DATA AND TYPES OF DATA

Data is unorganized information that is processed to make it meaningful.

Generally, data comprises of facts, observations, perceptions, numbers, characters, symbols, and images that can be interpreted to derive meaning.

One of the ways in which data can be categorized is by its structure.

Data can be:

Structured;

Semi-structured, or

Unstructured.

Structured data

--Structured data has a well-defined structure or adheres to a specified data model, can be stored in well-defined schemas such as databases, and in many cases can be represented in a tabular manner with rows and columns.

--Structured data is objective facts and numbers that can be collected, exported, stored, and organized in typical databases.

--Some of the sources of structured data could include:

1.SQL Databases and Online Transaction Processing (or OLTP) Systems that focus on business transactions,

2.Spreadsheets such as Excel and Google Spreadsheets, Online forms,

3.Sensors such as Global Positioning Systems (or GPS) and Radio Frequency Identification (or RFID) tags; and

4.Network and Web server logs.

structured data is stored in relational or SQL databases.

You can also easily examine structured data with standard data analysis methods and tools.

Semi-structured data

Semi-structured data is data that has some organizational properties but lacks a fixed or rigid schema.

Semi-structured data cannot be stored in the form of rows and columns as in databases.

It contains tags and elements, or metadata, which is used to group data and organize it in a hierarchy.

Some of the sources of semi-structured data could include:

E-mails, XML, and other markup languages, Binary executables, TCP/IP packets, Zipped files,

Integration of data from different sources.

XML and JSON allow users to define tags and attributes to store data in a hierarchical form and are used widely to store and exchange semi-structured data.

Unstructured data

Unstructured data is data that does not have an easily identifiable structure and, therefore, cannot be organized in a mainstream relational database in the form of rows and columns.
It does not follow any particular format, sequence, semantics, or rules.
Unstructured data can deal with the heterogeneity of sources and has a variety of business intelligence and analytics applications.
Some of the sources of unstructured data could include:
Web pages, Social media feeds, Images in varied file formats (such as JPEG, GIF, and PNG), video and audio files, documents and PDF files, PowerPoint presentations, media logs; and surveys.
Unstructured data can be stored in files and documents (such as a Word doc) for manual analysis or in NoSQL databases that have their own analysis tools for examining this type of data.
To summarize, structured data is data that is well organized in formats that can be stored in databases and lends itself to standard data analysis methods and tools;
Semi-structured data is data that is somewhat organized and relies on meta tags for grouping and hierarchy; and
Unstructured data is data that is not conventionally organized in the form of rows and columns in a particular format

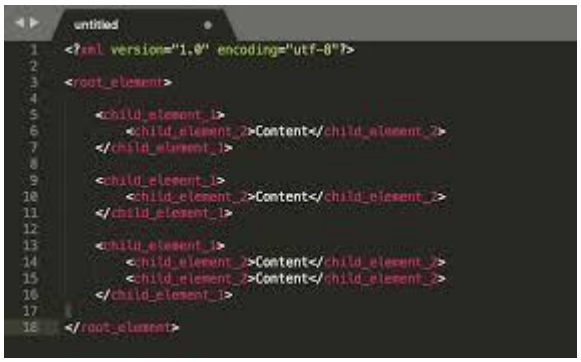
DATA SOURCES

some common sources such as
Relational Databases;
Flatfiles and XML Datasets
APIs and Web Services;
Web Scraping;
Data Streams and Feeds
Some of the standard file formats that we use
Delimited text file formats,
Microsoft Excel Open XML Spreadsheet, or XLSX
Extensible Markup Language, or XML,
Portable Document Format, or PDF,
JavaScript Object Notation, or JSON

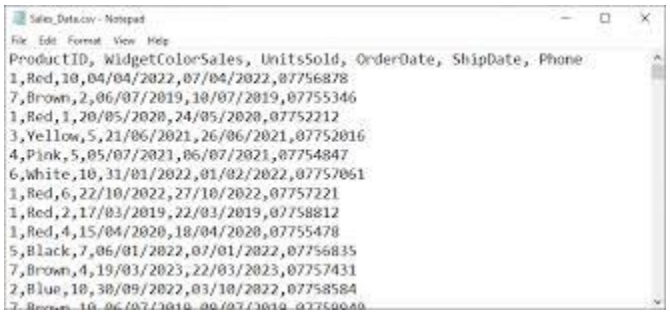
Flat File Model

	Route No.	Miles	Activity
Record 1	I-95	12	Overlay
Record 2	I-495	05	Patching
Record 3	SR-301	33	Crack seal

XML



CSV FILE



JSON FILE

```
{
  "first_name": "John",
  "last_name": "Smith",
  "is_alive": true,
  "age": 27,
  "address": {
    "street_address": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postal_code": "10021-3100"
  },
  "phone_numbers": [
    {
```

```
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [
    "Catherine",
    "Thomas",
    "Trevor"
  ],
  "spouse": null
}
```

Typically, organizations have internal applications to support them in managing their day to day business activities, customer transactions, human resource activities, and their workflows.
These systems use relational databases such as SQL Server, Oracle, MySQL, and IBM DB2, to store data in a structured way.
Data stored in databases and data warehouses can be used as a source for analysis.
For example, data from a retail transactions system can be used to analyze sales in different regions, and data from a customer relationship management system can be used for making sales projections.
External to the organization, there are other publicly and privately available datasets.
For example, government organizations releasing demographic and economic datasets on an ongoing basis.
Then there are companies that sell specific data, for example, Point-of-Sale data or Financial data, or Weather data, which businesses can use to define strategy, predict demand, and make decisions related to distribution or marketing promotions, among other things.
Such data sets are typically made available as flat files, spreadsheet files, or XML documents.
Flat files, store data in plain text format, with one record or row per line, and each value separated by delimiters such as commas, semi-colons, or tabs.
Data in a flat file maps to a single table, unlike relational databases that contain multiple tables.
One of the most common flat-file format is CSV in which values are separated by commas.
Spreadsheet files are a special type of flat files, that also organize data in a tabular format–rows and columns.
But a spreadsheet can contain multiple worksheets, and each worksheet can map to a different table.
Although data in spreadsheets is in plain text, the files can be stored in custom formats and include additional information such as formatting, formulas, etc.

Microsoft Excel, which stores data in .XLS or .XLSX format is probably the most common spreadsheet.
Others include Google sheets, Apple Numbers, and LibreOffice.
XML files, contain data values that are identified or marked up using tags.
While data in flat files is “flat” or maps to a single table, XML files can support more complex data structures, such as hierarchical.
Some common uses of XML include data from online surveys, bank statements, and other unstructured data sets.
Many data providers and websites provide APIs, or Application Program Interfaces, and Web Services, which multiple users or applications can interact with and obtain data for processing or analysis.
APIs and Web Services typically listen for incoming requests, which can be in the form of web requests from users or network requests from applications, and return data in plain text, XML, HTML, JSON, or media files.
Let’s look at some popular examples of APIs being used as a data source for data analytics:
The use of Twitter and Facebook APIs to source data from tweets and posts for performing tasks such as opinion mining or sentiment analysis—which is to summarize the amount of appreciation and criticism on a given subject, such as policies of a government, a product, a service, or customer satisfaction in general.
Stock Market APIs used for pulling data such as share and commodity prices, earnings per share, and historical prices, for trading and analysis.
Data Lookup and Validation APIs, which can be very useful for Data Analysts for cleaning and preparing data, as well as for co-relating data—for example, to check which city or state a postal or zip code belongs to.
APIs are also used for pulling data from database sources, within and external to the organization.
Web Scraping Web scraping is used to extract relevant data from unstructured sources.
Also known as screen scraping, web harvesting, and web data extraction, web scraping makes it possible to download specific data from web pages based on defined parameters.
Web scrapers can, among other things, extract text, contact information, images, videos, product items, and much more from a website.
Some popular uses of web scraping include
collecting product details from retailers, manufacturers, and eCommerce websites to provide price comparisons; generating sales leads through public data sources;
extracting data from posts and authors on various forums and communities;
and collecting training and testing datasets for machine learning models
Some of the popular web scraping tools include BeautifulSoup, Scrapy, Pandas, and Selenium.
Data streams are another widely used source for aggregating constant streams of data flowing

from sources such as instruments, IoT devices, and applications, GPS data from cars, computer programs, websites, and social media posts.
This data is generally timestamped and also geo-tagged for geographical identification.
Some of the data streams and ways in which they can be leveraged include:
stock and market tickers for financial trading;
retail transaction streams for predicting demand and supply chain management;
surveillance and video feeds for threat detection;
social media feeds for sentiment analysis;•sensor data feeds for monitoring industrial or farming machinery;
web click feeds for monitoring web performanceand improving design; and
real-time flight events for rebooking and rescheduling.
Some popular applications used to process data streams include Apache Kafka, Apache Spark Streaming, and Apache Storm.
RSS (or Really Simple Syndication) feeds, are another popular data source.
These are typically used for capturing updated data from online forums and news sites where data is refreshed on an ongoing basis.
Using a feed reader, which is an interface that converts RSS text files into a stream of updated data, updates are streamed to user devices.

PHILOSOPHIES OF DATA SCIENCE

Data science as a set of processes and concepts that act as a guide for making progress and decisions within a data-centric project.
This contrasts with the view of data science as a set of statistical and software tools and the knowledge to use them
The thought processes of a data scientist is more important than the specific tools used and how certain concepts pervade nearly all aspects of work in data science.
The origins of data science as a field of study or vocational pursuit lie somewhere between statistics and software development. Statistics can be thought of as the schematic drawing and software as the machine.
In addition to statistics and software, many folks say that data science has a third major component, which is something along the lines of subject matter expertise or domain knowledge.
Although it certainly is important to understand a problem before you try to solve it, a good data scientist can switch domains and begin contributing relatively soon.
Although it certainly is important to understand a problem before you try to solve it, a good data scientist can switch domains and begin contributing relatively soon.

Data should drive software only when that software is being built expressly for moving, storing, or otherwise handing the data. Software that's intended to address project or business goals should not be driven by data.

Problems and goals exist independently of any data, software, or other resources, but those resources may serve to solve the problems and to achieve the goals. The term data-centric reflects that data is an integral part of the solution, and we need to view the problems not from the perspective of the data but from the perspective of the goals and problems that data can help us address.

Locating and maintaining focus on the most important aspects of a project is one of the most valuable skills

Data scientists must have many hard skills—knowledge of software development and statistics among them—but this soft skill of maintaining appropriate perspective and awareness of the many moving parts in any data-centric problem to be very difficult yet very rewarding for most data scientists

Sometimes data quality becomes an important issue; sometimes the major issue is data volume, processing speed, parameters of an algorithm, interpretability of results, or any of the many other aspects of the problem. Ignoring any of these at the moment it becomes important can compromise or entirely invalidate subsequent results. The goal of a data scientist is to make sure that no important aspect of a project goes awry unnoticed. When something goes wrong—and something will—data scientist has to notice it and can fix it. Data scientist must also maintain awareness of all aspects of a project, particularly those in which there is uncertainty about potential outcomes.

1. DATA SCIENCE PROCESS OR LIFE CYCLE OF DATA SCIENCE

The lifecycle of a data science project can be divided into three phases, as illustrated in below figure. This is organized around these phases. The first part covers preparation, emphasizing that a bit of time and effort spent gathering information at the beginning of the project

The second part covers building a product for the customer, from planning to execution, using what you've learned from the first section as well as all of the tools that statistics and software can provide.

The third and final part covers finishing a project: delivering the product, getting feedback, making revisions, supporting the product, and wrapping up a project neatly.

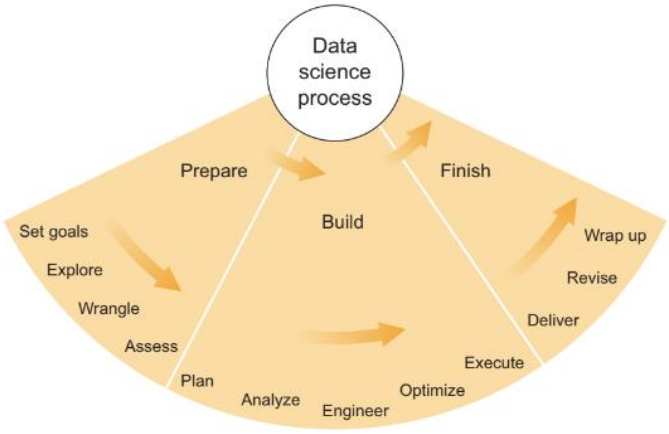


Figure 1.2 The data science process

2. AWARENESS IS VALUABLE

The most pervasive discrepancies between the perspective of a data scientist and that of a "pure" software developer is—one who doesn't normally interact with raw or "unwrangled" data.

DATA that is to be processed may come in all shapes and sizes, and parsing them for useful information is a challenge.

Two main strategies for addressing this challenge:

manual brute force and scripting.

We could also use some mixture of the two.

Given that brute force would entail creating a template for each data format as well as a new template every time the format changed. A script that could parse any data format and extract the relevant information is a better method, but it is extremely complex and almost impossible to write.

A compromise between the two extreme approaches seemed best, as it usually does.

Compromise between brute force and pure scripting: develop some simple templates for the most common formats, check for similarities and common structural patterns, and then write a simple script that could match and extract data. Practically this solution may not give 100% success.

awareness is incredibly valuable when working on problems involving data. A good developer using good tools to address what seems like a very tractable problem can run into trouble if they haven't considered the many possibilities that can happen when code begins to process data.

A data scientist's main responsibility is to try to imagine all of the possibilities, address the ones that matter, and reevaluate them all as successes and failures happen.

3. DEVELOPER VS. DATA SCIENTIST

A good software developer (or engineer) and a good data scientist have several traits in common. Both are good at designing and building complex systems with many interconnected parts; both are

familiar with many different tools and frameworks for building these systems; both are adept at foreseeing potential problems in those systems before they're actualized.

---In general, software developers design systems consisting of many well-defined components, whereas data scientists work with systems wherein at least one of the components isn't well defined prior to being built, and that component is usually closely involved with data processing or analysis. That component is usually closely involved with data processing or analysis.

The systems of software developers and those of data scientists can be compared with the mathematical concepts of logic and probability, respectively. The logical statement "if A, then B" can be coded easily in any programming language. The probabilistic statement "if A, then probably B" isn't nearly as straightforward.

Examples : Any good data-centric application contains many such statements—
consider the Google search engine ("These are probably the most relevant pages"), product recommendations on Amazon.com ("We think you'll probably like these things"),website analytics ("Your site visitors are probably from North America and each views about three pages").

Data scientists specialize in creating systems that rely on probabilistic statements about data and results.

A software developer needs expertise, but a data scientist needs exploration.

4. **DO I NEED TO BE A SOFTWARE DEVELOPER?**

Knowledge of a statistical software tool is a prerequisite for doing practical data science, but this can be as simple as a common spreadsheet program (for example, the divisive but near-ubiquitous Microsoft Excel). In theory, someone could be a data scientist without ever touching a computer or other device. Understanding the problem, the data, and relevant statistical methods could be enough, as long as someone else can follow your intentions and write the code. In practice, this doesn't happen often.

the process of thinking about and doing data science, but clearly software can't be ignored. Software—as an industry and its products—is the data scientist's toolbox. The tools of the craft are the enablers of work that's beyond the capabilities of the human mind and body alone.

5. **DO I NEED TO KNOW STATISTICS?**

As with software, expert knowledge of statistics certainly helps but isn't necessary. Statistics provides a framework for theorizing about how intangible thoughts, needs, and reactions are eventually translated into measurable actions that create data and formulate models of data-generating processes and translating those models into statistical terminology, equations, and, ultimately, code.

6. **PRIORITIES: KNOWLEDGE FIRST, TECHNOLOGY SECOND, OPINIONS THIRD**

There are various concerns of every data science project—for example, software versus statistics, changing business need versus project timeline, data quality versus accuracy of results. Each individual concern pushes and pulls on the others as a project progresses, and we're forced to make choices whenever two of them disagree on a course of action.

Knowledge, technology, and opinions are typically what you have at the beginning of any project; they are the three things that turn data into answers. Knowledge is what you know for a fact. Technology is the set of tools you have at your disposal. Opinions are those little almost facts you want to consider true but shouldn't quite yet. It's important to establish a hierarchy for your thought processes so that less-important things don't steamroll more-important ones because they're easier or more popular In practice, the hierarchy looks like this:

 Knowledge first—Get to know your problem, your data, your approach, and your goal before you do anything else, and keep those at the forefront of your mind.

 Technology second—Software is a tool that serves you. It both enables and constrains you. It shouldn't dictate your approach to the problem except in extenuating circumstances.

 Opinions third—Opinions, intuition, and wishful thinking are to be used only as guides toward theories that can be proven correct and not as the focus of any project.

Goals are not certain to be achieved but are required for any project, so it's imperative not to take the goal and its attainability for granted. You should always consider current knowledge first and seek to expand that knowledge incrementally until you either achieve the goal or are forced to abandon it In data science, a goal is much less likely to be achievable in exactly its original form. Remember: knowledge first, then technology, and then opinion. It's not a perfect framework, but it will be helpful.

7. BEST PRACTICES by DATA SCIENTIST

7.1 Documentation

- Comment your code so that a peer unfamiliar with your work can understand what the code does.
- For a finished piece of software—even a simple script—write a short description of how to use it and put this in the same place (for example, the same file folder) as the code.
- Make sure everything—files, folders, documents, and so on—has a meaningful name.

7.2 Code Repositories and Versioning

• Source code repositories (repos) are software products designed to manage source code.
• Modern repos are based on versioning systems, which track changes and allow for creation and comparison of different versions.
• Repos and versioning take time to learn and integrate into workflows.
• Bitbucket.org and GitHub.com offer free web hosting of code repos, with Git being the most popular versioning system.
• Remote repo-hosting services serve as a backup, ensuring code safety even in the event of a computer crash.
• Some code-hosting services offer web interfaces for viewing code history, versions, and development status.
• Remote repos allow code access from any location with web access, with language-specific code highlighting and other useful features.
• Tips for repos and versioning include using a remote source code repo, learning Git or another versioning system, committing code changes frequently, working in a location that doesn't affect production version or development by other team members, using versioning, branching, and forking instead of copying and pasting code, and asking Git gurus for best practices.

7.3 Code Organization

• Use common coding patterns for the programming language.
• Use meaningful names for variables and objects for better understanding.
• Use informative comments for better code organization.
• Avoid copying and pasting code.
• Code in chunks with specific functionalities for scripts and applications.
• Avoid premature optimization.
• Code in a logical, coherent fashion.
• Consider potential contributors to your project.
• Spend time organizing and commenting early to ensure understanding.

7.4 Ask Questions

Data Scientists' Awareness and Domain Knowledge

• Data scientists possess a strong sense of awareness, which can be a strength or a weakness.
• The stereotype of introverted academics being too shy to ask for help is often misguided.
• Data scientists, including software engineers, business strategists, sales executives, marketers, and researchers, possess vast knowledge about their domains or projects.
• In a business setting, it's important to learn about the company and the industry, or domain knowledge.
• Nontechnical business people often treat data scientists as smart, but they also have a deeper understanding of project goals and business problems.
• Engaging in discussions with those who understand the business side of the project can illuminate projects and contribute to domain knowledge.

7.5 Staying Close to Data in Data Analysis

• The concept of being "close to the data" involves minimizing the complexity of the methods

and algorithms used.
• Complex methods, such as black box methods, can be beneficial in certain fields. (but be conscious of the possibility of mistakes)
• In cases where complex methods have advantages, the concept of being close to the data can be adapted.
• This can involve verifying, justifying, or supporting results from complex methods with simpler, close-to-data methods.
• Overstraying from the data without a safety line can lead to difficulties in diagnosing problems.