Data Analysis Approach for Large Data Volumes in a Connected Community

Supriya Chinthavali*, Sangkeun Lee*, Michael Starke*, Junghoon Chae*, Varisara Tansakul*, Jeff Munk*, Helia Zandi*, Teja Kuruganti*, Heather Buckberry*, Mahabir Bhandari* James Leverette†
*Oak Ridge National Laboratory, Oak Ridge , TN
{chinthavalis, lees4, starkemr, chaej, tansakulv, munkjd, zandih, kurugantipv, buckberryhl, bhandarims}@ornl.gov
†Southern Company, Birmingham, AL
{jalevere}@southernco.com

***Abstract*—Recent advancements within smart neighborhoods where utilities are enabling automatic control of appliances such as heating, ventilation, and air conditioning (HVAC) and water heater (WH) systems are providing new opportunities to minimize energy costs through reduced peak load. This requires systematic collection, storage, management, and in-memory processing of large volumes of streaming data for fast performance. In this paper, we propose a multi-tier layered IoT software framework that enables effective descriptive and predictive data analysis for understanding live operation of the neighborhood, fault identification, and future opportunities for further optimization of load curves. We then demonstrate how we achieve live situational awareness of the connected neighborhood through a suite of visualization components . Finally, we discuss a few analytic dashboards that address questions such as peak load reductions obtained due to optimization, customer preference for automatic control of appliances (do they override the automatic control of HVAC?, etc.).** [1]

***Index Terms*—IoT, agents, data analytics, behind-the-meter**

I. Introduction

With the growing volume of intelligent devices such as low-cost sensors with embedded communications and building loads with microprocessor controls, the fierce volume of reported information is expected to become a storing and processing challenge. Early projections from a study developed in 2011-2012 for 2020 estimated that 50 billion internet of things (IoT) devices would be deployed [1]. As the storing of reported data by devices is growing rapidly, applications utilizing this data are also expanding. Machine learning algorithms [2]–[6], optimization and control techniques [7], and visualization [8] are in development to better support both building owner and the modern electric grid.

Machine learning utilizes large and significant data sets to establish higher-quality information often for supporting optimization and control decision making. This requires that the data is available and can be retrieved at a high volume. Furthermore, the machine learning approach and information to be repeated is often stored for post evaluation.

Optimization and control, depending on the method, can provide a wealth of data. Model predictive control techniques develop projections on expected system responses over a time a window [7], [9]. Performing optimization at a 5-minute resolution for 24 hours, easily leads to 288 data arrays of the unknown variables within the optimization problem. Depending on the frequency of the optimizations and the number of undetermined variables this can quickly lead into gigabytes (GB) or terabytes (TB) over an extended period. Finally, retrieving this information for inspection dynamically and quickly for a user can be very challenging.

II. Background

Alabama Power has constructed a neighborhood called Smart Neighborhood, to develop, deploy, and evaluate the incorporation of intelligent systems within a community. In this demonstration project, the 62 home community has been retrofitted with IoT enabled heating, ventilation, and air conditioning (HVAC) systems, water heater (WH) systems, refrigerators, and other devices. Each home is also sub-metered by a circuit to identify general electrical consumption behaviors and evaluate other potential opportunities in energy savings and demand reductions. As part of this work, optimization and control of the HVAC and WH systems to support the grid in a transactive vision is under investigation [7]. A cloud-based distributed control architecture has been developed and is under evaluation for supporting both the home owner and the electric grid needs.

In this paper, a platform for a post evaluation of large data sets has been developed that reviews the model predictive control accuracy, establishes energy savings and cost savings from both the customer side and utility side, customer overrides, and outage statistics. This is for a community with intelligent HVAC and WH systems, optimization and controls, transactive negotiation, and learning.

III. SOFTWARE FRAMEWORK FOR ANALYTICS

In this section, we describe the software architecture for connected community data analysis and its components. Figure 1 shows the conceptual architecture of the overall system.

Request URL	Description
/get_hvac_status	Get the most recent status of all HVAC systems
/get_list_of_ids	Get the list of house identifiers
/get_forecast_hvac	Get the forecast data for HVAC of a specific house for a given timestamp
/get_pg_channel	Get the raw sensor data for a given time period
...	...

Table I: Example of REST APIs provided by the analytic server

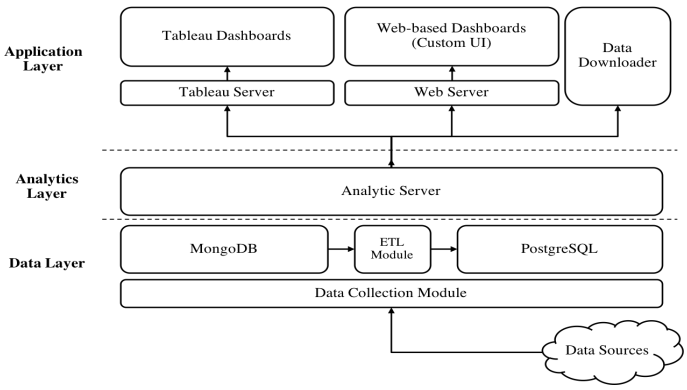

Figure 1: Connected Community Architecture

The architecture is composed of three layers - *Data Layer*, *Analytics Layer*, and *Application Layer*. The goal and details of each layer are as follows.

A. Data Layer

The goal of the data layer is to collect, transform, and store a large volume of data from various data sources in a connected community. Data including sensors, microgrid and from optimization tasks need to be stored for analytic purposes. Dealing with such data is challenging because they are not only often large volume and have to be processed on time. Also, because the existing variety of sensors and the possibility of new sensors can be added in the future, database systems should be flexible enough to adapt to changing data structure or formats.

Each home in the community has connected equipment and devices, and the community is supported by a microgrid. And, a multi-agent system (MAS) distributes, collects information from single homes and perform optimization tasks [7]. On the MAS side, the historical data is not permanently stored; however, one of the goals of our analytic tasks is to understand the yearly, monthly, and weekly data trends for a specific home or the entire community, and it requires to maintain all historical data in the database. A *Data Collection Module* is responsible for pulling device, optimization, and microgrid data from the MAS via RES API. The module pulls data every 5 minutes and stores the data. To speed up the process, the data pulling task is multi-threaded, where each thread takes care of specific kind of data (e.g., optimization, sensor, etc.). The original data we receive is in JSON (JavaScript Object Notation) format, which is a flexible open-standard file format consisting of attribute-value pairs and array data types. We chose MongoDB [10] to store the retrieved data as MongoDB can naturally store JSON-like documents with schema and provides expressive querying interface and indexing. Scalability is another important factor, as data being collected every 5 minutes from many homes can grow rapidly. MongoDB scales horizontally using sharding and supports load balancing across shards.

Although MongoDB is great for storing semi-structured data like JSON documents; many analytic tools such as Tableau and statistical methods still prefer tabular (i.e., relational) data. Thus, *ETL (Extract, Transform, Load) Module* extracts data from MongoDB collections, transforms the JSON documents into tabular data rows, then loads them into relational tables in a PostgreSQL [11] database. Unlike MongoDB, a relational database like PostgreSQL requires predefined table schema to store data. The data is stored in five tables named *channel*, *sensor*, *thermostat*, *waterheater*, *forecast_hvac*, and *forecast_wh*. The module also collects data from an FTP server where the additional microgrid data is stored in CSV (Comma Separated Values) format. Microgrid data is stored in one table in a PostgreSQL database which contains data collected from battery, environment, generator, load bank, main feed, relays, and solar. We created indices for both MongoDB and PostgreSQL based on frequently requested queries.

B. Analytics Layer

The analytics layer has a single component *analytic server*, which is a REST (Representational State Transfer) server [12]. Components in the application layer (e.g., data downloader, web dashboards, Tableau dashboards) request data to the *analytic server*, and the server is responsible for maintaining the connections to MongoDB and PostgreSQL databases and providing a unified interface to applications. MongoDB and PostgreSQL use different query mechanisms, scripting, and SQL respectively; however, via requesting data to the analytic server, application developers do not need to use two different ways to access data. We implemented the server using Tornado Python web framework and asynchronous networking library [13]. The current version of APIs includes 26 functions. Table I shows several example of implemented functions and their descriptions.

Note that some of the queries requested by the application may take a very long time to process. We also notice that same queries are often requested more than once in many cases. For example, when a user uses a web dashboard, queries initiated by default will be the same for the same date. Since our database is dealing with historical data, where the data created in the past will not be updated, the same API call always

returns the same results. When the analytic server receives an API call, it first checks its local cache repository to see if the same request has been processed recently. The URL of the requested call along with its input arguments is used as a search key for the cache repository. If the server cannot hit a cache, then it sends database queries to the databases and constructs the JSON response object that will be returned to the client. Before returning the result to the client, the result is converted into a byte stream and saved as a file using Python object serialization. On the other hand, if the server can find a cached file using the information of requested API call, instead of sending queries to the databases, it will identify the corresponding file and convert the byte stream into a Python object. This process allows bypassing Input/Output between the analytic server and the databases. How long and how much size of cached query results can persist on the server is configurable and can be adjusted by the server administrator.

Another important role of the analytic server is to standardize the data and perform additional descriptive and predictive based analysis on the existing data. For instance, different time zones may have been used for different which requires standardization before the data is exposed to applications; and aggregation of data such as summarizing at daily, weekly and monthly resolution, calculating summation, average, minimum and maximum values across different data points can be processed within the API functions.

C. Application Layer

The application layer is composed of several components that can be used by users to perform analytic tasks. The components include *Community Data download toolkit*, *Tableau dashboards* [14], and *Web dashboards*. These components communicate with the analytic server or directly with databases to access the data and fulfill various users analytic needs.

Community Data download toolkit is a simple toolkit that allows users to download data for a specific home and time in CSV format file. To download data directly from multiple devices, optimization data, and microgrid data is not a straightforward task, because every device and system uses different standards and/or APIs. Since our data layer and analytic layer abstracts such complexity, the toolkit can pull the data by using analytic servers APIs. Being able to download data in CSV for users is very useful, as many external data analytics tools (e.g., Excel, R, Jupyter notebook, etc.) can easily use this data format.

Tableau-based visual analytics dashboards is an application that provides visual analytics capabilities to users. Tableau [14] is a software platform that can create interactive data visualization dashboards, and it can either access our data layer via Web Data Connector through the analytic server or access PostgreSQL directly. An advantage of using Tableau is that we can quickly explore the data and create visualizations with various perspectives. We created Tableau workbooks and host the workbooks using *Tableau Server* so that can users can them using via their web browsers.

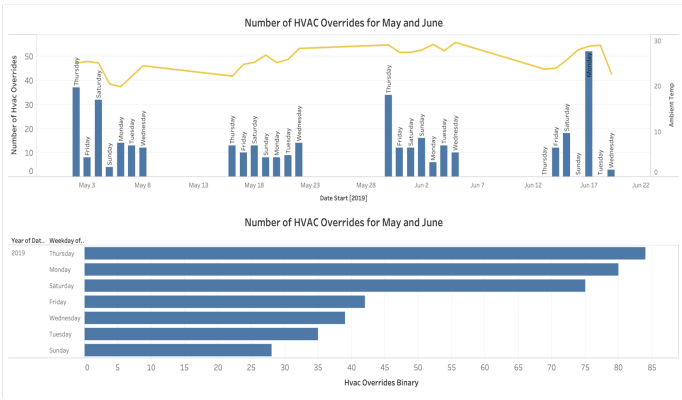

Figure 2: Overrides Dashboard: May and June 2019

Lastly, we implemented an in-house *Web-based visual analytics dashboards* to provide more custom data visualizations. Although Tableau is a great tool for quickly exploring the data and creating interactive visualizations, we needed to create specialized visualization widgets designed for connected community, which are not supported by Tableau. We implemented a web server using Tornado [13]. The web server renders web pages according to HTTP requests by communicating with the data server. We mainly used D3.js [15] and Twitter's bootstrap library [16] for our visualization components.

The details of the implemented visual analytic dashboards will be discussed in Section IV.

IV. Visual Analytics

A. Tableau-based visual analytics dashboards

We developed several Tableau-based dashboards to gain a better understanding of impacts of automatic control of appliances such as HVAC and WH on the peak load shifts and potential cost savings shown in Figure 3. We chose two days (May 17 and May 26, 2019) where the optimization was not dispatched to the neighborhood for HVAC control on one of the days and the dispatch was on the other day. Comparing the peak power difference between these two days during the peak price duration and off-peak price, it is clear that we save nearly 12KW per day for a residential neighborhood of about 62 homes.

Figure 2 is another example of a web-based analytic dashboard that displays the home owner's preference to override the automatic control of HVAC for the months of May and June, 2019 aggregated by the day of the week for weeks that automatic control (optimization dispatch is ON) was enabled. The ambient temperature during those months is being overlaid to understand the correlation between weather changes and a home owner's willingness to override their thermostat schedule.

We provide a suite of visualization components in the form of web dashboards. This visual analytics dashboard allows users to explore, understand, and monitor large-scale data. The dashboard provides multiple types of visualizations to show different types and levels of data according to the purposes of

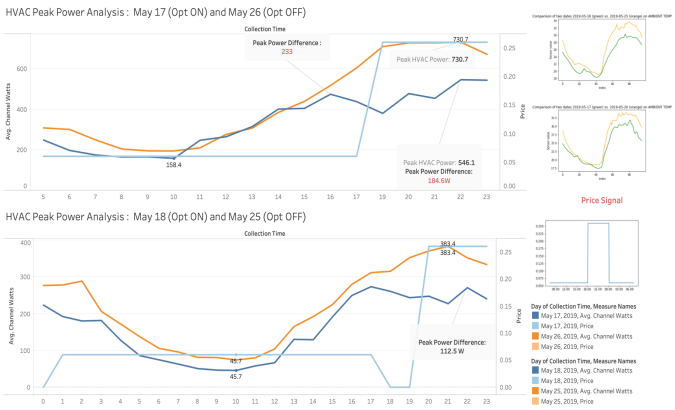

Figure 3: Peak Analysis Dashboard: Comparison between 2 days in May 2019 when optimization is ON and OFF

analysis. In this section, we describe the features and benefits of each component.

B. Web-based dashboards

Our visual analytics system queries data from the analytics layer described in Section III-B to provide the live status of the neighborhood, for example, total power consumption for 24 hours, HVAC and WH connection status, weather, optimization data for a user-selected date. The main dashboard visualizes the data in different visual representations, such as a stacked stream graph and multiple coordinated views, as shown in Figure 4. The multiple coordinated views are for displaying various types of data for the individual house. The following sections describe more details of the visualizations.

1) Main Power Streamgraph: The stacked streamgraph provides an overview of the total main power usage patterns for neighborhood and house level. The x-axis and y-axis of the streamgraph are for the time in UTC and the power usage, respectively. Each stream of the stacked stream graph represents the main power usages for the individual house, where the order of each stack and color for each stream account for usage pattern. The pink dots at the bottom of main power streamgraph in Figure 4 indicate how many homes' devices got disconnected throughout the day.

2) Multiple Coordinated View: The multiple coordinated views at the bottom of the dashboard are to display more detailed information for a selected house in the streamgraph, such as the power usages and temperatures of HVAC and water heater and temperatures at zone levels. Note that every house has different floor plans which differ in the number of zones.

The line graphs display HVAC and WH power usages (left column), HVAC and WH temperatures (middle column), and actual and setpoints temperature for each zone (right column).

The small plots under the line plots allow users to select a specific time window which line graphs will then show the plot at a specific time.

3) Area Charts: We provide specific area charts shown in Figure 5. The top chart shows HVAC temperatures and the bottom chart shows WH power. These charts display forecasted and actual values and indicate when the forecasted values are under or over-forecasting. If a forecasted value is higher than an actual value, the area between is filled by green color and vice versa by orange color. The filled areas between actual and forecasted lines help recognize the differences. There are two examples shown in Figure 5.

V. Challenges and Future Work

The databases tend to grow exponentially in size with thousands of IoT devices reporting data to the cloud instance for even a small residential neighborhood. The web-based application providing overview dashboard functionality was performing with slow response times. Performance profiling of the application quickly identified database queries were the bottleneck in performance. Also, SATA disk drives across NFS with hundreds of millions of rows in the pertinent tables (with even distribution) and very low memory settings for shared-buffers and work-mem parameters was an issue.

Performance tuning for poor performing queries was implemented through memory, SQL and index optimization. Further optimization that we have experimented is clustering tables based on the primary index that the table will be searched by. Conceptually, the data is stored on disk sequentially by the chosen index. This can lead to better performance from Bitmap Index or Heap Scan operations by limiting the number of pages read from disk. Also, a potential future optimization step is table partitioning. Since PostgreSQL supports table partitioning, enabling this feature allows the creation of parent-child relationships between tables, with the child tables being driven by a particular data field - for example, timestamp_utc. This will drastically improve the performance of the query against the analytic server.

VI. Conclusion

Automatic control of appliances such as HVAC and WH systems are enabling utilities to reduce energy costs through peak load shifting at a neighborhood level. This necessitates systematic collection and management of streaming big data. In this paper, we proposed a multi-tier software framework that enables effective data analysis and demonstrated the benefits of this implementation in providing live situational awareness of a smart neighborhood of 62 homes in Alabama and supporting daily operating and maintenance. We also demonstrate a few analytic dashboards that answer important questions related to peak load reductions obtained due to optimization, and customer consumption and behavior pattern when they are offered automatic control of appliances. Future studies will focus on further refining this framework to enable development of deep learning-based predictive model for quantification of cost savings through optimized control of HVAC and WH appliances based on electricity price.

References

[1] E. Dave *et al.*, "How the next evolution of the internet is changing everything," *The Internet of Things*, 2011.

[2] H. Zandi, M. Starke, J. Munk, T. Kuruganti, J. Leverette, and J. Gregor, "An Automatic Learning Framework for Smart Residential Communities," *International Journal of Smart Grid and Clean Energy*, 2019.

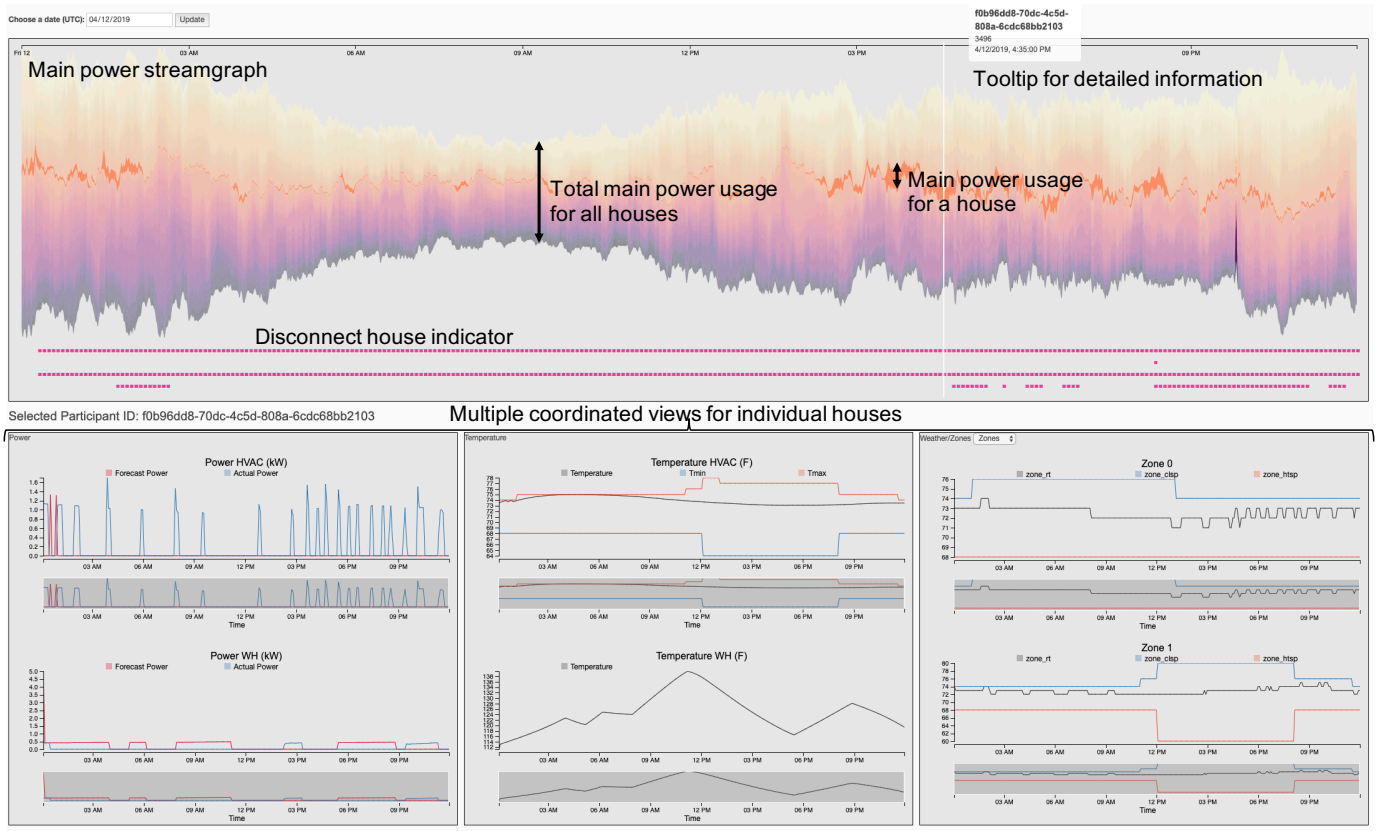

Figure 4: Main Power Streamgraph and Multiple coordinated views for individual houses: The dashboard consists of a stacked stream graph and multiple coordinated views

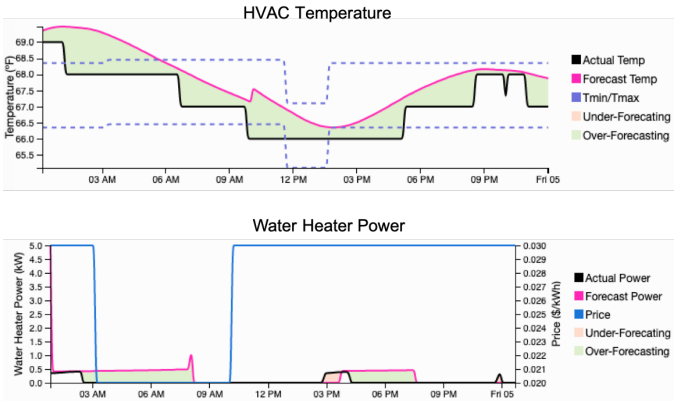

Figure 5: Area Charts: The filled areas between actual and forecasted lines (under-forecasting: orange, over-forecasting: green) help recognize the differences.

[3] J. L. G. Ortega, L. Han, N. Whittacker, and N. Bowring, "A machine-learning based approach to model user occupancy and activity patterns for energy saving in buildings," in *2015 Science and Information Conference (SAI)*, July 2015, pp. 474–482.

[4] B. Cui, J. Munk, R. Jaccksон, D. Fugate, and M. Starke, "building thermal model development of typical house in u.s. for virtual storage control of aggregated building loads based on limited available information."

[5] D. Suh, H. Kim, and J. Kim, "Estimation of water demand in residential building using machine learning approach," in *2015 5th International Conference on IT Convergence and Security (ICITCS)*. IEEE, 2015, pp. 1–2.

[6] A. Delorme-Costil and J.-J. Bezian, "Forecasting domestic hot water demand in residential house using artificial neural networks," in *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2017, pp. 467–472.

[7] M. Starke, H. Zandi, J. Munk, T. Kuruganti, and J. Hall, "Agent-Based System for Transactive Control of Smart Residential Neighborhoods," in *IEEE Power and Energy Society General Meeting*, 2019.

[8] R. Godina, E. M. Rodrigues, E. Pouresmaeil, and J. P. Catalão, "Home hvac energy management and optimization with model predictive control," in *2017 IEEE International Conference on Environment and Electrical Engineering and 2017 IEEE Industrial and Commercial Power Systems Europe (EEEIC/I&CPS Europe)*. IEEE, 2017, pp. 1–5.

[9] J. Nutaro, T. Kuruganti, D. Fugate, and M. Starke, "An inexpensive retrofit technology for reducing peak power demand in small and medium commercial buildings," 2014.

[10] K. Banker, *MongoDB in action*. Manning Publications Co., 2011.

[11] B. Momjian, *PostgreSQL: introduction and concepts*. Addison-Wesley New York, 2001, vol. 192.

[12] M. Masse, *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces*. " O'Reilly Media, Inc.", 2011.

[13] M. Dory, A. Parrish, and B. Berg, *Introduction to Tornado: Modern Web Applications with Python*. " O'Reilly Media, Inc.", 2012.

[14] D. G. Murray, *Tableau your data!: fast and easy visual analysis with tableau software*. John Wiley & Sons, 2013.

[15] N. Q. Zhu, *Data visualization with D3. js cookbook*. Packt Publishing Ltd, 2013.

[16] J. Spurlock, *Bootstrap: responsive web development*. " O'Reilly Media, Inc.", 2013.