STAT 455 Project

2025-11-29

Read/inspect data

Source: https://www.eia.gov/dnav/ng/ng_enr_drill_s1_m.htm

```
library(ggplot2)
library(forecast)
```

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

```
library(zoo)
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library(acp)
```

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

```
## Loading required package: quantmod
```

```
## Loading required package: xts
```

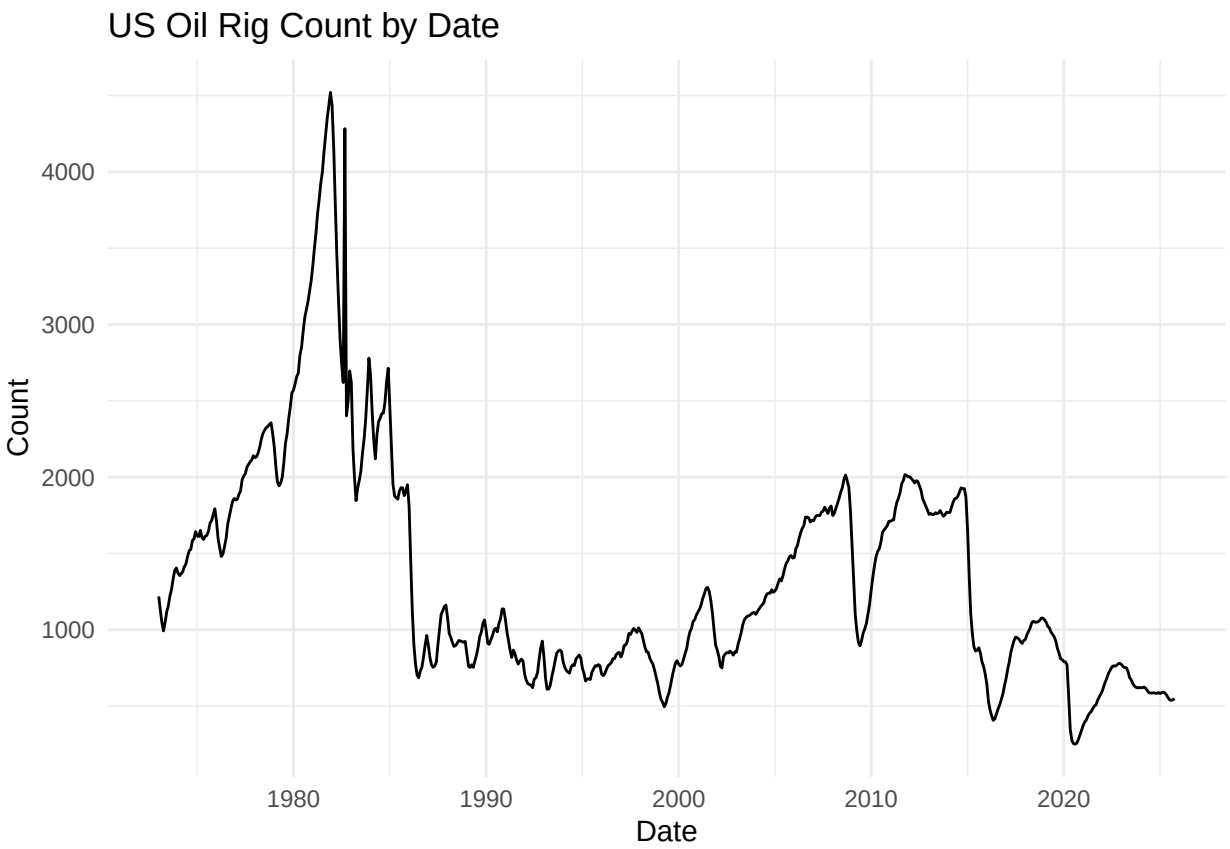
```
## Loading required package: TTR
```

```
library(tscount)
library(ggplot2)
library(dplyr)
```

```
##
## ########################## Warning from 'xts' package ##########################
## #                                                                              #
## # The dplyr lag() function breaks how base R's lag() function is supposed to  #
## # work, which breaks lag(my_xts). Calls to lag(my_xts) that you type or       #
## # source() into this session won't work correctly.                             #
```

```
## #                                                                              #
## # Use stats::lag() to make sure you're not using dplyr::lag(), or you can add #
## # conflictRules('dplyr', exclude = 'lag') to your .Rprofile to stop           #
## # dplyr from breaking base R's lag() function.                                #
## #                                                                              #
## # Code in packages is not affected. It's protected by R's namespace mechanism #
## # Set `options(xts.warn_dplyr_breaks_lag = FALSE)` to suppress this warning.  #
## #                                                                              #
## ################################################################################


##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:xts':
##
##     first, last

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
data <- read.csv('rigs.csv')

data$date <- as.Date(paste0("01-", data$date), format = "%d-%b-%y")

data %>%
  ggplot(aes(date, count)) +
  geom_line() +
  theme_minimal() +
  labs(x = 'Date',
       y = 'Count',
       title = 'US Oil Rig Count by Date')
```



Convert data to time series

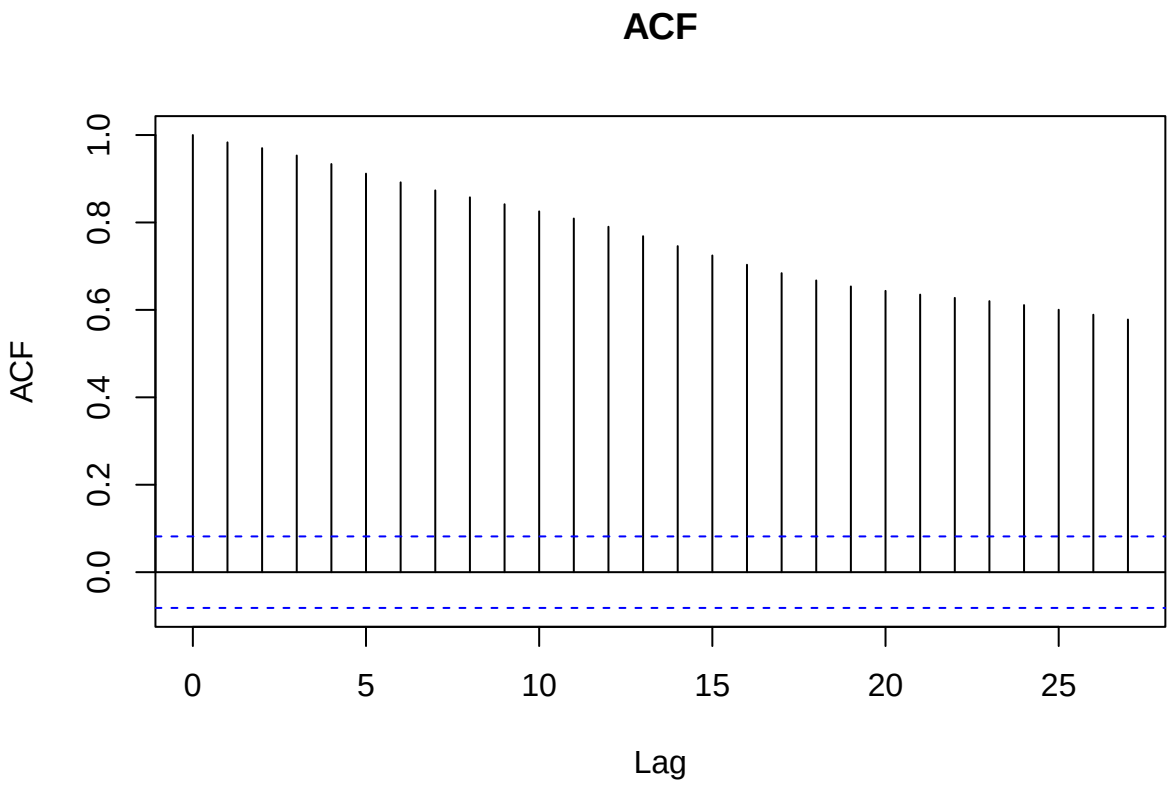
```
data$date <- as.yearmon(data$date, "%b-%y")

z <- ts(data$count,
        start = c(as.numeric(format(data$date[1], "%Y")),
                  as.numeric(format(data$date[1], "%m"))),
        frequency = 12)
```

Specifying test set

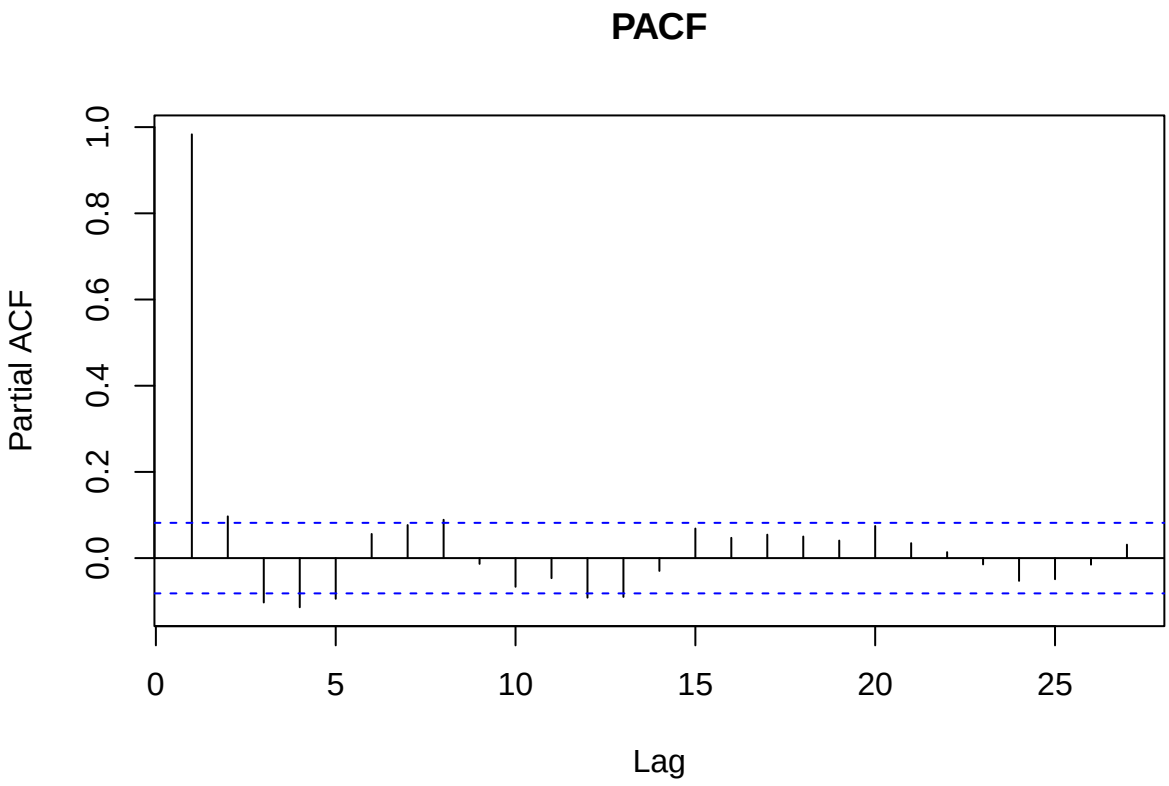
```
# use a 5-year test window
test_years <- 5
test_horizon <- test_years * 12

n <- length(z)
start_test <- n - test_horizon
h <- 1 # 1-step forecast
```

ACF and PACF Plots

```
acf(as.numeric(z)[1:start_test], main = "ACF")
```




```
pacf(as.numeric(z)[1:start_test], main = "PACF")
```



Running models and evaluating

```r
# Function to return metrics
evaluate_metrics <- function(actual, forecasted) {
  acc <- accuracy(forecasted, actual)
  data.frame(
    ME   = acc["Test set", "ME"],
    RMSE = acc["Test set", "RMSE"],
    MAE  = acc["Test set", "MAE"],
    MPE  = acc["Test set", "MPE"],
    MAPE = acc["Test set", "MAPE"]
  )
}
```

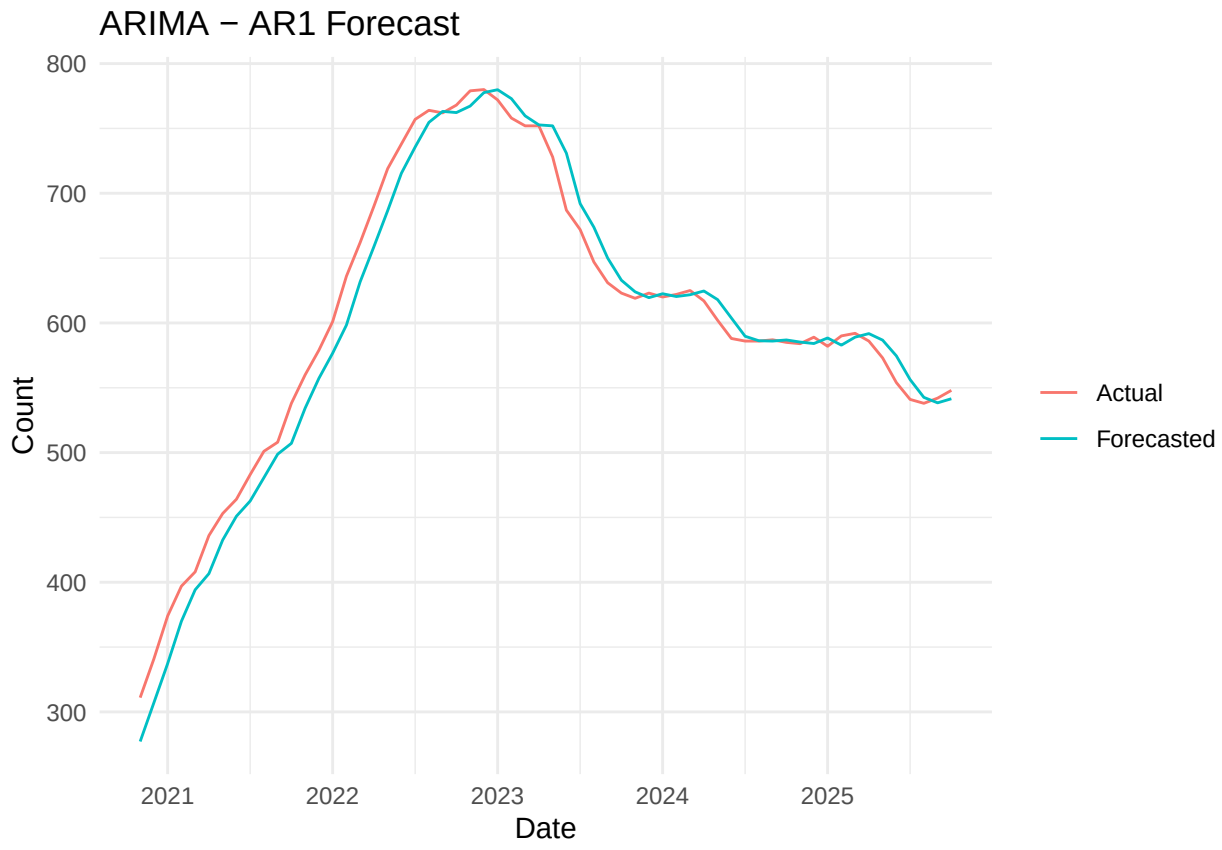
```r
# Function to plot forecast vs actual
evaluate_plot <- function(actual, forecasted, dates, title = "Forecast vs Actual") {
  df <- data.frame(
    date = rep(dates, 2),
    value = c(actual, forecasted),
    type = rep(c("Actual", "Forecasted"), each = length(actual))
  )

  ggplot(df, aes(x = date, y = value, color = type)) +
```

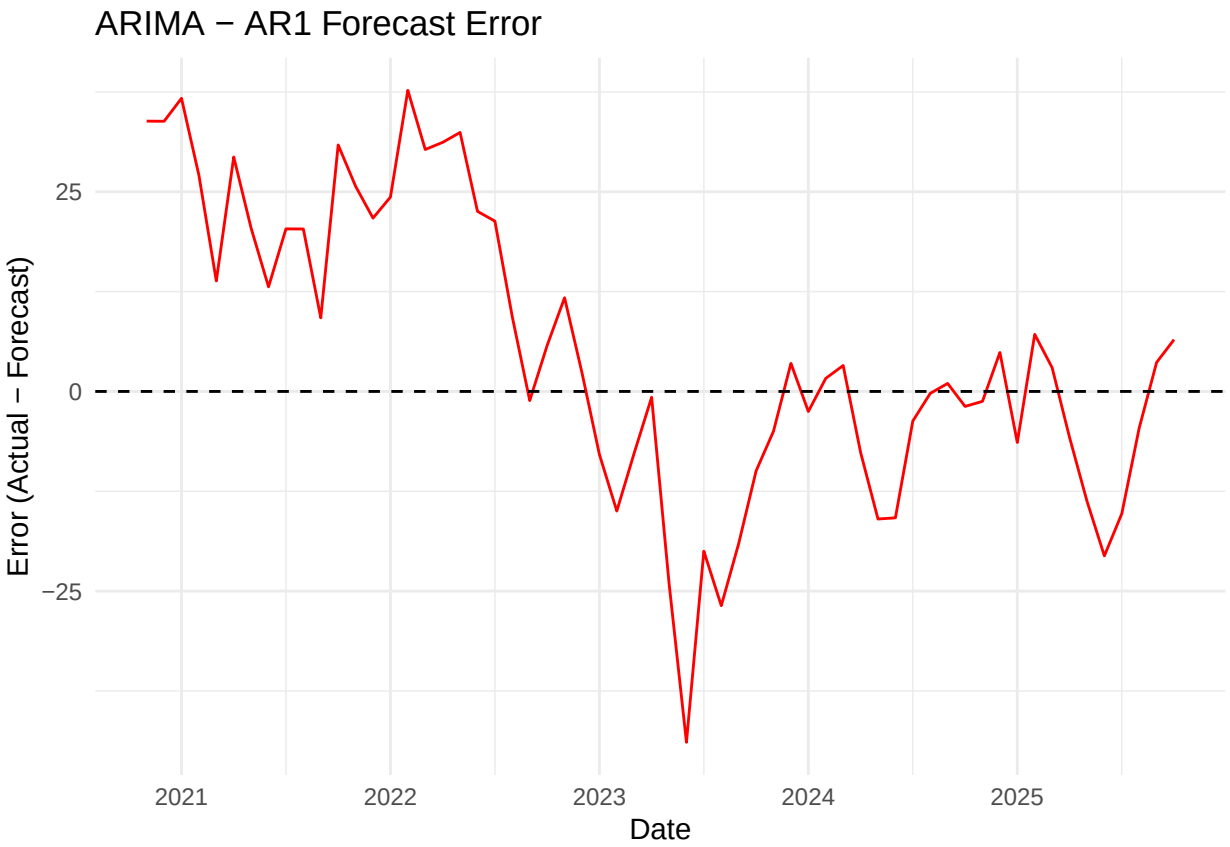
```
    geom_line() +
    theme_minimal() +
    labs(
      x = "Date",
      y = "Count",
      title = title
    ) +
    theme(legend.title = element_blank())
}
```

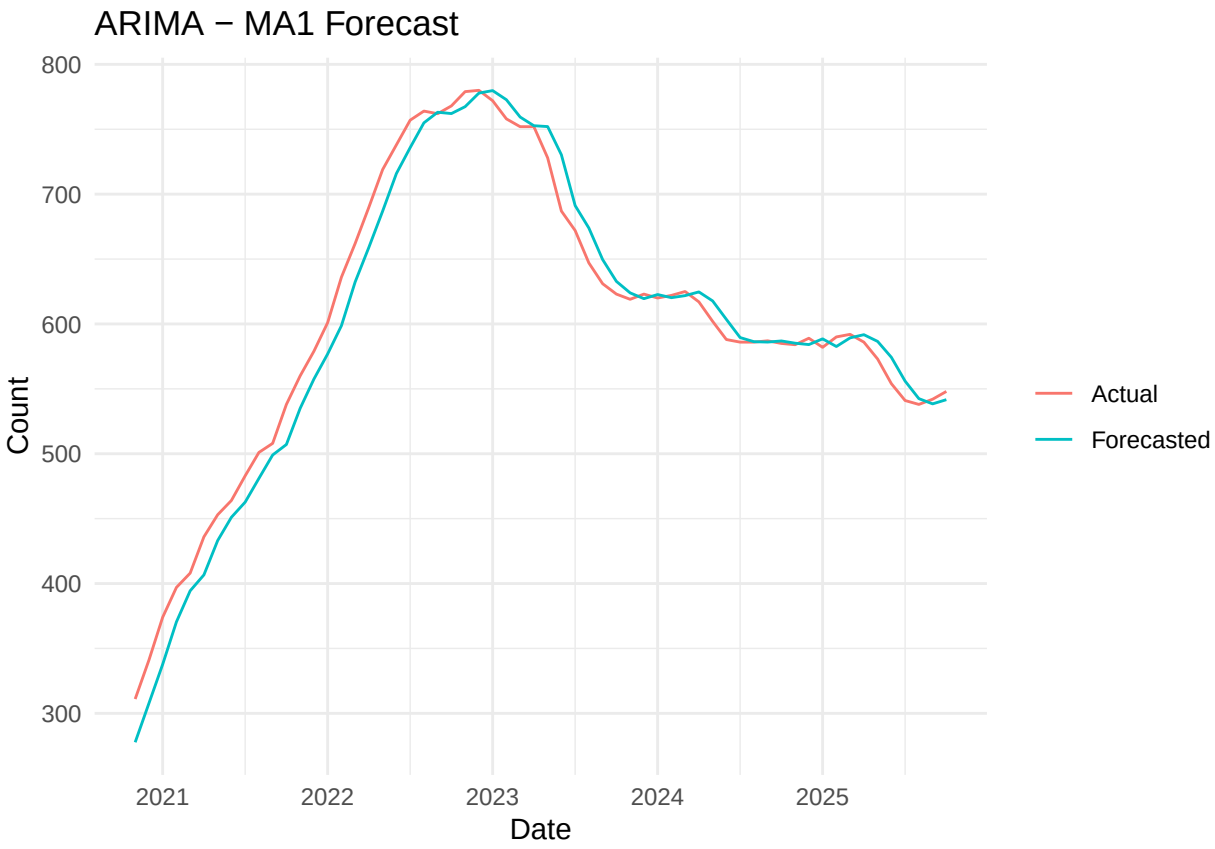
```
# Function to plot errors
error_plot <- function(actual, forecasted, dates, title = "Forecast Error") {
  df <- data.frame(
    date = dates,
    error = actual - forecasted
  )

  ggplot(df, aes(x = date, y = error)) +
    geom_line(color = "red") +
    geom_hline(yintercept = 0, linetype = "dashed") +
    theme_minimal() +
    labs(
      x = "Date",
      y = "Error (Actual - Forecast)",
      title = title
    )
}
```

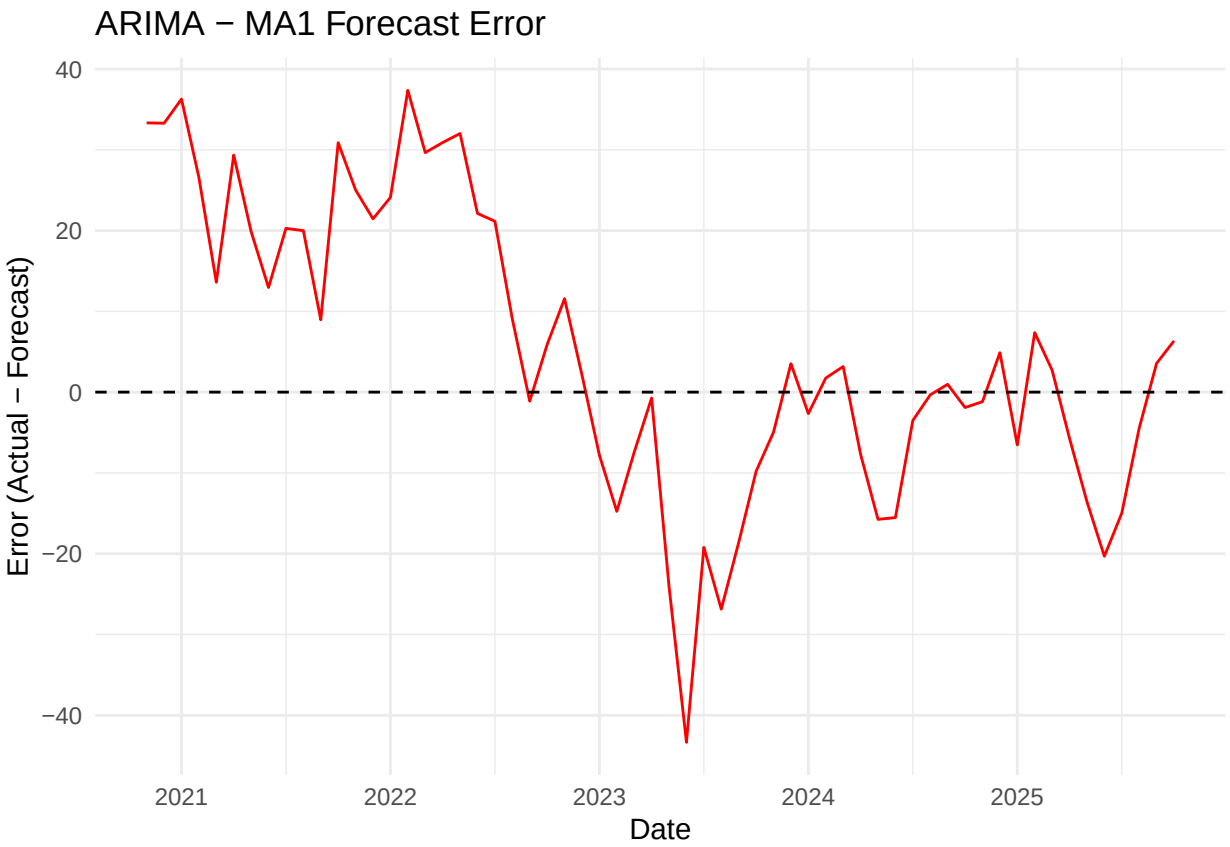
```
# Setup
test_years <- 5
test_horizon <- test_years * 12
n <- length(z)
start_test <- n - test_horizon
h <- 1
dates <- index(z)

results_list <- list()    # store metrics
forecast_plots <- list()  # store forecast vs actual plots
error_plots <- list()     # store error plots

# ARIMA models
arima_models <- list(
  AR1    = c(1,1,0),
  MA1    = c(0,1,1),
  ARMA11 = c(1,1,1)
)

for (name in names(arima_models)) {
  pred <- rep(NA, n)

  for (i in start_test:(n - h)) {
    train <- window(z, end = time(z)[i])
    fit <- tryCatch(Arima(train, order = arima_models[[name]]), error = function(e) NULL)
```

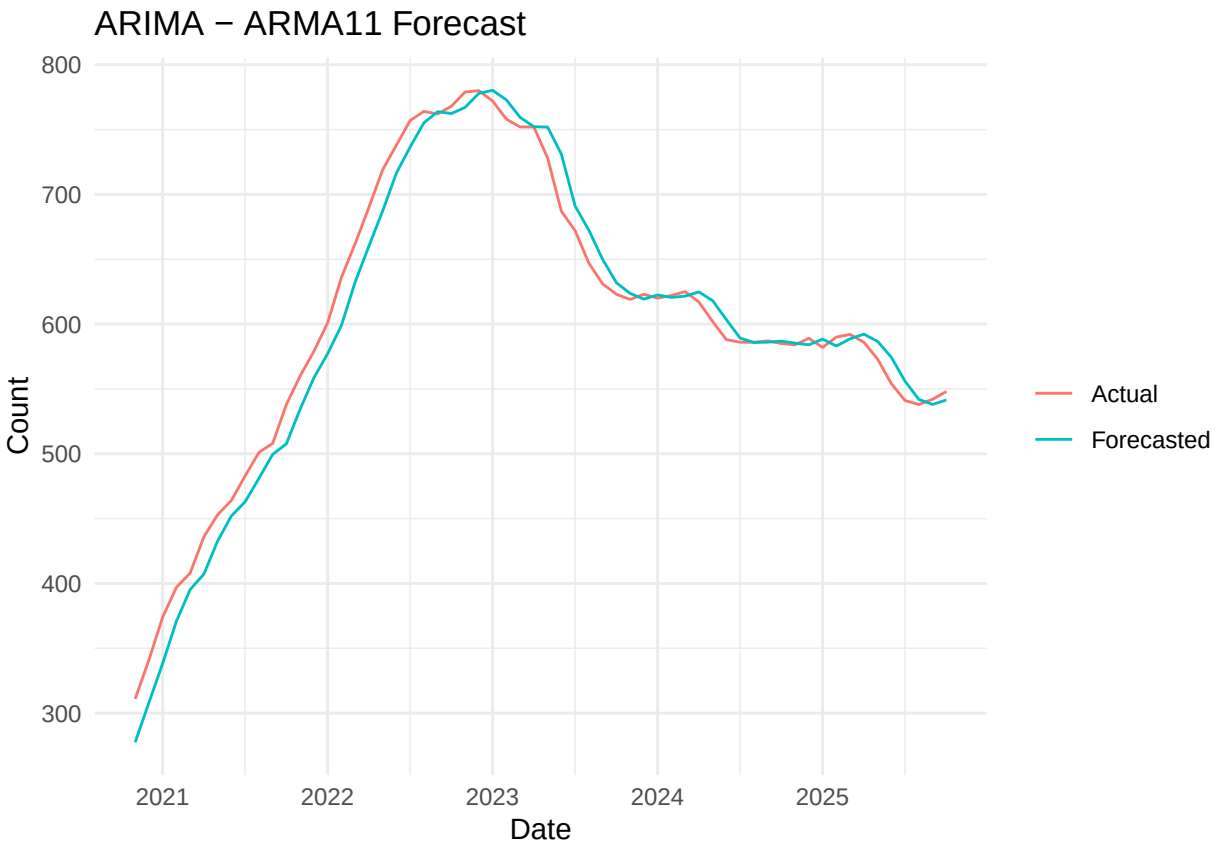
```
    if (!is.null(fit)) pred[i + h] <- forecast(fit, h = h)$mean[h]
  }

  idx <- start_test + h
  actual <- z[idx:n]
  forecasted <- pred[idx:n]
  valid <- !is.na(forecasted)

  # Metrics
  metrics <- evaluate_metrics(actual[valid], forecasted[valid])
  metrics <- cbind(Model = name, metrics)
  results_list[[name]] <- metrics

  # Forecast vs actual plot
  p_forecast <- evaluate_plot(actual[valid], forecasted[valid], dates[idx:n][valid],
                              title = paste("ARIMA -", name, "Forecast"))
  print(p_forecast)
  forecast_plots[[name]] <- p_forecast

  # Error plot
  p_error <- error_plot(actual[valid], forecasted[valid], dates[idx:n][valid],
                        title = paste("ARIMA -", name, "Forecast Error"))
  print(p_error)
  error_plots[[name]] <- p_error
}
```

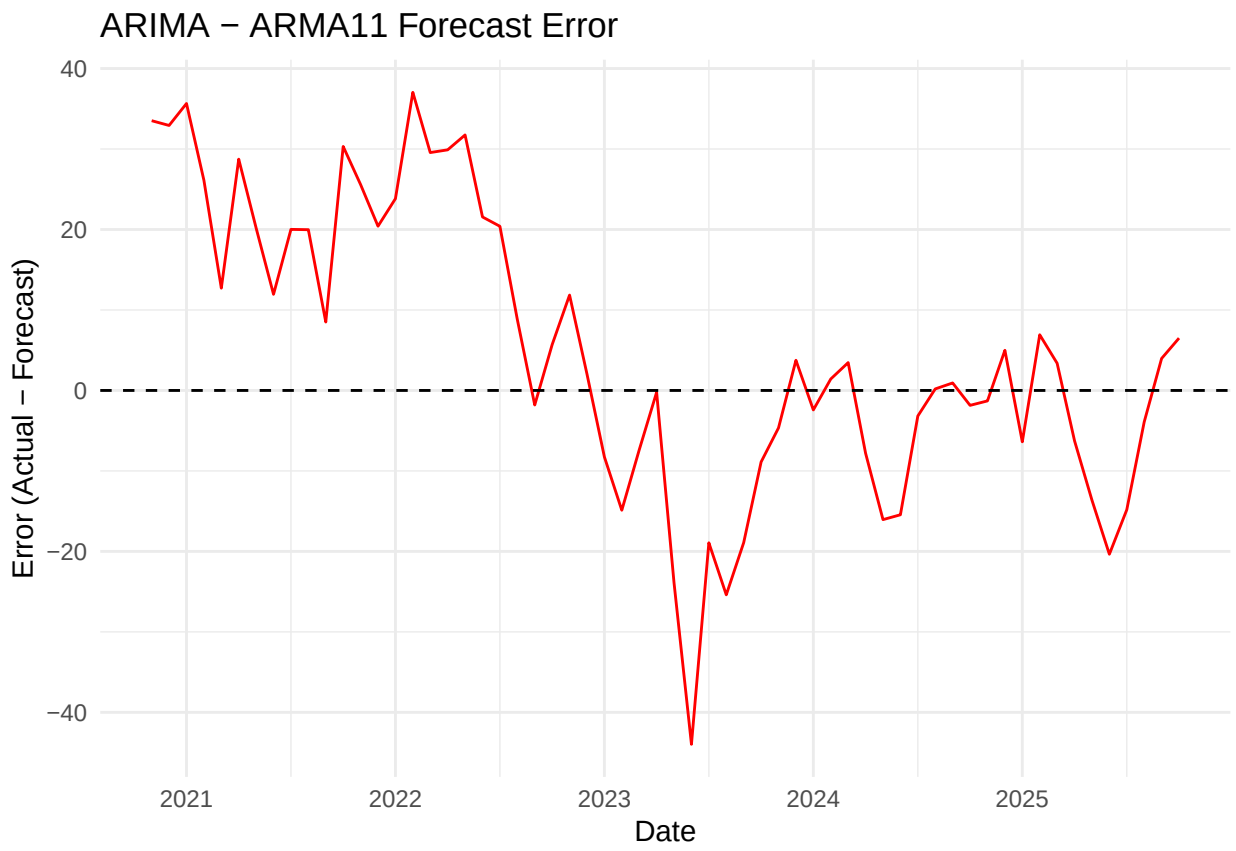



ARIMA − AR1 Forecast Error
Error (Actual − Forecast)
25
0
−25
2021
2022
2023
2024
2025
Date


ARIMA − MA1 Forecast
800
700
600
500
400
300
Count
2021
2022
2023
2024
2025
Date
Actual
Forecasted

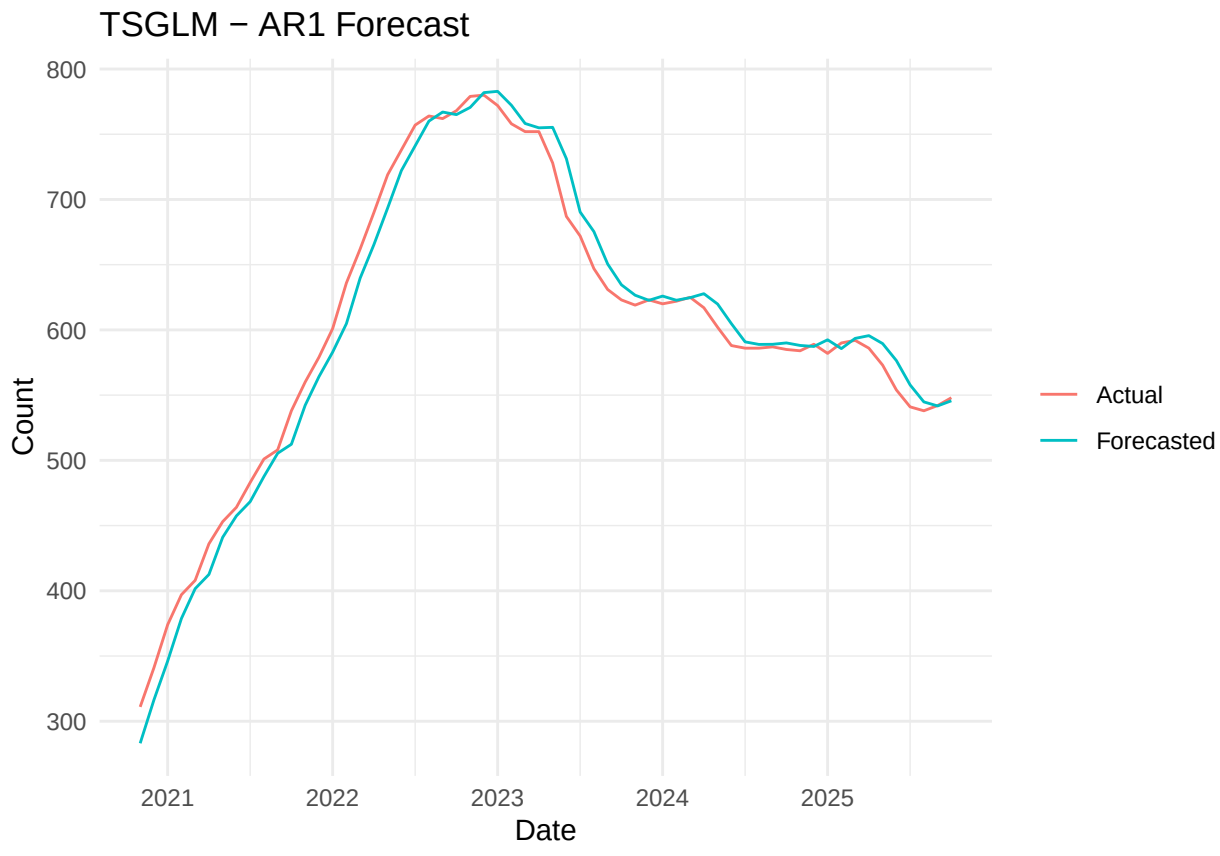

ARIMA − MA1 Forecast Error
Error (Actual − Forecast)
40
20
0
−20
−40
2021
2022
2023
2024
2025
Date

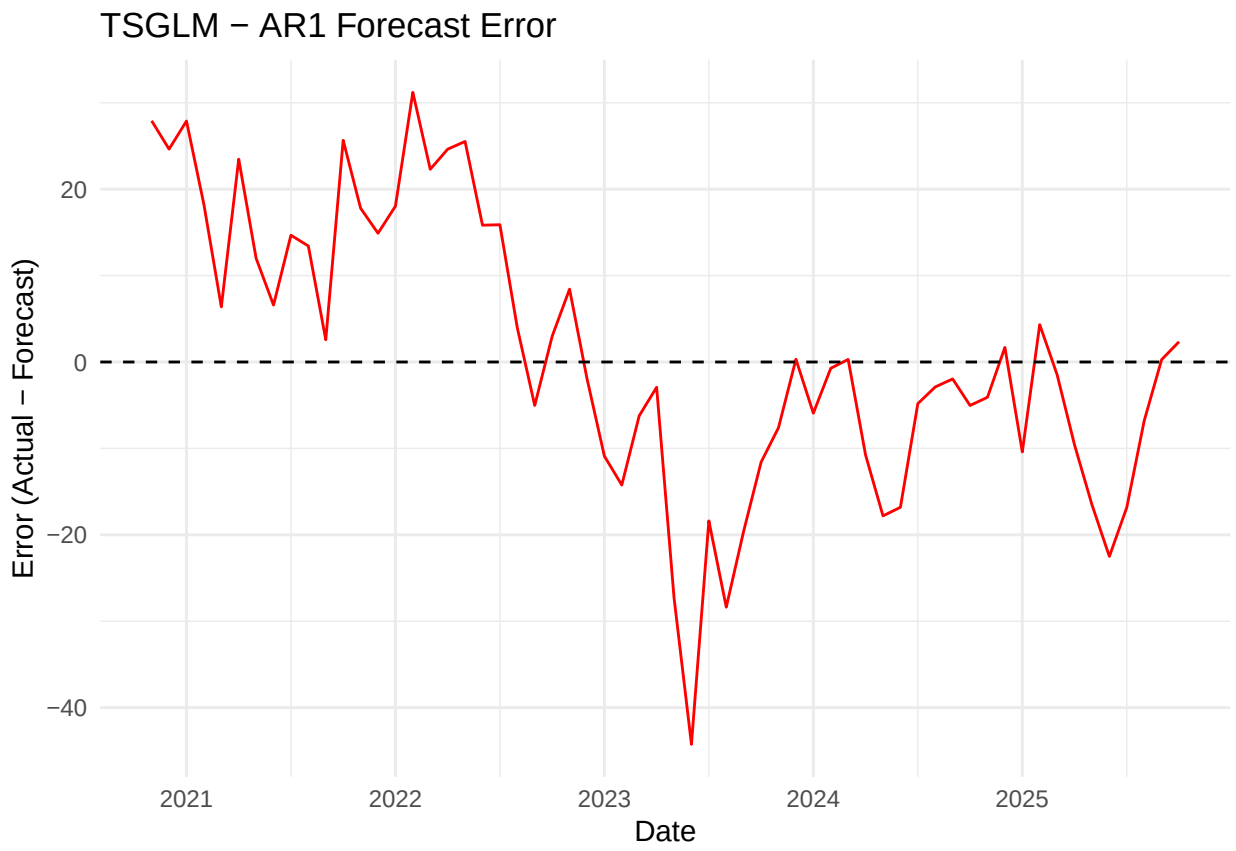

ARIMA − ARMA11 Forecast
800
700
600
500
400
300
Count
2021
2022
2023
2024
2025
Date
Actual
Forecasted

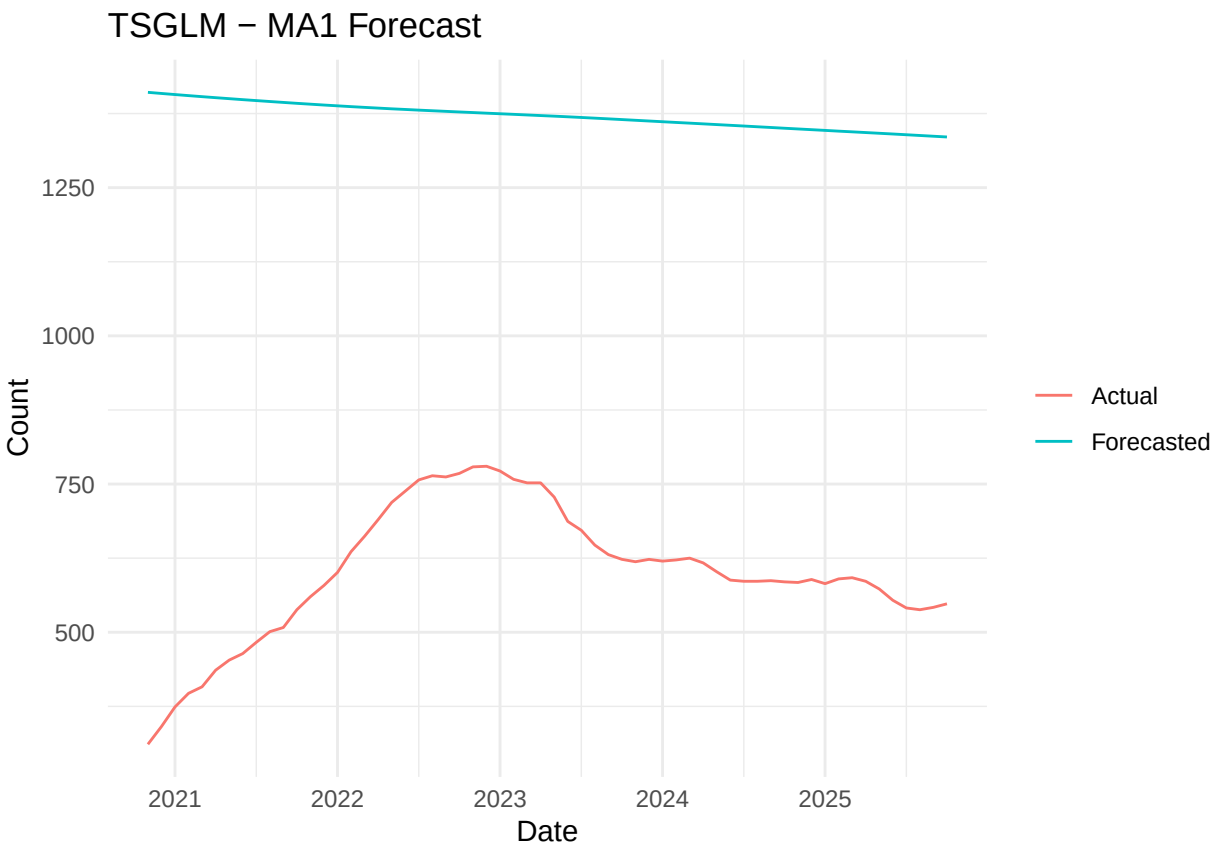


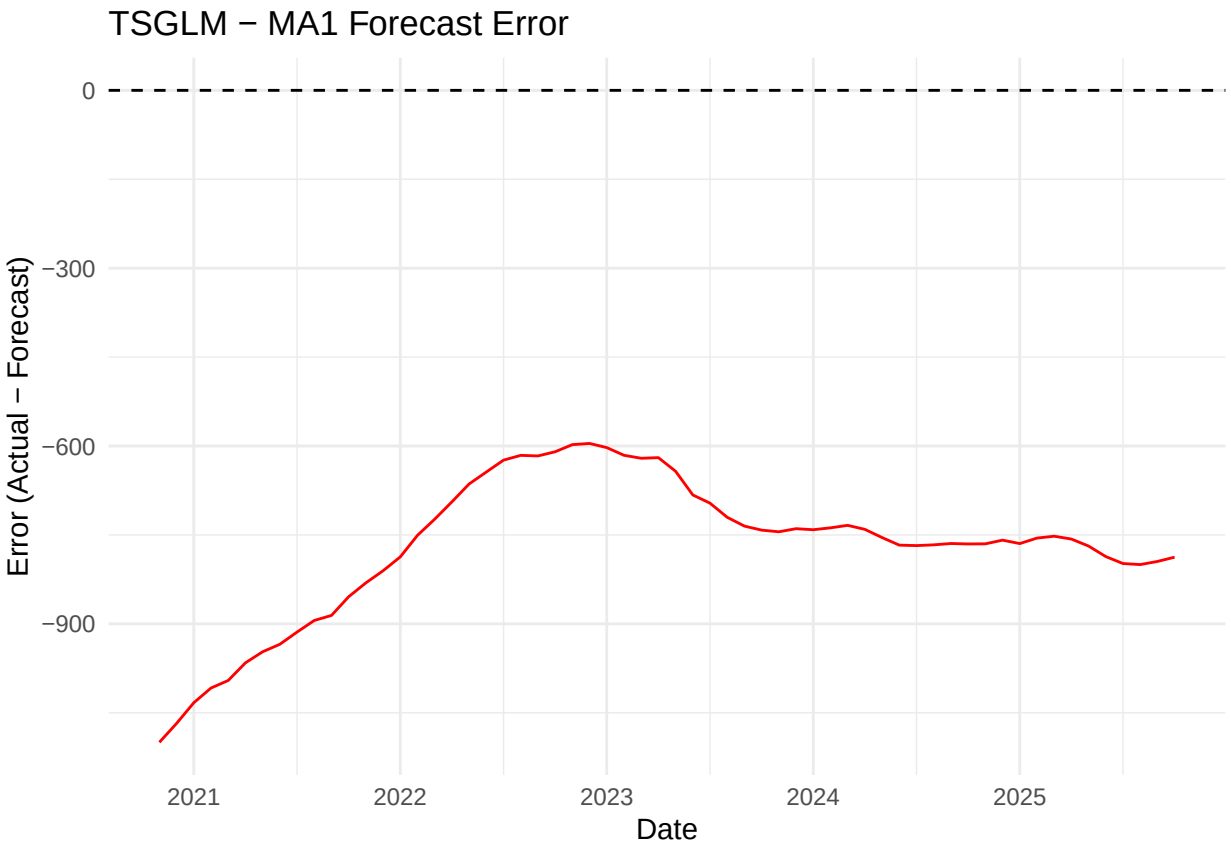

```r
# TSGLM models
z_scaled <- z / 100

tsglm_models <- list(
  AR1    = list(past_obs = 1),
  MA1    = list(past_mean = 1),
  ARMA11 = list(past_obs = 1, past_mean = 1)
)

for (name in names(tsglm_models)) {
  pred <- rep(NA, n)
  max_lag <- max(unlist(tsglm_models[[name]]))
  start <- max(start_test, max_lag + 5)

  for (i in start:(n - h)) {
    train <- z_scaled[1:i]
    fit <- suppressWarnings(
      tryCatch(
        tsglm(train, model = tsglm_models[[name]], link = "identity", distr = "poisson",
              init.method = "marginal", init.drop = FALSE),
        error = function(e) NULL
      )
    )
    lambda_next <- if (!is.null(fit)) predict(fit, n.ahead = h)$pred[h] else mean(train)
    pred[i + h] <- lambda_next * 100
  }
```

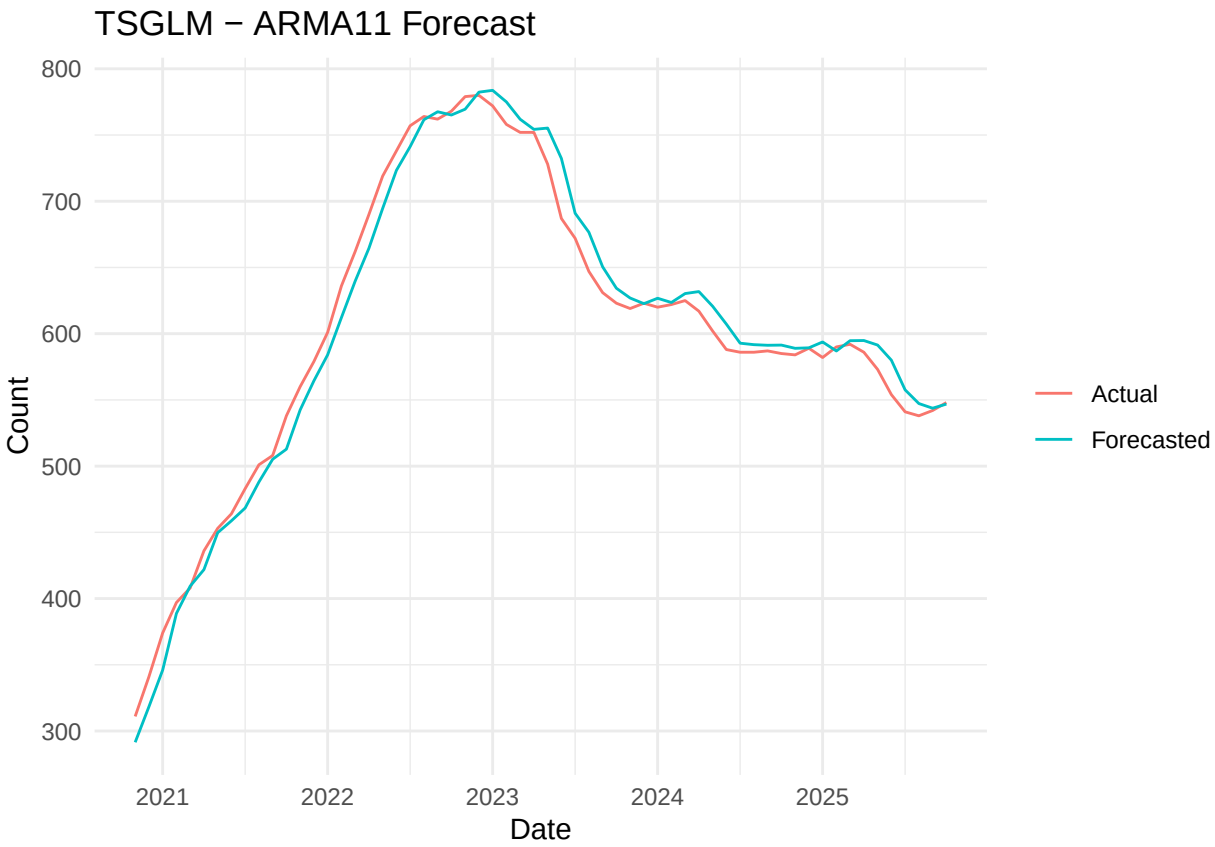
```r
  idx <- start + h
  actual <- z[idx:n]
  forecasted <- pred[idx:n]
  valid <- !is.na(forecasted)

  # Metrics
  metrics <- evaluate_metrics(actual[valid], forecasted[valid])
  metrics <- cbind(Model = paste0("TSGLM_", name), metrics)
  results_list[[paste0("TSGLM_", name)]] <- metrics

  # Forecast vs actual plot
  p_forecast <- evaluate_plot(actual[valid], forecasted[valid], dates[idx:n][valid],
                              title = paste("TSGLM -", name, "Forecast"))
  print(p_forecast)
  forecast_plots[[paste0("TSGLM_", name)]] <- p_forecast

  # Error plot
  p_error <- error_plot(actual[valid], forecasted[valid], dates[idx:n][valid],
                        title = paste("TSGLM -", name, "Forecast Error"))
  print(p_error)
  error_plots[[paste0("TSGLM_", name)]] <- p_error
}
```

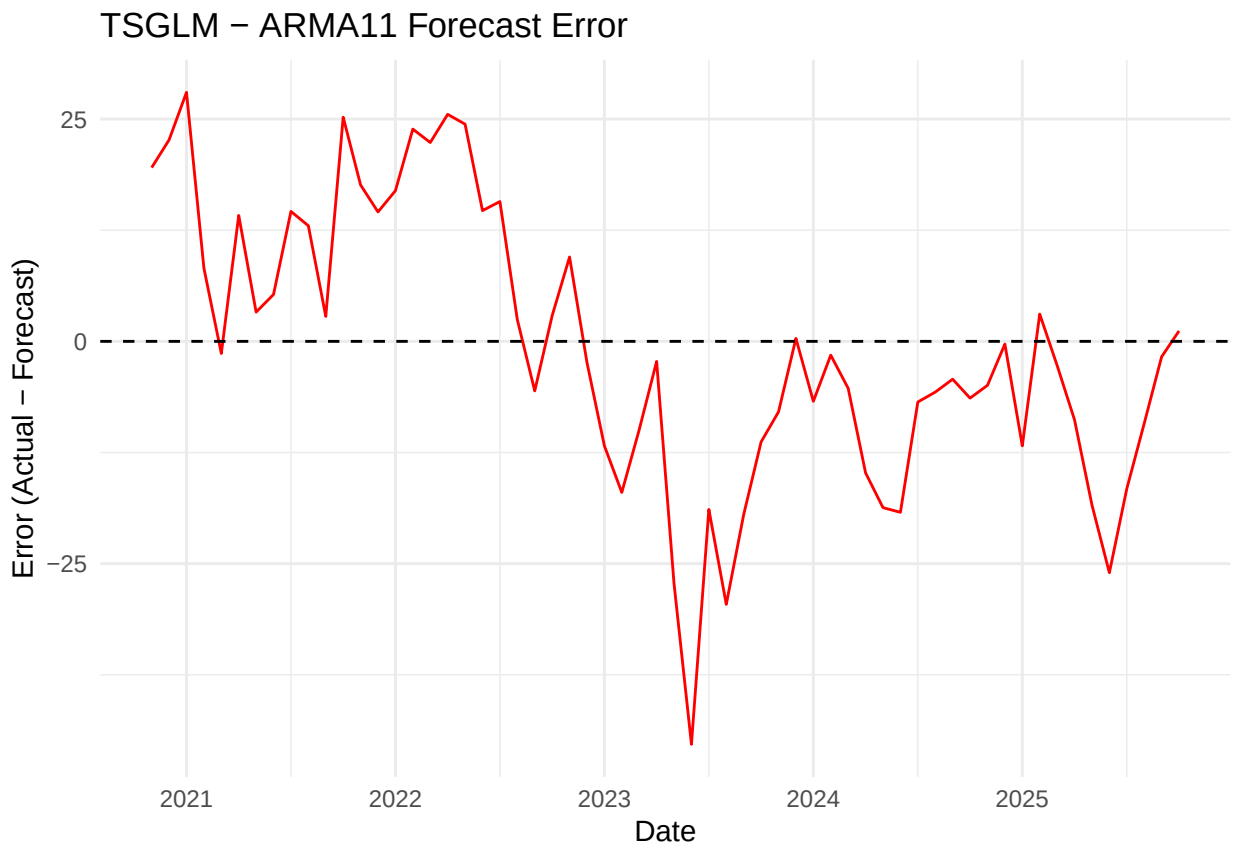



TSGLM − AR1 Forecast Error
Error (Actual − Forecast)
20
0
−20
−40
2021
2022
2023
2024
2025
Date


TSGLM − MA1 Forecast
Count
1250
1000
750
500
2021
2022
2023
2024
2025
Date
Actual
Forecasted


TSGLM − MA1 Forecast Error
0
−300
−600
−900
Error (Actual − Forecast)
2021
2022
2023
2024
2025
Date


TSGLM − ARMA11 Forecast
800
700
600
500
400
300
Count
2021
2022
2023
2024
2025
Date
Actual
Forecasted




```
# Combine results
comparison_table <- do.call(rbind, results_list)
rownames(comparison_table) <- NULL

comparison_table
```

```
##          Model          ME      RMSE       MAE          MPE       MAPE
## 1          AR1    5.053356  18.86039  14.93825    1.1935105   2.745920
## 2          MA1    4.986523  18.62340  14.74655    1.1775409   2.710276
## 3       ARMA11    4.897235  18.43444  14.58590    1.1597454   2.679870
## 4    TSGLM_AR1    1.014541  16.08024  12.79012    0.4412349   2.299766
## 5    TSGLM_MA1 -769.216588 778.51376 769.21659 -137.8471038 137.847104
## 6 TSGLM_ARMA11   -0.803470  15.63198  12.53037    0.0741234   2.189756
```