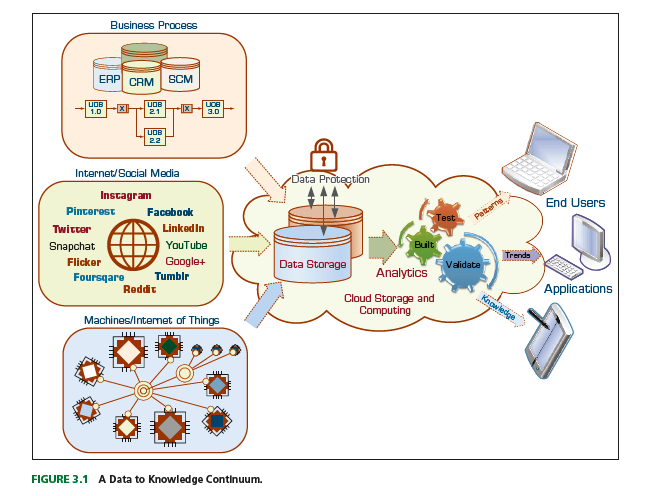
**MODULE 2**

**Chapter 1: Descriptive Analytics I**

**3.2 The Nature of Data in Analytics**

**Why data matters**

* Data is the **raw material** for BI/Analytics/AI: without reliable data, models and dashboards mislead.
* Decisions increasingly shift from **intuition → evidence**, so *data quality + governance* are foundational.



**Data lifecycle (end-to-end)**

1. **Create/Capture** (transactions, sensors, apps, web logs).
2. **Ingest** (batch files, CDC, APIs, streaming).
3. **Store** (RDBMS, data lake/object store, NoSQL, warehouse).
4. **Process/Transform** (ETL/ELT, feature engineering).
5. **Analyze/Model** (descriptive → prescriptive; ML/DL).
6. **Publish/Act** (reports, apps, automations).
7. **Monitor/Retain/Retire** (lineage, audits, archival, deletion).

**Structure & storage styles**

* **Structured** (tables): RDBMS/warehouse; strict schema; fast SQL.
* **Semi-structured** (JSON, XML, logs): document stores (MongoDB), data lakes.
* **Unstructured** (text, images, audio, video): lake/object storage; needs NLP/CV.
* **Specialized**:
  + **Columnar** (Parquet, columnar DBs) for analytics scans.
  + **Time-series** (InfluxDB, Prometheus) for metrics.
  + **Graph** (Neo4j) for networks/relationships.
  + **Key–value** (Redis) for caching/serving.

**Granularity, metadata, lineage**

* **Grain**: atomic (every transaction) vs aggregated (daily totals). Finer grain = flexibility, higher cost.
* **Metadata**: data about data (definitions, units, owner); critical for reuse.
* **Lineage/provenance**: how data moved/changed—enables trust, debugging, and compliance.

**Quality dimensions (with fixes)**

* **Accuracy** (validate against trusted source), **Completeness** (missing value rules),  
  **Consistency** (one truth across systems; MDM), **Timeliness** (freshness SLAs),  
  **Validity** (within ranges, legal values), **Uniqueness** (dedupe/keys).
* Techniques: profiling, constraints, reference tables, audit trails.

**Latency & access**

* **Batch** (hours/days), **micro-batch** (minutes), **real-time/streaming** (ms–seconds).
* Choose latency to match business need (e.g., fraud detection needs streaming).

**Governance, ethics, regulation**

* Data owners/stewards, policies, catalogs, access control.
* Privacy (consent, minimization), bias awareness, regulations (e.g., GDPR-style concepts).
* Secure by design: encryption, masking, role-based access, audit logs.

**3.3 A Simple Taxonomy of Data**

**By structure**

* Structured / Semi-structured / Unstructured (as above).

**By source & generation**

* **Human-generated** (forms, surveys, social posts).
* **Machine-generated** (sensors, IoT, clickstreams, app logs).
* **Internal** (ERP/CRM/SCM) vs **External** (open data, partners, social, purchased).

**By business role**

* **Transactional (OLTP)**: orders, payments (high volume, current state).
* **Analytical (OLAP)**: cleaned, conformed for analysis.
* **Master data**: core entities (Customer, Product) managed via MDM.
* **Reference data**: codes/standards (country, currency).
* **Metadata**: technical + business glossaries.

**By time horizon & dynamics**

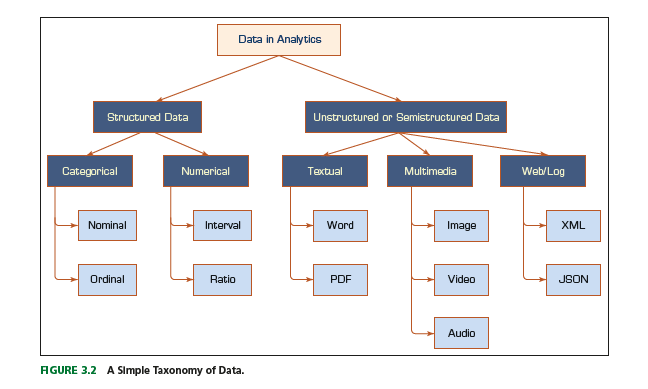
* **Historical** (archives), **near-real-time**, **streaming** (continuous).
* **Static** (rarely changes) vs **volatile** (fast-changing markets/prices).

**By ownership & openness**

* Proprietary, shared/partner, open/public datasets; also licensed data.

**By grain & sensitivity**

* Atomic vs summarized; **PII/PHI** vs anonymized/pseudonymized.

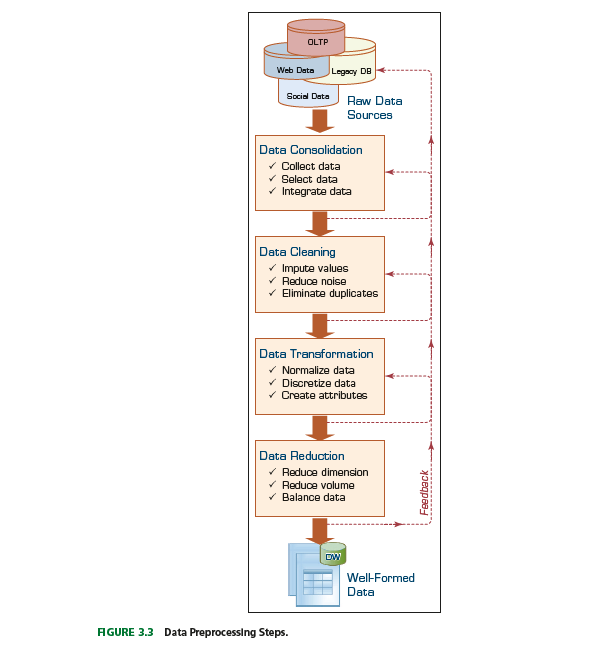
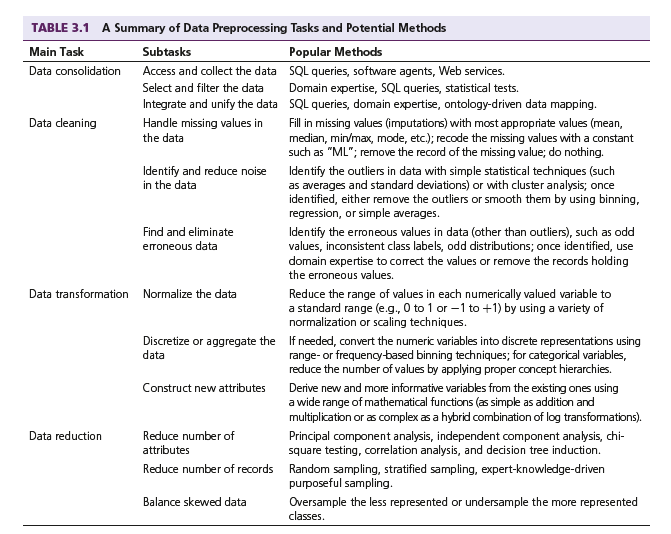


**3.4 The Art and Science of Data Preprocessing**

**Goal:** turn messy, heterogeneous raw data into high-signal, analysis-ready datasets.

**Typical pipeline**

1. **Data audit & sampling** (understand distributions, missingness, leakage risks).
2. **Split** into train/validation/test (or time-based splits for time series).
3. **Cleaning**
   * Missing values: listwise deletion (carefully), simple imputations (mean/median/mode), KNN/MICE, domain-specific rules.
   * Outliers: IQR/z-score rules, winsorization, robust models, isolation forest.
   * Noise: smoothing, aggregation, denoising (e.g., wavelets for signals).
   * Duplicates & identity resolution (fuzzy matching).
4. **Integration**
   * Join disparate sources; schema/semantic matching; dedupe across systems.
   * Unit harmonization (currency, time zones), key generation.
5. **Transformation & scaling**
   * Normalization/standardization (z-score, min–max, robust).
   * Log/Box-Cox for skew.
   * Categorical encoding: one-hot, ordinal, frequency/target encoding, hashing.
   * Time series: resampling, lag/lead, rolling stats, seasonality features.
   * Text: tokenization, stopword removal, stemming/lemmatization, n-grams; modern: sentence embeddings.
   * Images: resize, normalize, augmentation (flip/rotate/crop).
6. **Feature engineering**
   * Ratios, interactions, domain rules, percentiles, age/tenure, geospatial features.
7. **Reduction & selection**
   * Filter (variance, correlation, chi-square), wrapper (RFE), embedded (L1/Lasso).
   * Dimensionality reduction: **PCA** (modeling), **t-SNE/UMAP** (visualization).
8. **Imbalance handling**
   * Class weights, resampling (SMOTE/ADASYN), threshold tuning, cost-sensitive metrics.
9. **Validation & monitoring**
   * Data quality tests (expectations), drift checks, reproducible pipelines (DataOps/MLops).

****

**3.5 Definition of Big Data**

**Big data** = data whose **size/complexity/speed** exceed the capabilities of traditional tools to

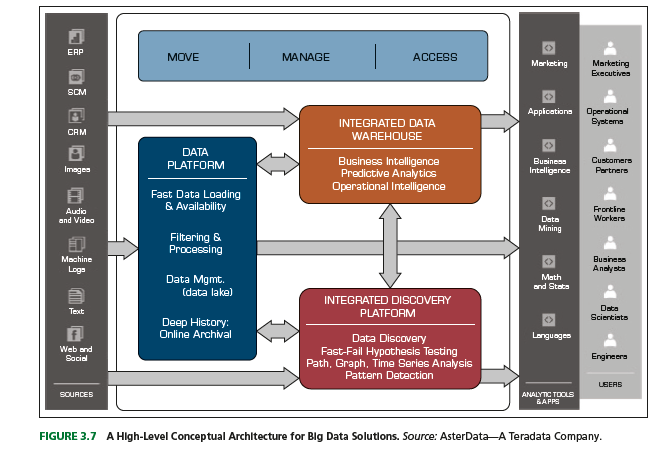
capture, store, process, and analyze economically.

**The “V’s” (with classroom examples)**

1. **Volume** – TB→PB→EB scale (clickstreams, HD video).
2. **Velocity** – streams that must be acted on *now* (fraud, IoT telemetry).
3. **Variety** – tables + logs + images + text + graphs.
4. **Veracity** – uncertainty/noise; need quality checks and probabilistic thinking.
5. **Value** – business outcomes (cost saved, revenue gained, risk reduced).  
   *(Often extended with Variability, Visualization, and Vulnerability for security.)*

**Technology Insights 3.1 – “Big, Bigger, Biggest”**

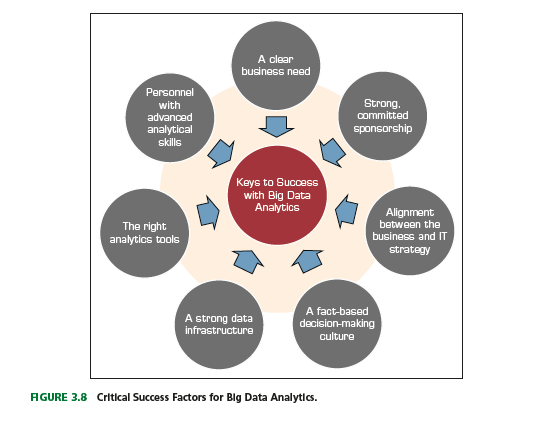
* **Drivers of growth:** smartphones, 4K/8K media, sensors/IoT, cloud apps, collaboration tools.
* **Why traditional RDBMS struggles:** rigid schema, vertical scaling limits, single-node I/O bottlenecks.
* **Modern stack (at a glance):**
  + Storage: HDFS / S3-style object stores (cheap, durable).
  + Compute: MapReduce (batch), **Spark** (fast in-memory), **Ray** (distributed Python).
  + Stream: **Kafka/Kinesis**; processing with **Flink/Spark Structured Streaming**.
  + SQL on lake: **Hive/Presto/Trino/Spark SQL**.
  + NoSQL: key-value (Redis), document (MongoDB), wide-column (Cassandra), graph (Neo4j).
  + Orchestration: Airflow, dbt; Infra: Kubernetes; Catalog/Lineage: Amundsen/DataHub.

****

**3.6 Fundamentals of Big Data Analytics**

**Core principles**

* **Scale-out, not up:** distribute data/compute across commodity nodes.
* **Bring compute to data:** minimize shuffles; pushdown predicates.
* **Schema-on-read (lakes) vs schema-on-write (warehouses):** flexibility vs reliability.
* **Separation of storage & compute** (cloud) for elastic cost control.



**Analytical workflows**

1. **Ingest**: CDC, streams, file drops; validate with checks.
2. **Store**: bronze (raw) → silver (clean) → gold (curated marts).
3. **Process**: Spark/SQL/dbt transforms; feature stores for ML reuse.
4. **Model**: scalable ML (Spark MLlib, xgboost-distributed, deep learning on GPUs).
5. **Serve**: APIs, dashboards, alerts, recommender services.
6. **Ops**: CI/CD for data & ML, drift/decay monitoring, A/B testing, feedback loops.

**Algorithms & patterns at scale**

* **Clustering** (k-means on Spark), **classification/regression** (GBT, logistic), **anomaly detection** (isolation forest), **recommendation** (ALS matrix factorization), **graph analytics** (PageRank, community detection).
* **Real-time**: sliding windows, CEP (complex event processing), online learning.

**Pitfalls & success factors**

* Pitfalls: schema drift, silent data failures, spurious correlations, training-serving skew, privacy violations.
* Success: business-owned use cases, strong governance, incremental delivery, cost observability, cross-functional teams.

**Business problems addressed (with data & value)**

1. **Personalization/Recommenders** (clickstream + catalog) → higher conversion/AOV.
2. **Fraud & Risk** (transactions + device/IP) → fewer chargebacks, safer credit.
3. **Predictive Maintenance** (sensor telemetry) → less downtime, lower spares.
4. **Supply Chain Optimization** (orders, lead times, weather) → reduced stockouts.
5. **Dynamic Pricing & Revenue Mgmt** (demand signals, competitors) → margin lift.
6. **Customer 360/Churn** (CRM, support, usage) → retention programs.
7. **Healthcare Analytics** (EHR, imaging, wearables) → earlier interventions.
8. **Smart Cities** (traffic sensors, public safety feeds) → congestion/crime reduction.
9. **Marketing Mix/Attribution** (ads, conversions, cost) → efficient spend.
10. **ESG & Risk Reporting** (energy, emissions, supplier data) → compliance & reputation.

**3.7 Big Data Technologies**

There are a number of technologies for processing and analyzing Big Data, but most have some common characteristics (Kelly, 2012).

Namely, they take advantage of commodity hardware to enable scale-out and parallel-processing techniques; employ nonrelational data storage capabilities to process unstructured and semi structured data; and apply advanced analytics and data visualization technology to Big Data to convey insights to end users.

The three Big Data technologies that stand out that most believe will transform the business analytics and data management markets are Hadoop, MapReduce, and NoSQL.

**Hadoop**

* Hadoop is an open-source framework for processing, storing, and analyzing massive amounts of distributed, unstructured data.
* It was designed to handle petabytes and exabytes of data distributed over multiple nodes in parallel.
* Hadoop clusters run on inexpensive commodity hardware so projects can scale-out without breaking the bank.
* Scales across commodity hardware, offers fault tolerance.

**How Does Hadoop Work?**

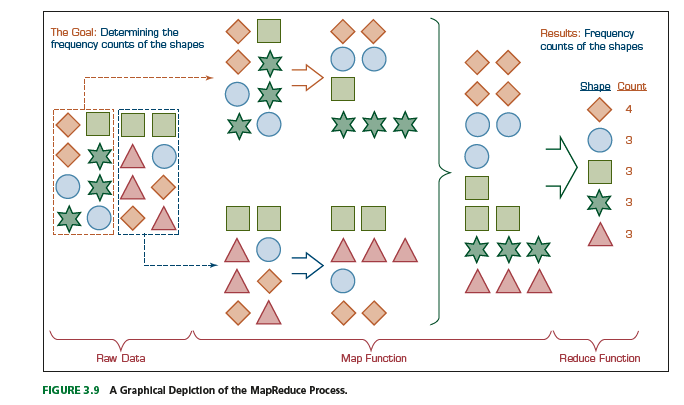
* Data split into blocks → stored across nodes (**HDFS**).
* File systems such as HDFS are adept at storing large volumes of unstructured and semi structured data as they do not require data to be organized into relational rows and columns.
* Each “part” is replicated multiple times and loaded into the file system so that if a node fails, another node has a copy of the data contained on the failed node.
* A Name Node acts as facilitator, communicating back to the client information such as which nodes are available, where in the cluster certain data resides, and which nodes have failed
* Jobs divided into tasks → executed in parallel.
* The client submits a “Map” job—usually a query written in Java—to one of the nodes in the cluster known as the Job Tracker.
* The Job Tracker refers to the Name Node to determine which data it needs to access to complete the job and where in the cluster that data is located.
* Once determined, the Job Tracker submits the query to the relevant nodes.
* Rather than bringing all the data back into a central location for processing, the processing occurs at each node simultaneously, or in parallel.
* When each node has finished processing its given job, it stores the results.
* The client initiates a “Reduce” job through the Job Tracker in which results of the map phase stored locally on individual nodes are aggregated to determine the “answer” to the original query, and then are loaded onto another node in the cluster.
* Resource manager allocates and monitors.

**MapReduce**

* **Map**: processes input data into key–value pairs.
* **Reduce**: aggregates intermediate results into final output.

**Why Use MapReduce?**

* Handles **large-scale datasets**.
* Provides fault tolerance & horizontal scalability.
* Cost-effective alternative to supercomputers.



**Hadoop Technical Components**

A Hadoop “stack” is made up of a number of components, which include the following:

**• *Hadoop Distributed File System (HDFS):*** The default storage layer in any given Hadoop cluster.

**• *Name Node:*** The node in a Hadoop cluster that provides the client information on where in the cluster particular data is stored and if any nodes fail.

**• *Secondary Node:*** A backup to the Name Node, it periodically replicates and stores data from the Name Node should it fail.

**• *Job Tracker:*** The node in a Hadoop cluster that initiates and coordinates MapReduce jobs or the processing of the data.

**• *Worker Nodes:*** The grunts of any Hadoop cluster, worker nodes store data and takedirection to process it from the Job Tracker.

***Hive:*** Hive is a Hadoop-based data warehousing–like framework originally developed by Facebook. It allows users to write queries in an SQL-like language called HiveQL, which are then converted to MapReduce. This allows SQL programmers with no MapReduce experience to use the warehouse and makes it easier to integrate with business intelligence (BI) and visualization tools such as Microstrategy, Tableau, Revolutions Analytics, and so forth.

***Pig:*** Pig is a Hadoop-based query language developed by Yahoo! It is relatively easy to learn and is adept at very deep, very long data pipelines (a limitation of SQL).

***HBase:*** HBase is a nonrelational database that allows for low-latency, quick lookups in Hadoop. It adds transactional capabilities to Hadoop, allowing users to conduct updates, inserts, and deletes. eBay and Facebook use HBase heavily.

***Flume:***Flume is a framework for populating Hadoop with data. Agents are populated throughout one’s IT infrastructure—inside Web servers, application servers, and mobile devices, for example—to collect data and integrate it into Hadoop.

***Oozie:*** Oozie is a workflow processing system that lets users define a series of jobs written in multiple languages—such as MapReduce, Pig, and Hive—and then intelligently link them to one another. Oozie allows users to specify, for example, that a particular query is only to be initiated after specified previous jobs on which it relies for data are completed.

***Ambari:*** Ambari is a Web-based set of tools for deploying, administering, and monitoring Apache Hadoop clusters. Its development is being led by engineers from Hortonworks, which includes Ambari in its Hortonworks Data Platform.

***Avro:*** Avro is a data serialization system that allows for encoding the schema of Hadoop files. It is adept at parsing data and performing removed procedure calls.

***Mahout:*** Mahout is a data mining library. It takes the most popular data mining algorithms for performing clustering, regression testing, and statistical modeling and implements them using the MapReduce model.

***Sqoop:*** Sqoop is a connectivity tool for moving data from non-Hadoop data stores—such as relational databases and data warehouses—into Hadoop. It allows users to specify the target location inside of Hadoop and instructs Sqoop to move data from Oracle, Teradata, or other relational databases to the target.

***HCatalog:*** HCatalog is a centralized metadata management and sharing service for Apache Hadoop. It allows for a unified view of all data in Hadoop clusters and allows diverse tools, including Pig and Hive, to process any data elements without needing to know physically where in the cluster the data is stored.

**Hadoop Pros and Cons**

* **Pros**: Scalable, open-source, flexible data handling, fault-tolerant.
* **Cons**: Batch-only, slower for real-time, complex setup.

**Technology Insight 3.2 – Demystifying Hadoop**

* Hadoop = **ecosystem**, not one product.
* Complements databases, not a replacement.
* Spark often runs on Hadoop clusters.

**Spark vs Hadoop**

* **Spark**: In-memory, faster, supports batch + streaming + ML.
* **Hadoop MapReduce**: Disk-based, batch only.
* Spark increasingly preferred for analytics.

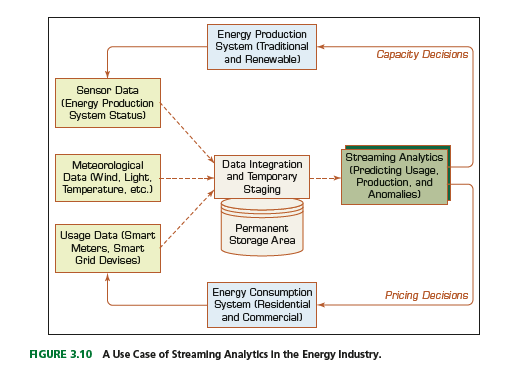
**NoSQL**

* Non-relational DBs designed for scalability and flexible schema.
* Types: Key–Value, Document, Column-family, Graph.
* Example: MongoDB, Cassandra, Redis, Neo4j.

**Data for Good**

* Using big data for **social good**:
  + Disaster relief (real-time mapping).
  + Healthcare (epidemic monitoring).
  + Environment (climate data).
  + Agriculture (yield optimization).

**3.8 Big Data and Stream Analytics**

****

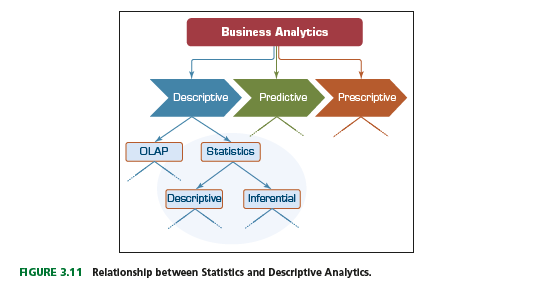
**Stream Analytics**

* Processing **continuous, real-time data streams**.
* Tools: Kafka, Flink, Spark Streaming, Storm.
* Supports event detection, windowing, and anomaly detection.

**Applications**

1. **e-Commerce** – Real-time product recommendations, fraud detection, dynamic pricing.
2. **Telecommunications** – Network monitoring, call-drop detection, real-time billing.
3. **Law Enforcement & Cybersecurity** – Intrusion detection, video surveillance, social media monitoring.
4. **Power Industry** – Smart grid monitoring, outage detection, load forecasting.
5. **Financial Services** – Algorithmic trading, real-time fraud prevention.
6. **Health Sciences** – IoT-enabled patient monitoring, early detection of emergencies.
7. **Government** – Smart cities, real-time traffic management, emergency services.

**3.9 Statistical Modeling for Business Analytics**

****

**Descriptive Statistics**

* Summarizes and describes data features.
* Includes measures of central tendency & dispersion.

**Measures of Central Tendency**

* **Mean** – arithmetic average.
* **Median** – middle value in ordered data.
* **Mode** – most frequent value.

**Measures of Dispersion**

* **Range** – max − min.
* **Variance** – average squared deviation from mean.
* **Standard Deviation** – square root of variance.
* **Mean Absolute Deviation (MAD)** – average absolute deviation.
* **Quartiles/IQR** – spread of middle 50%.

Quartiles help us identify spread within a subset of the data.

A quartile is a quarter of the number of data points given in a data set.

Quartiles are determined by first sorting the data and then splitting the sorted data into four disjoint smaller data sets.

Quartiles are a useful measure of dispersion because they are much less affected by outliers or a skewness in the data set than the equivalent measures in the whole data set.

Quartiles are often reported along with the median as the best choice of measure of dispersion and central tendency, respectively, when dealing with skewed and/or data with outliers.

A common way of expressing quartiles is as an interquartile range, which describes the

difference between the third quartile (Q3) and the first quartile (Q1), telling us about the

range of the middle half of the scores in the distribution.

The quartile-driven descriptive measures (both centrality and dispersion) are best explained with a popular plot called a box plot (or box-and-whiskers plot).

* **Box-and-Whiskers Plot** – visualization of distribution and outliers.

The box-and-whiskers plot (or simply a box plot) is a graphical illustration of several descriptive statistics about a given data set.

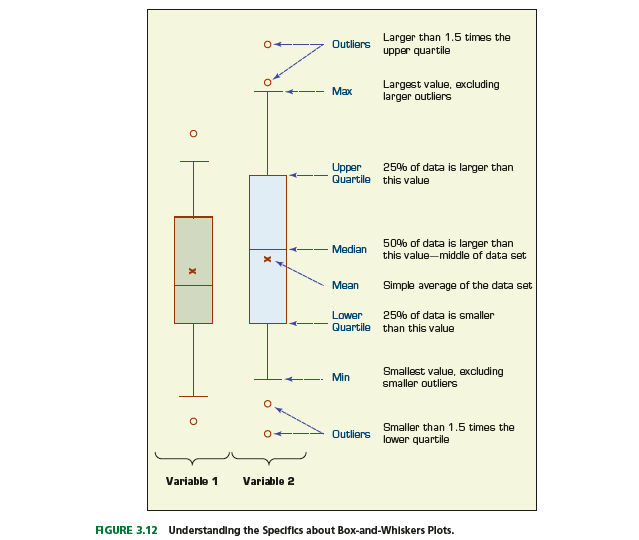
They can be either horizontal or vertical, but vertical is the most common representation, especially in modern-day analytics software products.

Box plot is often used to illustrate both centrality and dispersion of a given data set (i.e., the distribution of the sample data) in an easy-to-understand graphical notation.

The box plot shows the centrality (median, and sometimes also mean) as well as the dispersion (the density of the data within the middle half—drawn as a box between the first and third quartile), the minimum and maximum ranges (shown as extended lines from the box, looking like whiskers, that are calculated as 1.5 times the upper or lower end of the quartile box) along with the outliers that are larger than the limits of the whiskers.

A box plot also shows whether the data is symmetrically distributed with respect to the mean, or it sways one way or another.

The relative position of the median versus mean and the lengths of the whiskers on both side of the box give a good indication of the potential skewness in the data.



**Shape of a Distribution**

* **Normal (bell curve)**, skewed, uniform, bimodal.
* Shape affects choice of statistical methods.

Distribution is the frequency of data points counted and plotted over a small number of class labels or numerical ranges (i.e., bins).

In a graphical illustration of distribution, the y-axis shows the frequency (count or %), and the x-axis shows the individual classes or bins in a rank-ordered fashion.

A very well-known distribution is called normal distribution, which is perfectly symmetric on both sides of the mean and has several well-founded mathematical properties that make it a very useful statistical tool for research and practice.

As the dispersion of a data set increases, so does the standard deviation, and the shape of the distribution looks wider.

There are two commonly used measures to calculate the shape characteristics of a distribution: skewness and kurtosis. A histogram (i.e., frequency plot) is often used to visually illustrate both skewness and kurtosis.

**Skewness** is a measure of asymmetry (sway) in a distribution of the data that portrays a unimodal (i.e., only one peak exists in the distribution) structure of the data.

Because normal distribution is a perfectly symmetric unimodal distribution, it does not have skewness, that is, its skewness measure (i.e., the value of the coefficient of skewness) is equal to zero.

The skewness measure/value can be either positive or negative.

If the distribution sways left (i.e., the peak is on the left and the long tail is on the right side, and the mean is greater than median), then it produces a positive skewness measure, and if the distribution sways right (i.e., the peak is on the right and the long tail is on the left side, and the mean is smaller than median), then it produces a negative skewness measure.

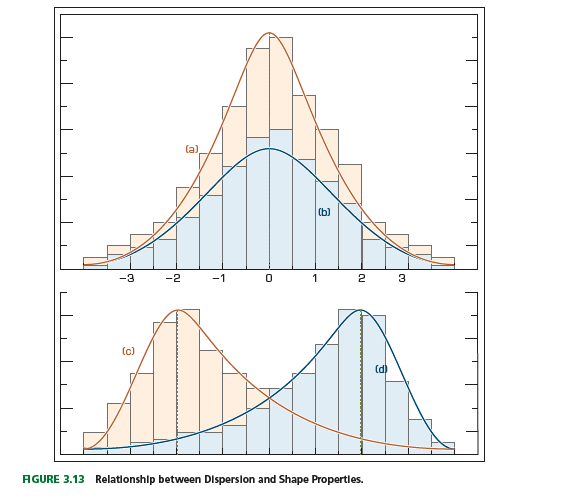
**Kurtosis** is another measure to use in characterizing the shape of a unimodal distribution.

As opposed to the sway in shape, kurtosis focuses on characterizing the peakiness/tallness nature of the distribution.

Specifically, kurtosis measures the degree to which a distribution is more or less peaked than a normal distribution.

Whereas a positive kurtosis indicates a relatively peaked/tall distribution, a negative kurtosis indicates a relatively flat/short distribution.

As a reference point, a normal distribution has a kurtosis of 3.



**3.10 Regression Modeling for Inferential Statistics**

Regression, especially linear regression, is perhaps the most widely known and used

analytics technique in statistics.

Regression has become the statistical technique for characterization of relationships between explanatory (input) variable(s) and response (output) variable(s).

As popular as it is, essentially, regression is a relatively simple statistical technique to model the dependence of a variable (response or output variable) on one (or more) explanatory (input) variables. Once identified, this relationship between the variables can be formally represented as a linear/additive function/equation.

Regression aims to capture the functional relationship between and among the characteristics of the real world and describe this relationship with a mathematical model, which may then be used to discover and understand the complexities of reality—explore and explain relationships or forecast future occurrences.

Regression can be used for one of two purposes: hypothesis testing—investigating potential relationships between different variables, and prediction/forecasting—estimating values of a response variables based on one or more explanatory variables. These two uses are not mutually exclusive. The explanatory power of regression is also the foundation of its prediction ability.

In hypothesis testing (theory building), regression analysis can reveal the existence/strength and the directions of relationships between a number of explanatory variables (often represented with xi ) and the response variable (often represented with y).

In prediction, regression identifies additive mathematical relationships (in the form of an equation) between one or more explanatory variables and a response variable. Once determined, this equation can be used to forecast the values of the response variable for a given set of values of the explanatory variables.

Correlation versus Regression: Correlation makes no a priori assumption of whether one variable is dependent on the other(s) and is not concerned with the relationship between variables; instead it gives an estimate on the degree of association between the variables.

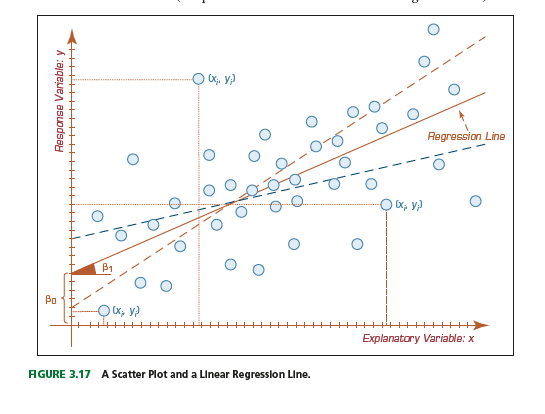
On the other hand, regression attempts to describe the dependence of a response variable on one (or more) explanatory variables where it implicitly assumes that there is a one-way causal effect from the explanatory variable(s) to the response variable, regardless of whether the path of effect is direct or indirect.

Also, although correlation is interested in the low-level relationships between two variables, regression is concerned with the relationships between all explanatory variables

Simple versus Multiple Regression: If the regression equation is built between one response variable and one explanatory variable, then it is called simple regression. For instance, the regression equation built to predict/explain the relationship between a height of a person (explanatory variable) and the weight of a person (response variable) is a good example of simple regression.

Multiple regression is the extension of simple regression where the explanatory variables are more than one. For instance, in the previous example, if we were to include not only the height of the person but also other personal characteristics (e.g., BMI, gender, ethnicity) to predict the weight of a person, then we would be performing multiple regression analysis.

In both cases, the relationship between the response variable and the explanatory variable(s) are linear and additive in nature.

****

**Linear Regression**

* Models relationship between dependent (Y) and independent (X) variables.

**How Do We Develop the Linear Regression Model?**

To understand the relationship between two variables, the simplest and most intuitive thing that one can do is to create a scatter plot, where the y-axis represents the values of the response variable, and the x-axis represents the values of the explanatory variable. A scatter plot would show the changes in the response variable as a function of the changes in the explanatory variable.

Simple regression analysis aims to find a mathematical representation of the relationship between two variables.

In reality, it tries to find the signature (i.e., algebraic representation) of a straight line passing through right in between the plotted dots (representing the observation/historical data) in such a way that the distance between the dots and the line is minimized (the predicted values on the theoretical regression line).

One of the most commonly used method is called the ordinary least squares (OLS) method. The OLS method aims to minimize the sum of squared residuals (squared vertical distances between the observation and the regression point) and leads to a mathematical expression for the estimated value of the regression line.

Equation: Y = β0 + β1X

In this equation, β0 ­ is called the intercept and β1 ­ is called the slope. Once OLS determines the values of these two coefficients, the simple equation can be used to forecast the values of y for given values of x. The sign and the value of β1 ­ also reveal the direction and the strengths of relationship between the two variables.

If the model is of a multiple linear regression type, then there would be more coefficients to be determined, one for each additional explanatory variable.

**How Do We Know If the Model Is Good Enough?**

Because of a variety of reasons, sometimes models as representations of the reality do not prove to be good. Regardless of the number of explanatory variables included, there is always a possibility of not having a good model, and therefore the linear regression model needs to be assessed for its fit (the degree at which it represents the response variable).

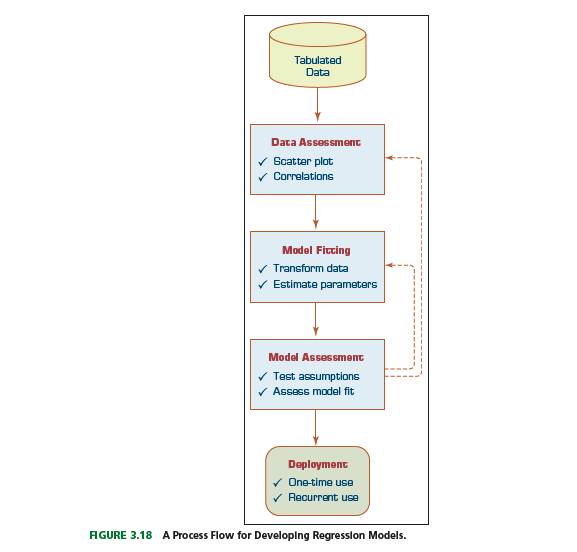
In the simplest sense, a well-fitting regression model results in predicted values close to the observed data values.

For the numerical assessment, three statistical measures are often used in evaluating the fit of a regression model. *R*2 (*R*-squared), the overall F-test, and the root mean square error (RMSE).

All three of these measures are based on the sums of the square errors (how far the data are from the mean and how far the data are from the model’s predicted values).

Different combinations of these two values provide different information about how the regression model compares to the mean model.

Of the three, *R*2 has the most useful and understandable meaning because of its intuitive scale. The value of *R*2 ranges from zero to one (corresponding to the amount of variability explained in percentage) with zero indicating that the relationship and the prediction power of the proposed model is not good, and one indicating that the proposed model is a perfect fit that produces exact predictions (which is almost never the case).

The improvement in the regression model can be achieved by adding mode explanatory variables, taking some of the variables out of the model, or using different data transformation techniques, which would result in comparative increases in an *R*2 value. 

**Model Development**

1. Collect data.
2. Estimate coefficients (least squares).
3. Test significance.
4. Validate with residuals and R².

**Model Evaluation**

* **Goodness of Fit** – R², Adjusted R².
* **Residual Analysis** – errors should be random.
* **F-test/t-test** – check coefficient significance.

**Assumptions of Linear Regression**

* Linearity, independence, homoscedasticity, normality of errors.

Linearity. This assumption states that the relationship between the response variable and the explanatory variables are linear. That is, the expected value of the response variable is a straight-line function of each explanatory variable, while holding all other explanatory variables fixed. Also, the slope of the line does not depend on the values of the other variables. It also implies that the effects of different explanatory variables on the expected value of the response variable are additive in nature.

Independence (of errors). This assumption states that the errors of the response variable are uncorrelated with each other. This independence of the errors is weaker than actual statistical independence, which is a stronger condition and is often not needed for linear regression analysis.

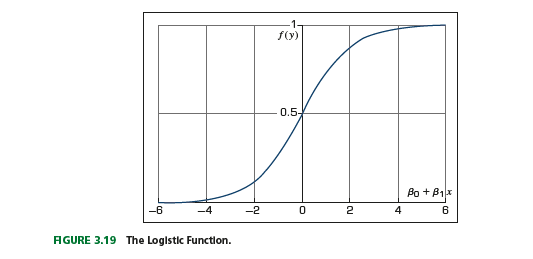
Normality (of errors). This assumption states that the errors of the response variable are normally distributed. That is, they are supposed to be totally random and should not represent any non random patterns.

Constant variance (of errors). This assumption, also called homoscedasticity, states that the response variables have the same variance in their error, regardless of the values of the explanatory variables. In practice, this assumption is invalid if the response variable varies over a wide enough range/scale.

Multicollinearity. This assumption states that the explanatory variables are not correlated (i.e., do not replicate the same but provide a different perspective of the information needed for the model). Multicollinearity can be triggered by having two or more perfectly correlated explanatory variables presented to the model (e.g., if the same explanatory variable is mistakenly included in the model twice, one with a slight transformation of the same variable). A correlation-based data assessment usually catches this error.

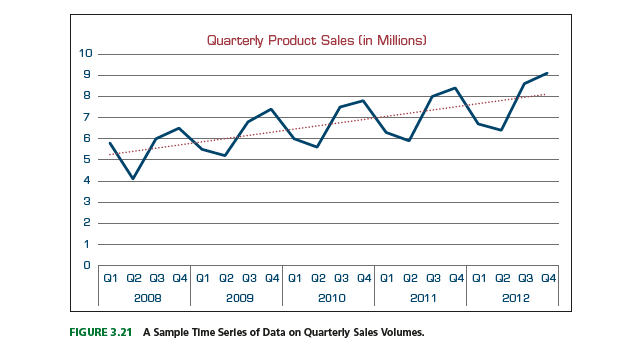
* Violations may require transformation or alternative models.

**Logistic Regression**

****

* Used for **binary outcomes** (Yes/No, Success/Failure).
* Estimates probability using **logit function**.

**Time Series Forecasting**

****

* Predicts values over time using historical data.
* Techniques: moving averages, exponential smoothing, ARIMA.
* Applied in demand forecasting, stock market prediction, weather forecasting.

Top of Form

Bottom of Form