UK Energy Demand Analysis

MLDS Group 4

2025-03-03

1. Data Introduction and Cleaning

Loading the Data

```r
energy_data <- read.csv("../energy_demand_uk.csv")

# Ensure date column is Date type
energy_data$date <- as.Date(energy_data$date)

# Display data summary
cat("Data dimensions:", nrow(energy_data), "rows,", ncol(energy_data), "columns\n")
```

```
## Data dimensions: 1461 rows, 10 columns
```

```r
head(energy_data)
```

```
##   X       date national_demand wind_generation solar_generation min_temp
## 1 1 2020-06-01           27193            1007             1490       11
## 2 2 2020-06-02           27255             796             1170       10
## 3 3 2020-06-03           27369            2284              394        6
## 4 4 2020-06-04           28676            1501              577        9
## 5 5 2020-06-05           25246            3954             1280        7
## 6 6 2020-06-06           25723            2070              981        7
##   max_temp rain_mm wind_speed average_price_daily
## 1       22     0.0         14               2.810
## 2       22     0.0          9               3.434
## 3       15     5.1         16               2.011
## 4       15     2.9         16               3.066
## 5       14     1.3         26               1.516
## 6       14     3.2         25               0.525
```

```r
summary(energy_data)
```

```
##        X             date            national_demand wind_generation
##  Min.   :   1   Min.   :2020-06-01   Min.   :21329   Min.   : 233
##  1st Qu.: 366   1st Qu.:2021-06-01   1st Qu.:28845   1st Qu.: 891
##  Median : 731   Median :2022-06-01   Median :32241   Median :1471
##  Mean   : 731   Mean   :2022-06-01   Mean   :32826   Mean   :1704
##  3rd Qu.:1096   3rd Qu.:2023-06-01   3rd Qu.:36760   3rd Qu.:2319
##  Max.   :1461   Max.   :2024-05-31   Max.   :44873   Max.   :5501
##
##  solar_generation    min_temp         max_temp        rain_mm
##  Min.   :   0.0   Min.   :-4.000   Min.   :-1.00   Min.   : 0.000
##  1st Qu.:   0.0   1st Qu.: 4.000   1st Qu.: 9.00   1st Qu.: 0.000
```

```
##  Median :   1.0   Median : 7.000   Median :13.00   Median : 0.500
##  Mean   : 340.2   Mean   : 7.492   Mean   :13.82   Mean   : 2.364
##  3rd Qu.: 631.0   3rd Qu.:11.000   3rd Qu.:18.00   3rd Qu.: 3.200
##  Max.   :1760.0   Max.   :23.000   Max.   :38.00   Max.   :32.400
##                   NA's   :1        NA's   :1       NA's   :1
##    wind_speed    average_price_daily
##  Min.   : 3.00   Min.   :-1.691
##  1st Qu.:10.00   1st Qu.: 5.874
##  Median :14.00   Median : 8.920
##  Mean   :14.86   Mean   :11.412
##  3rd Qu.:19.00   3rd Qu.:14.892
##  Max.   :36.00   Max.   :77.790
##  NA's   :1
```

Handling Missing Values

```r
# Check and print rows with missing values
missing_rows <- energy_data %>% filter(if_any(everything(), is.na))
cat("Number of missing rows:", nrow(missing_rows), "\n")
```

```
## Number of missing rows: 1
```

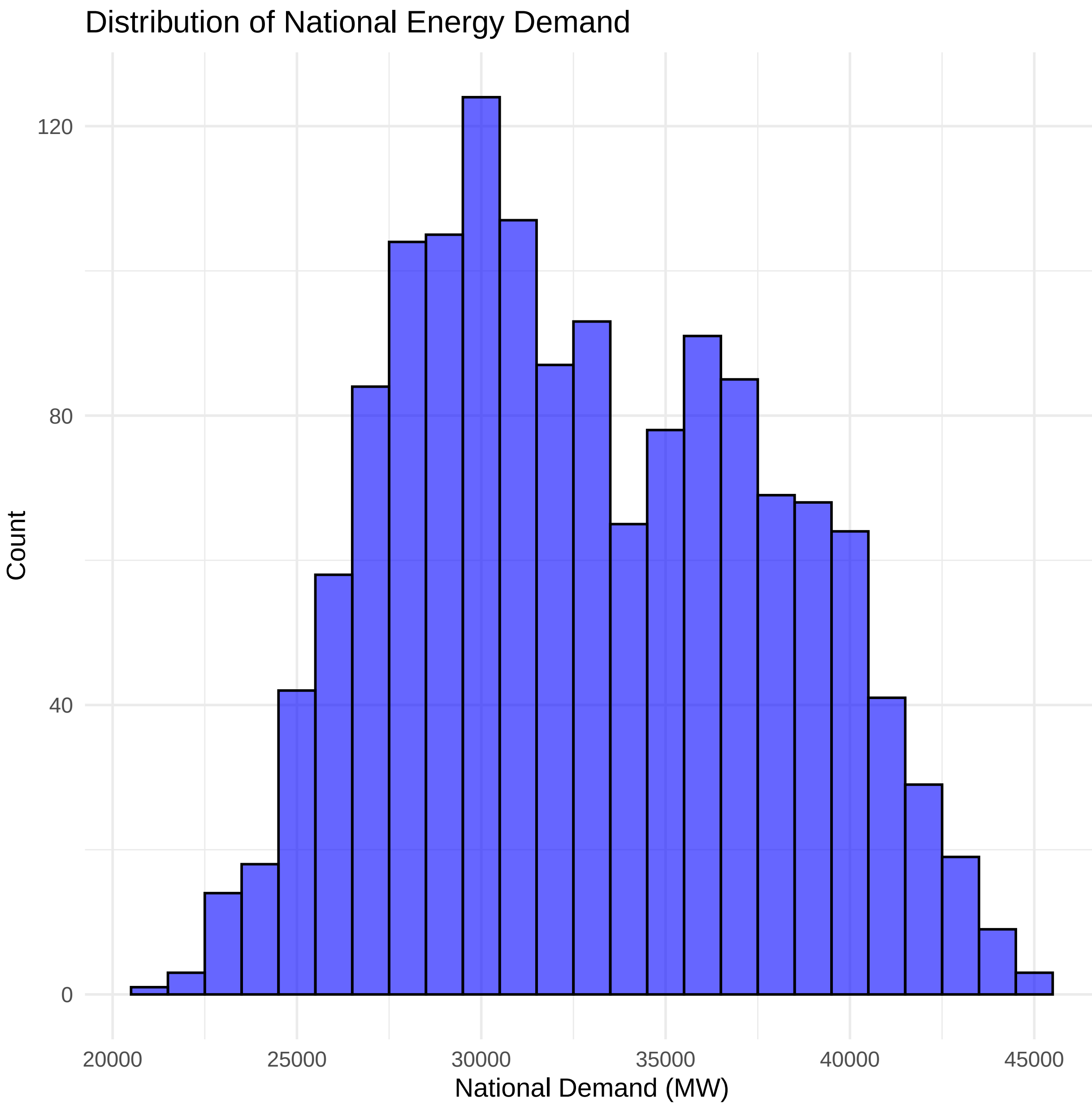
```r
# Identify which columns have missing values
if(nrow(missing_rows) > 0) {
  missing_cols <- colnames(missing_rows)[colSums(is.na(missing_rows)) > 0]
  cat("Columns with missing values:", paste(missing_cols, collapse=", "), "\n")

  # View the missing rows
  print(missing_rows)

  # Forward fill missing values
  energy_data <- na.locf(energy_data)

  # Verify missing values have been filled
  cat("Remaining missing values:", sum(is.na(energy_data)), "\n")
}
```

```
## Columns with missing values: min_temp, max_temp, rain_mm, wind_speed
##      X       date national_demand wind_generation solar_generation min_temp
## 1 1369 2024-02-29           37958            1489                0       NA
##   max_temp rain_mm wind_speed average_price_daily
## 1       NA      NA         NA               6.373
## Remaining missing values: 0
```

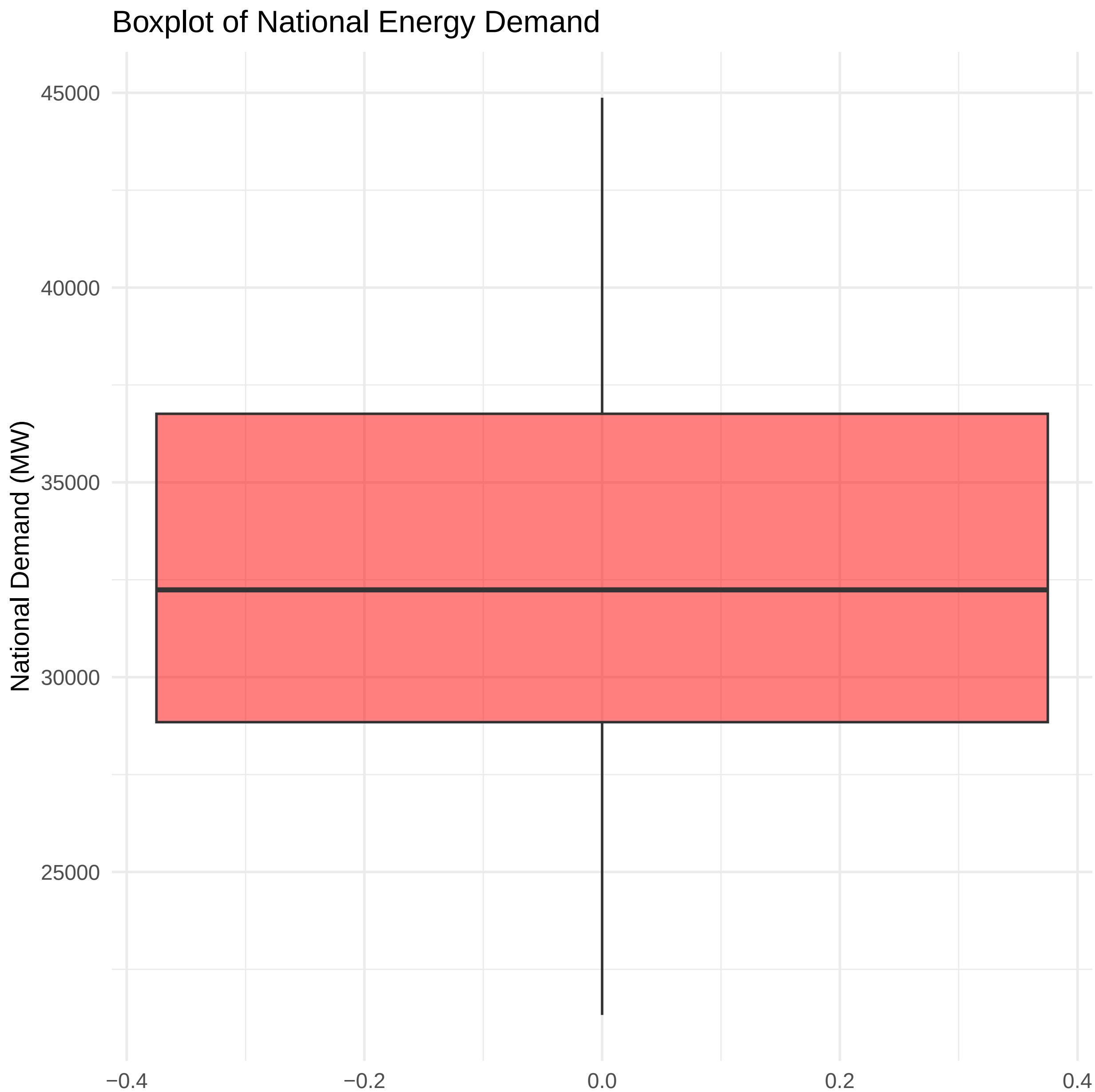
2. Univariate Exploratory Data Analysis

Distribution of National Energy Demand

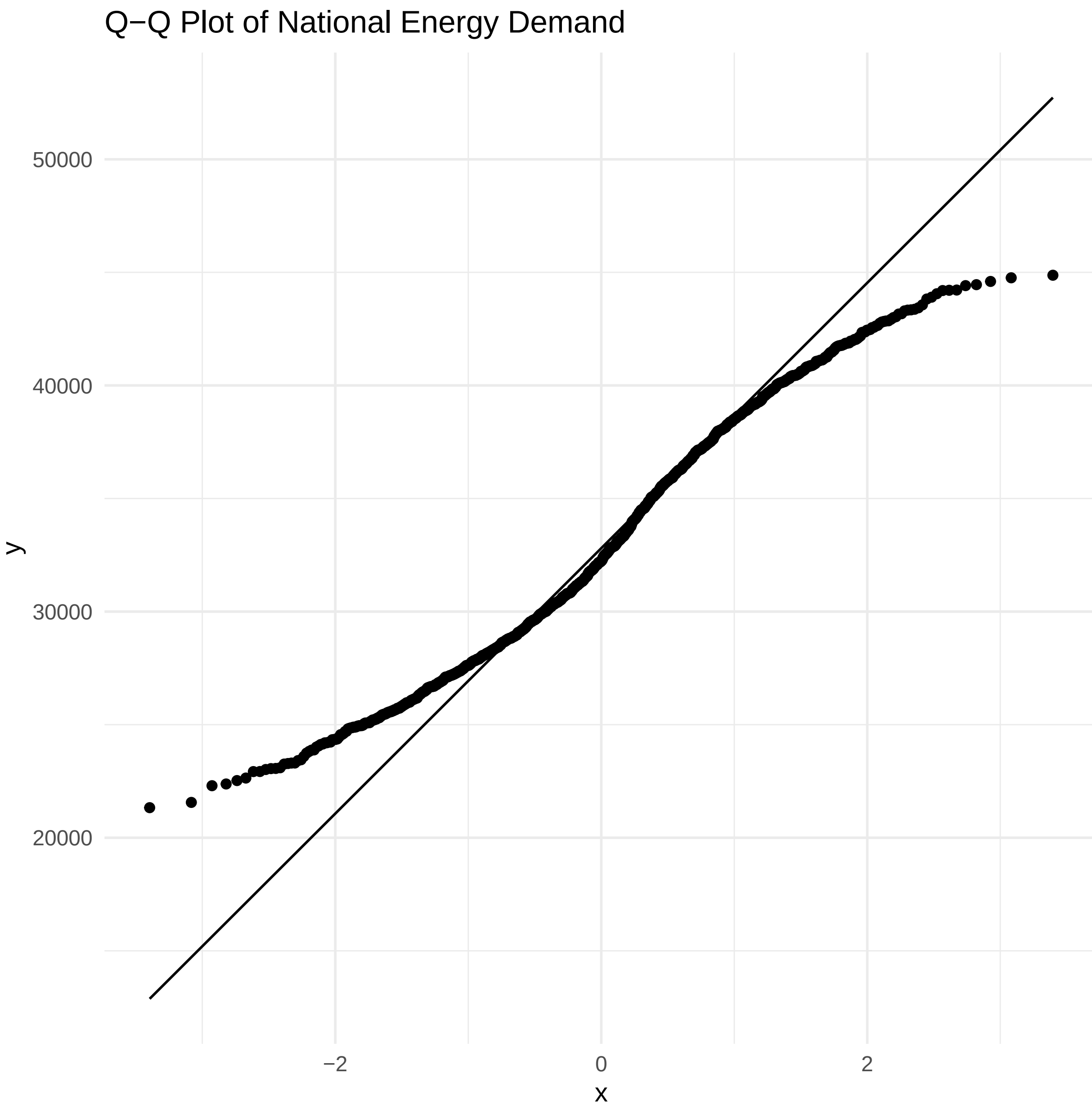
```r
# Histogram
ggplot(energy_data, aes(x = national_demand)) +
  geom_histogram(binwidth = 1000, fill = "blue", alpha = 0.6, color = "black") +
  labs(title = "Distribution of National Energy Demand", x = "National Demand (MW)", y = "Count") +
  theme_minimal()
```



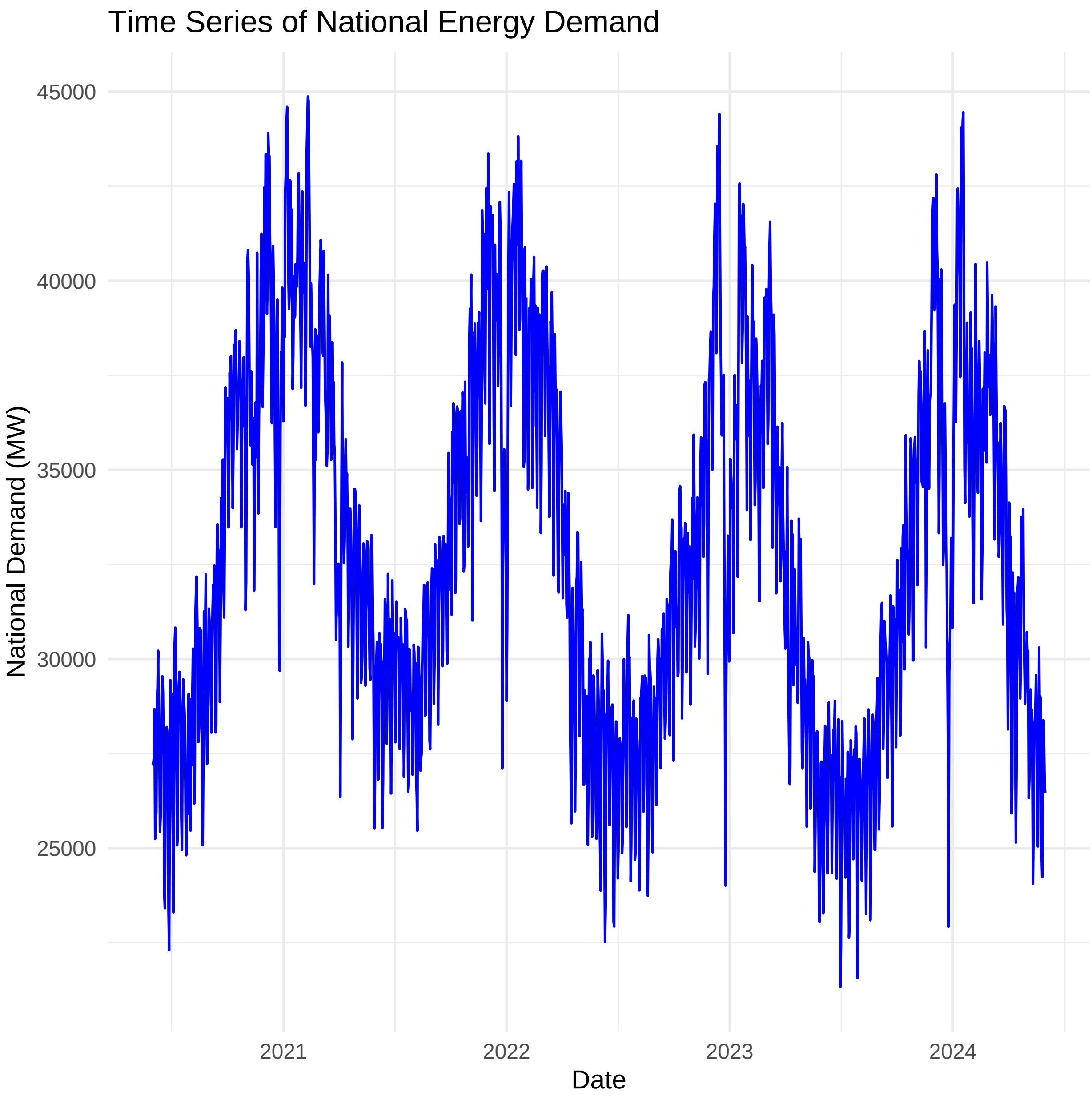

```r
# Boxplot
ggplot(energy_data, aes(y = national_demand)) +
  geom_boxplot(fill = "red", alpha = 0.5) +
  labs(title = "Boxplot of National Energy Demand", y = "National Demand (MW)") +
  theme_minimal()
```




```r
# QQ plot
ggplot(energy_data, aes(sample = national_demand)) +
  geom_qq() +
  geom_qq_line() +
  labs(title = "Q-Q Plot of National Energy Demand") +
  theme_minimal()
```




```r
# Time series plot
ggplot(energy_data, aes(x = date, y = national_demand)) +
  geom_line(color = "blue") +
  labs(title = "Time Series of National Energy Demand",
       x = "Date", y = "National Demand (MW)") +
  theme_minimal()
```


Time Series of National Energy Demand
45000
40000
35000
30000
25000
National Demand (MW)
2021
2022
2023
2024
Date

Conclusion:

From the histogram, the data appears to be right skewed, i.e., longer tail on the right.

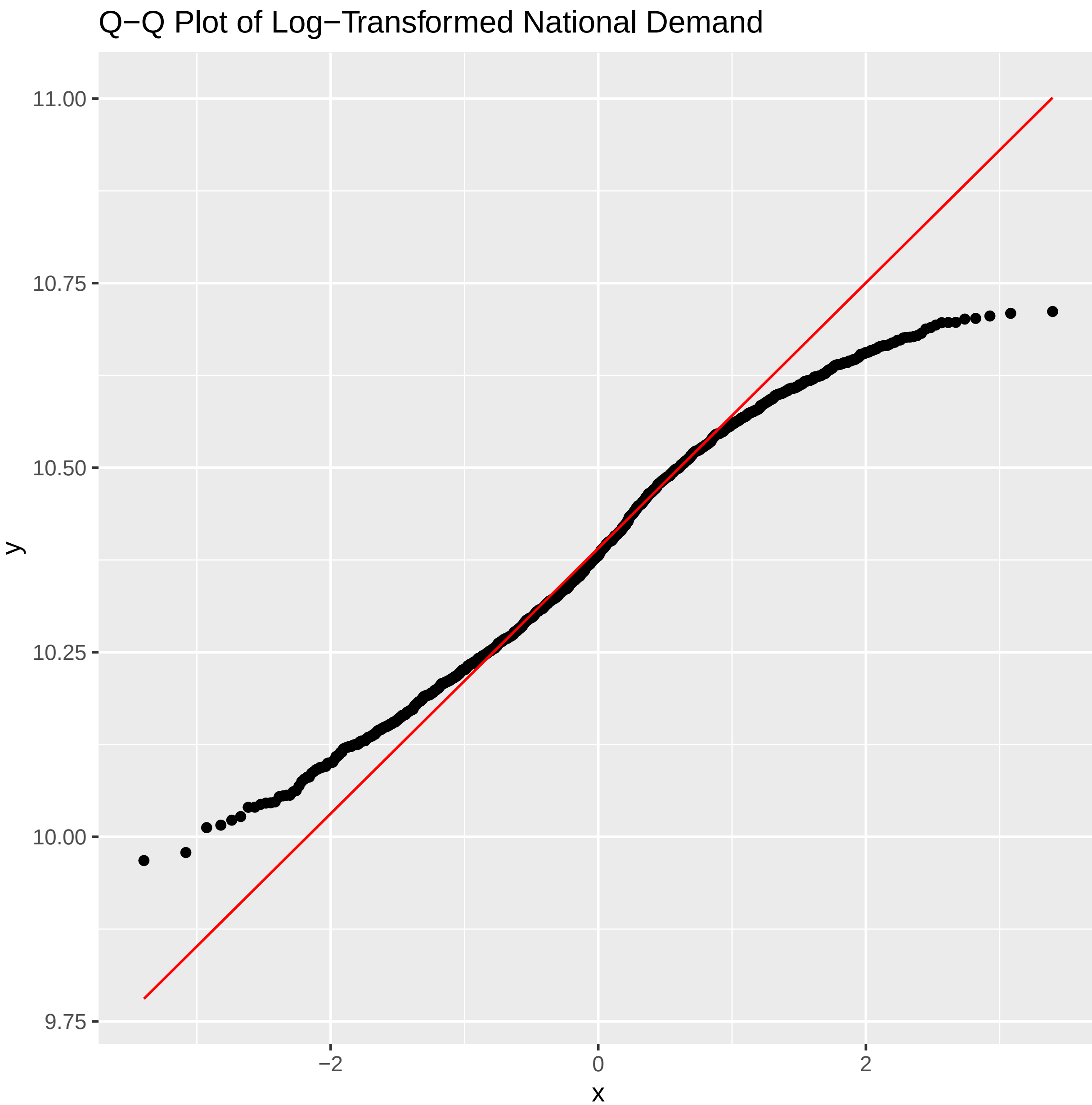
From the QQ plot, the data doesn't follow a normal distribution, based on the deviation from the extreme right and left side of the plot.

The national demand data does not follow normal distribution.

So in the following steps, let's try some transformation, to see if the data can follow normal distribution after transformation.

Testing Different Distributions

```
# QQ plot for log-transformed data
ggplot(energy_data, aes(sample = log(national_demand))) +
  geom_qq() +
  geom_qq_line(color = "red") +
  labs(title = "Q-Q Plot of Log-Transformed National Demand")
```



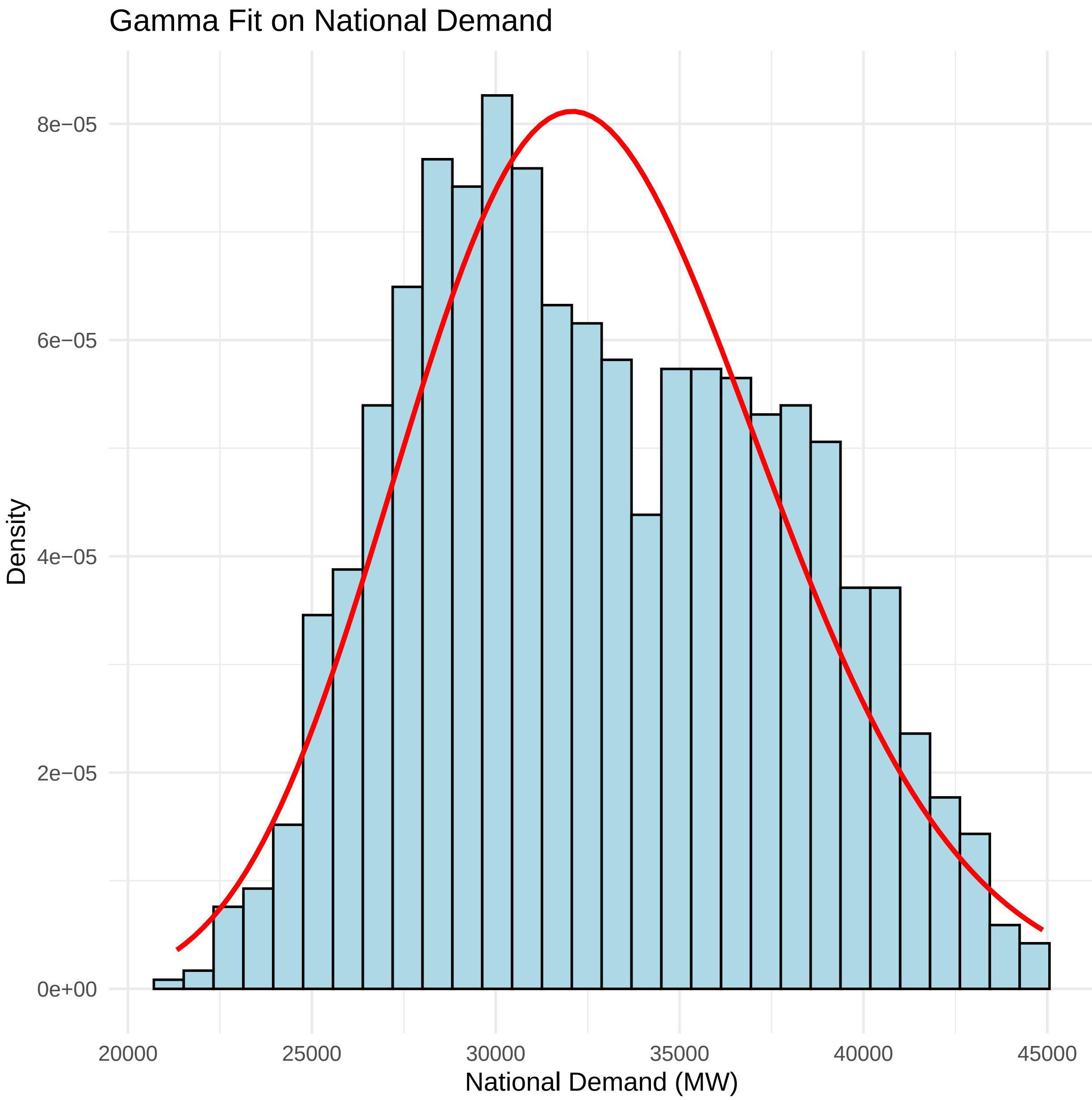

```r
# Fitting Gamma Distribution
fit_gamma <- fitdistr(energy_data$national_demand, "gamma")
print(fit_gamma)
```

```
##       shape            rate
##   4.373693e+01   1.332391e-03
##  (2.623343e-01) (5.311510e-06)
```

```r
shape_param <- fit_gamma$estimate["shape"]
rate_param <- fit_gamma$estimate["rate"]

# Plot histogram with gamma density using ggplot2 approach
ggplot(energy_data, aes(x = national_demand)) +
  geom_histogram(aes(y = ..density..), bins = 30, fill = "lightblue", color = "black") +
  stat_function(fun = dgamma, args = list(shape = shape_param, rate = rate_param),
                color = "red", linewidth = 1) +
```

```r
  labs(title = "Gamma Fit on National Demand", x = "National Demand (MW)", y = "Density") +
  theme_minimal()
```

```
## Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(density)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```




```r
# KS test for Gamma fit
ks_gamma <- ks.test(energy_data$national_demand, "pgamma", shape_param, rate_param)
```

```
## Warning in ks.test.default(energy_data$national_demand, "pgamma", shape_param,
## : ties should not be present for the one-sample Kolmogorov-Smirnov test
```

```r
print(ks_gamma)
```

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  energy_data$national_demand
## D = 0.0484, p-value = 0.002129
## alternative hypothesis: two-sided
```

```r
# Fitting Log Normal Distribution
fit_lognormal <- fitdistr(energy_data$national_demand, "lognormal")
print(fit_lognormal)
```

```
##      meanlog         sdlog
##   10.387501142    0.151818757
##  ( 0.003971918) ( 0.002808570)
```

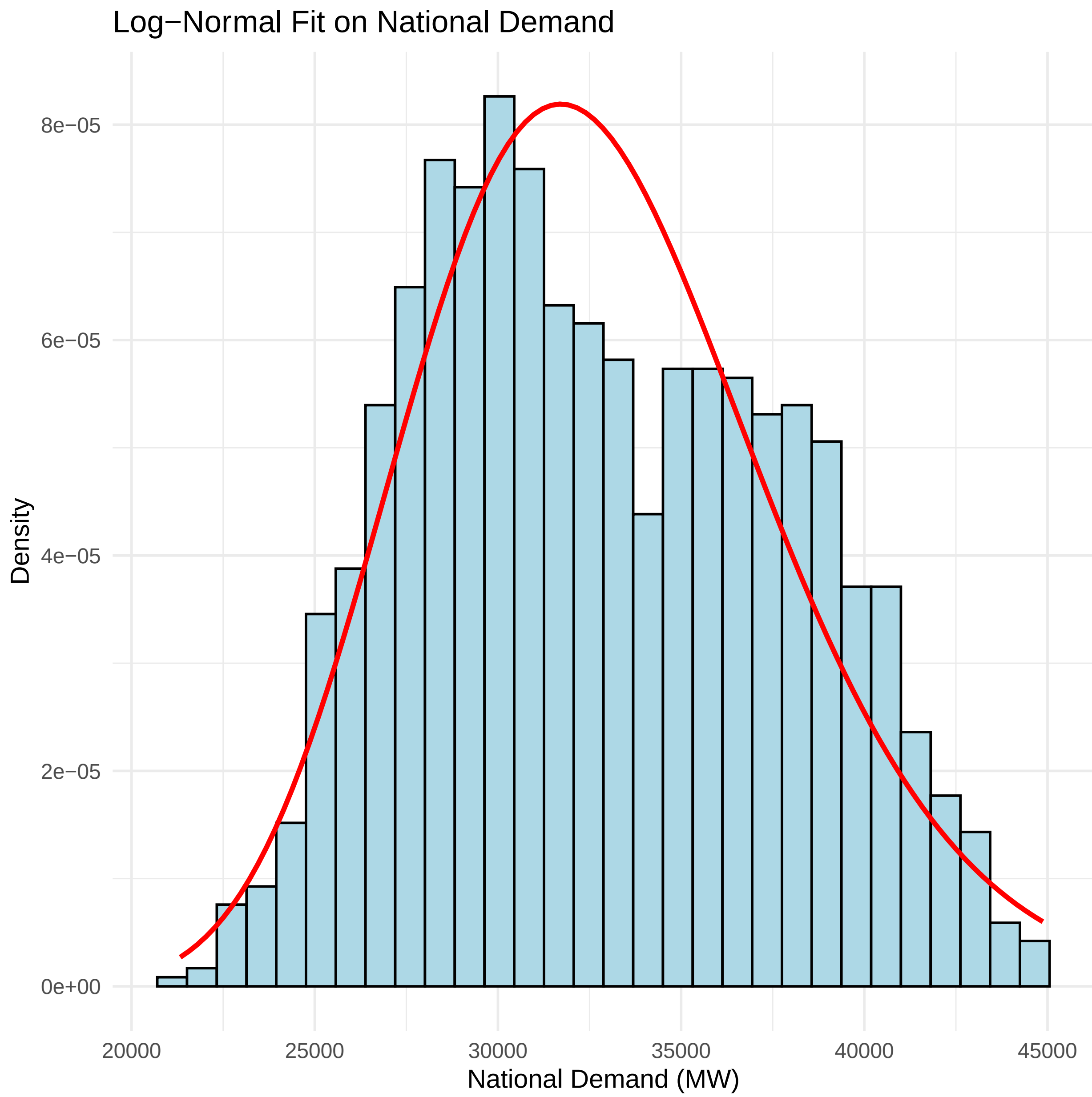
```r
meanlog <- fit_lognormal$estimate["meanlog"]
sdlog <- fit_lognormal$estimate["sdlog"]

# KS test for Log-Normal fit
ks_lognormal <- ks.test(energy_data$national_demand, "plnorm", meanlog, sdlog)
```

```
## Warning in ks.test.default(energy_data$national_demand, "plnorm", meanlog, :
## ties should not be present for the one-sample Kolmogorov-Smirnov test
```

```r
print(ks_lognormal)
```

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  energy_data$national_demand
## D = 0.051592, p-value = 0.000838
## alternative hypothesis: two-sided
```

```r
# Plot histogram with log-normal density using ggplot2 approach
ggplot(energy_data, aes(x = national_demand)) +
  geom_histogram(aes(y = ..density..), bins = 30, fill = "lightblue", color = "black") +
  stat_function(fun = dlnorm, args = list(meanlog = meanlog, sdlog = sdlog),
                color = "red", linewidth = 1) +
  labs(title = "Log-Normal Fit on National Demand", x = "National Demand (MW)", y = "Density") +
  theme_minimal()
```


Log−Normal Fit on National Demand
Density
8e−05
6e−05
4e−05
2e−05
0e+00
20000
25000
30000
35000
40000
45000
National Demand (MW)

Conclusion:

From the histogram, the data appears to be right skewed, i.e., longer tail on the right.

From the QQ plot, the data doesn't follow a normal distribution, based on the deviation from the extreme right and left side of the plot.

We did further testing on the log transformed of national demand, and from the QQ plot we can see that it still doesn't fit normal distribution.

Based on the right skewed feature of the data, we tried the following:

1. Fitting gamma distribution, as the gamma distribution is only for positive values (X>0), and tend to be used for data that are right skewed. However, from the result for the KS test, we see that p-value from the KS test is very small (< 0.05), we reject H0. This concludes Gamma is not a good fit for the energy data.

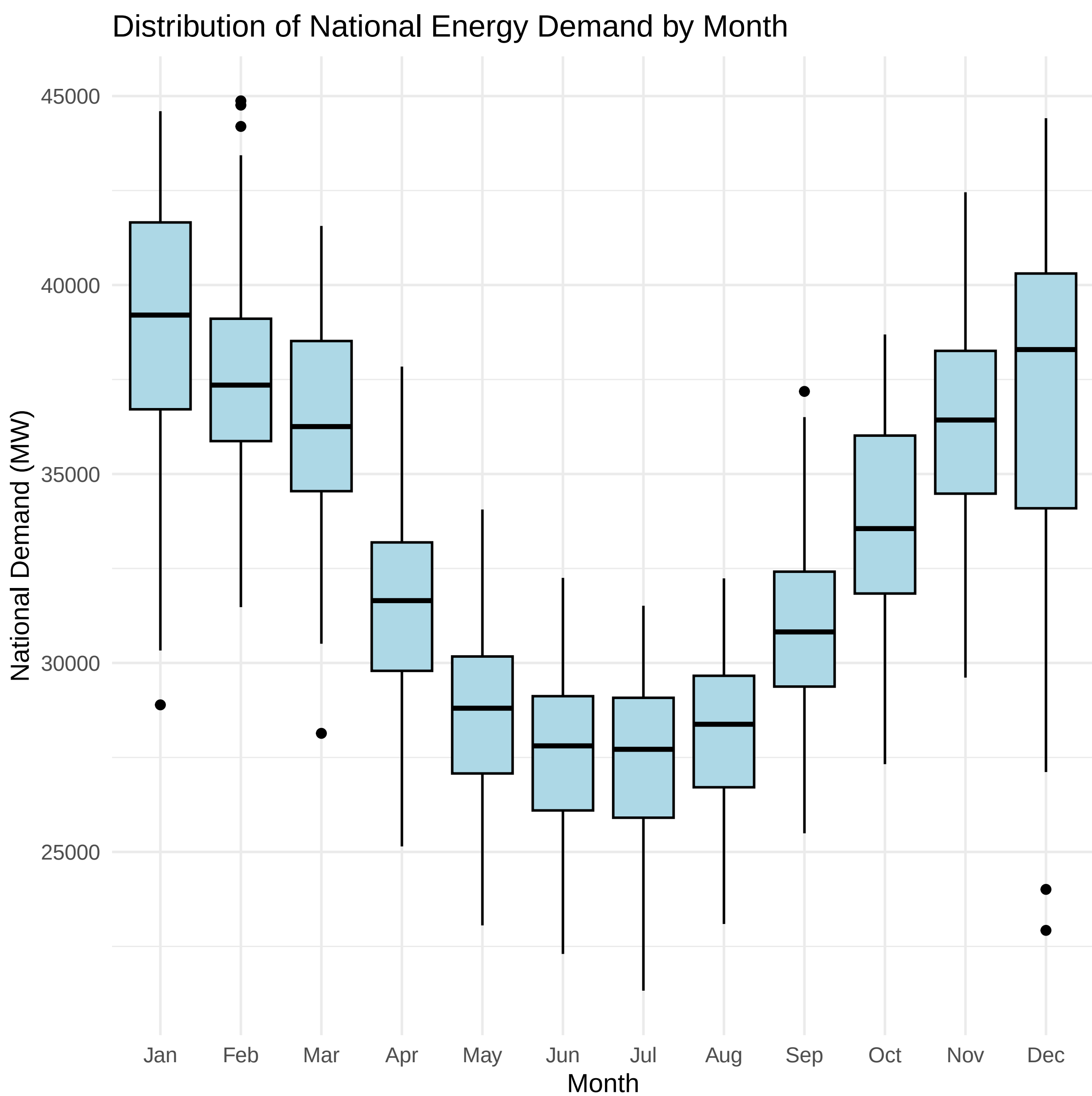
2. Fitting Log Normal Distribution:

Log-Normal distribution is also often used for positively skewed data.

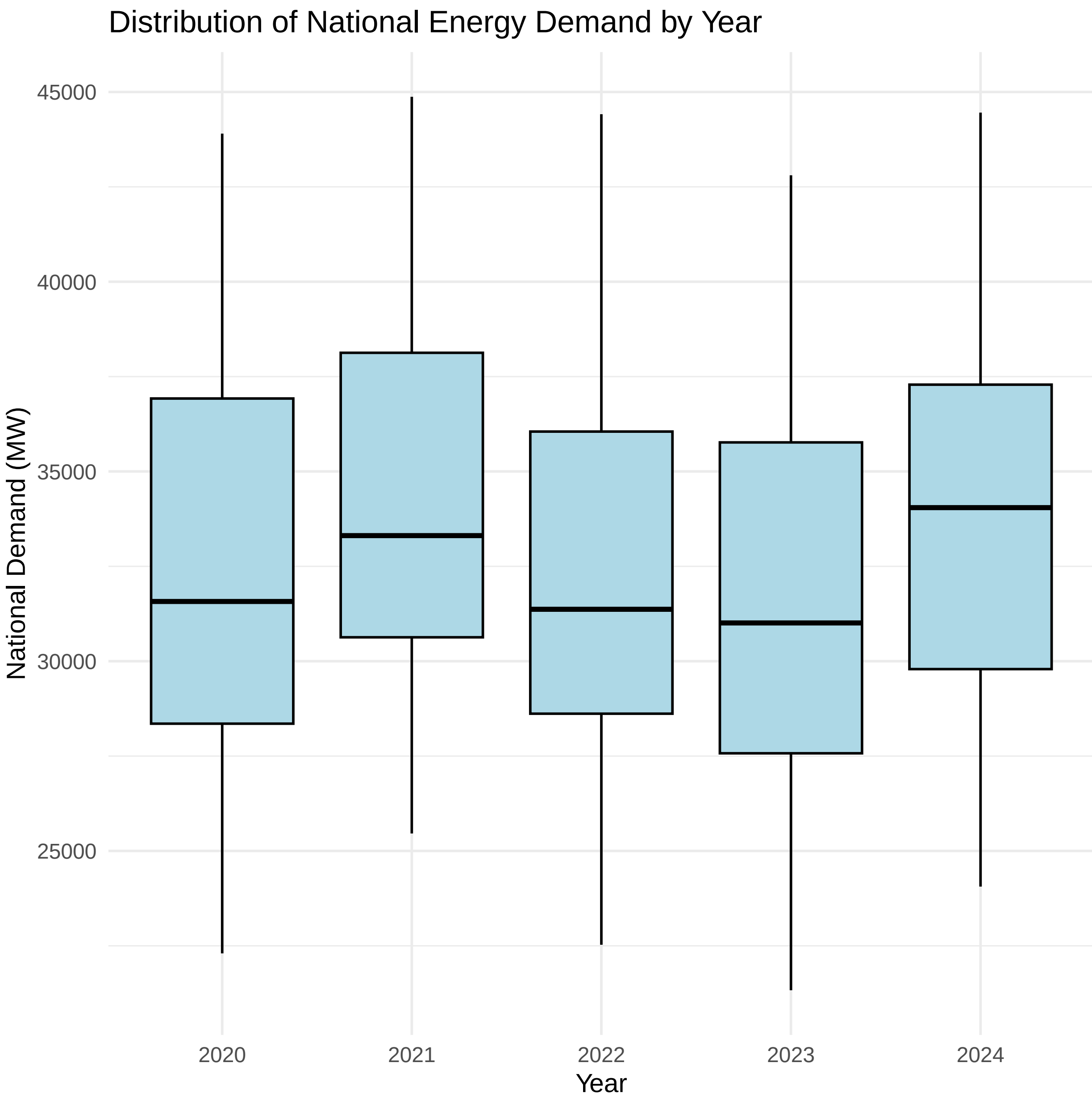
However from the KS test result, we reject H0. This concludes log normal is not a good fit for the energy data.

Distribution by Month and Year

```
# Boxplot by Month
energy_data$month <- factor(month(energy_data$date, label = TRUE), levels = month.abb)
ggplot(energy_data, aes(x = month, y = national_demand)) +
  geom_boxplot(fill = "lightblue", color = "black") +
  labs(title = "Distribution of National Energy Demand by Month",
       x = "Month", y = "National Demand (MW)") +
  theme_minimal()
```




```r
# Boxplot by Year
energy_data$year <- factor(year(energy_data$date))
ggplot(energy_data, aes(x = year, y = national_demand)) +
  geom_boxplot(fill = "lightblue", color = "black") +
  labs(title = "Distribution of National Energy Demand by Year",
       x = "Year", y = "National Demand (MW)") +
  theme_minimal()
```

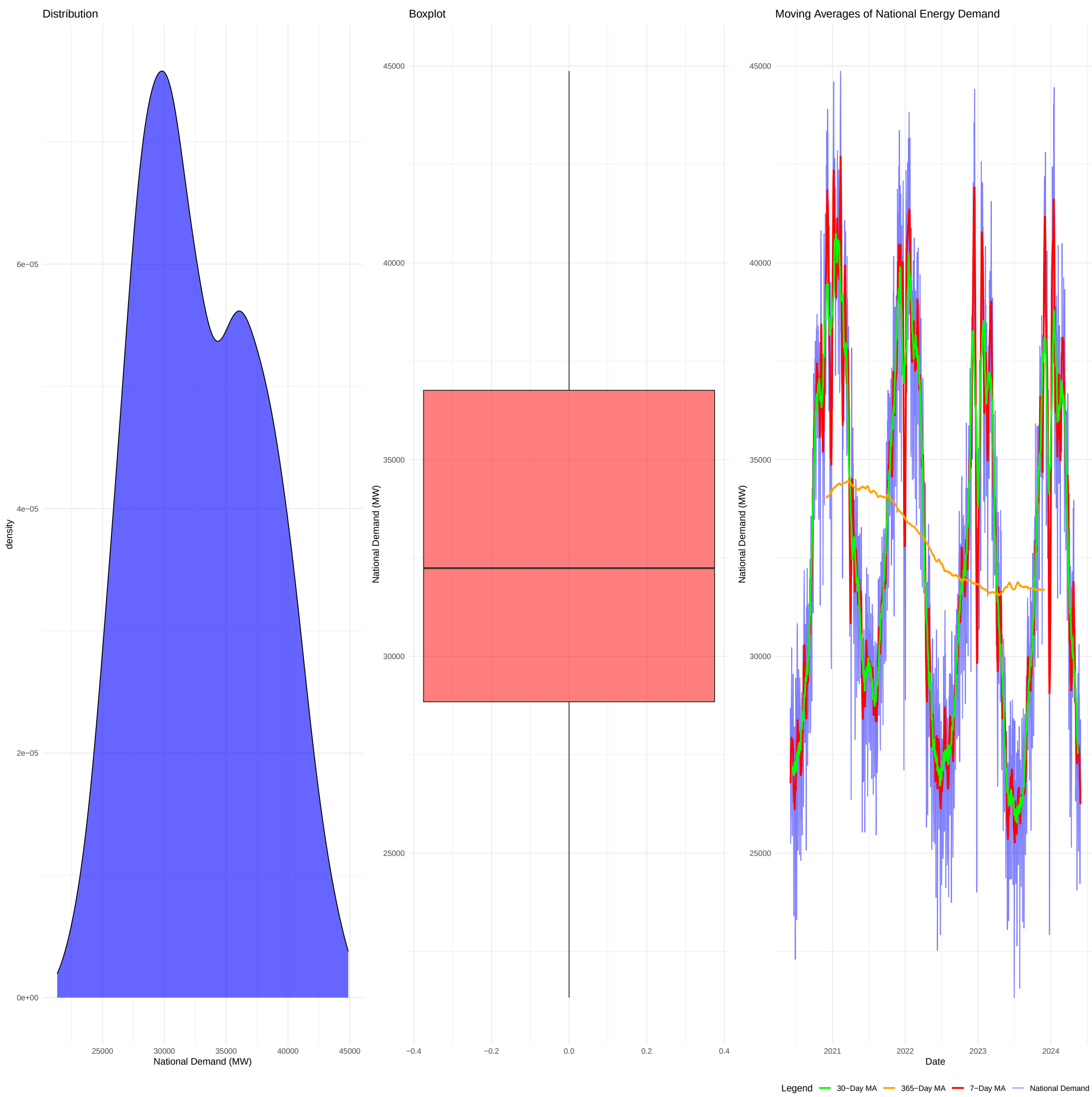



```
#############################################################################
#############################################################################
```

Added to the report 1

```
#charts combinations

dens <- ggplot(energy_data) +
  geom_density(aes(x = national_demand, y = after_stat(density)),fill = "blue", alpha = 0.6, color = "bl
  labs(title = "Distribution", x = "National Demand (MW)") +
  theme_minimal()

bplot <- ggplot(energy_data, aes(y = national_demand)) +
  geom_boxplot(fill = "red", alpha = 0.5) +
  labs(title = "Boxplot", y = "National Demand (MW)") +
```

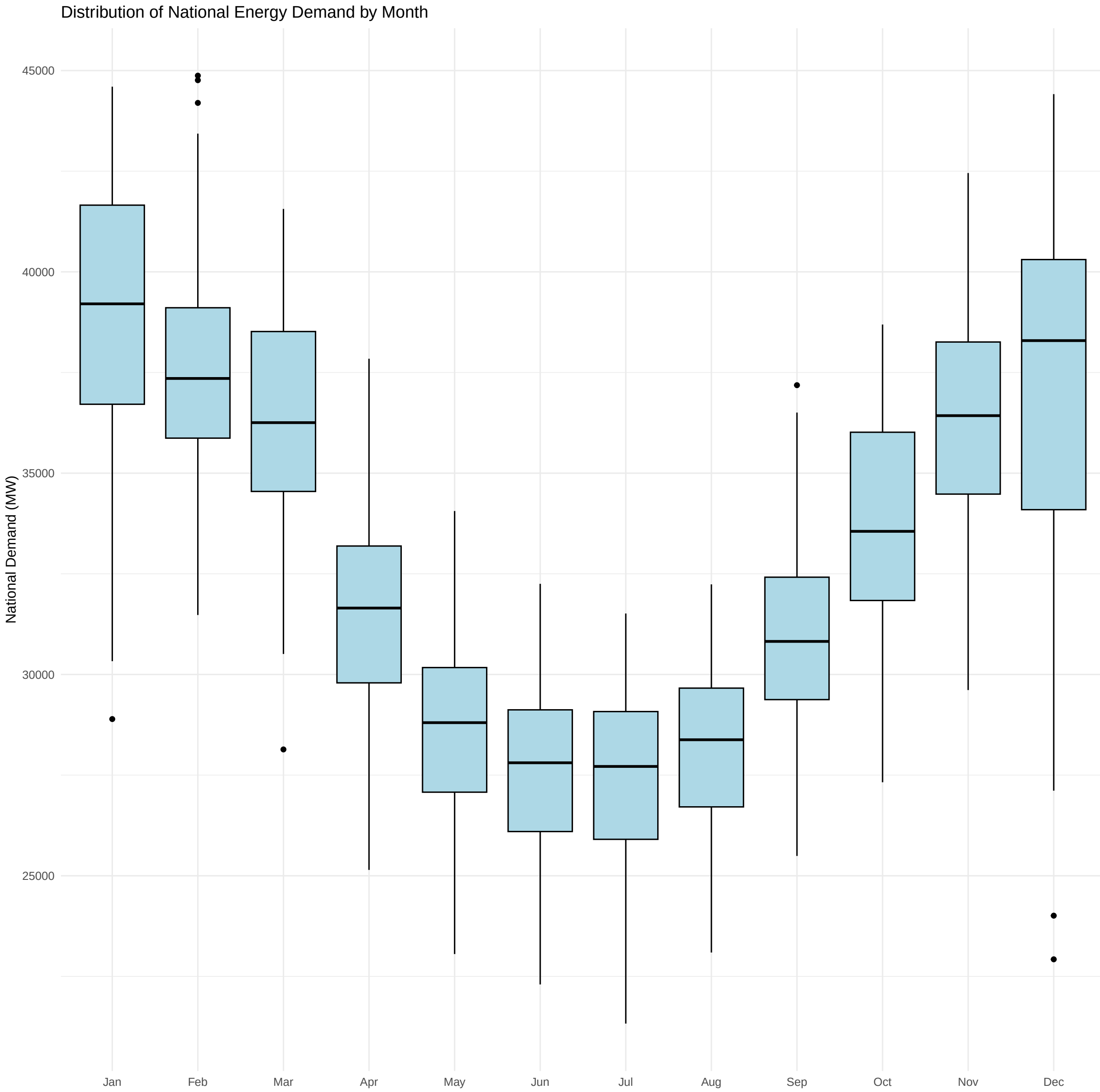
```r
  theme_minimal()

# Moving averages
energy_data <- energy_data %>%
  mutate(ma7 = zoo::rollmean(national_demand, k = 7, fill = NA),  # 7-day moving average
         ma30 = zoo::rollmean(national_demand, k = 30, fill = NA), # 30-day moving average
         ma365 = zoo::rollmean(national_demand, k = 365, fill = NA)) # 365-day moving average

ts <- ggplot(energy_data, aes(x = date)) +
  geom_line(aes(y = national_demand, color = "National Demand"), alpha = 0.5) +
  geom_line(aes(y = ma7, color = "7-Day MA"), size = 1) +
  geom_line(aes(y = ma30, color = "30-Day MA"), size = 1) +
  geom_line(aes(y = ma365, color = "365-Day MA"), size = 1) +
  labs(title = "Moving Averages of National Energy Demand",
       x = "Date",
       y = "National Demand (MW)",
       color = "Legend") +  # Add legend title
  scale_color_manual(values = c("National Demand" = "blue",
                                "7-Day MA" = "red",
                                "30-Day MA" = "green",
                                "365-Day MA" = "orange")) +
  theme_minimal()+
  theme(legend.position = "bottom")


(dens | bplot | ts)
```



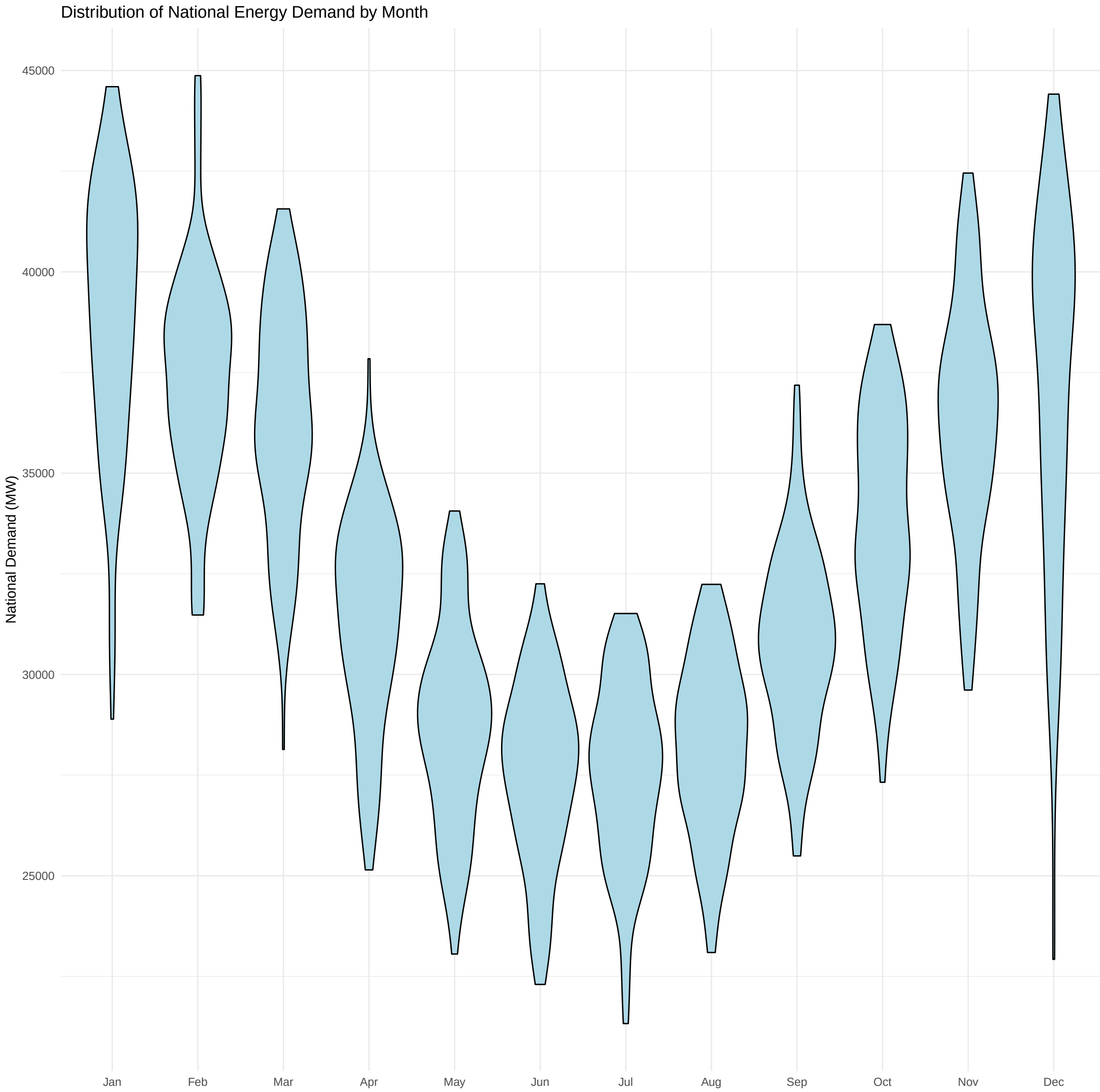
Added to the report 2

Boxplot/Violin plot Monthly split - seasonality

```
energy_data$month <- factor(month(energy_data$date, label = TRUE), levels = month.abb)
ggplot(energy_data, aes(x = month, y = national_demand)) +
  geom_boxplot(fill = "lightblue", color = "black") +
  labs(title = "Distribution of National Energy Demand by Month",
       x = "", y = "National Demand (MW)") +
  theme_minimal()
```



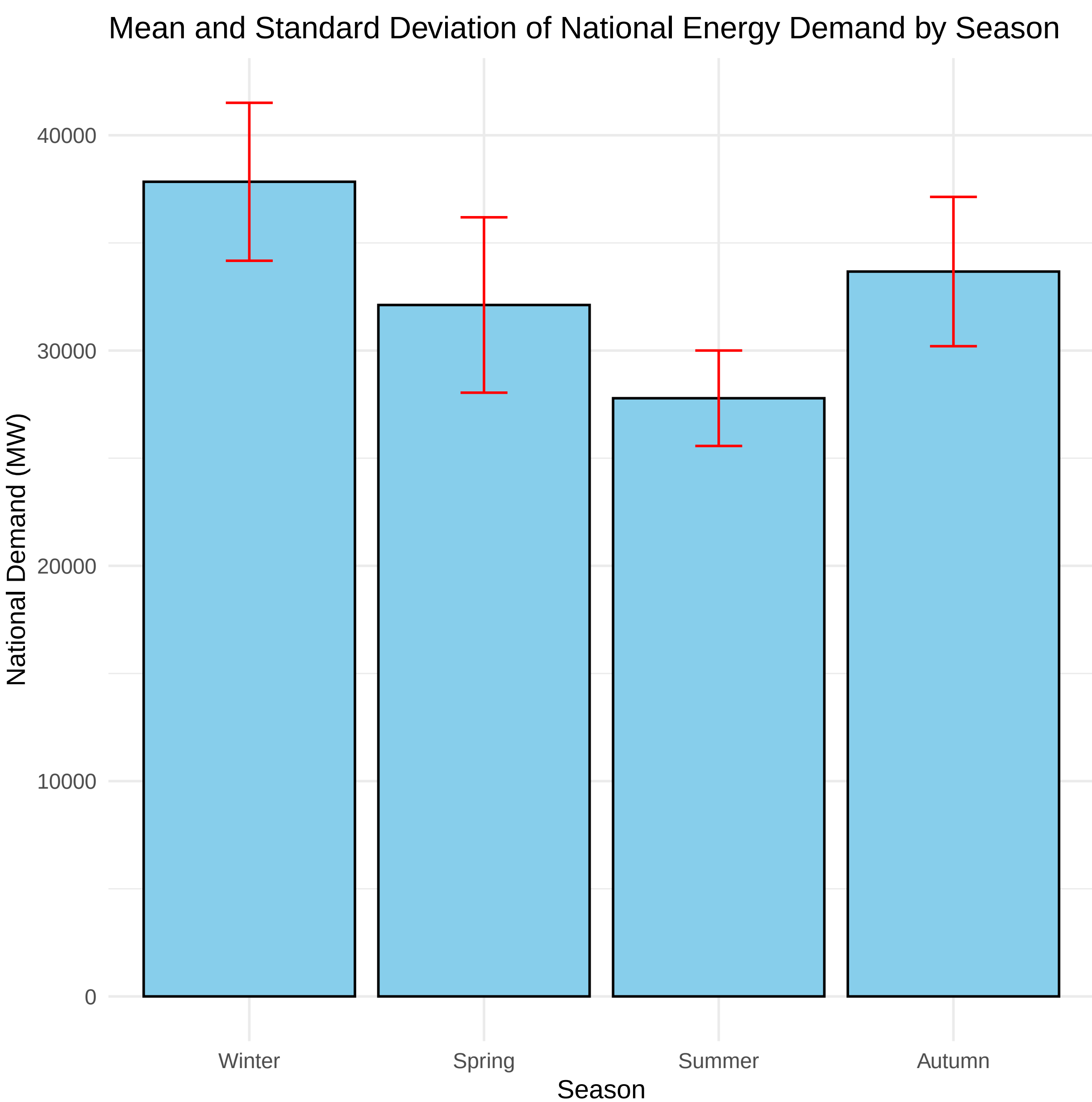

```r
energy_data$month <- factor(month(energy_data$date, label = TRUE), levels = month.abb)
ggplot(energy_data, aes(x = month, y = national_demand)) +
  geom_violin(fill = "lightblue", color = "black") +
  labs(title = "Distribution of National Energy Demand by Month",
       x = "", y = "National Demand (MW)") +
  theme_minimal()
```



Seasonal Analysis

```r
# Define seasons
energy_data <- energy_data %>%
  mutate(season = case_when(
    month(date) %in% c(12, 1, 2) ~ "Winter",
    month(date) %in% c(3, 4, 5) ~ "Spring",
    month(date) %in% c(6, 7, 8) ~ "Summer",
    month(date) %in% c(9, 10, 11) ~ "Autumn"
  ))

# Ensure correct order of seasons
energy_data$season <- factor(energy_data$season,
                            levels = c("Winter", "Spring", "Summer", "Autumn"))
```

```r
# Compute seasonal mean and standard deviation
seasonal_stats <- energy_data %>%
  group_by(season) %>%
  summarise(mean_demand = mean(national_demand, na.rm = TRUE),
            sd_demand = sd(national_demand, na.rm = TRUE))

# Plot seasonal mean with error bars (standard deviation)
ggplot(seasonal_stats, aes(x = season, y = mean_demand)) +
  geom_bar(stat = "identity", fill = "skyblue", color = "black") +
  geom_errorbar(aes(ymin = mean_demand - sd_demand, ymax = mean_demand + sd_demand),
                width = 0.2, color = "red") +
  labs(title = "Mean and Standard Deviation of National Energy Demand by Season",
       x = "Season", y = "National Demand (MW)") +
  theme_minimal()
```



Conclusion:

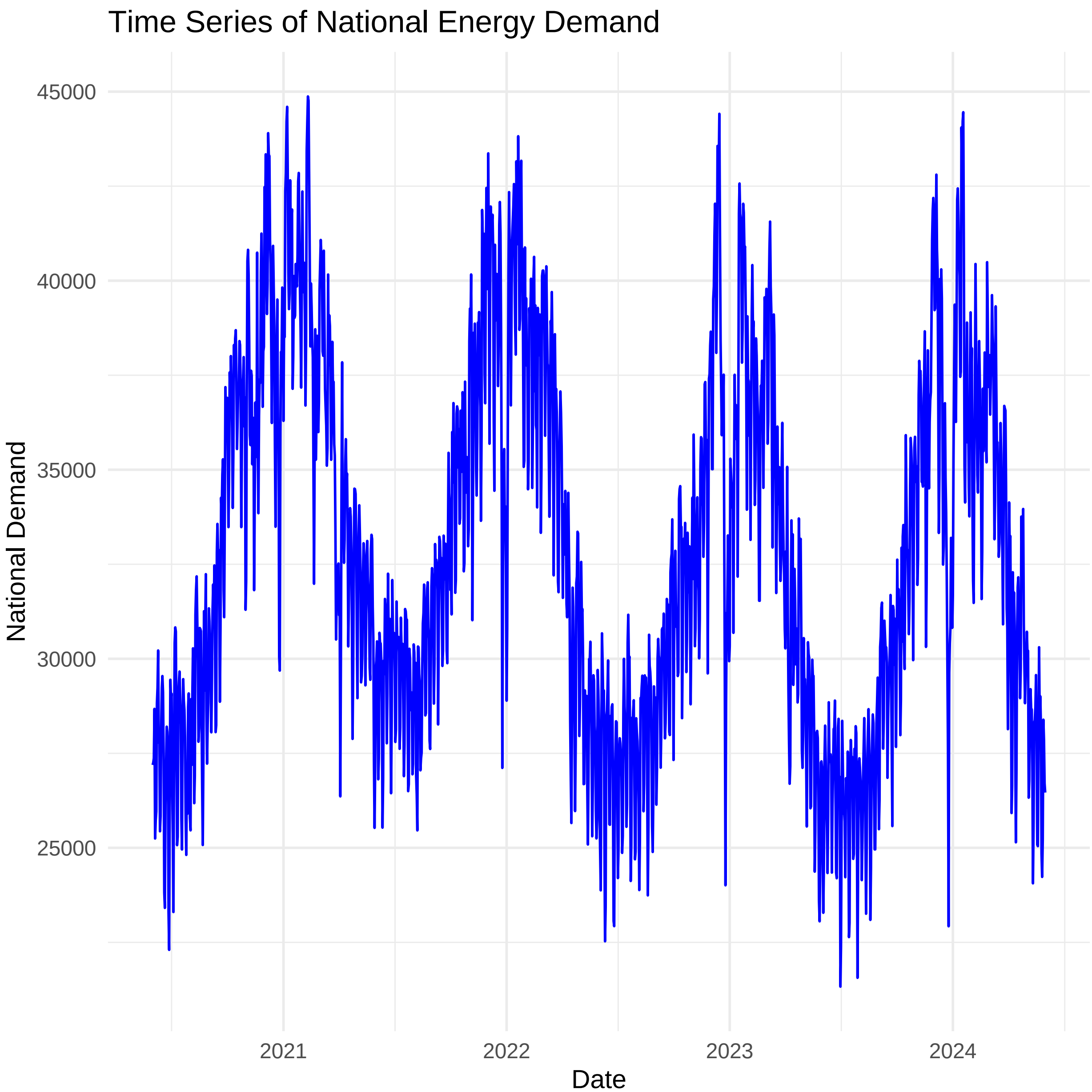
The bar plot represents the mean of the energy demand.

We can see in both winter and autumn, there is high demand for energy.

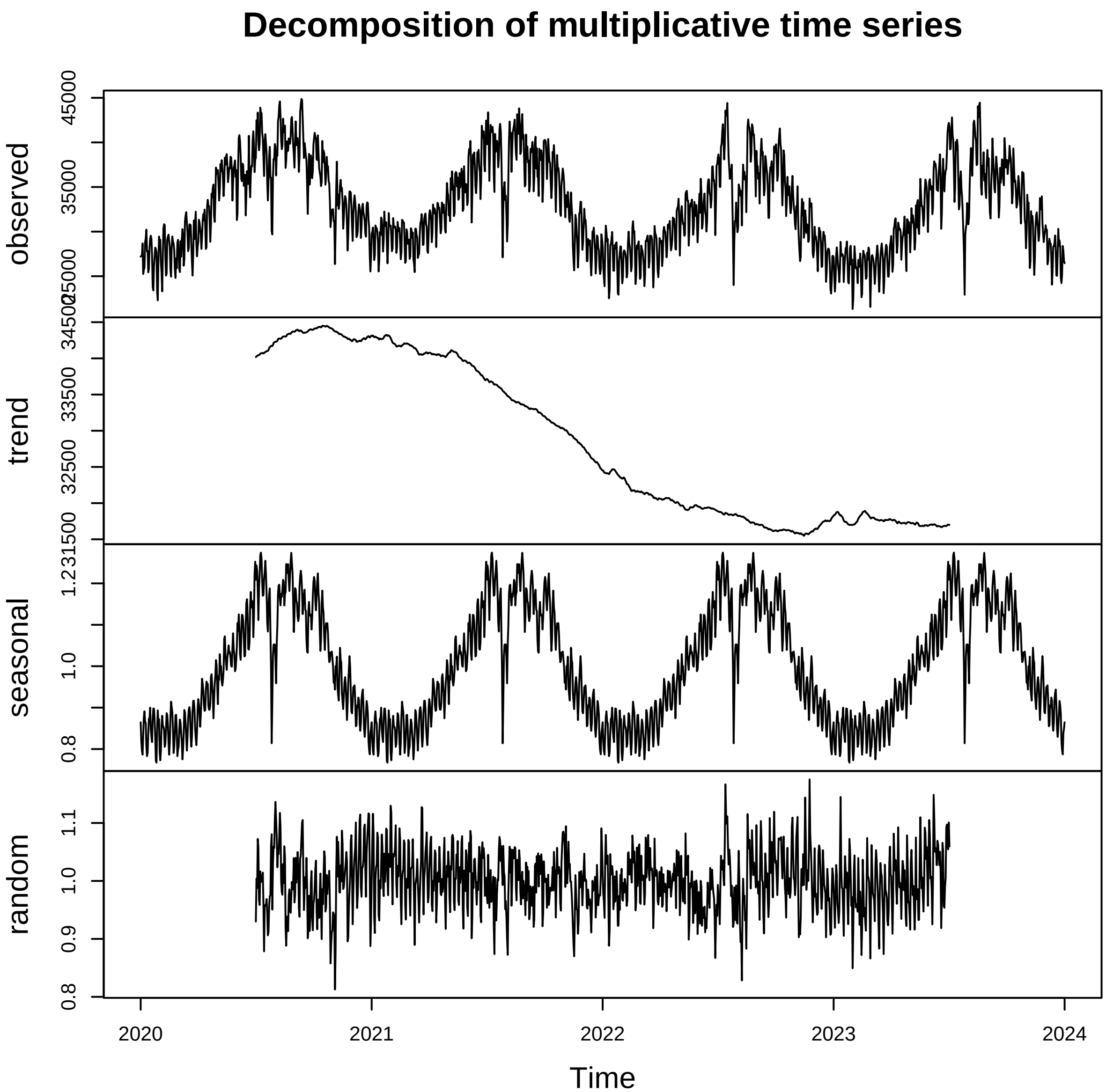
Winter, Spring and Autumn have the largest fluctuation of the demand (shown by the red error bars).

Time Series Analysis

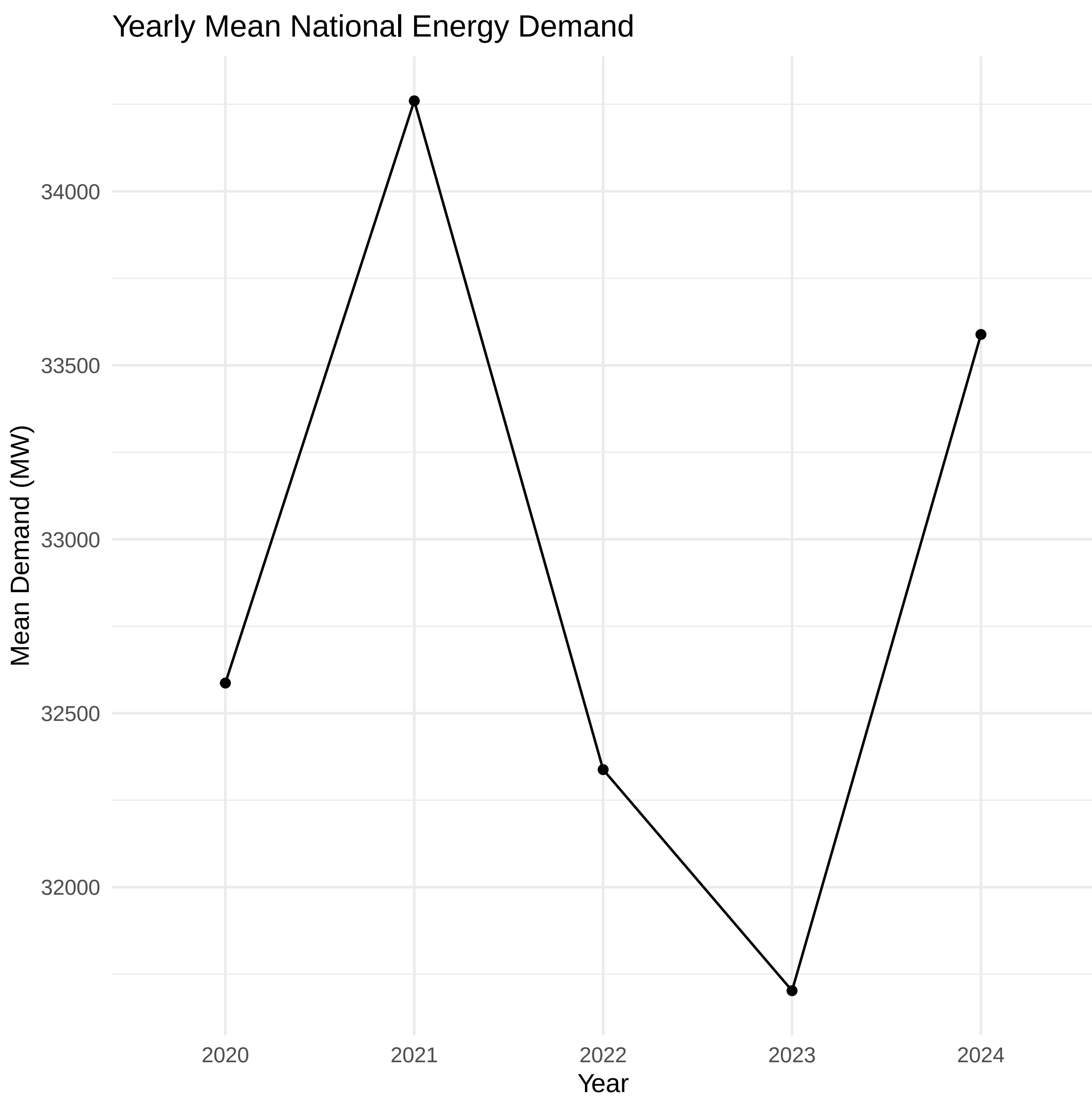
```r
# Create time series object
ts_data <- ts(energy_data$national_demand,
              start = c(2020, 1),
              frequency = 365)

# Time series plot with ggplot2
ggplot(energy_data, aes(x = date, y = national_demand)) +
  geom_line(color = "blue") +
  labs(title = "Time Series of National Energy Demand",
       x = "Date",
       y = "National Demand") +
  theme_minimal()
```




```r
# Decompose time series
decomposed_ts <- decompose(ts_data, type = "multiplicative")

# Plot the decomposition
plot(decomposed_ts)
```




```r
# Yearly mean trend
yearly_mean <- energy_data %>%
  group_by(year) %>%
  summarise(mean_demand = mean(national_demand))

ggplot(yearly_mean, aes(x = year, y = mean_demand)) +
  geom_line(group = 1) +
  geom_point() +
  labs(title = "Yearly Mean National Energy Demand",
       x = "Year",
       y = "Mean Demand (MW)") +
  theme_minimal()
```




```r
# Check for Stationarity with Augmented Dickey-Fuller Test
adf_test_result <- adf.test(ts_data)
print(adf_test_result)
```

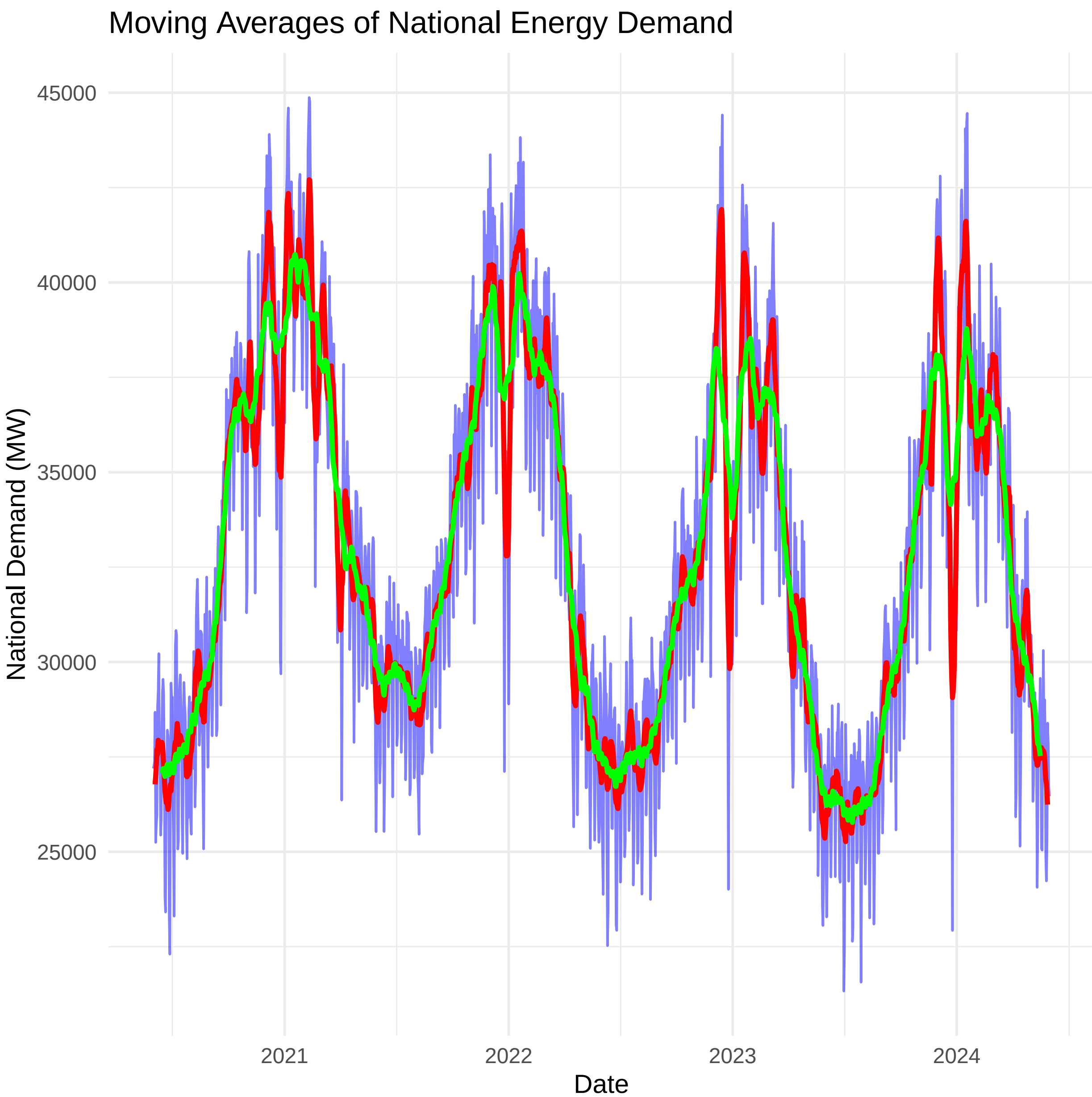
```
##
##  Augmented Dickey-Fuller Test
##
## data:  ts_data
## Dickey-Fuller = -3.5576, Lag order = 11, p-value = 0.03675
## alternative hypothesis: stationary
```

```r
# Moving averages
energy_data <- energy_data %>%
  mutate(ma7 = zoo::rollmean(national_demand, k = 7, fill = NA),  # 7-day moving average
         ma30 = zoo::rollmean(national_demand, k = 30, fill = NA)) # 30-day moving average
```

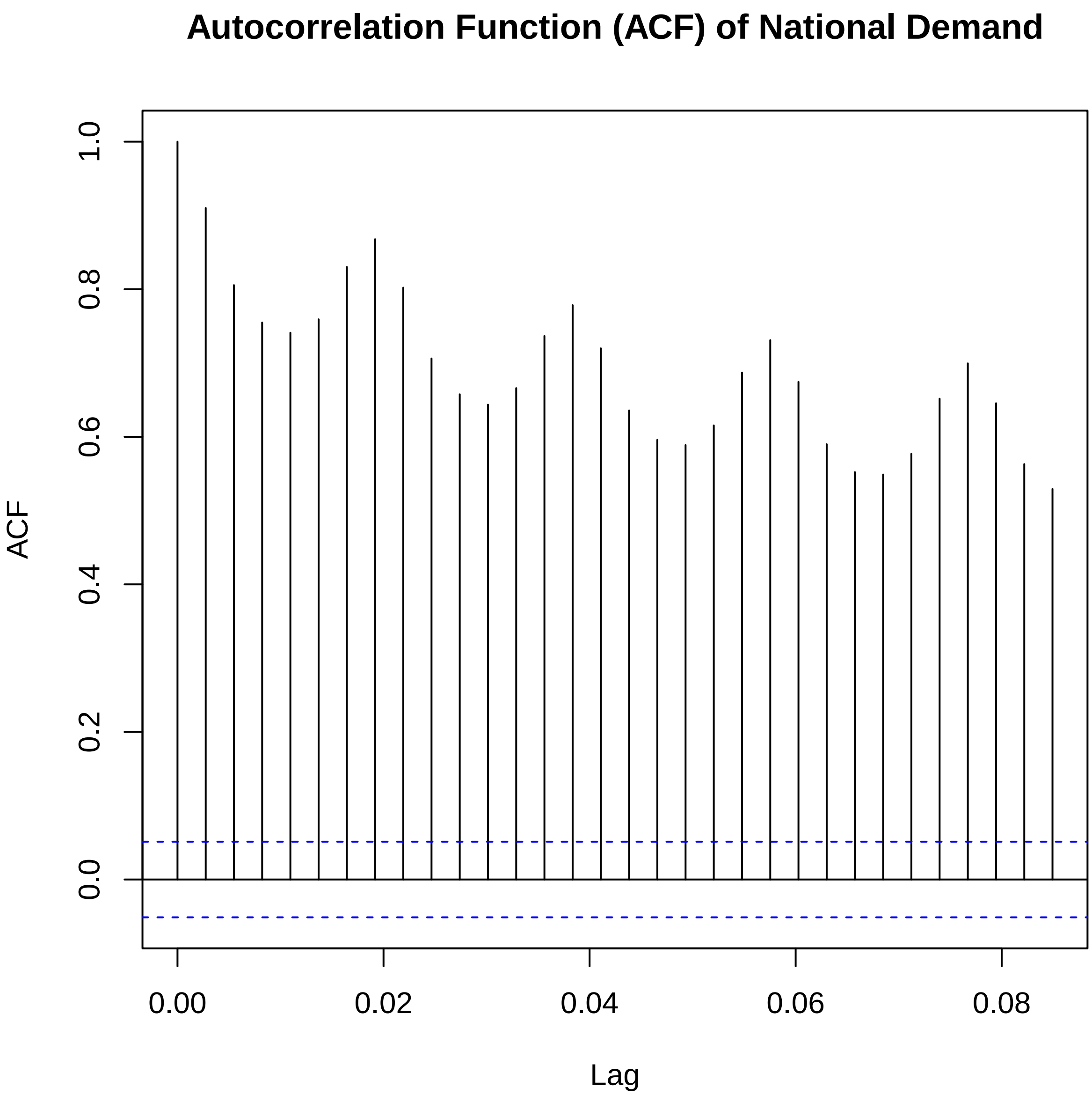
```
ggplot(energy_data, aes(x = date)) +
  geom_line(aes(y = national_demand), color = "blue", alpha = 0.5) +
  geom_line(aes(y = ma7), color = "red", size = 1) +
  geom_line(aes(y = ma30), color = "green", size = 1) +
  labs(title = "Moving Averages of National Energy Demand",
       x = "Date",
       y = "National Demand (MW)") +
  theme_minimal()
```

```
## Warning: Removed 6 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

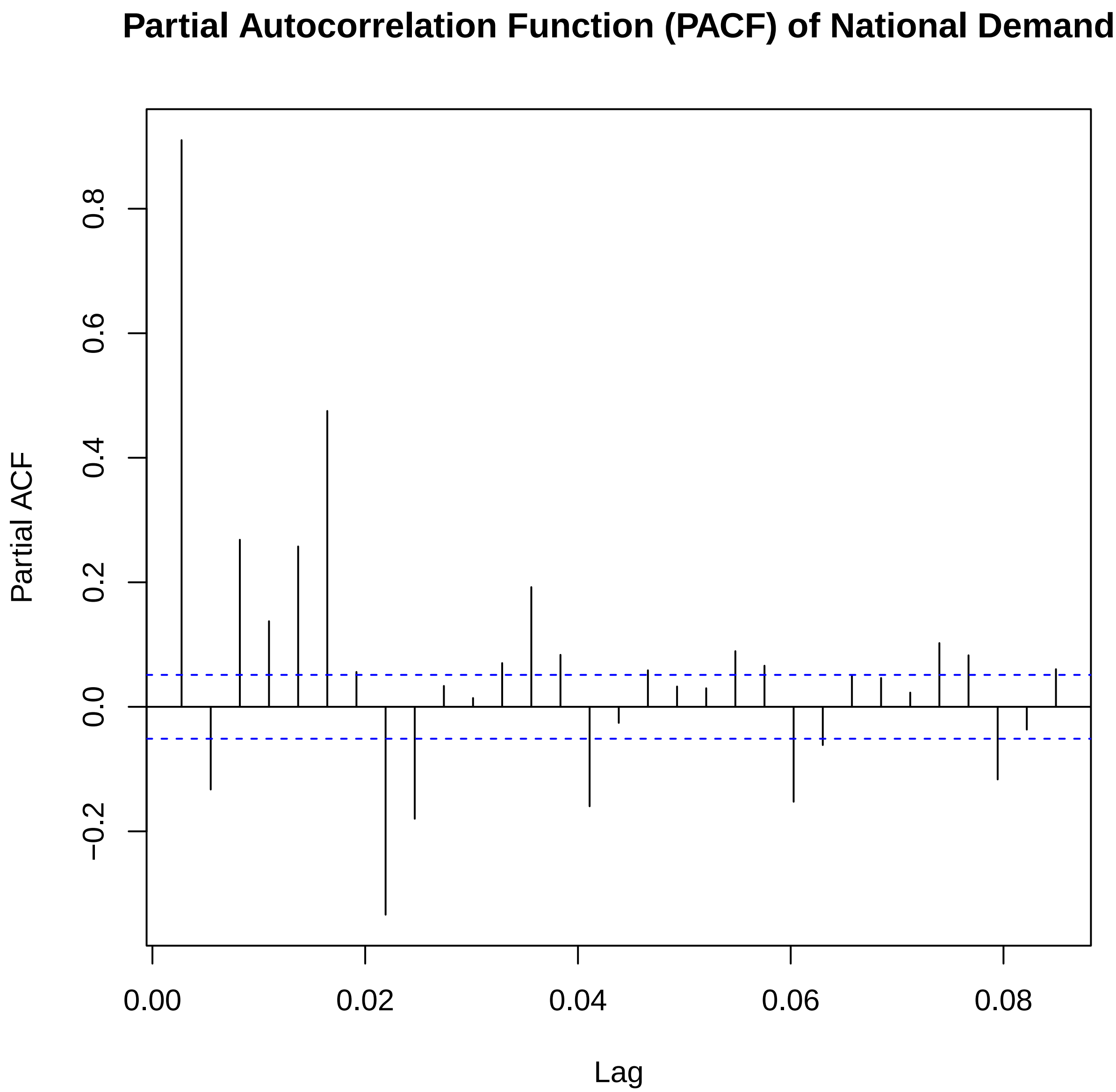
```
## Warning: Removed 29 rows containing missing values or values outside the scale range
## (`geom_line()`).
```



```r
# ACF and PACF
acf(ts_data, main = "Autocorrelation Function (ACF) of National Demand")
```




```r
pacf(ts_data, main = "Partial Autocorrelation Function (PACF) of National Demand")
```


Partial Autocorrelation Function (PACF) of National Demand
Partial ACF
-0.2
0.0
0.2
0.4
0.6
0.8
0.00
0.02
0.04
0.06
0.08
Lag

Conclusion:

If we compare the overall trend from the decomposition graph vs the trend of the yearly mean, the decomposition graph suggests strictly decreasing demand, while the yearly mean doesn't suggest this conclusion.

The decomposition graph did suggest seasonality.

The noise graph looks completely random, where there is no pattern.

Given the p-value < 0.05 from the ADF test, we reject H0, and we conclude that the time series is stationary.

Moving Average Analysis:

From the original blue graph -> we can see that there are a lot of noise, and sudden fluctuations (could be due to sudden change / requirement of energy).

From the red plot (7-day MA) -> smooths out some fluctuations.

From the green plot (30-day MA) -> smooths out more fluctuations, and shows clear seasonality.

SARIMA Model Fitting

```r
# Fit SARIMA model
#sarima_model <- auto.arima(ts_data, seasonal = TRUE,
#                           stepwise = FALSE, approximation = FALSE)
#summary(sarima_model)

# Forecast
#forecast_sarima <- forecast(sarima_model, h = 365)  # Forecast for next year
#autoplot(forecast_sarima)
```

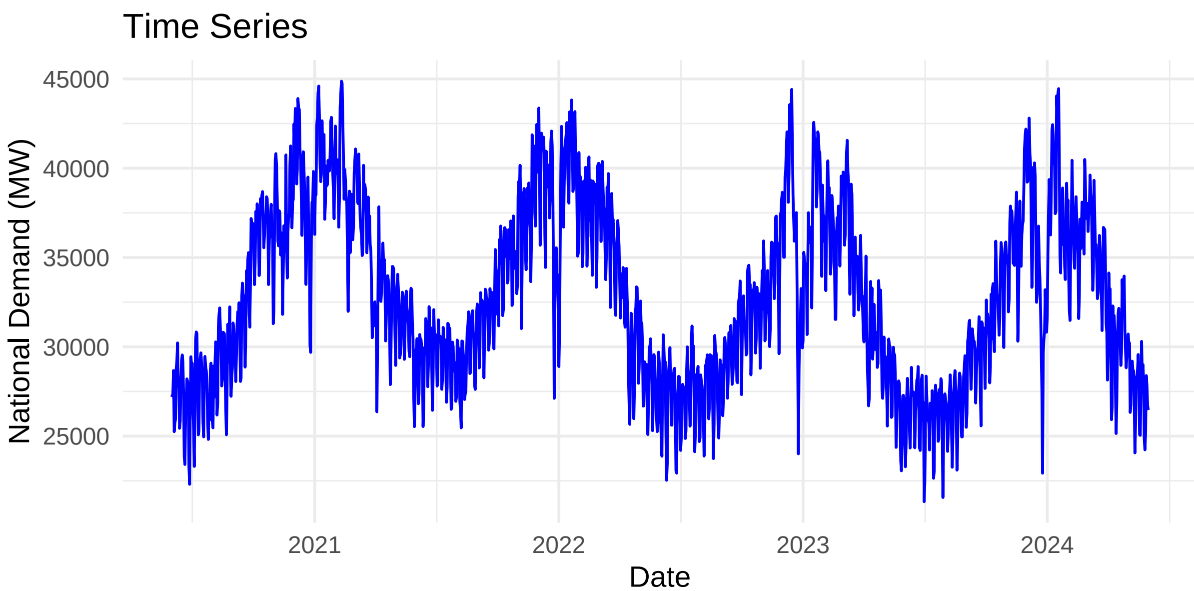
Added to the report 2

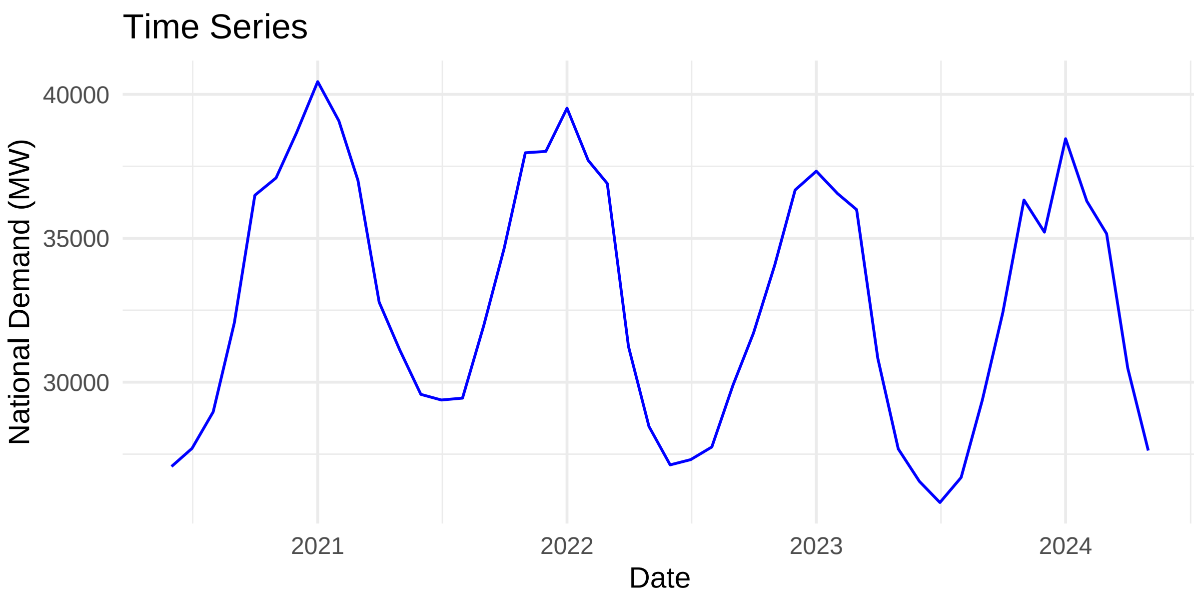
```r
# Monthly mean trend
month_year_mean <- energy_data %>%
  group_by(month,year) %>%
  summarise(mean_demand = mean(national_demand))
```

```
## `summarise()` has grouped output by 'month'. You can override using the
## `.groups` argument.
```

```r
month_year_mean <- month_year_mean %>%
  arrange(year,month)

ts_month_year_mean <- ts(month_year_mean$mean_demand,
              start = c(2020, 6),
              frequency = 12)
month_year_mean$date <- as.Date(time(ts_month_year_mean))

ts <- ggplot(energy_data, aes(x = date, y = national_demand)) +
  geom_line(color = "blue") +
  labs(title = "Time Series",
       x = "Date", y = "National Demand (MW)") +
```

```
  theme_minimal()

ts_month <- ggplot(month_year_mean, aes(x = date, y = mean_demand)) +
  geom_line(color = "blue") +
  labs(title = "Time Series",
       x = "Date", y = "National Demand (MW)") +
  theme_minimal()


(ts) / (ts_month)
```

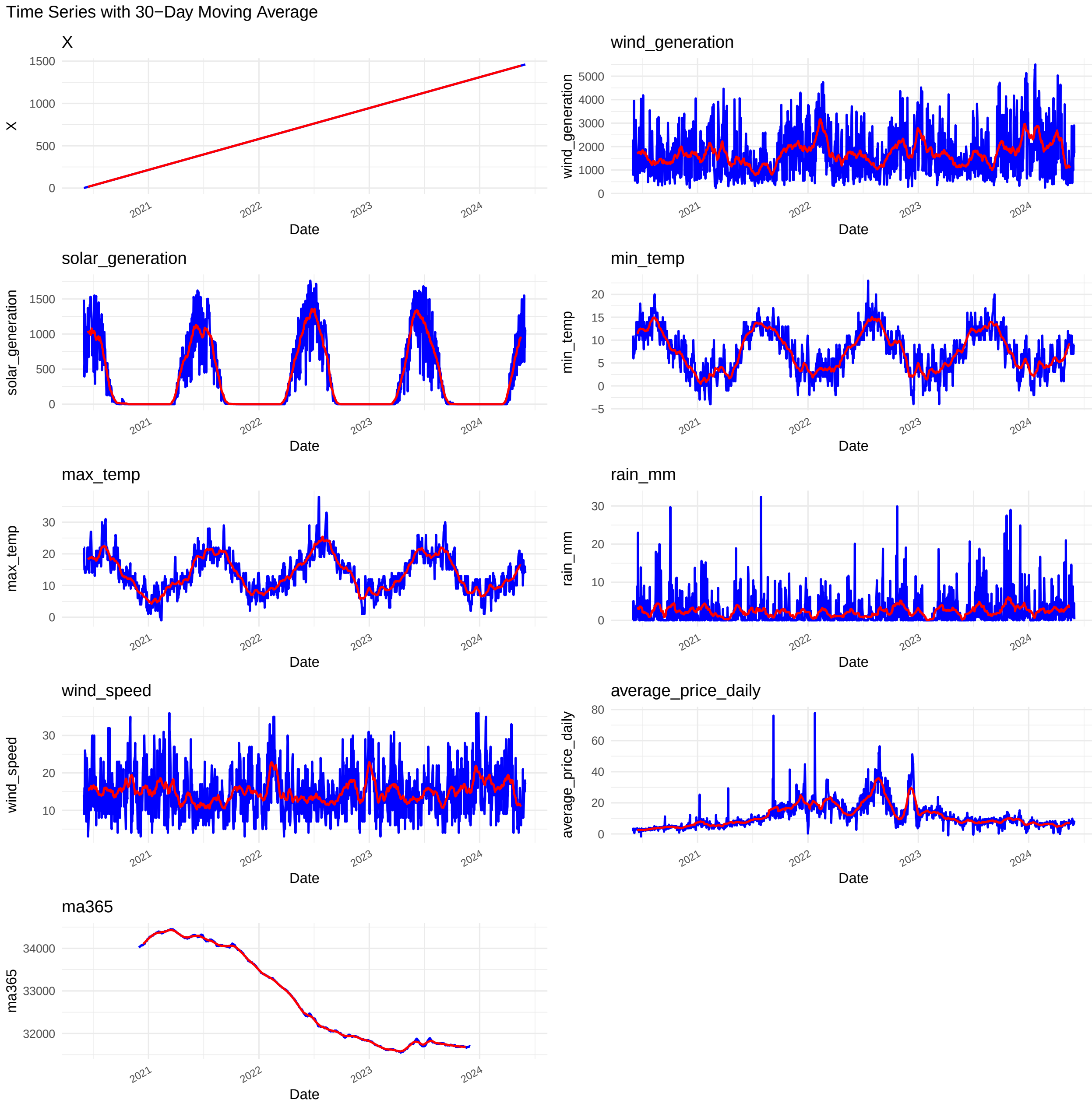




3. Multivariate Exploratory Data Analysis

Time Series of All Variables

```r
# Variables to exclude
exclude_cols <- c("national_demand", "date", "month", "year", "season", "ma7", "ma30")

# Identify columns to analyze
columns_to_consider <- setdiff(names(energy_data), exclude_cols)

# MA time window
MA_TIME <- 30

# Create list of ggplots
plot_list <- lapply(columns_to_consider, function(col) {
  ggplot(energy_data, aes(x = date, y = .data[[col]])) +
    geom_line(color = "blue", linewidth = 0.8) +
    geom_line(aes(y = rollmean(.data[[col]], k=MA_TIME, fill = NA)),
              color = "red", linewidth = 0.8) +
    labs(title = col, x = "Date", y = col) +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 30, hjust = 1, size = 8))
})

# Combine all plots into a grid layout
combined_plot <- wrap_plots(plot_list, ncol = 2) +
  plot_annotation(title = "Time Series with 30-Day Moving Average")

# Print the combined plot
print(combined_plot)
```



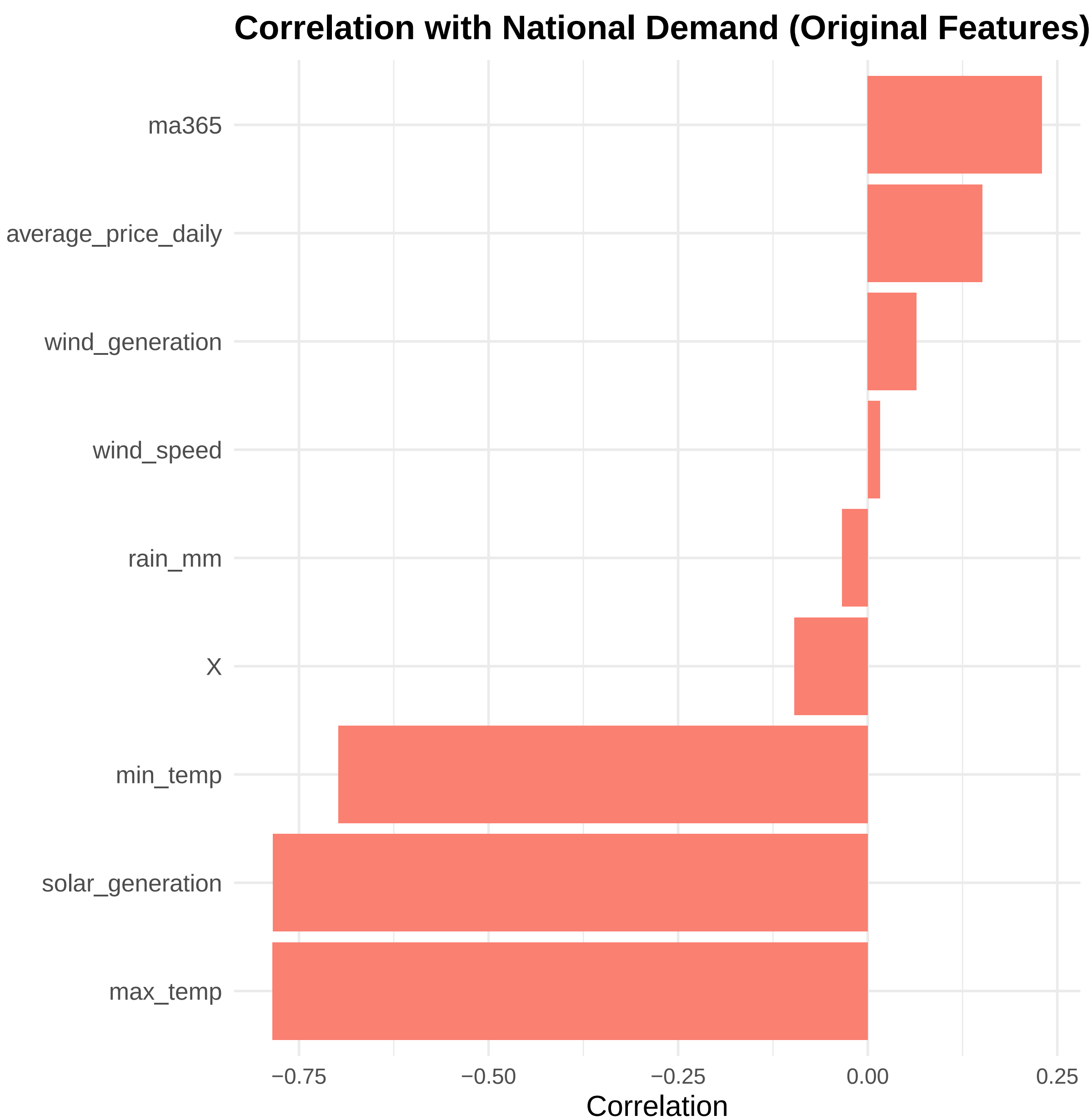
Correlation Analysis

```
# Calculate Spearman correlation for original features
correlations <- sapply(energy_data[columns_to_consider],
                       function(column) cor(column, energy_data$national_demand,
                                            method = "spearman",
                                            use = "complete.obs"))

# Display correlations
print(correlations)
```

```
##                X     wind_generation     solar_generation            min_temp
##      -0.09688236          0.06421924          -0.78463666         -0.69836736
```

```
##            max_temp             rain_mm          wind_speed average_price_daily
##         -0.78485635         -0.03406155          0.01612035          0.15122004
##               ma365
##          0.22966760
```

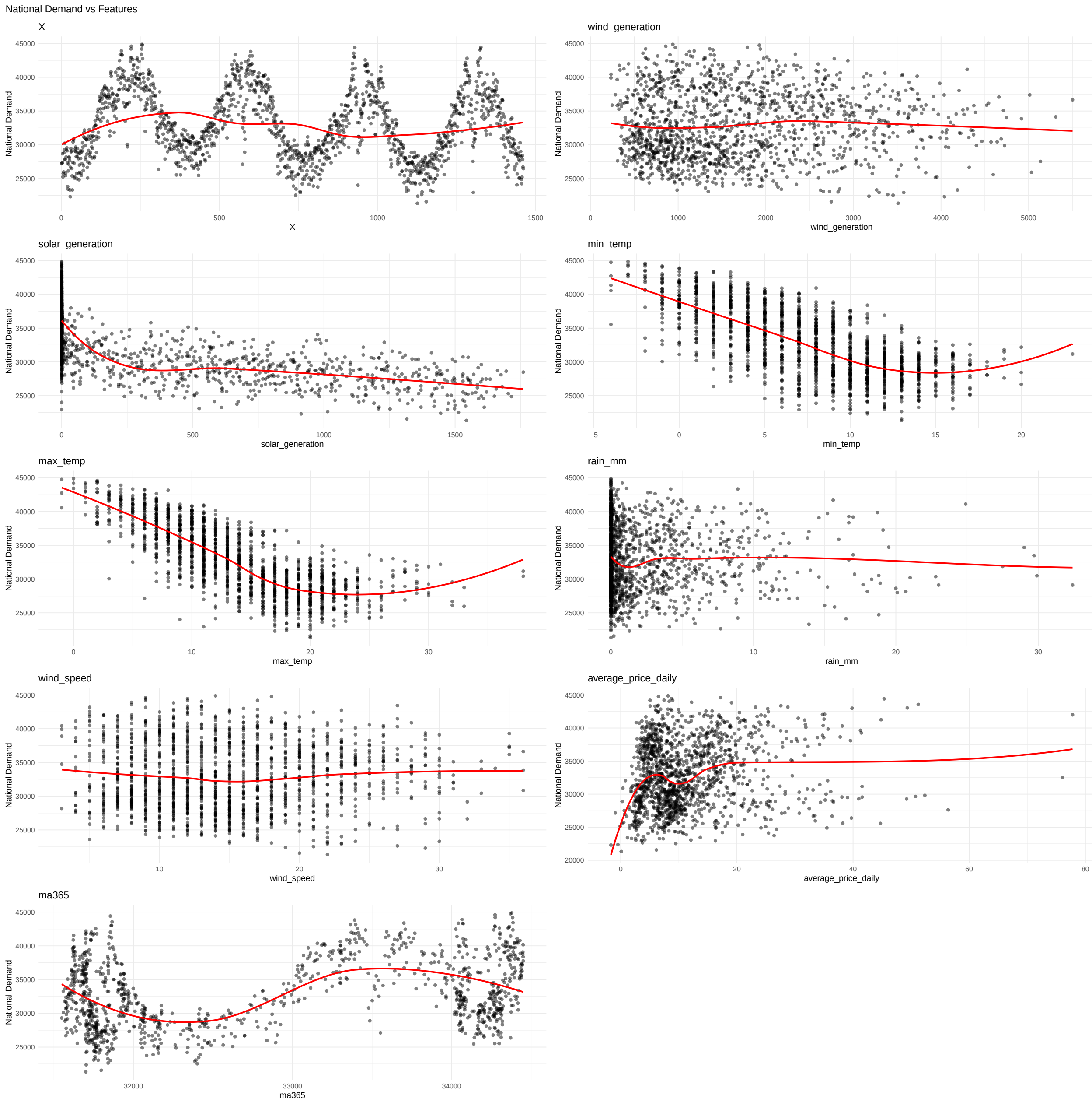
```r
# Convert correlations to data frame for plotting
correlations_df <- data.frame(
  feature = names(correlations),
  correlation = as.numeric(correlations)
)

# Plot correlations
ggplot(correlations_df, aes(x = correlation, y = reorder(feature, correlation))) +
  geom_bar(stat = "identity", fill = "salmon") +
  theme_minimal() +
  labs(
    title = "Correlation with National Demand (Original Features)",
    x = "Correlation",
    y = ""
  ) +
  theme(
    axis.text.y = element_text(size = 10),
    axis.title.y = element_text(size = 12),
    axis.title.x = element_text(size = 12),
    plot.title = element_text(size = 14, face = "bold"),
    legend.position = "none"
  )
```



Plot national_demand against Original Features

```r
# Create list of plots
plot_list <- lapply(columns_to_consider, function(col) {
  ggplot(energy_data, aes(x = .data[[col]], y = national_demand)) +
    geom_point(color = "black", alpha = 0.5) +  # Scatter plot
    geom_smooth(color = "red", method = "loess", se = FALSE) +  # Smooth trend line
    labs(title = paste(col),
         x = col,
         y = "National Demand") +
    theme_minimal()
})

# Combine all plots into a grid using patchwork
combined_plot <- wrap_plots(plot_list, ncol = 2) +
  plot_annotation(title = "National Demand vs Features")
```

```r
# Show the combined plot
print(combined_plot)
```

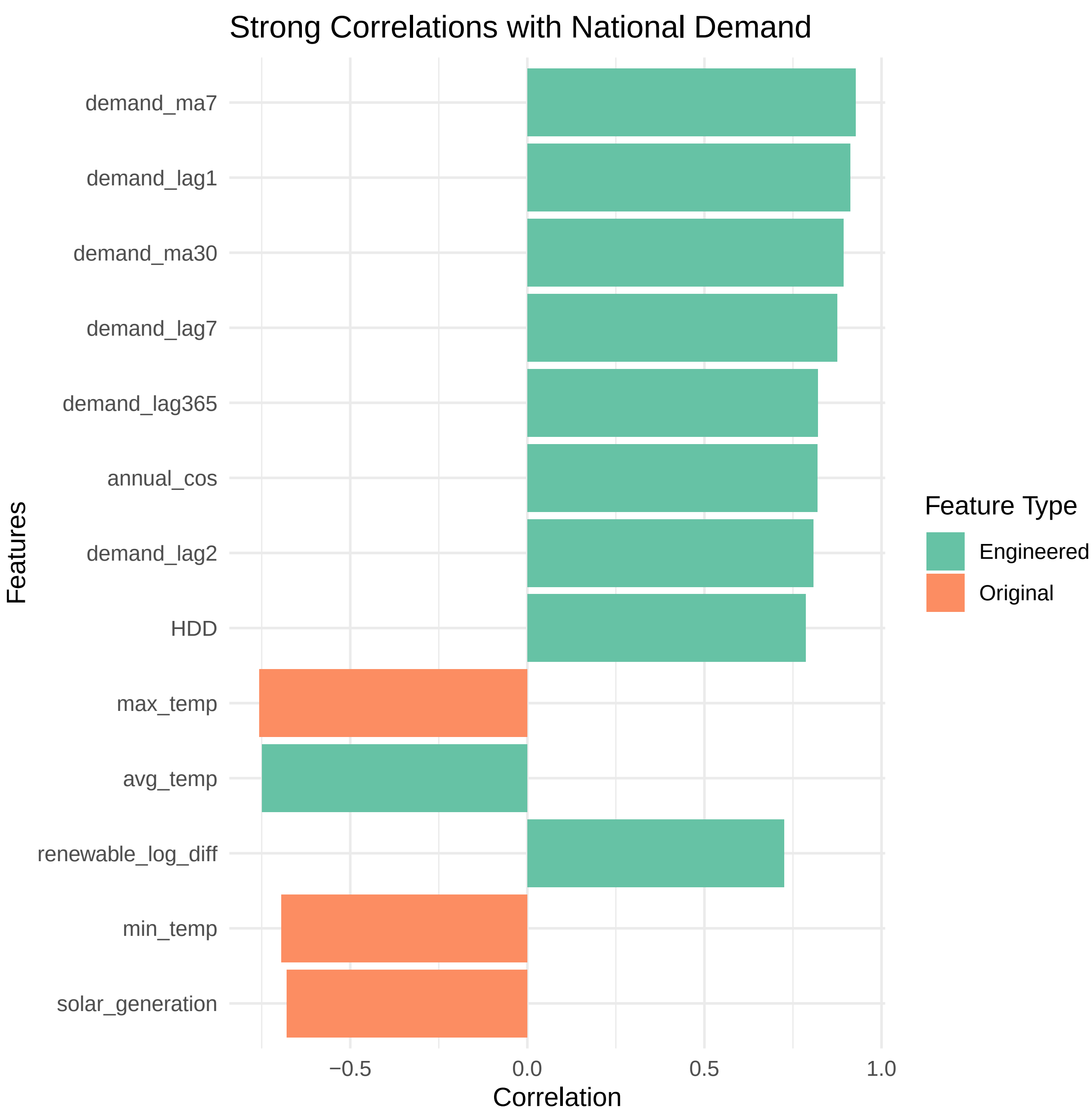


Feature Engineering

```r
# Add engineered features
energy_data <- energy_data %>%
  mutate(
    avg_temp = (min_temp + max_temp) / 2,
    day_of_year = yday(date),
    annual_cos = cos(2 * pi * day_of_year / 365),
    HDD = pmax(15.5 - avg_temp, 0),  # Heating Degree Days (base 15.5°C)

    # Lag features
```

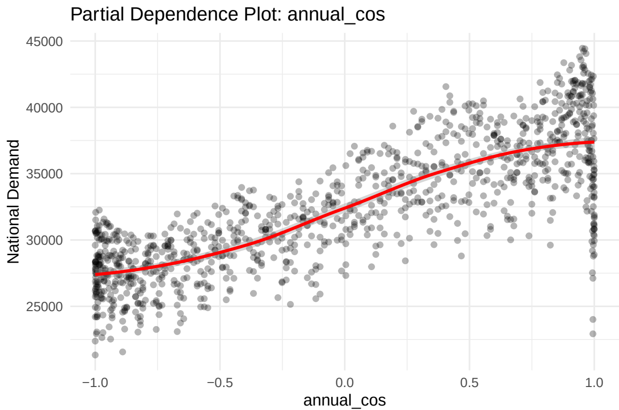
```r
    demand_lag1 = lag(national_demand, 1),
    demand_lag2 = lag(national_demand, 2),
    demand_lag7 = lag(national_demand, 7),
    demand_lag365 = lag(national_demand, 365),

    # Moving averages
    demand_ma7 = if("ma7" %in% names(energy_data)) ma7 else
                 rollmean(national_demand, k = 7, fill = NA, align = "right"),
    demand_ma30 = if("ma30" %in% names(energy_data)) ma30 else
                  rollmean(national_demand, k = 30, fill = NA, align = "right"),
    demand_ma365 = rollmean(national_demand, k = 365, fill = NA, align = "right"),

    # Renewable energy ratio
    renewable_log_diff = log(wind_generation + 1) - log(solar_generation + 1)
  )
if("ma7" %in% names(energy_data)) {
  energy_data <- energy_data %>% dplyr::select(-ma7)
}
if("ma30" %in% names(energy_data)) {
  energy_data <- energy_data %>% dplyr::select(-ma30)
}
# Identify original and engineered features
original_features <- c("wind_generation", "solar_generation", "min_temp",
                      "max_temp", "rain_mm", "wind_speed", "average_price_daily")

# Get all numeric column names
all_numeric_names <- names(select_if(energy_data, is.numeric))

# Create engineered features list (excluding national_demand and original features)
engineered_features <- setdiff(all_numeric_names,
                              c("national_demand", original_features))

# Calculate correlations with national demand
numeric_cols <- energy_data %>%
  select_if(is.numeric) %>%
  select_at(vars(-national_demand))

all_correlations <- data.frame(
  term = names(numeric_cols),
  correlation = sapply(numeric_cols, function(x)
                      cor(x, energy_data$national_demand, use = "pairwise.complete.obs"))
) %>%
  # Add feature type
  mutate(feature_type = case_when(
    term %in% original_features ~ "Original",
    term %in% engineered_features ~ "Engineered",
    TRUE ~ "Other"
  ))

# Filter for strong correlations (|r| > 0.5)
strong_correlations <- all_correlations %>%
  filter(!is.na(correlation) & abs(correlation) > 0.5) %>%
  arrange(desc(abs(correlation)))
```

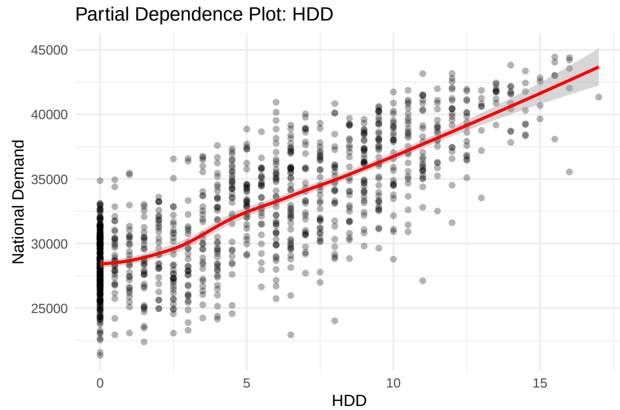
```r
# Plot strong correlations
ggplot(strong_correlations,
       aes(x = reorder(term, abs(correlation)),
           y = correlation,
           fill = feature_type)) +
  geom_col() +
  coord_flip() +
  labs(title = "Strong Correlations with National Demand",
       x = "Features",
       y = "Correlation",
       fill = "Feature Type") +
  theme_minimal() +
  scale_fill_brewer(palette = "Set2")
```

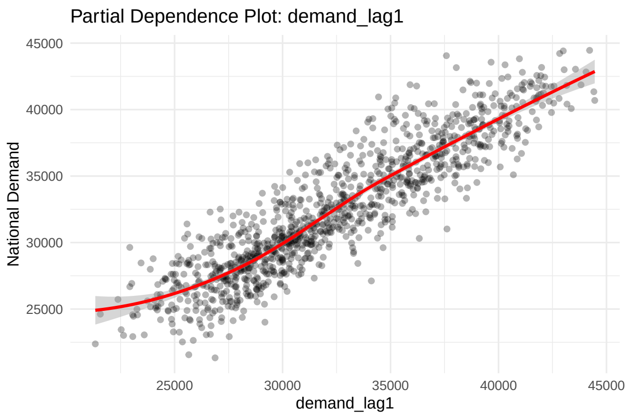


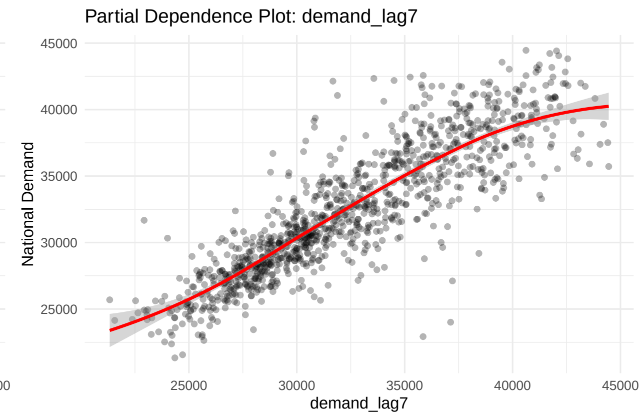
Partial Dependence Plots

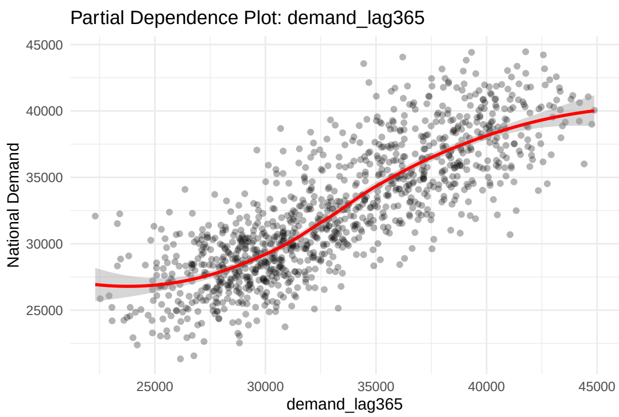
```r
# Select key variables for partial dependence
demand_dependence_vars <- c("national_demand",
                            "annual_cos",
                            "HDD",
                            "demand_lag1",
                            "demand_lag7",
                            "demand_lag365",
                            "renewable_log_diff")

# Prepare data
plot_data <- energy_data %>%
  dplyr::select(all_of(demand_dependence_vars)) %>%
  na.omit()

create_partial_dependence_plot <- function(feature) {
  ggplot(plot_data, aes_string(x = feature, y = "national_demand")) +
    geom_point(alpha = 0.3) +
    geom_smooth(method = "loess", color = "red", se = TRUE) +
    labs(title = paste("Partial Dependence Plot:", feature),
         x = feature,
         y = "National Demand") +
    theme_minimal()
}

partial_plots <- lapply(setdiff(demand_dependence_vars, "national_demand"),
                        create_partial_dependence_plot)
combined_partial_plots <- wrap_plots(partial_plots, ncol = 2)
print(combined_partial_plots)
```

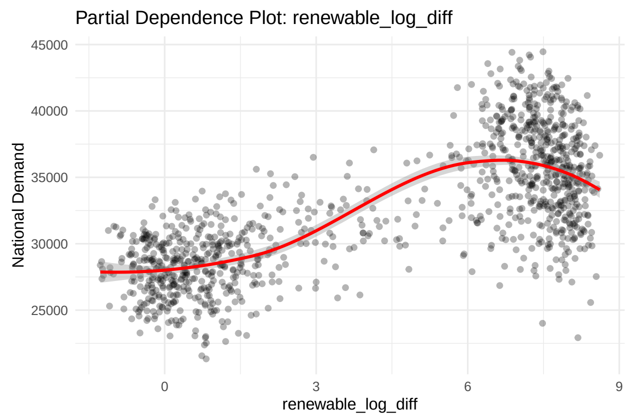












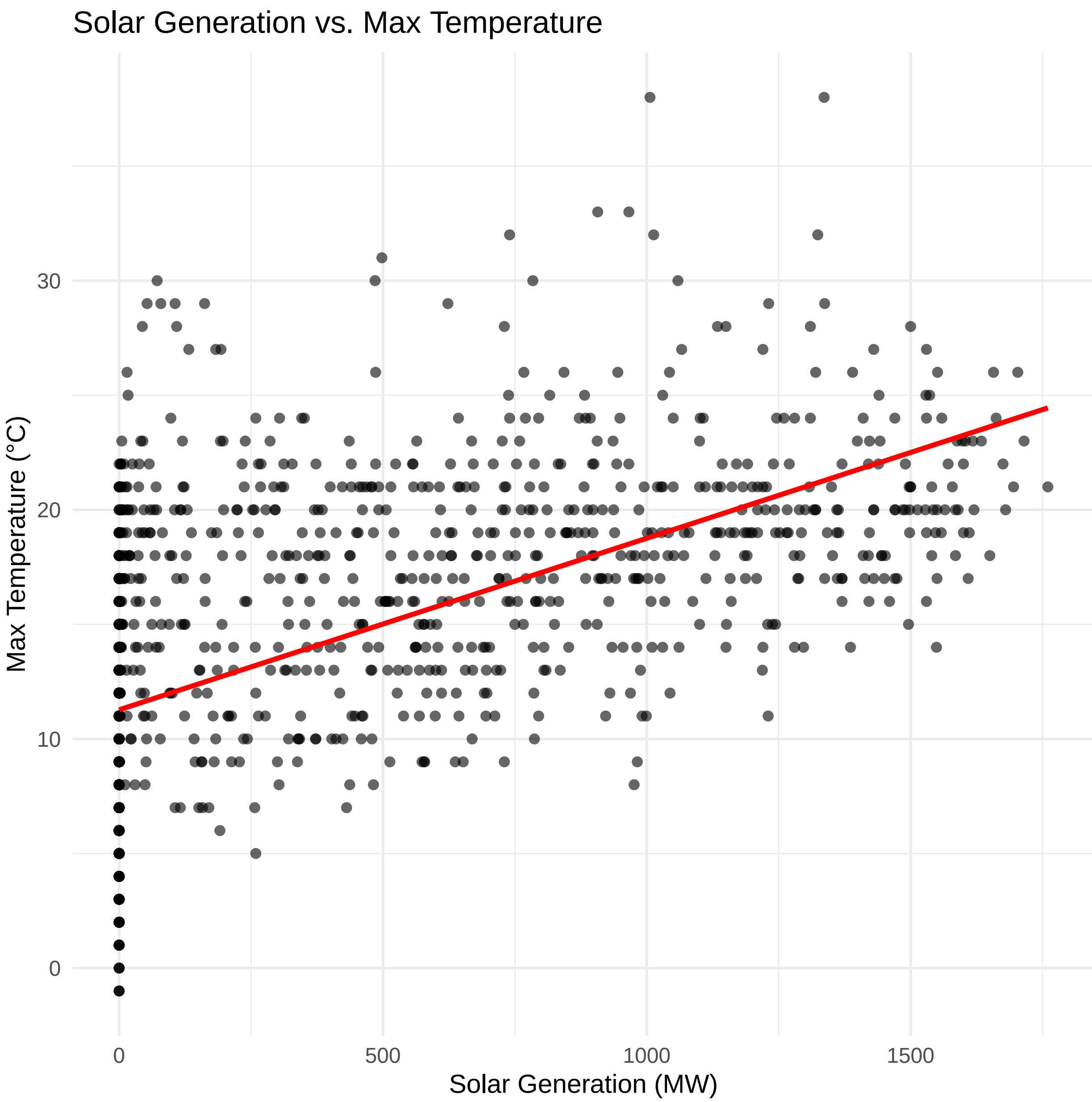
Confounding Effect: Solar Generation vs. Temperature

It can be seen that Solar generation correlates to demand, but is it causality? Because temperature correlates strongly not only with demand, but also with solar_generation, it is a confounder. In fact, demand is seasonal and is not affected by how much solar power is generated.

```r
# Calculate correlation between solar generation and max temperature
solar_temp_corr <- cor(energy_data$solar_generation, energy_data$max_temp,
                       method = "spearman", use = "complete.obs")
cat("Correlation between solar generation and max temperature:", solar_temp_corr, "\n")
```

```
## Correlation between solar generation and max temperature: 0.7104063
```

```r
# Plot Solar Generation vs. Temperature
ggplot(energy_data, aes(x = solar_generation, y = max_temp)) +
  geom_point(color = "black", alpha = 0.6) +
  geom_smooth(method = "lm", color = "red", se = FALSE) +
  labs(
    title = "Solar Generation vs. Max Temperature",
    x = "Solar Generation (MW)",
    y = "Max Temperature (°C)"
  ) +
  theme_minimal()
```



Daily Temperature Variation Effect

```r
# Calculate avg_temp_ma_30 (if not already created)
if(!"avg_temp_ma" %in% names(energy_data)) {
  energy_data <- energy_data %>%
    mutate(avg_temp_ma = rollmean(avg_temp, k = 30, fill = NA, align = "right"))
}

# Calculate daily temperature variation
if(!"dail_variation_avg_temp" %in% names(energy_data)) {
  energy_data <- energy_data %>%
    mutate(dail_variation_avg_temp = avg_temp - avg_temp_ma)
}

# Define temperature thresholds
cold_threshold <- 5
hot_threshold <- 13

# Create temperature segments
energy_data <- energy_data %>%
  mutate(day_type = case_when(
    min_temp < cold_threshold ~ "cold_day",
    min_temp >= cold_threshold & min_temp < hot_threshold ~ "mild_day",
    TRUE ~ "hot_day"
  ))

# Calculate correlations by segment
cold_days <- energy_data %>% filter(min_temp < cold_threshold)
mild_days <- energy_data %>% filter(min_temp >= cold_threshold & min_temp < hot_threshold)
hot_days <- energy_data %>% filter(min_temp >= hot_threshold)

# Calculate correlations of demand vs temperature variation by segment
cat("Correlation in cold days:",
    cor(cold_days$dail_variation_avg_temp, cold_days$national_demand,
        method="spearman", use="complete.obs"), "\n")
```

```
## Correlation in cold days: -0.423068
```

```r
cat("Correlation in mild days:",
    cor(mild_days$dail_variation_avg_temp, mild_days$national_demand,
        method="pearson", use="complete.obs"), "\n")
```

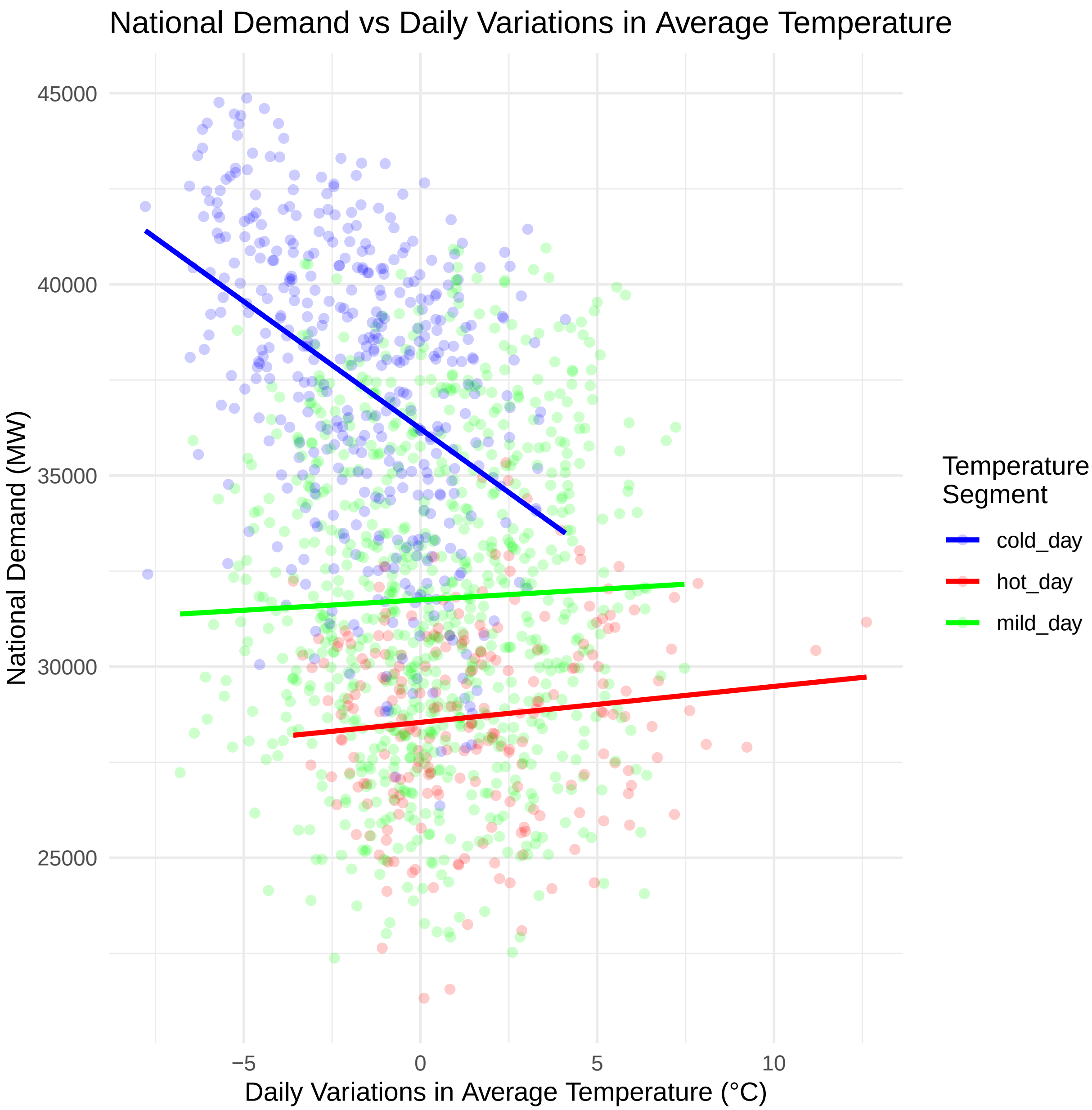
```
## Correlation in mild days: 0.03631178
```

```r
cat("Correlation in hot days:",
    cor(hot_days$dail_variation_avg_temp, hot_days$national_demand,
        method="spearman", use="complete.obs"), "\n")
```

```
## Correlation in hot days: 0.08510393
```

```r
# Plot daily variations correlation to demand by temperature segment
ggplot(energy_data, aes(x = dail_variation_avg_temp, y = national_demand, color = day_type)) +
  geom_point(alpha = 0.2) +
  geom_smooth(method = "lm", se = FALSE) +
  labs(
    title = "National Demand vs Daily Variations in Average Temperature",
```

```r
    x = "Daily Variations in Average Temperature (°C)",
    y = "National Demand (MW)"
  ) +
  scale_color_manual(values = c(
    "cold_day" = "blue", "mild_day" = "green", "hot_day" = "red"),
    name = "Temperature\nSegment") +
  theme_minimal()
```



During cold periods, daily temperature drops cause demand to increase. During mild and hot days, this behavior is not seen.

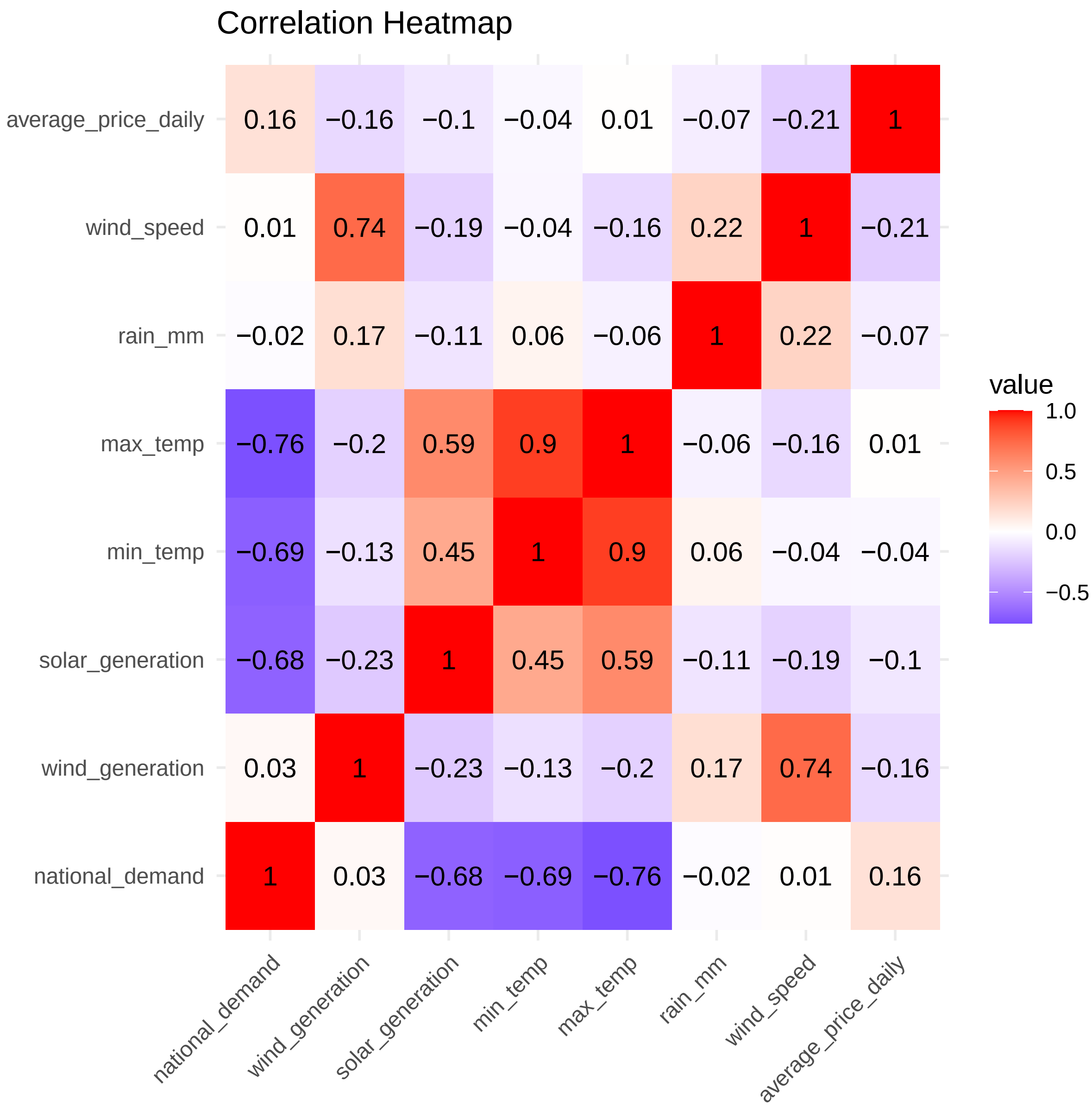
PART 4 DATA VISUALISATION

#Pair Plot

```r
temp_file <- tempfile(fileext = ".png")
png(temp_file)
ggpairs(data = energy_data,
        columns = c(3:10),na.rm = TRUE)
```

HEATMAP

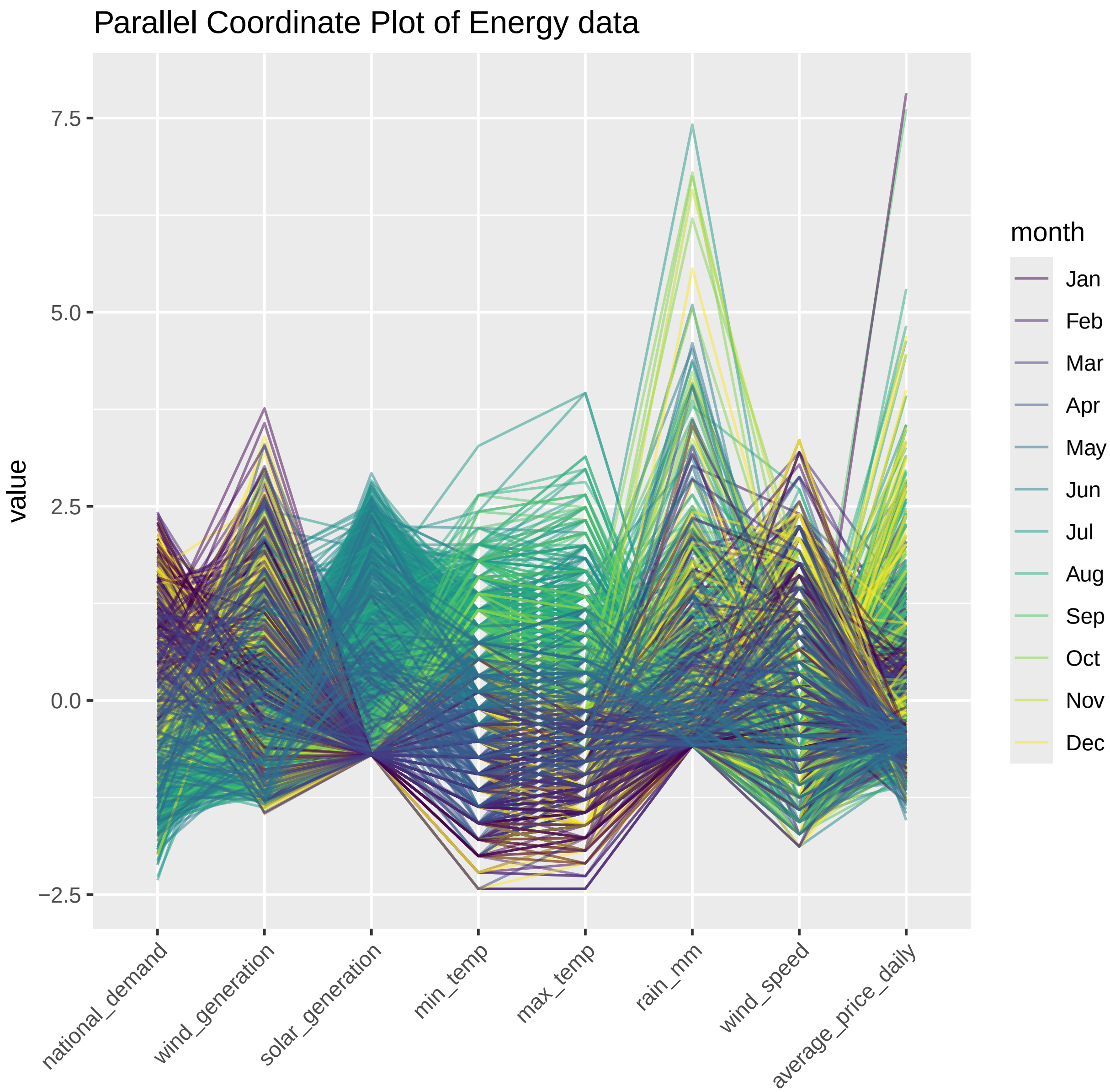
```r
corrmat <- cor(energy_data[3:10],use = "complete.obs")
df_cormat <- melt(corrmat,na.rm = TRUE)

corr_heatmap <- ggplot(df_cormat, aes(x = Var1, y = Var2, fill = value)) +
  geom_tile() +
  scale_fill_gradient2(low = "blue", mid = "white", high = "red", midpoint = 0) +
  geom_text(aes(label = round(value, 2)), color = "black", size = 4) +
  theme_minimal() +
  labs(title = "Correlation Heatmap", x = "", y = "") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

corr_heatmap
```



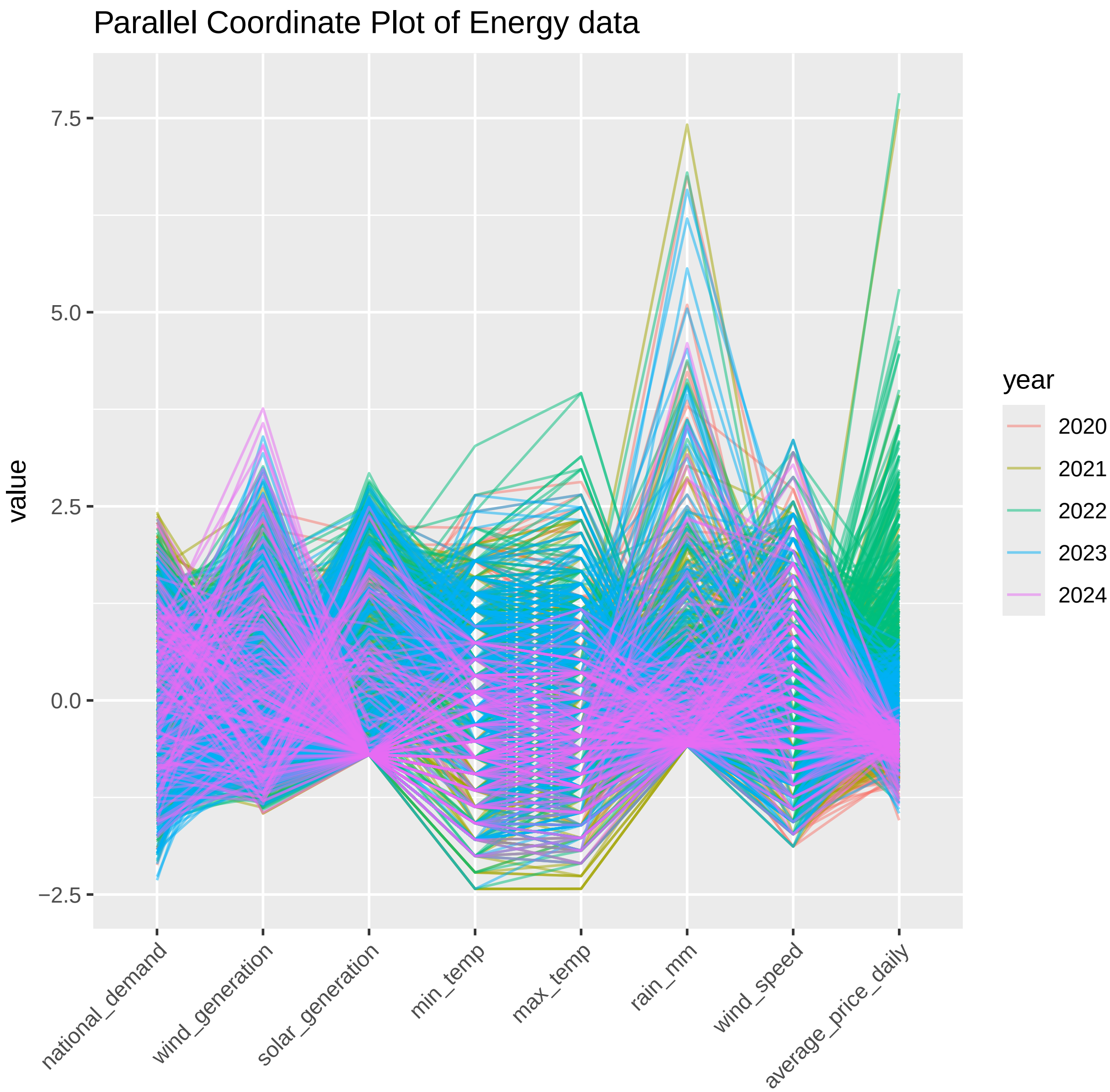
##
##
#Added to the report 3

##parallel coordinate plot

```
ggparcoord(data = energy_data,
           columns = c(3:10),  # Select numeric columns to plot
           scale = "std",  # Standardizes variables
           groupColumn = "month",
           alphaLines = 0.5) +  # Adjust transparency
  labs(title = "Parallel Coordinate Plot of Energy data", x="")+
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

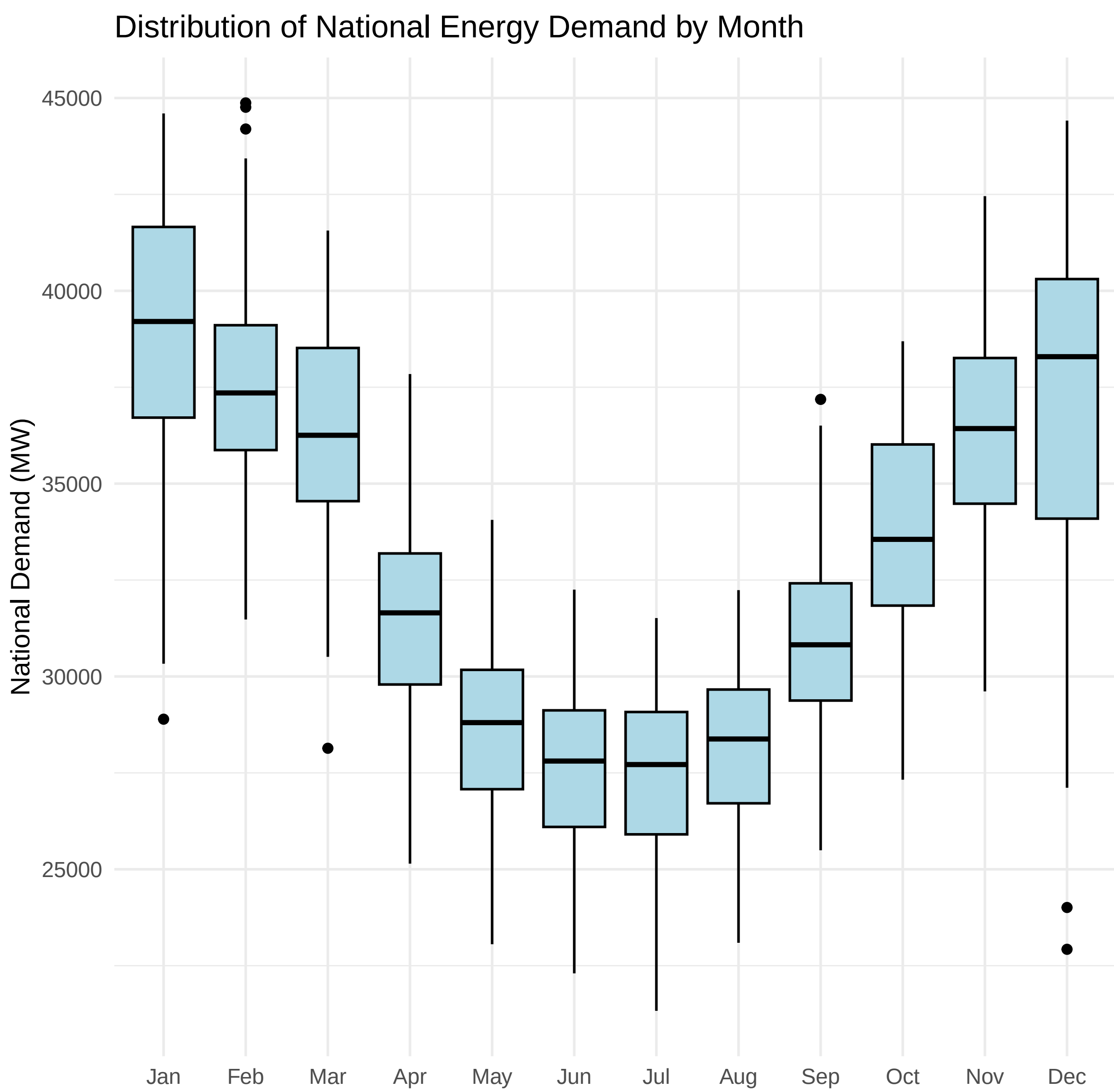



```
ggparcoord(data = energy_data,
           columns = c(3:10),  # Select numeric columns to plot
           scale = "std",  # Standardizes variables
           groupColumn = "year",
           alphaLines = 0.5) +  # Adjust transparency
  labs(title = "Parallel Coordinate Plot of Energy data", x="")+
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

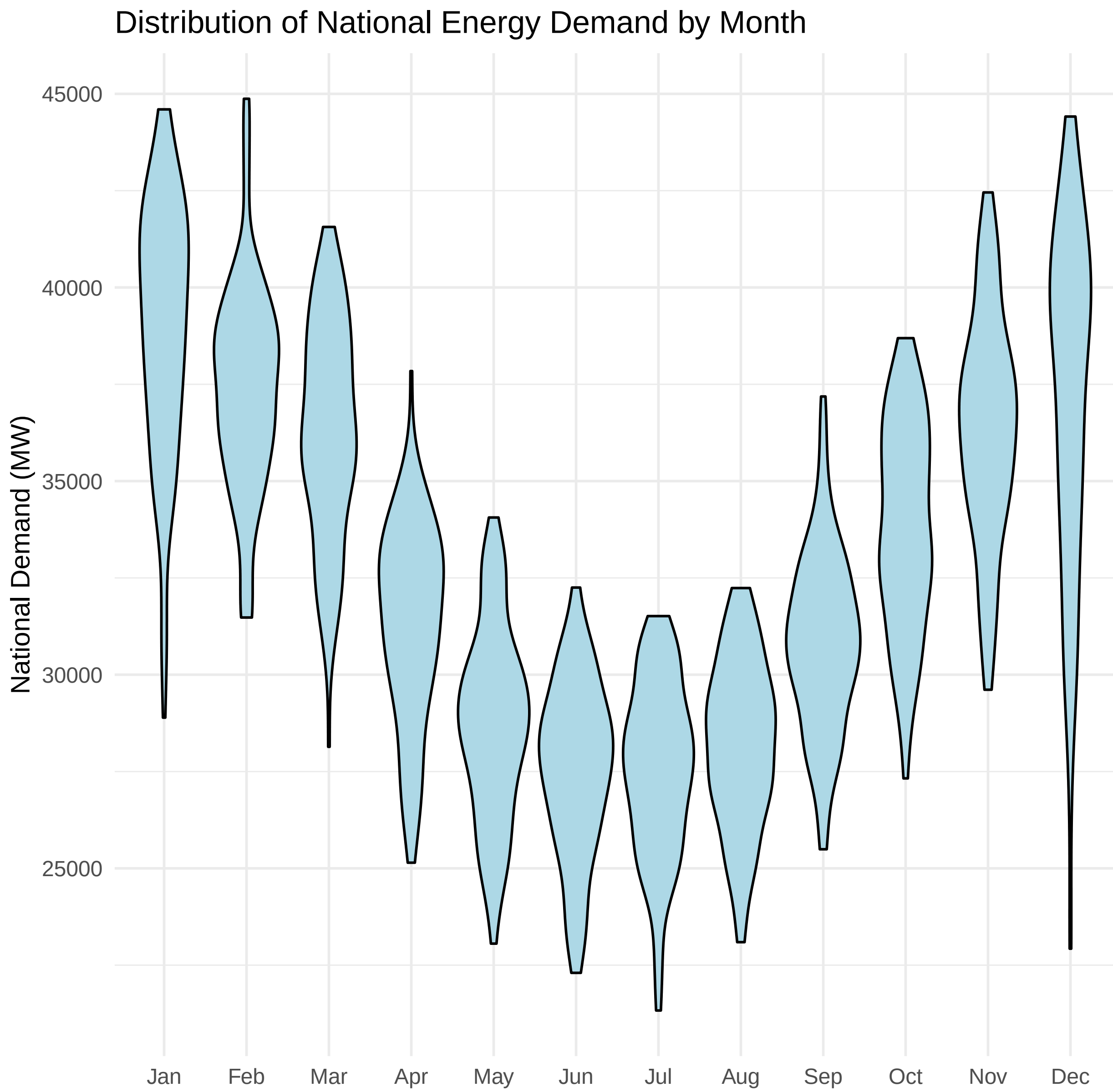


Boxplot/Violin plot Monthly split - seasonality

```r
energy_data$month <- factor(month(energy_data$date, label = TRUE), levels = month.abb)
ggplot(energy_data, aes(x = month, y = national_demand)) +
  geom_boxplot(fill = "lightblue", color = "black") +
  labs(title = "Distribution of National Energy Demand by Month",
       x = "", y = "National Demand (MW)") +
  theme_minimal()
```




```r
energy_data$month <- factor(month(energy_data$date, label = TRUE), levels = month.abb)
ggplot(energy_data, aes(x = month, y = national_demand)) +
  geom_violin(fill = "lightblue", color = "black") +
  labs(title = "Distribution of National Energy Demand by Month",
       x = "", y = "National Demand (MW)") +
  theme_minimal()
```


Distribution of National Energy Demand by Month
National Demand (MW)
45000
40000
35000
30000
25000
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec