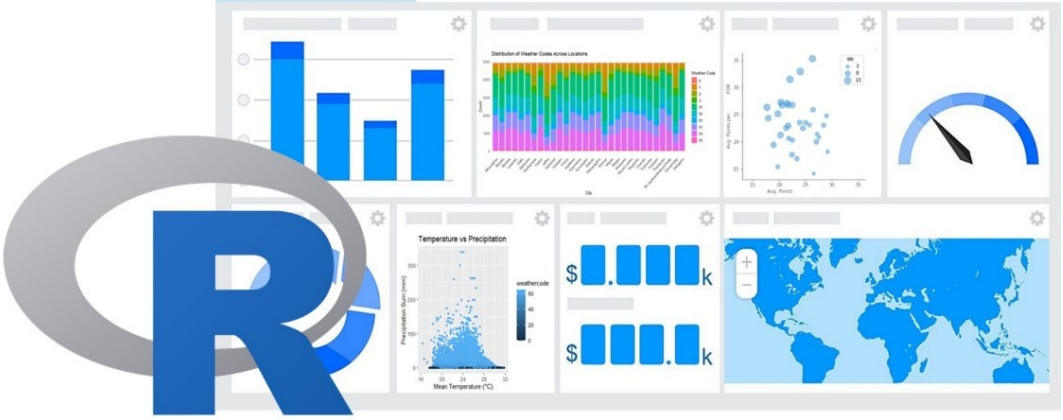
Teaching & Learning
Step-by-Step Guide:

Introduction to Data Visualization with R



Amila Jayasinghe

Niroshan Sanjaya

Farasath Hasan

Samith Madusanka

Teaching & Learning Step by - Step Guide:

Introduction to Data Visualization with R

Authors

Amila Jayasinghe

Niroshan Sanjaya

Farasath Hasan

Samith Madusanka

Publisher

University of Moratuwa

Author contribution

1. Amila Jayasinghe, (Supervision, Methodology), Department of Town & Country Planning, University of Moratuwa, Sri Lanka.

2. Niroshan Sanajaya, (Methodology, visualization), Department of Town & Country Planning, University of Moratuwa, Sri Lanka.

3. Farasath Hasan, (Writing—original draft preparation, Validation), Department of Town & Country Planning, University of Moratuwa, Sri Lanka.

4. Samith Madusanka, (Project Administration, Review and Editing), Department of Town & Country Planning, University of Moratuwa, Sri Lanka.

All authors have read and agreed to the published version of the book.

Contact authors amilabj@uom.lk

This book was produced with the valuable support of the Erasmus+ Capacity Building in Higher Education (CBHE) project 'Curricula Enrichment for Sri Lankan Universities delivered through the application of Location-Based Services to Intelligent Transport Systems' (LBS2ITS https://lbs2its.net/)

Project Number: 618657-EPP-1-2020-1-AT-EPPKA2-CBHE-JP

Programme: Erasmus+

Key Action: Cooperation for innovation and the exchange of good practices

Action Type: Capacity Building in Higher Education

Co-funding: Erasmus+ Programme of the European Union

This book was reviewed as an Open Education Resource for University students by Dr Rico Wittwer (Technische Universität Dresden — TU Dresden, Germany) under the LBS2ITS project

lbs2its.net

LBS2ITS

Curricula Enrichment delivered through

the Application of Location-based Services

to Intelligent Transport Systems

Co-funded by the Erasmus+ Programme of the European Union

Edition

First Edition - May 2025

ISBN 978-624-6745-00-4 (ebook)

Citation

Jayasinghe, A., Sanjaya, N., Hasan, F., & Madusanka, S. (2025). *Teaching & learning step-by-step guide—Introduction to data visualization with R* (1st ed.). University of Moratuwa.

Publisher

University of Moratuwa

PREFACE

This book serves as open educational resource for both academic programs and professional development, delivering a detailed roadmap for analyzing urban data using R programming. Tailored to bridge the gap between theory and practice, this book is meticulously designed to cater to the needs of students, educators, and practitioners alike.

In this book, readers will discover comprehensive guidance on leveraging data visualization with R to unlock insights into urban planning. The book provides step-by-step instructions on harnessing R's capabilities to analyze urban data, create insightful visualizations, and interpret findings to inform decision-making in urban development projects. By combining R's powerful statistical tools with its rich visualization libraries, users gain a deep understanding of complex urban datasets and can effectively communicate findings to stakeholders.

The book is developed based on the knowledge acquired from the Train the Teacher (TTT) Course on "Data and Models in Transportation," held at TU Dresden from Monday, 7th to Friday, 11th November 2022. This interactive course, which included "Data in Transport Planning – Introduction to Data Analysis and Visualization With R," provided the foundation upon which this guide was developed. The knowledge gained from the TTT course has been seamlessly integrated into the book, ensuring that it reflects cutting-edge methodologies and pedagogical approaches for teaching R in the context of transportation and urban data analysis.

5-day Train-the-Teachers Course on "Data and models in transportation", November 2022 @ TUD

This book not only serves as a resource for academic learning by offering practical applications and case studies but also empowers industry professionals to conduct sophisticated spatial analysis and contribute meaningful insights to their respective fields. Whether you're a student seeking to master data visualization in urban planning, an educator in search of robust teaching materials, or a practitioner aiming to enhance your technical skills, this guide provides invaluable support. It ensures that users at all levels can proficiently utilize R to explore and address urban challenges, ultimately fostering evidence-based decision-making and transformative urban interventions.

TABLE OF CONTENTS

TABLE OF CONTENTS vi
LIST OF FIGURES vii
1. INTRODUCTION TO DATA VISUALIZATION WITH R 1
2. CHART TYPES IN R 2
3. WORD CLOUD CHART 4
4. CHOROPLETH CHART MAPPING 6
5. LOLLIPOP CHART 9
6. BAR CHART 11
7. CHOROPLETH CHART MAPPING FOR HIGHEST LAND USE 13
8. BUBBLE CHART 15
9. BASIC RIDGELINE PLOT 17
10. TIME SERIES LINE PLOT 18
11. SCATTER PLOT 19
12. VIOLIN PLOT 20
13. HEATMAPS 21
14. STACKED BAR 22

LIST OF FIGURES

Figure 1 - Creating a new R Project in R studio 2
Figure 2 - Example chart type 2
Figure 3 - Code example 5
Figure 4 - Code example 8
Figure 5 - Generated Output Map 8
Figure 6 - Lollipop Chart 10
Figure 7 - Coding Screen shots 12
Figure 8 - Choropleth Chart mapping output 14
Figure 9 - Bubble Chart 16
Figure 10 - Basic ridgeline plot 17
Figure 11 - Time series plot 18
Figure 12 - Scatter Plot 19
Figure 13 - Violin plot 20
Figure 14 - Heat Maps 21
Figure 15 - Stacked bar 23

1. INTRODUCTION TO DATA VISUALIZATION WITH R

Required: R Studio and R programming language

In today's world, where vast amounts of data are generated every second, the ability to effectively visualize and interpret data is a crucial skill for professionals in various fields, including urban planning. R, a powerful open-source programming language and environment for statistical computing and graphics, offers a wide range of tools and packages for creating informative and visually appealing graphs and plots.

This book is designed to provide you with a comprehensive guide to creating various types of graphs and visualizations using R, specifically tailored to the field of urban planning. Whether you're a beginner looking to learn the basics of data visualization or an experienced R user seeking to expand your repertoire of graphing techniques, this book will walk you through the process step by step, from importing data to customizing plot aesthetics.

In this book, you will learn how to:

- Import and preprocess urban planning data from different sources, including CSV files, Excel spreadsheets, and spatial data formats such as shapefiles.
- Explore and visualize spatial patterns and trends in urban data using maps, choropleth maps, and spatial graphs.
- Analyze relationships between different urban characteristics using scatter plots, bar charts, and correlation matrices.
- Create time series plots to visualize temporal trends in urban data over time.
- Customize plot aesthetics, including colors, labels, titles, and annotations, to enhance the clarity and visual appeal of your graphs.
- Generate interactive and dynamic visualizations using R packages such as ggplot2, plotly, and leaflet.

Whether you're conducting research, making policy recommendations, or communicating findings to stakeholders, effective data visualization is essential for conveying insights and driving informed decision-making in urban planning. By mastering the techniques outlined in this book, you'll be equipped with the skills to create compelling and insightful graphs that unlock the hidden stories within your urban data.

2. CHART TYPES IN R

In the world of data visualization, charts are powerful tools for presenting data in a clear and meaningful way. R, a versatile programming language and environment for statistical computing, offers a diverse range of chart types to suit different data types and analysis goals. Let's explore some of the interactive R chart types used in Urban Planning. R provides a diverse range of pre-existing functions and packages for generating different kinds of graphs. Graphs are an effective instrument for visualizing data, making it easier to understand complicated patterns, trends, and correlations. Graphs are fundamental mathematical constructs used to depict connections between entities. Graphs are composed of vertices, also known as nodes that represent the items, and edges, which reflect the connections or interactions between these things. Within the field of graph theory, there are several kinds of graphs that possess unique attributes and serve specific purposes.

Another crucial difference exists between simple and non-simple graphs. Basic graphs do not have self-loops (edges that link a vertex to itself) or multiple edges (more than one edge connecting the same set of vertices). Non-simple graphs do not adhere to these requirements and instead, allow for the inclusion of such characteristics. Although simple graphs are often used in theoretical settings, non-simple graphs have practical uses in situations where many connections or self-referential linkages are significant, such as in social network analysis or transportation networks.

- **Create a New R Project and R Script**

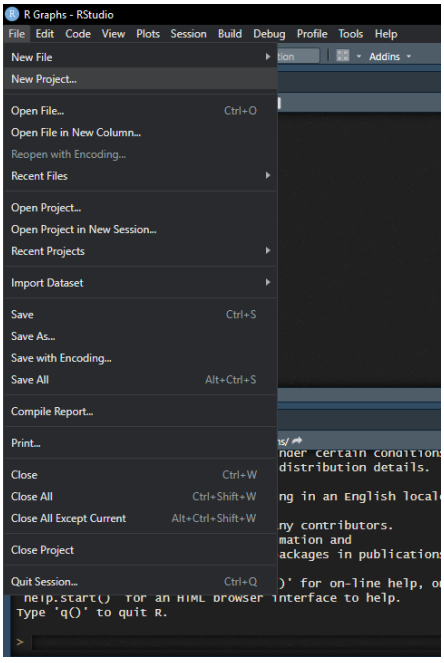


Figure 2 - Creating a new R Project in R studio.

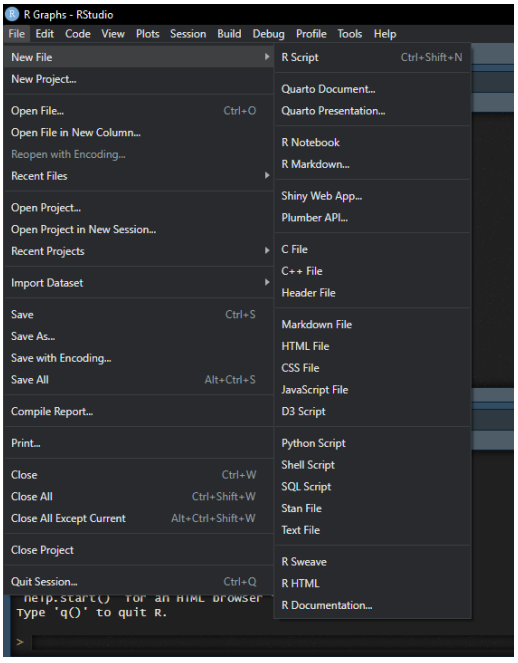
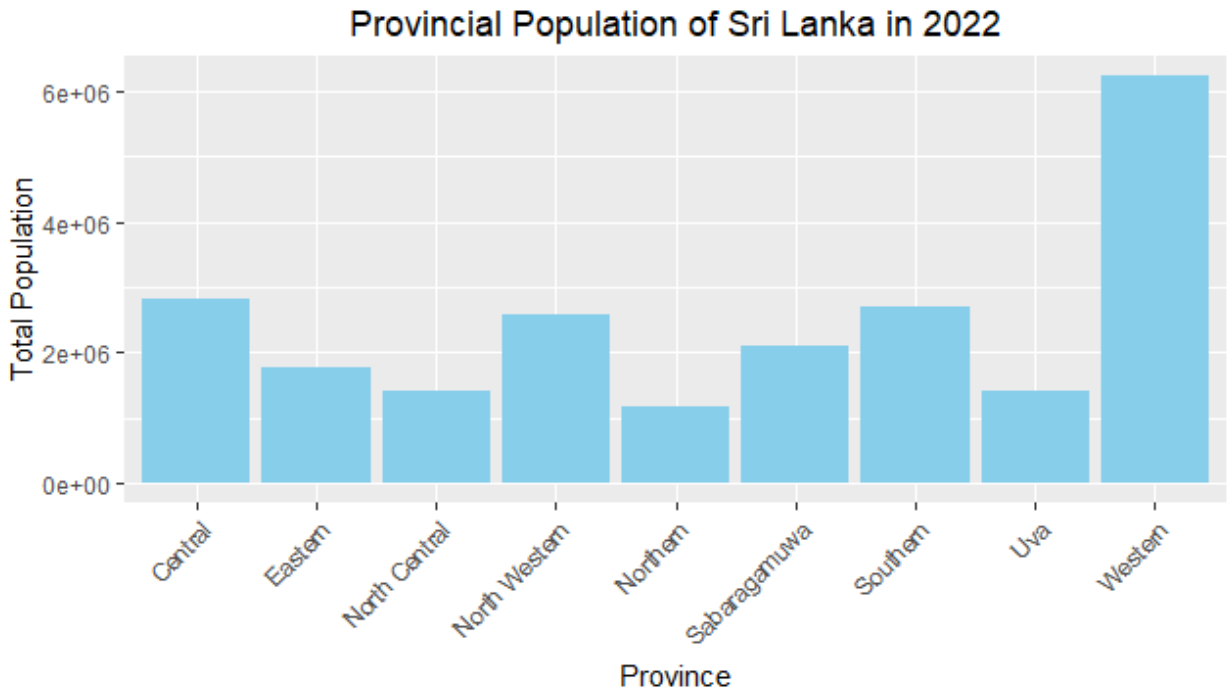


Figure 1 - Example chart type



Editable Code (As per the purpose of the study, user need to change the code)

Fixed\Common Code (User can use this code as it is)

3. WORD CLOUD CHART

Word cloud to represent the highest population provinces in Sri Lanka can be an engaging way to visualize the data.

Code:

```
install.packages("wordcloud")

# Load required packages

library(wordcloud)

#load Data                                   Change data File Path

data <- read.csv("C:\\Users\\User\\Downloads\\R graph\\Population CSV File.csv")

# Sort the data by population in descending order

data <- data[order(-data$Total), ]

# Create a word cloud

wordcloud(words = data$Province, freq = data$Total,

        min.freq = 1, random.order = FALSE,
        colors = brewer.pal(8, "Dark2"),
        scale=c(3,0.5),
        max.words=50,
        random.color=TRUE,
        rot.per=0.35,
        use.r.layout=FALSE,
        fixed.asp=TRUE)
# Export the word cloud as a PNG image

png("wordcloud.png", width = 800, height = 600)

wordcloud(words = data$Province, freq = data$Total,

        min.freq = 1, random.order = FALSE,

        colors = brewer.pal(8, "Dark2"),

        scale=c(3,0.5),

        max.words=50,

        random.color=TRUE,

        rot.per=0.35,

        use.r.layout=FALSE,

        fixed.asp=TRUE,

        ...)

dev.off()
```

Change these values with your column names

```
# Sort the data by population in descending order
data <- data[order(-data$Total), ]

# Create a word cloud
wordcloud(words = data$Province, freq = data$Total,
          min.freq = 1, random.order = FALSE,
          colors = brewer.pal(8, "Dark2"),
          scale=c(3,0.5),
          max.words=50,
          random.color=TRUE,
          rot.per=0.35,
```

Figure 3 - Code example

Output:

- **Data Import:** The code uses the read.csv() function to import the population data from a CSV file.
- **Data Sorting:** The population data is sorted in descending order based on the total population of each province using the order() function.
- **Word Cloud Creation:** The word cloud () function from the word cloud package is used to generate the word cloud. The words in the word cloud are the names of provinces, and the size of each word is proportional to the population of the corresponding province.
- **Parameters:** Several parameters are passed to the word cloud () function to customize the appearance of the word cloud. These include min. Freq, and random. Order, colors, scale, max. Words, random.color, rot. per, use.r.layout, and fixed. asp.
- **Exporting:** The resulting word cloud is exported as a PNG image using the PNG () function, specifying the width and height of the image. The dev. off() function is then used to close the PNG device.

4. CHOROPLETH CHART MAPPING

Visualize land use patterns across a geographic area using different colors or shades to represent different land use types. You can use the ggplot2 package for this purpose.

Code:

```
# Load required libraries
library(sf)
library(ggplot2)
# Load land use shapefile
land_use <- st_read("C:\\Users\\User\\Downloads\\R graph\\data\\Landuse_colombo.shp")
# Display unique land use categories
unique_categories <- unique(land_use$DESCRIPTIO)
print(unique_categories)
```

Input your shape file path

Please Enter double backslashes (\) between folder paths

Add your graph Category field column name and then run the process.

You can see your category names.

```
> print(unique_categories)
 [1] "Built up Area"         "Canals (wide)"       "Garden"
 [4] "Grassland"             "Lake"                "Main Road (A)"
 [7] "Marsh"                 "Minor Road"          "Paddy"
[10] "River"                 "Sand or Beach"       "Scrub"
[13] "Transport & Utilities" "Water Hole"
>
```

Code:

```
ggplot() +
  geom_sf(data = land_use, aes(fill = DESCRIPTIO)) +
  scale_fill_manual(values = c("Built up Area" = "#CC0000", "River" = "#3333FF", "Garden" = "#00FF00","Main Road (A)" ="#330000","Canals (wide)"="#66B2FF",
                               "Grassland"="#4C9900","Lake" ="#0080FF"  ,
                               "Marsh"  ="#FFFF00"       ,  "Minor Road"="#990000"    ,     "Paddy"   ="#66CC00"
                               , "Sand or Beach"="#FF8000"    ,    "Scrub"  ="#CCFF99"
                               ,"Transport & Utilities"="#FF00FF", "Water Hole"="#99CCFF")) + # Add custom colors
  labs(title = "Land Use Choropleth Map",
       caption = "Source: COM_TRANS_DATA",
       fill = "Land Use Category",
       x = "Longitude",
       y = "Latitude") + # Add axis labels
  theme_minimal() +
  theme(plot.title = element_text(size = 10, face = "bold", margin = margin(b = 10)), # Increase bottom margin of the title
        plot.caption = element_text(size = 10),
        axis.text.x = element_text(angle = 45, hjust = 1, margin = margin(t = 10)), # Rotate x-axis labels and adjust top margin
        axis.text.y = element_text(angle = 0, hjust = 1, margin = margin(r = 10)), # Adjust right margin of y-axis labels
        legend.title = element_text(margin = margin(b = 10)), # Increase bottom margin of the legend title
        legend.spacing.y = unit(0.5, "cm")) + # Increase vertical spacing between legend items
  coord_sf(expand = TRUE) # Expand plot area
```

You need to change the Land use categories and color according to your preference. The below site will provide color codes.

https://www.rapidtables.com/web/color/RGB_Color.html

```r
# Display unique land use categories
unique_categories <- unique(land_use$DESCRIPTIO)
print(unique_categories)

ggplot() +
  geom_sf(data = land_use, aes(fill = DESCRIPTIO)) +
  scale_fill_manual(values = c("Built up Area" = "#CC0000", "River" = "#3333FF", "Garden" = "#00FF00","Main Road (A)" ="#330000","Canals (wide)"
                               "Grassland"="#4C9900","Lake" ="#0080FF" ,
                               "Marsh"    ="#FFFF00"      ,     "Minor Road"="#990000"      ,       "Paddy"     ="#66CC00"
                               , "Sand or Beach"="#FF8000"      ,     "Scrub"   ="#CCFF99"
                               ,"Transport & Utilities"="#FF00FF", "Water Hole"="#99CCFF")) + # Add custom colors
  labs(title = "Land Use Choropleth Map",
       caption = "Source: COM_TRANS_DATA",
       fill = "Land Use Category",
       x = "Longitude",
       y = "Latitude") + # Add axis labels
  theme_minimal() +
  theme(plot.title = element_text(size = 10, face = "bold", margin = margin(b = 10)), # Increase bottom margin of the title
        plot.caption = element_text(size = 10),
        axis.text.x = element_text(angle = 45, hjust = 1, margin = margin(t = 10)), # Rotate x-axis labels and adjust top margin
        axis.text.y = element_text(angle = 0, hjust = 1, margin = margin(r = 10)), # Adjust right margin of y-axis labels
        legend.title = element_text(margin = margin(b = 10)), # Increase bottom margin of the legend title
        legend.spacing.y = unit(0.5, "cm")) + # Increase vertical spacing between legend items
  coord_sf(expand = TRUE) # Expand plot area
```

Figure 4 - Code example

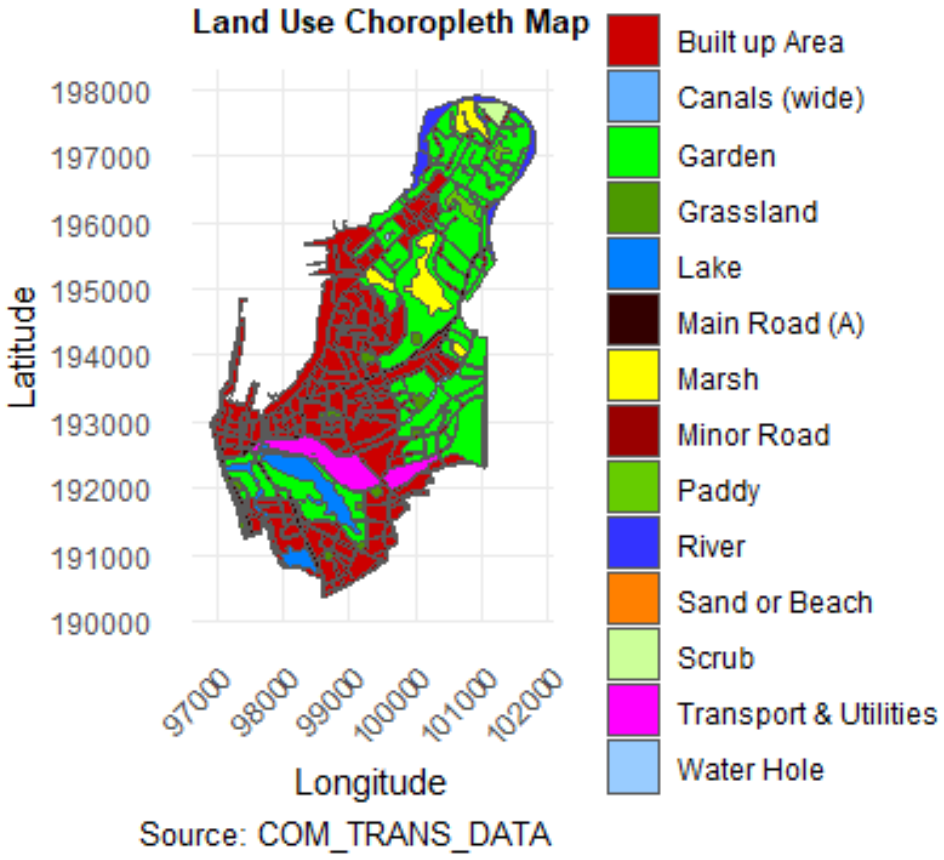


Figure 5 - Generated Output Map

5. LOLLIPOP CHART

A lollipop chart is a variation of a bar chart where the bars are replaced with circles or points, and a line segment (the "stick") connects each circle or point to the horizontal axis. It's often used to display categorical data along with their corresponding values. In the context of land use analysis, a lollipop chart can be used to visualize different land use types and their respective areas, where the circles represent the areas and are connected to the horizontal axis by lines.

Code:

```
# Load necessary libraries
library(tidyverse)
library(sf)
library(ggplot2)
# Read the shapefile
```

Load your shp

```
shapefile <- st_read("C:\\Users\\User\\Downloads\\R graph\\data\\Landuse_colombo.shp")
# View the attribute fields
head(shapefile)
```

Identify shp attribute columns.

```
# Aggregate data by land use type
land_use_summary <- shapefile %>%
  group_by(DESCRIPTIO) %>%
  summarize(total_area = sum(Area))
```

Change land use type and area column head

```
# Sort the data by total area
land_use_summary <- land_use_summary %>%
  arrange(desc(total_area))
# Define a color palette for land use types
color_palette <- c("Built up Area" = "#FF0000", "River" = "#3333FF", "Garden" = "#00FF00","Main Road (A)" ="#330000","Canals (wide)"="#66B2FF",
                   "Grassland"="#4C9900","Lake" ="#0080FF"  ,
                   "Marsh"   ="#FFFF00"        ,    "Minor Road"="#990000"    ,      "Paddy"    ="#66CC00"
                   , "Sand or Beach"="#FF8000"    ,     "Scrub"  ="#CCFF99"
                   ,"Transport & Utilities"="#FF00FF", "Water Hole"="#99CCFF") # Define your own colors
# Plot lollipop chart with custom colors and bubble size based on area
ggplot(land_use_summary, aes(x = reorder(DESCRIPTIO, total_area), y = total_area)) +
  geom_point(aes(color = DESCRIPTIO, size = total_area)) + # Assign different colors and size based on land use types and area
  geom_segment(aes(x = DESCRIPTIO, xend = DESCRIPTIO, y = 0, yend = total_area)) +
```

Customize colors and types according to your land use types

```
coord_flip() +
theme_minimal() +
labs(x = "Land Use Type", y = "Total Area (sqkm)") +
scale_color_manual(values = color_palette) + # Use manual color palette
scale_size_continuous(range = c(1, 10)) # Adjust the range of bubble sizes as needed
```

Output :

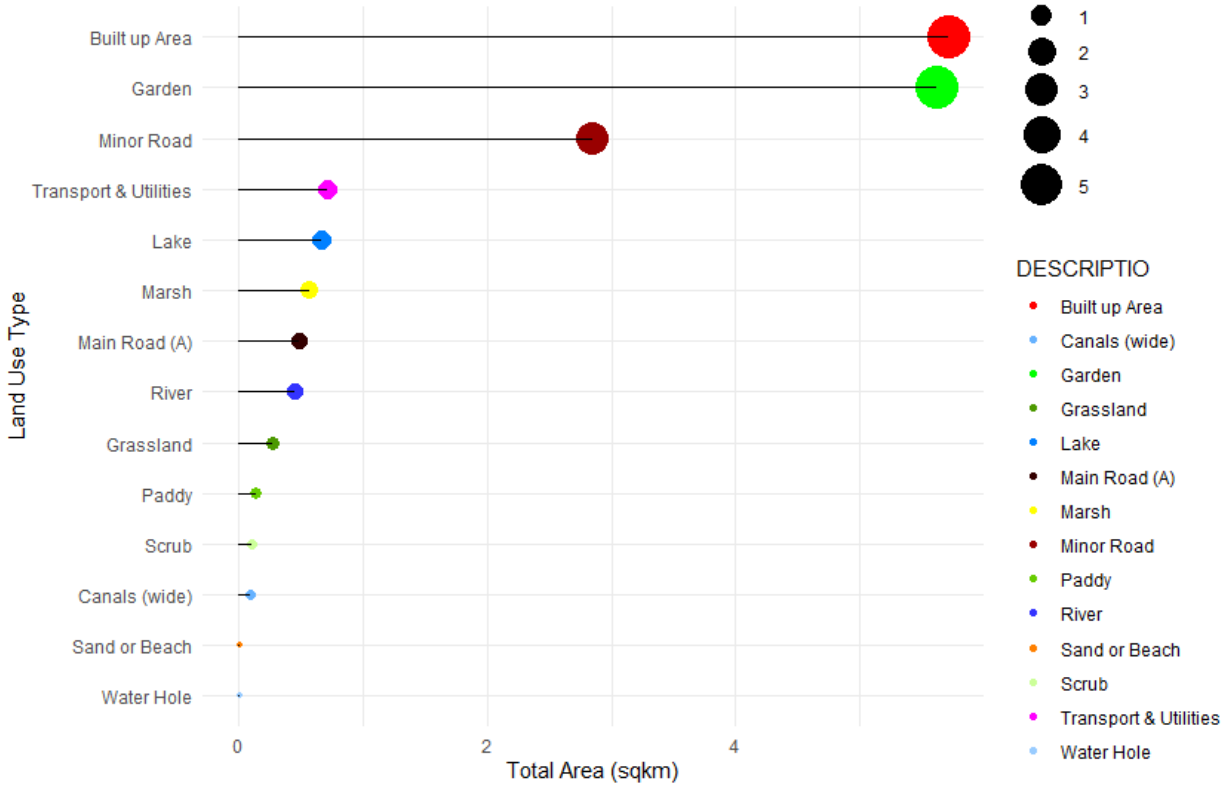


Figure 6 - Lollipop Chart

- **Color Palette:**A custom color palette is defined using hexadecimal color codes for each land use type. This palette is used to assign specific colors to each land use type in the plot.

- **Plotting:**The ggplot() function initializes the plot. The aes() function maps the x-axis to the land use types (reordered by total area) and the y-axis to the total area.

6. BAR CHART

1. Load the Data: First, you'll need to load your CSV file into R. You can use the read.csv() function for this purpose.
2. Prepare the Data: Ensure that your data is in the correct format and that the column names match the ones you provided.
3. Create the Bar Graph: Once your data is loaded and prepared, you can use the ggplot2 package to create the bar graph.

Code :

 # Load the required packages

```
library(ggplot2)
```

 # Load the CSV data

```
data <- read.csv("C:\\Users\\User\\Downloads\\R graph\\Population CSV File.csv")
```

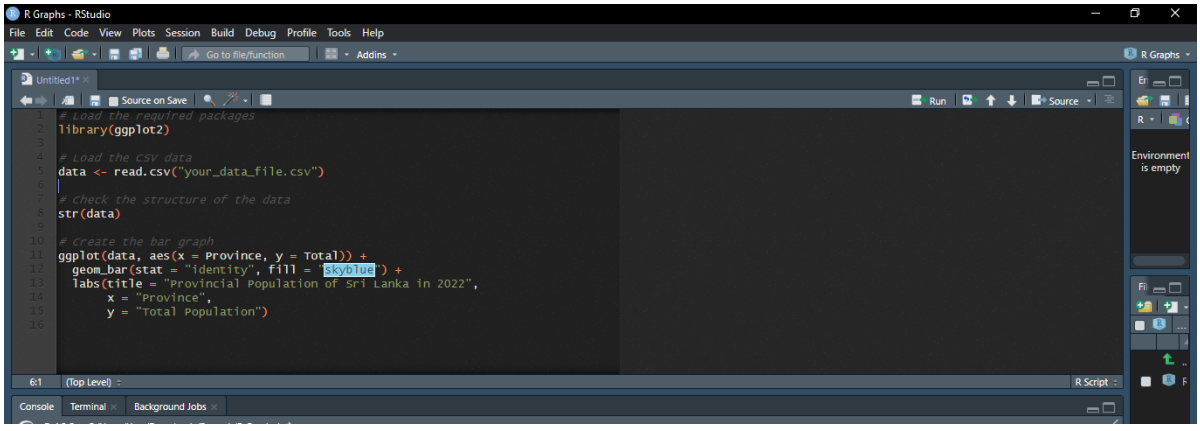
 # Check the structure of the data

```
str(data)
```

 # Create the bar graph with adjusted theme settings

```
ggplot(data, aes(x = Province, y = Total)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(title = "Provincial Population of Sri Lanka in 2022",
       x = "Province",
       y = "Total Population") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        plot.title = element_text(hjust = 0.5))
```

- **Enter Your Code as below.**

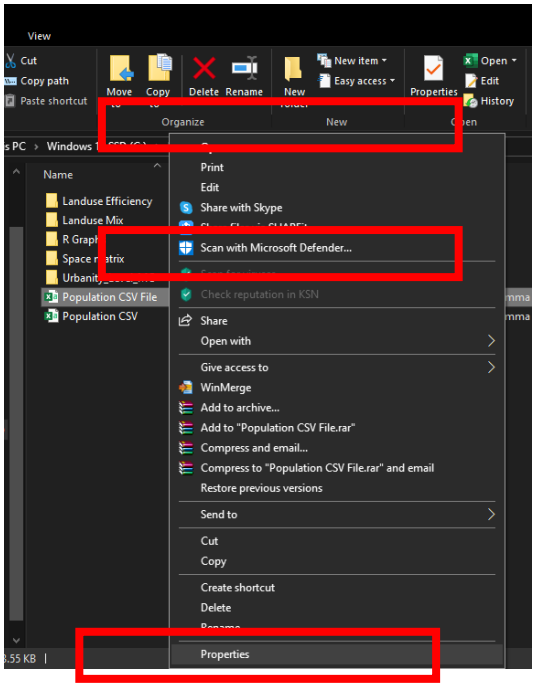


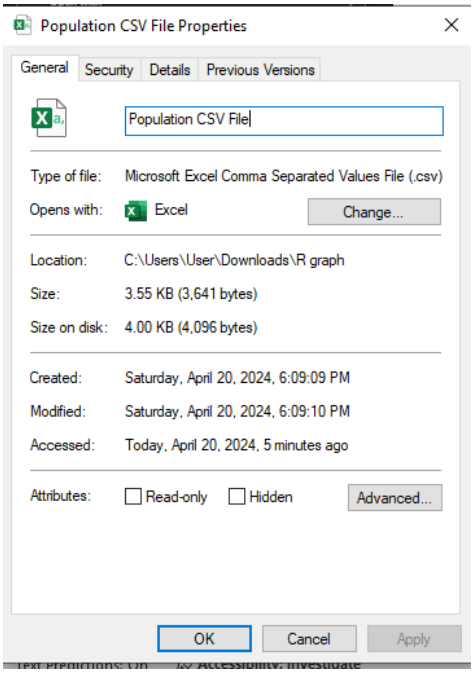
- **Load Your CSV**

```
# Load the required packages
library(ggplot2)

# Load the CSV data
data <- read.csv("your_data_file.csv")
```

- **Add your CSV file path.**





Ex: C:\Users\User\Downloads\R graph\Population CSV File.csv

Important – Please Enter double backslashes (\) between folder paths

Ex:"C:\\Users\\User\\Downloads\\R graph\\Population CSV File.csv"

- **Rename with your XY Axes with data column names**

```
# Create the bar graph
ggplot(data, aes(x = Province, y = Total)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(title = "Provincial Population of Sri Lanka in 2022",
       x = "Province",
       y = "Total Population")
```

Figure 7 - Coding Screenshots

7. CHOROPLETH CHART MAPPING FOR HIGHEST LAND USE

Code:

```
# Load necessary libraries
library(sf)
library(dplyr)
# Read the shapefiles
landuse_gnd <- st_read("C:\\Users\\User\\Downloads\\R graph\\data\\Landuse_GND.shp")
cmc_gnd <- st_read("C:\\Users\\User\\Downloads\\R graph\\data\\CMC_GND.shp")
# Calculate the dominant land use type within each GND
dominant_land_use <- landuse_gnd %>%
  group_by(GND_N_2012, DESCRIPTIO) %>%
  summarize(total_area = sum(Land_area)) %>%
  slice_max(total_area)
## Merge with GND boundaries
merged_data <- sf::st_join(cmc_gnd, dominant_land_use, by = "GND_N_2012")
# Define a color palette for different land use types
color_palette <- c("Built up Area" = "#FF0000", "River" = "#3333FF", "Garden" = "#00FF00","Main Road (A)" ="#330000","Canals (wide)"="#66B2FF",
                   "Grassland"="#4C9900","Lake" ="#0080FF" ,
                   "Marsh" ="#FFFF00" , "Minor Road"="#990000" , "Paddy" ="#66CC00"
                   , "Sand or Beach"="#FF8000" , "Scrub" ="#CCFF99"
                   ,"Transport & Utilities"="#FF00FF", "Water Hole"="#99CCFF")
# Plot the GND polygons and color them based on the dominant land use type
plot(merged_data$geometry, col = color_palette[merged_data$DESCRIPTIO])
# Extract unique GND names
Gnd_names<-unique(merged_data$GND_N_2012)
```

Add Shp data

Replace with your Landuse area column names

Replace with your common column name in shp

Customize colors and types according to your land use types

Output:

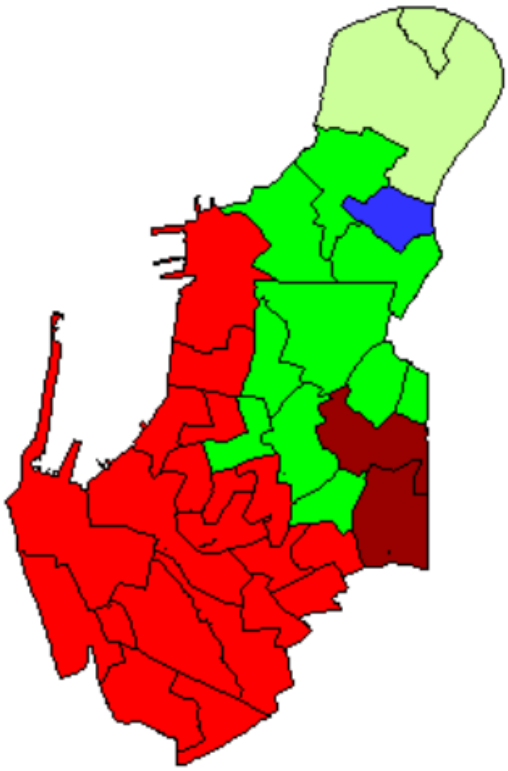


Figure 8 - Choropleth Chart mapping output

8. BUBBLE CHART

Bubble charts are versatile and can be used with various datasets depending on the context of your analysis. Here are a few examples of datasets commonly used for creating bubble charts.

Code:

```
#Install Packeges

install.packages("hrbrthemes")

install.packages("gapminder")

# Libraries

library(ggplot2)

library(dplyr)

library(hrbrthemes)

library(viridis)


# The dataset is provided in the gapminder library
                                                    Change Year and areas as you want
library(gapminder)

data_asia <- gapminder %>% filter(year == "2007" & continent == "Asia") %>% select(-year)

# Define custom colors for countries

custom_colors <- c("China" = "#FF5733", "India" = "#FFC300", "Indonesia" = "#C70039",
                   "Iran" = "#900C3F", "Japan" = "#6A1B9A", "Pakistan" = "#21618C",
                   "Philippines" = "#008F39", "Russia" = "#5D6D7E")

# Most basic bubble plot

plot <- data_asia %>%
  arrange(desc(pop)) %>%
  mutate(country = factor(country, levels = names(custom_colors))) %>%
  ggplot(aes(x = gdpPercap, y = lifeExp, size = pop, fill = country)) +
  geom_point(alpha = 0.7, shape = 21, color = "black") +
  scale_size(range = c(1, 24), name = "Population (M)") +
  scale_fill_manual(values = custom_colors, name = "Country") +
  theme_ipsum() +
  ylab("Life Expectancy") +
  xlab("Gdp per Capita") +
  theme(legend.position = "bottom")
```

 # Save the plot as a PNG file

ggsave("bubble_plot.png", plot, width = 10, height = 8, units = "in", dpi = 300)

Output:

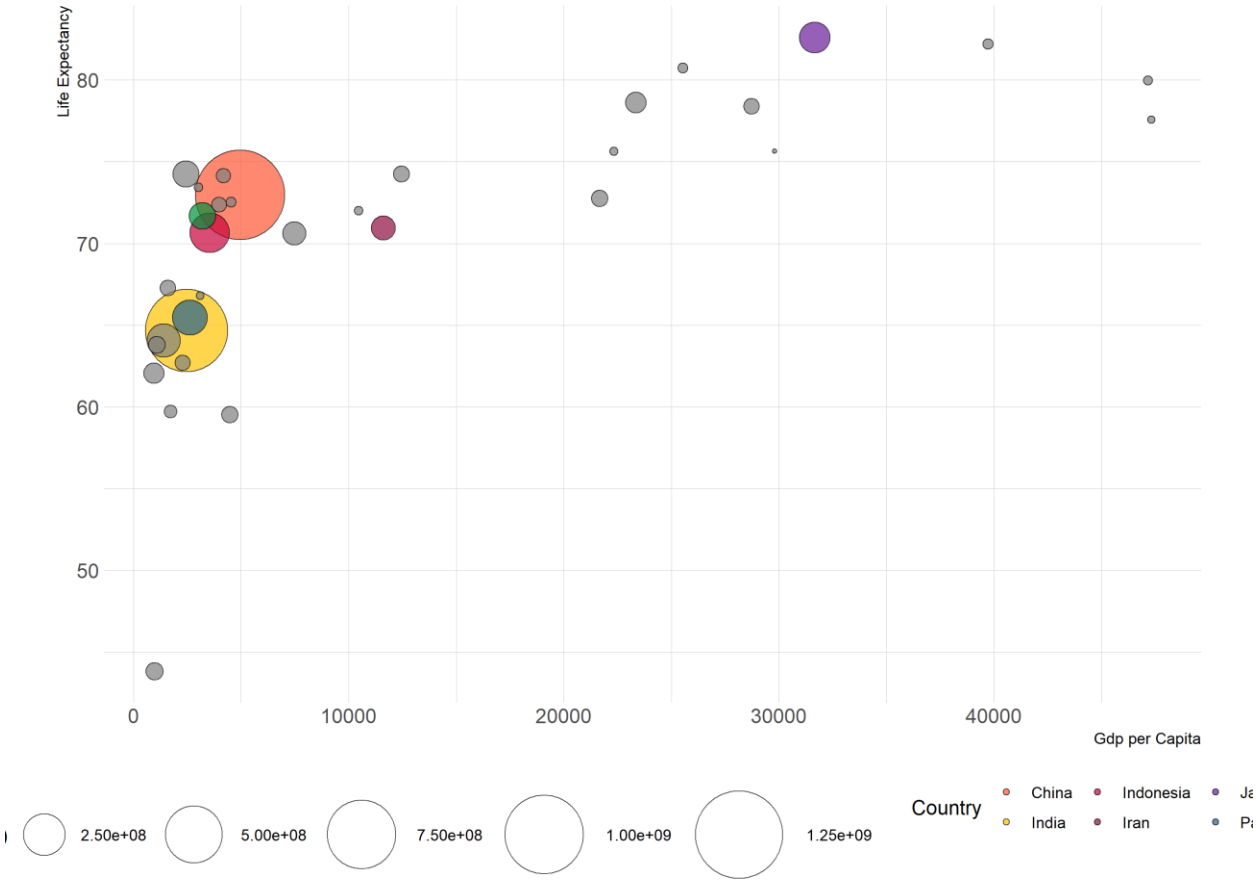


Figure 9 - Bubble Chart

9. BASIC RIDGELINE PLOT

Code:

```
# library
library(ggridges)
library(ggplot2)
# Diamonds dataset is provided by R natively
#head(diamonds)
# basic example
ggplot(diamonds, aes(x = price, y = cut, fill = cut)) +
  geom_density_ridges() +
  theme_ridges() +
  theme(legend.position = "none")
```

Output:

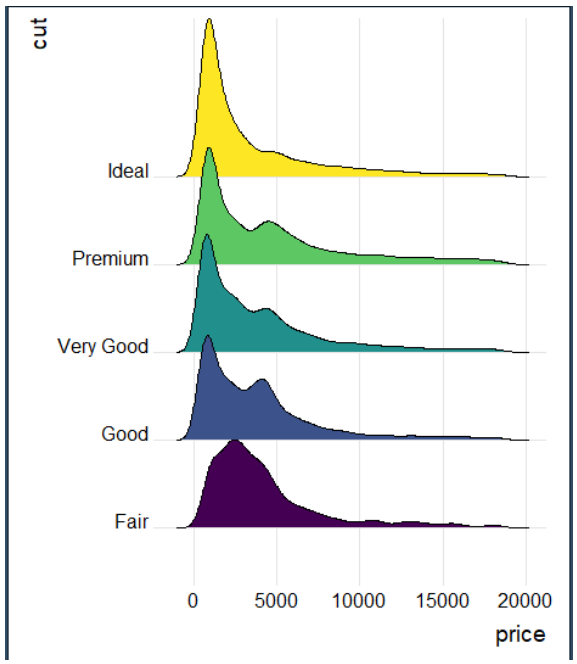


Figure 10 - Basic ridgeline plot

Geom Layer: geom_density_ridges() from the ggridges package is used to create the ridge plot. This function calculates kernel density estimates for the distribution of diamond prices for each cut and stacks them vertically.

Themes: theme_ridges() is applied to the plot to provide the specific theme designed for ridge plots. Additionally, theme(legend.position = "none") is used to remove the legend, as the fill color is based on the cut variable, which is already represented on the y-axis.

10. TIME SERIES LINE PLOT

Code:

```
# Load required libraries
library(ggplot2)
library(dplyr)
# Read the CSV file
                                                    Change With your CSV data.
weather_data                                                                    <-
read.csv("C:\\Users\\User\\Downloads\\SriLanka_Weather_Dataset.csv\\SriLanka_Weather_Dataset.csv")
# Check the structure of the data
str(weather_data)
# Plot temperature data over time
ggplot(weather_data, aes(x = time, y = temperature_2m_mean)) +
  geom_line() +
  labs(x = "Time", y = "Mean Temperature (°C)", title = "Mean Temperature Over Time")
# Plot precipitation data over time
ggplot(weather_data, aes(x = time, y = precipitation_sum)) +
  geom_bar(stat = "identity", fill = "blue") +
  labs(x = "Time", y = "Precipitation (mm)", title = "Precipitation Over Time")
# Add more plots as needed for different variables
```

Output:

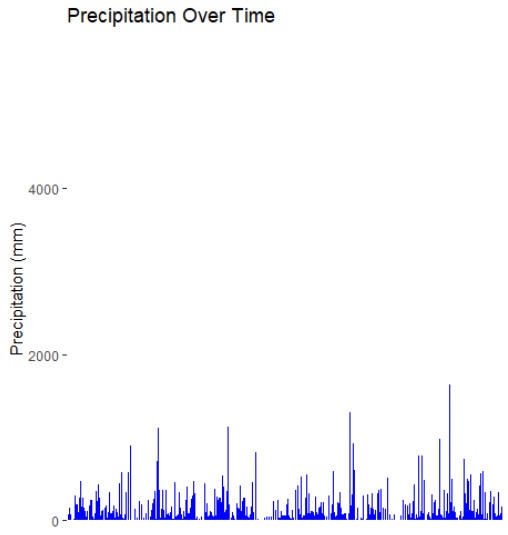


Figure 11 - Time series plot

11. SCATTER PLOT

Code:

```
# Load necessary libraries
library(ggplot2)

# Read the CSV file                                    Replace with Your CSV
weather_data <- read.csv("C:\\Users\\User\\Downloads\\SriLanka_Weather_Dataset\\SL.csv")

# Create a scatter plot with colorful points
ggplot(weather_data, aes(x = temperature_2m_mean, y = precipitation_sum, color = weathercode)) +
  geom_point() +                                       Replace with Your Colum names
  labs(title = "Temperature vs Precipitation",
       x = "Mean Temperature (°C)",
       y = "Precipitation Sum (mm)")
```

Output :

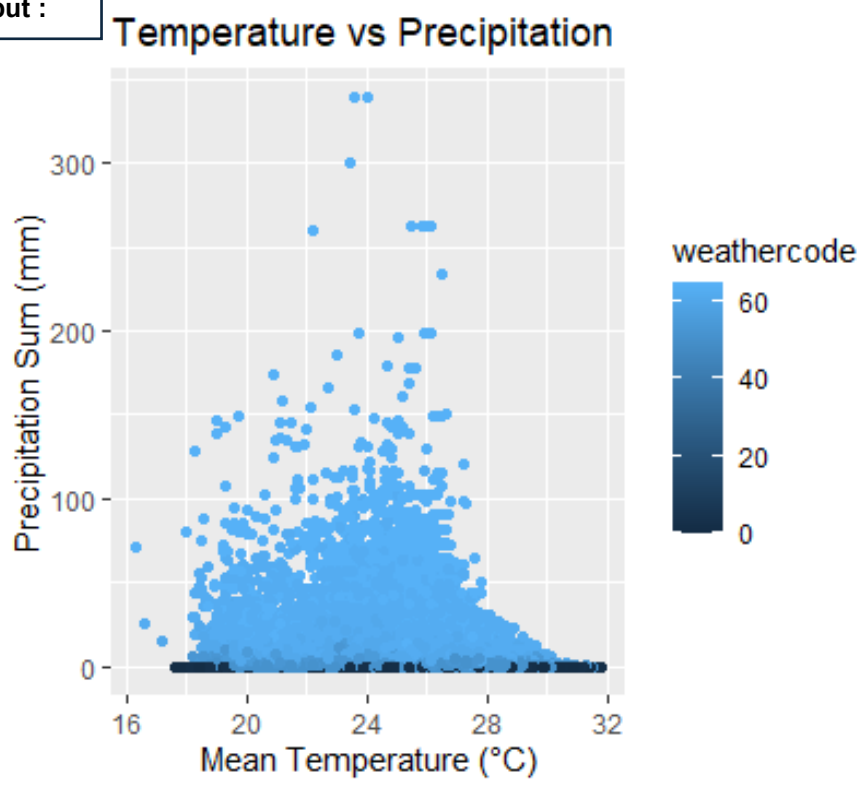


Figure 12 - Scatter Plot

12. VIOLIN PLOT

Code:

Replace with Your CSV

```
# Read the CSV file
weather_data <- read.csv("C:\\Users\\User\\Downloads\\SriLanka_Weather_Dataset\\SL.csv")

# Create a violin plot
ggplot(weather_data, aes(x = weathercode, y = temperature_2m_mean, fill = weathercode)) +
  geom_violin(alpha = 0.8) +
  geom_boxplot(width = 0.1, color = "black", alpha = 0.5) +
  scale_fill_viridis_d() +
  labs(title = "Distribution of Temperature Across Weather Codes",
       x = "Weather Code",
       y = "Mean Temperature (°C)") +
  theme_minimal()
```

Output:

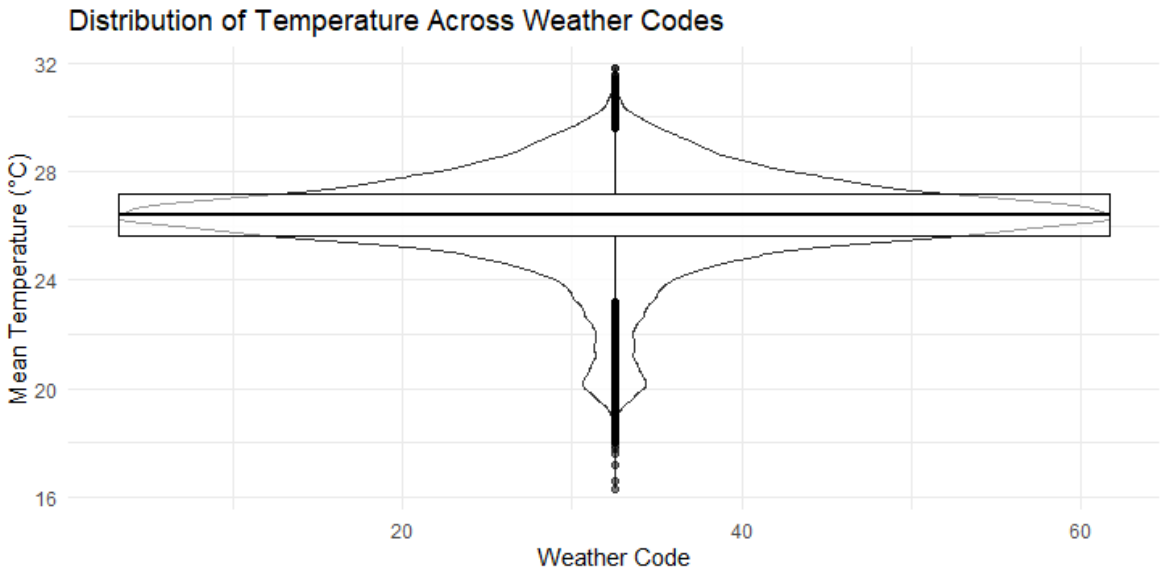


Figure 13 - Violin plot.

13. HEATMAPS

Code:

```
# Load necessary libraries
library(ggplot2)
library(reshape2)
# Read the CSV file
```

Replace with Your CSV

```
weather_data <- read.csv("C:\\Users\\User\\Downloads\\SriLanka_Weather_Dataset\\SL.csv")
# Select relevant columns for the heatmap
heatmap_data <- weather_data[, c("temperature_2m_max", "temperature_2m_min", "temperature_2m_mean", "precipitation_sum", "latitude", "longitude")]
# Compute correlation matrix
correlation_matrix <- cor(heatmap_data[, c("temperature_2m_max", "temperature_2m_min", "temperature_2m_mean", "precipitation_sum")])
# Convert correlation matrix to long format
correlation_matrix_long <- melt(correlation_matrix)
# Create a heatmap
ggplot(correlation_matrix_long, aes(x = Var1, y = Var2, fill = value)) +
  geom_tile() +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white", midpoint = 0, limit = c(-1, 1), name = "Correlation") +
  labs(title = "Correlation Heatmap of Weather Variables",
       x = "Weather Variables",
       y = "Weather Variables") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1))
```

Output:

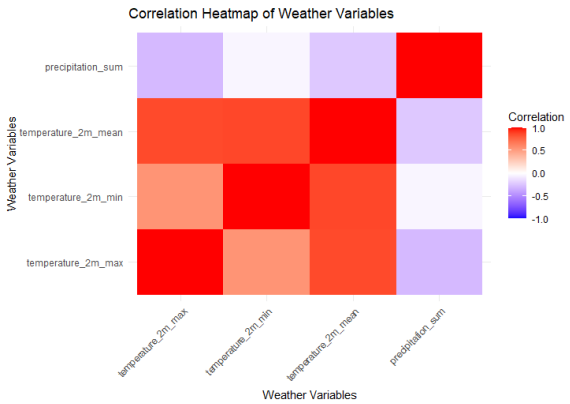


Figure 14 - Heat Maps

14. STACKED BAR

Code:

```
# Load necessary libraries
library(ggplot2)
```

Replace with Your CSV

```
# Read the CSV file
weather_data <- read.csv("C:\\Users\\User\\Downloads\\SriLanka_Weather_Dataset\\SL.csv")


# Create the stacked bar plot
ggplot(weather_data, aes(x = city, fill = factor(weathercode))) +
  geom_bar() +
  labs(title = "Distribution of Weather Codes Across Locations",
       x = "City",
       y = "Count",
       fill = "Weather Code") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

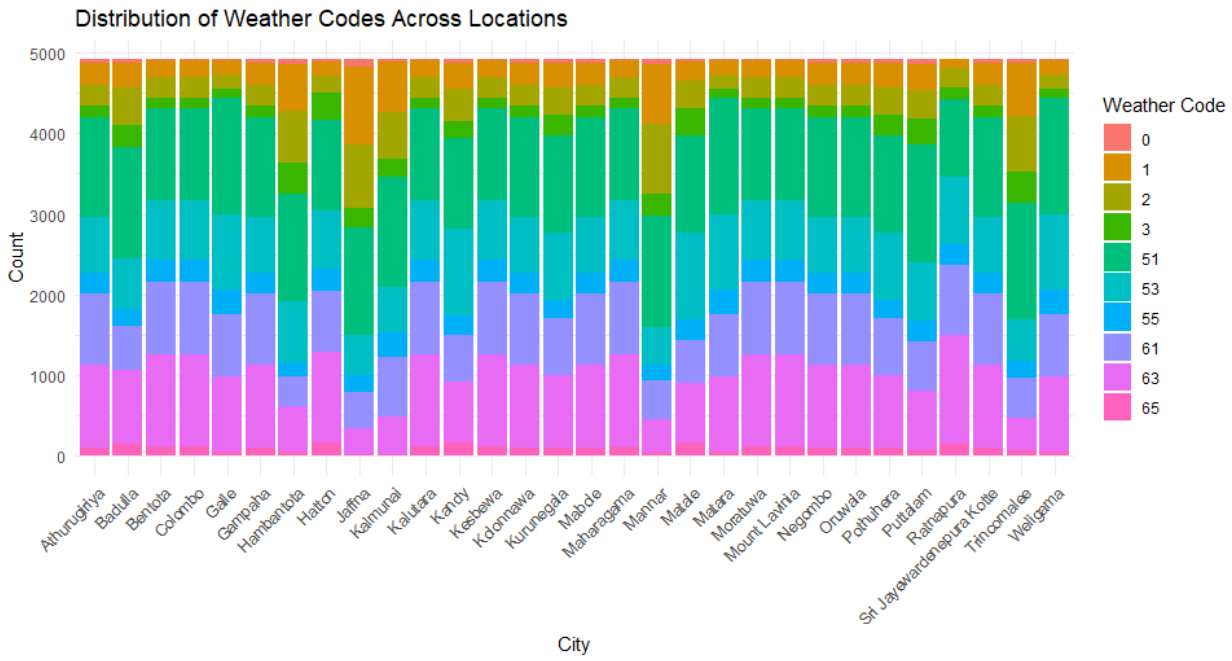


Figure 15 - Stacked bar.

- **Plotting:** The ggplot() function initializes the plot. Inside aes(), x is mapped to the cities (city), and fill is mapped to the weather codes (weathercode).
- **Geometric Layer:** geom_bar() is used to create the stacked bar plot. Each bar represents a city, and the height of each segment within the bar represents the count of different weather codes for that city.
- **Labels and Titles:** The labs() function is used to set the plot title, x-axis label, y-axis label, and legend title.
- **Themes:** theme_minimal() is applied to provide a clean and minimalistic appearance to the plot. Additionally, theme(axis.text.x = element_text(angle = 45, hjust = 1)) rotates the x-axis labels by 45 degrees for better readability.

AFTERWORD

Urban Informatics relies on the effective communication of spatial and statistical insights. This guide introduces R programming as a powerful tool for data visualization in urban studies, enabling users to convert complex datasets into clear, impactful graphics that support transparent and actionable planning outcomes.

Drawing from international best practices and open-source methodologies, this resource strengthens the analytical capabilities of those working at the interface of data science and urban planning. We trust it will inspire continued exploration in visual storytelling, making urban data accessible, meaningful, and transformative.

lbs2its.net

618657-EPP-1-2020-1-AT-EPPKA2-CBHE-JP