What is Data Visualization

What is Data Visualization?

Data visualization is the graphical representation of information and data using visual elements like charts, graphs, maps, and dashboards. It helps to make complex data more understandable and accessible.

Why is it important?

* **Simplifies complex data**
* **Identifies trends and patterns**
* **Supports decision-making**
* **Improves communication** – Helps in effectively communicating data insights to others, especially non-technical stakeholders.
* **Engages audience** – Interactive and visual formats keep the audience more engaged than raw numbers.

Challenges in Big Data Visualization

**Challenges in Big Data Visualization**

**Data Volume** – Big data involves massive datasets that can overwhelm traditional visualization tools, making it hard to process and display all data at once.

**Data Variety** – Big data comes in various formats (structured, semi-structured, unstructured), making it difficult to unify and visualize consistently.

**Real-Time Processing** – Visualizing data that updates in real-time (e.g., live dashboards) requires fast processing and efficient rendering, which can be technically challenging.

**Scalability** – Visualization tools must scale efficiently as data grows; otherwise, performance degrades and insights are delayed.

**Data Quality** – Inaccurate, incomplete, or inconsistent data can lead to misleading visualizations and incorrect interpretations.

**User Interpretation** – Designing visuals that are understandable to a wide range of users, including non-experts, can be difficult and may require balancing detail with simplicity.

Types of Data Visualization

Types of Data Visualization

<https://www.datalabsagency.com/2024/06/24/15-most-common-types-of-data-visualization/?v=0b3b97fa6688>

Data visualization can be categorized into various types based on the nature of the data and the insights one aims to derive. Below are four key types:

**1. Multidimensional Visualization**

Multidimensional visualizations represent data with more than two variables, enabling the exploration of complex relationships. These visualizations often use visual elements like position, size, color, and shape to encode multiple dimensions.

**Examples:**

* **Scatter Plot Matrix**: Displays pairwise relationships between multiple variables.
* **Parallel Coordinates Plot**: Each variable is represented by a parallel axis; data points are lines that intersect each axis at the corresponding value.
* **Bubble Chart**: An extension of the scatter plot where a third variable is represented by the size of the marker.

**2. Temporal Visualization**

Temporal visualizations are designed to represent data over time, highlighting trends, patterns, and changes. They are particularly useful for time-series data.

Examples:

* **Line Chart**: Plots data points connected by lines to show trends over intervals.
* **Gantt Chart**: Illustrates project schedules, showing tasks over time.
* **Timeline**: Displays events in chronological order, often used in historical data representation.

**3. Hierarchical Visualization**

Hierarchical visualizations depict data organized in a tree-like structure, representing parent-child relationships. They are effective for showing structures like organizational charts or file directories.

Examples:

* **Tree Diagram**: Displays the hierarchy with branches connecting nodes.
* **Treemap**: Uses nested rectangles to represent the hierarchy, where the size and color of each rectangle can denote additional variables.
* **Sunburst Chart**: A radial space-filling visualization similar to a treemap, but uses concentric circles to represent levels of the hierarchy

4. **Network Visualization**

Network visualizations are used to represent relationships and interactions between entities, often depicted as nodes (entities) and edges (connections). They are essential in analyzing social networks, communication patterns, and biological systems.

Examples:

* **Node-Link Diagram**: Shows nodes connected by lines, indicating relationships.
* **Force-Directed Graph**: Positions nodes based on simulated physical forces, often revealing clusters or communities.
* **Chord Diagram**: Displays inter-relationships between data in a matrix format, with arcs connecting related entities.

Data Presentation Techniques

Data Presentation Techniques

Data presentation techniques can be broadly categorized into graphical and non-graphical methods, with further subtypes under each.

**Graphical Methods**

Graphical methods involve visual representations of data, making it easier to identify patterns, trends, and outliers. They are widely used due to their intuitive appeal and effectiveness in communicating complex information.

Common graphical techniques include:

* Bar Charts: Used to compare quantities across different categories.
* Line Charts: Ideal for showing trends over time
* Pie Charts: Depict proportions of a whole.
* Histograms: Show frequency distributions of numerical data.
* Scatter Plots: Illustrate relationships between two variables.
* Heatmaps: Represent data values through color intensity.

These graphical methods are essential tools in data analysis and presentation, facilitating better understanding and decision-making.

**Non-Graphical Methods**

Non-graphical methods present data in textual or tabular formats without visual elements. They are useful for detailed data analysis and when precise values are required.

Common non-graphical techniques include:

* **Tables**: Organize data in rows and columns for easy lookup.
* **Descriptive Statistics**: Summarize data using measures like mean, median, mode, and standard deviation.
* **Frequency Distributions**: Show how often each value occurs in a dataset.
* **Stem-and-Leaf Plots**: Display quantitative data to retain original values while showing distribution.

Applications of Data Visualization

Applications of Data Visualization along with the use of the suitable plot

1. **Business Intelligence and Decision-Making**

Organizations utilize data visualization to interpret key performance indicators (KPIs), monitor sales trends, and analyze customer behaviors. Tools like dashboards and interactive charts enable stakeholders to quickly grasp insights, facilitating informed decision-making.

For instance, **a line chart** displaying monthly sales can help identify seasonal trends, while a **heatmap** can highlight regions with high customer engagement.

2. **Healthcare**

In the medical field, visualizations assist in tracking disease outbreaks, patient statistics, and treatment outcomes. Modern applications include dashboards monitoring cases, helping authorities allocate resources effectively.

**Line charts** are useful for showing trends over time, such as patient recovery rates or disease incidence.

3. **Financial Market Analysis**

Data visualization helps investors and analysts monitor stock prices, trading volumes, and market trends in real time.

Dashboards display key indicators like RSI, MACD, or moving averages, aiding in quick decision-making.

**Candlestick charts** show open, close, high, and low prices of stocks, making them perfect for technical analysis.

4. **Social Media and Web Analytics**

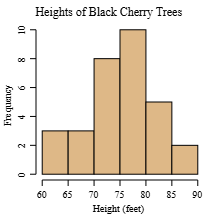
Marketers and web analysts use data visualization to track user engagement, traffic sources, and content performance.

Tools like **Google Analytics** and **social media dashboards** use graphs to show bounce rates, session durations, and user demographics.

**Heatmaps** highlight user interaction levels, such as clicks or scrolls on a webpage.

What is a histogram

What is a histogram? How is it used to visualize the distribution of data?

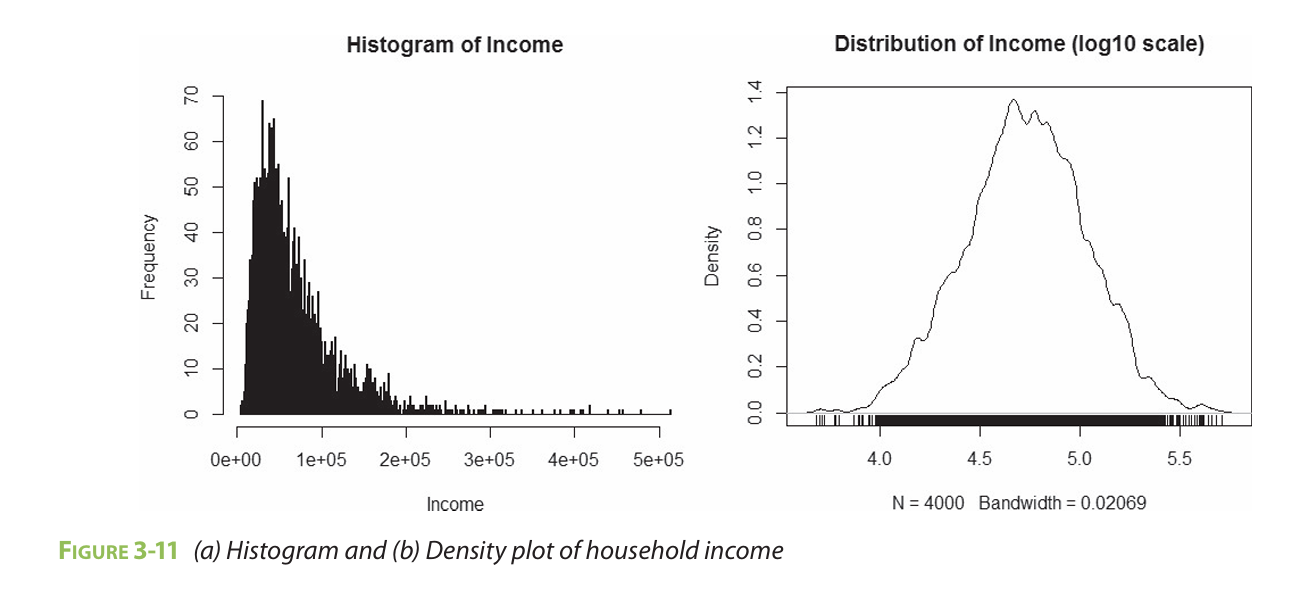


* A histogram is a graphical representation used to visualize the distribution of a continuous dataset.
* It is created by **dividing the entire range** of values into a series of **intervals or bins**, and then counting the number of observations that fall into each bin.
* These counts are represented as rectangular bars. The height of each bar shows the frequency (or relative frequency) of data points within that interval.
* Helps identify patterns, such as:

1. Central tendency (where most values lie)
2. Spread (how wide the data is distributed)
3. Skewness (asymmetry of the distribution)
4. Outliers (unusually high or low values)

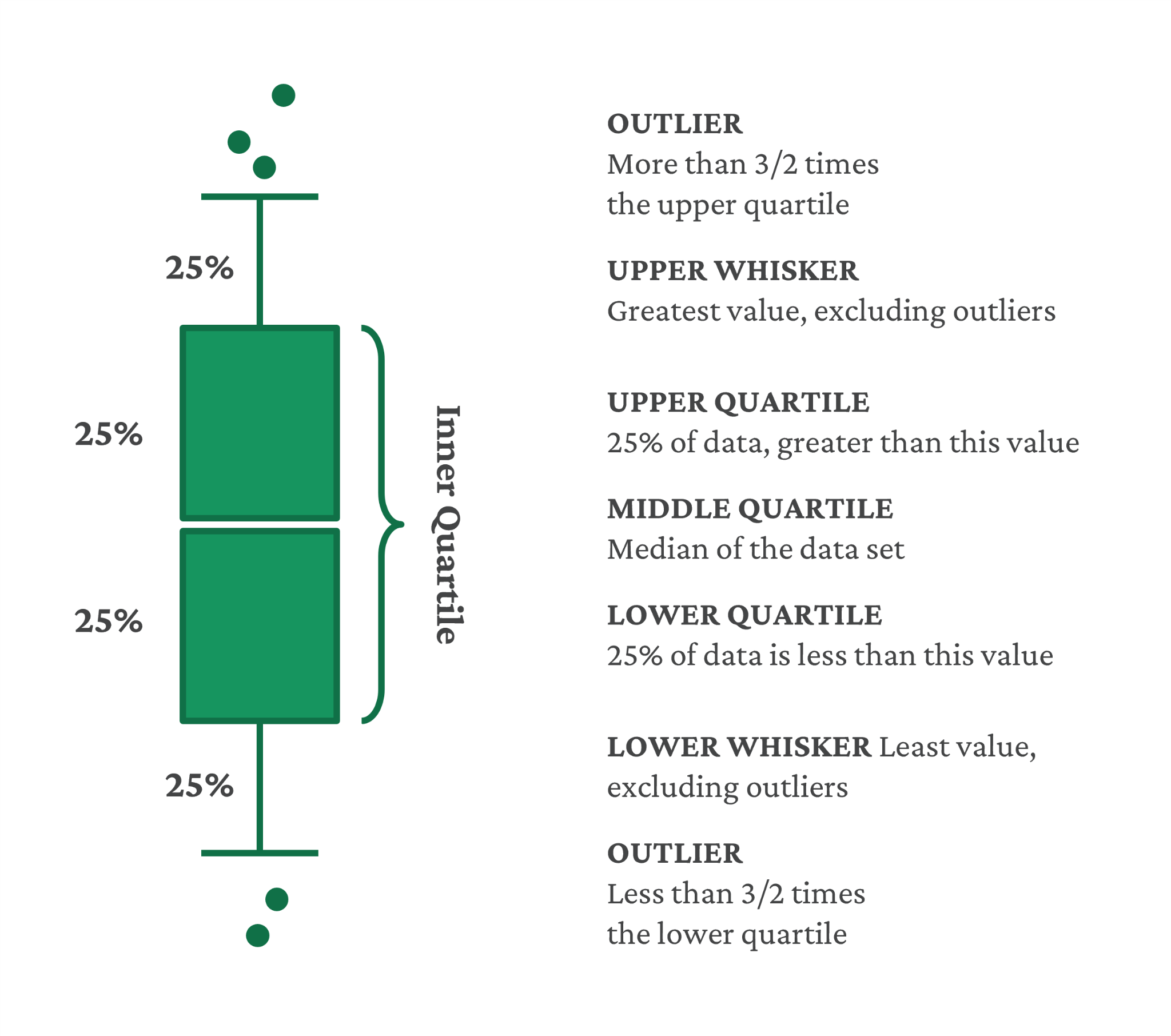
Difference Between Histogram and Density Plot

| **Feature** | **Histogram** | **Density Plot** |
| --- | --- | --- |
| **Definition** | A bar chart showing the **frequency** of data in bins | A **smoothed curve** that estimates the **probability density** of a dataset |
| **Appearance** | Discrete bars touching each other | Continuous smooth curve |
| **Binning** | Requires manually setting **bins (intervals)** | No fixed bins; uses **kernel smoothing** technique |
| **Interpretation** | Shows **count or frequency** of observations per bin | Shows the **likelihood (density)** of values over the range |
| **Scale (Y-axis)** | Represents **number of observations** in each bin | Represents **density**, not count – area under the curve = 1 |
| **Best Use Case** | Quick summary of **distribution shape and frequency** | More precise view of **distribution shape**, especially for large data |
| **Customization** | Bin width affects appearance significantly | Bandwidth (smoothing parameter) affects smoothness |



What is a box plot

What is a box plot? Explain the different components of a box plot?



Box and Whisker Plot is defined as a visual representation of the five-point summary. The Box and Whisker Plot is also called as **Box Plot**. It consists of a rectangular "box" and two "whiskers." Box and Whisker Plot contains the following parts:

* **Box:** The box in the plot spans from the first quartile (Q1) to the third quartile (Q3). This box contains the middle 50% of the data and represents the interquartile range (IQR). The width of the box provides insights into the data's spread.
* **Whiskers**: The whiskers extend from the minimum value to Q1 and from Q3 to the maximum value. They signify the range of the data, excluding potential outliers. The whiskers can vary in length, indicating the data's skewness or symmetry.
* **Median Line**: A line within the box represents the median (Q2). It divides the data into two halves, revealing the central tendency.
* **Outliers**: Individual data points lying beyond the whiskers are considered outliers and are often plotted as individual points.

The Interquartile Range (IQR) in a box plot represents the middle 50% of the data — that is, the range between the first quartile (Q1) and the third quartile (Q3): IQR = Q3 − Q1

Key Points About IQR:

* Q1 (25th percentile): 25% of the data falls below this value.
* Q3 (75th percentile): 75% of the data falls below this value.
* IQR: Measures the spread of the central data, excluding extreme values (outliers).

List of data visualization tools

List of data visualization tools.

Pentaho

* An open-source business intelligence suite that includes ETL, reporting, dashboards, and data mining.
* It’s Java-based data integration engine integrates with the Hadoop Map Reduce to for automatic deployment of tasks
* It provides various visualization features like

1. Drag-and-drop dashboard designer
2. Interactive charts (bar, line, pie, scatter, heat-map) via the Pentaho Analyzer
3. Extensible via plugins and community contributions

Datameer

* A Hadoop-native analytics platform designed to simplify big data preparation and visualization.
* It simplifies the integration, analysis, and visualization of large volumes of both structured and unstructured data.
* It offers a cost-effective, easy-to-use platform that enables users to derive insights from complex datasets
* Features:

1. Visual data profiling to identify outliers early in the analysis process.
2. Supports integration with multiple data sources (structured, semi-structured, unstructured).
3. Built on Apache Hadoop, which provides scalability and distributed processing.
4. End-to-end analytics: From data integration to visualization in one platform.

JasperReports

* An open-source Java reporting library, often embedded into Java applications to generate pixel-perfect reports.
* Visualization Features:

1. Chart components for bar, line, pie, area, and Gantt charts
2. Crosstabs and subreports for complex layouts
3. Export to PDF, HTML, Excel, and other formats

* Strengths:

1. Developer-friendly API for report generation from Java code
2. Integration with JasperReports Server for scheduling and distribution
3. Flexible layout control for static, printable reports

Dygraphs

* An open-source JavaScript charting library for highly interactive time-series visualizations.
* Visualization Features:

1. Pan and zoom on large datasets (hundreds of thousands of points)
2. Multiple axes, fill-between curves, and error bars
3. Annotation support for marking events on the graph

* Strengths:

1. Excellent performance for real-time or streaming data
2. Lightweight and embeddable in any web page
3. Customizable callbacks and plugins

Tableau

* A leading commercial BI and visualization platform known for its intuitive, drag-and-drop interface.
* Visualization Features:

1. Wide array of chart types: bar, line, pie, scatter, map, tree map, bullet, and more
2. “Show Me” recommendation engine suggests the best chart based on selected fields
3. Interactive dashboards with filters, parameters, and actions
4. Geospatial analytics with built-in map layers and custom shapefiles

* Strengths:

1. Extremely user-friendly for non-technical users
2. Fast in-memory data engine (Hyper) for rapid querying
3. Extensive connectivity: spreadsheets, databases, cloud services, and big-data platforms
4. Strong community and marketplace for extensions (Tableau Public, Tableau Exchange)

Tableau

Tableau

Tableau is a leading visual analytics platform that enables users to **connect to almost any data source**, **drag-and-drop** to create interactive visualizations, and **share** insights via dashboards and reports—all without coding

Its in-memory Hyper engine accelerates querying, while built-in AI features (Tableau Ask Data, Explain Data) and extensive connectivity (databases, spreadsheets, cloud services) empower both technical and non-technical users to explore data rapidly

**Working of Tableau:**

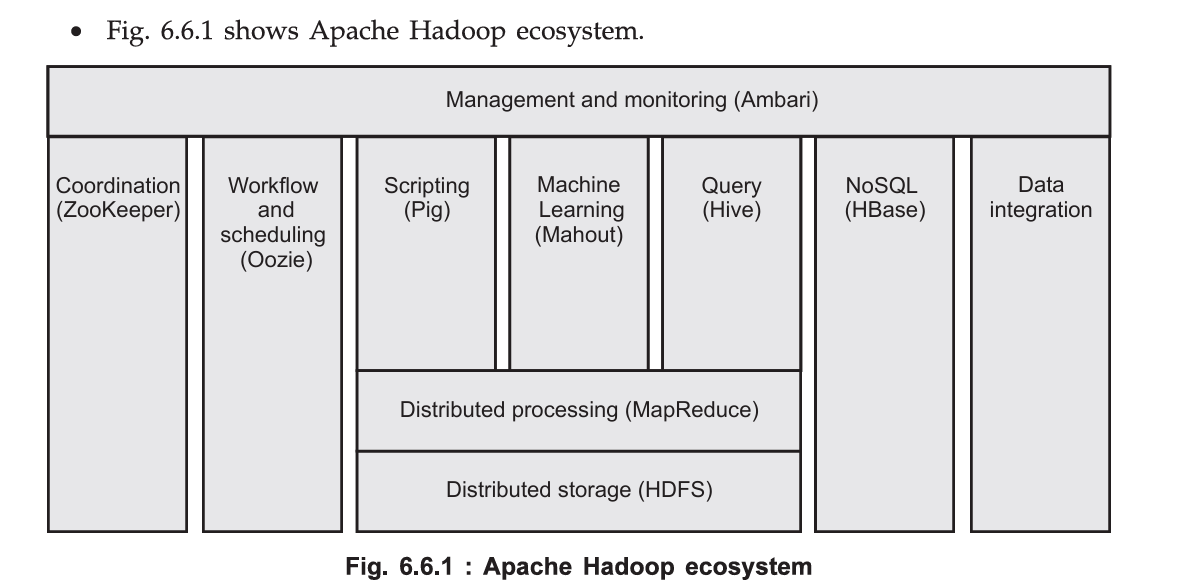
1. Tableau connects to data sources and extracts the necessary data.
2. It then processes this data in memory using its fast data engine and allows users to build visualizations through an intuitive interface.
3. Once the visuals or dashboards are created, they can be shared live through Tableau Server, Tableau Public, or on mobile and web platforms.
4. Tableau separates the data layer from the presentation layer, making it easier to update and maintain reports.

**Applications of Tableau (in brief):**

1. **Business Intelligence & Executive Dashboards**Organizations leverage Tableau to monitor KPIs (sales, revenue, churn), build executive dashboards, and drive data-driven decision-making
2. **Financial Market & Finance Analytics**Finance teams use Tableau to analyze P&L, cash flow, risk metrics, and forecast models. Prebuilt finance accelerators help departments break free from manual spreadsheets and deliver faster, governed reports
3. **Healthcare & Life Sciences Analytics**Healthcare providers and payers employ Tableau for patient outcomes, resource optimization, and treatment effectiveness dashboards—meeting strict HIPAA standards while unifying disparate data
4. **Social Media & Web Analytics**Marketers integrate Tableau with Google Analytics and social platforms to visualize traffic sources, engagement metrics, and campaign ROI. Heat maps reveal click and scroll behavior for UX improvements.
5. **Supply Chain & Operations**Manufacturers and retailers use Tableau to track inventory levels, logistics performance, and supplier metrics, enabling proactive interventions and cost reductions.
6. **Education & Research**Academic institutions and researchers create interactive visual stories—timelines, network graphs, and geographical mappings—to communicate findings and engage stakeholders

Hadoop Ecosystem

Hadoop Ecosystem



Apache Hadoop is an open-source framework designed for the distributed storage and large-scale processing of vast amounts of data across clusters of commodity hardware.

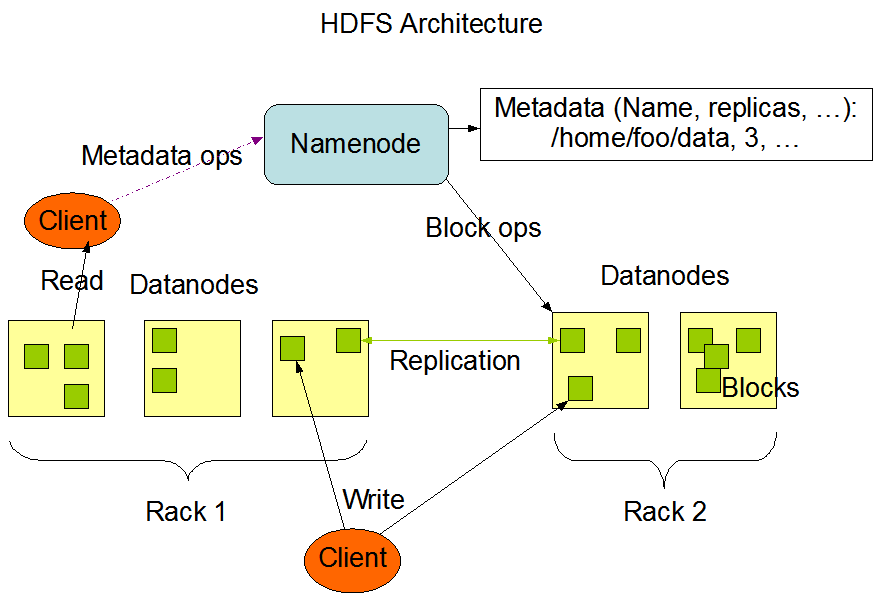
Hadoop ecosystem refers to the rich collection of open-source projects and tools that have grown up around the two core Hadoop components—**HDFS** (the distributed file system) and **MapReduce** (the batch‐processing engine)—to provide a complete, end-to-end big-data platform.

| **Component & Role** | **Description** |
| --- | --- |
| **HDFS (Hadoop Distributed File System)** | The fundamental storage layer. Splits files into blocks and replicates them across the cluster for fault tolerance and high throughput. |
| **MapReduce** | The core batch‐processing framework. Maps input data into key/value pairs, shuffles and sorts them, and reduces them to produce aggregated output. |
| **Apache Ambari** | A management and monitoring dashboard. Simplifies provisioning, managing, and monitoring of Hadoop services via a web UI and REST APIs. |
| **Apache ZooKeeper** | Provides distributed coordination: leader election, configuration management, and group services for other Hadoop components (e.g., HBase). |
| **Apache Oozie** | A workflow and job‐scheduler system. Defines complex job dependencies (MapReduce, Pig, Hive, Sqoop, etc.) as directed acyclic graphs (DAGs). |
| **Apache Pig** | A high‐level scripting language (Pig Latin) that compiles into MapReduce jobs, enabling data transformations without writing Java. |
| **Apache Mahout** | A scalable machine‐learning library. Provides algorithms for clustering, classification, and collaborative filtering running on Hadoop. |
| **Apache Hive** | A data warehousing layer. Offers SQL-like querying (HiveQL) on HDFS data, translating queries into MapReduce, Tez or Spark jobs. |
| **Apache HBase** | A NoSQL, column-oriented database built on HDFS. Provides real-time read/write access to large tables (suitable for random, sparse data). |
| **Data Integration** | Tools such as Sqoop (bulk import/export with RDBMS) and Flume (streaming ingest) that move data into and out of Hadoop. |
| **YARN (Yet Another Resource Negotiator)** | YARN is the resource manager of the Hadoop system.  It handles job scheduling, resource allocation, and monitors the cluster performance. |

HDFS and it’s architecture

HDFS and it’s architecture

HDFS is the primary storage system used by Hadoop applications. It is designed to store very large files (gigabytes to terabytes) across multiple machines in a reliable, fault-tolerant manner. By distributing data in chunks across a cluster of commodity servers, HDFS achieves both horizontal scalability and resilience to hardware failures.

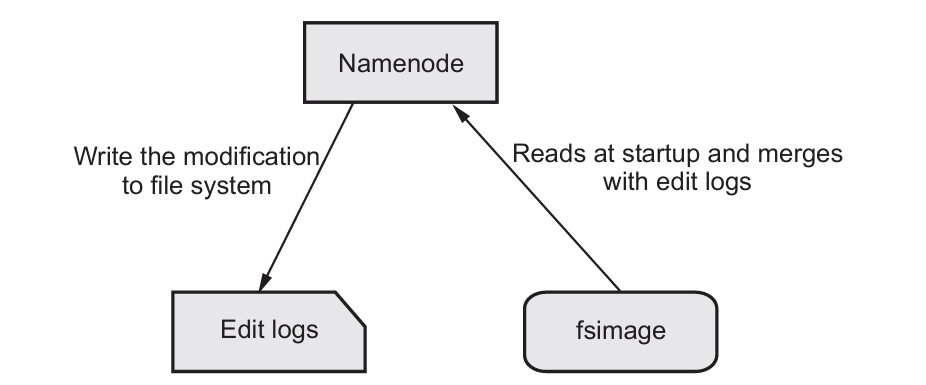


| **Component** | **Role** |
| --- | --- |
| **NameNode** | *Master server* that manages the filesystem namespace (directories, files) and metadata (block locations, permissions). |
| **DataNode** | *Worker servers* that actually store HDFS data blocks on local disk and serve read/write requests from clients. |
| **Secondary NameNode** | Periodically merges the NameNode’s edit-log with its checkpointed filesystem image (FSImage) to prevent the edit-log from growing without bound. |

The NameNode holds the directory tree and mapping of files → blocks → DataNodes, all kept in memory for fast access.

Metadata is persisted on disk as two files:

* FSImage: A checkpointed snapshot of the namespace.
* Edit Log: A log of recent filesystem changes.



Files are split into fixed-size blocks (default 128 MB).  
Each block is stored independently and replicated across multiple DataNodes.

**Write**: Client asks NameNode for DataNode block targets → streams data to the first DataNode → that node forwards to the next, and so on. → Once all replicas acknowledge, the write is confirmed.

**Read**: Client asks NameNode for block locations → reads data in parallel from the nearest DataNodes.

The Secondary NameNode’s job is to off-load the checkpointing work from the primary NameNode, keeping the metadata files lean and ensuring quicker restarts and consistent namespace snapshots

Map Reduce

Map Reduce

1. **MapReduce** is a programming model and processing framework designed to perform large-scale data processing on distributed clusters.
2. It was popularized by Google and implemented in Apache Hadoop to enable parallel computation over massive datasets.
3. MapReduce programs consist of two user-defined functions:

* **Map (k1, v1) → list(k2, v2)**

Processes each input record (key k1, value v1).

Emits zero or more intermediate key–value pairs (k2, v2).

* **Reduce (k2, list(v2)) → list(k3, v3)**

Receives each unique intermediate key k2 with the full list of its associated values.

Aggregates or processes those values to emit final output pairs (k3, v3).

1. Steps

**Input**: Raw data stored in HDFS serves as input to the MapReduce job.

**Split**: Data is split into chunks, each assigned to a different mapper.

**Map Phase**: Mapper processes the split and emits intermediate (key, value) pairs.

**Combine (optional)**:

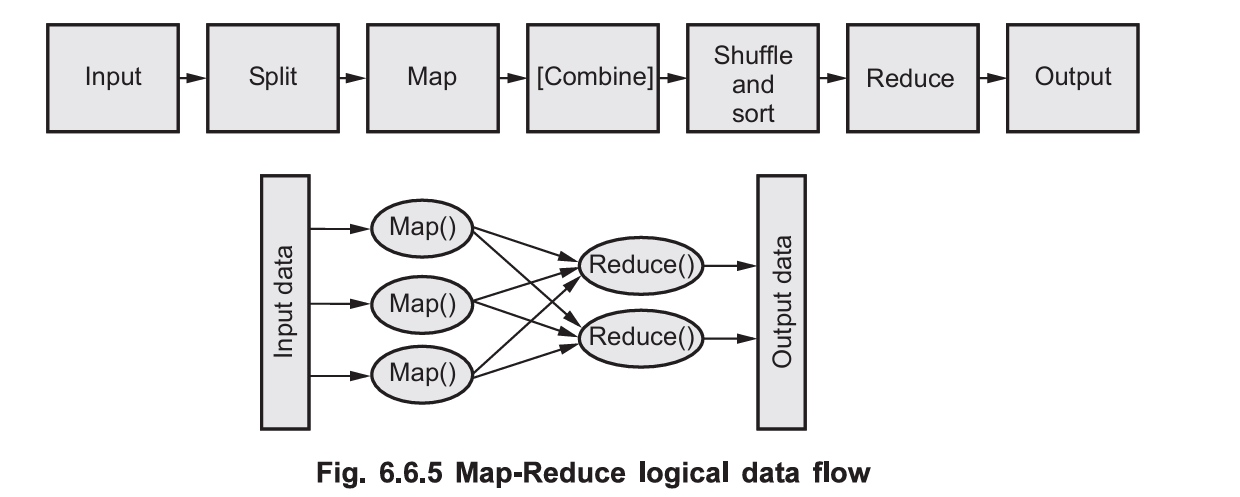
- Local aggregation on mapper to reduce data sent across the network.

**Shuffle & Sort**:

- Groups intermediate outputs by key and sends them to the appropriate reducers.

**Reduce Phase**:

- Reducer processes grouped values by key and generates final output.



Apache Pig

Apache Pig

1. **Apache Pig** is a high-level platform for creating programs that run on Apache Hadoop.
2. It simplifies big-data processing by providing a declarative language—**Pig Latin**—that abstracts away Java MapReduce coding.
3. **Client Layer**

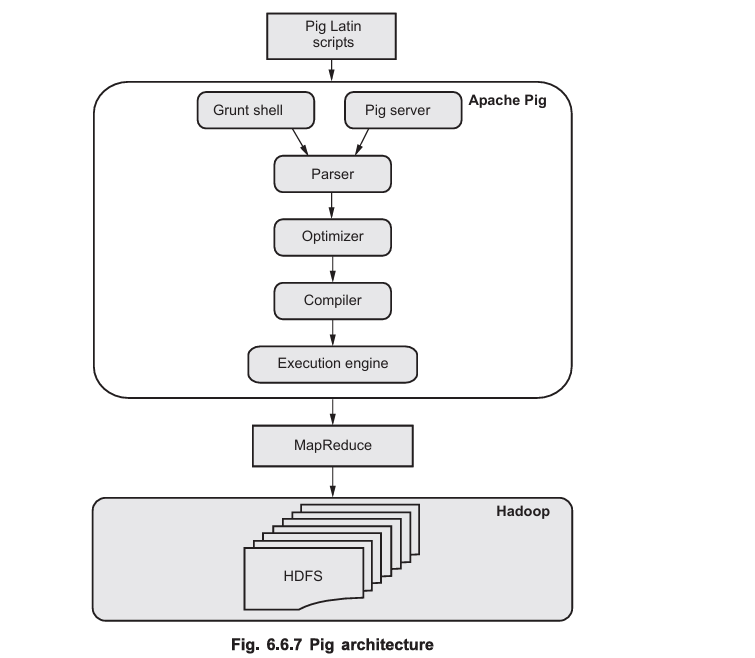
* Grunt Shell: Interactive command-line interface for entering Pig Latin statements.
* Scripts: Batch Pig Latin programs stored as .pig files.

1. **Pig Latin Processing Flow**

* Parser: Converts Pig Latin statements into a logical plan
* Optimizer: Applies rule-based optimizations to the logical plan.
* Compiler: Translates the optimized logical plan into a physical plan, then into one or more MapReduce jobs.
* Execution Engine: Submits jobs to Hadoop (or runs locally), monitors progress, and returns results.

1. **Execution Modes**

* Local Mode: Runs entirely on the local filesystem using a single JVM—ideal for development and small datasets.
* MapReduce Mode: Submits compiled jobs to a Hadoop cluster for distributed processing of large datasets.



Apache Hive

Apache Hive

1. Apache Hive is an open-source data warehouse infrastructure built on top of Hadoop.
2. It allows users to read, write, and manage large datasets stored in the Hadoop Distributed File System (HDFS) and other compatible storage systems like Apache HBase.
3. Hive provides a SQL-like query language called HiveQL (or HQL), making it easier for data analysts and developers familiar with SQL to interact with Hadoop data.
4. Features:

**Hive Metastore**: Stores metadata like table names, columns, data types, and storage details, separate from the actual data, which is typically stored in HDFS

**OLAP-oriented**: Designed for Online Analytical Processing (OLAP), not transaction-heavy tasks.

**HiveQL/HQL**: A query language similar to SQL used for data manipulation and querying. Hive queries (written in HiveQL) are compiled into MapReduce or Spark jobs depending on the execution engine configured.

**Multiple storage formats**: Supports TEXTFILE, SEQUENCEFILE, and RCFILE formats.

**Extensible and Scalable**: Can scale with the Hadoop cluster and be extended with custom UDFs (User Defined Functions).

1. Architecture

**Hive Driver**: Acts as the controller. It receives HiveQL queries from the user, manages the session, and handles query execution.

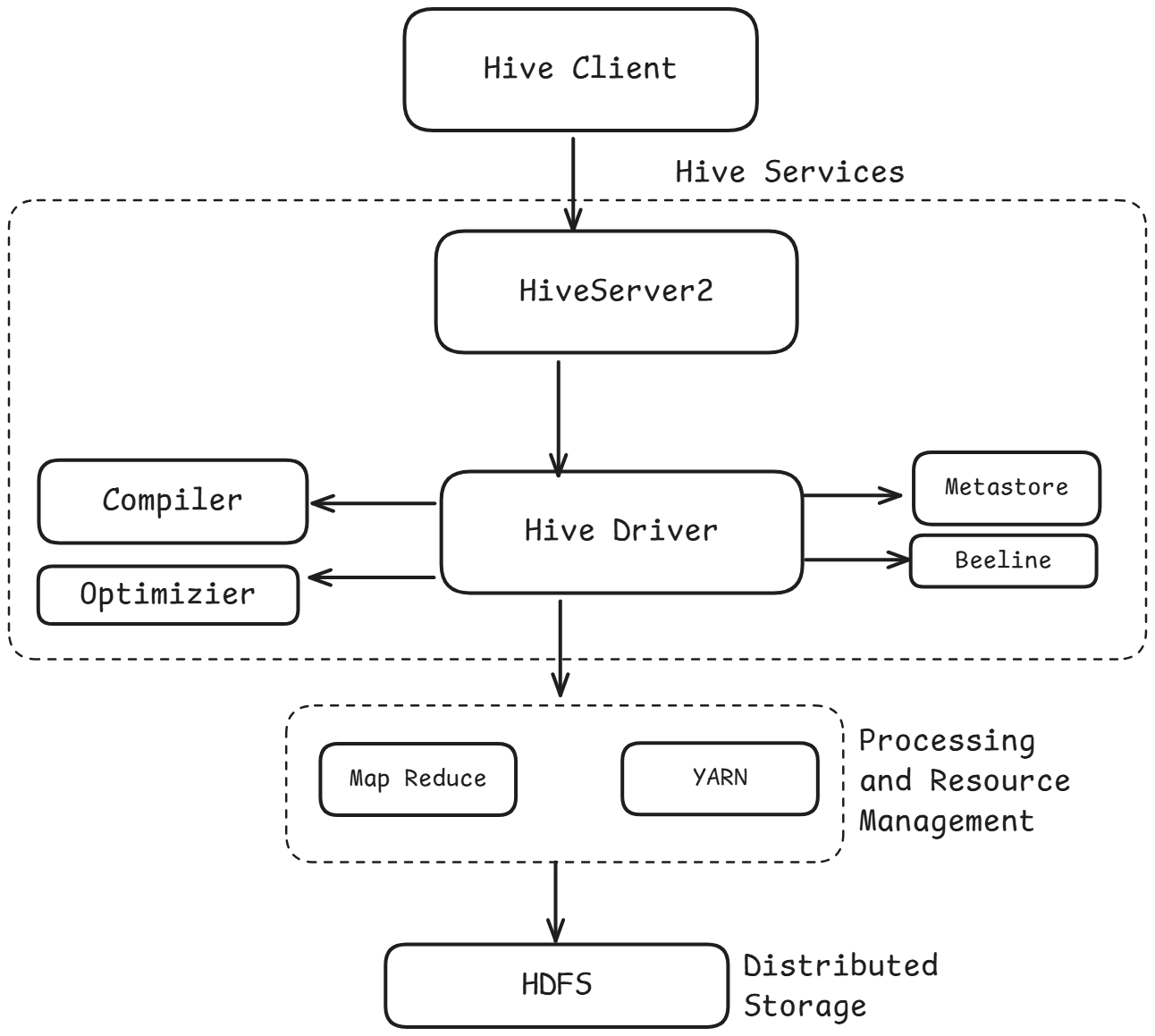
**Compiler**: Converts the HiveQL query into an execution plan.

**Optimizer**: Performs transformations on the execution plan such as predicate pushdown, join optimization, etc to improve efficiency.

**Metastore**: A central repository that stores metadata about Hive databases, tables, partitions, columns, and their data types. It is typically backed by an RDBMS like MySQL or PostgreSQL.

**Execution Engine**: Executes the query plan generated by the compiler. It runs the job across the Hadoop cluster.

**HDFS (or other storage)**: Actual data resides in Hadoop Distributed File System (HDFS). Hive reads and writes data from/to HDFS during query processing.



Apache Spark

Apache Spark

1. Apache Spark is an open-source, fast, and general-purpose in-memory data processing engine designed for big data analytics.
2. It was developed at UC Berkeley and later donated to the Apache Software Foundation.
3. Spark can handle batch, interactive, streaming, and machine learning workloads, making it more versatile than traditional MapReduce.
4. It complements Hadoop by enhancing the speed, flexibility, and ease of use of big data processing.

**How Spark Complements Hadoop:**

**Speed**:

Spark can be up to 100x faster than Hadoop MapReduce for certain workloads due to in-memory computing.

**Ease of Use:**

Spark uses high-level APIs (like DataFrames and Datasets) that simplify complex data workflows compared to verbose MapReduce code.

**Advanced Analytics:**

Spark provides built-in support for machine learning, streaming, and graph processing, which are not natively available in Hadoop MapReduce.

**Runs on Hadoop**:

Spark can run on top of Hadoop using YARN for resource management and HDFS for data storage. It can also integrate with other storage systems like HBase.

**Batch and Real-Time Processing:**

While Hadoop is more suited for batch processing, Spark supports both batch and stream processing, making it ideal for real-time analytics.

