Renewable Electricity Production Forecasting

Time Series Assignment 3

Team Residuals Sri Karthick Selvam – S4038609 Mohammad Reyaz Mohammad Rafi – S4067516 Prajw

2025-06-21

Introduction

The global push toward sustainable energy has placed renewable electricity at the forefront of national energy policies. In this context, understanding the trends and patterns in renewable electricity production is crucial for ensuring energy security, planning future capacity, and meeting climate goals.

This project analyzes the time series of monthly renewable electricity production in **Australia** from **January 2010 to February 2025**. The dataset was obtained from the **International Energy Agency (IEA)** and includes net electricity production from:

- **Hydro**
- **Solar**
- **Wind**
- **Geothermal**
- **Other renewables**

The data was sourced from: **IEA Monthly Electricity Statistics**

Research Question

> **How has Australia's renewable electricity production evolved over the past decade, and what do forecasts for the next 10 months suggest about future trends?**

Australia is rapidly transitioning toward a low-carbon and sustainable energy system. Renewable energy plays a vital role in this shift. Analyzing historical production trends and forecasting future outputs is essential for:

- Monitoring **progress toward clean energy targets**
- Supporting **policy development** and **energy planning**
- Identifying **seasonal patterns**, **growth behavior**, and **potential bottlenecks**
- Guiding **investments** in energy infrastructure and innovation

Objectives

Examine the structure, summary statistics, and distribution of the renewable electricity production data. - **Visualize patterns and trends**: Identify key features in the time series, such as long-term trend, seasonality, and variability. - **Evaluate time series properties**: Check for **stationarity** and assess

normality using statistical tests and visual tools (e.g., ACF, PACF, QQ plots, histograms). - **Apply transformations if needed**: Use techniques like Box-Cox or log transformation to stabilize variance and make the series more suitable for modeling. - **Explore and compare different models**: Fit and assess models such as linear, quadratic, cubic, seasonal, and ARIMA/SARIMA to identify best fit. - **Analyze model diagnostics**: Perform residual analysis, AIC/BIC comparisons, and goodness-of-fit evaluations to select the most appropriate model. - **Forecast future values**: Generate short-term forecasts to anticipate upcoming renewable electricity production and support pla nning and decision-making.

```r
library(forecast)
library(tseries)
library(TSA)
library(lmtest)
library(dplyr)
library(lubridate)
library(ggplot2)
```

Explore the dataset - Renewable electricity production data for Australia

Before performing any statistical modeling, it's important to begin by thoroughly understanding the dataset. This step ensures we know what the data represents, how it's structured, and whether it's ready for analysis.

In this study, we are working with monthly renewable electricity production data for Australia, covering the period from January 2010 to February 2025. The dataset includes net production from key renewable sources such as hydro, wind, solar, geothermal, and others.

We start by examining basic properties of the dataset, including: - The time range and frequency of observations - Summary statistics (e.g., mean, median, min, max) - Data types and completeness (checking for missing values or inconsistencies)

This initial exploration provides a foundation for the rest of the analysis, helping us decide which transformations or cleaning steps are needed before modeling.

```r
# Read the data
data <- read.csv("MES_0225.csv", skip = 8, stringsAsFactors = FALSE)

# Filter for Country = Australia, Balance = Net Electricity Production, Product = Total Renewables
filtered_data <- data %>%
  filter(
    Country == "Australia",
    Balance == "Net Electricity Production",
    Product == "Total Renewables (Hydro, Geo, Solar, Wind, Other)"
  ) %>%
  mutate(date = my(Time)) %>%    # Converts "Jan-10" to Date
  arrange(date)

# Extract start year and month for time series
start_year <- year(min(filtered_data$date))
start_month <- month(min(filtered_data$date))

# Create monthly time series
ts_data <- ts(filtered_data$Value, frequency = 12, start = c(start_year, start_month))
```

```r
# Plot the time series
# Convert the Time column to date format
filtered_data$date <- my(filtered_data$Time)

# Plot using ggplot2
ggplot(filtered_data, aes(x = date, y = Value)) +
  geom_line(color = "darkgreen", size = 1.2) +
  geom_point(color = "red", size = 2) +
  labs(
    title = "Renewable Electricity Production in Australia (GWh)",
    subtitle = "From Hydro, Geo, Solar, Wind, and Other Renewables",
    x = "Year",
    y = "Electricity Production (GWh)"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    plot.title = element_text(face = "bold", size = 16),
    plot.subtitle = element_text(size = 13),
    axis.text.x = element_text(angle = 45, hjust = 1)
  )
```

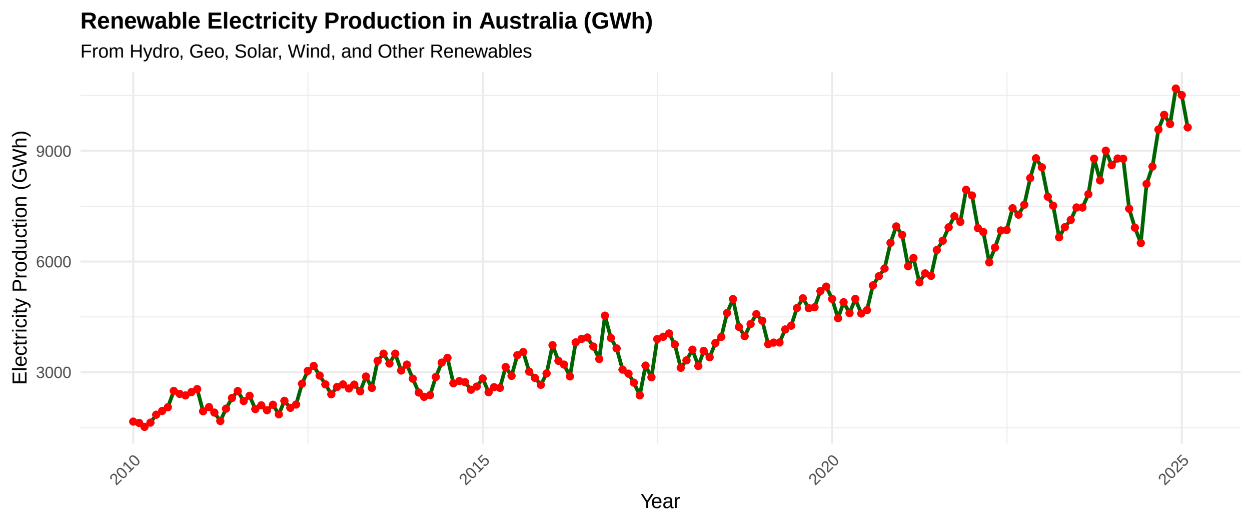


Figure 1: Figure 1: Time Series of Monthly Renewable Electricity Production in Australia

This Figure1 time series plot shows how much electricity Australia has generated from renewable sources every month from 2010 to early 2025. The data includes production from sources like hydro, solar, wind, geothermal, and other renewables.

- The **green line** represents the monthly renewable electricity production over time.
- The **red points** show the actual data values for each month. These points help identify **seasonal patterns**—which months typically have higher or lower production.
- The x-axis shows the years, and the y-axis shows the amount of electricity produced in **gigawatt-hours (GWh)**. Valid Points:

Key understanding from the Time Series Figure 1

- *Trend* : A **strong upward trend** is clearly observed throughout the series. From around 2010 to early 2025, renewable electricity production in Australia has consistently increased, with especially rapid growth after 2018.
- *Seasonality* : A **clear seasonal pattern** is visible. Each year shows regular rises and falls in production, likely due to weather or climate-related factors affecting solar, wind, and hydro output.
- *Changing Variance* : There is **slightly changing variance over time**. While not extreme, the variability in monthly production becomes more noticeable in recent years compared to the earlier part of the series.
- *Behaviour (AR or MA)* : Because of the **clear seasonality**, it's hard to directly interpret autoregressive (AR) or moving average (MA) behavior just from the plot. Further analysis with ACF and PACF plots would be needed to explore this.
- *Change Point* : There is **no sharp, obvious change point**, but there is a noticeable **acceleration in the upward trend starting around 2018**, suggesting possible policy or infrastructure shifts in renewable energy during that time.
- Overall, this time series is **non-stationary** due to the presence of trend and seasonality, and mild changes in variance.

```r
df_ts <- data.frame(
  date = as.Date(as.yearmon(time(ts_data))),
  value = as.numeric(ts_data),
  season = factor(cycle(ts_data), labels = month.abb)  # e.g., Jan, Feb, ...
)

# Step 2: Plot using ggplot2
ggplot(df_ts, aes(x = date, y = value)) +
  geom_line(color = "red", size = 1.2) +                # Line plot
  geom_point(aes(shape = season), color = "darkgreen", size = 2.5) +  # Points with seasonal shape
  scale_shape_manual(values = 1:12) +                      # Different shapes for months
  labs(
    title = "Time Series plot of Total Electricity Production from Renewable Sources in Australia",
    x = "Year",
    y = "Electricity Production (GWh)",
    shape = "Month"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    plot.title = element_text(face = "bold", size = 16, hjust = 0.5),
    axis.text.x = element_text(angle = 45, hjust = 1)
  )
```

Figure 2 shows the total electricity production from renewable sources in Australia, plotted month by month from 2010 to early 2025.

Each point on the graph represents one month, and different **shapes** are used to represent different months (e.g., circles for January, triangles for February, etc.). The red line connects these points to show how production changes over time.

This plot helps us see **two key things at once**:

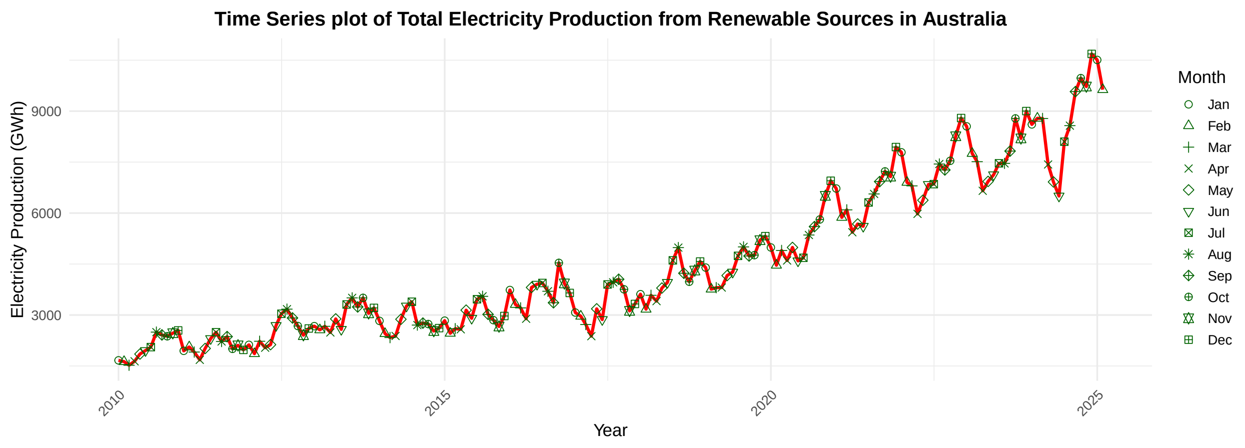


Figure 2: Figure 2: Time Series with Monthly Seasonal Indicators

1. **Overall trend** – The red line shows that renewable electricity production has been increasing steadily over the years, especially after 2018. This tells us that Australia is producing more clean energy now than in the past.

2. **Seasonal pattern** – Because the same shapes (months) repeat along the timeline, we can see that certain months tend to have higher or lower values regularly. For example, some months like summer or winter might consistently show peaks or dips, depending on how much sun, wind, or water is available for energy production.

This graph makes it easier to spot these seasonal cycles, as well as to notice how monthly behavior changes over time. Even though the shapes repeat every year, their position on the y-axis gradually moves upward—showing both **seasonality and long-term growth** at the same time.

In summary, this graph clearly illustrates that Australia's renewable energy output is not only growing, but also follows a consistent yearly pattern influenced by the seasons.

Understanding the Data

Before building any statistical models or forecasting future trends, it's important to first explore and understand the data. This step helps us identify the basic structure, patterns, and characteristics of the time series, and check whether certain assumptions hold (like normality or stationarity).

In this section, we'll use a combination of visual and numerical methods to explore the renewable electricity production data. Here's what we'll do:

- **Summary statistics**: Get a quick overview of the data using measures like mean, median, minimum, maximum, and standard deviation. This helps us understand the central tendency and spread of the production values.
- **Histogram**: Visualize the distribution of the production values to see whether they are symmetric, skewed, or follow any particular shape (like normal or bell-shaped).
- **Q-Q plot**: Compare the distribution of our data to a normal distribution. This helps us visually assess whether the data is approximately normally distributed.

- **Normality test**: Use a formal test (like the Shapiro-Wilk test) to check if the data follows a normal distribution, which can impact modeling choices.
- **ACF and PACF plots**: Analyze autocorrelation and partial autocorrelation. These plots help us understand how current values are related to past values and whether there are signs of autoregressive (AR) or moving average (MA) behavior.
- **Lag plots**: Use scatter plots of the data against its lagged values to visually check for linear relationships between observations across time lags.

This overall analysis will give us a deeper understanding of the time series structure, which is essential for choosing appropriate models later on (like decomposition or ARIMA).

Summary statistics

```
summary(ts_data)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1522    2673    3746    4469    6258   10685
```

```
str(ts_data)
```

```
##  Time-Series [1:182] from 2010 to 2025: 1666 1629 1522 1638 1852 ...
```

```
sum(is.na(ts_data))
```

```
## [1] 0
```

The table above shows basic summary statistics of monthly renewable electricity production in Australia (in GWh):

- **Minimum**: 1,522 GWh – the lowest monthly production observed.
- **1st Quartile (Q1)**: 2,673 GWh – 25% of the data falls below this value.
- **Median**: 3,746 GWh – the middle value; half of the months produced less, and half produced more.
- **Mean**: 4,469 GWh – the average monthly production.
- **3rd Quartile (Q3)**: 6,258 GWh – 75% of the data falls below this value.
- **Maximum**: 10,685 GWh – the highest monthly production recorded.

No missing values found on the dataset

This shows that production levels have a wide range and have increased significantly over time. The mean being higher than the median also suggests a **right-skewed distribution**, where some very high values are pulling the average up.

Histogram of Monthly Renewable Electricity Production

```r
ggplot(filtered_data, aes(x = Value)) +
  geom_histogram(binwidth = 500, fill = "#2C77B8", color = "white", alpha = 0.8) +
  labs(
    title = "Histogram of Monthly Renewable Electricity Production in Australia",
    x = "Electricity Production (GWh)",
    y = "Frequency"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    plot.title = element_text(face = "bold", hjust = 0.5)
  )
```

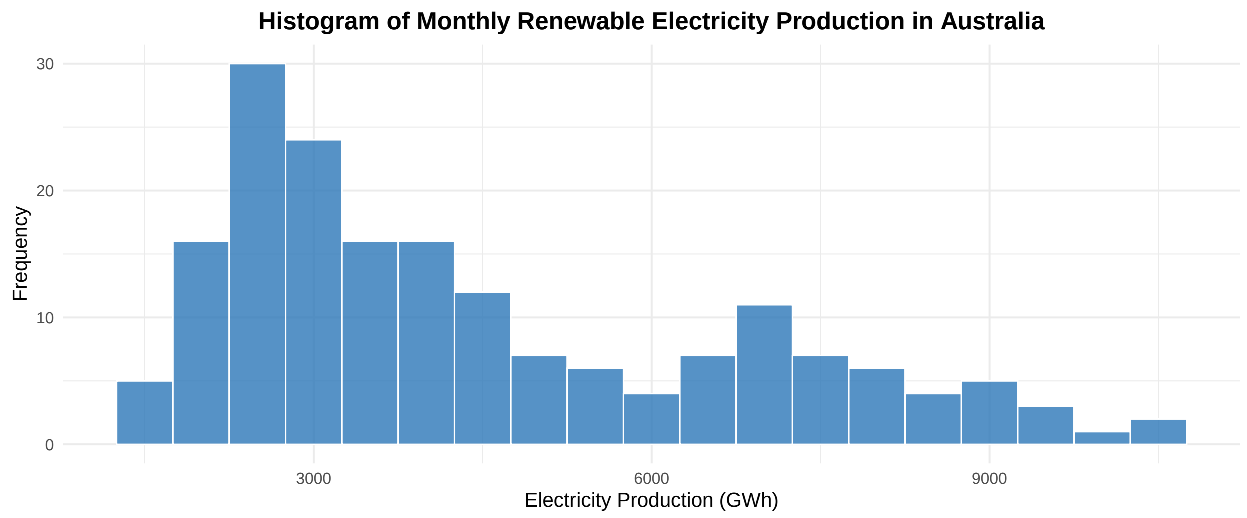


Figure 3: Figure 3: Histogram of Monthly Renewable Electricity Production

Fig 3 shows the distribution of monthly renewable electricity production in Australia (measured in GWh). Each bar represents the number of months where production fell within a specific range.

Interpretation:

- The distribution is **right-skewed**, meaning most of the production values are on the lower end (between 2,000 and 5,000 GWh), with fewer months producing very high amounts.
- The most common monthly production level is around **2,500 to 3,000 GWh**, as shown by the tallest bar.
- There are relatively few months where production exceeded **8,000 GWh**, which aligns with the high-end values from the summary statistics.
- This skewed distribution suggests that although average production is increasing, **lower values were more common in earlier years**, while higher production values became more frequent in recent years.

This histogram confirms that the data is **not normally distributed**, and it helps visually support the findings from the summary statistics section.

Augmented Dickey-Fuller Test (ADF)

```r
adf.test(ts_data)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ts_data
## Dickey-Fuller = -2.8026, Lag order = 5, p-value = 0.2407
## alternative hypothesis: stationary
```

To test whether the time series is stationary, we applied the Augmented Dickey-Fuller (ADF) test. Stationarity is an important assumption in many time series models, especially ARIMA or SARIMA.

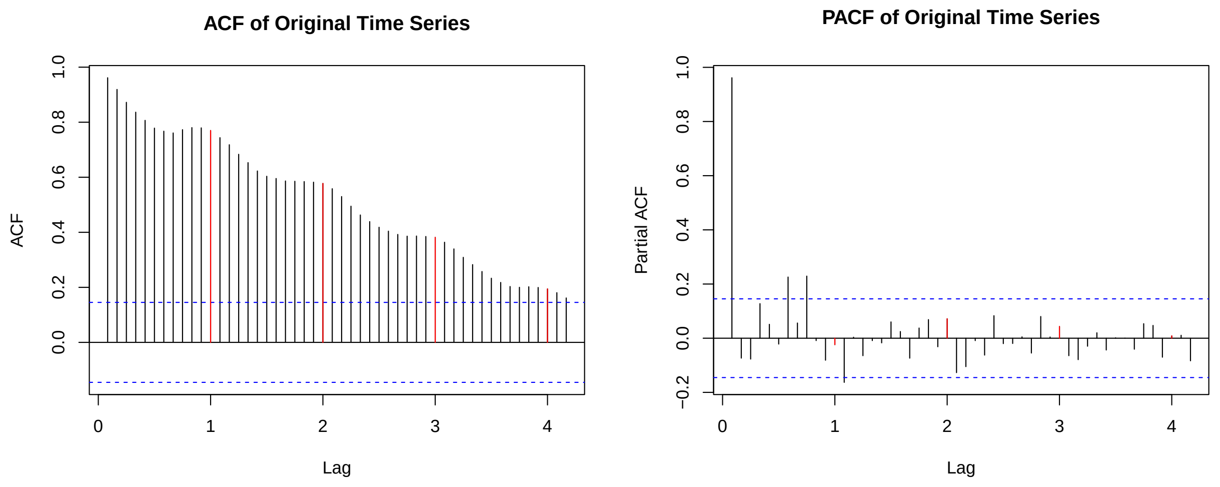
Interpretation: Since the **p-value is 0.2407**, which is **greater than 0.05**, we **fail to reject the null hypothesis**. This means there is **not enough evidence to conclude that the series is stationary**.

This result aligns with what we saw in the earlier plots—there is a strong trend and clear seasonality in the data, both of which are characteristics of a **non-stationary** time series.

To make the series stationary, we may need to apply transformations such as **differencing** or **seasonal adjustment**, especially before fitting models like ARIMA.

ACF and PACF Plots of the Original Time Series

```r
par(mfrow = c(1, 2))
seasonal_acf(ts_data, lag.max=50, main = "ACF of Original Time Series")
seasonal_pacf(ts_data, lag.max=50, main = "PACF of Original Time Series")
```



The plots above in figure 4 show the **seasonal autocorrelation structure** of the original time series. These are useful for identifying repeating patterns at regular time intervals—particularly yearly seasonality in monthly data.

ACF (left plot):

- The ACF shows a **slow decay**, typical of a **non-stationary** time series.
- The **red spikes** appear at regular intervals (around lag 12, 24, etc.), clearly indicating the presence of **seasonality**.
- High autocorrelation at these seasonal lags suggests that the electricity production pattern **repeats every year**.

PACF (right plot):

- The PACF shows a **strong spike at lag 1**, and smaller ones at later lags.
- The seasonal lags are also marked in red, though their influence is weaker than in the ACF.
- This pattern implies that **current values are influenced by the previous month and possibly the same month in previous years**.

  The combination of **non-stationarity, seasonal spikes**, and a **trend** suggests that we should consider using **seasonal differencing** and a **seasonal ARIMA model** for forecasting.

These plots provide key guidance for selecting seasonal ARIMA parameters, especially the seasonal order `(P, D, Q)[12]`.

Lag-1 Plot

```r
plot(ts_data[-length(ts_data)], ts_data[-1],
     xlab = "Lag 1", ylab = "Current",
     main = "Lag-1 Plot of ts_data")

abline(lm(ts_data[-1] ~ ts_data[-length(ts_data)]), col = "black", lwd = 2)
```

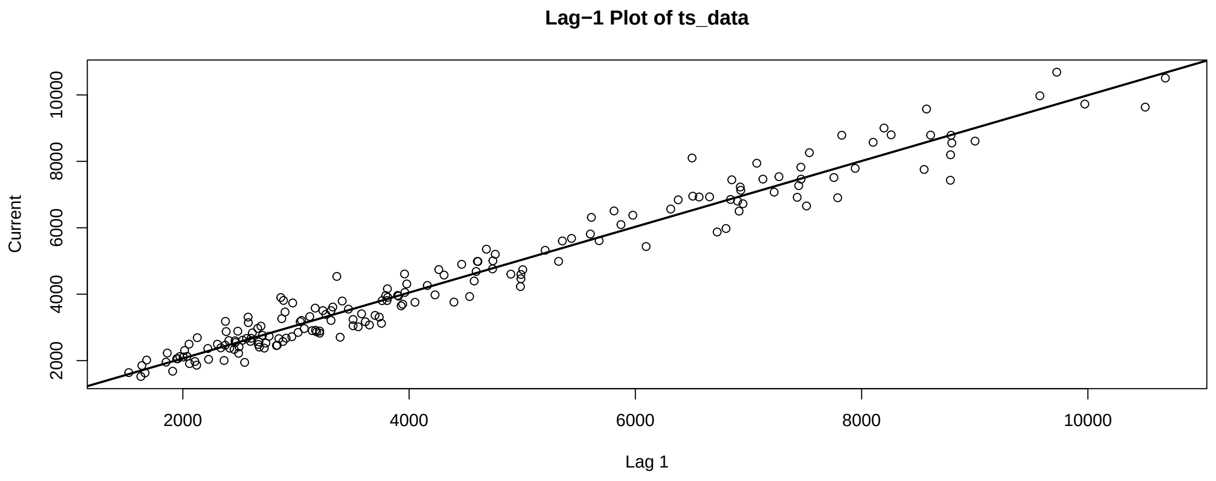


Figure 4: Figure 5: Lag-1 Scatter Plot of Monthly Renewable Production

The plot above in figure 5 shows the **Lag-1 relationship**, where each point represents a month. The x-axis shows the electricity production from the **previous month**, and the y-axis shows the production from the **current month**.

We added a black regression line to help visualize the overall trend between the two.

- The points form a **tight upward-sloping cloud**, indicating a **strong positive linear relationship** between consecutive months.
- This means that if production was high last month, it's likely to be high this month too, and vice-versa.
- The clear linear pattern confirms that the time series has **strong autocorrelation at lag 1**, which aligns with what we observed in the ACF and PACF plots.

This type of behavior is common in time series with a trend or seasonal component and supports the idea that past values can be useful for forecasting future ones.

Lag-2 Plot

```r
# Lag-2 plot: compare each value with its value two months earlier
plot(ts_data[-c(length(ts_data)-1, length(ts_data))], ts_data[-c(1, 2)],
     xlab = "Lag 2", ylab = "Current",
     main = "Lag-2 Plot of ts_data")

# Add regression line
abline(lm(ts_data[-c(1, 2)] ~ ts_data[-c(length(ts_data)-1, length(ts_data))]), col = "black", lwd = 2)
```

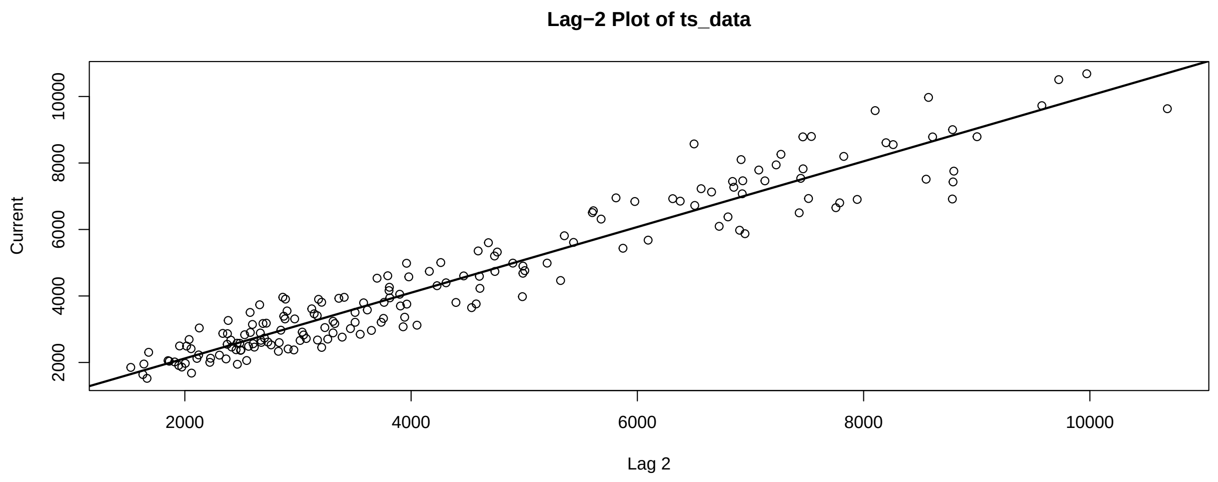


Figure 5: Figure 6: Lag-2 Scatter Plot of Monthly Renewable Electricity Production

The plot above in figure 6 shows the relationship between each month's electricity production and the production **two months earlier**. Each point represents a pair of observations: one from time t and one from $t-2$.

- The points form a **tight upward-sloping pattern**, indicating a strong positive relationship between the current value and the value two months ago.
- A regression line is added, showing a clear linear trend.
- This means that electricity production in a given month is **closely related to values from two months earlier**, suggesting persistent patterns or momentum in the time series.

Lag-1 vs Lag-2 Comparison

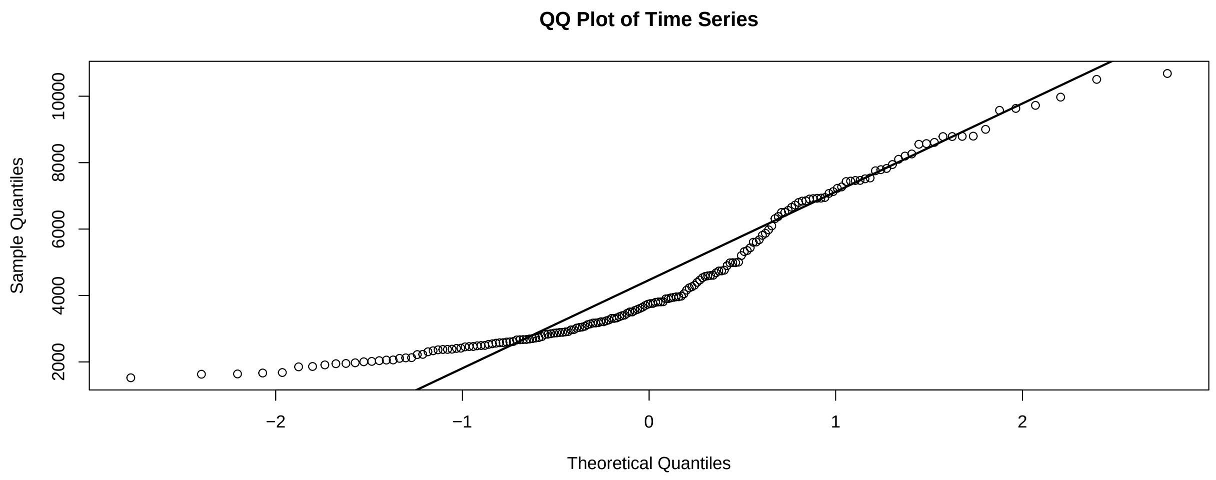
Aspect	Lag-1 Plot	Lag-2 Plot
Relationship	Very strong linear relationship	Still strong, but slightly more spread out
Tightness of points	Points are more closely clustered around the line	Slightly wider spread compared to lag-1
Interpretation	Electricity production is highly dependent on the previous month	There is also a meaningful connection with production two months back
Conclusion	Strong short-term autocorrelation	Evidence of medium-term autocorrelation

Both plots support the idea that the time series is **autocorrelated**—current values are influenced by previous ones. This is important for model selection, as it suggests that **AR terms** may be useful in future modeling.

Normality Check

To check whether the time series values are normally distributed, we used two methods: a **Q-Q plot** and the **Shapiro-Wilk test**.

```r
qqnorm(ts_data, main="QQ Plot of Time Series")
qqline(ts_data, col="black", lwd=2)
```



#### Q-Q Plot (Quantile-Quantile Plot):

- The Q-Q plot compares the quantiles of our data with the quantiles of a normal distribution.
- In the plot above, the points deviate noticeably from the straight diagonal line, especially at the **tails** (both ends).
- This indicates that the data is **not normally distributed**, as a normal distribution would result in points falling roughly along the line.

```r
#### Shapiro-Wilk Test:
shapiro.test(ts_data)
```

```
##
```

```
##  Shapiro-Wilk normality test
##
## data:  ts_data
## W = 0.89918, p-value = 8.732e-10
```

The **p-value is far below 0.05**, which means we **reject the null hypothesis** of normality. This confirms that the data **does not follow a normal distribution**.

Both the Q-Q plot and the Shapiro-Wilk test strongly suggest that the time series values are **not normally distributed**. This is not surprising, given the data has a clear trend, seasonality, and is right-skewed (as seen in the histogram).

While normality is not always required for time series modeling (especially for ARIMA), it's good to be aware of the data's distribution for proper model choice and interpretation.

The exploratory analysis provided valuable insights into the structure and behavior of Australia's monthly renewable electricity production over time.

We observed a **strong upward trend**, especially after 2018, reflecting the country's increasing investment in clean energy. The data also displayed a **clear and consistent seasonal pattern**, with certain months showing higher or lower production levels year after year—likely due to environmental and weather-related factors affecting solar, wind, and hydro generation.

The **histogram and Q-Q plot** showed that the distribution of production values is **right-skewed**, meaning lower production months were more common in earlier years, with a growing number of high-output months appearing more recently. This was supported by the **Shapiro-Wilk test**, which confirmed that the data is **not normally distributed**.

In terms of time series properties, the **Augmented Dickey-Fuller test** indicated that the series is **non-stationary**, which was further reinforced by the ACF and PACF plots. Additionally, **lag plots** confirmed the presence of autocorrelation, especially at short lags, which is an important factor to consider in model building.

While normality is not strictly required for time series modeling - especially for models like ARIMA - it's still important to understand the data distribution, trends, and seasonality to guide effective model selection.

Having uncovered these patterns and statistical properties, we are now in a strong position to move into the **model selection phase**. The next step is to explore a variety of models that can appropriately capture the trend, seasonality, and autocorrelation observed in the data, and identify the one that provides the best fit and forecasting potential.

Model Selection

After performing an in-depth exploration of the time series data, we identified several important characteristics:

- A **strong long-term upward trend**, especially visible after 2018.
- A **clear seasonal pattern** repeating annually.
- Signs of **non-linearity** in the growth pattern.
- **Non-stationarity**, confirmed by the ADF test and ACF/PACF plots.
- The data is also **not normally distributed**.

These observations guide us toward selecting suitable models that can account for the data's complexity. In this section, we fit and compare a range of models to identify which one best captures the structure of the time series.

Models Considered

- **Linear model** – captures basic upward trend but not curvature or seasonality.
- **Quadratic and cubic models** – handle nonlinear growth better.
- **Cyclic model** – captures seasonal variation using the month as a categorical variable.
- **Seasonal model (trend + month)** – combines both the trend and seasonal components explicitly.
- **SARIMA model** – an extension of ARIMA that also includes seasonal differencing and seasonal AR/MA terms, making it well-suited for series with strong seasonality like ours.

We will evaluate each model both visually (using plots of fitted values) and statistically (using criteria like AIC, BIC, and residual diagnostics). This will help us determine which model provides the best balance of **fit**, **interpretability**, and **forecasting ability**.

```r
# Create a data frame with time index and month as a factor
df_model <- data.frame(
  time = 1:length(ts_data),
  value = as.numeric(ts_data),
  month = factor(cycle(ts_data))
)

# Fit models
linear_model <- lm(value ~ time, data = df_model)
quadratic_model <- lm(value ~ time + I(time^2), data = df_model)
cubic_model <- lm(value ~ time + I(time^2) + I(time^3), data = df_model)
cyclic_model <- lm(value ~ month, data = df_model)
seasonal_model <- lm(value ~ time + month, data = df_model)

# Add fitted values
df_model$linear_fit <- predict(linear_model)
df_model$quad_fit <- predict(quadratic_model)
df_model$cubic_fit <- predict(cubic_model)
df_model$cyclic_fit <- predict(cyclic_model)
df_model$seasonal_fit <- predict(seasonal_model)

# Common theme for consistency
custom_theme <- theme_minimal(base_size = 14) +
  theme(
    plot.title = element_text(face = "bold", hjust = 0.5, size = 16),
    axis.title = element_text(size = 14),
    axis.text = element_text(size = 12)
  )
```

```r
summary(linear_model)
```

```
##
## Call:
## lm(formula = value ~ time, data = df_model)
##
```

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1954.68  -658.40     1.53   598.61  2725.94
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  861.663    129.636   6.647 3.45e-10 ***
## time          39.429      1.229  32.091  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 870.8 on 180 degrees of freedom
## Multiple R-squared:  0.8512, Adjusted R-squared:  0.8504
## F-statistic:  1030 on 1 and 180 DF,  p-value: < 2.2e-16
```

```r
# Plot Linear Fit
ggplot(df_model, aes(x = time, y = value)) +
  geom_line(color = "gray30", size = 1, alpha = 0.6) +
  geom_line(aes(y = linear_fit), color = "#0072B2", size = 1.2, linetype = "dashed") +
  labs(title = "Linear Model Fit",
       x = "Time Index",
       y = "Electricity Production (GWh)") +
  custom_theme
```

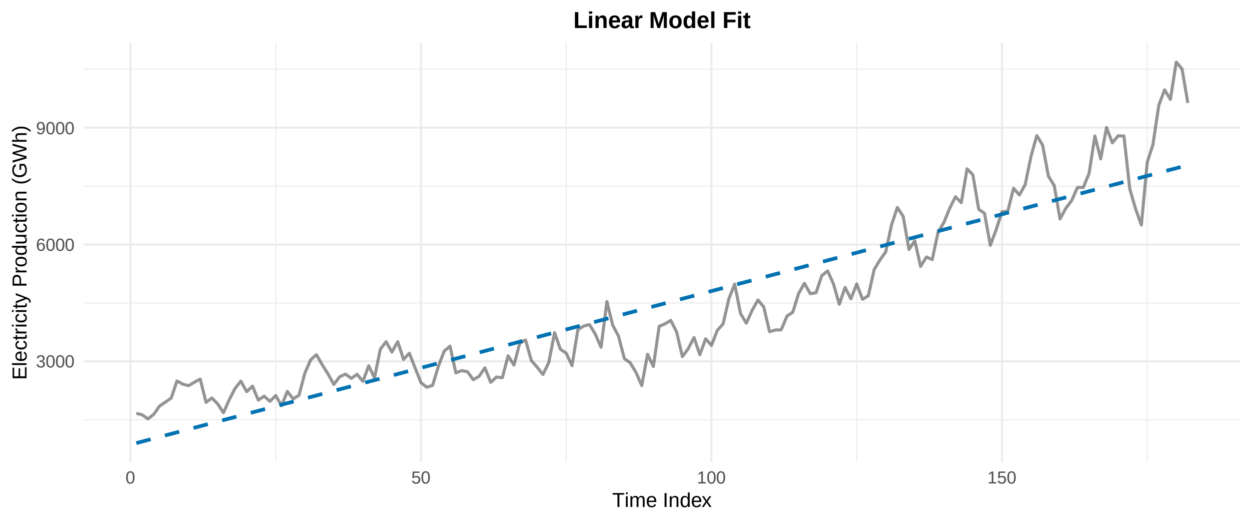


Figure 6: Figure 8: Linear Model Fit (GWh)

```r
residual.analysis(model = linear_model)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.98765, p-value = 0.1127
```

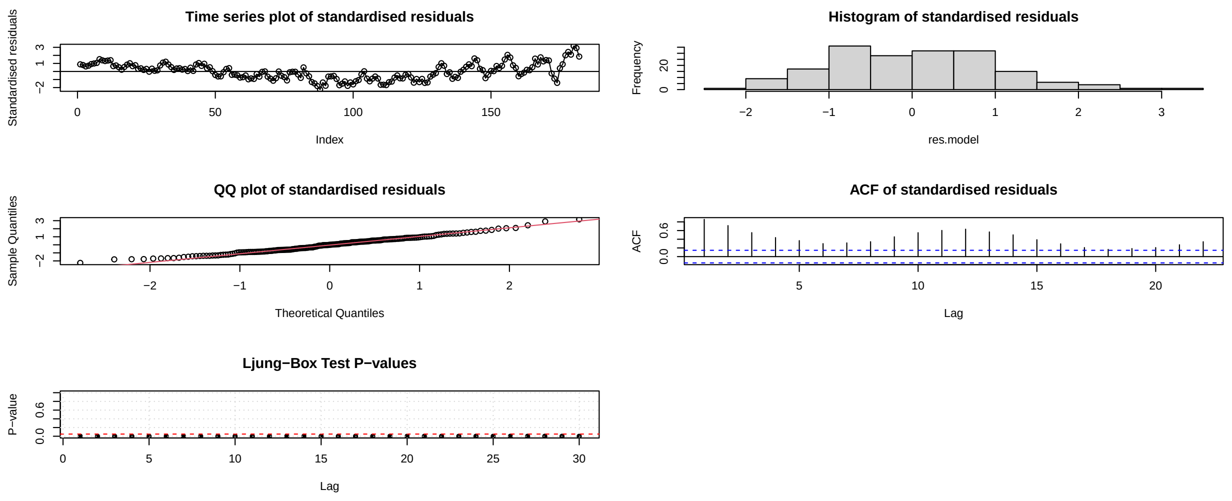


Figure 7: Figure 8: Linear Model Fit (GWh)

Observations: Linear Model

A linear regression model was fitted to capture the relationship between time and renewable electricity production in Australia. This model assumes that production increases at a constant rate over time.

The summary output shows a highly significant time coefficient, indicating a strong upward trend in renewable electricity generation. The model explains a large portion of the variability, as reflected by an R-squared value of approximately 0.85.

- **Captures Overall Trend**: The model successfully identifies a clear long-term upward trend in electricity production.
- **Interpretability**: The relationship between time and production is easy to interpret.
- **Good Starting Point**: Linear models are useful as a baseline to understand data behavior before applying more complex models.
- **Misses Nonlinearity**: The residual plot and fitted line reveal curvature, suggesting the relationship is not purely linear.
- **Ignores Seasonality**: The model does not account for recurring monthly or seasonal fluctuations.
- **Residual Structure**: Residual diagnostics show non-random patterns and autocorrelation, indicating that key dynamics (like seasonality or lagged effects) are not captured.
- **Assumption Violations**: Although the residuals appear roughly normal (as indicated by the Shapiro-Wilk test), the presence of autocorrelation and non-stationarity violates linear model assumptions.

Residual Diagnostics

Residual analysis reveals structured and correlated residuals, particularly in the ACF plot. The Ljung-Box test further suggests that residuals are not white noise. These patterns imply that important temporal dependencies remain unmodeled.

Given these limitations, the linear model is insufficient for accurate forecasting. To better capture the curvature and seasonality seen in the data, we next explore **Quadratic and Cubic Polynomial Models**: To capture nonlinear trends. - **Seasonal and Cyclical Models**: To model repeating monthly patterns. - **ARIMA and SARIMA Models**: To address autocorrelation, seasonality, and non-stationarity rigorously.

```r
summary(quadratic_model)
```

```
##
## Call:
## lm(formula = value ~ time + I(time^2), data = df_model)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2271.79  -362.42    -0.14   304.52  1421.22
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2319.01289  132.00941   17.57   <2e-16 ***
## time          -8.09313    3.33070   -2.43   0.0161 *
## I(time^2)      0.25968    0.01763   14.73   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 587.1 on 179 degrees of freedom
## Multiple R-squared:  0.9327, Adjusted R-squared:  0.932
## F-statistic:  1241 on 2 and 179 DF,  p-value: < 2.2e-16
```

```r
# Plot Quadratic Fit
ggplot(df_model, aes(x = time, y = value)) +
  geom_line(color = "gray30", size = 1, alpha = 0.6) +
  geom_line(aes(y = quad_fit), color = "#D55E00", size = 1.2, linetype = "dotdash") +
  labs(title = "Quadratic Model Fit",
       x = "Time Index",
       y = "Electricity Production (GWh)") +
  custom_theme
```

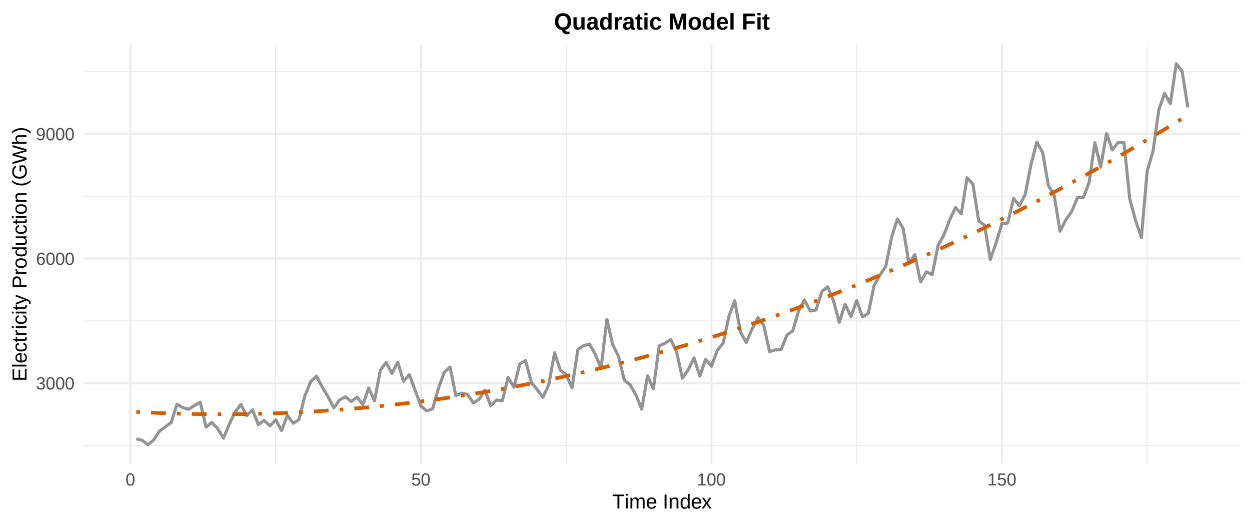
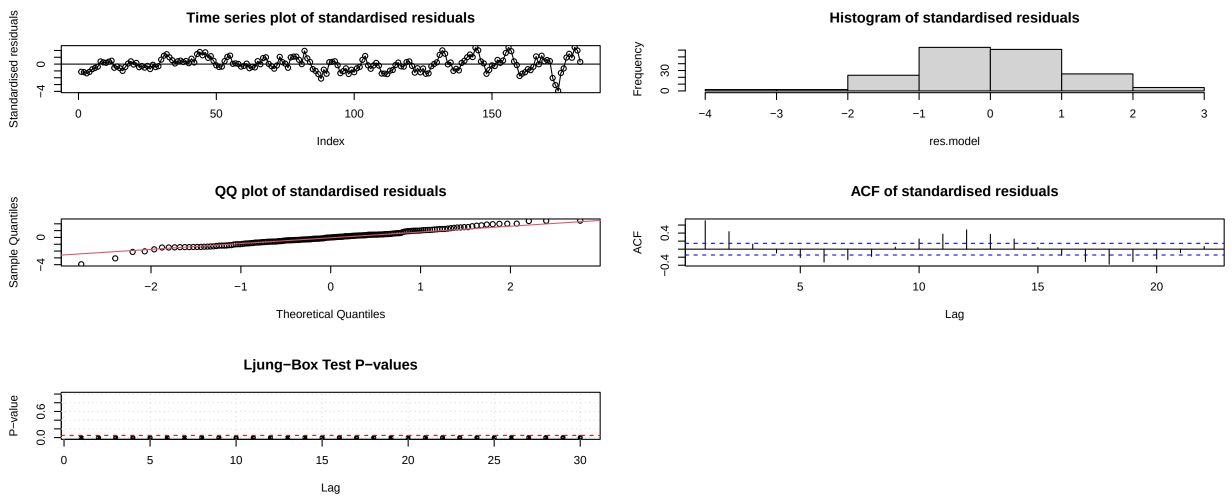


Figure 8: Figure 9: Quadratic Model Fit

```r
residual.analysis(model = quadratic_model)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.98462, p-value = 0.04326
```



### Observations: Quadratic Model

The quadratic model incorporates both linear (`time`) and nonlinear (`time`^{2}) terms, allowing it to better capture the curved growth trend observed in Australia's renewable electricity production.

- **Adjusted $R^2 = 0.932$**, indicating a significant improvement in fit compared to the linear model.
- Both the `time` and `time`^{2} terms are statistically significant ($p < 0.05$), validating the inclusion of a nonlinear trend.
- The **Shapiro-Wilk test** p-value (0.04326) is close to 0.05, suggesting mild deviation from normality but better than the linear model.
- Effectively captures **nonlinear trend** in the data.
- Offers a better overall fit compared to the linear model.
- **Seasonality is not captured**, leaving structured residuals.
- **Autocorrelation** persists, violating the independence assumption.

Residual Diagnostics

- The **Q-Q plot** shows improved alignment with the normal distribution, though some deviation remains.
- **Histogram of residuals** appears more symmetric and centered.
- **ACF plot** and **Ljung-Box test** indicate the presence of **autocorrelation**, which the model does not address.
- Residuals over time show **non-random patterns**, hinting at missed seasonal structure.

While the quadratic model provides a better trend fit, it still fails to account for seasonality and time dependence. Therefore, more advanced models like **cubic**, **seasonal**, or **ARIMA/SARIMA** are needed to fully capture the behavior of the time series.

```r
# Plot Cubic Fit
summary(cubic_model)
```

```
##
## Call:
## lm(formula = value ~ time + I(time^2) + I(time^3), data = df_model)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2440.87  -365.21   -35.31   328.91  1534.28
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.972e+03  1.739e+02  11.336  < 2e-16 ***
## time         1.437e+01  8.209e+00   1.751  0.08167 .
## I(time^2)   -4.640e-02  1.041e-01  -0.446  0.65629
## I(time^3)    1.115e-03  3.739e-04   2.982  0.00326 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 574.6 on 178 degrees of freedom
## Multiple R-squared:  0.9359, Adjusted R-squared:  0.9349
## F-statistic:   867 on 3 and 178 DF,  p-value: < 2.2e-16
```

```r
ggplot(df_model, aes(x = time, y = value)) +
  geom_line(color = "gray30", size = 1, alpha = 0.6) +
  geom_line(aes(y = cubic_fit), color = "#CC79A7", size = 1.2, linetype = "longdash") +
  labs(title = "Cubic Model Fit",
       x = "Time Index",
       y = "Electricity Production (GWh)") +
  custom_theme
```

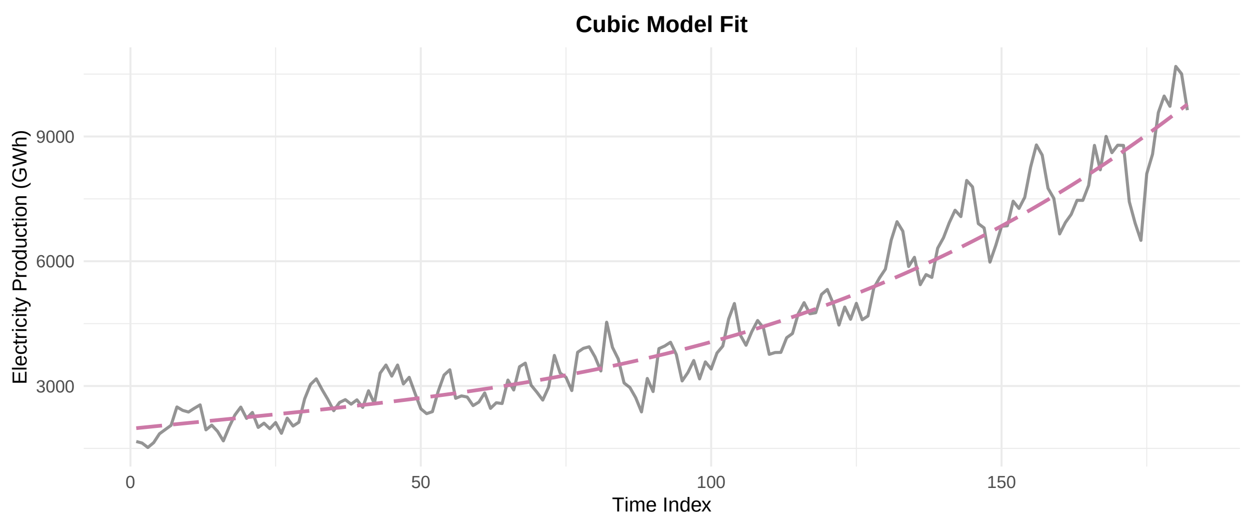
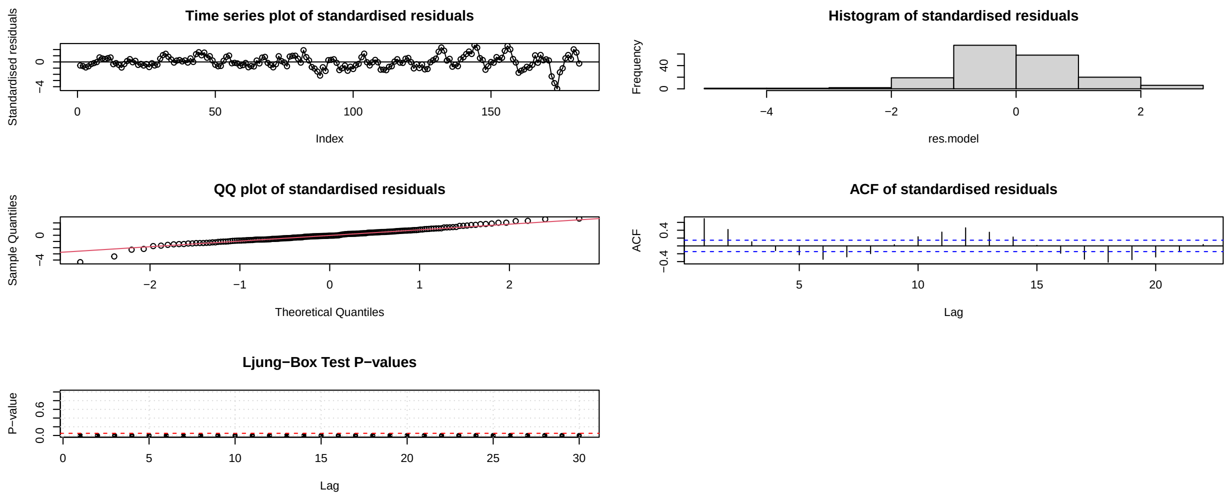


Figure 9: Figure 10 : Cubic Model Fit

```r
residual.analysis(model = cubic_model)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.97405, p-value = 0.001797
```



### Observations: Cubic Model Fit

The cubic model extends the quadratic model by adding a `time³` term, allowing the curve to adapt to more complex nonlinear growth patterns.

- **Adjusted R^2 = 0.9349**, showing a slight improvement over the quadratic model.
- **Residual standard error = 574.6**, indicating better model accuracy.
- **F-statistic** is high and the overall model is statistically significant ($p < 2.2e-16$).
- The **linear term (`time`)** is not statistically significant ($p = 0.0817$), indicating it may not contribute much after accounting for higher-order terms.
- The **quadratic term (`time²`)** is also not significant ($p = 0.6563$), suggesting possible multicollinearity or overfitting.
- The **cubic term (`time³`)** is statistically significant ($p = 0.00326$), capturing complex curvature in the data.
- The **intercept** and overall trend show meaningful baseline and directional movement.
- Still **does not model seasonality**, which leaves autocorrelation in residuals.
- Presence of **insignificant coefficients** raises concerns about overfitting.

Residual Diagnostics

- **Shapiro-Wilk test**: $p = 0.001797$, indicating residuals deviate from normality.
- **Q-Q plot**: Some deviation from the theoretical line, especially in tails.
- **Histogram**: Residuals appear more centered but not perfectly symmetric.
- **ACF plot**: Shows continued autocorrelation in residuals.
- **Ljung-Box test**: Confirms lack of white noise, suggesting room for improvement.
- **Residuals are not independent or normally distributed**, violating key regression assumptions.

The cubic model improves trend modeling but introduces complexity without fully resolving autocorrelation or seasonality. This motivates the move toward **seasonal and time series models (e.g., SARIMA)** that can more rigorously address these characteristics.

To fully represent the behavior of the time series, we now move on to **cyclic and seasonal models**, which explicitly model monthly variation. These models will help incorporate the repeating seasonal behavior that the trend-based models have consistently missed.

```r
# Plot Cyclic Fit
summary(cyclic_model)
```

```
##
## Call:
## lm(formula = value ~ month, data = df_model)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3104.6 -1782.0  -815.2  1826.7  5753.6
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4753.49     571.65   8.315 2.83e-14 ***
## month2       -400.19     808.43  -0.495    0.621
## month3       -708.96     821.80  -0.863    0.390
## month4      -1060.75     821.80  -1.291    0.199
## month5       -704.04     821.80  -0.857    0.393
## month6       -662.71     821.80  -0.806    0.421
## month7       -196.35     821.80  -0.239    0.811
## month8        -40.33     821.80  -0.049    0.961
## month9        -67.54     821.80  -0.082    0.935
## month10        80.45     821.80   0.098    0.922
## month11        16.11     821.80   0.020    0.984
## month12       324.44     821.80   0.395    0.693
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2287 on 170 degrees of freedom
## Multiple R-squared:  0.03125,    Adjusted R-squared:  -0.03144
## F-statistic: 0.4985 on 11 and 170 DF,  p-value: 0.9023
```

```r
ggplot(df_model, aes(x = time, y = value)) +
  geom_line(color = "gray30", size = 1, alpha = 0.6) +
  geom_line(aes(y = cyclic_fit), color = "#009E73", size = 1.2, linetype = "twodash") +
  labs(title = "Cyclic Model Fit (by Month)",
       x = "Time Index",
       y = "Electricity Production (GWh)") +
  custom_theme
```

```r
residual.analysis(model = cyclic_model)
```

```
##
##  Shapiro-Wilk normality test
```

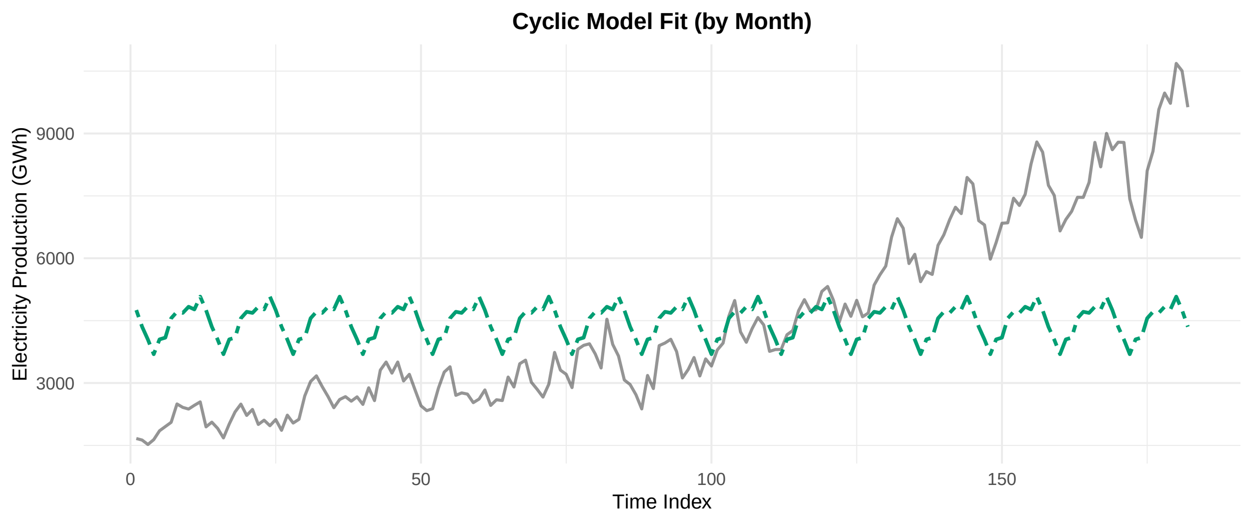
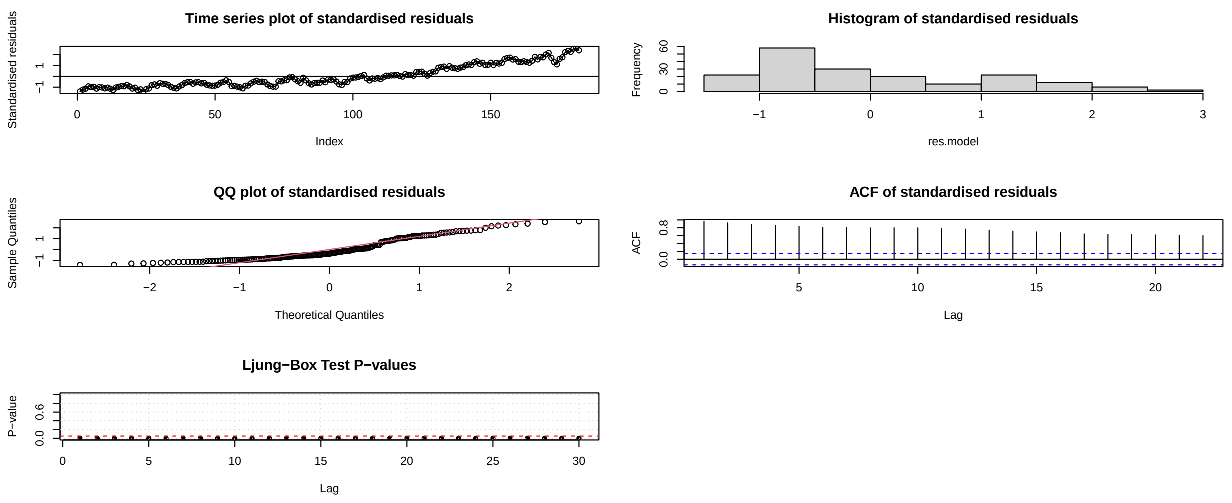


Figure 10: Figure 11: Cyclic Model Fit

```
##
## data:  res.model
## W = 0.90559, p-value = 2.188e-09
```



### Observations: Cyclic Model Fit (by Month)

This model captures potential seasonal effects by including monthly indicators as predictors.

- **Adjusted R² = -0.031**, indicating very poor fit.
- **F-statistic p-value = 0.9023**, suggesting the model is not statistically significant.
- Residual standard error remains high (**2287**), confirming that month-based seasonality alone does not explain the variation.
- Most monthly coefficients are statistically insignificant.
- The intercept is the only significant term, suggesting no strong seasonal effect by month.

Residual Diagnostics

- **Shapiro-Wilk test p-value = 2.19e-09**: strong evidence against normality.

- **Residual plots** show patterns and autocorrelation.
- **Ljung-Box test** confirms residuals are not white noise.

The cyclic model based on month fails to capture the data's structure. It underfits the trend and completely misses variance and autocorrelation patterns. This model alone is insufficient, motivating the need for more sophisticated models like **SARIMA**, which can handle both trend and seasonality effectively.

```r
# Plot Seasonal Fit
summary(seasonal_model)
```

```
##
## Call:
## lm(formula = value ~ time + month, data = df_model)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1752.51  -681.25     2.37   576.08  2314.65
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1186.852    235.939   5.030 1.24e-06 ***
## time          39.194      1.186  33.058  < 2e-16 ***
## month2      -439.380    296.740  -1.481  0.14055
## month3      -552.180    301.680  -1.830  0.06896 .
## month4      -943.168    301.663  -3.127  0.00208 **
## month5      -625.648    301.652  -2.074  0.03959 *
## month6      -623.517    301.645  -2.067  0.04025 *
## month7      -196.348    301.642  -0.651  0.51598
## month8       -79.521    301.645  -0.264  0.79239
## month9      -145.926    301.652  -0.484  0.62919
## month10      -37.131    301.663  -0.123  0.90219
## month11     -140.660    301.680  -0.466  0.64163
## month12      128.473    301.701   0.426  0.67077
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 839.3 on 169 degrees of freedom
## Multiple R-squared:  0.8702, Adjusted R-squared:  0.861
## F-statistic: 94.46 on 12 and 169 DF,  p-value: < 2.2e-16
```

```r
ggplot(df_model, aes(x = time, y = value)) +
  geom_line(color = "gray30", size = 1, alpha = 0.6) +
  geom_line(aes(y = seasonal_fit), color = "#E69F00", size = 1.2, linetype = "solid") +
  labs(title = "Seasonal Model Fit (Time + Month)",
       x = "Time Index",
       y = "Electricity Production (GWh)") +
  custom_theme
```

```r
residual.analysis(model = seasonal_model)
```

```
##
##  Shapiro-Wilk normality test
```

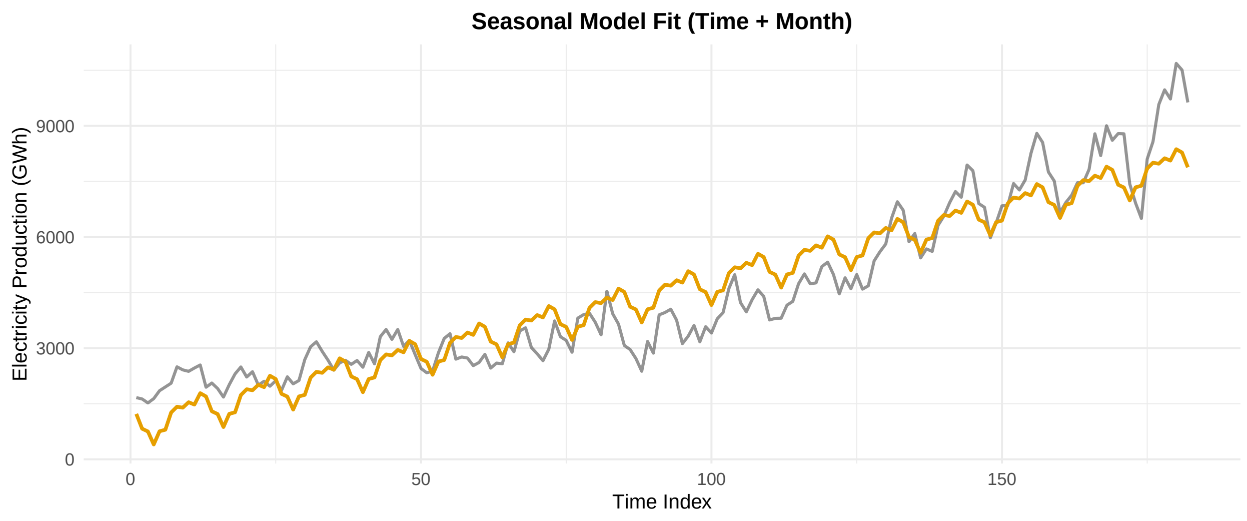
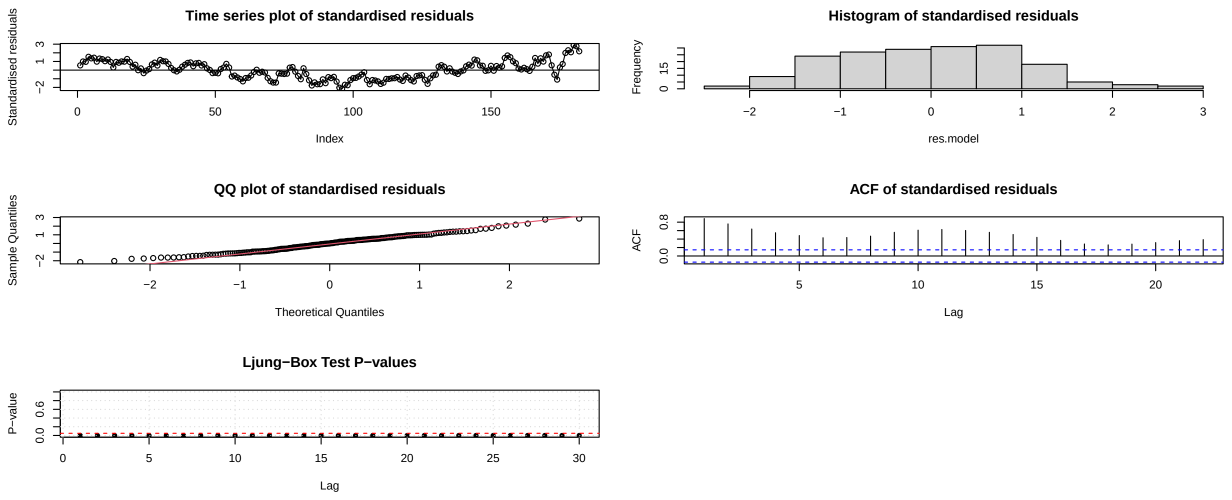


Figure 11: Figure 12: Seasoanl Model Fit

```
##
## data:  res.model
## W = 0.98777, p-value = 0.1173
```



### Observations: Seasonal Model Fit (Time + Month)

This model includes both a linear time trend and monthly dummy variables to capture seasonal patterns in electricity production.

- **Adjusted R² = 0.861**, indicating a strong overall fit.
- Several monthly coefficients (e.g., months 4, 5, 6) are statistically significant, suggesting clear seasonal effects.
- The time trend remains highly significant (**p < 2e-16**), confirming the continued growth of renewable electricity production over time.

Residual Diagnostics

- **Residual Standard Error = 839.3**, lower than in simpler models.

- **Shapiro-Wilk p-value = 0.1173** shows no strong evidence against normality.
- **ACF and Ljung-Box test** indicate some remaining autocorrelation, but reduced compared to previous models.
- Residual plots show more stability, though not entirely white noise.

The seasonal model successfully captures both trend and monthly seasonality, offering a meaningful improvement over purely linear or cyclic models. However, residual autocorrelation remains, suggesting that advanced time series models like SARIMA are needed for better forecasting and inference.

Based on our exploration and fitting of several models to the renewable electricity production data, we can summarize the strengths and weaknesses of each:

- **Linear Model**: Captures the general upward trend but fails to account for curvature or seasonality.
- **Quadratic and Cubic Models**: Improve trend estimation by modeling nonlinear growth, but still ignore seasonal patterns.
- **Cyclic Model**: Captures seasonal variation using monthly categories, but ignores the upward trend over time.
- **Seasonal Model (Time + Month)**: Successfully models both trend and seasonality. It is the **most comprehensive model** so far and fits the data well.

While the seasonal model (time + month) gives a strong fit, it is still a **regression-based model**, which assumes independent residuals. However, our earlier ACF and PACF plots indicate the presence of **autocorrelation**, meaning that **previous values influence current ones**. This is something the seasonal regression model does **not explicitly account for**.

Why Move to SARIMA?

- The time series is **non-stationary** and exhibits both **trend** and **seasonality**.
- There is strong **autocorrelation**, as shown by the ACF/PACF and lag plots.
- Regression-based models fail to address the **time-dependent structure** in residuals.
- **ARIMA models** are designed to handle trend and autocorrelation through **differencing and lag terms**.
- **SARIMA models** extend ARIMA by incorporating **seasonal patterns**, making them ideal for time series like this one with **clear annual cycles**.

SARIMA Model Development Approach

As we have already seen from the Q-Q plot, histogram, and Shapiro-Wilk test, the time series values are **not normally distributed**, and the Augmented Dickey-Fuller test confirmed that the series is **non-stationary**. Additionally, the ACF and PACF plots show strong autocorrelation and seasonality.

Since both **normality** and **stationarity** are important assumptions in SARIMA-type models—particularly for valid residuals and accurate forecasting—we need to transform and difference the series before model fitting. Here's the step-by-step modeling process:

Stabilize Variance

- To address **non-constant variance** and **skewness**, a **Box-Cox transformation** is applied.
- This transformation helps normalize the series and makes the variance more consistent across time.
- The transformed series is then visually inspected to ensure it behaves more regularly.

Initial Model and Differencing

- We begin with a **baseline SARIMA(0,0,0)(0,0,0)**[**12**] model on the transformed series.
- Next, we apply:
  - **Seasonal differencing (lag = 12)** to eliminate yearly seasonality.
  - **First-order differencing** to remove the trend component.
- These steps are essential to make the series **stationary**, which is a requirement for SARIMA-based models.
- Stationarity is confirmed through **plots**, **ADF test**, and observing the flattening of ACF/PACF at high lags.

Model Identification and Refinement

- Once stationarity is achieved, we examine the **ACF and PACF plots** of the differenced series to identify significant autocorrelation and partial autocorrelation lags.
- This guides the selection of appropriate SARIMA parameters:
  - **(p, d, q)**: non-seasonal autoregressive, differencing, and moving average orders.
  - **(P, D, Q)**: seasonal counterparts with seasonal period [**s = 12**].
- Models are fitted iteratively, and **residual diagnostics** are performed after each to check for:
  - White noise residuals,
  - Normality,
  - Independence (via ACF and Ljung-Box test),
  - Model performance (AIC/BIC).

Model Fitting

- Fit candidate models using various estimation methods:
  - **ML (Maximum Likelihood)** – accurate and recommended for final models.
  - **CSS (Conditional Sum of Squares)** – faster for initial fits.
  - **ML+CSS** – hybrid of speed and accuracy.

Model Evaluation

- Assess models using **AIC/BIC** scores and residual diagnostics (e.g., Ljung-Box test).
- Ensure residuals resemble white noise (i.e., uncorrelated and normally distributed).
- Compare **forecast accuracy** using metrics like RMSE or cross-validation error.
- Plot actual vs. fitted values to visually judge the model's performance.

Select the Best Model

- Choose the model that achieves:
  - The **lowest AIC/BIC**,
  - **Well-behaved residuals** (no autocorrelation),
  - **Good forecast performance**.

By going through this full process, we ensure that the selected ARIMA or SARIMA model is robust, accounts for both **trend and seasonality**, and respects the **time-dependent structure** of the data—making it well-suited for accurate forecasting.

Stabilizing Variance with Box-Cox Transformation

Why Transform the Data?

The original time series showed **non-constant variance** (heteroscedasticity) and a **right-skewed distribution**, which can violate assumptions of common time series models. This affects the reliability of statistical tests and can reduce the accuracy of forecasts.

To address this, we applied the **Box-Cox transformation**—a power transformation used to stabilize variance and improve the distributional shape of the data, making it more symmetric and closer to normal.

```
BC <- BoxCox.ar(y = ts_data, lambda = seq(-2, 2, 0.1))
```

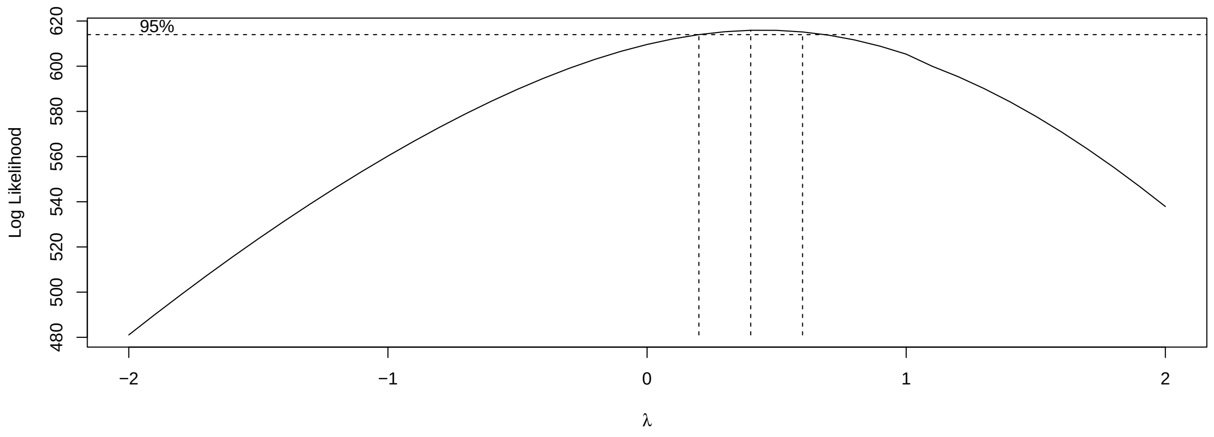


Figure 12: Figure 13: BoxCox Transformation parameter lambda

```
lambda <- BC$lambda[which.max(BC$loglike)]
lambda
```

```
## [1] 0.4
```

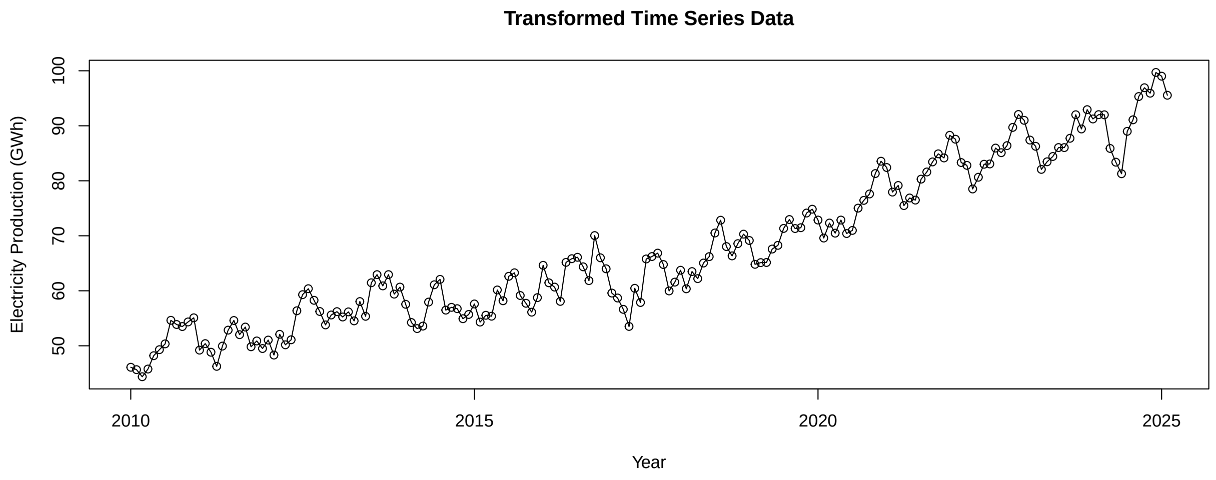
Lambda Selection

The optimal value of the Box-Cox transformation parameter (λ) was determined by maximizing the log-likelihood. The best λ was found to be **0.4**, and the confidence interval around it confirms its suitability.

This choice of λ helps transform the data in a way that **reduces skewness** and brings the distribution closer to a normal shape while also helping stabilize variance across time.

```r
ts_data_transformed <- ((ts_data^lambda) - 1) / lambda

plot(ts_data_transformed,
     main = "Transformed Time Series Data",
     xlab = "Year", ylab = "Electricity Production (GWh)", type = "o")
```



### Transformed Series

After applying the Box-Cox transformation using $\lambda = 0.4$, the time series still shows an upward trend and seasonality, but the **variance is more stable** across the entire period. This is especially noticeable in the later years, where the original series showed increasingly large fluctuations.

```r
qqnorm(ts_data_transformed, main="QQ Plot of Time Series")
qqline(ts_data_transformed, col="black", lwd=2)
```

```r
shapiro.test(ts_data_transformed)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  ts_data_transformed
## W = 0.94476, p-value = 1.724e-06
```

Variance Stabilization Decision

- The Box-Cox analysis suggested an optimal lambda of approximately **0.4**, indicating that a transformation could help stabilize variance.

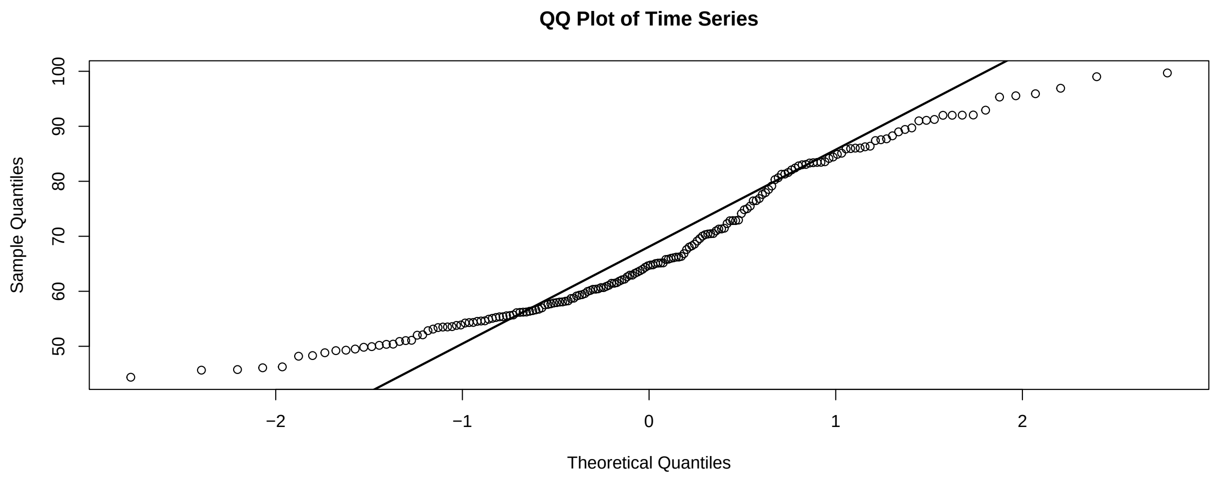


Figure 13: Figure 15: QQ Plot of Time Series

- After applying the transformation, the spread of values appeared more consistent, and the Q-Q plot showed slight improvement.
- However, the **Shapiro-Wilk test p-value**, while improved, still indicated non-normality (p = 1.72e-06). ### Final Decision: Use of Original Data > Although transformation was considered, the original data is used for model fitting due to its sufficient performance and ease of interpretation.
- The **improvement from transformation was minor**, and the original dataset already exhibited stable patterns suitable for modeling.
- To maintain **interpretability** of the results and avoid complicating the inverse transformation during forecasting, the original (untransformed) series is retained for model fitting.
- Residual checks and model diagnostics will be thoroughly reviewed to ensure assumptions are reasonably met.

SARIMA Model fitting

To begin the modeling process, the base model was fitted by setting only the seasonal differencing parameter to account for the strong annual seasonal pattern observed in the data. All other parameters were initialized to zero, resulting in a SARIMA(0,0,0)(0,1,0)[12] model. However, the residuals exhibited clear autocorrelations, indicating that seasonal components were not adequately addressed.

```r
m1.ts_data = Arima(ts_data,order=c(0,0,0),seasonal=list(order=c(0,1,0), period=12))
res.m1 = residuals(m1.ts_data);
par(mfrow=c(1,1))
plot(res.m1,xlab='Time',ylab='Residuals',main="Time series plot of the residuals")
```

```r
par(mfrow=c(1,2))
seasonal_acf(res.m1, lag.max = 50, main = "ACF plot of the residuals")
seasonal_pacf(res.m1, lag.max = 50, main = "PACF plot of the residuals")
```

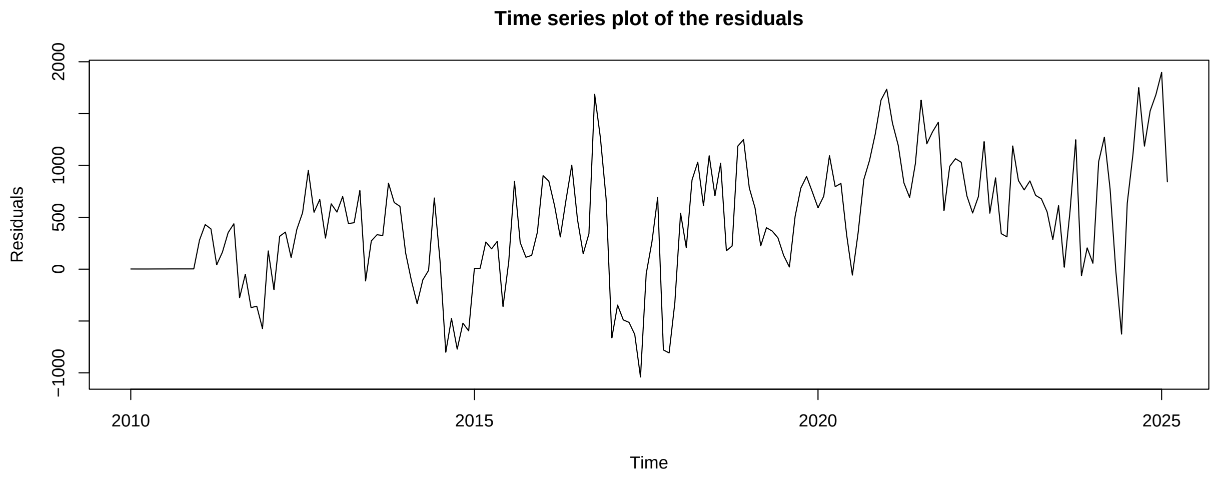


Figure 14: Figure 16: Time series,ACF & PCF of seasonal differencing of residuals

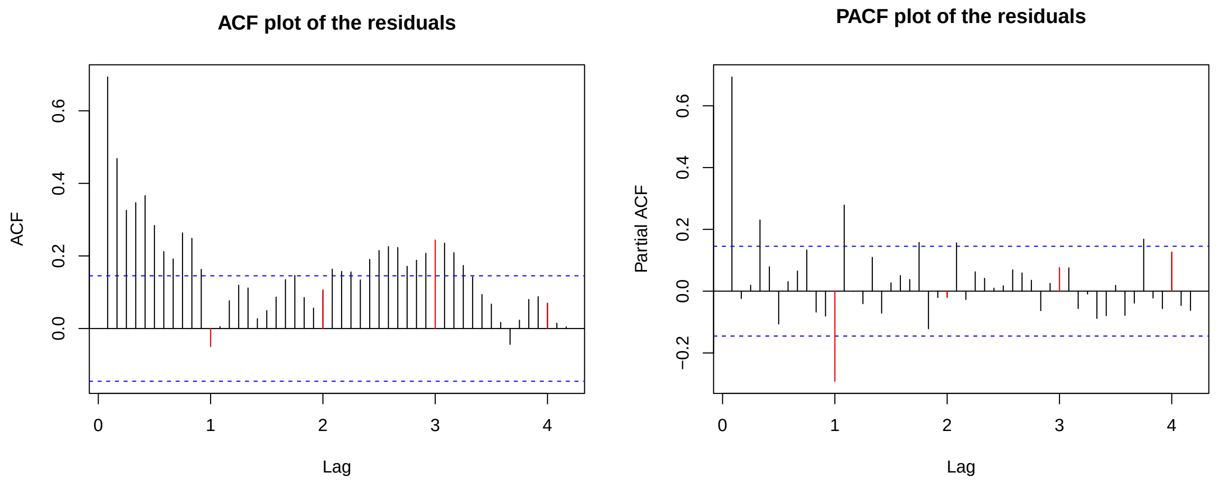


Figure 15: Figure 16: Time series,ACF & PCF of seasonal differencing of residuals

The model was then updated to SARIMA(0,1,0)(1,1,1)[12], introducing both first-order differencing to address non-stationarity and seasonal AR and MA terms. The ACF and PACF of residuals showed significant autocorrelations at early lags, suggesting underfitting and motivating the inclusion of non-seasonal AR and MA components.

```r
m2.ts_data = Arima(ts_data,order=c(0,1,0),seasonal=list(order=c(1,1,1), period=12))
res.m2 = residuals(m2.ts_data);
par(mfrow=c(1,1))
plot(res.m2,xlab='Time',ylab='Residuals',main="Time series plot of the residuals")
```

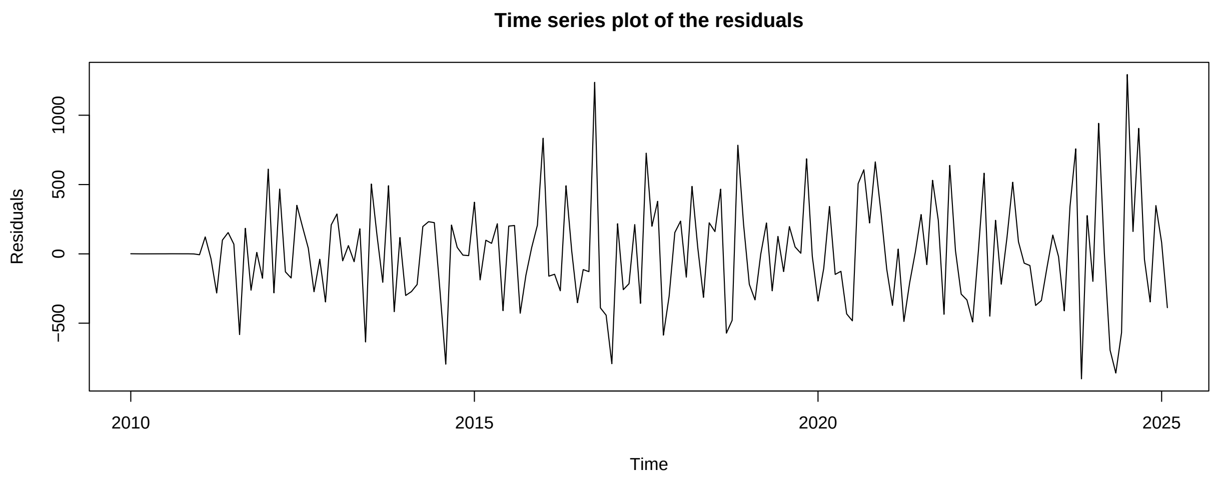
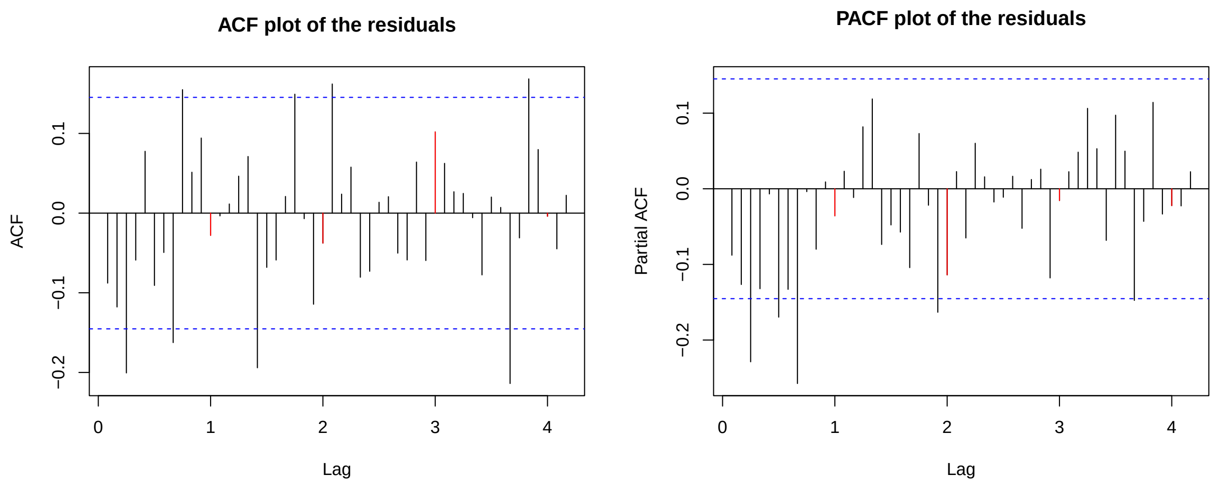


Figure 16: Figure 17: Time series,ACF & PCF of first-order differencing 1 of residuals

```r
par(mfrow=c(1,2))
seasonal_acf(res.m2, lag.max = 50, main = "ACF plot of the residuals")
seasonal_pacf(res.m2, lag.max = 50, main = "PACF plot of the residuals")
```



Based on the residuals of the SARIMA(0,1,0)(1,1,1)[12] model, the ACF and PACF plots indicated significant lags within the first few periods, leading to the trial of SARIMA(3,1,3)(1,1,1)[12]. This model removed significant autocorrelations in the residuals, indicating an adequate fit.

```r
m3.ts_data = Arima(ts_data,order=c(3,1,3),seasonal=list(order=c(1,1,1), period=12))
res.m3 = residuals(m3.ts_data);
par(mfrow=c(1,1))
plot(res.m3,xlab='Time',ylab='Residuals',main="Time series plot of the residuals")
```

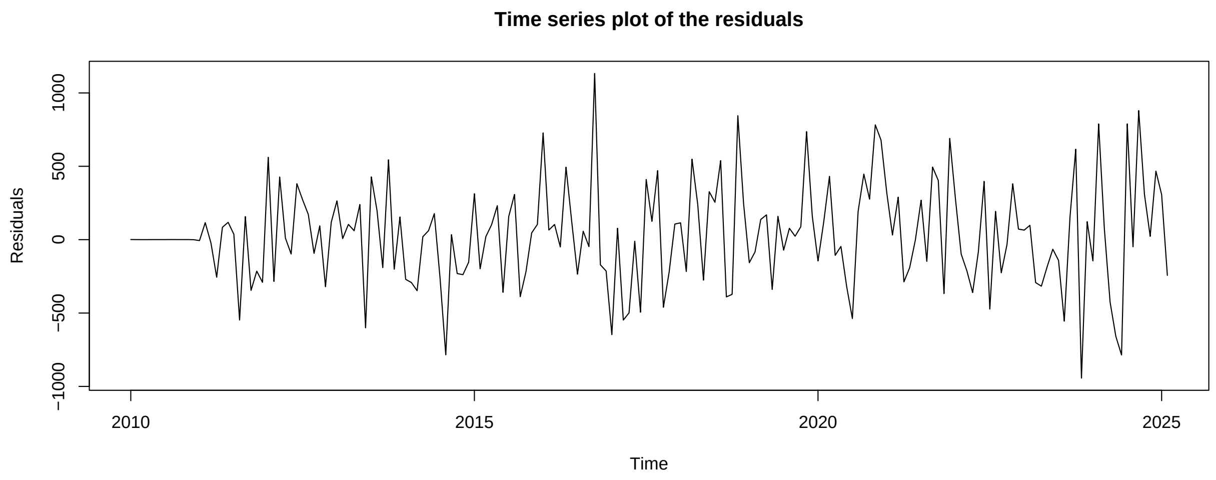
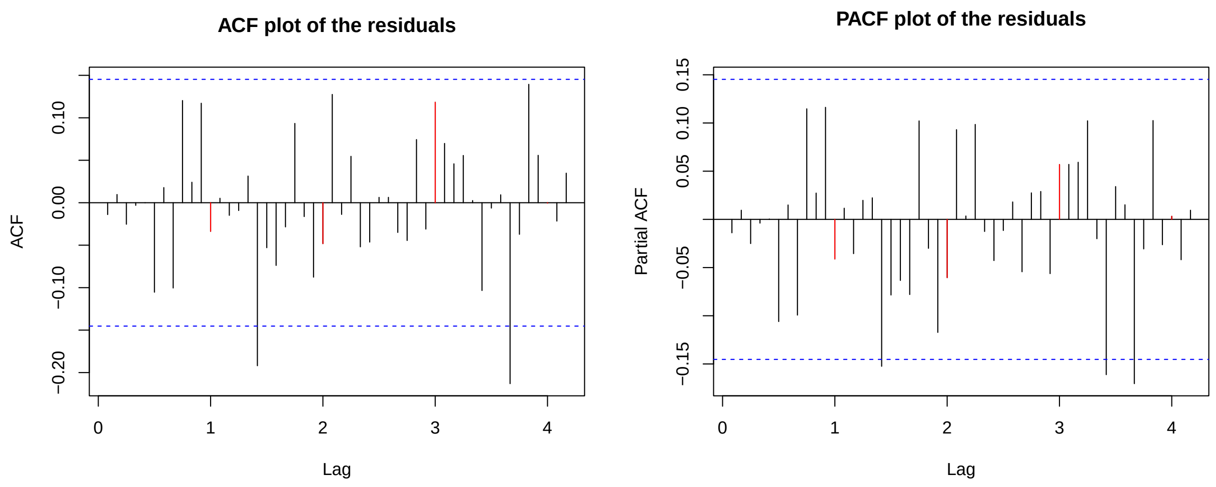


Figure 17: Figure 18: Time series,ACF & PCF of first-order differencing 2 of residuals

```r
par(mfrow=c(1,2))
seasonal_acf(res.m3, lag.max = 50, main = "ACF plot of the residuals")
seasonal_pacf(res.m3, lag.max = 50, main = "PACF plot of the residuals")
```



To refine the model selection process, both the Extended Autocorrelation Function (EACF) and the BIC-based armasubsets plot were used.

```r
eacf(res.m2)
```

```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 o o x o o o o x x o o  o  o  o
## 1 x o o o o o o o x o o  o  o  o
## 2 x o o o o o o o o o o  o  o  o
## 3 x x x o o o o o o o o  o  o  o
## 4 o o x x o o o o o o o  o  o  o
## 5 o x o x x o o x o o o  o  o  o
## 6 x x o x x x o o o o o  o  o  o
## 7 x x x x x x o o o o o  o  o  o
```

EACF suggested the following candidates:

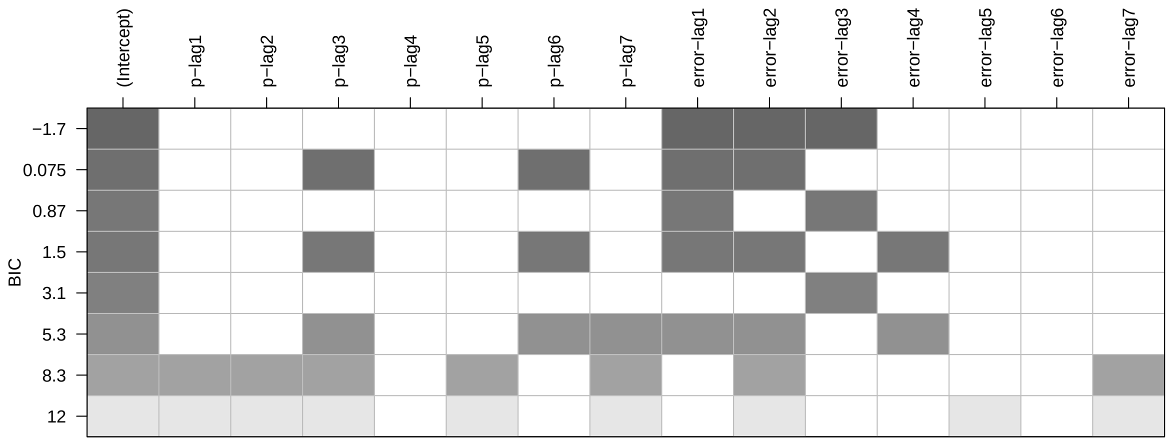
SARIMA(1,1,1)(1,1,1)[12]

SARIMA(1,1,2)(1,1,1)[12]

SARIMA(2,1,1)(1,1,1)[12]

SARIMA(2,1,2)(1,1,1)[12]

```
par(mfrow=c(1,1))
bic = armasubsets(y=res.m2,nar=7,nma=7,y.name='p',ar.method='ols')
plot(bic)
```



BIC analysis recommended additional models:

SARIMA(0,1,1)(1,1,1)[12], SARIMA(0,1,2)(1,1,1)[12], SARIMA(0,1,3)(1,1,1)[12]

SARIMA(3,1,1)(1,1,1)[12], SARIMA(3,1,2)(1,1,1)[12]

SARIMA(6,1,1)(1,1,1)[12], SARIMA(6,1,2)(1,1,1)[12]

Combining these, a total of 12 models were shortlisted for fitting and evaluation.

All shortlisted models were fitted using the Arima() function with ML and CSS estimation methods. The z-statistics and p-values from coeftest() were used to evaluate parameter significance.

```
m_011_ml = Arima(ts_data,order=c(0,1,1), seasonal=list(order=c(1,1,1), period=12), method = "ML")
print("Coefficient of 011-ML")
```

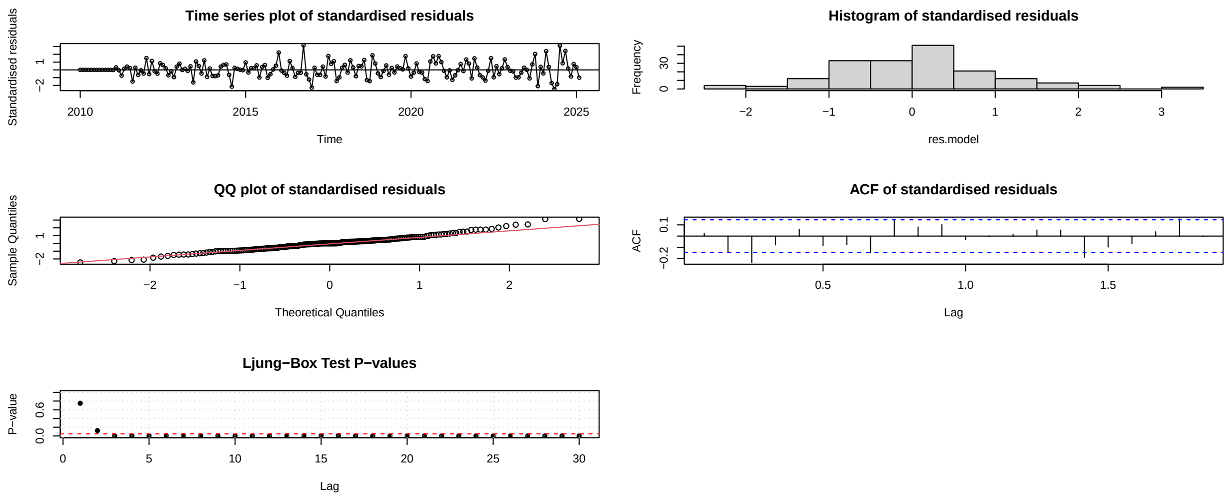
```
## [1] "Coefficient of 011-ML"
```

```r
coeftest(m_011_ml)
```

```
##
## z test of coefficients:
##
##       Estimate Std. Error z value  Pr(>|z|)
## ma1  -0.13827    0.10374 -1.3329    0.1826
## sar1 -0.13676    0.13122 -1.0422    0.2973
## sma1 -0.54448    0.12237 -4.4493 8.616e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
residual.analysis(model = m_011_ml)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.98411, p-value = 0.03688
```



```r
m_011_css = Arima(ts_data,order=c(0,1,1), seasonal=list(order=c(1,1,1), period=12), method = "CSS")
print("Coefficient of 011-CSS")
```

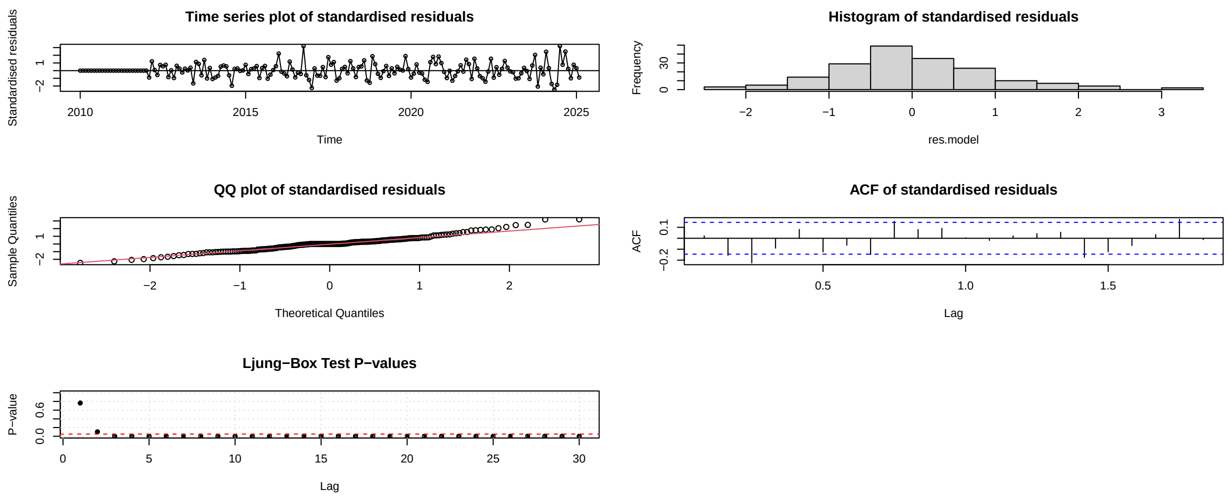
```
## [1] "Coefficient of 011-CSS"
```

```r
coeftest(m_011_css)
```

```
##
## z test of coefficients:
##
##       Estimate Std. Error z value  Pr(>|z|)
## ma1  -0.12701    0.10272 -1.2364   0.21629
## sar1 -0.20228    0.11473 -1.7631   0.07789 .
## sma1 -0.52423    0.10830 -4.8405 1.295e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
residual.analysis(model = m_011_css)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.97765, p-value = 0.005082
```



```
m_012_ml = Arima(ts_data,order=c(0,1,2), seasonal=list(order=c(1,1,1), period=12), method = "ML")
print("Coefficient of 012-ML")
```

```
## [1] "Coefficient of 012-ML"
```

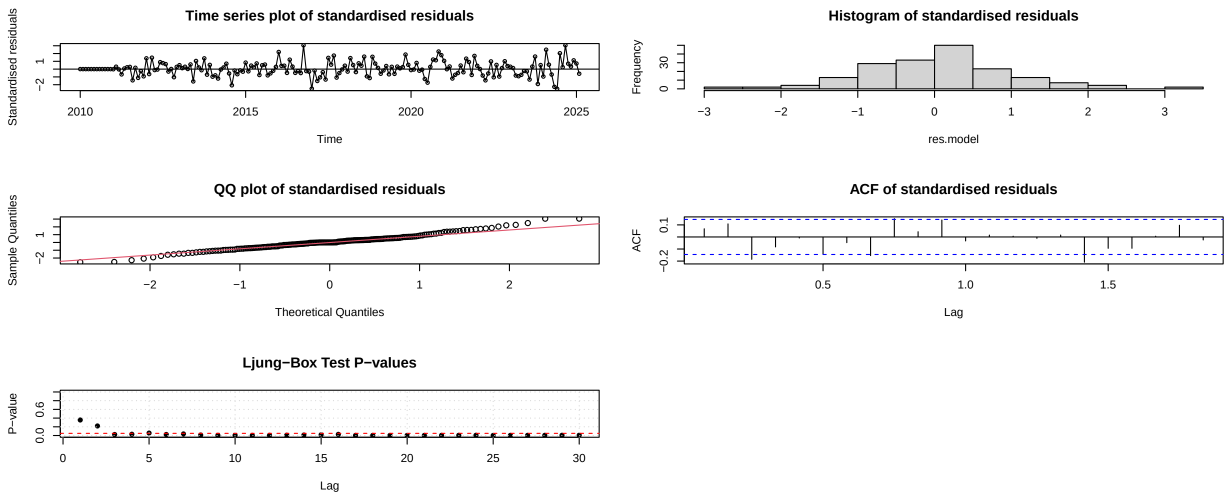
```
coeftest(m_012_ml)
```

```
##
## z test of coefficients:
##
##        Estimate Std. Error z value  Pr(>|z|)
## ma1  -0.299296   0.077685 -3.8527 0.0001168 ***
## ma2  -0.318160   0.080083 -3.9729 7.100e-05 ***
## sar1 -0.178173   0.126284 -1.4109 0.1582759
## sma1 -0.452134   0.112990 -4.0015 6.293e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
residual.analysis(model = m_012_ml)
```

```
##
```

```
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.98466, p-value = 0.04383
```



```r
m_012_css = Arima(ts_data,order=c(0,1,2), seasonal=list(order=c(1,1,1), period=12), method = "CSS")
print("Coefficient of 012-CSS")
```

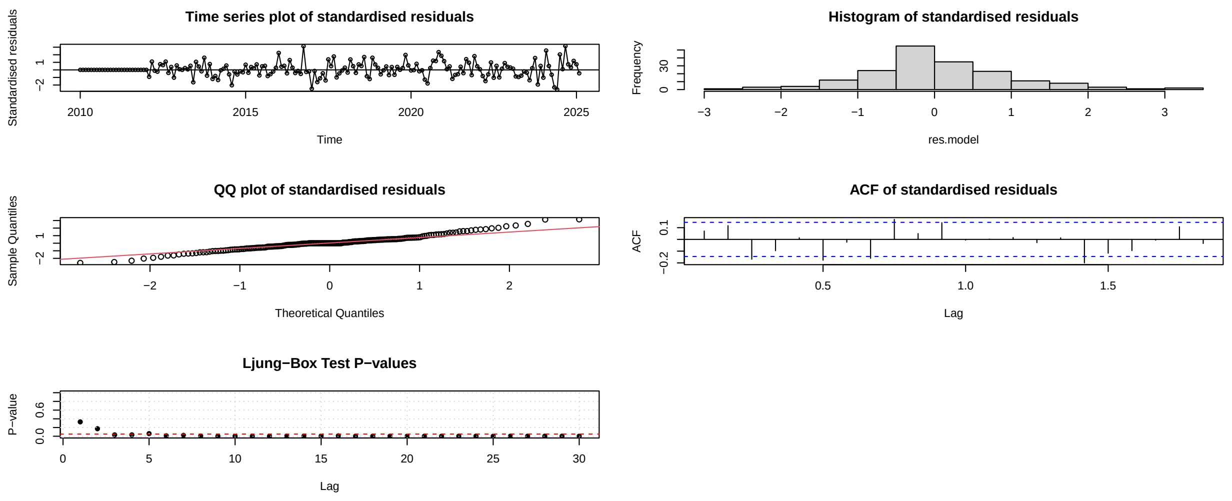
```
## [1] "Coefficient of 012-CSS"
```

```r
coeftest(m_012_css)
```

```
##
## z test of coefficients:
##
##       Estimate Std. Error z value  Pr(>|z|)
## ma1  -0.298243   0.074977 -3.9778 6.956e-05 ***
## ma2  -0.348544   0.078666 -4.4307 9.393e-06 ***
## sar1 -0.239321   0.112085 -2.1352   0.03275 *
## sma1 -0.427198   0.105134 -4.0633 4.837e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
residual.analysis(model = m_012_css)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.97595, p-value = 0.003095
```



```
m_013_ml = Arima(ts_data,order=c(0,1,3), seasonal=list(order=c(1,1,1), period=12), method = "ML")
print("Coefficient of 013-ML")
```

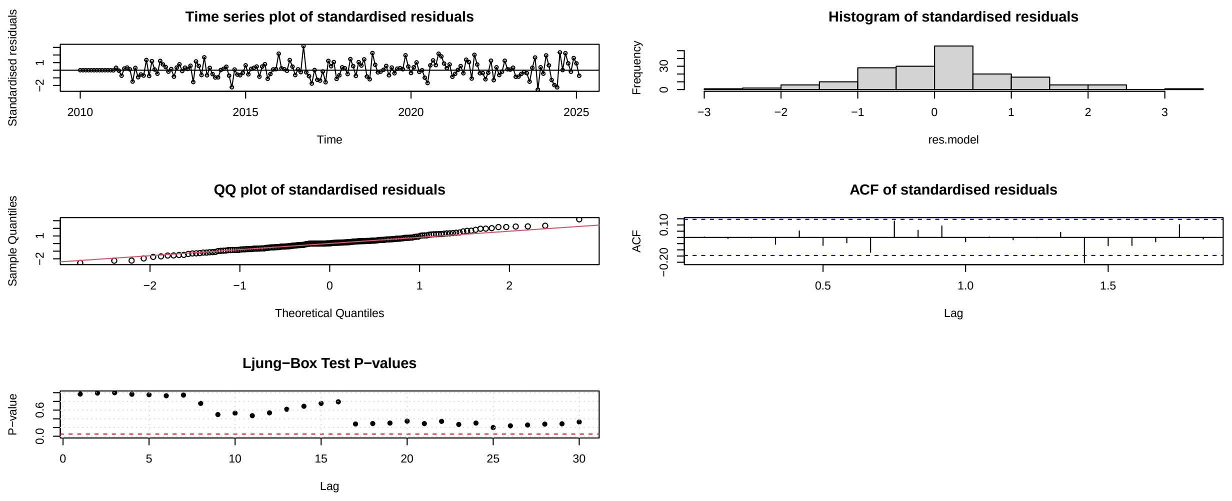
```
## [1] "Coefficient of 013-ML"
```

```
coeftest(m_013_ml)
```

```
##
## z test of coefficients:
##
##       Estimate Std. Error z value  Pr(>|z|)
## ma1  -0.244093   0.073608 -3.3161 0.0009127 ***
## ma2  -0.199833   0.078967 -2.5306 0.0113868 *
## ma3  -0.286133   0.076235 -3.7533 0.0001745 ***
## sar1 -0.186448   0.132296 -1.4093 0.1587392
## sma1 -0.440553   0.124276 -3.5450 0.0003927 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
residual.analysis(model = m_013_ml)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.98858, p-value = 0.151
```



```
m_013_css = Arima(ts_data,order=c(0,1,3), seasonal=list(order=c(1,1,1), period=12), method = "CSS")
print("Coefficient of 013-CSS")
```

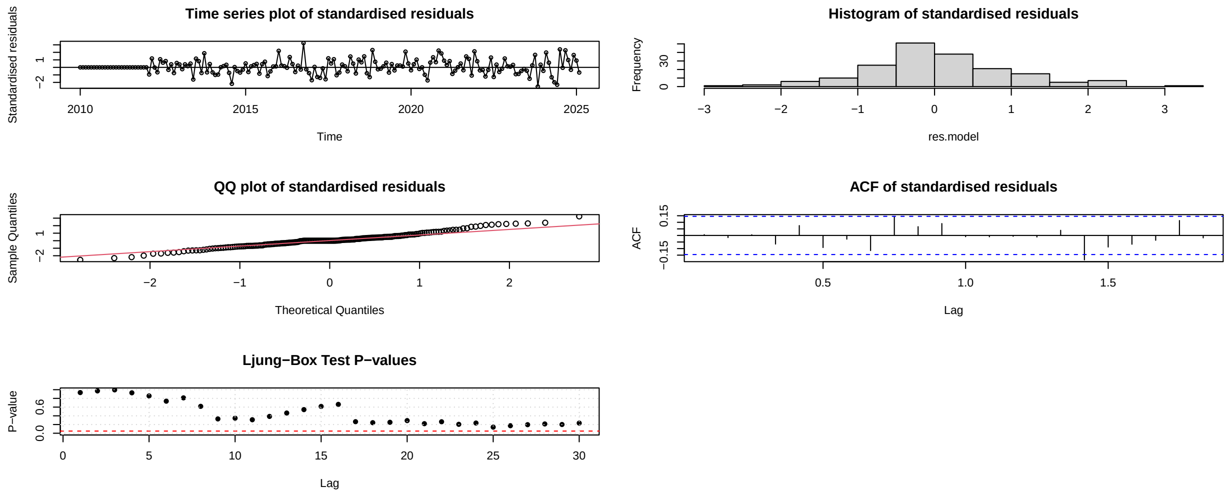
```
## [1] "Coefficient of 013-CSS"
```

```
coeftest(m_013_css)
```

```
##
## z test of coefficients:
##
##       Estimate Std. Error z value  Pr(>|z|)
## ma1  -0.242174   0.072212 -3.3537 0.0007975 ***
## ma2  -0.201071   0.080829 -2.4876 0.0128601 *
## ma3  -0.296067   0.078574 -3.7680 0.0001646 ***
## sar1 -0.240640   0.118591 -2.0292 0.0424426 *
## sma1 -0.418592   0.114985 -3.6404 0.0002722 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
residual.analysis(model = m_013_css)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.98141, p-value = 0.01582
```



```
m_111_ml = Arima(ts_data,order=c(1,1,1), seasonal=list(order=c(1,1,1), period=12), method = "ML")
print("Coefficient of 111-ML")
```

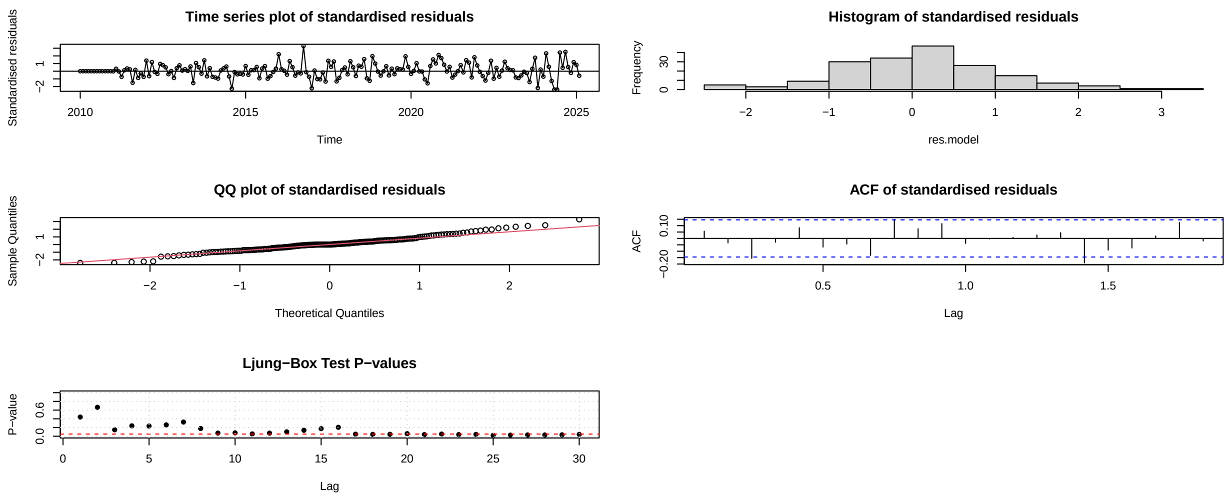
```
## [1] "Coefficient of 111-ML"
```

```
coeftest(m_111_ml)
```

```
##
## z test of coefficients:
##
##       Estimate Std. Error  z value  Pr(>|z|)
## ar1   0.616043   0.082042   7.5089 5.963e-14 ***
## ma1  -0.906228   0.041785 -21.6881 < 2.2e-16 ***
## sar1 -0.161196   0.130114  -1.2389    0.2154
## sma1 -0.482339   0.120283  -4.0100 6.071e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
residual.analysis(model = m_111_ml)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.98386, p-value = 0.03403
```



```
m_111_css = Arima(ts_data,order=c(1,1,1), seasonal=list(order=c(1,1,1), period=12), method = "CSS")
print("Coefficient of 111-CSS")
```

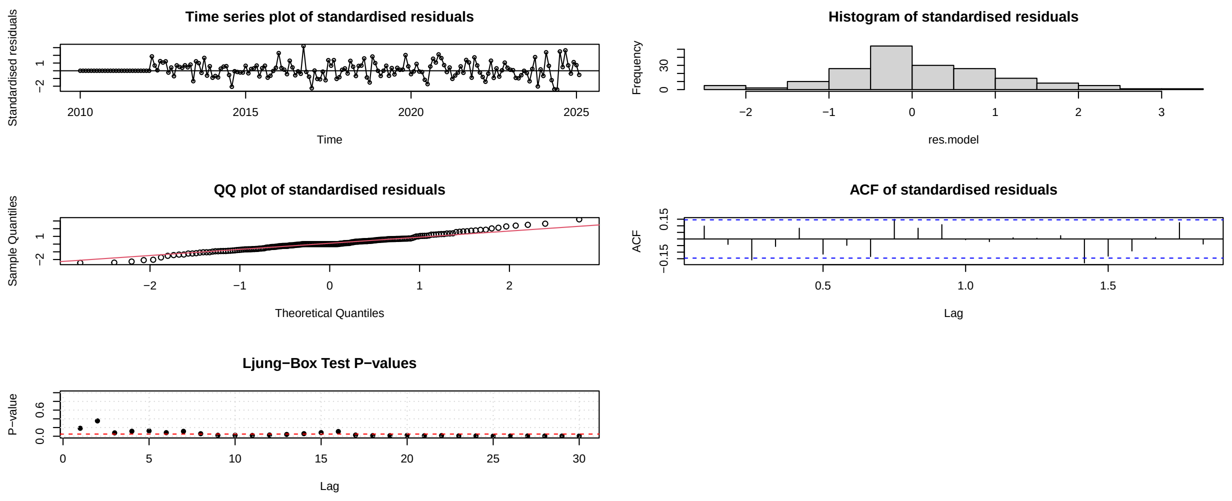
```
## [1] "Coefficient of 111-CSS"
```

```
coeftest(m_111_css)
```

```
##
## z test of coefficients:
##
##       Estimate Std. Error  z value  Pr(>|z|)
## ar1   0.546955   0.084048   6.5077 7.633e-11 ***
## ma1  -0.840166   0.056294 -14.9245 < 2.2e-16 ***
## sar1 -0.261463   0.113152  -2.3107   0.02085 *
## sma1 -0.432847   0.109866  -3.9398 8.156e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
residual.analysis(model = m_111_css)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.98052, p-value = 0.01206
```



```r
m_112_ml = Arima(ts_data,order=c(1,1,2), seasonal=list(order=c(1,1,1), period=12), method = "ML")
print("Coefficient of 112-ML")
```

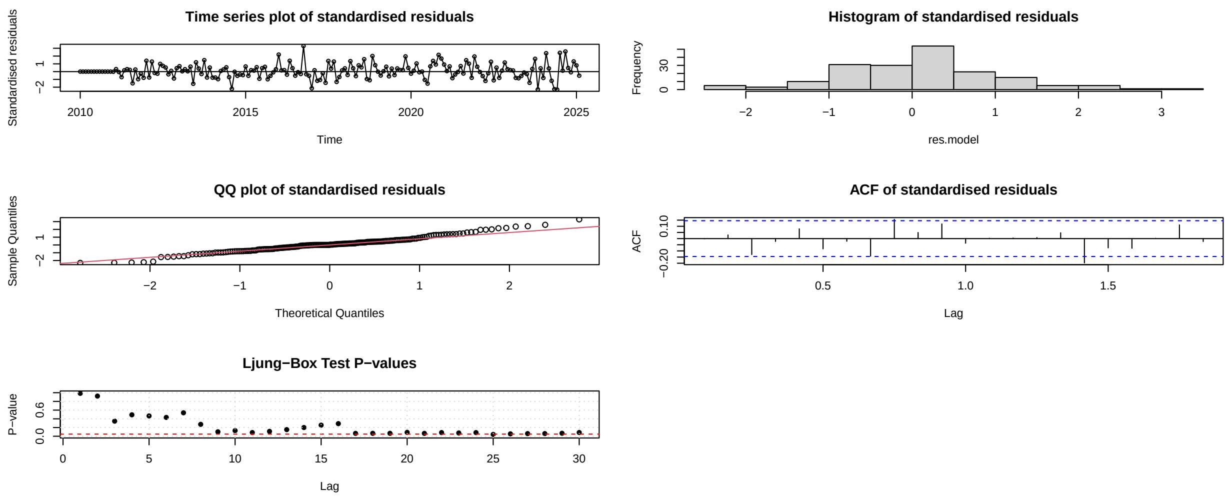
```
## [1] "Coefficient of 112-ML"
```

```r
coeftest(m_112_ml)
```

```
##
## z test of coefficients:
##
##       Estimate Std. Error z value  Pr(>|z|)
## ar1   0.506094   0.129741  3.9008 9.587e-05 ***
## ma1  -0.744683   0.133098 -5.5950 2.206e-08 ***
## ma2  -0.130630   0.097722 -1.3367    0.1813
## sar1 -0.167153   0.129256 -1.2932    0.1959
## sma1 -0.482342   0.119430 -4.0387 5.375e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
residual.analysis(model = m_112_ml)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.98203, p-value = 0.01919
```



```
m_112_css = Arima(ts_data,order=c(1,1,2), seasonal=list(order=c(1,1,1), period=12), method = "CSS")
print("Coefficient of 112-CSS")
```

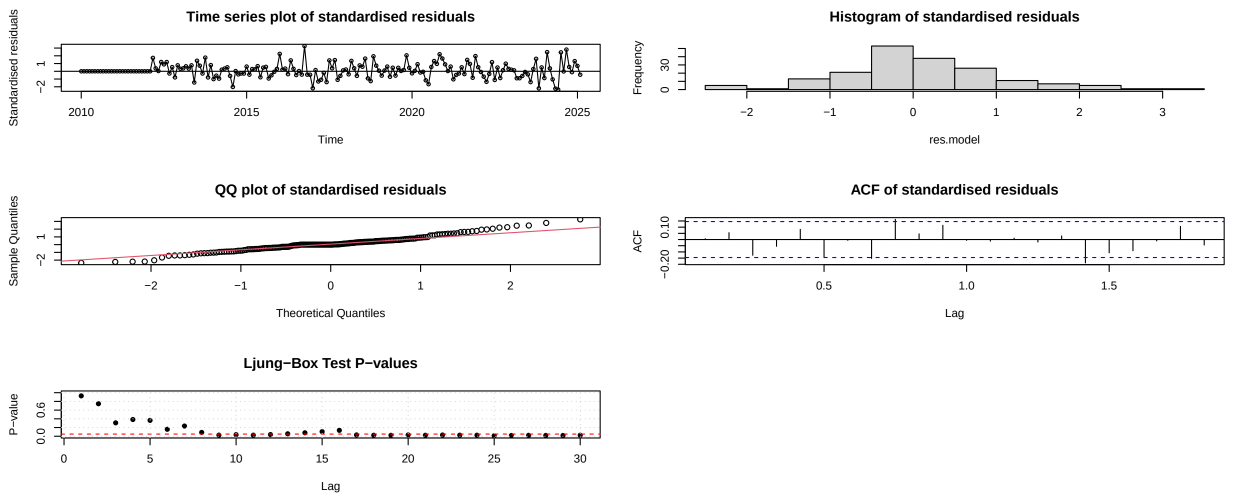
```
## [1] "Coefficient of 112-CSS"
```

```
coeftest(m_112_css)
```

```
##
## z test of coefficients:
##
##       Estimate Std. Error z value  Pr(>|z|)
## ar1   0.379457   0.134905  2.8128  0.004912 **
## ma1  -0.600442   0.129689 -4.6299 3.659e-06 ***
## ma2  -0.197936   0.090233 -2.1936  0.028263 *
## sar1 -0.257289   0.111497 -2.3076  0.021021 *
## sma1 -0.437272   0.107989 -4.0492 5.139e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
residual.analysis(model = m_112_css)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.9761, p-value = 0.003226
```



```
m_211_ml = Arima(ts_data,order=c(2,1,1), seasonal=list(order=c(1,1,1), period=12), method = "ML")
print("Coefficient of 211-ML")
```

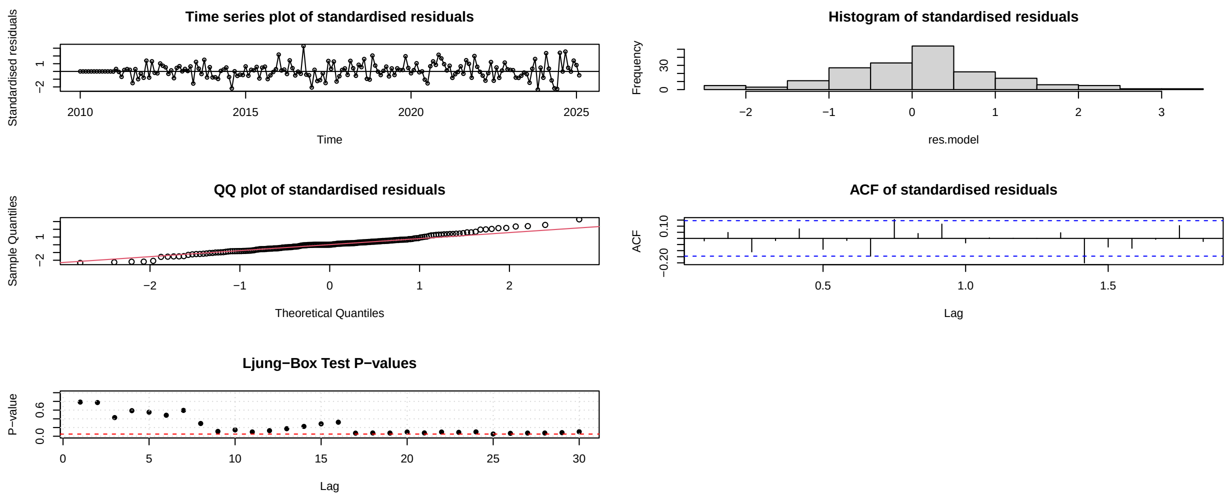
```
## [1] "Coefficient of 211-ML"
```

```
coeftest(m_211_ml)
```

```
##
## z test of coefficients:
##
##       Estimate Std. Error  z value  Pr(>|z|)
## ar1   0.655893   0.091909   7.1363 9.586e-13 ***
## ar2  -0.125745   0.081760  -1.5380    0.1241
## ma1  -0.877901   0.055857 -15.7169 < 2.2e-16 ***
## sar1 -0.170359   0.129294  -1.3176    0.1876
## sma1 -0.480894   0.119612  -4.0205 5.808e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
residual.analysis(model = m_211_ml)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.98276, p-value = 0.02407
```



```
m_211_css = Arima(ts_data,order=c(2,1,1), seasonal=list(order=c(1,1,1), period=12), method = "CSS")
print("Coefficient of 211-CSS")
```

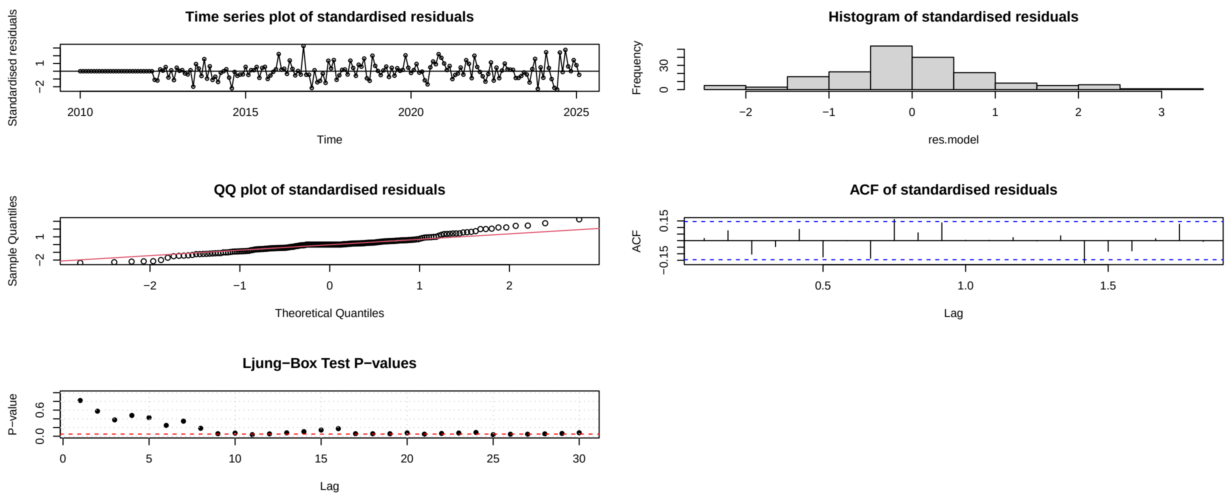
```
## [1] "Coefficient of 211-CSS"
```

```
coeftest(m_211_css)
```

```
##
## z test of coefficients:
##
##       Estimate Std. Error  z value  Pr(>|z|)
## ar1   0.592948   0.084502   7.0169 2.268e-12 ***
## ar2  -0.161697   0.079557  -2.0325   0.04211 *
## ma1  -0.818294   0.057934 -14.1245 < 2.2e-16 ***
## sar1 -0.253260   0.110013  -2.3021   0.02133 *
## sma1 -0.422734   0.108360  -3.9012 9.573e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
residual.analysis(model = m_211_css)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.96765, p-value = 0.0003168
```



```
m_212_ml = Arima(ts_data,order=c(2,1,2), seasonal=list(order=c(1,1,1), period=12), method = "ML")
print("Coefficient of 212-ML")
```

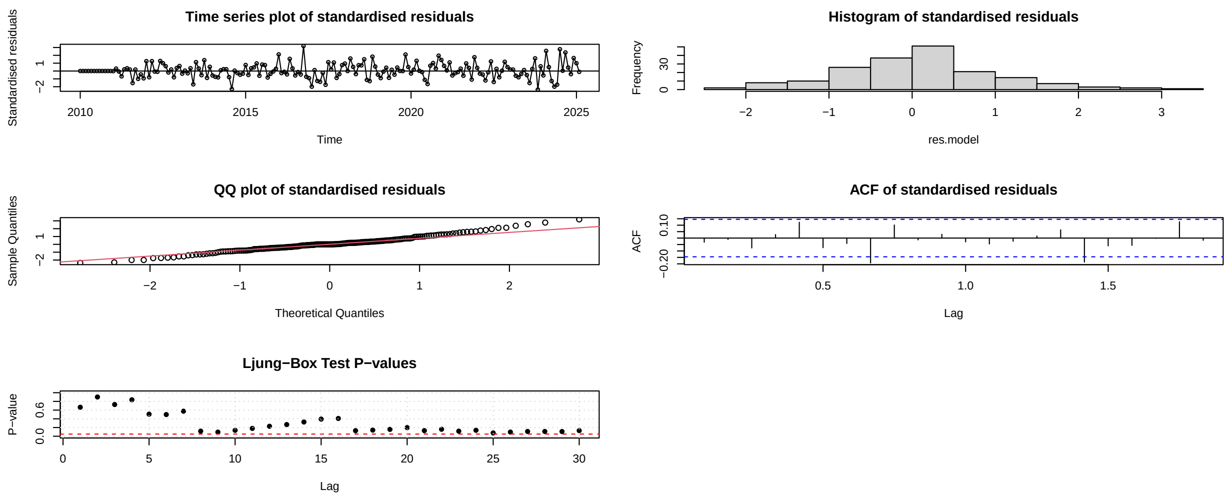
```
## [1] "Coefficient of 212-ML"
```

```
coeftest(m_212_ml)
```

```
##
## z test of coefficients:
##
##       Estimate Std. Error  z value  Pr(>|z|)
## ar1   1.546140   0.102551  15.0769 < 2.2e-16 ***
## ar2  -0.755299   0.093846  -8.0483 8.397e-16 ***
## ma1  -1.742530   0.090185 -19.3217 < 2.2e-16 ***
## ma2   0.842560   0.085186   9.9908 < 2.2e-16 ***
## sar1 -0.168585   0.128447  -1.3125    0.1894
## sma1 -0.593988   0.134631  -4.4120 1.024e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
residual.analysis(model = m_212_ml)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.98498, p-value = 0.04852
```



```
m_212_css = Arima(ts_data,order=c(2,1,2), seasonal=list(order=c(1,1,1), period=12), method = "CSS")
print("Coefficient of 212-CSS")
```

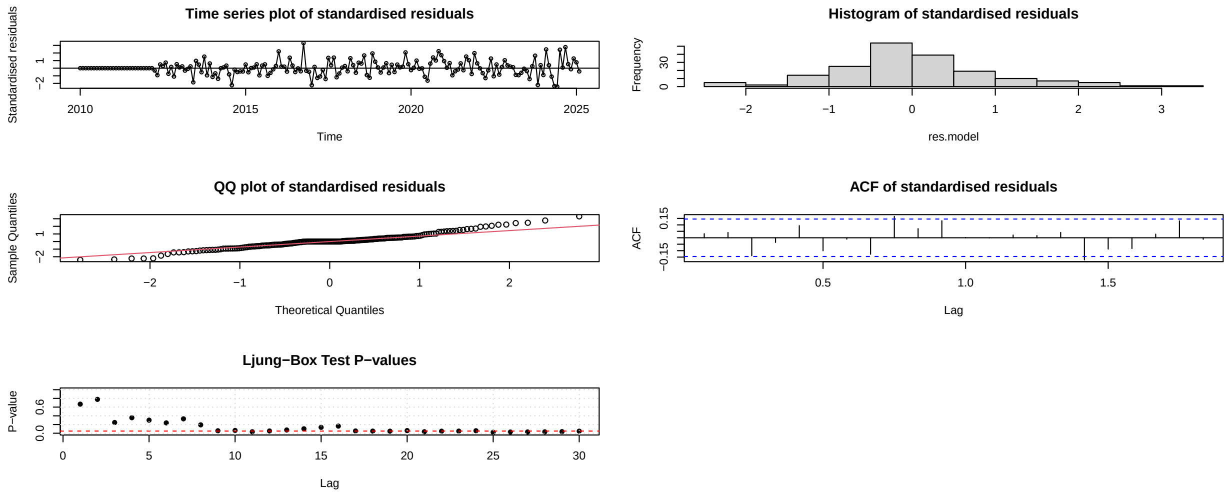
```
## [1] "Coefficient of 212-CSS"
```

```
coeftest(m_212_css)
```

```
##
## z test of coefficients:
##
##       Estimate Std. Error z value  Pr(>|z|)
## ar1    0.22889    0.52892  0.4328   0.66519
## ar2    0.14391    0.43979  0.3272   0.74350
## ma1   -0.47573    0.49777 -0.9557   0.33921
## ma2   -0.35435    0.50312 -0.7043   0.48124
## sar1  -0.23656    0.11340 -2.0860   0.03697 *
## sma1  -0.45368    0.11537 -3.9323 8.415e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
residual.analysis(model = m_212_css)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.96717, p-value = 0.0002799
```



```
m_311_ml = Arima(ts_data,order=c(3,1,1), seasonal=list(order=c(1,1,1), period=12), method = "ML")
print("Coefficient of 311-ML")
```

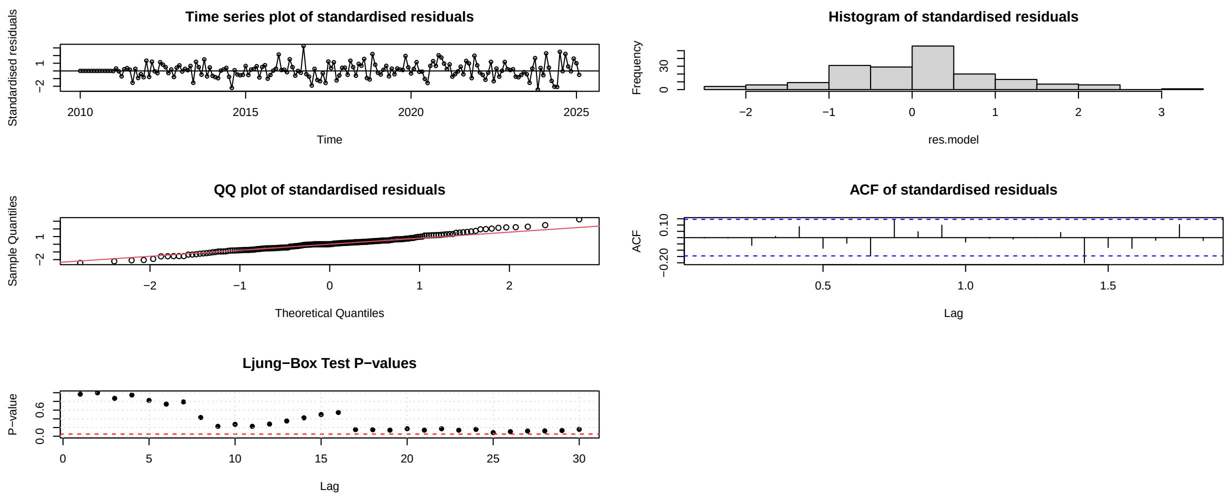
```
## [1] "Coefficient of 311-ML"
```

```
coeftest(m_311_ml)
```

```
##
## z test of coefficients:
##
##       Estimate Std. Error z value  Pr(>|z|)
## ar1   0.603156   0.111617  5.4038 6.524e-08 ***
## ar2  -0.077102   0.089949 -0.8572 0.3913524
## ar3  -0.108926   0.086518 -1.2590 0.2080333
## ma1  -0.834575   0.084265 -9.9042 < 2.2e-16 ***
## sar1 -0.174768   0.131428 -1.3298 0.1835961
## sma1 -0.478116   0.124189 -3.8499 0.0001182 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
residual.analysis(model = m_311_ml)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.98697, p-value = 0.09103
```



```
m_311_css = Arima(ts_data,order=c(3,1,1), seasonal=list(order=c(1,1,1), period=12), method = "CSS")
print("Coefficient of 311-CSS")
```

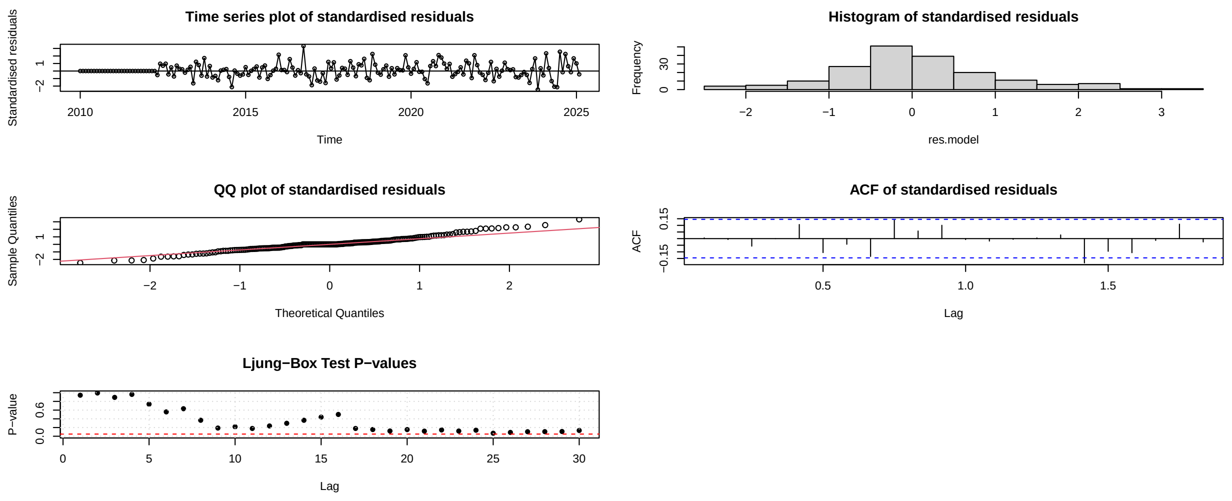
```
## [1] "Coefficient of 311-CSS"
```

```
coeftest(m_311_css)
```

```
##
## z test of coefficients:
##
##       Estimate Std. Error  z value  Pr(>|z|)
## ar1   0.607345   0.107609   5.6440 1.662e-08 ***
## ar2  -0.085432   0.089036  -0.9595   0.33730
## ar3  -0.116395   0.085614  -1.3595   0.17398
## ma1  -0.832835   0.080870 -10.2984 < 2.2e-16 ***
## sar1 -0.228518   0.114510  -1.9956   0.04598 *
## sma1 -0.459808   0.114851  -4.0035 6.240e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
residual.analysis(model = m_311_css)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.97675, p-value = 0.003906
```



```r
m_312_ml = Arima(ts_data,order=c(3,1,2), seasonal=list(order=c(1,1,1), period=12), method = "ML")
print("Coefficient of 312-ML")
```

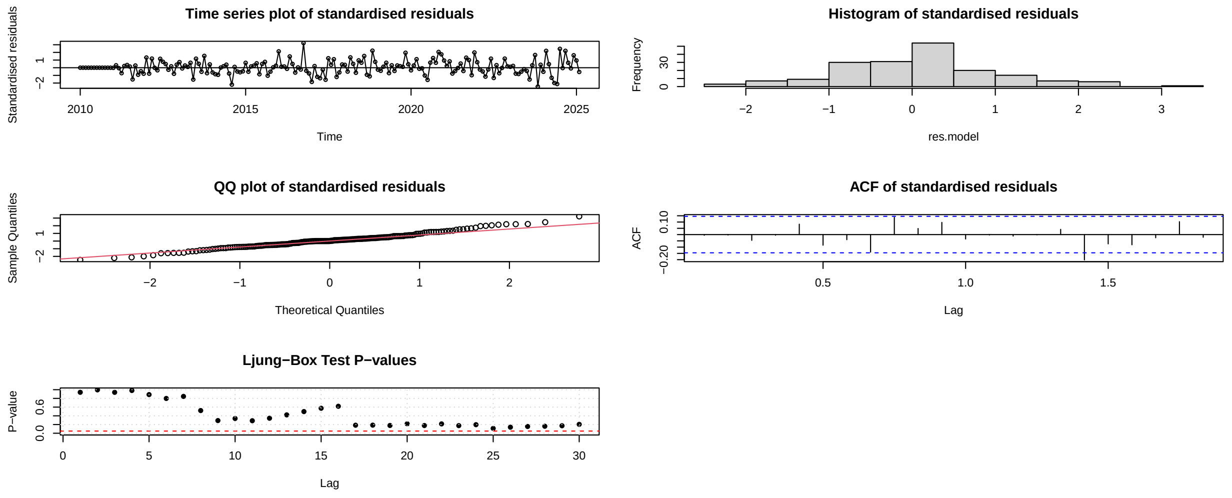
```
## [1] "Coefficient of 312-ML"
```

```r
coeftest(m_312_ml)
```

```
##
## z test of coefficients:
##
##       Estimate Std. Error z value  Pr(>|z|)
## ar1   0.357345   0.375954  0.9505 0.3418574
## ar2   0.084096   0.253043  0.3323 0.7396337
## ar3  -0.145284   0.091368 -1.5901 0.1118135
## ma1  -0.588429   0.372339 -1.5804 0.1140246
## ma2  -0.215054   0.328549 -0.6546 0.5127519
## sar1 -0.176798   0.132011 -1.3393 0.1804848
## sma1 -0.469087   0.125282 -3.7442 0.0001809 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
residual.analysis(model = m_312_ml)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.98732, p-value = 0.1017
```



```
m_312_css = Arima(ts_data,order=c(3,1,2), seasonal=list(order=c(1,1,1), period=12), method = "CSS")
print("Coefficient of 312-CSS")
```

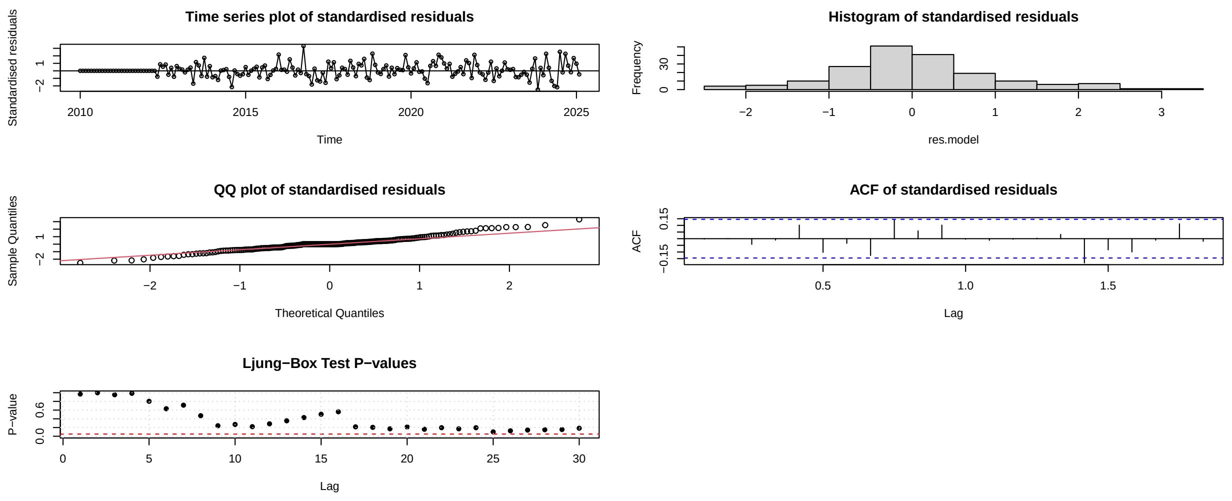
```
## [1] "Coefficient of 312-CSS"
```

```
coeftest(m_312_css)
```

```
##
## z test of coefficients:
##
##       Estimate Std. Error z value Pr(>|z|)
## ar1   0.392714   0.298805  1.3143 0.188752
## ar2   0.053359   0.197589  0.2701 0.787119
## ar3  -0.147136   0.089950 -1.6357 0.101893
## ma1  -0.612295   0.295624 -2.0712 0.038341 *
## ma2  -0.195453   0.254924 -0.7667 0.443253
## sar1 -0.237554   0.114067 -2.0826 0.037290 *
## sma1 -0.447696   0.115146 -3.8881 0.000101 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
residual.analysis(model = m_312_css)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.97533, p-value = 0.002588
```



```r
m_313_ml = Arima(ts_data,order=c(3,1,3), seasonal=list(order=c(1,1,1), period=12), method = "ML")
print("Coefficient of 313-ML")
```

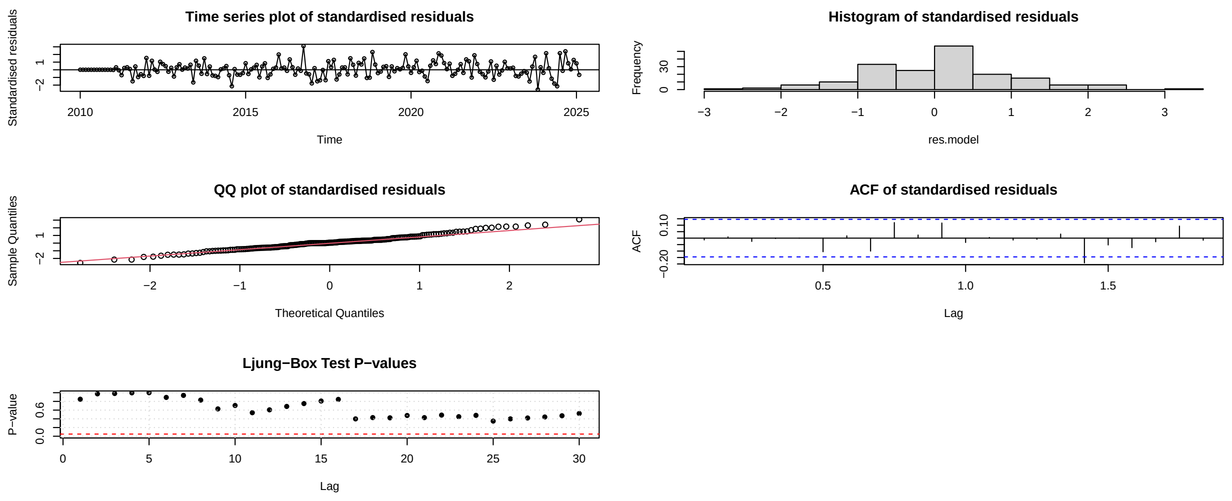
```
## [1] "Coefficient of 313-ML"
```

```r
coeftest(m_313_ml)
```

```
##
## z test of coefficients:
##
##       Estimate Std. Error z value  Pr(>|z|)
## ar1   0.66068    0.36271  1.8215 0.0685285 .
## ar2  -0.73857    0.37194 -1.9857 0.0470660 *
## ar3   0.30220    0.21456  1.4085 0.1589810
## ma1  -0.88939    0.34750 -2.5594 0.0104863 *
## ma2   0.67473    0.41803  1.6141 0.1065150
## ma3  -0.58410    0.21444 -2.7238 0.0064529 **
## sar1 -0.16348    0.14162 -1.1544 0.2483449
## sma1 -0.44959    0.12879 -3.4908 0.0004816 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
residual.analysis(model = m_313_ml)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.98936, p-value = 0.192
```



```
m_313_css = Arima(ts_data,order=c(3,1,3), seasonal=list(order=c(1,1,1), period=12), method = "CSS")
print("Coefficient of 313-CSS")
```

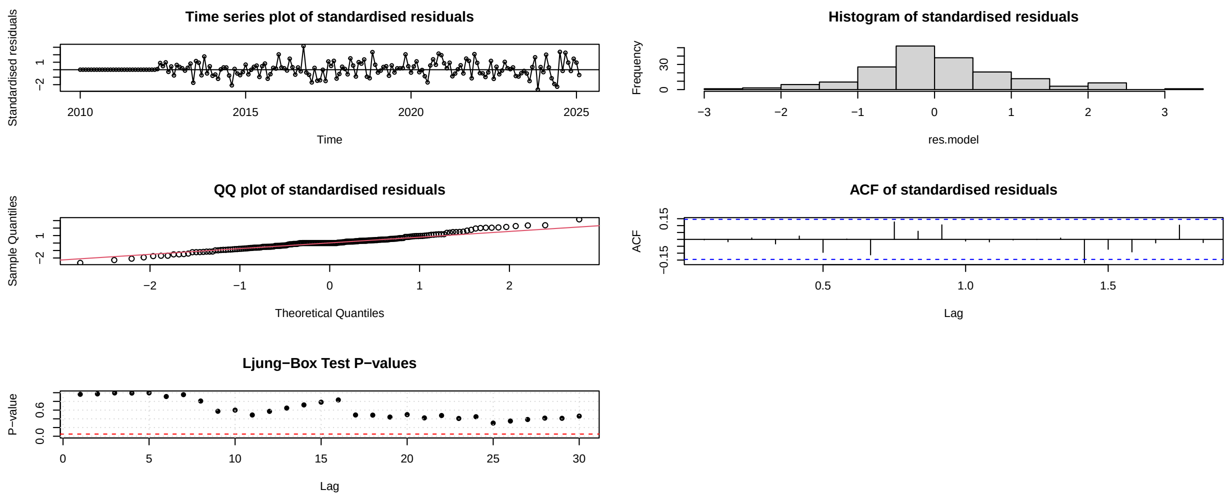
```
## [1] "Coefficient of 313-CSS"
```

```
coeftest(m_313_css)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error z value  Pr(>|z|)
## ar1   0.31426    0.25972  1.2100 0.2262829
## ar2  -0.36835    0.23860 -1.5438 0.1226348
## ar3   0.13382    0.16495  0.8113 0.4172155
## ma1  -0.53459    0.24354 -2.1950 0.0281609 *
## ma2   0.24066    0.26745  0.8998 0.3682053
## ma3  -0.46337    0.18824 -2.4616 0.0138311 *
## sar1 -0.24442    0.11633 -2.1011 0.0356281 *
## sma1 -0.41334    0.11503 -3.5932 0.0003267 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
residual.analysis(model = m_313_css)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.98144, p-value = 0.01601
```



```r
m_611_ml = Arima(ts_data,order=c(6,1,1), seasonal=list(order=c(1,1,1), period=12), method = "ML")
print("Coefficient of 611-ML")
```

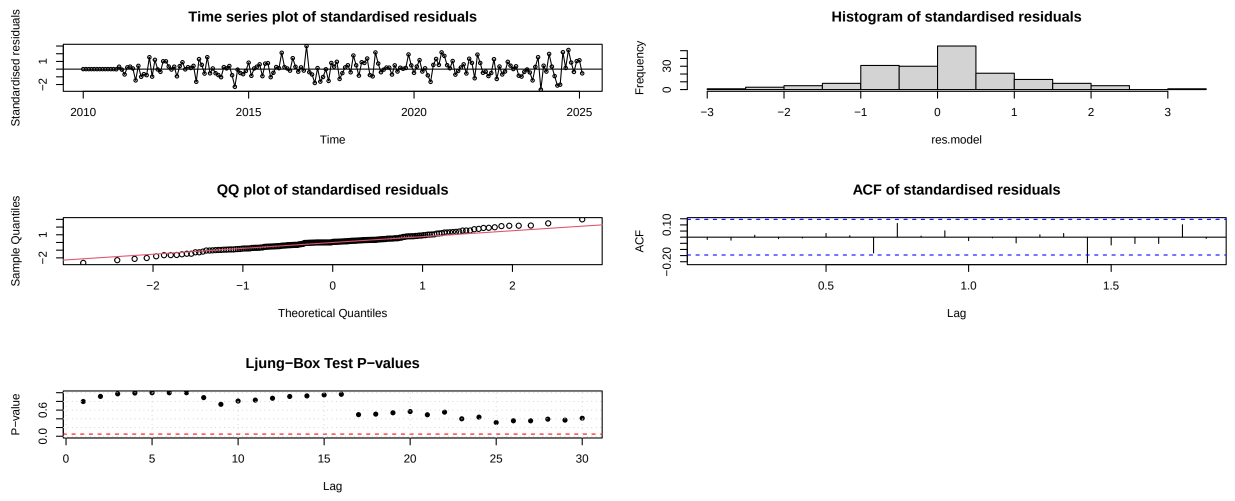
```
## [1] "Coefficient of 611-ML"
```

```r
coeftest(m_611_ml)
```

```
##
## z test of coefficients:
##
##        Estimate Std. Error z value  Pr(>|z|)
## ar1   0.494430   0.133941  3.6914 0.0002230 ***
## ar2  -0.092570   0.086893 -1.0653 0.2867257
## ar3  -0.196212   0.090572 -2.1664 0.0302834 *
## ar4   0.006372   0.091506  0.0696 0.9444844
## ar5   0.048731   0.087377  0.5577 0.5770402
## ar6  -0.206970   0.084059 -2.4622 0.0138085 *
## ma1  -0.713912   0.119600 -5.9692 2.385e-09 ***
## sar1 -0.178892   0.134650 -1.3286 0.1839895
## sma1 -0.493944   0.136201 -3.6266 0.0002872 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
residual.analysis(model = m_611_ml)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.98807, p-value = 0.1287
```



```r
m_611_css = Arima(ts_data,order=c(6,1,1), seasonal=list(order=c(1,1,1), period=12), method = "CSS")
print("Coefficient of 611-CSS")
```

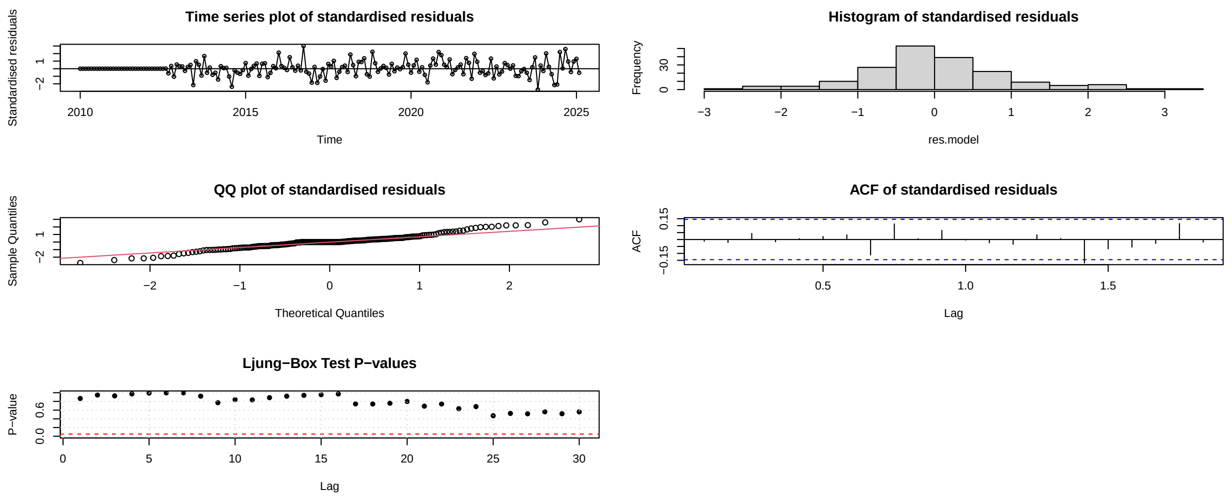
```
## [1] "Coefficient of 611-CSS"
```

```r
coeftest(m_611_css)
```

```
##
## z test of coefficients:
##
##       Estimate Std. Error z value  Pr(>|z|)
## ar1   0.453507   0.129573  3.5000 0.0004652 ***
## ar2  -0.122052   0.084067 -1.4518 0.1465478
## ar3  -0.224592   0.090233 -2.4890 0.0128092 *
## ar4  -0.014191   0.090790 -0.1563 0.8757948
## ar5   0.067468   0.083348  0.8095 0.4182427
## ar6  -0.256294   0.080882 -3.1687 0.0015311 **
## ma1  -0.665798   0.115259 -5.7766 7.624e-09 ***
## sar1 -0.263243   0.110829 -2.3752 0.0175384 *
## sma1 -0.428593   0.119342 -3.5913 0.0003290 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
residual.analysis(model = m_611_css)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.97035, p-value = 0.0006479
```



```
m_612_ml = Arima(ts_data,order=c(6,1,2), seasonal=list(order=c(1,1,1), period=12), method = "ML")
print("Coefficient of 612-ML")
```

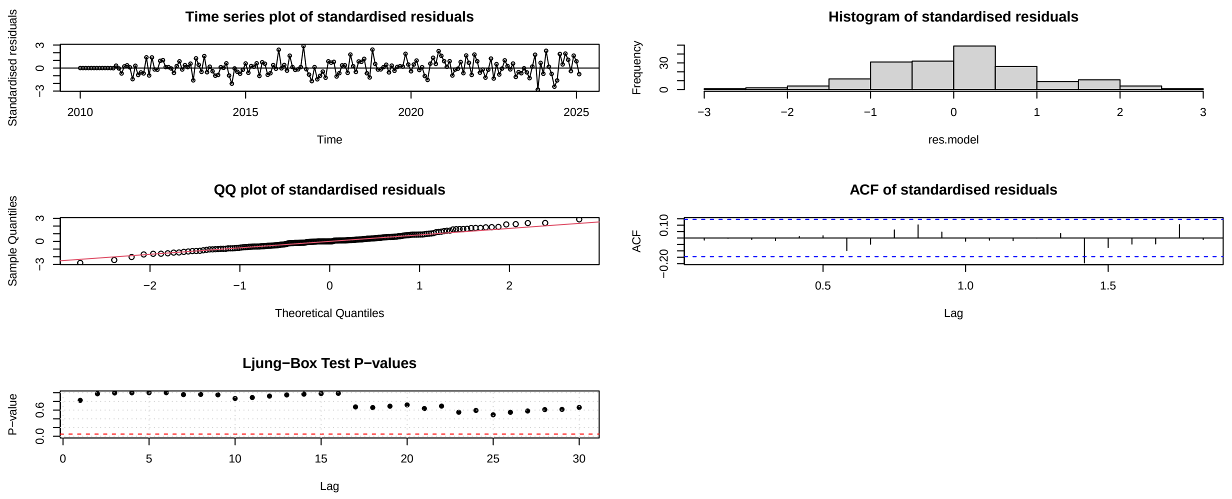
```
## [1] "Coefficient of 612-ML"
```

```
coeftest(m_612_ml)
```

```
##
## z test of coefficients:
##
##       Estimate Std. Error z value  Pr(>|z|)
## ar1  -0.265418   0.135541 -1.9582  0.050205 .
## ar2   0.467888   0.162726  2.8753  0.004036 **
## ar3  -0.223480   0.099567 -2.2445  0.024798 *
## ar4  -0.064691   0.102842 -0.6290  0.529327
## ar5   0.111645   0.118679  0.9407  0.346844
## ar6  -0.103978   0.103789 -1.0018  0.316432
## ma1   0.085266   0.121291  0.7030  0.482062
## ma2  -0.785208   0.142256 -5.5197 3.396e-08 ***
## sar1 -0.188330   0.136579 -1.3789  0.167924
## sma1 -0.426329   0.139326 -3.0599  0.002214 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
residual.analysis(model = m_612_ml)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.98874, p-value = 0.1588
```



```r
m_612_css = Arima(ts_data,order=c(6,1,2), seasonal=list(order=c(1,1,1), period=12), method = "CSS")
print("Coefficient of 612-CSS")
```

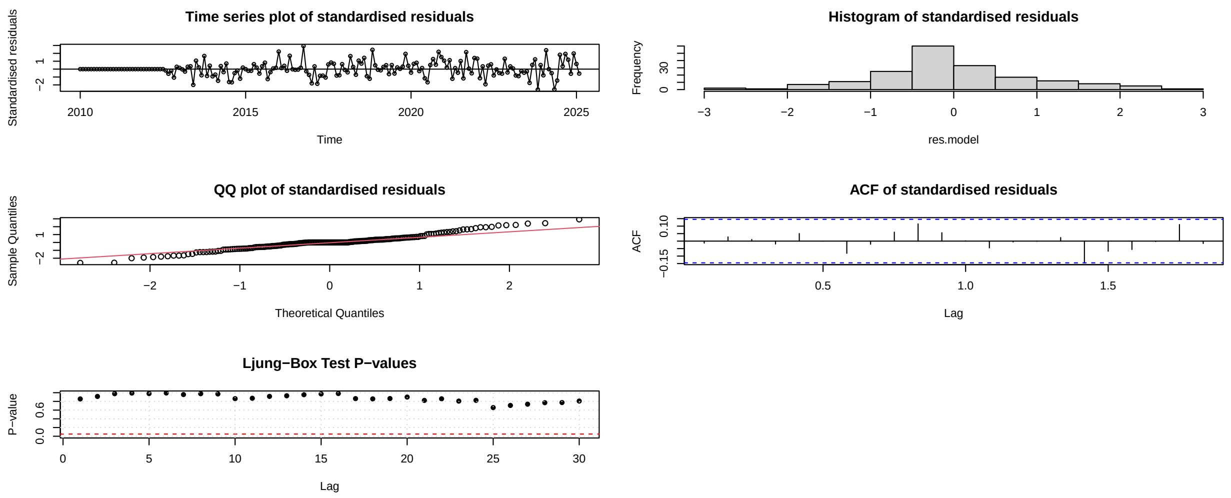
```
## [1] "Coefficient of 612-CSS"
```

```r
coeftest(m_612_css)
```

```
##
## z test of coefficients:
##
##       Estimate Std. Error z value  Pr(>|z|)
## ar1  -0.307028   0.106309 -2.8881 0.0038761 **
## ar2   0.427604   0.116869  3.6588 0.0002534 ***
## ar3  -0.286701   0.085619 -3.3486 0.0008123 ***
## ar4  -0.081452   0.091924 -0.8861 0.3755761
## ar5   0.100923   0.092842  1.0870 0.2770148
## ar6  -0.117049   0.074846 -1.5639 0.1178508
## ma1   0.218299   0.082926  2.6324 0.0084773 **
## ma2  -0.841494   0.089876 -9.3628 < 2.2e-16 ***
## sar1 -0.319597   0.084504 -3.7820 0.0001556 ***
## sma1 -0.311415   0.111380 -2.7960 0.0051744 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
residual.analysis(model = m_612_css)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.97115, p-value = 0.0008046
```



### Key findings include:

- SARIMA(0,1,3)(1,1,1)[12]: MA(1–3) and SMA(1) were statistically significant ($p < 0.05$), SAR(1) was not. Shapiro-Wilk p-value $= 0.151 \rightarrow$ Residuals approximately normal. Residual ACF showed no significant lags; Ljung-Box test p-values > 0.05.
- SARIMA(0,1,3)(0,1,1)[12]: All coefficients significant. Residuals slightly more normal and simpler model structure.
- SARIMA(2,1,2)(1,1,1)[12]: One of the lowest RMSE and MAPE scores. However, AR2 and MA2 were not consistently significant; residuals showed deviations from normality.
- SARIMA(1,1,1)(1,1,1)[12]: Simpler model with decent BIC but SAR(1) was not significant. Shapiro-Wilk $p = 0.034$, indicating slight deviation from normality.
- SARIMA(3,1,2), SARIMA(6,1,2): Showed overparameterization.Multiple insignificant coefficients and poor residual diagnostics.

  Model selection was guided by both information criteria and diagnostic measures.

AIC and BIC: These are model selection criteria that penalize model complexity. AIC (Akaike Information Criterion) balances fit and complexity, while BIC (Bayesian Information Criterion) imposes a stronger penalty for the number of parameters. Models with lower AIC or BIC are preferred, provided they do not overfit.

```
sc.AIC=AIC(m_011_ml, m_012_ml, m_013_ml, m_111_ml, m_112_ml, m_211_ml, m_212_ml, m_311_ml, m_312_ml,
m_313_ml, m_611_ml, m_612_ml)
sc.BIC=AIC(m_011_ml, m_012_ml, m_013_ml, m_111_ml, m_112_ml, m_211_ml, m_212_ml, m_311_ml, m_312_ml,
m_313_ml, m_611_ml, m_612_ml,
           k = log(length(ts_data)))

sort.score(sc.AIC, score = "aic")
```

```
##          df      AIC
## m_013_ml  6 2485.100
## m_212_ml  7 2486.670
## m_612_ml 11 2487.452
## m_211_ml  6 2487.598
## m_111_ml  5 2487.914
## m_311_ml  7 2488.031
## m_611_ml 10 2488.188
## m_112_ml  6 2488.229
## m_313_ml  9 2489.582
## m_312_ml  8 2489.726
## m_012_ml  5 2495.087
## m_011_ml  4 2504.064
```

```r
sort.score(sc.BIC, score = "aic")
```

```
##          df      AIC
## m_111_ml  5 2503.934
## m_013_ml  6 2504.324
## m_211_ml  6 2506.822
## m_112_ml  6 2507.453
## m_212_ml  7 2509.098
## m_311_ml  7 2510.459
## m_012_ml  5 2511.107
## m_312_ml  8 2515.358
## m_011_ml  4 2516.880
## m_313_ml  9 2518.418
## m_611_ml 10 2520.228
## m_612_ml 11 2522.696
```

Error Metrics: RMSE (Root Mean Square Error), MAE (Mean Absolute Error), MAPE (Mean Absolute Percentage Error), and MASE (Mean Absolute Scaled Error) were used to assess model accuracy on in-sample predictions. Lower values indicate better performance.

```r
Sm_011_ml <- accuracy(m_011_ml)[1:7]
Sm_012_ml <- accuracy(m_012_ml)[1:7]
Sm_013_ml <- accuracy(m_013_ml)[1:7]
Sm_111_ml <- accuracy(m_111_ml)[1:7]
Sm_112_ml <- accuracy(m_112_ml)[1:7]
Sm_211_ml <- accuracy(m_211_ml)[1:7]
Sm_212_ml <- accuracy(m_212_ml)[1:7]
Sm_311_ml <- accuracy(m_311_ml)[1:7]
Sm_312_ml <- accuracy(m_312_ml)[1:7]
Sm_313_ml <- accuracy(m_313_ml)[1:7]
Sm_611_ml <- accuracy(m_611_ml)[1:7]
Sm_612_ml <- accuracy(m_612_ml)[1:7]


df.Smodels <- data.frame(
  rbind(Sm_011_ml, Sm_012_ml, Sm_013_ml,
        Sm_111_ml, Sm_112_ml, Sm_211_ml,
        Sm_212_ml, Sm_311_ml, Sm_312_ml,
        Sm_313_ml, Sm_611_ml, Sm_612_ml
        )
)
colnames(df.Smodels) <- c("ME", "RMSE", "MAE", "MPE", "MAPE",
                          "MASE", "ACF1")
rownames(df.Smodels) <- c("SARIMA(0,1,1)x(1,1,1)_12",
                          "SARIMA(0,1,2)x(1,1,1)_12",
                          "SARIMA(0,1,3)x(1,1,1)_12",
              "SARIMA(1,1,1)x(1,1,1)_12",
                          "SARIMA(1,1,2)x(1,1,1)_12",
                          "SARIMA(2,1,1)x(1,1,1)_12",
                          "SARIMA(2,1,2)x(1,1,1)_12",
                          "SARIMA(3,1,1)x(1,1,1)_12",
                          "SARIMA(3,1,2)x(1,1,1)_12",
                          "SARIMA(3,1,3)x(1,1,1)_12",
                          "SARIMA(6,1,1)x(1,1,1)_12",
                          "SARIMA(6,1,2)x(1,1,1)_12"
                          )
round(df.Smodels,  digits = 3)
```

```
##                              ME    RMSE     MAE    MPE  MAPE  MASE   ACF1
## SARIMA(0,1,1)x(1,1,1)_12 10.520 368.891 277.727 -0.262 6.726 0.443  0.023
## SARIMA(0,1,2)x(1,1,1)_12 20.564 357.887 266.677 -0.231 6.519 0.425  0.068
## SARIMA(0,1,3)x(1,1,1)_12 26.345 345.052 259.910 -0.064 6.370 0.414  0.003
## SARIMA(1,1,1)x(1,1,1)_12 29.670 349.523 259.313 -0.025 6.314 0.413  0.057
## SARIMA(1,1,2)x(1,1,1)_12 29.063 347.736 257.766 -0.034 6.319 0.411 -0.001
## SARIMA(2,1,1)x(1,1,1)_12 28.491 347.106 257.582 -0.043 6.331 0.411 -0.020
## SARIMA(2,1,2)x(1,1,1)_12 21.410 342.862 254.377 -0.180 6.304 0.406 -0.032
## SARIMA(3,1,1)x(1,1,1)_12 26.867 345.593 259.591 -0.063 6.386 0.414 -0.003
## SARIMA(3,1,2)x(1,1,1)_12 26.830 345.395 259.672 -0.060 6.382 0.414 -0.006
## SARIMA(3,1,3)x(1,1,1)_12 27.311 343.379 259.856 -0.035 6.430 0.414 -0.014
## SARIMA(6,1,1)x(1,1,1)_12 25.533 339.422 254.553 -0.065 6.313 0.406 -0.019
## SARIMA(6,1,2)x(1,1,1)_12 24.470 336.211 253.203 -0.073 6.195 0.404 -0.016
```

Among all candidates, **SARIMA(2,1,2)(1,1,1)[12]** yielded the lowest error metrics but included non-significant coefficients and failed some residual tests, making it less robust.

SARIMA(0,1,3)(1,1,1)[12] provided a balanced performance with significant non-seasonal coefficients and acceptable diagnostics. However, the seasonal AR(1) term was not significant.

To simplify the model and adhere to the principle of parsimony, overparameterized models like SARIMA(3,1,2) and SARIMA(6,1,2) were excluded due to poor coefficient significance and degraded residual behavior.

After identifying **SARIMA(0,1,3)(1,1,1)[12]** as a strong candidate based on coefficient significance, residual diagnostics, and error metrics, further analysis was conducted to evaluate whether model performance could be enhanced through overparameterization. This involved deliberately increasing the non-seasonal AR and MA orders beyond the chosen model to assess any improvement in fit or forecast accuracy.

Two extended models were tested:

- **SARIMA(1,1,3)(1,1,1)[12]**
- **SARIMA(0,1,4)(1,1,1)[12]**

```
m_113_ml = Arima(ts_data,order=c(1,1,3), seasonal=list(order=c(1,1,1), period=12), method = "ML")
print("Coefficient of 113-ML")
```

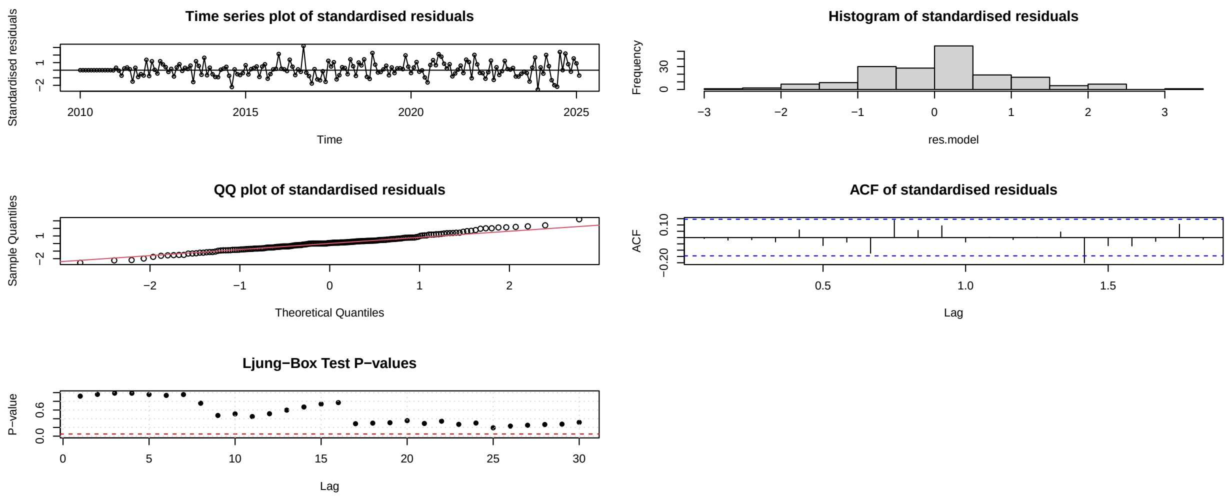
```
## [1] "Coefficient of 113-ML"
```

```r
coeftest(m_113_ml)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error z value  Pr(>|z|)
## ar1   0.12001    0.24192  0.4961 0.6198491
## ma1  -0.35206    0.23092 -1.5246 0.1273591
## ma2  -0.16548    0.10382 -1.5938 0.1109708
## ma3  -0.25324    0.10811 -2.3425 0.0191572 *
## sar1 -0.18050    0.13352 -1.3518 0.1764277
## sma1 -0.45122    0.12731 -3.5442 0.0003939 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
residual.analysis(model = m_113_ml)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.98846, p-value = 0.1453
```



```r
m_014_ml = Arima(ts_data,order=c(0,1,4), seasonal=list(order=c(1,1,1), period=12), method = "ML")
print("Coefficient of 014-ML")
```

```
## [1] "Coefficient of 014-ML"
```

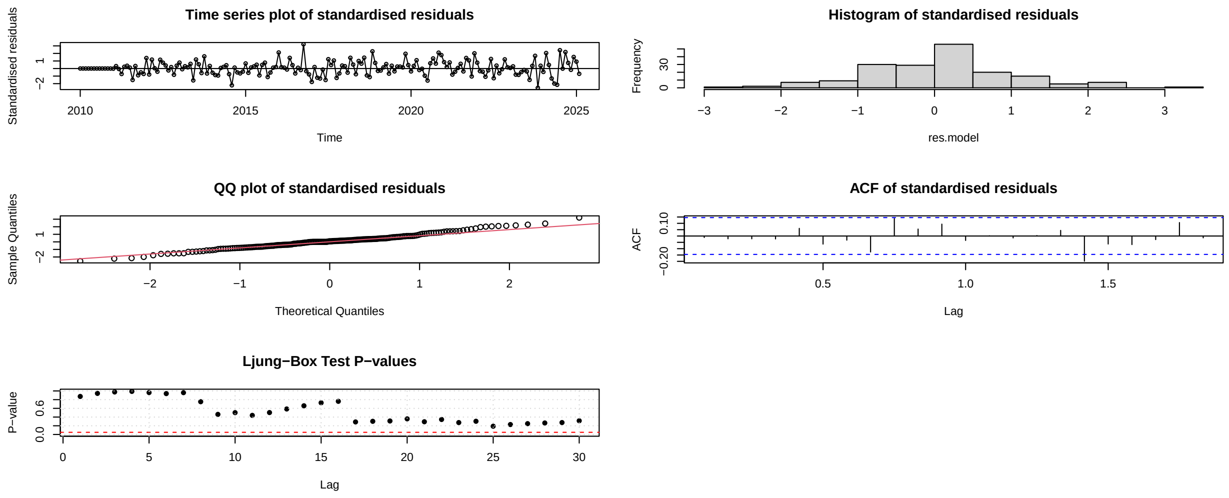
```r
coeftest(m_014_ml)
```

```
##
## z test of coefficients:
##
```

```
##        Estimate Std. Error z value  Pr(>|z|)
## ma1  -0.226727   0.080076 -2.8314 0.0046346 **
## ma2  -0.192617   0.077338 -2.4906 0.0127531 *
## ma3  -0.271693   0.081283 -3.3426 0.0008301 ***
## ma4  -0.050266   0.086676 -0.5799 0.5619572
## sar1 -0.178092   0.133873 -1.3303 0.1834170
## sma1 -0.455354   0.128074 -3.5554 0.0003774 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
residual.analysis(model = m_014_ml)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.98853, p-value = 0.1487
```



The rationale was to determine whether the inclusion of additional lags would capture any residual structure unaccounted for by the original model. However, the results showed that:

- The added parameters were not statistically significant (p-values > 0.05).
- Residual diagnostics did not improve, with Shapiro-Wilk and Ljung-Box p-values either remaining similar or slightly worsening.
- The increased complexity introduced risk of overfitting and reduced interpretability.

Therefore, these models were deemed overparameterized and were excluded from final consideration based on the principle of parsimony.

During the evaluation of SARIMA(0,1,3)(1,1,1)[12], the seasonal AR(1) component (SAR1) was observed to be statistically insignificant. To test the necessity of this component, a simplified model — SARIMA(0,1,3)(0,1,1)[12] was fitted by removing SAR1.

```r
m_013_ml_0AR = Arima(ts_data,order=c(0,1,3), seasonal=list(order=c(0,1,1), period=12), method = "ML")
print("Coefficient of 013-ML")
```

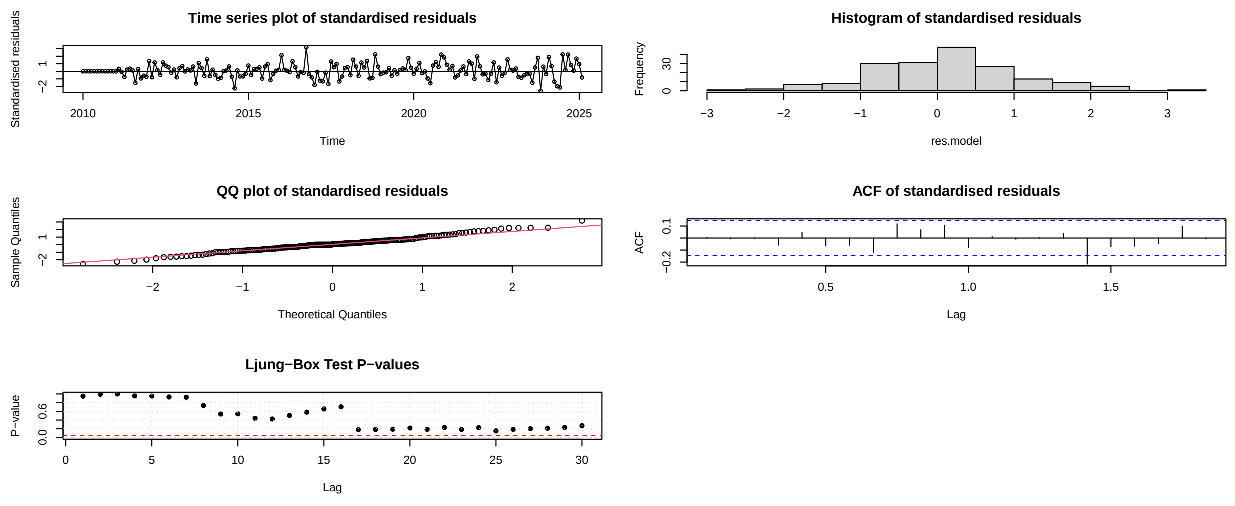
```
## [1] "Coefficient of 013-ML"
```

```r
coeftest(m_013_ml_0AR)
```

```
##
## z test of coefficients:
##
##       Estimate Std. Error z value  Pr(>|z|)
## ma1  -0.244953   0.073917 -3.3139 0.0009200 ***
## ma2  -0.197377   0.079977 -2.4679 0.0135902 *
## ma3  -0.285888   0.076111 -3.7562 0.0001725 ***
## sma1 -0.572835   0.069744 -8.2134 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
residual.analysis(model = m_013_ml_0AR)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.99071, p-value = 0.2874
```



All coefficients in this simplified model were statistically significant. Residual diagnostics improved: Shapiro-Wilk test p = 0.28 (suggesting normality), and Ljung-Box test p-values > 0.05 (no autocorrelation). The model structure became simpler and more interpretable, enhancing transparency without compromising diagnostic quality.

```r
Sm_013_ml <- accuracy(m_013_ml)[1:7]
Sm_013_ml_0AR <- accuracy(m_013_ml_0AR)[1:7]
```

```
df.Smodels <- data.frame(
  rbind(Sm_013_ml, Sm_013_ml_0AR
        )
)
colnames(df.Smodels) <- c("ME", "RMSE", "MAE", "MPE", "MAPE",
                          "MASE", "ACF1")
rownames(df.Smodels) <- c(
                          "SARIMA(0,1,3)x(1,1,1)_12",
              "SARIMA(0,1,3)x(0,1,1)_12"
                          )
round(df.Smodels,  digits = 3)
```

```
##                               ME    RMSE     MAE    MPE  MAPE  MASE  ACF1
## SARIMA(0,1,3)x(1,1,1)_12 26.345 345.052 259.910 -0.064 6.370 0.414 0.003
## SARIMA(0,1,3)x(0,1,1)_12 28.848 347.033 262.833 -0.034 6.444 0.419 0.005
```

The final model selection process was guided by multiple criteria including:

- Information Criteria (AIC, BIC) to assess model fit and penalize complexity
- Statistical Significance of Coefficients to ensure each parameter meaningfully contributes
- Residual Diagnostics (Shapiro-Wilk test, Ljung-Box test, residual ACF, QQ plots) to confirm white noise behavior
- In-sample Error Metrics (ME, RMSE, MAE, MPE, MAPE, MASE, ACF1) to assess prediction accuracy and bias

Information Criteria:

- SARIMA(0,1,3)(1,1,1)[12]: AIC = 2485.10, BIC = 2503.88
- SARIMA(0,1,3)(0,1,1)[12]: AIC = 2484.88, BIC = 2500.53

While both models have similar BIC values, the simplified model actually has slightly lower AIC, suggesting marginally better fit even with fewer parameters.

Coefficient Significance:

- SARIMA(0,1,3)(1,1,1)[12] includes a seasonal AR(1) term (SAR1) which was not statistically significant.
- SARIMA(0,1,3)(0,1,1)[12] showed all coefficients to be significant, improving interpretability and adherence to the principle of parsimony.

Residual Diagnostics:

- Normality:
  - SARIMA(0,1,3)(1,1,1)[12]: Shapiro-Wilk p = 0.151
  - SARIMA(0,1,3)(0,1,1)[12]: Shapiro-Wilk p = 0.280 → Indicates better normality in the simplified model.

- Autocorrelation (ACF1):
  - SARIMA(0,1,3)(1,1,1)[12]: ACF1 = 0.003
  - SARIMA(0,1,3)(0,1,1)[12]: ACF1 = 0.005 → Both are very low, indicating residuals are approximately white noise.

Ljung-Box test p-values for both models exceed 0.05 at multiple lags, affirming independence in residuals.

While **SARIMA(0,1,3)(1,1,1)[12]** has marginally lower error metrics, it also includes a non-significant seasonal AR(1) parameter, which reduces the interpretability and increases complexity without providing substantial benefit.

In contrast, **SARIMA(0,1,3)(0,1,1)[12]** has all parameters statistically significant, demonstrates better residual normality, offers comparable error metrics and AIC, and aligns with the principle of parsimony, favoring simpler models with similar predictive performance.

```r
par(mfrow=c(1,1))
m1_013.f0 = Arima(ts_data,order=c(0,1,3),
                     seasonal=list(order=c(0,1,1), period=12),
                     method = "ML")
m1_013.f0.Afrc = forecast::forecast(m1_013.f0, h = 10)
plot(m1_013.f0.Afrc)
lines(fitted(m1_013.f0.Afrc), col= "red")
legend("topleft", lty=1, pch=1, col=c("red","black","blue"), text.width = 4, c("Data", "Fitted", "Foreca
```

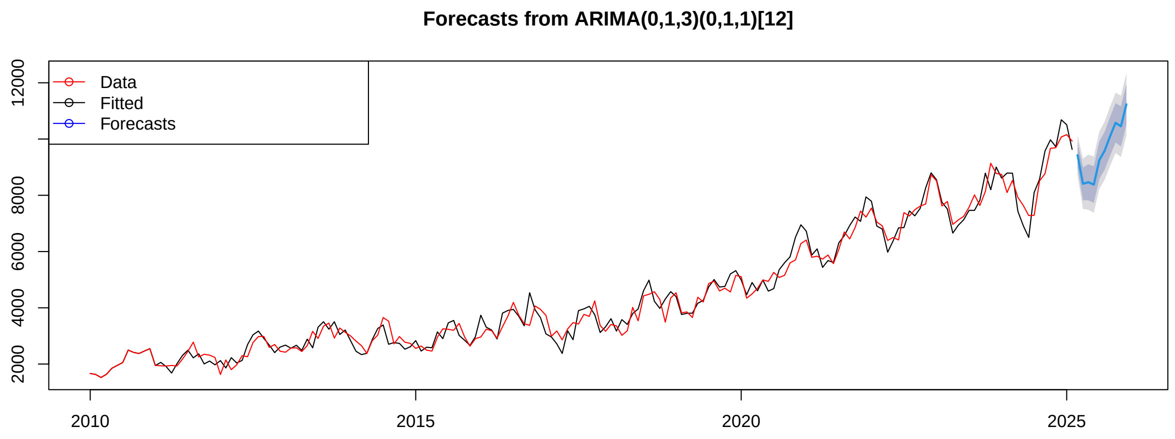


Figure 18: Figure 20: ARIMA(0,1,3),(0,1,1),1[2] Forecasting

```r
par(mfrow=c(1,1))
m1_013.f1 = Arima(ts_data,order=c(0,1,3),
                     seasonal=list(order=c(1,1,1), period=12),
                     method = "ML")
m1_013.f1.Afrc = forecast::forecast(m1_013.f1, h = 10)
plot(m1_013.f1.Afrc)
lines(fitted(m1_013.f1.Afrc), col= "red")
legend("topleft", lty=1, pch=1, col=c("red","black","blue"), text.width = 4, c("Data", "Fitted", "Foreca
```

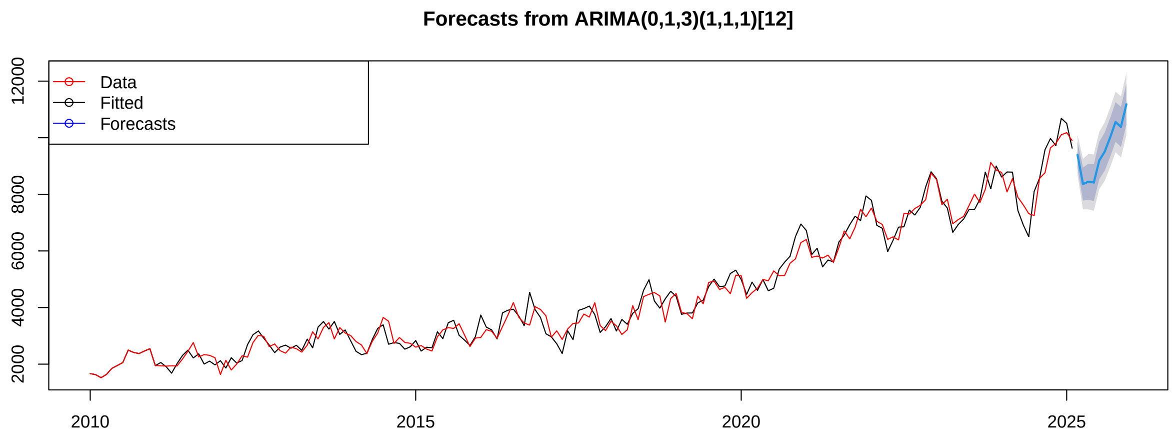


Figure 19: Figure 21: ARIMA(0,1,3),(1,1,1),1[2] Forecasting

Beyond statistical metrics and diagnostics, visual inspection of the forecast plots for both SARIMA(0,1,3)(1,1,1)[12] and SARIMA(0,1,3)(0,1,1)[12] was conducted to evaluate their predictive behavior.

Both models produced forecasts that closely follow the historical seasonal trend, show appropriate confidence interval widening across the forecast horizon, display no unexpected jumps or instability.

However, the forecast trajectories and bounds were nearly identical for both models, indicating that removing the seasonal AR(1) component did not lead to any substantial deterioration in forecast quality. The model SARIMA(0,1,3)(0,1,1)[12], despite its slightly higher RMSE and MAE, preserved the structural integrity of seasonal patterns while maintaining smoother forecasts with statistically significant parameters.

Conclusion

As part of the exploratory modeling process, several traditional regression-based models were tested to understand their ability to represent the structure of Australia's monthly renewable electricity production. The linear model, while providing a simple interpretation of the long-term upward trend, failed to capture the nonlinear growth and completely overlooked the recurring seasonal fluctuations present in the data. Progressing to a quadratic model allowed us to better fit the curvature in the trend, addressing the nonlinearity to some extent, yet it too lacked any mechanism to represent seasonality. The cubic model offered improved flexibility in capturing complex nonlinear trends, but still could not account for the evident seasonal cycles.

To explicitly address the periodic nature of the data, the cyclic model was introduced using the month as a categorical variable. This successfully captured seasonal variation but disregarded the long-term upward trend, which is a critical component of this time series. The seasonal model, which combined time (to model trend) and month (to model seasonality), provided the most comprehensive fit among regression-based approaches. It addressed both major components—trend and seasonality—effectively, but lacked the capacity to model residual autocorrelation or incorporate stochastic patterns over time. As a result, its forecasting capability remained limited compared to time-series-specific techniques.

In contrast, the SARIMA model, particularly SARIMA(0,1,3)(0,1,1)[12], emerged as a superior modeling approach. Unlike the previous models, SARIMA is explicitly designed for time series forecasting and handles non-stationarity through differencing, seasonality via seasonal AR and MA terms, and autocorrelation through non-seasonal ARMA components. This model demonstrated strong statistical properties: all coefficients were significant, residuals passed normality and independence checks, and diagnostic tests such as the Shapiro-Wilk and Ljung-Box tests confirmed the residuals resembled white noise. Forecast performance was stable and aligned well with the seasonal dynamics of the historical data.

Ultimately, the SARIMA model not only outperforms other approaches in terms of capturing the inherent structure of the data but also provides a robust forecasting framework. This makes it particularly well-suited to answering the research question: "How has Australia's renewable electricity production evolved over the past decade, and what do forecasts for the next 10 months suggest about future trends?" By accurately modeling both the trend and seasonal effects, while accounting for underlying temporal dependencies, the SARIMA model provides insightful and reliable projections that can inform policy, investment, and sustainability planning.

Following are the helper functions used in the analysis:

```r
sort.score <- function(x, score = c("bic", "aic")){
  if (score == "aic"){
    x[with(x, order(AIC)),]
  } else if (score == "bic") {
    x[with(x, order(BIC)),]
  } else {
    warning('score = "x" only accepts valid arguments ("aic","bic")')
  }
}

residual.analysis <- function(model, std = TRUE,start = 2, lag.max = 30, class = c("ARIMA","SARIMA", "G
  library(TSA)
  # library(FitAR)
  if ((class == "ARIMA") | (class == "SARIMA")){
    if (std == TRUE){
      res.model = rstandard(model)
    }else{
      res.model = residuals(model)
    }
  }else if (class == "GARCH"){
    res.model = model$residuals[start:model$n.used]
  }else if (class == "garch"){
    res.model = model$residuals[start:model$n.used]
  }else if (class == "ARMA-GARCH"){
    res.model = model@fit$residuals
  }else if (class == "fGARCH"){
    res.model = model@residuals
  }else {
    stop("The argument 'class' must be either 'ARIMA' or 'GARCH' ")
  }
  par(mfrow=c(3,2))
  plot(res.model,type='o',ylab='Standardised residuals', main="Time series plot of standardised residua
  abline(h=0)
  hist(res.model,main="Histogram of standardised residuals")
  qqnorm(res.model,main="QQ plot of standardised residuals")
  qqline(res.model, col = 2)
  if (class == "SARIMA") {
    seasonal_acf(res.model,main="ACF of standardised residuals")
  } else {
    acf(res.model,main="ACF of standardised residuals")
  }
  print(shapiro.test(res.model))
  k=0
  LBQPlot(res.model, lag.max = lag.max, SquaredQ = FALSE)
  par(mfrow=c(1,1))
}

helper <- function(class = c("acf", "pacf"), ...) {

  # Capture additional arguments
  params <- match.call(expand.dots = TRUE)
  params <- as.list(params)[-1]

  # Calculate ACF/PACF values
  if (class == "acf") {
    acf_values <- do.call(acf, c(params, list(plot = FALSE)))
  } else if (class == "pacf") {
    acf_values <- do.call(pacf, c(params, list(plot = FALSE)))
  }

  # Extract values and lags
  acf_data <- data.frame(
    Lag = as.numeric(acf_values$lag),
    ACF = as.numeric(acf_values$acf)
  )

  # Identify seasonal lags to be highlighted
  seasonal_lags <- acf_data$Lag %% 1 == 0

  # Plot ACF/PACF values
  if (class == "acf") {
    do.call(acf, c(params, list(plot = TRUE)))
  } else if (class == "pacf") {
    do.call(pacf, c(params, list(plot = TRUE)))
  }

  # Add colored segments for seasonal lags
  for (i in which(seasonal_lags)) {
    segments(x0 = acf_data$Lag[i], y0 = 0, x1 = acf_data$Lag[i], y1 = acf_data$ACF[i], col = "red")
  }
}


# seasonal_acf -------------------------------------------------------------

seasonal_acf <- function(...) {
  helper(class = "acf", ...)
}


# seasonal_pacf ------------------------------------------------------------

seasonal_pacf <- function(...) {
  helper(class = "pacf", ...)
}

LBQPlot <- function(residuals, lag.max = 30, SquaredQ = FALSE) {
  # Calculate Ljung-Box p-values for each lag
  p.values <- numeric(lag.max)

  for (i in 1:lag.max) {
    if (SquaredQ) {
      p.values[i] <- Box.test(residuals^2, lag = i, type = "Ljung-Box")$p.value
    } else {
      p.values[i] <- Box.test(residuals, lag = i, type = "Ljung-Box")$p.value
    }
  }

  # Plot p-values as dots
  plot(1:lag.max, p.values, type = "p", pch = 16, col = "black",
       ylim = c(0, 1),
       xlab = "Lag", ylab = "P-value",
       main = "Ljung-Box Test P-values")

  # Add horizontal reference line at 0.05
  abline(h = 0.05, col = "red", lty = 2)

  # Optional: Add grid
  grid()
}
```