Time Oriented Data Visualization Library for Data Scientists

Peter Bak, *Member, IEEE*, and Avi Yaeli, and Inna Skarbovsky, and Jonathan Bnayahu

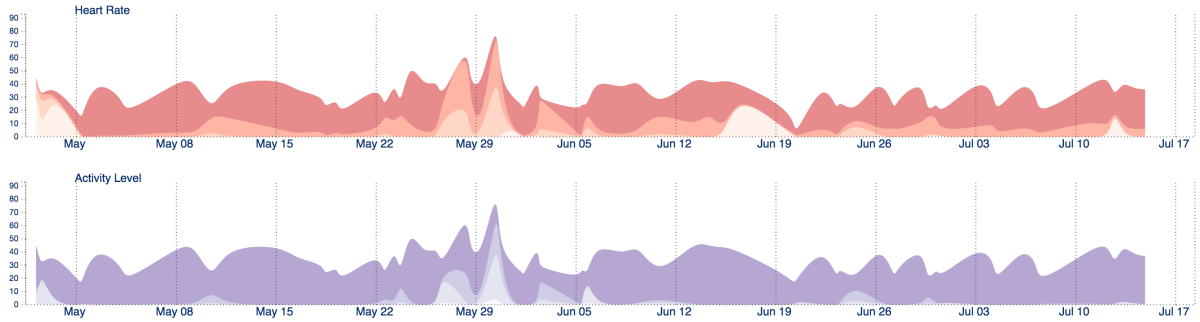

Fig. 1. Time-series visualization using a dedicated library for data scientists.

Abstract—The current paper demonstrates the use and design of time-oriented data visualization library for data scientists working in a Notebook environment. We provide visual interaction of event pattern for exploration, as exemplified on Fitbit data in Fig. 1.

1 Motivation

Data science is emerging from being a purely statistician led work to domain experts having deeper and broader understanding of the subject matter, but requiring more consumable environments to access data and analytic models. Visual analytics is one methodology that provides a solution for domain experts. It enables to use visualization as an interface between the user and the analytic algorithms, and provides a feedback by displaying the analytic results. The current works demonstrates the design and use of interactive visualization in data science environment, such as iPython Notebook, in a highly consumable environment. The contributions of this demonstration are in creating a dedicated library for data scientists - as opposed to visualization experts or programmers, and in accommodating for different properties of temporal data, rather than a general purpose visualization framework.

2 Architecture Design

There are an increasing number of visualization libraries for different environments, or even cross-environment that aim to provide programmers or visualization experts the tools to create representations of data. Many of these also support direct manipulation of properties of the display selection and filtering of the data itself. Just to emphasize some the existing solution, we mention here plot.ly (plot.ly/), shiny (shiny.rstudio.com/) and MPLD3 (mpld3.github.io/) among the many.

Our design has a different strategy. The primary goal is to shorten the time from data to insight, through visualization in one particular domain, without requiring knowledge in programming or in visualization per se. Our focus has been to support common visualization tasks in health-care in particular. What characterizes this domain is the longitudinal (such as electronic health care records) and sensor data from medical devices and wearables. Users often aim at discovering event patterns of particular sequence, or temporal alignment relative to common events and index dates. Supporting these tasks from a visualization perspective requires a rich set of visual layouts, and interaction techniques. In order to achieve this, we only use one configuration API to define and parameterize the visualization, which can be adopted to a specific domain language,.

The most inner layer of our architecture is a javascript library, which wraps a highly configurable D3 charts and provides automatic computation of all parameters, when these are not provided by the user. This library can be used from a web programming environment on its own, if required. The benefit of this approach is that it allows us easily integrate this implementation into any programming language and environment that allows HTML display.

We initially made this charting library available in IPython. We provide a thin Python client library as a package that the user can install in her/his environment and import into the notebook. The Python library leverages IPython rich display system to dynamically construct an HTML that contains both the JSON data and configuration, and in return calls the javascript library to produce the visualization. This approach can be extended easily to other Jupyter kernel languages, for example leveraging IRDisplay in R.

The chart configuration includes the date-time attribute shared among all the data-records, This variable is converted into a canonical representation supporting point and interval events. Uses can also configure the identifiers of events, derived events, and references thereof to originating events. A setting parameter of the configuration aims at exactly how these event types should be visualized. The user can set the marker and channels of the view, as proposed by Tamara Munzner's nested model for information visualization [2]. Marks

- *Peter Bak is with IBM Watson Health Technologies. E-mail: peter.bak@il.ibm.com*
- *Avi Yaeli is with IBM Watson Health Technologies. E-mail: aviy@il.ibm.com*
- *Inna Skarbovsky is with IBM Research, E-mail: inna@il.ibm.com*
- *Jonathan Bnayahu is with IBM Watson Health Technologies, E-mail: bnayahu@il.ibm.com*

Proceedings of the IEEE VIS 2016 Workshop on Temporal & Sequential Event Analysis. Available online at: eventevent.github.io

include simple elements, such as bars, icons, symbols, but also more complex structures such as areas, lines or even stacked charts. Channels include all visual attributes, such as color, size, shape, and others. that are freely configurable within a set of available options. Further, part of the setting is the value attribute, which encoded in the chart as the y-axes variable. The value attribute, in order to support a large variety of options, can be one of the data attributes, or a function based on attributes of the data.

We support several types of charts for difference natures of time, as described by Aigner et al. [1]. Sequential time representation for trends and tendencies is supported by timeline charts, which is demonstrated in this work. In addition, beyond the scope of this paper, we support radial and calendar charts to detect seasonality patterns. Relationships between events and patterns is made assessable though interaction techniques and through alignment functions to index dates that support absolute and relative time dimensions. Configurations share a common syntax for each chart.

A simple example of a configuration could be like:

```
{"settings": [{
  "key":"eventType",
  "valueAttr":"value",
  "chart":"symbol",
  "symbol":"diamonds",
  "color":"#ff7f00"
]}
```

An example with more sophisticated configuration to compute the BMI value on height and weight during a diet:

```
{"settings": [{
  "key":"eventType",
  "valueAttr": function(d){
    var bmi = d.weight/(d.height/100*d.height/100);
    return bmi;
  },
  "chart":"line",
  "color": {
    scale:"threshold",
    range:['#ef8a62','#f7f7f7','#67a9cf'],
    domain:[18.5,25,infinite]
  }
}]}
```

In addition, since all charts are aligned along the same temporal dimension, event types can freely be merged into one single chart with multiple axes using the *mergeTo* property with reference to the key attribute. Interaction is key in visualization, which is supported out of the box using the *brush, pan, zoom* and *mouse events* for each setting. The former generate direct manipulation on the charts, and the latter shows tooltip with all attribute of a data record.

3 Instantiation

We instantiated the described methodology on a fitbit (https://www.fitbit.com/) dataset as it is available through their official API (https://dev.fitbit.com/docs/). For demonstration purposes, we showed in our ipython Jupyter Notebook only the heart-rate and activity levels. Each of these variables shows the relative amount of time spend during a workout in one of four different levels reaching from low-to high categories. Results are shown in Fig 2. The first chart on the top shows bars for each of the conducted workout. The heart-rate and the activity levels are shown in the two consecutive charts as stacked area-charts. Color in theses charts correspond to low (pale) to high levels (intensive colors).

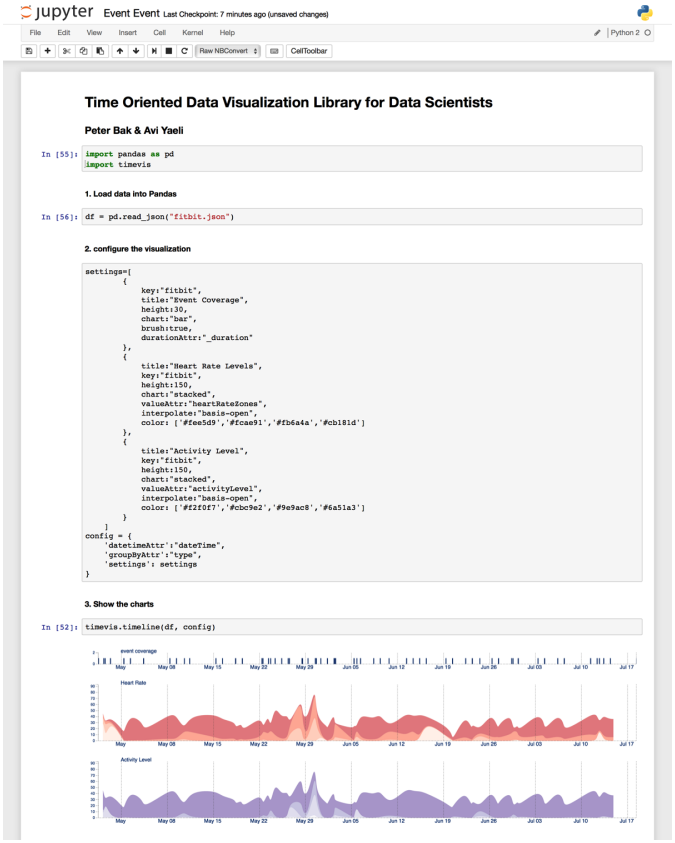

Fig. 2. Time-series visualization of fitbit dataset in a Jupyter notebook.

The notebook is divided into three parts. First, the user is required to load the pandas library together with our python wrapper for the visualization called *timevis*. Second, the user has to load the data in to pandas' data-frame using the appropriate data format and source for the import. Third, the user has to configure the charts and define the time attribute of the data. In the current example, we simplified the data during preprocessing, to easier accommodate for stacked charts. This step can of course be conducted within the configuration as described previously.

The results as shown on the charts are somewhat expected, but contain some interesting events. It is clearly visible that the data was obtained by a highly active person with periodic and consistent activity, indicated by dominant intensive colors. As typical for sporty people, heart-rate and activity levels go hand-in-hand, almost parallel. Also the values show consistently high activity levels, and high heart-rate levels, which are expected when data is obtained mainly during workouts. However, at several occasions, heart-rate is significantly low, but activity level is very high. We inquired further what type of activity would result in such a pattern. Activity levels combine several measurements together into a higher level concept, not just heart-rate. In our case, this pattern was due to Yoga and Pilates that did not raise heart-rate so much, but due to the movements of the arm are captured as high activity.

References

[1] W. Aigner, S. Miksch, H. Schumann, and C. Tominski. Visualization of time-oriented data, 2011.

[2] T. Munzner. A nested model for visualization design and validation. *IEEE transactions on visualization and computer graphics*, 15(6):921–928, 2009.