Contents

1 Introduction

1.1 Dataset

1.1.1 Data Sets

1. **Mtcars dataset**: The mtcars dataset in R is a built-in dataset that contains information about various car models. It provides data on the characteristics of 32 different car models, which were available in the early 1970s. The dataset includes a total of 11 variables, each representing different attributes of these cars, such as miles per gallon (mpg), horsepower (hp), number of cylinders (cyl), and more. The mtcars dataset is often used for data analysis, visualization, and statistical modeling, making it a useful resource for exploring and practicing data science techniques in R.

2. **Annual fire in Brazil**: it is obtained from NASA. Each dataset from 2013 to 2022 contains over 200,000 observations. Over the decade, there are more than 3 million observations.

The *ToothGrowth* dataset is a built-in dataset in R. It is used to study the impact of vitamin C on the tooth growth of Guinea pigs. The dataset consists of 60 observations and 3 variables:

- **len**: Represents the length of the Guinea pigs' teeth.
- **supp**: Indicates the method of vitamin C supplementation, with two levels: "VC" for vitamin C in their diet and "OJ" for vitamin C in orange juice.
- **dose**: Denotes the dosage of vitamin C in milligrams per day, with three levels: 0.5, 1, and 2.

The iris dataset is a classic dataset in the field of statistics. It was introduced by the British biologist Ronald A. Fisher in 1936 as an example of discriminant analysis. The dataset consists of 150 samples from three species of iris flowers: setosa, versicolor, and virginica. For each sample, four features were measured: the lengths and the widths of the sepals and petals, all in centimeters. The dataset is often used for classification tasks to differentiate between the three species based on the given measurements. It has become a standard test case for many classification algorithms and is widely recognized in the data science community.

Time series of the daily CNY, CAN, EUR, HKD, USD versus GBP exchange reference rate data published by the European Central Bank over the time period from 01 Jan 2013 to 12 Oct 2023 (without weekends). The exchange rate tells you how many pounds you need to buy/sell 1 CNY, CAN, EUR, HKD, USD.

The data set has the format as below:

Date	**CNYtoGBP**	**CANtoGBP**	**EURtoGBP**	**HKDtoGBP**	**USDtoGBP**
%d-%m-%y	Value	Value	Value	Value	Value

Table 1: Field Information: CNY, CAN, EUR, HKD, USD to GBP

2 Theoretical Foundations of Data Visualisation

This chapter, "Theoretical Foundations of Data Visualisation," delves deep into the core principles and concepts that serve as the bedrock of this dynamic field. We seek to understand not only the "how" but also the "why" behind the creation of visualisations that captivate and inform.

2.1 Introduction to Data Visualisation Theory

In the pursuit of creating effective data visualisations, it is crucial to understand that behind every chart, graph or plot lies a solid theoretical framework. Theoretical underpinnings provide the foundation upon which data visualisation is built, shaping not only how we represent data but also how we perceive, understand, and interpret it.

2.1.1 Guiding Principles for Data Representation

Within this theoretical framework, we encounter a set of guiding principles that dictate how data should be represented visually. These principles encompass fundamental concepts such as:

- **Accuracy**: Data visualisations should accurately reflect the underlying data, minimising distortion or misinterpretation.
- **Simplicity**: The "less is more" principle applies to data visualisation. Simplified visuals often convey information more effectively than cluttered ones.
- **Clarity**: Visualisations should be clear and understandable to the intended audience, avoiding unnecessary complexity.
- **Relevance**: Information presented should be relevant to the message or question being addressed.
- **Consistency**: Visual elements, such as colour coding and labelling, should be used consistently throughout a visualisation.

2.1.2 Theoretical Framework and Visual Perception

One of the fundamental aspects of data visualisation theory is an understanding of how the human brain perceives visual information. This knowledge is instrumental in designing visualisations that resonate with viewers. It includes considerations like:

- **Gestalt Principles**: The Gestalt principles of visual perception, including proximity, similarity, and continuity, influence how we group and interpret visual elements in a visualisation.
- **Colour Theory**: The effective use of colour, including colour contrasts and harmonies, can enhance the clarity and impact of a visualisation.
- **Cognitive Load**: Minimising the mental effort required to process information is vital.

2.2 Visual Perception and Cognition

Understanding the intricacies of how humans interpret visual information is pivotal to the art and science of data visualisation. Thus, we explore human visual perception, along with the application of cognitive psychology principles in data visualisation and highlight the crucial role of pre-attentive attributes in shaping our perception of data.

2.2.1 Human Visual Perception: Decoding Visual Information

Human visual perception is a remarkable cognitive process that allows us to decode and make sense of the world around us. When applied to data visualisation, it illuminates how viewers interact with and derive meaning from visual representations of data. Key aspects of human visual perception in the context of data visualisation include:

- **Pattern Recognition**: The human brain excels at recognizing patterns, making it adept at identifying trends, outliers, and relationships in data visualisations.
- **Perceptual Grouping**: We tend to group visually similar elements together, a principle known as perceptual grouping. This informs how we interpret clusters of data points and shapes in a visualisation.
- **Hierarchy of Perception**: Certain visual attributes are processed more quickly and efficiently than others. For example, colour is often processed faster than text, influencing the viewer's attention hierarchy.

By harnessing the principles of human visual perception, applying insights from cognitive psychology, and leveraging pre-attentive attributes, data visualisation designers can create visualisations that are not only aesthetically pleasing but also cognitively efficient.

2.2.2 Gestalt Principles

Gestalt psychology principles have long been recognised as fundamental to the field of visual perception and design. Gestalt psychology is a school of thought that emphasises how humans perceive and organise visual information. It posits that the mind seeks to create meaningful patterns and wholes from individual visual elements. The Gestalt principles thus, provide a framework for understanding how viewers naturally group and interpret visual stimuli.

Several key Gestalt principles play a pivotal role in shaping our perception of visual information. These principles include:

- **Proximity**: Elements that are close to each other are perceived as related or belonging to the same group. In data visualisation, proximity can be used to group data points or related information.
- **Similarity**: Elements that share similar visual attributes, such as colour, shape, or size, are perceived as belonging together. Similarity can be harnessed to encode categorical data or highlight relationships.
- **Continuity**: The human mind tends to perceive continuous patterns or lines as a single entity. In data visualisation, continuity can aid in representing trends or connections between data points.
- **Closure**: Viewers tend to mentally complete incomplete shapes or patterns. Closure can be employed to suggest relationships or connections even when not explicitly shown in the visualisation.
- **Symmetry**: Symmetrical elements are often perceived as more balanced and harmonious. Symmetry can be used for aesthetically pleasing and easily understandable visualisations.

2.2.3 Application of Gestalt Principles in Designing Visualisations

The application of Gestalt principles in designing data visualisations can lead to more intuitive and effective communication of information. Designers can strategically leverage these principles to:

- Group-related data points to enhance clarity and reduce visual clutter.
- Use colour or shape to signify meaningful categories or groupings.
- Create smooth, continuous lines or paths to guide the viewer's gaze through the visualisation.
- Suggest connections or patterns even in complex datasets.

2.3 Data abstraction and Representation

The transformation of raw data into meaningful representations is a pivotal step in data visualisation. This process, known as data abstraction, involves distilling complex datasets into visual forms that convey insights. In this section, we explore data abstraction, the hierarchies and levels of abstraction in data visualisation, and the critical trade-offs between abstraction and the potential loss of information.

2.3.1 Data Abstraction: Transforming Raw Data

Data abstraction is the art of simplifying and structuring raw data into formats that are comprehensible and insightful. It is the bridge that transforms numbers, text, and variables into visual elements that convey patterns, trends, and relationships. Effective data abstraction is at the heart of creating informative data visualisations.

2.3.2 Hierarchies and Levels of Abstraction

In data visualisation, abstraction operates on multiple levels of granularity. Hierarchies of abstraction allow us to represent data at varying levels of detail:

1. **Low-Level Abstraction**: At the lowest level, raw data is preserved in its most detailed form. This might include individual data points, measurements, or unprocessed text.
2. **Mid-Level Abstraction**: As we move up the hierarchy, data is grouped or aggregated to provide a broader overview. For example, hourly data points may be aggregated into daily or weekly averages.
3. **High-Level Abstraction**: At the highest level, data is represented in a condensed and abstracted form, often as summary statistics or key insights. This level provides a big-picture view.

2.3.3 Trade-offs Between Abstraction and Information Loss

While abstraction is essential for simplifying complex data, it comes with trade-offs. Increasing levels of abstraction can lead to information loss, where fine-grained details or outliers are overlooked. It is crucial for data visualisation designers to strike a balance:

- **Clarity vs. Detail**: Increasing abstraction can enhance the clarity of a visualisation but may sacrifice detailed information that is important for certain analytical tasks.

- **Generalisation vs. Specificity**: Abstraction can provide a more generalised view of data, making it accessible to a wider audience. However, it may miss specific nuances that experts may require.
- **Context vs. Precision**: High-level abstraction can provide valuable context, but it may lack the precision needed for precise decision-making.

In data visualisation, the art of data abstraction lies in finding the right level of detail that effectively conveys the intended message while minimising the risk of information loss. This balancing act is a critical consideration in the design of informative and meaningful data visualisations.

2.4 Data Types and Visualisation Techniques

In the world of data visualisation, understanding the nature of your data is key. Data comes in various types, and selecting the appropriate visualisation technique is contingent upon recognising these distinctions. In this section, we categorise data types, and demonstrate how to match each data type with suitable visualisation techniques.

2.4.1 Categorisation of Data Types

Data types can be broadly categorised into four main types:

- **Nominal data**: nominal data represents categories or labels without any inherent order. Examples include colours, gender categories, and city names.
- **Ordinal data**: ordinal data implies a meaningful order or ranking among categories but lacks equal intervals between them. Examples include survey responses (eg. "very satisfied", "satisfied", "neutral", "dissatisfied", "very dissatisfied")
- **Interval data**: interval data possesses ordered categories with equal intervals between them, but it lacks a true zero point. Temperature is measured in Celsius or Fahrenheit as an example.
- **Ratio data**: ratio data includes ordered categories with equal intervals and a meaningful zero point. Examples are age, income, and weight.

2.4.2 Matching Data Types with Appropriate Visualisation Techniques

Selecting the right visualisation technique for your data type is pivotal to effective communication. Here are some examples of visualisation techniques matches with corresponding data types:

- **Nominal data**: Techniques such as bar charts and stacked bar charts are effective in displaying categorical information and relative proportions.
- **Ordinal data**: Ordinal data can be visualised using ordered bar charts, dot plots, or stacked bar charts, which maintain the ranking and order of the categories.
- **Interval data**: Interval data benefits from visualisation methods like line charts, histograms, and box plots, which highlight trends and distributions without assuming a true zero point.
- **Ratio data**: Ratio data can be effectively presented using scatter plots, histograms, and line charts, allowing for precise comparisons and measurements due to the presence of a meaningful zero point.

2.5 Colour Theory in Data Visualisation

Here, we explore the significance of colour in data visualisation, the principles of colour perception and encoding, and the importance of avoiding misleading visualisations through thoughtful colour choices.

2.5.1 The Importance of Colour in Conveying Information

Colour is a potent tool for enhancing the understanding and impact of data visualisations. It enables the differentiation of data points, highlights trends, and provides context. Colour can be used to:

- **Encode Categorical Data**: Distinguish between different groups or classes using distinct colours.
- **Represent Quantitative Data**: Use colour intensity or gradients to represent values or magnitudes.
- **Add Context**: Apply colour to background elements, labels, or annotations to provide context and meaning to the visualisation.

2.5.2 Colour Perception and Colour Encoding in Visualisations

Understanding how colour is perceived by viewers is essential in data visualisation. Key principles include:

- **Colour Discrimination**: Consider that not all viewers may perceive colour in the same way. Ensure your colour choices are accessible to individuals with colour vision deficiencies (colour blindness).
- **Colour Encoding**: Select colour schemes that align with the message you want to convey. For example, warm colours like red and orange often signify caution or heat, while cool colours like blue and green convey calmness or coldness.
- **Colour Combinations**: Pay attention to how colours interact when placed adjacent to each other. Some combinations may create visual vibrations or make text difficult to read.

2.5.3 Avoiding Misleading Visualisations Due to Colour Choices

Misleading visualisations can result from inappropriate or deceptive use of colour. To avoid this:

- **Consistency**: Maintain consistency in colour usage throughout your visualisation. Use the same colour scheme for similar data categories or elements.
- **Avoid Distortion**: Ensure that colour choices do not exaggerate or distort the data. Overly intense or contrasting colours can lead to misinterpretation.
- **Legend Clarity**: Provide a clear and concise legend to explain the meaning of colours, especially when dealing with complex or unfamiliar colour schemes.
- **Test with Users**: Conduct user testing to ensure that your colour choices effectively convey the intended message and do not confuse or mislead the audience.

2.6 Theoretical Properties of Visualisations

Effective data visualisation is not solely about creating aesthetically pleasing graphics; it's also about adhering to key theoretical properties that optimise the expressiveness, precision, accuracy, and scalability of visual representations. In this section, we delve into these properties, including expressiveness and effectiveness, the data-ink ratio, and the principles of minimal ink, as well as precision, accuracy, and scalability.

2.6.1 Expressiveness and Effectiveness

- **Expressiveness**: Visualisations should be expressive, meaning they should effectively communicate the intended message or insights within the data. Expressive visualisations capture the richness and complexity of the underlying data, revealing patterns, trends, and relationships.
- **Effectiveness**: An effective visualisation is one that successfully conveys information to its audience. It allows viewers to understand the data, draw meaningful conclusions, and make informed decisions based on the presented information.

2.6.2 Data-Ink Ratio and the Principle of Minimal Ink

REVISE REFERENCES!! This **Data-Ink Ratio principle**, introduced by Edward Tufte, emphasises maximising the ink (or pixels in digital formats) used to represent the actual data while minimising non-essential ink. A higher data-ink ratio results in a cleaner, more efficient visualisation that reduces clutter and enhances comprehension.
The **Principle of Minimal Ink** builds on the data-ink ratio. This principle advocates for the removal of any visual elements that do not contribute to the viewer's understanding of the data. Eliminating unnecessary ink (e.g., excessive gridlines or decorations) simplifies the visualisation without sacrificing its effectiveness.

2.6.3 Precision, Accuracy, and Scalability

- **Precision**: Precision in data visualisation refers to the level of detail and granularity in the representation of data. Visualisations should strike a balance between showing enough detail to support accurate interpretation while avoiding overwhelming viewers with excessive complexity.
- **Accuracy**: Accuracy pertains to how faithfully the visualisation represents the true values in the data. Misleading or distorted visualisations can lead to incorrect conclusions. Therefore, maintaining accuracy is essential.
- **Scalability**: Scalability addresses how well a visualisation adapts to different data sizes or resolutions. Effective visualisations should be scalable, and capable of representing both small and large datasets without sacrificing clarity or performance.

2.7 Cognitive Load and Visual Complexity

In data visualisation, achieving a balance between complexity and cognitive load is crucial. This section explores the concept of cognitive load in visualisations, strategies to reduce cognitive load while maintaining complexity, and techniques to combat information overload through simplification.

2.7.1 Exploring the Concept of Cognitive Load in Visualisations

In data visualisations, cognitive load plays a significant role in how viewers engage with and understand the presented data. It is essential to strike a balance that ensures the visualisation conveys information effectively without overwhelming or overtaxing the viewer's cognitive capacity.

2.7.2 Strategies to Reduce Cognitive Load While Maintaining Complexity

- **Visual Hierarchy**: Establish a clear visual hierarchy that guides viewers' attention to the most important elements of the visualisation. Use techniques such as size, colour, and contrast to emphasise key information.
- **Simplify Labels and Text**: Reduce cognitive load by using concise labels and text. Avoid jargon and unnecessary complexity in annotations, ensuring that labels are informative and straightforward.
- **Interactive Features**: Implement interactive elements, such as tooltips and drill-down functionality, to provide additional information when viewers need it, reducing the need for dense, static visualisations.
- **Progressive Disclosure**: Present complex information gradually, allowing viewers to digest it in stages. Start with an overview and provide opportunities for users to explore details as needed.
- **Data Aggregation**: Consider aggregating data when it makes sense. Summarising data can reduce the cognitive load associated with interpreting fine-grained details.

2.7.3 Information Overload and Simplification Techniques

Information overload occurs when a visualisation overwhelms viewers with excessive data or visual elements, hindering comprehension. To combat information overload, the following simplification techniques can be applied:

- **Filtering**: Allow viewers to filter data based on specific criteria, enabling them to focus on the most relevant information.
- **Data Reduction**: Aggregate or summarise data to present overarching trends or patterns instead of inundating viewers with raw data points.
- **Studyboarding**: Use storytelling techniques to guide viewers through the data in a structured manner, helping them understand the context and relevance of the information presented.
- **Prioritisation**: Identify the most critical information and prioritise its display, relegating less essential data to secondary views or interactions.

3 Modern Methods of Data Visualisation

In this chapter, we explore a variety of powerful visualisation methods, from classic scatter plots and bar charts to advanced techniques like heatmaps and network graphs. Through vivid examples, we'll show when and why each method is used, and delve into the theoretical and mathematical foundations that empower these visualisations to unveil insights hidden within the data.

3.1 Introduction to Modern Data Visualization Methods

As data grows increasingly complex and vast, the tools and techniques for effectively conveying this information continue to expand and refine. In this section we introduce the data sets that will be used to illustrate each of the different visualisatio techniques.

3.2 Scatter Plots and Bubble Charts

Scatter plots and bubble charts are fundamental data visualisation techniques that provide valuable insights into the relationships and patterns within datasets. These visualisations are particularly effective for representing discrete data through data points, since this bring out easily identifiable comparisons, and reveals trends.

3.2.1 Scatter Plots

Scatter plots, also known as dot charts or dot density plots, offer a straightforward yet mathematically intriguing method for visualizing data. At their core, they display individual data points as dots along a single axis, where each dot representing a single observation.

The mathematical interest of dot plots lies in their ability to provide a simple visual representation of data distribution, center, and spread. While they don't rely on complex equations or statistical principles, dot plots make it easy to observe important characteristics of data, such as the mode (the most frequent value), skewness (asymmetry), and potential outliers.

They're particularly useful for comparing multiple data sets, identifying patterns, and detecting data irregularities. Their simplicity is what makes dot plots a valuable tool for both introductory statistics education and exploratory data analysis.

3.2.2 Scatter Plots in Practice

In this example, we'll create a scatter plot that visualizes the relationship between two variables - the weight of cars and the amount of miles traveled per gallon of petrol. We'll use the "mtcars" R dataset.

```
## `geom_smooth()` using formula = 'y ~ x'
```

3.2.3 Analysis

In Figure 1, provides insights into the relationship between cars' weight and their MPG, with the added dimension of color-coded cylinders.

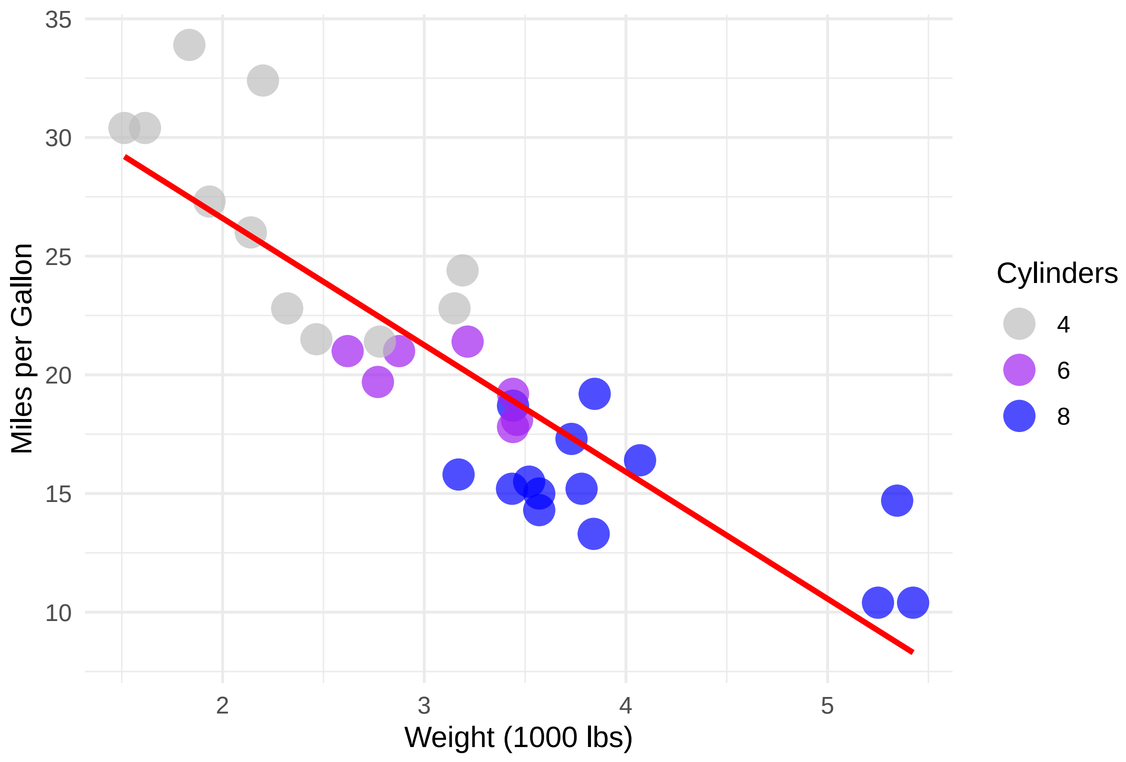


Figure 1: Scatter plot of car weights vs MPG

Clusterings
The scatter plot reveals distinct clustering of data points, highlighting specific patterns within the dataset. Cars with four cylinders (color "grey") are predominantly clustered in the lower weight and higher MPG region, representing smaller and more fuel-efficient vehicles. In contrast, cars with eight cylinders (color "blue") tend to be clustered in the higher weight and lower MPG area, indicating larger and less fuel-efficient cars. The identification of this clustering aids in visualising how the number of cylinders influences the trade-off between weight and fuel efficiency.

Linear Regression Line
The regression line provides a visual representation of the overall relationship between car weight and fuel efficiency.If the line has a positive slope, it indicates that as car weight increases, MPG decreases. Conversely, a negative slope suggests that heavier cars tend to have higher MPG. The steepness of the line represents the strength of this relationship. In this case, the reed regression line indicates a negative correlation—cars tend to have lower fuel efficiency as their weight increases.

3.2.4 Regression and the Regression Line

Linear regression is a fundamental statistical method used to model the relationship between a dependent variable (often denoted as Y) and one or more independent variables (commonly denoted as X). The objective of linear regression is to find a linear equation that best represents this relationship. In simple linear regression, with one independent variable, the linear regression line is expressed as:

$$Y = \beta_0 + \beta_1 X + \epsilon$$

Here, β_0 is the intercept, β_1 is the slope, X is the independent variable, and ϵ represents the error term. The objective of the linear regression line is to minimise the sum of squared differences between the observed and predicted values of Y, which helps us understand how changes in X affect Y.

3.2.5 Bubble Charts

Bubble charts are a captivating data visualization tool that extends beyond the typical two-dimensional scatter plot by introducing an extra dimension. They represent data points as bubbles or circles on a two-dimensional plane, where the size of each bubble encodes a third variable. This technique enhances data visualization by facilitating the exploration of multivariate data and uncovering patterns that may be hidden in traditional scatter plots.

Bubble Chart's Utility in Visualising Data
Bubble charts excel in scenarios where three key variables need to be conveyed simultaneously. The x-axis and y-axis represent two variables, as in a standard scatter plot, while the size of the bubble encodes a third variable, often a quantitative one. This allows for the visualization of relationships between three variables in a single, intuitive graphic.
For instance, in economics, bubble charts can illustrate economic indicators, with the x-axis showing time, the y-axis displaying GDP growth, and the bubble size representing a related factor like population or inflation.

Mathematical Intricacies
The mathematical intricacies of constructing bubble charts involve scaling the data values to determine the size of each bubble accurately. The size of the bubble is typically proportional to the square

root of the variable it represents. The choice of scaling method depends on the data distribution and the message the chart aims to convey.

The formula for calculating the bubble size (S) often involves applying a linear or nonlinear scaling function:

$$S = k \cdot \sqrt{V}$$

Where:

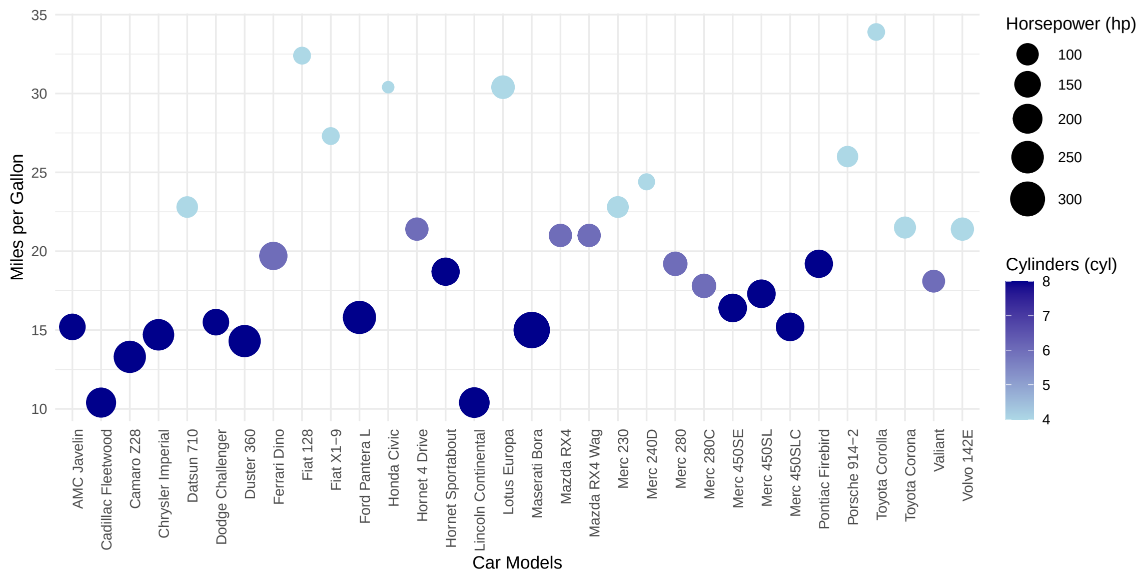
- S is the size of the bubble,
- V is the value of the variable being represented, and
- k is a scaling factor to control the bubble size.

Selecting an appropriate scaling factor (k) is critical for maintaining the proportionality between the bubble size and the variable being represented.

3.2.6 Bubble Charts in Practice

This bubble plot visualizes data from the same dataset as above. The purpose of this plot is to depict the relationship between car models and their fuel efficiency (mpg) while using the size of the bubbles to represent the car's horsepower (hp) and color-coding the bubbles based on the number of cylinders (cyl).

```r
#Create bubble plot
ggplot(mtcars, aes(x = rownames(mtcars), y = mpg, size = hp, color = cyl)) +
  geom_point() +
  labs(
    x = "Car Models",
    y = "Miles per Gallon",
    size = "Horsepower (hp)",
    color = "Cylinders (cyl)"
  ) +
  scale_size_continuous(range = c(3, 10)) +
  scale_color_gradient(low = "lightblue", high = "darkblue") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust =1))
```



3.2.7 Analysis

The plot's title, axis labels, and legends provide context and clarity to the visualization, making it accessible and informative. Additionally, the choice of a gradient color scale for the number of cylinders enhances the visual appeal and aids in interpreting the data.This bubble plot allows for quick comparisons between multiple characteristics of different car models.The resulting bubble plot effectively conveys several key insights:

1. **Car Model vs. MPG**: The x-axis displays the car models, offering a clear representation of each vehicle in the dataset. The bubble plot is particularly useful for displaying nominal data, such as car model names, as it allows easy identification and comparison.

2. **Miles per Gallon (MPG)**: The y-axis measures miles per gallon, representing the fuel efficiency of each car model. Higher bubbles indicate better fuel efficiency. This variable, which is continuous, is positioned vertically to demonstrate how each car model's fuel efficiency relates to others.

3. **Horsepower (HP)**: The size of each bubble represents the car's horsepower (hp). Larger bubbles correspond to higher horsepower, providing an additional dimension to the data. The size encoding helps identify more powerful cars.

4. **Cylinders (Cyl)**: The color of each bubble is determined by the number of cylinders (cyl) in the car's engine. The color scheme adds a categorical aspect to the visualization, making it easy to differentiate between cars with different cylinder counts.

3.3 Bar Charts and Histograms

3.3.1 Bar Charts

A bar chart is a very important method to present data. It organizes information into vertical bars. Bar charts have lots of advantages in data visualization. It can present data categories in a frequency

distribution. A bar chart is best for comparing classified data. Especially when the values are close, because the human perception of height is better than other visual elements (such as area, angle, etc.), the use of a bar chart is more appropriate. These bars usually have different lengths, and every length is proportional to the size of the information they present.
R uses the function *barplot*() to create bar charts. R can draw both vertical and Horizontal bars in the bar chart. In the bar chart, each of the bars can be given different colors.
R is a programming language for data analysis and statistical computing, and its advent has made data visualization more straightforward and accessible. Among the various tools available in R, ggplot2 stands out as one of the most renowned and powerful tools for creating data visualizations. It offers a wealth of data visualization capabilities and is celebrated for its versatility and aesthetic appeal. In this chapter, we will focus on how to use ggplot2 to create bar charts for data visualization.

3.3.2 Different Types of Bar Charts

Here is an overview of the different types of bar charts.

Vertical Bar Chart This is the most common bar chart. We use different vertical columns to display and compare the values of different categories in the same dimension, where the X-axis represents the contrasting categories and the Y-axis represents the frequency or count of their categories.

Horizontal Bar Chart This is very similar to a vertical bar chart but rotated 90 degrees. Categories are shown on the y-axis and frequency or count are shown on the x-axis. Horizontal bar charts are especially useful when category names are long or when there are numerous categories.

Multi-set Bar Chart Also known as a grouped bar chart or clustered bar chart. A multi-set bar chart is used to represent and compare different sub-groups within individual categories. This type of chart is useful when you want to show and compare multiple sets of data side-by-side. Multi-set Bar charts can be horizontal or vertical like the other normal bar charts, and the length of each bar represents the frequency or count of their categories.

Stacked bar chart Similar to bar charts, stacked bar charts are often used to compare different classes of values and, within each class of values, are divided into sub-classes, which are often referred to by different colors. Each segment's size is proportional to the frequency or count that it represents from the sub-category. The entire bar's length represents the cumulative total of all the sub-categories. However, it is very easy to get confused when there are too many categories.

3.3.3 Advantages of Bar Charts

1. **Clarity and Simplicity**: Bar charts are structurally simple, making them easy to read and understand, allowing audiences to quickly grasp key information.
2. **Effective Comparison**: They provide a visual representation that makes comparing the size or value of different categories straightforward, especially when comparing a limited number of categorical data.
3. **High Flexibility**: They can be used to represent any type of data, be it continuous or discrete.

4. **Multilevel Representation**: Stacked or grouped bar charts can be used to represent multiple data series.

3.3.4 Disadvantages of Bar Charts

1. **Limited Data Representation**: They might not be suitable for representing large datasets as things can get cluttered.
2. **Potential Misinterpretation**: Without a zero baseline, bar charts can be misleading as they might exaggerate differences.
3. **Overcomplexity with Many Categories**: If there are too many bars, it can be challenging to discern information effectively.
4. **Requires Categorical Data**: Bar charts are not ideal for representing trends over continuous data, where line graphs might be more appropriate.

3.3.5 ToothGrowth Dataset

The *ToothGrowth* dataset is a built-in dataset in R. It is used to study the impact of vitamin C on the tooth growth of Guinea pigs. The dataset consists of 60 observations and 3 variables:

- **len**: Represents the length of the Guinea pigs' teeth.
- **supp**: Indicates the method of vitamin C supplementation, with two levels: "VC" for vitamin C in their diet and "OJ" for vitamin C in orange juice.
- **dose**: Denotes the dosage of vitamin C in milligrams per day, with three levels: 0.5, 1, and 2.

This dataset is commonly used for conducting statistical analyses such as analysis of variance (ANOVA) to determine if vitamin C supplementation significantly affects the tooth growth of Guinea pigs. It serves as a typical example for learning statistics and data visualization.

This bar chart illustrates the tooth growth in relation to varying doses of a vitamin. The key observations are:

1. **X-axis Description:** The X-axis represents different dosages of the vitamin (mg/day). There are three distinct dosage levels.
2. **Y-axis Description:** The Y-axis signifies the length of tooth growth (len). This represents the average tooth growth at the given vitamin dosage.
3. **Data Observation:** From the heights of the bars, it is evident that as the vitamin dosage increases, the tooth growth also appears to increase. This might suggest that higher doses of the vitamin may promote tooth growth.

```r
library(datasets)
# load ToothGrowth dataset from R
data(ToothGrowth)
ggplot(ToothGrowth, aes(x = factor(dose), y = len)) +# Create a bar chart with different vitamin d
  geom_bar(stat = "identity", position = "dodge", fill = "grey") +
  labs(title = "Tooth Growth by Dose", x = "Dose (mg/day)", y = "Tooth Length") +
  theme_minimal()
```

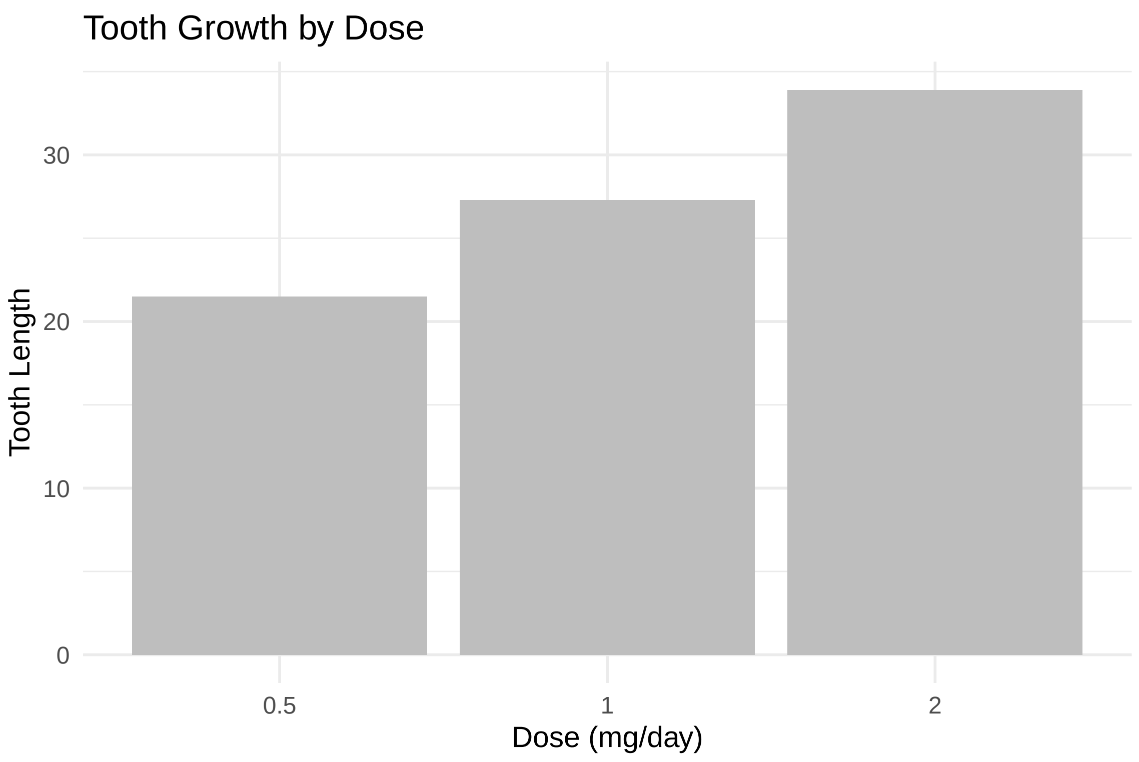


Figure 2: Tooth growth by dose

The provided bar chart elucidates tooth growth as influenced by varying doses of a vitamin and further differentiates based on the type of supplement. Below are the primary observations:

1. **X-axis Description:** The X-axis categorizes different dosages of the vitamin, with three distinct dosage levels: 0.5, 1, and 2 mg/day.
2. **Y-axis Description:** The Y-axis measures the length of tooth growth, representing the average growth at the specific vitamin dosage and supplement.
3. **Data Observation:** It's discernible from the heights and grouping of the bars that the tooth growth tends to increase with higher doses of the vitamin. However, the effectiveness seems to vary based on the type of supplement. A closer inspection would be required to determine which supplement is more effective at each dosage.

```r
ggplot(filtered_data, aes(x = factor(dose), y = len, fill = supp)) +
  geom_bar(stat = "identity", position = position_dodge(width = 0.7)) +
  labs(
    title = "Tooth Growth by Dose and Supplement",
    x = "Dose (mg/day)",
    y = "Tooth Length",
    fill = "Supplement"
  ) +
  theme_minimal()
```

The displayed bar chart provides insights into tooth growth influenced by varying doses of a vitamin, further categorized by the type of supplement (‘supp‘). The key insights from this chart are:

1. **X-axis Description:** The X-axis demarcates different vitamin dosages, categorized into three distinct levels: 0.5, 1, and 2 mg/day.
2. **Y-axis Description:** The Y-axis quantifies tooth growth length, representing the combined average growth for both supplements at the respective vitamin dosages.
3. **Data Observation:** The total height of each bar signifies the combined tooth growth for both supplements at the given dosage. From the stacked sections, it's evident that the impact on tooth growth varies based on the supplement type. A detailed inspection might elucidate the relative effectiveness of the supplements at each dosage level.

```r
ggplot(filtered_data, aes(x = factor(dose), y = len, fill = supp)) +
  geom_bar(stat = "identity", position = "stack") +
  labs(
    title = "Tooth Growth by Dose and Supplement",
    x = "Dose (mg/day)",
    y = "Tooth Length",
    fill = "Supplement"
  ) +
  theme_minimal()
```

In the next part of our section, we will look at another plot which called Histogram.

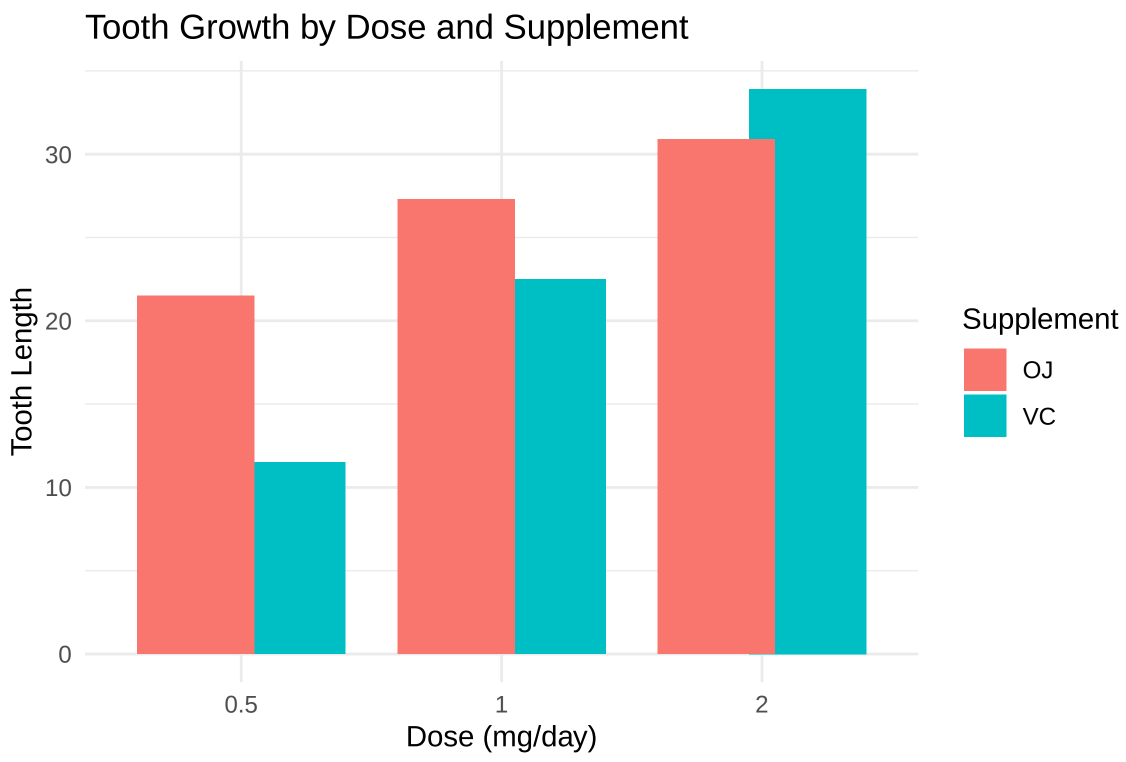


Figure 3: Tooth Growth by Dose and Supplement(grouping bar chart)

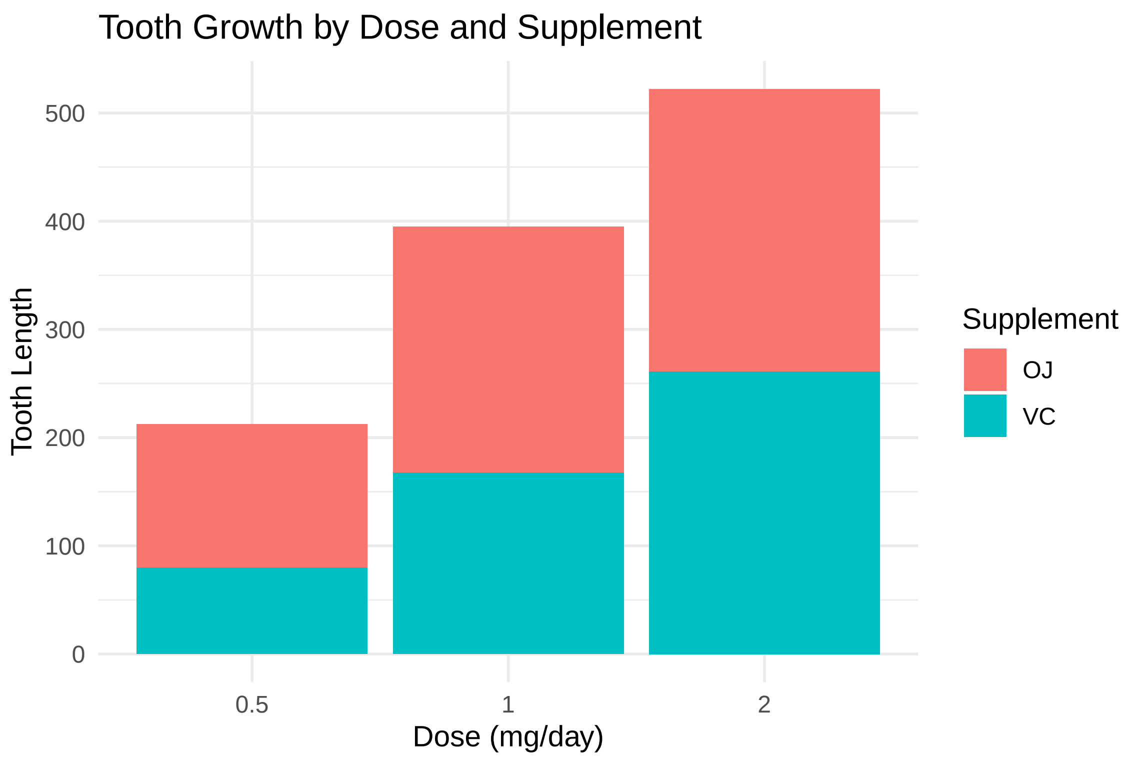


Figure 4: Tooth Growth by Dose and Supplement(stacked bar chart)

3.3.6 Histograms

Histograms, although visually similar to bar charts, convey different meanings. A histogram involves concepts of statistics. It requires data to be categorized into groups and then counts the data points within each of those groups. On a Cartesian coordinate system, the x-axis shows the endpoints of each group, and the y-axis represents frequency. The height of each rectangle indicates the corresponding frequency, making it a frequency distribution histogram. In order to determine the quantity of each group in the histogram, a multiplication of the frequency by the group interval is necessary. Since every histogram has a fixed group interval, if we use the y-axis to directly show quantity and each rectangle's height indicates the number of data points, we can both retain the distribution and simultaneously see the number in each group at a glance. All examples in this text use the non-standard histogram depiction with the y-axis denoting quantity.

3.3.7 Uses of Histograms:

Histograms demonstrates the distribution of frequency or quantity across groups.
Facilitates the visualization of differences in frequency or quantity among groups.
The R language uses the *hist*() function to create histograms. This function takes vectors as input and uses a few more parameters to plot the histogram.
Now, we want to creat a better graph with ggplot2 thanks to the $geom_histogram()$ function and iris dataset.
The iris dataset is a classic dataset in the field of statistics. It was introduced by the British biologist Ronald A. Fisher in 1936 as an example of discriminant analysis. The dataset consists of 150 samples from three species of iris flowers: setosa, versicolor, and virginica. For each sample, four features were measured: the lengths and the widths of the sepals and petals, all in centimeters. The dataset is often used for classification tasks to differentiate between the three species based on the given measurements. It has become a standard test case for many classification algorithms and is widely recognized in the data science community.

```r
bar_diagram <- ggplot(iris, aes(x = Sepal.Length)) +
  geom_histogram(
    binwidth = 0.2,  # Adjust the box width to 0.2 for smaller data sets
    fill = "grey",
    color = "black"
  ) +
  labs(
    title = "Histogram of Sepal Length in Iris Dataset",
    x = "Sepal Length/cm",
    y = "Frequency"
  ) +
  theme_minimal()
print(bar_diagram)
```

3.4 Heatmaps and Tree Maps

In this chapter, we explore two powerful data visualisation techniques: heatmaps and treemaps. These methods are instrumental for conveying intricate data structures and patterns, offering unique

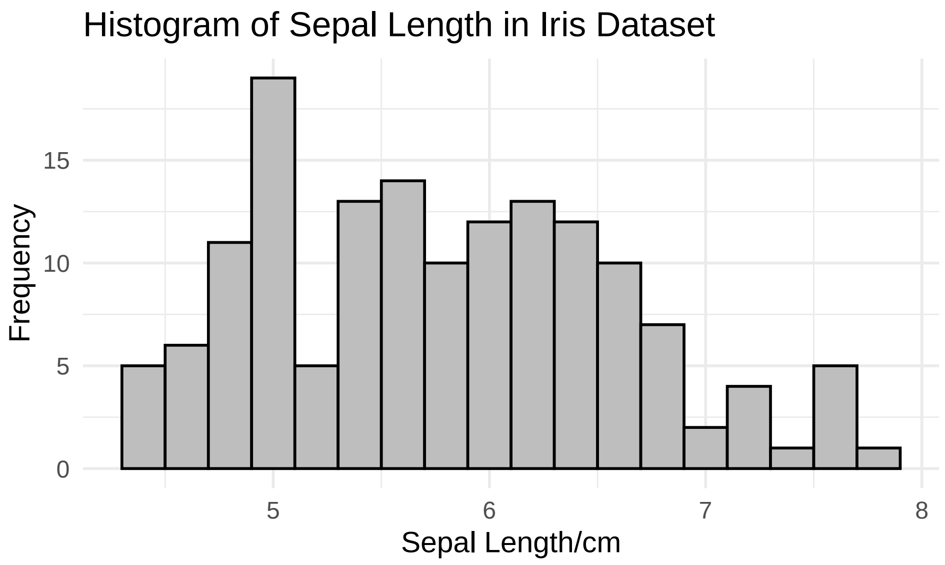


Figure 5: Histogram of Sepal Length in Iris Dataset

ways to represent multivariate information, making them indispensable tools for data scientists.

We will delve into the theory behind heatmaps and treemaps, understand how to create them using popular data visualization libraries, and demonstrate their practical applications with real-world examples. By the end of this chapter, you will be well-equipped to leverage heatmaps and treemaps to gain insights from complex and hierarchical datasets.

3.4.1 Heatmaps - Fire in Brazil

The heatmap is a data visualisation technique that uses colour coding to represent different intensity.

In this illustrative example, heatmaps are used to visualise fire occurrences in Brazil. These heatmaps provide a spatially coherent representation, highlighting regions at high risk and seasonal patterns. Here, the heatmap is a powerful tool for identifying the occurrence of fire incidents. The data-driven insights could empower policymakers to make informed decisions regarding preventive measures and firefighting strategies.

In Figure 6, it can be observed that significantly higher fire counts are found in certain locations. The presence of two strips with high frequencies of fires are highly unusual. The vertical trend corresponds to the location of BR-230 (Trans-Amazonian Highway) passing through the city of Apuí, State of Amazonas, where a high frequency of fire occurrence is observed. The horizontal trend corresponds to BR-163 (Brazil highway) passing through Três Pinheiros in Novo Progresso, State of Pará. The western coastal area with a high frequency of fire occurrence corresponds to regions in close proximity to the cities of Vista Alegre do Abunã and Rio Branco. Research has indicated that 95 % of active fires and the most intense ones (FRP ¿ 500 megawatts) occurred at the edges in forests.

From the same figure, it can be observed that August and September are the riskiest months in terms of fire hazard, whereas little risk is posed from November to July. The follow-up question naturally arises: How does FY22 compare to previous years? Is it valid to claim that August and September constitute the fire hazard season?

In Figure 7, the data shows a higher number of fire occurrences in the months of August to October compared to the rest of the year, indicating a greater number of fire hazards during these months.

```r
# Obtain the Brazil map data
brazil_map <- map_data("world", region = "Brazil")

# Create the heatmap of fire occurrences
space_heatmap <- ggplot(confident_fire_fy22, aes(x = longitude, y = latitude)) +
  geom_polygon(data = brazil_map, aes(x = long, y = lat, group = group),
               fill = "#bdbdbd") +
  geom_bin2d(bins = 300) +
  scale_fill_gradient(low = "#fee6ce", high = "#d7301f") +
  coord_fixed(ratio = 1) +
  theme_minimal()+
  theme(axis.text = element_text(size = 9))

interactive_plot <- ggplotly(space_heatmap)

time_heatmap <- ggplot(confident_fire_months_fy22,
                       aes(x = abb_month, y = as.character(2022), fill = count)) +
  geom_tile(width = 0.9, height = 0.5) +  # Create the heatmap tiles
  scale_fill_gradient(low = "#fff7ec", high = "#d7301f") +
  labs(x = " ", y = " ", name = "count") +
  theme_minimal() +
  theme(axis.text = element_text(size = 9))

spacetime_fy22 <- grid.arrange(space_heatmap, time_heatmap, nrow = 2,
                               heights = c(2,0.5))

print(spacetime_fy22)

## TableGrob (2 x 1) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (2-2,1-1) arrange gtable[layout]
```

```r
heatmap_plot <- ggplot(pivot_table,
                       aes(x = factor(abb_month, levels = custom_order),
                           y = as.character(year), fill = count)) +
  geom_tile() +
  scale_fill_gradient(low = "#fff7ec", high = "#d7301f") +
```

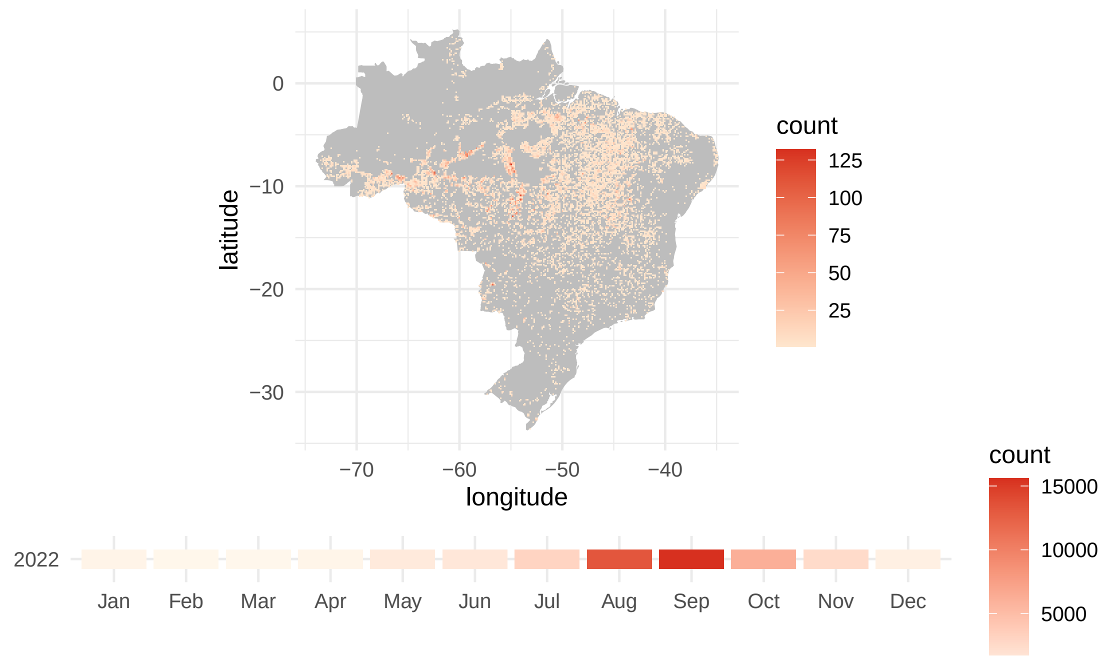


Figure 6: Frequency of Fire by Space and Time, FY22

```
  labs(x = " ", y = " ") +
  theme_minimal() +
  theme(axis.text = element_text(size = 9))

print(heatmap_plot)
```

3.4.2 Heatmaps, correlation matrix and AIC score

The foundation of a heatmap is a data matrix M, where each entry in this matrix represents an observation.

$$M = \begin{bmatrix} M_{11} & M_{12} & \dots & M_{1j} \\ M_{21} & M_{22} & \dots & M_{2j} \\ \vdots & \vdots & \ddots & \vdots \\ M_{i1} & M_{i2} & \dots & M_{ij} \end{bmatrix}$$

Therefore, the first step to create a heatmap is to organize the data into columns and rows. In Figure 7, the structured data is displayed as a grid of coloured cells, where the colour intensity corresponds to the underlying frequency.

Heatmaps serve as powerful tools for visualizing relationships between covariables within a model. An example of the necessity to analyze a matrix of correlations between variables is found in regression models. In the real world, variables are often correlated, and completely independent relationships are seldom encountered. Therefore, the analysis of pairwise correlations becomes essential.

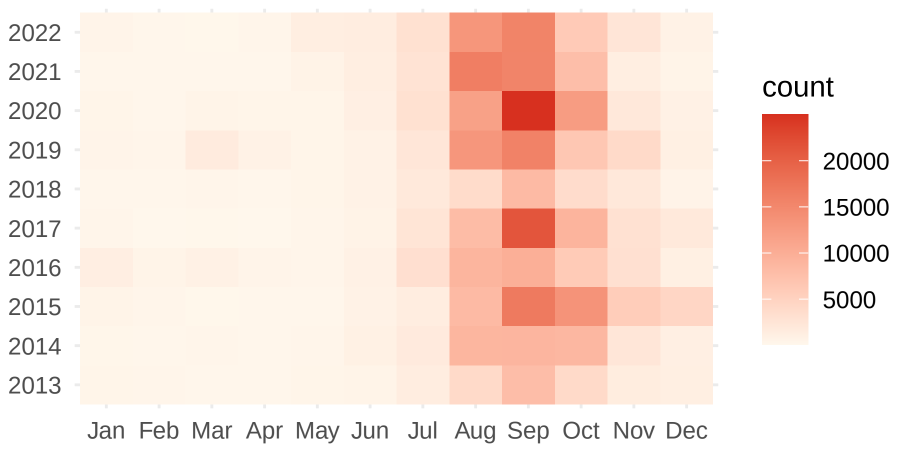


Figure 7: Frequency of Fire Occurrences, FY13-22

Significantly impacted by highly correlated variables, the regression model requires the selection of one variable from the correlated set. The selection is based on the identification of a regression model with the lowest Akaike Information Criterion (AIC) score among these variables.

$$AIC = -2l(\hat{\theta}) + 2\dim(\theta)$$

where $l(\hat{\theta})$ is the log-likelihood function, which is used to find the Maximum Likelihood Estimator (MLE) of a distribution.

The AIC measures the extent to which the linear model fits the dataset. To obtain the best model, minimise the AIC score. In other words, the objective is to have the trend explained by the regression model, while avoiding overfitting that captures the noise in the dataset, ultimately leading to inaccurate predictions, as demonstrated in Figure 8.

Figure 8: Danger of overfitting the regression model

3.4.3 Treemaps

Treemaps are a visualisation method specifically designed for hierarchical data structures. They represent data as nested rectangles, where each rectangle represents a part of the whole. Treemaps offer a visually appealing and efficient way to convey the hierarchical composition of data. The size and color of each rectangle can be used to encode additional information.

3.4.4 Use Cases for Treemaps

Treemaps are highly effective when dealing with hierarchical data. Some common use cases include:

- **Disk Space Visualization**: Treemaps can be employed to visualize disk space usage, where the outermost rectangle represents the entire disk, and inner rectangles represent folders and files. The size of each rectangle reflects the space they occupy.
- **Market Share Analysis**: In business, treemaps are useful for visualizing market share data. The top-level rectangle represents the total market, and inner rectangles represent individual segments, brands, or products. The size and color of each segment can represent its share and performance.

XXX

3.5 Line Charts and Time Series Visualization

In this chapter, we are going to investigate the intricacies of the line chart and its most common application: time series. First, a line chart is a statistical representation that uses a Cartesian coordinate system, where each point on the chart corresponds to a pair of coordinates (x, y), to depict changes in numerical values over continuous time intervals or ordered categories. The x-axis typically represents these intervals or categories, while the y-axis conveys quantified data. Hence, data points, represented by a coordinates $\{(x_i, y_i)\}_{1 \leq i \leq n}$, n being the total number of data points. In a line chart, consecutive data points are typically connected by straight lines. The line segment between two points (x_i, y_i) and (x_{i+1}, y_{i+1}) can be described by the equation of a line in the slope-intercept form: $y = mx + b$, where m is the slope and b is the y-intercept.

Suitability for Displaying Trends Over Time:
Line charts effectively visualize data trends over time. By plotting data at intervals like days or years, they highlight trends and patterns. Multiple lines on one chart enable easy data comparison, such as contrasting sales of two products. They aid in recognizing seasonal changes, cyclic events, and unexpected shifts, making them invaluable for forecasting. Due to their simplicity, they're accessible to those with minimal data analysis background. Line charts are a prime choice for time series visualization.

Time series visualization is essential in data analysis, showcasing time-ordered data. It reveals long-term trends, helping analysts discern patterns for future planning. It's crucial for spotting seasonality in datasets. This method also identifies anomalies, suggesting areas needing investigation. Predictive modeling, based on historical patterns, becomes feasible, fostering proactive choices. Overlaying multiple data series offers richer comparative analysis. Overall, time series visualization provides quick insights into chronological data, driving informed decisions by highlighting trends, seasonal changes, and outliers.

Limitations:
While line charts are excellent for displaying trends over time, they have limitations. They may not be suitable for showing individual data distributions or for data where there's no logical order. eg. too many points, too many lines, too many zeros.

3.5.1 Showcase real-world examples of time series visualisations

Here, we are going to investigate exchange rate data set and plot all daily and 21-day moving average exchange rates in one figure.

```
# Plot exchange rates of CNY, CAN, EUR, HKD, USD to GBP

# First plot
p1 <- ggplot(plot_dt, aes(x=Date, y=Rate, color=Currency)) +
  geom_line() +
  labs(title="Exchange rates of CNY, CAN,\n  EUR, HKD, USD to GBP",
       y="Exchange Rate to GBP",
       x="Date",
       color="Currency")+
  theme_minimal()+
  theme(legend.position="none")

# Second plot
p2 <- ggplot(plot_data, aes(x=Date, y=Rate, color=Currency)) +
  geom_line() +
  labs(title="21-Day moving average \n of exchange rates",
       y="Exchange Rate to GBP",
       x="Date",
       color="Currency")+
  theme_minimal()+
  theme(legend.position="none")
```

From Figure 9, we can see the the daily CNY, CAN, EUR, HKD, USD versus GBP exchange in the same plot, which provide us an overview of the trend and comparison. Then we can make the plot smooth by taking averages, which enables easier recognition of trend.

From above plot, a 21-day moving average for each currency allows us to view the trend easily. However, from Figure 9, for example there are huge gaps between ”CNYtoGBP” and ”EURtoGBP”. This may not suitable for viewing their trends simutaneosly. If we want to display two different time series that measure two different quantities at the same time points, we can draw the second series again on the second Y-axis on the right side.

From Figure 10, the

3.5.2 Identification of anomalies

A common approach to anomaly detection in time series data is to use the Z-score. The Z-score measures how many standard deviations a data point is from the mean. Data points with a Z-score above a certain threshold (e.g., 2 or 3) can be considered anomalies.

```
#Plot of time series with anomalies

# Calculate the Z-scores for the EURtoGBP exchange rate
eur_to_gbp_series <- MyData$EURtoGBP
mean_val <- mean(eur_to_gbp_series, na.rm = TRUE)
std_val <- sd(eur_to_gbp_series, na.rm = TRUE)
z_scores <- abs((eur_to_gbp_series - mean_val) / std_val)

# Define a threshold for anomalies (e.g., Z-score > 2)
threshold <- 3
```

```
# Extract the legend
p2_legend <- cowplot::get_legend(p2 + theme(legend.position="bottom"))

# Combine the plots with adjusted widths using cowplot
combined_plot <- cowplot::plot_grid(p1, p2, labels = c("1", "2"),
                                    rel_widths = c(1, 1), nrow=1)

# Combine the plots and the legend
final_plot <- cowplot::plot_grid(combined_plot, p2_legend, ncol=1,
                                 rel_heights = c(1, .1))

print(final_plot)
```

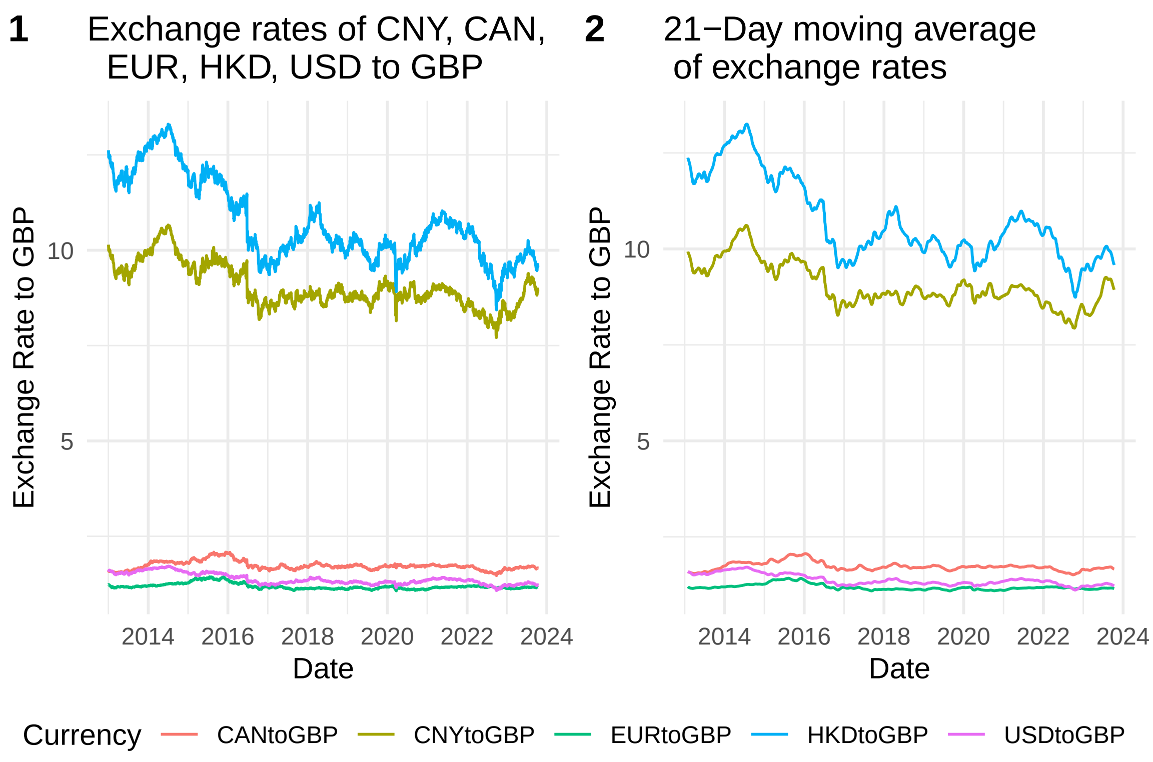


Figure 9: 'Daily and 21-day moving average exchange rates of CNY, CAN, EUR, HKD, USD to GBP'

```r
# Plot Double y-axis time series of CNY and EUR to GBP
# Define scale factor
scale_factor <- max(MyData$CNYtoGBP) / max(MyData$EURtoGBP)

df <- MyData %>% select(Date,CNYtoGBP, EURtoGBP)

# Create plot with dual y-axes
p <- ggplot(df, aes(Date)) +
  geom_line(aes(y = EURtoGBP, color = "EURtoGBP")) +
  geom_line(aes(y = CNYtoGBP / scale_factor, color = "CNYtoGBP")) +
  scale_y_continuous(
    name = "EURtoGBP Exchange Rate axis",
    sec.axis = sec_axis(~ . * scale_factor, name = "CNYtoGBP Exchange Rate axis")
  ) +
  labs(title = 'Double y-axis time series of CNY and EUR to GBP',
       color = "Currency") +
  theme_minimal()


print(p)
```

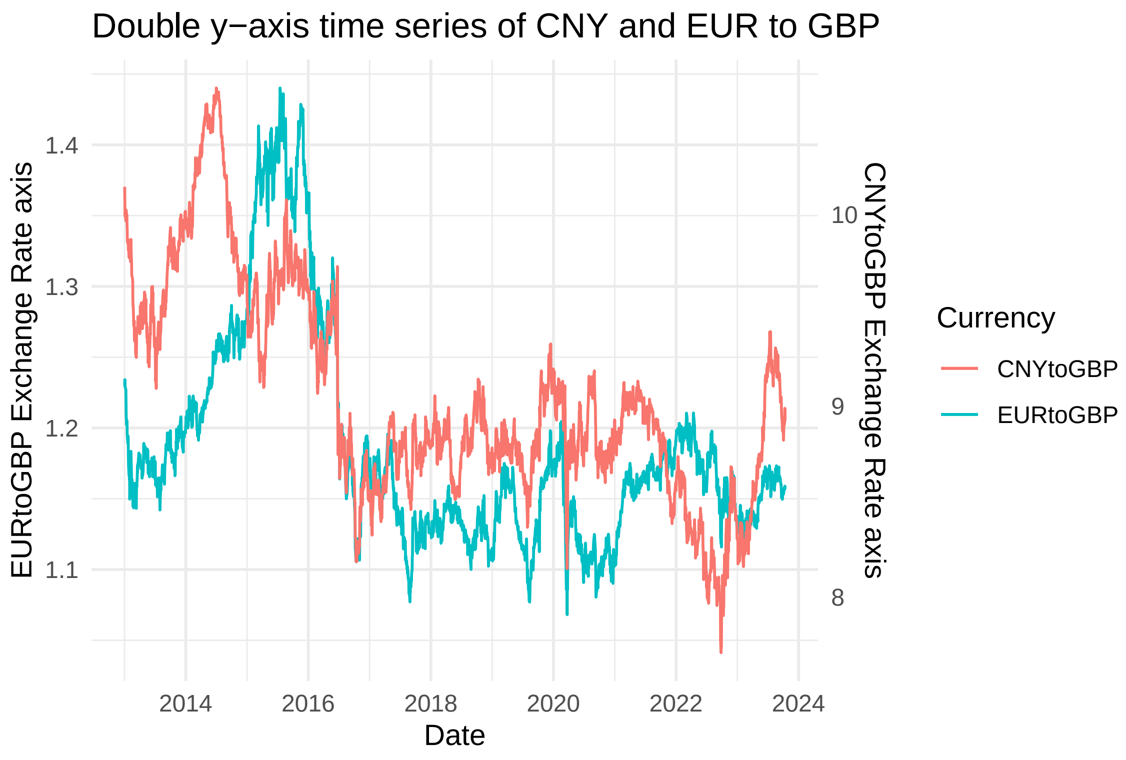


Figure 10: 'Double y-axis time series of CNY and EUR to GBP'

```r
ggplot(plot_data_z, aes(x = Date, y = Rate)) +
  geom_line(aes(color = "EURtoGBP")) +
  geom_point(aes(y = Anomaly, color = "Anomalies"), na.rm = TRUE) +
  labs(title = "EURtoGBP Exchange Rate with Anomalies Highlighted",
       y = "Exchange Rate to GBP",
       color = "Legend") +
  scale_color_manual(values = c("EURtoGBP" = "blue",
                                "Anomalies" = "red"))+
  theme_minimal()
```

Figure 12: 'Time series with anomalies'

```r
anomalies <- which(z_scores > threshold)

plot_data_z <- data.frame(Date =  MyData$Date,
                          Rate = eur_to_gbp_series,
                          Anomaly = ifelse(z_scores > threshold,
                                        eur_to_gbp_series, NA))
```

3.5.3 Decomposition of one time series into trend, seasonal, and random.

One of the primary advantages of time series visualization is the ease with which it allows analysts to identify long-term upward or downward trends in data and patterns that repeat over specific intervals. By decomposing the time series, it would be easy to see those features.

Time series data, Y_t, can often be described as a combination of several distinct components:

- **Trend (T_t):** The underlying progression in the series.
- **Seasonal (S_t):** Periodic fluctuations due to seasonal factor.
- **Residual (R_t):** The irregular or error component.

The decomposition of a time series can be described in two main models:
Additive Model: In the additive model, the components are added together:

$$Y_t = T_t + S_t + R_t$$

Multiplicative Model: In the multiplicative model, the components are multiplied together:

$$Y_t = T_t \times S_t \times R_t$$

In practice, the choice between the additive and multiplicative models often depends on the nature of the time series. If the magnitude of the seasonal fluctuations or the variation around the trend does not vary with the level of the time series, then an additive model is appropriate. If the magnitude of the seasonal fluctuations or the variation around the trend increases or decreases as the time series level changes, then a multiplicative model may be more suitable.

Hence, in this way, we will be able to view the seasonal trend and overall trend of CNY to GBP obviously.

```r
# Plot decomposition of addictive time series model
decomposed_ts <- stats::decompose(ts_data$CNYtoGBP)
plot(decomposed_ts, xlab="Date")
```

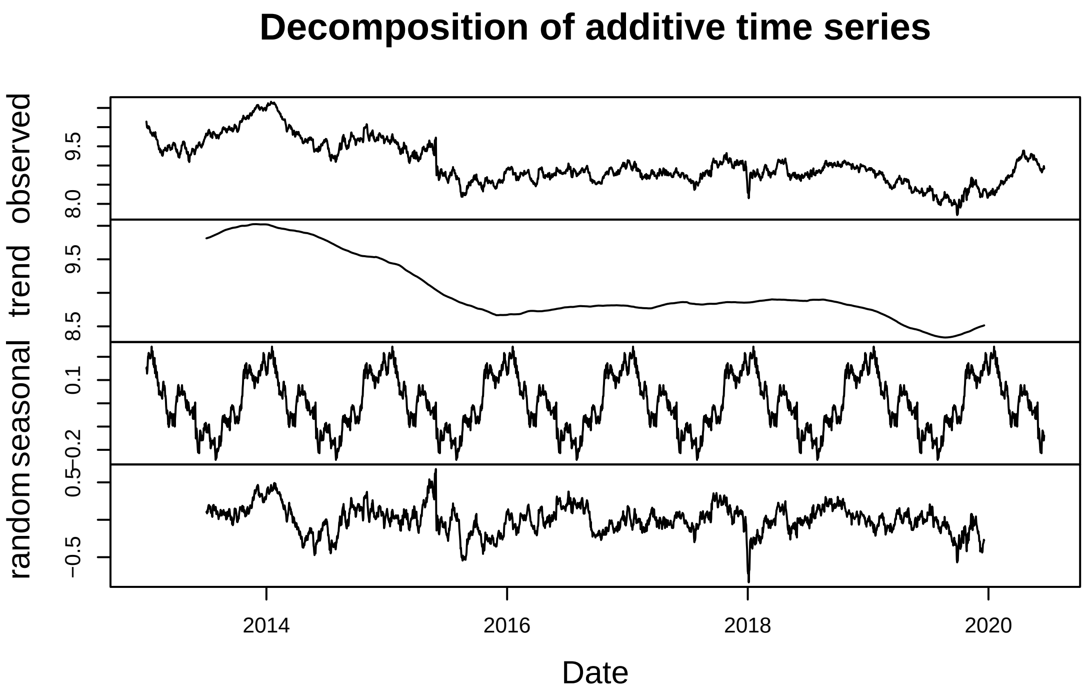


Figure 13: 'Decomposition of addictive time series model of CNY to GBP exchange rates'

3.6 Network Graphs

Definition and Utility: Network graphs, often referred to as graphs or networks, are a powerful data visualization method used to depict relationships between entities. These entities, known as nodes, are interconnected by edges or links, which represent relationships, connections, or interactions. Network graphs find extensive utility in various fields, such as social network analysis, transportation systems, and even biological networks like protein-protein interactions. They excel at revealing complex dependencies and structures, making them a critical tool for understanding relational data.

3.6.1 The Mathematics behind Network Graphs:

Constructing network graphs involves several mathematical intricacies. Here we present just a few of the many concepts that play a role in the creation of such graphs:

1. **Nodes and Edges**: Mathematically, a network graph, G, is defined as $G = (V, E)$, where V represents the set of nodes and E represents the set of edges connecting these nodes.
2. **Node Degree**: The degree of a node is the number of edges connected to it. In a directed graph, nodes can have both in-degrees and out-degrees.
3. **Centrality Measures**: Centrality metrics like degree centrality, betweenness centrality, and closeness centrality provide insights into the relative importance or influence of nodes within a network.
4. **Graph Metrics**: Graph theory concepts like shortest paths, connected components, and clustering coefficients are used to analyze the network's structure.

Formulas used in Network Graphs:

1. **Degree of a Node (Undirected Graph)**:

$$Degree(v) = \sum_{w \in V} A(v, w)$$

where $A(v, w)$ is the adjacency matrix element, indicating whether there is a connection between nodes v and w.

2. **Degree of a Node (Directed Graph)**:

$$In - Degree(v) = \sum_{w \in V} A(w, v)$$

$$Out - Degree(v) = \sum_{w \in V} A(v, w)$$

3. **Betweenness Centrality (for unweighted graphs)**:

$$C_B(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

where σ_{st} is the number of shortest paths from node s to t, and $\sigma_{st}(v)$ is the number of those paths passing through node v.

```
# Plot the graph
#plot(lesmis_graph, layout = layout, vertex.label.cex = 0.7, main = "Character Interactions in Les
```

3.6.2 Network Graphs in Practice

3.7 Sankey Diagrams

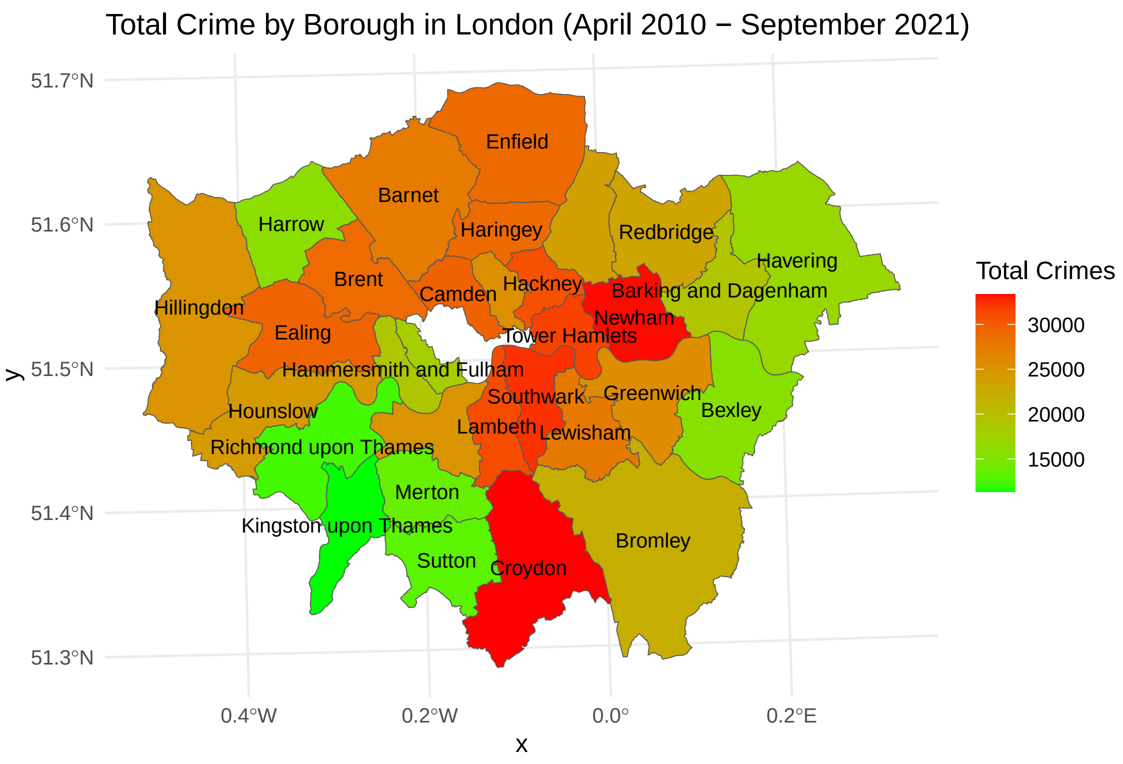
xxx

3.8 Geographic Maps and Spatial Data Visualisation

https://data.london.gov.uk/dataset/statistical-gis-boundary-files-london: :text=,2014.ziphttps://data.london.gov.uk/d
text $=$ *Recordedhttps* : //*public.tableau.com/app/profile/metropolitan.police.service/viz/MonthlyCrimeDataNew*
https : //*medium.com/@davemorison/an* $-$ *interactive* $-$ *visualization* $-$ *of* $-$ *londons* $-$ *crime* $-$ *data* $-$ *using* $-$ *sh*

```
# Load the shapefile
london_boroughs <- st_read("London_Ward.shp")

## Reading layer ‘London_Ward’ from data source
##   ‘C:\Users\chuyu\Downloads\Data-Visualisation-Project\London_Ward.shp’
##   using driver ‘ESRI Shapefile’
## Simple feature collection with 657 features and 6 fields
## Geometry type: POLYGON
## Dimension:     XY
## Bounding box:  xmin: 503568.2 ymin: 155850.8 xmax: 561957.5 ymax: 200933.9
## Projected CRS: OSGB36 / British National Grid

# Load the crime data (assuming it’s in CSV format)
crime_data <- read.csv("BoroughLevelCrime.csv")

# Summing the total crimes for each borough over the entire timeframe
borough_totals <- crime_data %>%
  group_by(LookUp_BoroughName) %>%
  summarise(Total_Crimes = sum(across(starts_with("X2020"))))

# Merge the crime data with the shapefile
merged_data <- merge(london_boroughs, borough_totals,
                     by.x="DISTRICT", by.y="LookUp_BoroughName")

# Aggregate geometries by district
aggregated_data <- merged_data %>%
  group_by(DISTRICT) %>%
  summarise(geometry = st_union(geometry), Total_Crimes = first(Total_Crimes))

# Calculate centroids for labeling
aggregated_data$centroid <- st_centroid(aggregated_data$geometry)
```

```r
# Plot the Crime numbers in London by boroughs
plot <- ggplot(data=aggregated_data) +
  geom_sf(aes(fill=Total_Crimes)) +
  geom_sf_text(aes(label = DISTRICT, geometry = centroid),
               size = 3, check_overlap = TRUE) +
  scale_fill_gradient(low="Green", high="red") +
  theme_minimal() +
  labs(title="Total Crime by Borough in London (April 2010 - September 2021)",
       fill="Total Crimes")

print(plot)
```



3.9 3D and Interactive Visualisations

ggplot2 is one of the most popular data visualization libraries in R, but it is primarily designed for 2D data visualization. Directly creating 3D views with ggplot2 can be challenging.

rgl: This is a widely-used package for 3D visualizations. It allows you to create interactive 3D scatter plots, line plots, and more, and view them in a separate window.

3D data visualization is an approach that employs three-dimensional graphics to represent complex data structures, allowing for an immersive exploration of information. Unlike traditional 2D visualizations (like bar graphs or line charts), 3D visualizations can convey an additional dimension of data, making them particularly valuable in specific contexts.

Our first example will be a scatter plot. We can use scatterplot3d package to help us for data visualization.

4 Trees Dataset in R

The `trees` dataset is one of R's built-in datasets. It contains measurements from 31 felled black cherry trees and provides insights into the relationship between a tree's girth, its height, and the volume of timber it can produce. The dataset comprises the following variables:

- **Girth:** The diameter of the tree, measured in inches at 4 ft 6 in above the ground.
- **Height:** The height of the tree, measured in feet.
- **Volume:** The volume of timber, measured in cubic feet, that the tree can produce.

```r
data(trees)
# Generate colors based on the Volume variable
colors <- colorRampPalette(c("blue", "red"))(length(unique(trees$Volume)))
color_assign <- colors[as.numeric(as.factor(trees$Volume))]
# Create 3d scatter plot with colors
scatterplot3d(trees$Girth, trees$Height, trees$Volume,
              color=color_assign,
              main="3D Scatterplot of trees data",
              xlab="Girth (inches)",
              ylab="Height (ft)",
              zlab="Volume (cubic ft)")
```

4.1 Advanced Visualisation Techniques

xxx

5 Practical Implementations

XXX

6 Case Studies

6.1 Market Analysis Dashboards

XXX

6.2 Healthcare Data Visualisation

XXX

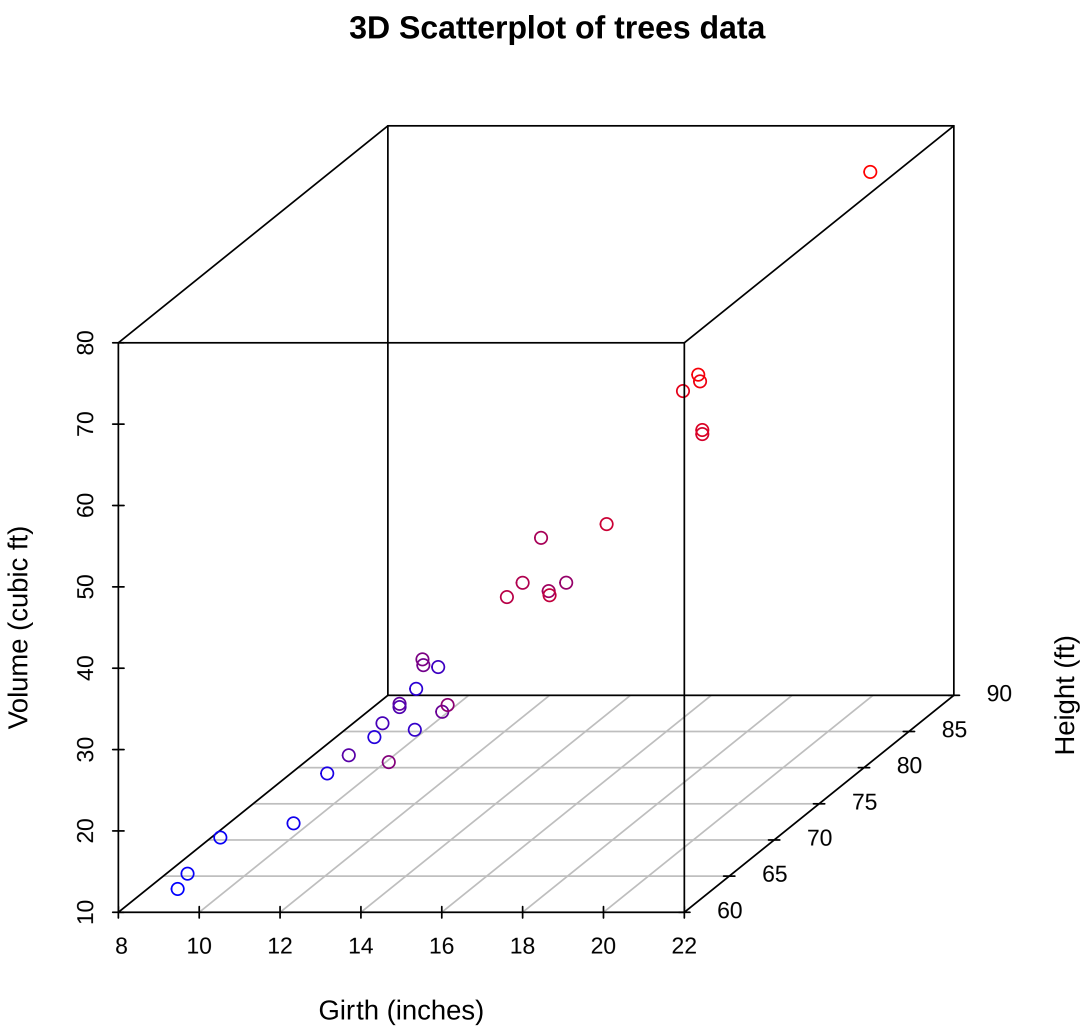


Figure 14: 3d scatter plot

7 State-of-the-Art Approaches

xxx

8 Conclusion

xxx