Data Visualisation: Theory and Practice

Yujie Chu, Pia Fullaondo, Qinqing Li, Jacko Zhou

February 25, 2024

Contents

Overview

Graphical statistics is an indispensable tool in the arsenal of every contemporary data project. Primarily, it enables one to delve into the data, explore its nuances, and effectively communicate empirical discoveries through visual means. This project aims to provide an overview of modern data visualisation methods, delve into their theoretical underpinnings, and demonstrate the potency of these methodologies using both synthetic and authentic datasets.

Furthermore, the project is aware of the ongoing evolution of novel data visualisation techniques, recognising it as an active area of research (e.g. Rodu and Kafadar, 2022). Thus, it aims to cover both textbook methods as well as state-of-the-art approaches and open problems.

The paper explores the theoretical foundations that underlie various visualisation methodologies, studies their mathematical construction, and illustrates their practical application. In particular, while chapter 2 focuses on the theoretical framework of data visualisation, the main body is made up of Chapters 3, 4, and 5. These chapters concentrate on standard implementations of graphical statistics for univariate, bivariate, and multivariate datasets, respectively.

Chapter 3 discusses Histograms, the Kernel Density Estimate, Bar Charts, Time Series Analysis, and the ROC Curve. In Chapter 4, the focus shifts to bivariate data, covering Heatmaps, Scatter and Bubble Plots, as well as regression, including Simple Linear Regression and LOESS Regression theory. Chapter 5 introduces multivariate data visualisation methods, presenting Principal Component Analysis, along with the theoretical foundation and practical applications of Biplots, Principal Curves and t-SNE. The final chapter, Chapter 6, examines active research areas in graphical statistics, with a particular emphasis on Box Plots.

Throughout the thesis, vocabulary referencing the programme R is used. The R package names such as *ggplot2* are italicised, while R commands like `faceting` are displayed in a teletype font. The R code chunks for each figure in the report can be found in a separate GitHub code folder.

1 Introduction

The introductory chapter serves to elucidate the motivation for this study, and to contextualise the paper. It underscores the enduring importance of data visualisation. Case examples both of the influential applications, but also dangerous misuses of data visualisations are showcased and analysed. These are sourced from BBC Ideas [19]. Futhermore, an introduction of essential R packages including *ggplot2* is provided, along with a presentation of the datasets used throughout paper.

Literature and Research
To appreciate the significance and impact of data visualisation, it is imperative to recognise its prominence within various method-specific publications. Notably, the Journal of Computational and Graphical Statistics stands out as a significant source of ongoing research in the realm of computational and graphical methods within statistics. This prestigious journal publishes ongoing research on the latest techniques in computational and graphical methods in statistics, encompassing data analysis and numerical graphical displays. Additionally, the Journal of statistical software publishes peer-reviewed articles about statistical software, together with the source code. Collectively, these publications reveal the vast potential for current and future research in this ever-evolving domain, particularly as our digital landscape and data sphere expand rapidly.

1.1 Historical Background and Misuses of Data Visualisation

This section begins with an analysis of two case studies demonstrating the effectiveness and influence of data visualisation, followed by an exploration of two instances illustrating problematic data misuse. The primary aim is to emphasise the importance of data visualisation while also stressing the need for careful attention to detail, especially considering the potential for malicious exploitation of visualised data.

Motivations for Using Data Visualisations — Case 1
Florence Nightingale was not only a social reformer and the founder of modern nursing but also a pioneering statistician. It was her application of data visualisation during the Crimean War that transformed the field of healthcare and pushed for social reform.

During the Crimean War, Nightingale recognised that unsanitary hospital conditions were claiming more lives than the battlefield itself. With the help of William Farr, Nightingale created the coxcomb aimed to illustrate the toll of preventable mortality on soldiers, as shown in Figure 1. The coxcomb, resembling an unconventional pie chart, partitioned mortality by causes. Blue indicates preventable deaths, red indicates deaths by wounds, and black indicates other causes. The blue areas outweighed the red and black sections combined, highlighting the disproportionate impact of unsanitary hospital conditions on the mortality rate.

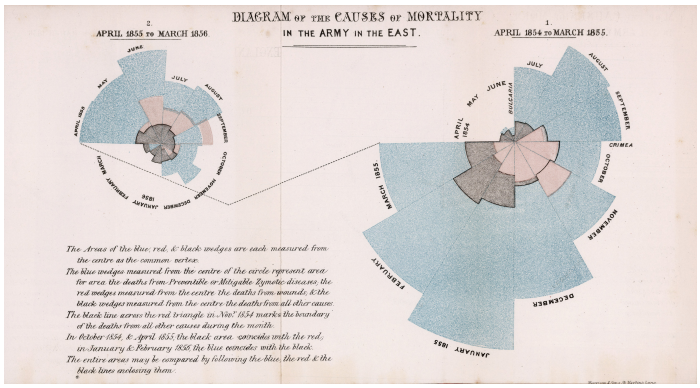

Figure 1: "Diagram of the causes of mortality in the army in the East" (1858) by Florence Nightingale [17]

Nightingale leveraged the compelling visualisations in her advocacy efforts, presenting them to MPs and government officials who otherwise are unlikely to read or understand statistical reports. She successfully persuaded Queen Victoria, head of the British Army at the time, to allocate funding for the improvement of better conditions in military hospitals.

Motivations for Using Data Visualisations — Case 2
Sometimes, one glance is enough to convey the most powerful idea. Edward Hawkins, a British climate scientist and Professor of climate science at the University of Reading, is renowned for his exceptional data visualisations of climate change.

In 2018, Edward Hawkins was invited to deliver a lecture on climate change in Wales to an audience with diverse backgrounds. It was important to effectively convey the growing urgency surrounding global warming. To achieve this, he created a chart that used only colours, without any words, titles, or legends, as shown in Figure 2. This seemingly simple yet remarkably powerful chart visually illustrated the Earth's warming trend since 1850.

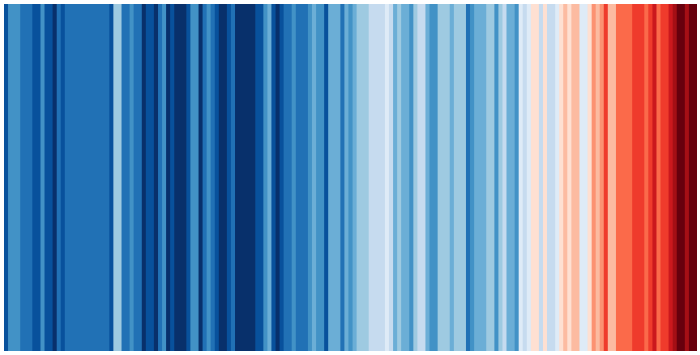

Figure 2: Latest global stripes (1850-2020) by Edward Hawkins [13]

Known as the "warming stripes," this chart cleverly employs blues to indicate cooler-than-average years and reds to signify hotter-than-average years. Its influence reached far and wide, gracing the

front pages of major media outlets and featured in news broadcasts worldwide. It became a symbol in climate change demonstrations. Arguably, it stands as one of the most iconic graphics in modern times.

Misuses of Data Visualisation — Case 1
Having observed the remarkable effectiveness of data visualisations, the significance of employing them correctly becomes apparent. The improper use of data visualisations holds the potential to significantly influence the public in misleading ways, resulting in undesired consequences.

Inappropriate data visualisation can conceal trends rather than reveal them. Figure 3 illustrates an instance of this issue. On the left-hand side, an inappropriate scale is used — the y-scale ranging from 0 to 30 million dollars, obscuring the fluctuations in payroll spending. Conversely, on the right-hand side, observe that there's a significant increase of over 500,000 dollars in just two months. This revelation is substantial; considering inflation, 500,000 dollars in 1937 is worth well over 10 million dollars today [1].

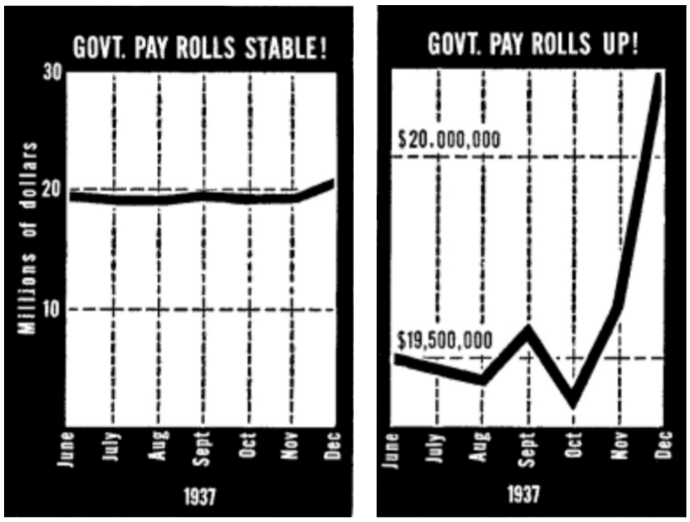

Figure 3: Incomplete Data Analysis, "How to lie with statistics" by Miguel de Carvalho [6]

Misuses of Data Visualisation — Case 2
One striking example of data visualisation misuse is found in the Kallikak Family tree — one of the most prominent eugenic narratives of the 20th century.

The visualisation, shown in Figure 4, was created by the psychologist Henry Goddard and presented in his 1912 book, "The Kallikak Family: A Study in the Heredity of Feeble-Mindedness." Goddard's narrative centered around Martin Kallikak, a soldier who, in addition to his marriage to a respected citizen, had a one-night stand with a "feeble-minded" maid. Goddard believed that intellectual disabilities were inherited traits. In Goddard's account, the legitimate family was successful, while the children of the "feeble-minded" maid were labeled as "the lowest types of human beings." However, research has since revealed that the entire story was fictitious, as there was no record of the maid's existence [24].

Regrettably, the Kallikak family tree became a central element in the eugenics movement. Figure 4 was featured in the 1935 Nazi propaganda film "Das Erbe" (The Inheritance), which was used to promote public acceptance of Nazi eugenics laws. This propaganda laid the groundwork for the forced sterilization of approximately 400,000 people under Nazi eugenics policies [22].

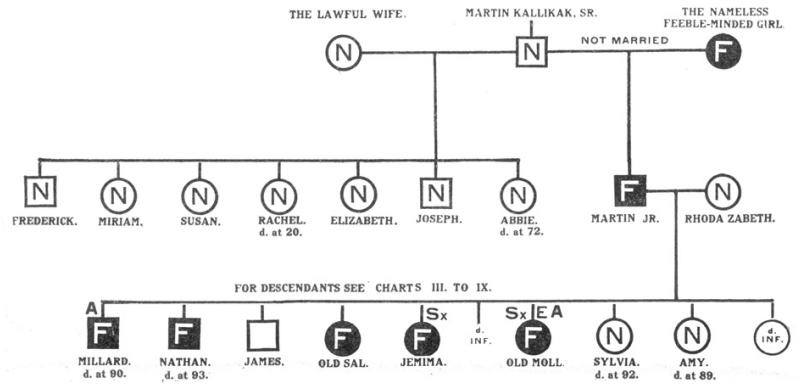

Figure 4: The Kallikak Family tree (1912) by Henry Goddard [4]

1.2 Computing and Data Visualisation

As a powerful tool for statistical analysis and graphic creation, the programme R offers a wide array of functionalities to cater to a range of data visualisation needs. We will first introduce the commonly used R package *ggplot2*, then we will present key packages mobilised to produce clear and compelling graphs.

ggplot2 is the main tool used to create plots representing various types of datasets. The foundation of *ggplot2* lies in the Grammar of Graphics, allowing the sequential construction of individual elements composing a graph. These elements are then combined to create a unified graphical representation.

The R package *ggplot2* is notable for its robustness and flexibility, enabling the creation of customised graphics rather than adhering strictly to pre-set options. Despite the initial learning curve, *ggplot2* is crafted to be user-friendly, offering sensible defaults and an iterative method for constructing plots. Its focus lies in uncovering the underlying message within the data. Furthermore, *ggplot2* is structured to support layered and annotated graphics that enhance data analysis, especially for beginners and those new to data exploration.

Key ggplot2 Commands
As shown in subsequent sections, it is possible to produce visualisations rather effortlessly by using internal data from R, or even external data, together with *ggplot2*. When using this package, certain key tools are used repeatedly to create visual plots from datasets. The tools below mentioned are integral functions within the ggplot2 package in R for creating sophisticated visualizations [31]:

`geom` refers to geometric objects, pivotal components within *ggplot2* used to define the visual representation of data in a plot. Each `geom` function corresponds to a specific graphical representation type in a chart. Additionally, in *ggplot2*, `scales` functions enable the adjustment of various mapping

details, including colour choices, label formatting, legend arrangement, and more.

`coord` provides the axes and gridlines which aid in interpreting the graph. Various coordinate systems such as cartesian, polar, and map projections are available. Furthermore, `faceting` is a robust feature enabling the partitioning of a single plot into multiple plots based on factors present in the dataset. This functionality proves particularly beneficial for exploring and presenting data with multiple groups or categories.

The `theme` function holds significance in customising the non-data elements of plots. The theme system within *ggplot2* allows precise adjustment of aesthetic aspects such as fonts, labels, legends, and background colours. This tool is essential for enhancing plot readability and creating visually captivating graphics tailored to specific audiences or publication requirements.

Key *ggplot2* Packages
The R language is a powerful tool for data analysis, owing to the wide array of R packages that facilitate diverse functions. Throughout this project, we will use many different packages, each enhancing the analysis and presentation of data.

The *tidyverse* package serves as a comprehensive collection of R packages tailored to provide a cohesive set of tools for data science. This collection includes several widely used packages such as *ggplot2* and *dplyr*. Particularly, the *dplyr* package plays a pivotal role in data manipulation and transformation. It enables us to filter, sort, summarise, and transform datasets, thereby streamlining data analysis and management.

Furthermore, the *Rtsne* package facilitates convenient analysis using the *Rtsne* function, particularly useful for visualising high-dimensional data. Lastly, the *qqboxplot* package aids in constructing Q–Q boxplots, which integrate both Q–Q plots and boxplots, offering a more comprehensive visual assessment tool.

1.3 Datasets

This section introduces the different datasets analysed and visualised in the subsequent chapters.

mtcars: The built-in R dataset, mtcars was extracted from the 1974 Motor Trend US magazine. It includes the fuel consumption and 10 aspects of automobile design for 32 automobiles (1973–74 models).

iris: The built-in R dataset iris was collected by Edgar Anderson. This dataset was famously used by British statistician Ronald Fisher to demonstrate linear discriminant analysis in 1936. It encompasses 5 flower characteristics for 3 types of iris, each with 50 samples.

palmerpenguins: The built-in R dataset palmerpenguins was provided by the Palmer Station Long Term Ecological Research. This dataset contains biometric measurements from three distinct penguin species inhabiting the Palmer Archipelago in Antarctica, namely the Adélie, Chinstrap, and Gentoo penguins.

ToothGrowth: The built-in R dataset, ToothGrowth, measures the tooth growth of 60 guinea

pigs. Each animal was either administered with vitamin C or orange juice.

Fire in Brazil: Open-source fire observation data is provided by NASA. The analysis focuses on Brazil (2013-2022). Note that the dataset contains the variable "confidence", ranging from 0% to 100%. The dataset was filtered with a confidence level of $\geq 95\%$ to ensure an accurate account of fire occurrences [7].

Exchange Rate: The exchange rate data, available at the Bank of England, provides daily spot exchange rates against the pound Sterling (2005-present). This report examines the daily spot rates of the Canadian Dollar, Euro, and US Dollar against the pound Sterling.

Taipei Housing: The historical real estate valuation from Sindian Dist., New Taipei City, Taiwan, available at UC Irvine Machine Learning Repository.

2 Theoretical Foundations of Data Visualisation

This chapter, "Theoretical Foundations of Data Visualisation," delves into the core principles and concepts that serve as the base of this 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

Creating effective data visualisations requires a robust theoretical framework underlying every chart, graph, or plot. These theoretical underpinnings not only form the basis of data visualisation but also influence how we represent, perceive, understand, and interpret data.

Guiding Principles for Data Representation
The theoretical framework of data visualisation involves guiding principles dictating visual representation of data. These principles include accuracy, emphasising faithful reflection of underlying data to reduce distortion or misinterpretation; simplicity, advocating for streamlined visuals to convey information effectively; clarity, ensuring visuals are easily understood without unnecessary complexity; relevance, presenting information pertinent to the message or question addressed; and consistency, maintaining uniform use of visual elements like colour coding and labeling throughout a visualisation.

Theoretical Framework and Visual Perception
Understanding how the human brain processes visual information is a fundamental aspect of data visualisation theory. This knowledge plays a crucial role in designing visualisations that effectively connect with viewers. It encompasses several key considerations which will be studied in order: the Gestalt Principles, which encompass proximity, similarity, and continuity, affecting how visual elements are grouped and interpreted; Colour Theory, involving the strategic use of colour contrasts and harmonies to improve clarity and impact; and the management of Cognitive Load, which emphasises the importance of reducing mental effort needed to process information.

2.2 Data Types and Visualisation Techniques

To have a discussion about data representation, understanding the nature of the data is key. Data comes in various types, and selecting the appropriate visualisation technique is contingent upon recognising these distinctions. In this section, the data types are categorised and matched with their suitable visualisation techniques.

2.2.1 Categorisation of Data Types

Data types can be broadly categorised into four main types:

Nominal data: represents categories or labels without any inherent order. Examples include colours, gender categories, and city names.

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: 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: includes ordered categories with equal intervals and a meaningful zero point. Examples are age, income, and weight.

Matching Data Types with Appropriate Visualisation Techniques

Selecting appropriate visualisation techniques is essential for effective data communication. Various data types demand specific visualisation methods for optimal representation [14]. For nominal data, bar charts and stacked bar charts are effective in displaying categorical information and relative proportions. Ordinal data benefits from ordered bar charts, dot plots, or stacked bar charts, maintaining the ranking and order of categories. Interval data is best visualised using line charts, histograms, and box plots, showcasing trends and distributions without assuming a true zero point. Finally, ratio data finds effective representation through scatter plots, histograms, and line charts, enabling precise comparisons and measurements due to the presence of a meaningful zero point. These are represented in Figure 5, where artificial data from a class of 4th-year university students is visualised.

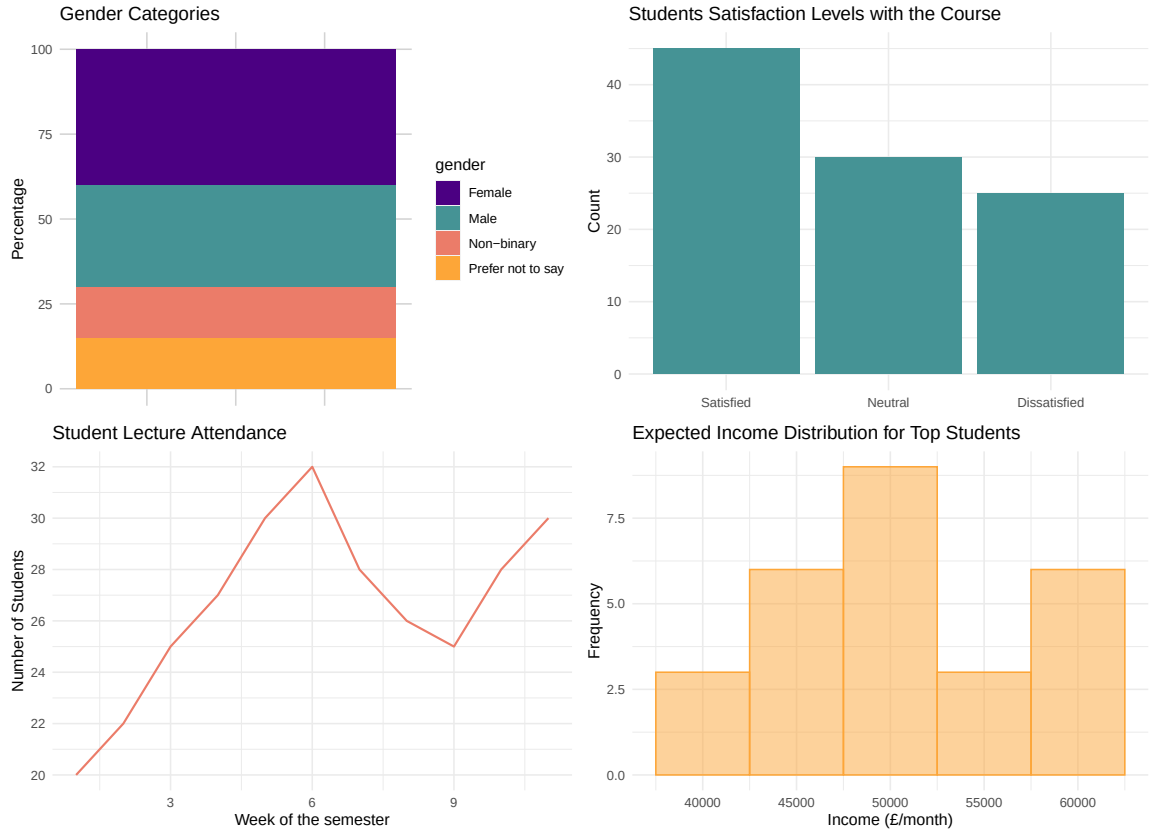

Figure 5: Different types of visualisation techniques according to the data type

2.3 Data Abstraction and Representation

The transformation of raw data into meaningful representations is a pivotal step in data representation. 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.

Data Abstraction: Transforming Raw Data
Data abstraction involves simplifying and structuring raw data into comprehensible and insightful formats. This process serves as the bridge, transforming numbers, text, and variables into visual elements that convey patterns, trends, and relationships, forming the core of informative data visualisations [28].

2.3.1 Hierarchies and Levels of Abstraction

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**: At the mid-level, 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.

These different levels of abstraction are represented in Figure 6. The first visualisations of the mtcars dataset is a scatter plot that provides detailed information about the relationship between car weight and miles per gallon, with points coloured by the number of cylinders. The second is an abstract visualisation using a box-and-whisker plot to provide a high-level summary of the distribution of miles per gallon for different numbers of cylinders. Finally, the third visualisation is a bar plot presenting aggregated information about the average miles per gallon for different numbers of cylinders.

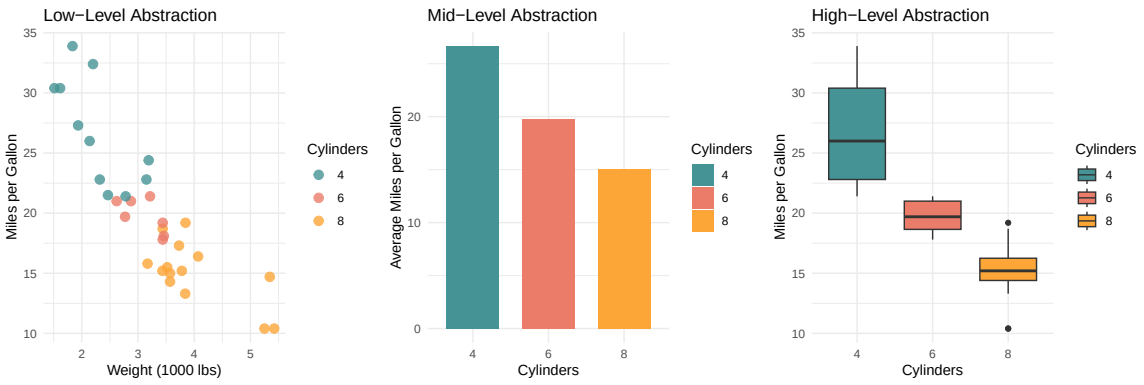

Figure 6: Mtcars dataset visualised on 3 different levels of abstraction

Trade-offs Between Abstraction and Information Loss
While abstraction simplifies complex data, it presents trade-offs. Designers of data visualisation

must strike a balance between clarity and detail, generalisation and specificity, and context versus precision. Abstraction increases clarity but may sacrifice crucial detailed information necessary for some analytical tasks. It offers a more generalised view accessible to a wider audience but might overlook specific nuances essential for experts. While providing valuable context, high-level abstraction may lack the precision required 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 Visual Perception and Cognition

In this first section, human visual perception is explored, along with the application of cognitive psychology principles in data visualisation.

Human Visual Perception: Decoding Visual Information
Human visual perception profoundly influences our understanding of the surrounding world. When applied to data visualisation, it shapes the way that individuals engage with and derive meaning from visual data representations.

Significant aspects of human visual perception within data visualisation encompass pattern recognition, adept at identifying trends, outliers, and relationships in data representations. Additionally, perceptual grouping, where visually similar elements are grouped together, influences the interpretation of data clusters and shapes. Moreover, the =hierarchy of perception dictates that certain visual attributes are processed more swiftly and effectively than others, such as colour being processed faster than text, influencing the viewer's attention hierarchy [5].

The Gestalt Principles
Furthermore, the Gestalt Principles play an important role in the realm of visual perception and design [21]. Key Gestalt principles crucial in shaping visual information perception include proximity, which groups related elements, similarity that links similar attributes, continuity aiding trend representation, closure for implying connections, and symmetry for balance and aesthetics in visualisations [27].

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.

2.5 Colour Theory in Data Visualisation

In this section, 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 are explored.

The Importance of Colour in Conveying Information
Colour significantly enhances the impact and comprehension of data visualisations. It serves multiple purposes: distinguishing data points, emphasising trends, and offering contextual information. It is utilised to encode categorical data, differentiating between various groups with distinct colours,

and to represent quantitative data by using colour intensity or gradients to portray values or magnitudes. Additionally, colour is instrumental in adding context to visualisations through background elements, labels, or annotations, imparting meaning to the data [14].

Colour Perception and Colour Encoding in Visualisations
Understanding colour perception in data visualisation is crucial. Key principles involve considering colour discrimination, ensuring accessibility for individuals with colour vision deficiencies, as is illustrated by Figure 7. Careful selection of colour schemes aligned with the intended message is essential — for instance, using warm colours like red and orange to indicate caution or warmth, and cool colours like blue and green to convey calmness or coldness. Additionally, attention should be paid to how colours interact when combined; certain combinations might create visual vibrations or impact text legibility.

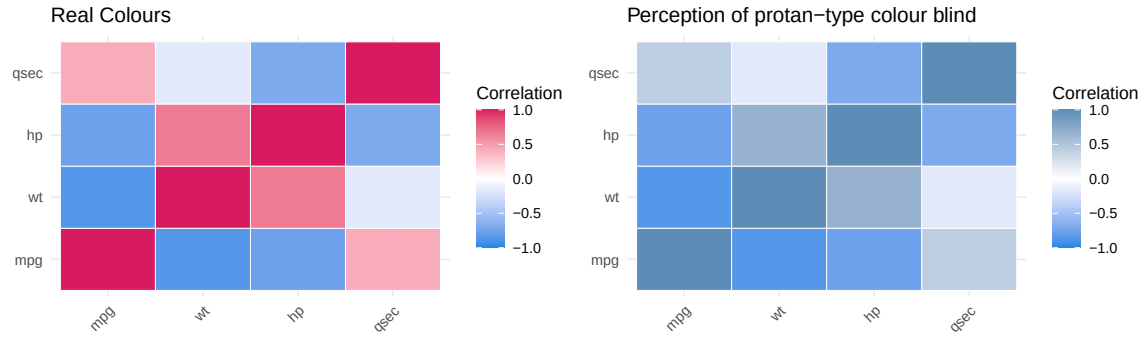

Figure 7: Colour perception of a heatmap for by a colour blind person

Avoiding Misleading Visualisations Due to Colour Choices
Misleading visualisations often stem from inappropriate or deceptive use of colour, requiring precautions to prevent such occurrences. First, maintaining consistency in colour usage throughout the visualisation is essential. Employing a uniform colour scheme for similar data categories or elements helps establish coherence and understanding. Furthermore, it's crucial to avoid colour choices that could distort or exaggerate the data. Overly intense or contrasting colours might mislead interpretations, emphasising the necessity for judicious colour selection.

Additionally, providing a clear and concise legend becomes imperative to explain the meaning of colours, especially when dealing with complex or unfamiliar colour schemes. A comprehensive legend helps viewers decipher the represented data accurately.

2.6 Cognitive Load and Visual Complexity

In data visualisation, achieving a balance between complexity and cognitive load is crucial. Cognitive load significantly influences how viewers engage with and comprehend presented data. Finding a balance is crucial to effectively convey information without overwhelming the viewer's cognitive capacity. 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.

2.6.1 Strategies to Reduce Cognitive Load While Maintaining Complexity

To reduce cognitive load while maintaining complexity in data visualisation, several strategies can be employed. Firstly, establishing a clear visual hierarchy using size, colour, and contrast helps direct attention to crucial elements. Additionally, simplifying labels and text by avoiding unnecessary complexity ensures information is clear and easily digestible.

Furthermore, employing interactive features like tooltips and drill-down functionality assists in providing additional information when required, reducing the density of static visualisations. A final approach involves the use of progressive disclosure, presenting complex information gradually, beginning with an overview and allowing users to explore details as needed.

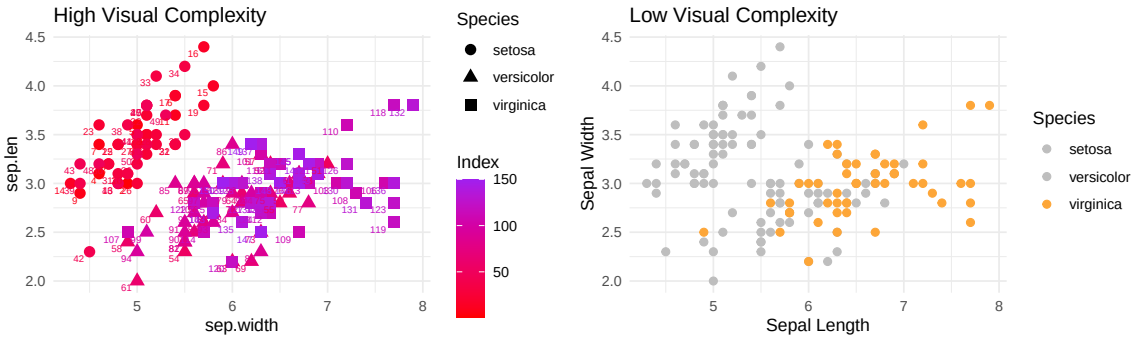

Figure 8: High versus low cognitive load demand through the reduction of visual complexity

2.6.2 Information Overload and Simplification Techniques

Addressing information overload in visualisations necessitates the strategic application of simplification techniques. Filtering enables focused data selection, while data reduction aggregates information to highlight overarching trends. Storyboarding structures data presentation, aiding in contextual comprehension, and prioritisation ensures critical information is prominently displayed, elevating the visualisation's clarity and impact. These strategies collectively combat overwhelming data or excessive visual elements, enhancing comprehension and the effective communication of insights to viewers.

3 Univariate Data Visualisation Methods

This chapter, initiates the discussion regarding data visualisation methods. This chapter particularly introduces techniques for graphically representing for datasets of the simplest nature - that is, single-variable datasets. It focuses on histograms, bar charts, Kernel Density Estimate, ROC Curves, and Time Series Analysis.

3.1 Histograms

A classical histogram is a visual representation of the distribution of a (numerical) dataset. It is composed by a series of contiguous rectangles, also known as "bins", each representing a range of data values. The height of each of these corresponds to the frequency or count of data points within that range, depending on the type of histogram. On a cartesian coordinate system, typically, the x-axis shows the endpoints of each bin, and the y-axis represents count or frequency.

3.1.1 Classic Frequency Histograms

Mathematical Defintion
Suppose the data points within a datset are partitioned into k non-overlapping bins, denoted as $B_1, \ldots, B_k$ such that $B_i = [t_i, t_{i+1})$ for $i = 1, \ldots, k$. Here, t_i and t_{i+1} represent the lower and upper boundaries of the i-th bin, respectively.

Then, the classical frequency histogram of the data can be represented as a set of k bars, where the height of the i-th bar corresponds to the frequency or count of data points falling within the interval B_i.

Theory of Bin Number and Bin Width
The classical frequency histogram can be fully described by two main factors: the bin width h, and the bin origin t_0. However, in order for the bin counts to be comparable, the bins should all have the same width. Furthermoree, it is essential to correctly choose the number of bins, since these have a huge impact on how the data is displayed and interpreted. Too few bins may hide the information in a dataset, and too many bins can cause a lot of noise in a dataset.

Sturges' Rule, established by Herbert Sturges in 1926, offers a systematic approach to determining the optimal number of bins constructing a frequency histogram [23]. This rule is based on the concept of normality, with the binomial distribution, $B(n, p = 0.5)$, serving as a model for an optimally constructed histogram. It links the binomial distribution with normally distributed data, providing a basis for selecting the number of bins to achieve a histogram resembling a normal density curve.

It suggests setting the number of bins, denoted as k, to

$$k = 1 + \log_2 n,$$

where n represents the sample size. This formula stems from constructing a frequency histogram with k bins, each of width 1 and centered on the points $i = 0, 1, \ldots, k-1$. The bin count of the i-th bin is chosen to be the binomial coefficient $\binom{k-1}{i}$. As k increases, this ideal frequency histogram assumes the shape of a normal density with mean $(k-1)/2$ and variance $(k-1)/4$. The total sample

size is

$$n = \sum_{i=0}^{k-1} \binom{k-1}{i} = (1+1)^{k-1} = 2^{k-1}$$

by the binomial expansion. Hence, Sturges' rule follows immediately: $k = 1 + \log_2 n$ [23].

Note that Sturges' rule functions as a number-of-bins rule rather than a bin-width rule. Assuming all bins are of equal width, to find the bin-width, Sturges' rule is implemented by partitioning the sample range of the data into the recommended number bins [23].

3.1.2 Density Histograms

A frequency histogram differs from a density histogram by its normalisation, which integrates to 1. That is, let $B_i = [t_i, t_{i+1})$ represents the i-th bin, as defined previously. If $t_{i+1}t_i = h$ for all i, the histogram has a fixed bin width of h. In a frequency histogram, blocks of height 1 and width h are stacked in the appropriate bins, resulting in an integral equal to nh. Conversely, a density histogram uses blocks of height $1/(nh)$ to ensure each block has an area of $1/n$. In this way, the height of each bin represents the probability distribution of the data, such that the total area of the histogram equals 1.

Let v_i denote the bin count of the i-th bin, representing the number of sample points falling in bin B_i. Thus, the density histogram is mathematically defined as:

$$\hat{f}(x) = \frac{v_k}{nh} = \frac{1}{nh} \sum_{i=1}^{n} I_{[t_i, t_{i+1})}(x_i) \quad \text{for } x \in B_k.$$

With this definition of a histogram, it is straightforward to check that $\hat{f}(x) \geq 0$ and that $\int \hat{f}(x)\, dx = 1$, so that $\hat{f}(x)$ is a proper density function.

The density histogram's bin counts v_k can be considered as binomial random variables, where each bin count v_k follows a binomial distribution $B(n, p_k)$. The probability p_k for each bin is the integral of the density function $f(t)$ over the bin interval B_k:

$$p_k = \int_{B_k} f(t)\, dt.$$

3.1.3 Histogram in Practice

The R language uses the `hist` function to create histograms. This function takes vectors as input, along with other parameters to plot a histogram.

By plotting histograms of a large number of sample means, one can observe the shape gradually resembling a normal distribution.

Consider Figure 9 which was created on *ggplot2* with the use of to the `geom_histogram` function and iris dataset.

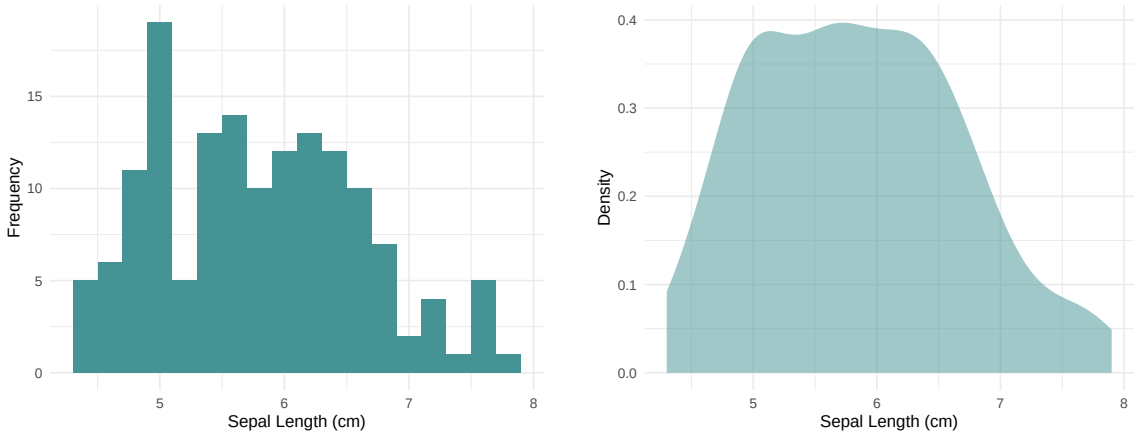

Figure 9: Histogram and Kernel Density Estimation of Sepal Length in Iris Dataset

3.2 Kernel Density Estimation

Kernel Density Estimation(KDE) evolves from the concept of histogram, offering a method for estimating the the probability density function of a dataset.

Kernel Density Estimation (KDE) is an very useful tool in statistics. Instead of discrete histograms, it helps create a smooth curve from values in a dataset. KDE is used to infer the distribution of a population based on a limited sample. Thus, the result of the kernel density estimation is an estimate of the sample's probability density function. Based on this estimated probability density function, one can ascertain certain characteristics of the data distribution, such as the regions where data is concentrated.

The KDE algorithm takes a parameter called bandwidth, that affects how smooth the resulting curve is. Changing the bandwidth changes the shape of the kernel: a lower bandwidth implies only points very close to the chosen position are given any weight, which leads to the estimate looking squiggly. In contrast, a higher bandwidth means having a shallow kernel, where distant points can contribute. Thus, leading to a smoother curve.

3.2.1 Theory of Kernel Density Estimation

We can express KDE as follows, where the K represent the kernel function:

$$\hat{f}(x) = \sum_{\text{observations}} K\left(\frac{x - \text{observation}}{\text{bandwidth}}\right).$$

In KDE, the variable x represents the point of density estimation. The total number of data points, denoted as n, constitutes the sample size, and all these points are used for estimating the dataset's probability density function. The bandwidth (h) is a crucial parameter that controls the smoothness of the estimation, with smaller bandwidths leading to closer fits to data and larger ones smoothing out details. The kernel function (K), such as the Gaussian or triangular kernel, assigns mass around each data point, influencing the density estimate, while X_i represents the individual sample points

that collectively contribute to the estimated density function, guided by the kernel function and the bandwidth[30]. Here is the formula for the kernel density estimator:

$$\hat{f}(x;h) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right),$$

Here, the kernel function $K(u)$ is a normalized non-negative function that satisfies:

$$\int K(u)\,du = 1.$$

By introducing the rescaling notation $K_h(u) = h^{-1}K(\frac{u}{h})$, we can write the formula in a more compact way:

$$\hat{f}(x;h) = n^{-1}\sum_{i=1}^{n} K_h(x - X_i).$$

In Figure 9, the kernel density estimation of the dataset is presented. The distribution of sepal lengths is depicted by the kernel density curve. The peaks observed in the curve correspond to the primary concentration trends of sepal length within the data. A unimodal curve signifies a concentration of sepal lengths for most irises in that specific region. Conversely, a bimodal or multimodal curve suggests the existence of multiple concentration areas.

3.3 Bar Charts

Bar charts are a crucial tool in data presentation, arranging data into vertical or horizontal bars. The varying lengths of these bars directly correspond to the magnitude of the information they represent. Bar charts excel in comparing classified data, especially when values are closely aligned. This stems from the nature of human perception, as our visual acuity for height surpasses that of other visual elements like area or angle.

Bar charts represent a versatile tool for data visualisation, frequently employed to compare distinct categories. The vertical bar chart, commonly recognised, exhibits categories along the X-axis and their frequencies or counts along the Y-axis. Horizontal bar charts, rotated 90 degrees, prove beneficial for extended category names or numerous categories, displaying categories on the Y-axis and frequencies on the X-axis. Multi-set or grouped bar charts facilitate side-by-side comparisons of sub- groups within categories, available in both vertical and horizontal orientations. Stacked bar charts illustrate classes of values subdivided into sub-classes, often differentiated by colour, where each segment's size signifies its frequency or count, and the total bar length reflects the cumulative total.

3.3.1 Bar Charts in Practice

The two bar charts on in Figure 10 illustrate the impact of varying vitamin dosages on tooth growth, further categorised by supplement type. On the X-axis, three distinct levels of vitamin dosage are presented, while the Y-axis indicates the average lenth of tooth for each dosage.

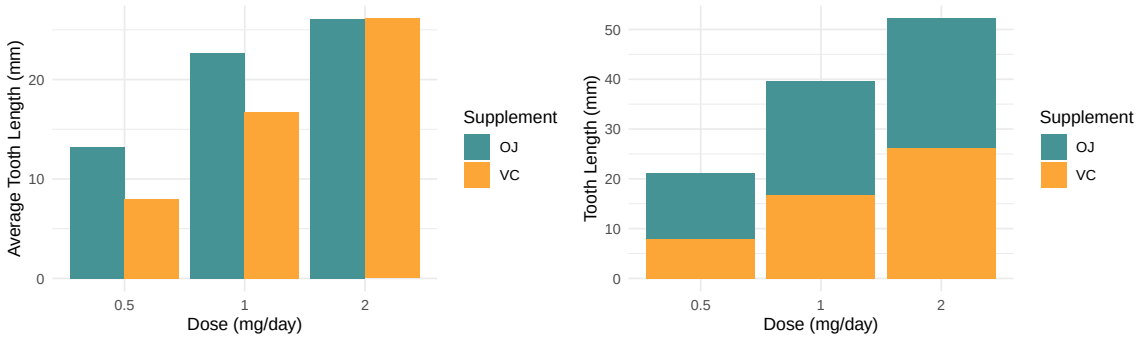

Figure 10: Visualising tooth growth by dosage: A comparison of grouped and stacked bar chart techniques

From the grouped bar, chart found on the left of Figure 10, due to the side-by-side positioning of the bars, it is easy to note that tooth growth varies not only with the dosage but also with the supplement type. In contrast, the stacked bar graph on the right of Figure 10 facilitates the understanding of the combined effects of the two supplements at each dosage level. However, compared to the latter, it becomes more challenging to differentiate the individual contribution of each supplement.

3.4 Line Charts and Time Series

Previous subsections explored histograms, kernel density estimation, and bar charts, which are univariate methods used for analysing independent variables in isolation, revealing patterns and intrinsic properties within data. Now, the focus shifts to univariate methods of dependent variables, specifically line charts, which are crucial for visualising relationships between variables to understanding their interactions and dependencies over time.

3.4.1 Theory of Line Charts

Line charts are fundamental tools in data visualisation, particularly useful for displaying time series data. A line chart represents n data points $\{(x_i, y_i)\}_{1 \leq i \leq n}$ on a Cartesian coordinate system, with the x-axis often denoting time intervals or ordered categories and the y-axis representing the measured values.

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. Also, a series of linear interpolations between pairs of data points could be used. These interpolations assume that the change between two points is uniform or linear. This linear approach is mathematically represented as:

$$y = y_i + \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \times (x - x_i), \quad x_i \leq x \leq x_{i+1}.$$

This equation highlights that for any point x between x_i and x_{i+1}, the corresponding value of y on the line chart is determined by a linear relation. This method effectively "fills the gaps" between actual observed data points and provides a continuous view of the data.

3.4.2 Time Series Analysis

Time series visualisation is particularly suited to line charts because they effectively display changes and trends over time, allowing for easy visualization of relationships between time points. A time series[3] is a collection of observations x_t, where $t = 0, \ldots, n$ denotes the time point at which the observation is recorded. An index set T_0 which collects all the time points when observations are available. For instance, $T_0 = \{0, \ldots, n\}$ for $n \in \mathbb{N}$. By plotting these data points over time, line charts help in identifying long-term trends, seasonal patterns, and anomalies.

A time series can be viewed as a realisation of a stochastic process. A stochastic process[3] $X = (X_t)_{t \in T_0}$ is a collection of random variables X_t, where t denotes the time index and T_0 the index set. For a fixed event $\omega \in \Omega$ we obtain the realisation of the stochastic process (sometimes also called a sample path) which is given by $x_t = X_t(\omega)$, $t \in T_0$. In practice, line charts of time series data provide insights into the behavior of such stochastic processes over time.

3.4.3 Case Example: Time Series Visualisation of Exchange rates

Here, plot daily and 49-day moving average exchange rates of exchange rate data in one figure.

```
# PLOT OF DAILY AND 49-DAY MOVING AVERAGE EXCHANGE RATES OF CAN, EUR, USD TO GBP
p1 <- ggplot(plot_dt, aes(x = Date, y = Rate, color = Currency)) + geom_line() +
  labs(y = "Exchange Rate to GBP", x = "", color = "Currency") + theme_minimal() +
  theme(legend.position = "none") + scale_color_manual(values = color_palette)
p2 <- ggplot(plot_data, aes(x = Date, y = Rate, color = Currency)) + geom_line() +
  labs(y = "Exchange Rate to GBP", x = "", color = "Currency") + theme_minimal() +
  theme(legend.position = "none") + scale_color_manual(values = color_palette)
# Extract the legend and combine the plots
p2_legend <- cowplot::get_legend(p2 + theme(legend.position = "right"))
combined_plot <- cowplot::plot_grid(p1, p2, rel_widths = c(1, 1), nrow = 2)
cowplot::plot_grid(combined_plot, p2_legend, nrow = 1, rel_widths = c(2, 0.5))
```

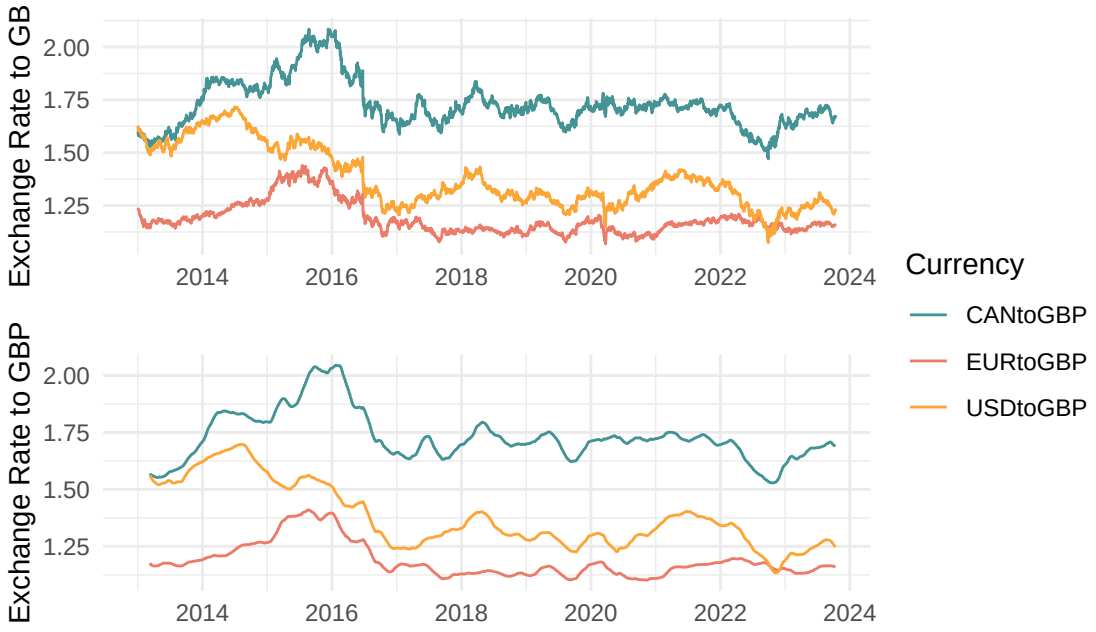

Figure 11: Daily (top) and 49-day moving average (bottom) exchange rates of CAN, EUR, USD to GBP

The first plot in Figure 11 presents a comparative visualisation of daily exchange rates for CAD, EUR, and USD against GBP, offering an overview of their trends and relative performance. This plot enables the identification of overall trends and periods of volatility for each currency pair, allowing for an assessment of their stability and strength relative to GBP. Notice that, there was a simultaneous and significant drop in all three currencies relative to the GBP. The concurrent nature of these declines across diverse currency pairs suggests that the driving factor is a depreciation of the GBP, rather than independent appreciations of the USD, EUR, and CAN. This notable depreciation of the GBP occurred around 2016, which coincides with the commencement of BREXIT process. Usually, the long term trend attracts financial analyst most. Therefore, filtering fluctuations and anomalies is important. Then, a 49-day moving average (MA49) was employed, shown in the second plot of Figure 11, to elucidate long-term trends while mitigating short-term fluctuations. This is mathematically represented as

$$\mathrm{MA}_{49}(t) = \frac{1}{49} \sum_{k=t-48}^{t} x_k,$$

where x_k denotes the exchange rate on day k.

This method effectively filters out daily noise, allowing a clearer view of overarching trends in currency movements against the GBP. The overlay of these moving averages on the daily exchange rates in visualisations provides both a clear comparative and a quantitative perspective.

Decomposition of Time Series

One of the primary advantages of time series visualisation 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, X_t, can often be described as a combination of several distinct components: Trend component t_t: The underlying progression in the series, Seasonal component s_t: Periodic fluctuations due to seasonal factor, Residual r_t: The irregular or error component. Hence, the decomposition of a time series can be described in two main models:

Additive Model [3]: In the additive model, the components are added together:

$$X_t = t_t + s_t + r_t.$$

Multiplicative Model [3]: In the multiplicative model, the components are multiplied together:

$$X_t = t_t \times s_t \times r_t \quad \text{or} \quad \log(X_t) = \log(t_t) + \log(s_t) + \log(r_t).$$

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.

R function `decompose()` is able to decompose the time series by additive model or multiplicative model. And 12 is the demonstration of decomposition.

```
# PLOT OF DECOMPOSITION OF ADDICTIVE TIME SERIES MODEL
ggplot(decomposed_df, aes(x = time, y = value)) + geom_line() +
  facet_wrap(~component, scales = "free_y", ncol = 1) + labs(x = "Date",y = "Exchange Rate to GBP") + theme_minimal()
```

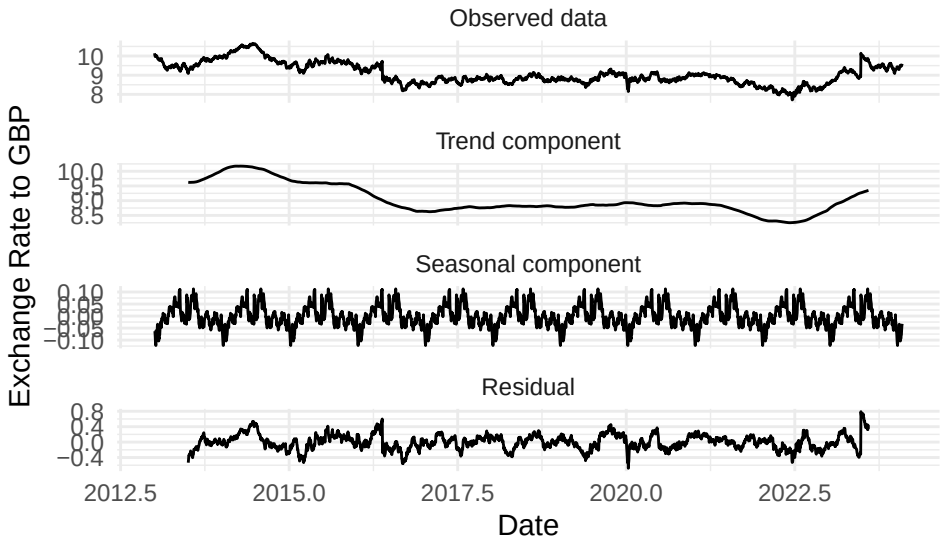

Figure 12: Decomposition of addictive time series model of CNY to GBP exchange rates

From Figure 12, the CNY to GBP exchange rate time series was decomposed into its fundamental components: trend, seasonality, and residual noise by additive model. This additive model, represented mathematically as $X_t = t_t + s_t + r_t$.

As illustrated in Figure 12, the trend component t_t of CNY to GBP exchange rate exhibits a distinct pattern over time: initially, it shows a gradual decrease and reached crest in 2022, followed by a sudden increase afterwards. It coincides with the tax reduction policy issued by UK government in 2022, which leads to a depreciation of GBP. This trend is pivotal for understanding the broader economic relationship between these currencies.

Moreover, the seasonal component s_t of the decomposition highlights cyclical fluctuations, indicative of recurrent patterns within the year. These could be attributed to seasonal economic activities, policy changes, or other cyclical factors influencing the currency market. The clear demarcation of these cyclical trends in the seasonal component helps in isolating such effects from the overarching trend.

Lastly, the residual component r_t encompasses the random, unexplained variations after accounting for the trend and seasonal factors. analysing these residuals is crucial for understanding the unpredictability in the exchange rate and can be pivotal in risk management and forecasting.

Autocorrelation Analysis of CNY to GBP Exchange Rate
Autocorrelation, also referred to as serial correlation, is a crucial concept in time series analysis. It describes the correlation of a time series with its own past and future values. The autocorrelation function (ACF) measures the linear predictability of the series at lag h, which is the time t with its values at a previous time $t - h$. The mathematical fomulation of ACF of time series will be given in the following paragraph.

Suppose we have a time series with observations denoted by $x_1, \ldots, x_n$. Then the sample mean is given by $\bar{x} = \frac{1}{n}\sum_{t=1}^{n} x_t$. And the sample autocovariance function[3] at lag h in days of our time series is

$$\hat{\gamma}(h) := \frac{1}{n}\sum_{t=1}^{n-|h|}(x_{t+|h|} - \bar{x})(x_t - \bar{x}), \quad -n < h < n.$$

Hence, the sample autocorrelation function[3] at lag h in days is given by

$$\hat{\rho}(h) := \frac{\hat{\gamma}(h)}{\hat{\gamma}(0)} = \frac{\sum_{t=1}^{n-|h|}(x_{t+|h|} - \bar{x})(x_t - \bar{x})}{\sum_{t=1}^{n}(x_t - \bar{x})^2}, \quad -n < h < n.$$

The value of $\hat{\rho}(h)$ lies between -1 and +1. A value close to +1 indicates a strong positive correlation, while a value close to -1 indicates a strong negative correlation. A value near 0 suggests little to no linear correlation. A slow decay in the ACF plot indicates a strong relationship between past and present values, while spikes at specific lags may suggest seasonality. Autocorrelations outside the 95% confidence interval are considered statistically significant.

Next, visualise the ACF for the CNY to GBP exchange rate to understand its time-dependent structure better.

```
# PLOT OF THE AUTOCORRELATION FUNCTION
acf_data <- acf(MyData$CNYtoGBP, plot = FALSE)
plot(acf_data, main = "", xlab = "Lag h", ylab = "ACF")
```

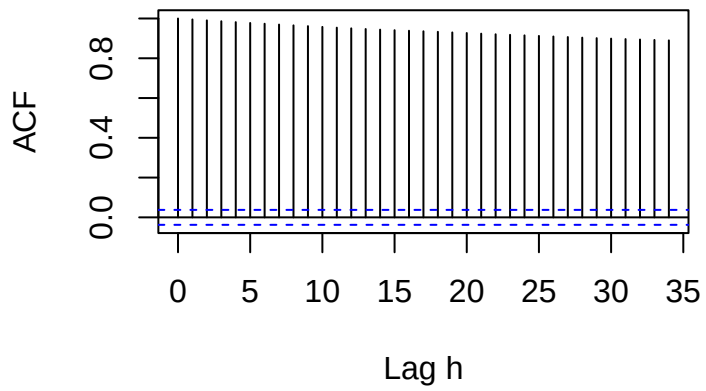

Figure 13: Autocorrelation Function at lag h in days of CNY to GBP Exchange Rate

From Figure 13, the ACF plot for the CNY to GBP exchange rate series reveals that the ACF starts near 1 and decreases gradually. This pattern suggests a strong persistence in the time series, indicating that past values have a significant influence on future values. In time series analysis, such a slow decay in the ACF is indicative of a non-stationary series, where the mean, variance, and autocorrelation structure do not remain constant over time.

This persistent autocorrelation suggests that short-term movements in the CNY to GBP exchange rate are heavily influenced by its recent history. Such a characteristic is crucial for forecasting models, as it implies that recent historical data can be a powerful predictor of near-future trends. Models like ARIMA (Autoregressive Integrated Moving Average), which are well-suited for data with high autocorrelation, may be particularly effective in this context.

3.5 ROC Curve

The Receiver Operating Characteristic (ROC) anaysis is a technique for assessing the performance of classification models as its discrimination threshold is varied. Unlike univariate methods in previous subsections that are constructed from a single dataset representing one variable, ROC analysis divides a single dataset into two samples based on the actual condition of each observation, but focuses on evaluating the performance of a classification model on one variable, typically the predicted probability that a given observation belongs to a positive class. These two populations are defined as follows: the positive class (true condition positive), which includes observations that actually belong to the class of interest. The negative class (true condition negative), which includes observations that do not belong to the class of interest.

3.5.1 Theory of ROC curve

Let Y be a binary variable indicating the true class of an instance, with $Y = 1$ for positive instances and $Y = 0$ for negative instances. Let X be a continuous variable representing the predicted score or probability of an instance being classified as positive by the classifier, with $X \geq c$ for positive and $X < c$ for negative given a threshold c.

Since ROC curve plots two parameters[10]: True Positive Rate (TPR, also known as Sensitivity, is the proportion of positive instances correctly identified) and False Positive Rate (FPR, 1 - Specificity, is the proportion of negative instances incorrectly identified as positive). Then define the TPR and the FPR using conditional probabilities as follows[18]:

$$\mathrm{TPR}(c) = \mathbb{P}[X \geq c | Y = 1], \quad \mathrm{FPR}(c) = \mathbb{P}[X \geq c | Y = 0].$$

The ROC curve is then the set of points $(\mathrm{FPR}(c), \mathrm{TPR}(c))$ for all possible values of threshold c. This curve plots the trade-off between sensitivity (or TPR) and specificity (1 - FPR) across different thresholds. Let $t = \mathrm{FPR}(c)$, then $c = \mathrm{FPR}^{-1}(t)$ and $\mathrm{TPR}(c) = \mathrm{TPR}(\mathrm{FPR}^{-1}(t))$. Hence, let $\mathrm{ROC}(t) = \mathrm{TPR}(\mathrm{FPR}^{-1}(t))$, the ROC curve can be represented as a parametric function of t[18] , with t varying from 0 to 1:

$$\mathrm{ROC} = \{(t, \mathrm{ROC}(t)) \,|\, t \in [0, 1]\}.$$

Also, the Area Under the Curve (AUC)[18] provides a single measure of overall performance of a classifier. This definition represents the integral of $\mathrm{ROC}(t)$ over the interval from 0 to 1. It can be defined as:

$$\mathrm{AUC} = \int_0^1 \mathrm{ROC}(t) dt.$$

The larger the AUC, the better the classifier. An AUC of 0.5 suggests no discriminative ability, while an AUC of 1.0 represents perfect classification.

3.5.2 ROC analysis in COVID-19 test

The COVID-19 pandemic underscores the need for accurate diagnostic tests to differentiate between positive and negative cases[11]. An effective mean for evaluating accuracy of various diagnostic tests is the ROC analysis. ROC analysis comprehensively examines test accuracy by integrating TPR, FPR, and AUC. These metrics are calculated by comparing test outcomes against a recognized gold standard, thereby determining the true disease status. This approach is vital for understanding the efficacy of COVID-19 diagnostic tools and guiding public health decisions.

The TPR, or sensitivity, measures the proportion of actual positives correctly identified, reflecting the test's ability to detect COVID-19 cases. The FPR, conversely, indicates the rate of false alarms, where non-infected individuals are wrongly identified as infected. An ideal test minimises FPR, avoiding unnecessary treatments or quarantine. The ROC curve visualizes the trade-off between TPR and FPR at various thresholds. A curve arching towards the upper left indicates high sensitivity and low FPR — the hallmarks of a reliable test. The AUC consolidates the ROC performance into a single value: 0.5 denotes no better accuracy than random chance, while 1.0 represents perfect accuracy. A higher AUC indicates a better overall ability of the test.

Here, the simulated data will be used to demonstrate the ROC analysis in COVID-19 test. Score_Test1 and Score_Test2 are the simulated test scores from two different diagnostic tests for COVID-19. These scores are continuous variables that typically would represent the likelihood of a positive diagnosis. The scores might come from a lab test result, such as a PCR test or a rapid antigen test, where higher scores indicate a higher likelihood of infection. The Condition column indicates the true condition of the patient, with 1 representing a positive COVID-19 case and 0 representing a negative case.

```
# PLOT OF ROC CURVES
plot(roc_test1, col="#459395", lwd=2)
lines(roc_test2, col="#FDA638", lwd=2, lty=2)
legend("bottomright", legend=c("Diagnostic Test 1", "Diagnostic Test 2"),
       col=c("#459395", "#FDA638"), lwd=2, lty=c(1, 2), cex=0.5)
# Add the perfect lines
abline(h=1, lty=2, col="red")
abline(v=1, lty=2, col="red")
```

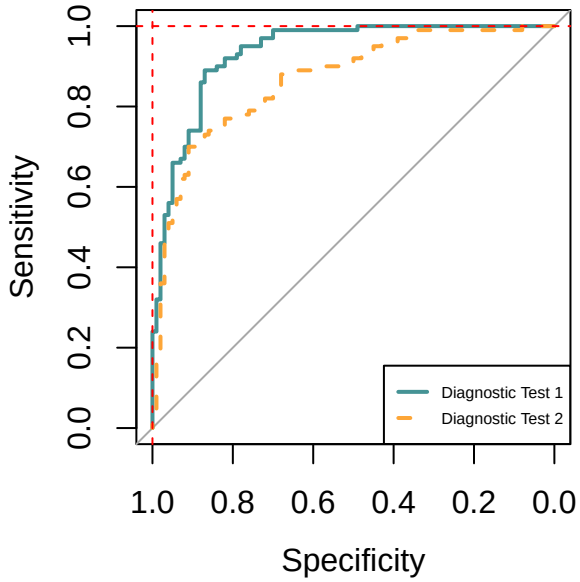

Figure 14: ROC Curves for Two Diagnostic Tests

From the plot 14, the Diagnostic Test 1 has higher AUC. This test is likely to be more accurate in diagnosing COVID-19. It suggests that the test has a higher combined sensitivity and specificity, meaning it can identify positive cases more correctly and has fewer false alarms. Also, the Diagnostic Test 2 has a lower AUC than Test 1, This test is less accurate but still better than random guessing. It may miss more true cases (lower sensitivity) or incorrectly identify healthy individuals as having COVID-19 (higher false positive rate).

In practice, the choice of which test to use may depend on other factors. For instance, if the cost of missing a COVID-19 case is high (for example, in a nursing home setting), a test with higher

sensitivity might be preferred. Conversely, if the cost of false positives is high (for example, leading to unnecessary quarantine), a test with higher specificity might be favored.

4 Bivariate Data Visualisation Methods

This chapter transitions from the study of univariate data visualisations to the exploration of bivariate data. Fundamental bivariate data visualisation methods, including heatmaps, scatter plots, and bubble charts, are introduced, scrutinised, and modelled throughout this chapter. In order to facilitate a more profound analysis of bivariate data, the chapter further delves into the examination of linear regression, as well as LOESS regression. Notably, within the context of scatter plots, an intriguing deviation is observed with the incorporation of animated data visualisations.

4.1 Heatmaps

The heatmap is a data visualisation technique that uses colour coding to represent different intensities. It can be represented as an $m \times n$ matrix $\mathbf{M}$, with m observations for variable 1 and n observations for variable 2:

$$\mathbf{M} = \begin{bmatrix} \mathrm{M}_{11} & \mathrm{M}_{12} & \dots & \mathrm{M}_{1n} \\ \mathrm{M}_{21} & \mathrm{M}_{22} & \dots & \mathrm{M}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{M}_{m1} & \mathrm{M}_{m2} & \dots & \mathrm{M}_{mn} \end{bmatrix},$$

where each entry M_{mn} represent an observation.

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. The data-driven insights could empower policymakers to make informed decisions regarding preventive measures and firefighting strategies.

In Figure 15, 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. 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 [8].

The seasonal pattern of fire frequency is shown in Figure 16. Observe that more fire occur in the months of August to October compared to the rest of the year.

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

print(heatmap_plot)
```

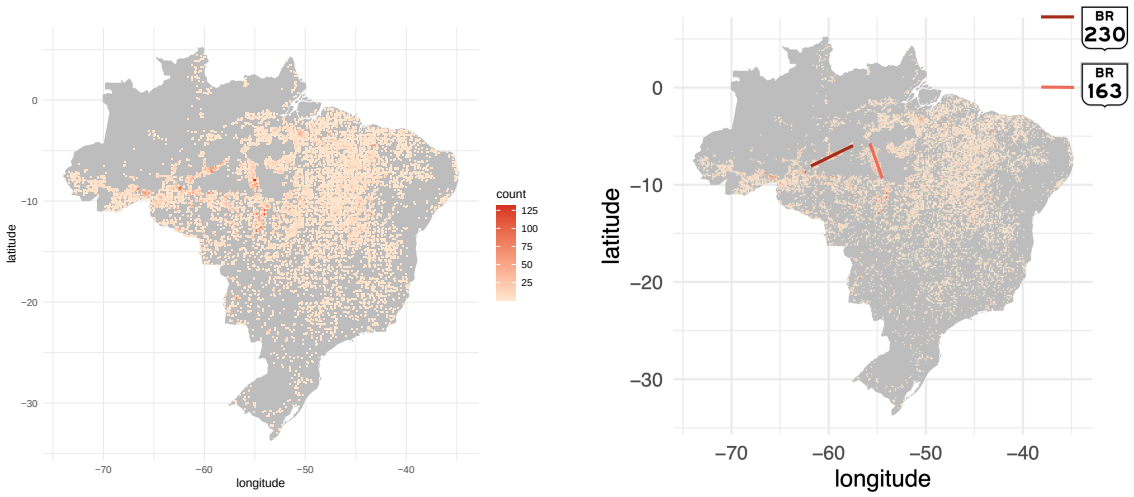

Figure 15: Frequency of fire in Brazil (2022), two strips of high frequencies of fires are highly unusual

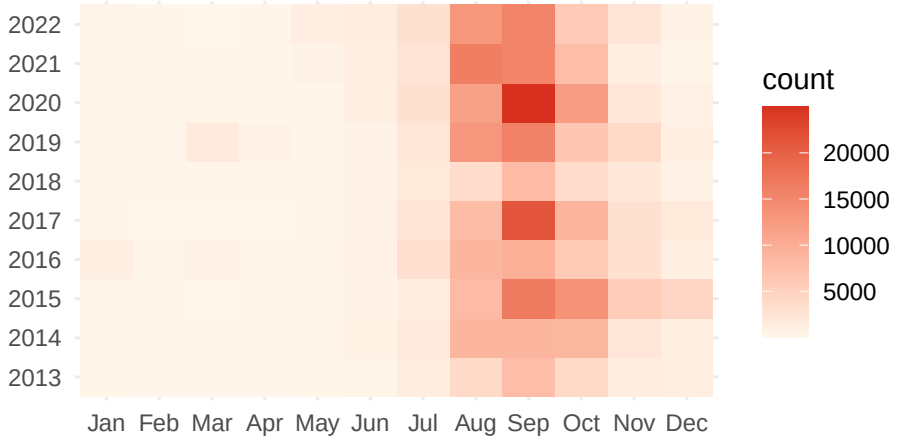

Figure 16: Frequency of fire in Brazil (2013-2022)

4.2 Scatter Plots

A scatter plot is a graphical representation of a set of data points in a two-dimensional coordinate system. Each data point is represented by a dot, and the position of the dot is determined by the values of two variables. Some examples of this type of visualisation can be found in Figures 6, 8, and 17.

In general, the Y-axis denotes the response, or dependent, variable and the X-axis denotes the

explanatory, or independent, variable. Each observation of a dataset is mapped to a dot in the 2-dimentional space. Let (x_i, y_i) represent the coordinates of the i-th data point on the scatter plot produced by mapping a set of data. The scatter plot can be mathematically described as a set of points: $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ where n is the number of observations in the set.

The scatter plots in Figure 17 visualise data from the mtcars dataset. Each of these three graphs represents different information drawn from the dataset. The first plot displays the various car models and their corresponding miles per gallon. The middle plot, once again, shows the car models, but this time their horsepower is represented. Finally, the last scatter plot visualises the relationship between the cars' miles per gallon and their horsepower.

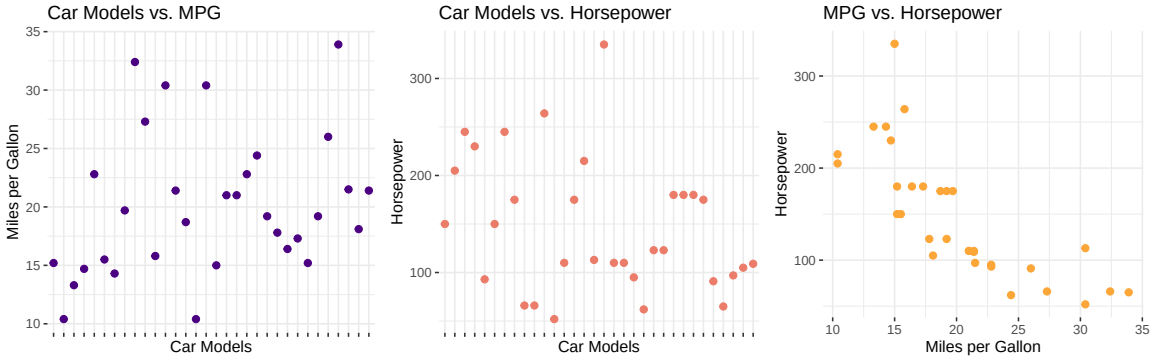

Figure 17: Partial scatter plot deconstruction of the mtcars dataset

4.2.1 Animated Scatter Plots

While static scatter plots are effective in depicting relationships between two variables, animated scatter plot visualisations take this a step further by introducing, for example, a temporal dimension to the data visualisation. Unlike static plots, animated scatter plots enable the depiction of changes in relationships, clusters, or outliers over time. In R, the *gganimate* package, building on the foundation of *ggplot2* package, facilitates the creation of animated plots, including animated scatter plots.

4.2.2 gganimate in Practice

The figure below depicts a static scatter plot illustrating the relationship between GDP per capita and life expectancy across various countries. GDP per capita is displayed on a logarithmic scale along the x-axis, and life expectancy is shown on the y-axis. The size of each point on the plot corresponds to the population size of the country it represents, and the colour of each point distinguishes one country from another.

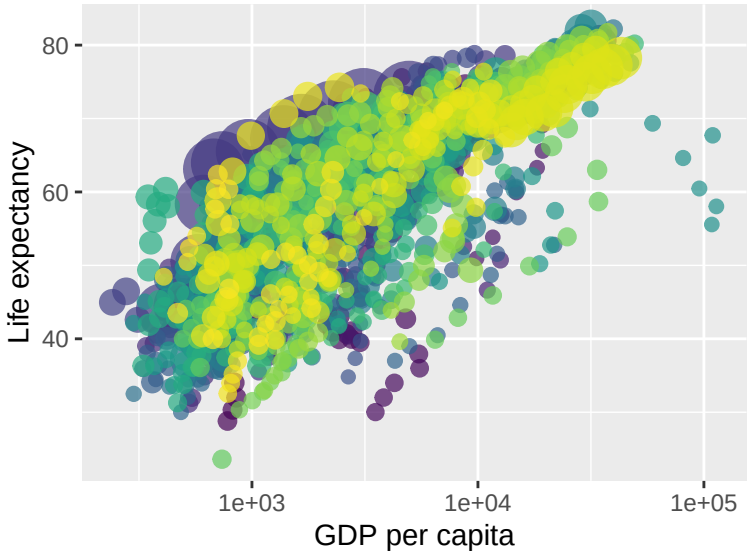

Figure 18: staticl scatter plot deconstruction of the gapminder dataset

In the graph we can see that there appears to be a positive correlation between GDP per capita and life expectancy, as GDP per capita increases, life expectancy also tends to be higher.

However, by using the animation function, we can see how life expectancy and GDP per capita change through the years for each country, so it helps to show how relationships between variables evolve over time or across different categories.

```
# create animated ggplot
gapplot<-ggplot(gapminder, aes(gdpPercap, lifeExp, size = pop, colour = country)) +
  geom_point(alpha = 0.7, show.legend = FALSE) +
  scale_colour_manual(values = country_colors) +
  scale_size(range = c(2, 12)) +
  scale_x_log10() +
  facet_wrap(~continent) +
  # Here comes the gganimate specific bits
  labs(title = 'Year: {frame_time}', x = 'GDP per capita', y = 'life expectancy') +
  transition_time(year) +
  ease_aes('linear')
```

4.3 Bubble Charts

4.3.1 Theory of Bubble Charts

Bubble charts are a captivating data visualisation 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.

The construction of bubble charts, is parallel to that of a scatter plot, but involves the scaling the data values to determine the size of each bubble. While it isn't always the case, the size of these is typically proportional to the variable it represents. The choice of scaling method depends on the

data distribution and the message the chart aims to convey.

Generally, the formula for calculating the bubble radius (R) involves applying the scaling function

$$R = kV,$$

where R represents the size of the bubble, V the value of the variable being represented, and k 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.

Hence, bubble chart represents data points in three dimensions: x-coordinate, y-coordinate, and bubble size. Each data point is denoted by a triplet of values (x_i, y_i, C_i), where x_i and y_i represent the coordinates, and C_i is the equation of the circle for the i-th observation. In the context of a bubble chart, the radius R_i of each bubble is expressed as $k \cdot V_i$, allowing us to formulate the equation for each circle as:

$$(x - x_i)^2 + (y - y_i)^2 = (kV_i)^2,$$

Here, x_i and y_i denote the coordinates of the circle's center.

Since each bubble B_i can be defined mathematically by the triplet (x_i, y_i, C_i), akin to the scatter plot, the bubble plot can be described mathematically as a set of "bubbles" $\{B_1, \ldots, B_n\}$, where n represents the number of observations in the set.

4.3.2 Bubble Charts in Practice

The bubble chart shown in Figure 19, as the scatter plots above, visualises data from the mtcars dataset. The single graph deplics 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 even colour-coding the bubbles based on the number of cylinders (cyl).

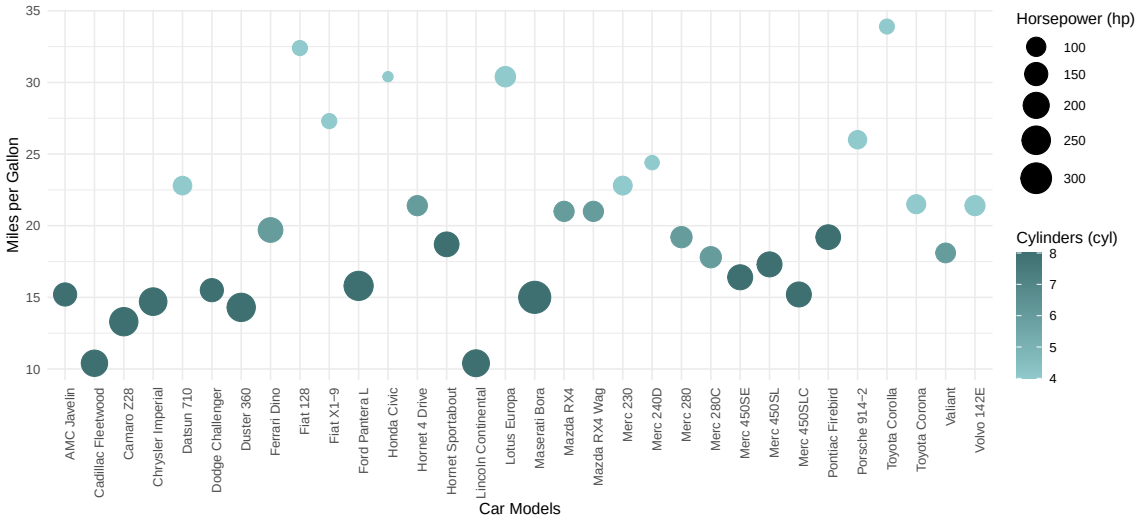

Figure 19: Bubble plot illustrating trends in car types and performance

In Figure 19 a strong correlation between the cars' number of cylinder and their horsepower can be quickly identified. This is due to the fact that as size of the dots (representing horsepower) increases, their colour (representing the number of cylinders) simultaneously becomes darker. Furthermore, a correlation between the miles per gallon consumed by the different types of cars and their number of cylinders and horsepower is easily identifiable in Figure 19. This is shown by the fact that, in the same way that the darker dots cluster towards the bottom of the graph and become lighter as they reach the top of the graph, those with a large diameter also cluster near the lower part and decrease in size as they ascend.

The efficiency of bubble charts is highlighted by the fact that capturing just some of the information provided by this single bubble chat - that is, the information regarding the car models, the miles covered per gallon, and the horsepower of each of these, requires three different scatter plots. These are displayed in Figure 17. The contrast between the simplicity and readability of Figure 19, and the density and complexity of Figure 17 emphasises the advantages of bubble charts in representing datasets with a higher number of variables.

4.4 Simple Linear Regression

Regression models are statistical tools that provide functions to estimate the relationship between the response variable and one or more explanatory variables. Regression analysis is widely adopted by data scientists, who use large datasets to build predictive models for trend forecasting. The following paragraphs will introduce simple linear regression models and demonstrate their usage using the mtcars dataset.

4.4.1 Theory of Simple Linear Regression

Let $\mathbf{x} = (x_1, \ldots, x_n)^T$ denote n explanatory variables and let $\mathbf{Y} = (Y_1, \ldots, Y_n)^T$ denote n corresponding response variables.

In a simple linear model, it is assumed that the response variables $Y_1, \ldots, Y_n$ are uncorrelated with a common variance σ^2, and their expectations are given by $E(Y_i|x_i) = \beta_0 + \beta_1 x_i$. The expectations generated by β_0 and β_1 given x_i can be expressed as:

$$\mathrm{E}(\mathbf{Y}|\mathbf{x}) = \begin{pmatrix} \beta_0 + \beta_1 x_1 \\ \vdots \\ \beta_0 + \beta_1 x_n \end{pmatrix} = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \beta_0 + \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \beta_1 = \mathbf{1}_n \beta_0 + \mathbf{x}\beta_1, \tag{4.1}$$

where $\mathbf{1}_n$ is an n-vector of 1's.

Given design matrix $\mathbf{X}$ where $\mathbf{x}_i = (1, x_i)$ and $\beta = (\beta_0, \beta_1)^T$, then $E(Y_i|x_i) = \mathbf{x}_i\beta$. These assumptions can be equivalently written in the vector form:

$$\mathrm{E}(\mathbf{Y}|\mathbf{x}) = \begin{pmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} = \mathbf{X}\boldsymbol{\beta}, \quad \mathrm{var}(\mathbf{Y}|\mathbf{x}) = \begin{pmatrix} \sigma^2 & 0 & \cdots & 0 \\ 0 & \sigma^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma^2 \end{pmatrix} = \sigma^2 \mathbf{I}_n. \tag{4.2}$$

4.4.2 Theory of Least Squares Estimation

The residual sum of squares (RSS) is a measure of the goodness of fit in a regression model, where residuals are the differences between the response variables y_i and responses generated by the regression model $\mathrm{E}(\mathbf{Y}_i|\mathbf{x})$. In least squares estimation, the goal is to find values of parameters $\boldsymbol{\beta} = (\beta_0, \beta_1)^T$ to minimise the RSS, denoted by Q:

$$\mathrm{Q} = \sum_{i=1}^{n} [y_i - \mathrm{E}(Y_i|\mathbf{x})]^2 = [\mathbf{y} - \mathrm{E}(\mathbf{Y}|\mathbf{x})]^T [\mathbf{y} - \mathrm{E}(\mathbf{Y}|\mathbf{x})] = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}), \tag{4.3}$$

where $\mathbf{y}$ is n-vector of response variables and $\mathbf{X}$ is the $n \times 2$ design matrix. The partial derivative of Q with respect to vector $\boldsymbol{\beta}$ is:

$$\frac{\partial Q}{\partial \boldsymbol{\beta}} = 2(\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} - \mathbf{X}^T\mathbf{y}), \tag{4.4}$$

Equating $\frac{\partial Q}{\partial \boldsymbol{\beta}} = \mathbf{0}$, the vector $\hat{\boldsymbol{\beta}}$, the least squares estimate of $\boldsymbol{\beta}$, can be written as:

$$\mathbf{X}^T(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}) = \mathbf{0}. \tag{4.5}$$

The least squares estimate of $\boldsymbol{\beta}$ is given by:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}. \tag{4.6}$$

4.4.3 Case example: 1970s automobiles

In this section, we will study the performance of 1970s automobiles using the mtcars dataset, employing the method of linear regression. Performance is measured in Miles per Gallon (mpg); the higher the mileage, the more efficient the automobile. We will start with the visualisation of a simple linear regression model, followed by the discussion of linear regression models and the model selection method.

In the preliminary stages of data exploration, calculating the correlation matrix is a crucial step before engaging in regression modeling, as shown in Figure 20. In real-world scenarios, variables are often correlated, and entirely independent relationships are seldom encountered. Therefore, analysing pairwise correlations becomes essential. This helps in understanding multicollinearity issues within the model. Multicollinearity occurs when one covariate within the model can be accurately predicted from another covariate. When this happens, the coefficient estimates of the model can change unpredictably due to minor changes in the data.

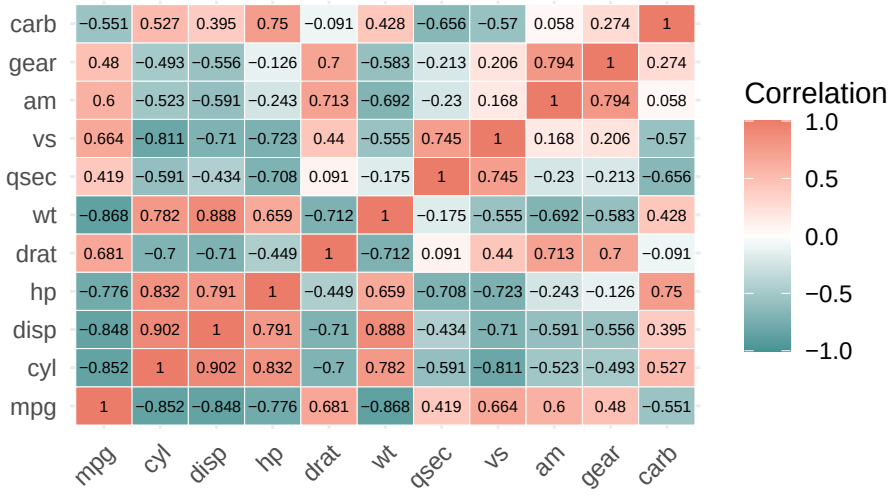

Figure 20: Correlation matrix of all variables in mtcars dataset

For the simple linear regression model, the response variable is Miles per Gallon (mpg), and we select weight (wt) as the explanatory variable. Note that mpg and wt are highly correlated, with a correlation coefficient of -0.868. This suggests that wt may have strong predictive power for mpg. Use the R function `lm()` to calculate the linear regression model, with the summary displayed below.

Observe that the t-test yields a p-value of 1.29×10^{-10}, which is less than 0.001. This indicates that the variable wt holds high statistical significance in this model. For the fitted model, the slope is $\beta_1 = -5.3445$, meaning that for every increase of 1000 lbs, the car efficiency decreases by 5 miles per gallon. The RSS is 3.046. The simple linear regression line is displayed in Figure 21.

```
# Summary for simple linear regression, with response variable Miles per Gallon
Modelwt <- lm(formula = mpg ~ wt, data = mtcars)
```

```
summary(Modelwt)

##
## Call:
## lm(formula = mpg ~ wt, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5432 -2.3647 -0.1252  1.4096  6.8727
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  37.2851     1.8776  19.858  < 2e-16 ***
## wt           -5.3445     0.5591  -9.559 1.29e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.046 on 30 degrees of freedom
## Multiple R-squared:  0.7528,Adjusted R-squared:  0.7446
## F-statistic: 91.38 on 1 and 30 DF,  p-value: 1.294e-10
```

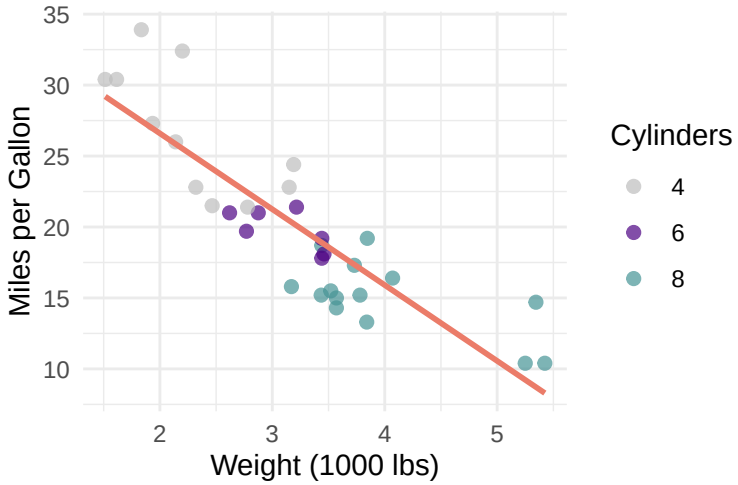

Figure 21: Scatter plot of car weights vs MPG

4.5 LOESS Regression

Locally weighted scatterplot smoothing (LOWESS) is a local regression method for a response variable Y on a single predictor X. LOWESS was proposed by William S. Cleveland in 1979, it is the special case to higher dimention locally estimated scatterplot smoothing (LOESS) regression method. They are non-parametric regression methods that fits a linear regression model for each neighbourhood of data. Unlike GLM, the LOESS model does not provide a global function to fit the data; rather, it fits neighborhoods of the data.

4.5.1 Theory of LOESS regression

Suppose we have n ordered observations $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ with predictors x_i and response variables y_i. Assume a model of the form

$$y_i = g(x_i) + \varepsilon_i,$$

where g is an unknown smooth function and ε_i is i.i.d.Gaussian error terms with mean 0 and variance σ^2.

Let $\Delta_i(x) = |x - x_i|$ represent the horizontal distance between x and x_i. Let $\Delta_{(i)}(x)$ denote the ordered distances to x, arranged from smallest to largest.

For each point (x, y) consider the neighbourhood to fit a line segment around it. The size of the neighbourhood indicates the number of data points used to fit the line segments and is determined by the smoothing parameter α. For $\alpha \leq 1$, each neighbourhood consists of $\frac{n}{\alpha}$ number of points [20]. Each point (x_i, y_i) in the neighbourhood of point (x, y) is assigned with a weight, "importance" of the data point to the line segment [25]. The weight of (x_i, y_i) is denoted as $w_i(x)$, which is defined as:

$$w_i(x) = T(\Delta_i(x); \Delta_{(q)}(x)). \tag{4.7}$$

Tricube weight function $T(u, t)$ is defined as:

$$T(u; t) = \begin{cases} (1 - (u/t)^3)^3 & \text{for } 0 \leq u < t \\ 0 & \text{for } u \geq t \end{cases}, \tag{4.8}$$

where u is the horizontal distance between neighbourhood point of interest (x_i, y_i) to point (x, y) and t is the maximum distance threshold. For points in the neighbourhood with horizontal distance exceeding threshold t, they are assigned with weight 0 by the tricube weight function.

For $\alpha > 1$, for each point (x, y) the neighbourhood include all points. The maximum distance threshold is assumed to be $\alpha^{\frac{1}{p}}$, for p explanatory variables. The weight is given by:

$$w_i(x) = T(\Delta_i(x); \Delta_{(n)}(x))\alpha. \tag{4.9}$$

The line segment is fitted by weighted least square, and this is the preliminary estimation $\hat{g}(x)$ for every point (x, y).

Both LOWESS and LOESS penalise the outliers in the preliminary estimation by assigning points further away from $\hat{g}(x)$ with less weight than before. The robustness weight (the adjusted weight) is given by:

$$r_i = B(\hat{\varepsilon}_i, 6\text{m}),$$

where bisquare weight function is defined as:

$$B(u; b) = \begin{cases} (1 - (u/b)^2)^2 & \text{for } 0 \leq |u| < b \\ 0 & \text{for } |u| \geq b \end{cases}, \tag{4.10}$$

and the median absolute residual m is defined as $\text{m} = \text{median}(|\hat{\varepsilon}_i|)$, the residual $\hat{\varepsilon}_i$ is defined as $\hat{\varepsilon}_i = y_i - \hat{y}(x_i)$.

Note the process of weight adjustment using previous estimation is iterated several times until a smooth curve is obtained.

The updated model estimate $\hat{g}(x)$, is computed using the local fitting method, but with the neighborhood weights $w_i(x)$ replaced by $r_i w_i(x)$ [9].

4.5.2 Case example: Taiwan Housing dataset

Property valuation can be modelled as a regression problem, in this analysis, we delve into the pricing of properties based on their age. Using the Taipei Housing dataset, we investigate the relationship between the age of houses (measured in years) and their prices (measured in New Taiwan dollars per unit area). Linear regression model and the LOESS regression model are used, as shown in Figure 22.

The linear model has a residual standard error of 13.32, while the LOESS model has a residual standard error of 12.16. Cross-validation is a method to compare model goodness of fit. LOESS, with its ability to capture local variations may outperform linear regression, which minimises the least squares estimate $\hat{\beta}$.

```
# linear regression vs. LOESS model
ggplot(data = estate, aes(x = house_age, y = price_per_area)) +
  geom_point(size =0.5) +
  theme_minimal()+
  scale_x_continuous(labels = scales::number_format(scale = 1)) +
  scale_y_continuous(labels = scales::number_format(scale = 1)) +
  geom_smooth(method = "loess", se = FALSE, color = "#EB7C69", span = 0.5) +
  geom_smooth(method = "lm", se = FALSE, color = "#459395", formula = y ~ x) +
  labs(title = "House Age vs. House Price",
       x = "house age (year)",
       y = "price (unit area)") +
  theme_minimal()
```

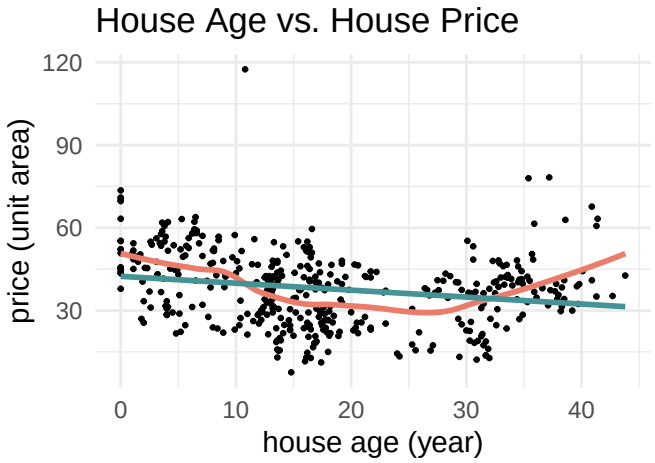

Figure 22: Estate valuation using house age, linear regression and smooth regression methods

5 Visualising Beyond Two Dimensions

This chapter extends beyond the realm of two dimensional data visualisations. Within this chapter, advanced techniques are explored to unravel complex relationships involving multiple variables. The Principal Component Analysis (PCA) is introduced as a powerful method for dimensionality reduction, enabling a concise representation of high-dimensional datasets. Multiple innovative approaches facilitating the visualisation of high-dimensional data in lower-dimensional biplots and t-distributed Stochastic Neighbor Embedding (t-SNE) are studied.

5.1 Principal Component Analysis (PCA)

5.1.1 Theory of PCA

Principal Component Analysis (PCA) is one of the most widely-used dimensionality reduction techniques. The basic idea behind PCA is to project high-dimensional data into a low-dimensional subspace, aiming to create a visual representation that captures most of the variability present in the data.

The feature matrix $\mathbf{X}$ is an $N \times D$ real-valued matrix collecting the observed data:

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_n^T \end{pmatrix} = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,D} \\ & \vdots & & \\ x_{N,1} & x_{N,2} & \cdots & x_{N,D} \end{pmatrix},$$

with N observations and D variables (also known as features).

Define a linear and orthogonal mapping, denoted as $\mathbf{W} : \mathbf{x} \to \mathbf{z}$ which acts as a 'transformer' between high-dimensional data point $\mathbf{x}$ and its low-dimensional projection $\mathbf{z}$. Matrix $\mathbf{W}$ is defined as a $D \times L$ orthogonal and normalised matrix. Therefore, it can be expressed as column vectors $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_L]$, where $\mathbf{w}_i \mathbf{w}_j = 0$ for all $i \neq j$ and $\mathbf{w}_i^T \mathbf{w}_i = 1$. This matrix allows us to project the data from its original high-dimensional space to the low-dimensional subspace using $\mathbf{z} = \mathbf{W}^T \mathbf{x}$; this process is called encoding. To unproject $\mathbf{z}$ and obtain a high-dimensional approximation, denoted as $\hat{\mathbf{x}} = \mathbf{W}\mathbf{z}$, this process is called decoding.

The optimal solution of PCA is obtained by minimising the reconstruction error subject to constraint $\mathbf{W}$, which is defined below:

$$\mathcal{L}(\mathbf{W}) = \frac{1}{N} \sum_{n=1}^{N} ||\mathbf{x}_n - \hat{\mathbf{x}}||_2^2.$$

5.1.2 Derivation of PCA

In general, the vector $\mathbf{w}_i$ is given by the equation $\mathbf{\Sigma}\mathbf{w}_i = \lambda \mathbf{w}_i$, where $\mathbf{\Sigma}$ is the covariance matrix and λ is the i^{th} largest eigenvalue of $\mathbf{\Sigma}$, for $i = 1, \ldots, L$.

Define the covariance matrix $\mathbf{\Sigma}$ as $\mathbf{\Sigma} = \frac{1}{N} \sum_{n=1}^{N} (\mathbf{x}_n - \bar{x})(\mathbf{x}_n - \bar{x})^T$. Without loss of generality, let's assume that the data matrix $\mathbf{X}$ is centred around 0 mean denoted as $\bar{x} = 0$. Then, the covariance matrix can be written as:

$$\mathbf{\Sigma} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n\mathbf{x}_n^T = \frac{1}{N}\mathbf{X}\mathbf{X}^T. \tag{5.1}$$

The optimal encoding $\mathbf{Z}$ ensures the least amount of information loss when data is projected from a high-dimensional space to a lower-dimensional one. To determine this optimal encoding, we minimise the reconstruction error with respect to $\mathbf{w}$ and $\mathbf{z}$.

$$\begin{aligned}\mathcal{L}(\mathbf{w},\mathbf{z}) &= \frac{1}{N}\sum ||\mathbf{x}_n - \mathbf{W}\mathbf{z}_n||^2 \\ &= \frac{1}{N}\sum (\mathbf{x}_n - \mathbf{W}\mathbf{z}_n)^T(\mathbf{x}_n - \mathbf{W}\mathbf{z}_n) \\ &= \frac{1}{N}\sum \mathbf{x}_n^T\mathbf{x}_n - 2\mathbf{x}_n^T\mathbf{W}\mathbf{z}_n + \mathbf{z}_n^T\mathbf{W}^T\mathbf{W}\mathbf{z}_n.\end{aligned} \tag{5.2}$$

To determine the optimal encoding $\mathbf{z}_n$ for the n^{th} observation, we set the derivative with respect to $\mathbf{z}_n$ equal to 0:

$$\frac{\partial\mathcal{L}(\mathbf{w},\mathbf{z})}{\partial\mathbf{z}_n} = \frac{1}{N}(-2\mathbf{x}_n^T\mathbf{W} + 2\mathbf{z}_n) = \mathbf{0}.$$

Thus we find that $\mathbf{z}_n = \mathbf{W}^T\mathbf{x}_n$ yields the optimal encoding for the n^{th} observation.

Next, we aim to find the matrix $\mathbf{W}$ which gives the optimal encoding. We start by expanding and simplifying the loss function $\mathcal{L}(\mathbf{W})$ from equation 5.2:

$$\mathcal{L}(\mathbf{W}) = \frac{1}{N}\sum_{n=1}^{W}\mathbf{x}_n^T\mathbf{x}_n - \sum_{n=1}^{L}\mathbf{W}_l^T(\frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n\mathbf{x}_n^T)\mathbf{W}_l. \tag{5.3}$$

WLOG, our data is assumed to be centred around 0 mean (by 5.1), the loss function can be represented as:

$$\mathcal{L}(\mathbf{W}) = \text{constant} - \sum_{l=1}^{L}\mathbf{W}_l^T\mathbf{\Sigma}\mathbf{W}_l. \tag{5.4}$$

Since $\mathbf{W}$ is a normalised matrix, it has the property $\mathbf{W}_i^T\mathbf{W}_i = 1$ for all $i = 1,\ldots,N$. Therefore, when computing the partial derivative of $\mathcal{L}(\mathbf{W})$ with respect to vector $\mathbf{w}_1$, we can apply the Lagrange multipliers to $\mathcal{L}(\mathbf{w}_1)$ and calculate the derivative of $\hat{\mathcal{L}}(\mathbf{w}_1)$ instead:

$$\hat{\mathcal{L}}(\mathbf{w}_1) = -\mathbf{w}_1^T\mathbf{\Sigma}\mathbf{w}_1 + \lambda(\mathbf{w}_1^T\mathbf{w}_1 - 1), \tag{5.5}$$

Equating $\frac{\partial\hat{\mathcal{L}}(\mathbf{w}_1))}{\partial\mathbf{w}_1} = \mathbf{0}$, the weight vector $\mathbf{w}_1$ which minimise the loss function can be expressed as:

$$\frac{\partial\hat{\mathcal{L}}(\mathbf{w}_1))}{\partial\mathbf{w}_1} = 2\mathbf{\Sigma}\mathbf{w}_1 + 2\lambda\mathbf{w}_1 = \mathbf{0}. \tag{5.6}$$

Therefore $\mathbf{\Sigma}\mathbf{w}_1 = \lambda\mathbf{w}_1$ that is, $\mathbf{w}_1$ is the eigenvector of $\mathbf{\Sigma}$ with eigenvalue λ.

Premultiply by $\mathbf{w}_1^T$, $\mathbf{w}_1^T\mathbf{\Sigma}\mathbf{w}_1 = \lambda\mathbf{w}_1^T\mathbf{w}_1 = \lambda$. To minimise the loss $\mathcal{L}(\mathbf{w}_1)$ we maximise $\mathbf{w}_1^T\mathbf{\Sigma}\mathbf{w}_1$. Thus the largest eigenvalue λ would minimise the loss $\mathcal{L}(\mathbf{w}_1)$.

Adopting the same methodology, the second largest eigenvalue λ_2 would minimise the loss $\mathcal{L}(\mathbf{w}_2)$.

Therefore, order the eigenvectors by their corresponding eigenvalues $\{\mathbf{w}_1, \ldots, \mathbf{w}_L\}$. These form the columns of the weight matrix $\mathbf{W} \in \mathbb{R}^{D \times L}$.

5.2 Biplots

A biplot is a graphical representation that succinctly captures the relationships between variables and observations in a reduced-dimensional space defined by the principal components. This visualisation method facilitates the exploration of complex multivariate data by providing a simultaneous display of both samples (observations) and variables in a single plot. They are commonly understood to be a generalisation of the simple two-variable scatterplot.

At their core, biplots offer a graphical representation of a standardised data matrix, whose rows n are the samples, and whose columns p are the variables, by projecting these onto a two-dimensional plane. To do this, mathematical computations derived from techniques like Singular Value Decomposition (SVD) and Principal Component Analysis (PCA) are required.

In recent years, there have been substantial developments in biplots [12]. The technique has expanded far beyond what is now referred to as the "classical biplots" or "PCA biplots". The concept now finds broad application in conjunction with various other techniques of multivariate analysis. Nevertheless, this section studies the foundations of this visualisation technique, and thus, the emphasis is on biplots within the framework of PCA.

5.2.1 Construction of PCA Biplots

Singular Value Decomposition
Principal component biplots are based on singular value decomposition of the $(n \times p)$ data matrix $\mathbf{X}$ of n observations on p variables with rank r. The singular value decomposition of the standardised matrix $\mathbf{X}$ is found to be the following:

$$\mathbf{X} = \mathbf{UBS}^T,$$

where $\mathbf{U}$ is an $(n \times r)$ orthogonal matrix, and $\mathbf{S}$ is an $(p \times r)$ orthogonal matrix. The columns of $\mathbf{U}$ and $\mathbf{S}$ are the left and right singular vectors. $\mathbf{B}$ is a $(r \times r)$ diagonal matrix with entries $b_1, \ldots, b_r$ such that $b_1 \geq \ldots b_r \geq 0$ [16]. These entries the singular values discussed later on.

In order to find this singular value decomposition, firstly we require the computation of the orthogonal diagonalisation

$$\mathbf{X}^T\mathbf{X} = \mathbf{PDP}^T,$$

where $\mathbf{D}$ is the diagonal matrix of eigenvalues of $\mathbf{X}^T\mathbf{X}$ and $\mathbf{P}$ is an orthogonal matrix composed of the normalised eigenvectors of $\mathbf{X}^T\mathbf{X}$.

Now, $\mathbf{B}$ and $\mathbf{S}$ are computed. Matrix $\mathbf{S}$ is equivalent to the matrix $\mathbf{P}$ previously computed, and matrix $\mathbf{B}$ is a $(n \times p)$ diagonal matrix, where the diagonal entries are the "singular values" with any additional rows and columns of zero to make the dimensions of $\mathbf{B}$ match those of $\mathbf{X}$. The singular values referenced are square root of the eigenvalues of $\mathbf{X}^T\mathbf{X}$.Hence matrix $\mathbf{B}$ is of the form

$$\mathbf{B} = \begin{bmatrix} \mathbf{D} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}.$$

Finally, the $\mathbf{U}$ orthogonal matrix is calculated. The i-th column of $\mathbf{U}$ is given by

$$\mathbf{u}_i = \frac{1}{\sigma_1}\mathbf{X}\mathbf{s}_i,$$

where σ_i are the singular values, and $\mathbf{s}_i$ are the columns of the matrix $\mathbf{S}$ [2].

PCA Biplot Construction

Now, the heuristic argument motivating PCA biplots is the following. Define $\mathbf{B}^\alpha$, for $0 \leq \alpha \leq 1$, as a diagonal matrix who's elements are $b_1^\alpha, \ldots, b_r^\alpha$, with a similar definition for $\mathbf{B}^{1-\alpha}$ [15]. Then, with the decomposition of matrix $\mathbf{X}$ found through SVD, the data can be further rewritten as

$$\mathbf{X} = \mathbf{U}\mathbf{B}^\alpha\mathbf{B}^{1-\alpha}\mathbf{S}^T = \mathbf{U}\mathbf{B}\mathbf{S}^T = \mathbf{G}\mathbf{H}^T,$$

where

$$\mathbf{G} = [\mathbf{g}_1, \ldots, \mathbf{g}_n] = \mathbf{U}\mathbf{B}^\alpha, \quad \mathbf{H}^T = [\mathbf{h}_1, \ldots, \mathbf{h}_p]^T = \mathbf{B}^{1-\alpha}\mathbf{S}^T.$$

Hence, the ith observation of the jth variable can be rewritten as

$$x_{ij} = \mathbf{g}_i^T\mathbf{h}_j.$$

Both the $\mathbf{g}_i$ and $\mathbf{h}_j$ have r elements, and hence, if $\mathbf{X}$ has rank 2, all could be plotted as points in two-dimensional space [15]. Note that for the more general case, where $r > 2$, x_{ij} can be written as

$$x_{ij} = \sum_{k=1}^{r} u_{ik}b_k s_{jk}$$

which is often well approximated by

$$_m\tilde{x}_{ij} = \sum_{k=1}^{m} u_{ik}b_k s_{jk}, \quad \text{with } m < r.$$

But this can be written

$$_m\tilde{x}_{ij} = \sum_{k=1}^{m} \mathbf{g}_{ik}\mathbf{h}_{jk} = \mathbf{g}_i^{*T}\mathbf{h}_j^*,$$

where $\mathbf{g}_i^*, \mathbf{h}_j^*$ contain the first m elements of $\mathbf{g}_i$ and $\mathbf{h}_j$ respectively. Hence, suggesting that if instead of using the $\mathbf{g}_i$ and the $\mathbf{h}_j$, we can just use their first elements (say, 2), respectively denoted by $\mathbf{g}_i^*$ and $\mathbf{h}_j^*$, to get

$$x_{ij} \approx \mathbf{g}_i^{*T}\mathbf{h}_j^*.$$

Thus, $\mathbf{g}_i$ and $\mathbf{h}_j$ should provide a reasonable two-dimensional approximation of the n observations and the p variables [16].

5.2.2 Biplots in Practice: The mtcars Dataset

This section presents a comparative analysis of PCA biplots generated through two distinct methodologies: manual mathematical construction and use of R's in-built functions. The focus lies on elucidating the mathematical intricacies, exploring relations, and highlighting differences between the two approaches. The mtcars dataset will serve as a benchmark.

Methodologies Discussion
Traditionally, biplots are constructed manually through mathematical operations based on SVD discussed previously. However, with computational tools like R, in-built functions such as `prcomp()` and `biplot()` streamline the process, removing tidious mathematical intricacies, which becomes particularly interesting when working with larger datasets.

The manual construction, which results in the biplot shown in Figure 23, begins with data standardisation followed by SVD decomposition to obtain matrices representing row and column contributions to principal components. These matrices are then utilised to construct the biplot. Conversely, the in-built function approach employs R's `prcomp()` function for PCA and directly generates the biplot using `biplot()`, without the need for explicit mathematical operations. The product of this method is shown in Figure 23.

```
#MANUAL CONSTRUCTION OF A BIPLOT IN R
# Standardise the data
scaled_data <- scale(mtcars)

# Set the alpha parameter for SVD
alpha <- 0

# Preprocess the data using SVD
preprocess <- scaled_data - matrix(colMeans(scaled_data), nrow = nrow(scaled_data), ncol = ncol(scaled_data), byrow = TRUE)
svd_mtcars <- svd(preprocess)
U <- svd_mtcars$u
L <- diag(svd_mtcars$d)
A <- svd_mtcars$v

# Calculate G and H matrices
G <- U %*% (L%^%(alpha))
H <- t((L%^% (1-alpha)) %*% t(A))

# Extract the first two columns for biplot
G2 <- G[, 1:2]
H2 <- H[, 1:2]

# Set up a side-by-side layout
par(mfrow = c(1, 2))

# Create a biplot-like plot for manual construction with title
plot(G2, xlim = c(-0.25, 0.4), xlab = "PC1", ylab = "PC2", main = "Manual Mathematical Construction")

par(new = TRUE)
plot(H2, xlim = c(-4.5, 8), ylim = c(-4.5, 8), col = "#EB7C69",
     xaxt = "n", yaxt = "n", xlab = "", ylab = "")
for (i in 1:ncol(scaled_data)) {
  lines(c(0, H2[i, 1]), c(0, H2[i, 2]), col = "#EB7C69", lwd = 1.5)
}

axis(3)
axis(4)

#GENERATION OF A BIPLOT WITH R'S BUILT-IN FUNCTIONS
# Standardise the data
scaled_data <- scale(mtcars)

# Perform Principal Component Analysis (PCA)
```

```r
pca_result <- prcomp(scaled_data, scale. = TRUE)

# Create a biplot with title
biplot(pca_result, cex = 0.7, main = "R's Built-in Function")
```

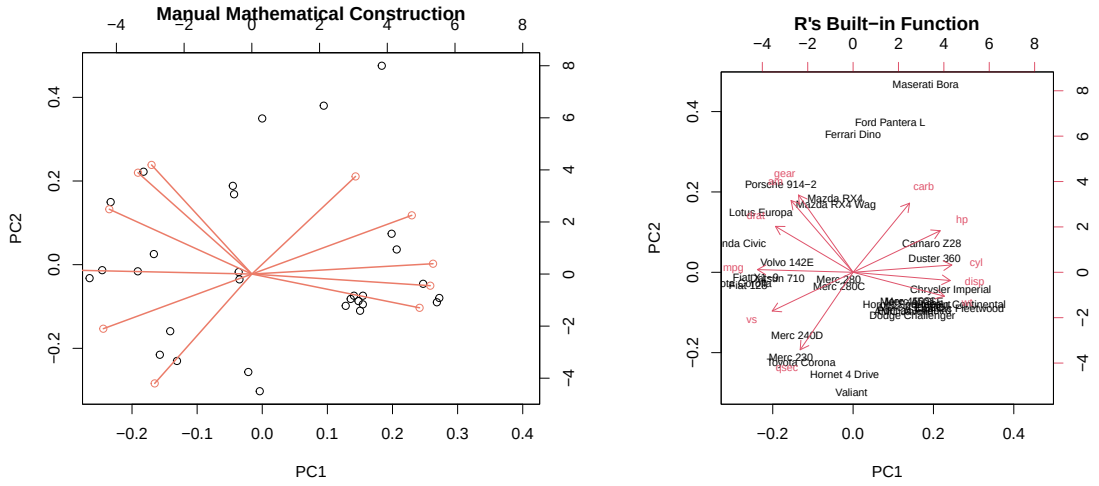

Figure 23: PCA biplots of mtcars dataset

Comparative analysis of these two plots reveals several key findings. The manual construction approach offers meticulous control over the biplot's customisation, allowing for fine-tuning of parameters and appearance. Conversely, the in-built function approach provides a user-friendly ideal for practitioners less inclined towards mathematical intricacies the visualisation. However, despite differences in implementation, both methodologies produce visually comparable biplots, showcasing identical patterns and relationships within the data.

5.3 Principal Curves

Principal curves stand as a vital tool in exploratory data analysis, providing a nonlinear analogue to PCA for capturing the underlying structure of multidimensional data. While PCA focuses on finding linear projections that maximise variance, principal curves seek to identify smooth, nonlinear trajectories that capture the most significant variations in the data. These curves are essentially one-dimensional paths embedded within the multidimensional space of the data points.

Hence, if the points don't fall in a linear subspace, that is, they fall in a non-linear subspace such as a curve in 2-dimension, PCA is insufficient. An example of this is the scatter plot on the left side of Figure 24. In this case, a straight line can't be placed in the middle of the plot, however, one could fit a curve that passes through the centre of the points. This is the purpose that principal curves serve

5.3.1 Construction of Principal Curves

As with PCA, take $(n \times p)$ data matrix $\mathbf{X}$, with row vectors $\mathbf{x}_i$. The goal is to extract a lower-dimensional set of "factors" or "scores" $\mathbf{y} = (y_1, \ldots, y_n)^T$. These satisfy the following model:

$$\mathbf{x}_i = f(y_i) + \epsilon_i,$$

where ϵ_i are errors with mean 0 and variance $\sigma^2 \mathbf{I}_p$ and $f : \mathbb{R}^1 \to \mathbb{R}^p$ is a continuous function called a curve [16].

For curve $\mathbf{f}$, the projection index $y_f(\mathbf{x})$ maps observation $\mathbf{x} \in \mathbb{R}^p$ to the point of $\mathbf{f}$ that is closest, returning the score (index). If there are several such points, it returns the largest (this is arbitrary choice to ensure it is a well defined function). Hence, the projection index is [16]:

$$y_f(\mathbf{x}) = \sup_y \{y : ||\mathbf{x} - \mathbf{f}(y)|| = \inf_\mu ||\mathbf{x} - \mathbf{f}(\mu)||\}.$$

As seen in a similar form in PCA, principal curves estimates the following least-squares objective function

$$\min_{\mathbf{f}} \sum_{i=1}^{n} ||\mathbf{x}_i - \mathbf{f}(y_{\mathbf{f}}(\mathbf{x}_i))||^2.$$

To construct the principal curve, an algorithm put to work. This iterates between finding a curve $\mathbf{f}$ and then projecting the points onto that. More explicitly, the algorithms has the following form [16]:

1. Initialise: let iteration counter $h = 1$ and set $\mathbf{y}^{(0)} = \mathbf{X}\mathbf{u}$, where $\mathbf{u}$ is the first PC vector.
2. Smoothing step: With $\mathbf{y}^{(h-1)}$ fixed, estimate $\hat{\mathbf{x}}_i^{(h)}$ with a smoother.
3. Projection step: With $\hat{\mathbf{X}}^{(h)}$ fixed, use the projection index to update the scores $\mathbf{y}(h)$ so that they have unit speed.
4. Loop: Increment h and return to step 1 while the change in the objective function above some threshold.

5.3.2 Principal Cuves in Practice

In Figure 24, we see an example of how a principal curve is fitted to the observations of an artificial dataset. The points in the plots are not grouped in a linear subspace, but rather follow a similar shape to that of cubic function.

Using the built-in `principal-curve()` function, a principal curve can be fitted to the dataset. The principal curve traverses through the densest regions of the data distribution, capturing the underlying nonlinear structure. The panel on the left of Figure 24 depicts the principal curve overlaid on the dataset, providing insights into the central tendency and inherent curvature of the data. Additionally, whiskers are drawn to visualise the perpendicular distances between data points and the principal curve.

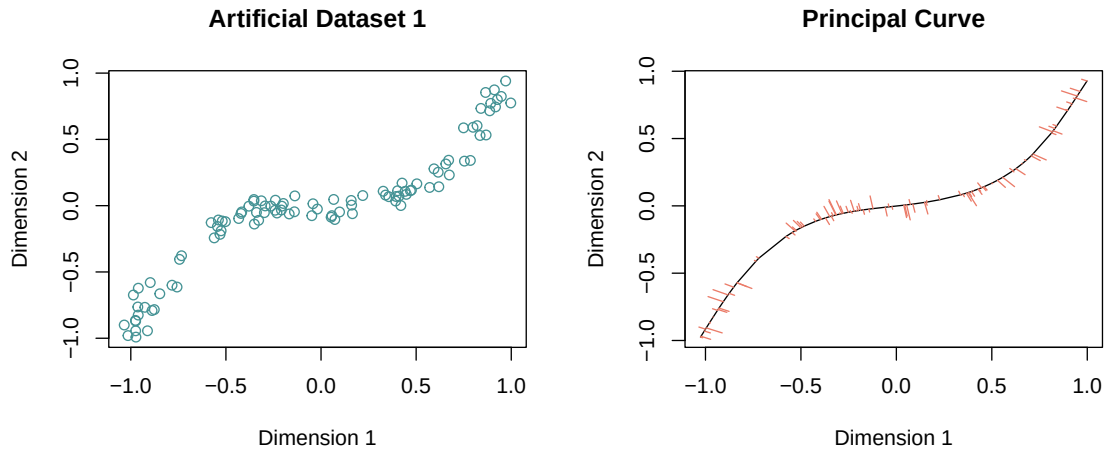

Figure 24: Principal curve fitting illustration on artificial dataset

The principal curve effectively captures the nonlinear relationship between the dimensions of the artificial dataset. By flexibly adapting to the curvature of the data distribution, the principal curve reveals intricate patterns and trends that may not be captured by linear methods like PCA. The visualisation aids in understanding the underlying structure of the dataset.

5.4 t-Distributed Stochastic Neighbor Embedding (t-SNE)

Since PCA is a linear method and may not perform well with non-linear data structures, often missing complex non-linear relationships. Although effective in capturing the global structure of data, PCA focuses on maintaining large pairwise distances to maximise variance and overlooks important local patterns and structures.

Unlike PCA, t-SNE[29] is a nonlinear technique for embedding high-dimensional data for visualisation in a low-dimensional space of two or three dimensions, focusing on preserving the small pairwise similarities between data points thus retain the local structure of the dataset in a lower-dimensional space. It starts by converting high-dimensional Euclidean distances into probabilities that reflect the similarity between points, then maps these points to a lower-dimensional space in a way that tries to preserve these similarities.

5.4.1 Theory of t-SNE

For a given high-dimensional dataset $\{x_1, ..., x_n\}$, the t-SNE algorithm quantifies the similarity between pairs of data points, x_i and x_j, using conditional probabilities $p_{j|i}$. These probabilities are calculated under the assumption that if x_i were to choose its neighbours, it would do so in proportion to their probability density under a Gaussian distribution centered at x_i. The conditional probability $p_{j|i}$[29] for $i \neq j$ is mathematically expressed as:

$$p_{j|i} = \frac{\exp(-||\mathbf{x}_i - \mathbf{x}_j||^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-||\mathbf{x}_i - \mathbf{x}_k||^2/2\sigma_i^2)},$$

where σ_i denotes the variance of the Gaussian that is centered on data point x_i. Also, set similarities of the same pair wise points $p_{i|i} = 0$ and $\sum_j p_{j|i} = 1$ for all i.

To create a symmetrical joint probability distribution in the high-dimensional space, t-SNE averages the conditional probabilities $p_{j|i}$ and $p_{i|j}$[29], resulting in:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n},$$

where n is a number of data points. Also, $p_{ii} = 0$ and $\sum_{i,j} p_{ij} = 1$.

In the low-dimensional space (usually 2D or 3D), we have the corresponding dataset $\{y_1, ..., y_n\}$. Since t-SNE aims to find mapped points y_i and y_j in a low-dimensional space that reflects the similarities p_{ij} as well as possible. In the low-dimensional space, t-SNE computes similarities q_{ij} using a similar formula but with a Student t-distribution (with one degree of freedom, equivalent to the Cauchy distribution) which has heavier tails than a Gaussian distribution to compute the joint probabilities[29] of the mapped points:

$$q_{ij} = \frac{(1 + ||\mathbf{y}_i - \mathbf{y}_j||^2)^{-1}}{\sum_{k \neq l}(1 + ||\mathbf{y}_k - \mathbf{y}_l||^2)^{-1}}.$$

Similarly, set $q_{ii} = 0$.

The goal of t-SNE is to minimize the discrepancy between the probabilities p_{ij} in the high-dimensional space and q_{ij} in the low-dimensional space. Therefore, the objective is to minimise the Kullback-Leibler (KL) divergence[29] between the joint probability distributions P and Q:

$$C = \mathrm{KL}(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}},$$

where the C means the cost function that we wish to minimize, and the Kullback-Leibler (KL) divergence is a measure of how probability distribution P differs from probability distribution Q. In addition, the lower the KL divergence value, the closer the two distributions are. A KL divergence of zero means that these two distributions are the same.

To continue with the minimisation, compute the gradient of symmetric SNE[29] as follows:

$$\frac{\partial C}{\partial y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + ||y_i - y_j||^2)^{-1}.$$

Once the gradient of the cost function with respect to the positions of the points in the low-dimensional space, $\frac{\partial C}{\partial y_i}$, is computed, t-SNE employs gradient descent to minimizes the Kullback-Leibler (KL) divergence between the high-dimensional probability distribution P and the low-dimensional probability distribution Q. Gradient descent is an iterative optimization algorithm used to minimize the cost function by updating the parameters in the opposite direction of the gradient of the cost function. In the case of t-SNE, the parameters are the positions of the points in the low-dimensional space. The update rule[29] at each iteration t for a point y_i is given by:

$$y_i^{(t+1)} = y_i^{(t)} - \eta \frac{\partial C}{\partial y_i},$$

where η is the learning rate, a positive scalar determining the step size.

The process is repeated for a number of iterations or until the change in the cost function between iterations is below a predetermined threshold, indicating convergence. The gradient descent process in t-SNE is crucial for effectively reducing the dimensionality of high-dimensional data while preserving the local structure of the data. By iteratively updating the positions of the points in the low-dimensional space, t-SNE minimizes the KL divergence, resulting in a meaningful representation of the data in lower dimensions.

5.4.2 Case Example: t-SNE in Practice

R provides us with a dedicated package for t-SNE analysis, *Rtsne*, which allows us to conduct the analysis conveniently using the *Rtsne* function. In the graph presented below, we observe a t-SNE visualization of the penguin dataset.

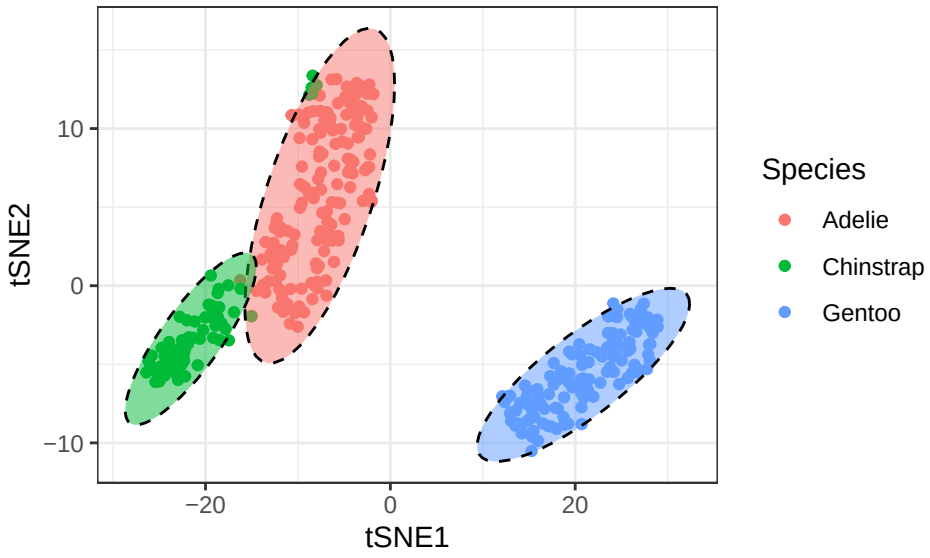

Figure 25: t-SNE of Penguins Dataset

The provided plot in Figure 25 is a t-SNE visualization of the Penguins dataset, rendered using the *ggplot*2 package in R. In this case, using the T-sne algorithm helps us to reduce the multidimensional numerical data of the penguin dataset to a 2-dimensional space. We also use scatter plots and elliptical areas to visualize the distribution of different penguin species. The data points are colour-coded to distinguish between three penguin species: Adelie in red, Chinstrap in green, and Gentoo in blue. This colour coding aids in the visual differentiation of the species clusters.

The axes are labeled $tSNE1$ and $tSNE2$, corresponding to the dimensions reduced by the t-SNE algorithm. Dashed ellipses overlaid on the scatter plot demarcate the clusters of each species, providing a visual guide to the density and separation of the species within the transformed feature space. The plot effectively uses the t-SNE technique to illustrate the grouping of species, highlight-

ing the algorithm's utility in discerning inherent data patterns in a lower-dimensional representation.

While t-SNE is an exceptionally potent tool for data visualization, it does come with its own set of constraints. Firstly, it has a significant memory footprint and can be time-consuming to run, which may pose challenges when dealing with large datasets.

Secondly, t-SNE is tailored specifically for visualization, which constrains the embedding space to two or three dimensions. This limitation means that t-SNE is optimised for human interpretability rather than for capturing higher-dimensional relationships.

Additionally, it requires experimentation with different initialisation to mitigate the risk of local suboptimal solutions affecting the results. Therefore, it's crucial to carefully consider and select the most suitable method based on the specific requirements and nature of the data at hand.

6 State-Of-The-Art Modern Approaches

In this chapter, we will introduce some state-of-the-art graphical statistical methods published in recent research. In Chapter 6.1, we will introduce the foundational visualisation methods box plots and Q–Q plots. Building on this, we will discuss the theory and implementation of functional boxplots and Q–Q Box plots, as published in recent articles in the *Journal of Computational and Graphical Statistics*.

6.1 Introduction

6.1.1 Box Plots in Practice

Box plots, also known as box-and-whisker plots, were introduced by John W. Tukey in 1977. Since then, they have become essential in exploratory data analysis, offering a summary of a dataset's central tendency, skewness, and outliers. These plots typically showcase the median, quartiles, and range, making them valuable for comparing and discerning patterns across different variables.

An example of box plots using the iris dataset is displayed in Figure 26. Specifically, the right graph showcases a violin plot — which is a recent advancement in box plot techniques. The violin plot combines a box plot with a kernel density function, providing a mirrored density plot displayed in the same way as a box plot.

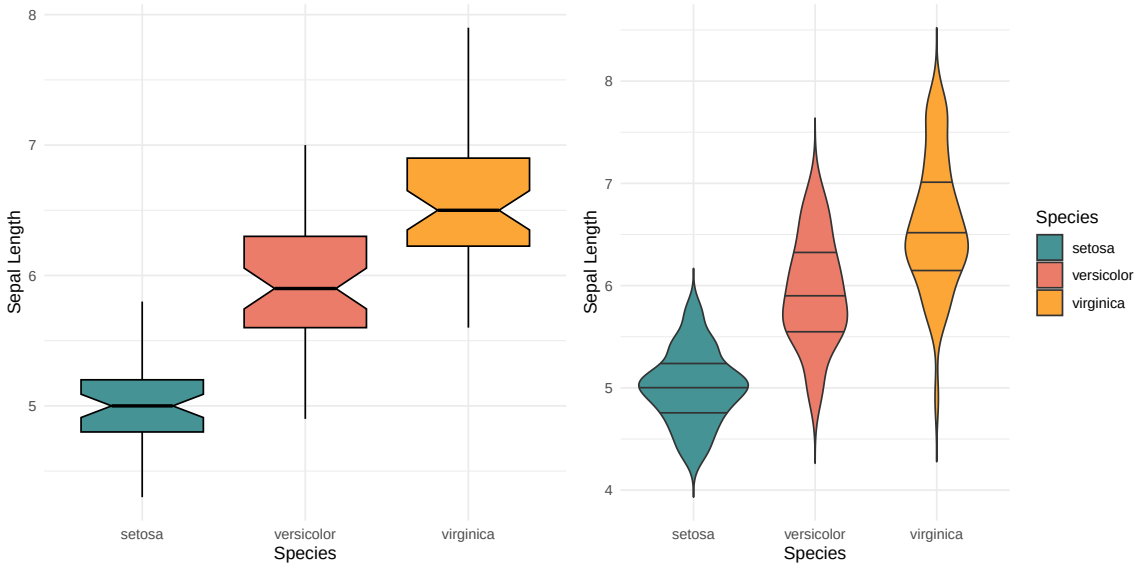

Figure 26: Comparison of Violin and Box Plots of Sepal Length by Species

6.1.2 Q–Q Plots in Practice

Q–Q plots are constructed by plotting the quantiles of two datasets against each other. Typically, one dataset serves as the basis for theoretical quantiles, often assumed to follow a particular probability distribution, while the other dataset comprises observed values. This comparative depiction

enables a direct assessment of how closely the observed data align with the theoretical distribution, facilitating the identification of deviations from expected patterns.

In this section, side-by-side QQ plots are displayed offering a visual comparison of the distributional properties of two variables from the mtcars and the iris datset. These plots serve as a means to assess normality assumptions and identify potential deviations from expected distributions. Through this visual exploration, we aim to gain insights into the underlying structure of the data and guide further investigation.

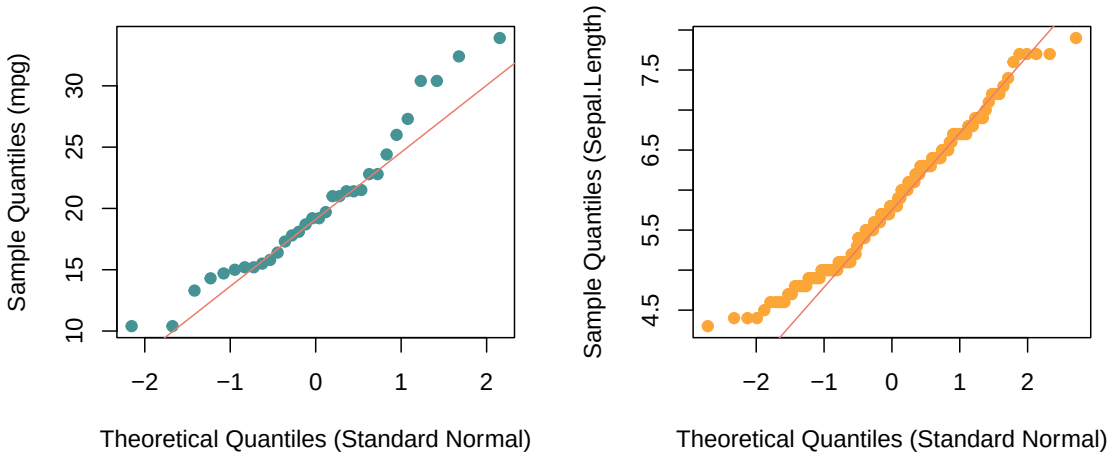

Figure 27: Q–Q plot of mpg in mtcars dataset

In the first plot in Figure 27, which examines the mpg variable in the mtcars dataset, deviations, particularly in the tails of the distribution, are noticeable. These deviations suggest potential departures from normality, possibly indicating the presence of outliers or non-normal behaviour in the data.

The second plot examines the distribution of the Sepal.Length variable in the iris dataset. Here, the points exhibit a slight departure from the theoretical quantiles line. The deviation is less pronounced compared to the previous plot. The distribution of sepal lengths within the iris dataset appears to be approximately normal.

In general, the Q–Q plots provide a succinct and insightful representation of the distributional properties of the variables under examination. They offer a visual means to assess normality assumptions and identify potential outliers or deviations from expected distributions. As such, Q–Q plots serve as a valuable tool in exploratory data analysis and model diagnostics, aiding researchers in understanding the underlying structure of their data and guiding subsequent analyses.

6.2 Functional Boxplot

Building upon the traditional box plot, the functional boxplot is an innovative extension that provides a comprehensive visual representation of functional data. The i^{th} observation of the population is represented as a real function $y_i(t)$, where $i = 1, \ldots, n, t \in \mathcal{I}$, where $\mathcal{I}$ is an interval on the real line.

This section aims to unravel the methodology behind functional boxplots, elucidating their construction, interpretation, and application in real-world datasets, which enables a deeper understanding of the patterns and anomalies inherent in functional data.

6.2.1 Theory of Funtional Boxplot

The construction of a box plot relies on data ordering, where you simply arrange univariate observations from smallest to largest. However, with multivariate data, this process becomes significantly more complex, necessitating advanced methodologies to establish a meaningful order. Various versions of data depth have been developed to assess the centrality (or depth) of an observation or its deviation (outlyingness), from a population distribution with density P.

Consider a set of observations $y_1(t), \ldots, y_n(t)$ defined over an interval $\mathcal{I} \in \mathbb{R}$. The graph of a function $y(t)$ is defined as $G(y) = (t, y(t)) : t \in I$. Applying the concept of range from univariate data to two dimensions, we derive a band in $\mathbb{R}^2$ bounded (also known as delimited) by curves $y_{i_1}, \ldots, y_{i_k}$ for time (or states) $1, \ldots, k$. Band is defined as $B(y_{i_1}, \ldots, y_{i_k}) = \{(t, x(t)) : t \in \mathcal{I}, \min_{r=1,\ldots,k} y_{i_r}(t) \leq x(t) \leq \max_{r=1,\ldots,k} y_{i_r}(t)\}$. Let J denote the number of curves that determines the band, where J is a fixed integer $2 \leq J \leq n$.

Band depth is a measure of how deeply a curve lies within a population of curves, indicating its centrality or outlyingness relative to others. This measure is valuable for understanding the distributional characteristics of functional data across a population, offering insights into patterns and variability within the dataset.

For a stochastic process $Y(t)$ generating the observations $y_1(t), \ldots, y_n(t)$, the population version of the band depth for a curve $y(t)$ with respect to the probability measure P is defined as:

$$BD_J(y, P) = \sum_{j=2}^{J} BD^{(j)}(y, P) = \sum_{j=2}^{J} P\{G(y) \subset B(Y_1, \ldots, Y_j)\}, \tag{6.1}$$

where $B(Y_1, \ldots, Y_j)$ is a band delimited by j random curves [26]. Here, $BD^{(j)}(y, P)$ calculates the probability that the graph of $y(t)$ lies entirely within a band formed by any selection of j functions. These bands are delineated by the minimum and maximum values at each point t among the selected j functions. This formulation of BD facilitates a nuanced understanding of curve centrality and variability within functional data, laying the groundwork for the construction of functional boxplots. Then we can define the sample version of $BD(j)(y, P)$, which is the fraction of the bands determined by j different sample curves containing the whole graph of the curve $y(t)$ [26]:

$$BD_n^{(j)}(y) = \binom{n}{j}^{-1} \sum_{1 \leq i_1 < i_2 < \cdots < i_j \leq n} I\{G(y) \subseteq B(y_{i_1}, \ldots, y_{i_j})\},$$

where $I\{\}$ denotes the indicator function. Hence, by computing the fraction of the bands containing the curve $y(t)$, the bigger the value of band depth, the more central position the curve has. Then,

the sample band depth of a curve $y(t)$ is [26]

$$BD_{n,J}(y) = \sum_{j=2}^{J} BD_n^{(j)}(y).$$

A sample median function is a curve from the sample with largest depth value, defined by [26]

$$\text{argmax}_{y \in y_1,\ldots,y_n} BD_{n,J}(y).$$

In a traditional boxplot, the box represents the central 50% of the data. In functional box plot, estimate the central region of data using a band defined by the α proportion (where $0 < \alpha < 1$) of the deepest curves in the sample. Specifically, the central region for 50% of the sample, denoted as $C_{0.5}$, is calculated using the formula [26]:

$$C_{0.5} = \{(t, y(t)) : \min_{r=1,\ldots,\lceil n/2 \rceil} y[r](t) \leq y(t) \leq \max_{r=1,\ldots,\lceil n/2 \rceil} y[r](t)\},$$

where $\lceil n/2 \rceil$ is the smallest integer not less than half of the sample size. This 50% central region functions similarly to the inter-quartile range (IQR) in a boxplot, providing a measure of the spread of the central 50% of the curves. This method is robust, less affected by outliers or extreme values, and offers a clearer visualization of the distribution's central spread. Within this central region, the curve that represents the median, denoted as $y_{[1]}(t)$, signifies the most centrally located curve with the highest band depth, serving as a robust measure of central tendency.

6.2.2 Functional Box Plot Example

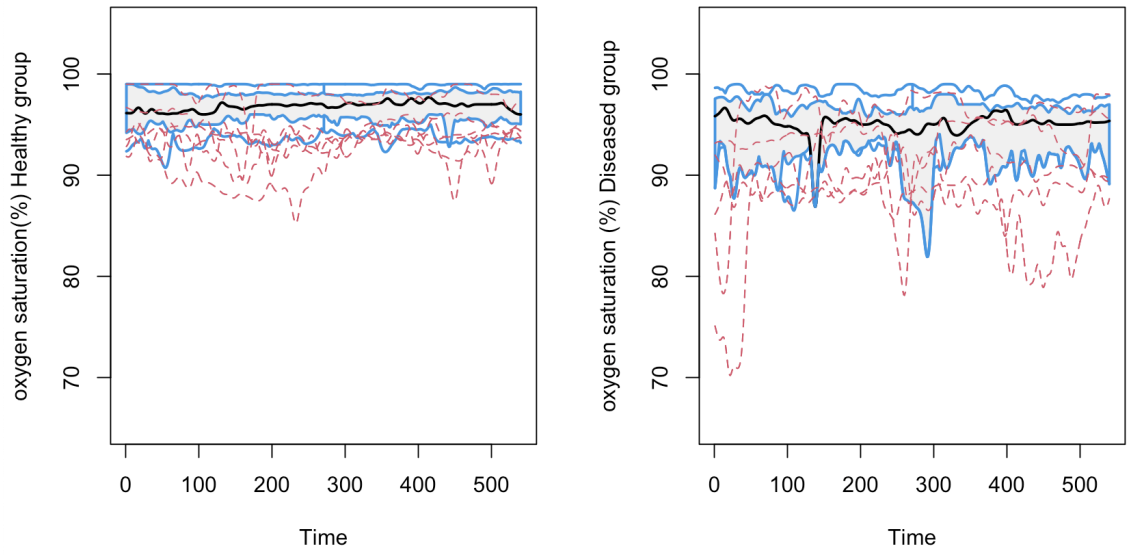

Figure 28: Left graph demostrates curves of arterial oxygen saturation (%) for 80 women without metabolic syndrome and right graph demostrates curves of arterial oxygen saturation (%) for 35 women with metabolic syndrome. Outliers from the population are highlighted in red.

6.3 Q–Q Boxplots

In this section, we will introduce the Q–Q boxplot, which combines the advantages from the boxplot and the Q–Q plot. The Q–Q boxplot integrates the box plot's ability to summarize data through its quartiles and the Q–Q plot's capability to compare data distributions to a theoretical distribution (often the normal distribution).

Although Q–Q plots and box plots are very effiective in data visualization, they have there limitations as well. Q–Q plot is less effective than box plot in highlighting summary quantitles, and it also suffers the readability when we place multiple samples together. Box plot usually has unreliable tail information when $n < 100$, therefor may not suitable for some research questions that need more accurate estimates of extreme quantiles.

In this section, we will introduce the Q–Q boxplot, which combines the advantages from the boxplot and the Q–Q plot. The Q–Q boxplot integrates the box plot's ability to summarize data through its quartiles and the Q–Q plot's capability to compare data distributions to a theoretical distribution (often the normal distribution).

6.3.1 Construction of Q–Q Boxplots

The construction of a Q–Q boxplot involves a three-step process. The first setp is to construct the box, which is similar to the standard boxplot and represents the interquartile range of the data. The second step is to draw the whiskers and confidence bands. The third step is to draw the outside values. The first and third steps are as same as in box plots.

The unique aspect of the Q–Q boxplot comes in the second step. In ordor to draw the whiskers and confidence bands, we have to standardize both the comparison distribution and the reference distribution. This standardization process involves matching the quantiles from the reference distribution to those in the comparison distribution, either by subsampling or interpolation. Once this is done, the deviations of the comparison distribution from the reference distribution are calculated relative to each data point. Alongside this, the point-wise confidence bands are also determined, which provides a visual indication of how closely the data follows the expected theoretical distribution. To integrate the relative deviations and confidence bands into the Q–Q boxplot, we include both the x and y coordinates in the plot. The y-axis coordinates correspond to the quantile values from the comparison dataset, while the x-axis coordinates represent the deviations relative to the reference distribution.

To calculate the x-coordinate for a Q–Q boxplot, we set a scale for the plot and determin the maximum x-coordinate, $x_{\max} = (\frac{b_w}{2})$ where b_w is the box width. Let dev_{max} be the maximum of the absolute values of the whisker deviations and confidence band values. The value of a relative deviaition d is computed by either taking the difference between the standardized values from the comparison distribution and those from the reference distribution, or by using a value from the confidence band associated with the reference distribution. We can then set:

$$x^* = \frac{d \cdot x_{\max}}{\mathrm{dev}_{\max}}.$$

The x coordinate is then $x_b + x*$, where x_b marks the center of the boxplot.

6.3.2 Q–Q Boxplots in Practice

The resulting Q–Q boxplot shows the distribution of sepal length values within each iris species, allowing comparison whether they come from a normal distribution. Outliers are also indicated in the plot.

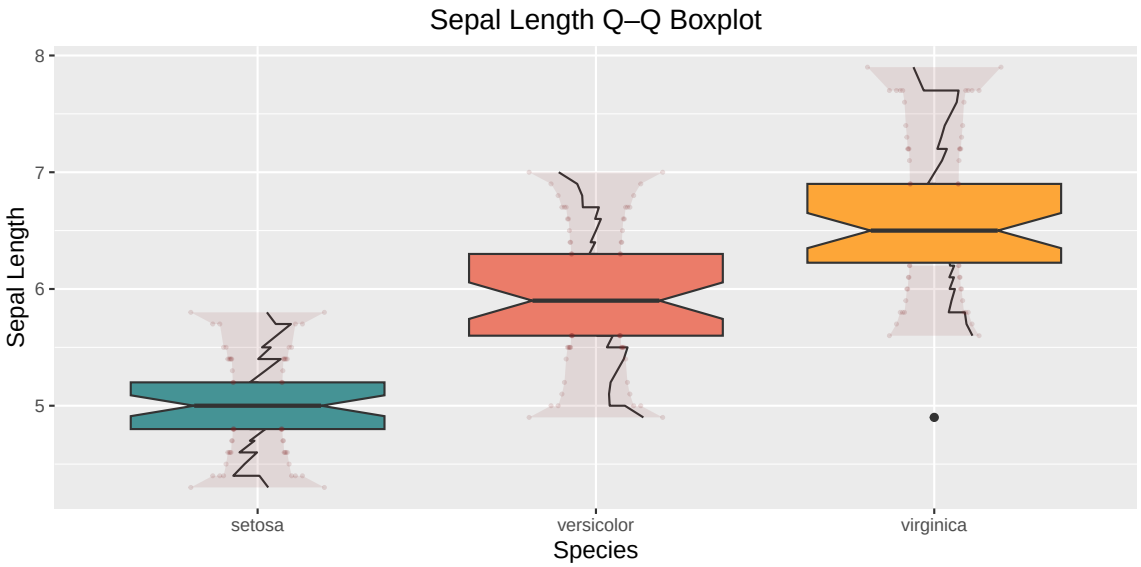

Figure 29: Q–Q boxplot of iris dataset

From the above Figure 29, we can find out easily that the middle of our Q-Q boxplot is same as in the boxplot as we seen in Figure 26. In addition, there are more details in tails behavior simultaneously. We can see that the right-tail deviations and left-tail deviations all lie inside of the confidence bands calculated with the bootstrap method, which means the sample distribution of our dataset is consistent with the theoretical distribution being compared to, that is a normal distribution.

7 Index

Accuracy: Emphasising faithful reflection of data to reduce distortion or misinterpretation.

Clarity: Ensuring visuals are easily understood without unnecessary complexity.

Cognitive Load: The mental effort required to process information presented visually.

Cognitive psychology: Understanding how mental processes affect perception and understanding of visual data.

Colour Theory: Principles guiding the strategic use of colour in visualisations for clarity and impact.

Colour encoding: Utilising colours to represent data categories or values in visualisations.

Colour perception: Understanding how humans perceive and interpret colours in visualisations.

Consistency: Maintaining uniform use of visual elements throughout a visualisation.

Data abstraction: Simplifying and structuring raw data into comprehensible visual forms.

Gestalt Principles: Rules affecting how visual elements are grouped and interpreted in perception.

Hierarchies of abstraction: Representing data at various levels of detail in visualisation.

Information overload: Occurs when excessive data or visual elements overwhelm understanding.

Interval data: Ordered categories with equal intervals but lacking a true zero point.

Nominal data: Represents categories or labels without any inherent order.

Ordinal data: Implies a meaningful order among categories but lacks equal intervals.

Ratio data: Includes ordered categories with equal intervals and a meaningful zero point.

Relevance: Presenting information pertinent to the addressed message or question.

References

[1] $500,000 in 1937 is how much today? https://www.calculateme.com/inflation/500000-dollars/from-1937/to-now, n.d. Accessed 9 Nov. 2023.

[2] Kirk Baker. Singular value decomposition tutorial. *The Ohio State University*, 24:22, 2005.

[3] Peter J. Brockwell and Richard A. Davis. *Introduction to Time Series and Forecasting*. Springer, 2016. Accessed 13 Nov. 2023.

[4] K. Dakwa. The kallikak family – historical influences, current controversies, teaching resources. https://intelltheory.com/intelli/the-kallikak-family/, 2001. Accessed 9 Nov. 2023.

[5] Mehdi Dastani. The role of visual perception in data visualization. *Journal of Visual Languages & Computing*, 13(6):601–622, 2002.

[6] M. de Carvalho. Incomplete data analysis. University of Edinburgh, Lecture 2, 9-10, 2023.

[7] Earth Science Data Systems. Firms frequently asked questions — earthdata. https://www.earthdata.nasa.gov/faq/firms-faq#ed-confidence, 2023. Accessed 12 Nov. 2023.

[8] Celso Silva et. al. Deforestation-induced fragmentation increases forest fire occurrence in central brazilian amazonia. *Forest*, 9(6):305, 2018.

[9] William S. Cleveland et. al. *Local regression models. Chapter 8 of Statistical Models in S*. Wadsworth Brooks/Cole, 1992.

[10] Tom Fawcett. Introduction to roc analysis. *Pattern Recognition Letters*, 27:861–874, 06 2006.

[11] JP García, JC Ferreira, and CM Patino. Receiver operating characteristic analysis: an ally in the pandemic. *J Bras Pneumol*, 47(2):e20210139, Apr 30 2021.

[12] John C Gower and David J Hand. *Biplots*, volume 54. CRC Press, 1995.

[13] E. Hawkins. Warming stripes — climate lab book. https://www.climate-lab-book.ac.uk/warming-stripes/, n.d. Accessed 9 Nov. 2023.

[14] Kieran Healy. *Data visualization: a practical introduction*. Princeton University Press, 2018.

[15] M. J. Blanchet, de Carvalho. Applied statistics. Ecole Polytechnique Federale De Lausanne, Lecture 6, 2011.

[16] Ian T Jolliffe. Principal component analysis. *Technometrics*, 45(3):276, 2003.

[17] F. Nightingale. Diagram of the causes of mortality in the army in the east. https://www.davidrumsey.com/luna/servlet/detail/RUMSEY~8~1~327826~90096398:Diagram-of-the-Causes-of-Mortality-;JSESSIONID=802dd785-32fc-4b43-a53d-72ed57e15cc8?qvq=q%3Aauthor%3D%22Nightingale%2C%20Florence%22%3Blc%3ARUMSEY%7E8%7E1&mi=1&trs=10, 1859. Accessed 20 Oct. 2023.

[18] Margaret Sullivan Pepe. *The Statistical Evaluation of Medical Tests for Classification and Prediction*, volume 28. Oxford University Press, United Kingdom, 2003. Print.

[19] A. Proctor. Five charts that changed the world - bbc ideas. https://www.bbc.co.uk/ideas/videos/five-charts-that-changed-the-world/p0fb69c1, 2023. Accessed 20 Oct. 2023.

[20] B. D. Ripley. loess: Local polynomial regression fitting. https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/loess, 1998. Accessed 20 Jan. 2024.

[21] Muhammad Hafiz Wan Rosli and Andres Cabrera. Gestalt principles in multimodal data representation. *IEEE computer graphics and applications*, 35(2):80–87, 2015.

[22] A. Rutherford. Where science meets fiction: the dark history of eugenics. https://www.theguardian.com/science/2022/jun/19/where-science-meets-fiction-the-dark-history-of-eugenics, 2022. Accessed 9 Nov. 2023.

[23] David W Scott. *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons, 2015.

[24] J.D. Smith and M.L. Wehmeyer. Who was deborah kallikak? *Intellectual and Developmental Disabilities*, 50(2):169–178, 2012.

[25] Josh Starmer. Lowess and loess, clearly explained!!! https://www.youtube.com/watch?v=Vf7oJ6z2LCc, 2018. Accessed 3 Jan. 2024.

[26] Ying Sun and Marc G. Genton. Functional boxplots. *Journal of Computational and Graphical Statistics*, 20(2):316–334, 2011.

[27] Dejan Todorovic. Gestalt principles. *Scholarpedia*, 3(12):5345, 2008.

[28] Edward R Tufte. *The visual display of quantitative information*, volume 2. Graphics press Cheshire, CT, 2001.

[29] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[30] Matt P Wand and M Chris Jones. *Kernel smoothing*. CRC press, 1994.

[31] Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016.