WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Real-Time Stock Market Analytics Using Python and Big Data Tools

Ponagantiadarsh[1], Varshini Priyam Vada [2]

[1,] P.G. Research Scholar, Department of MCA-Data Science, Aurora Deemed to Be University, Hyderabad, India
[2]Assistant Professor, Department of MCA, Aurora Deemed to Be University, Hyderabad, Telangana, India
Email: [1] ponagantiadarshpatel@gmail.com, [2]varshini@aurora.edu.in

ABSTRACT:

The continuous and fast-growing large-scale financial data require strong supports for real-time analysis, visualization, and decision making. 2 WH Y B I G DAT A A NALYTI CS Nowadays, analysis on a large scale or on fast-evolving data can be challenging for traditional systems, calling for scalable and interactive analytics systems. This project provides us a holistic outline of Real-Time Stock Market Analytics in Python and Big Data based on data challenges have been confronted so far.

The system proposed includes multiple technologies: MongoDB for scalable and flexible storage, Pandas for data preprocessing and exploratory analysis, and Matplotlib for dynamic visualizations, whilst a Tkinter-based Graphical User Interface (GUI) delivers the most advanced and user-friendly experience for both technical and non-technical staff. The tool provides data ingestion from APIs and CSV files, and users can easily manage and analyze the historical as well as live market data.

It offers interactive record addition, dynamic chart generation (by market; bar, line, and pie charts) and real-time data updates so investors, analysts and researchers can easily track market trends. Experimental results demonstrate that the system aids in simplifying complex financial analytics tasks, increases financial literacy, and also provides an economically viable alternative to commercial tool infrastructure in banks.

Some ideas for the future, ie inclusion of ARIMA, machine learning price forecasting, realtime api for price updates and more advanced dashboards for a deeper understanding of the trend.

Keywords: Stock Market, Big Data, Python, MongoDB, Real-Time Analytics, Data Visualization, Tkinter, Pandas, Predictive Analytics

I. INTRODUCTION

In the data rich world of finance, informing investors, assessing risk, predicting economy, stock market analytics is of prime importance. Stock market data arrives at the rate of terabytes per second: price movements, volume, and various global economic indicators and sentiment.

As a result, it has become more important than ever for investors, analysts and policymakers to be able to digest this data in real-time and context and make informed decisions. Nevertheless, the large amounts, fast pace, and diversity of stock market data make its storage, processing, and visualization decidedly challenging.

Classic approaches to stock analysis sometimes result in static datasets and time consuming manual work, which limit the amount of information they can process from large-scale and rapidly evolving data flows. Moreover, the vast majority of marketing analytics tools cater to enterprises and are prohibitively expensive or don't offer the level of customization needed by individual investors, researchers and academic institutions.

These limitations demonstrate the demand for a real-time, scalable, and end-user-friendly system, which is able to effectively handle, analyze and visualize the stock market information.

This study tackles these challenges by presented Real-Time Stock Market Analytics System combining Python and Big Data technologies to provide a flexible and real-time analytics system that is low-cost, modular and interactive. The system makes use of MongoDB for scalable data storage, Pandas for lightweight pre-processing and analysis of data, and Matplotlib for easy-to-use visualisation of the data. Finally, a GUI written with Tkinter provides an accessible environment to the system for lay and technical users.

The main purposes of this study are:

1. The objective Establish a real-time stock market analytic system that integrates data storage, Pre-processing and visualization in one place.
2. Interactive functionalities are aimed at providing dynamic data updates, visual trend drawing and record control implemented as Python tools.

3. This is posed to assess the usability, scalability, and application potential of the system among the investors, academicians, researchers.

By meeting these objectives, the research will build a decision support tool that facilitates decision-makers, financial literacy and can also reduce the disparity between the increasing capability of advanced analytics methods, and its practical applicability in real financial environments.

II.LITERATURE SURVEY

A. Existing Methodology

Existing stock market analytics research has followed various methodologies for data retrieval, pre-processing, analysis, and visualization:

1. Data Collection and Storage
    - The conventional method was to use APIs like Alpha Vantage or Yahoo Finance to obtain current or historical stock data.
    - Hadoop and Apache Spark platforms were employed for managing big financial data, whereas MongoDB supported data storage with the schema absence for high frequency trading data (Grolinger et al., 2013).
2. Data Analysis Techniques
    a. ARIMA (Box & Jenkins, 1976) based statistical forecasting models were initially applied for time series prediction.
    b. Subsequently, machine learning approaches, including Random Forests and Long Short-Term Memory (LSTM) networks, obtained enhanced predictive accuracies by plotting non-linear structures into the market (Fischer & Krauss, 2018).
3. Real-Time Processing
    a. Apache Kafka and Spark Streaming were Low-latency processing pipelines that were able to handle Real time event and make fast decisions in financial systems (Kambatla et al., 2014).
4. Visualization Tools
    a. Matplotlb and Seaborn generated charts such as line plot line graph and bar chart were widely used for visulization (Few, 2006).
    b. Plotly and Dash have brought interactive dashboards for improved analysis of financial data trends.

Limitations: Nevertheless, available tools could have high computational demand, may not be user friendly or could be expensive for small investors and academic use. This required scalable, inexpensive and real-time analytics system with a responsive interface.

B. Proposed Methodology

The Real-Time Stock Market Analytics System integrates storage, real-time analytics, visualization and a GUI in a single environment, which is implemented with Python language and open-source technology.

B. 1 System Architecture

1. Data Capture:
    - For retrospective research, stock data will be imported directly from CSV files and/or API's (Alpha Vantage).
    - This system will have dynamic updates for almost real time research.
2. Data Storage:
    - This database will be managed using MongoDB which will hold large quantities (terabits) of semi-structured data from the financial market.
    - The collections in MongoDB will allow Python scripts to interact with the data using the PyMongo library.
3. Data Preprocessing:
    - Pandas & NumPy libraries will remove missing values and format errors, apply statistical calculations, etc. to provide processed clean data usable in the application.
4. Visualization Module:
    - Matplotlib and Plotly libraries will create pie charts, bar charts or line graphs for interactive market trend research.
    - The charts will be customizable in the GUI by users.
5. GUI:

- The whole system is developed in Tkinter, and it gives a GUI so users can load and view datasets, add new records and view visualizations without coding.

6. Real-Time Analytics:

   - The system is using improved stream data pipelines, enabling the user to see development dashboards and derived insights in real-time.The system is designed to support real-time and accurate decision-making for stakeholders using the model, including analysts, traders and investors.

Gap Analysis

The literature split and existing solutions when assessed posed many gaps and limitations.

1. High Complexity and Resource Costs.

   Many predictive models, particularly deep-learning models come with demanding computational resource requirements, as well the complexity of building the models which limits opportunities for students, smaller traders, or independent researchers seeking lightweight and simplified alternatives.

2. Commercial Options are expensive.

   Almost all commercial platforms focused on analytics have substantial licensing fees associated with their product, which are intended for enterprise, unlike a low-cost open-source alternative.

3. Little Available Real-time Visualization With Facilitated Interfaces.

   While there are real-time analytics tools available, integrated platforms that can handle real-time processing, synchronous visualization, and have an easy to use GUI are few and far between, especially for local or academic users.

4. No Available Customizable open-source Platforms.

   Most tools function separately therefore require a significant degree of technical knowledge to connect them together. Very few systems offer some kind of template or modular open-source framework with which smaller clients could easily install and modify.

This project sets out to close these gaps by building a lightweight, flexible, scalable and low-cost platform with open source technology. This platform will provide:

- User friendly ease of use via a Tkinter interface,
- Flexible structured data storage with MongoDB,
- Fully open-source access with Python, Pandas, and Matplotlib, and
- Future-wide scalabilitity into predictive analytics and machine learning.

III. METHODOLOGY

1. Dataset Retrieval

Stock market datasets are the integral component of the analytic system.

- Dataset Real-Time / Historical - APIs (such as Yahoo Finance and Alpha Vantage) allow for both historical datasets and real-time data streaming. The datasets are timely for analysis.
- Dataset Storage - Stock market datasets are stored in MongoDB which is a NoSQL database, and permits a lot of data to be stored with each record containing a lot of fields such as Ticker, Date, Open, High, Low, Close and Volume. The MongoDB schema-less structure allows for rapid ingestion of both structured and semi-structured datasets and allows for scalable loading.

2. Data Cleaning / Data Preparation

Frequently, the information gathered from the internet contains missing values, duplicates, inconsistent formats, etc. Therefore, all the datasets must be pre-processed before any analysis.

- Pre-Processing: Using the pandas and NumPy libraries the pre-processing of of datasets includes dealing with missing values, standardizing date formats, and removing duplicates.
- Statistical Pre-Processing: Statistical pre-processing will be applied to things such as Normalization and outliers before we visualize datasets and before we use them in machine learning.

3. Tools Coffee Orc and Justification:

The system uses a collection of open-source tools, with each serving a purpose:

- MongoDB - Scalable, high-performance backend storage.
- Pandas & NumPy, these two open-source libraries facilitate programming and steps on manipulating data faster.
- Matplotlib and Plotly - Support both static (matplotlib) and interactive (Plotly) data visualizations.
- Tkinter - Provides users with an interface to interact with datasets through visualizations and controls.
- Scikit-learn - Will allow for machine learning models to install within the systems for predictive analytics in the future.

4. A Single Sheet Structure for Data Interaction

The front-end of the system is a GUI based in Tkinter that was developed for usability:

- Treeview widget - Exposed a statistically tabled presentation of the stock data
- Record Management - Besides viewing the data in tabular format, users can manually insert records, delete individual records or multiple records, and update records interactively from the GUI interface, while also myself visualize the data. The system locally published the Invaluable portion of the interactions for tiny- or individual-scale users, not for commercial purposes.

5. Components of A Dashboard

Data visualizations are at the heart of what the system accomplishes, with a clear intentional purpose to make market understanding more efficient:

- Types a charts - Line charts predominantly to track stock price mappings over time, bar charts comparing trading volumes, pie charts for proportion relationships
- Interactive Dashboards - Using Plotly, the system provided the framework for background interaction of zoming, filtering and patient searching features, whereby incorporating a richer avenue of analyses for traders and researchers.

6. Storytelling with Tableau (A Future possibility)

In future releases, tableau dashboards could be integrated for more advanced users and give way to storytelling analytics:

- Guided Insights: The storytelling feature in tableau can break financial data apart, in order, and help users to intuitively see more complex trends.
- Interactive Storylines: Allowing users to investigate financial scenarios for learning or professional use.

IV. IMPLEMENTATION

System Modules

The proposed real-time stock market analytics system will be built in a modular architecture. Therefore, each layer has its own functionality, while it is seamlessly nested within the other modules. The modules will be:

1. Database layer (MongoDB)

The database layer is the backend for the system. This uses MongoDB as a means to store stock market data and retrieve it whenever necessary.

- Functionality: It stores both historical and real-time data fields for example Ticker, Date, Open, High, Low, Close, and Volume.
- Benefits: MongoDB is a schema-less database in a NoSQL architecture. Therefore it is scalable, retrieval of data is quick, and it has flexibility for semi-structured data and it is capable of producing large amounts of data at higher frequencies; exactly what a financial data use-case needs.

2. Data Processing layer (Pandas, NumPy)

This layer executes data cleaning, data transformation, and the data analysis prior to visualization and/or model training.

- Cleaning: It will manage missing values, duplicates, and formatting.
- Transform: This is transforming raw financial data into more common formats and making data that is consistent for group data processing and analysis.
- Exploratory Data Analysis (EDA): It is here will generate summary statistics and provide insights about trends that should lead to more important interpretations.

3. Visualization Layer (Matplotlib, Plotly)

The visualization layer is responsible for presenting the data in chart or dashboard format.

- Static Visualizations: The user will mostly use Matplotlib in order to present trends of interest as line, pie, and bar charts.
- Interactive Dashboards: The user will be able to use Plotly to create real-time charts that allow for zooming, filtering, and a variety of ways to interact with the graphic. With systems like this, the user will be able to drill down further and increase their ability to make better decisions based on the identified trends.

4. Interface Layer (Tkinter GUI)

The interface layer is built with a Tkinter GUI which is the front-end for the user. The front-end provides the user the controls of a systems for a non-technical user.

- Functionality: The controls on the front-end consist of buttons and menus that allow the user to load a dataset, add a record, and create visualizations.
- Overall Purpose: The front-end was designed to provide a user-friendly interface to allow for some interaction with the system at any level of programming experience as a front-end. The system easily be used by students, small stock traders, or anyone else that needs to make a decision without programming as part of the process.

5. Development Layer for Machine learning (Scikit-learn)

In the future, Scikit-learn could be implemented for predictive analysis.

- Forecasting - For example, machine learning models – linear regression, random forest, and time series. These examples will help with identifying price trends.
- Decision Support - Also involves prediscriptive and predictive models.

V. RESULTS

Main Dashboard (Initial View)

Figure:1

The first figure shows the main graphical user interface (GUI) of the Stock Market Analytics system developed with Python's Tkinter library. The top of the interface contains labeled entry fields for users to insert important stock attributes, such as ticker symbol, date (homogenized and converted into YYYY-MM-DD format), open, high, low, close, adjusted close, and volume. Beneath these entry fields are two buttons — "Add Record" and "Generate Visualizations" — which will allow users to insert records into the system or to begin graphical analysis based on the previously-recorded dataset. The bottom half of the interface contains a scrollable table (specifically, a Treeview widget in Tkinter) which will display the stock data in tabular format consisting of the Ticker, Date, Open, High, Low, Close, Adj Close, and Volume. This allows users to manage records and simultaneously view and analyze substantial amounts of financial data in an organized setting.

Data Entry (Before Adding a Record)

The second image highlights the **data-entry process** within the system. In this case, the input fields have been populated with stock market details for the ticker symbol **NSEI** on the date **2025-01-05**. The open price is recorded as 5000, the high price as 7500.526, the low price as 4500, while the close and adjusted close are both 4250.36985. The trading volume entered is 25,685,656. This step demonstrates how the system allows users to manually input

new records, ensuring that any missing or external data can be incorporated into the dataset. The interface maintains simplicity by following a structured input format, making it accessible even to users with limited technical expertise.

Figure:2

The second image displays the data-entry process for the system. For this specific example, the entry fields have been filled with information from the stock market for the ticker symbol NSEI on the date of 2025-01-05. Open price is recorded as 5000, high price as 7500.526, low price as 4500, close price and adjusted close price is recorded as 4250.36985, which appear to be the same, and traded volume is entered as 25,685,656. This process is a good example of how the application allows users to manually log new records, so users can enter any data that is missing or from an external source into the data sets. There is great simplicity to the process at this point, as the users follow a standard entry structure format, allowing even the most non-technical users the ability to use the application.

Visualization Output (Bar Chart)

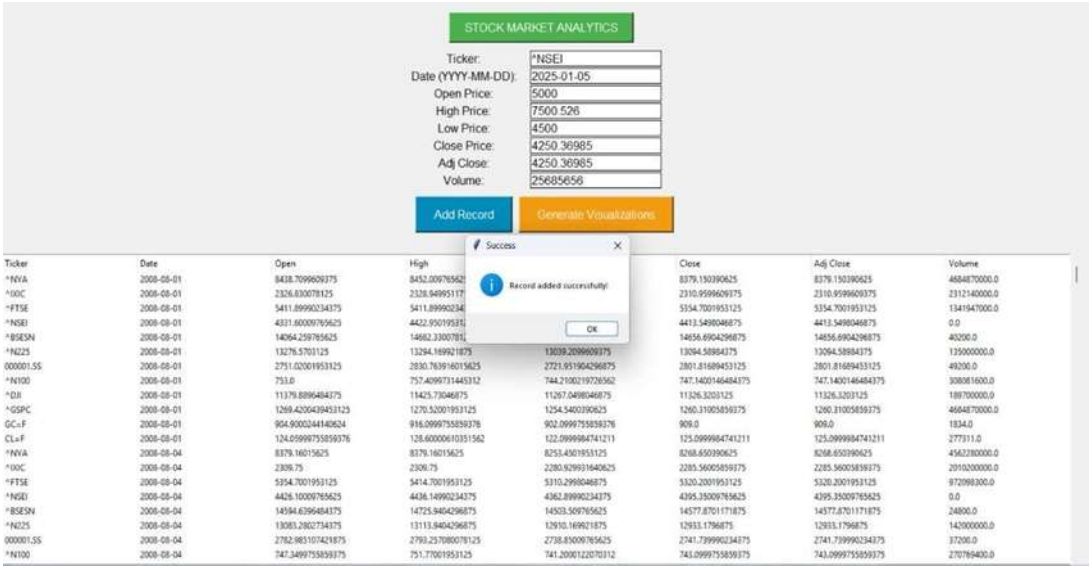

Figure:3

The third image shows the output from the visualization module of the application. After specifying the parameters, the application calls Matplotlib to produce a bar graph, relating stock tickers to their corresponding opening values. Each bar shows the opening value of a stock. Hence, users can observe deviations in market behavior among a number of companies or indices where the user specifies tickers. Though the dataset of tickers is large, causing many of the x-axis labels to be somewhat crowded, the visual clearly illustrates that the system can transform tabular stock data into a graphical form. Visual analyses such as these help users easily spot patterns, anomalies, and trends within the data presented which would be less obvious or appear hidden with raw numbers.

Confirmation Dialog (After Adding a Record)

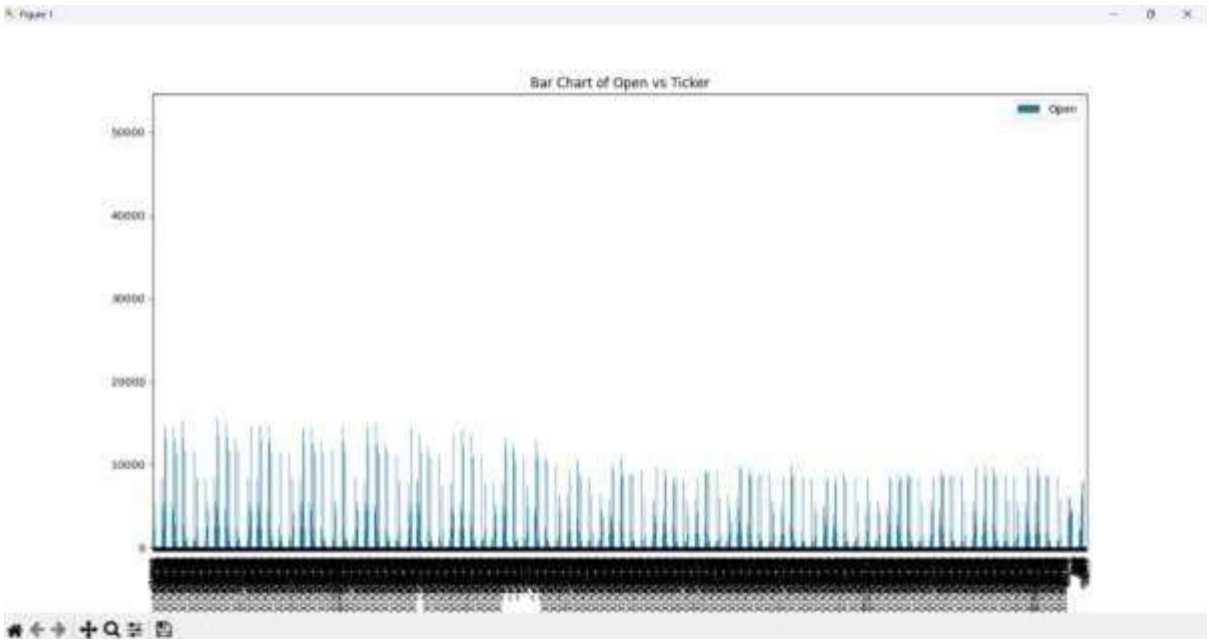

Figure:4

The fourth image is an example of the system's feedback of the system, for once, the record is added. When the user completes the particular details for the ticker NSEI and clicks submit, a confirmation dialog will appear with the message: "Record added successfully". This pop-up message also provides a level of confidence and trust from the user's viewpoint, confirming that the input has some verification and is stored in the base database. However, the success message is also part of the overall user experience; confirmation - as simply as possible - that user's input was noted and accepted, and nothing was ambiguous or obscurely generated, is part of a level of trust built into the application and positive user experience. Trust along with this level of feedback, completes the cycle of record-insertion, from filling-out, submitting and permanently saving of records.

Visualization Selector (Chart Configuration)

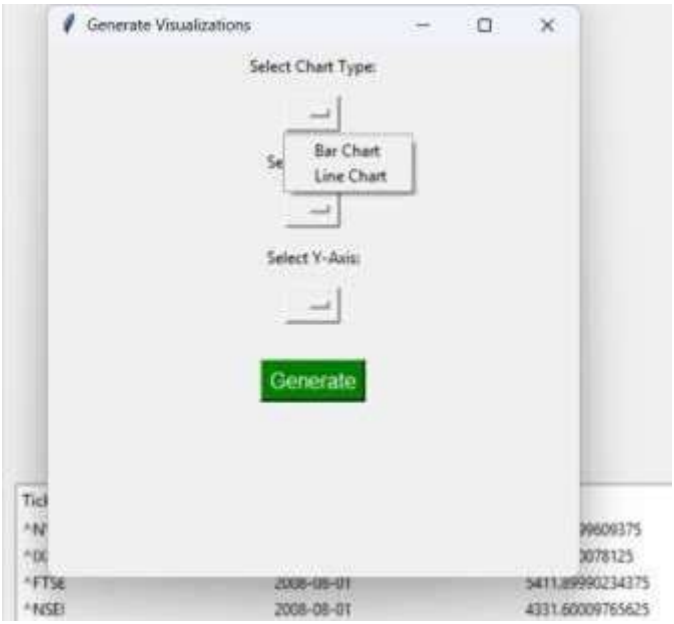

Figure:5

The fifth image shows the configuration window of the visualization of the system. Users have dropdown options to specify the chart type they want to create (Bar Chart or Line Chart) and also select the X-axis and Y-axis variables. Once the configuration is complete, the user can simply press the “Generate” button with the variables selected to produce the visualization. This was configurable so that a user is flexibly completing their analysis based on the data set they have and the type of analysis they wanted to perform. The system provides multiple chart types, and the user can select the axis variable to add further flexibility into their data analysis process and encourage deeper dive into their financial data.

VI. DISCUSSION

The system demonstrates several strengths:

- Database Integration: MongoDB deals with structured as well as semi-structured data at increased processed speed.
- Visualization: Matplotlib, Plotly to visualize the trends in a better way.
- User friendly: Due to the existence of Tkinter-based GUI The system is available also for non-programmer researchers.
- But the project was also constrained: There are a couple other visualization types that are missing: scatter, candlestick, histograms, etc.
- Static Data the project depended on pre-existing CSVs, not live APIs. error checking was rudimentary and could be improved basic input verification was in place but could be improved.

Opportunities for enhancement include:

- To integrate real time stock APIs for live updates.
- The support of other visualization types, such as candlestick and area charts.
- Developing predict models that will use regression, LSTM, or reinforcement learning.
- Developing mobile/web-based apps for wider availability.

VII. CONCLUSION

The project has shown that Python and Big Data tools can be used together to create a scalable, fast and intuitive stock market analysis system. The system features MongoDB as a backend storage, Pandas and NumPy for preprocessing, and Tkinter for GUI interaction, such that data is processed and accessed in a streamlined manner opening the system up not only to students but also to small-scale investors. In the modular construction, each element can operate individually and further without effecting operation of the other elements within the system.

We will show (and you will understand) the ease of financial data interpretation and visualization, thanks to visualization libraries as Matplotlib, as Plotly. The line, bar, and pie charts, that can be created, gives users the ability to have more insight into the trends in the market by changing the selected fields parameter and the type of chart. The integration of Plotly for interactive dashboards increases user interaction with zooms and filters, creating an experience similar to that of commercial analytics

Beyond just technical implementation, the project's overall contribution is the democratization of financial analytics. With a lightweight, open-source and cost effective solution, it enables students, researchers and small traders to process financial data without commercial software that is often expensive. More Feature Inputs such as Machine learning for prediction, Real Time API calls, Advance web bases dashboards to grow this system towards professional world.

REFERENCES

- Python Official Documentation: https://docs.python.org
- MongoDB Documentation: https://www.mongodb.com/docs
- Matplotlib Documentation: https://matplotlib.org
- Box, G.E.P. & Jenkins, G.M. (1976). *Time Series Analysis: Forecasting and Control.*
- Fischer, T. & Krauss, C. (2018). Deep learning with LSTMs for financial market predictions.
- Grolinger, K. et al. (2013). *Data management using NoSQL databases.*
- Kambatla, K. et al. (2014). Trends in Big Data Analytics. *Journal of Parallel and Distributed Computing.*
- McKinney, W. (2011). *Python for Data Analysis.*
- Few, S. (2006). *Information Dashboard Design.*