Regression and Exponential Smoothing Time-Series Forecasting

Lab 12 Forecasting

AUTHOR
Aaron Younger

PUBLISHED
November 21, 2025

Business Understanding

In this analysis, non-seasonally adjusted sales time series data will be used to generate forecasts. The goal is to forecast August through December as accurately as possible.

Data Understanding

Correct R-Version

```
options(scipen = 999)
suppressWarnings(RNGversion("3.5.3"))
```

Libraries

```
library(readxl)
library(DataExplorer)
library(tidyverse)
library(dplyr)
library(ggplot2)
library(strucchange)
library(forecast)
library(lubridate)
```

Import Dataset

```
library(readxl)
my_data <- read_excel("MRTSSM451USN.xlsx",
    sheet = "Monthly")
View(my_data)

my_data %>% head(1) ## Beginning date 1992-01-01
```

```
# A tibble: 1 × 2
  observation_date    MRTSSM451USN
  <dttm>                     <dbl>
1 1992-01-01 00:00:00         2982
```

```
my_data %>% tail(1) ## End date 2025-07-01
```

```
# A tibble: 1 × 2
  observation_date    MRTSSM451USN
  <dttm>                     <dbl>
1 2025-07-01 00:00:00         8010
```

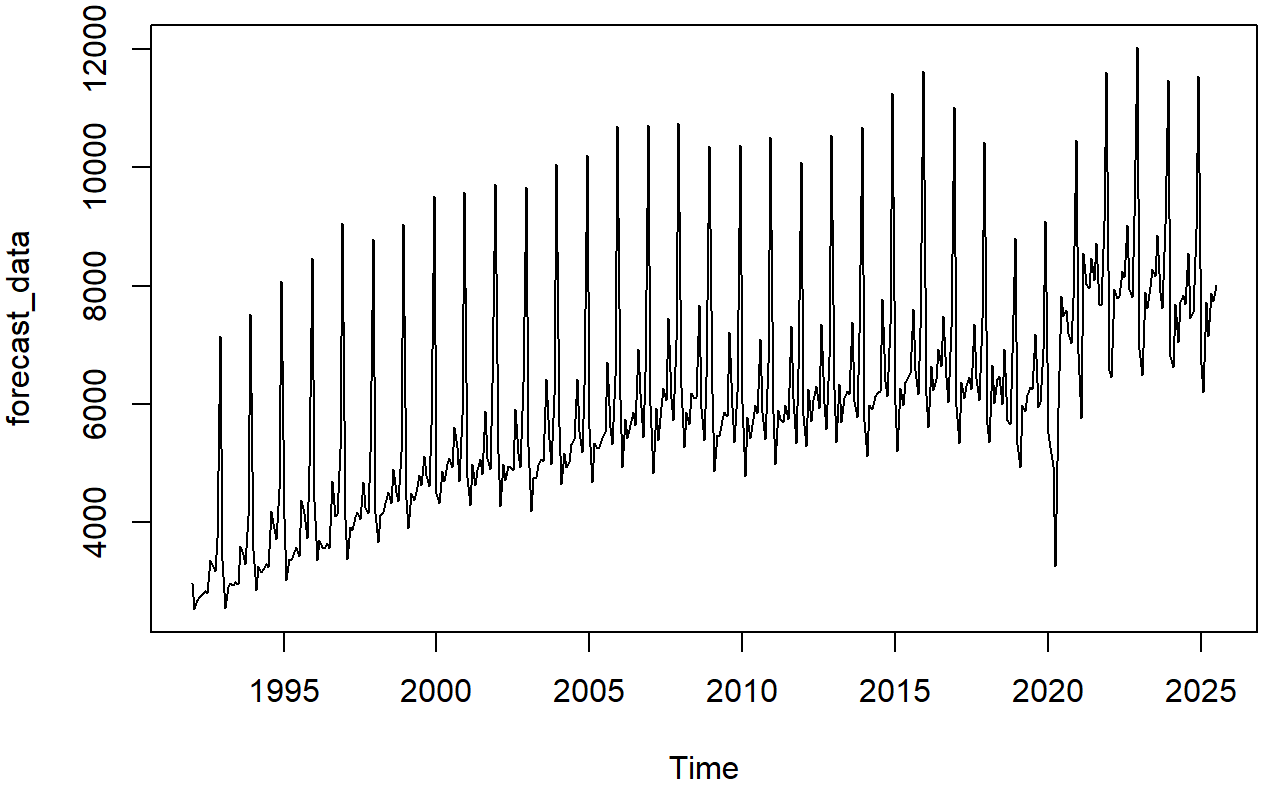
```
my_data <- my_data %>%
  rename(sales = MRTSSM451USN)
```

Comments on Import Dataset:

After importing the forecast data that will be used for analysis, I checked the beginning and end date of the dataset to know the range of dates. The beginning date in this dataset is January 1, 1992 and the end date in this dataset is July 1, 2025.

Create New Forecasting Dataset

```
my_data <- my_data %>%
  arrange(observation_date)

start_year  <- year(min(my_data$observation_date))
start_month <- month(min(my_data$observation_date))

forecast_data <- ts(
  my_data$sales,
  start     = c(start_year, start_month),
  frequency = 12
)

plot.ts(forecast_data)
```



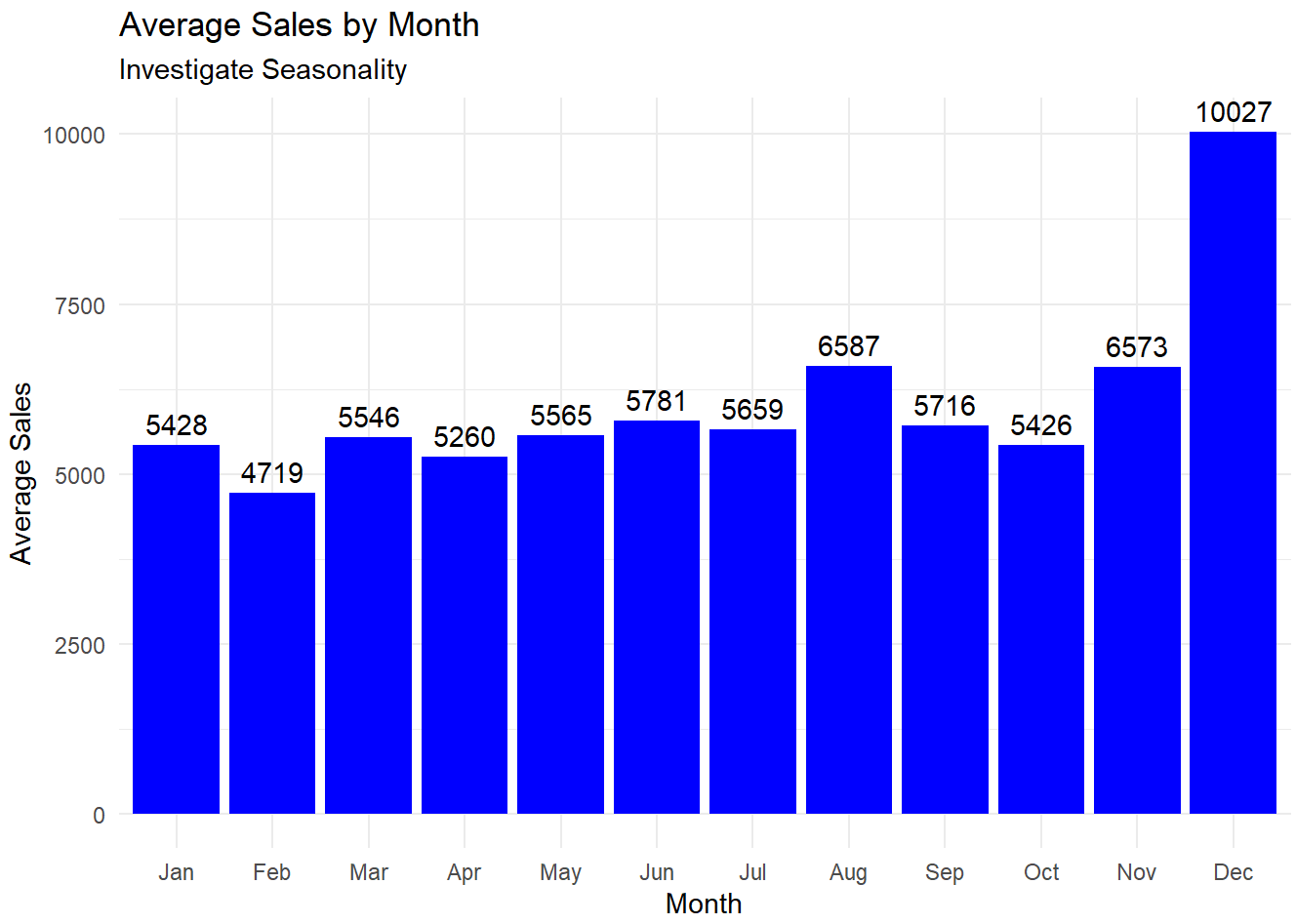
Comments on Creation of time series dataset:

Based on the graph of sales throughout the dates, it is clear there is both an upward trend and yearly seasonality during the end of each year. There is also a significant drop in sales around 2019-2021. This is due to COVID-19 and needs to be addressed during analysis for a potential structure break.

Quantitatively show the affects of seasonality on sales

```
monthly_avg <- my_data %>%
  mutate(month = month(observation_date, label = TRUE, abbr = TRUE)) %>%
  group_by(month) %>%
  summarise(avg_sales = mean(sales))

ggplot(data = monthly_avg, aes(x=month, y = avg_sales)) +
  geom_col(fill = "blue") +
  geom_text(aes(label = round(avg_sales,0)),
            vjust = -0.5) +
  labs(
    title = "Average Sales by Month",
    subtitle = "Investigate Seasonality",
    x = "Month",
    y = "Average Sales"
```

```
  ) +
  theme_minimal()
```

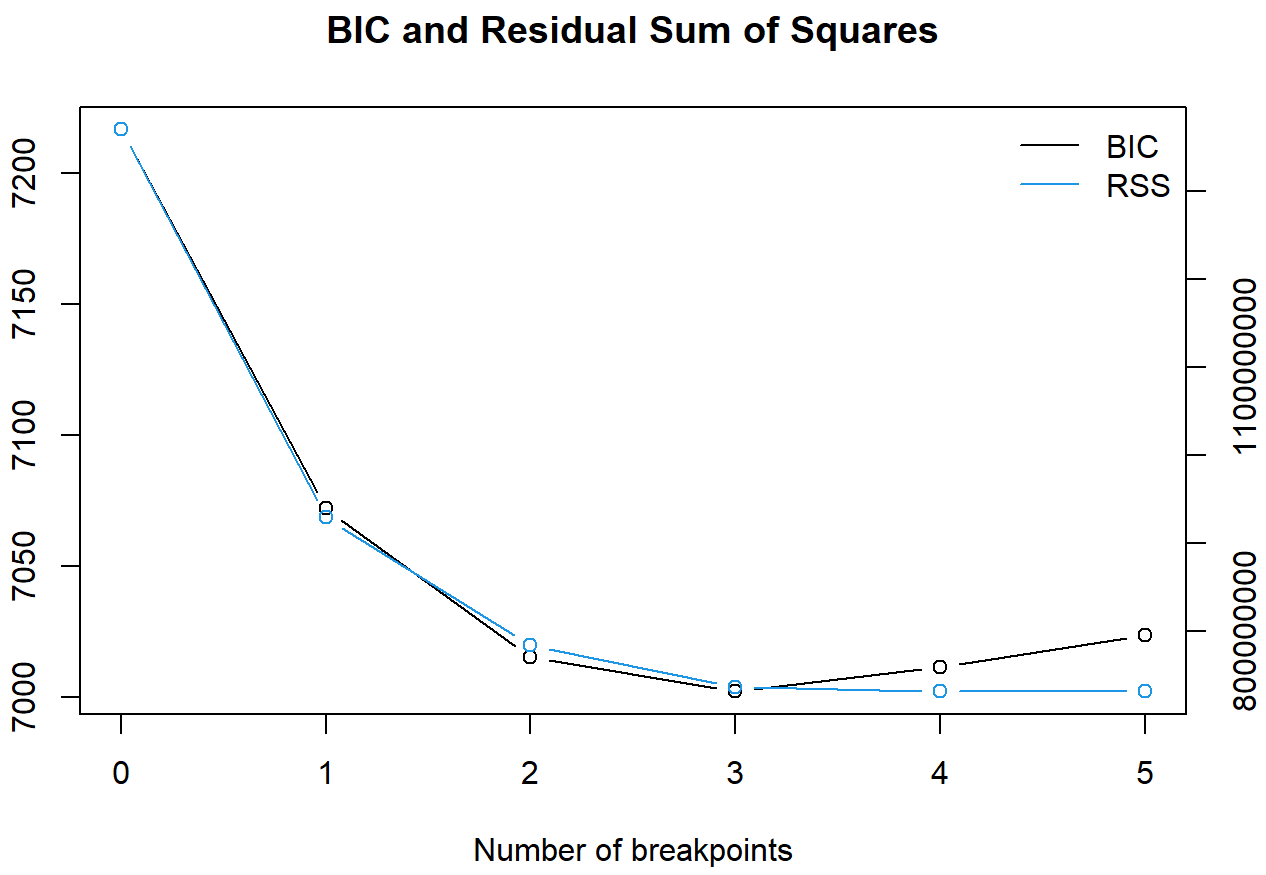


Comments on Seasonality:

It is apparent from the bar chart that December has by far the highest average sales. This is most likely due to the holiday season.

Bai-Perron, test for Structure Breaks

```
sb <- strucchange::breakpoints(forecast_data ~ 1)
plot(sb) ## Figure out what this graph reads
```




```
strucchange::breakdates(sb)
```

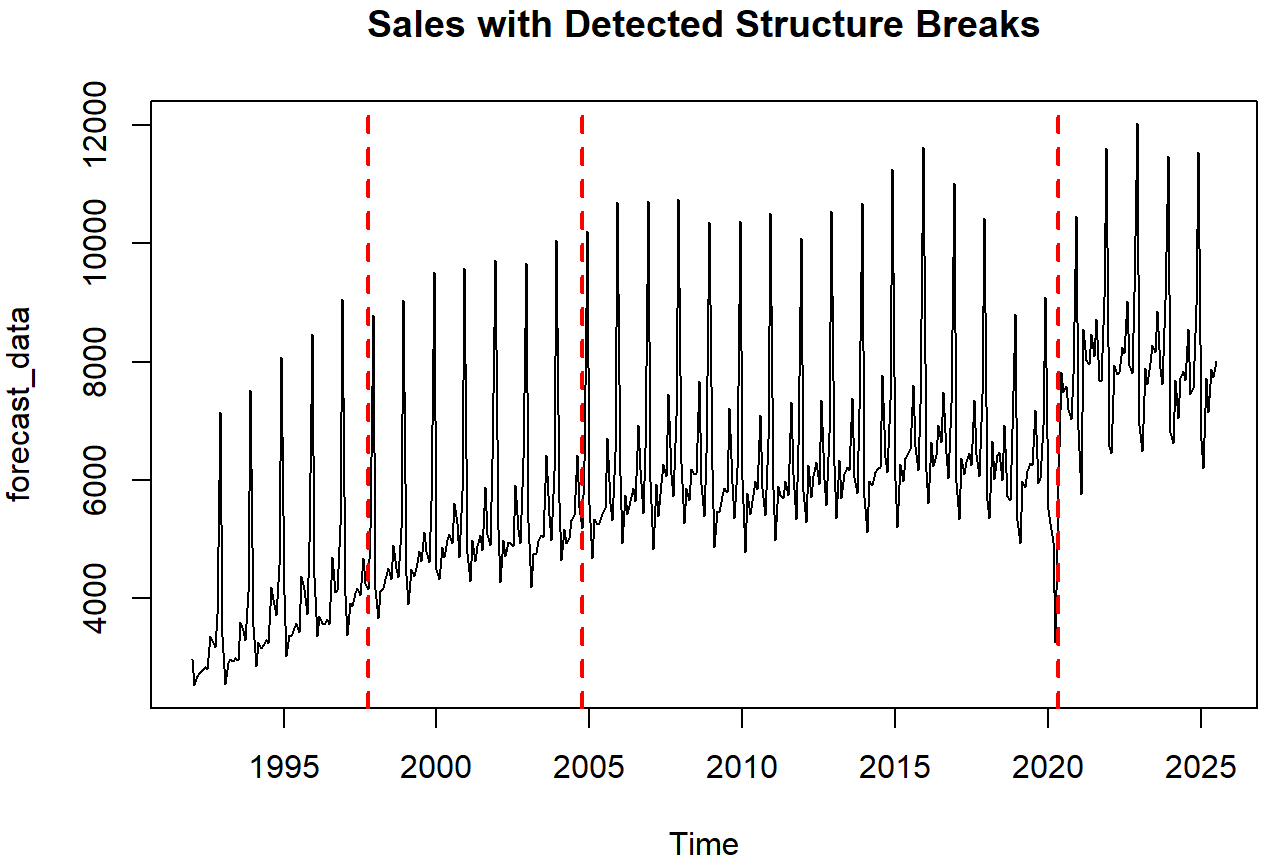
```
[1] 1997.750 2004.750 2020.333
```

```
confint(sb)
```

```
     Confidence intervals for breakpoints
     of optimal 4-segment partition:

Call:
confint.breakpointsfull(object = sb)

Breakpoints at observation number:
  2.5 % breakpoints 97.5 %
1    59          70     80
2   136         154    172
3   334         341    350

Corresponding to breakdates:
  2.5 %    breakpoints 97.5 %
1 1996(11) 1997(10)    1998(8)
```

```
2 2003(4)  2004(10)    2006(4)
3 2019(10) 2020(5)     2021(2)
```

```
plot(forecast_data, main = "Sales with Detected Structure Breaks")
lines(sb, col = "red", lwd = 2)
```



Comments on Bai-Perron Test:

The Bai-Perron test is a statistical test that shows breakpoints in the forecasting data. Breakpoints in forecast data refer to significant changes or discontinuities that occur in the data, which can affect the accuracy and reliability of forecasts. To look for breakpoints I first graphed a BIC and RSS graph. RSS, the blue line, measures model error, whereas the BIC, the black line, penalizes model complexity. The best model is the one with the lowest BIC, BIC will drop as breakpoints are added but will rise again once to many are added. This BIC and RSS graph show the optimal number of breakpoints is 3. The optimal number of breakpoints is further supported by the confidence interval table for structural break dates. The estimated breakdates for the three detected structural changes fall within the following ranges: November 1996 to August 1998, April 2003 to April 2006, and October 2019 to February 2021. These breakpoints are visually displayed on the plotted forecast series, where they appear as red vertical lines intersecting the time series curve.

Chow Test

```
chow_test <- sctest(forecast_data ~ 1, type = "Chow", point = 334)
chow_test
```

```
    Chow test

data:  forecast_data ~ 1
F = 97.836, p-value < 0.00000000000000022
```

Comments on Chow Test:

The Chow Test tests for whether a given date represents a statistically significant structural break in the time series data. Since I want the time series to start as early as possible while still being structurally consistent I will use the Chow test on October, 2019. The results of the chow test showed that October of 2019 is significant due to the extremely small p-value. What this means is for modeling my new start date will be October 1, 2019.

Data Preparation: Regressoin

Create Dummy Variable for Covid Era

```
covid_start <- c(2020, 3)   # March 2020
covid_end   <- c(2021, 12)  # December 2021

# Create the dummy on the same time index as forecast_data
covid_dummy <- ifelse(
  time(forecast_data) >= covid_start[1] + (covid_start[2] - 1) / 12 &
  time(forecast_data) <= covid_end[1] + (covid_end[2] - 1) / 12,
  1, 0
)

covid_dummy <- ts(covid_dummy,
                  start = start(forecast_data),
                  frequency = frequency(forecast_data))
```

Comments on Creating Dummy Variable for Covid Era:

A binary dummy variable was constructed to represent the COVID-19 era. Observations occurring during the defined COVID period were assigned a value of 1, and all other observations were assigned a 0. Including this variable allows the model to test whether the COVID-era period had a statistically significant effect by comparing model performance with and without the dummy term. The COVID-19 era is defined as March 2020 - December 2021 since this data range captures the full shock and full recovery from COVID-19.

Partition, forecast data

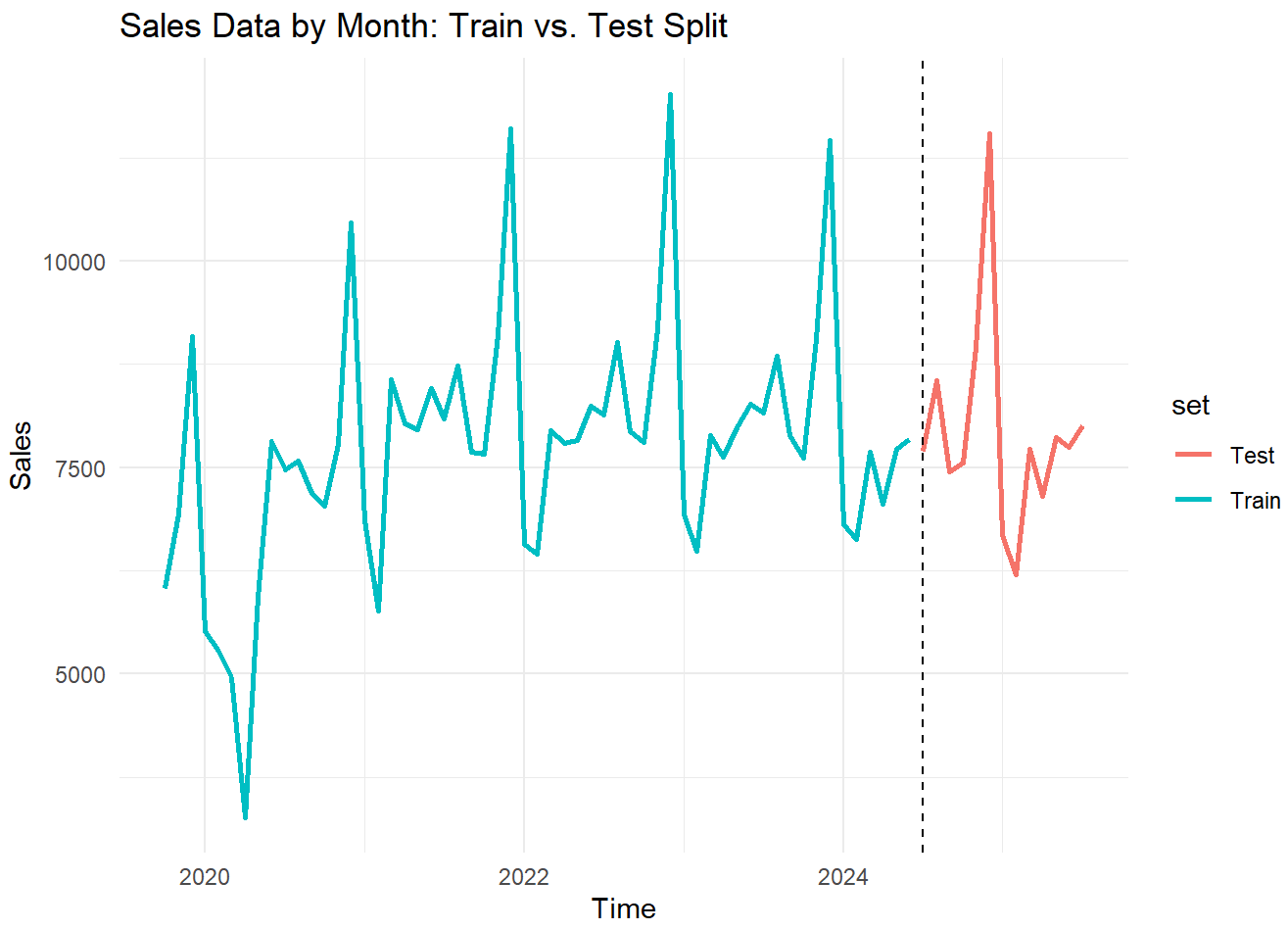
```
trainset <- window(forecast_data, start = c(2019,10), end =c(2024,6))
testset <- window(forecast_data, start = c(2024,7), end = c(2025,7))

cat("trainset has", length(trainset), "observation.")
```

```
trainset has 57 observation.
```

```
cat("\ntestset has", length(testset), "observation.")
```

```
testset has 13 observation.
```

```
plot_set <- bind_rows(data.frame(Time = time(trainset), sales = as.numeric(trainset), set =
        "Train"),
                      data.frame(Time = time(testset), sales = as.numeric(testset), set =
        "Test"))
ggplot(plot_set, aes(Time, sales, color = set)) +
  geom_line(linewidth = 1) +
  geom_vline(xintercept = 2024.5, linetype = 2) +
  labs(title = "Sales Data by Month: Train vs. Test Split",
       x = "Time", y = "Sales") +
  theme_minimal()
```



Comments on Partitioning the Forecast Data:

A training and test dataset was made for modeling. The training dataset ranged from October 2019 - June 2024. The test dataset was exactly one year, it ranged from July 2024 to the end of the time series dataset July 2025.

Partition, covid binary

```
covid_train <- window(covid_dummy, start = c(2019, 10), end = c(2024, 6))
```

Modeling: Regression

Cross Fold Validation: Regression

```
t_results <- data.frame()  # store all folds + models here

# FOLD 1:

# y (sales)
cv_train1_y <- window(trainset, end   = c(2021,12))
cv_val1_y   <- window(trainset, start = c(2022,1), end = c(2022,12))

# covid dummy (aligned)
cv_train1_c <- window(covid_train, end   = c(2021,12))
cv_val1_c   <- window(covid_train, start = c(2022,1), end = c(2022,12))

## Model 1: trend
model1_M1    <- tslm(cv_train1_y ~ trend)
forecast1_M1 <- forecast(model1_M1, h = length(cv_val1_y))
acc1_M1      <- forecast::accuracy(forecast1_M1, cv_val1_y)

t_results <- rbind(t_results,
  data.frame(Fold = 1,
             Model = "trend",
             RMSE  = acc1_M1[2,"RMSE"],
             MAPE  = acc1_M1[2,"MAPE"])
)

## Model 2: trend + season
model1_M2    <- tslm(cv_train1_y ~ trend + season)
forecast1_M2 <- forecast(model1_M2, h = length(cv_val1_y))
acc1_M2      <- forecast::accuracy(forecast1_M2, cv_val1_y)

t_results <- rbind(t_results,
  data.frame(Fold = 1,
             Model = "trend + season",
```

```
             RMSE  = acc1_M2[2,"RMSE"],
             MAPE  = acc1_M2[2,"MAPE"])
)

## Model 3: trend + season + covid
model1_M3    <- tslm(cv_train1_y ~ trend + season + cv_train1_c)
forecast1_M3 <- forecast(
  model1_M3,
  h = length(cv_val1_y),
  newdata = data.frame(cv_train1_c = cv_val1_c)
)
acc1_M3      <- forecast::accuracy(forecast1_M3, cv_val1_y)

t_results <- rbind(t_results,
  data.frame(Fold = 1,
             Model = "trend + season + covid",
             RMSE  = acc1_M3[2,"RMSE"],
             MAPE  = acc1_M3[2,"MAPE"])
)


# FOLD 2:


cv_train2_y <- window(trainset,    end   = c(2022,12))
cv_val2_y   <- window(trainset,    start = c(2023,1), end = c(2023,12))

cv_train2_c <- window(covid_train, end   = c(2022,12))
cv_val2_c   <- window(covid_train, start = c(2023,1), end = c(2023,12))

## Model 1: trend
model2_M1    <- tslm(cv_train2_y ~ trend)
forecast2_M1 <- forecast(model2_M1, h = length(cv_val2_y))
acc2_M1      <- forecast::accuracy(forecast2_M1, cv_val2_y)

t_results <- rbind(t_results,
  data.frame(Fold = 2,
             Model = "trend",
             RMSE  = acc2_M1[2,"RMSE"],
             MAPE  = acc2_M1[2,"MAPE"])
)

## Model 2: trend + season
model2_M2    <- tslm(cv_train2_y ~ trend + season)
forecast2_M2 <- forecast(model2_M2, h = length(cv_val2_y))
acc2_M2      <- forecast::accuracy(forecast2_M2, cv_val2_y)

t_results <- rbind(t_results,
  data.frame(Fold = 2,
             Model = "trend + season",
             RMSE  = acc2_M2[2,"RMSE"],
```

```
            MAPE  = acc2_M2[2,"MAPE"])
)

## Model 3: trend + season + covid
model2_M3    <- tslm(cv_train2_y ~ trend + season + cv_train2_c)
forecast2_M3 <- forecast(
  model2_M3,
  h = length(cv_val2_y),
  newdata = data.frame(cv_train2_c = cv_val2_c)
)
acc2_M3      <- forecast::accuracy(forecast2_M3, cv_val2_y)

t_results <- rbind(t_results,
  data.frame(Fold = 2,
             Model = "trend + season + covid",
             RMSE  = acc2_M3[2,"RMSE"],
             MAPE  = acc2_M3[2,"MAPE"])
)

# FOLD 3:

cv_train3_y <- window(trainset,    end   = c(2023,12))
cv_val3_y   <- window(trainset,    start = c(2024,1), end = c(2024,6))

cv_train3_c <- window(covid_train, end   = c(2023,12))
cv_val3_c   <- window(covid_train, start = c(2024,1), end = c(2024,6))

## Model 1: trend
model3_M1    <- tslm(cv_train3_y ~ trend)
forecast3_M1 <- forecast(model3_M1, h = length(cv_val3_y))
acc3_M1      <- forecast::accuracy(forecast3_M1, cv_val3_y)

t_results <- rbind(t_results,
  data.frame(Fold = 3,
             Model = "trend",
             RMSE  = acc3_M1[2,"RMSE"],
             MAPE  = acc3_M1[2,"MAPE"])
)

## Model 2: trend + season
model3_M2    <- tslm(cv_train3_y ~ trend + season)
forecast3_M2 <- forecast(model3_M2, h = length(cv_val3_y))
acc3_M2      <- forecast::accuracy(forecast3_M2, cv_val3_y)

t_results <- rbind(t_results,
  data.frame(Fold = 3,
             Model = "trend + season",
             RMSE  = acc3_M2[2,"RMSE"],
             MAPE  = acc3_M2[2,"MAPE"])
)
```

```
## Model 3: trend + season + covid
model3_M3    <- tslm(cv_train3_y ~ trend + season + cv_train3_c)
forecast3_M3 <- forecast(
  model3_M3,
  h = length(cv_val3_y),
  newdata = data.frame(cv_train3_c = cv_val3_c)
)
acc3_M3      <- forecast::accuracy(forecast3_M3, cv_val3_y)

t_results <- rbind(t_results,
  data.frame(Fold = 3,
             Model = "trend + season + covid",
             RMSE  = acc3_M3[2,"RMSE"],
             MAPE  = acc3_M3[2,"MAPE"])
)


t_results
```

```
  Fold                  Model     RMSE     MAPE
1    1                  trend 2020.458 24.94214
2    1         trend + season 1329.589 15.56136
3    1 trend + season + covid 2451.921 29.64513
4    2                  trend 1804.887 22.12157
5    2         trend + season 1190.354 14.07086
6    2 trend + season + covid 1143.092 13.40971
7    3                  trend 1950.668 26.63008
8    3         trend + season 1049.182 13.65372
9    3 trend + season + covid 1132.616 14.98762
```

Comments on Cross Fold Validation:

Three different models were tested over three different time periods. Each models RMSE and MAPE was collected from each cross validation fold. The RMSE and MAPE will be evaluated and whichever model has the lowest average RMSE and MAPE from the three fold cross validation will be the model used to forecast.

Evaluation: Regression

```
# Average CV error per model (this is what you report)
cv_summary <- t_results %>%
  group_by(Model) %>%
  summarise(
    mean_RMSE = mean(RMSE),
    mean_MAPE = mean(MAPE),
    .groups = "drop"
  )

cv_summary
```

```
# A tibble: 3 × 3
  Model                  mean_RMSE mean_MAPE
  <chr>                      <dbl>     <dbl>
1 trend                      1925.      24.6
2 trend + season             1190.      14.4
3 trend + season + covid     1576.      19.3
```

Comments on Evaluation:

The model with the lowest mean RMSE and MAPE is the model that includes just trend and season. This Model will be used to forecast the sales of August - December 2025.

Deployment: Regression

Forecasting 2025-08-01 - 2025-12-01

```
newdata2025 <- window(forecast_data, start = c(2019,5), end = c(2025,7))
new_covid <- window(covid_dummy, start = c(2019,5), end = c(2025,7))


final_model1 <- tslm(newdata2025 ~ trend + season)
summary(final_model1)
```

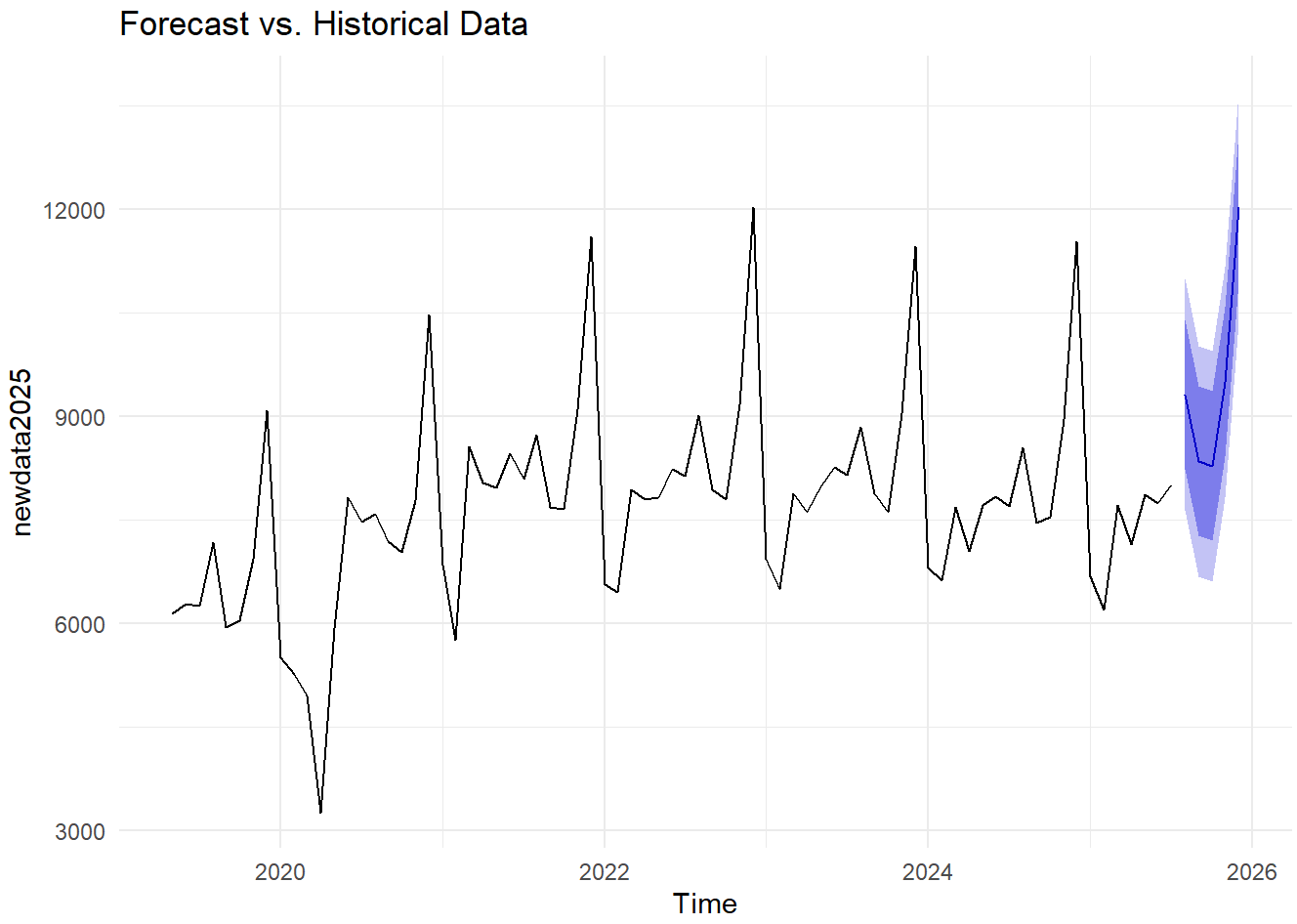
```
Call:
tslm(formula = newdata2025 ~ trend + season)

Residuals:
     Min       1Q   Median       3Q      Max
-2841.58  -436.25    61.15   468.95  1643.19

Coefficients:
            Estimate Std. Error t value             Pr(>|t|)
(Intercept) 5631.900    347.525  16.206 < 0.0000000000000002 ***
trend         23.853      4.063   5.871  0.00000018362276058 ***
season2     -449.520    437.439  -1.028              0.30812
season3      849.961    437.496   1.943              0.05659 .
season4      187.441    437.590   0.428              0.66988
season5      833.396    421.587   1.977              0.05252 .
season6     1274.258    421.529   3.023              0.00364 **
season7     1129.405    421.509   2.679              0.00943 **
season8     1876.432    437.892   4.285  0.00006478977553863 ***
season9      885.079    437.722   2.022              0.04750 *
season10     794.559    437.590   1.816              0.07424 .
season11    1985.706    437.496   4.539  0.00002658858924580 ***
season12    4495.853    437.439  10.278  0.00000000000000512 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 757.6 on 62 degrees of freedom
Multiple R-squared:  0.7752,    Adjusted R-squared:  0.7317
F-statistic: 17.82 on 12 and 62 DF,  p-value: 0.000000000000000884
```

```
future_covid <- data.frame(new_covid = rep(0, 5))

forcast <- forecast(final_model1, h = 5, newdata = future_covid)
forcast
```

```
         Point Forecast     Lo 80     Hi 80     Lo 95     Hi 95
Aug 2025       9321.159  8238.320 10403.999  7650.140 10992.179
Sep 2025       8353.659  7270.820  9436.499  6682.640 10024.679
Oct 2025       8286.993  7204.153  9369.832  6615.973  9958.012
Nov 2025       9501.993  8419.153 10584.832  7830.973 11173.012
Dec 2025      12035.993 10953.153 13118.832 10364.973 13707.012
```

```
autoplot(forcast)+
  labs(title = "Forecast vs. Historical Data") +
  theme_minimal()
```



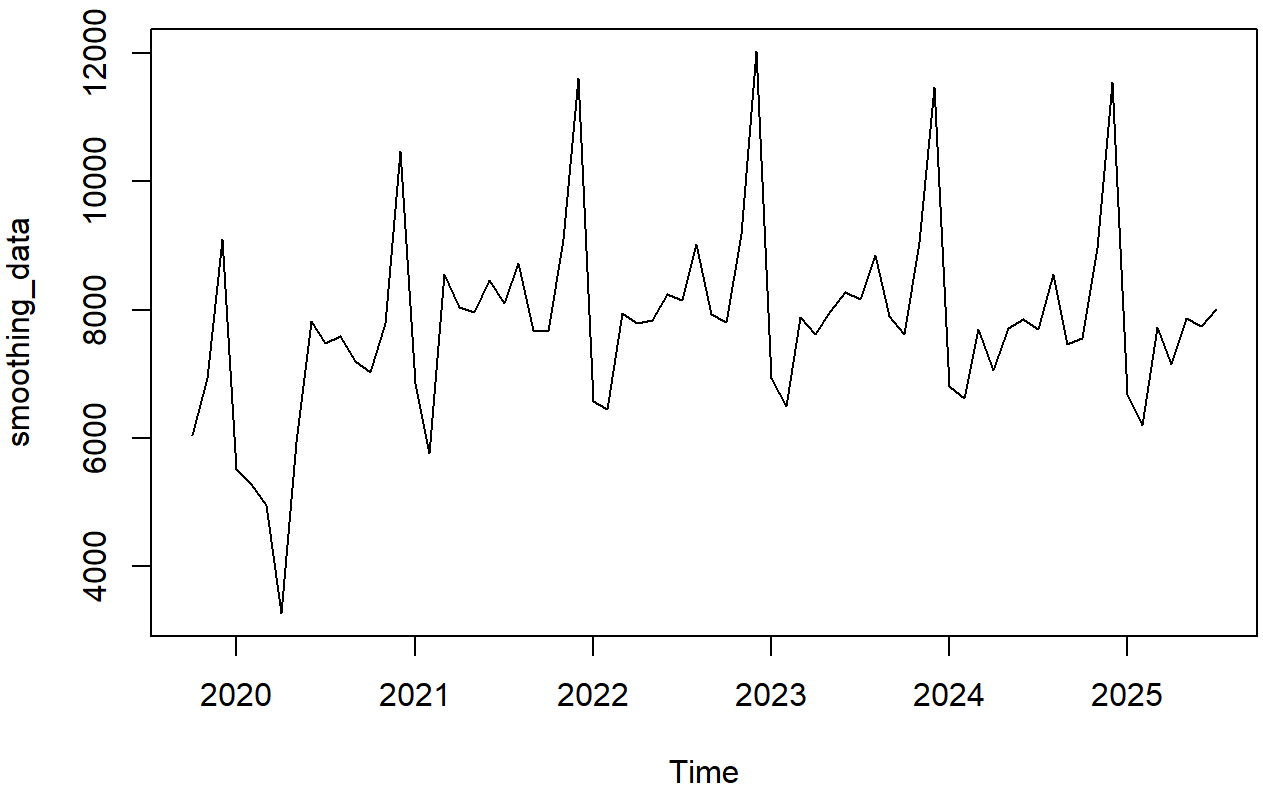
Comments on Forecasting Using Regression:

Overall, the final model accounts for 73 percent variability within the dataset. The final regression model predicts that August will have a sales value of 9,321, September will have a sales value of 8,354, October will have a sales value of 8,287, November will have a sales value of 9,502, December will have a sales value of 12,036. These estimates are reasonable as they follow a similar trend to the other Aug-Dec months in the dataset.

Data Understanding: Smoothing

Create data time period for Smoothing

```
my_data_recent <- my_data %>%
  filter(observation_date >= as.Date("2019-10-01")) %>%
  arrange(observation_date)


start_year1  <- year(min(my_data_recent$observation_date))
start_month1 <- month(min(my_data_recent$observation_date))

smoothing_data <- ts(my_data_recent$sales,
                     start = c(start_year1, start_month1), frequency = 12)
plot(smoothing_data)
```



Comments on smoothing data:

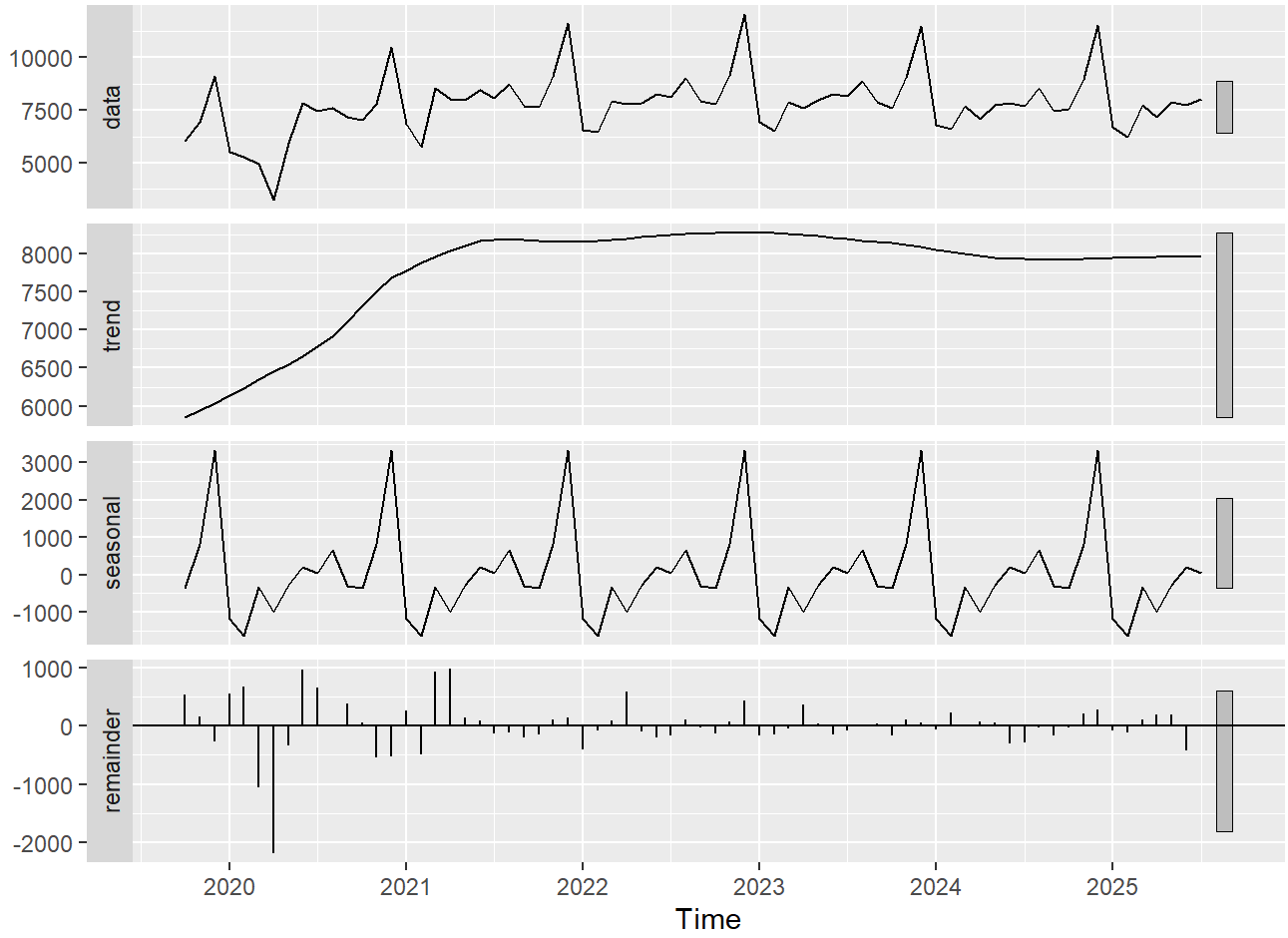
For the smoothing dataset I will use the data that is found in the most recent structure break, cutting off all dates behind this. Inluding older dates that also have their own structure breaks will distort forecast accuracy of the smoothing models. The most recent structure break and the following data is most likely to produce the most accurate forecasts.

Comments on trend and seasonality:

Based on the time-series graph, there is seasonality. This is due to the spike in sales during December. To confirm trend and or seasonality I will use another method that tests for seasonality and trend.

STL Decomposition; test for both trend and seasonality

```
fit <- stl(smoothing_data, s.window = "periodic")
autoplot(fit)
```



Comments on STL Decomposition Graphs:

Plotting STL produces four different graphs. The top graph labeled data, shows the raw time series data. The second graph labeled trend shows a strong trend upward from 2020 to mid-2022, most likely representing the shift out of COVID. The trend then flattens though 2022 to 2025, with there being a slight increase from 2024 to 2025. The third graph labeled seasonal shows a repeating yearly cycle of high sales.

The fourth graph remainder shows what the time series graph looks like when trend and seasonality is removed. Overall, it is safe to say that the period of time series data that smoothing will be done over does contain both trend and seasonality. For modeling this means Holt-Winters should be used.

Modeling: Holt-Winters Smoothing

Cross Validation - Holt-Winters Smoothing

```
s_results <- data.frame()

## FOLD 1: train <= 2021-12, validate 2022-01 .. 2022-12
s_train1 <- window(smoothing_data, end   = c(2021, 12))
s_val1   <- window(smoothing_data, start = c(2022, 1), end = c(2022, 12))

fit1_AAA <- ets(s_train1, model = "AAA", restrict = FALSE)
fc1_AAA  <- forecast(fit1_AAA, h = length(s_val1))
acc1_AAA <- accuracy(fc1_AAA, s_val1)

s_results <- rbind(s_results,
  data.frame(Fold = 1, Model = "AAA",
             RMSE = acc1_AAA[2, "RMSE"],
             MAPE = acc1_AAA[2, "MAPE"])
)

fit1_AAM <- ets(s_train1, model = "AAM", restrict = FALSE)
fc1_AAM  <- forecast(fit1_AAM, h = length(s_val1))
acc1_AAM <- accuracy(fc1_AAM, s_val1)

s_results <- rbind(s_results,
  data.frame(Fold = 1, Model = "AAM",
             RMSE = acc1_AAM[2, "RMSE"],
             MAPE = acc1_AAM[2, "MAPE"])
)

## FOLD 2: train <= 2022-12, validate 2023
s_train2 <- window(smoothing_data, end   = c(2022, 12))
s_val2   <- window(smoothing_data, start = c(2023, 1), end = c(2023, 12))

fit2_AAA <- ets(s_train2, model = "AAA", restrict = FALSE)
fc2_AAA  <- forecast(fit2_AAA, h = length(s_val2))
acc2_AAA <- accuracy(fc2_AAA, s_val2)

s_results <- rbind(s_results,
  data.frame(Fold = 2, Model = "AAA",
             RMSE = acc2_AAA[2, "RMSE"],
             MAPE = acc2_AAA[2, "MAPE"])
)
```

```
fit2_AAM <- ets(s_train2, model = "AAM", restrict = FALSE)
fc2_AAM  <- forecast(fit2_AAM, h = length(s_val2))
acc2_AAM <- accuracy(fc2_AAM, s_val2)

s_results <- rbind(s_results,
  data.frame(Fold = 2, Model = "AAM",
             RMSE = acc2_AAM[2, "RMSE"],
             MAPE = acc2_AAM[2, "MAPE"])
)

## FOLD 3: train <= 2023-12, validate 2024-01 .. 2024-06
s_train3 <- window(smoothing_data, end   = c(2023, 12))
s_val3   <- window(smoothing_data, start = c(2024, 1), end = c(2024, 6))

fit3_AAA <- ets(s_train3, model = "AAA", restrict = FALSE)
fc3_AAA  <- forecast(fit3_AAA, h = length(s_val3))
acc3_AAA <- accuracy(fc3_AAA, s_val3)

s_results <- rbind(s_results,
  data.frame(Fold = 3, Model = "AAA",
             RMSE = acc3_AAA[2, "RMSE"],
             MAPE = acc3_AAA[2, "MAPE"])
)

fit3_AAM <- ets(s_train3, model = "AAM", restrict = FALSE)
fc3_AAM  <- forecast(fit3_AAM, h = length(s_val3))
acc3_AAM <- accuracy(fc3_AAM, s_val3)

s_results <- rbind(s_results,
  data.frame(Fold = 3, Model = "AAM",
             RMSE = acc3_AAM[2, "RMSE"],
             MAPE = acc3_AAM[2, "MAPE"])
)

s_results
```

```
  Fold Model      RMSE      MAPE
1    1   AAA 1869.4543 21.501937
2    1   AAM 1752.9373 19.919131
3    2   AAA 1406.0059 16.153866
4    2   AAM  397.0064  3.860729
5    3   AAA  638.2105  7.069445
6    3   AAM  387.9680  4.073419
```

Comments on Cross Fold Validation:

Two different types of Holt-Winter models were tested. The AAA smoothing method is the additive version of the Holt-Winters seasonal smoothing and AAM is the multiplicative Holt-Winters smoothing. The AAA model is used when seasonality stays constant and AAM is used when the size of seasonal swing grows or shrinks as the time changes. Each smoothing models RMSE and MAPE was collected from each cross

validation fold. The RMSE and MAPE will be evaluated and whichever model has the lowest average RMSE and MAPE from cross validation will be the model used to forecast.

Evaluation: Holt-Winters Smoothing

```
s_results_summary <- s_results %>%
  group_by(Model) %>%
  summarise(
    mean_RMSE = mean(RMSE),
    mean_MAPE = mean(MAPE),
    .groups = "drop"
)
s_results_summary
```

```
# A tibble: 2 × 3
  Model mean_RMSE mean_MAPE
  <chr>     <dbl>     <dbl>
1 AAA       1305.     14.9
2 AAM        846.      9.28
```

Comments on Evaluation:

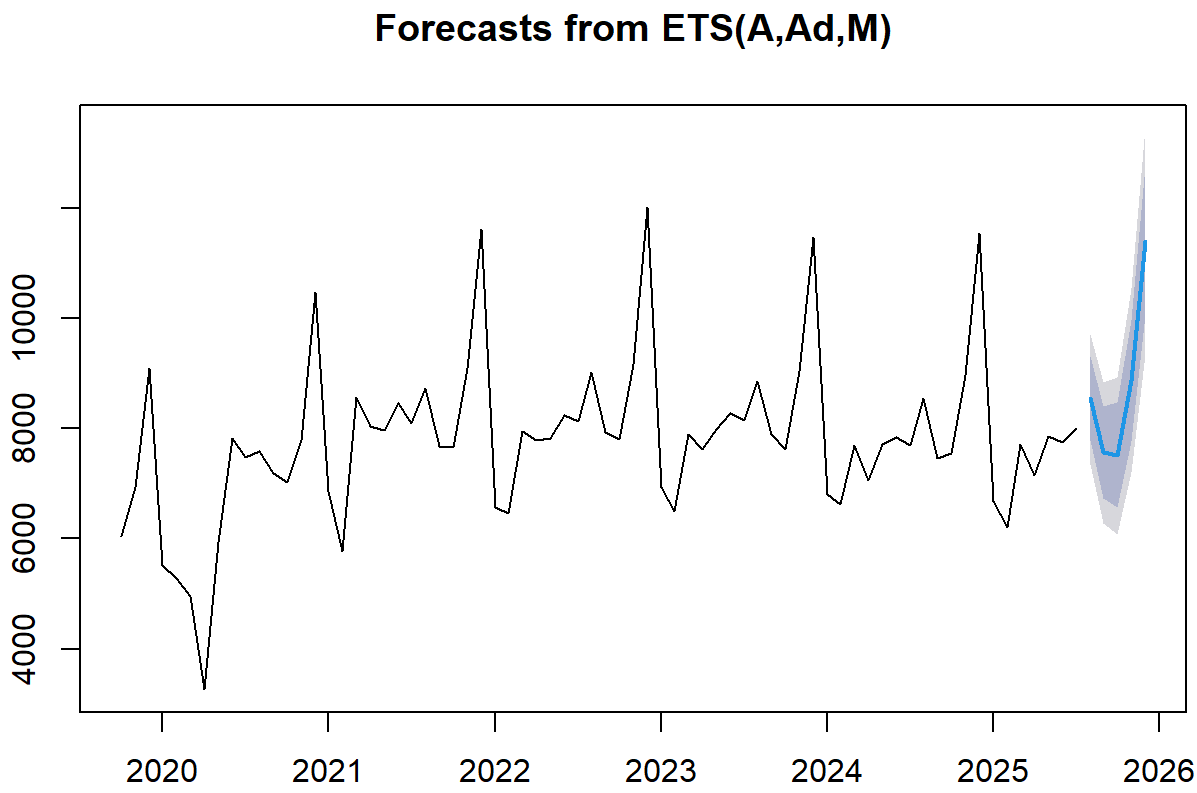
The Model with the lowest RMSE and MAPE is the AAM version of Holt-Winters. This model will be used to forecast the sales of August - December 2025.

Deployment - Forecast sales for August - December 2025

```
sfinal <- ets(smoothing_data, model = "AAM", restrict = FALSE)
sf_fcst <- forecast::forecast(sfinal, h=5)
sf_fcst
```

```
         Point Forecast     Lo 80     Hi 80    Lo 95     Hi 95
Aug 2025       8544.612  7787.469  9320.988 7360.140  9706.227
Sep 2025       7575.036  6734.312  8409.806 6277.007  8839.335
Oct 2025       7512.327  6571.779  8468.210 6075.740  8935.495
Nov 2025       8869.347  7766.350  9962.485 7209.020 10546.128
Dec 2025      11392.514 10055.070 12779.738 9342.565 13461.565
```

```
plot(sf_fcst)
```



Comments on Forecasting using Smoothing:

The Final AAM Holt-Winters smoothing model predicts that August will have a sales value of 8545, September will have a sales value of 7,575, October will have a sales value of 7,512, November will have a sales value of 8,869, December will have a sales value of 11,393. These estimates are reasonable as they follow a similar trend to the other Aug-Dec months in the dataset.

Which is the Better Model: Regresion vs Holt-Winters Smoothing

```
model_compare <- data.frame(
  Model = c("Regression", "Holt-Winters AAM"),
  RMSE  = c(mean(c(acc1_M2[2,"RMSE"], acc2_M2[2,"RMSE"], acc3_M2[2,"RMSE"])),
            mean(c(acc1_AAM[2,"RMSE"], acc2_AAM[2,"RMSE"], acc3_AAM[2,"RMSE"]))),
  MAPE  = c(mean(c(acc1_M2[2,"MAPE"], acc2_M2[2,"MAPE"], acc3_M2[2,"MAPE"])),
            mean(c(acc1_AAM[2,"MAPE"], acc2_AAM[2,"MAPE"], acc3_AAM[2,"MAPE"])))
)

model_compare
```

```
            Model      RMSE      MAPE
1      Regression 1189.7087 14.428648
```

```
2 Holt-Winters AAM  845.9706  9.284426
```

Comments on Best Model:

Based off the best model for both Regression and Holt-Winters Smoothing, Holt-Winters AAM should be used to forecast and not the best regression model. Holt-Winters AAM is preferred over regression for forecasting August-December 2025 because it has a lower RMSE and MAPE. This means the Holt-Winters AAM has lower average error producing more accurate sales numbers.