoop, which provides for scaling to large complex data.

3.2 Sampling

A natural approach to viewing many panels is simply scrolling through all pages of panels tiled across a screen. This alone can be very powerful, but can become an unrealistic approach when the number of panels is large. In our own applications, the numbers of subsets range from thousands to millions. We use statistical sampling methods to choose a sample of subsets to visualize. The sampling is a data reduction, but one that is systematic and rigorous. It is guided by sampling variables, each with one value per subset. We have used two methods of sampling: *representative* and *cognostic*.

A representative sample covers the joint region of values of a set of sampling variables. In most cases the variables are known to have an effect, and we seek to study their effect within the region of the multidimensional space that they collectively occupy. For example, suppose 10^6 subsets have variable sample sizes n. Suppose from each subset we get an estimate of a parameter, and 5000 bootstrap values of the estimate to describe the sampling distribution of the estimator for the subset. Suppose we want to study the normality of the bootstrap distribution for each subset by a normal quantile plot, and how it changes with n. Of course our expectation is that the estimator gets closer to normal as n increases. We could check normality by making normal quantile plots for 1000 subsets whose values of n on the log scale are as close to equal as possible.

Cognostics is a very general notion of Tukey. We describe it for D&R subset sampling. We visualize subsets for which the values of cognostic sampling variables are in certain regions of their multidimensional space. These become subsets of interest. Tukey described the problem of having too many potential visualizations of the data as "drown[ing] in a sea of many different displays" [5]. To deal with this, he put forth the idea of computing diagnostic variables that judge the relative interest or importance of applying a visualization method to each subset. In his words, "Since these are intended for computer consumption, not human consumption, it seems natural to call such computer guiding diagnostics 'cognostics'" [5].

As with representative sampling, cognostic sampling variables can be known to have an effect, and we seek to study their effect in a focused region. They can also be variables that measure statistical behavior. For example, if we fit a logistic regression model to each subset, we might take a cognostic variable to be a measure of how well the model fits the data, and sample subsets for which the variable is very large.

In many cases, we have found that what is most powerful is a combination of the values of sampling variables together with what the analyst sees in visualizing samples. Cognostic sampling provides a way to look at interesting subsets of data, providing a new mode of interaction where the user is able to sort and filter the panels of the display based on the variables. This takes the analyst to a new level of interactive exploration of the data. We will illustrate this in Section 5.3.

4 Design and Implementation of Trelliscope

The Trellis extension concepts introduced in the previous section are quite straightforward – the idea is to use a distributed computing interface to create the panels and leverage cognostics and sampling to interactively view the panels. However, inserting these ideas into an implementation that provides meaningful interactivity and an efficient workflow for a data analyst, while being interoperable with a distributed storage and computing backend, is not trivial.

We have created an R implementation of the Trellis extensions, called Trelliscope. As a system, Trelliscope builds upon D&R by providing methods to apply plot and cognostic functions to divided datasets to create displays with many panels, providing a web-based viewer that allows users to interact with the panels, all with minimal effort. For example, users can sort and filter millions of panels using a wide variety of cognostics values. The filtered panels are tiled on a screen where the user can then rapidly page through them.

To support this at-scale flexibility for visualization, Trelliscope takes as input a dataset divided into subsets using the `divide()` method from the `datadr` D&R package [13]. This data can either be a divided local data frame or a distributed divided dataset on the Hadoop distributed file system (HDFS). A *prepanel* function can be applied to the data to determine axis limits for each panel. If limits are not specified, scales are chosen such that the data fills the panel. The analyst then specifies a plot function and a cognostics function to be applied to each subset and sends these to a command `trsPlot()`. This function applies the plot and cognostics functions to each subset and stores the results, along with meta data about the display. Trelliscope employs R as a visualization system to create the plots and compute the cognostics. By using the R-based interfaces of base R graphics, lattice, or ggplot [21, 18], it is possible to create nearly any visual representation that a user requires. There are several options for how plots and cognostics are stored, which are discussed in greater detail later in Sections 4.1.1 and 4.1.2.

4.1 Hardware Architecture

Trelliscope can work on a single workstation operating on in-memory data frames in R. When dealing with very large data sets, Trelliscope can interface with scalable, distributed hardware architectures. For example, Figure 3 shows how Trelliscope can operate seamlessly with very large datasets stored on HDFS. Our analysis environment has been set up in a way similar to that of Figure 3, except that the web server and cognostics server are both running on the master node of the Hadoop cluster.

4.1.1 Panel Storage

We have investigated and implemented several options for panel storage. There are three possibilities for pre-generating and storing raster images. These are: 1) images stored on disk, 2) images stored on

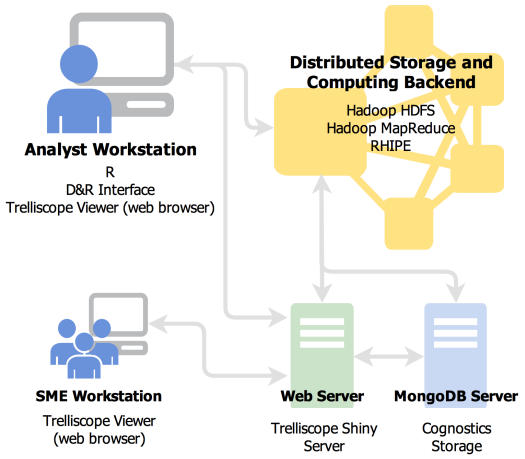

Fig. 3. Distributed architecture of Trelliscope. Trelliscope is capable of running on a single workstation, but has the ability to run in a distributed architecture, one possibility of which is shown here. All analysis work is done on the analyst's workstation in R, and the D&R interface and Trelliscope R package coordinate communication with the other services. Other users, such as subject-matter experts (SMEs) can access the viewer through a web browser.

HDFS, 3) images stored in MongoDB. It is also possible to specify that the images be rendered on-the-fly. For this option, the user can specify for the data to either be stored and retrieved from local disk or from HDFS. In this case, pointers to each subset on disk are stored in the image's place and the images are rendered on-the-fly by reading the associated data when requested by the viewer.

The advantage of rendering on-the-fly is much quicker initial creation of a display (only cognostics are computed), as well as the ability to dynamically modify the panel function from within the viewer. This particularly makes sense when there are millions of subsets and the probability of viewing all subsets is very low, in which case there is not much sense in precomputing all of them. It typically is not expensive in terms of compute time or storage space to compute cognostics for every subset, even when the number of subsets is in the millions. The only possible downside to the render-on-the-fly approach is the case when the plot function is very expensive to apply. In that case, it can be worth the time expense to pre-generate the plots so that time isn't spent on rendering during the viewing process.

In our implementation, the viewer knows how to link to a panel image or data through a special cognostic automatically computed for each display called `panelKey`. Panel images or data can be uniquely located through the combination of the display name and panel key. Panel data or images stored to HDFS are stored as key/value pair Hadoop mapfiles [20] with the key being the panel key. The images or data can then be rapidly retrieved by key when the viewer requests them. For panel images stored on disk, the files are named according to panel key and are retrievable by name.

4.1.2 Cognostics Storage

Cognostics for a display can be stored either as an R data frame on disk or as a collection in a database. The R data frame storage approach is the simplest and does not require additional hardware or software to be available aside from R on a workstation. The potential drawback to this approach is that it does not scale well beyond a few million subsets as the viewer must load the entire cognostics data frame into memory to show the display.

Storing the cognostics in a database is another option that provides several benefits over a data frame approach. Most databases will provide the ability to index columns of the database for fast filtering, while not being required to be loaded into memory on the web server. We have chosen MongoDB as a database option for storing cognostics in our system. MongoDB scales very easily and has a flexible aggregation framework that we plan to leverage to perform different viewer-directed transformations and aggregations of cognostics.

4.2 Creating a Display in R

Creating a display from an R instance is as simple as calling the `trsPlot()` function and specifying 1) a divided input dataset, 2) a name for the display, and 3) a plot function for the panels. Other optional settings (e.g., display group, axis limits, and a cognostics function) can be specified as well. With respect to data types, `trsPlot()` can handle different classes of divided data objects (e.g. local data frame or HDFS) and there are different parameter defaults for how plots and cognostics are stored based on the input type. Detailed documentation about all of the options with examples is provided in the online documentation [14].

An example of creating a display could look like this:

```
trsPlot(
  data = gridCounts,
  group = "exploratory",
  name = "gridCounts",
  desc = "Number of grid hits over time",
  lims = list(x="free", y="free"),
  plotFn = gcPlot,
  cogFn = gcCog,
  plotDim = list(height = 200, width = 800),
  storage = "hdfsData"
)
```

Here, `gridCounts` is a divided HDFS data object. We have created and tested a plot function `gcPlot` and a cognostics function `gcCog` that we are specifying to be applied to the subsets of this data. We specify the x and y axis limits to be "free". The storage option chosen is `"hdfsData"`, meaning that only cognostics will be computed which will include a `panelKey` that will be used to retrieve data from the `gridCounts` data when viewing the display

Prior to calling `trsPlot()`, there are functions that aid the analyst in specifying the panel axis limits. A `trsPrepanel()` function computes the axis limits for each subset and returns them. The analyst can visualize the limits with a special R plot method to get a feel for the distribution of the axis limit values. This can be helpful in discovering subsets with large outlying values which are sure to abound when the data is very large, and the analyst can specify to trim outliers so that they do not inflate the axis limits of all panels.

4.3 The Trelliscope Viewer

The Trelliscope viewer is a web application written in R using the R Shiny [17] package. The server for the viewer can either run on the user's workstation or on a web server. The viewer is accessed through a web browser. Shiny provides communication from the web browser to a live R session on the web server through a WebSocket connection, providing an interactive user interface controlled by JavaScript.

4.3.1 Panel View

The panel view is the main component of the viewer; Figure 4 shows the panel view for a display created during the analysis of high intensity physics data where the user has specified to view six panels per screen. Once the user has selected a display to view, the panel view tiles the panels of the chosen display across the screen. Left and right keys or the navigation buttons on the header bar provide simple navigation through the collection of panels.

By clicking the "view" button, the analyst can specify how many panels are to be viewed in a single screen and can also specify whether cognostic values should be displayed alongside each panel. There is also the "view" option to edit the panel plotting function for displays that are not pre-rendered. This ability to dynamically change what is being plotted is an additional benefit to not pre-rendering the plots. The user can react to what is being seen in the panels directly through the viewer (e.g. adding a reference line) without going back to R to recreate the entire display. An example screenshot of the view pane is shown in Figure 5.

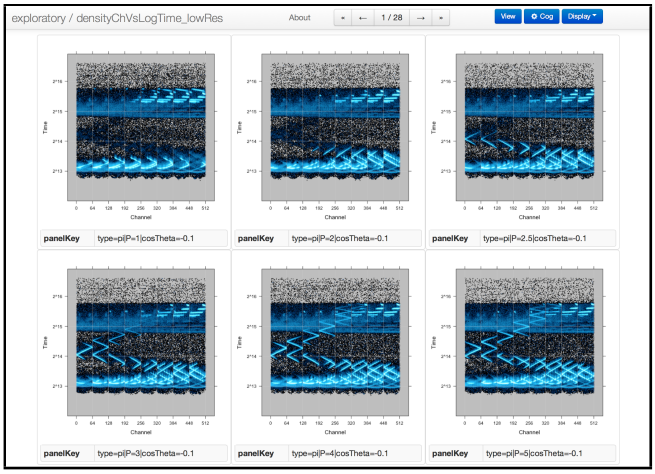

Fig. 4. Screenshot of the panel view of the Trelliscope viewer for a display created during the analysis of high intensity physics data.

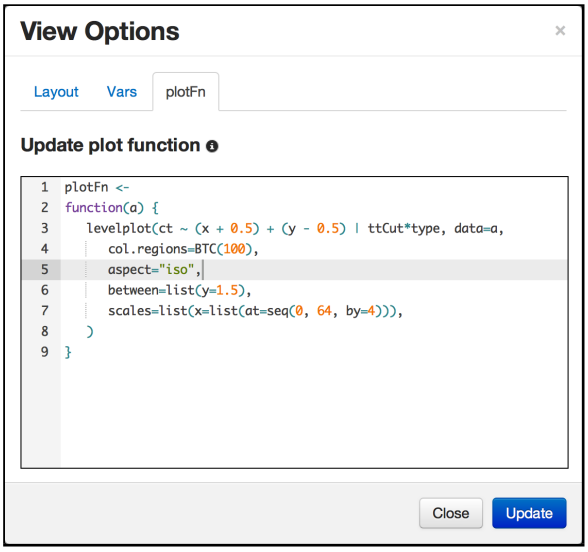

Fig. 5. Screenshot of view pane of the Trelliscope viewer. The panel plot editing tab is visible.

4.3.2 Cognostics Interaction

A major feature of the Trelliscope viewer is the ability to sort and filter panels according to the cognostics. Clicking the "Cog" button brings up an interactive table that displays the cognostics variables. Clicking column headers in the cognostics panel allows for sorting, with multi-column sorting available with shift-click. In addition to sorting, there are filters available for each variable. Regular expressions can be used to filter qualitative variables and numerical ranges can be used to filter quantitative variables. A screenshot of the cognostics pane for a display is shown in Figure 6. This figure depicts an example of filtering and sorting where operations done in the cognostics pane determine how the panels will be displayed back in the panel view.

The cognostics pane also allows for interactive visual filtering of cognostics for quantitative variables. Visual filtering is useful in it can help the user identify regions of interest in the cognostics space using the distribution of the cognostic as a visual cue. The visual filters are created using d3.js [3].

For univariate filtering, the user can click the histograms at the bottom of the cognostics table (see Figure 6). Clicking in this way activates the histogram so that users can select a range of the histogram's values; a screenshot of this type of selection is shown in Figure 7.

Multivariate filtering is also possible. In the cognostics pane's

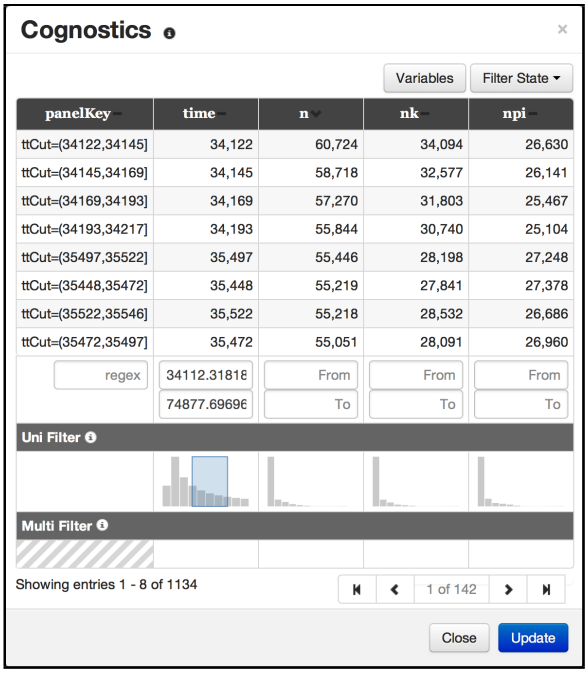

Fig. 6. Screenshot of the cognostics pane in the Trelliscope viewer. In addition to `panelKey`, 4 cognostics variables are visible. The `time` cognostic has been range filtered and the filtered cognostics are sorted in reverse order by the cognostic variable `n`.

"Multi Filter" section, the user can select 2 variables and bring up an interactive scatterplot from which a identify bivariate range of values can be identified to filter the cognostics on. If there are more than 5,000 cognostics, a hexagonal binning [4] is computed and plotted instead of individual points. In the future, the viewer will support more than 2 variables by using projection pursuit to project the multivariate data onto 2 a two-dimensional plane which the user can then interact with. An example of bivariate filtering is shown in Figure 8.

4.3.3 Viewing related displays

It is very common to create multiple displays for the same division of the data. When this is the case, it is useful to view multiple displays side-by-side for each subset. Trelliscope provides a way to do this by keeping track of which displays were created from the same division. By selecting the "Displays" button and selecting "Related Displays" from the menu, the user can select any number of related displays to be shown in the viewer.

5 Applications

We have employed the principles behind Trelliscope in many projects [12, 15, 22]. Here, we discuss three recent projects that have made direct use of the Trelliscope implementation. The main goal of this paper is to introduce the Trelliscope methodology and implementation, and therefore we are not able to go into details for these applications. Our aim is to briefly illustrate the use of Trelliscope in real-world data analysis projects and demonstrate a few use cases where it was an integral part of deep analysis.

5.1 Power Systems Data Cleaning and Event Detection

Sensors called Phasor Measurement Units (PMUs) provide power grid operators with the status of the electric power grid at a very high frequency, 30-60 times per second. Our analysis covered a data set of 1.5 years of data from 38 PMUs, a data set 2TB in size. The data consists of about 1.5 billion time points with measurements from 38 PMU locations, where each PMU records on average 14 measurements. These measurements include the measured frequency at which the grid is operating as well as several phase angle measurements.

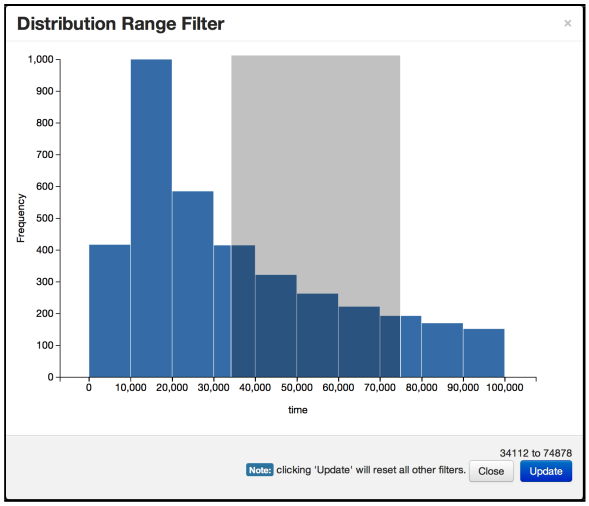

Fig. 7. Screenshot of the interactive univariate histogram cognostic filter in the Trelliscope viewer.

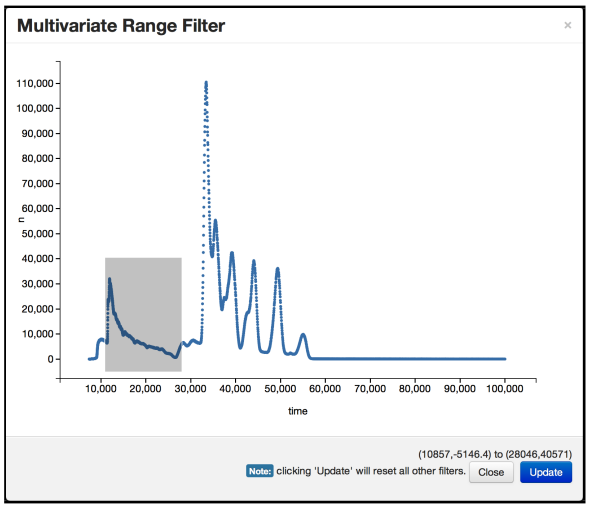

Fig. 8. Screenshot of the interactive bivariate cognostic filter in the Trelliscope viewer.

The initial goal for our analysis of this data was to build event detection algorithms for specific grid activities of interest such as generator trips and grid islanding. Our initial inquiry for the data was simply to understand the behavior of each series over time. As all behaviors of interest are time-local, we chose a subsetting scheme that was short enough to be able to process and visualize at high enough time resolution, but long enough to be sure to capture the time-local behavior within subsets. Based on these requirements, we settled on 5 minute time intervals for subsetting. There are about 158,000 subsets of the data, and when additionally splitting out by location, there are about 6 million subsets.

We investigated the data using Trelliscope to create and view time series plots of the subsets. Trelliscope provided a way for us to have immediately at our fingertips a detailed view of any time sequence of the 2TB dataset. We developed a list of cognostics for the subsets that highlighted different behaviors such as the variability of the data around a local polynomial regression fit of the data, number of missing values, length of repeated sequences, etc. This list of cognostics was built iteratively – sorting and filtering on one cognostic variable would typically bring subsets to our attention where other interesting behavior was occurring. Some of these behaviors were so rare that it is doubtful that they would have been discovered without the ability to interactively look at the data at the greatest level of detail. A large proportion of the behaviors we uncovered were deemed to be erroneous records. The combination of Trelliscope for detailed views with the analytical recombination methods of D&R enabled us to identify, validate, and build algorithms to remove these records from the data to focus on the event detection. Many of the types of erroneous records had gone unnoticed in several prior analyses where tools like D&R and Trelliscope were not available.

Another notable use-case of Trelliscope was in fine-tuning our generator trip algorithm. Based on examples of known trips, we were able to identify characteristics of a generator trip: all frequency measurements experience a sharp sudden drop followed by a gradual recovery. We constructed features from the data that captured these types of behaviors, but were not certain at which point the combination of feature metrics constituted a bona fide generator trip. We used the metrics to procure a candidate set of over 3000 time periods in which we thought a generator trip might have occurred. This data constituted a new division with each subset containing the data corresponding to the suspected generator trip. We created a Trelliscope time series display of the data and presented them to a domain expert to view and classify using the Trelliscope viewer. Classification narrowed the list down to about 500 verified trips and from this narrowed list we were able to create a rules-based algorithm for detecting generator trips. A screenshot of the Trelliscope viewer being used for this purpose is shown in Figure 5.1.

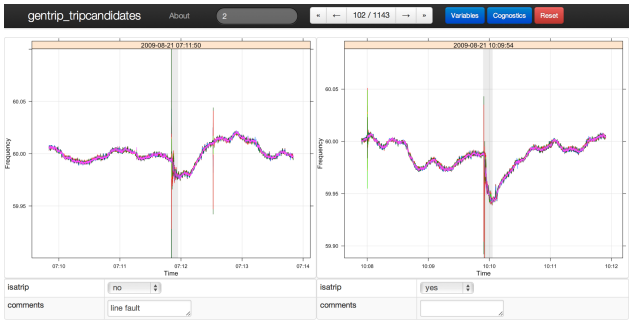

Fig. 9. Trelliscope viewer being used to view and verify generator trips in a power systems engineering application.

5.2 High Intensity Physics Particle Classification

The discovery of New Physics (NP) is key to extend the Standard Model (SM) of particle physics. This model is a theory that allows scientists to make verifiable predictions about the nature of quantum causality. One of the key aspects to generating NP is to correctly verify the presence and behavior of specific sub-atomic particles during a high intensity physics experiment. In this application, we are specifically looking at detecting and differentiating between two sub-atomic particles, pion and kaon, that are present during a single 100ns event from simulated data that models a detector in the Belle II experiment [8]. Within a single event, the numbers of these particles, their ratio to each other, and the energy distribution of each particle type is key to identifying the SM inconsistencies that lead to NP.

This is an ongoing project, but our initial task, in anticipation of real data becoming available in the coming years, is to see if it is feasible to do kaon / pion classification from simulated data. We are able to stochastically simulate data from a variety of possible scenarios. From a high-level, the scenario of interest begins when a kaon or pion particle hits the exterior of the detector and releases a small array of photons into one of the detector's quartz chamber (see Figure 5.2). The particles themselves enter the detector at a known momentum, P, and propagate photons radially at an unknown angle, θ. The particles bounce around the detector until they hit a sensor grid defined by 64×8 channels (labeled $1, \ldots, 512$. We have simulated hundreds of thousands of events from 168 combinations of P, θ, and particle type (kaon/pion). The simulation reports the hit time and location on

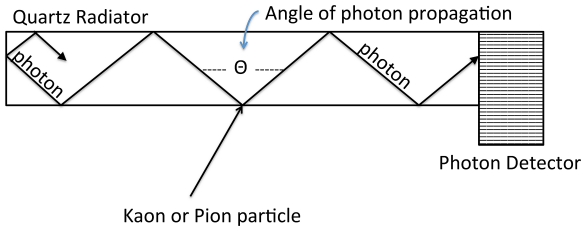

Fig. 10. Illustration of a detector in the Belle II experiment. Particles hit the exterior of the detector and release an array of photons that disperse an a fixed angle, θ, based on the particles mass and momentum, P. Photons bounce around the quartz chamber until they are detected by the photon detector.

the sensor. The complete realizations we are currently studying generate about 12GB of data but we will soon be looking at much larger collections of events. This is not nearly as much data as in our previous example, but still enough that we cannot tackle this data with traditional approaches.

In our first step, we split the data into subsets for each combination of P, θ, and particle type in order to study the distribution of hit times and locations on the sensor grid according to channel. Then we create a hexagonal binned scatterplot of time vs. channel. An example of this for pions with $\cos(\theta) = 0.1$ and P ranging from 1 to 5 (left to right, top to bottom) is seen in Figure 4. The next step relies on iterations of exploratory data analysis that are facilitated by stepping through Treliscope's side-by-side panel views of kaon and pion plots for fixed P and θ. During this task, we noticed that for small P, it is easy to visually distinguish kaons and pions, but as P increases (from $P \geq 3$), the two begin to look very similar. Big P is where the scientific interest is.

To further investigate the kaon / pion behavior, we partitioned the data by time. More specifically, we choose one set of parameters, $P = 3$ and $\cos(\theta) = 0.1$ to study and contrast kaon / pion behavior in detail. An initial look at this case over time led us to desire a finer time resolution and hence more sample runs. We generated 1 million sample runs for this case and divided the data up into 3739 subsets according to equal-width log time intervals. The display method for this data is a heat map of the hit distribution within each time chunk for each of kaon and pion, plotted according to the actual grid layout of the sensor. An example of this display can be seen in the background of Figure 1. Viewing the data in this way helped us identify time periods of a clear signal moving back and forth across the sensor. The activity comes in bursts, and the signal is only present when the activity is high (see Figure 8 for the corresponding cognostics). Viewing this display also gave us a better view of the differences between kaon and pion and we were able to see many times where the signal was visibly very different between the two. Based on our observations, we constructed a simple Bayes classifier for any new sample. The sample hits are matched to the time partitions of our time-divided data, and then based on the distribution of the training data in that time period, we compute the probability that the point is a kaon or pion given its hit location on the grid. We found that if we weight these classifications based on the time regions where the difference between kaon and pion is most pronounced, we can get accurate classification in the 90% range. While here we have omitted several of the iterations and steps in this process, Trelliscope and the `datadr` package played a crucial role in doing this analysis effectively in a very short time.

5.3 Proteomics Biomarker Discovery

The goal of our proteomics research project is to extract meaningful signatures of biological processes from peptide and protein abundance data. The process of biomarker discovery and characterization provides opportunities both for purely statistical and expert knowledge-based approaches. In this project we investigated the use of Trelliscope in an effort to improve integration of the two.

We used proteomics data from a mouse model of chronic obstructive pulmonary disease (COPD) that involves deletion of a gene, ADA. Animals with ADA+/- are controls (no disease) ADA+/+ are diseased. We analyzed data from plasma samples of control and disease groups over time to study biomarkers for COPD. The abundance data underwent an extensive preprocessing procedure, resulting in data for 2,927 peptides which were rolled up into 414 proteins. Statistical and functional clustering tools were applied to the data, forming potential groups of biomarkers.

We then created a visual representation of the data in Trelliscope to present to a domain expert. We divided the protein data into subsets by protein and likewise for the peptide data. The panel plotting function for each protein or peptide displays the abundance over time by disease and control groups. An example of two panels from the protein display is shown in Figure 5.3. The dots represent the abundances for each subject, while the crosshairs and lines denote the mean abundance at each time point for each group and its progression over time. The top and bottom sets of boxes colored in gray and green indicate whether a statistically significant difference exists between the two groups at each time point, the top row for a G-test (test for presence/absence) and the bottom for a t-test for difference between group means [19]. A statistically significant positive difference between control and disease for the given test at the given time point, gray means insignificant, and red means a statistically significant negative difference. These boxes helped create a visual cue that can be rapidly digested while paging through many panels. These panels illustrate the ability to create very specific graphics in R in a simple and flexible way.

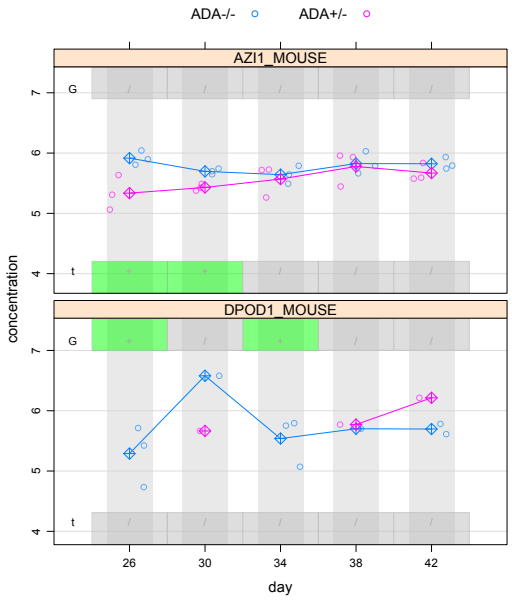

Fig. 11. Two panels from a display of protein abundance vs. time for disease (ADA+/-) and control (ADA-/-) groups.

We computed several cognostics for each panel. Some simple statistical cognostics for each panel include the following: number of total, ADA-/-, and ADA+/- observations in the panel, average difference between ADA+/- and ADA-/- before and after day 34, t-test and g-test p-values on each day, slope of ADA-/- and ADA+/- abundance over time. Domain-specific cognostics for each panel include protein annotation and function, cluster assignment from hierarchical clustering, enriched functional cluster assignment and gene functional classification from DAVID [7], and assigned generalized biological process from GO [1].

Using these cognsotics, data analysts and domain experts were able to interact with the panels in many ways. Panels were sorted and filtered on proteins of interest, protein functions of interest, etc. to help the domain expert understand what the data had to say. Panels were also sorted by statistical significance trends of interest, such as

abundances that are not significantly different in the beginning and become significant over time. Another interaction was filtering the panels based on the clustering results and determining if the resulting protein groupings made any sense, as well as trying to determine if there was anything apparent in the data that would explain and provide some interpretability of the protein groupings selected.

This analysis facilitated by Trelliscope provided valuable insight into the data for the domain experts in several key aspects. Visualization of the peptides (portions of the whole protein) contributing to quantitation of key proteins was evaluated. In several cases it was apparent that there were two trends in the peptides mapping to one protein indicating that the mapping was incorrect. Grouping of the proteins into functional groups was used to identify subgroups for incorporation into a classification algorithm that allowed identification of improved biosignatures of disease in the COPD samples.

The data for this example is by far the smallest (less than 10MB) and illustrates the utility of Trelliscope when the data is not large. Detailed views of the data still required panels for hundreds or thousands of subsets and thus benefitted from the interactivity of Trelliscope.

6 Discussion and Future Work

Trelliscope provides a framework for scalable detailed, interactive visualization of large complex data. It has been developed out of necessity from analysis of many real-world large complex datasets and has proven to be immensely useful.

Trelliscope is a research effort that to this point has been mainly focused on the architecture and practicality of a scalable Trellis Display system. In the future, we would like to give much more attention to the design and features of the viewer.

The main focus of future work on the viewer is based around sampling and cognostics. We envision an interface that allows the analyst to specify different sampling schemes of the panels. We also anticipate new cognostic types that provide new ways to interact with the data. One of these is relational cognostics, where the cognostic for one panel is a reference to another panel or panels either in the same display or in other displays. Such cognostics could be displayed and interacted with in the viewer through a network graph.

We also envision more options with how the quantitative and qualitative cognostics are interacted with. This would include more complex visual interactions with multiple cognostic variables.

We have given consideration to the idea of using more modern web-based technologies to create and render the panel images. This could bring a new level of interactivity. However, we feel there is sufficient power (and ease of use) for the analyst by allowing them to stick to tools they are familiar with to create the panel-level plots while still being able to have rich interactions with these plots at the cognostics level.

Acknowledgments

This work was supported in part by the Pacific Northwest National Laboratory Signature Discovery Initiative, Future Power Grid Initiative, and a grant "D&R for Large Complex Data" from the DARPA XDATA Program.

References

[1] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, et al. Gene ontology: tool for the unification of biology. *Nature genetics*, 25(1):25–29, 2000.

[2] R. A. Becker, W. S. Cleveland, and M.-J. Shyu. The visual design and control of trellis display. *Journal of Computational and Graphical Statistics*, 5(2):123–155, 1996.

[3] M. Bostock, V. Ogievetsky, and J. Heer. D^3 data-driven documents. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2301–2309, 2011.

[4] D. B. Carr, R. J. Littlefield, W. Nicholson, and J. Littlefield. Scatterplot matrix techniques for large n. *Journal of the American Statistical Association*, 82(398):424–436, 1987.

[5] W. Cleveland. *The Collected Works of John W. Tukey: Graphics 1965-1985*, volume 5. Chapman & Hall/CRC, 1988.

[6] W. S. Cleveland. *Visualizing Data*. Hobart Press, Chicago, 1993.

[7] G. Dennis Jr, B. T. Sherman, D. A. Hosack, J. Yang, W. Gao, H. C. Lane, R. A. Lempicki, et al. David: database for annotation, visualization, and integrated discovery. *Genome Biol*, 4(5):P3, 2003.

[8] T. et al. Belle II Technical Design Report, 2010.

[9] S. Guha. *Computing Environment for the Statistical Analysis of Large and Complex Data*. PhD thesis, Purdue University Department of Statistics, 2010.

[10] S. Guha, R. Hafen, J. Rounds, J. Xia, J. Li, B. Xi, and W. S. Cleveland. Large complex data: divide and recombine (d&r) with rhipe. *Stat*, 1(1):53–67, 2012.

[11] S. Guha, R. P. Hafen, P. Kidwell, and W. S.Cleveland. Visualization Databases for the Analysis of Large Complex Datasets. *Journal of Machine Learning Research*, 5:193–200, 2009.

[12] S. Guha, P. Kidwell, A. Barthur, W. S. Cleveland, J. Gerth, and C. Bullard. A Streaming Statistical Algorithm for Detection of SSH Keystroke Packets in TCP Connections. In R. K. Wood and R. F. Dell, editors, *Operations Research, Computing, and Homeland Defense*. Institute for Operations Research and Management Sciences, 2011.

[13] R. Hafen. "datadr" github documentation. http://hafen.github.io/datadr/. Accessed: 2013-05-11.

[14] R. Hafen. "trelliscope" github documentation. http://hafen.github.io/trelliscope/. Accessed: 2013-05-11.

[15] R. P. Hafen, D. E. Anderson, W. S. Cleveland, R. Maciejewski, D. S. Ebert, A. Abusalah, M. Yakout, M. Ouzzani, and S. Grannis. Syndromic Surveillance: STL for Modeling, Visualizing, and Monitoring Disease Counts. *BMC Medical Informatics and Decision Making*, 9:21:doi:10.1186/1472–6947–9–21, 2009.

[16] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2012. ISBN 3-900051-07-0.

[17] RStudio and Inc. *shiny: Web Application Framework for R*, 2013. R package version 0.5.0.

[18] D. Sarkar. *Lattice: multivariate data visualization with R*. Springer Verlag, 2008.

[19] B.-J. M. Webb-Robertson, L. A. McCue, K. M. Waters, M. M. Matzke, J. M. Jacobs, T. O. Metz, S. M. Varnum, and J. G. Pounds. Combined statistical analyses of peptide intensities and peptide occurrences improves identification of significant peptides from ms-based proteomics data. *Journal of proteome research*, 9(11):5748–5756, 2010.

[20] T. White. *Hadoop: The definitive guide*. O'Reilly Media, Inc., 2012.

[21] H. Wickham. *ggplot2: elegant graphics for data analysis*. Springer Publishing Company, Incorporated, 2009.

[22] B. Xi, H. Chen, W. S. Cleveland, and T. Telkamp. Statistical Analysis and Modeling of Internet VoIP Traffic for Network Engineering. *Electronic Journal of Statistics*, 4:58–116, 2010.