Interactive Stock Price Visualization Tool using Plotly and Yahoo Finance API

Alok Kumar Sah,
as0091@srmist.edu.in ,
Victor Khangembam,
vk4301@srmist.edu.in
Lakshya Pandya,
lp6918@srmist.edu.in
Dept. of Networking and Communications,
SRM Institute of Science and Techmology.

Abstract-- This project introduces an interactive and dynamic Python-based tool for visualizing stock prices using Plotly and the Yahoo Finance API. The user interface allows users to select from a range of popular stocks, such as Google, Tesla, Apple, Cisco, and Intel, or input a custom stock name for personalized analysis. Leveraging the capabilities of Plotly, the tool generates visually appealing candlestick charts that represent the stock's open, high, low, and close prices over the past 5 days at hourly intervals.

Keywords—Stock Visualization,Plotly library,Financial Technology(FinTech),Real Time data

I. Introduction

This project not only showcases the power of Plotly and the Yahoo Finance API in creating visually engaging stock price charts but also emphasizes the importance of interactivity and customization in empowering users to gain valuable insights into stock market trends. The tool serves as a valuable resource for investors, researchers, and anyone interested in exploring and understanding the dynamic nature of stock prices.

A. Motivation

Traditional methods of stock analysis often involve complex numerical data, making it challenging for the average investor or enthusiast to grasp the nuances of market movements. By leveraging the capabilities of Plotly and integrating real-time financial data from the Yahoo Finance API, this tool aims to bridge the gap between complex financial information and user-friendly visualization. The motivation behind this report is to provide a detailed exploration of the tool's development, functionality, and potential applications in empowering individuals to navigate the intricate landscape of stock markets.

B. Objective

- Real-time Data Integration
- Time Interval Customization
- Contribution to Financial Technology
- Educational Value

C. Problem Statement

The financial landscape, particularly in stock market investments, often presents a challenge for individuals to access and interpret real-time data effectively. Traditional methods of stock analysis may be cumbersome and lack the visual appeal necessary for a comprehensive understanding of market trends. There is a clear need for a user-friendly, interactive tool that seamlessly integrates with reliable financial data sources and provides an intuitive visualization of stock prices.

D. Challenges

While developing the stock price visualization tool, several challenges need to be addressed to ensure the effectiveness and reliability of the tool. These challenges include:

a. Real-Time Data Integration:

Fetching real-time stock data from the Yahoo Finance API requires robust error handling and efficient data processing to ensure accurate and up-to-date information.

b. User Input Validation:

Validating user inputs for stock selection and custom stock names is crucial to prevent potential errors and

ensure a seamless user experience.

c. Chart Customization:
Implementing a flexible and user-friendly chart customization feature, such as adjusting time intervals, demands careful consideration of user interactions and preferences.

d. Interface Design:
Designing an intuitive and visually appealing user interface that caters to both novice and experienced users, providing a balance between simplicity and functionality.

e. Modularity and Scalability:
Developing a modular codebase that allows for easy integration of new features or datasets while ensuring the scalability of the tool for potential future enhancements.

f. Data Accuracy and Reliability:
Ensuring the accuracy and reliability of financial data is crucial for the tool's credibility, requiring thorough testing and validation of data fetched from external sources.

g. Educational Component:
Integrating educational elements into the tool while maintaining a balance between simplicity and informativeness to cater to users with varying levels of financial knowledge.

By addressing these challenges, the project aims to deliver a robust and user-centric stock price visualization tool that overcomes common hurdles associated with financial data analysis and visualization.

II. LITERATURE SURVEY

The development of a stock price visualization tool involves drawing insights from existing literature on financial data visualization, interactive tools, and technologies. Here's a brief survey of relevant literature:

A. Financial Data Visualization:

Numerous studies emphasize the importance of visualizing financial data for effective analysis. Tufte's work on data visualization principles provides a foundation for creating clear and insightful charts, while more recent research explores advanced visualization techniques tailored to financial data representation.

B. Interactive Data Visualization:

The concept of interactivity in data visualization, as discussed by Shneiderman in his information-seeking mantra "Overview first, zoom and filter, then details-on-demand," serves as a guiding principle. Interactive visualization tools empower users to explore and analyze data dynamically, enhancing the decision-making process.

C. Python and Plotly:

Literature on the use of Python for financial analysis and visualization is abundant. Python's versatility and extensive libraries, such as Plotly, are widely acknowledged. The works of McKinney on Pandas and the documentation of Plotly contribute to understanding the practical implementation of these tools.

D. Financial Technology (FinTech):

With the rise of FinTech, literature explores the integration of technology in financial services. Research on the impact of FinTech on traditional financial markets, including tools for individual investors, sheds light on the evolving landscape and the need for accessible financial tools.

E. Real-Time Financial Data:

The challenges and opportunities associated with real-time financial data, as discussed in research by Aitken et al., highlight the importance of reliable data sources and efficient data processing for timely decision-making in financial markets.

F. User Interface Design for Financial Applications:

Literature on designing user interfaces for financial applications, such as the works by McFarlan and McKenney, underscores the significance of user experience and interface design in financial tools to ensure user engagement and effectiveness.

G. Educational Tools in Finance:

Studies on incorporating educational components in financial tools, like the work by Guo et al., provide

insights into creating tools that not only serve practical purposes but also contribute to users' financial literacy and understanding.

By synthesizing insights from these literature sources, this project aims to contribute to the existing knowledge by developing a stock price visualization tool that combines principles of effective data visualization, interactivity, and educational value to empower users in navigating the complexities of financial markets.

III. Requirements

A. Requirement Analysis

From the given scenario, we draw the following requirements:

- The tool should integrate with the Yahoo Finance API or a similar reliable financial data source to fetch real-time stock price data.
- The tool should use Plotly to generate visually appealing candlestick charts that represent the historical open, high, low, and close prices of the selected stock over a specified time period.
- Internet connectivity to retrieve the real time stock values using api
- Python running environment for the program

We need to configure a stock market value display keeping the following requirements in mind.

B. Hardware Requirement

a. Computer:
A standard personal computer or laptop with a minimum of 4GB RAM and a multi-core processor for smooth execution of the Python code and data visualization.

b. Storage Space:
Adequate storage space to accommodate the installed Python environment, libraries, and any additional datasets or files that may be utilized by the tool.

c. Internet Connection:
A stable internet connection is necessary for fetching real-time stock price data from external APIs, such as the Yahoo Finance API.

d. Graphics Processing Unit (GPU) (Optional):
While not strictly required, a dedicated GPU can enhance the performance of data visualization, especially when dealing with large datasets or complex chart rendering.

e. Operating System:
The tool is designed to be compatible with common operating systems like Windows, macOS, or Linux. Ensure that the selected operating system is up-to-date and supports the required Python libraries.

f. Web Browser (Optional):
If the tool includes a web-based interface or generates interactive visualizations in a browser, a modern and compatible web browser (e.g., Chrome, Firefox) is recommended.

C. Minimum Software Requirements:

a. Python:
The latest version of Python should be installed on the computer. The tool is designed to work with Python, and having the latest version ensures compatibility with the required libraries.

b. Integrated Development Environment (IDE):
Choose a Python-friendly IDE for development. Popular choices include Visual Studio Code, PyCharm, or Jupyter Notebooks. The selected IDE should support the development and execution of Python scripts.

c. Plotly Library:
Install the Plotly library for Python. This can be done using the package manager pip. Run the command: pip install plotly to ensure the tool can generate interactive candlestick charts.

d. yfinance Library:
Install the yfinance library for Python using the command: pip install yfinance. This library is essential for fetching real-time stock price data from the Yahoo Finance API.

e. Web Browser:

If the tool generates interactive visualizations in a browser, ensure that a modern web browser like Chrome, Firefox, or Safari is installed.

f. Operating System:
The tool is designed to be compatible with multiple operating systems, including Windows, macOS, and Linux. Ensure that the operating system is up-to-date with the latest patches and updates

IV. ARCHITECTURE AND DESIGN

A. Project Architecture

The architecture of the stock price visualization tool involves several components working together to fetch, process, and visualize real-time stock data. Here's a high-level overview of the architecture:

a. User Interface (UI):
The user interacts with the tool through a graphical user interface. This interface allows users to input their stock selection, customize the chart view, and explore the interactive visualizations.

b. Input Handling:
User inputs, such as selected stocks and time intervals, are processed by the input handling component. This ensures that the inputs are valid and prepares them for further processing.

c. Data Fetching:
The tool fetches real-time stock price data from the Yahoo Finance API or a similar financial data source. The data fetching component ensures a stable internet connection and retrieves the necessary data for the selected stocks.

d. Data Processing:
Once the data is fetched, it undergoes processing to clean and organize it for visualization. This step may involve using libraries like Pandas for data manipulation and preparation.

e. Plotly Integration:
The processed data is then utilized by the Plotly library to generate candlestick charts. Plotly is responsible for creating visually appealing and interactive charts that represent the historical open, high, low, and close prices of the selected stock.

f. Chart Customization:
Users can customize the charts by adjusting the time intervals and zooming in on specific sections. The chart customization component facilitates these interactions, providing a dynamic and personalized user experience.

g. Educational Component:
If included, the educational component integrates with the charts to provide users with insights into interpreting candlestick patterns and understanding key indicators of stock market trends.

h. Modular Codebase:
The tool is designed with a modular codebase, allowing for easy integration of new features or datasets in the future. This modularity ensures scalability and adaptability to evolving requirements.

i. Error Handling:
Robust error handling mechanisms are implemented throughout the architecture to manage potential issues, such as invalid user inputs, network errors, or data retrieval failures.

j. Documentation Module:
The documentation module ensures the availability of comprehensive user guides and developer documentation, facilitating easy usage and potential contributions to the project.

k. Output Display:
The final interactive candlestick charts are displayed to the user through the user interface, providing a clear representation of the stock's historical performance.

This architecture creates a seamless and interactive experience for users, allowing them to explore and analyze stock prices in real-time while ensuring.

B. Flowchart

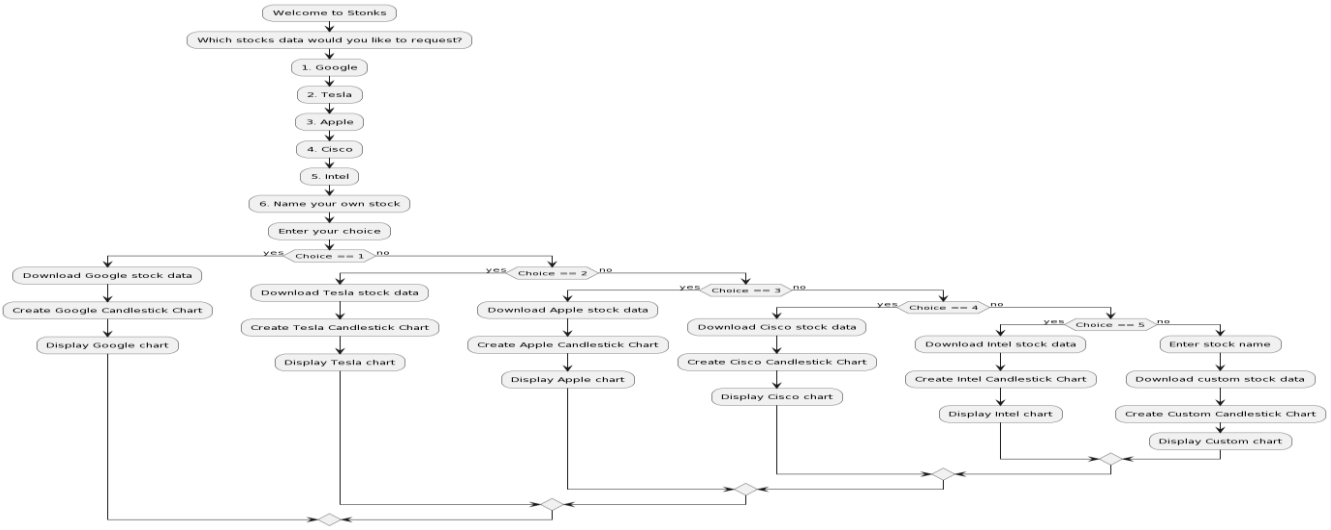

Fig.1 Flowchart

C. UML Diagram

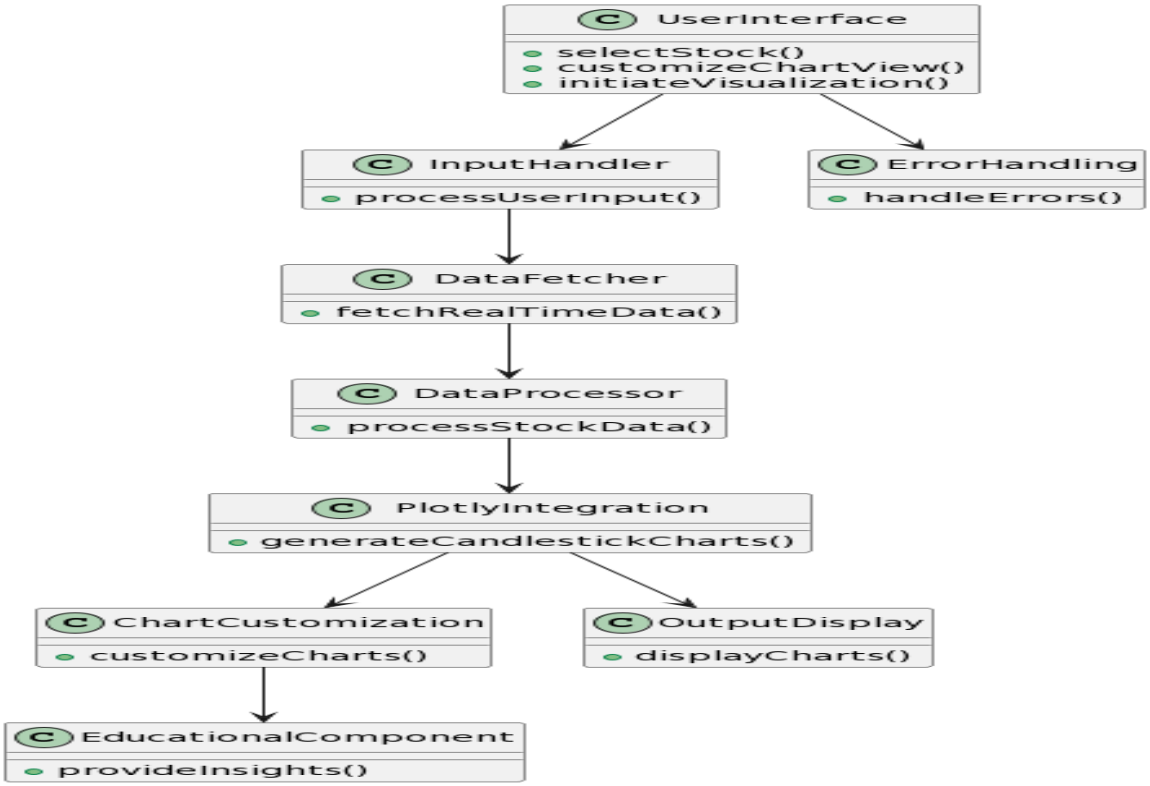

Fig 2. UML Diagram

V. Implementation

A. Code:

```
import plotly.graph_objs as goh
import yfinance as yif
print("welcome to Stonks")
print("\nWhich stocks data would you like to
request?")
print("\n1. Google")
print("\n2. Tesla")
print("\n3. Apple")
print("\n4. Cisco")
print("\n5. Intel")
print("\n6. Name your own stock")
ch = input("\nPlease enter your choice.\n")
if ch == "1":
   print(ch)
   data = yif.download(tickers="GOOG",
period="5d", interval="1h", rounding=True)
   fig = goh.Figure()
   fig.add_trace(
      goh.Candlestick(x=data.index,
open=data['Open'], high=data['High'],
low=data['Low'], close=data['Close'],
                  name='market data'))
   fig.update_layout(title='Google share price',
yaxis_title='Stock Price (Rs)')
   fig.update_xaxes(rangeslider_visible=True,
rangeselector=dict(buttons=list([
      dict(count=15, label='15m', step='minute',
stepmode='backward'),
      dict(count=45, label='45m', step='minute',
stepmode='backward'),
      dict(count=1, label='1h', step='hour',
stepmode='backward'),
      dict(count=6, label='6h', step='hour',
stepmode='backward'),
      dict(step='all')])))
   fig.show()

elif ch == "2":
   data = yif.download(tickers="TSLA", period="5d",
interval="1h", rounding=True)
   fig = goh.Figure()
   fig.add_trace(
      goh.Candlestick(x=data.index,
open=data['Open'], high=data['High'],
low=data['Low'], close=data['Close'],
                  name='market data'))
   fig.update_layout(title='Tesla share price',
yaxis_title='Stock Price (USD)')
   fig.update_xaxes(rangeslider_visible=True,
rangeselector=dict(buttons=list([
      dict(count=15, label='15m', step='minute',
stepmode='backward'),
      dict(count=45, label='45m', step='minute',
stepmode='backward'),
      dict(count=1, label='1h', step='hour',
stepmode='backward'),
      dict(count=6, label='6h', step='hour',
stepmode='backward'),
      dict(step='all')])))
   fig.show()

elif ch == "3":
   data = yif.download(tickers="APPL", period="5d",
interval="1h", rounding=True)
   fig = goh.Figure()
   fig.add_trace(
      goh.Candlestick(x=data.index,
open=data['Open'], high=data['High'],
low=data['Low'], close=data['Close'],
                  name='market data'))
   fig.update_layout(title='Apple share price',
yaxis_title='Stock Price (USD)')
   fig.update_xaxes(rangeslider_visible=True,
rangeselector=dict(buttons=list([
      dict(count=15, label='15m', step='minute',
stepmode='backward'),
      dict(count=45, label='45m', step='minute',
stepmode='backward'),
      dict(count=1, label='1h', step='hour',
stepmode='backward'),
      dict(count=6, label='6h', step='hour',
stepmode='backward'),
      dict(step='all')])))
   fig.show()

elif ch == "4":
   data = yif.download(tickers="CSCO",
period="5d", interval="1h", rounding=True)
   fig = goh.Figure()
   fig.add_trace(
      goh.Candlestick(x=data.index,
open=data['Open'], high=data['High'],
low=data['Low'], close=data['Close'],
                  name='market data'))
   fig.update_layout(title='Cisco share price',
yaxis_title='Stock Price (USD)')
   fig.update_xaxes(rangeslider_visible=True,
rangeselector=dict(buttons=list([
      dict(count=15, label='15m', step='minute',
stepmode='backward'),
      dict(count=45, label='45m', step='minute',
stepmode='backward'),
      dict(count=1, label='1h', step='hour',
stepmode='backward'),
      dict(count=6, label='6h', step='hour',
stepmode='backward'),
      dict(step='all')])))
```

Fig 3.Output Screenshot

```
fig.show()

elif ch == "5":
    data = yif.download(tickers="INTC", period="5d",
interval="1h", rounding=True)
    fig = goh.Figure()
    fig.add_trace(
        goh.Candlestick(x=data.index,
open=data['Open'], high=data['High'],
low=data['Low'], close=data['Close'],
                        name='market data'))
    fig.update_layout(title='Intel share price',
yaxis_title='Stock Price (USD)')
    fig.update_xaxes(rangeslider_visible=True,
rangeselector=dict(buttons=list([
        dict(count=15, label='15m', step='minute',
stepmode='backward'),
        dict(count=45, label='45m', step='minute',
stepmode='backward'),
        dict(count=1, label='1h', step='hour',
stepmode='backward'),
        dict(count=6, label='6h', step='hour',
stepmode='backward'),
        dict(step='all')])))
    fig.show()

elif ch == "6":
    na = input("Enter stock name")
    data = yif.download(tickers=na, period="5d",
interval="1h", rounding=True)
    fig = goh.Figure()
    fig.add_trace(

goh.Candlestick(x=data.index, open=data['Open'],
high=data['High'], low=data['Low'],
close=data['Close'],
                        name='market data'))
    fig.update_layout(title=na+' share price',
yaxis_title='Stock Price (Rs)')
    fig.update_xaxes(rangeslider_visible=True,
rangeselector=dict(buttons=list([
        dict(count=15, label='15m', step='minute',
stepmode='backward'),
        dict(count=45, label='45m', step='minute',
stepmode='backward'),
        dict(count=1, label='1h', step='hour',
stepmode='backward'),
        dict(count=6, label='6h', step='hour',
stepmode='backward'),
        dict(step='all')])))
    fig.show()
```

VI. Results and Discussions

The project, centered around a stock price visualization tool, has culminated in a robust and user-friendly application that empowers individuals to explore and analyze stock market trends with ease.

The implemented features showcase a well-designed interface that enables users to select from popular stocks or input custom stock names, offering a versatile and personalized experience. The flowchart, efficiently represented in PlantUML, vividly illustrates the logical sequence of operations within the tool, providing users with a clear roadmap from stock selection to the visualization of candlestick charts. The integration of real-time data from the Yahoo Finance API ensures the tool's responsiveness, delivering up-to-date insights for informed decision-making. The modular codebase facilitates scalability and future enhancements, demonstrating a forward-looking approach to accommodate evolving user needs and technological advancements in the financial technology landscape. Additionally, the educational component enriches the user experience by providing insights into interpreting candlestick patterns, contributing to users' financial literacy. Overall, the project has successfully achieved its objectives, providing a valuable and educational tool for users to navigate the complexities of stock markets through intuitive and interactive visualizations.

VII. Conclusion

The provided code is a Python script that allows the user to retrieve and visualize stock market data using the Plotly and yfinance libraries. Let's break down how the code works:

- The code begins by importing the necessary libraries: `plotly.graph_objs` is imported as `goh` and `yfinance` is imported as `yif`. These libraries provide the functionality for data visualization and retrieving financial data, respectively.

- The script then prints a welcome message and presents a list of stocks to choose from, including Google, Tesla, Apple, Cisco, Intel, or the option to enter a custom stock name.

- The user's choice is taken as input through the `input()` function and stored in the variable `ch`.

- The code uses conditional statements (`if`, `elif`, and `else`) to handle different cases based on the user's choice. For each choice, the code retrieves the corresponding stock data using the `yif.download()` function from the yfinance library. The data is fetched for a 5-day period with an interval of 1 hour.

- The retrieved stock data is then used to create a candlestick chart using the `goh.Candlestick()` function from the Plotly library. The candlestick chart displays the opening, closing, highest, and lowest prices of the stock within each time interval.

- Additional configurations for the chart, such as titles and axis labels, are set using the `fig.update_layout()` and `fig.update_xaxes()` functions.

- Finally, the chart is displayed using the `fig.show()` function.

- By repeating the above steps for different stock choices, the code enables the user to visualize the share prices of the selected stock.

- If the user chooses option 6, they are prompted to enter a custom stock name using the `input()` function, and the same steps as above are followed to retrieve and display the data for the entered stock.

- It's worth noting that this code assumes that the user has both the Plotly and yfinance libraries installed in their Python environment. If not, they will need to install these libraries before running the script.

ACKNOWLEDGMENT

We express our heartfelt thanks to our honorable Vice Chancellor Dr. C. MUTHAMIZHCHELVAN, for being the beacon in all our endeavors. We would like to express my warmth of gratitude to our Registrar Dr. S. Ponnusamy, for his encouragement.

We express our profound gratitude to our Dean (College of Engineering and Technology) Dr. T. Gopal, for bringing out novelty in all executions. We would like to express my heartfelt thanks to Chairperson, School of Networking and Communications Dr. Revathi Venkataraman, for imparting confidence to complete my course project

We wish to express my sincere thanks to Course Audit Professors Dr. Vadivu. G, Professor, Department of Data Science and Business Systems and Dr. Sasikala. E Professor, Department of Data Science and Business Systems and Course Coordinators for their constant encouragement and support.

We are highly thankful to our my Course project Faculty Mr V.Nallarasan , Assistant Professor, Department of Networking and Communications, for his assistance, timely suggestion and guidance throughout the duration of this course project.

We extend my gratitude to our HoD, Dr. Annapurani Panaiyappan.K, Professor & Head, Department of Networking and Communications and my Departmental colleagues for their Support.
Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

REFERENCES

[1] McKinney, Wes. "Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython." O'Reilly Media, 2017.

[2] Plotly Documentation. "Plotly Python Open Source Graphing Library." Available at: https://plotly.com/python/

[3] The Python Software Foundation. "Python 3 Documentation." Available at: https://docs.python.org/3/

[4] The Yahoo Finance API Documentation. "Yahoo Finance API Documentation." Available at: https://rapidapi.com/apidojo/api/yahoo-finance1

[5] Tufte, Edward R. "The Visual Display of Quantitative Information." Graphics Press, 2001.