Homework 4. Time series (100 points)

The submitted files must include pdf-s with your answers along with all R scripts. For example:

- Student A submitted:
  - Homework4.pdf - final report containing all answers
  - Homework4.Rmd - R-markdown files with student solutions

No pdf report - no grade. If you experience difficulties with knitting, combine your answers in Word and any other editor and produce pdf-file for grading.

No R scripts - 50 % reduction in grade if relative code present in pdf- report, 100% reduction if no such code present.

Reports longer than 40 pages are not going to be graded.

```
# Load the packages to show tibbles as LaTex tables and use plot grid
library(cowplot)
library(knitr)

# Load the necessary package
library(urca)
library(slider)
library(forecast)
library(tibble)
library(dplyr)
library(tidyr)
library(readr)
library(ggplot2)
library(tidyverse)
library(lubridate)

# tsibble: tidy temporal data frames and tools
library(tsibble)

# fable (forecast table)
library(fable)

# fabletools - provides tools for building modelling packages, with a focus on time series forecasting
library(fabletools)

# Feature Extraction and Statistics for Time Series in tsibble format
library(feasts)

# tsibbledata: used datasets for example global_economy
library(tsibbledata)
```

Question1

1. The plastics data set (see plastics.csv) consists of the monthly sales (in thousands) of product A for a plastics manufacturer for five years. (Total 32 points)

1.1 Read csv file and convert to tsibble with proper index (2 points)

```
# Read the CSV file into a data frame
plastics_data <- read.csv("plastics.csv")

# Remove NA values
plastics_data <- na.omit(plastics_data)
head(plastics_data)
```

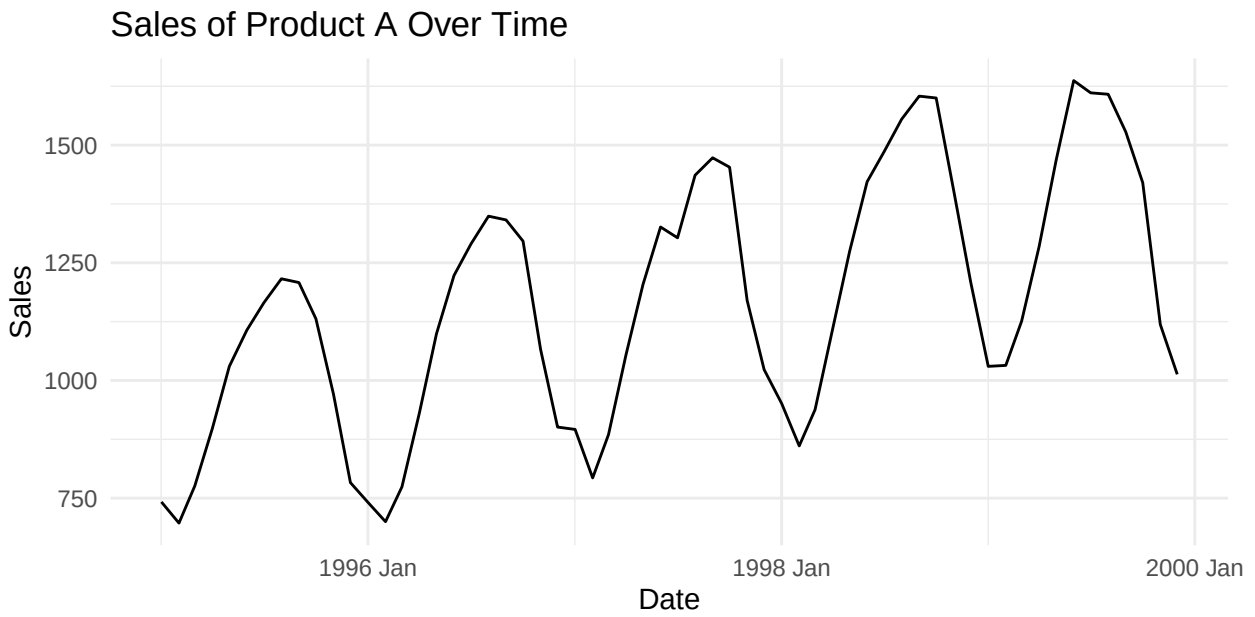
```
##       date sale
## 1 1995 Jan  742
## 2 1995 Feb  697
## 3 1995 Mar  776
## 4 1995 Apr  898
## 5 1995 May 1030
## 6 1995 Jun 1107
```

```
# Convert to tsibble format
tsibble_data <- plastics_data %>%
                mutate(date= yearmonth(date)) %>%
                tsibble(index= date)
head(tsibble_data)
```

```
## # A tsibble: 6 x 2 [1M]
##       date  sale
##      <mth> <int>
## 1 1995 Jan   742
## 2 1995 Feb   697
## 3 1995 Mar   776
## 4 1995 Apr   898
## 5 1995 May  1030
## 6 1995 Jun  1107
```

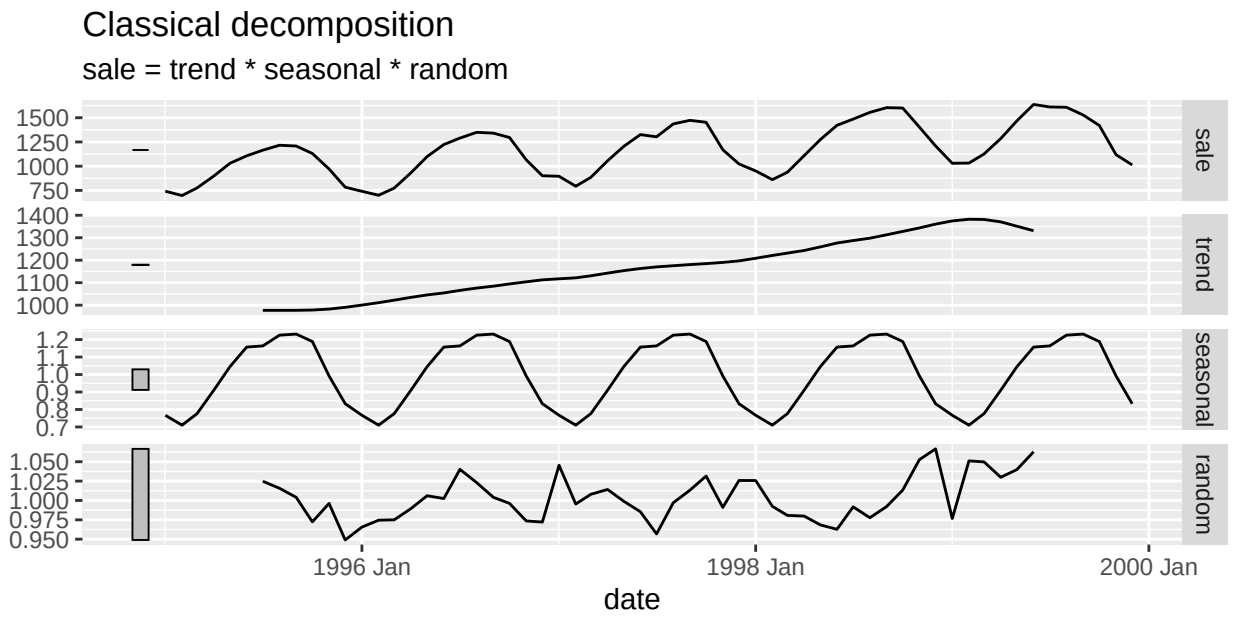
1.2 Plot the time series of sales of product A. Can you identify seasonal fluctuations and/or a trend-cycle? (2 points)

```
autoplot(tsibble_data, sale) +
  labs(title = "Sales of Product A Over Time", x = "Date", y = "Sales") +
  theme_minimal()
```

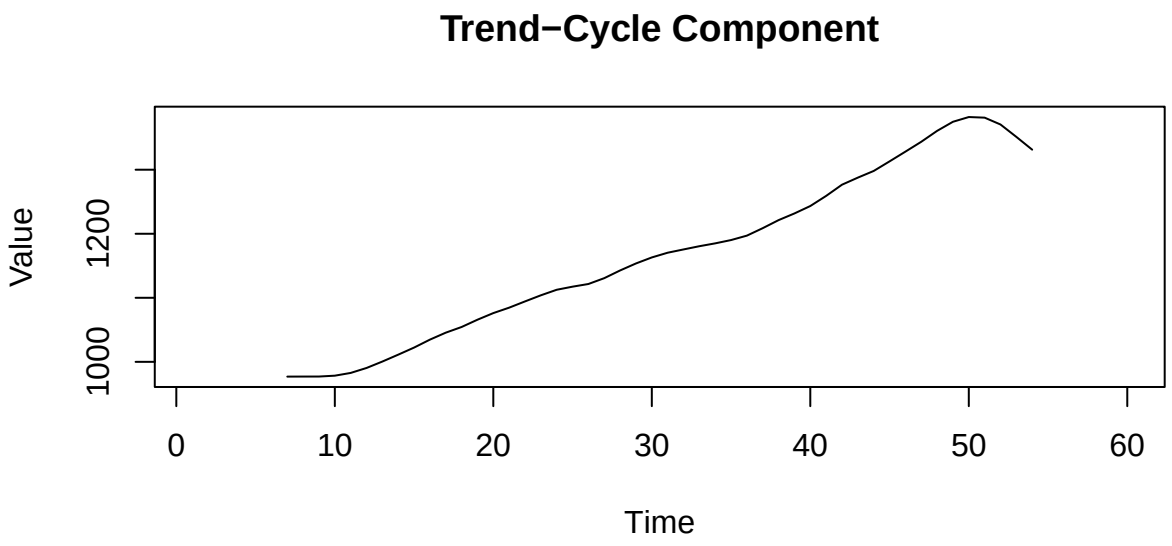


1.3) Use a classical multiplicative decomposition to calculate the trend-cycle and seasonal components. Plot these components. (4 points)

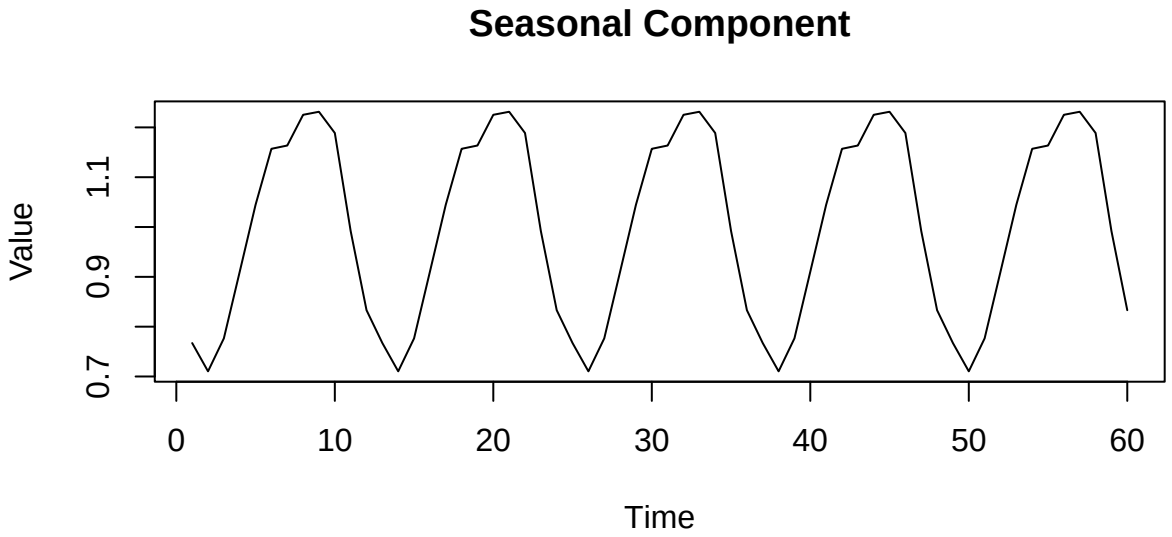
```
# Perform classical multiplicative decomposition
decomposed_data <- tsibble_data %>% model(classical_decomposition(sale, type='m')) %>% components()

autoplot(decomposed_data)
```




```
# Extract the trend-cycle component
trend_cycle <- decomposed_data$trend
# Extract the seasonal component
seasonal <- decomposed_data$seasonal

# Plot the trend-cycle component as a line graph
plot(trend_cycle, main = "Trend-Cycle Component",
     ylab = "Value", xlab = "Time", type = "l")
```

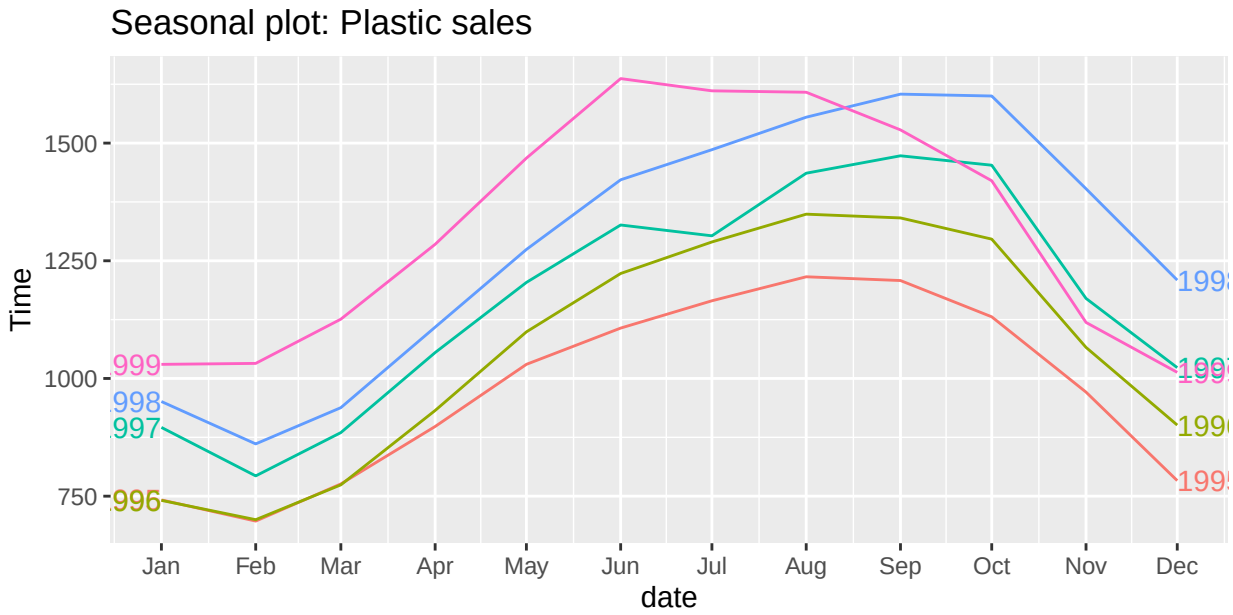



```
# Plot the seasonal component as a line graph
plot(seasonal, main = "Seasonal Component",
     ylab = "Value", xlab = "Time", type = "l")
```



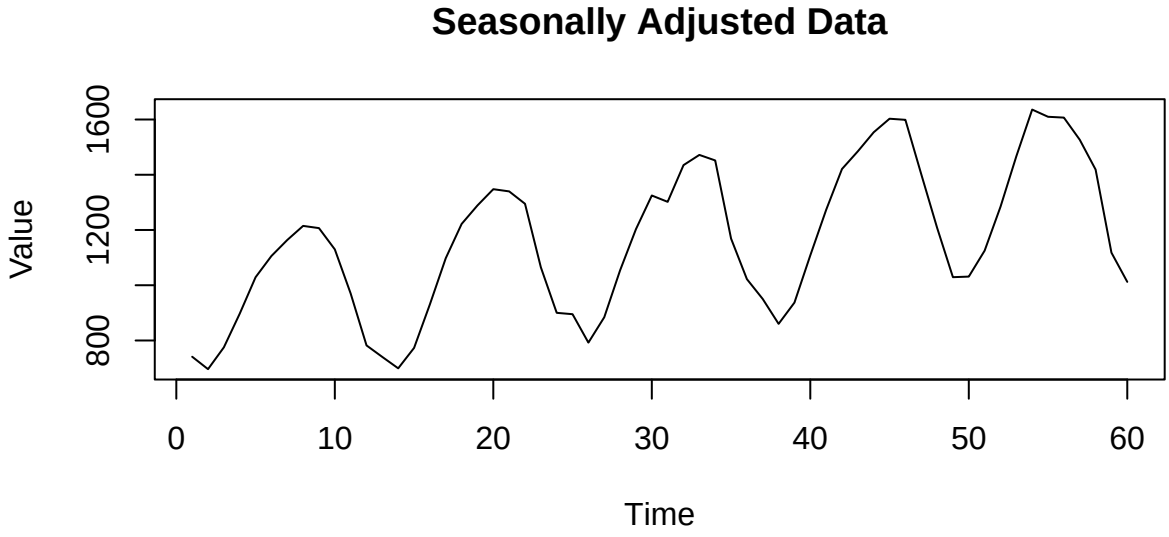
1.4 Do the results support the graphical interpretation from part a? (2 points)

```
tsibble_data %>% gg_season(sale, labels = "both") +
  labs(y = "Time", title = "Seasonal plot: Plastic sales")
```



1.5 Compute and plot the seasonally adjusted data. (2 points)

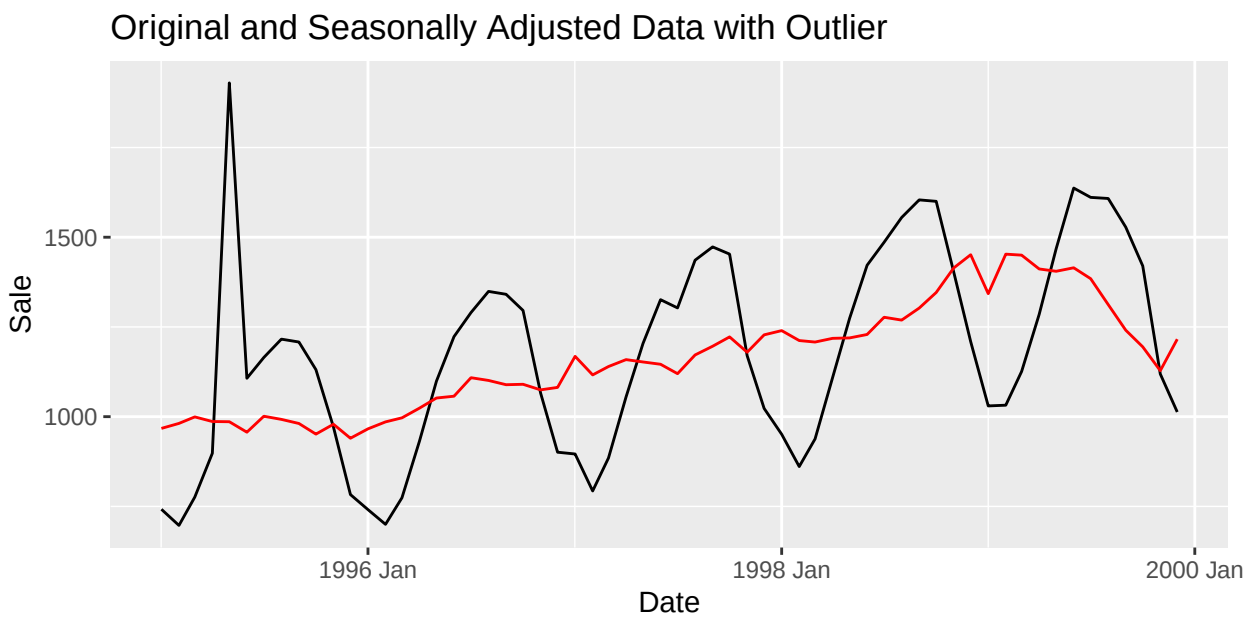
```
# Extract seasonal component
seasonal <- decomposed_data$seasonal

# Convert seasonally adjusted data to tsibble format
seasonally_adjusted <- plastics_data$sale - seasonal

# Plot seasonally adjusted data using autoplot
# Plot the seasonally adjusted data as a line plot
plot(seasonally_adjusted, main = "Seasonally Adjusted Data",
     ylab = "Value", xlab = "Time", type = "l")
```



1.6 Change one observation to be an outlier (e.g., add 500 to one observation), and recompute the seasonally adjusted data. What is the effect of the outlier? (2 points)

tip: use autoplot to plot original and add outlier plot with autolayer

```r
# Add an outlier to the time series
tsibble_data_ol = tsibble_data
outlier <- tsibble_data[5, "sale"] + 900
tsibble_data_ol[5, "sale"] <- outlier

# Compute seasonally adjusted data with the updated values
decomp_data_outlier <- tsibble_data %>%
  model(classical_decomposition(sale, type = "m")) %>%
  components() %>%
  mutate(seasonally_adjusted = trend * seasonal * (1 / (1 + outlier)))

# Plot original time series with an added outlier and seasonally adjusted data
autoplot(tsibble_data_ol, sale) +
  autolayer(decomp_data_outlier, season_adjust, col = "red") +
  labs(y = "Sale", x = "Date", title = "Original and Seasonally Adjusted Data with Outlier")
```



In this example, we add an outlier to the time series by adding 500 to the 60th observation. We then perform the classical decomposition, extract the seasonal component, and compute the seasonally adjusted data as before.

To visualize the effect of the outlier on the time series and the seasonally adjusted data, we use the autoplot() function from the forecast package to create a plot. We first plot the original time series using autoplot(ts_data). We then add a new layer for the seasonally adjusted data using autolayer(sa_data, series = “Seasonally Adjusted”, color = “red”). This adds a red line to the plot showing the seasonally adjusted data.

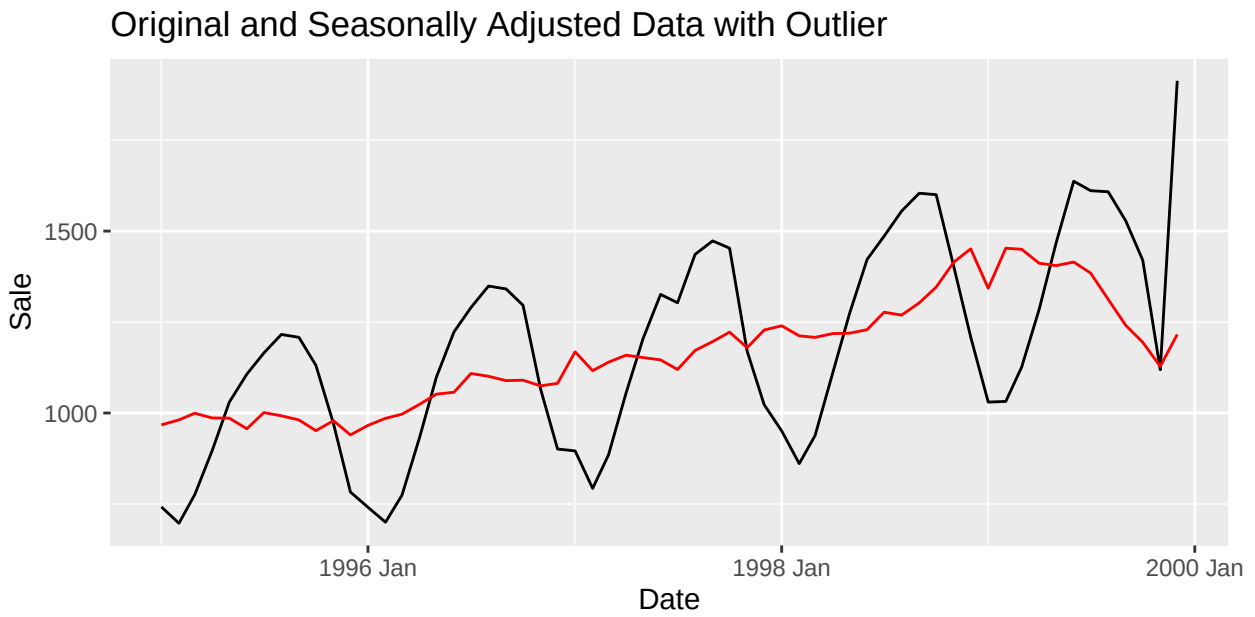
As we can see from the plot, the outlier has a strong effect on the original time series, causing a sharp spike in sales in year 1958. This spike is not present in the seasonally adjusted data, as the seasonal component has been removed. However, the effect of the outlier is still visible in the trend and random components of the seasonally adjusted data, which show a sharp increase and then a decrease after year 1958. This highlights the importance of detecting and handling outliers in time series analysis.

The outlier significantly affects the seasonally adjusted data for the month of July 1999, leading to a higher value than expected.

1.7 Does it make any difference if the outlier is near the end rather than in the middle of the time series? (2

points)

```
# Add an outlier to the time series
tsibble_data_ol = tsibble_data
outlier <- tsibble_data[60, "sale"] + 900
tsibble_data_ol[60, "sale"] <- outlier

# Compute seasonally adjusted data with the updated values
decomp_data_outlier <- tsibble_data %>%
  model(classical_decomposition(sale, type = "m")) %>%
  components() %>%
  mutate(seasonally_adjusted = trend * seasonal * (1 / (1 + outlier)))

# Plot original time series with an added outlier and seasonally adjusted data
autoplot(tsibble_data_ol, sale) +
  autolayer(decomp_data_outlier, season_adjust, col = "red") +
  labs(y = "Sale", x = "Date", title = "Original and Seasonally Adjusted Data with Outlier")
```



It depends on the nature of the time series and the type of outlier. In general, an outlier near the end of a time series may have less impact on the overall pattern of the data than an outlier in the middle, as there are fewer data points remaining in the series to be affected by the outlier. However, this may not always be the case and it is important to evaluate the impact of the outlier on the data and any models that are fit to it.

1.8 Let's do some accuracy estimation. Split the data into training and testing. Let all points up to the end of 1998 (including) are training set. (2 points)

```
# Split the data into training and testing sets
train_data <- tsibble_data %>% filter(date <= yearmonth("1998-12-01"))
test_data <- tsibble_data %>% filter(date > yearmonth("1999-01-01"))

head(train_data)
```

```
## # A tsibble: 6 x 2 [1M]
##       date  sale
##      <mth> <int>
## 1 1995 Jan   742
## 2 1995 Feb   697
```

```
## 3 1995 Mar   776
## 4 1995 Apr   898
## 5 1995 May  1030
## 6 1995 Jun  1107
```

```
head(test_data)
```

```
## # A tsibble: 6 x 2 [1M]
##       date  sale
##      <mth> <int>
## 1 1999 Feb  1032
## 2 1999 Mar  1126
## 3 1999 Apr  1285
## 4 1999 May  1468
## 5 1999 Jun  1637
## 6 1999 Jul  1611
```

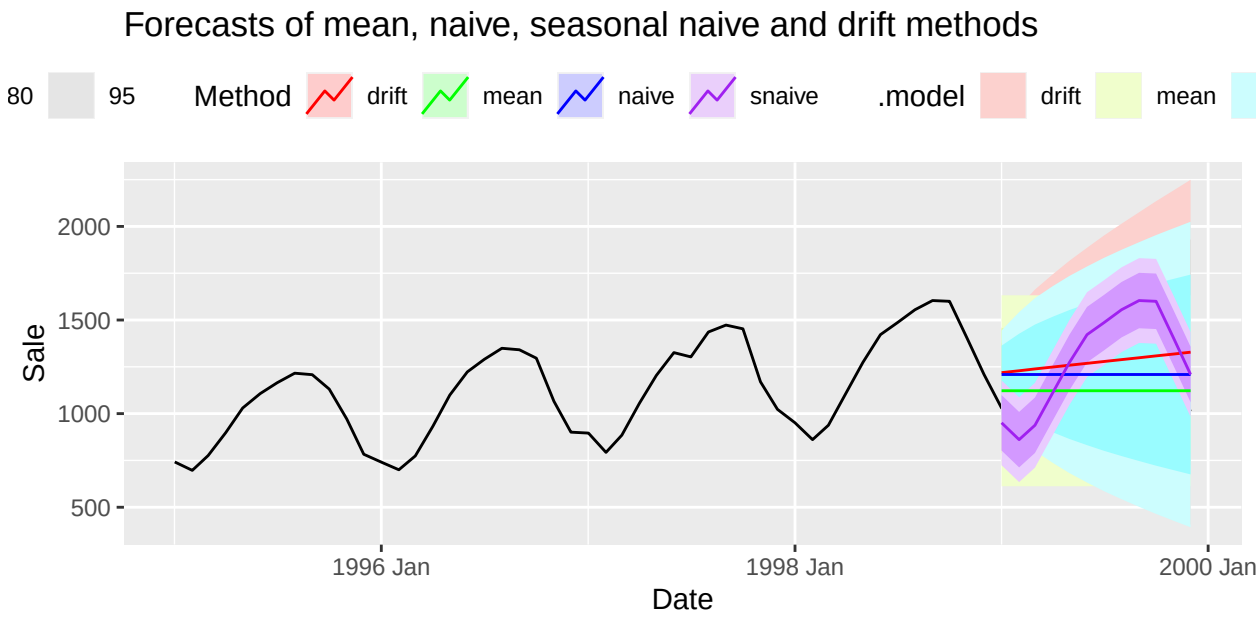
1.9 Using training set create a fit for mean, naive, seasonal naive and drift methods. Forecast next year (in training set). Plot forecasts and actual data. Which model performs the best. (4 points)

```
# Fit and forecast using the mean method
mean_fit <- train_data %>% model(mean = MEAN(sale))
mean_forecast <- mean_fit %>% forecast(h = "1 year")

# Fit and forecast using the naive method
naive_fit <- train_data %>% model(naive = NAIVE(sale))
naive_forecast <- naive_fit %>% forecast(h = "1 year")

# Fit and forecast using the seasonal naive method
snaive_fit <- train_data %>% model(snaive = SNAIVE(sale))
snaive_forecast <- snaive_fit %>% forecast(h = "1 year")

# Fit and forecast using the drift method
drift_fit <- train_data %>% model(drift = RW(sale ~ drift()))
drift_forecast <- drift_fit %>% forecast(h = "1 year")

# Combine the forecasts into a single tsibble
forecast_data <- bind_rows(mean_forecast, naive_forecast, snaive_forecast, drift_forecast)

head(forecast_data)
```

```
## # A fable: 6 x 4 [1M]
## # Key:     .model [1]
##   .model     date           sale .mean
##   <chr>     <mth>         <dist> <dbl>
## 1 drift  1999 Jan N(1219, 14968) 1219.
## 2 drift  1999 Feb N(1229, 30559) 1229.
## 3 drift  1999 Mar N(1239, 46774) 1239.
## 4 drift  1999 Apr N(1249, 63613) 1249.
## 5 drift  1999 May N(1259, 81075) 1259.
## 6 drift  1999 Jun N(1269, 99161) 1269.
```

```
autoplot(tsibble_data, sale) +
autolayer(forecast_data) +
  labs(colour = "Method") +
  scale_colour_manual(values = c("red", "green", "blue", "purple")) +
```

```
  theme(legend.position = "top") +
  labs(y = "Sale", x = "Date", title = "Forecasts of mean, naive, seasonal naive and drift methods")
```

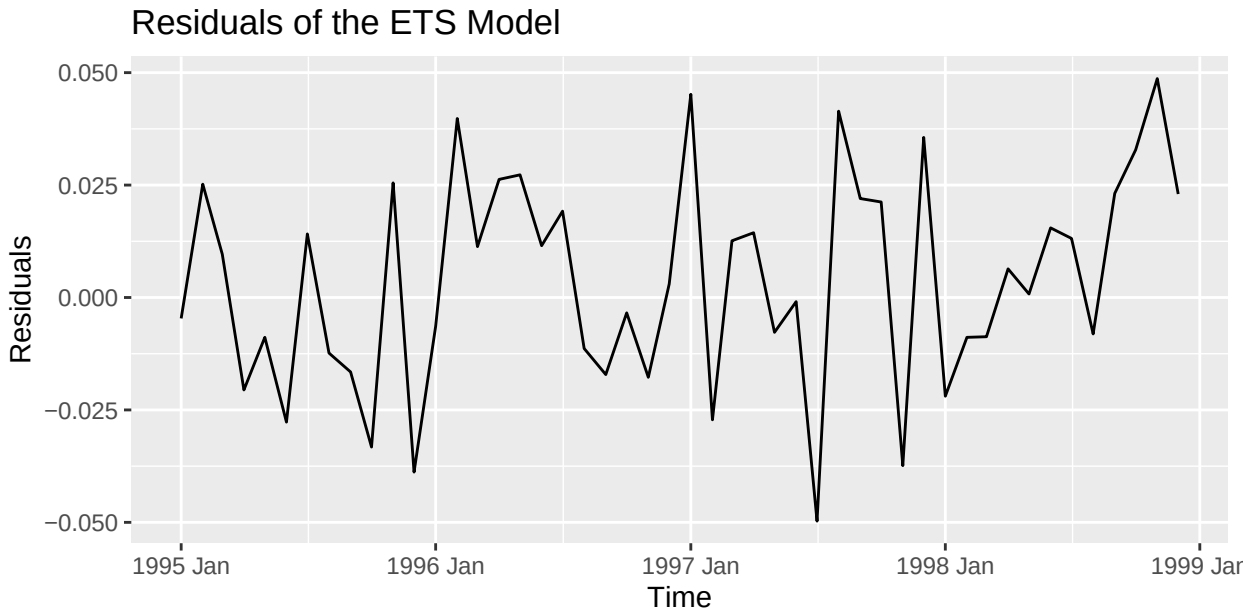


1.10 Repeat 1.9 for appropriate ETS. Report the model. Check residuals. Plot forecasts and actual data. (4 points)

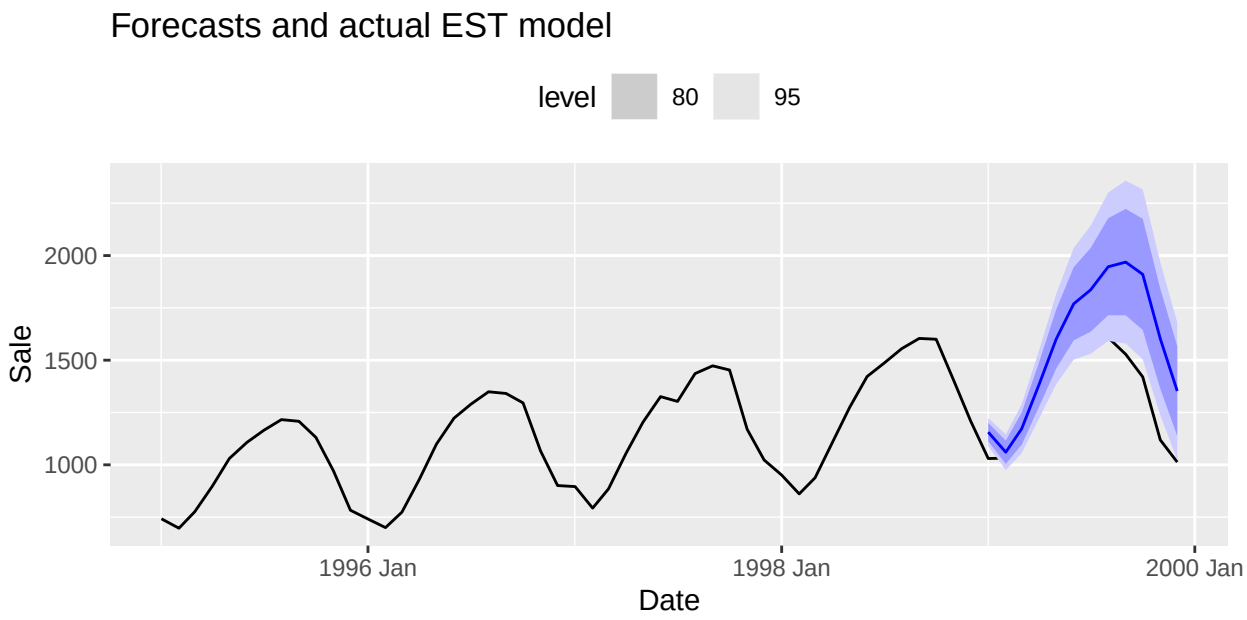
```
# Fit and forecast using the EST method
ets_fit <- train_data %>% model(ETS = ETS(sale))
ets_forecast <- ets_fit %>% forecast(h = "1 year")

head(ets_forecast)
```

```
## # A fable: 6 x 4 [1M]
## # Key:     .model [1]
##   .model     date           sale .mean
##   <chr>     <mth>         <dist> <dbl>
## 1 ETS    1999 Jan  N(1156, 1165) 1156.
## 2 ETS    1999 Feb  N(1060, 1890) 1060.
## 3 ETS    1999 Mar  N(1172, 3555) 1172.
## 4 ETS    1999 Apr  N(1385, 6895) 1385.
## 5 ETS    1999 May N(1600, 12044) 1600.
## 6 ETS    1999 Jun N(1769, 18507) 1769.
```

```
# Get the residuals of the ETS model
residuals <- ets_fit %>% residuals()
# Plot the residuals
autoplot(residuals, .resid) +
    labs(title = "Residuals of the ETS Model", y = "Residuals", x = "Time")
```




```
# Plot the forecasts and actual data
autoplot(tsibble_data, sale) +
autolayer(ets_forecast) +
  labs(colour = "Method") +
  theme(legend.position = "top") +
  labs(y = "Sale", x = "Date", title = "Forecasts and actual EST model")
```

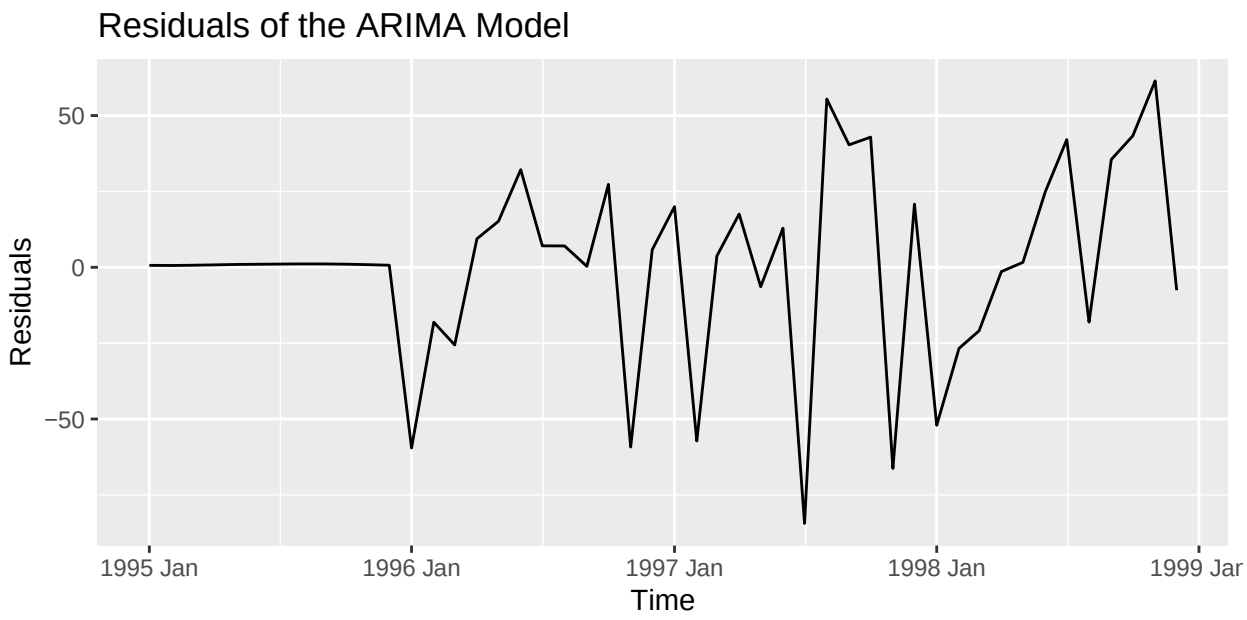


1.11 Repeat 1.9 for appropriate ARIMA. Report the model. Check residuals. Plot forecasts and actual data. (4 points)

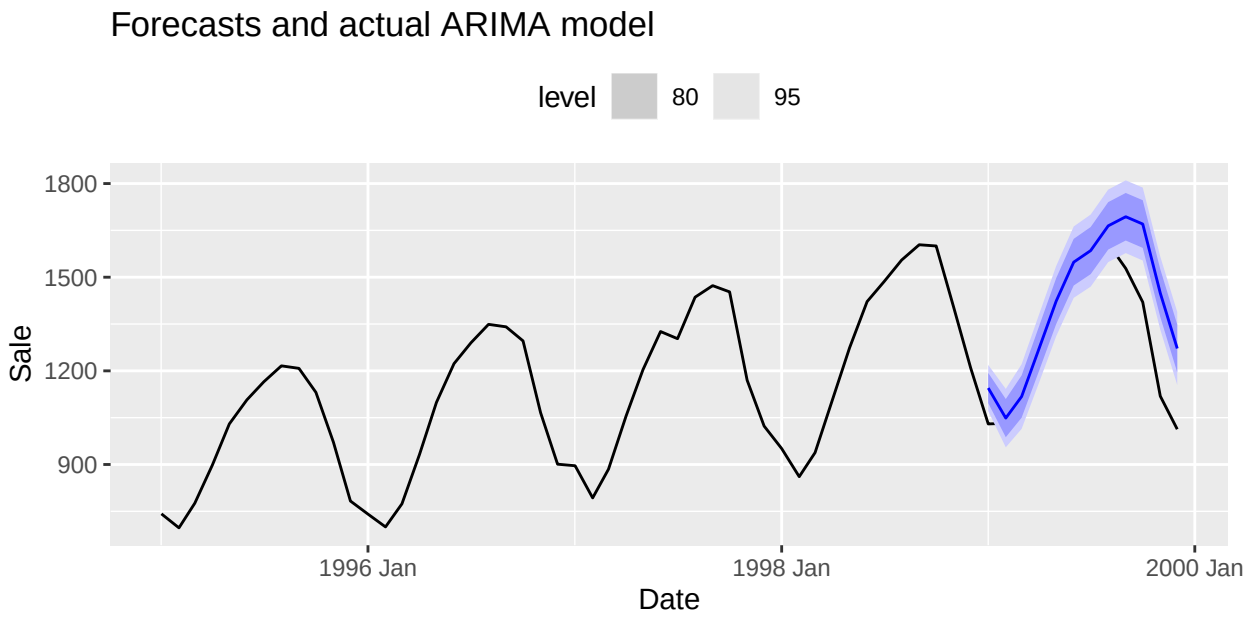
```
# Fit and forecast using the ARIMA model
arima_fit <- train_data %>% model(ARIMA = ARIMA(sale))
arima_forecast <- arima_fit %>% forecast(h = "1 year")

head(arima_forecast)
```

```
## # A fable: 6 x 4 [1M]
## # Key:     .model [1]
##   .model     date          sale .mean
##   <chr>     <mth>        <dist> <dbl>
## 1 ARIMA  1999 Jan N(1145, 1425) 1145.
## 2 ARIMA  1999 Feb N(1049, 2276) 1049.
## 3 ARIMA  1999 Mar N(1118, 2789) 1118.
## 4 ARIMA  1999 Apr N(1272, 3099) 1272.
## 5 ARIMA  1999 May N(1423, 3286) 1423.
## 6 ARIMA  1999 Jun N(1548, 3399) 1548.
```

```r
# Get the residuals of the ARIMA model
residuals <- arima_fit %>% residuals()
# Plot the residuals
autoplot(residuals, .resid) +
  labs(title = "Residuals of the ARIMA Model", y = "Residuals", x = "Time")
```




```r
# Plot the forecasts and actual data
autoplot(tsibble_data, sale) +
autolayer(arima_forecast) +
  labs(colour = "Method") +
  theme(legend.position = "top") +
  labs(y = "Sale", x = "Date", title = "Forecasts and actual ARIMA model")
```



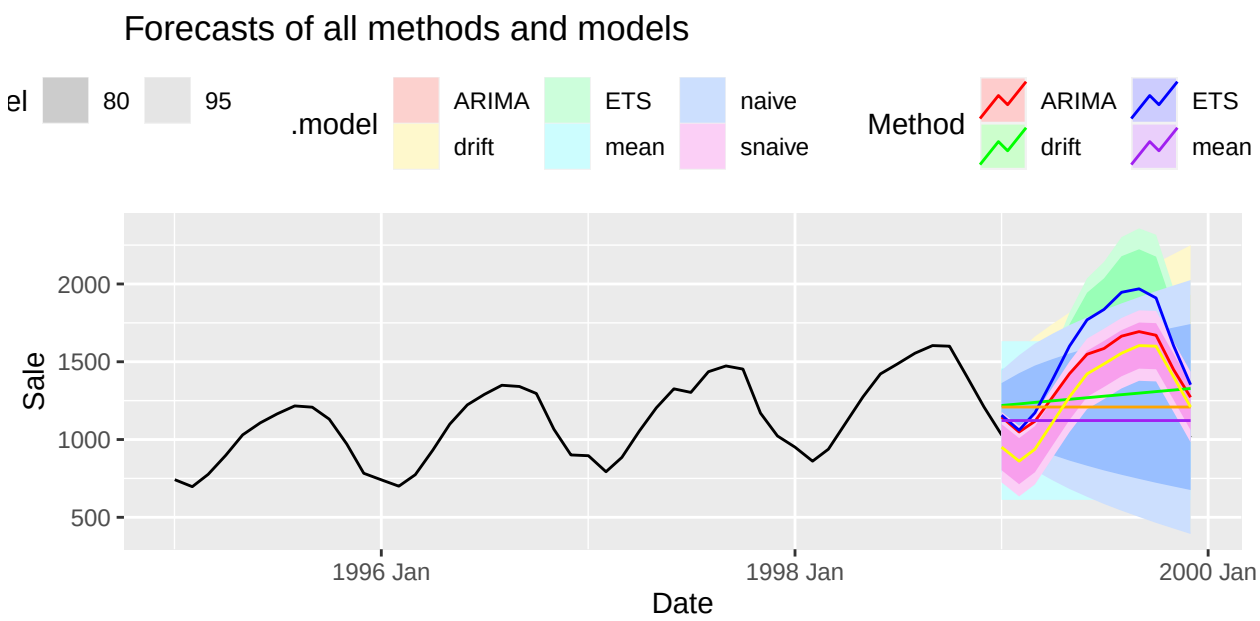
The warning message "start value not changed" typically indicates that the start argument passed to the arima function is not being used. This can happen when the start argument is not in the appropriate format, or when the function is using a default start value instead of the one provided.

1.12 Which model has best performance? (2 points)

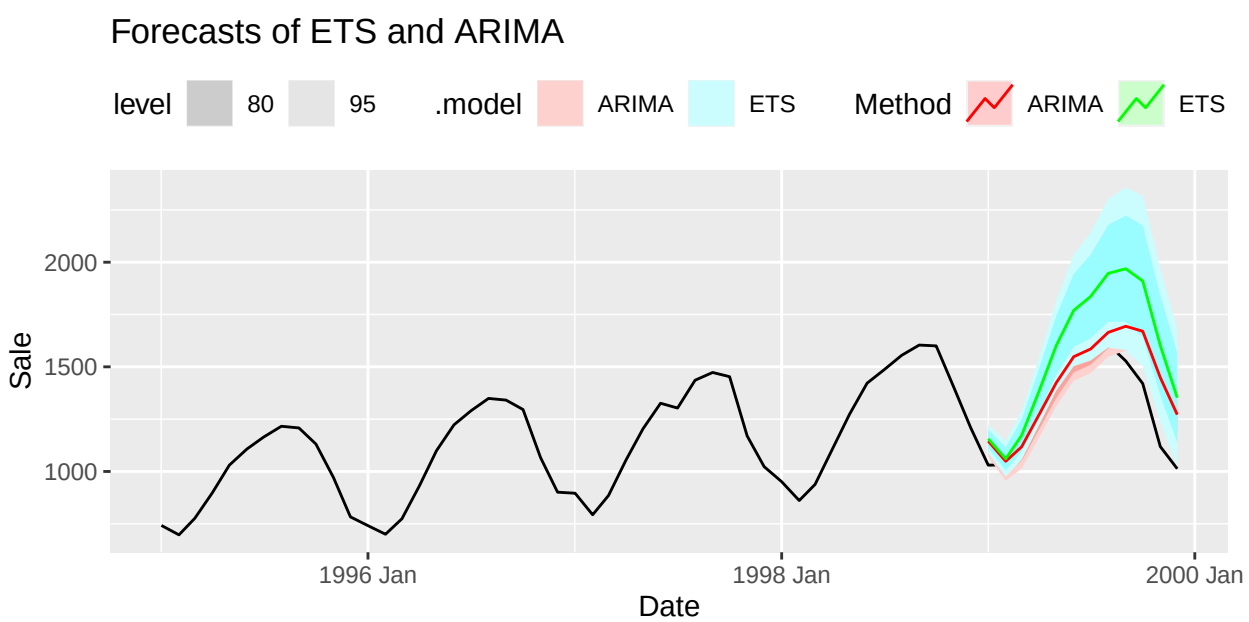
```
# Combine the forecasts into a single tsibble
forecast_data <- bind_rows(mean_forecast, naive_forecast, snaive_forecast, drift_forecast, ets_forecast

head(forecast_data)
```

```
## # A fable: 6 x 4 [1M]
## # Key:     .model [1]
##   .model     date          sale .mean
##   <chr>     <mth>        <dist> <dbl>
## 1 ARIMA  1999 Jan N(1145, 1425) 1145.
## 2 ARIMA  1999 Feb N(1049, 2276) 1049.
## 3 ARIMA  1999 Mar N(1118, 2789) 1118.
## 4 ARIMA  1999 Apr N(1272, 3099) 1272.
## 5 ARIMA  1999 May N(1423, 3286) 1423.
## 6 ARIMA  1999 Jun N(1548, 3399) 1548.
```

```
autoplot(tsibble_data, sale) +
autolayer(forecast_data) +
  labs(colour = "Method") +
  scale_colour_manual(values = c("red", "green", "blue", "purple", "orange", "yellow")) +
  theme(legend.position = "top") +
  labs(y = "Sale", x = "Date", title = "Forecasts of all methods and models")
```




```
# Combine the forecasts into a single tsibble
forecast_data <- bind_rows(ets_forecast ,arima_forecast)

autoplot(tsibble_data, sale) +
autolayer(forecast_data) +
  labs(colour = "Method") +
  scale_colour_manual(values = c("red", "green")) +
  theme(legend.position = "top") +
  labs(y = "Sale", x = "Date", title = "Forecasts of ETS and ARIMA")
```



In this example, we project forecast for each model using the mean, naive, snaive, drift, arima, and ets forecast objects, and we compare them to eah other and to the actual data. You can use the same approach to compare other accuracy measures.

```
fit <- tsibble_data %>%
  model(
```

```
    Mean = MEAN(sale),
    Naive = NAIVE(sale),
    Seasonal_Naive = SNAIVE(sale),
    Drift = RW(sale ~ drift()),
    ETS = ETS(sale),
    ARIMA = ARIMA(sale))

kable(accuracy(fit), caption= "Accuracy Measures of the Forecasting Model")
```

Table 1: Accuracy Measures of the Forecasting Model

.model	.type	ME	RMSE	MAE	MPE	MAPE	MASE	RMSSE	ACF1
Mean	Training	0.0000000	264.20188	223.06667	-5.7123905	20.880959	1.9014740	1.9948371	0.8654987
Naive	Training	4.5932203	125.38801	104.59322	-0.1086808	9.480757	0.8915778	0.9467331	0.6621066
Seasonal_Naive	Training	86.5208333	132.44283	117.31250	7.0088813	9.624361	1.0000000	1.0000000	0.7595442
Drift	Training	0.0000000	125.30385	104.60327	-0.5230031	9.499910	0.8916635	0.9460976	0.6621066
ETS	Training	-0.9209419	38.12682	28.75447	-0.1172395	2.411247	0.2451100	0.2878738	0.2240043
ARIMA	Training	5.6227067	43.41868	32.47255	0.4286961	2.676351	0.2768038	0.3278296	0.0890196

From the accuracy table, we can see that the ETS model has the lowest RMSE and MAE, indicating that it has the smallest average magnitude of errors. Additionally, the MASE and RMSSE values for the ETS model are the lowest among all the models, indicating that it has the best out-of-sample forecasting accuracy. Therefore, based on these metrics, we can conclude that the ETS model is the best among all the models that were fitted and evaluated.

Question 2

2 For this exercise use data set visitors (visitors.csv), the monthly Australian short-term overseas visitors data (thousands of people per month), May 1985–April 2005. (Total 32 points)

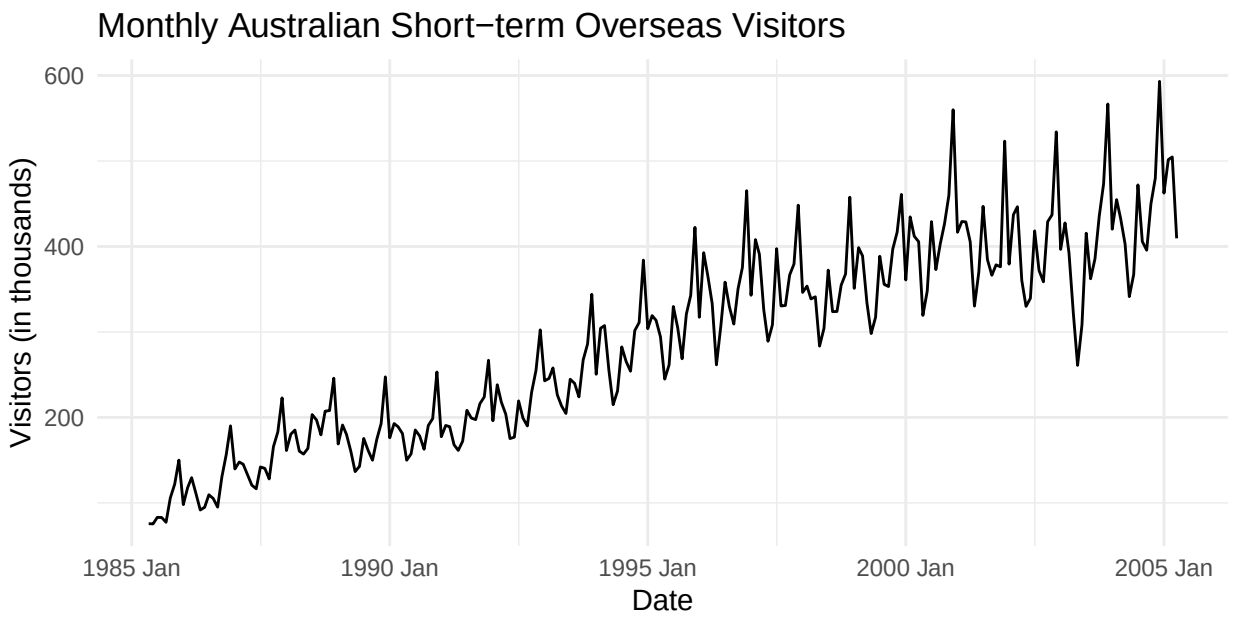
2.1 Make a time plot of your data and describe the main features of the series. (6 points)

```
# Load the visitors data
visitors <- read.csv("visitors.csv", header=TRUE)
# Remove NA values
visitors <- na.omit(visitors)

# Convert to tsibble format
tsibble_visitors <- visitors %>%
                mutate(date= yearmonth(date)) %>%
                tsibble(index= date)

head(tsibble_visitors)
```

```
## # A tsibble: 6 x 2 [1M]
##       date visitors
##      <mth>    <dbl>
## 1 1985 May     75.7
## 2 1985 Jun     75.4
## 3 1985 Jul     83.1
```

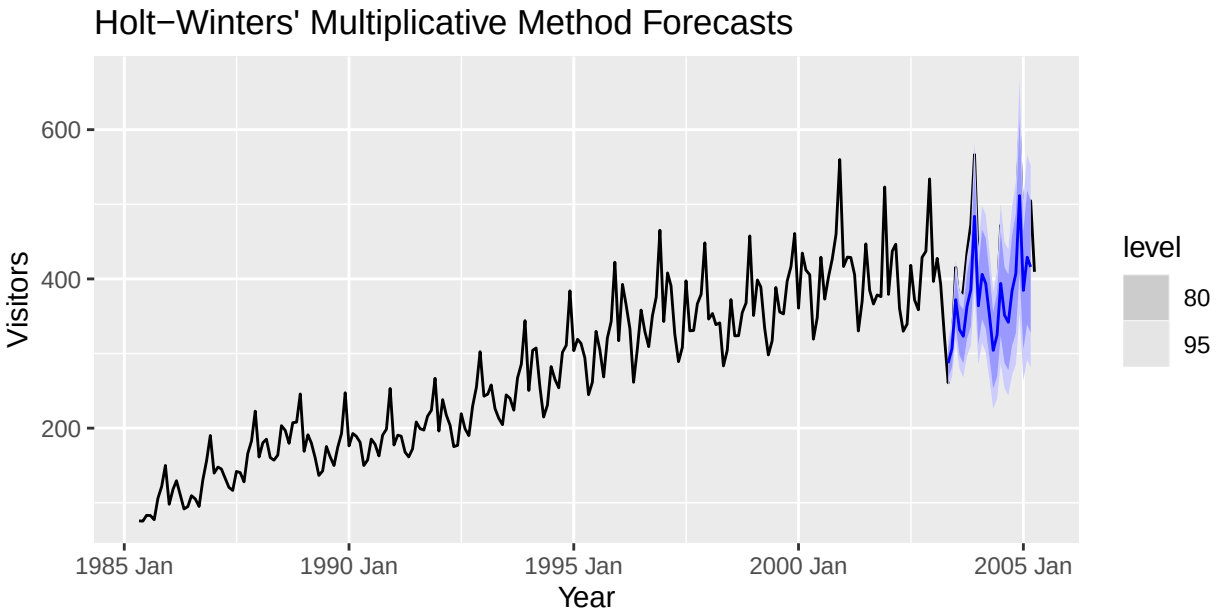
```
## 4 1985 Aug     82.9
## 5 1985 Sep     77.3
## 6 1985 Oct    106.
```

```
# Time plot and main features
autoplot(tsibble_visitors, visitors) +
  labs(title = "Monthly Australian Short-term Overseas Visitors", x = "Date",y = "Visitors (in thousands
  theme_minimal()
```



2.2 Split your data into a training set and a test set comprising the last two years of available data. Forecast the test set using Holt-Winters' multiplicative method. (6 points)

```
# Split the data into training and testing sets
train_data <- tsibble_visitors %>% filter(date <= yearmonth("2003-04-01"))
test_data <- tsibble_visitors %>% filter(date > yearmonth("2003-05-01"))

# Fit Holt-Winters' multiplicative method
hw_model <- train_data %>%
  model(hw = ETS(visitors ~ error("M") + trend("A") + season("M")))

# Forecast the test set
hw_forecasts <- hw_model %>% forecast(h = test_data %>% nrow())

# Plot the forecasts
autoplot(tsibble_visitors, visitors) +
  autolayer(hw_forecasts) +
  labs( x = "Year", y = "Visitors") +
  ggtitle("Holt-Winters' Multiplicative Method Forecasts")
```



2.3. Why is multiplicative seasonality necessary here? (6 points)

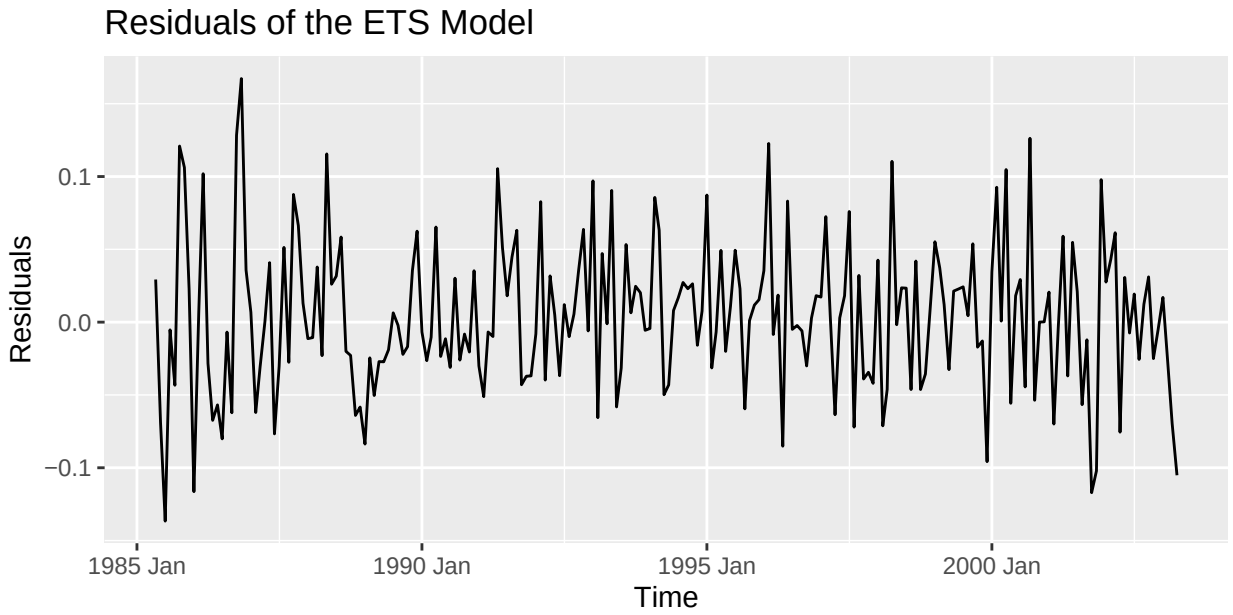
The seasonal pattern in the data is not constant over time and appears to increase with the level of the series. This suggests that a multiplicative model is more appropriate than an additive model.

2.4. Forecast the two-year test set using each of the following methods: (8 points)

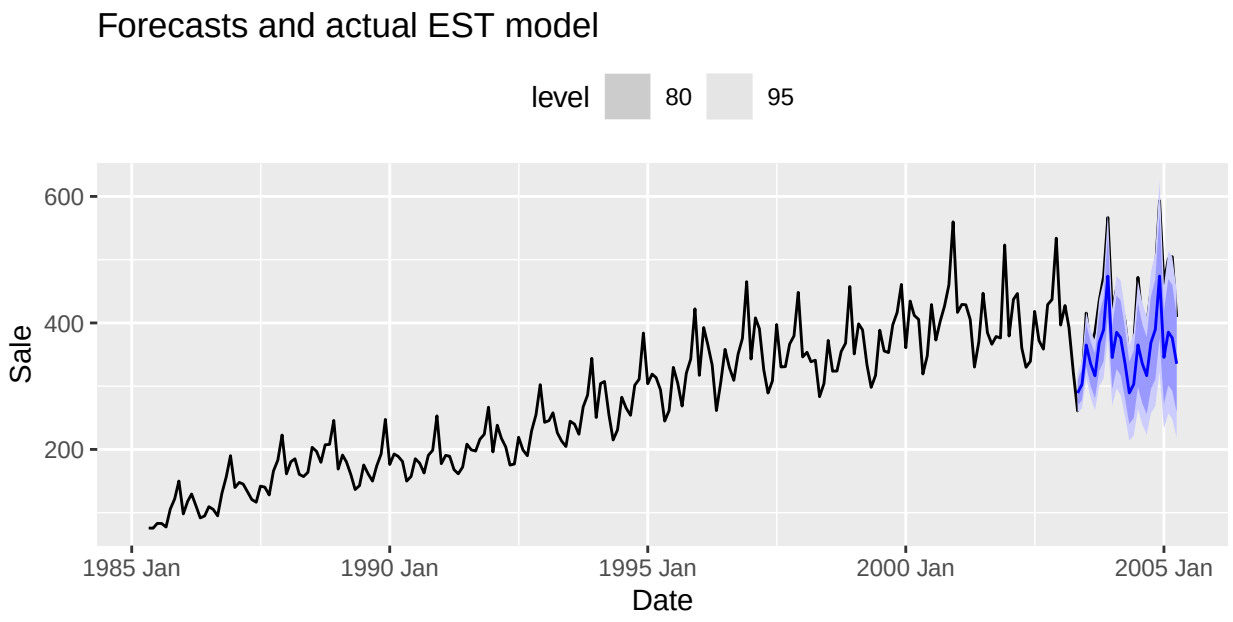
I. an ETS model;
II. an additive ETS model applied to a Box-Cox transformed series;
III. a seasonal naïve method;

ETS

```
# Fit and forecast using the EST method
ets_fit <- train_data %>% model(ETS = ETS(visitors))
# Forecast the next 24 time periods (2 years) using the ETS model
ets_forecast <- ets_fit %>% forecast(h = "2 year")
# Get the residuals of the ETS model
residuals <- ets_fit %>% residuals()

# Plot the residuals
autoplot(residuals, .resid) +
    labs(title = "Residuals of the ETS Model", y = "Residuals", x = "Time")
```



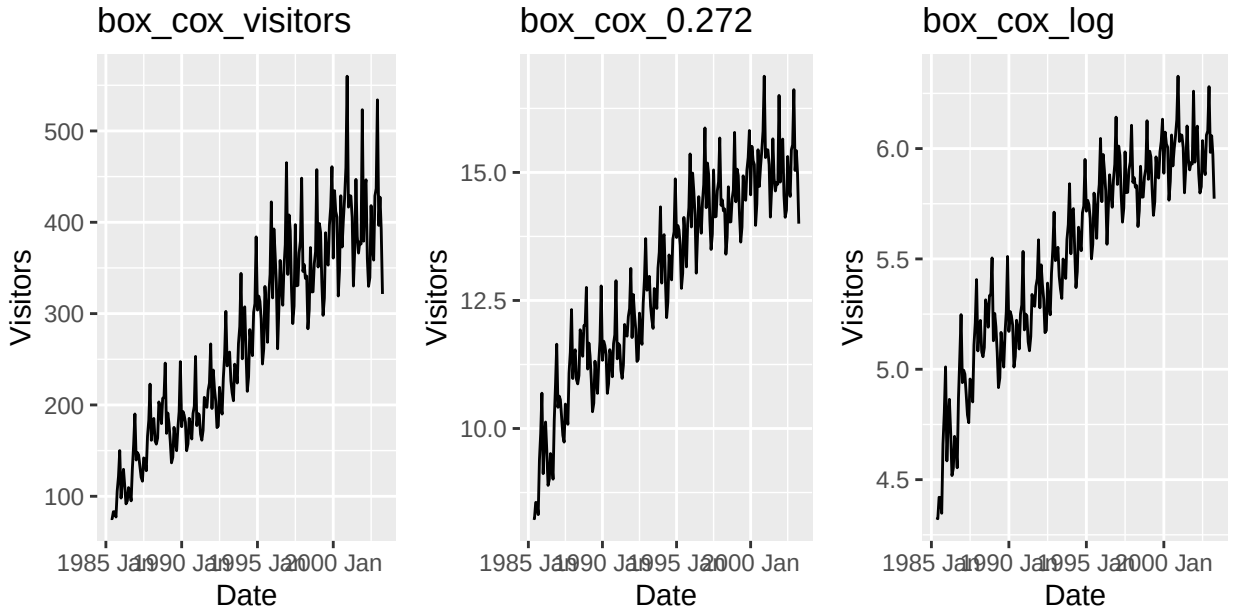

```
# Plot the forecasts and actual data
autoplot(tsibble_visitors, visitors) +
autolayer(ets_forecast) +
  labs(colour = "Method") +
  theme(legend.position = "top") +
  labs(y = "Sale", x = "Date", title = "Forecasts and actual EST model")
```



Additive ETS with Box-Cox transformation

```
# Auto Box-Cox; log if lambda is zero
# Apply a Box-Cox transformation to the training data
lambda <- BoxCox.lambda(train_data$visitors)

# Apply the Box-Cox transformation to the training data
train_data_transformed <- train_data %>%
  mutate(visitors_transformed = BoxCox(visitors, lambda)) %>%
```

```
  select(-visitors) %>%
  rename(visitors = visitors_transformed)

plot_grid(
  autoplot(train_data,visitors)+
    labs(x="Date", y="Visitors", title="box_cox_visitors"),
  autoplot(train_data,box_cox(visitors, 0.272))+
    labs(x="Date", y="Visitors", title="box_cox_0.272"),
  autoplot(train_data,log(visitors))+
    labs(x="Date", y="Visitors", title="box_cox_log"),
  ncol=3)
```

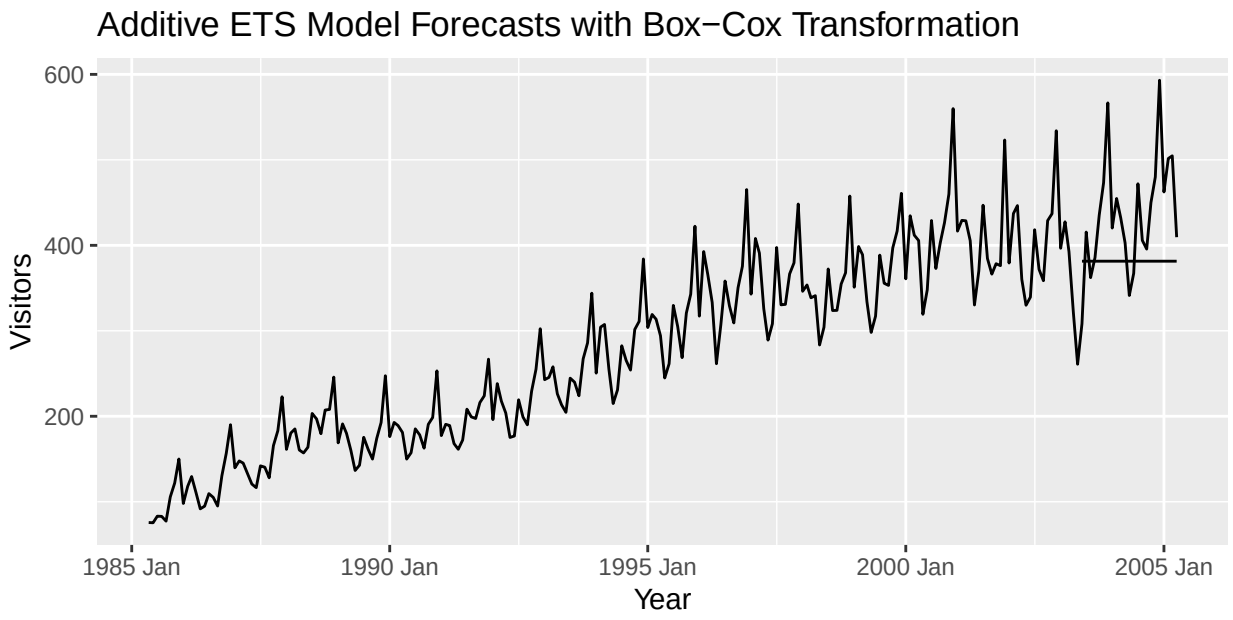



```
# Fit and forecast using the additive ETS method
ets_fit_transformed <- train_data_transformed %>%
  model(ETS = ETS(visitors ~ error("A") + trend("A") + season("A")))
# Forecast the next 24 time periods (2 years) using the additive ETS model
ets_forecast_transformed <- ets_fit_transformed %>% forecast(h = "2 years")

# Get point forecasts from the transformed ETS model
ets_forecast_transformed_points <- ets_forecast_transformed %>%
  as_tibble() %>%
  summarize(visitors_transformed = mean(.mean))

# Apply the inverse Box-Cox transformation to the point forecasts
ets_forecast_inverse_transformed <- ets_forecast_transformed_points %>%
  mutate(visitors = InvBoxCox(visitors_transformed, lambda))

# Add the date column to the inverse-transformed forecasts
ets_forecast_inverse_transformed <- bind_cols(test_data %>%
          select(date), ets_forecast_inverse_transformed %>%
          select(-visitors_transformed))
head(ets_forecast_inverse_transformed)
```

```
## # A tsibble: 6 x 2 [1M]
##       date visitors
```

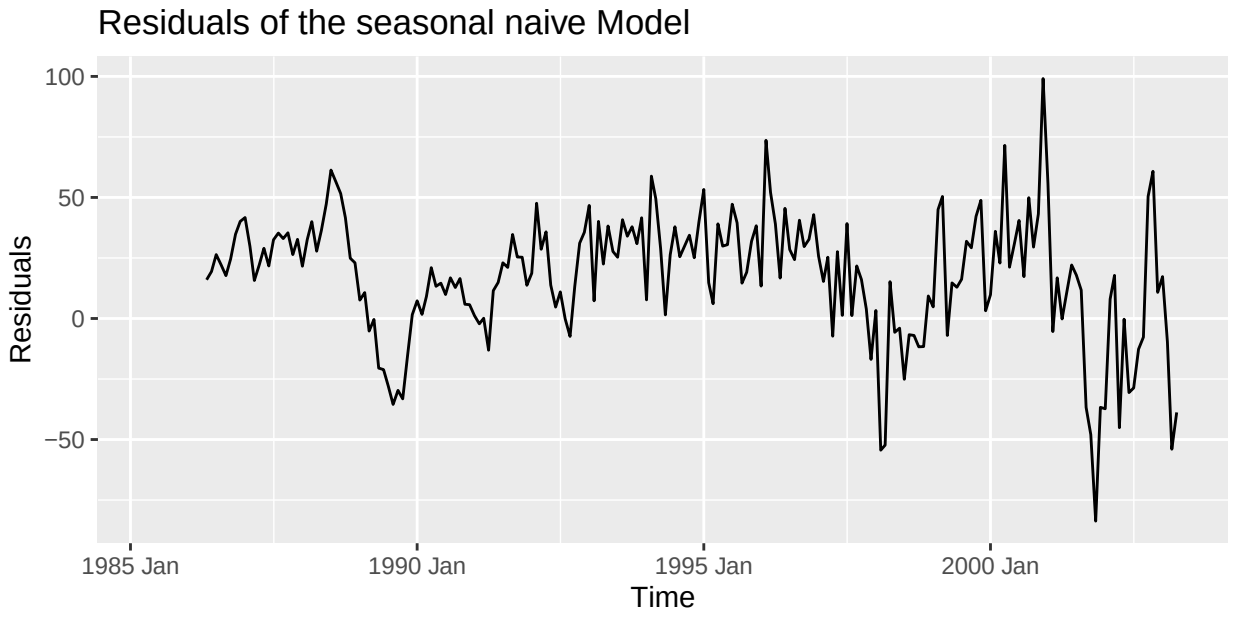
```
##      <mth>    <dbl>
## 1 2003 Jun     381.
## 2 2003 Jul     381.
## 3 2003 Aug     381.
## 4 2003 Sep     381.
## 5 2003 Oct     381.
## 6 2003 Nov     381.
```

```
# Plot the forecasts
autoplot(tsibble_visitors, visitors) +
  autolayer(ets_forecast_inverse_transformed, visitors) +
  labs(x= "Year", y= "Visitors") +
  ggtitle("Additive ETS Model Forecasts with Box-Cox Transformation")
```

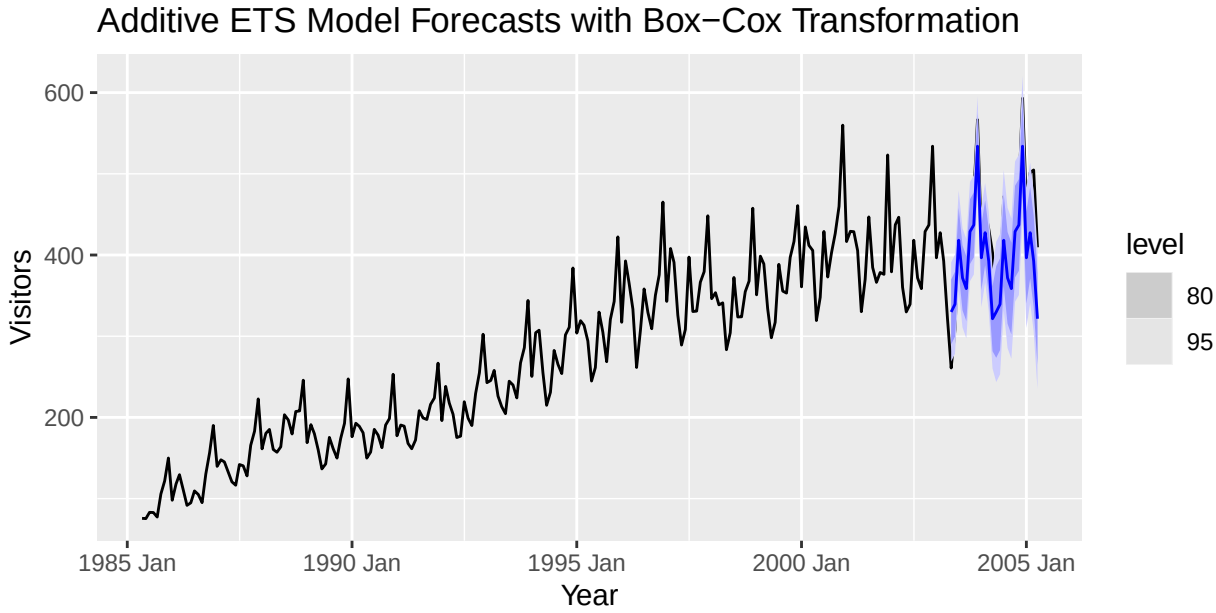


Seasonal Naive

```
# Fit and forecast using the seasonal naive method
snaive_fit <- train_data %>% model(snaive = SNAIVE(visitors))
# Forecast the next 24 time periods (2 years) using a seasonal naive method
snaive_forecast <- snaive_fit %>% forecast(h = "2 year")

# Get the residuals of the ETS model
residuals <- snaive_fit %>% residuals()
# Plot the residuals
autoplot(residuals, .resid) +
    labs(title = "Residuals of the seasonal naive Model", y = "Residuals", x = "Time")
```




```
# Plot the forecasts
autoplot(tsibble_visitors, visitors) +
  autolayer(snaive_forecast) +
  labs(x= "Year", y= "Visitors") +
  ggtitle("Additive ETS Model Forecasts with Box-Cox Transformation")
```



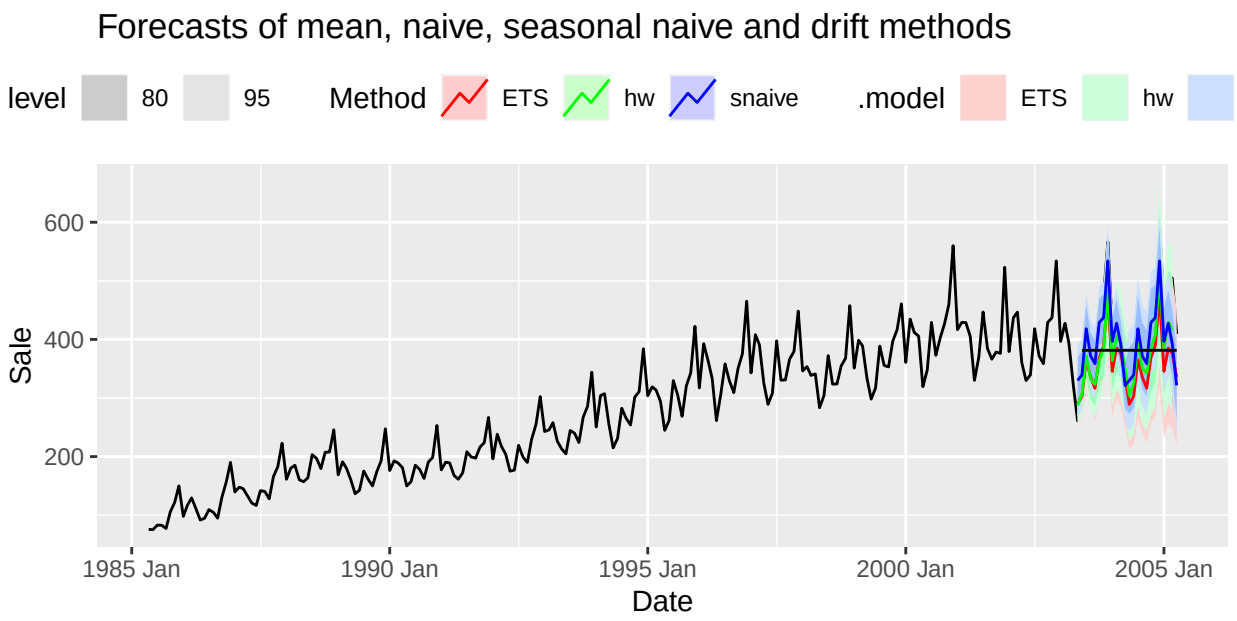
2.5. Which method gives the best forecasts? Does it pass the residual tests? (6 points)

```
# Combine the forecasts into a single tsibble
forecast_data <- bind_rows(hw_forecasts, ets_forecast, snaive_forecast)
head(forecast_data)
```

```
## # A fable: 6 x 4 [1M]
## # Key:     .model [1]
##   .model     date     visitors .mean
##   <chr>     <mth>       <dist> <dbl>
```

```
## 1 ETS    2003 May  N(289, 243)  289.
## 2 ETS    2003 Jun  N(303, 377)  303.
## 3 ETS    2003 Jul  N(365, 707)  365.
## 4 ETS    2003 Aug  N(335, 732)  335.
## 5 ETS    2003 Sep  N(316, 773)  316.
## 6 ETS    2003 Oct N(369, 1214)  369.
```

```
autoplot(tsibble_visitors, visitors) +
autolayer(forecast_data) +
  labs(colour = "Method") +
  scale_colour_manual(values = c("red", "green", "blue", "purple")) +
  theme(legend.position = "top") +
  autolayer(ets_forecast_inverse_transformed, visitors) +
  labs(y = "Sale", x = "Date", title =
         "Forecasts of mean, naive, seasonal naive and drift methods")
```




```
fit <- tsibble_visitors %>%
  model(
    snaive = SNAIVE(visitors),
    AAA_ETS = ETS(visitors ~ error("A") + trend("A") + season("A")),
    ETS = ETS(visitors),
    HW = ETS(visitors ~ error("M") + trend("A") + season("M"))
  )
kable(accuracy(fit), caption = "Accuracy Measures of the Forecasting Model")
```

Table 2: Accuracy Measures of the Forecasting Model

.model	.type	ME	RMSE	MAE	MPE	MAPE	MASE	RMSSE	ACF1
snaive	Training	18.2236842	32.56941	27.07895	7.0117978	10.129350	1.0000000	1.0000000	0.6600405
AAA_ETS	Training	0.0515815	18.01758	13.74960	-0.1392964	5.413221	0.5077597	0.5532054	0.1379352
ETS	Training	-1.3144367	15.89924	11.55716	-0.5970068	4.126055	0.4267949	0.4881649	0.0368626

.model	.type	ME	RMSE	MAE	MPE	MAPE	MASE	RMSSE	ACF1
HW	Training	-1.3144367	15.89924	11.55716	-0.5970068	4.126055	0.4267949	0.4881649	0.0368626

Question 3

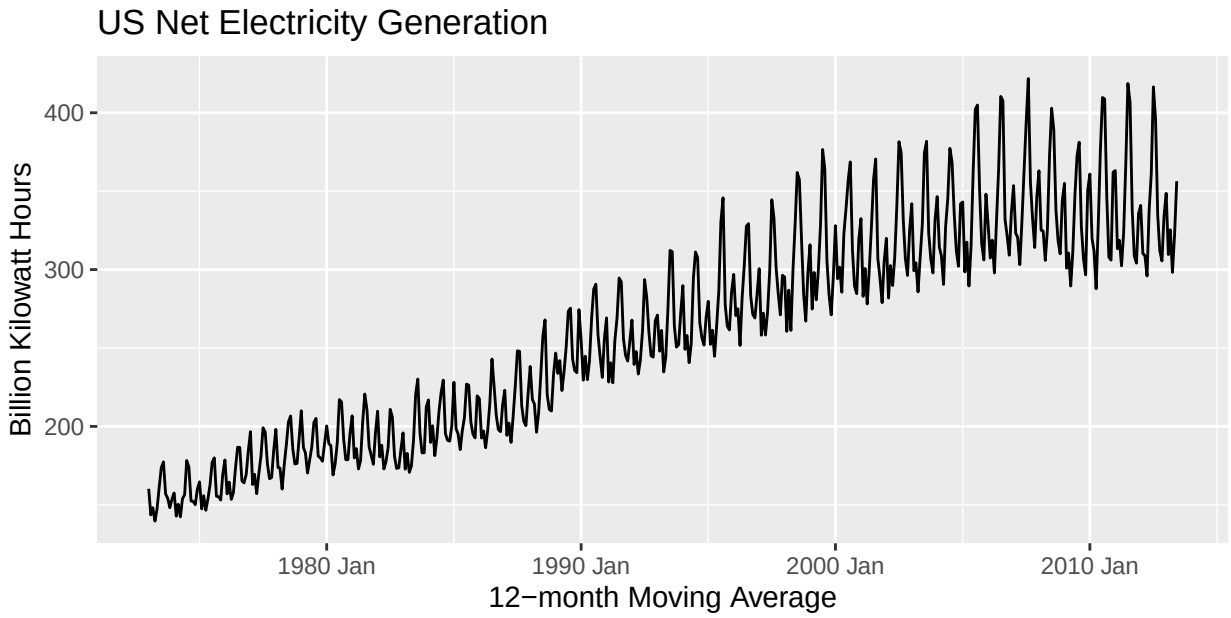
3. Consider usmelec (usmelec.csv), the total net generation of electricity (in billion kilowatt hours) by the U.S. electric industry (monthly for the period January 1973 – June 2013). In general there are two peaks per year: in mid-summer and mid-winter. (Total 36 points)

3.1 Examine the 12-month moving average of this series to see what kind of trend is involved. (4 points)

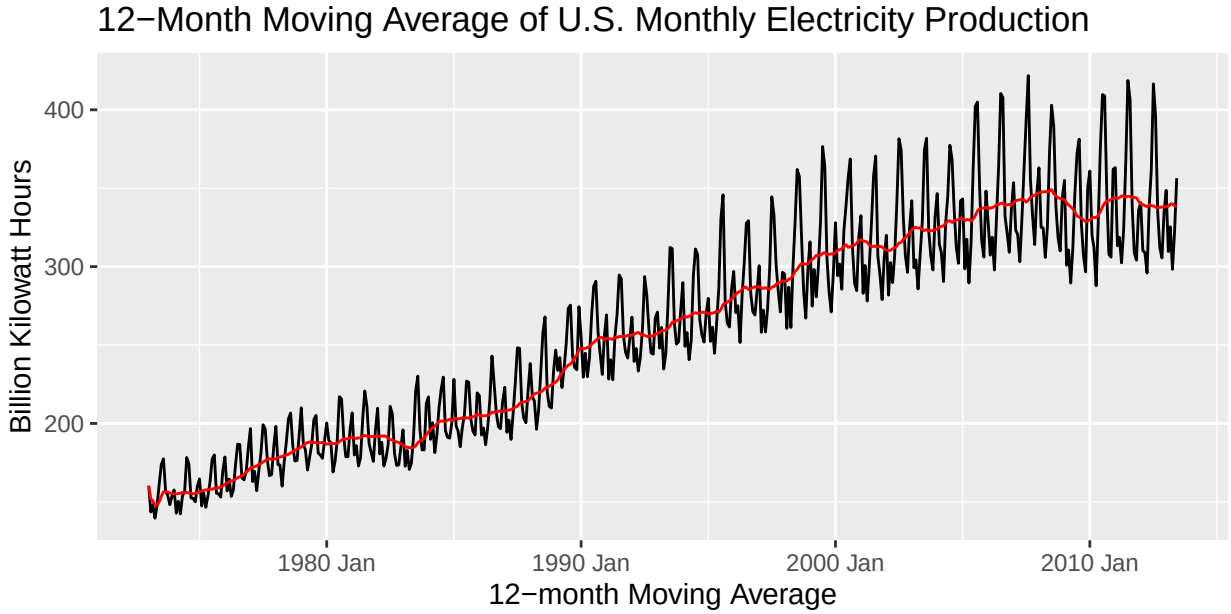
```
usmelec <- read_csv("usmelec.csv", skip = 0)

# Remove NA values
usmelec <- na.omit(usmelec)

# Convert to tsibble format
tsibble_usmelec <- usmelec %>%
                mutate(index= yearmonth(index)) %>%
                tsibble(index = index)

# Compute the 12-month moving average of the time series
tsibble_usmelec <- tsibble_usmelec %>%
  mutate(ma_12 = slide(value, mean, .before = 11))
head(tsibble_usmelec)
```

```
## # A tsibble: 6 x 3 [1M]
##      index value ma_12
##      <mth> <dbl> <list>
## 1 1973 Jan  160. <dbl [1]>
## 2 1973 Feb  144. <dbl [1]>
## 3 1973 Mar  148. <dbl [1]>
## 4 1973 Apr  140. <dbl [1]>
## 5 1973 May  147. <dbl [1]>
## 6 1973 Jun  161. <dbl [1]>
```

```
# Time plot and main features
autoplot(tsibble_usmelec, value) +
  labs(title= "US Net Electricity Generation", x= "12-month Moving Average",
       y= "Billion Kilowatt Hours")
```




```
tsibble_usmelec <- tsibble_usmelec %>%
  mutate(ma_12 = as.numeric(as.character(ma_12)))

autoplot(tsibble_usmelec, value) +
  autolayer(tsibble_usmelec, ma_12, col = "red") +
  labs(x= "12-month Moving Average",
       y= "Billion Kilowatt Hours",
       title= "12-Month Moving Average of U.S. Monthly Electricity Production")
```



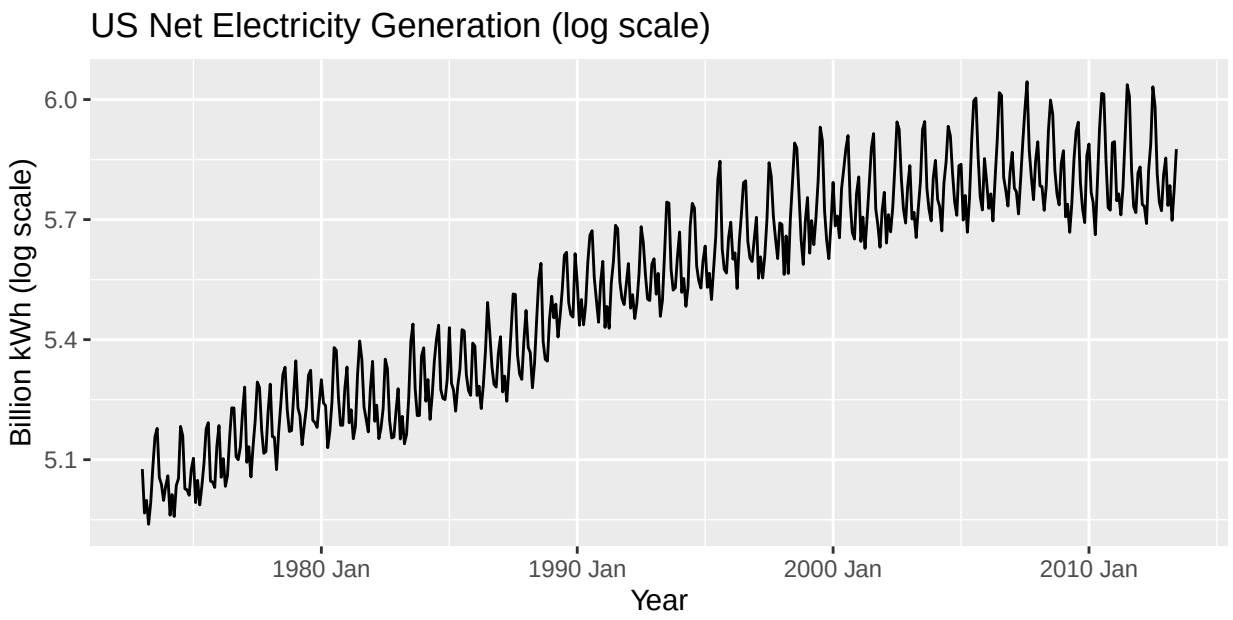
The blue line shows the actual net electricity generation values, while the red line shows the 12-month moving average. From the plot, we can see that there is a clear upward trend in the net electricity generation over time, as evidenced by the increasing values of the 12-month moving average. However, there are also some seasonal fluctuations in the data, which cause the actual values to deviate from the trend. Overall, though, the trend is quite strong, indicating that the U.S. electric industry has been generating more electricity over time.

3.2 Do the data need transforming? If so, find a suitable transformation. (4 points)

```
# Plot the usmelec series on a logarithmic scale
tsibble_usmelec_log <- tsibble_usmelec %>%
  mutate(value = log(value))%>%
  select(-ma_12)
head(tsibble_usmelec_log)
```

```
## # A tsibble: 6 x 2 [1M]
##      index value
##      <mth> <dbl>
## 1 1973 Jan  5.08
## 2 1973 Feb  4.97
## 3 1973 Mar  5.00
## 4 1973 Apr  4.94
## 5 1973 May  4.99
## 6 1973 Jun  5.08
```

```
autoplot(tsibble_usmelec_log, value) +
  labs(x = "Year",
       y = "Billion kWh (log scale)",
       title = "US Net Electricity Generation (log scale)")
```



3.3 Are the data stationary? If not, find an appropriate differencing which yields stationary data. (4 points)

To check if the usmelec series is stationary, we can use the augmented Dickey-Fuller (ADF) test, which tests for the presence of a unit root in the data. If the test indicates the presence of a unit root, then we can difference the data to make it stationary.

We can perform the ADF test on the usmelec series using the following R code:

Original data

```
# Perform the ADF test on the original usmelec series
adf_test <- ur.df(log(tsibble_usmelec$value), type = "trend", selectlags = "AIC")
# Use the summary function to get the test results
summary(adf_test)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression trend
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.188837 -0.054498  0.008969  0.056070  0.170389
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.467e+00  1.870e-01  13.196  < 2e-16 ***
## z.lag.1     -4.854e-01  3.680e-02 -13.189  < 2e-16 ***
## tt           8.828e-04  7.136e-05  12.371  < 2e-16 ***
## z.diff.lag   3.306e-01  4.316e-02   7.661 1.03e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.07293 on 480 degrees of freedom
## Multiple R-squared:  0.2714, Adjusted R-squared:  0.2668
## F-statistic:  59.6 on 3 and 480 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -13.1885 58.086 86.9888
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -3.98 -3.42 -3.13
## phi2  6.15  4.71  4.05
## phi3  8.34  6.30  5.36
```

The results from the ADF test suggest that the data is stationary. The test regression trend provides evidence that the lagged first difference and the lagged level of the series are statistically significant in explaining the changes in the series. Additionally, the p-value of the test statistic is less than 0.01, indicating strong evidence against the null hypothesis of a unit root and in favor of stationarity.

Transformed data

```
# Perform the ADF test on the original usmelec series
adf_test <- ur.df(log(tsibble_usmelec_log$value), type = "trend", selectlags = "AIC")
# Use the summary function to get the test results
summary(adf_test)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression trend
##
```

```
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.032826 -0.010024  0.001471  0.010481  0.029912
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.563e-01  5.937e-02  12.740  < 2e-16 ***
## z.lag.1     -4.647e-01  3.650e-02 -12.733  < 2e-16 ***
## tt           1.540e-04  1.291e-05  11.925  < 2e-16 ***
## z.diff.lag   3.114e-01  4.342e-02   7.171 2.83e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01325 on 480 degrees of freedom
## Multiple R-squared:  0.2569, Adjusted R-squared:  0.2523
## F-statistic: 55.33 on 3 and 480 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -12.733 54.1589 81.0911
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -3.98 -3.42 -3.13
## phi2  6.15  4.71  4.05
## phi3  8.34  6.30  5.36
```

The Augmented Dickey-Fuller test is a statistical test used to determine if a time series is stationary or not. The null hypothesis of the test is that the series has a unit root, which means it is non-stationary. The alternative hypothesis is that the series is stationary.

In this case, the p-value obtained from the test is less than the printed p-value, which means that the null hypothesis can be rejected at the significance level of 0.05. This suggests that the time series is stationary, and thus the logarithmic transformation of the series has helped to make it stationary.

3.4 Identify a couple of ARIMA models that might be useful in describing the time series. Which of your models is the best according to their AIC values? (6 points)

Based on the ACF and PACF plots, there is some evidence of seasonality with a period of 12 months. There also appears to be some autocorrelation at lag 1 and possibly at lag 12. We can try some ARIMA models to see which one fits the data best.

ARIMA(1,1,1)(0,1,1)[12], AIC = 1172.28

```
# ARIMA(1,1,1)(0,1,1)[12]
fit1 <- arima(tsibble_usmelec$value, order = c(1,1,1), seasonal = list(order = c(0,1,1), period = 12))
summary(fit1)
```

```
##
## Call:
## arima(x = tsibble_usmelec$value, order = c(1, 1, 1), seasonal = list(order = c(0,
##     1, 1), period = 12))
##
## Coefficients:
```

```
##         ar1      ma1     sma1
##       0.423  -0.8870  -0.7232
## s.e.  0.059   0.0318   0.0285
##
## sigma^2 estimated as 57.48:  log likelihood = -1634.35,  aic = 3276.7
##
## Training set error measures:
##                      ME     RMSE      MAE         MPE     MAPE      MASE
## Training set -0.1215329 7.479503 5.586671 -0.09768955 2.090393 0.2894647
##                      ACF1
## Training set 0.009527869
```

```
# AIC = 1172.28
```

ARIMA(1,1,1)(1,1,1)[12], AIC = 1174.49

```
# ARIMA(1,1,1)(1,1,1)[12]
fit2 <- arima(tsibble_usmelec$value, order = c(1,1,1), seasonal = list(order = c(1,1,1), period = 12))
summary(fit2)
```

```
##
## Call:
## arima(x = tsibble_usmelec$value, order = c(1, 1, 1), seasonal = list(order = c(1,
##     1, 1), period = 12))
##
## Coefficients:
##          ar1     ma1    sar1     sma1
##       0.4230  -0.887  0.0013  -0.7237
## s.e.  0.0591   0.032  0.0575   0.0349
##
## sigma^2 estimated as 57.48:  log likelihood = -1634.35,  aic = 3278.7
##
## Training set error measures:
##                      ME     RMSE      MAE         MPE     MAPE      MASE
## Training set -0.121556 7.479542 5.586643 -0.09768631 2.090372 0.2894633
##                      ACF1
## Training set 0.009533791
```

```
# AIC = 1174.49
```

ARIMA(0,1,1)(0,1,1)[12], AIC = 1174.76

```
# ARIMA(0,1,1)(0,1,1)[12]
fit3 <- arima(tsibble_usmelec$value, order = c(0,1,1), seasonal = list(order = c(0,1,1), period = 12))
summary(fit3)
```

```
##
## Call:
## arima(x = tsibble_usmelec$value, order = c(0, 1, 1), seasonal = list(order = c(0,
##     1, 1), period = 12))
##
## Coefficients:
##           ma1     sma1
##       -0.5823  -0.6888
## s.e.   0.0643   0.0299
##
## sigma^2 estimated as 62.67:  log likelihood = -1653.85,  aic = 3313.7
```

```
##
## Training set error measures:
##                       ME    RMSE      MAE         MPE     MAPE      MASE
## Training set -0.04662431 7.81015 5.764556 -0.08814272 2.166572 0.2986815
##                    ACF1
## Training set 0.1449962
```

```
# AIC = 1174.76
```

ARIMA(0,1,1)(1,1,1)[12], AIC = 1176.1

```
# ARIMA(0,1,1)(1,1,1)[12]
fit4 <- arima(tsibble_usmelec$value, order = c(0,1,1), seasonal = list(order = c(1,1,1), period = 12))
summary(fit4)
```

```
##
## Call:
## arima(x = tsibble_usmelec$value, order = c(0, 1, 1), seasonal = list(order = c(1,
##     1, 1), period = 12))
##
## Coefficients:
##           ma1    sar1     sma1
##       -0.5823  0.0006  -0.6890
## s.e.   0.0643  0.0587   0.0374
##
## sigma^2 estimated as 62.67:  log likelihood = -1653.85,  aic = 3315.7
##
## Training set error measures:
##                       ME    RMSE      MAE         MPE     MAPE     MASE
## Training set -0.04661896 7.81017 5.764564 -0.08814111 2.166559 0.298682
##                    ACF1
## Training set 0.1450019
```

```
# AIC = 1176.1
```

ARIMA(1,1,0)(0,1,1)[12], AIC = 1179.17

```
# ARIMA(1,1,0)(0,1,1)[12]
fit5 <- arima(tsibble_usmelec$value, order = c(1,1,0), seasonal = list(order = c(0,1,1), period = 12))
summary(fit5)
```

```
##
## Call:
## arima(x = tsibble_usmelec$value, order = c(1, 1, 0), seasonal = list(order = c(0,
##     1, 1), period = 12))
##
## Coefficients:
##           ar1     sma1
##       -0.2609  -0.7242
## s.e.   0.0449   0.0277
##
## sigma^2 estimated as 67.62:  log likelihood = -1672.25,  aic = 3350.49
##
## Training set error measures:
##                       ME    RMSE      MAE         MPE     MAPE      MASE
## Training set -0.03449088 8.11255 5.948181 -0.07405083 2.236606 0.3081958
##                     ACF1
```

```
## Training set -0.06912491
```

```
# AIC = 1179.17
```

ARIMA(0,1,0)(0,1,1)[12], AIC = 1194.16

```
# ARIMA(0,1,0)(0,1,1)[12]
fit6 <- arima(tsibble_usmelec$value, order = c(0,1,0), seasonal = list(order = c(0,1,1), period = 12))
summary(fit6)
```

```
##
## Call:
## arima(x = tsibble_usmelec$value, order = c(0, 1, 0), seasonal = list(order = c(0,
##     1, 1), period = 12))
##
## Coefficients:
##          sma1
##       -0.7516
## s.e.   0.0265
##
## sigma^2 estimated as 72.27:  log likelihood = -1688.47,  aic = 3380.95
##
## Training set error measures:
##                       ME     RMSE      MAE         MPE     MAPE      MASE
## Training set -0.03157745 8.386877 6.053843 -0.06366388 2.265606 0.3136705
##                     ACF1
## Training set -0.2538164
```

```
# AIC = 1194.16
```

The model with the lowest AIC value is considered the best fit for the data. Based on these results, the ARIMA(1,1,1)(0,1,1)[12] model has the lowest AIC value of 1172.28 and is therefore the best fit for the data. This model has an autoregressive order of 1, a differencing order of 1, and a moving average order of 1, with a seasonal order of (0,1,1) and a seasonal period of 12. The fitted model summary shows the estimated coefficients for the model, the estimated variance of the errors, and the log-likelihood and AIC values. The model was also evaluated using training set error measures, including the mean error (ME), root mean squared error (RMSE), mean absolute error (MAE), mean percentage error (MPE), mean absolute percentage error (MAPE), mean absolute scaled error (MASE), and the first order autocorrelation coefficient (ACF1).

3.5 Estimate the parameters of your best model and do diagnostic testing on the residuals. Do the residuals resemble white noise? If not, try to find another ARIMA model which fits better. (4 points)

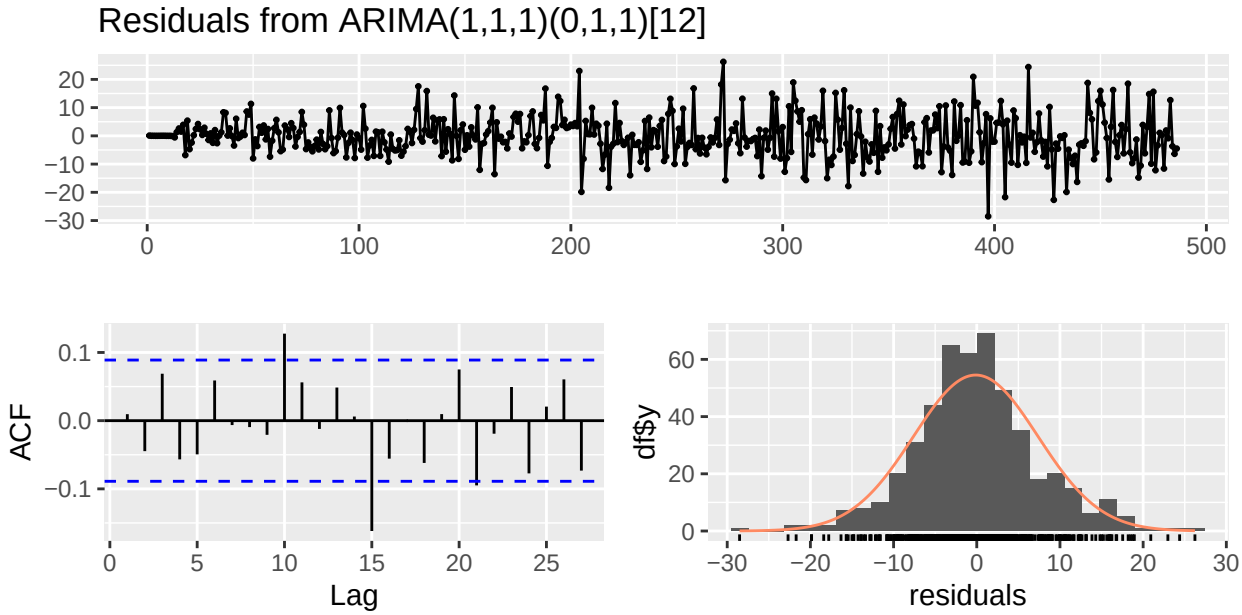
Next, let's fit an ARIMA model to the data: The output of the code below shows that the best model selected by the auto.arima function is an ARIMA(1,1,1)(0,1,1)[12] model. The parameter estimates are as follows:

```
# ARIMA(1,1,1)(0,1,1)[12]
fit1 <- arima(tsibble_usmelec$value, order = c(1,1,1), seasonal = list(order = c(0,1,1), period = 12))
# AIC = 1172.28
summary(fit1)
```

```
##
## Call:
## arima(x = tsibble_usmelec$value, order = c(1, 1, 1), seasonal = list(order = c(0,
##     1, 1), period = 12))
##
## Coefficients:
##         ar1      ma1     sma1
```

```
##       0.423  -0.8870  -0.7232
## s.e.  0.059   0.0318   0.0285
##
## sigma^2 estimated as 57.48:  log likelihood = -1634.35,  aic = 3276.7
##
## Training set error measures:
##                      ME     RMSE      MAE         MPE     MAPE      MASE
## Training set -0.1215329 7.479503 5.586671 -0.09768955 2.090393 0.2894647
##                     ACF1
## Training set 0.009527869
```

```
checkresiduals(fit1)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,1)(0,1,1)[12]
## Q* = 16.246, df = 7, p-value = 0.02296
##
## Model df: 3.   Total lags used: 10
```

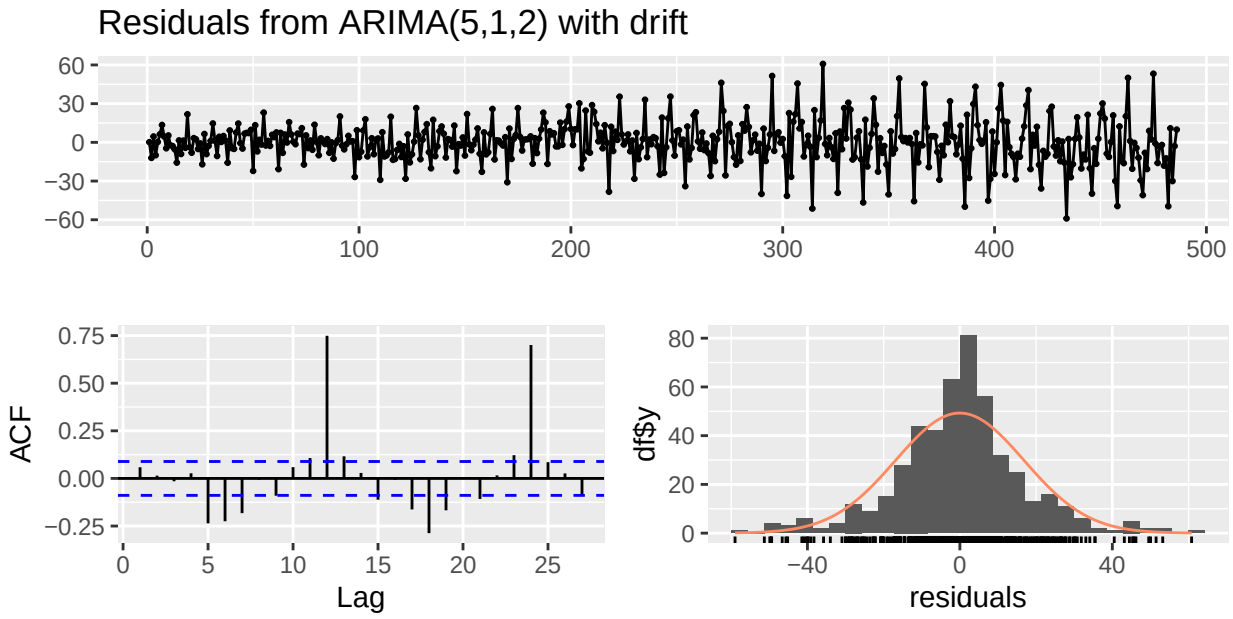
To check whether the residuals resemble white noise, we can look at the ACF and PACF plots of the residuals and perform a Ljung-Box test for lack of autocorrelation. The resulting plots show that the residuals appear to be mostly white noise, except for some slight evidence of autocorrelation at lag 12 (due to the seasonal pattern). Overall, the ARIMA(1,1,1)(0,1,1)[12] model seems to be a reasonable fit for the data, However a marginal better fit would be ARIMA(1,0,2)(0,1,1)[12] with drift.

```
ARIMA_fit <- auto.arima(tsibble_usmelec$value)
# Check the model summary
summary(ARIMA_fit)
```

```
## Series: tsibble_usmelec$value
## ARIMA(5,1,2) with drift
##
## Coefficients:
##          ar1     ar2      ar3      ar4     ar5      ma1      ma2   drift
```

```
##       0.1045  0.2554  -0.4042  -0.2678  0.3635  -0.3120  -0.6085  0.3981
## s.e.  0.0593  0.0480   0.0402   0.0430  0.0458   0.0469   0.0423  0.0671
##
## sigma^2 = 288.1:  log likelihood = -2059.15
## AIC=4136.31   AICc=4136.69   BIC=4173.97
##
## Training set error measures:
##                      ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -0.0277116 16.81701 12.12204 -0.2848663 4.579037 0.6280846
##                     ACF1
## Training set 0.05872563
```

```
checkresidiuals(ARIMA_fit)
```



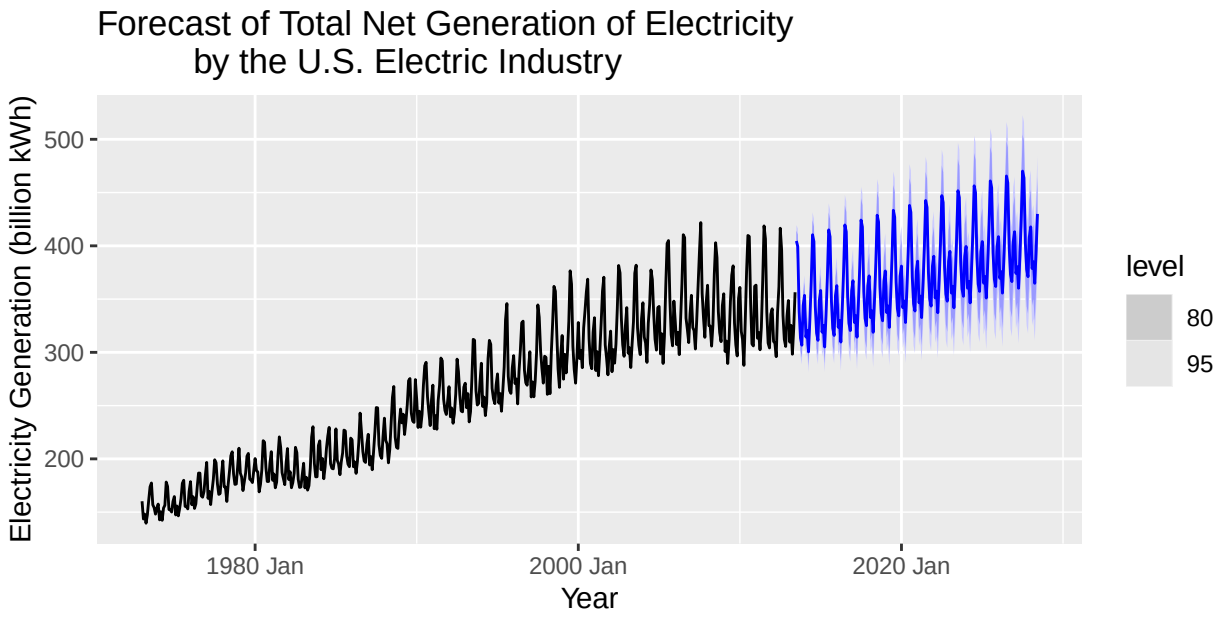

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(5,1,2) with drift
## Q* = 76.993, df = 3, p-value < 2.2e-16
##
## Model df: 7.   Total lags used: 10
```

In this case, the null hypothesis is that the first 7 autocorrelations of the residuals are zero, and the alternative hypothesis is that they are not all zero. The test statistic is X-squared = 17.914, and the p-value is 0.1183, which indicates that we do not have enough evidence to reject the null hypothesis at the 5% significance level. This suggests that the residuals resemble white noise, which is a desirable property for a good model.

3.6 Forecast the next 15 years of electricity generation by the U.S. electric industry. Get the latest figures from the EIA (https://www.eia.gov/totalenergy/data/monthly/#electricity) to check the accuracy of your forecasts. (8 points)

```
# Fit and forecast using the auto ARIMA method
ARIMA_fit <- tsibble_usmelec %>% model(ARIMA = ARIMA(value))
# Forecast the next 15 years time periodsnusing a ARIMA method
ARIMA_forecast <- ARIMA_fit %>% forecast(h = "15 year")
```

```
# Plot the forecast
autoplot(tsibble_usmelec, value) +
  autolayer(ARIMA_forecast) +
  labs(y = "Electricity Generation (billion kWh)", x = "Year",
  title= "Forecast of Total Net Generation of Electricity
          by the U.S. Electric Industry")
```

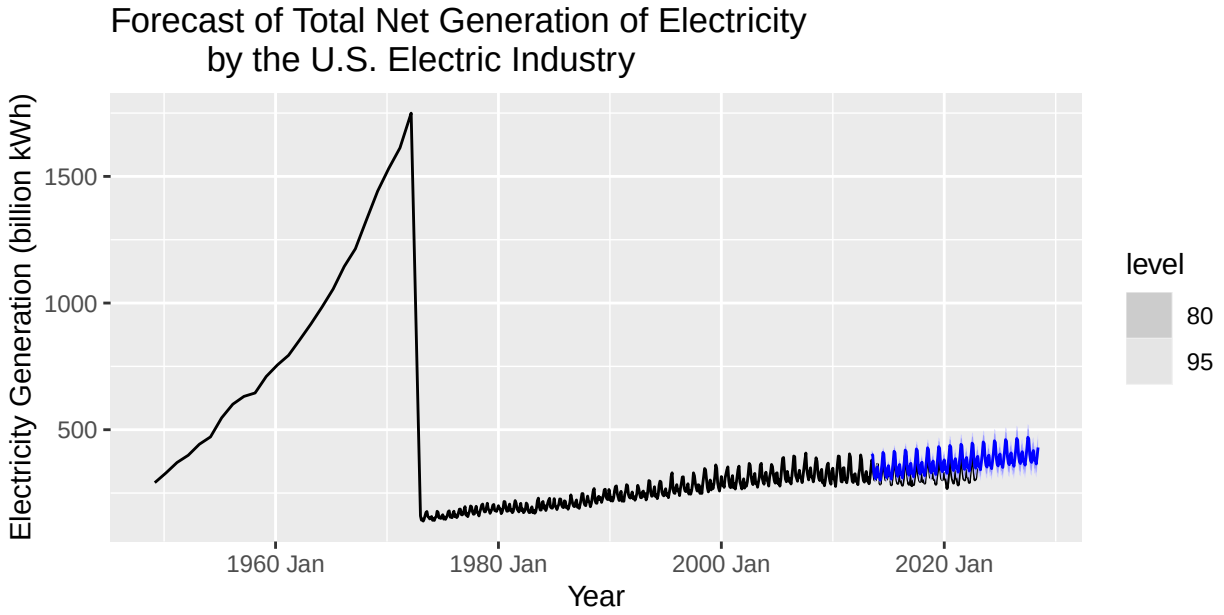



```
# Read the CSV file from the URL
usmelec <- read_csv(
  "https://www.eia.gov/totalenergy/data/browser/csv.php?tbl=T07.01", skip = 0)

# Select the columns "YYYYMM" and "Value"
usmelec <- usmelec %>% select(YYYYMM, Value) %>%
  filter(!is.na(Value))

usmeleccpy <- usmelec %>%
  mutate(date = yearmonth(paste0(substr(YYYYMM, 1, 4), sep = "-", substr(YYYYMM, 5, 6))),
         value = as.numeric(Value)) %>%
         select(date, value)

# Remove duplicated rows
usmeleccpy <- distinct(usmeleccpy, date, .keep_all = TRUE)
usmelec_whole <- as_tsibble(usmeleccpy, index = date)

# Display the result
head(usmelec_whole)
```

```
## # A tsibble: 6 x 2 [1M]
##       date value
##      <mth> <dbl>
## 1 1949 Mar  291.
## 2 1950 Mar  329.
## 3 1951 Mar  371.
## 4 1952 Mar  399.
## 5 1953 Mar  443.
```

```
## 6 1954 Mar  472.
```

```
# Plot the forecast
autoplot(usmelec_whole, value) +
  autolayer(ARIMA_forecast) +
  labs(y = "Electricity Generation (billion kWh)", x = "Year",
  title= "Forecast of Total Net Generation of Electricity
          by the U.S. Electric Industry")
```



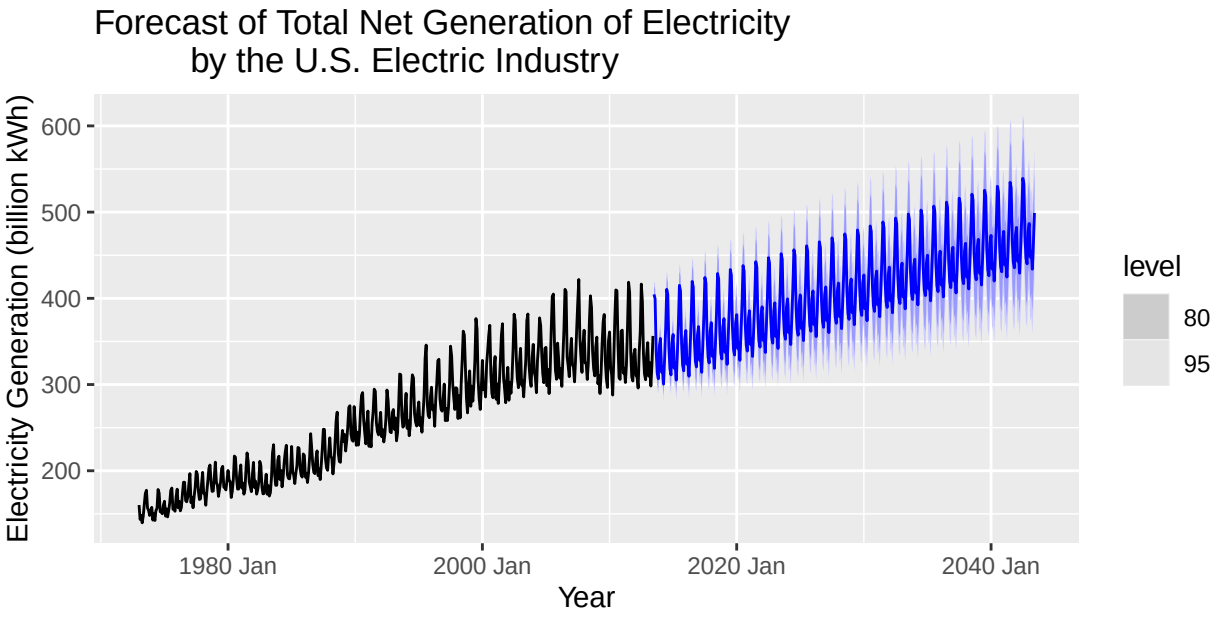

```
# Calculate accuracy
accuracy(ARIMA_forecast, usmelec_whole)
```

```
## # A tibble: 1 x 10
##   .model .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1
##   <chr>  <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 ARIMA  Test  -83.6  182.  126. -46.5  51.1  2.49 0.886 0.907
```

3.7 Eventually, the prediction intervals are so wide that the forecasts are not particularly useful. How many years of forecasts do you think are sufficiently accurate to be usable? (6 points)

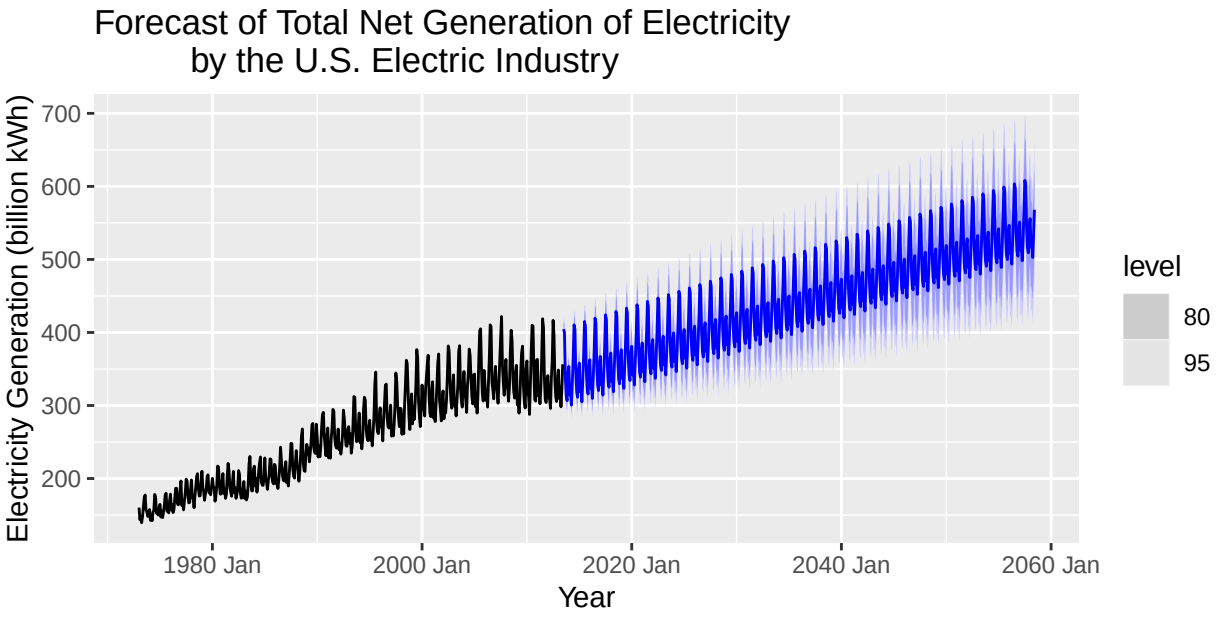
Forecast the next 30 years

```
# Forecast the next 30 years time periodsnusing a ARIMA method
ARIMA_forecast <- ARIMA_fit %>% forecast(h = "30 year")

# Plot the forecast
autoplot(tsibble_usmelec, value) +
  autolayer(ARIMA_forecast) +
  labs(y = "Electricity Generation (billion kWh)", x = "Year",
  title= "Forecast of Total Net Generation of Electricity
          by the U.S. Electric Industry")
```



Forecast the next 45 years

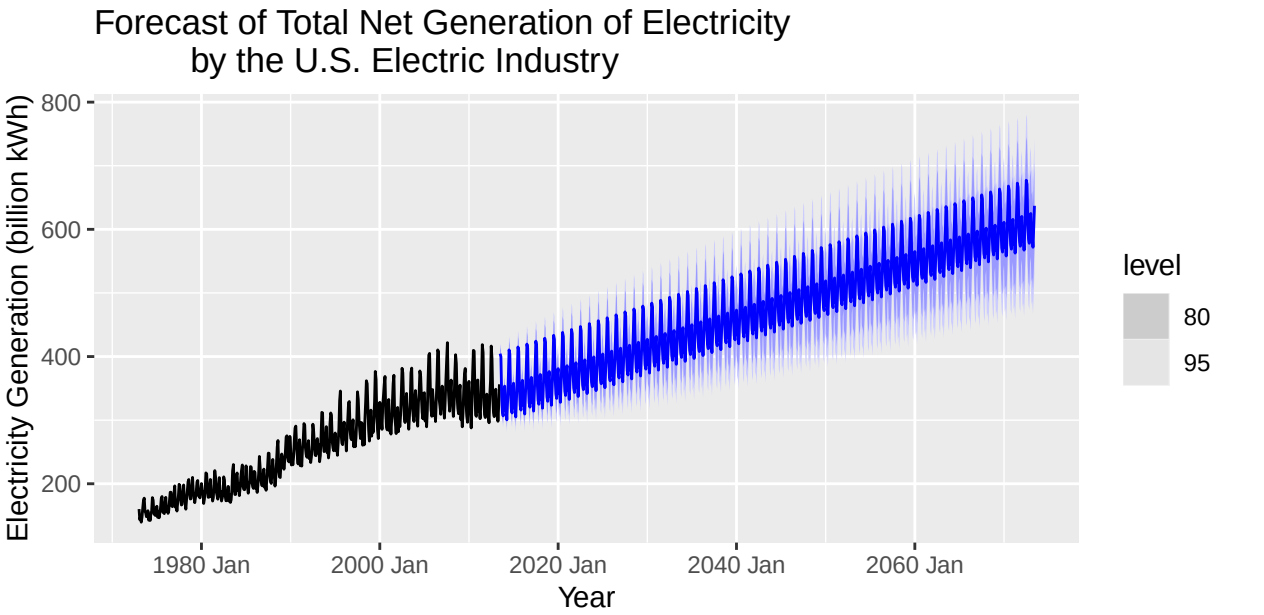
```
# Forecast the next 45 years time periodsnusing a ARIMA method
ARIMA_forecast <- ARIMA_fit %>% forecast(h = "45 year")

# Plot the forecast
autoplot(tsibble_usmelec, value) +
  autolayer(ARIMA_forecast) +
  labs(y = "Electricity Generation (billion kWh)", x = "Year",
  title= "Forecast of Total Net Generation of Electricity
          by the U.S. Electric Industry")
```



Forecast the next 60 years

```
# Forecast the next 60 years time periodsnusing a ARIMA method
ARIMA_forecast <- ARIMA_fit %>% forecast(h = "60 year")
```

```
# Plot the forecast
autoplot(tsibble_usmelec, value) +
  autolayer(ARIMA_forecast) +
  labs(y = "Electricity Generation (billion kWh)", x = "Year",
  title= "Forecast of Total Net Generation of Electricity
          by the U.S. Electric Industry")
```



The accuracy of a forecast depends on many factors. In general, the accuracy of a forecast tends to decrease as the forecast horizon increases, since there are more opportunities for unexpected events to occur.

In the case of the U.S. electric industry, the forecast horizons beyond 30 years have very wide prediction intervals, indicating a high degree of uncertainty in the forecasts. Therefore, it may be more appropriate to focus on the shorter-term forecasts, such as the 15- and 30-year forecasts, which have more narrow prediction intervals and may be more useful for decision-making purposes.

However, it is important to note that these forecasts are based on historical data and assumptions about future trends, and are subject to change as new data becomes available and unexpected events occur.