Working with Graphs: Graphical Parameters, Symbols, and Colors

Project submitted to the

SRM University – AP, Andhra Pradesh

for the partial fulfillment of the requirements to award the degree of

Bachelor of Technology/Master of Technology

In

Computer Science and Engineering

School of Engineering and Sciences

Submitted by

Candidate Name

Sai Rohith Kumar Banka	**(AP21110010795)**
Sreeroop Veerapaneni	**(AP21110010831)**
Revanth Upadhyayula	**(AP21110010834)**
Chakrapani Maale	**(AP21110010920)**
Kodali Hemanth	**(AP21110010949)**
Vishnu Vardhan Kondapalli	**(AP21110010989)**

Under the Guidance of

Prof. Saleti Sumalatha

SRM University–AP

Neerukonda, Mangalagiri, Guntur

Andhra Pradesh – 522 240

[Novermber, 2024]

Certificate

Date: 28-Nov-24

This is to certify that the work present in this Project entitled "***Working with Graphs***" has been carried out by **Rohit, Sreeroop, Revanth, Chakrapani, Hemanth, Vishnu** under my supervision. The work is genuine, original, and suitable for submission to the SRM University – AP for the award of Bachelor of Technology/Master of Technology in **School of Engineering and Sciences.**

Supervisor

(Signature)

Prof. Saleti Sumalatha

Assistant Professor,

SRM University AP.

Acknowledgements

Beyond our collective efforts, the success of our Course Project for Principles of Big Data Management is intricately tied to the guidance and encouragement of many. At this juncture, we wish to express our heartfelt gratitude to the individuals whose invaluable contributions played a pivotal role in achieving the successful completion of this project.

We extend our deepest appreciation to Prof. Saleti Sumalatha. We cannot adequately convey our thanks for his unwavering support and assistance. Every interaction with him leaves us motivated and inspired. Without his guidance and encouragement, our project would not have been possible.

The project's accomplishments owe much to the guidance and support of our mentor. His consistent help and support have been instrumental in our success, and we are truly grateful for the contributions.

Table of Contents

Abstract

Graphs are essential tools for simplifying complex datasets, transforming raw data into visual forms that reveal patterns, trends, and outliers. In the era of Big Data, where the volume and variety of information can be overwhelming, effective visualization is key to extracting actionable insights. This presentation explores the critical components of creating impactful graphs in R, focusing on graphical parameters, symbols, and colors.

Graphical parameters enable customization of axes, scaling, and layout, ensuring clarity and relevance in visualizations. Symbols and lines are instrumental in representing data categories and trends, offering flexibility to distinguish various datasets. Colors enhance both the aesthetics and accessibility of graphs, making them visually appealing and inclusive for diverse audiences.

Using R's robust graphing libraries such as **ggplot2**, this study illustrates the application of these components in Big Data scenarios, from identifying clusters in scatter plots to highlighting anomalies with heatmaps. By mastering these techniques, data professionals can communicate complex insights effectively, catering to the needs of diverse stakeholders. This work underscores the importance of well-designed visualizations in decision-making and offers a foundation for further exploration of advanced visualization methods in R.

Abbrevations

ggplot2: Grammar of Graphics Plotting Version 2

pch: Plotting Character

RGB: Red-Green-Blue (Color model used for digital graphics)

1 Introduction

In today's data-driven world, the ability to extract meaningful insights from large datasets is a critical skill. With the explosion of Big Data, which includes vast amounts of structured and unstructured information, making sense of this data has become increasingly complex. Visualizations serve as a bridge between raw data and actionable insights by transforming numerical or categorical information into graphical representations. These visuals simplify the process of identifying patterns, trends, and outliers, making complex datasets comprehensible to a wider audience.

The Role of Data Visualization in Big Data Contexts

Data visualization is indispensable in Big Data contexts for several reasons:

1. **Simplification**: Large datasets, often with millions of records, are difficult to interpret directly. Graphs and charts provide an overview by summarizing the data into easily digestible formats.
2. **Trend Identification**: Visual tools help uncover patterns that might otherwise go unnoticed. For example, time-series graphs can reveal seasonal trends in sales or customer behavior.
3. **Anomaly Detection**: Scatter plots and heatmaps can highlight outliers or unusual behavior in datasets, crucial for domains like fraud detection or system monitoring.
4. **Decision Support**: By presenting data in a visual manner, stakeholders can make informed decisions quickly, whether it's optimizing a marketing campaign or identifying resource allocation needs.

R: A Preferred Tool for Data Visualization

R is one of the most powerful and flexible tools for data visualization, widely adopted in both academic and industry settings. Its popularity stems from several key features:

1. **Rich Graphing Libraries**: R offers a range of libraries like ggplot2, lattice, and base, enabling users to create anything from simple bar charts to complex, interactive visualizations.
2. **Customizability**: R allows granular control over every aspect of a graph, including colors, symbols, labels, and layouts. This makes it suitable for creating both standard and highly customized visuals.
3. **Integration with Data Analysis**: As a statistical programming language, R seamlessly integrates data cleaning, manipulation, and analysis with visualization, ensuring a streamlined workflow.

4. **Community and Resources**: R boasts a large, active community that contributes packages, tutorials, and best practices, making it accessible even to beginners.

Real-World Applications of Graphs in Big Data

Graphs are employed across various domains to turn complex datasets into actionable insights. Some notable examples include:

1. **Financial Data Trends**: In the finance industry, line charts and candlestick graphs are used to track stock market trends, analyze portfolio performance, and predict future price movements. For instance, a time-series analysis can show the impact of economic events on market indices.
2. **Social Media Analytics**: Heatmaps and scatter plots help analyze user engagement on platforms like Twitter or Instagram. For example, businesses can visualize peak activity times or track how viral a campaign went using network graphs.
3. **Healthcare**: In the medical field, visualizations like boxplots or histograms are used to study patient data, identify risk factors, and monitor the spread of diseases, as seen during the COVID-19 pandemic.
4. **Urban Planning and Transportation**: Geographic visualizations such as density maps help city planners identify traffic congestion points, optimize public transport routes, and improve urban infrastructure.

By enabling users to draw insights from Big Data efficiently, data visualization has become an integral part of modern decision-making processes. R, with its unparalleled capabilities, continues to play a pivotal role in creating impactful visual narratives that drive innovation and problem-solving across industries.

2 Graphical Parameters in R

The visual representation of data in R is highly customizable through the use of graphical parameters. These parameters allow fine control over graph layouts, scaling, axes, and overall appearance, ensuring that visualizations are both clear and informative. In this section, we delve into the par() function, scaling, axis customization, graph layouts, and advanced graphical parameters.

2.1 The par() Function

The par() function in R is a powerful tool that sets or queries graphical parameters. These parameters control various aspects of a graph's appearance, such as margins, layout, colors, and axis labels. Once set, these parameters remain in effect for the current plotting session until explicitly changed.

Key Features of par()

- **Global Control**: Parameters set using par() apply to all subsequent plots in the session, making it easier to maintain consistency.
- **Flexibility**: Users can specify parameters individually or combine multiple settings in a single call.
- **Session Scope**: Changes made using par() persist only within the R session and can be reset using par(opar), where opar is a saved copy of the original parameters.

```r
# Setting margins and background color
opar <- par(mar = c(5, 4, 4, 2) + 0.1, bg = "lightblue")
plot(1:10, main = "Customized Plot", col = "blue", pch = 16)
par(opar)  # Reset to default settings
```

2.2 Scaling and Axes Customization

Scaling and axis adjustments ensure that the plotted data is presented within appropriate and meaningful ranges. R offers robust functionality for customizing axis limits and tick marks.

2.3 Setting Axis Limits (xlim, ylim)

The xlim and ylim parameters define the minimum and maximum values for the x-axis and y-axis, respectively. This is useful when you want to focus on a specific range of data or ensure consistency across multiple plots.

Example:

```r
# Scatter plot with customized axis limits
x <- rnorm(100)
y <- rnorm(100)
plot(x, y, xlim = c(-3, 3), ylim = c(-3, 3), main = "Customized Axis Limits")
```

2.4 Adjusting Tick Marks (axis())

The axis() function provides detailed control over axis labels, positions, and intervals. You can customize the placement and appearance of ticks for better readability.

Example:

```r
# Customizing tick marks
plot(1:10, xaxt = "n", yaxt = "n", main = "Custom Tick Marks")
axis(1, at = 1:10, labels = paste("X", 1:10))
axis(2, at = seq(0, 10, by = 2), las = 1)
```

2.5 Graph Layouts

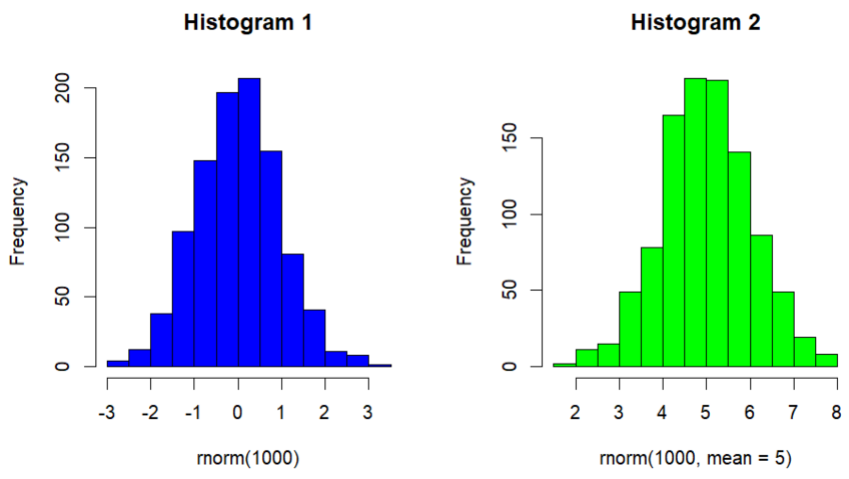
R enables the creation of multi-panel plots using the mfrow and mfcol parameters in par(). These parameters split the plotting area into a grid for displaying multiple graphs side by side or stacked vertically.

Using mfrow and mfcol

- **mfrow**: Arranges plots row-wise.
- **mfcol**: Arranges plots column-wise.

Example: Side-by-Side Histograms

```r
# Comparing two distributions using mfrow
opar <- par(mfrow = c(1, 2))  # 1 row, 2 columns
hist(rnorm(1000), main = "Histogram 1", col = "blue")
hist(rnorm(1000, mean = 5), main = "Histogram 2", col = "green")
par(opar)  # Reset layout
```



2.6 Advanced Parameters

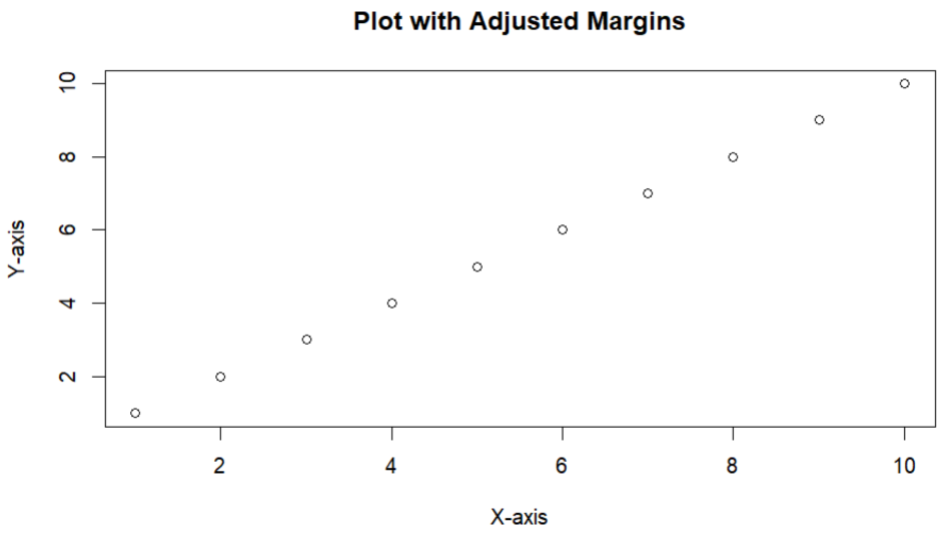
Advanced graphical parameters provide granular control over graph aesthetics, such as margins, spacing, and aspect ratios.

2.7 Margins (mar) and Spacing

The mar parameter adjusts the size of the margins surrounding the plotting area. It is specified as a numeric vector of length four, representing the bottom, left, top, and right margins.

Example:

```
# Adjusting margins
opar <- par(mar = c(5, 5, 4, 2))  # Increase left margin
plot(1:10, main = "Plot with Adjusted Margins", xlab = "X-axis", ylab = "Y-axis")
par(opar)  # Reset to default margins
```

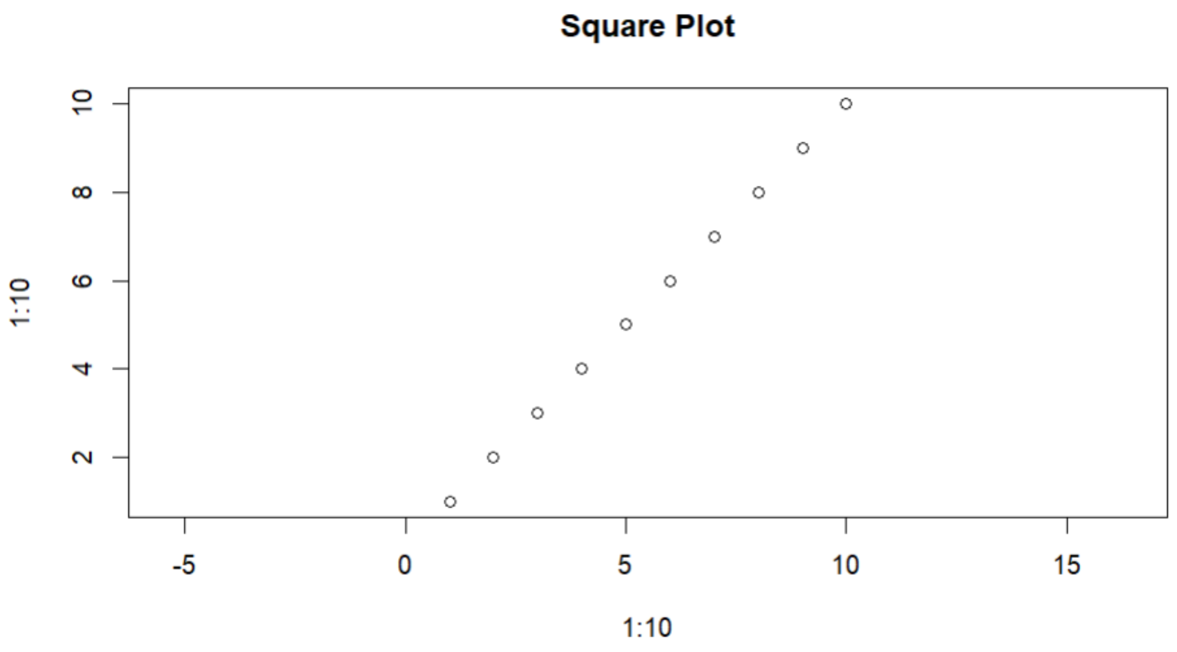


2.8 Aspect Ratio Control (asp)

The asp parameter sets the aspect ratio of the plot, ensuring that one unit on the x-axis is equal to a specified number of units on the y-axis. This is particularly useful for geographic or spatial data.

Example:

```r
# Setting aspect ratio for a square plot
plot(1:10, 1:10, asp = 1, main = "Square Plot")
```



Graphical parameters in R provide comprehensive tools for tailoring visualizations to meet specific needs. Whether it's setting axis limits, arranging multiple plots, or refining margins, these features allow users to create clear and aesthetically pleasing graphs. Mastering these parameters enhances the effectiveness of data communication, making R a preferred choice for visualization tasks.

3 Symbols and Lines

In R, symbols and lines play a vital role in customizing graphs, enhancing their readability, and highlighting specific features of the data. By using parameters such as pch, col, bg, lty, and lwd, you can represent data points and trends in a visually meaningful way. This section explores how symbols and lines are used in R to create effective visualizations.

3.1 Symbols

The pch Parameter

The pch parameter controls the plotting character (symbol) used for points in a graph. R provides 25 pre-defined symbols, ranging from basic shapes like circles and squares to more complex markers. The symbols are numbered from 0 to 25. For symbols 21 to 25, you can customize both the border color (col) and the fill color (bg).

Table of pch Values and Their Symbols

pch Value	**Symbol**	**Description**
0	□	Square
1	○	Circle
2	△	Upward triangle
3	+	Plus
4	×	Cross
5	⍰	Diamond
6	⍰	Downward triangle
21–25	●	Shapes with customizable fill and border colors

Example: Customizing Symbols in a Scatter Plot

```r
# Scatter plot with different symbols
x <- rnorm(50)
y <- rnorm(50)
plot(x, y, pch = 21, col = "blue", bg = "yellow", cex = 1.5,
     main = "Scatter Plot with Custom Symbols")
```

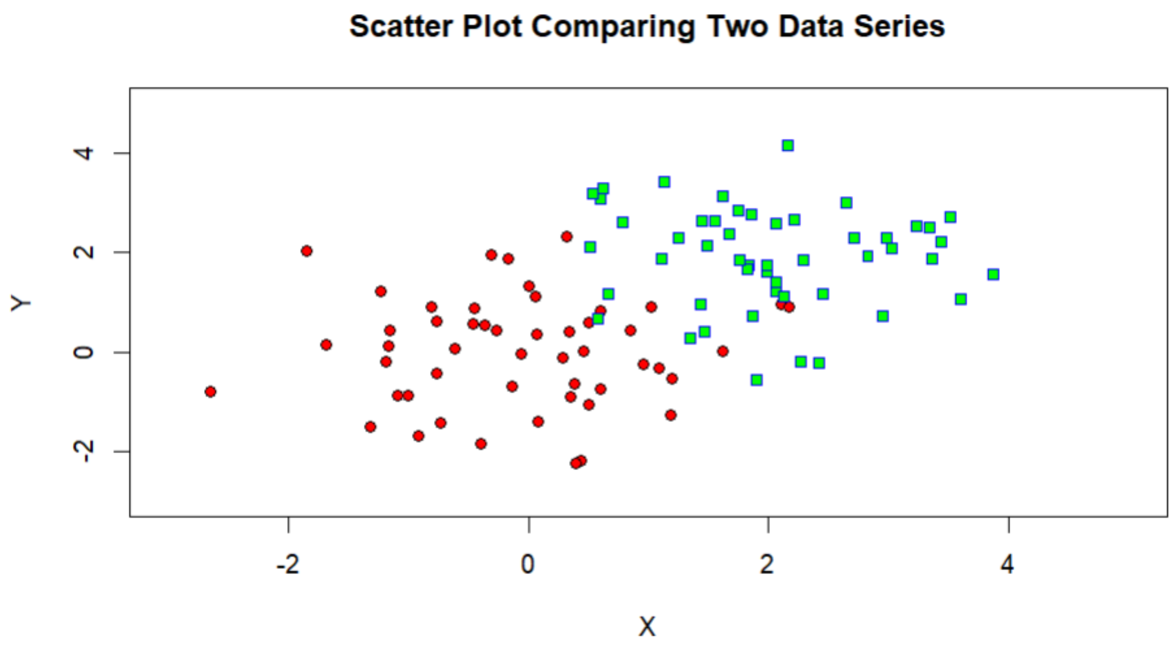
3.2 Using Border (col) and Fill Colors (bg)

Symbols 21 through 25 allow customization of both border (col) and fill (bg) colors. This feature is particularly useful for distinguishing categories within a dataset.

Example: Comparing Two Data Series with Different Shapes

```
# Two data series with distinct symbols and colors
x1 <- rnorm(50)
y1 <- rnorm(50)
x2 <- rnorm(50, mean = 2)
y2 <- rnorm(50, mean = 2)

plot(x1, y1, pch = 21, col = "black", bg = "red", xlim = c(-3, 5), ylim = c(-3, 5),
     main = "Scatter Plot Comparing Two Data Series", xlab = "X", ylab = "Y")
points(x2, y2, pch = 22, col = "blue", bg = "green")
```



3.3 Lines

Line Types (lty)

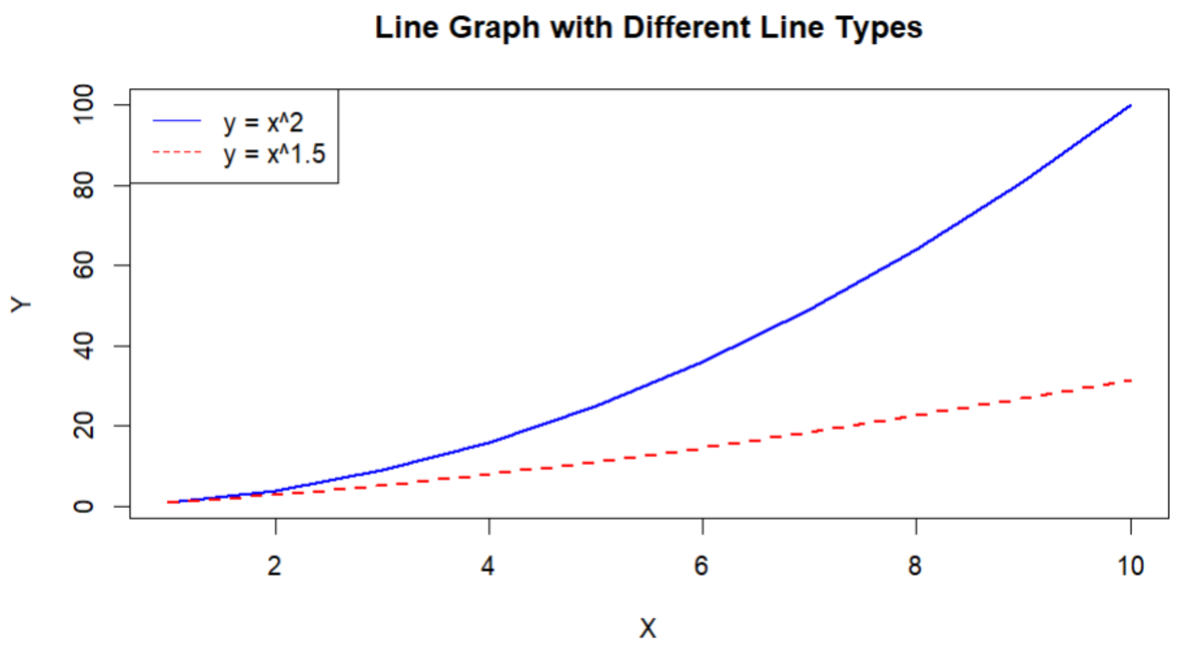
The lty parameter controls the style of lines in plots. R supports several predefined line types, each represented by a number or a string.

lty Value	Line Style	Description
1	—	Solid line
2	--	Dashed line
3	.	Dotted line
4	-.	Dash-dot line
5	Long dash	
6	Two-dash	

Example: Line Graph with Different Line Types

```
# Line graph with multiple line styles
x <- 1:10
y1 <- x^2
y2 <- x^1.5

plot(x, y1, type = "l", lty = 1, col = "blue", lwd = 2,
     main = "Line Graph with Different Line Types", xlab = "X", ylab = "Y")
lines(x, y2, lty = 2, col = "red", lwd = 2)
legend("topleft", legend = c("y = x^2", "y = x^1.5"), lty = c(1, 2), col = c("blue", "red
```



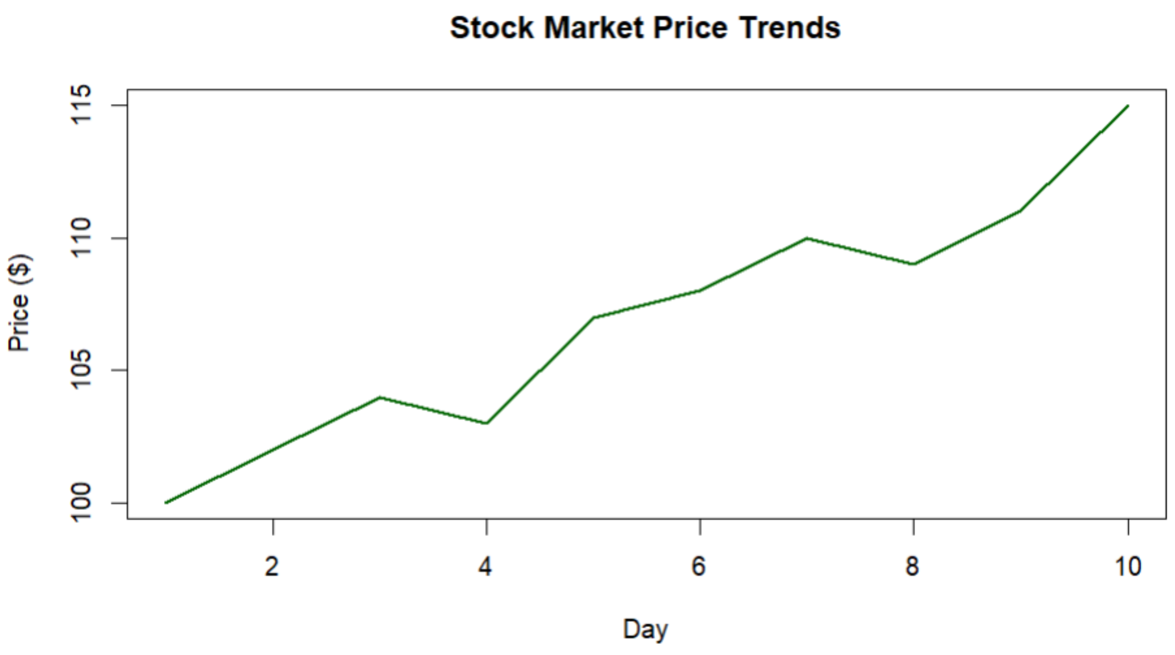
3.4 Line Width (lwd)

The lwd parameter adjusts the thickness of lines, making them more or less prominent. This is particularly useful for emphasizing certain trends.

Example: Emphasizing Trends in a Stock Market Graph

```r
# Stock market trend with emphasized lines
dates <- 1:10
prices <- c(100, 102, 104, 103, 107, 108, 110, 109, 111, 115)

plot(dates, prices, type = "l", col = "darkgreen", lwd = 2,
     main = "Stock Market Price Trends", xlab = "Day", ylab = "Price ($)")
```



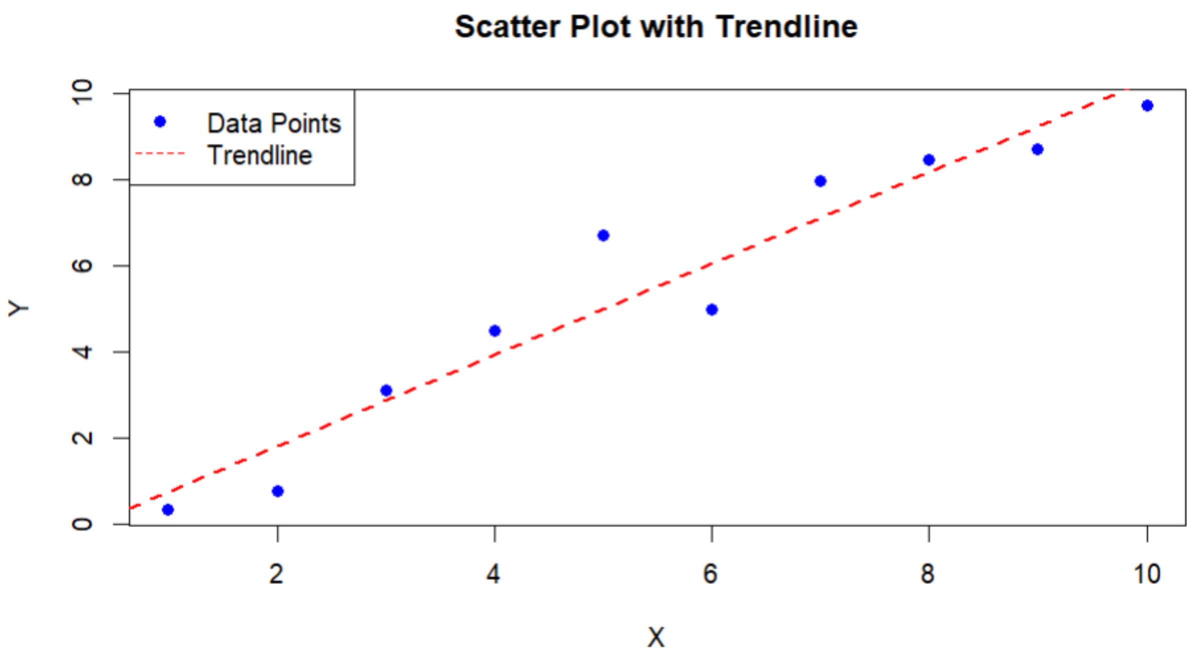
3.5 Integration: Combining Symbols and Lines

Symbols and lines can be combined within the same plot to represent data points and their trends simultaneously. This is especially useful for adding trendlines or reference lines to scatter plots.

Example: Trendline Overlaid on Data Points

```
# Scatter plot with a trendline
x <- 1:10
y <- x + rnorm(10)
fit <- lm(y ~ x)  # Linear regression model

plot(x, y, pch = 16, col = "blue", main = "Scatter Plot with Trendline", xlab = "X", ylab
abline(fit, lty = 2, col = "red", lwd = 2)  # Add trendline
legend("topleft", legend = c("Data Points", "Trendline"), pch = c(16, NA), lty = c(NA, 2)
```



Symbols and lines are fundamental elements of effective data visualization in R. By leveraging the **pch**, **col**, **bg**, **lty**, and **lwd** parameters, users can create graphs that are not only visually appealing but also highly informative. Mastering the integration of these elements enables the creation of detailed and clear visualizations that convey complex data relationships with ease.

4 Colors in Graphs

Colors play a vital role in data visualization, enhancing both the aesthetic appeal and interpretability of graphs. Proper use of colors can highlight critical insights, differentiate data categories, and improve the overall clarity of visualizations. This section covers the theory behind color representation, strategies for choosing effective color palettes, accessibility considerations, and case studies demonstrating the practical use of colors in R.

4.1 Color Theory

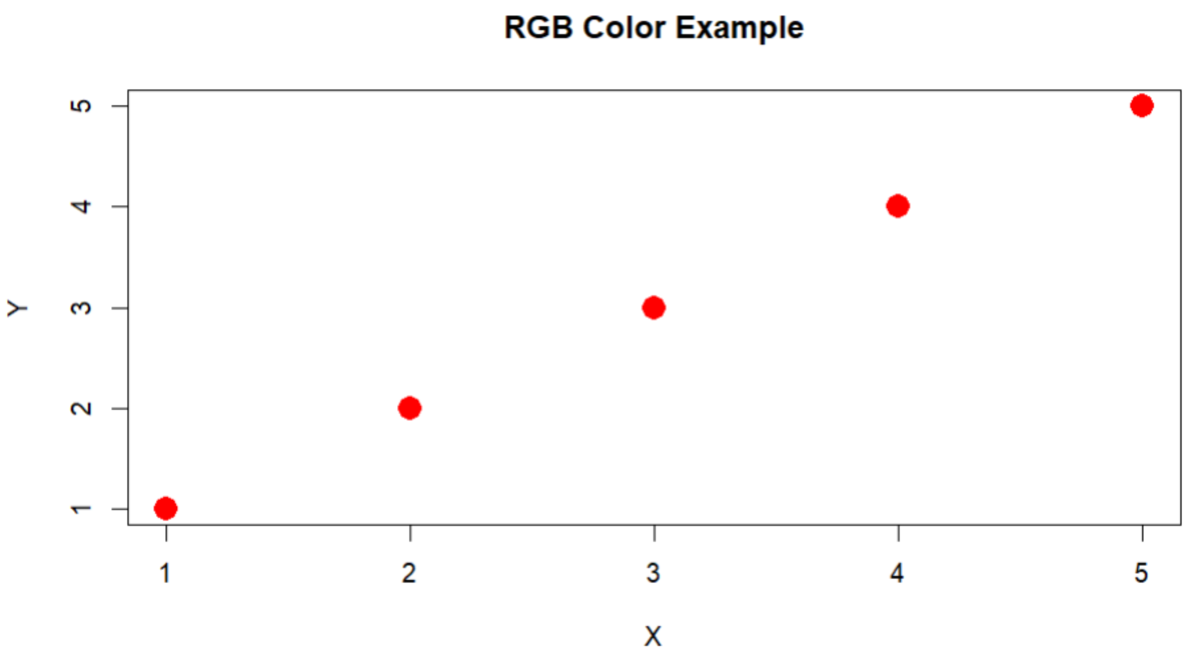
RGB Model

The RGB (Red, Green, Blue) color model is an additive color system where colors are created by combining red, green, and blue light in varying intensities. Each color component is typically represented on a scale from 0 to 255.

- Example: Pure red is represented as (255, 0, 0), while white is (255, 255, 255).
- The rgb() function in R can be used to define colors in this model.

Example:

```r
# Creating custom colors using RGB
plot(1:5, col = rgb(255, 0, 0, max = 255), pch = 16, cex = 2,
     main = "RGB Color Example", xlab = "X", ylab = "Y")
```



4.2 HSV Model

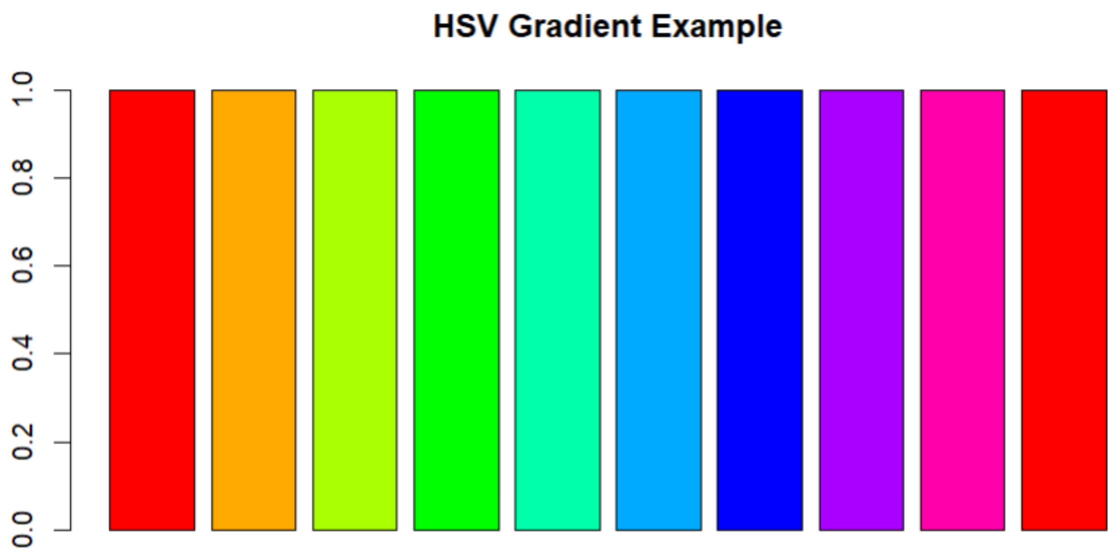
The HSV (Hue, Saturation, Value) model is an alternative color representation that is often more intuitive for visualizing data:

- **Hue**: Defines the base color (e.g., red, green, blue) on a 360° scale.
- **Saturation**: Represents the intensity of the color, ranging from 0 (gray) to 1 (full color).
- **Value**: Indicates the brightness of the color, from 0 (black) to 1 (bright).

R provides the hsv() function to define colors in this model.

Example:

```r
# Creating a gradient using HSV
barplot(rep(1, 10), col = hsv(seq(0, 1, length = 10), 1, 1),
        main = "HSV Gradient Example")
```



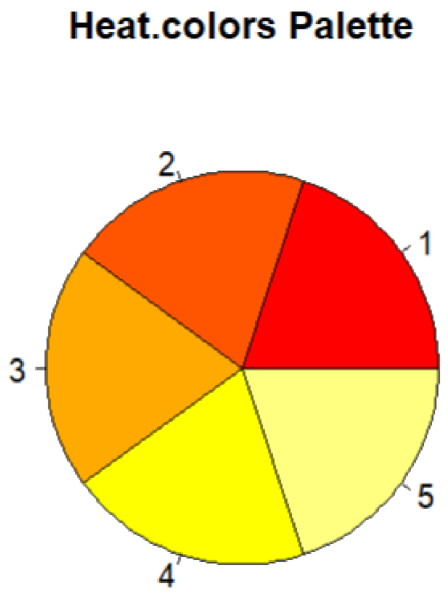
4.3 Choosing Palettes for Clarity

Predefined color palettes in R make it easier to assign colors to data points consistently:

- **rainbow()**: Produces a spectrum of colors. Ideal for displaying continuous data but may overwhelm when overused.
- **heat.colors()**: Ranges from red to yellow. Useful for heatmaps and density plots.

Example:

```
# Using rainbow and heat.colors palettes
pie(rep(1, 5), col = rainbow(5), main = "Rainbow Palette")
pie(rep(1, 5), col = heat.colors(5), main = "Heat.colors Palette")
```



4.4 Accessibility

Color-Blind Friendly Palettes

Approximately 8% of the male population and 0.5% of the female population have some form of color blindness. To ensure inclusivity, it's essential to use palettes designed for color-blind accessibility, such as viridis. This palette is perceptually uniform and suitable for grayscale conversion.

Example: Recoloring a Heatmap for Inclusivity

```
# Heatmap with a color-blind friendly palette
library(viridis)
matrix_data <- matrix(runif(100), nrow = 10)
heatmap(matrix_data, col = viridis(10), main = "Heatmap with Viridis Palette")
```

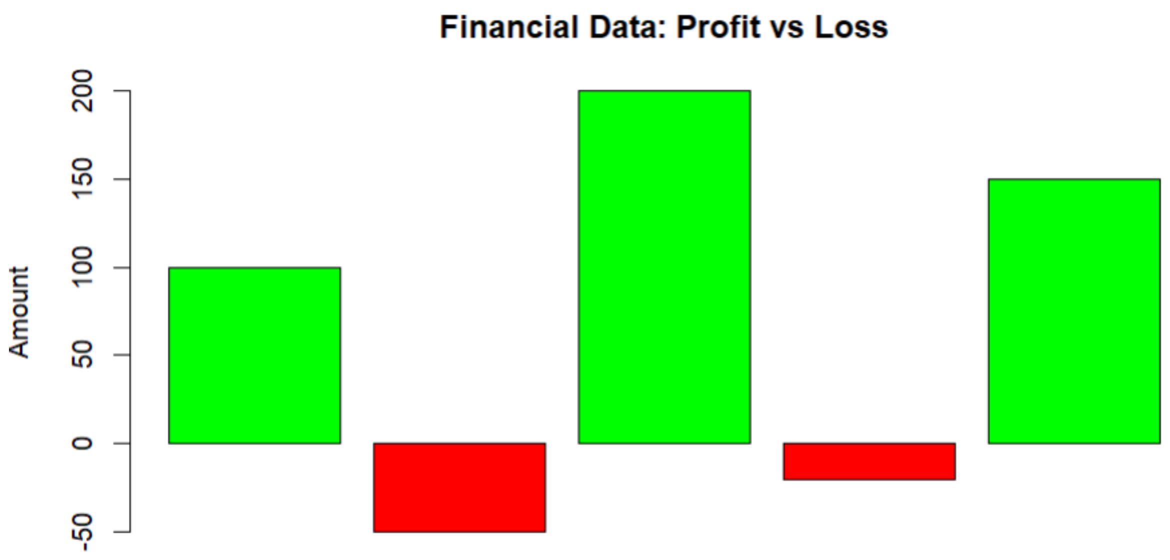
4.5 Case Studies

Highlighting Anomalies in Financial Data

Contrast is crucial when highlighting critical data points or anomalies. For instance, in a financial dataset, using red for losses and green for profits can quickly draw attention to trends or outliers.

Example:

```r
# Highlighting profit and loss
profit_loss <- c(100, -50, 200, -20, 150)
barplot(profit_loss, col = ifelse(profit_loss > 0, "green", "red"),
        main = "Financial Data: Profit vs Loss", ylab = "Amount")
```

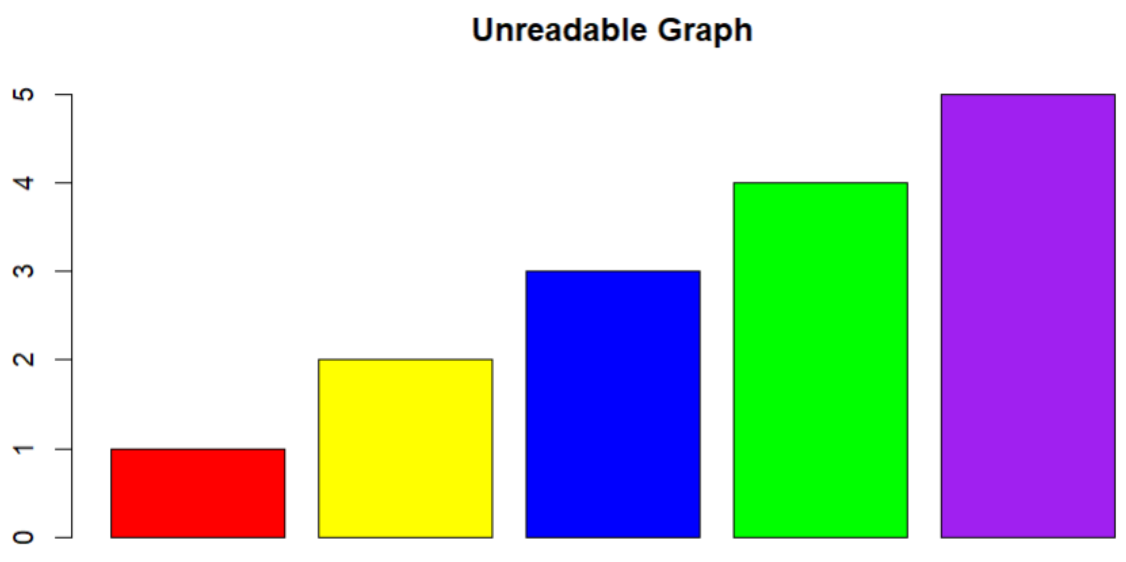


4.6 Improving an Unreadable Graph with a Better Palette

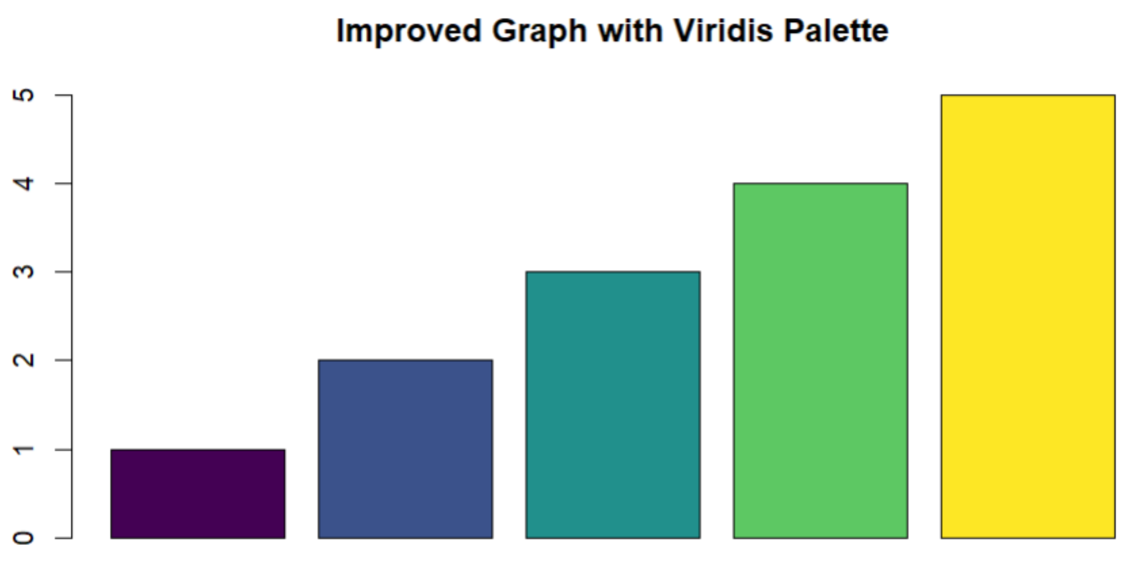
Overusing bright or clashing colors can make a graph difficult to interpret. Using a more subdued palette improves clarity and prevents viewer fatigue.

Before:

```r
# Unreadable graph with clashing colors
barplot(1:5, col = c("red", "yellow", "blue", "green", "purple"),
        main = "Unreadable Graph")
```



After:



The effective use of colors in graphs enhances their interpretability and impact. By understanding color models like RGB and HSV, selecting appropriate palettes for clarity, and ensuring accessibility through color-blind friendly schemes, data visualizations can be made both visually appealing and inclusive. Through practical applications, such as highlighting anomalies or improving poorly designed graphs, colors become a powerful tool for communication in data visualization.

5 Practical Examples and Exercises

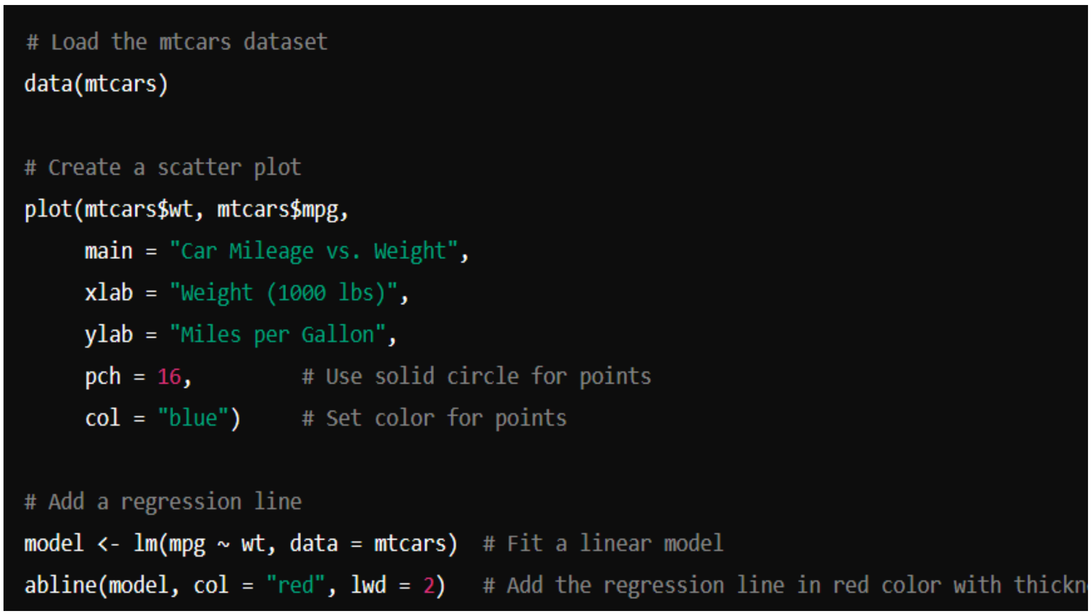
Exercise 1: Plot a Scatter Plot for a Dataset (Car Mileage vs. Weight)

Objective:

- Create a scatter plot comparing car mileage (mpg) and weight (wt).
- Add a regression line to model the relationship between the two variables.
- Customize colors for points and the regression line.

Step-by-Step Code:

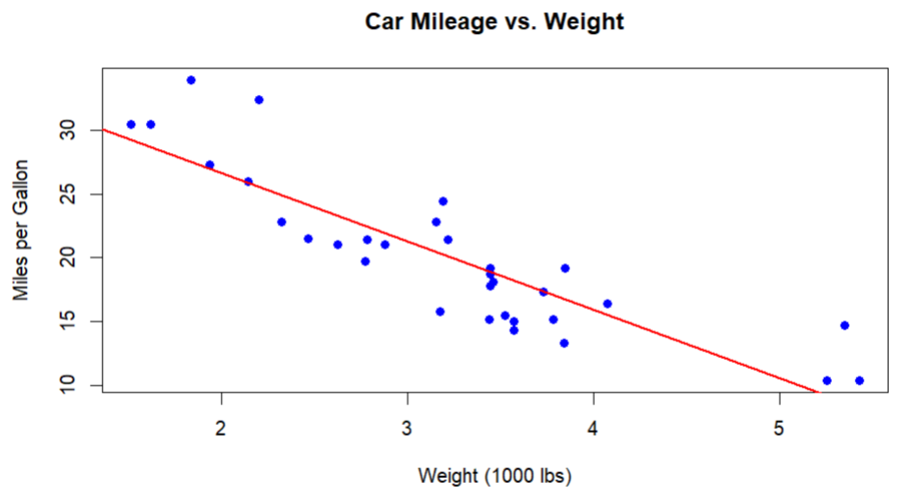
1. **Load Data**: Use the mtcars dataset available in R for this example.
2. **Create the Scatter Plot**: Plot mpg (miles per gallon) against wt (weight).
3. **Add a Regression Line**: Use abline() to add a linear regression line.
4. **Customize Colors**: Set custom colors for the points and the regression line.



```r
# Load the mtcars dataset
data(mtcars)

# Create a scatter plot
plot(mtcars$wt, mtcars$mpg,
     main = "Car Mileage vs. Weight",
     xlab = "Weight (1000 lbs)",
     ylab = "Miles per Gallon",
     pch = 16,          # Use solid circle for points
     col = "blue")      # Set color for points

# Add a regression line
model <- lm(mpg ~ wt, data = mtcars)  # Fit a linear model
abline(model, col = "red", lwd = 2)   # Add the regression line in red color with thickn
```



Exercise 2: Create a Multi-Panel Plot to Compare Income Distribution Across Different Regions

Objective:

- Create a multi-panel plot to compare income distribution across three regions.
- Use a boxplot to visualize the distribution of income.

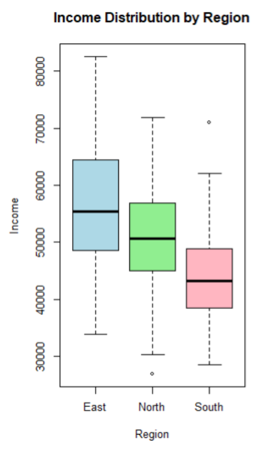
Step-by-Step Code:

1. **Simulate Data**: Create a data frame with income data for three regions.
2. **Create the Multi-Panel Plot**: Use par(mfrow = c()) to arrange multiple plots in one window.
3. **Create Boxplots**: Use boxplot() to show income distribution.

```r
# Simulate income data for three regions
set.seed(123)  # Set seed for reproducibility
region_data <- data.frame(
  Region = rep(c("North", "South", "East"), each = 100),
  Income = c(rnorm(100, mean = 50000, sd = 10000),  # North
             rnorm(100, mean = 45000, sd = 8000),   # South
             rnorm(100, mean = 55000, sd = 12000))   # East
)

# Set up a multi-panel plot layout (1 row, 3 columns)
par(mfrow = c(1, 3))  # Arrange 3 plots in a single row

# Create boxplots for income distribution by region
boxplot(Income ~ Region, data = region_data,
        main = "Income Distribution by Region",
        xlab = "Region", ylab = "Income", col = c("lightblue", "lightgreen", "lightpink"))
```



6 Conclusion

In this report, we explored the essential techniques for effective data visualization using R, focusing on graphical parameters, symbols, lines, and colors. R provides powerful tools, such as the **par()** function, for customizing plot appearance, from adjusting axis limits to modifying layout and margins. These features are crucial for presenting complex data in a clear, organized manner, ensuring that key insights are communicated effectively. By customizing symbols and lines, users can highlight specific data points or trends, making visualizations more informative and engaging. Symbols, controlled by the **pch** parameter, and lines, adjusted by **lty** and **lwd**, provide a flexible way to distinguish between data categories and emphasize relationships between variables. Additionally, color plays a pivotal role in enhancing the clarity and aesthetics of plots, with R's built-in color palettes and the ability to define custom colors using the **RGB** and **HSV** models.

The ability to create color-blind friendly visualizations, such as using the **viridis** palette, ensures accessibility, making visualizations more inclusive. Practical examples, like scatter plots with regression lines, multi-panel plots comparing distributions, and heatmaps visualizing data density, demonstrated the versatility of R in handling different data types. These visualizations provide a deeper understanding of data patterns, trends, and anomalies. Moving forward, integrating interactive features and advanced tools like **Shiny** will further enhance the power of data visualization in R, enabling real-time analysis and dynamic reporting. Mastering these techniques ensures that data insights are presented in an impactful, clear, and accessible manner, essential for informed decision-making in today's data-driven world.