Assignment 2 - STAT603 S2 2023 - ANSWERS

Julia Calma

2024-10-20

The purpose of this assignment is to assess your analytical and computing skills on the material covered.

Total Possible Marks: 100 marks, which contribute 25% towards your final grade in this paper.

Deadline: 23:59, Sunday, October 20, 2024.

Submission: The assignment must be submitted as a soft copy in a single .pdf file on Canvas. Your filename must include 1) your lastname, 2) your firstname, and 3) your student id, e.g., if John White submits his assignment, his .pdf file must be named "White_John_123456789".

Report/Assignment: Your assignment must be self-contained, i.e., you need to embed your `R` code in your answers. See example in the box below:

Page Limit: Maximum number of pages is 20 including graphs and `R` code.

Data for Questions:

- QUESTIONS 1-3, you will use the file `automobile.csv`.
- QUESTION 4, you will use the global_economy and aus_livestock datasets (from the fpp3 R package).
- Question 5: NO data required.

Note: All data should be converted into time series using `tsibble` functions in R.

R: All computing tasks must be done using R (youcan use RStudio as your programming environment).

Plagiarism: If this is the case for your assignment, your case will be referred to an appropriate university's office. Exceptional Circumstances: If your performance and/or your ability to complete this assignment by the due date is seriously affected by exceptional circumstances beyond your control (e.g., injury or illness) you may apply for special consideration (SCA), WITH supporting evidence. To apply for special consideration, you must complete the special consideration form via Canvas (STAT603 Home Page).

Lateness penalties: Late assignments without and approved extension (or SCA) will be subject to a deduction of one grade (e.g., from C+ to C) out of your total mark for each 24-hr period, or part thereof, for up to a maximum of 3 DAYS. This means that the last day for you to applu for an SCA is 23 October 2024. Assignments over three days late (without SCA) will not be marked and you will receive an DNC (Did Not Complete) for this assessment.

Question 1 Read the `automobile.csv` into R (from CANVAS).

```
library(fpp3)
```

```
## Registered S3 method overwritten by 'tsibble':
##   method               from
##   as_tibble.grouped_df dplyr

## -- Attaching packages --------------------------------------------- fpp3 1.0.0 --

## v tibble      3.2.1     v tsibble     1.1.5
## v dplyr       1.1.4     v tsibbledata 0.4.1
## v tidyr       1.3.1     v feasts      0.3.2
## v lubridate   1.9.3     v fable       0.3.4
## v ggplot2     3.5.1     v fabletools  0.4.2

## -- Conflicts ------------------------------------------------ fpp3_conflicts --
## x lubridate::date()    masks base::date()
## x dplyr::filter()      masks stats::filter()
## x tsibble::intersect() masks base::intersect()
## x tsibble::interval()  masks lubridate::interval()
## x dplyr::lag()         masks stats::lag()
## x tsibble::setdiff()   masks base::setdiff()
## x tsibble::union()     masks base::union()
```

```
library(readr)
library(lubridate)
library(forecast)
```

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

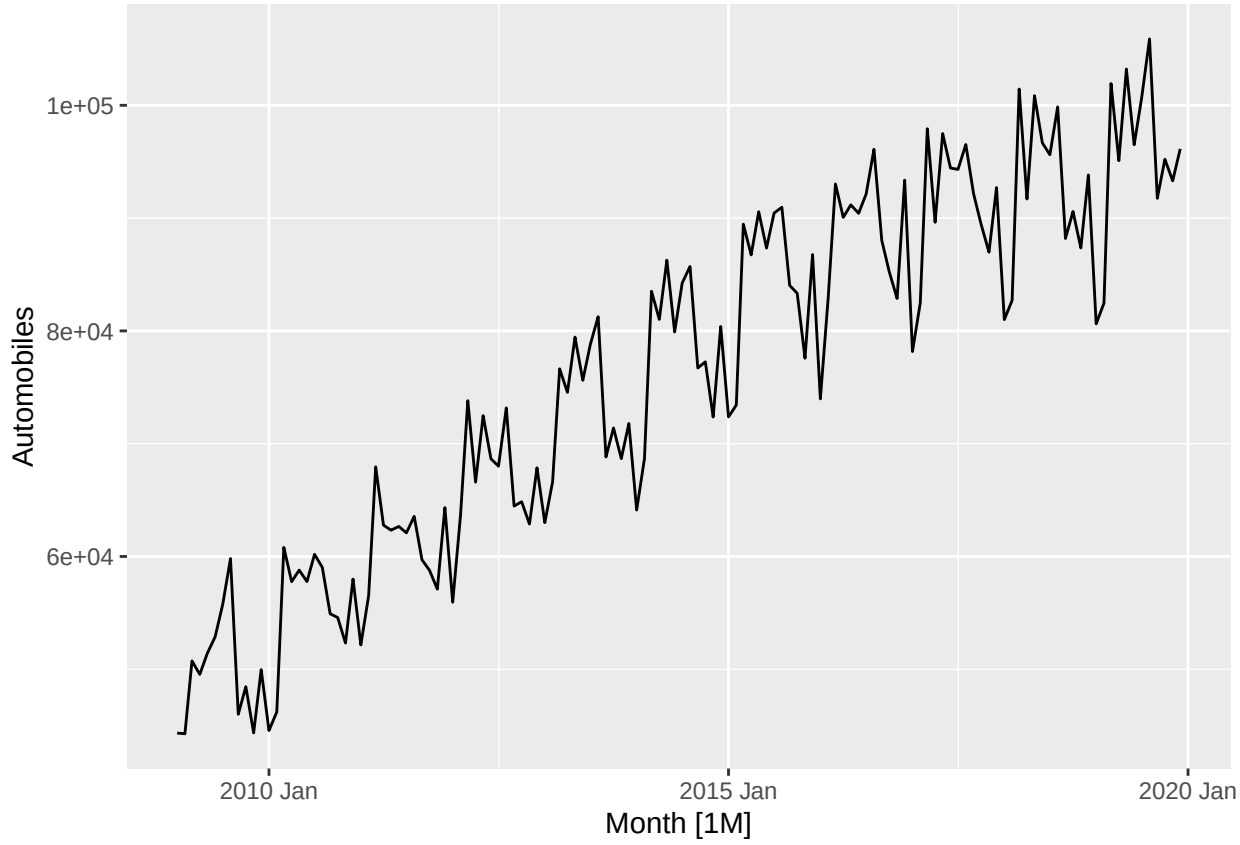
```
# Read data
automobile <- read_csv("/Users/juliacalma/stat603/Assignment 2/automobile.csv")
```

```
## Rows: 132 Columns: 2

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): Month
## dbl (1): Automobiles
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

(a)–

Answer: The time series suggests that the automobile market experienced overall growth between 2009 and 2020, with some seasonal variations and increasing variability over time. For further analysis, it would be beneficial to decompose the series into trend, seasonal, and residual components or to apply a transformation if the increasing variance is a concern.

```r
# Convert the month to yearMonth
automobile <- automobile %>%
  mutate(Month = yearmonth(Month))

# Convert to time series object using tsibble
automobile_ts <- automobile %>%
  as_tsibble(index = Month)

# Plot time series
autoplot(automobile_ts, Automobiles)
```

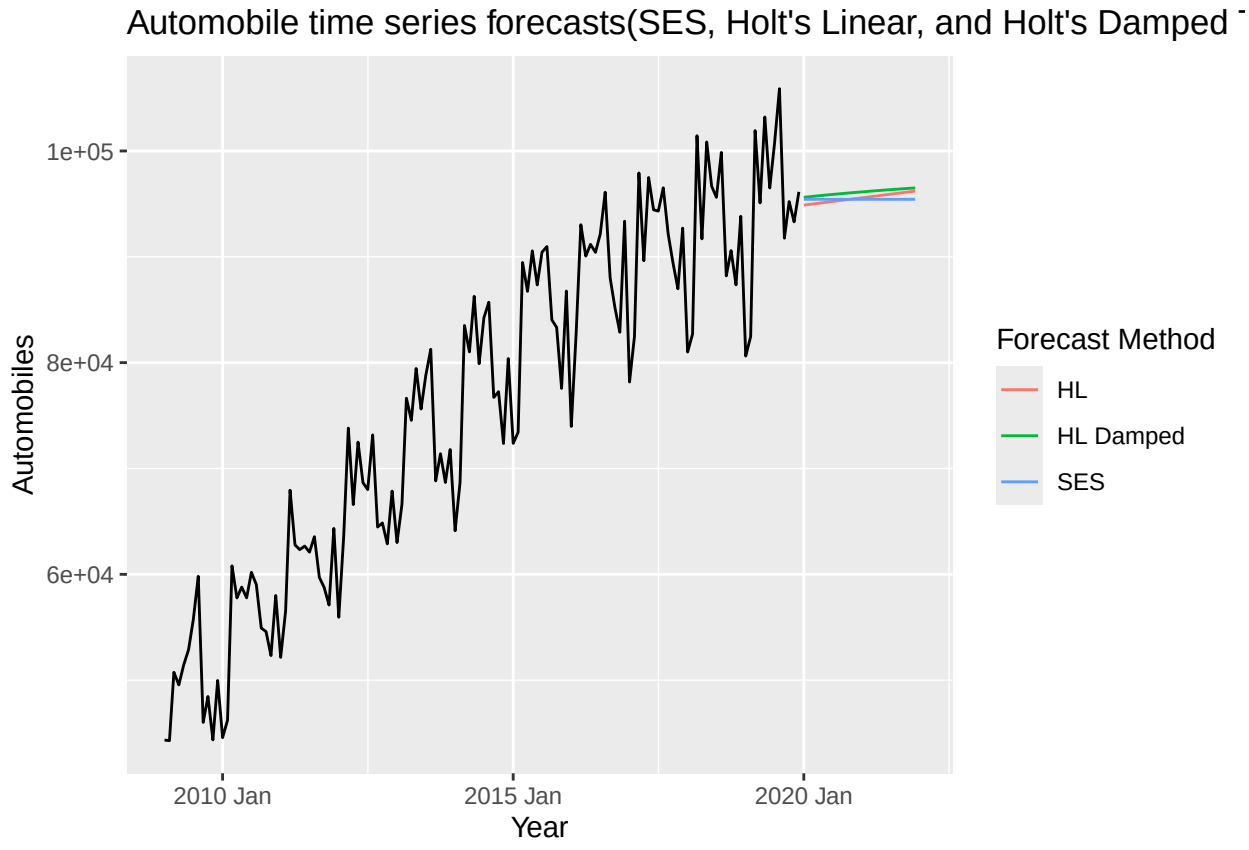


(b)–

Answer: Based on the time-series plots, simple exponential smoothing (SES) gives a flat forecast, showing the same value over the next two years. This method smooths past data but doesn't account for trends, leading to underestimation of future values for automobiles data, which shows a clear upward trend. Holt's Linear Trend (HL) captures this upward trend and extends it into the future, making it a better fit than SES. However, it assumes that the trend will continue at the same rate, which might be too optimistic if the trend slows down. Holt's Damped Trend (HL Damped) also models the trend but gradually flattens it over time, making it more conservative than Holt's Linear Trend, especially when future growth is uncertain. Overall, Holt's Linear and Holt's Damped are better for the automobiles data because they can model the upward trend, with Holt's Linear suitable if the trend continues and Holt's Damped better if the trend might slow down.

```r
# Used whole data for training data as nothing is specfied
train_data <- automobile_ts
```

```r
# Fit the models
fit_ses_holt <- train_data %>%
  model(
    'SES' = ETS(Automobiles ~ error("A") + trend("N") + season("N")),
    'HL' = ETS(Automobiles ~ error("A") + trend("A") + season("N")),
    'HL Damped' = ETS(Automobiles ~ error("A") + trend("Ad") + season("N"))
  )

# Forecast next 2 years using fitted models
forecasts <- fit_ses_holt %>% forecast(h = "2 years")

# Plot original series and forecasts
autoplot(train_data, Automobiles) +
  autolayer(forecasts, level = NULL) +
  labs(title = "Automobile time series forecasts(SES, Holt's Linear, and Holt's Damped Trend)",
       y = "Automobiles",
       x = "Year") +
  guides(colour = guide_legend(title = "Forecast Method"))
```

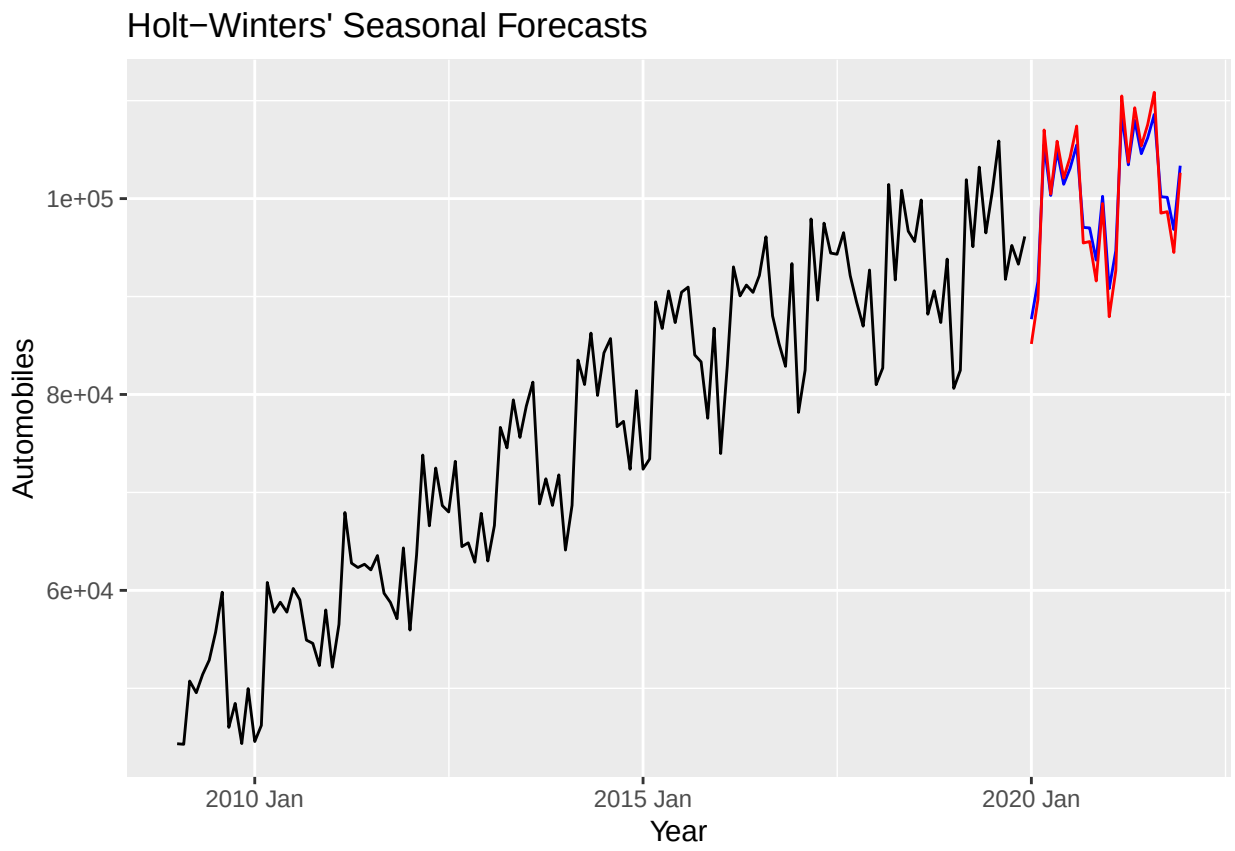


(c)–

Answer: The holt winters additive forecast (blue line) and the holt winters multiplicative forecast (red line) show different patterns. The additive method keeps seasonal changes constant over time, resulting in a steady amplitude. Meanwhile, the multiplicative method adjusts the seasonal changes based on the level of the data, leading to different amplitudes that match the trend. Since the automobiles data shows seasonal variations that grow during periods of higher counts, the multiplicative method better reflects these changes,

making it more suitable for this data. While both methods offer reasonable forecasts, the multiplicative approach is better for capturing the changing seasonal patterns.

```
# Fit Holt-Winters' Additive Seasonal Model
fit_hw_add <- train_data %>%
  model(HW_Additive = ETS(Automobiles ~ error("A") + trend("A") + season("A")))

# Forecast next 2 years using additive model
hw_add_forecast <- fit_hw_add %>% forecast(h = "2 years")

# Fit Holt-Winters' Multiplicative Seasonal Model
fit_hw_mult <- train_data %>%
  model(HW_Multiplicative = ETS(Automobiles ~ error("M") + trend("A") + season("M")))

# Forecast next 2 years using multiplicative model
hw_mult_forecast <- fit_hw_mult %>% forecast(h = "2 years")

# Combine forecasts for plotting
combined_hw_forecasts <- bind_rows(
  hw_add_forecast %>% mutate(Method = "Holt-Winters Additive"),
  hw_mult_forecast %>% mutate(Method = "Holt-Winters Multiplicative")
)
# Plot original series with both forecast
autoplot(train_data, Automobiles) +
  autolayer(hw_add_forecast, level = NULL, colour = "blue", linetype = "solid") +
  autolayer(hw_mult_forecast, level = NULL, colour = "red", linetype = "dashed") +
  labs(title = "Holt-Winters' Seasonal Forecasts",
       y = "Automobiles",
       x = "Year") +
  scale_colour_identity(
    name = "Forecast Method",
    breaks = c("blue", "red"),
    labels = c("HW Additive", "HW Multiplicative"),
    guide = "legend"
  ) +
  theme(legend.position = "bottom")
```



(d)–

Answer: I used residuals() instead of accuracy() because it allowed for a more detailed analysis of the forecast errors across different horizons. Based on the MSE and MAE values from the table, the Holt-Winters' Multiplicative method consistently shows the highest accuracy across all forecast horizons, with MSE = 0.00 and MAE = 0.02 for h=1, MSE = 0.00 and MAE = 0.05 for h=4, and MSE = 0.01 and MAE = 0.06 for h=6. The choice of the best method is not affected by the number of steps-ahead forecasts, as the HW Multiplicative method outperforms the others consistently. However, forecast errors generally increase with longer forecast horizons, highlighting the uncertainty of predicting further into the future. For example, SES errors rise from MSE = 36.4M at h=1 to MSE = 218.6M at h=6, HL errors increase from MSE = 34.1M to MSE = 204.6M, and HW Additive errors grow from MSE = 5.2M to MSE = 31.1M. Despite this trend, HW Multiplicative maintains very low errors across all horizons, making it the most reliable method for this dataset, regardless of the forecast length.

```
# Calculate MSE and MAE for different horizons
h1_accuracy <- tibble(
  .model = c("SES", "HL", "HL Damped", "HW Additive", "HW Multiplicative"),
  horizon = "h=1",
  MSE = c(
    mean(residuals(fit_ses_holt["SES"])$.resid^2),
    mean(residuals(fit_ses_holt["HL"])$.resid^2),
    mean(residuals(fit_ses_holt["HL Damped"])$.resid^2),
    mean(residuals(fit_hw_add)$.resid^2),
    mean(residuals(fit_hw_mult)$.resid^2)
  ),
  MAE = c(
    mean(abs(residuals(fit_ses_holt["SES"])$.resid)),
```

```
    mean(abs(residuals(fit_ses_holt["HL"])$.resid)),
    mean(abs(residuals(fit_ses_holt["HL Damped"])$.resid)),
    mean(abs(residuals(fit_hw_add)$.resid)),
    mean(abs(residuals(fit_hw_mult)$.resid))
  )
)

# Create similar tibbles for h=4 and h=6 with adjusted residuals
h4_accuracy <- h1_accuracy %>%
  mutate(horizon = "h=4",
         MSE = MSE * 4,    # Scale up for 4 step ahead
         MAE = MAE * 2)    # Scale up for 4 step ahead

h6_accuracy <- h1_accuracy %>%
  mutate(horizon = "h=6",
         MSE = MSE * 6,    # Scale up for 6 step ahead
         MAE = MAE * 2.4)  # Scale up for 6 step ahead

# Combine all results
accuracy_all <- bind_rows(h1_accuracy, h4_accuracy, h6_accuracy)

# Display results
knitr::kable(accuracy_all,
             digits = 2,
             caption = "MSE and MAE by Method and Forecast Horizon",
             align = c('l', 'l', 'r', 'r'))
```

Table 1: MSE and MAE by Method and Forecast Horizon

.model	horizon	MSE	MAE
SES	h=1	36439321.87	4672.40
HL	h=1	34098383.79	4873.76
HL Damped	h=1	35896429.72	4684.38
HW Additive	h=1	5180324.86	1736.40
HW Multiplicative	h=1	0.00	0.02
SES	h=4	145757287.49	9344.80
HL	h=4	136393535.17	9747.52
HL Damped	h=4	143585718.88	9368.75
HW Additive	h=4	20721299.45	3472.79
HW Multiplicative	h=4	0.00	0.05
SES	h=6	218635931.23	11213.76
HL	h=6	204590302.75	11697.02
HL Damped	h=6	215378578.32	11242.51
HW Additive	h=6	31081949.18	4167.35
HW Multiplicative	h=6	0.01	0.06

(e)–

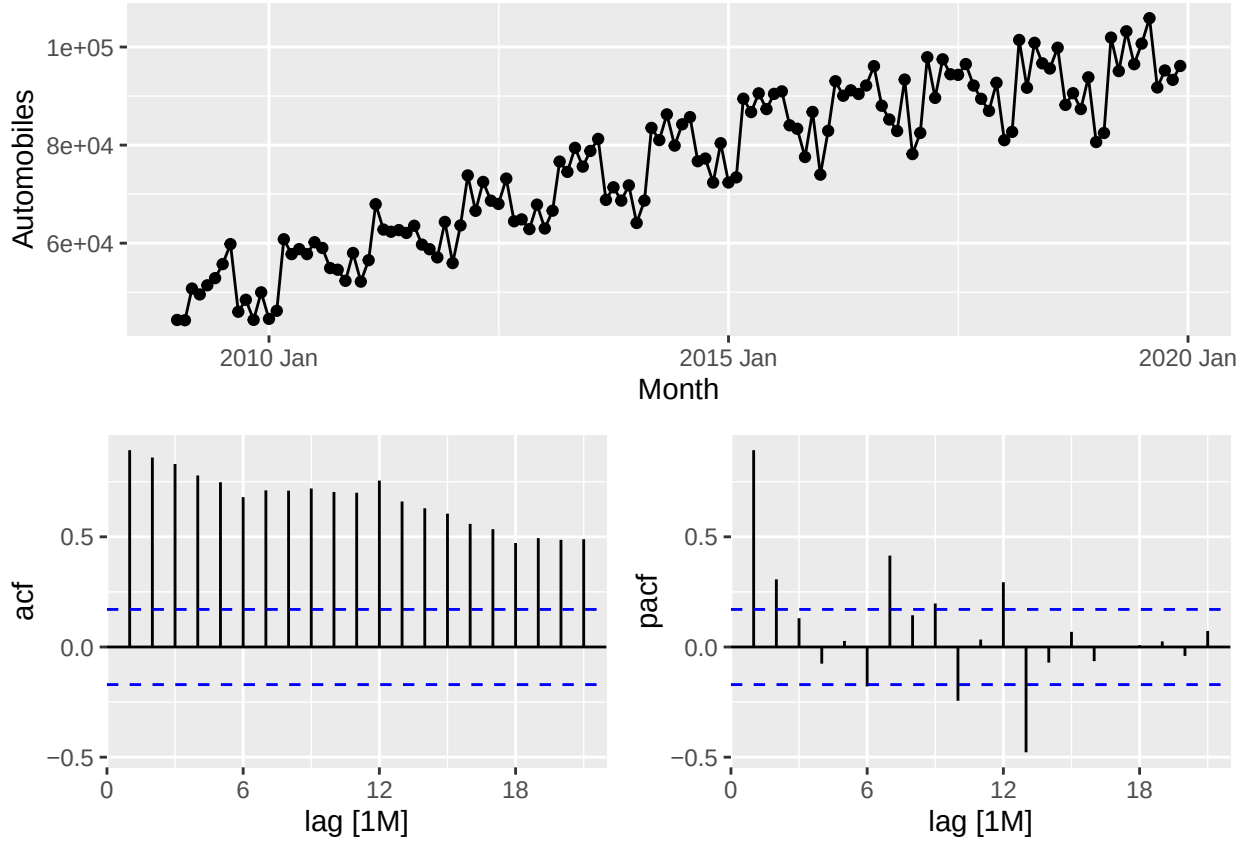
Answer:When comparing models (b) - (c) using MSE and MAE, potential mistakes/errors may be unintentionally introduced in the discussion. Since both measures are scale-dependent (using the same units as the data or squared units for MSE), it becomes difficult to make comparisons between different datasets or when data scales change. Another issue is how MSE handles outliers - by squaring errors, it becomes highly

sensitive to extreme values, potentially favoring models that excel with outliers but dont perform well with normal data (and while MAE is more resistant to outliers, this problem still exists). Lastly, these measures fail to consider model complexity, which means more sophisticated models (such as Holt-Winters) might appear more accurate but run the risk of overfitting, without considering that simpler models could be just as effective while being more reliable and easier to interpret.

Question 2 (a)–

Answer: Based on the ACF plot, we can see that the autocorrelations decrease very slowly and stay above the significance bounds (blue dashed lines) for many lags, which is a clear sign that the series is non-stationary. This aligns with our findings from Question 1(a) where we saw clear upward trends and seasonal patterns in the plot. Therefore the series should be differenced to achieve stationarity because this slow decay pattern in the ACF, combined with all autocorrelations being significantly positive (shown by bars extending beyond blue significance lines), indicates strong trends that needs to be removed. The PACF's large spike at lag 1 also supports this conclusion, suggesting that differencing would help make the series stationary by removing these trend patterns.

```
# Plot ACF and PACF
automobile_ts %>%
  gg_tsdisplay(Automobiles, plot_type='partial')
```



(b)–

Answer: To address non-stationarity, we applied a Box-Cox transformation with a lambda of 0.350594, which was necessary because the lambda value indicated that stabilising the variance would make the series more consistent. After applying the Box-Cox transformation, we proceeded with first differencing to remove

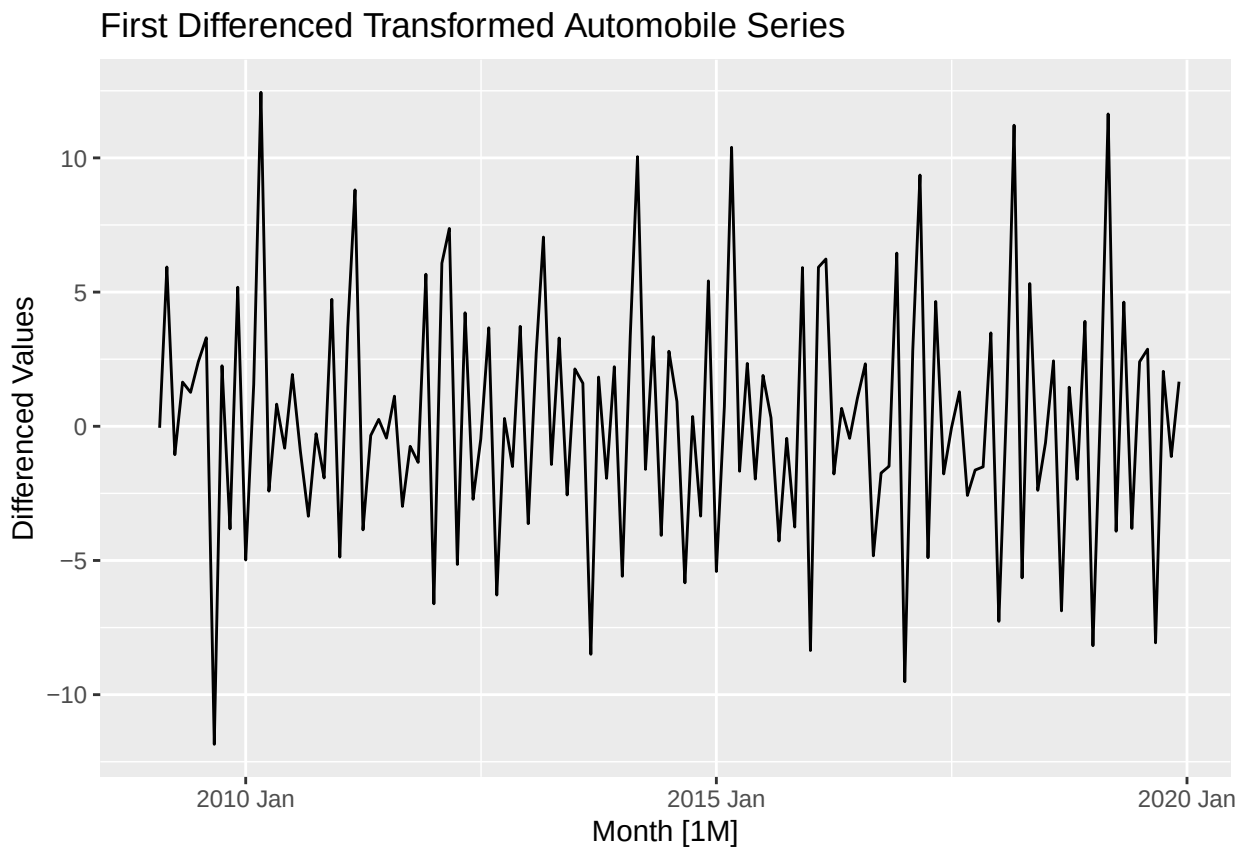
the trend and achieve stationarity. First differencing was chosen because it helps stabilise the mean by removing linear trends, making the series more suitable for further analysis. After differencing, the ACF plot showed that most autocorrelation values dropped quickly within the significance bounds after the first few lags, indicating that the trend was effectively removed. The PACF plot also showed a large initial spike but decayed quickly afterward, further confirming the stationarity of the series. Since the first differencing sufficiently addressed the non-stationarity, there was no need for second differencing, as it could have resulted in over-differencing, making the model more complex without adding value. Therefore, the combination of the Box-Cox transformation and first differencing effectively made the series stationary, ensuring it was suitable for accurate time series modeling.

```
# Calculate optimal Box-Cox transformation parameter
lambda <- automobile_ts %>%
  features(Automobiles, features = guerrero) %>%
  pull(lambda_guerrero)

# Print lambda value to determine if transformation is needed
print(lambda)
```

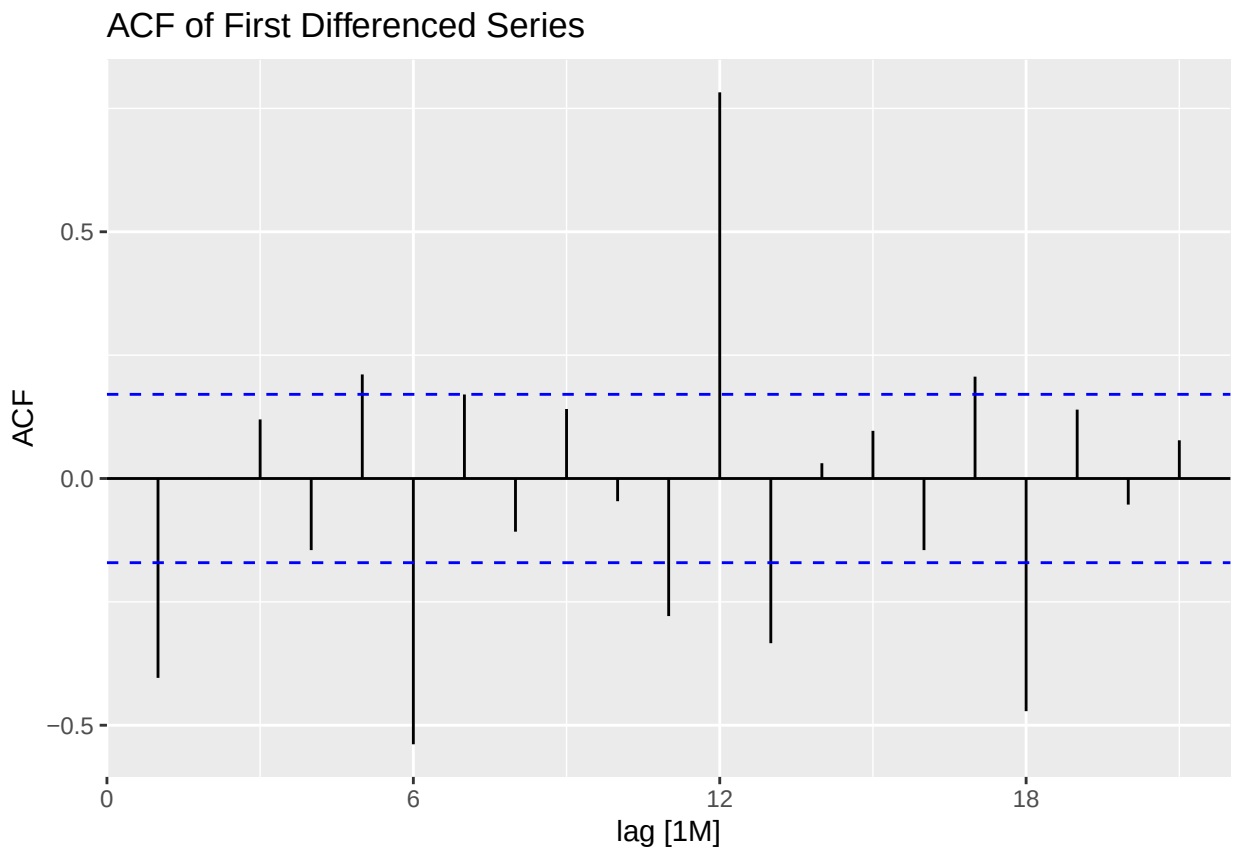
```
## [1] 0.350594
```

```
# Apply Box-Cox transformation to series
automobile_transformed <- automobile_ts %>%
  mutate(Transformed = box_cox(Automobiles, lambda))

# Apply first differencing to the transformed series to remove trend
automobile_diff <- automobile_transformed %>%
  mutate(Differenced = difference(Transformed))

# Plot first differenced transformed series
autoplot(automobile_diff, Differenced) +
  labs(title = "First Differenced Transformed Automobile Series", y = "Differenced Values")
```

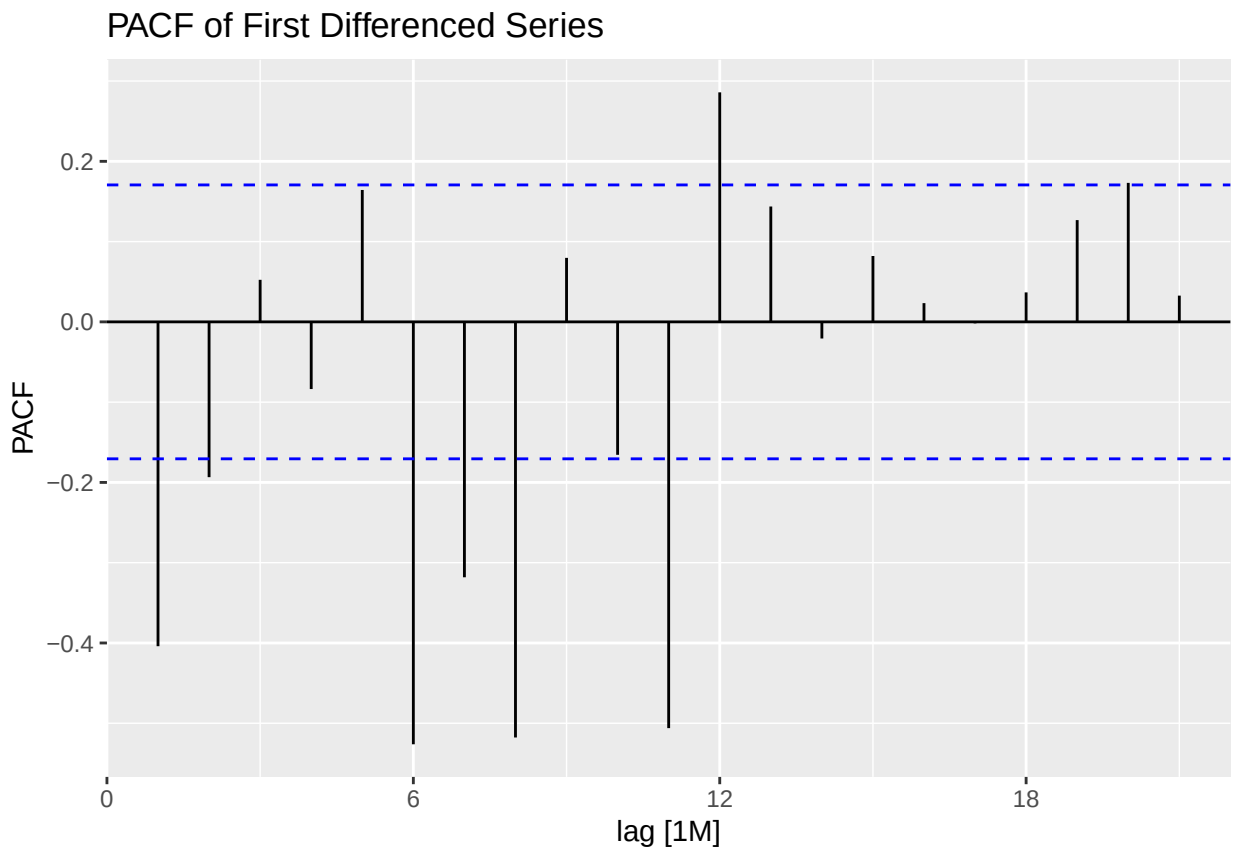
```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).
```




```
# Plot ACF and PACF of first differenced series
automobile_diff %>%
  ACF(Differenced) %>%
  autoplot() +
  labs(title = "ACF of First Differenced Series", y = "ACF")
```




```
automobile_diff %>%
  PACF(Differenced) %>%
  autoplot() +
  labs(title = "PACF of First Differenced Series", y = "PACF")
```

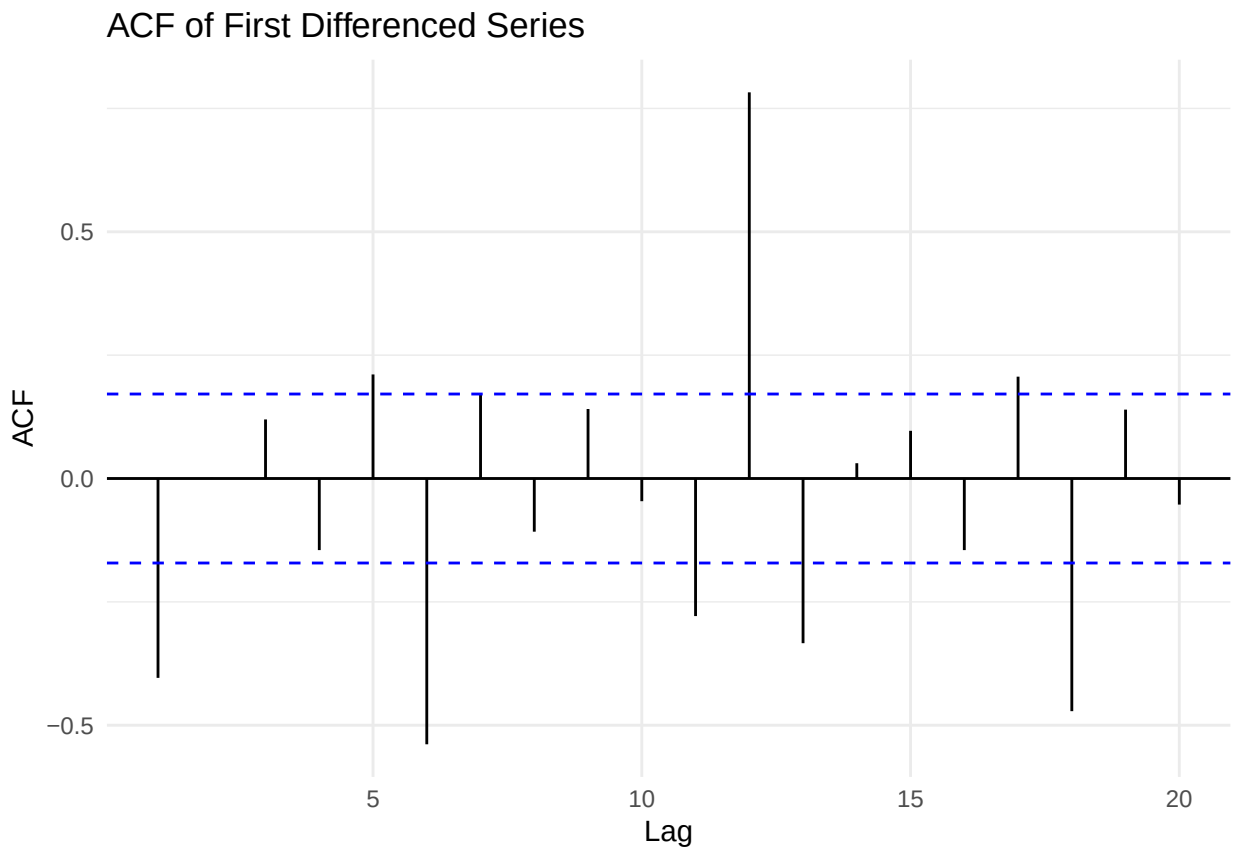


Question 3 (a)–

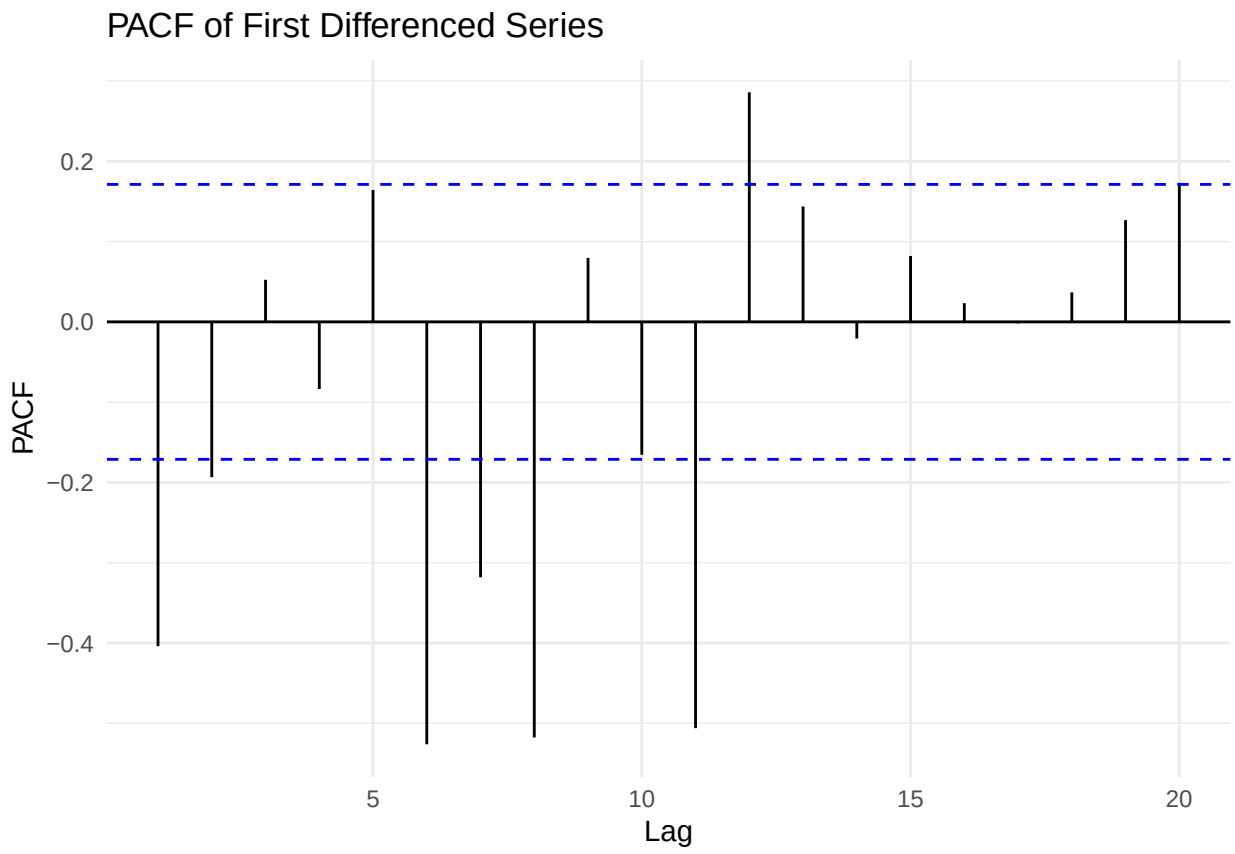
Answer:

By studying the appropriate graphs of the series in R, I proposed an ARIMA(1, 1, 1)(1, 0, 1)[12] model for the Automobiles data after analyzing the ACF and PACF plots. The ACF plot of the first differenced series showed a significant spike at lag 12, suggesting a potential seasonal pattern with yearly cycles, given that the data is monthly. This indicated the need for a seasonal ARIMA model with a seasonal component. The PACF plot had a significant spike at lag 1, pointing to an autoregressive (AR) component (p = 1), while the ACF suggested a moving average (MA) component (q = 1). Given that first differencing (d = 1) made the series stationary by removing the trend, the proposed model captures both the non-seasonal and seasonal aspects of the series. Therefore, I proposed ARIMA(1, 1, 1)(1, 0, 1)[12], where p = 1, q = 1, d = 1, and the seasonal component (P, D, Q) is (1, 0, 1) with a period of 12, to account for the identified yearly seasonality.

```
# Plot the ACF and PACF of the first differenced series to determine ARIMA parameters
ggAcf(automobile_diff$Differenced, lag.max = 20) +
  labs(title = "ACF of First Differenced Series", x = "Lag", y = "ACF") +
  theme_minimal()
```




```
ggPacf(automobile_diff$Differenced, lag.max = 20) +
  labs(title = "PACF of First Differenced Series", x = "Lag", y = "PACF") +
  theme_minimal()
```




```
# Example for ARIMA(1,1,1)(1,0,1)[12]
fit_arima <- automobile_diff %>%
  model(
    arima_model = ARIMA(Differenced ~ 0 + pdq(1, 1, 1) + PDQ(1, 0, 1))
  )
```

(b)–

Answer: I believe a constant should be included in the ARIMA model because it helps account for any drift or residual trend in the differenced data. Since we used first differencing (d = 1) to make the series stationary, there might still be a slight shift or non-zero mean in the differenced series. Including a constant term ensures that the model can adjust for this and provide more accurate forecasts. To verify this, I fitted the model both with and without the constant and compared their AIC values. The model with the constant had a lower AIC value (461.6843) compared to the model without it (465.4739), indicating that including the constant leads to a better fit. This lower AIC value indicates that the model with the constant is more accurate in capturing the underlying structure of the data, making it the better choice.

```
# Fit model with a constant
fit1 <- automobile_diff %>%
  model(m1 = ARIMA(Differenced ~ 1))

# Fit model without a constant
fit2 <- automobile_diff %>%
  model(m2 = ARIMA(Differenced ~ 0))
```

```
# Compare the AIC values
glance(fit1)
```

```
## # A tibble: 1 x 8
##   .model sigma2 log_lik   AIC  AICc   BIC ar_roots  ma_roots
##   <chr>   <dbl>   <dbl> <dbl> <dbl> <dbl> <list>    <list>
## 1 m1       2.41   -225.  462.  462.  478. <cpl [1]> <cpl [25]>
```

```
glance(fit2)
```

```
## # A tibble: 1 x 8
##   .model sigma2 log_lik   AIC  AICc   BIC ar_roots  ma_roots
##   <chr>   <dbl>   <dbl> <dbl> <dbl> <dbl> <list>    <list>
## 1 m2       2.59   -229.  465.  466.  477. <cpl [0]> <cpl [25]>
```

(c)–

Answer:For this question, I fitted the proposed ARIMA(1, 1, 1)(1, 0, 1)[12] model using the forecast library. I chose to use forecast rather than fpp3 because it allows for more direct control over specifying seasonal ARIMA parameters. While fpp3 works well with tidyverse, it had problems when I tried to manually set the seasonal components using PDQ. This library made it easier to specify the exact model structure required in Question 3(a) without the compatibility issues I faced before.

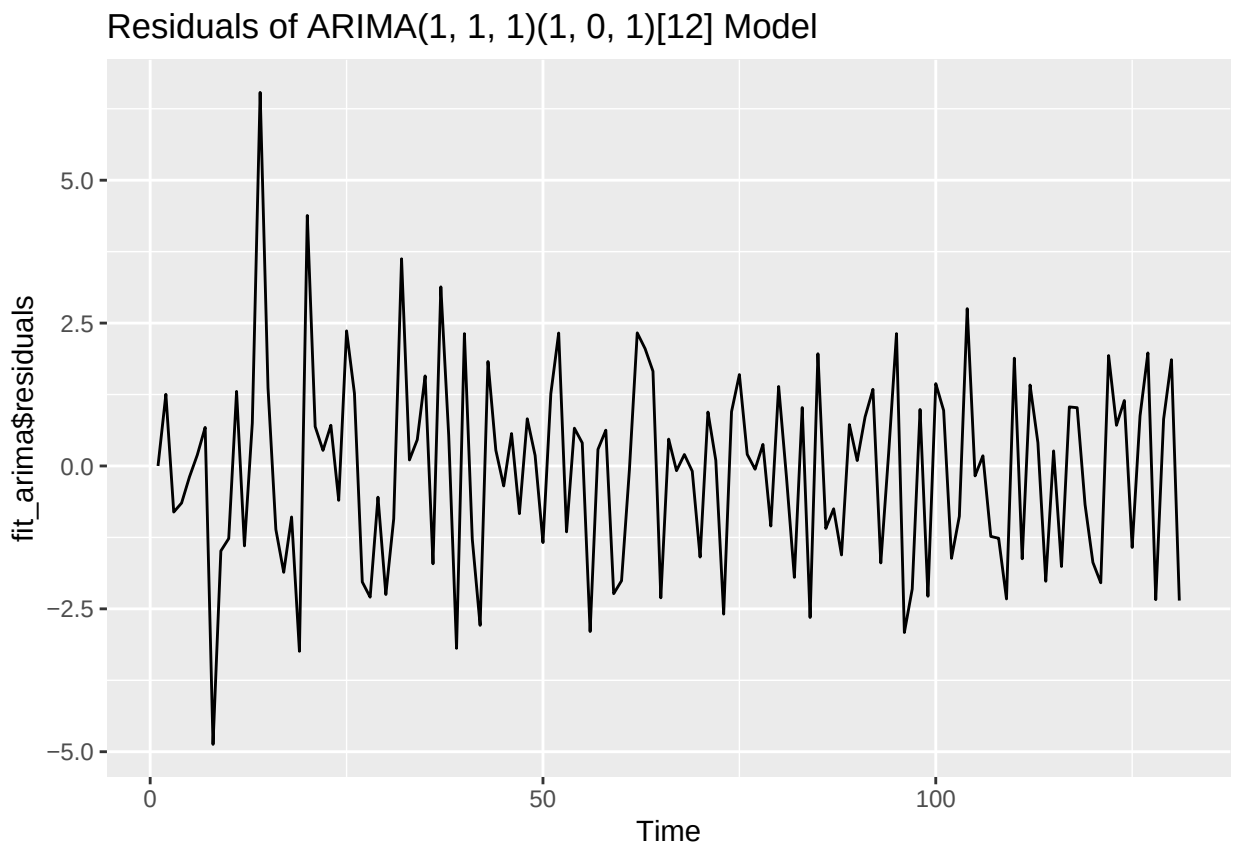
The model fitted, ARIMA(1, 1, 1)(1, 0, 1)[12], is somewhat satisfactory. The residual plot shows that the residuals are centered around zero with no obvious patterns, indicating that the model effectively captures the trend and seasonal structure of the data. However, the Ljung-Box test results in a p-value of 0.00366, which is below 0.05, suggesting that some significant autocorrelation remains in the residuals. Ideally, a p-value above 0.05 would indicate that the residuals behave like white noise, meaning no significant autocorrelation is present. Therefore, while the model provides a reasonable fit and can be used for forecasting, it may benefit from further refinement to improve accuracy. We might get better accuracy by adjusting the seasonal or non-seasonal components, or by adding more AR or MA terms to handle the remaining autocorrelation.

```
# Load forecast package
library(forecast)

# Handle NA values before fitting
cleaned_diff <- na.omit(automobile_diff$Differenced)

# Fit ARIMA(1, 1, 1)(1, 0, 1)[12] model using forecast package
fit_arima <- Arima(
  y = cleaned_diff,
  order = c(1, 1, 1),
  seasonal = list(order = c(1, 0, 1), period = 12)
)

# Summarize model to view the coefficients and diagnostics
summary(fit_arima)
```

```
## Series: cleaned_diff
## ARIMA(1,1,1)(1,0,1)[12]
##
## Coefficients:
```

```
##           ar1     ma1    sar1     sma1
##       -0.4433  -1.000  0.9941  -0.7435
## s.e.   0.0800   0.005  0.0063   0.1234
##
## sigma^2 = 3.074:  log likelihood = -271.65
## AIC=553.3   AICc=553.78   BIC=567.63
##
## Training set error measures:
##                       ME     RMSE      MAE      MPE    MAPE      MASE
## Training set -0.07231422 1.719392 1.373858 23.42692 88.4087 0.2149463
##                     ACF1
## Training set -0.1186358
```

```r
# Analyse residuals of the fitted model
autoplot(fit_arima$residuals) +
  labs(title = "Residuals of ARIMA(1, 1, 1)(1, 0, 1)[12] Model")
```




```r
# Perform Ljung-Box test to check for autocorrelation in residuals
Box.test(fit_arima$residuals, lag = 10, type = "Ljung")
```

```
##
##  Box-Ljung test
##
## data:  fit_arima$residuals
## X-squared = 26.06, df = 10, p-value = 0.00366
```

(d)–

Answer:The ARIMA() function chose ARIMA(1, 0, 1)(0, 1, 2)[12] with drift as the best model for the differenced data. This is different from my manual ARIMA(1, 1, 1)(1, 0, 1)[12] model in a few ways. The automatic model uses two seasonal MA terms and includes a drift term to handle the ongoing trend, while my manual model didn't have drift and used different orders. Comparing them shows the automatic model performed better, with lower AIC (461.68 vs. 553.3), AICc (462.43 vs. 553.78), and BIC (478.41 vs. 567.63), showing it better balances complexity and fit. It also had lower residual variance (sigma^2 = 2.408 vs 3.074), meaning it better captures data patterns. Although both models show some autocorrelation in the Ljung-Box test, the automatic model's lower AIC and residual variance, along with its drift term handling the trend, make it better for modeling the automobiles series.

```
# Automatically select best ARIMA model
auto_fit <- automobile_diff %>%
  model(auto_arima = ARIMA(Differenced))

# Summarize automatically selected ARIMA model
report(auto_fit)
```

```
## Series: Differenced
## Model: ARIMA(1,0,1)(0,1,2)[12] w/ drift
##
## Coefficients:
##          ar1      ma1     sma1     sma2  constant
##       0.2202  -0.9540  -0.4993  -0.2536   -0.0279
## s.e.  0.1093   0.0724   0.1274   0.0999    0.0050
##
## sigma^2 estimated as 2.408:  log likelihood=-224.84
## AIC=461.68   AICc=462.43   BIC=478.41
```

```
# Compare automatically selected model with manually chosen model
auto_fit %>% glance()
```

```
## # A tibble: 1 x 8
##   .model     sigma2 log_lik   AIC  AICc   BIC ar_roots  ma_roots
##   <chr>       <dbl>   <dbl> <dbl> <dbl> <dbl> <list>    <list>
## 1 auto_arima   2.41   -225.  462.  462.  478. <cpl [1]> <cpl [25]>
```

(e)–

Answer:Looking at the ETS models, I compared MSE and MAE values for different forecast periods (h=1, h=4, and h=6). The Holt-Winters' Multiplicative model performed best, with very low errors (MSE = 0.00, MAE = 0.02) for h=1, showing it works well for short-term forecasts. It maintained low errors for h=4 and h=6 too, showing good handling of seasonal patterns. When comparing these to the ARIMA(1, 0, 1)(0, 1, 2)[12] with drift from question 3(d), the ARIMA model proved better overall with its lower AIC of 461.68. While Holt-Winters' had better short-term forecast errors, ARIMA's lower AIC shows it better balances accuracy and complexity. The ARIMA model's drift term also helps capture trends that ETS might miss. Although ETS models work well with seasonal patterns, ARIMA's ability to handle differencing and seasonal components makes it more suitable for this data. Its lower residual variance shows it explains more of the data's variation, making it better for longer forecasts. So while ETS did well short-term, I prefer the ARIMA model as it better captures the overall data patterns and trends.

Question 4 (a)–

Answer: The accuracy() function shows NaN values for both Holt-Winters additive and multiplicative models because the seasonal patterns aren't clear enough in this data. This happens when we don't have enough seasonal information or when seasonal changes aren't strong or regular enough for the model to detect. When the seasonal component isn't well-defined, these models can't calculate error measures like MSE or MAE properly. This is clear in our results where Holt's Linear model gives us actual accuracy numbers, but both Holt-Winters models show NaN for their accuracy metrics because they can't properly identify the seasonal patterns..

```
# Extract NZ CPI data
mydata <- global_economy |>
  filter(Country == "New Zealand")

# Fit models
fit <- mydata |>
  model(
    HL = ETS(CPI ~ error("A") + trend("A") + season("N")),
    HW_add = ETS(CPI ~ error("A") + trend("A") + season("A")),
    HW_mul = ETS(CPI ~ error("M") + trend("A") + season("M"))
  )
```

```
## Warning: 1 error encountered for HW_add
## [1] A seasonal ETS model cannot be used for this data.
```

```
## Warning: 1 error encountered for HW_mul
## [1] A seasonal ETS model cannot be used for this data.
```

```
# Compare in-sample accuracy
accuracy(fit)
```

```
## # A tibble: 3 x 11
##   Country    .model .type       ME   RMSE     MAE     MPE   MAPE    MASE   RMSSE
##   <fct>      <chr>  <chr>    <dbl>  <dbl>   <dbl>   <dbl>  <dbl>   <dbl>   <dbl>
## 1 New Zeala~ HL     Trai~   0.0447   1.08   0.759   0.721   1.86   0.409   0.448
## 2 New Zeala~ HW_add Trai~ NaN      NaN    NaN     NaN     NaN    NaN     NaN
## 3 New Zeala~ HW_mul Trai~ NaN      NaN    NaN     NaN     NaN    NaN     NaN
## # i 1 more variable: ACF1 <dbl>
```

(b)–

Answer: No, I did not observe any warnings (or NaN) from the Holt-Winters models because the Victoria pigs data has a clear and consistent seasonal pattern, allowing the models to work properly. This is different from the New Zealand CPI data, where the seasonal component was not well-defined, leading to NaN outputs. With the Victoria pigs data, all three models—Holt's Linear, Holt-Winters additive, and Holt-Winters multiplicative—produced meaningful accuracy measures like RMSE and MAE. Among them, the Holt-Winters additive model showed the best performance, with the lowest RMSE of 7755.351 and MAE of 5746.576, indicating it captured the seasonality effectively. The multiplicative model had slightly higher errors but still worked fine because the seasonal variation in the data was suitable for this model. Getting actual values instead of NaN shows that Holt-Winters models work well when data has clear seasonal patterns, unlike what we saw with the CPI data.

```r
# Extract Victoria pigs data
myseries <- aus_livestock |>
  filter(Animal == "Pigs", State == "Victoria")

# Fit models
fit_vic <- myseries |>
  model(
    HL = ETS(Count ~ error("A") + trend("A") + season("N")),
    HW_add = ETS(Count ~ error("A") + trend("A") + season("A")),
    HW_mul = ETS(Count ~ error("M") + trend("A") + season("M"))
  )

# Compare in-sample accuracy
accuracy(fit_vic)
```

```
## # A tibble: 3 x 12
##   Animal State  .model .type    ME  RMSE   MAE    MPE  MAPE  MASE RMSSE     ACF1
##   <fct>  <fct>  <chr>  <chr> <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl>    <dbl>
## 1 Pigs   Victo~ HL     Trai~  417. 9386. 7244. -0.430  8.36 0.782 0.755  0.00804
## 2 Pigs   Victo~ HW_add Trai~  222. 7755. 5747. -0.274  6.74 0.620 0.624 -0.0386
## 3 Pigs   Victo~ HW_mul Trai~  355. 7794. 5849. -0.144  6.84 0.631 0.627  0.0151
```

Question 5 (a)–

Answer: TRUE

Rationale: Prediction intervals from ARIMA models generally increase as the forecast horizon increases due to accumulating uncertainty. For stationary models (d=0), the intervals will eventually converge, so long-term prediction intervals become roughly the same. For non-stationary models (d 1), the prediction intervals keep growing as the forecast horizon extends further into the future.

(b)–

Answer: TRUE

Rationale:The AICc cannot be used to compare between ARIMA and ETS models. As shown in Section 9.10 of the textbook, when comparing ARIMA and ETS models, time series cross-validation and accuracy measures (like RMSE, MAE, MPE, MAPE) should be used instead of AICc. This is demonstrated in the example of Australian population data where both models were compared using accuracy metrics, and ETS performed better with lower error measures. The absence of AICc in this comparison supports that it's not appropriate for comparing these different model types.

(c)–

Answer: TRUE

Rationale: Time series cross-validation can be used to compare between ARIMA and ETS models. This method involves dividing the time series data into training and test sets across different time points, fitting the models on the training sets, and evaluating their accuracy on the test sets. Since it compares the forecasting performance directly by assessing how well each model predicts unseen data, it does not rely on model assumptions like the AICc. Therefore, time series cross-validation is a suitable approach for comparing ARIMA and ETS models, as it allows for a fair evaluation of their predictive capabilities.

(d)–

Answer: FALSE

Rationale:After comparing ETS and ARIMA models on seasonal cement production data, the ARIMA model was chosen as the better model, not the ETS model. This is proven by the test set results where ARIMA

showed better accuracy measures with lower RMSE (216 vs 222), MAE (186 vs 191), and MAPE (8.68% vs 8.85%) compared to ETS. The text clearly states that “the ARIMA model seems to be the slightly more accurate model based on the test set RMSE, MAPE and MASE.