DATA EXPLORATION AND VISUALIZATION

Unit 2

Content

Exploratory data analysis (EDA) techniques:

summary statistics
data visualization
outlier detection

Visualization tools and techniques for communicating insights:

charts,
Graphs
Dashboards

Interactive data exploration and drill-down capabilities

Best practices for designing effective data visualizations

WHAT IS EDA?

The analysis of datasets based on various numerical methods and graphical tools.

Exploring data for patterns, trends, underlying structure, deviations from the trend, anomalies and strange structures.

It facilitates discovering unexpected as well as conforming the expected.

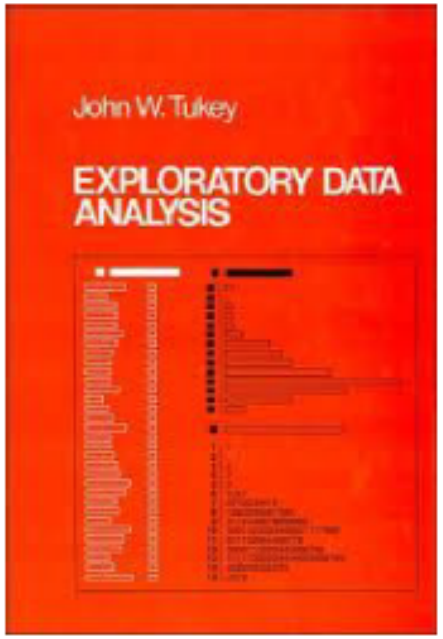
Another definition: An approach/ philosophy for data analysis that employs a variety of techniques (mostly graphical).

1977

By John W. Tukey

Exploratory Data Analysis 1977

- Based on insights developed at Bell Labs in the 60's
- Techniques for visualizing and summarizing data
- What can the data tell us? (in contrast to "confirmatory" data analysis)
- Introduced many basic techniques:
  - 5-number summary, box plots, stem and leaf diagrams,...
- 5 Number summary:
  - extremes (min and max)
  - median & quartiles
  - More robust to skewed & longtailed distributions



AIM OF THE EDA

1. Maximize insight into a dataset
2. Uncover underlying structure
3. Extract important variables
4. Detect outliers and anomalies
5. Test underlying assumptions
6. Develop valid models
7. Determine optimal factor settings (Xs)

Exploratory vs Confirmatory Data Analysis

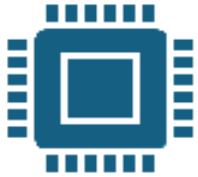
Feature	Exploratory Data Analysis (EDA)	Confirmatory Data Analysis (CDA)
Purpose	Discover patterns	Test hypotheses
Process	Flexible, open-ended	Structured, hypothesis-driven
Characteristics	Hypothesis-generating	Hypothesis-testing
Techniques	Descriptive statistics, visualization	Hypothesis testing, regression

EDA Steps in BIA

Data Ingestion and Streaming:

Continuous data flow: Set up pipelines to continuously ingest data from various sources (e.g., databases, APIs, IoT devices) in real time.

Data streaming platforms: Use technologies like Apache Kafka, Apache Flink, or AWS Kinesis to process and distribute data streams efficiently.



Real-Time Data Cleaning and Transformation:

Streaming data cleaning: Implement rules and algorithms to clean and preprocess data as it flows through the pipeline.

Data quality checks: Continuously monitor data quality metrics and trigger alerts for anomalies or inconsistencies.

Feature engineering: Apply feature engineering techniques on the fly to create relevant features for analysis.

Real-Time Data Exploration and Visualization:

Interactive dashboards: Develop dashboards that update in real time, allowing users to explore and visualize data as it arrives.

Visualization libraries: Leverage libraries like D3.js, Plotly, or Bokeh for creating dynamic and interactive visualizations.

Anomaly detection: Implement algorithms to detect unusual patterns or outliers in the data stream.

EDA Steps in BIA

Real-Time Hypothesis Testing and Alerting:

Hypothesis testing frameworks: Set up frameworks to continuously test hypotheses against incoming data.

Alerting systems: Trigger alerts when predefined conditions are met, such as significant changes in key metrics or anomalies.

Real-Time Insights and Decision-Making:

Automated insights: Use machine learning models to extract insights from the data stream automatically.

Decision support systems: Provide recommendations or suggestions based on the real-time analysis.

Integration with operational systems: Connect the real-time BI system with operational systems to enable immediate action.

Key Technologies and Tools:

Data streaming platforms:

Apache Kafka, Apache Flink, AWS Kinesis

Real-time databases:

Apache Cassandra, Apache Druid

Visualization libraries:

D3.js, Plotly, Bokeh

Machine learning frameworks:

TensorFlow, PyTorch, scikit-learn

Cloud platforms:

AWS, Azure, GCP

Challenges and Considerations

Latency:

Ensure low latency in data processing and visualization to enable timely decision-making.

Scalability:

Design the system to handle increasing data volumes and complexity.

Data quality:

Maintain data quality throughout the real-time pipeline.

Security:

Implement robust security measures to protect sensitive data.

Cost:

Consider the costs associated with hardware, software, and cloud infrastructure.

EXAMPLE 1

Data from the Places Rated Almanac *Boyer and Savageau, 1985)

9 variables fro 329 metropolitan areas in the USA

1. Climate mildness
2. Housing cost
3. Health care and environment
4. Crime
5. Transportation supply
6. Educational opportunities and effort
7. Arts and culture facilities
8. Recreational opportunities
9. Personal economic outlook

+ latitude and longitude of each city

Questions:

1. How is climate related to location?
2. Are there clusters in the data (excluding location)?
3. Are nearby cities similar?
4. Any relation bw economic outlook and crime?
5. What else???

EXAMPLE 3

In a project, investigating the well-being of teenagers after an economic hardship, main questions can be

Is there a positive (and significant) effect of economic problems on distress?

Which other factors can be most related to the distress of teenagers? e.g. age, gender,…?

EXAMPLE 4*

New cancer cases in the U.S. based on a cancer registry

- The rows in the registry are called observations they correspond to individuals
- The columns are variables or data fields they correspond to attributes of the individuals

Cancer Registry, 2001

ID	Sex	Age	Date	Site (ICD-9)
⋮	⋮	⋮	⋮	⋮
1023	M	58	1/1/01	1530
1024	F	69	1/2/01	1740
1025	F	29	1/2/01	1610
1026	M	46	1/4/01	1410
⋮	⋮	⋮	⋮	⋮

https://www.biostat.wisc.edu/~lindstro/2.EDA.9.10.pdf

Examples of Variables

Identifier(s):

- patient number,
- visit # or measurement date (if measured more than

Attributes at study start (baseline):

- enrollment date,
- demographics (age, BMI, etc.)
- prior disease history, labs, etc.
- assigned treatment or intervention group
- outcome variable

Attributes measured at subsequent times

- any variables that may change over time
- outcome variable

Data Types and Measurement Scales

Continuous Variables: (Quantitative, numeric).

- Continuous data can be rounded or \binned to create categorical data.

Categorical Variables: (Discrete, qualitative).

- Some categorical variables (e.g. counts) are sometimes treated as continuous.

Categorical Data

Unordered categorical data (nominal)

- 2 possible values (binary or dichotomous
  - Examples: gender, alive/dead, yes/no
- Greater than 2 possible values - No order to categorie
  - Examples: marital status, religion, country of birth, race.

Ordered categorical data (ordinal)

- Ratings or preferences
- Cancer stage
- Quality of life scales,
- National Cancer Institute's NCI Common Toxicity Criteria (severity grades 1-5)
- Number of copies of a recessive gene (0, 1 or 2)

Summary Statistics and Their Applications in BIA

Mean

- The average value of a dataset.
- Used to represent the central tendency of a dataset, especially when the data is normally distributed.
- For instance, in sales analysis, the mean sales per month can indicate the average performance of a sales team.

Median

- The middle value in a sorted dataset.
- Robust to outliers, making it suitable for skewed distributions.
- For example, in income analysis, the median income can better represent the typical income level, as it's less affected by extreme values like very high or very low incomes.

Mode

- The most frequent value in a dataset.
- Identifies the most common category or value in categorical data.
- In customer segmentation, the mode of a customer attribute (e.g., age group) can help identify the dominant segment.

Summary Statistics and Their Applications in BIA

Standard Deviation

Measures the spread of data around the mean.

Indicates the variability or dispersion of data.

A high standard deviation suggests a wider spread, while a low standard deviation indicates data points are clustered closer to the mean.

In quality control, standard deviation can help assess process variation.

Variance

The square of the standard deviation.

While less commonly used in direct analysis, variance is a key component of statistical tests and models.

Summary Statistics and Their Applications in BIA

Skewness

Measures the asymmetry of the distribution.

Identifies if the distribution is skewed to the left (negative skewness) or right (positive skewness).

In financial data, a right-skewed distribution (e.g., stock returns) often indicates the presence of outliers (e.g., extreme gains).

Kurtosis

Measures the "tailedness" of the distribution.

Helps identify if the distribution has heavy tails (leptokurtic) or light tails (platykurtic) compared to a normal distribution.

In risk management, kurtosis can help assess the likelihood of extreme events.

Percentiles

Indicate the values below which a certain percentage of the data lies.

Useful for understanding the distribution of data and identifying outliers.

For example, the 95th percentile of customer order values can help identify high-value customers.

Analyzing customer satisfaction data

Calculate the mean customer satisfaction rating

to get an overall sense of customer sentiment.

Examine the median rating

to understand the typical customer experience.

Identify the mode

to see if there are any particularly common satisfaction levels.

Analyze the standard deviation

to understand how much customer satisfaction varies.

Check for skewness

to see if there are a significant number of extremely satisfied or dissatisfied customers.

Data Visualization in (BIA)

Histograms:	Show the distribution of a single numerical variable.
Box Plots:	Visualize the distribution of a numerical variable, showing quartiles, median, and outliers.
Scatter Plots:	Display the relationship between two numerical variables.
Line Charts:	Plot the trend of a numerical variable over time.
Bar Charts:	Compare categorical data.
Pie Charts:	Represent proportions of a whole.
Heatmaps:	Visualize the relationship between two categorical variables.

Histograms

1

Collect Data:

- Gather customer order values, usually in a dataset where each row represents an order and a column holds the order value.

2

Choose Bins:

- Decide the number of bins (intervals) you want to divide your data into. This affects the granularity of your histogram.

3

Plot the Histogram:

- Use a tool like Matplotlib to plot the histogram, which shows the frequency of order values falling within each bin.

Code

```
import matplotlib.pyplot as plt

# Example data: List of customer order values
order_values = [50, 120, 300, 200, 150, 400, 500, 250, 100, 75, 80, 300, 275, 90, 110]

# Plotting the histogram
plt.figure(figsize=(10, 6))
plt.hist(order_values, bins=10, color='skyblue', edgecolor='black')

# Adding titles and labels
plt.title('Distribution of Customer Order Values')
plt.xlabel('Order Value')
plt.ylabel('Frequency')

# Display the plot
plt.show()
```

Explanation

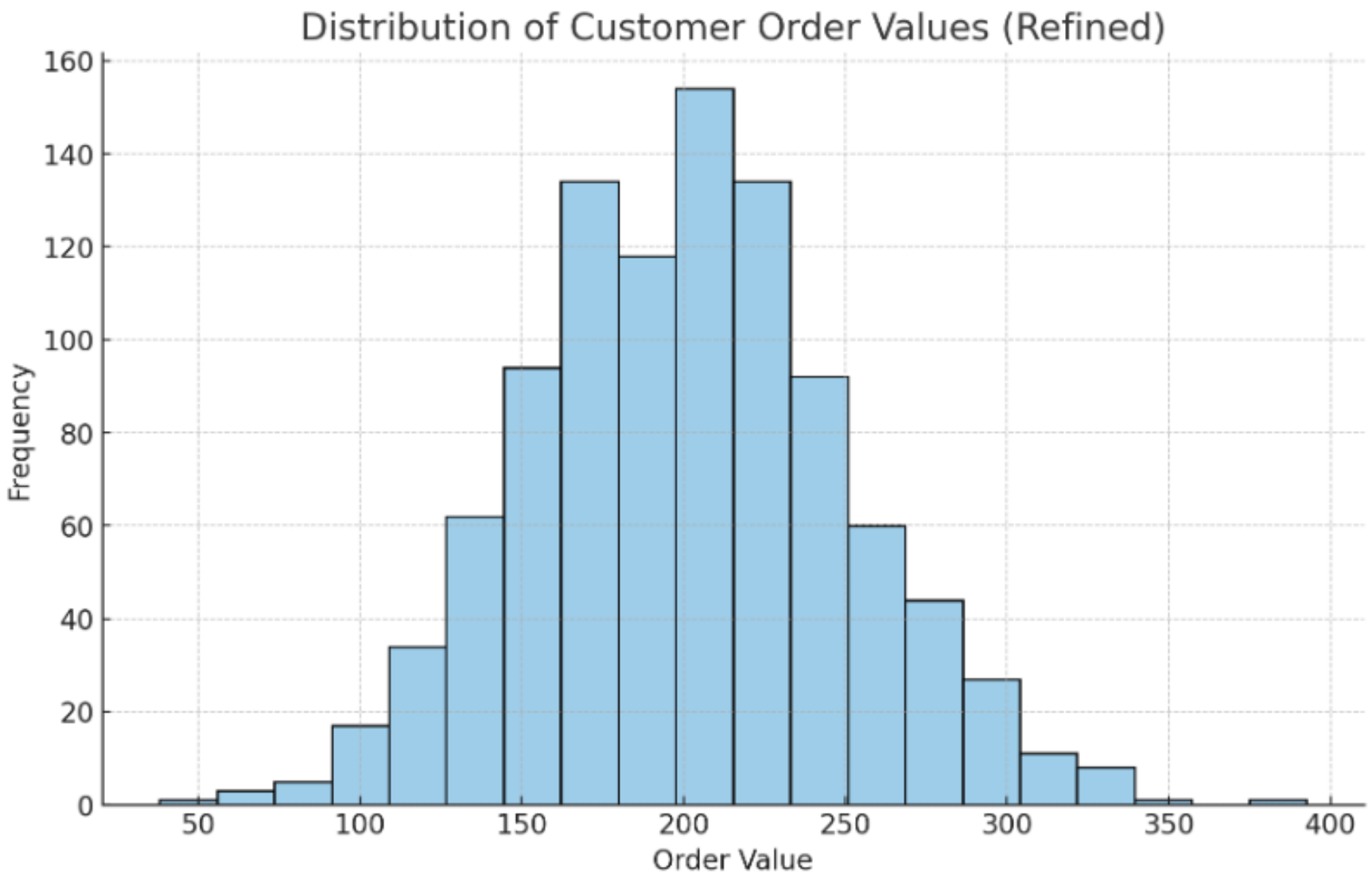
Bins: The histogram is divided into intervals or "bins" representing ranges of order values.

Frequency: The y-axis represents the number of orders within each bin.

Order Value: The x-axis represents the range of order values.

Customizations:

- You can adjust the number of bins for different levels of detail.
- Colors and labels can be customized to enhance readability.


Distribution of Customer Order Values (Refined)
Frequency
Order Value
0
20
40
60
80
100
120
140
160
50
100
150
200
250
300
350
400

Scatter plot

Collect Data: Gather data on marketing spend and corresponding sales revenue.

Plot the Scatter Plot: Use a tool like Matplotlib to create the scatter plot.

X-axis: Marketing spend in monetary units.

Y-axis: Sales revenue in monetary units.

Scatter Points: Each point represents the relationship between a specific marketing spend and the resulting sales revenue.

Code

```
import matplotlib.pyplot as plt
import numpy as np

# Example data: Random data for marketing spend and sales revenue
np.random.seed(42)
marketing_spend = np.random.normal(loc=5000, scale=1000, size=100)  # Mean=5000, SD=1000
sales_revenue = marketing_spend * 0.5 + np.random.normal(loc=0, scale=200, size=100)  # A linear relationship with noise

# Plotting the scatter plot
plt.figure(figsize=(10, 6))
plt.scatter(marketing_spend, sales_revenue, color='blue', alpha=0.6)

# Adding titles and labels
plt.title('Relationship Between Marketing Spend and Sales Revenue')
plt.xlabel('Marketing Spend')
plt.ylabel('Sales Revenue')

# Display the plot
plt.show()
```


Relationship Between Marketing Spend and Sales Revenue
Sales Revenue
Marketing Spend
3500
3000
2500
2000
1500
3000
4000
5000
6000
7000

Outlier Detection in (BIA)

Z-score:

Measures how many standard deviations a data point is from the mean.

Interquartile Range (IQR):

A measure of dispersion based on quartiles.

Isolation Forest:

An unsupervised anomaly detection algorithm.

One-Class SVM:

A supervised anomaly detection algorithm.

Z-score

The Z-score measures how many standard deviations a data point is from the mean of the dataset.

It is calculated using the formula:

- $Z=(X-\mu)/\sigma$

where X is the data point, μis the mean, and σ is the standard deviation.

Example

- Imagine you're analyzing customer purchase amounts to identify outliers.
- If the average purchase amount is $200 with a standard deviation of $50,
- a purchase of $400 would have a Z-score of: $Z=(400-200)/50=4$
- A Z-score of 4 indicates that the $400 purchase is 4 standard deviations away from the mean, making it a significant outlier.

Isolation Forest

The **Isolation Forest** method is an efficient algorithm for detecting outliers or anomalies in data.

It works by isolating points in the dataset.

Since anomalies are few and different, they are easier to isolate than normal data points.

This method works well for both small and large datasets.

Steps for Outlier Detection using Isolation Forest:

- **Fit the Isolation Forest model** on the dataset.
- **Predict anomalies** by assigning an anomaly score to each data point. Points with higher scores are more likely to be outliers.
- **Set a threshold** to label points as outliers (anomalies) based on their anomaly score.

Example code

```
# Step 1: Import necessary libraries
from sklearn.ensemble import IsolationForest
import numpy as np

# Step 2: Create some sample data (1D in this case, but it can be multi-dimensional)
data = [[10], [12], [14], [16], [18], [19], [120], [15], [17], [20], [13]]

# Step 3: Initialize Isolation Forest model
# 'contamination' parameter determines the proportion of outliers in the data (set to 0.1 for 10%)
iso_forest = IsolationForest(contamination=0.1, random_state=42)

# Step 4: Fit the model on the data
iso_forest.fit(data)
```

```
# Step 5: Predict outliers (-1 indicates an outlier, 1 indicates inlier)
predictions = iso_forest.predict(data)

# Step 6: Print the results
for i, point in enumerate(data):
    print(f"Data point: {point[0]}, Prediction: {'Outlier' if predictions[i] == -1 else 'Inlier'}")
```

```
Data point: 10, Prediction: Inlier
Data point: 12, Prediction: Inlier
Data point: 14, Prediction: Inlier
Data point: 16, Prediction: Inlier
Data point: 18, Prediction: Inlier
Data point: 19, Prediction: Inlier
Data point: 120, Prediction: Outlier
Data point: 15, Prediction: Inlier
Data point: 17, Prediction: Inlier
Data point: 20, Prediction: Inlier
Data point: 13, Prediction: Inlier
```

Explanation:

contamination: This parameter specifies the expected proportion of outliers in the dataset (e.g., 0.1 means we expect 10% of the data to be outliers).

fit: The model is fitted to the data.

predict: After fitting, each data point is predicted to be an inlier (1) or an outlier (-1).

Interactive data exploration and drill-down capabilities

Interactive data exploration and drill-down capabilities allow users to dynamically explore datasets, drill down into specific data points, and uncover insights through interactive dashboards, visualizations, and filtering mechanisms.

These capabilities are typically available in Business Intelligence (BI) tools and data analytics platforms, such as

- Tableau
- Power BI
- Python-based libraries like Plotly and Dash.

Key Features of Interactive Data Exploration and Drill-Down

Interactive Visualizations:

- Users can interact with charts and graphs by clicking, hovering, or zooming to see details. Visualizations update dynamically based on user inputs.

Filtering and Slicing:

- Users can apply filters (e.g., by time period, category, or location) to view specific subsets of data and drill down into a granular level of detail.

Drill-Down Capabilities:

- Drill-down functionality allows users to explore hierarchical data, starting from a high-level summary and drilling down into finer details (e.g., from country to city to specific store data).

Dynamic Dashboards:

- Dashboards aggregate multiple visualizations into a single view and allow users to manipulate data interactively. Changes in one visualization may automatically update others based on user actions.

Aggregations and Summarization:

- Users can view data at different levels of aggregation and summary statistics, such as average, sum, and count, and interactively switch between these views.

Data Filtering with Selectable Widgets:

- Use of dropdowns, sliders, and checkboxes to filter and explore specific data ranges or categories

Tools for Interactive Data Exploration and Drill-Down

Tableau:

- **Features**: Tableau provides interactive dashboards where users can filter, highlight, and drill down into data easily. It supports hierarchical views and advanced filtering.
- **Example**: Clicking on a sales region in a global sales dashboard will zoom into country-specific details, and further drill-down might reveal city-level sales figures.
- **Usage**: Ideal for non-technical users looking for fast, intuitive data exploration.

Power BI:

- **Features**: Power BI offers interactive filtering, slicers, and drill-through reports, allowing users to explore data at different granularities.
- **Example**: A report might allow users to drill from total revenue to regional sales and down to individual products.
- **Usage**: Works well for both technical and non-technical users and integrates tightly with Microsoft Excel.

Plotly + Dash (Python):

- **Features**: Plotly is a Python library for creating interactive visualizations, while Dash provides a web-based framework for building custom interactive dashboards.
- **Example**: Users can create interactive scatter plots where hovering over points shows additional information, or drill into details by dynamically filtering the dataset.
- **Usage**: Suitable for Python developers who need custom-built interactive data applications.

Tools for Interactive Data Exploration and Drill-Down

4. Jupyter Notebooks + ipywidgets:

- **Features**: Jupyter Notebooks combined with ipywidgets allow for interactive data exploration directly within Python notebooks. Users can create sliders, dropdowns, and buttons to interact with data.
- **Example**: An interactive widget in a notebook that allows users to filter and visualize data based on selected parameters, such as dates or categories.
- **Usage**: Ideal for data scientists and engineers working in Python, looking for quick exploration and analysis.

5. Google Data Studio:

- **Features**: Google Data Studio provides an easy-to-use interface for building interactive reports and dashboards. It supports drill-downs, interactive filters, and real-time data connections.
- **Example**: A user can create a dashboard with a global map, and clicking on a country reveals detailed sales data for that region.
- **Usage**: Best for Google Analytics and other Google data sources.

Example of Interactive Data Exploration with Plotly and Dash (Python)

```
import dash
from dash import dcc, html
import plotly.express as px
import pandas as pd

# Step 1: Prepare some sample data
df = px.data.gapminder()

# Step 2: Initialize the Dash app
app = dash.Dash(__name__)

# Step 3: Create the layout with dropdown for filtering and graph for visualizing
app.layout = html.Div([
    html.H1("Interactive Data Exploration with Drill-Down"),
    dcc.Dropdown(
        id='continent-dropdown',
        options=[{'label': c, 'value': c} for c in df['continent'].unique()],
        value='Asia',
        clearable=False
    ),
```

```
# Step 4: Define the callback function to update the plot dynamically
@app.callback(
    dash.dependencies.Output('scatter-plot', 'figure'),
    [dash.dependencies.Input('continent-dropdown', 'value')]
)
def update_graph(selected_continent):
    filtered_df = df[df['continent'] == selected_continent]
    fig = px.scatter(
        filtered_df, x="gdpPercap", y="lifeExp",
        size="pop", color="country", hover_name="country",
        log_x=True, size_max=60
    )
    return fig

# Step 5: Run the Dash app
if __name__ == '__main__':
    app.run_server(debug=True)
```

Explanation

Dropdown for filtering:

- The dropdown allows the user to select a continent, and the plot updates accordingly.

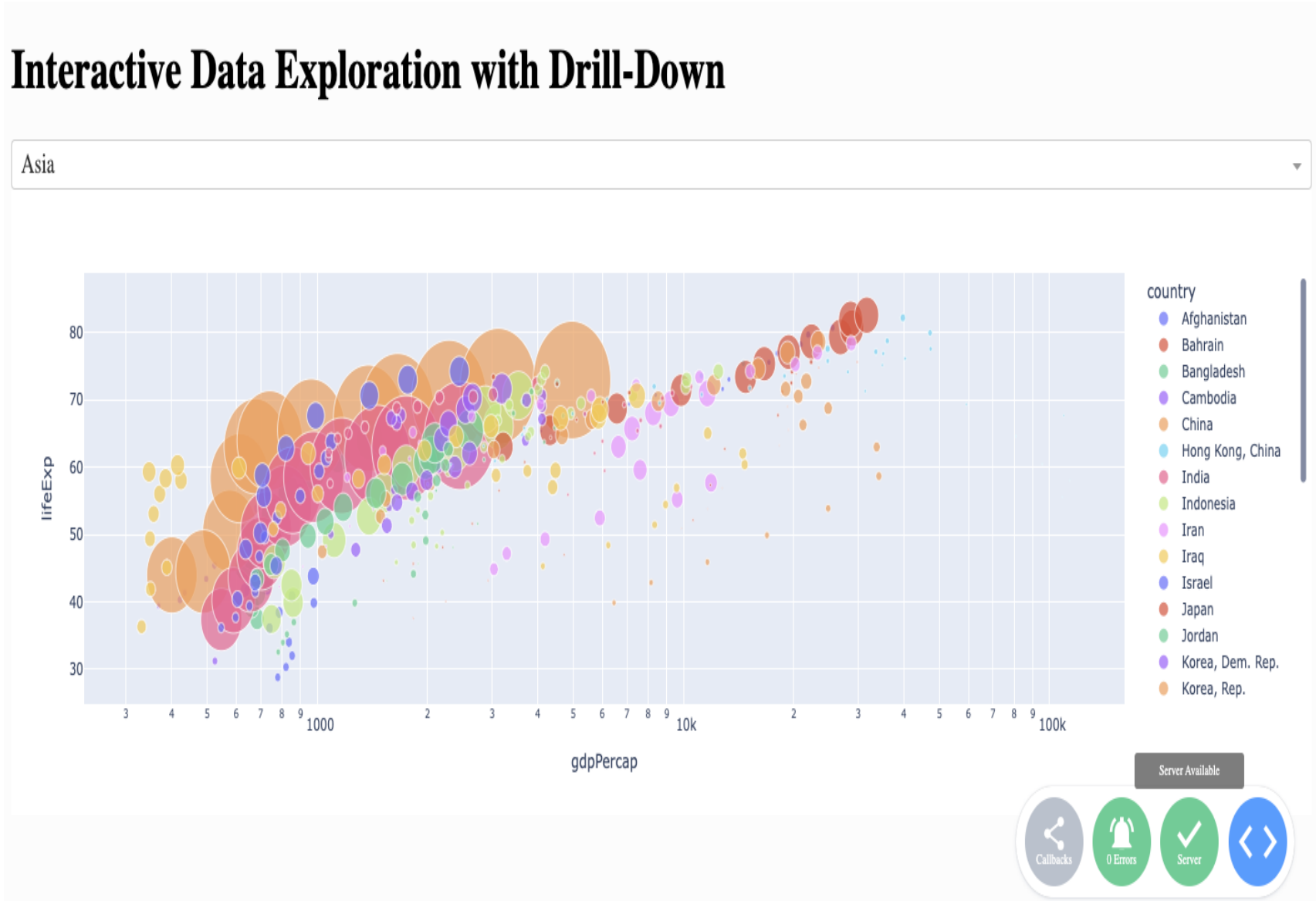
Scatter plot:

- The plot is generated using Plotly, and it visualizes GDP per capita vs. life expectancy, with each point representing a country.

Interactive Features:

- Hovering over points shows additional information, and selecting a different continent from the dropdown dynamically updates the graph.

Output


Interactive Data Exploration with Drill-Down
Asia
lifeExp
80
70
60
50
40
30
3 4 5 6 7 8 9 1000 2 3 4 5 6 7 8 9 10k 2 3 4 5 6 7 8 9 100k
gdpPercap
country
Afghanistan
Bahrain
Bangladesh
Cambodia
China
Hong Kong, China
India
Indonesia
Iran
Iraq
Israel
Japan
Jordan
Korea, Dem. Rep.
Korea, Rep.
Server Available
Callbacks
0 Errors
Server

Best Practices to ensure your visualizations




1. Know Your Audience

Tailor visualizations to the technical understanding and needs of the audience. For example, executives may prefer high-level summaries, while analysts may need detailed, granular data.

Consider **cultural context**, such as color meanings that might vary across regions or demographics.

2. Choose the Right Chart Type

Bar/Column charts: Ideal for comparing categories or showing discrete values.

Line charts: Best for showing trends over time.

Pie charts: Use sparingly, only when you need to show part-to-whole relationships with few categories.

Scatter plots: Effective for showing relationships between two continuous variables.

Heatmaps: Useful for showing data density or patterns across a matrix.

Always match the data type (categorical, time series, continuous) with the appropriate visualization format.

3. Simplify the Design

Avoid clutter: Remove unnecessary gridlines, borders, and visual effects (e.g., 3D elements) that don't add meaning.

Limit data points: Too many data points can make the visualization cluttered. Focus on key data and group similar points if necessary.

Use whitespace: Provide ample space around charts and between elements to avoid overwhelming the viewer.

4. Use Colors Effectively

Limit the color palette: Stick to 3-5 colors. Use consistent colors across charts and avoid using too many contrasting colors.

Use color for emphasis: Highlight key data points or trends by using a bolder or different color.

Be mindful of colorblindness: Use colorblind-friendly palettes (e.g., blue and orange instead of red and green) to make visualizations more accessible.

Ensure high contrast: Use contrasting colors for text and background to improve readability.

Best Practices to ensure your visualizations

- **5. Provide Context**
  - **Titles and labels**: Clearly label axes, data points, and provide a meaningful title that describes the key takeaway.
  - **Annotations**: Add notes or callouts to explain significant trends, outliers, or important insights.
  - **Legends**: If using multiple colors or line types, provide a clear legend to explain them.
  - **Units of measurement**: Always specify units (e.g., $, %, time) so that users can interpret values correctly.
- **6. Ensure Data Integrity**
  - **Avoid misleading scales**: Start axes at zero where possible to prevent exaggeration of trends. For example, truncating the y-axis can distort the magnitude of differences.
  - **Consistent intervals**: Use evenly spaced intervals on axes to ensure a uniform understanding of the data.
  - **Respect proportions**: Pie charts, bubble charts, or treemaps should represent values proportionally to avoid misrepresenting data.
- **7. Emphasize Key Insights**
  - **Highlight critical data**: Use color, size, or annotations to draw attention to key points or insights. For example, if a particular data point represents a milestone, it should stand out visually.
  - **Tell a story**: Arrange visualizations in a sequence that leads the viewer through the narrative of the data, helping them understand key trends or insights.
- **8. Interactive Visualizations**
  - **Allow filtering and drill-down**: In interactive dashboards, allow users to filter data, hover for more details, or drill down for finer levels of granularity.
  - **Tooltips**: Use hover-over tooltips to provide more information on specific data points without cluttering the main visualization.
  - **Responsive design**: Ensure that your visualizations are adaptable to different screen sizes and devices if they are part of a web or mobile application.

Best Practices to ensure your visualizations

- **9. Consider Data Density and Granularity**
  - **Avoid overloading**: Don't try to show too much data in a single visualization. If the dataset is large, consider breaking it down into smaller, more manageable views.
  - **Aggregation**: Sometimes showing aggregated data (e.g., averages, totals) helps users get the bigger picture without being overwhelmed by individual data points.
- **10. Iterate and Test**
  - **Get feedback**: Show your visualizations to colleagues or sample audience members to ensure clarity and effectiveness.
  - **Test readability**: Ensure that fonts, labels, and colors are easy to read and understand. If possible, use user testing or A/B testing to evaluate different designs.
  - **Update regularly**: If the data is time-sensitive or evolving, ensure that the visualizations stay relevant by updating them regularly.
- **11. Incorporate Consistency**
  - **Maintain uniform styles**: In dashboards or multiple charts, use consistent fonts, colors, and layouts for a coherent look and feel.
  - **Stick to the same scales**: If you're comparing similar data across charts, use the same axis scales to facilitate comparison.
- **12. Accessibility**
  - **Text alternatives**: If your visualization is part of a report or dashboard, ensure that alternative descriptions or text summaries are available for users who cannot interact directly with the visual.
  - **Keyboard navigation**: For interactive visualizations, ensure that they can be navigated with a keyboard, not just a mouse.