Load libraries

```
# Clear all variables in the workplace
rm(list=ls())

# Load the forecasting package
library(fpp2)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
## -- Attaching packages --------------------------------------------- fpp2 2.5 --
```

```
## v ggplot2   3.5.0      v fma       2.5
## v forecast  8.23.0     v expsmooth 2.3
```

```
##
```

```
library(pacman)

p_load(tidyverse, lubridate, stringr, scales, cowplot, dygraphs, xts, imputeTS, forecast, zoo, fma, exps
```

Load Data

```
path <- "D:/An I5/TSA/Mini Project/Code/household_power_consumption.txt"

# Import the data
data <- read_delim(path, ";", escape_double = FALSE, trim_ws = TRUE)
```

```
## Rows: 2075259 Columns: 9
## -- Column specification --------------------------------------------------------
## Delimiter: ";"
## chr  (7): Date, Global_active_power, Global_reactive_power, Voltage, Global_...
## dbl  (1): Sub_metering_3
## time (1): Time
##
## 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.
```

Data preprocessing

```
# Convert the Date column to Date type
data <- data %>%
  mutate(Date = dmy(Date))  # Assuming the format is Day-Month-Year

# Convert Time column to proper time format (if needed)
data <- data %>%
  mutate(Time = hms(Time))  # Requires lubridate for parsing time
```

Handling Invalid and Missing Values

Replace invalid by NA

Replace NA by mean

```
# Check for problematic rows in all columns
invalid_rows <- data %>%
  filter(!str_detect(Global_active_power, "^\\d+(\\.\\d+)?$") |
           !str_detect(Global_reactive_power, "^\\d+(\\.\\d+)?$") |
           !str_detect(Voltage, "^\\d+(\\.\\d+)?$") |
           !str_detect(Global_intensity, "^\\d+(\\.\\d+)?$") |
           !str_detect(Sub_metering_1, "^\\d+(\\.\\d+)?$") |
           !str_detect(Sub_metering_2, "^\\d+(\\.\\d+)?$") |
           !str_detect(Sub_metering_3, "^\\d+(\\.\\d+)?$"))

print(invalid_rows)
```

```
## # A tibble: 25,979 x 9
##    Date       Time       Global_active_power Global_reactive_power Voltage
##    <date>     <Period>   <chr>               <chr>                 <chr>
##  1 2006-12-21 11H 23M 0S ?                   ?                     ?
##  2 2006-12-21 11H 24M 0S ?                   ?                     ?
##  3 2006-12-30 10H 8M 0S  ?                   ?                     ?
##  4 2006-12-30 10H 9M 0S  ?                   ?                     ?
##  5 2007-01-14 18H 36M 0S ?                   ?                     ?
##  6 2007-01-28 17H 13M 0S ?                   ?                     ?
##  7 2007-02-22 22H 58M 0S ?                   ?                     ?
##  8 2007-02-22 22H 59M 0S ?                   ?                     ?
##  9 2007-03-25 17H 52M 0S ?                   ?                     ?
## 10 2007-04-28 21M 0S     ?                   ?                     ?
## # i 25,969 more rows
## # i 4 more variables: Global_intensity <chr>, Sub_metering_1 <chr>,
## #   Sub_metering_2 <chr>, Sub_metering_3 <dbl>
```

```
# Replace invalid values with NA for numeric columns
data <- data %>%
  mutate_at(vars(Global_active_power, Global_reactive_power, Voltage,
                 Global_intensity, Sub_metering_1, Sub_metering_2, Sub_metering_3),
            ~ ifelse(str_detect(., "^\\d+(\\.\\d+)?$"), as.numeric(.), NA))
```

```
## Warning: There were 6 warnings in `mutate()`.
## The first warning was:
## i In argument: `Global_active_power = (structure(function (..., .x = ..1, .y =
##   ..2, . = ..1) ...`.
## Caused by warning in `ifelse()`:
## ! NAs introduced by coercion
## i Run `dplyr::last_dplyr_warnings()` to see the 5 remaining warnings.
```

```
# Replace NA with column mean
data <- data %>%
  mutate_at(vars(Global_active_power, Global_reactive_power, Voltage,
                 Global_intensity, Sub_metering_1, Sub_metering_2, Sub_metering_3),
            ~ replace(., is.na(.), mean(., na.rm = TRUE)))
```

Convert character columns to numeric

```
# Combine Date and Time into a single Datetime column
data <- data %>%
  mutate(Datetime = as.POSIXct(paste(Date, Time)))

# Convert character columns to numeric
data <- data %>%
  mutate_at(vars(Global_active_power, Global_reactive_power, Voltage,
                 Global_intensity, Sub_metering_1, Sub_metering_2),
            ~ as.numeric(.))
```

View Data

```
# Display the structure of the dataset
print(str(data))
```

```
## tibble [2,075,259 x 10] (S3: tbl_df/tbl/data.frame)
##  $ Date                 : Date[1:2075259], format: "2006-12-16" "2006-12-16" ...
##  $ Time                 :Formal class 'Period' [package "lubridate"] with 6 slots
##   .. ..@ .Data : num [1:2075259] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..@ year  : num [1:2075259] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..@ month : num [1:2075259] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..@ day   : num [1:2075259] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..@ hour  : num [1:2075259] 17 17 17 17 17 17 17 17 17 17 ...
##   .. ..@ minute: num [1:2075259] 24 25 26 27 28 29 30 31 32 33 ...
##  $ Global_active_power  : num [1:2075259] 4.22 5.36 5.37 5.39 3.67 ...
##  $ Global_reactive_power: num [1:2075259] 0.418 0.436 0.498 0.502 0.528 0.522 0.52 0.52 0.51 0.51 ..
##  $ Voltage              : num [1:2075259] 235 234 233 234 236 ...
##  $ Global_intensity     : num [1:2075259] 18.4 23 23 23 15.8 15 15.8 15.8 15.8 15.8 ...
##  $ Sub_metering_1       : num [1:2075259] 0 0 0 0 0 0 0 0 0 0 ...
##  $ Sub_metering_2       : num [1:2075259] 1 1 2 1 1 2 1 1 1 2 ...
##  $ Sub_metering_3       : num [1:2075259] 17 16 17 17 17 17 17 17 17 16 ...
##  $ Datetime             : POSIXct[1:2075259], format: "2006-12-16" "2006-12-16" ...
## NULL
```

```
# Display a summary of the dataset
print(summary(data))
```

```
##       Date                 Time                          Global_active_power
##  Min.   :2006-12-16   Min.   :0S                         Min.   : 0.076
##  1st Qu.:2007-12-12   1st Qu.:6H 0M 0S                   1st Qu.: 0.310
##  Median :2008-12-06   Median :12H 0M 0S                  Median : 0.630
##  Mean   :2008-12-05   Mean   :11H 59M 32.7448092021295S  Mean   : 1.092
##  3rd Qu.:2009-12-01   3rd Qu.:18H 0M 0S                  3rd Qu.: 1.520
##  Max.   :2010-11-26   Max.   :23H 59M 0S                 Max.   :11.122
##  Global_reactive_power    Voltage      Global_intensity Sub_metering_1
##  Min.   :0.0000        Min.   :223.2   Min.   : 0.200   Min.   : 0.000
##  1st Qu.:0.0480        1st Qu.:239.0   1st Qu.: 1.400   1st Qu.: 0.000
##  Median :0.1020        Median :241.0   Median : 2.800   Median : 0.000
##  Mean   :0.1237        Mean   :240.8   Mean   : 4.628   Mean   : 1.122
##  3rd Qu.:0.1920        3rd Qu.:242.9   3rd Qu.: 6.400   3rd Qu.: 0.000
##  Max.   :1.3900        Max.   :254.2   Max.   :48.400   Max.   :88.000
##  Sub_metering_2   Sub_metering_3      Datetime
```

```
##  Min.   : 0.000   Min.   : 0.000   Min.   :2006-12-16 00:00:00.00
##  1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.:2007-12-12 00:00:00.00
##  Median : 0.000   Median : 1.000   Median :2008-12-06 00:00:00.00
##  Mean   : 1.299   Mean   : 6.458   Mean   :2008-12-05 19:13:27.25
##  3rd Qu.: 1.000   3rd Qu.:17.000   3rd Qu.:2009-12-01 00:00:00.00
##  Max.   :80.000   Max.   :31.000   Max.   :2010-11-26 00:00:00.00
```

```r
# Count missing values in each column
print(colSums(is.na(data)))
```

```
##                  Date                  Time   Global_active_power
##                     0                     0                     0
## Global_reactive_power               Voltage      Global_intensity
##                     0                     0                     0
##        Sub_metering_1        Sub_metering_2        Sub_metering_3
##                     0                     0                     0
##              Datetime
##                     0
```

Explore Global Active Power (GAP)

```r
# select datetime and Global
gap_data <- data %>%
  select(Date, Time, Datetime, Global_active_power)

# Display structure and summary
str(gap_data)
```

```
## tibble [2,075,259 x 4] (S3: tbl_df/tbl/data.frame)
##  $ Date               : Date[1:2075259], format: "2006-12-16" "2006-12-16" ...
##  $ Time               :Formal class 'Period' [package "lubridate"] with 6 slots
##   .. ..@ .Data : num [1:2075259] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..@ year  : num [1:2075259] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..@ month : num [1:2075259] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..@ day   : num [1:2075259] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..@ hour  : num [1:2075259] 17 17 17 17 17 17 17 17 17 17 ...
##   .. ..@ minute: num [1:2075259] 24 25 26 27 28 29 30 31 32 33 ...
##  $ Datetime           : POSIXct[1:2075259], format: "2006-12-16" "2006-12-16" ...
##  $ Global_active_power: num [1:2075259] 4.22 5.36 5.37 5.39 3.67 ...
```

```r
summary(gap_data)
```

```
##       Date                 Time
##  Min.   :2006-12-16   Min.   :0S
##  1st Qu.:2007-12-12   1st Qu.:6H 0M 0S
##  Median :2008-12-06   Median :12H 0M 0S
##  Mean   :2008-12-05   Mean   :11H 59M 32.7448092021295S
##  3rd Qu.:2009-12-01   3rd Qu.:18H 0M 0S
##  Max.   :2010-11-26   Max.   :23H 59M 0S
##     Datetime                      Global_active_power
##  Min.   :2006-12-16 00:00:00.00   Min.   : 0.076
##  1st Qu.:2007-12-12 00:00:00.00   1st Qu.: 0.310
##  Median :2008-12-06 00:00:00.00   Median : 0.630
##  Mean   :2008-12-05 19:13:27.25   Mean   : 1.092
##  3rd Qu.:2009-12-01 00:00:00.00   3rd Qu.: 1.520
```

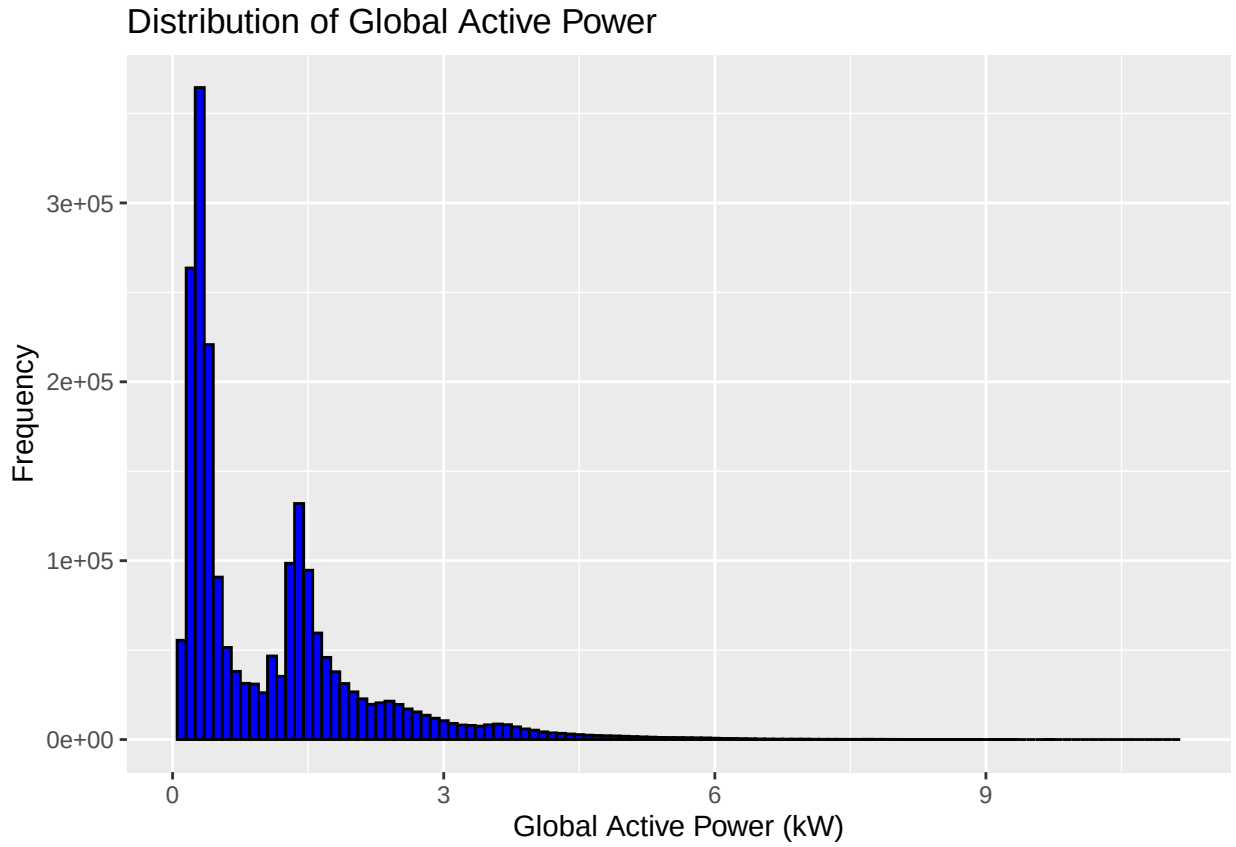
```
##  Max.   :2010-11-26 00:00:00.00   Max.   :11.122
```

```
colSums(is.na(gap_data))
```

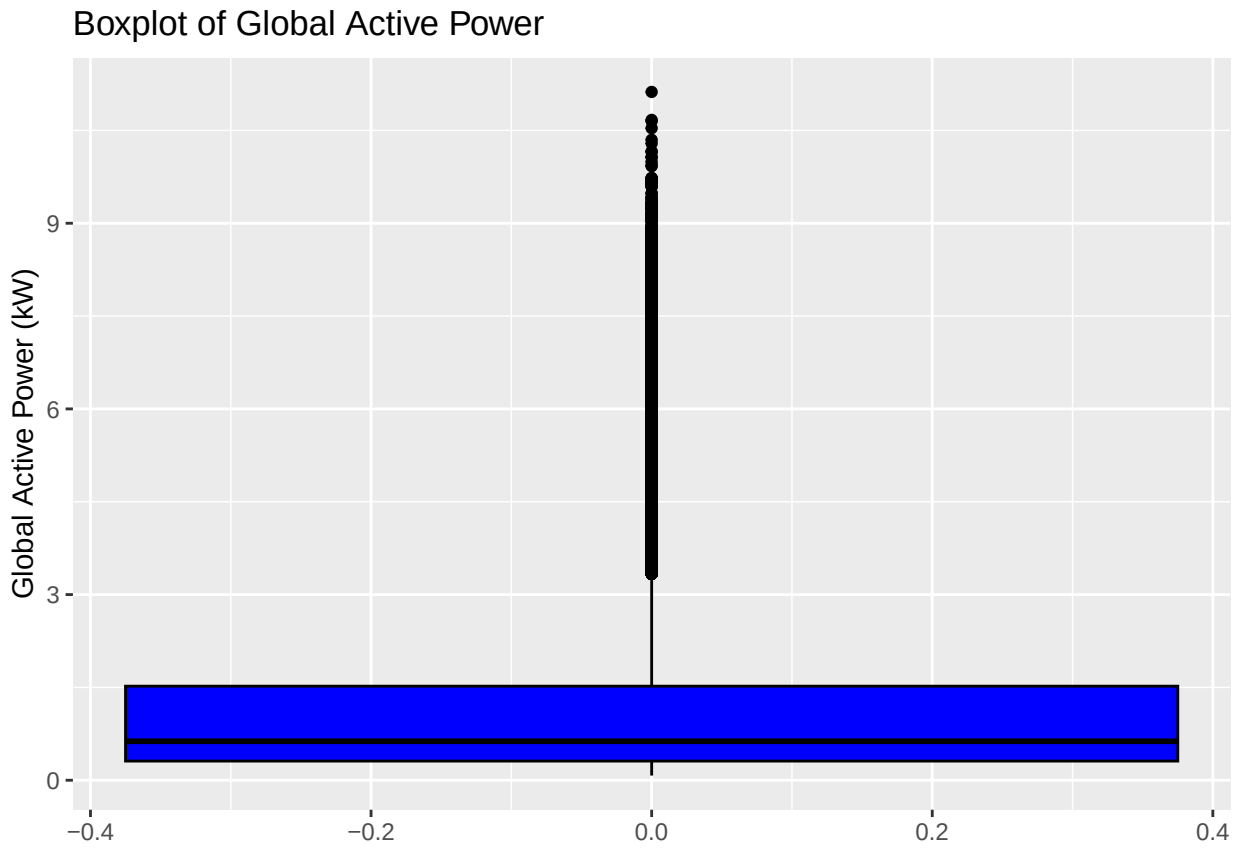
```
##                 Date                 Time             Datetime Global_active_power
##                    0                    0                    0                    0
```

Visualization

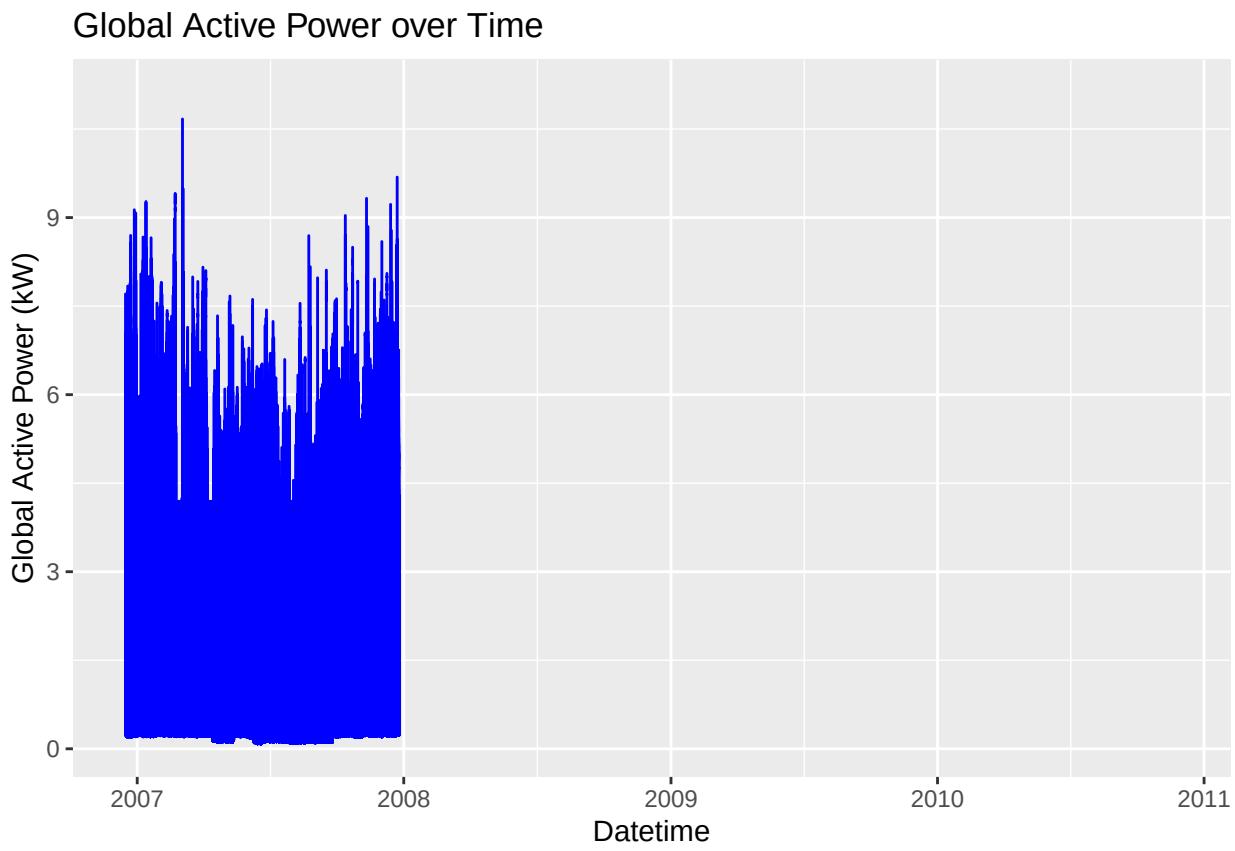
```
# Load necessary libraries
library(ggplot2)

# Histogram of Global_active_power
ggplot(gap_data, aes(x = Global_active_power)) +
  geom_histogram(binwidth = 0.1, fill = "blue", color = "black") +
  labs(title = "Distribution of Global Active Power", x = "Global Active Power (kW)", y = "Frequency")
```




```
# Boxplot of Global_active_power to check for outliers
ggplot(gap_data, aes(y = Global_active_power)) +
  geom_boxplot(fill = "blue", color = "black") +
  labs(title = "Boxplot of Global Active Power", y = "Global Active Power (kW)")
```




```
# Time series plot of Global_active_power over time
ggplot(gap_data, aes(x = Datetime, y = Global_active_power)) +
  geom_line(color = "blue") +
  labs(title = "Global Active Power over Time", x = "Datetime", y = "Global Active Power (kW)")
```

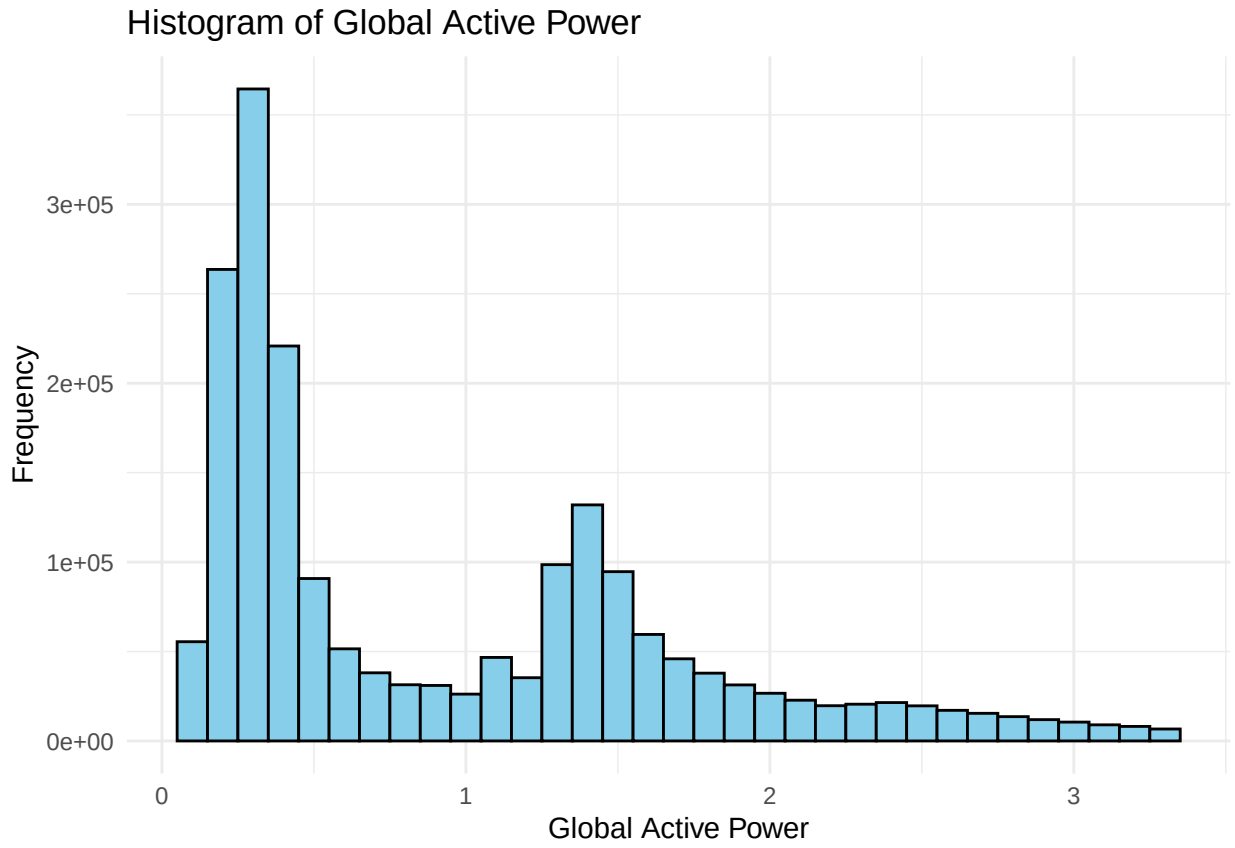


Remove outlier

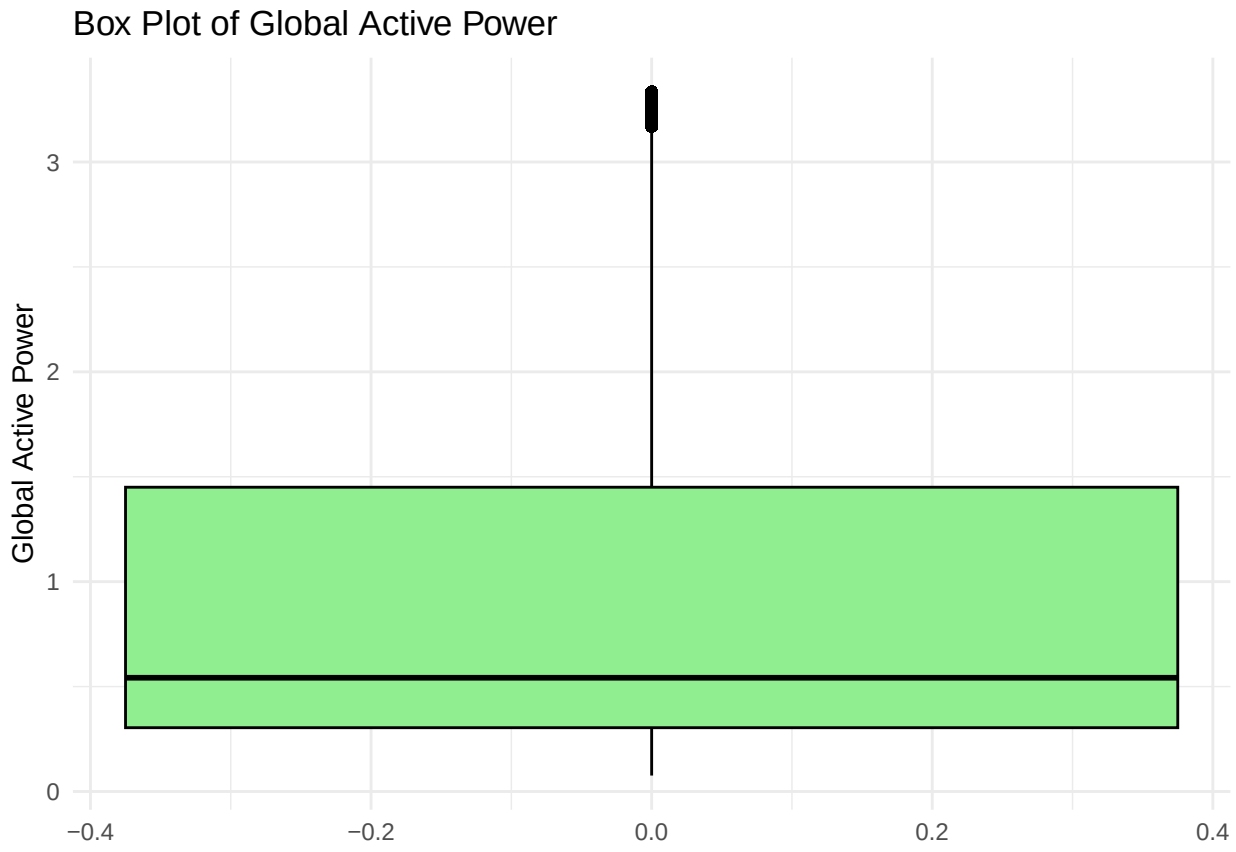
```r
# Function to remove rows with outliers in a column
remove_outliers_from_df <- function(df, column_name) {
  # Calculate the IQR
  IQR_value <- IQR(df[[column_name]], na.rm = TRUE)
  Q1 <- quantile(df[[column_name]], 0.25, na.rm = TRUE)
  Q3 <- quantile(df[[column_name]], 0.75, na.rm = TRUE)

  # Define lower and upper bounds for outliers
  lower_bound <- Q1 - 1.5 * IQR_value
  upper_bound <- Q3 + 1.5 * IQR_value

  # Filter the data frame to keep only rows within bounds
  df <- df[df[[column_name]] >= lower_bound & df[[column_name]] <= upper_bound, ]

  return(df)
}



# gap_data1 <- gap_data
# Apply the function to the entire data frame for the specified column
gap_data <- remove_outliers_from_df(gap_data, "Global_active_power")
```

Visualize again

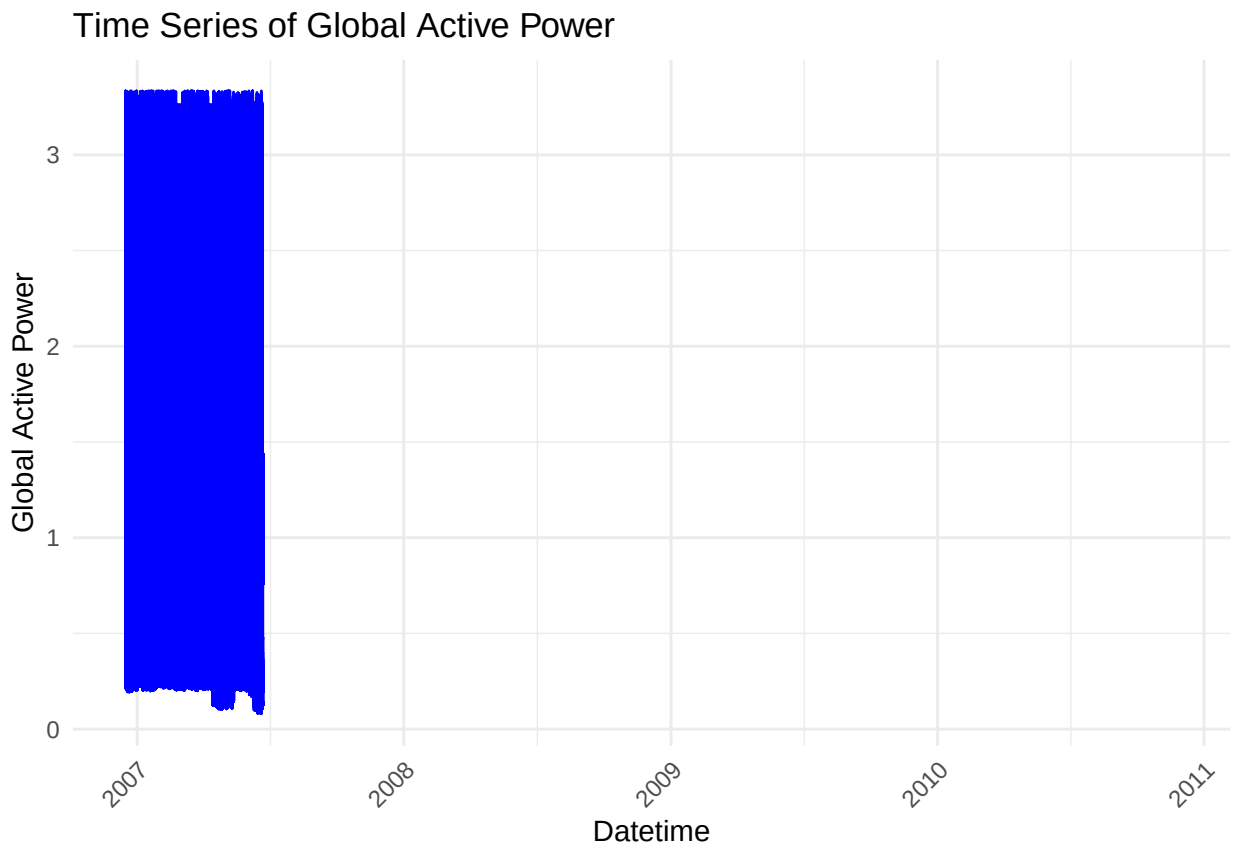
```r
# Load necessary libraries
library(ggplot2)

# Histogram for Global_active_power
ggplot(gap_data, aes(x = Global_active_power)) +
  geom_histogram(binwidth = 0.1, fill = "skyblue", color = "black") +
  labs(title = "Histogram of Global Active Power", x = "Global Active Power", y = "Frequency") +
  theme_minimal()
```




```r
# Box plot for Global_active_power
ggplot(gap_data, aes(y = Global_active_power)) +
  geom_boxplot(fill = "lightgreen", color = "black") +
  labs(title = "Box Plot of Global Active Power", y = "Global Active Power") +
  theme_minimal()
```

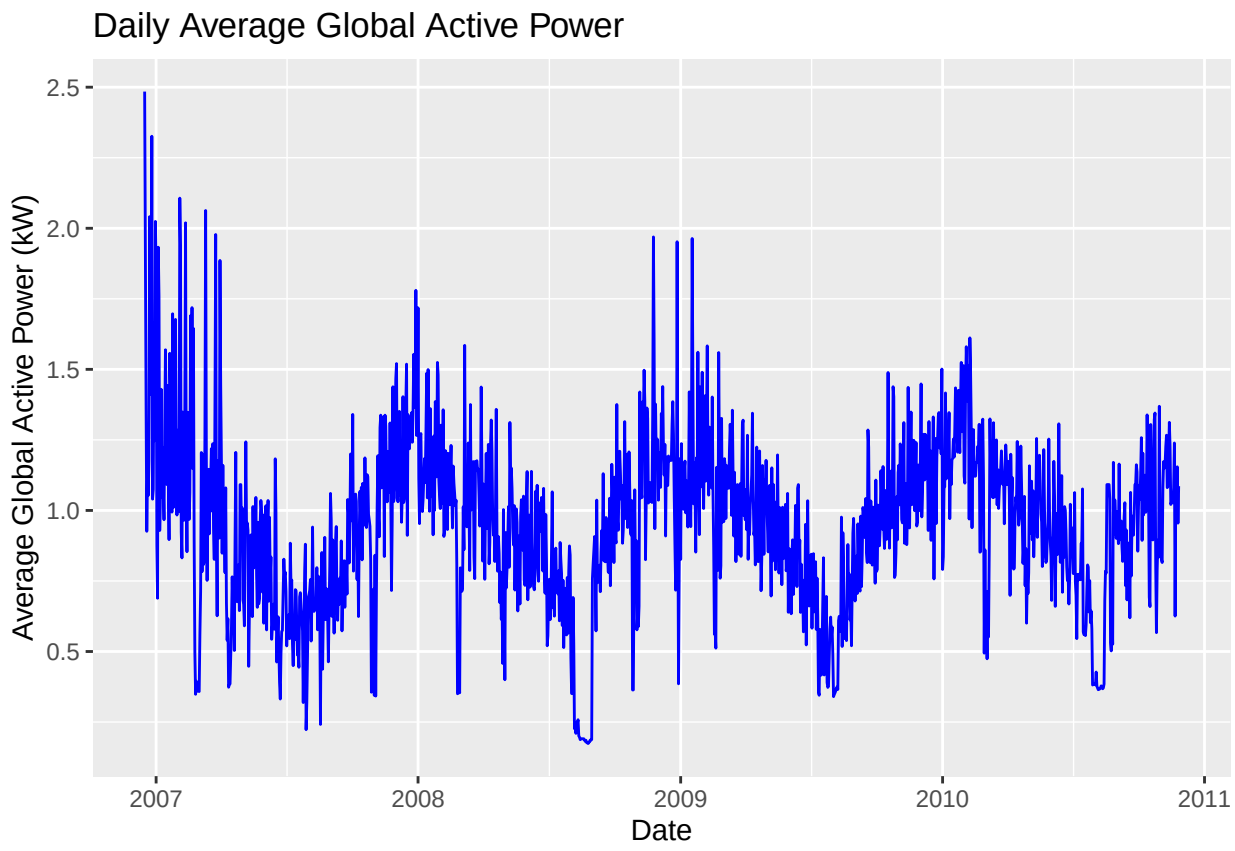



```
# Time series plot for Global_active_power over time
ggplot(gap_data, aes(x = Datetime, y = Global_active_power)) +
  geom_line(color = "blue") +
  labs(title = "Time Series of Global Active Power", x = "Datetime", y = "Global Active Power") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

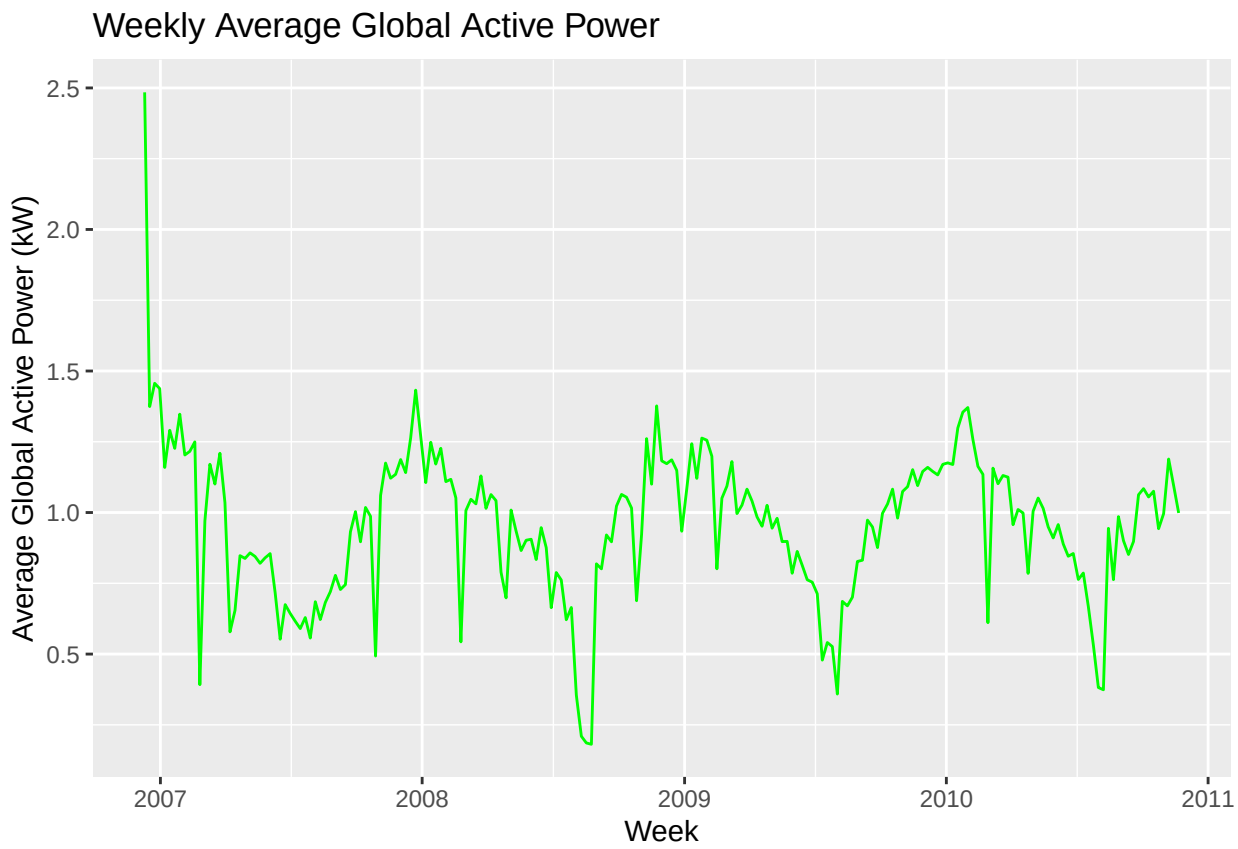



```r
library(dplyr)
library(ggplot2)
library(lubridate)

# Assuming `gap_data` has a "Date" column in Date format and "Global_active_power" column

# 1. Aggregate data to daily level
daily_gap_data <- gap_data %>%
  group_by(Date) %>%
  summarise(Daily_Avg = mean(Global_active_power, na.rm = TRUE))

# Convert to a daily time series
daily_ts <- ts(daily_gap_data$Daily_Avg, start = c(2007, 1), frequency = 365)

# 2. Aggregate data to weekly level
weekly_data <- daily_gap_data %>%
  mutate(Week = floor_date(Date, unit = "week")) %>%
  group_by(Week) %>%
  summarise(Weekly_Avg = mean(Daily_Avg, na.rm = TRUE))

# Convert to a weekly time series
weekly_ts <- ts(weekly_data$Weekly_Avg, start = c(2007, 1), frequency = 52)

# 3. Aggregate data to monthly level
monthly_data <- daily_gap_data %>%
  mutate(Month = floor_date(Date, unit = "month")) %>%
```

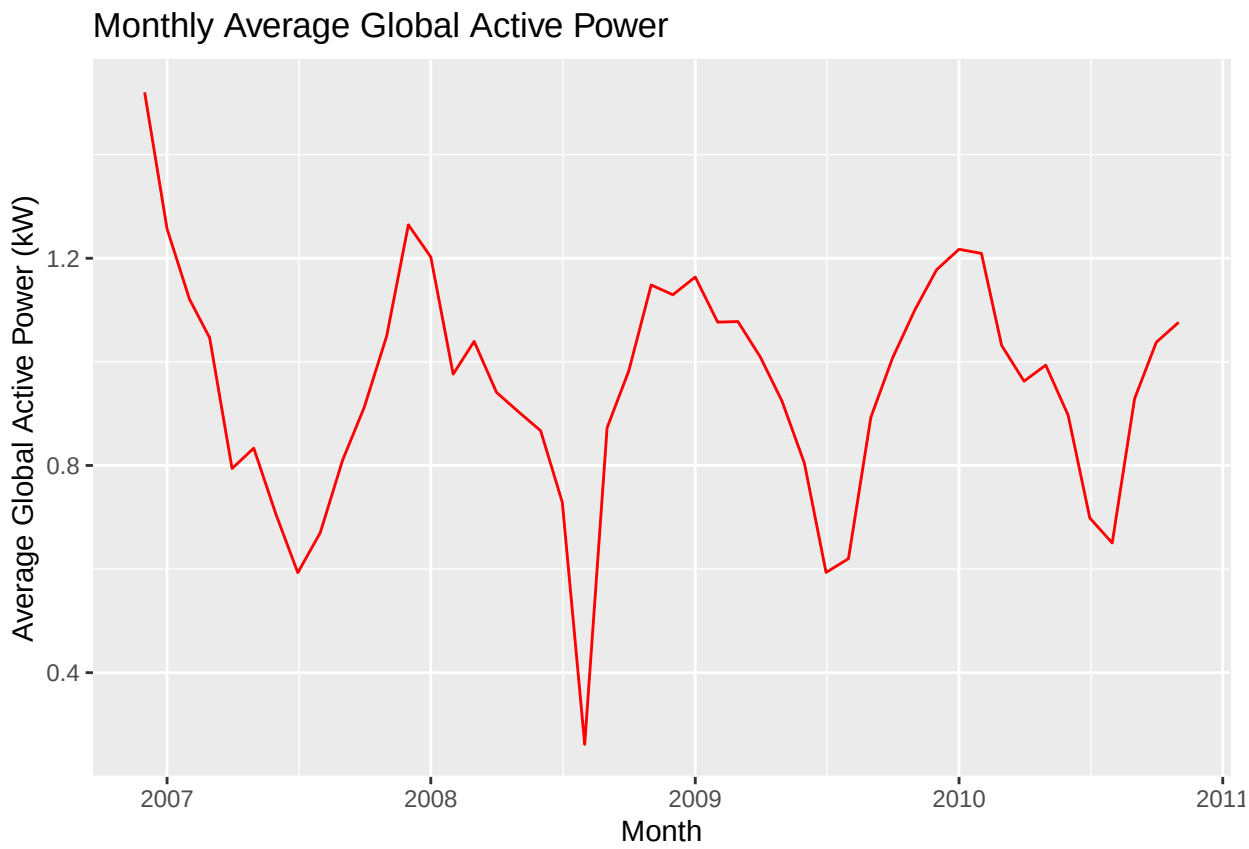
```
  group_by(Month) %>%
  summarise(Monthly_Avg = mean(Daily_Avg, na.rm = TRUE))

# Convert to a monthly time series
monthly_ts <- ts(monthly_data$Monthly_Avg, start = c(2007, 1), frequency = 12)

# 4. Aggregate data to quarterly level
quarterly_data <- daily_gap_data %>%
  mutate(Quarter = paste0(year(Date), " Q", quarter(Date))) %>%
  group_by(Quarter) %>%
  summarise(Quarterly_Avg = mean(Daily_Avg, na.rm = TRUE))

# Convert to a quarterly time series
quarterly_ts <- ts(quarterly_data$Quarterly_Avg, start = c(2007, 1), frequency = 4)

# 5. Plotting examples
# Daily Plot
ggplot(daily_gap_data, aes(x = Date, y = Daily_Avg)) +
  geom_line(color = "blue") +
  labs(title = "Daily Average Global Active Power", x = "Date", y = "Average Global Active Power (kW)")
```




```
# Weekly Plot
ggplot(weekly_data, aes(x = Week, y = Weekly_Avg)) +
  geom_line(color = "green") +
  labs(title = "Weekly Average Global Active Power", x = "Week", y = "Average Global Active Power (kW)"
```

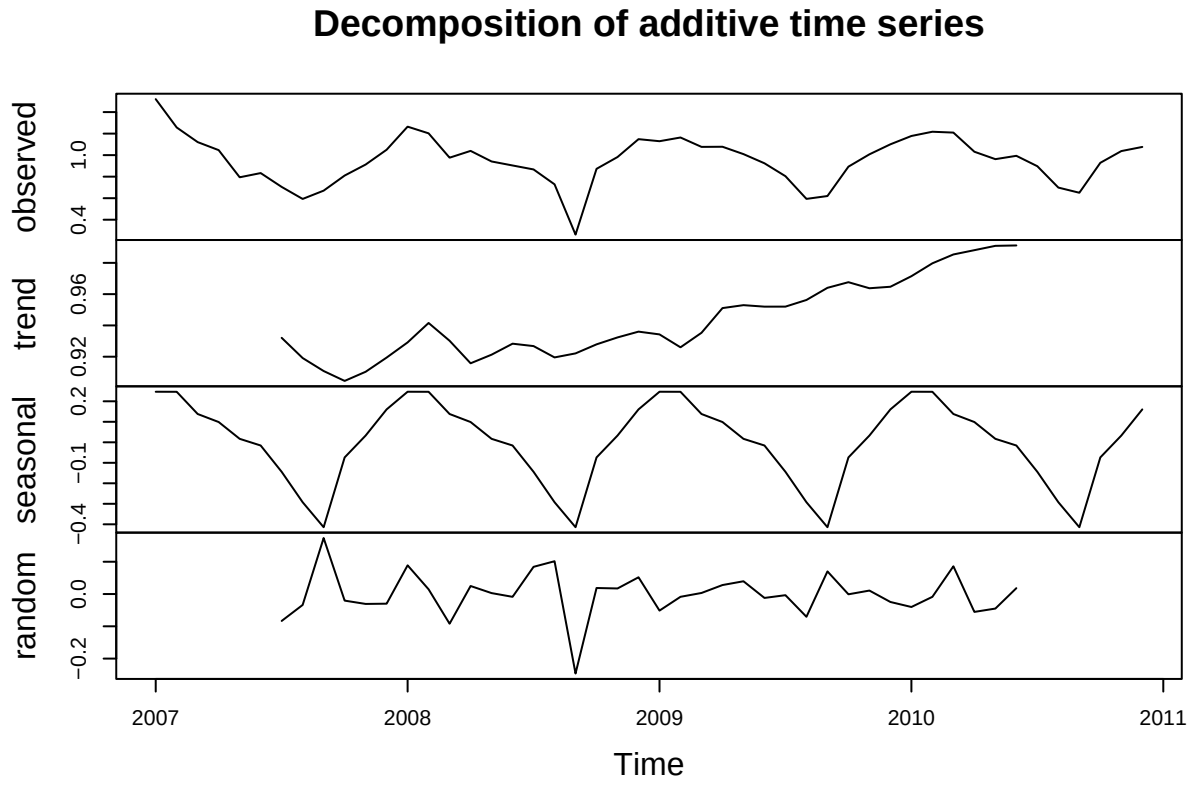



```
# Monthly Plot
ggplot(monthly_data, aes(x = Month, y = Monthly_Avg)) +
  geom_line(color = "red") +
  labs(title = "Monthly Average Global Active Power", x = "Month", y = "Average Global Active Power (kW
```



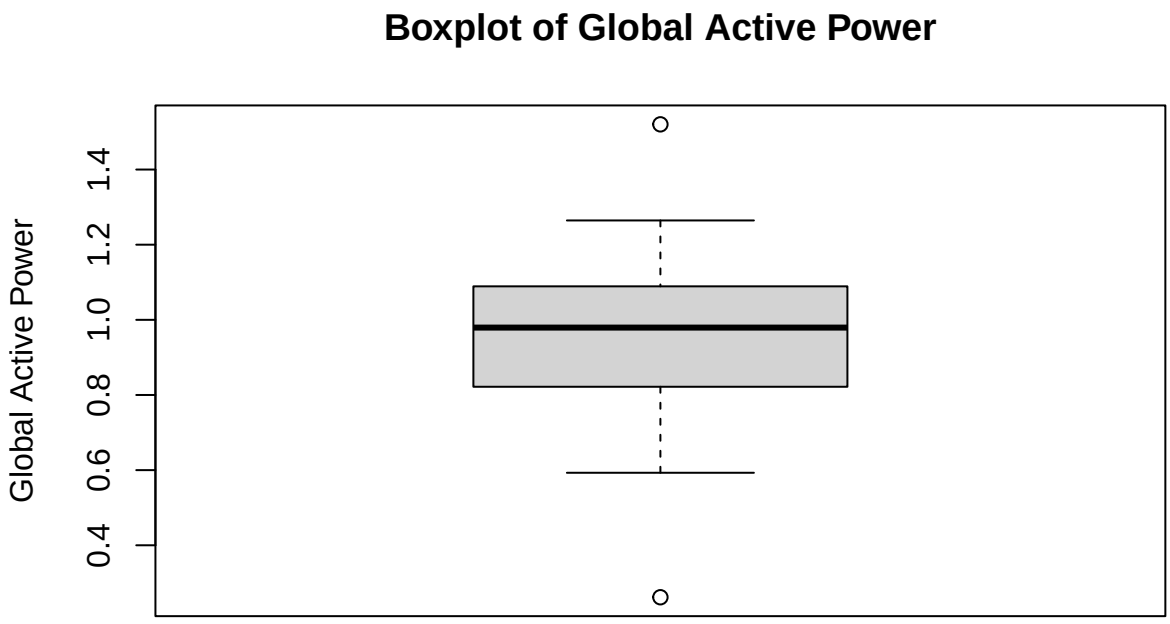
Daily time series decomposed (time plot, trend, season, error)

```r
library(dplyr)
Y <- monthly_ts

# Decompose the quarterly time series
decomposed_Y <- decompose(Y, type = "additive")

# Plot the decomposition
plot(decomposed_Y)
```



Box plot

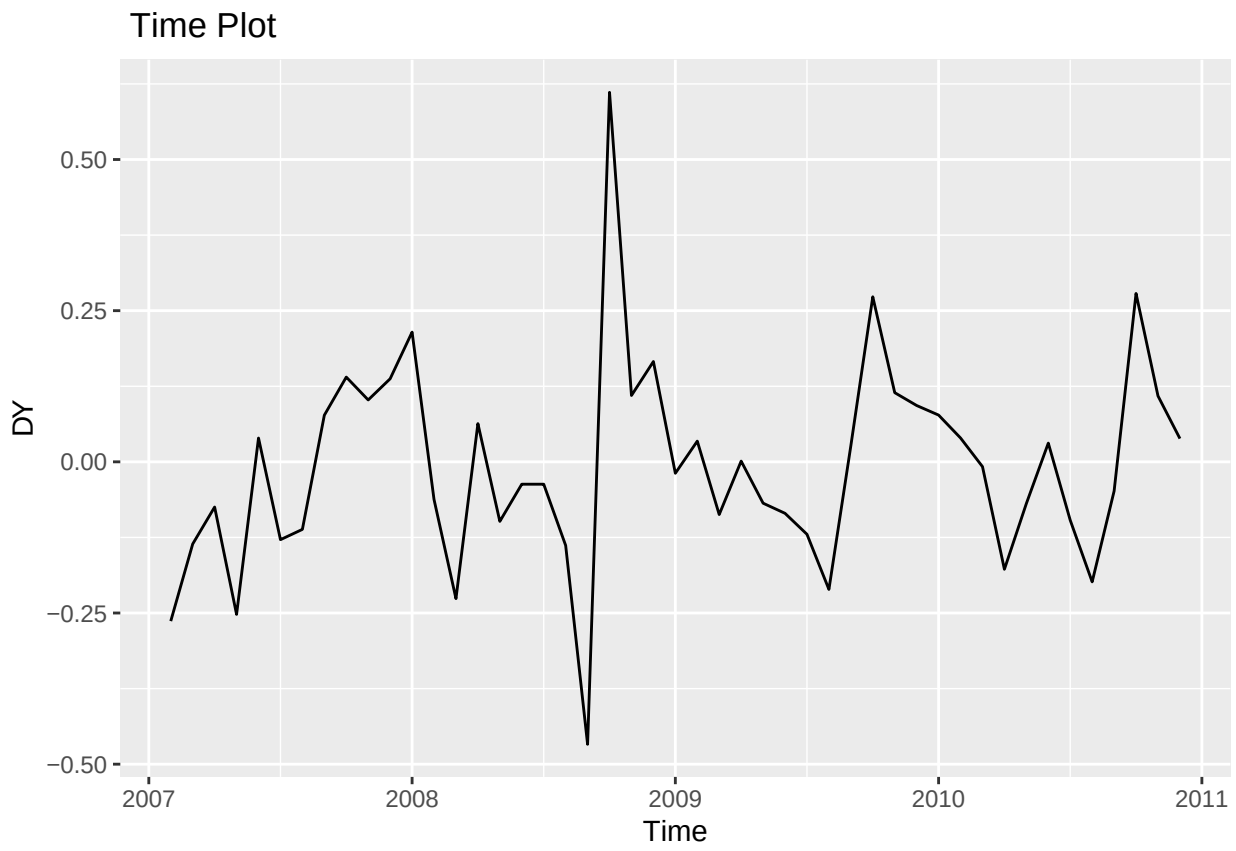
```
boxplot(Y, main = "Boxplot of Global Active Power", ylab = "Global Active Power")
```




```
Q1 <- quantile(Y, 0.25)
Q3 <- quantile(Y, 0.75)
IQR <- Q3 - Q1
lower_bound <- Q1 - 1.5 * IQR
upper_bound <- Q3 + 1.5 * IQR
outliers <- Y[Y < lower_bound | Y > upper_bound]
print(outliers)
```

```
## [1] 1.5203097 0.2616725
```

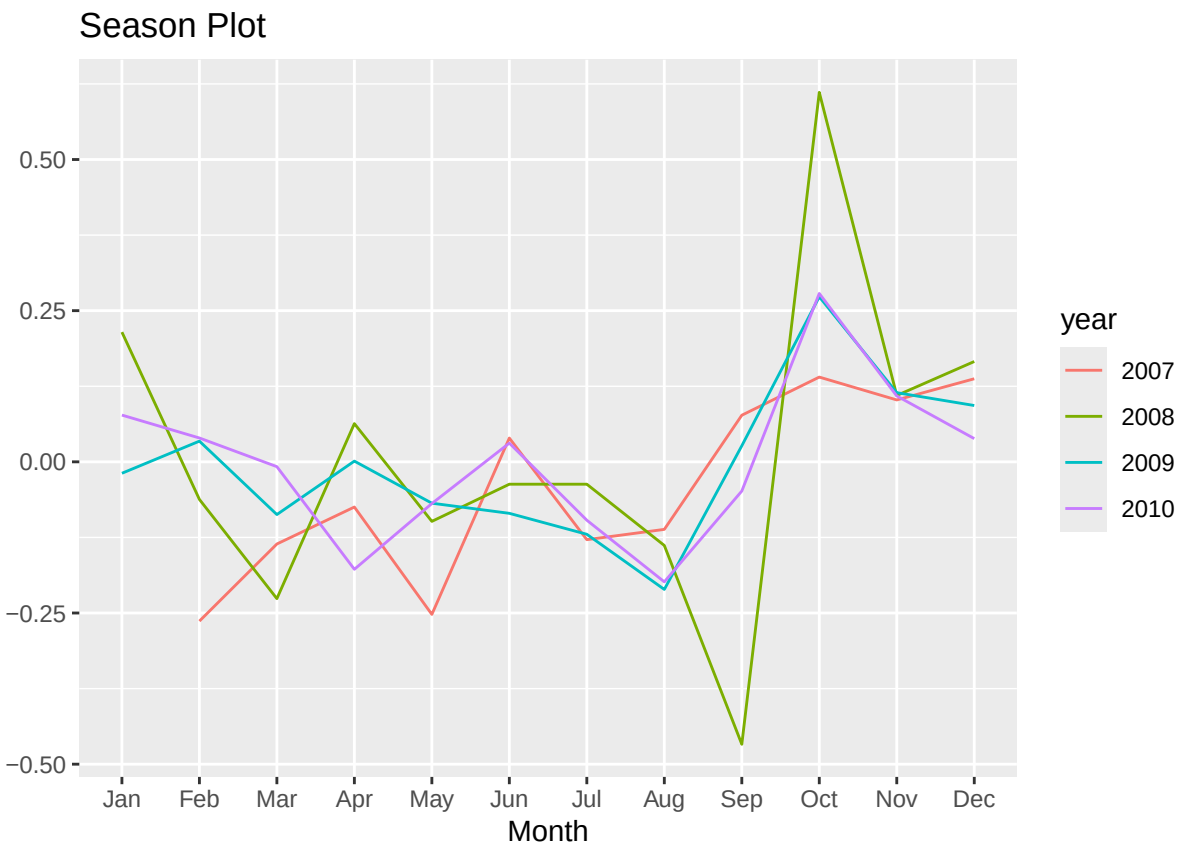
```
DY <- diff(Y)

autoplot(DY) +
  ggtitle(" Time Plot")
```




```
  ylab("KW")
```

```
## $y
## [1] "KW"
##
## attr(,"class")
## [1] "labels"
```

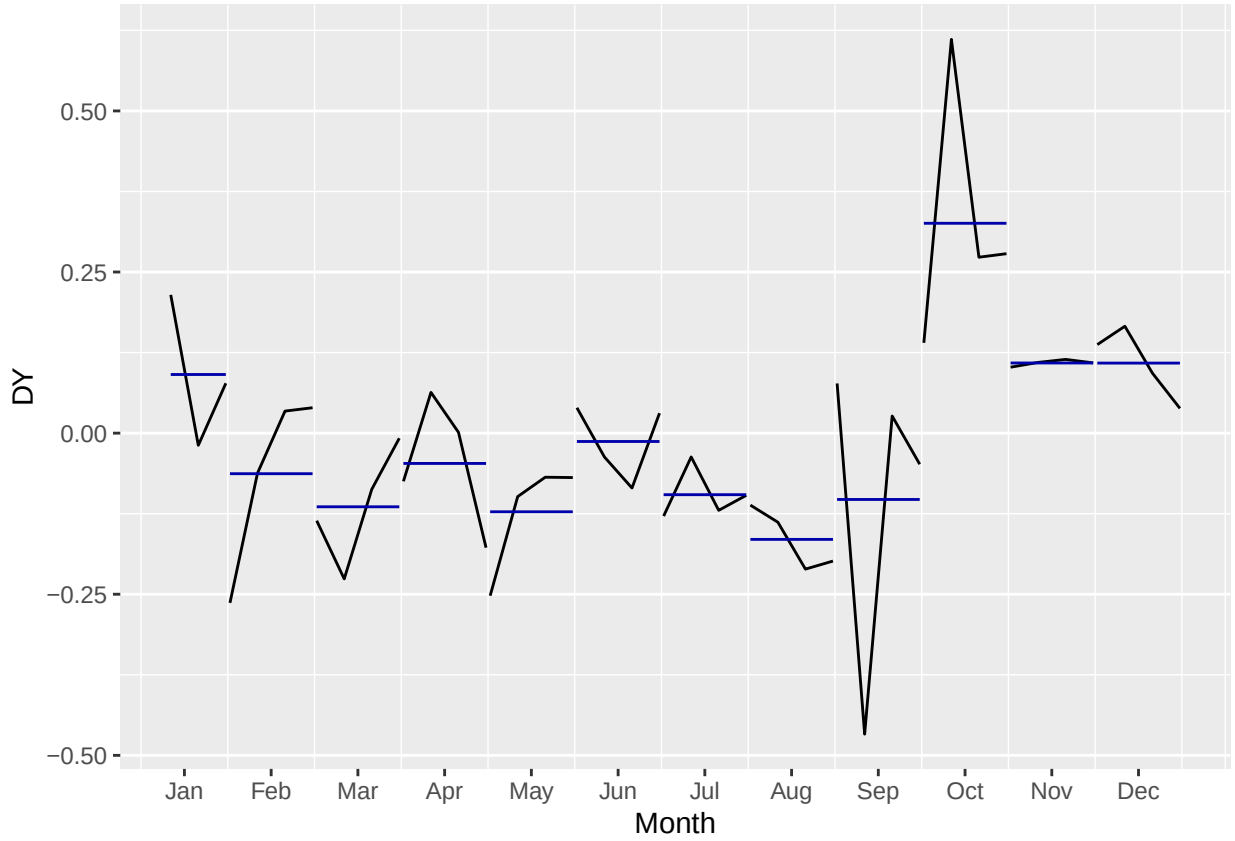
```
ggseasonplot(DY) +
  ggtitle("Season Plot")
```




```
  ylab("KW")
```

```
## $y
## [1] "KW"
##
## attr(,"class")
## [1] "labels"
```

```
ggsubseriesplot(DY)
```




```
if (!require(tseries)) install.packages("tseries")
```

```
## Loading required package: tseries
```

```
##
## Attaching package: 'tseries'
```

```
## The following object is masked from 'package:imputeTS':
##
##     na.remove
```

```
library(tseries)
adf_test <- adf.test(DY)
print(adf_test)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  DY
## Dickey-Fuller = -3.5608, Lag order = 3, p-value = 0.04636
## alternative hypothesis: stationary
```

```
# we can use Y instead of DY in auto.arima
fit_arima <- auto.arima(Y, stepwise = FALSE, approximation = FALSE, trace=TRUE)
```

```
##
##  ARIMA(0,0,0)(0,1,0)[12]                    : -43.61482
##  ARIMA(0,0,0)(0,1,0)[12] with drift         : -41.91935
##  ARIMA(0,0,0)(0,1,1)[12]                    : -45.94201
```

```
##  ARIMA(0,0,0)(0,1,1)[12] with drift         : -44.92186
##  ARIMA(0,0,0)(1,1,0)[12]                    : -45.11452
##  ARIMA(0,0,0)(1,1,0)[12] with drift         : -43.7827
##  ARIMA(0,0,0)(1,1,1)[12]                    : -43.63374
##  ARIMA(0,0,0)(1,1,1)[12] with drift         : Inf
##  ARIMA(0,0,1)(0,1,0)[12]                    : -41.44502
##  ARIMA(0,0,1)(0,1,0)[12] with drift         : -39.78399
##  ARIMA(0,0,1)(0,1,1)[12]                    : -43.86056
##  ARIMA(0,0,1)(0,1,1)[12] with drift         : -42.41833
##  ARIMA(0,0,1)(1,1,0)[12]                    : -42.88964
##  ARIMA(0,0,1)(1,1,0)[12] with drift         : -41.25696
##  ARIMA(0,0,1)(1,1,1)[12]                    : -41.3752
##  ARIMA(0,0,1)(1,1,1)[12] with drift         : Inf
##  ARIMA(0,0,2)(0,1,0)[12]                    : -39.88052
##  ARIMA(0,0,2)(0,1,0)[12] with drift         : -39.71918
##  ARIMA(0,0,2)(0,1,1)[12]                    : -41.33437
##  ARIMA(0,0,2)(0,1,1)[12] with drift         : -40.49412
##  ARIMA(0,0,2)(1,1,0)[12]                    : -40.41788
##  ARIMA(0,0,2)(1,1,0)[12] with drift         : -39.5999
##  ARIMA(0,0,2)(1,1,1)[12]                    : -38.6805
##  ARIMA(0,0,2)(1,1,1)[12] with drift         : -37.72143
##  ARIMA(0,0,3)(0,1,0)[12]                    : -37.80197
##  ARIMA(0,0,3)(0,1,0)[12] with drift         : -37.9322
##  ARIMA(0,0,3)(0,1,1)[12]                    : -38.68213
##  ARIMA(0,0,3)(0,1,1)[12] with drift         : -38.69753
##  ARIMA(0,0,3)(1,1,0)[12]                    : -37.91197
##  ARIMA(0,0,3)(1,1,0)[12] with drift         : -37.92248
##  ARIMA(0,0,3)(1,1,1)[12]                    : -35.82862
##  ARIMA(0,0,3)(1,1,1)[12] with drift         : -35.62703
##  ARIMA(0,0,4)(0,1,0)[12]                    : -35.16371
##  ARIMA(0,0,4)(0,1,0)[12] with drift         : -35.06909
##  ARIMA(0,0,4)(0,1,1)[12]                    : -35.7857
##  ARIMA(0,0,4)(0,1,1)[12] with drift         : -35.7376
##  ARIMA(0,0,4)(1,1,0)[12]                    : -35.03731
##  ARIMA(0,0,4)(1,1,0)[12] with drift         : -34.99525
##  ARIMA(0,0,5)(0,1,0)[12]                    : -32.8365
##  ARIMA(0,0,5)(0,1,0)[12] with drift         : -34.10325
##  ARIMA(1,0,0)(0,1,0)[12]                    : -41.42269
##  ARIMA(1,0,0)(0,1,0)[12] with drift         : -39.68322
##  ARIMA(1,0,0)(0,1,1)[12]                    : -43.83861
##  ARIMA(1,0,0)(0,1,1)[12] with drift         : -42.40898
##  ARIMA(1,0,0)(1,1,0)[12]                    : -42.87074
##  ARIMA(1,0,0)(1,1,0)[12] with drift         : -41.25314
##  ARIMA(1,0,0)(1,1,1)[12]                    : -41.3551
##  ARIMA(1,0,0)(1,1,1)[12] with drift         : Inf
##  ARIMA(1,0,1)(0,1,0)[12]                    : -39.66791
##  ARIMA(1,0,1)(0,1,0)[12] with drift         : -39.49368
##  ARIMA(1,0,1)(0,1,1)[12]                    : -41.32754
##  ARIMA(1,0,1)(0,1,1)[12] with drift         : -39.76651
##  ARIMA(1,0,1)(1,1,0)[12]                    : -40.36877
##  ARIMA(1,0,1)(1,1,0)[12] with drift         : -38.58986
##  ARIMA(1,0,1)(1,1,1)[12]                    : Inf
##  ARIMA(1,0,1)(1,1,1)[12] with drift         : Inf
##  ARIMA(1,0,2)(0,1,0)[12]                    : -37.5503
```

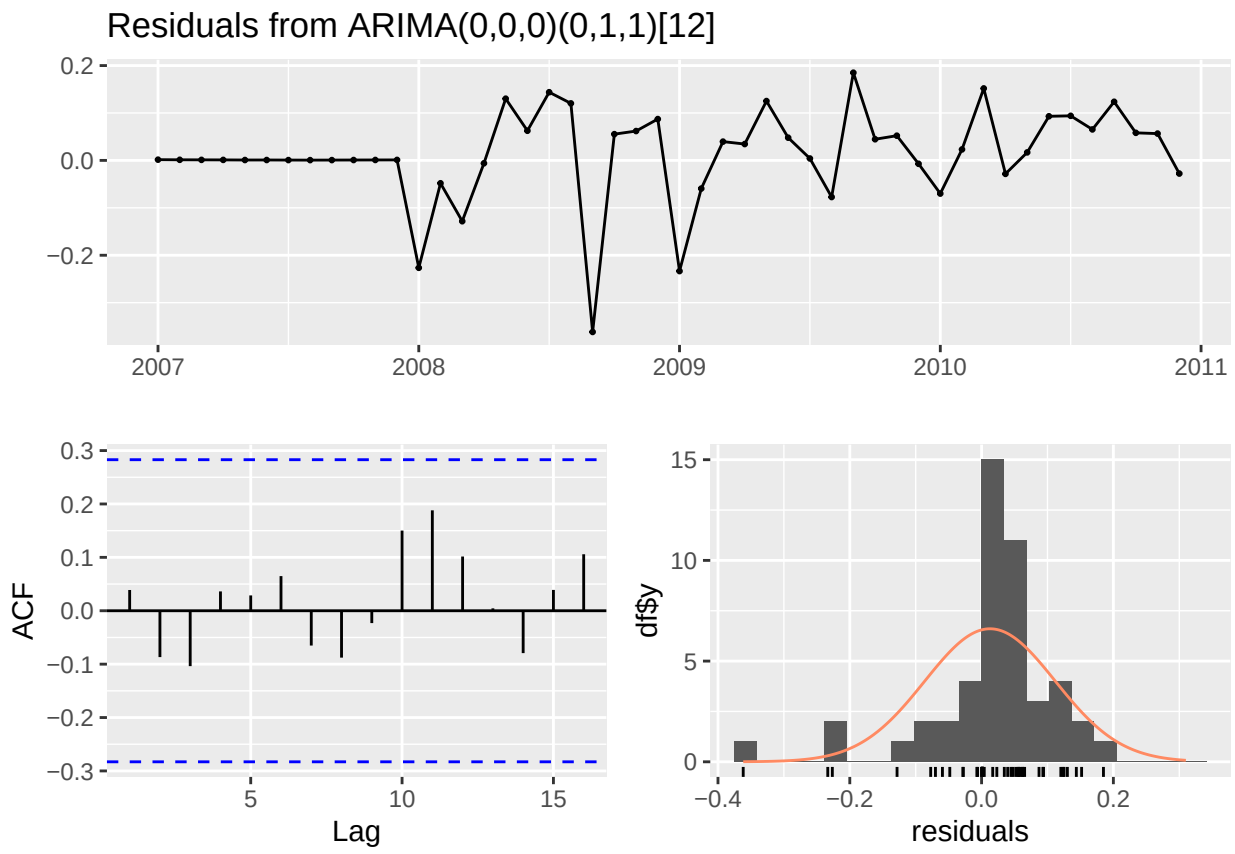
```
##  ARIMA(1,0,2)(0,1,0)[12] with drift         : -37.50306
##  ARIMA(1,0,2)(0,1,1)[12]                    : -38.64299
##  ARIMA(1,0,2)(0,1,1)[12] with drift         : -38.49105
##  ARIMA(1,0,2)(1,1,0)[12]                    : -37.8167
##  ARIMA(1,0,2)(1,1,0)[12] with drift         : -37.64725
##  ARIMA(1,0,2)(1,1,1)[12]                    : -35.7956
##  ARIMA(1,0,2)(1,1,1)[12] with drift         : Inf
##  ARIMA(1,0,3)(0,1,0)[12]                    : -35.12169
##  ARIMA(1,0,3)(0,1,0)[12] with drift         : -35.04706
##  ARIMA(1,0,3)(0,1,1)[12]                    : -35.78557
##  ARIMA(1,0,3)(0,1,1)[12] with drift         : -35.66823
##  ARIMA(1,0,3)(1,1,0)[12]                    : -35.02481
##  ARIMA(1,0,3)(1,1,0)[12] with drift         : -34.89664
##  ARIMA(1,0,4)(0,1,0)[12]                    : Inf
##  ARIMA(1,0,4)(0,1,0)[12] with drift         : Inf
##  ARIMA(2,0,0)(0,1,0)[12]                    : -39.65728
##  ARIMA(2,0,0)(0,1,0)[12] with drift         : -38.14887
##  ARIMA(2,0,0)(0,1,1)[12]                    : -41.35271
##  ARIMA(2,0,0)(0,1,1)[12] with drift         : -40.15719
##  ARIMA(2,0,0)(1,1,0)[12]                    : -40.45327
##  ARIMA(2,0,0)(1,1,0)[12] with drift         : -39.02291
##  ARIMA(2,0,0)(1,1,1)[12]                    : -38.69609
##  ARIMA(2,0,0)(1,1,1)[12] with drift         : Inf
##  ARIMA(2,0,1)(0,1,0)[12]                    : -37.75581
##  ARIMA(2,0,1)(0,1,0)[12] with drift         : -38.03227
##  ARIMA(2,0,1)(0,1,1)[12]                    : -38.66307
##  ARIMA(2,0,1)(0,1,1)[12] with drift         : -38.8709
##  ARIMA(2,0,1)(1,1,0)[12]                    : Inf
##  ARIMA(2,0,1)(1,1,0)[12] with drift         : -38.09345
##  ARIMA(2,0,1)(1,1,1)[12]                    : -35.81127
##  ARIMA(2,0,1)(1,1,1)[12] with drift         : Inf
##  ARIMA(2,0,2)(0,1,0)[12]                    : Inf
##  ARIMA(2,0,2)(0,1,0)[12] with drift         : Inf
##  ARIMA(2,0,2)(0,1,1)[12]                    : Inf
##  ARIMA(2,0,2)(0,1,1)[12] with drift         : Inf
##  ARIMA(2,0,2)(1,1,0)[12]                    : Inf
##  ARIMA(2,0,2)(1,1,0)[12] with drift         : Inf
##  ARIMA(2,0,3)(0,1,0)[12]                    : -32.26617
##  ARIMA(2,0,3)(0,1,0)[12] with drift         : Inf
##  ARIMA(3,0,0)(0,1,0)[12]                    : -38.16308
##  ARIMA(3,0,0)(0,1,0)[12] with drift         : -37.57579
##  ARIMA(3,0,0)(0,1,1)[12]                    : -38.69636
##  ARIMA(3,0,0)(0,1,1)[12] with drift         : -38.11293
##  ARIMA(3,0,0)(1,1,0)[12]                    : -37.9907
##  ARIMA(3,0,0)(1,1,0)[12] with drift         : -37.32143
##  ARIMA(3,0,0)(1,1,1)[12]                    : -35.83765
##  ARIMA(3,0,0)(1,1,1)[12] with drift         : -35.09466
##  ARIMA(3,0,1)(0,1,0)[12]                    : -35.50572
##  ARIMA(3,0,1)(0,1,0)[12] with drift         : -36.10696
##  ARIMA(3,0,1)(0,1,1)[12]                    : -35.80126
##  ARIMA(3,0,1)(0,1,1)[12] with drift         : -36.27761
##  ARIMA(3,0,1)(1,1,0)[12]                    : -35.11211
##  ARIMA(3,0,1)(1,1,0)[12] with drift         : -35.73178
##  ARIMA(3,0,2)(0,1,0)[12]                    : Inf
```

```
##  ARIMA(3,0,2)(0,1,0)[12] with drift         : Inf
##  ARIMA(4,0,0)(0,1,0)[12]                    : -35.52999
##  ARIMA(4,0,0)(0,1,0)[12] with drift         : -35.73204
##  ARIMA(4,0,0)(0,1,1)[12]                    : -35.80323
##  ARIMA(4,0,0)(0,1,1)[12] with drift         : -35.9086
##  ARIMA(4,0,0)(1,1,0)[12]                    : -35.13055
##  ARIMA(4,0,0)(1,1,0)[12] with drift         : -35.4184
##  ARIMA(4,0,1)(0,1,0)[12]                    : -32.63561
##  ARIMA(4,0,1)(0,1,0)[12] with drift         : -33.13168
##  ARIMA(5,0,0)(0,1,0)[12]                    : -32.64588
##  ARIMA(5,0,0)(0,1,0)[12] with drift         : -33.13475
##
##
##
##  Best model: ARIMA(0,0,0)(0,1,1)[12]
```

```
print(summary(fit_arima))
```

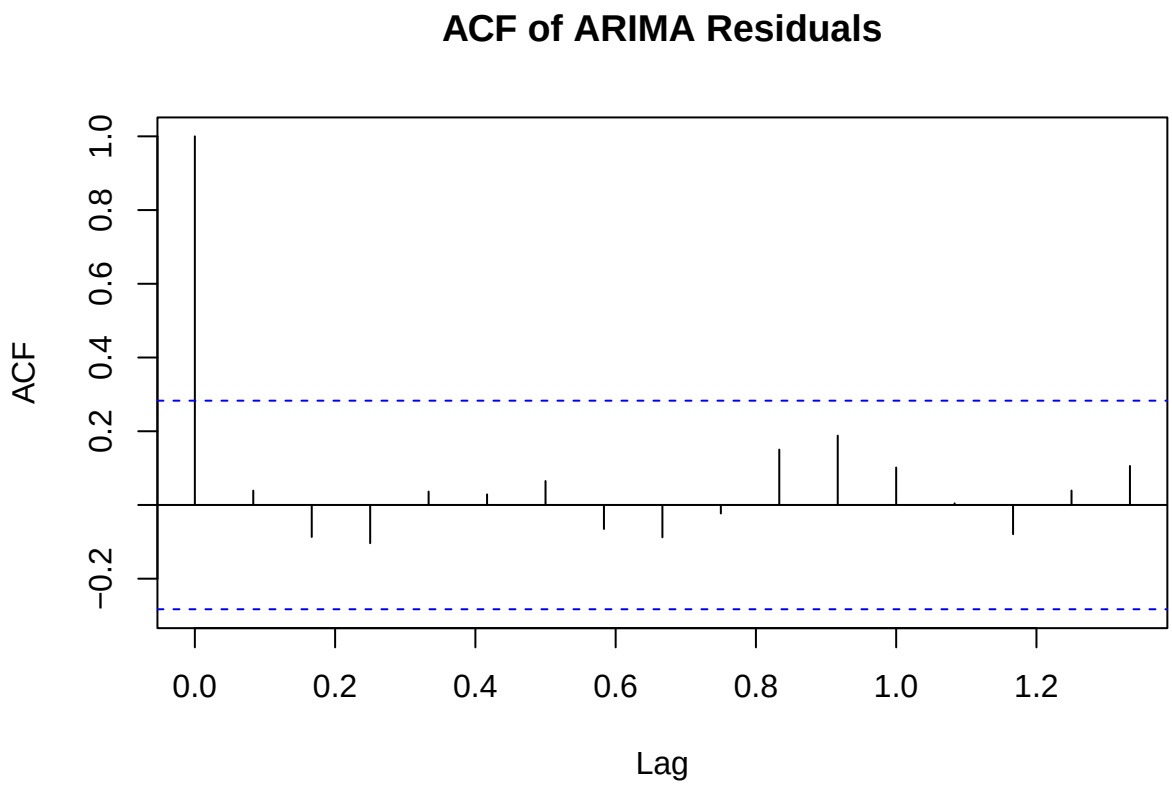
```
## Series: Y
## ARIMA(0,0,0)(0,1,1)[12]
##
## Coefficients:
##          sma1
##       -0.5243
## s.e.   0.2954
##
## sigma^2 = 0.0134:  log likelihood = 25.15
## AIC=-46.31   AICc=-45.94   BIC=-43.14
##
## Training set error measures:
##                       ME       RMSE        MAE        MPE     MAPE      MASE
## Training set 0.01278253 0.09885724 0.06587471 -0.1720544 9.186196 0.7010151
##                    ACF1
## Training set 0.03888812
```

```
checkresiduals(fit_arima)
```

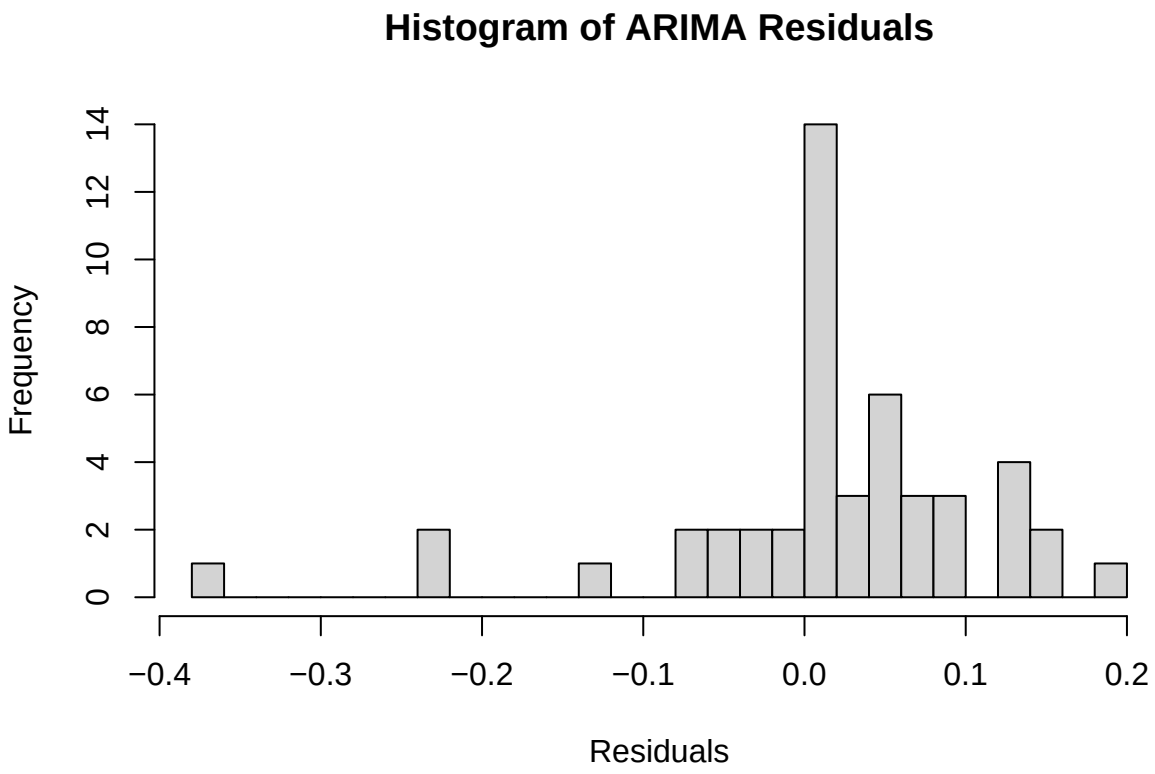



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,0,0)(0,1,1)[12]
## Q* = 3.5698, df = 9, p-value = 0.9374
##
## Model df: 1.   Total lags used: 10
```

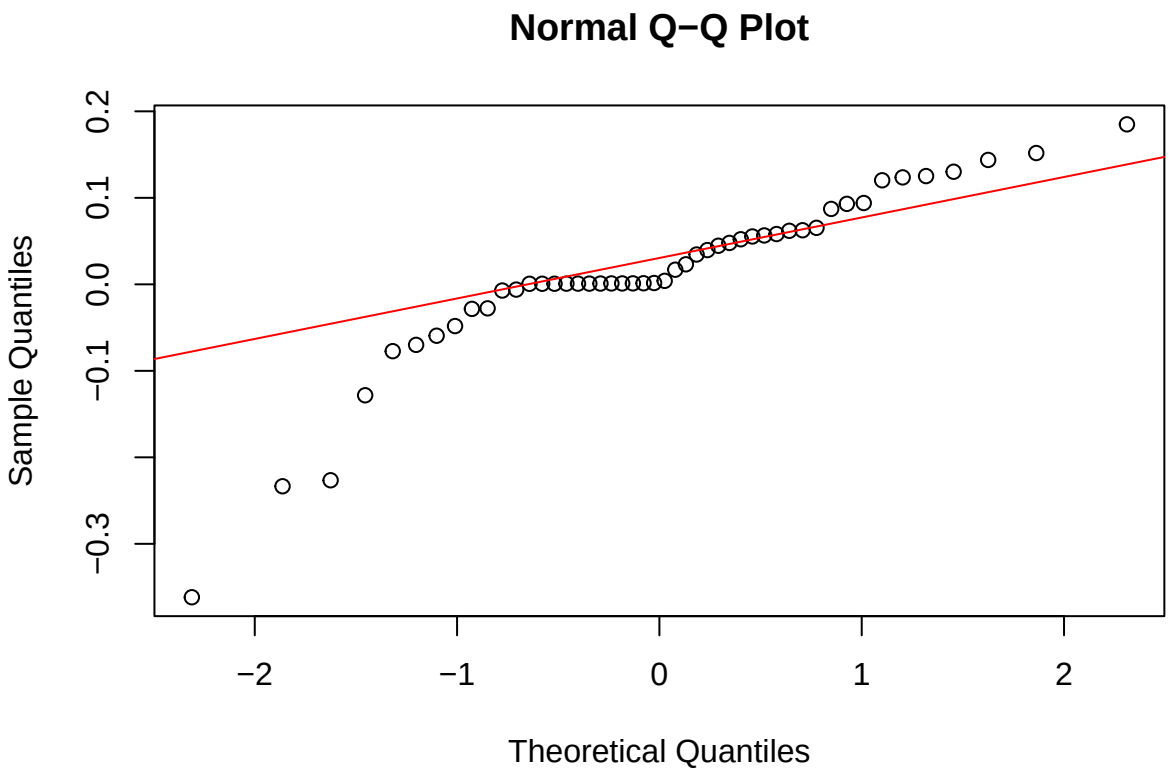
```
# ACF of residuals (manually if you want more control)
acf(residuals(fit_arima), main = "ACF of ARIMA Residuals")
```




```r
# Histogram of residuals
hist(residuals(fit_arima), main = "Histogram of ARIMA Residuals", xlab = "Residuals", breaks = 30)
```




```r
# QQ-plot of residuals
qqnorm(residuals(fit_arima))
qqline(residuals(fit_arima), col = "red")
```



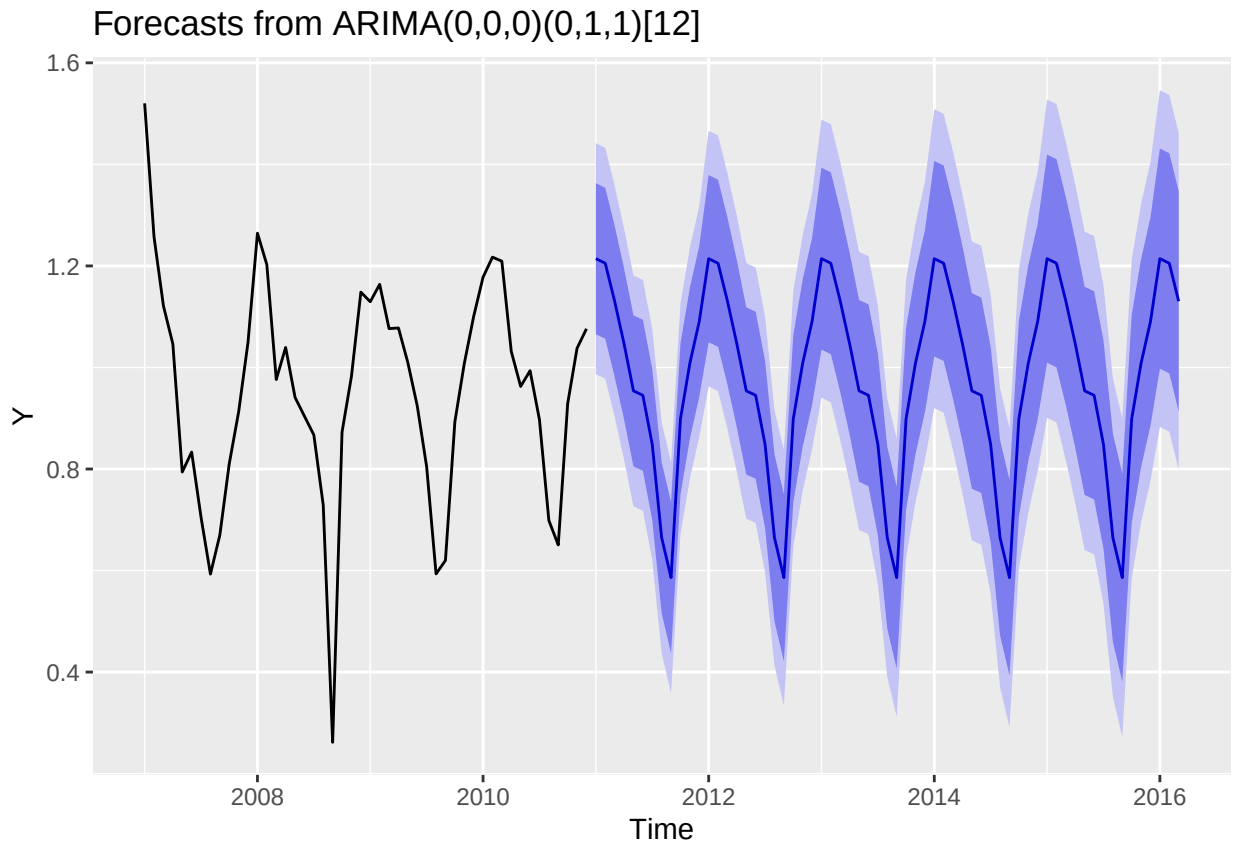

```
# Ljung-Box test to check for autocorrelation in residuals
Box.test(residuals(fit_arima), lag = 20, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  residuals(fit_arima)
## X-squared = 12.439, df = 20, p-value = 0.9001
```

```
### arima(2,1,3) SD^2 = 0.09049 SD=0.3008156
```

Daily Forecasting

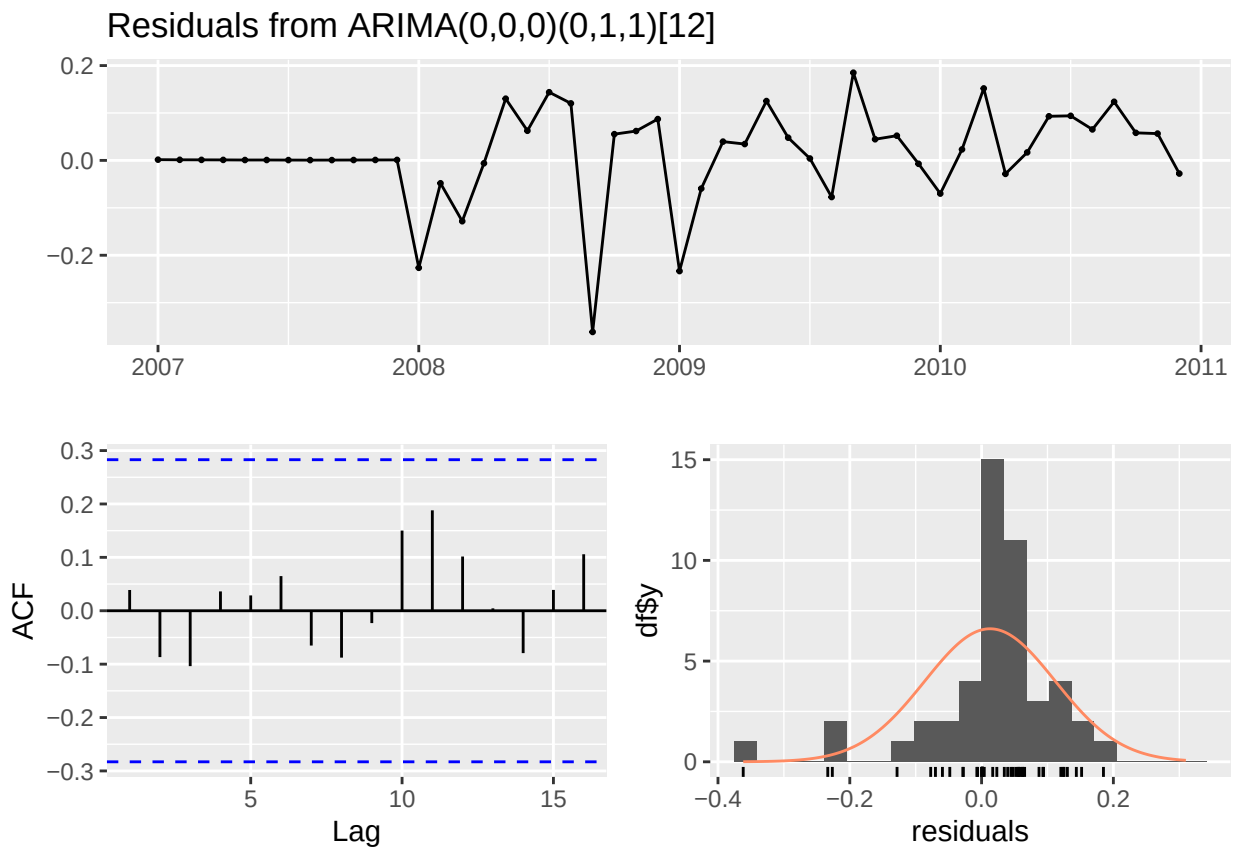
```
fcst <- forecast(fit_arima, h=63)
autoplot(fcst, include=1095)
```




```
# Fit ARIMA model
# fit_arima <- auto.arima(Y, stepwise = FALSE, approximation = FALSE, trace = TRUE)

# Print ARIMA model summary
print(summary(fit_arima))
```

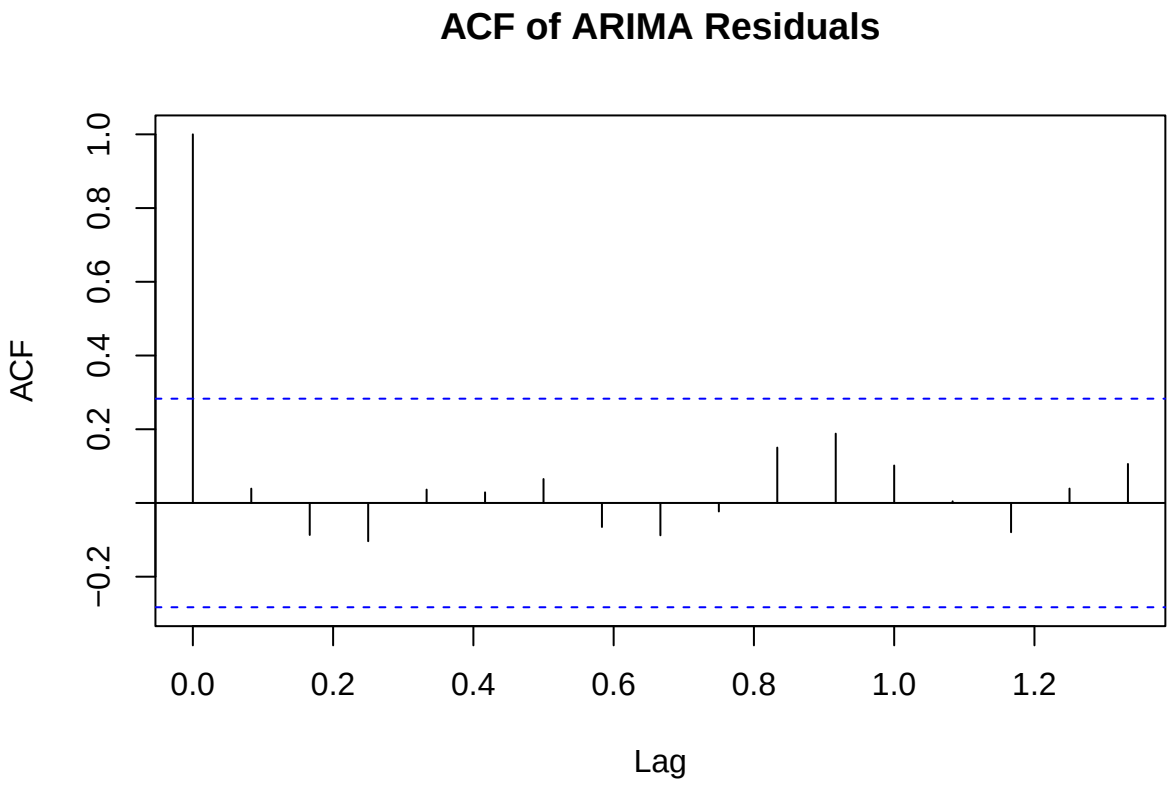
```
## Series: Y
## ARIMA(0,0,0)(0,1,1)[12]
##
## Coefficients:
##          sma1
##       -0.5243
## s.e.   0.2954
##
## sigma^2 = 0.0134:  log likelihood = 25.15
## AIC=-46.31   AICc=-45.94   BIC=-43.14
##
## Training set error measures:
##                       ME      RMSE       MAE        MPE     MAPE      MASE
## Training set 0.01278253 0.0988572 0.0658747 -0.1720544 9.186196 0.7010151
##                     ACF1
## Training set 0.03888812
```

```
# Check residual diagnostics
checkresiduals(fit_arima)
```

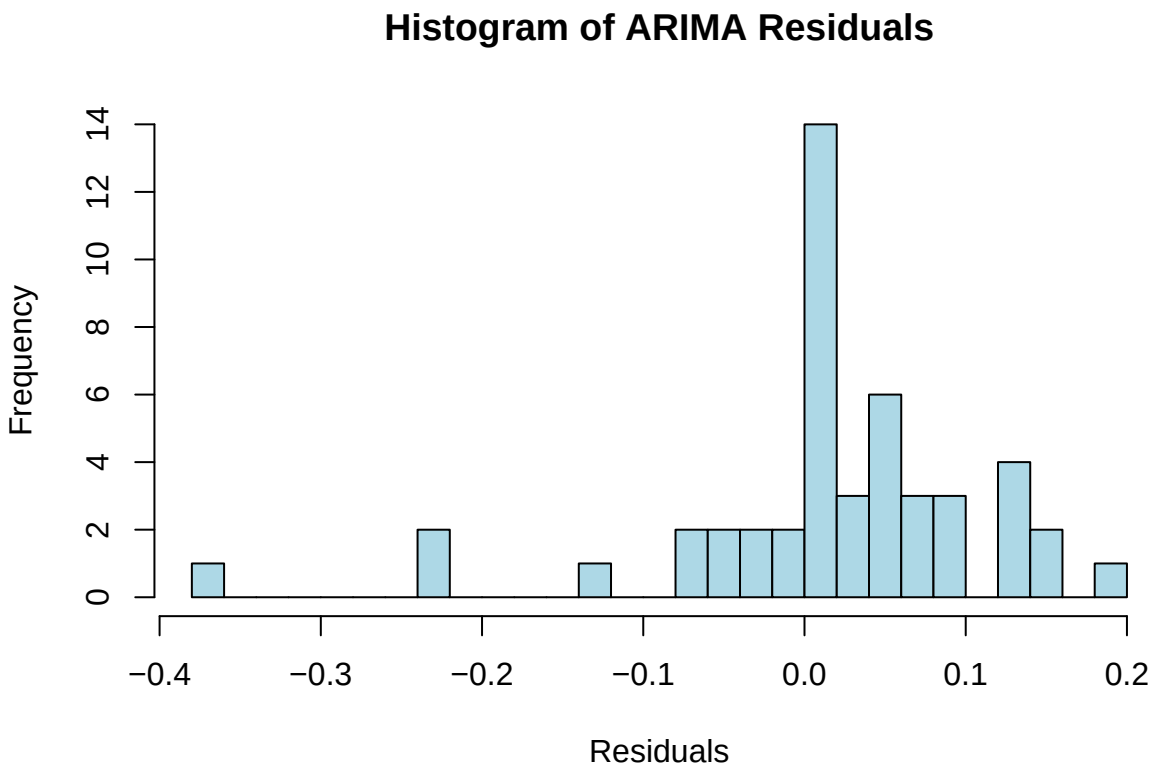



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,0,0)(0,1,1)[12]
## Q* = 3.5698, df = 9, p-value = 0.9374
##
## Model df: 1.   Total lags used: 10
```

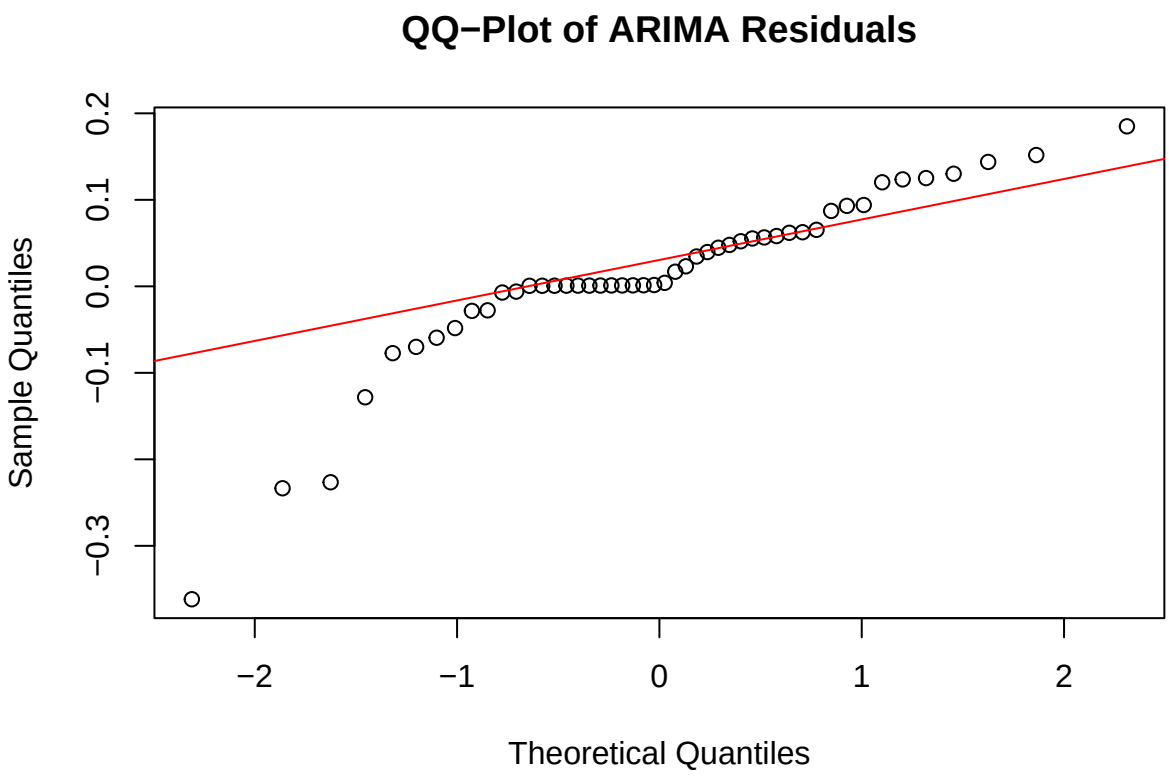
```r
# ACF of residuals
acf(residuals(fit_arima), main = "ACF of ARIMA Residuals")
```




```r
# Histogram of residuals
hist(residuals(fit_arima),
     main = "Histogram of ARIMA Residuals",
     xlab = "Residuals",
     breaks = 30,
     col = "lightblue",
     border = "black")
```




```r
# QQ-plot of residuals
qqnorm(residuals(fit_arima), main = "QQ-Plot of ARIMA Residuals")
qqline(residuals(fit_arima), col = "red")
```




```r
# Ljung-Box test for autocorrelation in residuals
box_test <- Box.test(residuals(fit_arima), lag = 20, type = "Ljung-Box")
print(box_test)
```

```
##
##  Box-Ljung test
##
## data:  residuals(fit_arima)
## X-squared = 12.439, df = 20, p-value = 0.9001
```

```r
# Shapiro-Wilk test for normality of residuals
shapiro_test <- shapiro.test(residuals(fit_arima))
print(shapiro_test)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(fit_arima)
## W = 0.86822, p-value = 6.874e-05
```

```r
# Jarque-Bera test for normality of residuals
if (!require(tseries)) install.packages("tseries")
library(tseries)
jarque_bera_test <- jarque.bera.test(residuals(fit_arima))
print(jarque_bera_test)
```

```
##
##  Jarque Bera Test
```

```
##
## data:  residuals(fit_arima)
## X-squared = 44.909, df = 2, p-value = 1.771e-10
```

```
# Runs Test for randomness of residuals
if (!require(randtests)) install.packages("randtests")
```

```
## Loading required package: randtests
```

```
##
## Attaching package: 'randtests'
```

```
## The following object is masked from 'package:tseries':
##
##     runs.test
```

```
library(randtests)
runs_test <- runs.test(residuals(fit_arima))
print(runs_test)
```

```
##
##  Runs Test
##
## data:  residuals(fit_arima)
## statistic = -3.5016, runs = 13, n1 = 24, n2 = 24, n = 48, p-value =
## 0.0004626
## alternative hypothesis: nonrandomness
```

```
# Calculate variance and standard deviation of residuals
residual_variance <- var(residuals(fit_arima))
residual_sd <- sqrt(residual_variance)
cat("Residual Variance:", residual_variance, "\n")
```

```
## Residual Variance: 0.009813815
```

```
cat("Residual Standard Deviation:", residual_sd, "\n")
```

```
## Residual Standard Deviation: 0.0990647
```

Interpretation of Diagnostic Tests: Box-Ljung Test:

X-squared = 12.886, df = 20, p-value = 0.8822 Interpretation: The high p-value (> 0.05) suggests that there is no significant autocorrelation in the residuals. This is a good sign, as it indicates that the ARIMA model has captured the time series' patterns effectively. Shapiro-Wilk Normality Test:

W = 0.87703, p-value = 0.0001235 Interpretation: The low p-value (< 0.05) indicates that the residuals deviate significantly from a normal distribution. This suggests that the residuals may not be normally distributed, which might affect the validity of statistical inferences based on the model. Jarque-Bera Test:

X-squared = 44.074, df = 2, p-value = 2.688e-10 Interpretation: The extremely low p-value (< 0.05) confirms that the residuals are not normally distributed. This aligns with the result of the Shapiro-Wilk test. The non-normality may be due to skewness, kurtosis, or other distributional characteristics. Runs Test:

Statistic = -3.5016, p-value = 0.0004626 Interpretation: The low p-value (< 0.05) indicates non-randomness in the residuals. This suggests that there may be patterns left in the residuals that the model has not captured, implying the ARIMA model may need improvement or adjustments. Residual Variance and Standard Deviation:

Residual Variance = 0.0091 Residual Standard Deviation = 0.0956 Interpretation: The residual variance and standard deviation indicate the average dispersion of the residuals. While these values are relatively low, the presence of non-randomness and non-normality suggests further refinement of the model may be required.