Analysis of Inpatient Admissions

Galih Fitriatmo

2025-01-31

Introduction

This project was carried out during my internship. The data used consists of inpatient admission records from a hospital covering the period from 07/03/2022 to 10/11/2023. However, to maintain institutional confidentiality, the data in this repository has been modified while retaining characteristics similar to the original data.

Load Library

```
library(stats)
library(tidyverse)
library(tseries)
library(forecast)
library(dplyr)
library(readxl)
library(MASS)
library(lmtest)
library(nortest)
library(readr)
```

Load Data

```
data <- read_csv("Data/data_pengunjung.csv", col_types = cols(
  Tanggal = col_date(format = "%d/%m/%Y"),
  Hari = col_character(),
  Value = col_double(),
  Subset = col_character()
))
print(data)
```

```
## # A tibble: 672 × 4
##    Tanggal    Hari  Value Subset
##    <date>     <chr> <dbl> <chr>
##  1 2022-03-07 Mon      48 Train
##  2 2022-03-08 Tue      96 Train
##  3 2022-03-09 Wed      84 Train
##  4 2022-03-10 Thu     126 Train
##  5 2022-03-11 Fri     102 Train
##  6 2022-03-12 Sat      84 Train
##  7 2022-03-13 Sun     108 Train
##  8 2022-03-14 Mon     126 Train
##  9 2022-03-15 Tue      90 Train
## 10 2022-03-16 Wed     114 Train
## # ℹ 662 more rows
```

Split Data

```
data_train <- data %>% filter(Subset == "Train")
data_fix <- ts(data_train$Value, frequency = 7, start = c(1, 1))
```

Descriptive Statistics

```
summary_stats <- data_train %>%
  summarize(
    Mean = mean(Value, na.rm = TRUE),
    Median = median(Value, na.rm = TRUE),
    Min = min(Value, na.rm = TRUE),
    Max = max(Value, na.rm = TRUE)
  )
summary_stats
```

```
## # A tibble: 1 × 4
##    Mean Median   Min   Max
##   <dbl>  <dbl> <dbl> <dbl>
## 1  123.    120    12   270
```

Based on the output, the average number of inpatient admissions at the hospital is 122.6645963 patients. The lowest number of admissions occurred on 2023-02-25 with 12 patients, while the highest occurred on 2023-05-03 with 270 patients.

Daily Patient arrivals

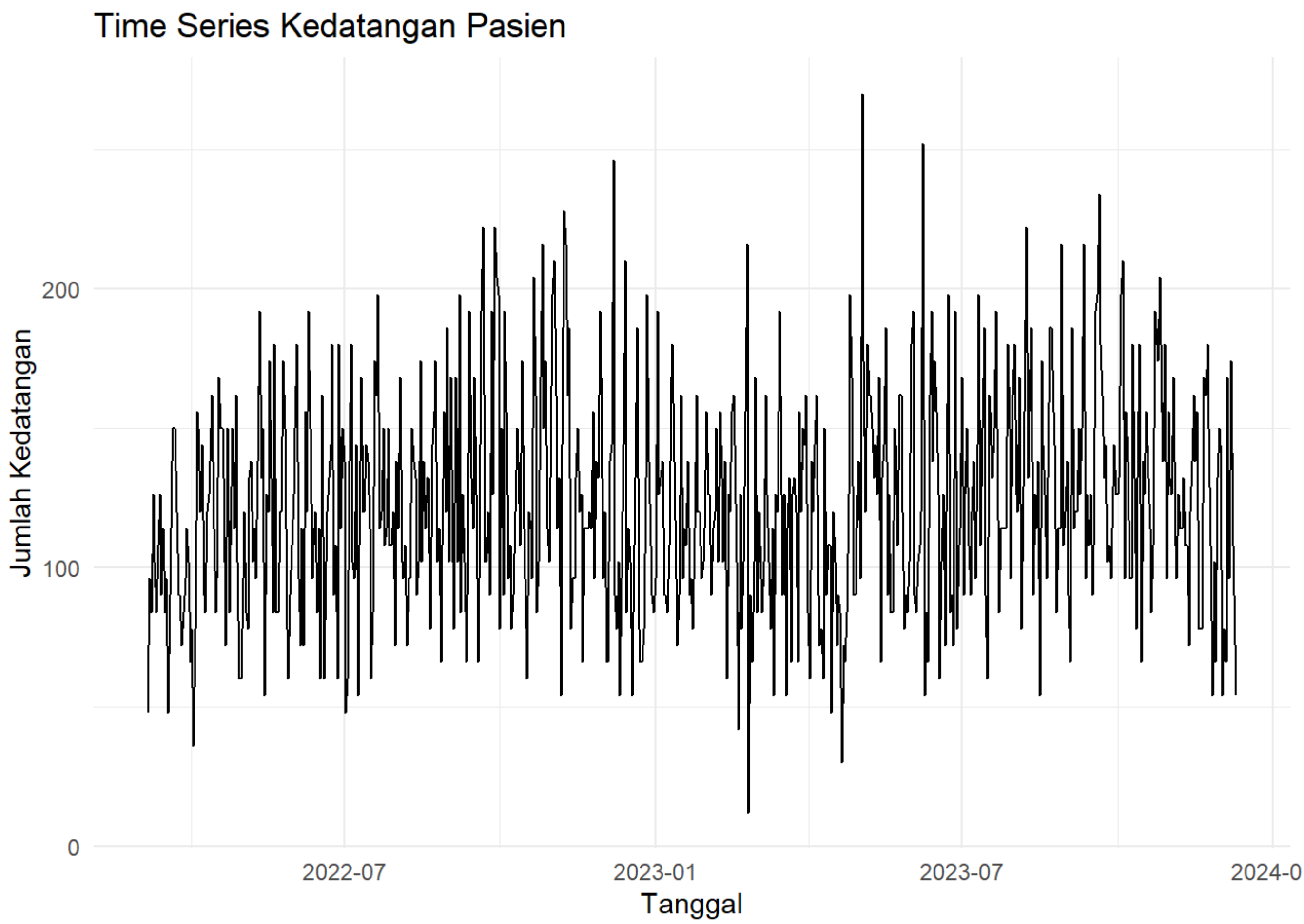
```
summary_by_day <- data_train %>%
  group_by(Hari) %>%
  summarize(Mean = mean(Value, na.rm = TRUE)) %>%
  arrange(factor(Hari, levels = c("Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun")))
summary_by_day
```

```
## # A tibble: 7 × 2
##   Hari   Mean
##   <chr> <dbl>
## 1 Mon   121.
## 2 Tue   140.
## 3 Wed   144.
## 4 Thu   132.
## 5 Fri   131.
## 6 Sat   104.
## 7 Sun    87.6
```

Based on the output, the highest average number of patient arrivals occurred on Wed and the lowest occurred on Sun.

Time Series Visualization

```
ggplot(data_train, aes(x = Tanggal, y = Value)) +
  geom_line() +
  labs(title = "Time Series Kedatangan Pasien", x = "Tanggal", y = "Jumlah Kedatangan") +
  theme_minimal()
```

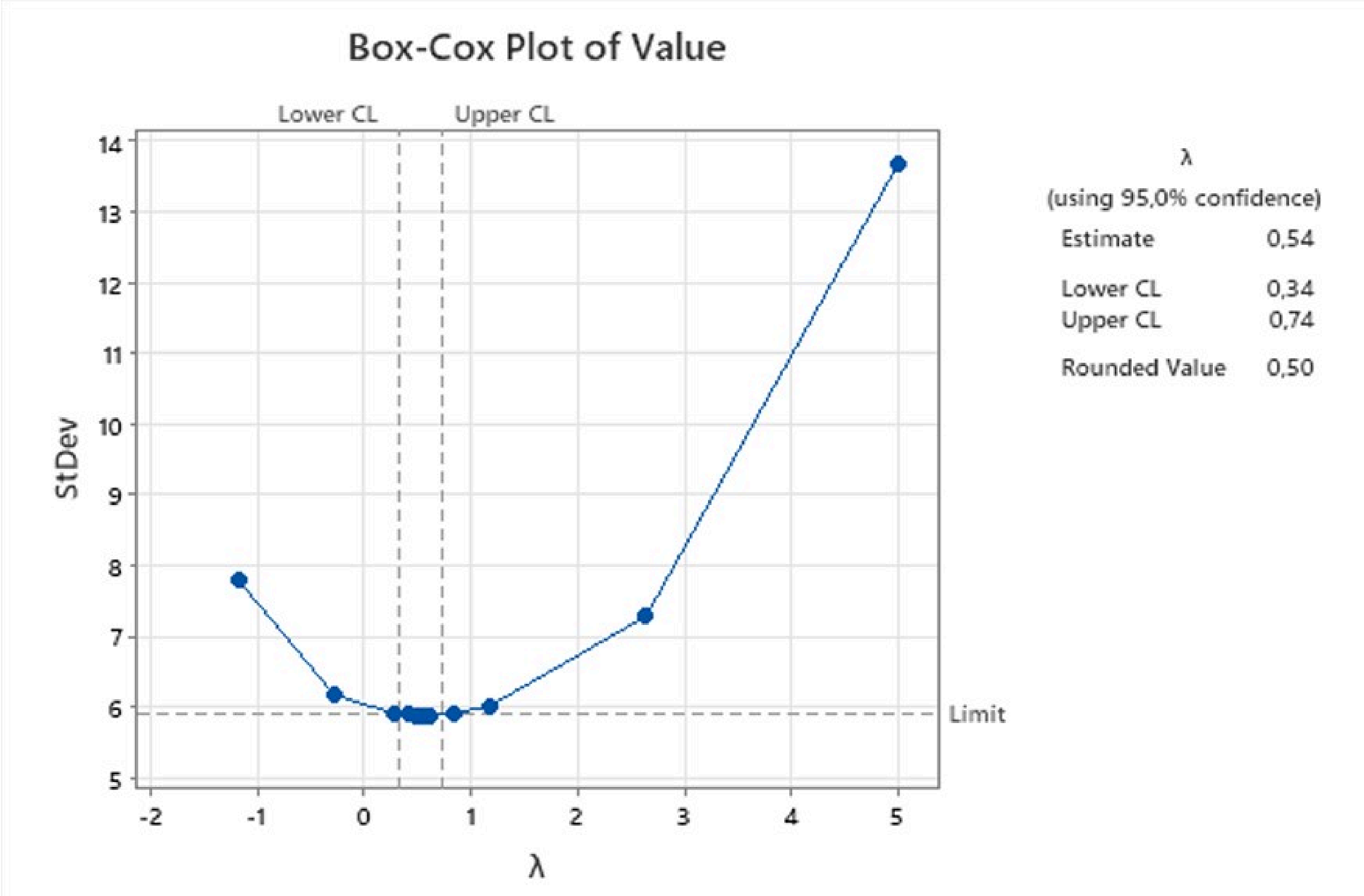


From the plot above, the number of inpatient admissions fluctuates significantly, but no clear trend is observed. It can also be seen that inpatient visits tend to follow a weekly cycle, rising and falling every seven days. The increase usually starts on Monday, peaks on Wednesday, and then begins to decline on Saturday. The lowest number of inpatient admissions typically occurs on Sunday, as some clinics are closed, preventing patient transfers from outpatient to inpatient care. This indicates a seasonal pattern in the number of inpatient admissions.

Stationarity Check

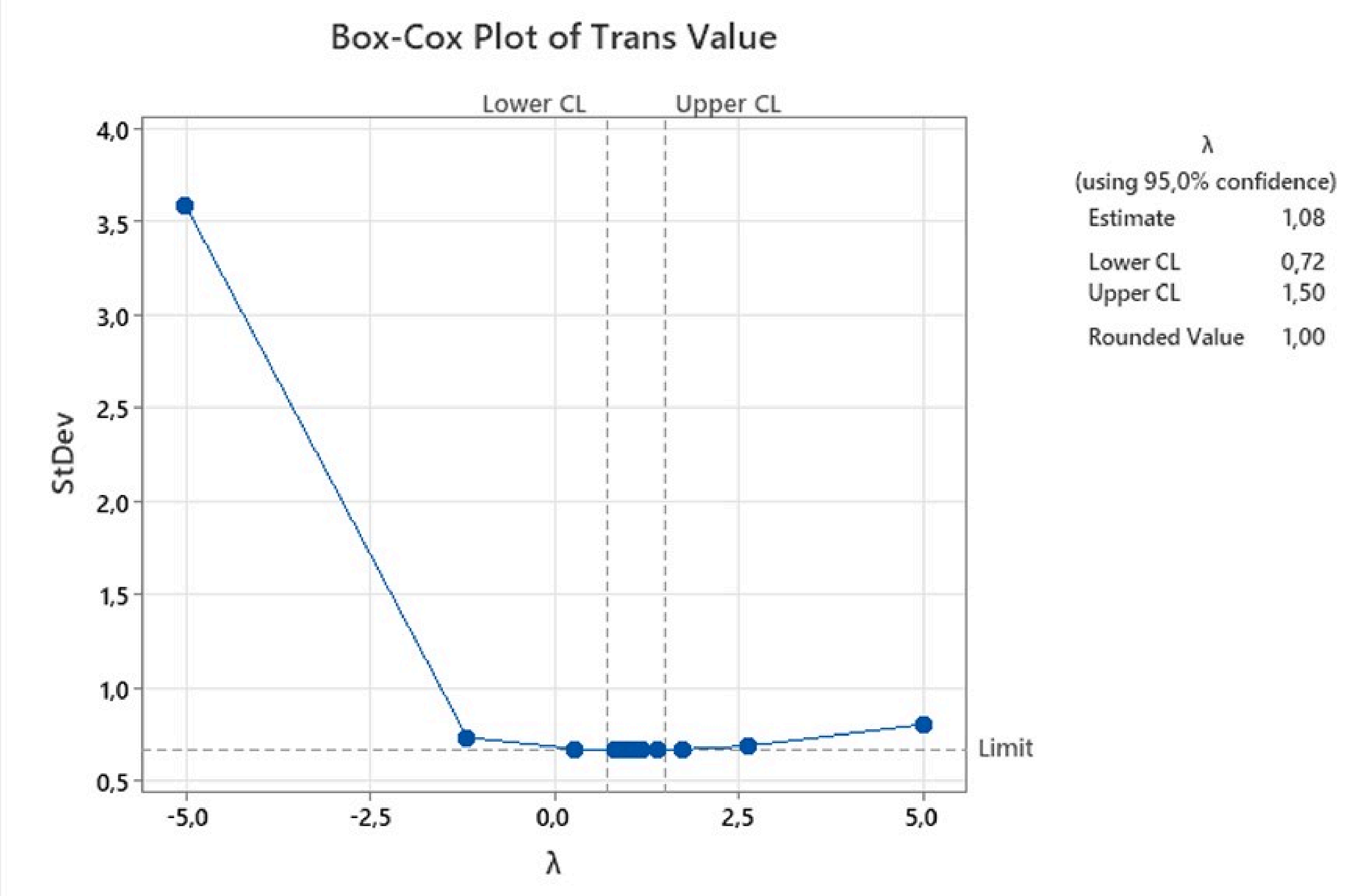
Stationarity Test on Variance

The stationarity check on variance is performed using Box-Cox transformation. The results are as follows



Based on the output, the rounded value obtained is 0.5, indicating the need for transformation. The transformation is performed using the formula $Zt = \sqrt{Yt}$, and the results are displayed in the output below.

```
lambda <- BoxCox.lambda(data_fix)
data_fix_transformed <- BoxCox(data_fix, lambda)
```

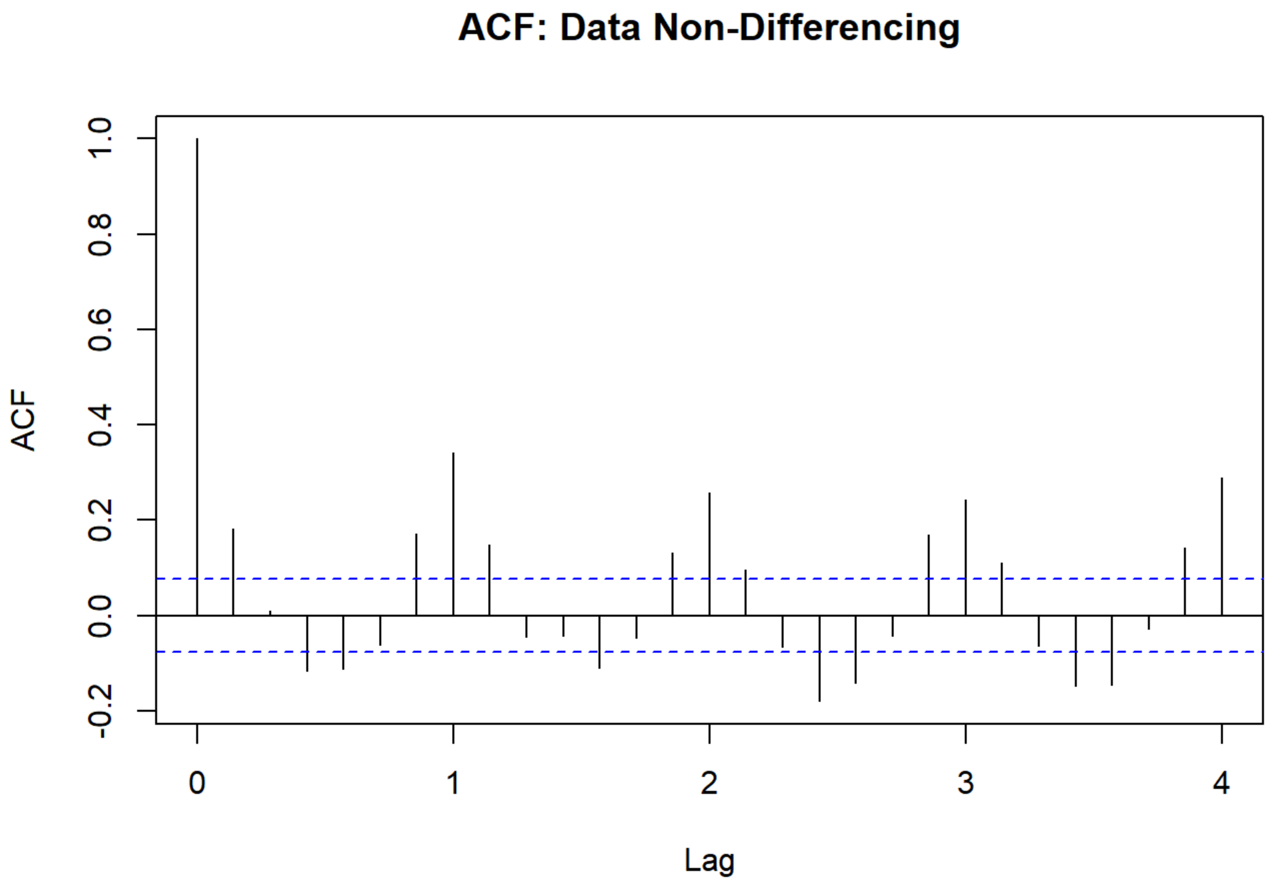


Based on the output, the rounded value obtained is 1, indicating that the transformed data is now stationary in variance. The next step is to check for stationarity in the mean.

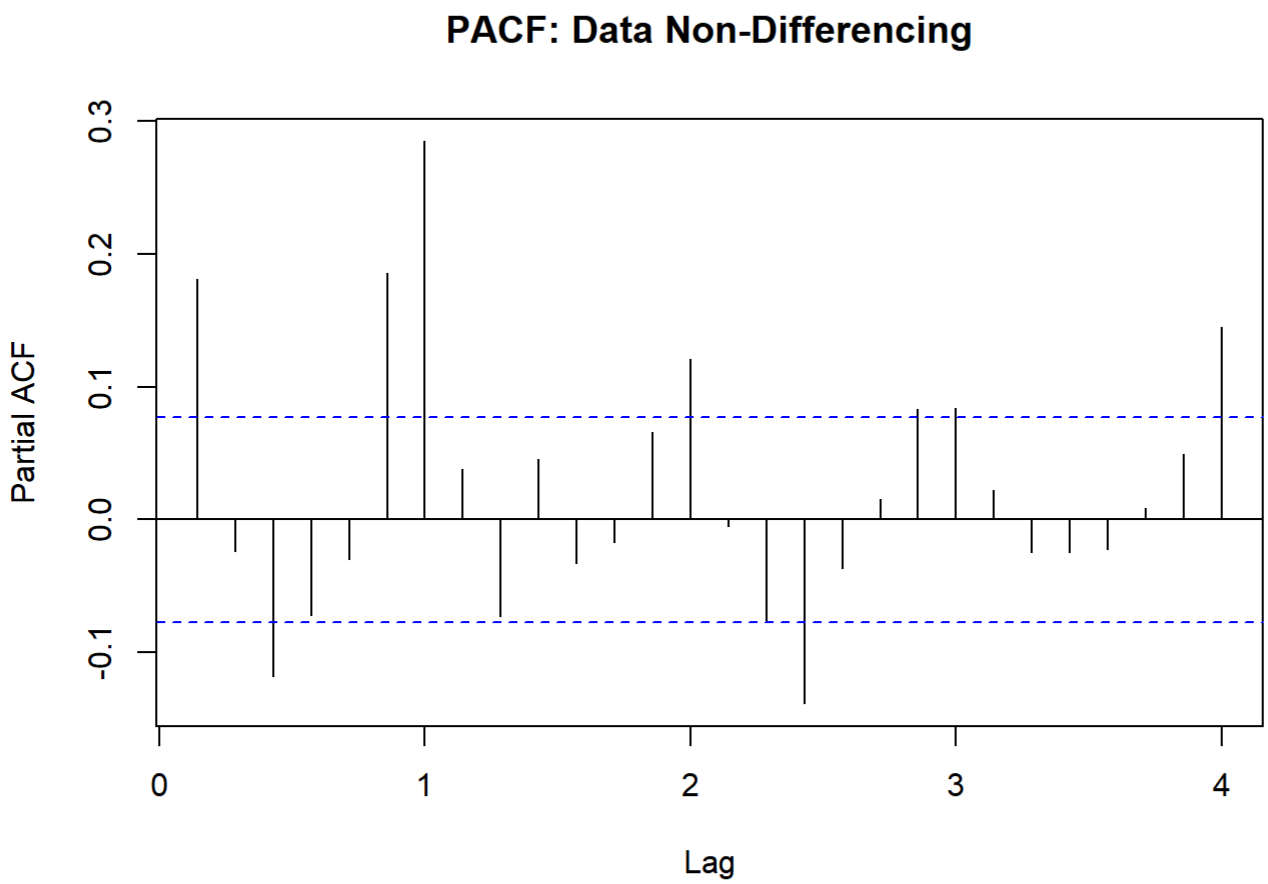
Stationarity Test on Mean

The stationarity check on the mean can be performed using the ACF and PACF plots of the transformed data.

```
acf(data_fix_transformed, main = "ACF: Data Non-Differencing")
```



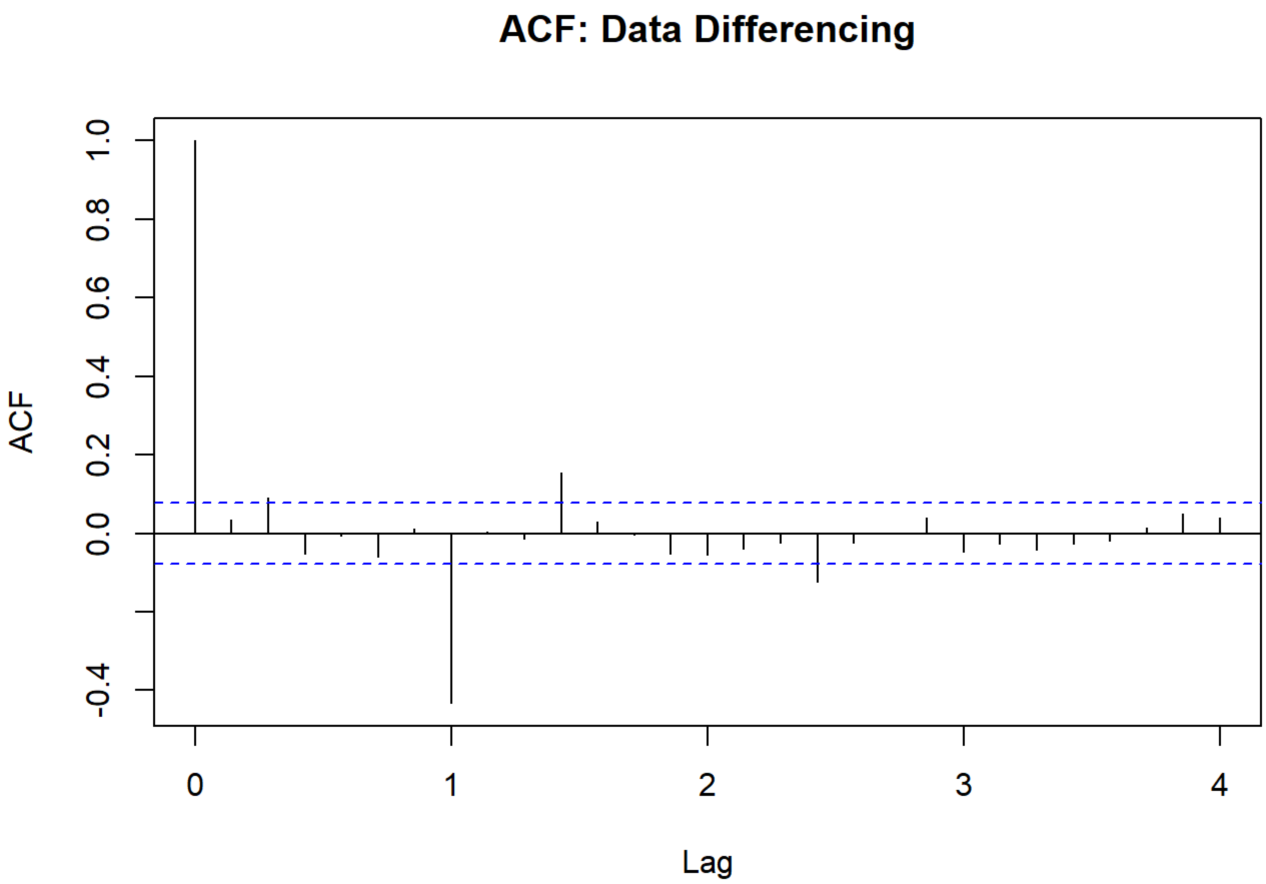

```
pacf(data_fix_transformed, main = "PACF: Data Non-Differencing")
```



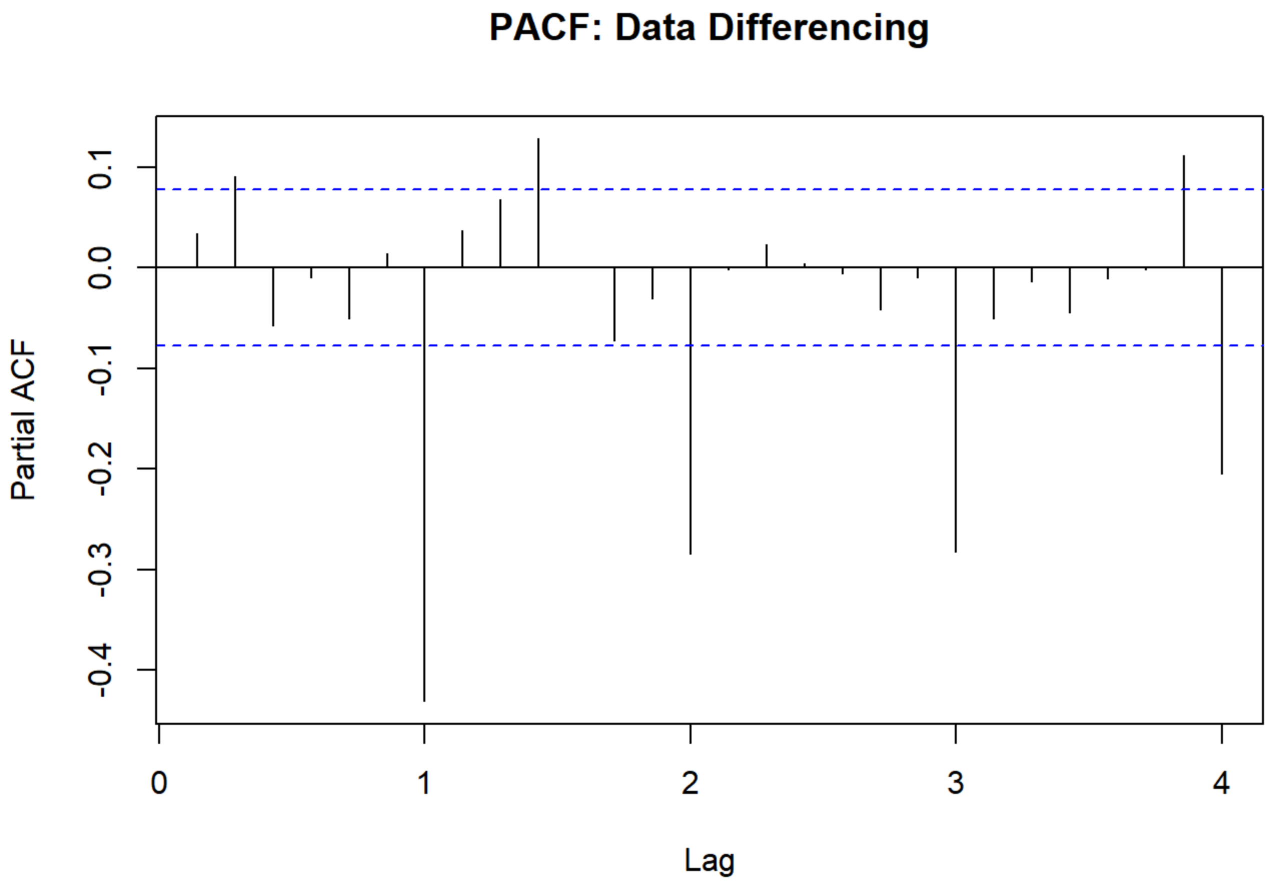
The output shows that ACF exhibits a slowly dying down sinusoidal pattern, indicated by autocorrelation values consistently exceeding the confidence interval at each lag. Therefore, the data is not yet stationary in the mean. Additionally, the sinusoidal pattern suggests that the data is seasonal. This pattern typically appears as a regular wave-like fluctuation at specific lags, reflecting a recurring seasonal period of 38 in the data. Subsequently, seasonal differencing is performed on the transformed data using a lag of 7.

Differencing for Stationarity

```
data_diff_fix <- diff(data_fix_transformed, lag = 7, differences = 1)
acf(data_diff_fix, main = "ACF: Data Differencing")
```




```
pacf(data_diff_fix, main = "PACF: Data Differencing")
```



Based on the output, it can be seen that the ACF pattern no longer slowly dies down, indicating that the differenced data is now stationary in terms of the mean. Data that is stationary in both variance and mean can be modeled based on the ACF and PACF plots. The output sequentially presents the ACF and PACF plots of the stationary data. Based on the ACF plot, there is a spike at lag 2 followed by a cut-off after lag 2. Additionally, for the seasonal component, there is a spike at lag 7 with a cut-off after lag 7. Meanwhile, the PACF plot shows a gradually decreasing pattern at seasonal levels, specifically at lags 7, 14, 21, and 28. Based on the plot analysis, the suspected SARIMA models are SARIMA(0,0,2)(0,1,1)7, SARIMA(0,0,[2])(0,1,1)7, and SARIMA(0,0,2)(1,1,1)7.

SARIMA Modeling

```
model_SARIMA1 <- Arima(data_fix_transformed, order = c(0,0,2), seasonal = list(order = c(0,1,
1), period = 7))

model_SARIMA2 <- Arima(data_fix_transformed, order = c(0,0,2), seasonal = list(order = c(1,1,
1), period = 7))

model_SARIMA3 <- Arima(data_fix_transformed, order = c(0,0,7), seasonal = list(order = c(0,1,
1), period = 7), fixed = c(0, NA, rep(0, 4), 0, NA))
```

Testing SARIMA Model Parameter

The next analysis involves estimating the parameters of all suspected models using the **Conditional Least Squares (CLS)** method. Then, the significance of these model parameters is tested.

```
coeftest(model_SARIMA1)
```

```
##
## z test of coefficients:
##
##       Estimate Std. Error  z value Pr(>|z|)
## ma1   0.074195   0.039905   1.8593  0.06298 .
## ma2   0.085187   0.038693   2.2016  0.02769 *
## sma1 -0.905111   0.030140 -30.0306  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
coeftest(model_SARIMA2)
```

```
##
## z test of coefficients:
##
##       Estimate Std. Error  z value  Pr(>|z|)
## ma1   0.072218   0.039773   1.8158  0.069408 .
## ma2   0.094769   0.038979   2.4312  0.015047 *
## sar1  0.128411   0.046157   2.7821  0.005401 **
## sma1 -0.958141   0.027136 -35.3093 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
coeftest(model_SARIMA3)
```

```
##
## z test of coefficients:
##
##       Estimate Std. Error  z value Pr(>|z|)
## ma2   0.088870   0.039193   2.2675  0.02336 *
## sma1 -0.899284   0.030163 -29.8143  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Based on the output, all p-values are greater than α = 0.05, indicating that the parameters of each SARIMA model are significant. Therefore, the analysis proceeds with the diagnostic check.

Diagnostic Check

The diagnostic check is conducted to assess whether the errors in the data are random or independent and follow a normal distribution. The randomness of errors is examined using the Ljung-Box test.

```
lags <- c(7,14,21,28,35)
diagnostic_tests <- function(model, lags) {
  ljung_box_results <- lapply(lags, function(lag) {
    test <- Box.test(model$residuals, type = "Ljung-Box", lag = lag)
    data.frame(Lag = lag, Statistic = test$statistic, p_value = test$p.value)
  })

  lillie_test <- lillie.test(model$residuals)
  coef_test <- coeftest(model)
  ljung_box_df <- do.call(rbind, ljung_box_results)
  lillie_test_df <- data.frame(Test = "Lilliefors", Statistic = lillie_test$statistic, p_valu
e = lillie_test$p.value)

  list(Coefficient_Test = coef_test, Ljung_Box = ljung_box_df, Lilliefors = lillie_test_df)
}
```

```
result_SARIMA1 <- diagnostic_tests(model_SARIMA1, lags)
print(result_SARIMA1)
```

```
## $Coefficient_Test
##
## z test of coefficients:
##
##       Estimate Std. Error  z value Pr(>|z|)
## ma1   0.074195   0.039905   1.8593  0.06298 .
## ma2   0.085187   0.038693   2.2016  0.02769 *
## sma1 -0.905111   0.030140 -30.0306  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## $Ljung_Box
##            Lag Statistic   p_value
## X-squared    7  7.552041 0.3737444
## X-squared1  14 17.867927 0.2128625
## X-squared2  21 28.284359 0.1322693
## X-squared3  28 32.783180 0.2438202
## X-squared4  35 42.826175 0.1705545
##
## $Lilliefors
##         Test  Statistic   p_value
## D Lilliefors 0.01828572 0.8666523
```

```
result_SARIMA2 <- diagnostic_tests(model_SARIMA2, lags)
print(result_SARIMA2)
```

```
## $Coefficient_Test
##
## z test of coefficients:
##
##       Estimate Std. Error  z value  Pr(>|z|)
## ma1   0.072218   0.039773   1.8158  0.069408 .
## ma2   0.094769   0.038979   2.4312  0.015047 *
## sar1  0.128411   0.046157   2.7821  0.005401 **
## sma1 -0.958141   0.027136 -35.3093 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## $Ljung_Box
##             Lag Statistic   p_value
## X-squared    7  2.970168 0.8877481
## X-squared1  14 13.568581 0.4823211
## X-squared2  21 22.346464 0.3797859
## X-squared3  28 27.351805 0.4991580
## X-squared4  35 39.201713 0.2869120
##
## $Lilliefors
##         Test  Statistic   p_value
## D Lilliefors 0.02342219 0.5309554
```

```
result_SARIMA3 <- diagnostic_tests(model_SARIMA3, lags)
print(result_SARIMA3)
```

```
## $Coefficient_Test
##
## z test of coefficients:
##
##       Estimate Std. Error  z value Pr(>|z|)
## ma2   0.088870   0.039193   2.2675  0.02336 *
## sma1 -0.899284   0.030163 -29.8143  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## $Ljung_Box
##             Lag Statistic    p_value
## X-squared    7  10.36175 0.16898339
## X-squared1  14  20.96717 0.10247683
## X-squared2  21  32.13707 0.05670544
## X-squared3  28  37.45709 0.10911813
## X-squared4  35  47.35736 0.07931509
##
## $Lilliefors
##         Test  Statistic  p_value
## D Lilliefors 0.02600699 0.361853
```

From the output, it can be seen that the suspected SARIMA(0,0,[2])$(0,1,1)_7$ model has a p-value < 0.05, indicating that the model does not meet the adequacy criteria or that its residuals do not satisfy the white noise assumption. Therefore, the SARIMA(0,0,[2])$(0,1,1)_7$ model cannot be used.

For models that satisfy the white noise assumption, the next step is to test whether the residuals follow a normal distribution. This is done using the Lilliefors normality test. According to the output, the p-values for both models are greater than alpha or 0.05, indicating that the residuals of SARIMA(0,0,2)(0,1,1)$_7$ and SARIMA(0,0,2)(1,1,1)$_7$ follow a normal distribution.

Since both models meet the white noise assumption and their residuals are normally distributed, they can be used for forecasting.

Model Evaluation with Test Data

Data Preparation

After going through various stages in the SARIMA method, several suitable models have been identified for forecasting. Next, these models are evaluated based on RMSE and MAPE, with the model having the smallest error selected for forecasting. The evaluation is conducted using test data, covering inpatient admission records from **December 11, 2023, to January 7, 2024**, consisting of **28 observations**. The test data is much smaller than the train data because SARIMA is not well-suited for long-term forecasting. The test data is not included in the training data.

```
data_test <- data[data$Subset == "Test",]
test_data <- data_test[, c("Tanggal", "Value")]
test_data_ts <- ts(test_data$Value, frequency = 7, start = c(1, 1))
test_data_transformed <- BoxCox(test_data_ts, lambda)
```

Performing Forecasting for All Models That Meet Assumptions

```
forecasts_test_SARIMA1 <- forecast(model_SARIMA1, h = length(test_data_transformed))
forecast_values_test_SARIMA1_original_scale <- InvBoxCox(forecasts_test_SARIMA1$mean, lambda)

forecasts_test_SARIMA2 <- forecast(model_SARIMA2, h = length(test_data_transformed))
forecast_values_test_SARIMA2_original_scale <- InvBoxCox(forecasts_test_SARIMA2$mean, lambda)
```

The forecasting results are shown in the output below.

```
results <- data.frame(
  Date = data_test$Tanggal,
  Day = data_test$Hari,
  Actual_Value = data_test$Value,
  Predicted_Value_SARIMA1 = forecast_values_test_SARIMA1_original_scale,
  Predicted_Value_SARIMA2 = forecast_values_test_SARIMA2_original_scale
)
results
```

```
##          Date    Day Actual_Value Predicted_Value_SARIMA1
## 1  2023-12-11  Senin          138               113.18169
## 2  2023-12-12 Selasa          156               139.37966
## 3  2023-12-13   Rabu          138               135.25537
## 4  2023-12-14  Kamis          144               141.51757
## 5  2023-12-15  Jumat          108               139.70984
## 6  2023-12-16  Sabtu           90               103.09303
## 7  2023-12-17 Minggu           60                88.11334
## 8  2023-12-18  Senin           96               117.40114
## 9  2023-12-19 Selasa          168               143.16959
## 10 2023-12-20   Rabu          174               135.25537
## 11 2023-12-21  Kamis          216               141.51757
## 12 2023-12-22  Jumat          132               139.70984
## 13 2023-12-23  Sabtu           96               103.09303
## 14 2023-12-24 Minggu           66                88.11334
## 15 2023-12-25  Senin          126               117.40114
## 16 2023-12-26 Selasa          174               143.16959
## 17 2023-12-27   Rabu          132               135.25537
## 18 2023-12-28  Kamis          126               141.51757
## 19 2023-12-29  Jumat           84               139.70984
## 20 2023-12-30  Sabtu           66               103.09303
## 21 2023-12-31 Minggu           66                88.11334
## 22 2024-01-01  Senin           60               117.40114
## 23 2024-01-02 Selasa          150               143.16959
## 24 2024-01-03   Rabu          156               135.25537
## 25 2024-01-04  Kamis          156               141.51757
## 26 2024-01-05  Jumat          168               139.70984
## 27 2024-01-06  Sabtu           90               103.09303
## 28 2024-01-07 Minggu           96                88.11334
##    Predicted_Value_SARIMA2
## 1                109.47444
## 2                141.51775
## 3                130.33310
## 4                140.86974
## 5                132.03281
## 6                102.12323
## 7                 83.76412
## 8                119.54554
## 9                142.43454
## 10               135.12188
## 11               136.86901
## 12               135.29389
## 13               103.73554
## 14                88.05828
## 15               120.87092
## 16               142.55247
## 17               135.74308
## 18               136.35946
## 19               135.71554
## 20               103.94349
## 21                88.61748
## 22               121.04164
## 23               142.56762
## 24               135.82296
## 25               136.29409
```

```
## 26                135.76973
## 27                103.97021
## 28                 88.68942
```

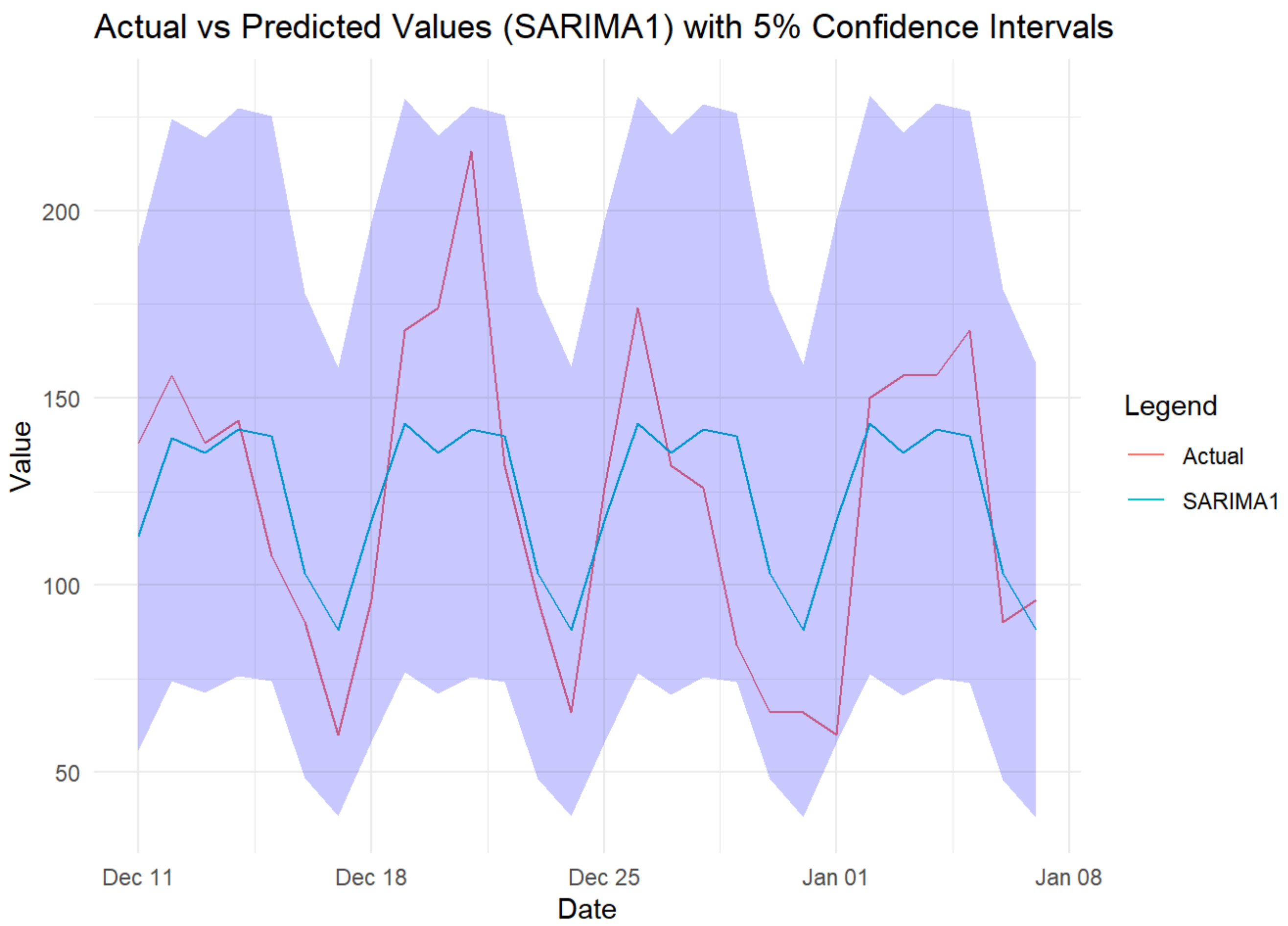
Model Evaluation

```
rmse <- function(actual, predicted) {
  sqrt(mean((actual - predicted)^2))
}

mape <- function(actual, predicted) {
  mean(abs((actual - predicted) / actual)) * 100
}

rmse_SARIMA1 <- rmse(results$Actual_Value, results$Predicted_Value_SARIMA1)
mape_SARIMA1 <- mape(results$Actual_Value, results$Predicted_Value_SARIMA1)

rmse_SARIMA2 <- rmse(results$Actual_Value, results$Predicted_Value_SARIMA2)
mape_SARIMA2 <- mape(results$Actual_Value, results$Predicted_Value_SARIMA2)

summary_df <- data.frame(
  Model = c("SARIMA1", "SARIMA2"),
  RMSE = c(rmse_SARIMA1, rmse_SARIMA2),
  MAPE = c(mape_SARIMA1, mape_SARIMA2)
)
summary_df
```

```
##     Model     RMSE     MAPE
## 1 SARIMA1 28.57020 22.19397
## 2 SARIMA2 28.99893 22.02917
```

Based on the output, it can be seen that the SARIMA(0,0,2)(0,1,1)$_7$ model has the smallest error, making it the best model for forecasting.

```
ggplot(results, aes(x = Date)) +
  geom_line(aes(y = Actual_Value, color = "Actual")) +
  geom_line(aes(y = Predicted_Value_SARIMA1, color = "SARIMA1")) +
  geom_ribbon(aes(ymin = InvBoxCox(forecasts_test_SARIMA1$lower[, 2], lambda),
                  ymax = InvBoxCox(forecasts_test_SARIMA1$upper[, 2], lambda)),
              alpha = 0.2, fill = "blue") +
  labs(title = "Actual vs Predicted Values (SARIMA1) with 5% Confidence Intervals",
       x = "Date",
       y = "Value",
       color = "Legend") +
  theme_minimal()
```



The output shows the prediction results for the test data used. From the plot, it can be seen that the SARIMA(0,0,2)(0,1,1)$_7$ model performs well in forecasting. This is indicated by the predicted values being close to the actual values, and the confidence interval covering all actual values, demonstrating that the model's uncertainty estimation is also reliable.