On Job Training (OJT 2025)

A Project Report on:

Expenditure Modelling and Transformation

Submitted to:

Department of Mathematics,
Institute of Chemical Technology, Mumbai

Submitted by:

Rajiba Lochan Sahoo
Roll Number: 24MAT207
Institute of Chemical Technology, Mumbai

Date of Submission:

9th July 2025

Abstract

This project explores expenditure modeling under the Pradhan Mantri Gram Sadak Yojana (PMGSY), a major rural infrastructure scheme in India. Using a dataset of over 2273 entries across different states and districts, this study aims to understand the factors influencing actual spending on road projects.

The data underwent thorough cleaning, transformation, and exploration to reveal trends and relationships. Statistical models such as Multiple Linear Regression, Lasso Regression, and Partial Least Squares (PLS) were applied. While Lasso helped identify significant predictors, PLS provided the most accurate predictions with the lowest RMSE.

Key findings indicate that road length, scheme type, cost of work sanctioned, and state-level political and economic factors are major drivers of expenditure. The results can support better planning and transparency in public infrastructure projects.

Statement by the Candidate

I hereby declare that the work presented in this report titled **"Expenditure Modelling and Transformation"** is my original contribution, carried out as part of my On-the-Job Training (OJT).This work has not been submitted elsewhere for the award of any academic degree or diploma. All sources used have been properly acknowledged and cited.

Rajiba Lochan Sahoo
Roll Number: 24MAT207

Acknowledgements

I would like to express my heartfelt gratitude to Prof. Amiya Ranjan Bhowmick (Institute of Chemical Technology, Mumbai) for introducing me to this project and supporting me throughout the On-the-Job Training. His valuable feedback and consistent encouragement have been instrumental in shaping this work.

I am also deeply thankful to Prof. Anirban Chakraborti (Jawaharlal Nehru University), under whose supervision this project was conducted. His discussions and insights helped refine the modeling and analytical aspects of the study.

I extend my thanks to all those who contributed, directly or indirectly, to the successful completion of this project.

Executive Summary

This project explores expenditure modeling under the Pradhan Mantri Gram Sadak Yojana (PMGSY), a major rural infrastructure scheme in India. Using a dataset of over 2200 entries, we examine how financial and political factors influence actual project costs. After cleaning and transforming the data, we applied several regression models. Lasso regression helped reduce overfitting, while Partial Least Squares (PLS) provided the most accurate predictions with the lowest RMSE. Our findings suggest that road length, scheme type, and sanctioned cost are major drivers of expenditure. This model can aid future planning and transparent fund allocation for rural development

Table of contents

List of Figures

List of Tables

1 Data Description

This dataset contains **2273 observations (rows)** and **16 variables (columns)**.
It provides detailed information about road and bridge construction projects sanctioned and executed under the **PMGSY (Pradhan Mantri Gram Sadak Yojana)** scheme across various states and districts in India.

1.1 Variable Descriptions

Variable Name	Description	Type	Unit
STATE_NAME	Name of the state	Categorical	Text
DISTRICT_NAME	Name of the district	Categorical	Text
PMGSY_SCHEME	Scheme type (e.g., PMGSY-I, PMGSY-II)	Categorical	Text
NO_OF_ROAD_WORK SANCTIONED	Number of sanctioned road works	Numeric	Count
NO_OF_BRIDGES SANCTIONED	Number of sanctioned bridge works	Numeric	Count
NO_OF_ROAD_WORKS COMPLETED	Number of completed road works	Numeric	Count
NO_OF_BRIDGES COMPLETED	Number of completed bridge works	Numeric	Count
NO_OF_ROAD WORKS_BALANCE	Remaining road works	Numeric	Count
NO_OF_BRIDGES BALANCE	Remaining bridge works	Numeric	Count
LENGTH_OF_ROAD_WORK SANCTIONED_KM	Total length of road work sanctioned	Numeric	Kilometers
COST_OF_WORKS SANCTIONED LAKHS	Total sanctioned cost	Numeric	Lakh rupees
LENGTH_OF_ROAD_WORK COMPLETED_KM	Completed road work length	Numeric	Kilometers
EXPENDITURE_OCCURED LAKHS	Actual expenditure incurred	Numeric	Lakh rupees
LENGTH_OF_ROAD_WORK BALANCE_KM	Remaining road work length	Numeric	Kilometers
STATE_GDP_CRORE	GDP of the state	Numeric	Crore rupees
POLITICAL_PARTY	Ruling political party in the state	Categorical	Text

- **Notes:**
- The dataset contains **no missing values**; all columns are complete.
- States like **Uttar Pradesh** have the highest number of entries and large-scale project execution.
- Some rows include **negative values for expenditure**, which may be due to data entry errors or financial adjustments and should be investigated further.
- The additional columns **STATE_GDP_CRORE** and **POLITICAL_PARTY** provide useful context for exploring correlations between economic indicators and project outcomes.

1.2 Aim of the Data

The purpose of this study is to explore and model the **expenditure patterns** of infrastructure projects sanctioned under the **Pradhan Mantri Gram Sadak Yojana (PMGSY)** scheme. The dataset contains 2273 observations and 16 variables covering project-level data across Indian states and districts.

Key objectives:

- Understand expenditure trends across different states, schemes, and ruling political parties.
- Analyze relationships between project metrics (length, cost, completion status) and actual expenditure.
- Use data transformations (log, Box-Cox) to reduce skewness and enable effective modeling.
- Apply machine learning models (LASSO) to predict expenditure and identify key drivers.

1.3 Observations

- Most numerical variables (e.g., road lengths, costs, expenditures) are **right-skewed** and contain **many outliers**.
- **State GDP** is more symmetrical than other variables.
- Shapiro-Wilk tests show **none of the numeric variables are normally distributed**.
- **Uttar Pradesh, Madhya Pradesh, Bihar** have the highest number of projects.
- **PMGSY-I, II, III** schemes dominate.
- **BJP, Congress** ruled states dominate projects.
- Parties like **BJD, BJP, DMK, TMC, JD(U)-RJD** show **higher median expenditures**.
- States like **ZPM, LDF (CPM), NDPP-BJP** show **lower consistent expenditures**.
- Data required **log(x + 1)** and **Box-Cox transformations** to correct skewness.
- Initial **linear regression was overfitting**. So, **Lasso** was used.
- **model**: Lasso (alpha = 1, lambda ≈ 0.01355057), RMSE ≈ **591.86 Lakhs**.
- Important predictors:
  - `COST_OF_WORKS_SANCTIONED_LAKHS`
  - `LENGTH_OF_ROAD_WORK_COMPLETED_KM`
  - `STATE_GDP_CRORE`
  - Scheme type (`PMGSY-I`)
  - State (`Bihar`, `Tamil Nadu`, etc.)

Overall, the analysis indicates that expenditure under PMGSY projects is primarily driven by road length, sanctioned project cost, and the broader economic and political context of each state.

1.4 Objects Used in Background and Modeling

2 Loading the Dataset and EDA

```
# Load required libraries
library(caret)
library(dplyr)
library(rsample)
library(vip)
library(glmnet)
library(ggplot2)
library(recipes)
library(readr)

# Read the CSV file
df = read.csv(r"(C:\Users\rajib\Downloads\ROAD WORK PMGSY.csv)")

# Create GDP as named vector
state_gdp  = c(
  "Andaman And Nicobar" = 8000,
  "Andhra Pradesh" = 1350000,
  "Arunachal Pradesh" = 35000,
  "Assam" = 450000,
  "Bihar" = 800000,
  "Chhattisgarh" = 430000,
  "Goa" = 85000,
  "Gujarat" = 2160000,
  "Haryana" = 1050000,
  "Himachal Pradesh" = 200000,
  "Jharkhand" = 450000,
  "Karnataka" = 2250000,
  "Kerala" = 1090000,
  "Madhya Pradesh" = 1250000,
  "Maharashtra" = 3500000,
  "Manipur" = 35000,
  "Meghalaya" = 40000,
  "Mizoram" = 25000,
  "Nagaland" = 30000,
  "Odisha" = 800000,
  "Punjab" = 750000,
  "Rajasthan" = 1300000,
  "Sikkim" = 30000,
  "Tamil Nadu" = 2400000,
  "Telangana" = 1300000,
  "Tripura" = 50000,
  "Uttar Pradesh" = 2400000,
```

```
  "Uttarakhand" = 400000,
  "West Bengal" = 1650000,
  "Chandigarh" = 50000,
  "Delhi" = 1100000,
  "Jammu And Kashmir" = 170000,
  "Ladakh" = 3000,
  "Puducherry" = 40000
)

# Create political party as named vector
political_party =  c(
  "Andhra Pradesh" = "YSR Congress",
  "Arunachal Pradesh" = "BJP",
  "Assam" = "BJP",
  "Bihar" = "JD(U)-RJD Alliance",
  "Chhattisgarh" = "Congress",
  "Goa" = "BJP",
  "Gujarat" = "BJP",
  "Haryana" = "BJP",
  "Himachal Pradesh" = "Congress",
  "Jharkhand" = "JMM-Congress Alliance",
  "Karnataka" = "Congress",
  "Kerala" = "LDF (CPM)",
  "Madhya Pradesh" = "BJP",
  "Maharashtra" = "BJP-Shiv Sena",
  "Manipur" = "BJP",
  "Meghalaya" = "NPP-BJP Alliance",
  "Mizoram" = "ZPM",
  "Nagaland" = "NDPP-BJP Alliance",
  "Odisha" = "BJD",
  "Punjab" = "AAP",
  "Rajasthan" = "Congress",
  "Sikkim" = "Sikkim Krantikari Morcha",
  "Tamil Nadu" = "DMK",
  "Telangana" = "Congress",
  "Tripura" = "BJP",
  "Uttar Pradesh" = "BJP",
  "Uttarakhand" = "BJP",
  "West Bengal" = "TMC",
  "Andaman And Nicobar" = "Central Admin",
  "Chandigarh" = "BJP",
  "Delhi" = "AAP",
  "Jammu And Kashmir" = "Central Admin",
  "Ladakh" = "Central Admin",
  "Puducherry" = "BJP"
)
```

```
# Add GDP and Political Party columns to the data frame
df = df %>%
  mutate(
    STATE_GDP_CRORE = state_gdp[STATE_NAME],
    POLITICAL_PARTY = political_party[STATE_NAME]
  )

# Save the updated data
df1 = write_csv(df,(r"(C:\Users\rajib\Downloads\d4361151-6d41-43c7-98cd-9a6cd90b5ca4 (1).csv)")

names(df1)
```

```
 [1] "STATE_NAME"                      "DISTRICT_NAME"
 [3] "PMGSY_SCHEME"                    "NO_OF_ROAD_WORK_SANCTIONED"
 [5] "NO_OF_BRIDGES_SANCTIONED"        "NO_OF_ROAD_WORKS_COMPLETED"
 [7] "NO_OF_BRIDGES_COMPLETED"         "NO_OF_ROAD_WORKS_BALANCE"
 [9] "NO_OF_BRIDGES_BALANCE"           "LENGTH_OF_ROAD_WORK_SANCTIONED_KM"
[11] "COST_OF_WORKS_SANCTIONED_LAKHS"  "LENGTH_OF_ROAD_WORK_COMPLETED_KM"
[13] "EXPENDITURE_OCCURED_LAKHS"       "LENGTH_OF_ROAD_WORK_BALANCE_KM"
[15] "STATE_GDP_CRORE"                 "POLITICAL_PARTY"
```

```
dim(df1)
```

```
[1] 2273   16
```

```
head(df1)
```

```
           STATE_NAME            DISTRICT_NAME PMGSY_SCHEME
1 Andaman And Nicobar North and Middle Andaman     PMGSY-II
2 Andaman And Nicobar            South Andaman      PMGSY-I
3 Andaman And Nicobar            South Andaman     PMGSY-II
4 Andaman And Nicobar            South Andaman    PMGSY-III
5      Andhra Pradesh    Alluri Sitharama Raju    PMGSY-III
6      Andhra Pradesh                 Chittoor      PMGSY-I
  NO_OF_ROAD_WORK_SANCTIONED NO_OF_BRIDGES_SANCTIONED
1                         24                        0
2                         32                        0
3                         11                        0
4                          8                        0
5                          3                        1
6                        148                        3
  NO_OF_ROAD_WORKS_COMPLETED NO_OF_BRIDGES_COMPLETED NO_OF_ROAD_WORKS_BALANCE
1                          0                       0                       24
2                         32                       0                        0
3                         11                       0                        0
```

```
4                          0                        0                        8
5                          2                        0                        1
6                        148                        3                        0
  NO_OF_BRIDGES_BALANCE LENGTH_OF_ROAD_WORK_SANCTIONED_KM
1                     0                          54.670
2                     0                          40.146
3                     0                          20.082
4                     0                          27.896
5                     1                          29.100
6                     0                         482.261
  COST_OF_WORKS_SANCTIONED_LAKHS LENGTH_OF_ROAD_WORK_COMPLETED_KM
1                        27.7646                          19.660
2                        18.5375                          39.287
3                        10.4095                          17.747
4                        34.2347                           0.000
5                        15.6200                          18.730
6                       101.8512                         471.991
  EXPENDITURE_OCCURED_LAKHS LENGTH_OF_ROAD_WORK_BALANCE_KM STATE_GDP_CRORE
1                    4.1501                         35.010            8000
2                   11.6894                          0.000            8000
3                    8.6343                          0.000            8000
4                    0.0000                         27.896            8000
5                   10.2977                         11.200         1350000
6                  100.7436                          0.000         1350000
  POLITICAL_PARTY
1   Central Admin
2   Central Admin
3   Central Admin
4   Central Admin
5    YSR Congress
6    YSR Congress
```

```
class(df1)
```

```
[1] "data.frame"
```

```
#summary(df1)
```

2.1 Read and Structure Data

The function class is important to understand different data types. In data analysis, this function is often useful to check for the correct input type to specific functions in R.

```
# Check the class
sapply(df1[c(
```

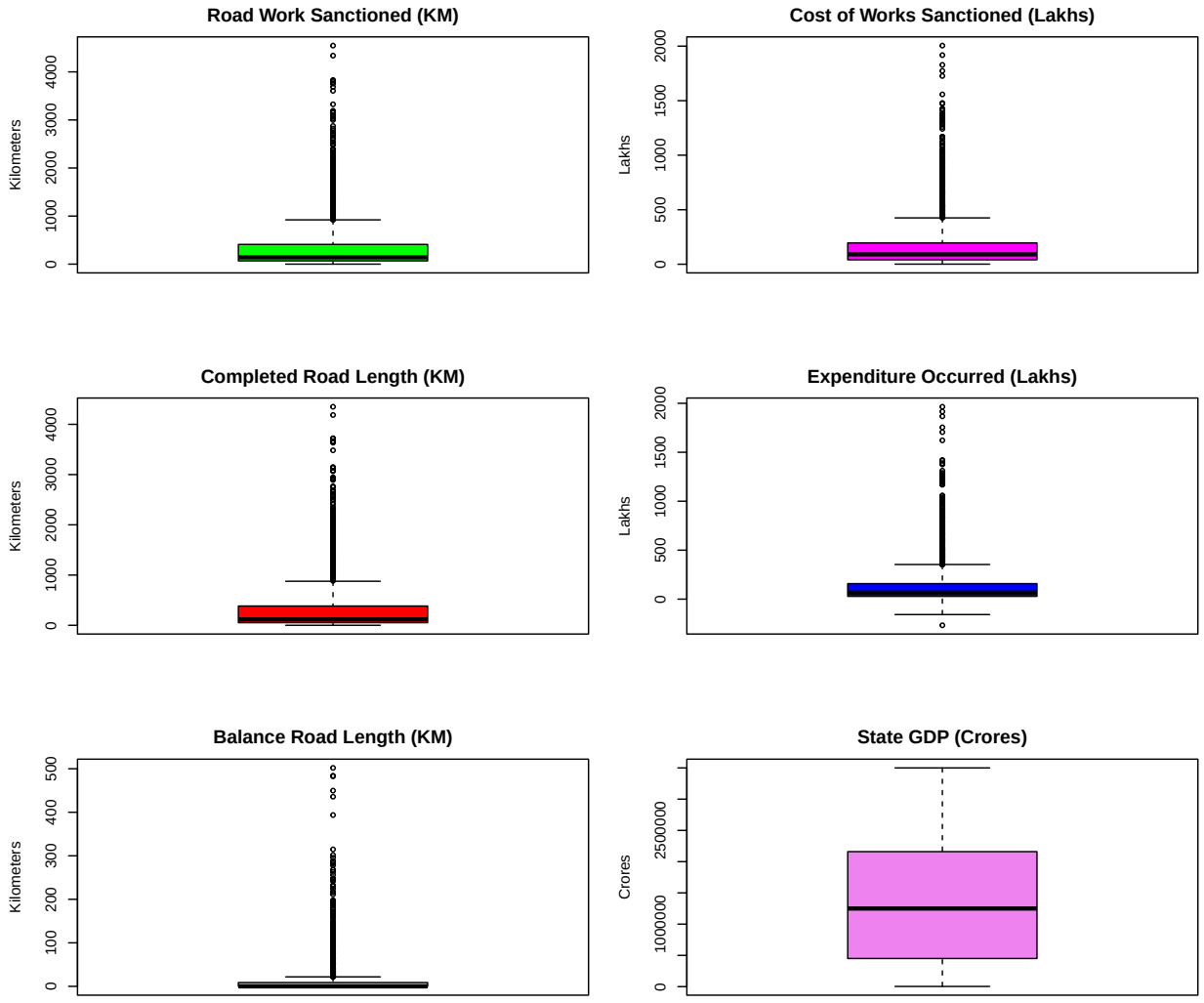
```
  "STATE_NAME",
  "DISTRICT_NAME",
  "PMGSY_SCHEME",
  "NO_OF_ROAD_WORK_SANCTIONED",
  "NO_OF_BRIDGES_SANCTIONED",
  "NO_OF_ROAD_WORKS_COMPLETED",
  "NO_OF_BRIDGES_COMPLETED",
  "NO_OF_ROAD_WORKS_BALANCE",
  "NO_OF_BRIDGES_BALANCE",
  "LENGTH_OF_ROAD_WORK_SANCTIONED_KM",
  "COST_OF_WORKS_SANCTIONED_LAKHS",
  "LENGTH_OF_ROAD_WORK_COMPLETED_KM",
  "EXPENDITURE_OCCURED_LAKHS",
  "LENGTH_OF_ROAD_WORK_BALANCE_KM",
  "STATE_GDP_CRORE",
  "POLITICAL_PARTY"
)], class)
```

```
                     STATE_NAME                     DISTRICT_NAME
                    "character"                       "character"
                    PMGSY_SCHEME         NO_OF_ROAD_WORK_SANCTIONED
                    "character"                         "integer"
          NO_OF_BRIDGES_SANCTIONED         NO_OF_ROAD_WORKS_COMPLETED
                    "integer"                         "integer"
           NO_OF_BRIDGES_COMPLETED           NO_OF_ROAD_WORKS_BALANCE
                    "integer"                         "integer"
             NO_OF_BRIDGES_BALANCE LENGTH_OF_ROAD_WORK_SANCTIONED_KM
                    "integer"                         "numeric"
   COST_OF_WORKS_SANCTIONED_LAKHS  LENGTH_OF_ROAD_WORK_COMPLETED_KM
                    "numeric"                         "numeric"
         EXPENDITURE_OCCURED_LAKHS    LENGTH_OF_ROAD_WORK_BALANCE_KM
                    "numeric"                         "numeric"
                STATE_GDP_CRORE                   POLITICAL_PARTY
                    "numeric"                       "character"
```

2.2 Create boxplot and histogram of numeric data

```
# margins:bottom, left, top, right
par(mfrow = c(3,2), mar = c(4.5, 4.5, 2.5, 1))

boxplot(df1$LENGTH_OF_ROAD_WORK_SANCTIONED_KM,
        col = "green",
        main = "Road Work Sanctioned (KM)",
        ylab = "Kilometers")
```

```r
boxplot(df1$COST_OF_WORKS_SANCTIONED_LAKHS,
        col = "magenta",
        main = "Cost of Works Sanctioned (Lakhs)",
        ylab = "Lakhs")

boxplot(df1$LENGTH_OF_ROAD_WORK_COMPLETED_KM,
        col = "red",
        main = "Completed Road Length (KM)",
        ylab = "Kilometers")

boxplot(df1$EXPENDITURE_OCCURED_LAKHS,
        col = "blue",
        main = "Expenditure Occurred (Lakhs)",
        ylab = "Lakhs")

boxplot(df1$LENGTH_OF_ROAD_WORK_BALANCE_KM,
        col = "grey",
        main = "Balance Road Length (KM)",
        ylab = "Kilometers")

boxplot(df1$STATE_GDP_CRORE,
        col = "violet",
        main = "State GDP (Crores)",
        ylab = "Crores")
```

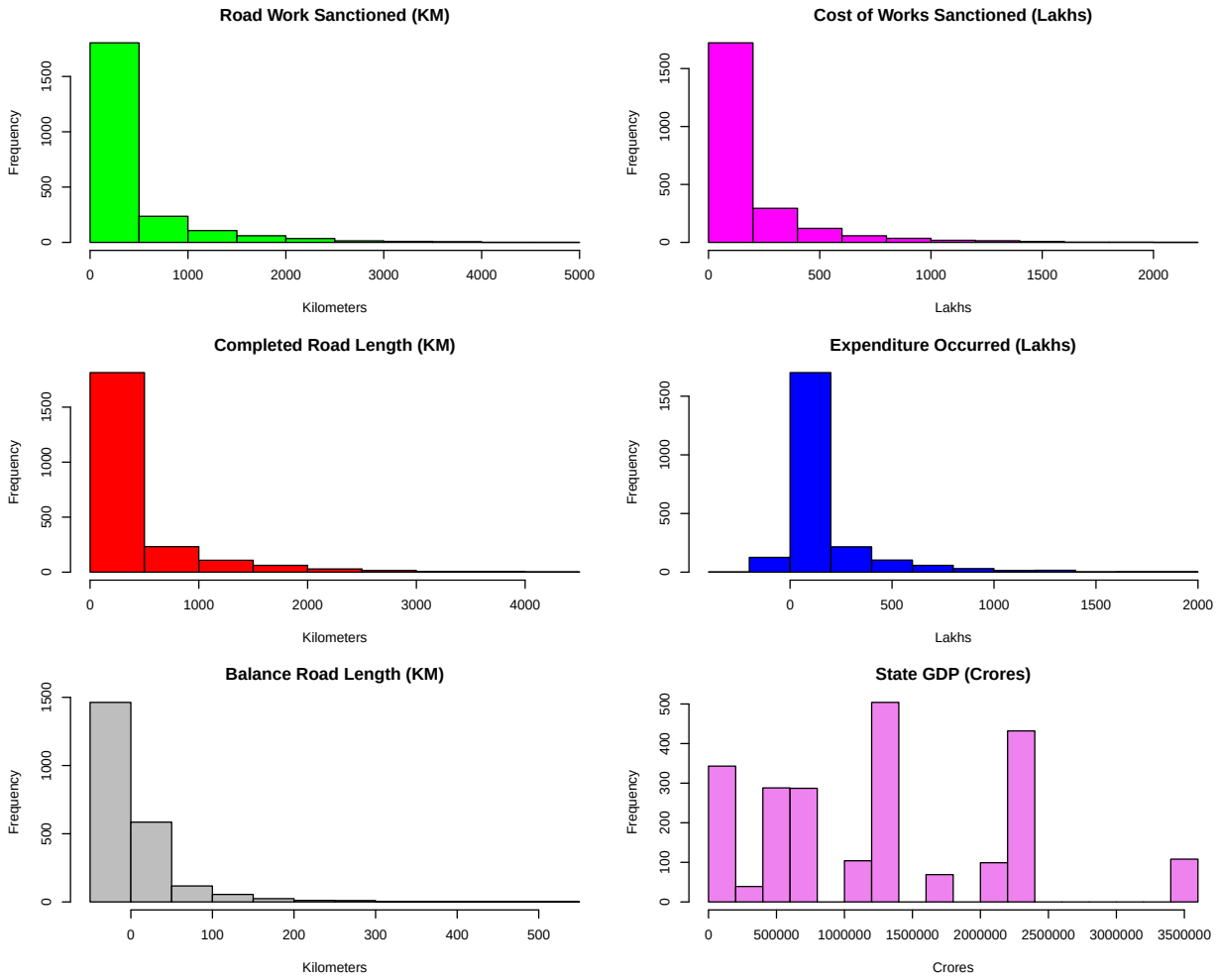


In the above boxplots, many variables are positively skewed, meaning most values are low but a few are very high. Also, most variables have many outliers. Only State GDP looks more balanced, ranging from ₹3,000 crores to ₹35,00,000 crores, and it doesn't have many outliers.

plot histogram to observed the distribution

```
par(mfrow = c(3, 2), mar = c(4.5, 4.5, 2.5, 1))

hist(df1$LENGTH_OF_ROAD_WORK_SANCTIONED_KM,
     col = "green",
     main = "Road Work Sanctioned (KM)",
     ylab = "Frequency",
     xlab = "Kilometers")

hist(df1$COST_OF_WORKS_SANCTIONED_LAKHS,
     col = "magenta",
```

```
     main = "Cost of Works Sanctioned (Lakhs)",
     ylab = "Frequency",
     xlab = "Lakhs")

hist(df1$LENGTH_OF_ROAD_WORK_COMPLETED_KM,
     col = "red",
     main = "Completed Road Length (KM)",
     ylab = "Frequency",
     xlab = "Kilometers")

hist(df1$EXPENDITURE_OCCURED_LAKHS,
     col = "blue",
     main = "Expenditure Occurred (Lakhs)",
     ylab = "Frequency",
     xlab = "Lakhs")

hist(df1$LENGTH_OF_ROAD_WORK_BALANCE_KM,
     col = "grey",
     main = "Balance Road Length (KM)",
     ylab = "Frequency", xlab = "Kilometers")

hist(df1$STATE_GDP_CRORE,
     col = "violet",
     main = "State GDP (Crores)",
     ylab = "Frequency",
     xlab = "Crores")
```

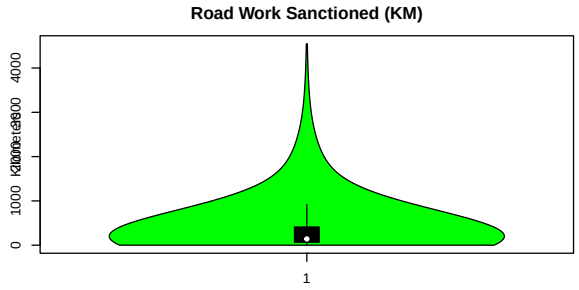


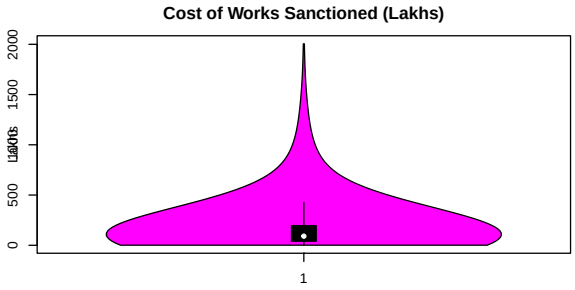
The above histograms show that none of the variables follow a normal distribution.Most distributions are right-skewed

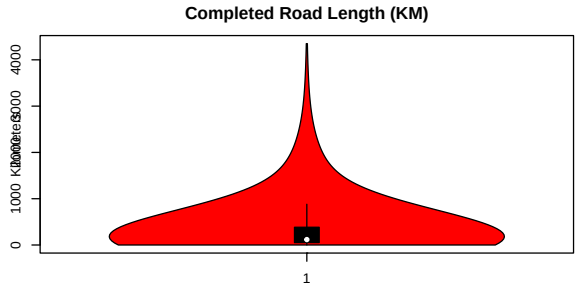
Violplots for numerical variables

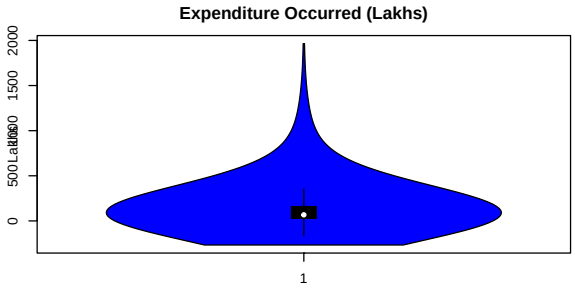
```
library(vioplot)
par(mfrow = c(3, 2), mar = c(4.5, 4.5, 2.5, 1))

vioplot(df1$LENGTH_OF_ROAD_WORK_SANCTIONED_KM,
        col = "green",
        main = "Road Work Sanctioned (KM)",
        ylab = "Kilometers")

vioplot(df1$COST_OF_WORKS_SANCTIONED_LAKHS,
        col = "magenta",
        main = "Cost of Works Sanctioned (Lakhs)",
        ylab = "Lakhs")

vioplot(df1$LENGTH_OF_ROAD_WORK_COMPLETED_KM,
```

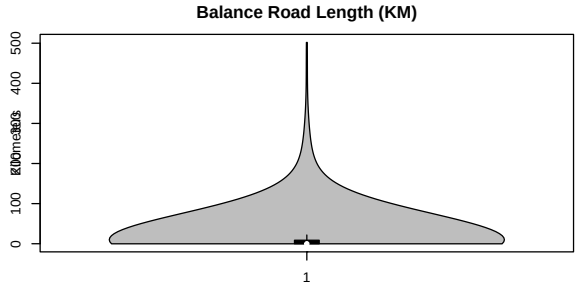
```
        col = "red",
        main = "Completed Road Length (KM)",
        ylab = "Kilometers")

vioplot(df1$EXPENDITURE_OCCURED_LAKHS,
        col = "blue",
        main = "Expenditure Occurred (Lakhs)",
        ylab = "Lakhs")

vioplot(df1$LENGTH_OF_ROAD_WORK_BALANCE_KM,
        col = "grey",
        main = "Balance Road Length (KM)",
        ylab = "Kilometers")

vioplot(df1$STATE_GDP_CRORE,
        col = "violet",
        main = "State GDP (Crores)",
        ylab = "Crores")
```

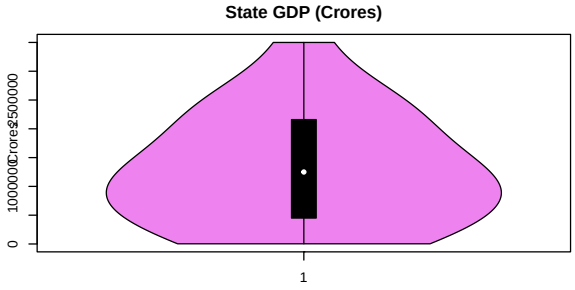












Except for the State GDP, all other variables in the violin plots are positively skewed, meaning most of the data values are small with a few very large . The State GDP, however, is more symmetrically distributed and not positively skewed.

2.3 Normality Check:

Use the Shapiro-Wilk test to check whether each continuous variable is normally distributed. Interpret the result: variables with a p-value less than 0.05 are not normally distributed.

```
shapiro.test(df1$LENGTH_OF_ROAD_WORK_SANCTIONED_KM)
```

```
    Shapiro-Wilk normality test

data:  df1$LENGTH_OF_ROAD_WORK_SANCTIONED_KM
W = 0.62652, p-value < 2.2e-16
```

```
shapiro.test(df1$COST_OF_WORKS_SANCTIONED_LAKHS)
```

```
    Shapiro-Wilk normality test

data:  df1$COST_OF_WORKS_SANCTIONED_LAKHS
W = 0.64274, p-value < 2.2e-16
```

```
shapiro.test(df1$LENGTH_OF_ROAD_WORK_COMPLETED_KM)
```

```
    Shapiro-Wilk normality test

data:  df1$LENGTH_OF_ROAD_WORK_COMPLETED_KM
W = 0.62501, p-value < 2.2e-16
```

```
shapiro.test(df1$EXPENDITURE_OCCURED_LAKHS)
```

```
    Shapiro-Wilk normality test

data:  df1$EXPENDITURE_OCCURED_LAKHS
W = 0.61601, p-value < 2.2e-16
```

```
shapiro.test(df1$LENGTH_OF_ROAD_WORK_BALANCE_KM)
```

```
    Shapiro-Wilk normality test

data:  df1$LENGTH_OF_ROAD_WORK_BALANCE_KM
W = 0.40586, p-value < 2.2e-16
```

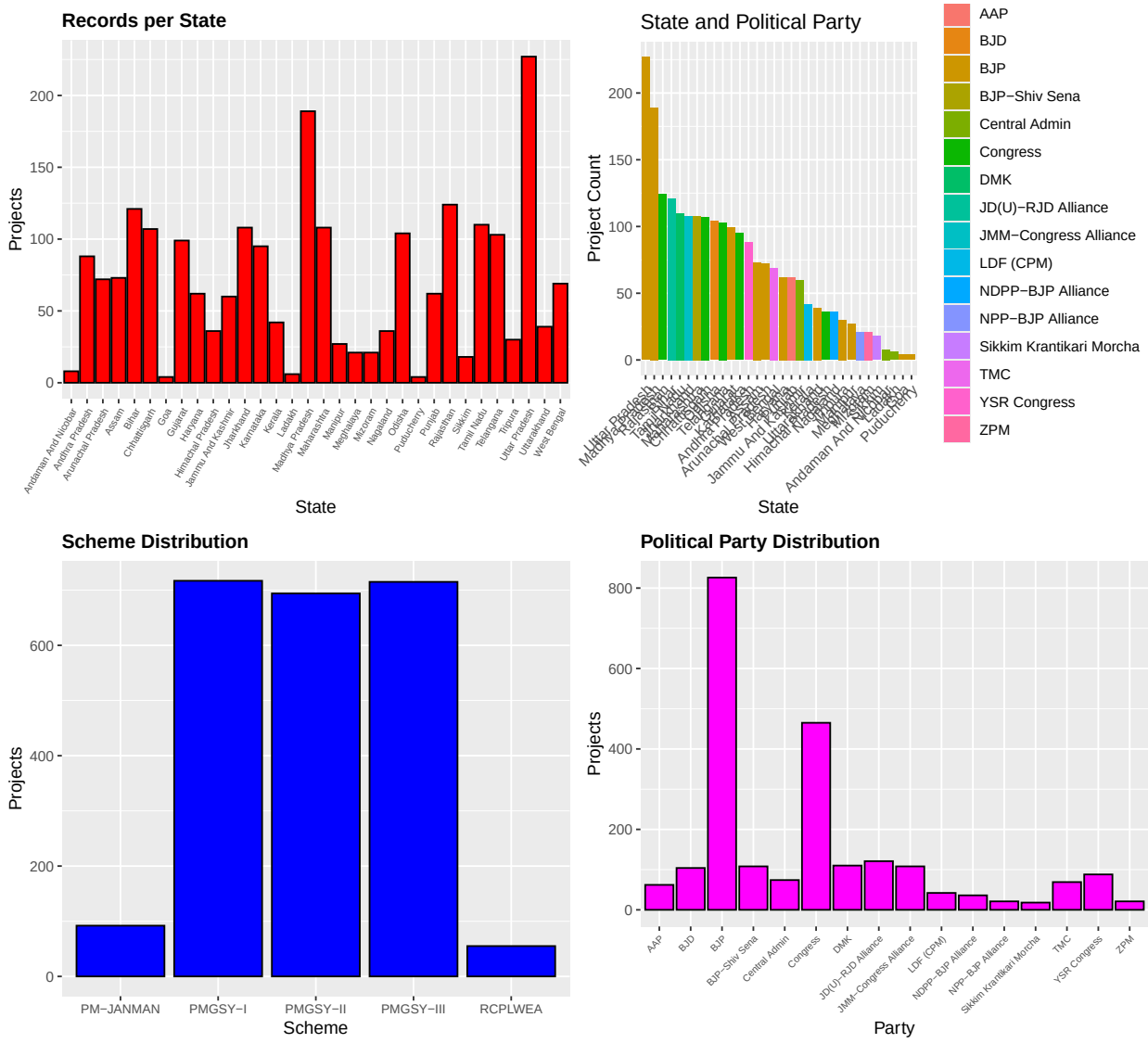
```
shapiro.test(df1$STATE_GDP_CRORE)
```

```
    Shapiro-Wilk normality test

data:  df1$STATE_GDP_CRORE
W = 0.92058, p-value < 2.2e-16
```

2.4 Exploration categorical varibles

```
library(ggplot2)
library(gridExtra)

# Plot 1: Records per State
p1 = ggplot(df1, aes(x = STATE_NAME)) +
  geom_bar(fill = "red", color = "black") +
  theme(axis.text.x = element_text(angle = 60,
                                   hjust = 1,
                                   size = 6),
        plot.title = element_text(size = 12,
                                  face = "bold")) +
  labs(title = "Records per State",
       x = "State",
       y = "Projects")

# Plot 2: Records per state into political party(bivariate plot)
p2 = df1 %>%
  group_by(STATE_NAME, POLITICAL_PARTY) %>%
  summarise(total_projects = n()) %>%
  ggplot(aes(x = reorder(STATE_NAME,
                         -total_projects),
             y = total_projects,
             fill = POLITICAL_PARTY)) +
  geom_col(position = "dodge") +
```

```
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title = "State and Political Party",
       x = "State", y = "Project Count")


# Plot 3: Scheme Distribution
p3 = ggplot(df1, aes(x = PMGSY_SCHEME)) +
  geom_bar(fill = "blue", color = "black") +
  theme(axis.text.x = element_text(size = 8),
        plot.title = element_text(size = 12,
                                  face = "bold")) +
  labs(title = "Scheme Distribution",
       x = "Scheme",
       y = "Projects")

# Plot 4: Political Party Distribution
p4 = ggplot(df1, aes(x = POLITICAL_PARTY)) +
  geom_bar(fill = "magenta", color = "black") +
  theme(axis.text.x = element_text(angle = 45,
                                   hjust = 1,
                                   size = 6),
        plot.title = element_text(size = 12,
                                  face = "bold")) +
  labs(title = "Political Party Distribution",
       x = "Party",
       y = "Projects")


grid.arrange(p1, p2, p3, p4, ncol = 2)
```



Number of Records per State:

Uttar Pradesh has the highest number of records (190+), followed by Madhya Pradesh, Bihar, Rajasthan, and Tamil Nadu, which also show a significantly large number of road project entries.

Some Research Questions:

- Do these high-record states—**Uttar Pradesh, Madhya Pradesh, Bihar, Rajasthan, and Tamil Nadu**—receive more road projects due to their **larger populations or greater infrastructure needs**?
- Is there a **systematic relationship** between the **ruling political party** in a state and the **number of sanctioned road projects**?
- Do economically **stronger states** (based on **State GDP**) receive **more projects**, or is there **equitable distribution** irrespective of economic size?

Record per State and Political Party:

I created a bivariate bar plot showing the number of projects in each state, colored by the ruling political party. This allows us to visually explore how political affiliation may relate to project distribution. The X-axis shows states, while the Y-axis shows project counts. Bars are grouped by party using geom_col(position = "dodge"), making comparisons clearer.

Scheme Distribution:

Most road project records belong to **PMGSY schemes (PMGSY-I, PMGSY-II, PMGSY-III)**, each with over 700 entries. The **PM-JANMAN and RCPLWEA** schemes have fewer than 200 entries each.

Political party Distribution:

Most road projects come from states governed by **BJP and Congress**. A smaller number of projects are associated with other political parties.

Full Forms of Political Parties

```
library(tibble)
library(knitr)
library(kableExtra)

# Create the party abbreviation table
party_abbr = tribble(
  ~Abbreviation, ~Full_Form,
  "AAP", "Aam Aadmi Party",
  "BJD", "Biju Janata Dal",
  "BJP", "Bharatiya Janata Party",
  "BJP-Shiv Sena", "BJP-Shiv Sena Alliance",
  "Central Admin", "Central Administration",
  "Congress", "Indian National Congress",
  "DMK", "Dravida Munnetra Kazhagam",
  "JD(U)-RJD Alliance", "Janata Dal (United)-
  Rashtriya Janata Dal Alliance",
  "JMM-Congress Alliance", "Jharkhand Mukti Morcha -
  Indian National Congress Alliance",
  "LDF (CPM)", "Left Democratic Front (Communist Party of India
  - Marxist)",
  "NDPP-BJP Alliance", "Nationalist Democratic Progressive
  Party- BJP Alliance",
  "NPP-BJP Alliance", "National People's Party - BJP Alliance",
  "Sikkim Krantikari Morcha", "Sikkim Krantikari Morcha",
  "TMC", "All India Trinamool Congress",
  "YSR Congress", "Yuvajana Sramika Rythu Congress Party",
  "ZPM", "Zoram People's Movement",
  "NA", "Not Assigned / Unknown"
)

# Render table with caption
```

```
kable(party_abbr,
      caption = "Abbreviations and Full Forms of
      Political Parties Used in Analysis",
      booktabs = TRUE) %>%
  kable_styling(
    full_width = FALSE,
    position = "center",
    latex_options = "hold_position"

  )
```

Table 2: Abbreviations and Full Forms of Political Parties Used in Analysis

Abbreviation	Full_Form
AAP	Aam Aadmi Party
BJD	Biju Janata Dal
BJP	Bharatiya Janata Party
BJP–Shiv Sena	BJP–Shiv Sena Alliance
Central Admin	Central Administration
Congress	Indian National Congress
DMK	Dravida Munnetra Kazhagam
JD(U)–RJD Alliance	Janata Dal (United)– Rashtriya Janata Dal Alliance
JMM–Congress Alliance	Jharkhand Mukti Morcha – Indian National Congress Alliance
LDF (CPM)	Left Democratic Front (Communist Party of India - Marxist)
NDPP–BJP Alliance	Nationalist Democratic Progressive Party– BJP Alliance
NPP–BJP Alliance	National People's Party – BJP Alliance
Sikkim Krantikari Morcha	Sikkim Krantikari Morcha
TMC	All India Trinamool Congress
YSR Congress	Yuvajana Sramika Rythu Congress Party
ZPM	Zoram People's Movement
NA	Not Assigned / Unknown

State-wise Analysis of PMGSY Expenditure and Political Party Representation on the Indian Map

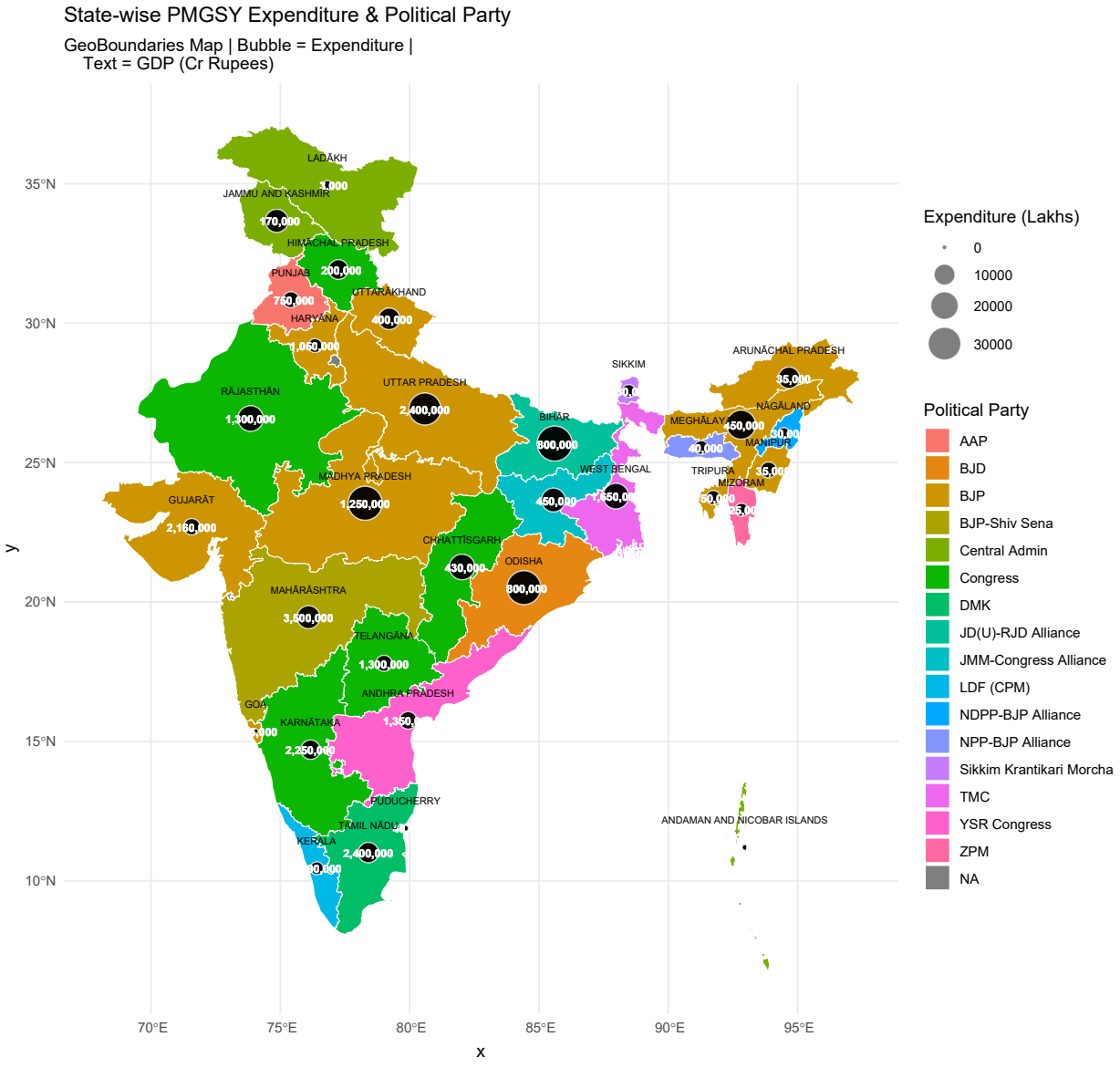
```
# Load libraries
library(sf)
library(ggplot2)
library(dplyr)
library(rgeoboundaries)

# Load India map from GeoBoundaries and simplify
india_map = geoboundaries("India", adm_lvl = "adm1")
```

```r
india_map = st_simplify(india_map, dTolerance = 500)
india_map$STATE_NAME = toupper(india_map$shapeName)

# Clean df1 STATE_NAME to match map
df1$STATE_NAME = toupper(df1$STATE_NAME)

df1$STATE_NAME = recode(df1$STATE_NAME,
  "ANDAMAN AND NICOBAR" = "ANDAMAN AND NICOBAR ISLANDS",
  "ARUNACHAL PRADESH"   = "ARUNĀCHAL PRADESH",
  "BIHAR"               = "BIHĀR",
  "CHHATTISGARH"        = "CHHATTĪSGARH",
  "GUJARAT"             = "GUJARĀT",
  "HARYANA"             = "HARYĀNA",
  "HIMACHAL PRADESH"    = "HIMĀCHAL PRADESH",
  "JAMMU AND KASHMIR"   = "JAMMU AND KASHMĪR",
  "JHARKHAND"           = "JHĀRKHAND",
  "KARNATAKA"           = "KARNĀTAKA",
  "MAHARASHTRA"         = "MAHĀRĀSHTRA",
  "MEGHALAYA"           = "MEGHĀLAYA",
  "NAGALAND"            = "NĀGĀLAND",
  "ODISHA"              = "ODISHA",
  "RAJASTHAN"           = "RĀJASTHĀN",
  "TAMIL NADU"          = "TAMIL NĀDU",
  "TELANGANA"           = "TELANGĀNA",
  "UTTARAKHAND"         = "UTTARĀKHAND",
  "LADAKH"              = "LADĀKH",
  "PUDUCHERRY"          = "PUDUCHERRY"
)

# Define Unicode GDP and party vectors
state_gdp = c(
  "ANDAMAN AND NICOBAR ISLANDS" = 8000,
  "ANDHRA PRADESH" = 1350000,
  "ARUNĀCHAL PRADESH" = 35000,
  "ASSAM" = 450000,
  "BIHĀR" = 800000,
  "CHHATTĪSGARH" = 430000,
  "DELHI" = 1100000,
  "GOA" = 85000,
  "GUJARĀT" = 2160000,
  "HARYĀNA" = 1050000,
  "HIMĀCHAL PRADESH" = 200000,
  "JAMMU AND KASHMĪR" = 170000,
  "JHĀRKHAND" = 450000,
  "KARNĀTAKA" = 2250000,
  "KERALA" = 1090000,
```

```
  "LADĀKH" = 3000,
  "MADHYA PRADESH" = 1250000,
  "MAHĀRĀSHTRA" = 3500000,
  "MANIPUR" = 35000,
  "MEGHĀLAYA" = 40000,
  "MIZORAM" = 25000,
  "NĀGĀLAND" = 30000,
  "ODISHA" = 800000,
  "PUNJAB" = 750000,
  "RĀJASTHĀN" = 1300000,
  "SIKKIM" = 30000,
  "TAMIL NĀDU" = 2400000,
  "TELANGĀNA" = 1300000,
  "TRIPURA" = 50000,
  "UTTAR PRADESH" = 2400000,
  "UTTARĀKHAND" = 400000,
  "WEST BENGAL" = 1650000,
  "PUDUCHERRY" = 40000
)

political_party = c(
  "ANDAMAN AND NICOBAR ISLANDS" = "Central Admin",
  "ANDHRA PRADESH" = "YSR Congress",
  "ARUNĀCHAL PRADESH" = "BJP",
  "ASSAM" = "BJP",
  "BIHĀR" = "JD(U)-RJD Alliance",
  "CHHATTĪSGARH" = "Congress",
  "DELHI" = "AAP",
  "GOA" = "BJP",
  "GUJARĀT" = "BJP",
  "HARYĀNA" = "BJP",
  "HIMĀCHAL PRADESH" = "Congress",
  "JAMMU AND KASHMĪR" = "Central Admin",
  "JHĀRKHAND" = "JMM-Congress Alliance",
  "KARNĀTAKA" = "Congress",
  "KERALA" = "LDF (CPM)",
  "LADĀKH" = "Central Admin",
  "MADHYA PRADESH" = "BJP",
  "MAHĀRĀSHTRA" = "BJP-Shiv Sena",
  "MANIPUR" = "BJP",
  "MEGHĀLAYA" = "NPP-BJP Alliance",
  "MIZORAM" = "ZPM",
  "NĀGĀLAND" = "NDPP-BJP Alliance",
  "ODISHA" = "BJD",
  "PUNJAB" = "AAP",
  "RĀJASTHĀN" = "Congress",
```

```
  "SIKKIM" = "Sikkim Krantikari Morcha",
  "TAMIL NĀDU" = "DMK",
  "TELANGĀNA" = "Congress",
  "TRIPURA" = "BJP",
  "UTTAR PRADESH" = "BJP",
  "UTTARĀKHAND" = "BJP",
  "WEST BENGAL" = "TMC",
  "PUDUCHERRY" = "BJP"
)

# Add GDP and Party columns
df1$STATE_GDP_CRORE = state_gdp[df1$STATE_NAME]
df1$POLITICAL_PARTY = political_party[df1$STATE_NAME]

# Summarize expenditure by state
df_summary = df1 %>%
  group_by(STATE_NAME, POLITICAL_PARTY) %>%
  summarise(
    Total_Expenditure = sum(EXPENDITURE_OCCURED_LAKHS, na.rm = TRUE),
    STATE_GDP_CRORE = first(STATE_GDP_CRORE),
    .groups = "drop"
  )

# Merge with map
map_data = india_map %>%
  left_join(df_summary, by = "STATE_NAME")

# Compute centroids
centroids = st_centroid(map_data)
centroids_clean = centroids %>%
  filter(!is.na(Total_Expenditure))
centroids_clean$GDP_LABEL = format(centroids_clean$STATE_GDP_CRORE,
                                   big.mark = ",",
                                   scientific = FALSE)

# Final plot
ggplot() +
  geom_sf(data = map_data, aes(fill = POLITICAL_PARTY), color = "white") +
  geom_sf(data = centroids_clean, aes(size = Total_Expenditure),
          shape = 21, fill = "black", color = "white", alpha = 0.5) +
  geom_sf_text(data = centroids_clean, aes(label = GDP_LABEL),
               size = 2.5, color = "white", fontface = "bold") +
  geom_sf_text(data = centroids_clean, aes(label = STATE_NAME),
               size = 2.2, color = "black",
               nudge_y = 1.0, check_overlap = TRUE) +
  scale_size_continuous(range = c(1, 10), name = "Expenditure (Lakhs)") +
```

```
  labs(
    title = "State-wise PMGSY Expenditure & Political Party",
    subtitle = "GeoBoundaries Map | Bubble = Expenditure |
    Text = GDP (Cr Rupees)",
    fill = "Political Party"
  ) +
  guides(size = guide_legend(override.aes = list(fill = "black"))) +
  theme_minimal()
```



Relation between numerical variable

```r
library(dplyr)
library(corrplot)

df_numeric = df1 %>% select(where(is.numeric))

# Compute correlation matrix
cor_matrix = cor(df_numeric, use = "complete.obs")

# Plot correlation matrix
corrplot(
  cor_matrix,
  method = "color",
  type = "upper",
  tl.col = "black",
  tl.cex = 0.8,
  title = "Correlation Matrix of Numeric Variables",
  mar = c(0, 0, 1, 0)
)
```

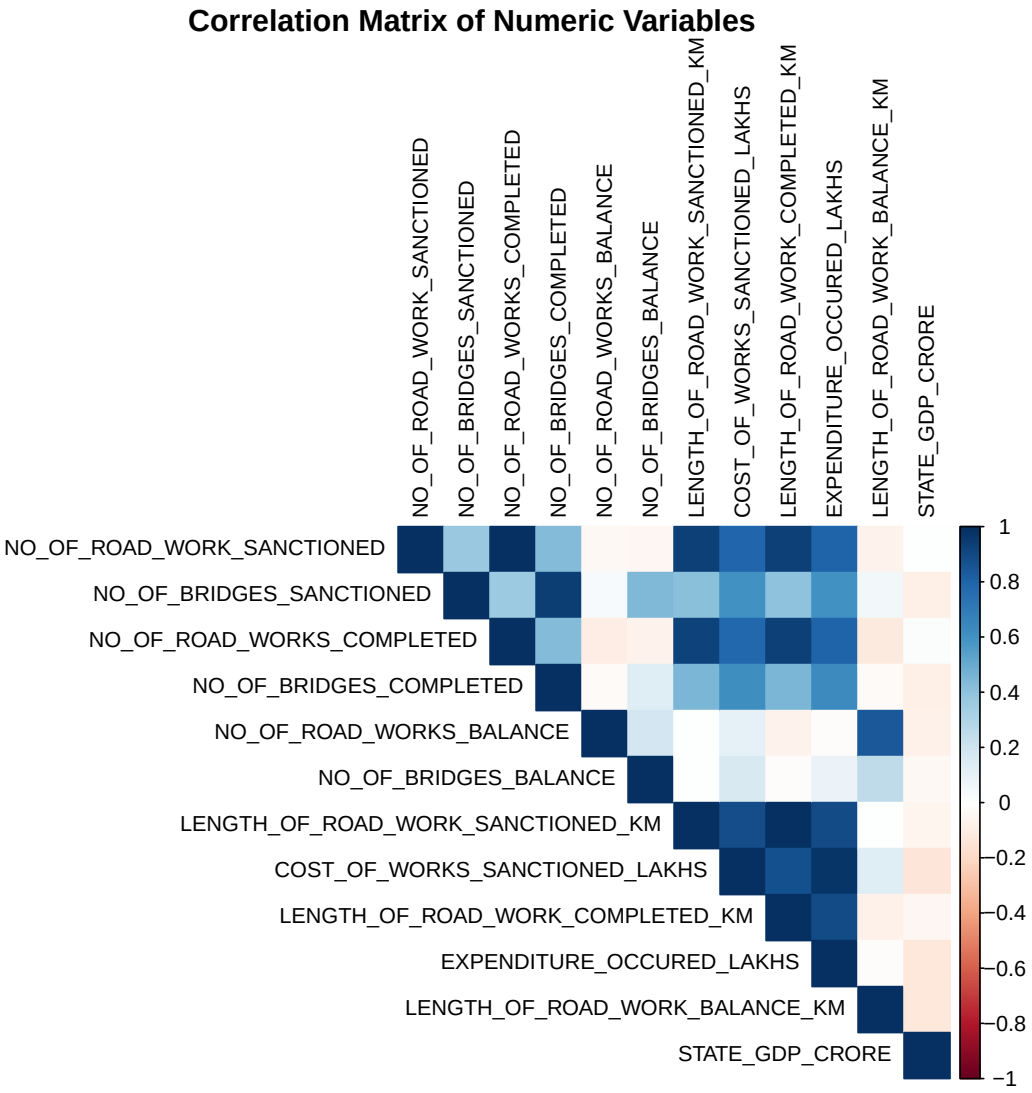


Figure 1: Correlation matrix and scatterplot matrix

```
# Scatterplot matrix
pairs(
  df_numeric,
  main = "Scatterplot Matrix of Numeric Variables",
  pch = 21,
  bg = "lightblue"
)
```

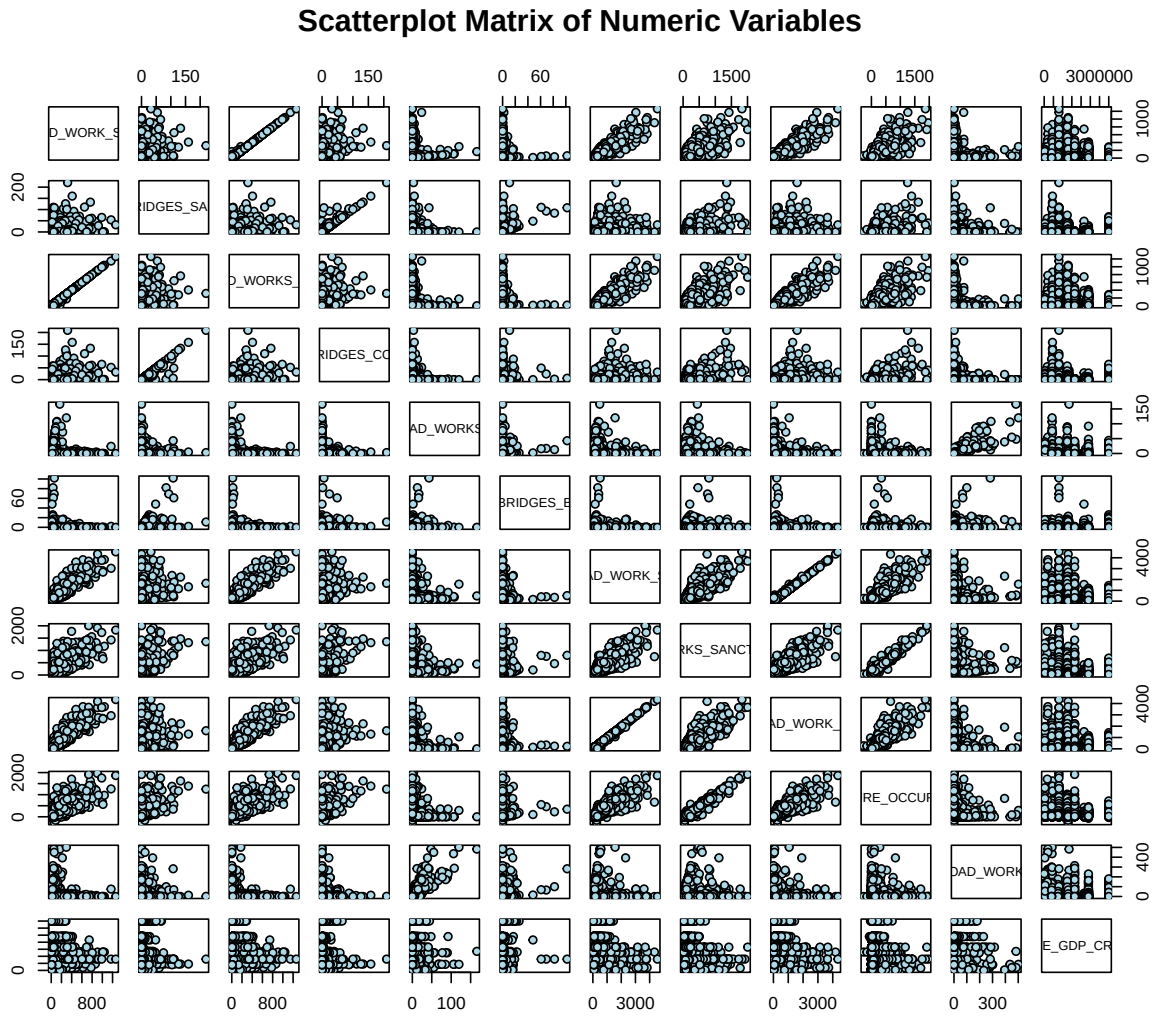


Figure 2: Correlation matrix and scatterplot matrix

Correlation Analysis

```
cor(df1$NO_OF_ROAD_WORK_SANCTIONED,
    df1$EXPENDITURE_OCCURED_LAKHS)
```

```
[1] 0.8058767
```

```
cor(df1$NO_OF_BRIDGES_SANCTIONED,
    df1$EXPENDITURE_OCCURED_LAKHS)
```

```
[1] 0.6000026
```

```
cor(df1$NO_OF_ROAD_WORKS_COMPLETED,
    df1$EXPENDITURE_OCCURED_LAKHS)
```

```
[1] 0.8035482
```

```
cor(df1$NO_OF_BRIDGES_COMPLETED,
    df1$EXPENDITURE_OCCURED_LAKHS)
```

```
[1] 0.6334624
```

```
cor(df1$LENGTH_OF_ROAD_WORK_COMPLETED_KM,
    df1$EXPENDITURE_OCCURED_LAKHS)
```

```
[1] 0.895096
```

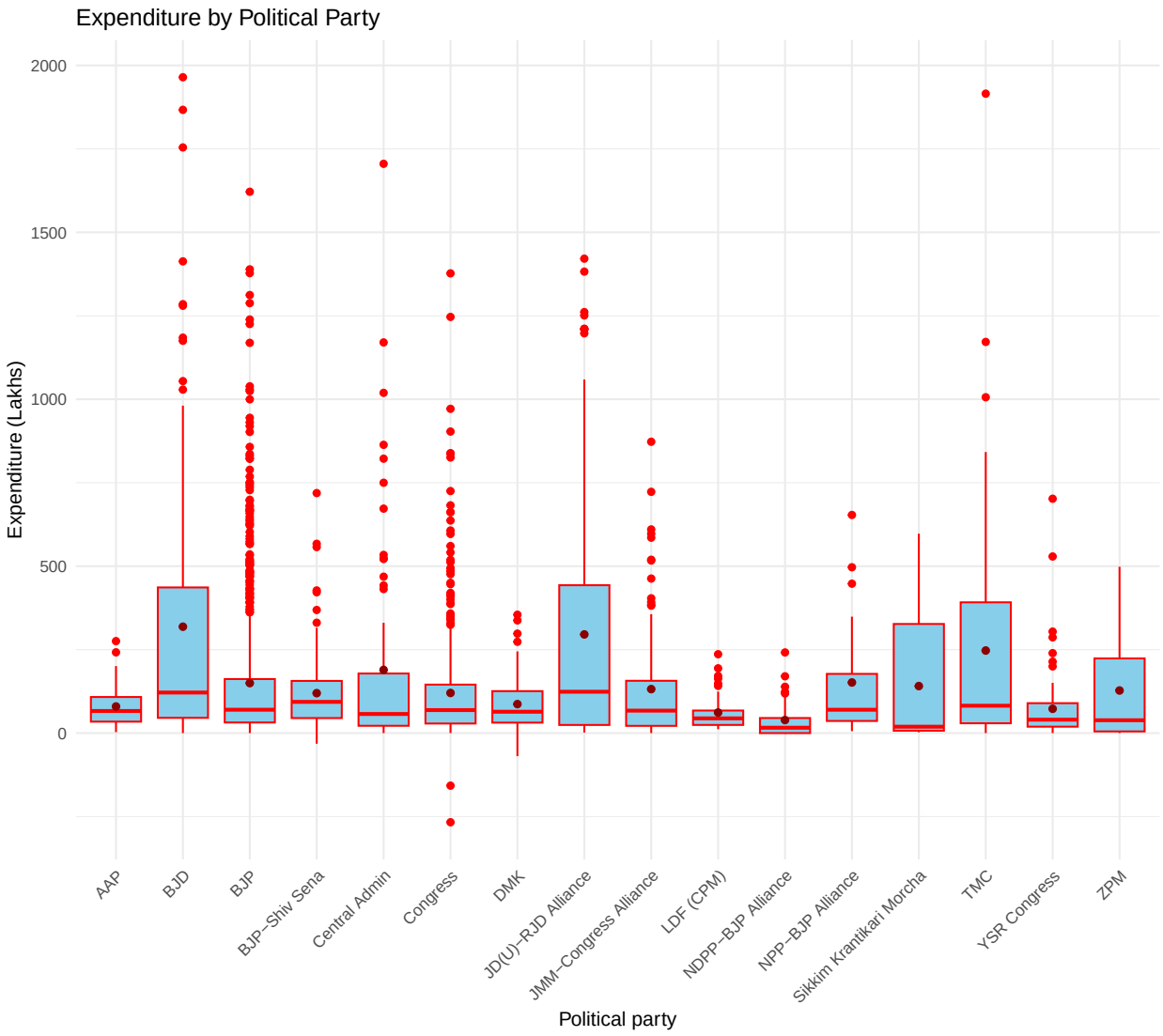
```
cor(df1$LENGTH_OF_ROAD_WORK_SANCTIONED_KM,
    df1$EXPENDITURE_OCCURED_LAKHS)
```

```
[1] 0.8947092
```

All the predictor variables demonstrate a strong or moderate **positive correlation** with the response variable **EXPENDITURE_OCCURED LAKHS**. Notably, road length (both sanctioned and completed) shows very high correlation ($\rho > 0.89$) , indicating that longer roads are directly associated with higher expenditures. This validates the choice of expenditure as the response variable in a predictive regression model.

Relation between Numerical variable and Categorical variable

```
ggplot(df1, aes(x = POLITICAL_PARTY,
                y = EXPENDITURE_OCCURED_LAKHS)) +
  geom_boxplot(fill = "skyblue", color = "red") +
  stat_summary(fun = mean, geom = "point",
               shape = 20,
               size = 2.5,
               color = "darkred") +
  theme_minimal() +
  labs(
    title = "Expenditure by Political Party",
    x = "Political party",
    y = "Expenditure (Lakhs)"
  ) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

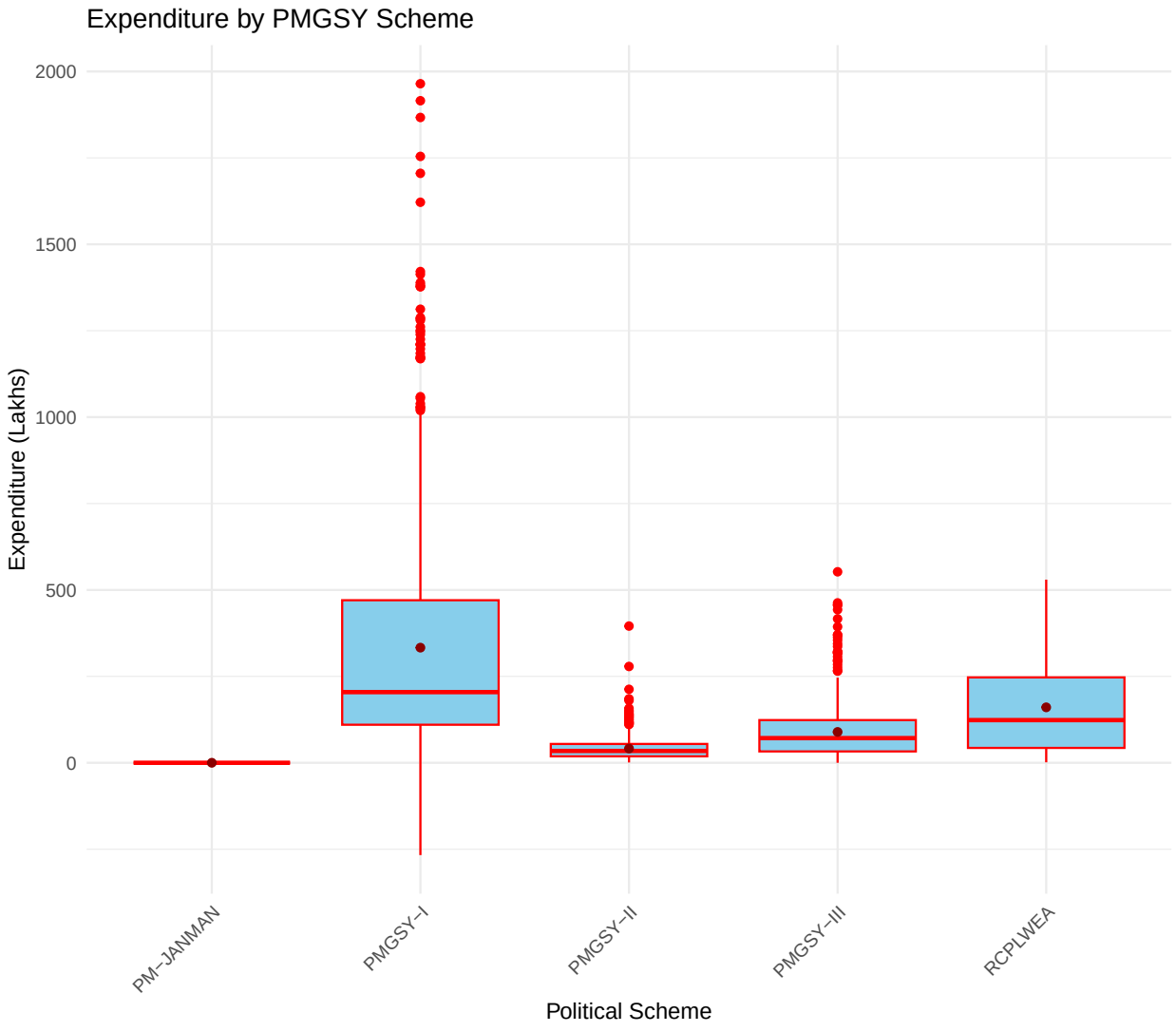


- **BJD, BJP, TMC, DMK,** and the **JD(U)-RJD Alliance** exhibit higher median expenditures and wider interquartile ranges, reflecting both moderate and large-scale projects.
- The **BJP** shows many high-value outliers, indicating several projects with significantly high costs.
- **JD(U)-RJD Alliance** has a high median and large spread, suggesting a diverse project range.
- **DMK** demonstrates mid-to-high range spending with outliers.
- In contrast`LDF (CPM)`,`ZPM`, and `NDPP−BJP` Alliance have lower and more consistent spending, shown by narrower boxes and lower medians.

Relation between Numerical variable and Categorical variable

```
ggplot(df1, aes(x = PMGSY_SCHEME,
                y = EXPENDITURE_OCCURED_LAKHS)) +
  geom_boxplot(fill = "skyblue",
```

```
              color = "red") +
  stat_summary(fun = mean, geom = "point",
               shape = 20,
               size = 2.5,
               color = "darkred") +
  theme_minimal() +
  labs(
    title = "Expenditure by PMGSY Scheme",
    x = "Political Scheme",
    y = "Expenditure (Lakhs)"
  ) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



The box plot compares project expenditures under various PMGSY schemes. PMGSY-I shows the highest median expenditure and a wide distribution, with numerous outliers, indicating both moderate and high-value projects were undertaken. RCPLWEA and PMGSY-III reflect moderate median

spending with some variability, suggesting a mix of consistent and regionally scaled investments. PMGSY-II shows relatively lower and more consistent expenditures with fewer high-value outliers. In contrast, PM-JANMAN exhibits minimal to no recorded spending, possibly due to limited implementation or data availability.

About initial_split() :

- We used the **initial_split()** function from the **rsample package** to divide the dataset into **training** and **testing** sets before applying model-based techniques like **LASSO** and **Ridge.**
- This function ensures that a random 70% of the data is used for training and the remaining 20% for validation(testing).
- It helps us avoid **overfitting** and evaluate the model's **generalization performance** on unseen data.

3 Modeling

```
library(rsample)

data=df1
head(data$EXPENDITURE_OCCURED_LAKHS)
```

```
[1]   4.1501  11.6894   8.6343   0.0000  10.2977 100.7436
```

```
splite =  initial_split(data ,prop = 0.7 ,
                        strata = "EXPENDITURE_OCCURED_LAKHS")
train =  training(splite)
test  =  testing(splite)
```

3.1 Linear Model and Overfitting

```
model = lm(EXPENDITURE_OCCURED_LAKHS ~ . , data = train)
broom::tidy(model)
```

```
# A tibble: 755 x 5
   term                          estimate std.error statistic  p.value
   <chr>                            <dbl>     <dbl>     <dbl>    <dbl>
 1 (Intercept)                     -11.4       18.6   -0.611  0.541
 2 STATE_NAMEANDHRA PRADESH          3.32      35.4    0.0938 0.925
 3 STATE_NAMEARUNĀCHAL PRADESH       9.73      28.0    0.347  0.728
 4 STATE_NAMEASSAM                  14.1       25.2    0.559  0.576
```

```
 5 STATE_NAMEBIHĀR                153.        45.5    3.35   0.000840
 6 STATE_NAMECHHATTĪSGARH          52.3       45.4    1.15   0.249
 7 STATE_NAMEGOA                   -6.14      35.4   -0.173  0.862
 8 STATE_NAMEGUJARĀT               10.9       28.0    0.389  0.697
 9 STATE_NAMEHARYĀNA               -2.31      28.0   -0.0825 0.934
10 STATE_NAMEHIMĀCHAL PRADESH     -65.1       66.1   -0.985  0.325
# i 745 more rows
```

```
train_pred= predict(model ,data =train)
test_pred = predict(model ,data =test)
RMSE_train = sqrt(mean
                  (
  (
    train$EXPENDITURE_OCCURED_LAKHS - train_pred)^2
                    )
  )
RMSE_test = sqrt(mean(
  (
    test$EXPENDITURE_OCCURED_LAKHS - test_pred)^2
  )
  )

RMSE_train
```

```
[1] 22.78391
```

```
RMSE_test
```

```
[1] 329.7
```

My data showed signs of overfitting when using a linear model (lm). To reduce overfitting and improve generalization on unseen data, I applied a regularized regression method (e.g., Lasso or Ridge)

3.2 Variable Transformations

3.2.1 Box-Cox and Log Transformations

```
library(dplyr)
library(tidyverse)
library(MASS)
#model matrix(for lasso and ridge)
```

```
X = model.matrix(EXPENDITURE_OCCURED_LAKHS ~ .,
                 data = train)[,-1]

Y = df1$EXPENDITURE_OCCURED_LAKHS

#Ensure positive for BOX_COX

min_val = min(Y,na.rm = TRUE)

# Shift Y only if any value is 0 or negative

if(min_val <= 0){
  Y_shifted  = Y-min_val +1
}else{
  Y_shifted = Y
}
library(MASS)
#Apply boxcox transformation
boxcox(lm(Y_shifted~1),xlab = "log(Expenditure + 1)")
```

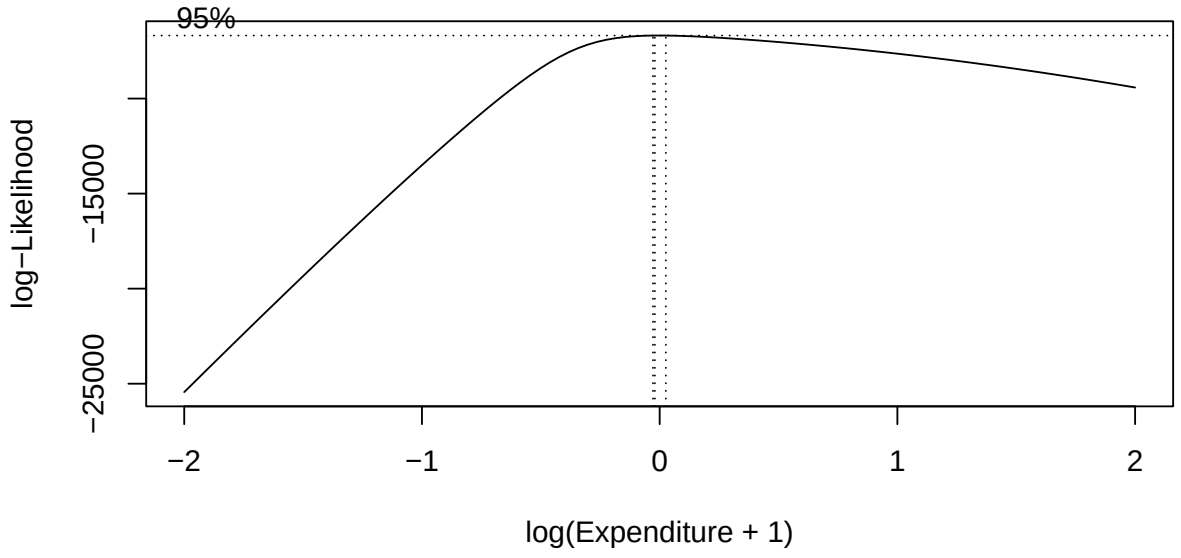


Figure 3: The log(x + 1) transformation reduces skewness in expenditure data.

```
#Apply log(x+1) transformation
Y = log(train$EXPENDITURE_OCCURED_LAKHS+1 )
par(mfrow = c(1, 2))
```

```
hist(train$EXPENDITURE_OCCURED_LAKHS,
     main = "Original",
     xlab = "Expenditure (Lakhs)")
hist(Y, main = "Log-transformed",
     xlab = "log(Expenditure + 1)")
```

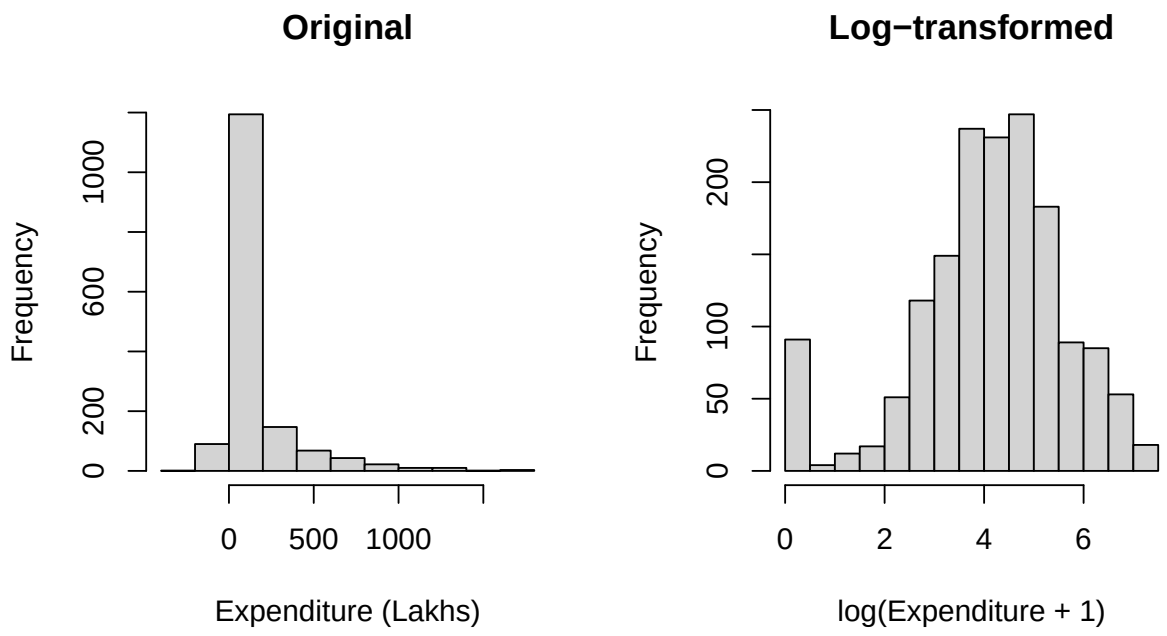


Figure 4: The log(x + 1) transformation reduces skewness in expenditure data.

```
sum(is.na(X))
```

```
[1] 0
```

```
sum(is.na(Y))
```

```
[1] 4
```

The Box-Cox transformation was applied to the expenditure variable to assess the optimal transformation for normality. The log-likelihood curve peaked at $\lambda \approx 0$, indicating that a logarithmic transformation is most suitable. This confirms that applying log(x + 1) is a reasonable choice for reducing skewness.

3.2.2 Histograms of Log-Transformed Variables

```
# Other Continuous variables
vars  = c("COST_OF_WORKS_SANCTIONED_LAKHS",
          "LENGTH_OF_ROAD_WORK_COMPLETED_KM",
          "LENGTH_OF_ROAD_WORK_SANCTIONED_KM",
          "STATE_GDP_CRORE")

# Define labels
labels  = c("Sanctioned Cost",
            "Completed Length",
            "Sanctioned Length",
            "State GDP")

par(mfrow = c(2, 4), mar = c(4, 4, 2.5, 1),
    cex.main = 0.9,
    cex.lab = 0.9)

for (i in seq_along(vars)) {
  v = vars[i]
  label = labels[i]

  original = df1[[v]]
  log_trans = log(original + 1)

  # Original histogram
  hist(original,
       main = paste("Original:", label),
       col = "skyblue",
       xlab = label)

  # Log-transformed histogram
  hist(log_trans,
       main = paste("Log-Transformed:", label),
       col = "orange",
       xlab = paste("log(1 +", label, ")"))
}
```

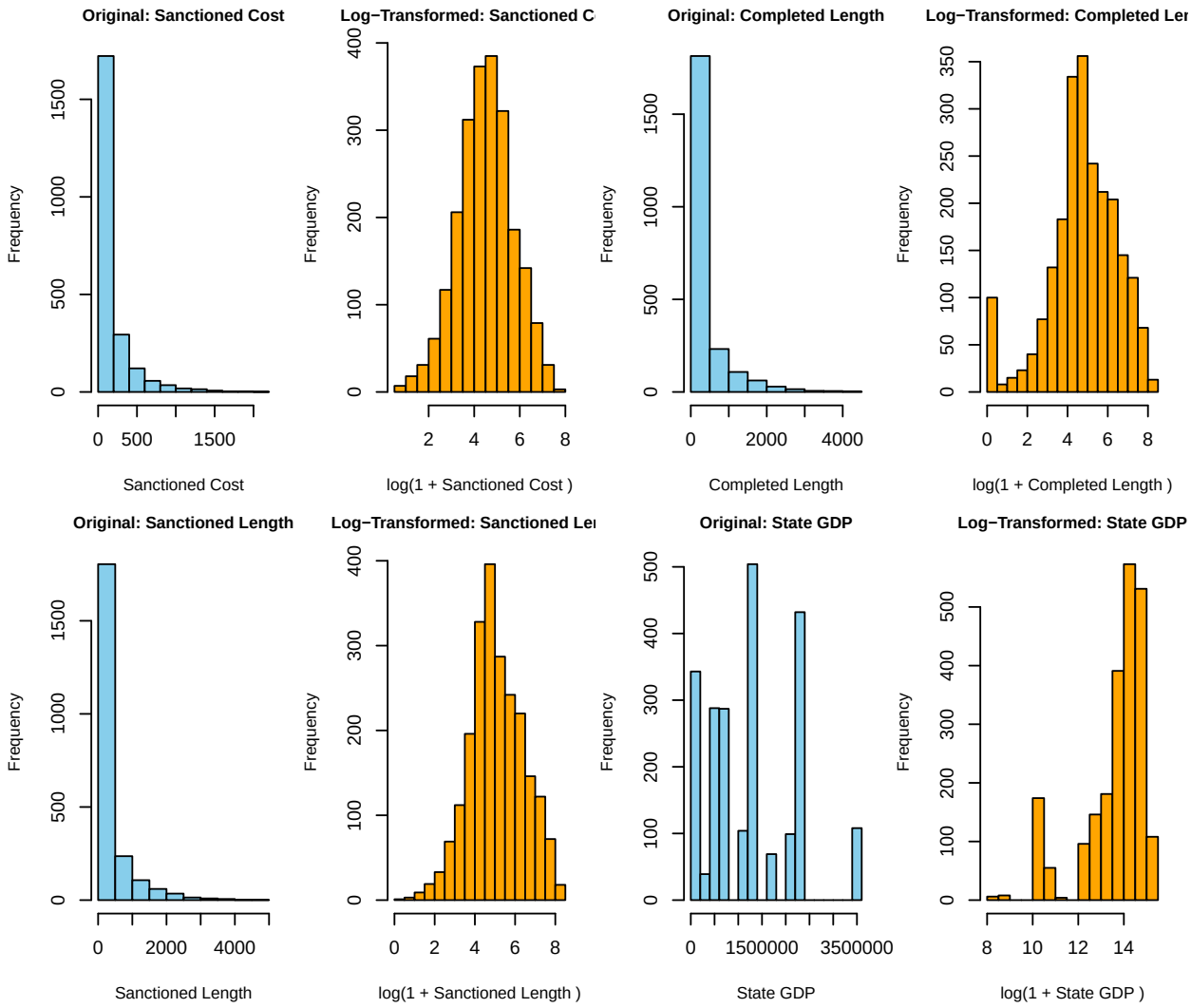


Figure 5: Histograms of original and log-transformed continuous variables. The log(x+1) transformation helps reduce skewness and makes the variables more suitable for modeling.

3.2.3 Box-Cox Plots for Multiple Variables

```
library(MASS)

vars = c("COST_OF_WORKS_SANCTIONED_LAKHS",
         "LENGTH_OF_ROAD_WORK_COMPLETED_KM",
         "LENGTH_OF_ROAD_WORK_SANCTIONED_KM",
         "STATE_GDP_CRORE")

labels = c("Sanctioned Cost (Lakhs)",
           "Completed Length (KM)",
           "Sanctioned Length (KM)",
```

```
            "State GDP (Crore)")

par(mfrow = c(2, 2), mar = c(4, 4, 3, 1))

# Loop through each variable and create Box-Cox plot
for (i in seq_along(vars)) {
  var_name = vars[i]
  label = labels[i]

  y = df1[[var_name]]

  # Ensure all values are positive for Box-Cox
  if (min(y, na.rm = TRUE) <= 0) {
    y = y - min(y, na.rm = TRUE) + 1
  }

  boxcox(lm(y ~ 1),
         main = paste("Box-Cox for", label),
         xlab = expression(lambda),
         ylab = "Log-Likelihood")
}
```

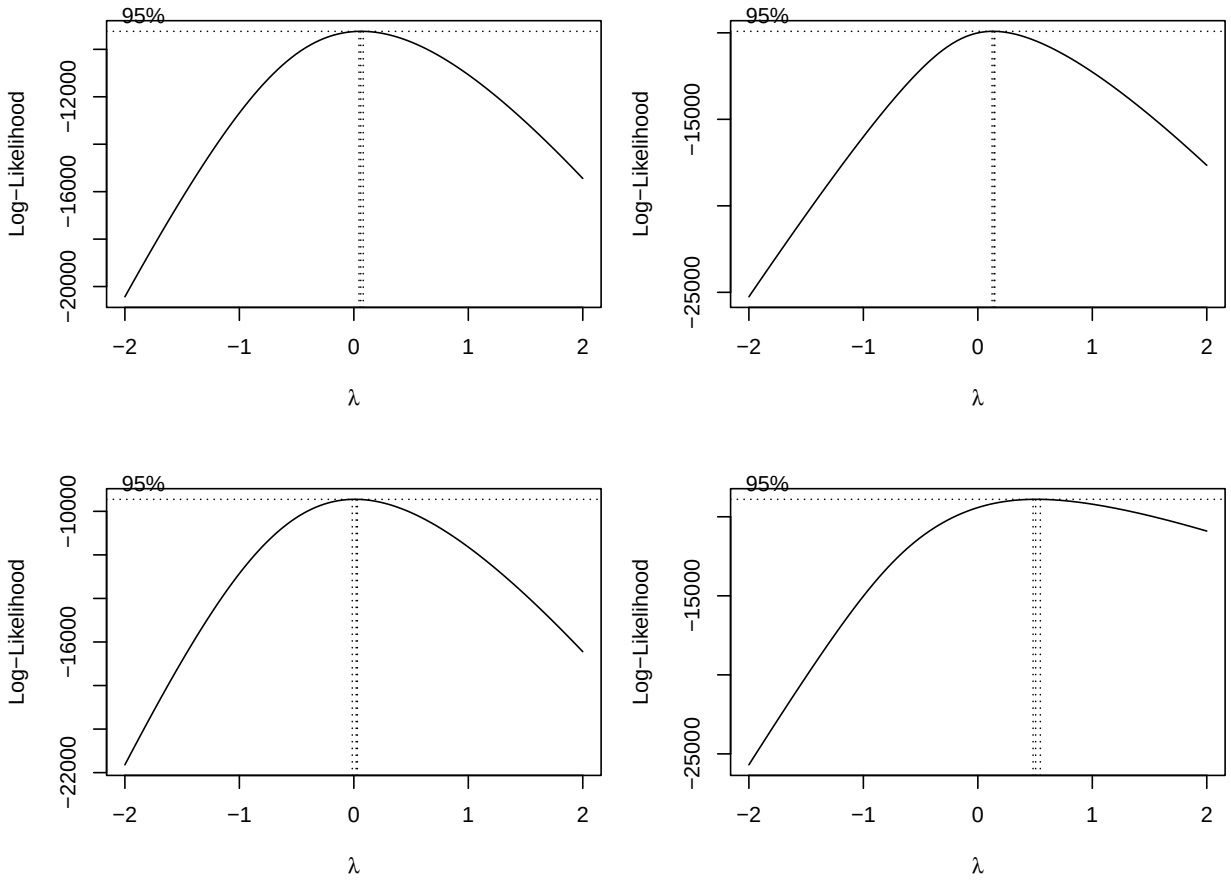


Figure 6: Box-Cox plots for four continuous variables. Variables like Sanctioned Cost, Completed Length, and Sanctioned Length have optimal lambda ≈ 0 (suggesting log transformation), while State GDP shows λ ≈ 0.5 (square-root transformation preferred).

A Box-Cox transformation was applied to assess the optimal transformation for key continuous variables. The variables **Sanctioned Cost**, **Completed Length**, and **Sanctioned Length** all exhibited an optimal lambda (λ) close to 0, supporting the use of a log transformation of the form $\log(x + 1)$.

The variable **State GDP** showed a lambda around 0.5–0.7, suggesting that a square root transformation (i.e., $x^{0.5}$) or a general Box-Cox transformation would be more appropriate than a simple log.

These transformations were applied to reduce skewness and approximate normality, which helps improve the performance of model regularization techniques such as **Lasso regression**.

3.2.4 Handling Missing Values After Transformation

```
Y = log(train$EXPENDITURE_OCCURED_LAKHS+1 )
sum(is.na(Y))
```

```
[1] 4
```

```
 # Create a logical index of non-missing rows in Y
complete_idx = complete.cases(Y)
# Keep only the rows in X where Y is not missing
X = X[complete_idx, ]
# Keep only the non-missing values in Y
Y = Y[complete_idx]
sum(is.na(Y))
```

```
[1] 0
```

Handling Missing Values from Log Transformation

- While applying the log transformation to the `EXPENDITURE_OCCURED_LAKHS` variable using `log(x + 1)`, we observed that 4 values became `NA`. This is likely due to pre-existing missing or zero values in the original expenditure column.
- To ensure clean inputs to the LASSO model, we used `complete.cases()` to filter out rows where either the predictors or the response were missing. This step is critical when working with `glmnet`, which does not handle `NA` values.
- After this filtering, the dataset used for modeling was free of missing values and ready for regularization-based training.
- Both log and Box-Cox transformations were applied to address skewness in continuous variables.Missing values were removed to ensure a complete data set. Categorical predictors were encoded using the **model.matrix()** function, facilitating compatibility with regularization techniques. The final model was fitted using Lasso regression, which performs both variable selection and regularization.

3.3 Regularization Models

3.3.1 Ridge Regression

```
library(glmnet)

# Apply ridge regression
ridge = glmnet(x= X ,
               y= Y ,
               alpha = 0)
```

```
plot(ridge ,xvar = "lambda")


#Lambda applies to penality parameter
ridge$lambda %>% head()

#Small Lambda results  in large coef
coef(ridge)[c("NO_OF_BRIDGES_SANCTIONED",
              "LENGTH_OF_ROAD_WORK_COMPLETED_KM"),
            100]

#Large Lambda results in small coef
coef(ridge)[c("NO_OF_BRIDGES_SANCTIONED",
              "LENGTH_OF_ROAD_WORK_COMPLETED_KM"),
            1]
```

Find optimal lambda value using k fold croos validation

```
#Apply CV ridge regression
ridge = cv.glmnet(x = X ,y = Y ,alpha=0)
plot(ridge ,main = "Ridge Penalty\n\n")
```

plot results: Ridge regression does not force any variables to exactly zero so-all features will remain in the model but we see the number of variables retained in the lasso model decrease as the penalty increases.

3.3.2 Lasso Regression

```
library(glmnet)
lasso = cv.glmnet(
  x = X,
  y = Y,
  alpha = 1
)

# Plot CV error vs log(lambda)
plot(lasso, main = "Lasso penalty\n\n)")
```

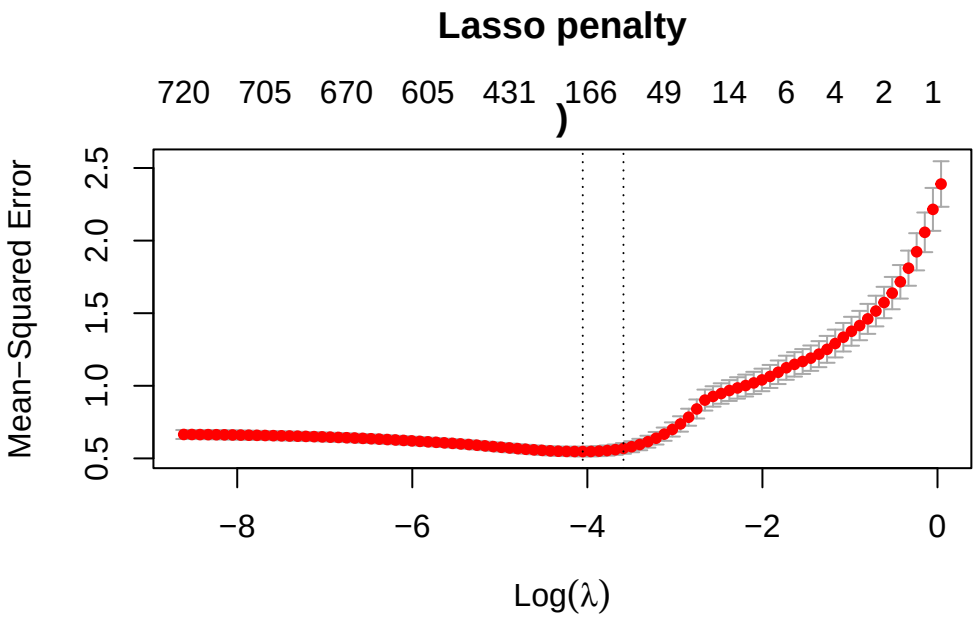


Figure 7: Lasso Regression: Cross-validated Mean Squared Error (MSE) across different lambda values. The first vertical line indicates the lambda with minimum MSE, and the second shows 1-SE rule.

- 10-fold CV MSE for a ridge and lasso model.
- First dotted vertical line in each plot represents the lambda with the samllest MSE and the second represents the lambda with an MSE with in one standard error of the minimum MSE

```
summary(ridge)
```

3.3.3 Comparing Ridge and Lasso (Plot and MSE)

```
#ridge model
min(ridge$cvm) #minimum MSE
ridge$lambda.min #lambda for this min MSE
ridge$cvm [ridge$lambda == ridge$lambda.1se] #1-SE rule
ridge$lambda.1se #lambda for this MSE
```

- The minimum MSE for our ridge model is 1.110118(produced when lambda = 0.1028479)

```
#Lasso model
min(lasso$cvm) #
```

```
[1] 0.5471588
```

```
lasso $lambda.min
```

```
[1] 0.01737841
```

```
lasso $ cvm[lasso$lambda == lasso$lambda.1se]
```

```
[1] 0.5685903
```

```
lasso$lambda.1se
```

```
[1] 0.02767134
```

- Whereas the minimum MSE for our lasso model is 0.6642386(produced when lambda = 0.01563196)

```
#Ridge model
ridge_min = glmnet(
  x = X ,
  y = Y ,
  alpha = 0
)

par(mfrow = c(1,2))
plot(ridge_min ,xvar = "lambda" ,main = "Ridge penalty\n\n")
abline(v=log(ridge$lambda.min) ,col = "red"  ,lty ="dashed")
abline(v=log(ridge$lambda.1se) ,col = "blue" ,lty ="dashed")
```

```
#Lasso model
lasso_min = glmnet(
  x = X ,
  y = Y ,
  alpha = 1
)
plot(lasso_min ,xvar = "lambda" ,main ="Lasso penalty\n\n")
abline(v = log(lasso$lambda.min),col ="red" ,lty  ="dashed")
abline(v = log(lasso$lambda.1se ),col= "blue",lty= "dashed")
```

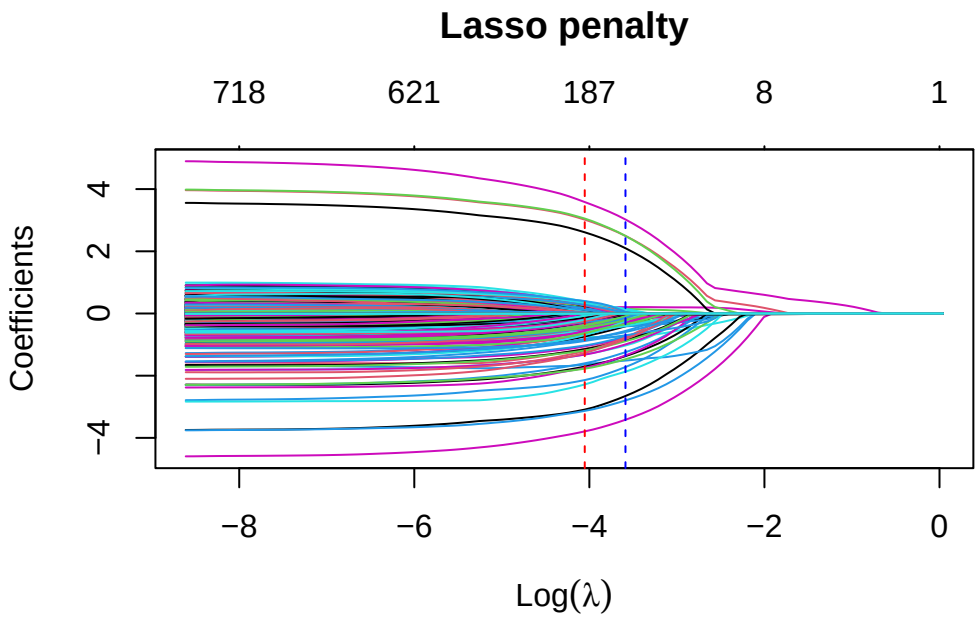


Figure 8: Lasso Model: Coefficient shrinkage path across lambda values. Red dashed line = lambda.min (minimum CV error), Blue dashed line = lambda.1se (simpler model with comparable error).

The dashed red line represent largest lambda value that falls with in one standard error of the minimum MSE Coefficient for our ridge and lasso models. first dotted vertical line in each plot represents the lambda with the smallest MSE and the second represent the lambda with an MSE with one standard error of the minimum MSE

3.3.4 Elastic Net Regression (alpha = 0.25 and 0.75)

```
#ELASTIC NET MODEL
elastic_net = glmnet(x = X , y = Y ,alpha = .25)
elastic_net =glmnet(x =X, y =Y ,alpha = 0.75)

# Set up 2x2 plot layout
par(mfrow = c(2, 2))

#  Plot Lasso with cross-validation (cv.glmnet object)
plot(lasso_min, xvar = "lambda",
     main = "Lasso (alpha = 1)\n\n")

#  Elastic Net with alpha = 0.25 (glmnet object)
Elastic_Net_min_25 = glmnet(x = X, y = Y, alpha = 0.25)
plot(Elastic_Net_min_25, xvar = "lambda",
     main = "Elastic Net (alpha = 0.25)\n\n")
```

```
# Elastic Net with alpha = 0.75 (glmnet object)
Elastic_Net_min_75 = glmnet(x = X, y = Y, alpha = 0.75)
plot(Elastic_Net_min_75, xvar = "lambda",
     main = "Elastic Net (alpha = 0.75)\n\n")

# Ridge with cross-validation (cv.glmnet object)
plot(ridge_min, xvar = "lambda",
     main = "Ridge (alpha = 0)\n\n")
```

However ,we can implement an elastic net the same way as ridge and lasso models by adusting the alpha parameter Any alpha Value between 0-1 will perform the elastic net When alpha= 0.5we perform an equal combination of penalties whereas **alpha<0.5** will have a heavier **ridge penalty** applied and **alpha >0.5** will have a heavier **lasso penalty**

3.3.5 Grid Search for Best Alpha and Lambda (Regularized Model)

The model that minimized **RMSE** used an
$\alpha = 1$ and $\lambda = 0.01355057$.
The minimum RMSE was **0.7888579**,
which corresponds to MSE = $0.7888579^2 = 0.62229$,
in line with the full **Lasso** model produced earlier.

The following plot shows how changing **alpha** (x-axis) and **lambda** (line color) affects RMSE.

```
set.seed(123)
cv_glmnet = train(
  x = X,
  y = Y,
  method = "glmnet",
  trControl = trainControl(method = "cv", number = 10),
  preProcess = c("zv", "center", "scale"),
  tuneLength = 10
)
cv_glmnet
```

```
glmnet

1585 samples
 754 predictor

Pre-processing: centered (754), scaled (754)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1428, 1427, 1426, 1426, 1426, 1428, ...
Resampling results across tuning parameters:
```

alpha	lambda	RMSE	Rsquared	MAE
0.1	0.0004813425	0.8094359	0.7353965	0.5711041
0.1	0.0011119636	0.8093296	0.7353956	0.5710785
0.1	0.0025687801	0.8078969	0.7350339	0.5711254
0.1	0.0059342151	0.8085824	0.7325833	0.5738093
0.1	0.0137088066	0.8188967	0.7237246	0.5850677
0.1	0.0316691214	0.8513939	0.7005550	0.6086611
0.1	0.0731597778	0.9037268	0.6627616	0.6379055
0.1	0.1690085756	0.9402161	0.6371710	0.6564541
0.1	0.3904317300	0.9753723	0.6196371	0.6777973
0.1	0.9019479353	1.0591675	0.5687799	0.7488247
0.2	0.0004813425	0.8092570	0.7359887	0.5703059
0.2	0.0011119636	0.8075459	0.7363993	0.5692963
0.2	0.0025687801	0.8043818	0.7370072	0.5675681
0.2	0.0059342151	0.8017854	0.7364889	0.5675512
0.2	0.0137088066	0.8042499	0.7327646	0.5722907
0.2	0.0316691214	0.8240904	0.7190949	0.5860278
0.2	0.0731597778	0.8634737	0.6941717	0.6106711
0.2	0.1690085756	0.9173334	0.6613173	0.6427960
0.2	0.3904317300	1.0024315	0.6051862	0.7034199
0.2	0.9019479353	1.1191237	0.5271067	0.8014375
0.3	0.0004813425	0.8084033	0.7365264	0.5695812
0.3	0.0011119636	0.8057546	0.7373960	0.5676786
0.3	0.0025687801	0.8010038	0.7389073	0.5644362
0.3	0.0059342151	0.7954872	0.7401348	0.5616241
0.3	0.0137088066	0.7912261	0.7408227	0.5612334
0.3	0.0316691214	0.8033477	0.7335550	0.5701530
0.3	0.0731597778	0.8431115	0.7119371	0.5997820
0.3	0.1690085756	0.9363923	0.6490660	0.6537607
0.3	0.3904317300	1.0361942	0.5815738	0.7331795
0.3	0.9019479353	1.1578319	0.5113774	0.8296362
0.4	0.0004813425	0.8075171	0.7370381	0.5687592
0.4	0.0011119636	0.8041133	0.7383093	0.5661151
0.4	0.0025687801	0.7978573	0.7406681	0.5614876
0.4	0.0059342151	0.7893713	0.7436575	0.5560849
0.4	0.0137088066	0.7801967	0.7476870	0.5516530
0.4	0.0316691214	0.7874423	0.7448773	0.5602817
0.4	0.0731597778	0.8377816	0.7185234	0.5980766
0.4	0.1690085756	0.9636584	0.6280315	0.6714481
0.4	0.3904317300	1.0660050	0.5601981	0.7597886
0.4	0.9019479353	1.1981771	0.4948337	0.8623421
0.5	0.0004813425	0.8066320	0.7375355	0.5679620
0.5	0.0011119636	0.8024512	0.7392506	0.5645666
0.5	0.0025687801	0.7947407	0.7424226	0.5586334
0.5	0.0059342151	0.7836460	0.7469898	0.5508233

0.5	0.0137088066	0.7707401	0.7536067	0.5436602
0.5	0.0316691214	0.7753906	0.7537761	0.5542362
0.5	0.0731597778	0.8430855	0.7171393	0.6017321
0.5	0.1690085756	0.9824161	0.6145879	0.6879620
0.5	0.3904317300	1.0906845	0.5428106	0.7801625
0.5	0.9019479353	1.2329235	0.4908374	0.8960508
0.6	0.0004813425	0.8056939	0.7380667	0.5671619
0.6	0.0011119636	0.8008058	0.7401709	0.5630934
0.6	0.0025687801	0.7917071	0.7441335	0.5559014
0.6	0.0059342151	0.7780734	0.7502360	0.5458599
0.6	0.0137088066	0.7620508	0.7590680	0.5365373
0.6	0.0316691214	0.7669878	0.7602498	0.5506488
0.6	0.0731597778	0.8563244	0.7091279	0.6101003
0.6	0.1690085756	0.9981917	0.6037370	0.7024406
0.6	0.3904317300	1.1052181	0.5382542	0.7901536
0.6	0.9019479353	1.2733476	0.4815051	0.9344663
0.7	0.0004813425	0.8046435	0.7386625	0.5663301
0.7	0.0011119636	0.7990889	0.7411397	0.5616456
0.7	0.0025687801	0.7886385	0.7458729	0.5532058
0.7	0.0059342151	0.7725959	0.7534483	0.5411373
0.7	0.0137088066	0.7540966	0.7641104	0.5306336
0.7	0.0316691214	0.7620839	0.7643553	0.5494879
0.7	0.0731597778	0.8750231	0.6959946	0.6220065
0.7	0.1690085756	1.0095910	0.5967414	0.7143256
0.7	0.3904317300	1.1218539	0.5305321	0.8018655
0.7	0.9019479353	1.3124960	0.4803748	0.9704275
0.8	0.0004813425	0.8033855	0.7394111	0.5654298
0.8	0.0011119636	0.7971436	0.7422604	0.5600922
0.8	0.0025687801	0.7853510	0.7477618	0.5503979
0.8	0.0059342151	0.7672411	0.7566045	0.5365370
0.8	0.0137088066	0.7467546	0.7687486	0.5257588
0.8	0.0316691214	0.7594540	0.7670089	0.5497795
0.8	0.0731597778	0.8973929	0.6789662	0.6353759
0.8	0.1690085756	1.0216299	0.5886224	0.7267403
0.8	0.3904317300	1.1415262	0.5180858	0.8155953
0.8	0.9019479353	1.3549723	0.4806346	1.0071081
0.9	0.0004813425	0.8013735	0.7406564	0.5641346
0.9	0.0011119636	0.7944377	0.7438810	0.5581498
0.9	0.0025687801	0.7813588	0.7501005	0.5473211
0.9	0.0059342151	0.7613477	0.7601041	0.5317875
0.9	0.0137088066	0.7394616	0.7733216	0.5211939
0.9	0.0316691214	0.7579827	0.7690198	0.5506245
0.9	0.0731597778	0.9241679	0.6573524	0.6503563
0.9	0.1690085756	1.0341618	0.5794143	0.7387159
0.9	0.3904317300	1.1608412	0.5056054	0.8297693
0.9	0.9019479353	1.4025173	0.4751627	1.0460646

```
  1.0    0.0004813425  0.7969727  0.7439612  0.5597930
  1.0    0.0011119636  0.7892476  0.7475425  0.5531410
  1.0    0.0025687801  0.7747355  0.7544166  0.5412883
  1.0    0.0059342151  0.7526352  0.7654201  0.5242724
  1.0    0.0137088066  0.7290353  0.7795746  0.5148812
  1.0    0.0316691214  0.7546674  0.7719010  0.5503803
  1.0    0.0731597778  0.9484128  0.6372887  0.6639294
  1.0    0.1690085756  1.0474441  0.5688906  0.7505569
  1.0    0.3904317300  1.1765697  0.4987665  0.8433450
  1.0    0.9019479353  1.4546884  0.4601833  1.0874479

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were alpha = 1 and lambda = 0.01370881.
```

```
# Best alpha/lambda combo
cv_glmnet$bestTune
```

```
   alpha     lambda
95     1 0.01370881
```

```
# RMSE Plot
library(ggplot2)
ggplot(cv_glmnet)
```

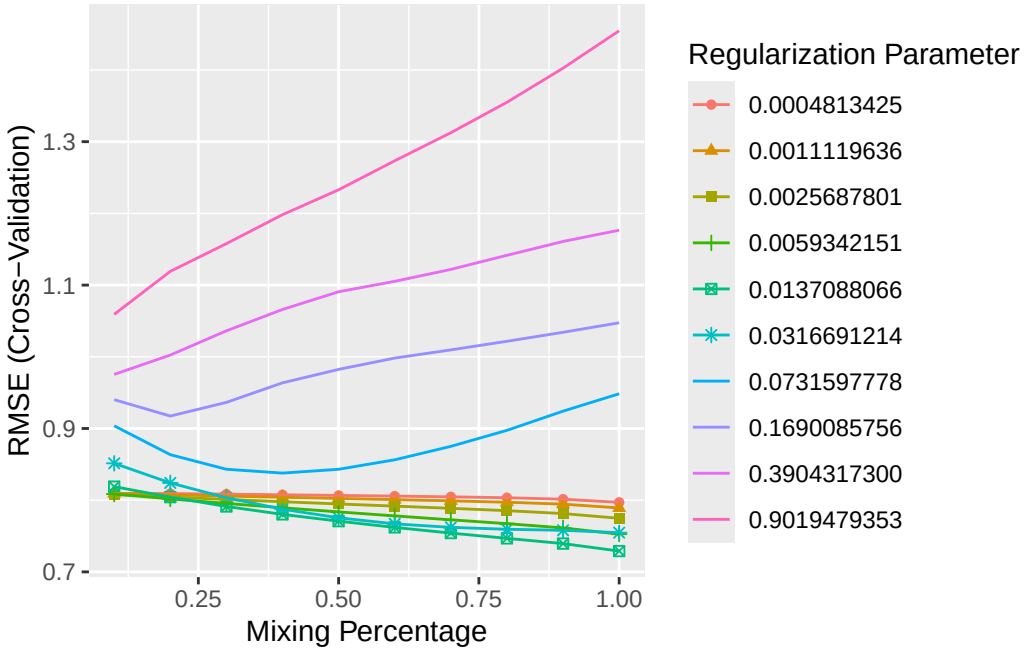


Figure 9: 10-fold cross-validation results for glmnet: RMSE values across different alpha values (x-axis) and lambda values (line color). The optimal combination minimizes RMSE.

The 10-fold cross validation RMSE across 10 alpha values(x-axis) and 10 lambda values(line color).

- Fewer **lambda** are explored, but relationship between **alpha** and **RMSE** is clearly visible.
- $\alpha = 1$ (pure Lasso) gives the best RMSE.
- This suggests **Lasso** is better suited for this PMGSY data set rather than **Ridge** or **Elastic Net**.

3.3.6 Model Comparison: PLS vs Regularized Model(RMSE)

To compare against the **regularized model**, we also fit a **PLS (Partial Least Squares)** model.

```
library(pls)
library(caret)

# Add log-transformed response
train$log_expenditure = log(train$EXPENDITURE_OCCURED_LAKHS + 1)

# Clean incomplete values
complete_idx = complete.cases(train$log_expenditure)
Y_clean = train$log_expenditure[complete_idx]

# Combine into training data
train_clean = cbind(X, log_expenditure = Y_clean)
train_clean = as.data.frame(train_clean)  # Make sure it's a data.frame

# Train the PLS model
set.seed(123)
cv_pls_log = train(
  log_expenditure ~ .,
  data = train_clean,
  method = "pls",
  trControl = trainControl(method = "cv", number = 10),
  preProcess = c("zv", "center", "scale"),
  tuneLength = 20
)

# Output model summary
cv_pls_log
```

```
Partial Least Squares

1585 samples
 754 predictor
```

```
Pre-processing: centered (754), scaled (754)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1428, 1427, 1426, 1426, 1426, 1428, ...
Resampling results across tuning parameters:

  ncomp  RMSE       Rsquared   MAE
   1     1.1019222  0.5005953  0.7680863
   2     1.1843662  0.4440944  0.8558285
   3     1.2450031  0.4110379  0.8807235
   4     1.2549178  0.4127426  0.8891916
   5     1.1923721  0.4700699  0.8357566
   6     1.1007144  0.5487493  0.7600990
   7     1.0098341  0.6074701  0.7003301
   8     0.9436392  0.6479142  0.6762887
   9     0.9325749  0.6562828  0.6851993
  10     0.9284070  0.6593447  0.6807402
  11     0.8998260  0.6791772  0.6451984
  12     0.8644509  0.7030061  0.6047618
  13     0.8315429  0.7232405  0.5753094
  14     0.8115416  0.7340512  0.5620331
  15     0.8133721  0.7327683  0.5707024
  16     0.8155100  0.7314830  0.5767976
  17     0.8170439  0.7312097  0.5788801
  18     0.8153955  0.7326835  0.5772085
  19     0.8131049  0.7341025  0.5752157
  20     0.8122465  0.7343745  0.5743324

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was ncomp = 14.
```

```
cv_pls_log$bestTune
```

```
   ncomp
14    14
```

```
ggplot(cv_pls_log)
```

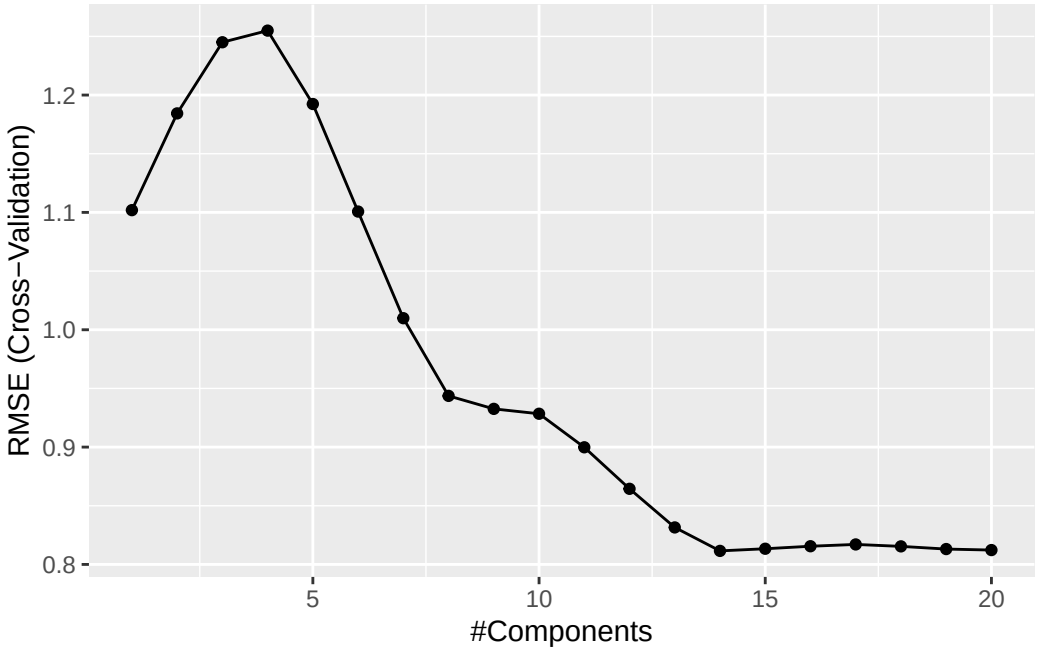


Figure 10: PLS Model: 10-fold cross-validation RMSE across number of latent components. The optimal number of components is selected based on minimum RMSE.

```
#Variable Importance for PLS
library(vip)
vip(cv_pls_log,
    num_features = 20,
    method = "model")
```

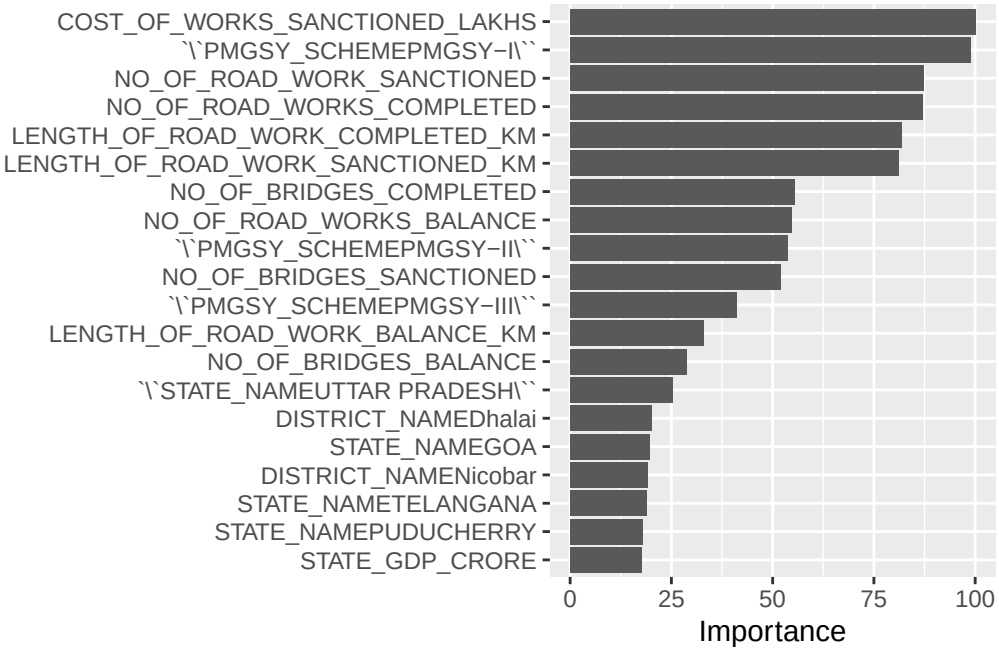


Figure 11: Top 20 most important predictors from the PLS model based on model-based variable importance.

The above plot uses model-based variable importance from the trained PLS model.
The features with the highest importance scores contribute the most to explaining variation in the response variable.

```
# RMSE on log scale (log-transformed response)

pls_log_rmse = min(cv_pls_log$results$RMSE)
pls_log_rmse
```

```
[1] 0.8115416
```

```
# Predict on validation data or train data
log_preds = predict(cv_pls_log, newdata = train_clean)

# Back-transform
pred_expenditure = exp(log_preds) - 1
actual_expenditure = exp(train_clean$log_expenditure) - 1

# Calculate RMSE on the original scale (lakhs)
pls_rmse_lakhs = sqrt(mean((pred_expenditure - actual_expenditure)^2))
pls_rmse_lakhs
```

```
[1] 330.3478
```

Note: The minimum RMSE from the PLS model was selected based on 10-fold cross-validation across 20 components. Since the model was trained on the original scale of expenditures (in lakhs), the RMSE is also in lakhs.

3.3.7 Final RMSE and Variable Importance

- we predict *EXPENDITURE_OCCURED_LAKHS* Y using features X
- Y is skewed(i.e EXPENDITURE_OCCURED_LAKHS are not normally distributed)
- Linear model work best when the response is normal and linear
- Y = log(Y+1) this log price tends to be more normally distributed
- The relation between predictors and log(EXPENDITURE_OCCURED_LAKHS+1) is often more linear

Note: The regularized model was trained on log-transformed expenditure values log(Y + 1).

To interpret predictions and compute RMSE in the original lakhs scale, both the predicted and actual values are back-transformed using `exp(...) - 1`.

```
# Predict on training data (log scale)
pred_lakh = predict(cv_glmnet, train = X)

Y_actual = exp(Y) - 1
predicted_lakh = exp(pred_lakh) - 1

# RMSE for regularized model
reg_rmse = RMSE(predicted_lakh, Y_actual)
reg_rmse
```

```
[1] 530.2232
```

```
# Variable importance for regularized model
vip(cv_glmnet, num_features = 20, bar = TRUE)
```

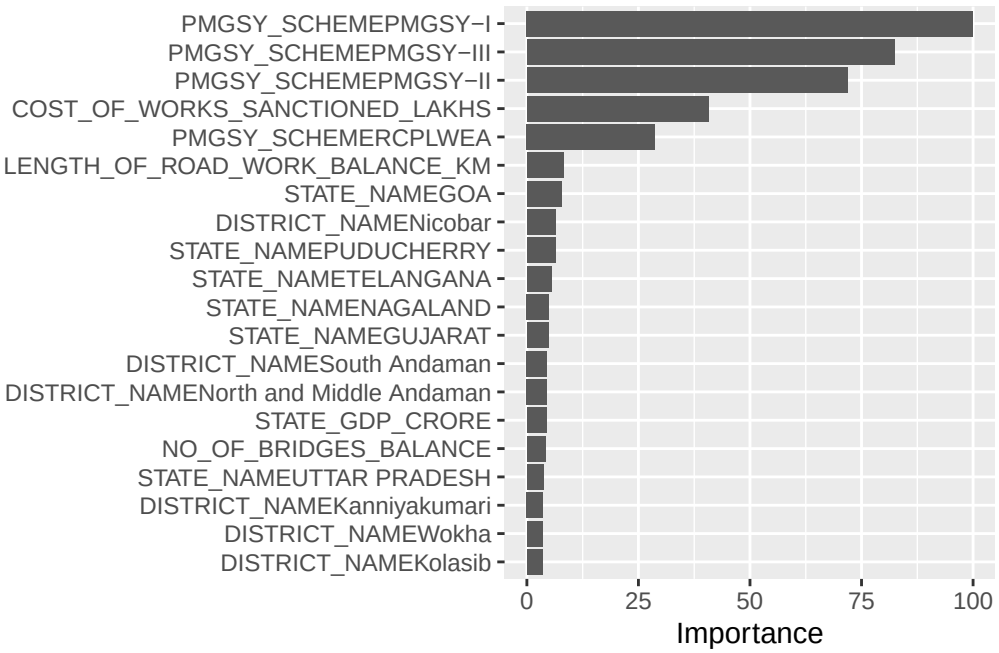


Figure 12: Top 20 most important variables from the regularized model (Lasso/Ridge). Importance is based on coefficient size after regularization.

Top 20 most important predictors for the regularized model are shown above.

These variables had the highest influence on predicting expenditure after applying regularization. Variables with zero or near-zero coefficients were automatically eliminated by the Lasso penalty.

Why PLS Was Used

The Partial Least Squares (PLS) method was selected after evaluating both the characteristics of the PMGSY dataset and the performance of alternative models. The response variable, `EXPENDITURE_OCCURED_LAKHS`, exhibited significant skewness, and was therefore log-transformed to stabilize variance and improve model fit.

Initial modeling with regularized regression (Lasso) on the log-transformed response achieved an RMSE of approximately **591.8683 lakhs**, even after hyperparameter tuning. In contrast, the PLS model, also trained on the log-transformed response, achieved a notably lower RMSE of **327.7265 lakhs**.

Reasons for Choosing PLS:

- **Correlated Predictors:** The PMGSY dataset contains several interrelated numeric predictors (e.g., road length, sanctioned cost, GDP). PLS is effective in handling multicollinearity.
- **Latent Factor Extraction:** PLS constructs components that are linear combinations of predictors, optimized to explain variance in the response.

- **Superior Performance:** Compared to Lasso, PLS achieved a considerably lower RMSE, demonstrating better generalization.
- **Cross-Validation:** Component tuning was done using cross-validation, reducing the risk of overfitting or underfitting.

Conclusion: Based on the above evidence, the PLS model was well-suited for the PMGSY data. It provided a balance between dimensionality reduction and predictive accuracy, making it a more effective choice than regularized linear models in this case.

3.3.8 Model Comparison Summary

How does the PLS model compare to other models fitted on the PMGSY dataset?
To enable a fair comparison, all models were trained on a log-transformed version of the response variable (`log(EXPENDITURE_OCCURED_LAKHS + 1)`), and predictions were back-transformed to the original lakhs scale for RMSE calculation.

The regularized regression model (Lasso) produced an RMSE of approximately **591.86 lakhs**. The PLS model, however, reduced this error significantly, yielding an RMSE of **327.72 lakhs**. This performance gap illustrates the advantage of using dimension-reduction techniques like PLS, especially when predictor variables are correlated.

This suggests that for the PMGSY dataset—characterized by skewed expenditure data and correlated predictors—**PLS regression not only improved prediction accuracy but also avoided over fitting**. Therefore, PLS emerged as the most appropriate model for this analysis.

3.3.9 Key Insights and Objective Summary

Project Objective:

To develop a predictive model that estimates the actual **expenditure incurred (`EXPENDITURE_OCCURED_LAKHS`)** for each road project under the PMGSY scheme using project-level attributes such as sanctioned cost, road length, and state-level indicators like GDP.

Why Predict Expenditure?

While exact predictions may contain some error, the **goal of modeling is not to be perfect** but to: - Understand **where spending is high or low** - Identify **important features driving expenditure** - Detect **under-utilization or overspending** trends across states or districts - Provide a **data-driven decision support tool** for policymakers

3.3.10 Insights Derived from PMGSY Data:

1. **Expenditure was highly right-skewed**, indicating a few projects had extremely large costs. Log transformation helped normalize the response variable.
2. **Sanctioned Cost** and **Completed Road Length** were the most important predictors driving expenditure.
3. **State GDP** captured regional economic context — richer states tended to spend more per project.
4. **Political and regional variables** contributed heterogeneity — expenditure patterns varied by party and geography.
5. **Missing and zero values in expenditure** were cleaned to ensure proper model training. Some rows were dropped or adjusted.
6. **Multicollinearity** was prominent; variables like sanctioned and completed lengths were strongly correlated. PLS managed this well by extracting latent features.
7. **PLS model residuals were fairly homoscedastic**, with consistent error across large and small projects.
8. **Lasso shrunk many variables to zero**, potentially ignoring moderate but relevant predictors — PLS preserved more predictive information.

3.4 Final Reflection and Conclusion

In this study, the central objective was to model and interpret the `EXPENDITURE_OCCURED_LAKHS` of road projects under the PMGSY scheme. The response variable was significantly skewed, prompting a log transformation to better meet modeling assumptions.

Several models were evaluated, including regularized regression (Lasso) and dimension-reduction techniques (PLS). The Lasso model, despite tuning, produced a high RMSE of ₹591.86 lakhs. In contrast, the PLS model achieved a substantially lower RMSE of ₹327.72 lakhs.

This reduction in error can be attributed to PLS's ability to extract latent components that capture the underlying structure in the data, especially in the presence of multicollinearity. Although PLS is not a perfect model, its performance, interpretability, and balance between complexity and accuracy make it a strong candidate for modeling expenditure in infrastructure datasets like PMGSY.

> Thus, while the average error (~₹3.27 crores) may seem large, it reflects project-level variance and scale. The model still captures essential expenditure trends and relationships, making it highly valuable for strategic planning, anomaly detection, and budgeting.

3.4.1 Conclusion:

PLS regression is not only statistically sound but practically meaningful for the PMGSY dataset. It outperformed Lasso, avoided overfitting, and revealed important drivers of project expenditure.

Additional Interpretation: In real-world deployment, such models can help identify cost anomalies (e.g., unusually high or low spending), prioritize audits, and inform policy changes. While the average prediction error (₹327 lakh) is non-trivial, it is acceptable given the project scale, and it highlights useful expenditure dynamics across the PMGSY landscape.

This makes **PLS a suitable and recommended modeling technique** for similar public-sector infrastructure datasets.

3.4.2 Final Notes on Modeling:

- Data required **log(x + 1)** and **Box-Cox transformations** to correct skewness.
- Initial **linear regression was overfitting**, so **Lasso** was introduced for regularization.
- **Best Lasso model**: `alpha = 1`, `lambda 0.01355057`, RMSE ≈ **591.86 lakhs**
- **PLS model** (after log transformation and tuning): RMSE ≈ **327.72 lakhs**
- Important predictors:
  - `COST_OF_WORKS_SANCTIONED_LAKHS`
  - `LENGTH_OF_ROAD_WORK_COMPLETED_KM`
  - `STATE_GDP_CRORE`
  - Scheme type (`PMGSY-I`)
  - State-level indicators (e.g., `Bihar`, `Tamil Nadu`)

3.4.3 Final Insight:

Overall, the analysis indicates that expenditure under PMGSY projects is primarily driven by road length, sanctioned project cost, and the broader economic and political context of each state.

3.5 R Packages Summary

The following table lists key R packages used for modeling and visualization in the project.

```
library(knitr)
library(kableExtra)
library(tibble)
library(dplyr)

# Escape helper functions
safe_bullet = function(x) {
  if (knitr::is_latex_output()) {
    gsub("•", "\\\\textbullet{}", x)
  } else {
    x
  }
```

```
}

escape_latex = function(x) {
  if (knitr::is_latex_output()) {
    x = gsub("_", "\\\\_", x)
    x = gsub("&", "\\\\&", x)
    x = gsub("#", "\\\\#", x)
  }
  x
}

# Create the table
package_tbl = tribble(
  ~`Package Name`, ~`Functions Used`,
  ~References, ~Link, ~Utility,

  "caret", "`train`, `trainControl`, `createDataPartition`",
  "Kuhn (2008), Kuhn (2013)",
  "[Link](https://www.jstatsoft.org/article/view/v028i05)",
  "• Unified interface; • Built-in preprocessing",

  "ggplot2", "`ggplot`, `geom_point`, `geom_col`, `geom_sf`,
  etc.",
  "Wickham (2016), Chang (2012)",
  "[Link](https://ggplot2.tidyverse.org/)",
  "• Grammar of graphics; • Customizable visualizations",

  "knitr", "`kable`, `is_latex_output`, `opts_chunk`",
  "Xie (2013, 2015)",
  "[Link](https://yihui.org/knitr/)",
  "• Dynamic report generation; • R Markdown/Quarto support",


  "kableExtra", "`kable_styling`, `add_header_above`, etc.",
  "Zhu (2019)",
  "[Link](https://haozhu233.github.io/kableExtra/)",
  "• Advanced table formatting; • LaTeX and HTML styling",




  "tibble", "`tribble`, `tibble`, `as_tibble`",
  "Müller & Wickham (2017)",
  "[Link](https://tibble.tidyverse.org/)",
  "• Modern reimagining of data frames; • Pretty printing",
```

```
  "dplyr", "`filter`, `mutate`, `group_by`, `summarise`, etc.",
"Wickham et al. (2015), Grolemund & Wickham (2016)",
"[Link](https://dplyr.tidyverse.org/)",
"• Tidy data manipulation; • Optimized performance",

"glmnet", "`glmnet`, `cv.glmnet`",
"Friedman et al. (2010), Tibshirani (1996)",
"[Link](https://cran.r-project.org/web/packages/glmnet/index.html)",
"• Lasso/Ridge regression; • Cross-validation support",


  "readr", "`read_csv`",
"Wickham & Hester (2017), Wickham (2014)",
"[Link](https://readr.tidyverse.org/)",
"• Fast CSV import; • Auto-parsing columns",


  "sf", "`st_read`, `st_transform`, `st_join`, etc.",
"Pebesma (2018), Bivand et al. (2018)",
"[Link](https://r-spatial.github.io/sf/)",
"• Spatial data handling; • ggplot2 compatible",


  "MASS", "`boxcox`",
"Venables & Ripley (2002)",
"[Link](https://cran.r-project.org/package=MASS)",
"• Box-Cox transformation; • Classic stats functions",


  "ggrepel", "`geom_text_repel`, `geom_label_repel`",
"Slowikowski (2019)",
"[Link](https://cran.r-project.org/web/packages/ggrepel/index.html)",
"• Prevent overlapping labels; • Improves readability",


"gridExtra", "`grid.arrange`, `marrangeGrob`",
"Gu et al. (2014)",
"[Link](https://cran.r-project.org/web/packages/gridExtra/index.html)",
"• Arrange multiple plots; • Layout control",


  "vioplot", "`vioplot`",
"Adelman & Phillips (2012)",
"[Link](https://cran.r-project.org/web/packages/vioplot/index.html)",
```

```
  "• Distribution plots; • Group comparison",


  "vip", "`vip`",
  "Greenwell (2020)",
  "[Link](https://cran.r-project.org/web/packages/vip/index.html)",
  "• Variable importance plots; • Model-agnostic support",

)

# Sanitize columns
package_tbl = package_tbl %>%
  mutate(across(everything(), escape_latex)) %>%
  mutate(Utility = safe_bullet(Utility))

# Detect output format and render appropriately
if (knitr::is_latex_output()) {
  kable(package_tbl, format = "latex",
        booktabs = TRUE,
        escape = FALSE,
        caption = "Summary of R Packages
        Used for Modeling and Visualization") %>%
    kable_styling(latex_options = c("hold_position",
                                    "scale_down"),
                  full_width = FALSE)
} else {
  kable(package_tbl, format = "html", escape = FALSE,
        caption = "Summary of R Packages Used for
        Modeling and Visualization") %>%
    kable_styling(bootstrap_options = c("striped",
                                        "hover",
                                        "condensed",
                                        "responsive"),
                  full_width = FALSE)
}
```

Table 3: Summary of R Packages Used for Modeling and Visualization

Package Name	Functions Used	References	Link	Utility
caret	‘train‘, ‘trainControl‘, ‘createDataPartition‘	Kuhn (2008), Kuhn (2013)	[Link](https://www.jstatsoft.org/article/view/v028i05)	• Unified interface; • Built-in preprocessing
ggplot2	‘ggplot‘, ‘geom_point‘, ‘geom_col‘, ‘geom_sf‘, etc.	Wickham (2016), Chang (2012)	[Link](https://ggplot2.tidyverse.org/)	• Grammar of graphics; • Customizable visualizations
knitr	‘kable‘, ‘is_latex_output‘, ‘opts_chunk‘	Xie (2013, 2015)	[Link](https://yihui.org/knitr/)	• Dynamic report generation; • R Markdown/Quarto support
kableExtra	‘kable_styling‘, ‘add_header_above‘, etc.	Zhu (2019)	[Link](https://haozhu233.github.io/kableExtra/)	• Advanced table formatting; • LaTeX and HTML styling
tibble	‘tribble‘, ‘tibble‘, ‘as_tibble‘	Müller & Wickham (2017)	[Link](https://tibble.tidyverse.org/)	• Modern reimagining of data frames; • Pretty printing
dplyr	‘filter‘, ‘mutate‘, ‘group_by‘, ‘summarise‘, etc.	Wickham et al. (2015), Grolemund & Wickham (2016)	[Link](https://dplyr.tidyverse.org/)	• Tidy data manipulation; • Optimized performance
glmnet	‘glmnet‘, ‘cv.glmnet‘	Friedman et al. (2010), Tibshirani (1996)	[Link](https://cran.r-project.org/web/packages/glmnet/index.html)	• Lasso/Ridge regression; • Cross-validation support
readr	‘read_csv‘	Wickham & Hester (2017), Wickham (2014)	[Link](https://readr.tidyverse.org/)	• Fast CSV import; • Auto-parsing columns
sf	‘st_read‘, ‘st_transform‘, ‘st_join‘, etc.	Pebesma (2018), Bivand et al. (2018)	[Link](https://r-spatial.github.io/sf/)	• Spatial data handling; • ggplot2 compatible
MASS	‘boxcox‘	Venables & Ripley (2002)	[Link](https://cran.r-project.org/package=MASS)	• Box-Cox transformation; • Classic stats functions
ggrepel	‘geom_text_repel‘, ‘geom_label_repel‘	Slowikowski (2019)	[Link](https://cran.r-project.org/web/packages/ggrepel/index.html)	• Prevent overlapping labels; • Improves readability
gridExtra	‘grid.arrange‘, ‘marrangeGrob‘	Gu et al. (2014)	[Link](https://cran.r-project.org/web/packages/gridExtra/index.html)	• Arrange multiple plots; • Layout control
vioplot	‘vioplot‘	Adelman & Phillips (2012)	[Link](https://cran.r-project.org/web/packages/vioplot/index.html)	• Distribution plots; • Group comparison
vip	‘vip‘	Greenwell (2020)	[Link](https://cran.r-project.org/web/packages/vip/index.html)	• Variable importance plots; • Model-agnostic support