ADS506 Assignment 2.2 PJME_MW Time Series Data

Gabriel E. Mancillas Gallardo

2024-11-17

Assignment: Propose a Time Series Dataset for Your Final Project ### Data Source

The data source is the PJM Hourly Energy Consumption Data. The dataset contains the hourly power consumption data from PJM from 2005 to 2018. The dataset contains the following columns:

```r
# Load necessary libraries
library(readr)
library(dplyr)
library(tsibble)
library(ggplot2)
library(feasts)
library(fable)
library(forecast)

# Load AEP consumption data
aep_data <- read_csv("/Users/home/Documents/GitHub/Energy-Consumption-Data/AEP_hourly.csv")

# Display the first few rows of the dataset
head(aep_data)
```

```
## # A tibble: 6 x 2
##   Datetime            AEP_MW
##   <dttm>               <dbl>
## 1 2004-12-31 01:00:00  13478
## 2 2004-12-31 02:00:00  12865
## 3 2004-12-31 03:00:00  12577
## 4 2004-12-31 04:00:00  12517
## 5 2004-12-31 05:00:00  12670
## 6 2004-12-31 06:00:00  13038
```

```r
# Display the total number of rows and columns
dim(aep_data)
```

```
## [1] 121273      2
```

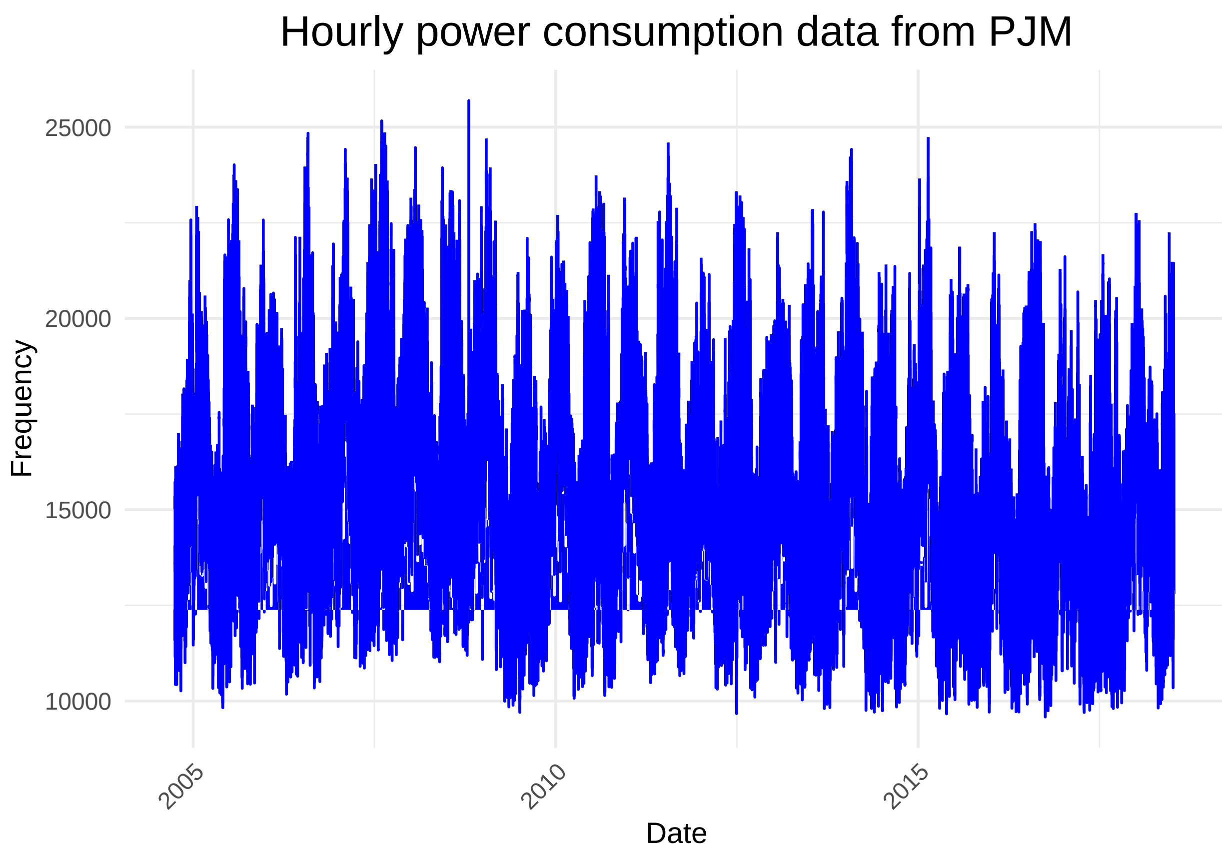
Data Source

Include public links to data if it is too large to upload (do not upload datasets larger than 50MB). **There is no need to upload the data (its under 40MB)** github repository

Time Series Plot

```r
# Check for duplicates
duplicates <- aep_data %>%
    group_by(Datetime) %>%
    filter(n() > 1)

# Remove duplicates if any
aep_data <- aep_data %>%
    distinct(Datetime, .keep_all = TRUE)
```

```r
# Convert to tsibble for time series structure
aep_data_ts <- aep_data %>%
    as_tsibble(index = Datetime)

# Time series plot for Adjusted Close prices
ggplot(aep_data_ts, aes(x = Datetime, y = `AEP_MW`)) +
    geom_line(color = "blue") +
    labs(
        title = "Hourly power consumption data from PJM",
        x = "Date",
        y = "Frequency"
    ) +
    theme_minimal() +
    theme(
        plot.title = element_text(hjust = 0.5, size = 16),
        axis.text.x = element_text(angle = 45, hjust = 1)
    )
```



Discussion

The hourly power consumption data from PJM reveals notable fluctuations over time, capturing trends that reflect varying levels of energy demand influenced by seasonality, time of day, and economic factors. From 2005 onwards, we observe a periodic increase in energy demand during summer and winter months, likely driven by seasonal heating and cooling needs. These patterns highlight the strong influence of temperature and weather conditions on energy usage.

Energy demand also reflects broader economic activities. Periods of economic growth are often marked by increased industrial and residential energy consumption, whereas economic downturns can lead to reduced demand. This dataset offers insights into how external factors, such as economic slowdowns, regulatory changes, and shifts in energy efficiency practices, can directly impact consumption patterns.

Predicting future consumption trends from this dataset can provide valuable insights for energy providers, policymakers, and businesses. Accurate forecasts help ensure resource availability, support grid stability, and inform pricing strategies. However, energy demand is inherently variable, affected by unpredictable factors like extreme weather events and changes in consumer behavior. Adding contextual data, such as temperature records or economic indicators, would likely improve forecast accuracy and provide a more comprehensive understanding of the factors driving energy consumption.

Assignment: Propose a Time Series Dataset for Your Final Project

```
##     Datetime                                 AEP_MW
##  Min.   :2004-10-01 01:00:00.00   Min.   : 9581
##  1st Qu.:2008-03-17 14:00:00.00   1st Qu.:13630
##  Median :2011-09-02 02:00:00.00   Median :15310
##  Mean   :2011-09-02 01:55:58.41   Mean   :15500
##  3rd Qu.:2015-02-16 15:00:00.00   3rd Qu.:17200
##  Max.   :2018-08-03 00:00:00.00   Max.   :25695
```

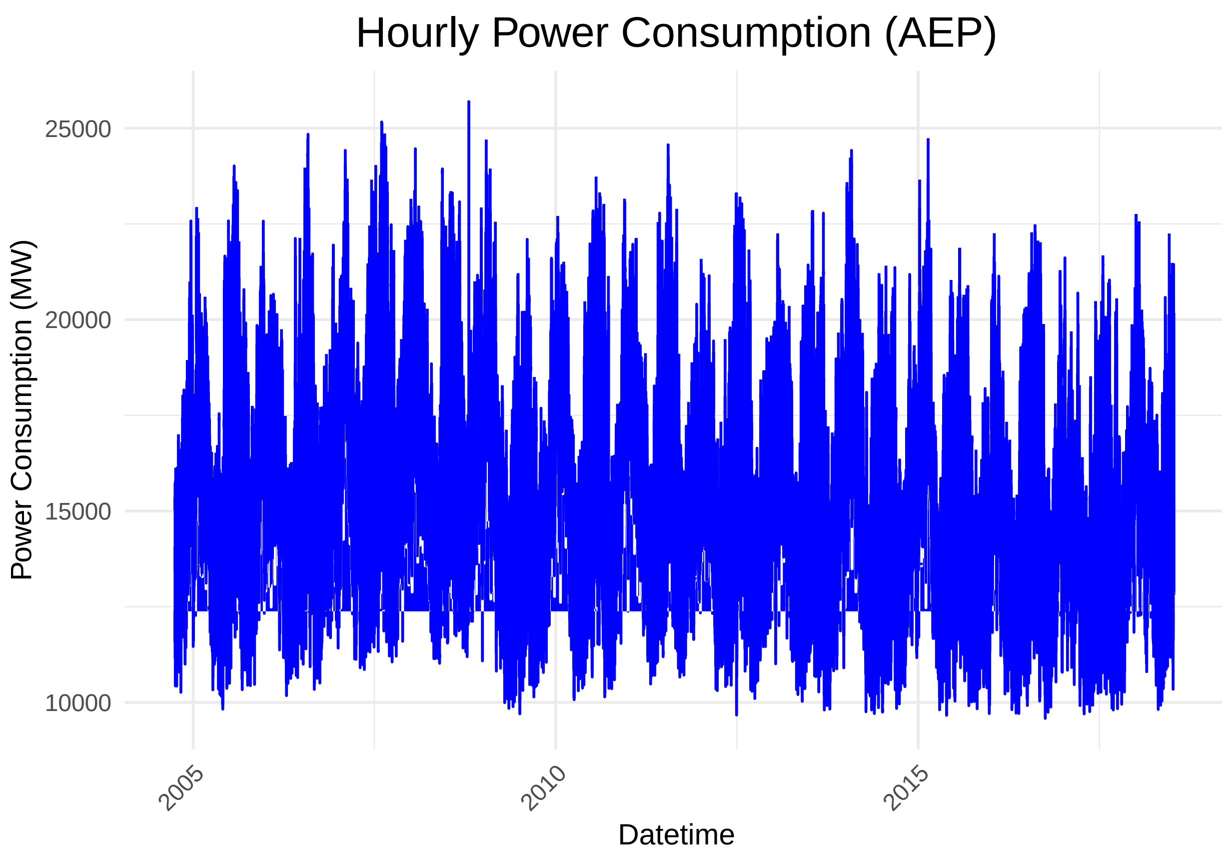
```
## # A tibble: 6 x 2
##   Datetime            AEP_MW
##   <dttm>               <dbl>
## 1 2004-12-31 01:00:00  13478
## 2 2004-12-31 02:00:00  12865
## 3 2004-12-31 03:00:00  12577
## 4 2004-12-31 04:00:00  12517
## 5 2004-12-31 05:00:00  12670
## 6 2004-12-31 06:00:00  13038
```

```
## Number of missing values: 0
```

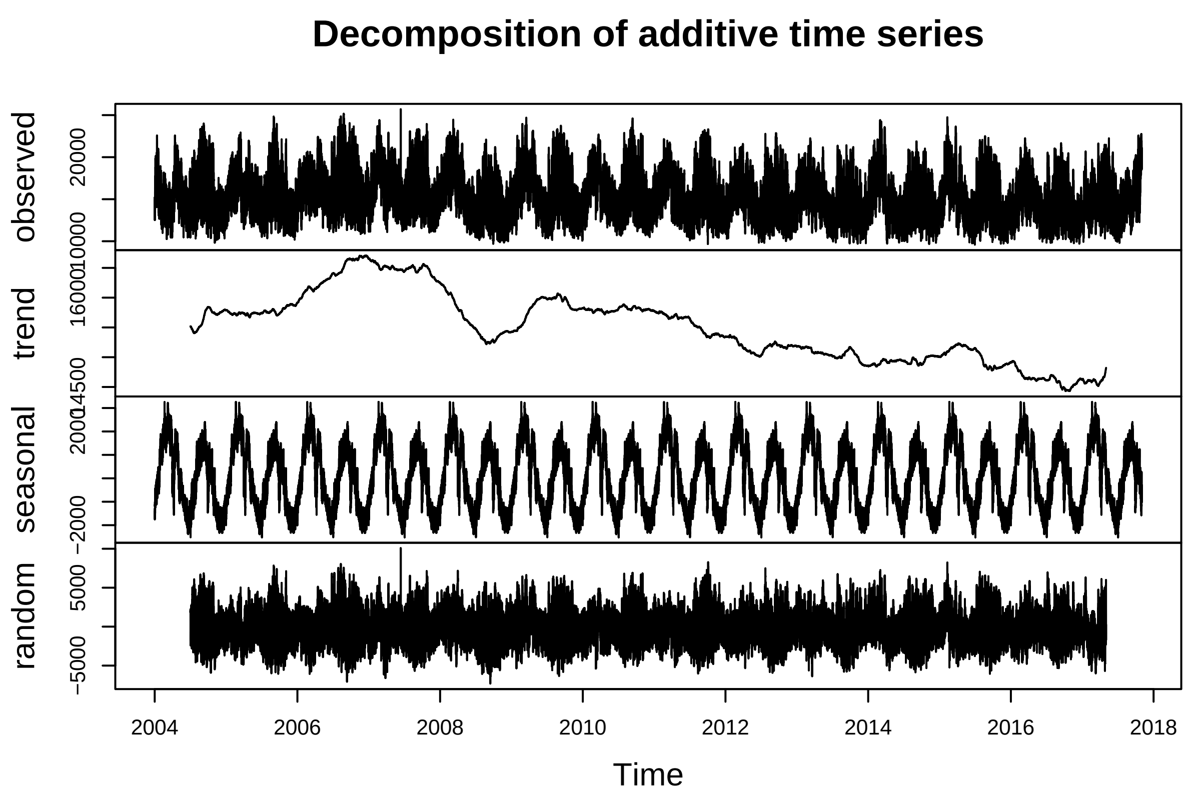
```
## Number of missing timestamps: 27
```

```
## [1] 1099188000 1112497200 1130637600 1143946800 1162087200 1173582000
```

```
## Number of missing values after interpolation: 0
```



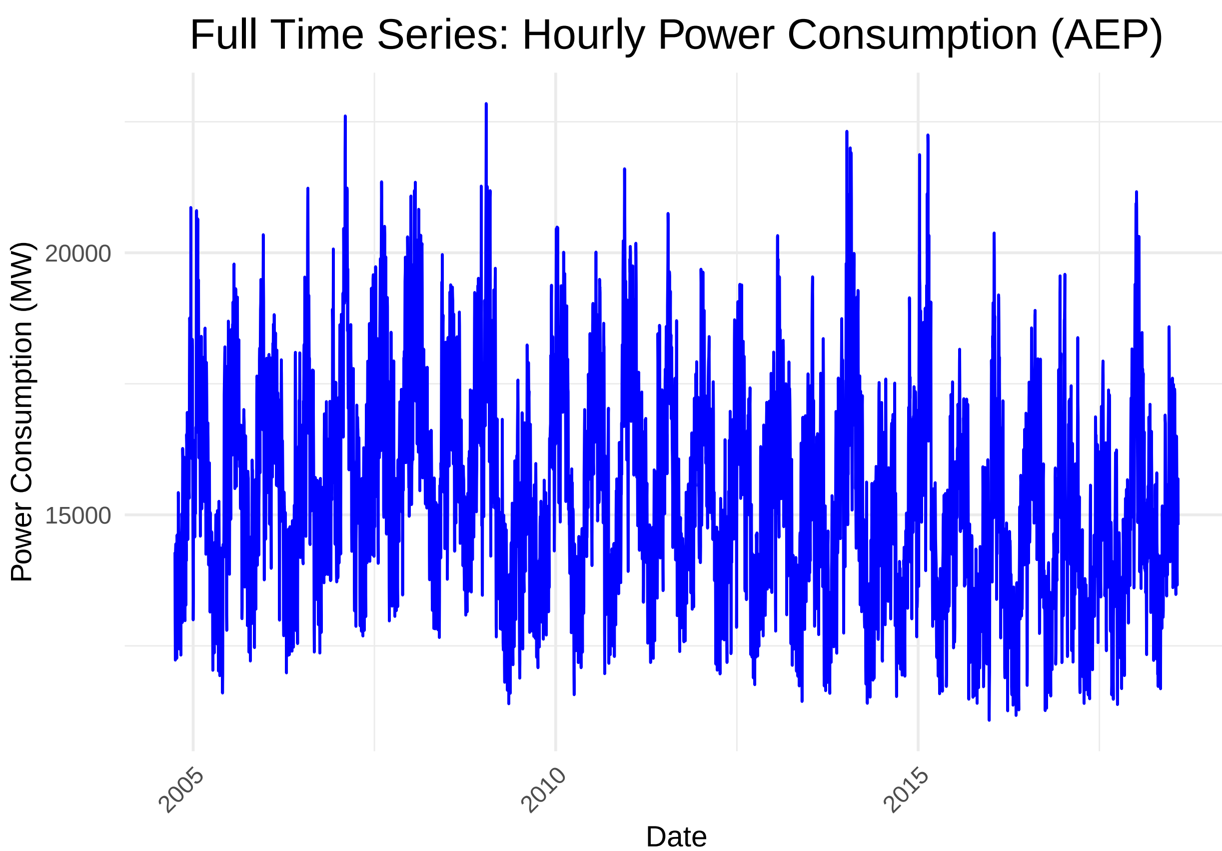
```
## Number of missing values in AEP_MW: 0

## Number of missing timestamps: 27

## # A tsibble: 121,269 x 2 [1h] <UTC>
##    Datetime            AEP_MW
##    <dttm>               <dbl>
##  1 2004-10-01 01:00:00  12379
##  2 2004-10-01 02:00:00  11935
##  3 2004-10-01 03:00:00  11692
##  4 2004-10-01 04:00:00  11597
##  5 2004-10-01 05:00:00  11681
##  6 2004-10-01 06:00:00  12280
##  7 2004-10-01 07:00:00  13692
##  8 2004-10-01 08:00:00  14618
##  9 2004-10-01 09:00:00  14903
## 10 2004-10-01 10:00:00  15118
## # i 121,259 more rows

## [1] "numeric"

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    9581   13630   15310   15500   17200   25695
```

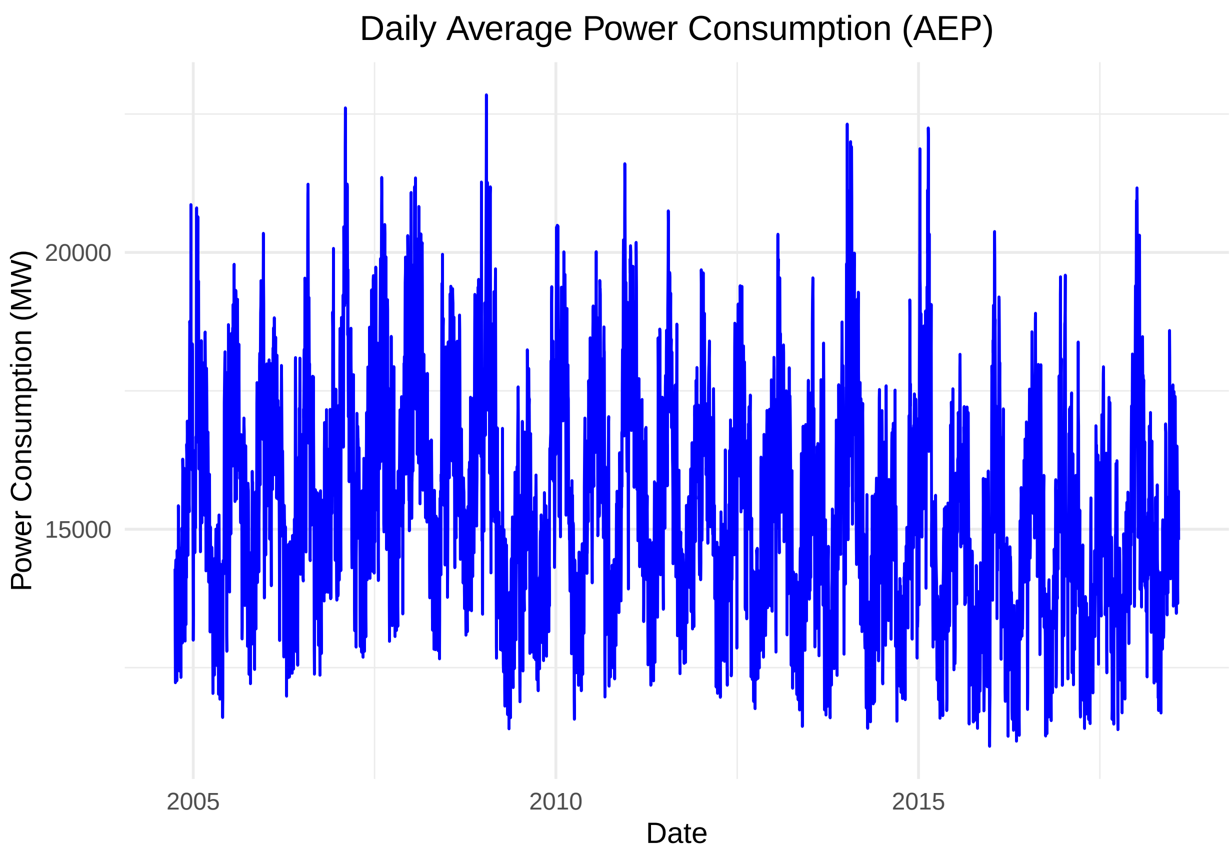


Lets proceed with the ARIMA of the time series data

```
## # A tibble: 6 x 2
##   Datetime            AEP_MW
##   <dttm>               <dbl>
## 1 2004-12-31 01:00:00  13478
## 2 2004-12-31 02:00:00  12865
## 3 2004-12-31 03:00:00  12577
## 4 2004-12-31 04:00:00  12517
## 5 2004-12-31 05:00:00  12670
## 6 2004-12-31 06:00:00  13038
```

```
## [1] 121273      2
```





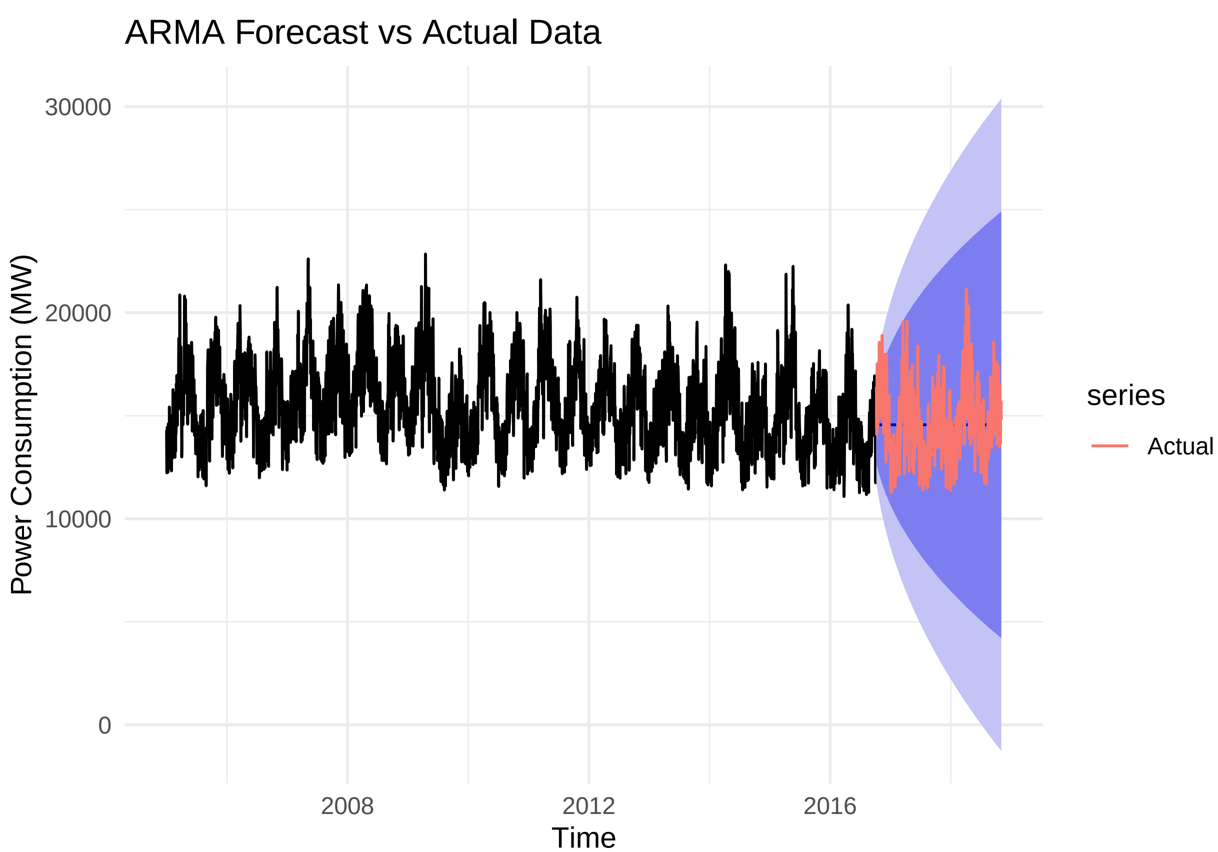

```r
# Convert to time series object
aep_ts <- ts(aep_data_daily$AEP_MW, start = c(2005, 1), frequency = 365.25)

# Split data into training (85%) and testing (15%) sets
split_point <- floor(0.85 * length(aep_ts))
train_data <- window(aep_ts, end = c(2005 + (split_point / 365.25)))
test_data <- window(aep_ts, start = c(2005 + (split_point / 365.25) + 1 / 365.25))

# Fit ARMA model
arma_model <- auto.arima(train_data, seasonal = FALSE)

# Display ARMA model summary
summary(arma_model)
```

```
## Series: train_data
## ARIMA(2,1,4)
##
## Coefficients:
##           ar1      ar2     ma1      ma2      ma3      ma4
##       -0.9343  -0.4117  1.0558  -0.0076  -0.8025  -0.5102
## s.e.   0.0442   0.0299  0.0406   0.0270   0.0241   0.0149
##
## sigma^2 = 849506:  log likelihood = -35418.76
## AIC=70851.52   AICc=70851.55   BIC=70896.08
##
## Training set error measures:
```

```
##                     ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 0.4519411 920.9354 726.4187 -0.2880247 4.667184 0.4924075
##                     ACF1
## Training set 0.002969507
```

```r
# Forecast future values
arma_forecast <- forecast(arma_model, h = length(test_data))

# Plot the forecast
autoplot(arma_forecast) +
  autolayer(test_data, series = "Actual", PI = FALSE) +
  labs(
    title = "ARMA Forecast vs Actual Data",
    x = "Time",
    y = "Power Consumption (MW)"
  ) +
  theme_minimal()
```

```
## Warning in ggplot2::geom_line(ggplot2::aes(x = .data[["timeVal"]], y =
## .data[["seriesVal"]], : Ignoring unknown parameters: ‘PI‘
```



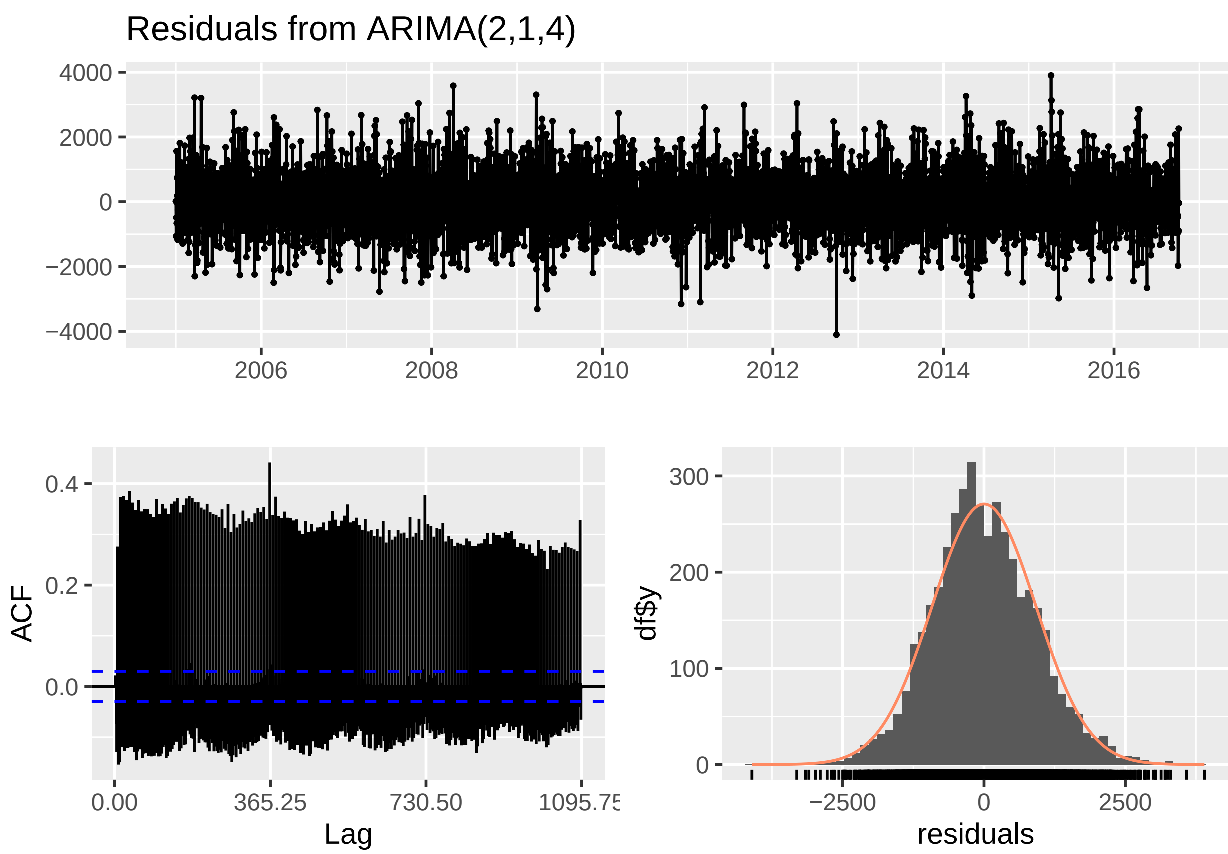

```r
# Calculate RMSE and MAE
arma_rmse <- sqrt(mean((test_data - arma_forecast$mean)^2, na.rm = TRUE))
arma_mae <- mean(abs(test_data - arma_forecast$mean), na.rm = TRUE)

# Print model evaluation metrics
cat("ARMA - RMSE:", arma_rmse, "\n")
```

```
## ARMA - RMSE: 1828.204
```

```r
cat("ARMA - MAE:", arma_mae, "\n")
```

```
## ARMA - MAE: 1453.795
```

```r
# Check residuals
checkresiduals(arma_model)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,1,4)
## Q* = 70508, df = 724.5, p-value < 2.2e-16
##
## Model df: 6.   Total lags used: 730.5
```

```r
# Random Walk Model
rw_forecast <- rwf(train_data, h = length(test_data), drift = FALSE)

# Calculate Random Walk metrics
rw_rmse <- sqrt(mean((test_data - rw_forecast$mean)^2, na.rm = TRUE))
rw_mae <- mean(abs(test_data - rw_forecast$mean), na.rm = TRUE)

# Print results
cat("Random Walk - RMSE:", rw_rmse, "\n")
```

```
## Random Walk - RMSE: 2291.718
```

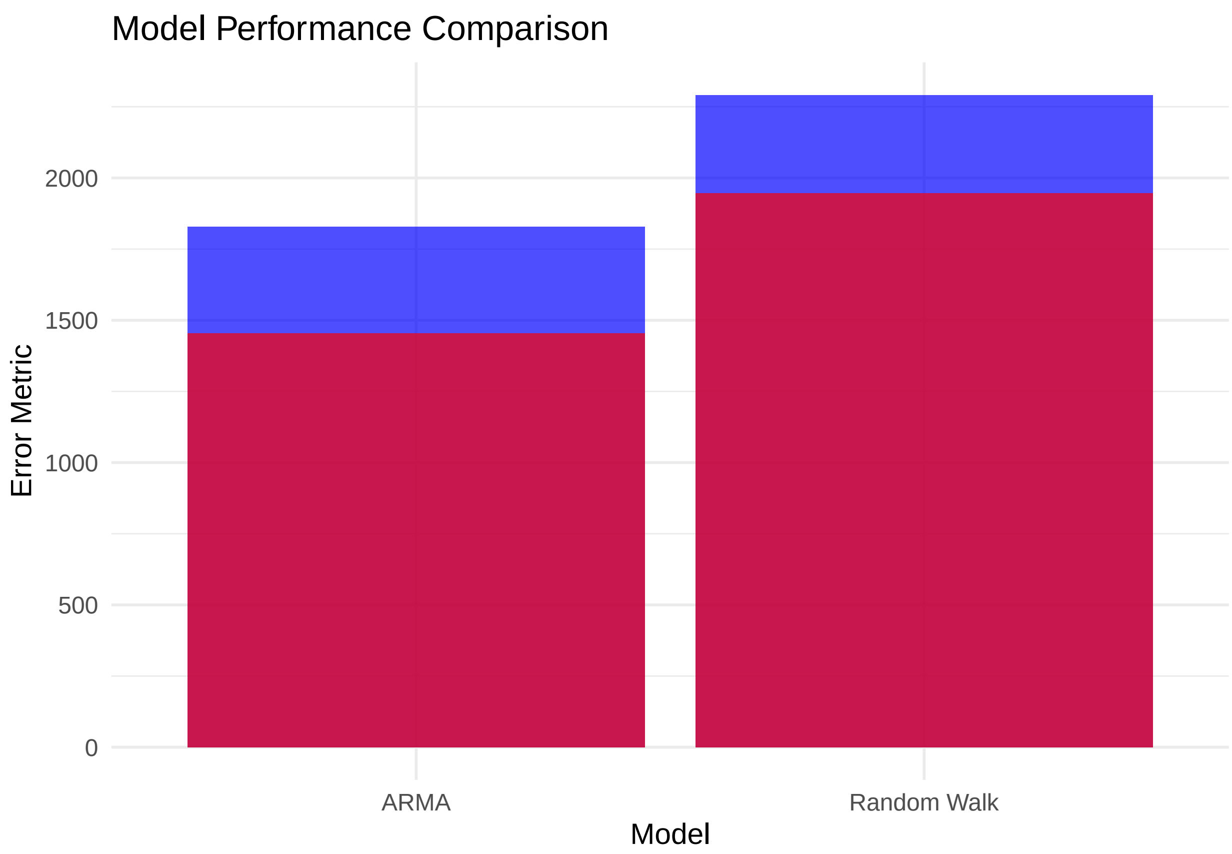
```r
cat("Random Walk - MAE:", rw_mae, "\n")
```

```
## Random Walk - MAE: 1947.201
```

```r
# Model Performance Comparison
model_comparison <- data.frame(
  Model = c("ARMA", "Random Walk"),
  RMSE = c(arma_rmse, rw_rmse),
  MAE = c(arma_mae, rw_mae)
)

# Display comparison
print(model_comparison)
```

```
##         Model     RMSE      MAE
## 1        ARMA 1828.204 1453.795
## 2 Random Walk 2291.718 1947.201
```

```r
# Plot comparison
ggplot(model_comparison, aes(x = Model)) +
  geom_bar(aes(y = RMSE), stat = "identity", fill = "blue", alpha = 0.7) +
  geom_bar(aes(y = MAE), stat = "identity", fill = "red", alpha = 0.7, position = "dodge") +
  labs(
    title = "Model Performance Comparison",
    x = "Model",
    y = "Error Metric"
  ) +
  theme_minimal()
```


Model Performance Comparison
2000
1500
1000
500
0
Error Metric
ARMA
Random Walk
Model

```r
# Load necessary libraries
library(readr)
library(dplyr)
library(tsibble)
library(ggplot2)
library(forecast)

# Load AEP consumption data
aep_data <- read_csv("/Users/home/Documents/GitHub/Energy-Consumption-Data/AEP_hourly.csv")
```

```
## Rows: 121273 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl  (1): AEP_MW
## dttm (1): Datetime
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# Display the first few rows of the dataset
head(aep_data)
```

```
## # A tibble: 6 x 2
##   Datetime            AEP_MW
##   <dttm>               <dbl>
## 1 2004-12-31 01:00:00  13478
## 2 2004-12-31 02:00:00  12865
## 3 2004-12-31 03:00:00  12577
## 4 2004-12-31 04:00:00  12517
## 5 2004-12-31 05:00:00  12670
## 6 2004-12-31 06:00:00  13038
```

```r
# Display the total number of rows and columns
dim(aep_data)
```

```
## [1] 121273      2
```

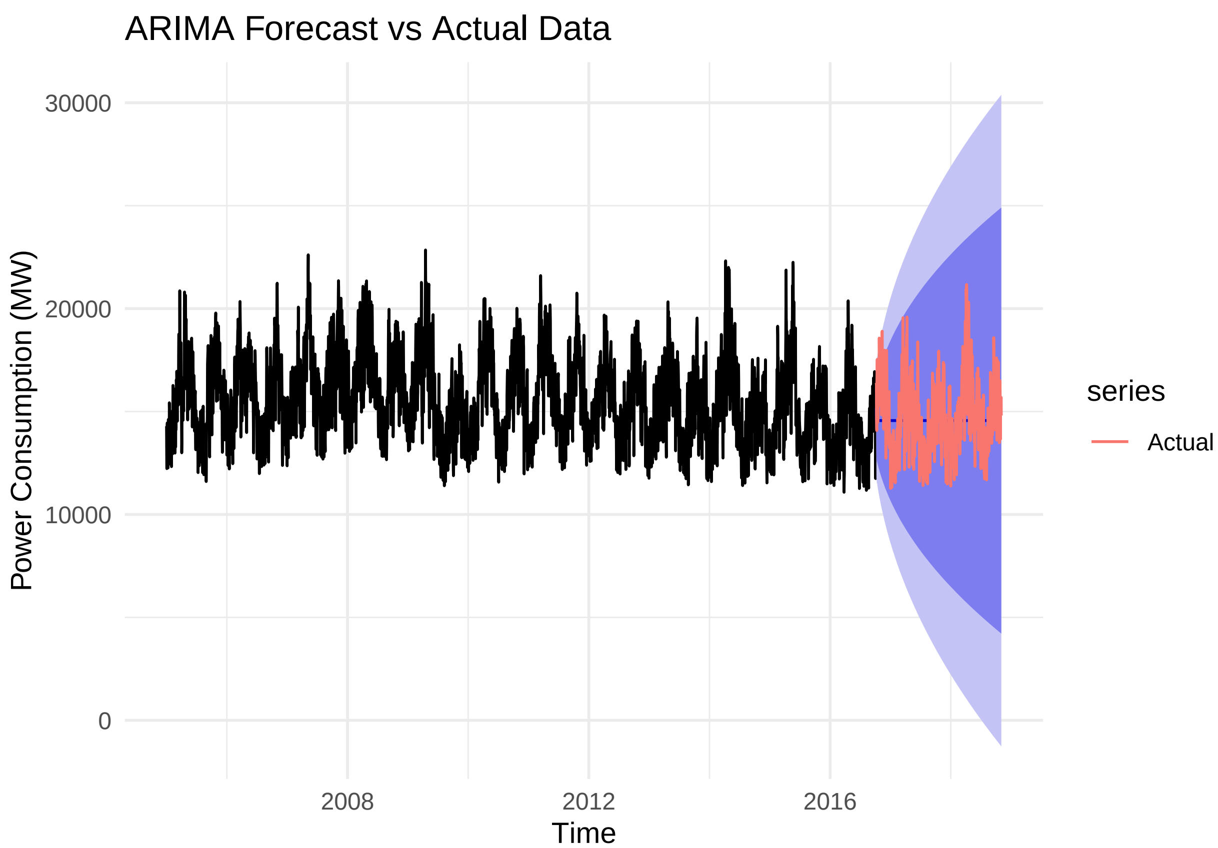
```r
# Remove duplicates
aep_data <- aep_data %>%
  distinct(Datetime, .keep_all = TRUE)

# Handle missing values
aep_data$AEP_MW <- zoo::na.approx(aep_data$AEP_MW, na.rm = FALSE)

# Convert to daily averages
aep_data_daily <- aep_data %>%
  mutate(Date = as.Date(Datetime)) %>%
  group_by(Date) %>%
  summarise(AEP_MW = mean(AEP_MW, na.rm = TRUE))

# Convert to a time series object
aep_ts <- ts(aep_data_daily$AEP_MW, start = c(2005, 1), frequency = 365.25)
```

```r
# Split data into training (85%) and testing (15%) sets
split_point <- floor(0.85 * length(aep_ts))
train_data <- window(aep_ts, end = c(2005 + (split_point / 365.25)))
test_data <- window(aep_ts, start = c(2005 + (split_point / 365.25) + 1 / 365.25))

# Fit ARIMA model using auto.arima
arima_model <- auto.arima(train_data, seasonal = FALSE)

# Display ARIMA model summary
summary(arima_model)
```

```
## Series: train_data
## ARIMA(2,1,4)
##
## Coefficients:
##           ar1      ar2     ma1      ma2      ma3      ma4
##       -0.9343  -0.4117  1.0558  -0.0076  -0.8025  -0.5102
## s.e.   0.0442   0.0299  0.0406   0.0270   0.0241   0.0149
##
## sigma^2 = 849506:  log likelihood = -35418.76
## AIC=70851.52   AICc=70851.55   BIC=70896.08
##
## Training set error measures:
##                     ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 0.4519411 920.9354 726.4187 -0.2880247 4.667184 0.4924075
##                     ACF1
## Training set 0.002969507
```

```r
# Forecast future values
arima_forecast <- forecast(arima_model, h = length(test_data))

# Plot the forecast
autoplot(arima_forecast) +
  autolayer(test_data, series = "Actual", PI = FALSE) +
  labs(
    title = "ARIMA Forecast vs Actual Data",
    x = "Time",
    y = "Power Consumption (MW)"
  ) +
  theme_minimal()
```

```
## Warning in ggplot2::geom_line(ggplot2::aes(x = .data[["timeVal"]], y =
## .data[["seriesVal"]], : Ignoring unknown parameters: ‘PI‘
```



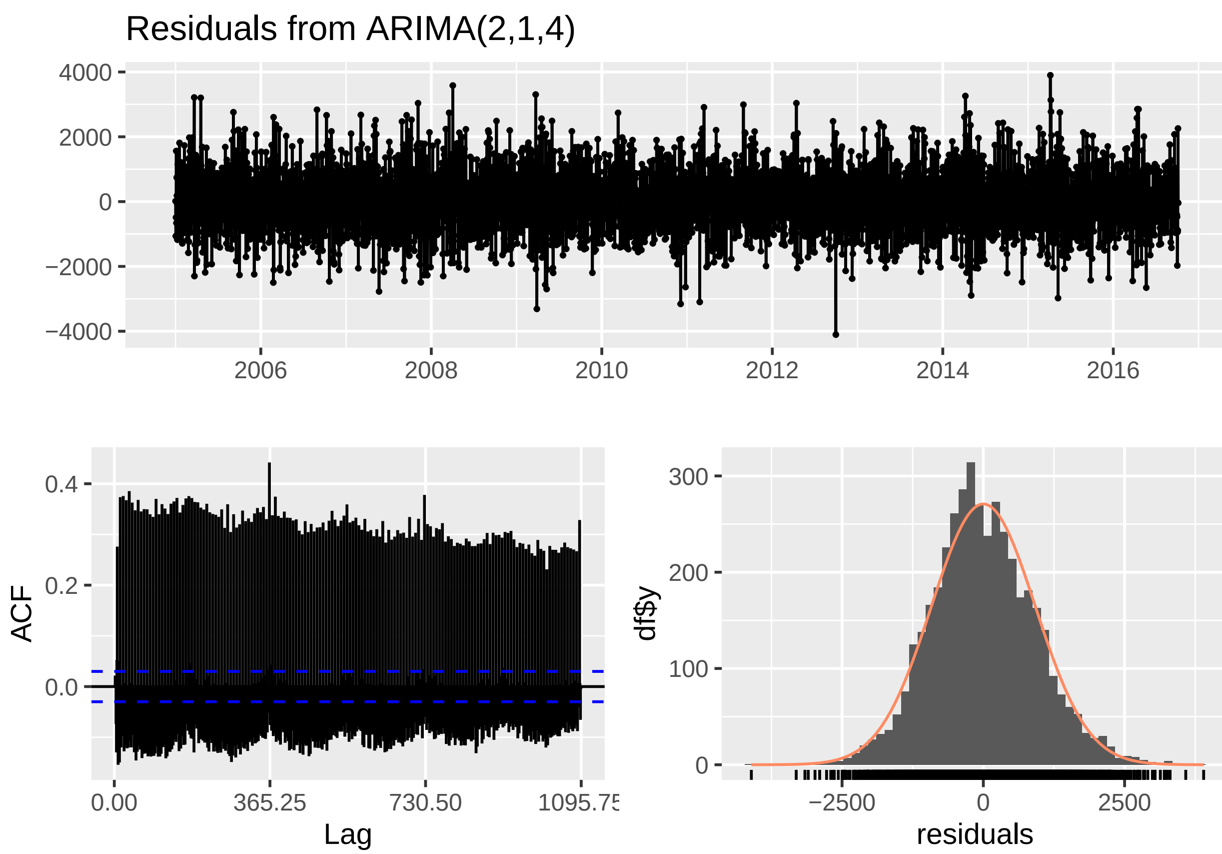

```r
# Calculate RMSE and MAE for ARIMA
arima_rmse <- sqrt(mean((test_data - arima_forecast$mean)^2, na.rm = TRUE))
arima_mae <- mean(abs(test_data - arima_forecast$mean), na.rm = TRUE)

# Print evaluation metrics
cat("ARIMA - RMSE:", arima_rmse, "\n")
```

```
## ARIMA - RMSE: 1828.204
```

```r
cat("ARIMA - MAE:", arima_mae, "\n")
```

```
## ARIMA - MAE: 1453.795
```

```r
# Check residuals
checkresiduals(arima_model)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,1,4)
## Q* = 70508, df = 724.5, p-value < 2.2e-16
##
## Model df: 6.   Total lags used: 730.5
```

```r
# Random Walk Model
rw_forecast <- rwf(train_data, h = length(test_data), drift = FALSE)

# Calculate Random Walk metrics
rw_rmse <- sqrt(mean((test_data - rw_forecast$mean)^2, na.rm = TRUE))
rw_mae <- mean(abs(test_data - rw_forecast$mean), na.rm = TRUE)

# Print results
cat("Random Walk - RMSE:", rw_rmse, "\n")
```

```
## Random Walk - RMSE: 2291.718
```

```r
cat("Random Walk - MAE:", rw_mae, "\n")
```

```
## Random Walk - MAE: 1947.201
```

```r
# Model Performance Comparison
model_comparison <- data.frame(
  Model = c("ARIMA", "Random Walk"),
  RMSE = c(arima_rmse, rw_rmse),
  MAE = c(arima_mae, rw_mae)
)

# Display comparison
print(model_comparison)
```

```
##         Model     RMSE      MAE
## 1       ARIMA 1828.204 1453.795
## 2 Random Walk 2291.718 1947.201
```

```r
# Plot comparison
ggplot(model_comparison, aes(x = Model)) +
  geom_bar(aes(y = RMSE), stat = "identity", fill = "blue", alpha = 0.7) +
  geom_bar(aes(y = MAE), stat = "identity", fill = "red", alpha = 0.7, position = "dodge") +
  labs(
    title = "Model Performance Comparison",
    x = "Model",
    y = "Error Metric"
  ) +
  theme_minimal()
```