Ethics and Privacy

Storytelling

Domain Knowledge

Big Data

Machine Learning

Visualization

Processing and cleaning

Data Collection

DATA SCIENCE

Hands-on studies compilation

Marcos Lima

Statistics

Ethics and
Privacy

Random Sampling
Stratification
Split

DATA COLLECTION

Archive
Data Lake
Storage

Values
Structured
Unstructured

from

Sources
IOT Devices
Web Scraping
API Integration

with

Quality
Cleansing
Validation

The process of data collection is the main building of data science. This chapter is the beginning of an exploration of diverse data sources dive into the intricacies of structured data, demonstrating how it is meticulously collected from databases and well-organized repositories. Unstructured data, represented by text, images, and videos, poses unique challenges. Semi-structured data, which exhibits a degree of organization but may not adhere to a strict schema.

Moving beyond data sources, there is a vast landscape of data acquisition techniques. The world of web scraping uses HTTP protocol and navigation system over the scripts to harvest data from websites. The integration of Application Programming Interfaces (APIs) can be used as a means to access a wealth of data from various online services and platforms.

Data quality is a crucial facet. All process bellow this point will reflect any failure or miscalculation. Data must have consistency for your results thrive. Techniques for data validation ensure that the data collected remains trustworthy.

In our data-driven age, data privacy and ethics play pivotal roles. Every chapter will bring ideas and techniques in concern to individual privacy.

Furthermore, the journey goes through the world of data sampling, a important branch of statistics and its history. Methods of random sampling, which involves selecting a representative subset of data, stratified sampling, a technique that divides data into subgroups and samples from each were discovered and applied into different scope of data.

Collect data and quality

The most used tools for data manipulation include programming languages such as Python, R and Matlab. These languages are considered 'Swiss Army knives' and share a common thread of versatility and functionality. The code examples will be primarily provided in Python or R. It's important to note that the intention here is to provide a practical perspective, concrete illustrations of the theoretical concepts, navigating to scenarios and showcase how these versatile languages can be applied to solve data-related problems.

Data frame

A data frame is a fundamental data structure used in statistics, data analysis and programming. It can be thought of as a two-dimensional table where data is organized in rows and columns. Each column typically represents a variable or a field, while each row corresponds to an observation or a data point. Data frames are versatile and can store a variety of data types, including numeric, character and factor data. They offer an efficient way to manipulate, analyze, and visualize data, making them a crucial component in data science and statistical analysis. Data frames allow for seamless integration of data from various sources, making them a popular choice for organizing and exploring datasets in data-centric tasks.

Installing R

To detailed instructions on how to install R visit the project website at https://www.r-project.org/

```
> # Example of a data frame
> df <- data.frame(x=c(1,2,3),
                   y=c(7,8,9),
                   z=c("A","B","C"))

> print(df)
  x y z
1 1 7 A
2 2 8 B
3 3 9 C
```

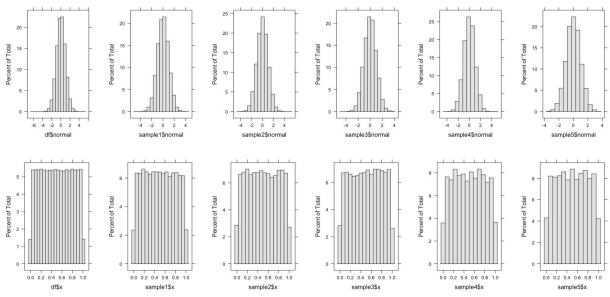
Data sources and their quality play pivotal roles. It's essential to consider the sources from which data is collected ensuring that data is collected from reliable, relevant, and diverse sources enhances the richness and representativeness of the data frame. Moreover, addressing data quality, which includes handling missing data, outliers, and inconsistencies, is crucial for maintaining the integrity of the data frame. Quality assurance processes, such as data cleansing and validation, are fundamental to producing accurate and reliable data frames. This subject will be described further with more details. In this page you can find simple examples how to handle a data frame.

```
# library rvest contains several functions
# to scrape data from internet
scrape_WC02_df <- function(){
  library(rvest)
  # Load World Cup data from wikipedia website
  url <- "https://en.wikipedia.org/wiki/2002_FIFA_World_Cup"
  html_code <- read_html(url)
  # You can find xpath using browser developer tools
  # identifying the table to scrape
  nodes <- html_nodes(html_code,
    xpath='//*[@id="mw-content-text"]/div[1]/table[26]')
  table <- html_table(nodes)
  return(data.frame(table))
}
> df <- scrape_WC02_df()
> print(paste("df has",dim(df)[2],"columns,",dim(df)[1],"rows"))
[1] "df has 12 columns, 32 rows"
> # Selecting columns
> print(df[1:5,c("Team","Pos")])
         Team Pos
1      Brazil   1
2     Germany   2
3      Turkey   3
4 South Korea   4
5       Spain   5
> # Search queries
> print(df[df$Team=="Brazil",c("Team","Pos","Pld","W","D","L")])
    Team Pos Pld W D L
1 Brazil   1   7 7 0 0
> print(df[df$L==0,c("Team","Pos","Pld","W","D","L","GF","GA")])
                  Team Pos Pld W D L GF GA
1               Brazil   1   7 7 0 0 18  4
5                Spain   5   5 3 2 0 10  5
12 Republic of Ireland  12   4 1 3 0  6  3
> # Make some calculations
> print(paste("Sum of goals in the tournament:",sum(df$GF)))
[1] "Sum of goals in the tournament: 161"
> print(paste("Average of goals:",sum(df$GF)/sum(df$Pld)))
[1] "Average of goals: 1.2578125"
> # Adding new column to the data frame
> df$goals_ratio <- df$GF/df$Pld
> # Top 10 scoring ratio teams
> print(head(
+   df[order(df$goals_ratio,decreasing=TRUE),
+      c("Team","Pos","GF","GA","goals_ratio")],10))
                  Team Pos GF GA goals_ratio
1               Brazil   1 18  4    2.571429
2              Germany   2 14  3    2.000000
5                Spain   5 10  5    2.000000
21            Portugal  21  6  4    2.000000
17        South Africa  17  5  5    1.666667
19          Costa Rica  19  5  6    1.666667
12 Republic of Ireland  12  6  3    1.500000
14             Belgium  14  6  7    1.500000
16            Paraguay  16  6  7    1.500000
3               Turkey   3 10  6    1.428571
```

Sampling

Statistical sampling involves the process of selecting a subset of data from a larger population to make inferences about that population. This field has a rich history dating back to the early days of statistical analysis, with pioneers like Francis Galton, who used sampling techniques to study heredity, and Ronald A. Fisher, who made substantial contributions to the theory of sampling. In modern data science, sampling is integral to efficiently handling large datasets, conducting surveys, and performing hypothesis testing. Understanding various sampling methods, such as random and stratified sampling, allows data scientists to extract valuable insights and draw conclusions from data with precision and reliability. Sampling is a powerful tool to data science to its analytical core.

```
> # Generate a data frame with random values
> # rnorm creates normal distribution and runif uniform
> df = data.frame(normal=rnorm(78**3),x=runif(78**3))
> summary(df)
     normal                x
 Min.   :-4.545741   Min.   :0.0000036
 1st Qu.:-0.673120   1st Qu.:0.2516106
 Median :-0.000155   Median :0.5012793
 Mean   : 0.001163   Mean   :0.5007995
 3rd Qu.: 0.674084   3rd Qu.:0.7507580
 Max.   : 4.615366   Max.   :0.9999989
> require(lattice)
> histogram(df$normal,col="gray")
> # Create samples with different sizes 10-1% of original
> sample1 <- df[sample(nrow(df),round(nrow(df)*.10)),]
> sample2 <- df[sample(nrow(df),round(nrow(df)*.08)),]
> sample3 <- df[sample(nrow(df),round(nrow(df)*.05)),]
> sample4 <- df[sample(nrow(df),round(nrow(df)*.02)),]
> sample5 <- df[sample(nrow(df),round(nrow(df)*.01)),]
> # Plot will have same distributions as population
```



```
> # Now, if you use the same seed...
> set.seed(9)
> df = data.frame(normal=rnorm(78**3),x=runif(78**3))
> # You will find the same "random numbers" as bellow
> summary(df)
     normal                x
 Min.   :-4.618437   Min.   :0.0000004
 1st Qu.:-0.676649   1st Qu.:0.2496231
 Median :-0.000465   Median :0.4991344
 Mean   :-0.001224   Mean   :0.4994693
 3rd Qu.: 0.674798   3rd Qu.:0.7493647
 Max.   : 4.832043   Max.   :0.9999978
```

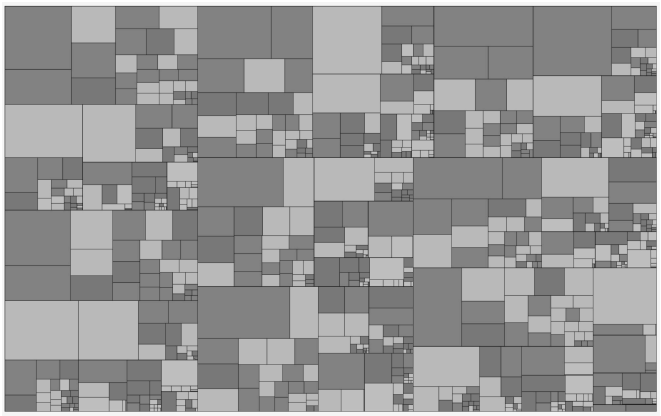
Randomization

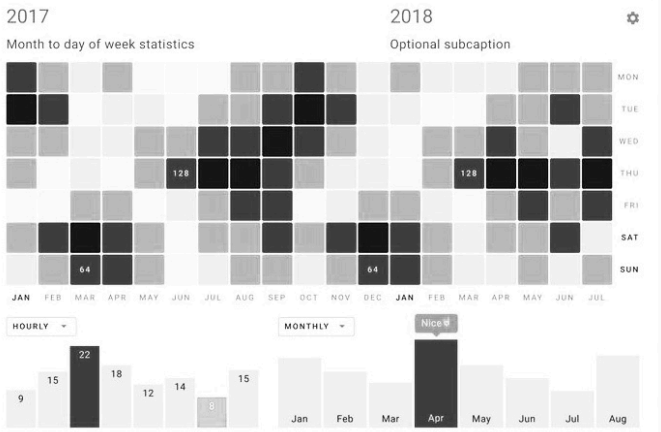
This is a process widely used in statistics and data science to ensure the selection of a representative subset of data from a larger dataset. It avoid bias and increase the generalizability of results. Randomness reduces the likelihood of unintentional patterns or biases enabling robust statistical analyses and dependable insights. Although random subsets of data have the same distribution shape as the population.

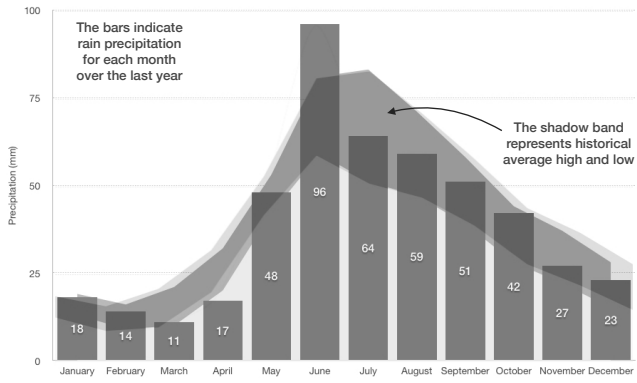
Randomization in computer is achieved using pseudorandom number generators (PRNGs). These algorithms generate sequences of numbers that appear to be random but are actually determined by an initial value called a seed. The seed is typically set based on some unpredictable factor, such as system time or user input. While PRNGs can't produce truly random numbers like radioactive decay or quantum processes, they are sufficient for most practical purposes. The key is to use well-designed PRNGs and frequently update the seed to minimize predictability.

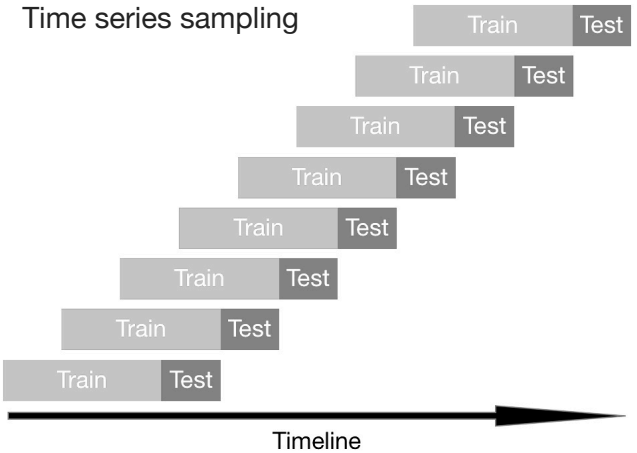
Seed

In random selection algorithms, the seed plays a crucial role, as it not only ensures randomness but also enables replicability. By setting the same seed, you can choose the same sample repeatedly and achieve same results. This capability has several valuable applications. First, it's a useful technique for troubleshooting as it allows you to retrace your steps and investigate issues in your analysis. Second, it promotes collaboration and idea sharing among colleagues. When you share your code and analysis with colleagues providing them with the same seed ensures they can reproduce your results precisely.









Stratification

Stratified sampling involves dividing a diverse dataset into homogeneous subgroups or strata based on specific characteristics or attributes. Each stratum represents a subset of the population, and then random sampling is applied within each stratum. The key idea is to ensure that each subgroup is well-represented in the sample, which can lead to more accurate and reliable inferences. Stratified sampling is particularly useful when dealing with a population that exhibits significant variability across certain characteristics, as it allows you to obtain a balanced and representative sample.

Time series sampling

It is a specialized form of sampling commonly used in analytics when dealing with time-ordered data. Time series data consists of observations or measurements collected and recorded in chronological order. It aims to select specific data points or segments within this chronological sequence for analysis. This type of sampling is instrumental in tasks such as forecasting future trends, identifying seasonality patterns, and understanding the temporal behavior of data. Some common techniques include simple random sampling from time intervals, systematic sampling at fixed intervals, or sampling specific time points of interest. Time series sampling allows data scientists to uncover patterns and insights within temporal data, making it a valuable component in various applications, from financial analysis to climate modeling.

Moreover, we will dive into the evaluation process of predictions. It's essential to have two distinct samples collected from different timeframes to ensure that our assessments out-of-sample are not influenced by any seasonal biases. This helps in providing a more comprehensive and unbiased evaluation of predictive models and their performance across various temporal scenarios.

Streamlined Ingestion Process

Value Added
Self-Service
Data-driven

Timeliness
Always Ready
Easy to Find

Scale
Robust
Support Growth

Flexibility
Easy Modified
Automated
Streamlined

Quality
Explicit Visibility
Trustworthy

A data lake is an integral component of modern data management, serving as a foundational element for organizations striving to become data-driven. It encompasses several crucial characteristics, each of which contributes significantly to its value and utility.

At its core, a data lake is designed for scalability, capable of accommodating vast volumes of data. This scalability is essential in today's data-rich environment, where information accumulates at an unprecedented rate. A robust infrastructure ensures that the data lake can handle diverse data types, ranging from structured to unstructured, without compromising its performance. This combination of scalability and robustness ensures that the data lake can support an organization's growth without data-related constraints.

One of the standout features of a data lake is its self-service capability. Users across the organization can access and retrieve data independently, reducing the burden on IT teams and accelerating decision-making processes. This self-service aspect adds substantial value by promoting agility and enabling users to extract insights swiftly. Additionally, it empowers individuals within the organization to explore and analyze data, fostering a culture of data-driven decision-making.

Data lakes are engineered to be always ready. This means that data is readily available for analysis, ensuring that time-sensitive insights can be derived promptly. The timeliness of data retrieval and analysis can be a critical factor in various business scenarios. Furthermore, data lakes are designed with accessibility in mind, making it easy for users to find the specific data they require. This accessibility streamlines workflows and minimizes the time spent searching for relevant information.

Flexibility is another key attribute of data lakes. They can seamlessly adapt to evolving business needs and changing data requirements. This adaptability is particularly crucial in dynamic environments where data formats and sources may vary over time. Whether it's incorporating new data sources or modifying existing data structures, a data lake's flexibility ensures that it remains aligned with the organization's goals.

Data quality is a fundamental aspect of any data management strategy, and data lakes are no exception. They offer explicit visibility into data quality, enabling organizations to establish and maintain trustworthy data sources. This visibility builds confidence among users that the data they access is accurate and reliable, reinforcing the credibility of data-driven insights.

Efficiency is a hallmark of data lakes. They are designed to automate and streamline data ingestion processes, reducing manual intervention and the risk of errors. This automation not only enhances data processing speed but also ensures that data is ingested consistently, preserving its integrity.

Data Lake serves as a comprehensive solution of self-service capability that empowers users, timeliness and accessibility to promote quick decision-making. Quality and trustworthiness ensure the reliability of insights. Together, these characteristics make the data lake an invaluable asset in the quest for data-driven excellence.

Storage

A virtual drive can be used for storing data and scripts in the context of a data lake. Many cloud storage providers offer virtual drives that are accessible via the internet. Examples like Google Drive, Dropbox, OneDrive, and Box. You can store data files, scripts, and other assets in these virtual drives. They provide the advantage of easy access and collaboration from various devices and locations.

In an enterprise or on-premises data lake setup, you can use network-mapped drives to access centralized storage resources. These drives can be used to store and share data and scripts across a local network.

Using shared drives can simplify data and script management, but it's essential to ensure proper security, access control, and versioning practices to maintain data integrity and manage scripts effectively.

Many data lakes use object storage solutions like Amazon S3, Azure Blob Storage, or Google Cloud Storage. Object storage provides a scalable and cost-effective way to store both data and scripts. You can organize your data into containers or buckets and include script files alongside the data.

Data lakes built on Hadoop-based platforms often use Hadoop Distributed File System (HDFS) or Hadoop-compatible distributed file systems. These systems allow you to store data and scripts within the same environment.

For managing data-related scripts version control, systems like Git are commonly used. Git allows you to track changes to your scripts, collaborate with team members, and maintain a history of script versions. While data itself is not stored in Git, you can include references to data files or data storage locations within your scripts.

The choice of storage depends on the specific architecture and tools used in your data lake ecosystem. It's common to use a combination of options to ensure that data and scripts are well-organized, versioned, and accessible for data processing and analysis.

Archive

In the dynamic field of data science, maintaining a structured and accessible data archive is indispensable. Data archiving involves the systematic storage and preservation of valuable datasets, ensuring their availability for future analysis.

Data is an asset, and like any asset, it requires safeguarding. Different datasets demand distinct archiving strategies: local, cloud-based or offline. Adequate documentation is a must-piece of data archiving creating comprehensive metadata, including descriptions, data dictionaries, and version histories.

Time Machine Archive is a concept. Much like its fictional namesake, it allows you to revisit and restore data as it existed at specific points in time. It can be built in a special file format known as .rda (R Data).

Storing archive in these files involves creating snapshots of your datasets and functions at different time points and saving them as .rda files. These files are essentially binary R objects that contain the state of your data at specific moments in time. To store a Time Machine Archive, you can follow a version control approach, such as Git, which helps you manage changes to your datasets over time. By creating and committing .rda files with clear labels or descriptions for each snapshot, you maintain a historical record of your data's evolution. When needed, you can retrieve, explore, or restore any version of your dataset from these files, ensuring data reproducibility and facilitating error correction in data science projects.

An essential aspect of data archiving is understanding the data's lifecycle. The lifecycle of data doesn't end with archiving; it also includes responsible data purging and deletion practices, especially in compliance with data privacy regulations.

In an era of ever-expanding data volumes, the knowledge of data archiving, Time Machine Archives, and their implementation through .rda files is useful for data scientists to maintain data integrity, support research reproducibility, and ensure data's long-term accessibility.