Data 624 - HW5 (Fall 2024)

Khyati Naik

```
library(fpp3)
```

```
## Warning: package 'fpp3' was built under R version 4.3.3
```

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

```
## v tibble      3.2.1     v tsibble     1.1.3
## v dplyr       1.1.4     v tsibbledata 0.4.1
## v tidyr       1.3.0     v feasts      0.3.2
## v lubridate   1.9.2     v fable       0.3.4
## v ggplot2     3.5.1     v fabletools  0.4.2
```

```
## Warning: package 'dplyr' was built under R version 4.3.2
```

```
## Warning: package 'tidyr' was built under R version 4.3.2
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

```
## Warning: package 'tsibble' was built under R version 4.3.2
```

```
## Warning: package 'tsibbledata' was built under R version 4.3.3
```

```
## Warning: package 'feasts' was built under R version 4.3.3
```

```
## Warning: package 'fabletools' was built under R version 4.3.3
```

```
## Warning: package 'fable' was built under R version 4.3.3
```

```
## -- 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()
```

8.1 Consider the the number of pigs slaughtered in Victoria, available in the aus_livestock dataset.

a. Use the ETS() function to estimate the equivalent model for simple exponential smoothing. Find the optimal values of alpha and l(0) , and generate forecasts for the next four months.

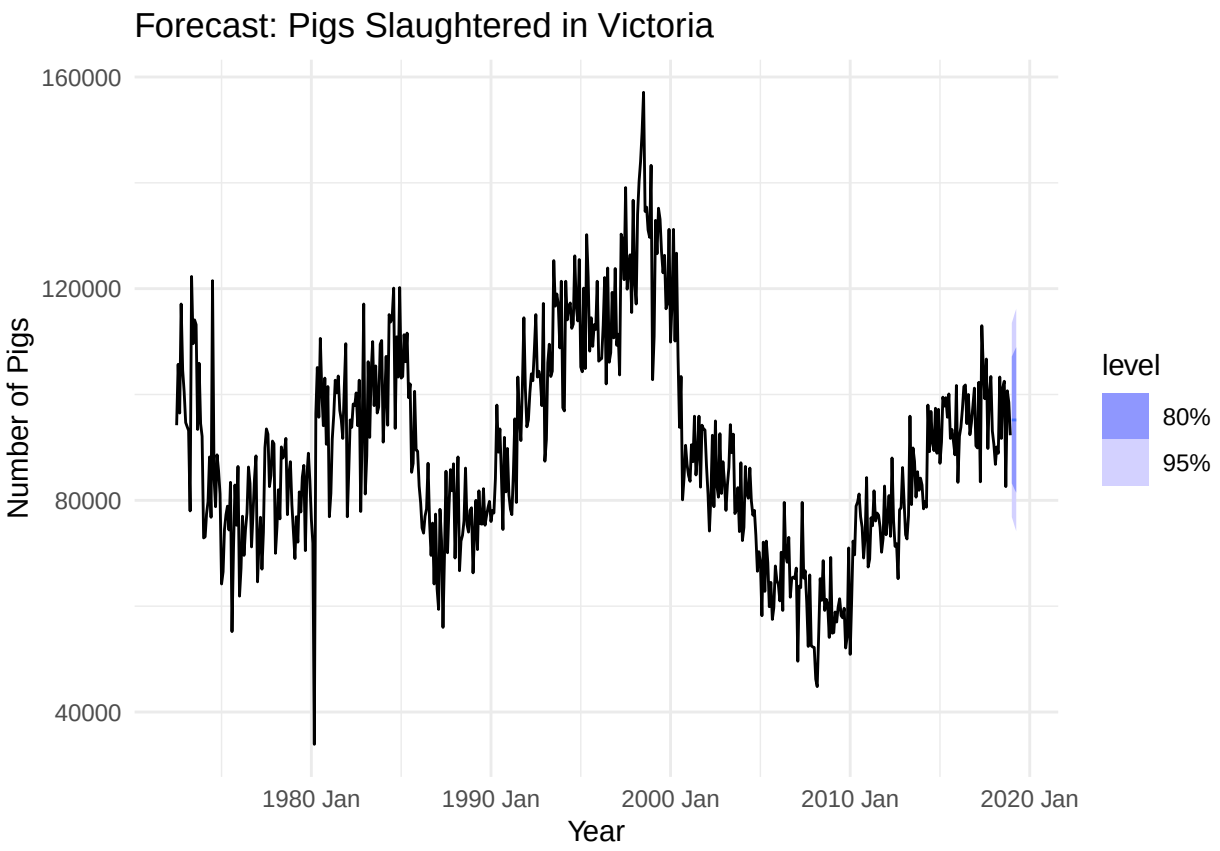
```
# Filter data for pigs in Victoria and fit an ETS model
vic_pigs_model <- aus_livestock %>%
  filter(State == "Victoria", Animal == "Pigs") %>%  # Streamlined filtering
  model(ETS(Count ~ error("A") + trend("N") + season("N")))  # ETS model with additive error, no trend

# Output model fit
report(vic_pigs_model)
```

```
## Series: Count
## Model: ETS(A,N,N)
##   Smoothing parameters:
##     alpha = 0.3221247
##
##   Initial states:
##      l[0]
##  100646.6
##
##   sigma^2:  87480760
##
##      AIC     AICc      BIC
## 13737.10 13737.14 13750.07
```

```
# Generate a 4-period forecast
vic_pigs_forecast <- vic_pigs_model %>%
  forecast(h = 4)

vic_pigs_forecast
```

```
## # A fable: 4 x 6 [1M]
## # Key:     Animal, State, .model [1]
##   Animal State    .model                          Month             Count  .mean
##   <fct>  <fct>    <chr>                           <mth>            <dist>  <dbl>
## 1 Pigs   Victoria "ETS(Count ~ error(\"A\") +~ 2019 Jan N(95187, 8.7e+07) 95187.
## 2 Pigs   Victoria "ETS(Count ~ error(\"A\") +~ 2019 Feb N(95187, 9.7e+07) 95187.
## 3 Pigs   Victoria "ETS(Count ~ error(\"A\") +~ 2019 Mar N(95187, 1.1e+08) 95187.
## 4 Pigs   Victoria "ETS(Count ~ error(\"A\") +~ 2019 Apr N(95187, 1.1e+08) 95187.
```

```
# Plot the forecast with custom styling
vic_pigs_forecast %>%
  autoplot(filter(aus_livestock, State == "Victoria", Animal == "Pigs")) +  # Plot forecast against act
  ggtitle("Forecast: Pigs Slaughtered in Victoria") +  # Improved title
  labs(y = "Number of Pigs", x = "Year") +  # Clear axis labels
  theme_minimal()  # Apply a clean theme for better visual aesthetics
```



b. Compute a 95% prediction interval for the first forecast using y(hat) +- 1.96s where s is the standard deviation of the residuals. Compare your interval with the interval produced by R.

```
# Compute the mean predicted value from the forecast
y_hat <- mean(vic_pigs_forecast$.mean[1])

# Get residuals by applying the augment function to the model
augmented_fit <- augment(vic_pigs_model)

# Compute the standard deviation of the residuals
residual_sd <- sd(augmented_fit$.resid)

# Calculate the 95% prediction interval
upper_limit_95 <- y_hat + (residual_sd * 1.96)
lower_limit_95 <- y_hat - (residual_sd * 1.96)

# Store the 95% prediction interval
interval_95 <- c(lower_limit_95, upper_limit_95)

# Display the interval values
interval_95
```

```
## [1]  76871.01 113502.10
```

```r
# Calculate the 95% forecast intervals using the updated forecast object
vic_pigs_hilo <- vic_pigs_forecast %>% hilo()

# Output the 95% prediction interval for the first forecast period
vic_pigs_hilo$`95%`[1]
```

```
## <hilo[1]>
## [1] [76854.79, 113518.3]95
```

8.5 Data set global_economy contains the annual Exports from many countries. Select one country to analyse.

a. Plot the Exports series and discuss the main features of the data.

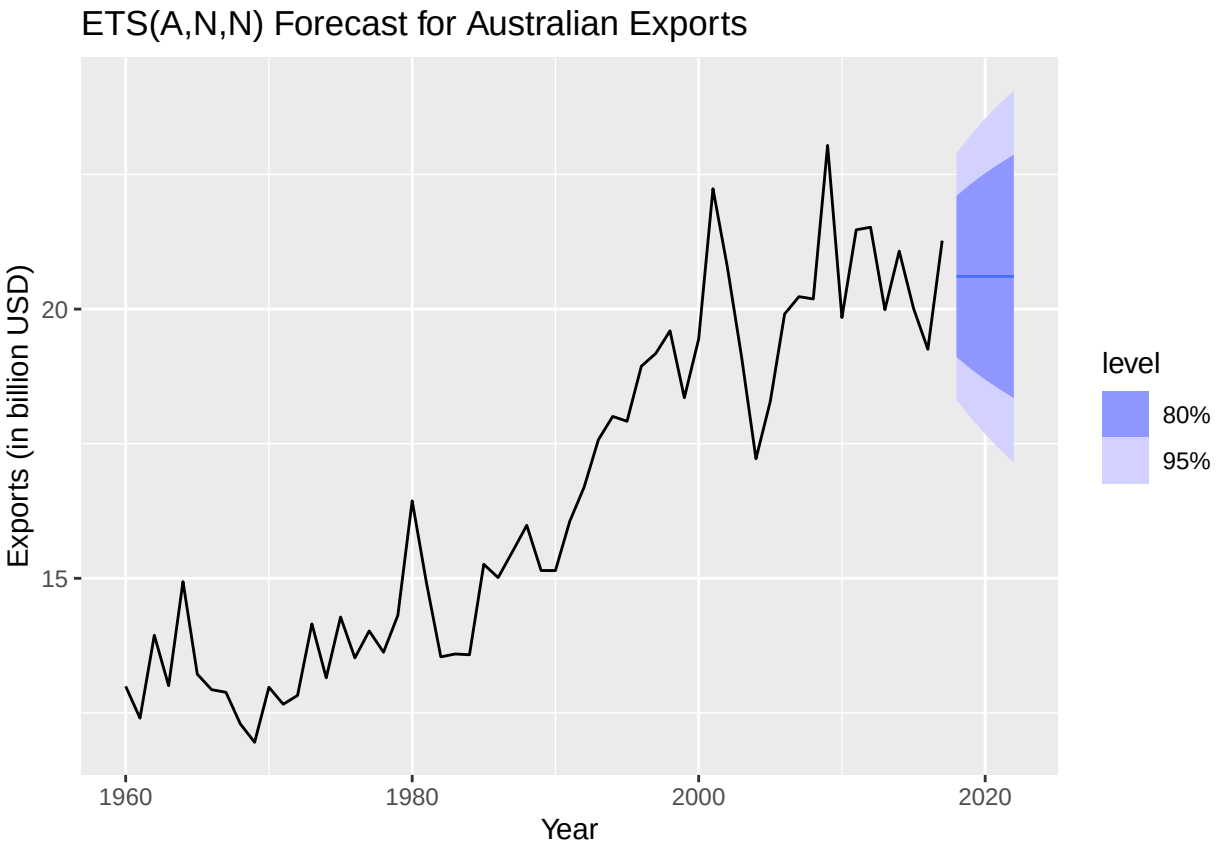
```r
# Select a country (e.g., Australia) from the global_economy dataset
aus_exports <- global_economy %>% filter(Country == "Australia")

# Plot the exports series
aus_exports %>%
  autoplot(Exports) +
  ggtitle("Annual Exports of Australia") +
  ylab("Exports (in billion USD)") +
  xlab("Year")
```

b. Use an ETS(A,N,N) model to forecast the series, and plot the forecasts.

```
# Fit an ETS(A,N,N) model (Exponential Smoothing with additive errors, no trend, and no seasonality)
aus_ets_ann <- aus_exports %>% model(ETS(Exports ~ error("A") + trend("N") + season("N")))

# Forecast the series
aus_ets_ann_fc <- aus_ets_ann %>% forecast(h = "5 years")

head(aus_ets_ann_fc)
```

```
## # A fable: 5 x 5 [1Y]
## # Key:     Country, .model [1]
##   Country   .model                                          Year    Exports .mean
##   <fct>     <chr>                                          <dbl>     <dist> <dbl>
## 1 Australia "ETS(Exports ~ error(\"A\") + trend(\"N\") +~  2018 N(21, 1.4)  20.6
## 2 Australia "ETS(Exports ~ error(\"A\") + trend(\"N\") +~  2019 N(21, 1.8)  20.6
## 3 Australia "ETS(Exports ~ error(\"A\") + trend(\"N\") +~  2020 N(21, 2.2)  20.6
## 4 Australia "ETS(Exports ~ error(\"A\") + trend(\"N\") +~  2021 N(21, 2.7)  20.6
## 5 Australia "ETS(Exports ~ error(\"A\") + trend(\"N\") +~  2022 N(21, 3.1)  20.6
```

```
# Plot the forecasts
aus_ets_ann_fc %>%
  autoplot(aus_exports) +
  ggtitle("ETS(A,N,N) Forecast for Australian Exports") +
  ylab("Exports (in billion USD)") +
  xlab("Year")
```



c. Compute the RMSE values for the training data.

```
aus_ets_ann %>% accuracy() %>% select(RMSE)
```

```
## # A tibble: 1 x 1
##    RMSE
##   <dbl>
## 1  1.15
```

d. Compare the results to those from an ETS(A,A,N) model. (Remember that the trended model is using one more parameter than the simpler model.) Discuss the merits of the two forecasting methods for this data set.

```
# Fit an ETS(A,A,N) model (Additive errors, additive trend, no seasonality)
aus_ets_aan <- aus_exports %>% model(ETS(Exports ~ error("A") + trend("A") + season("N")))

# Forecast using the ETS(A,A,N) model
aus_ets_aan_fc <- aus_ets_aan %>% forecast(h = "5 years")

# Compare RMSE values for both models
accuracy_ets_ann <- aus_ets_ann %>% accuracy()
accuracy_ets_aan <- aus_ets_aan %>% accuracy()
```

```r
# Display RMSE values for comparison
accuracy_ets_ann %>% select(RMSE)
```

```
## # A tibble: 1 x 1
##    RMSE
##   <dbl>
## 1  1.15
```

```r
accuracy_ets_aan %>% select(RMSE)
```

```
## # A tibble: 1 x 1
##    RMSE
##   <dbl>
## 1  1.12
```

e. Compare the forecasts from both methods. Which do you think is best?

```r
head(aus_ets_aan_fc)
```

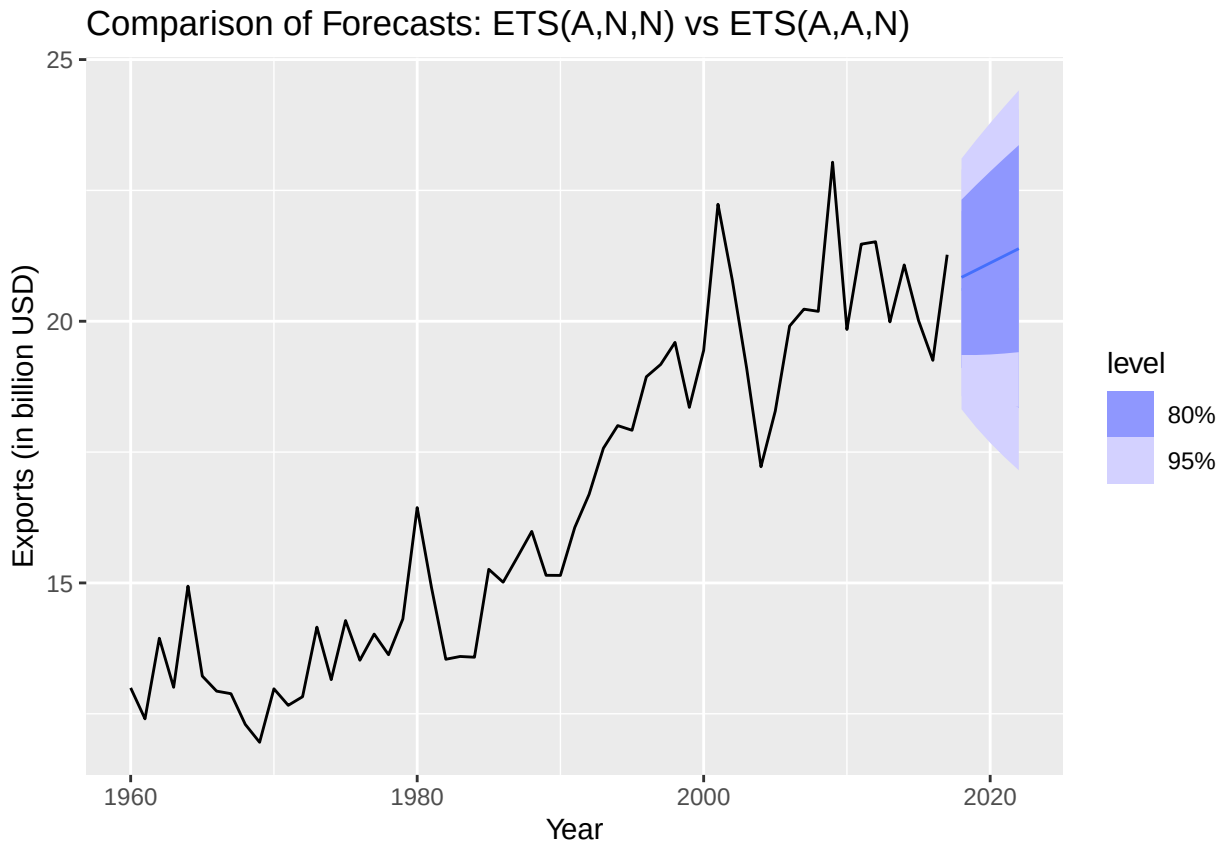
```
## # A fable: 5 x 5 [1Y]
## # Key:     Country, .model [1]
##   Country   .model                                          Year    Exports .mean
##   <fct>     <chr>                                          <dbl>     <dist> <dbl>
## 1 Australia "ETS(Exports ~ error(\"A\") + trend(\"A\") +~  2018 N(21, 1.3)  20.8
## 2 Australia "ETS(Exports ~ error(\"A\") + trend(\"A\") +~  2019 N(21, 1.6)  21.0
## 3 Australia "ETS(Exports ~ error(\"A\") + trend(\"A\") +~  2020 N(21, 1.9)  21.1
## 4 Australia "ETS(Exports ~ error(\"A\") + trend(\"A\") +~  2021 N(21, 2.1)  21.3
## 5 Australia "ETS(Exports ~ error(\"A\") + trend(\"A\") +~  2022 N(21, 2.4)  21.4
```

```r
# Plot forecasts from both models for comparison
aus_ets_ann_fc %>%
  autoplot(aus_exports) +
  autolayer(aus_ets_aan_fc, series = "ETS(A,A,N)", PI = FALSE) +
  ggtitle("Comparison of Forecasts: ETS(A,N,N) vs ETS(A,A,N)") +
  ylab("Exports (in billion USD)") +
  xlab("Year") +
  guides(colour = guide_legend(title = "Model"))
```

```
## Warning in ggdist::geom_lineribbon(without(intvl_mapping, "colour_ramp"), :
## Ignoring unknown parameters: `series` and `PI`
```

```
## Warning in geom_line(mapping = without(mapping, "shape"), data =
## unpack_data(object[single_row[["FALSE"]], : Ignoring unknown parameters:
## `series` and `PI`
```

```
## Scale for fill_ramp is already present.
## Adding another scale for fill_ramp, which will replace the existing scale.
```



f. Calculate a 95% prediction interval for the first forecast for each model, using the RMSE values and assuming normal errors. Compare your intervals with those produced using R.

```
# Get the RMSE values for each model
rmse_ets_ann <- accuracy_ets_ann %>% pull(RMSE)
rmse_ets_aan <- accuracy_ets_aan %>% pull(RMSE)

# Forecasted mean values for the first forecast
mean_ets_ann <- aus_ets_ann_fc$.mean[1]
mean_ets_aan <- aus_ets_aan_fc$.mean[1]

# Calculate the 95% prediction intervals using RMSE
upper_limit_ann <- mean_ets_ann + (1.96 * rmse_ets_ann)
lower_limit_ann <- mean_ets_ann - (1.96 * rmse_ets_ann)

upper_limit_aan <- mean_ets_aan + (1.96 * rmse_ets_aan)
lower_limit_aan <- mean_ets_aan - (1.96 * rmse_ets_aan)

# Compare to R's built-in intervals
aus_ets_ann_fc %>% hilo() %>% select(`95%`)
```

```
## # A tsibble: 5 x 2 [1Y]
##                    `95%`  Year
##                  <hilo> <dbl>
```

```
## 1 [18.31970, 22.89462]95  2018
## 2 [17.97872, 23.23560]95  2019
## 3 [17.67716, 23.53716]95  2020
## 4 [17.40386, 23.81046]95  2021
## 5 [17.15211, 24.06221]95  2022
```

```
aus_ets_aan_fc %>% hilo() %>% select(`95%`)
```

```
## # A tsibble: 5 x 2 [1Y]
##                     `95%`  Year
##                    <hilo> <dbl>
## 1 [18.57028, 23.10700]95  2018
## 2 [18.49679, 23.45482]95  2019
## 3 [18.43976, 23.78617]95  2020
## 4 [18.39583, 24.10441]95  2021
## 5 [18.36266, 24.41191]95  2022
```

```
# Output the intervals
interval_ann <- c(lower_limit_ann, upper_limit_ann)
interval_aan <- c(lower_limit_aan, upper_limit_aan)

# Display intervals
interval_ann
```

```
## [1] 18.35944 22.85488
```

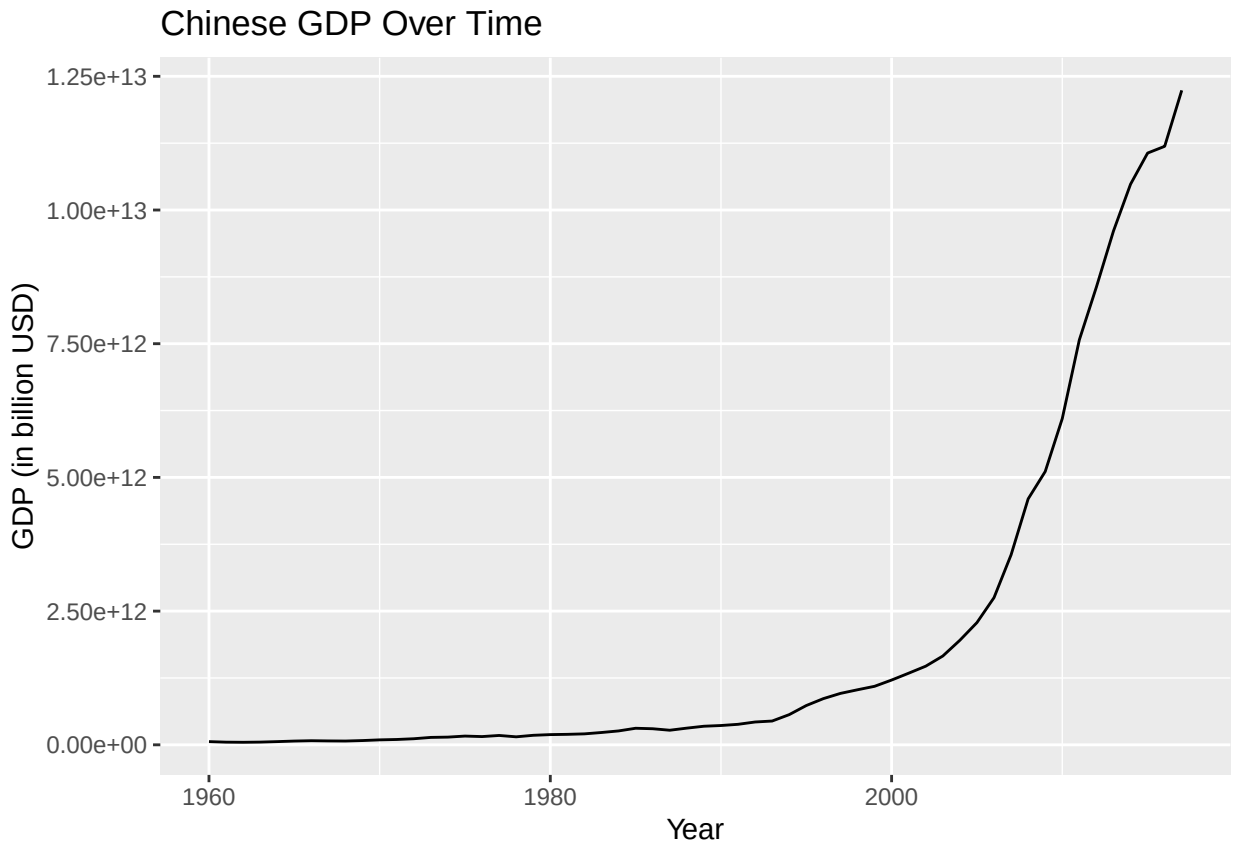
```
interval_aan
```

```
## [1] 18.64986 23.02743
```

8.6 Forecast the Chinese GDP from the global_economy data set using an ETS model. Experiment with the various options in the ETS() function to see how much the forecasts change with damped trend, or with a Box-Cox transformation. Try to develop an intuition of what each is doing to the forecasts.

[Hint: use a relatively large value of h when forecasting, so you can clearly see the differences between the various options when plotting the forecasts.]

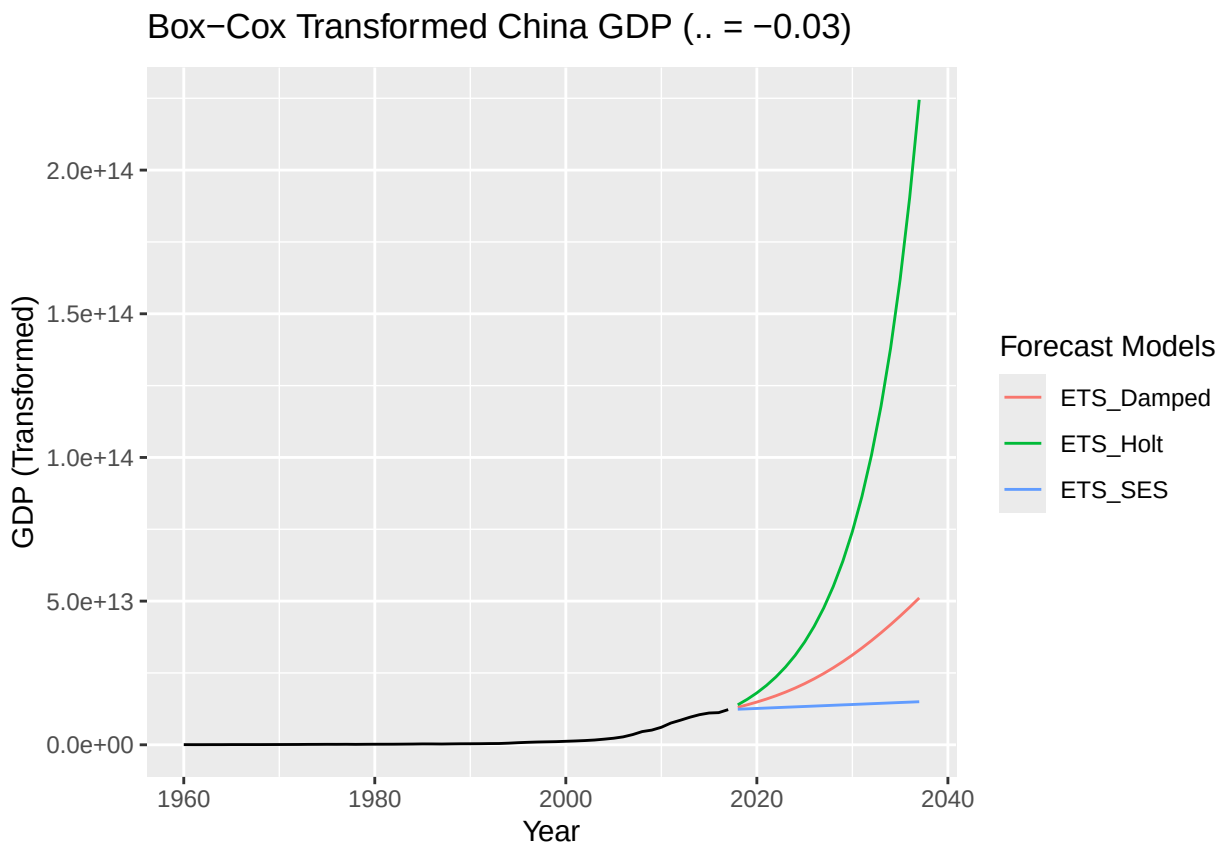
```
# Filter the global_economy dataset for China
china_gdp <- global_economy %>% filter(Country == "China")

# Plot the GDP data for visualization
china_gdp %>%
  autoplot(GDP) +
  ggtitle("Chinese GDP Over Time") +
  ylab("GDP (in billion USD)") +
  xlab("Year")
```




```
# Step 1: Calculate optimal Box-Cox lambda using Guerrero method
lambda_gdp <- china_gdp %>%
  features(GDP, features = guerrero) %>%
  pull(lambda_guerrero)

# Step 2: Fit three ETS models (SES, Holt, Damped) with Box-Cox transformation
gdp_ets_models <- china_gdp %>%
  model(
    ETS_SES = ETS(box_cox(GDP, lambda_gdp) ~ error("A") + trend("N") + season("N")),
    ETS_Holt = ETS(box_cox(GDP, lambda_gdp) ~ error("A") + trend("A") + season("N")),
    ETS_Damped = ETS(box_cox(GDP, lambda_gdp) ~ error("A") + trend("Ad") + season("N"))
  )

# Step 3: Generate forecasts for 20 years using fitted models
gdp_forecasts <- gdp_ets_models %>% forecast(h = 20)

# Step 4: Plot the original series with forecasts
gdp_forecasts %>%
  autoplot(china_gdp, level = NULL) +
  labs(
    y = "GDP (Transformed)",
    title = latex2exp::TeX(paste0("Box-Cox Transformed China GDP ( = ", round(lambda_gdp, 2), ")"))
  ) +
  guides(colour = guide_legend(title = "Forecast Models"))
```

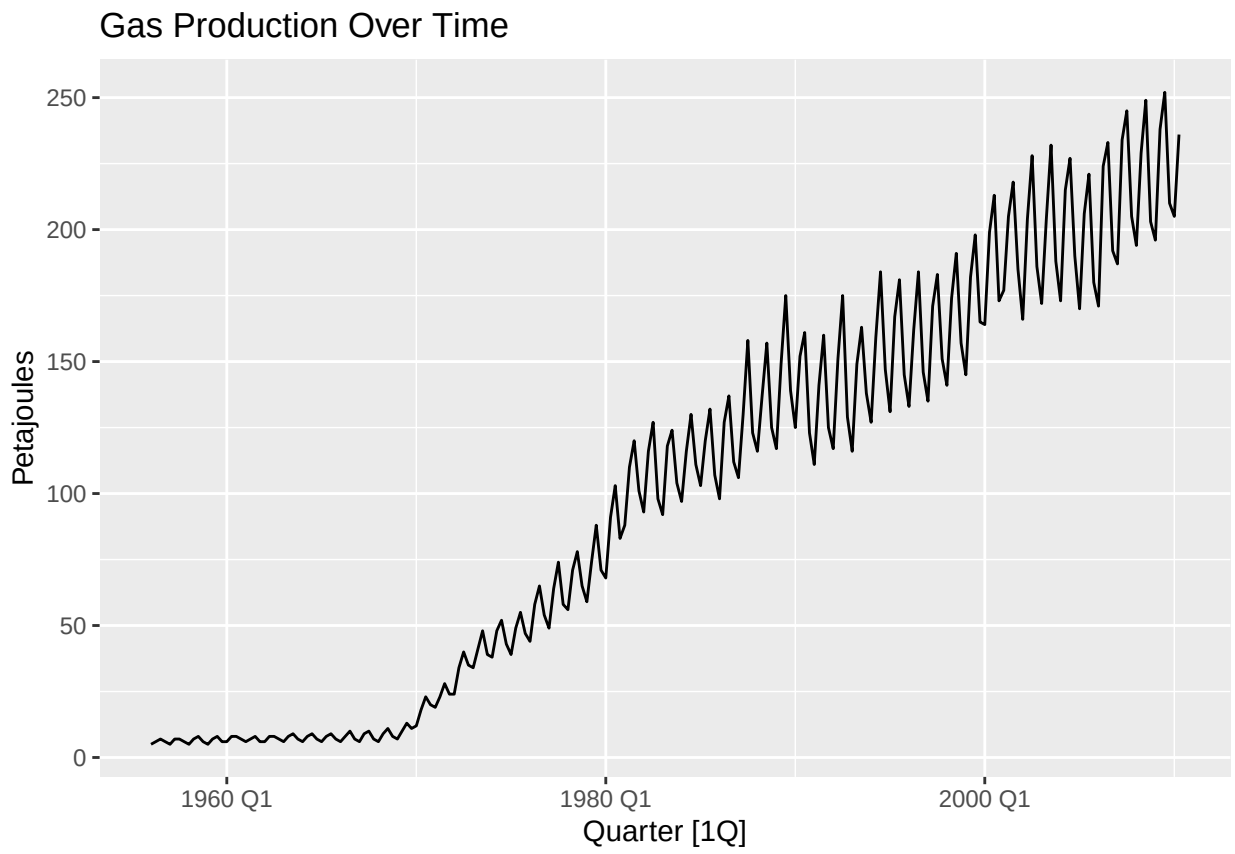
```
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
```

```
## metrics unknown for Unicode character U+03bb

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on 'Box-Cox Transformed China GDP (  = -0.03)' in
## 'mbcsToSbcs': dot substituted for <ce>

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on 'Box-Cox Transformed China GDP (  = -0.03)' in
## 'mbcsToSbcs': dot substituted for <bb>

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## metrics unknown for Unicode character U+03bb

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on 'Box-Cox Transformed China GDP (  = -0.03)' in
## 'mbcsToSbcs': dot substituted for <ce>

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on 'Box-Cox Transformed China GDP (  = -0.03)' in
## 'mbcsToSbcs': dot substituted for <bb>

## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
## font metrics unknown for Unicode character U+03bb

## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on 'Box-Cox Transformed China GDP (  = -0.03)' in
## 'mbcsToSbcs': dot substituted for <ce>

## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on 'Box-Cox Transformed China GDP (  = -0.03)' in
## 'mbcsToSbcs': dot substituted for <bb>

## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
## font metrics unknown for Unicode character U+03bb

## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on 'Box-Cox Transformed China GDP (  = -0.03)' in
## 'mbcsToSbcs': dot substituted for <ce>

## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on 'Box-Cox Transformed China GDP (  = -0.03)' in
## 'mbcsToSbcs': dot substituted for <bb>

## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on 'Box-Cox Transformed China GDP (  = -0.03)' in
## 'mbcsToSbcs': dot substituted for <ce>

## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on 'Box-Cox Transformed China GDP (  = -0.03)' in
## 'mbcsToSbcs': dot substituted for <bb>
```

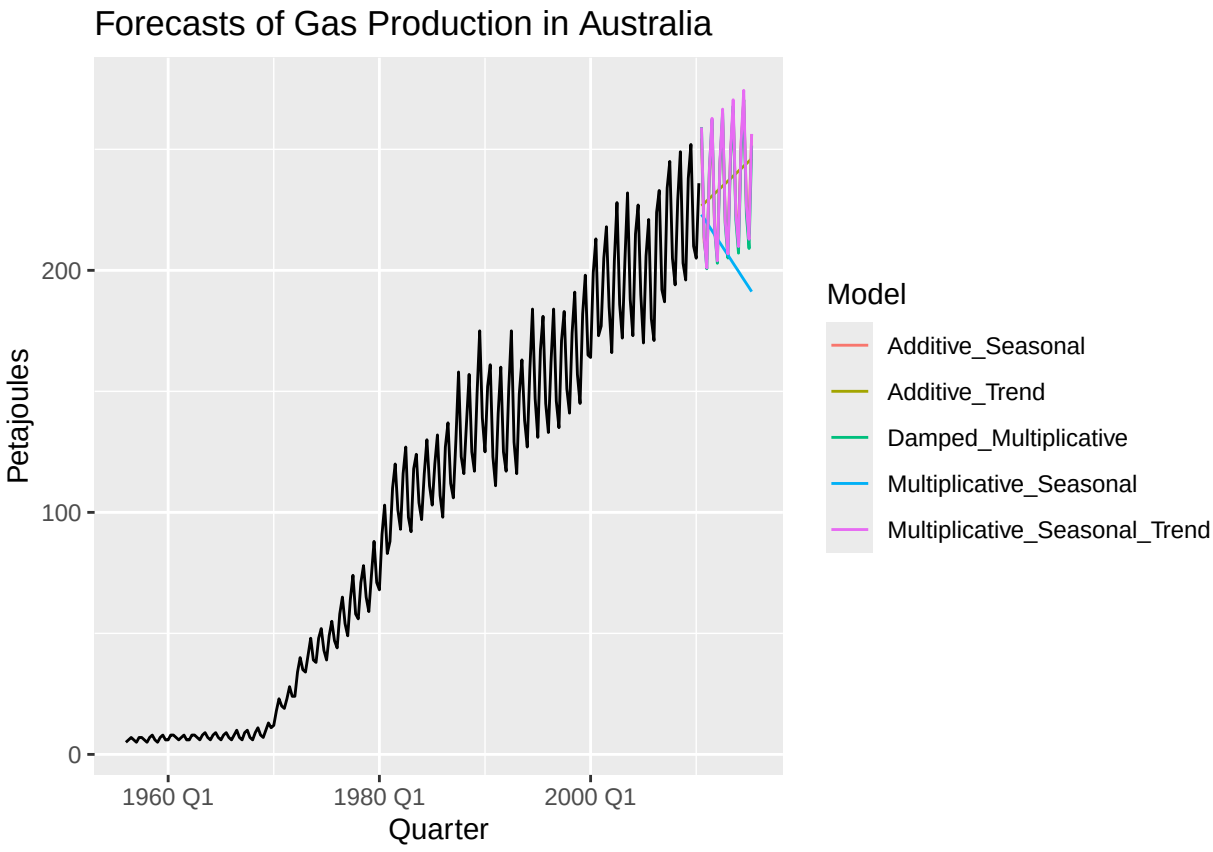


8.7 Find an ETS model for the Gas data from aus_production and forecast the next few years. Why is multiplicative seasonality necessary here? Experiment with making the trend damped. Does it improve the forecasts?

```
# Load and prepare the data for Gas production in petajoules
gas_data <- aus_production %>%
  select(Quarter, Gas) %>%
  mutate(Gas = as.numeric(Gas)) %>%
  tsibble(index = Quarter)

# Plot the Gas production series
gas_data %>%
  autoplot(Gas) +
  labs(title = "Gas Production Over Time", y = "Petajoules")
```




```
# Fit various ETS models
gas_models <- gas_data %>%
  model(
    Additive_Trend = ETS(Gas ~ error("A") + trend("A") + season("N")),
    Multiplicative_Seasonal = ETS(Gas ~ error("M") + trend("A") + season("N")),
    Additive_Seasonal = ETS(Gas ~ error("A") + trend("A") + season("A")),
    Multiplicative_Seasonal_Trend = ETS(Gas ~ error("M") + trend("A") + season("M")),
    Damped_Multiplicative = ETS(Gas ~ error("M") + trend("Ad") + season("M"))
  )

# Forecast for the next 5 years (20 quarters)
gas_forecasts <- gas_models %>%
  forecast(h = 20)

# Plot the forecasts with the original data
gas_forecasts %>%
  autoplot(gas_data, level = NULL) +
  labs(y = "Petajoules", title = "Forecasts of Gas Production in Australia") +
  guides(colour = guide_legend(title = "Model"))
```

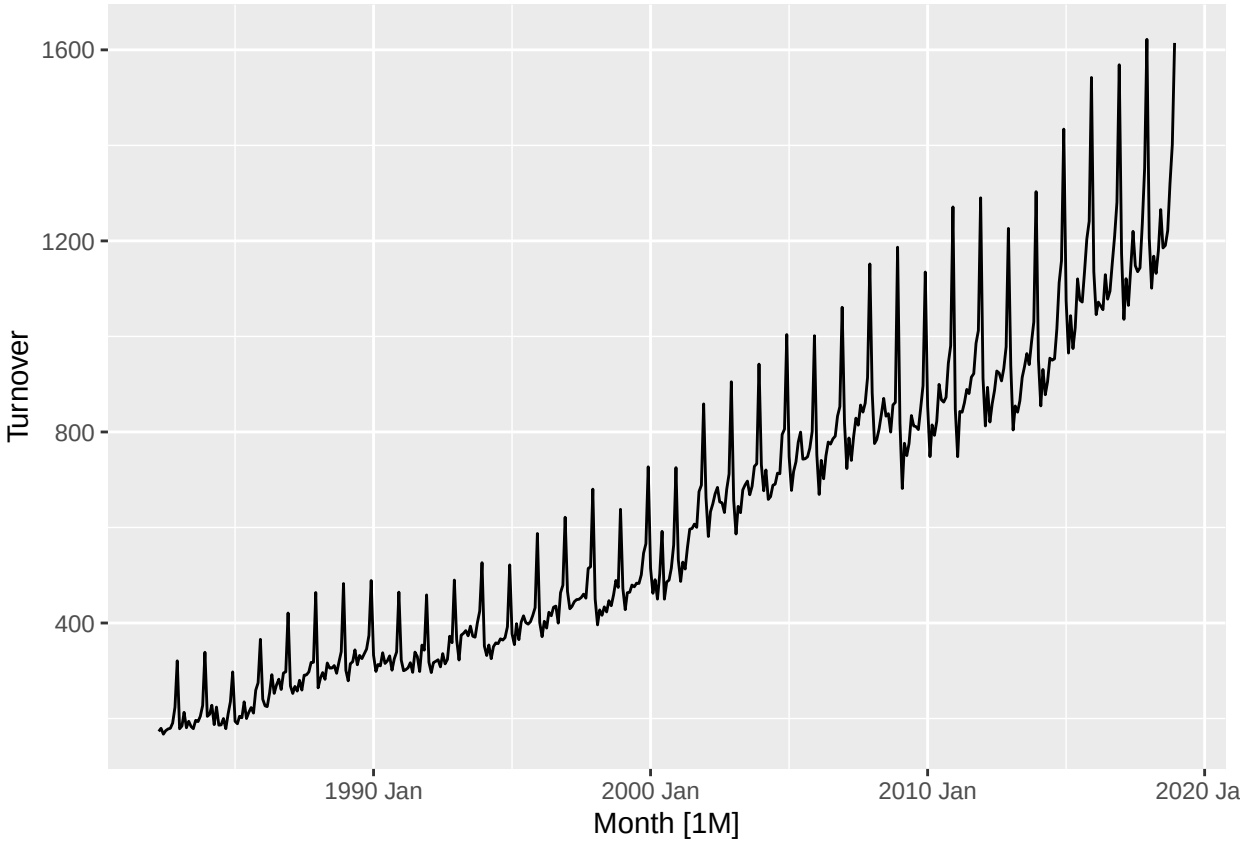


8.8 Recall your retail time series data (from Exercise 7 in Section 2.10).

a. Why is multiplicative seasonality necessary for this series?

```
set.seed(123)

myseries <- aus_retail %>%
  filter(`Series ID` == sample(aus_retail$`Series ID`,1))

autoplot(myseries)
```

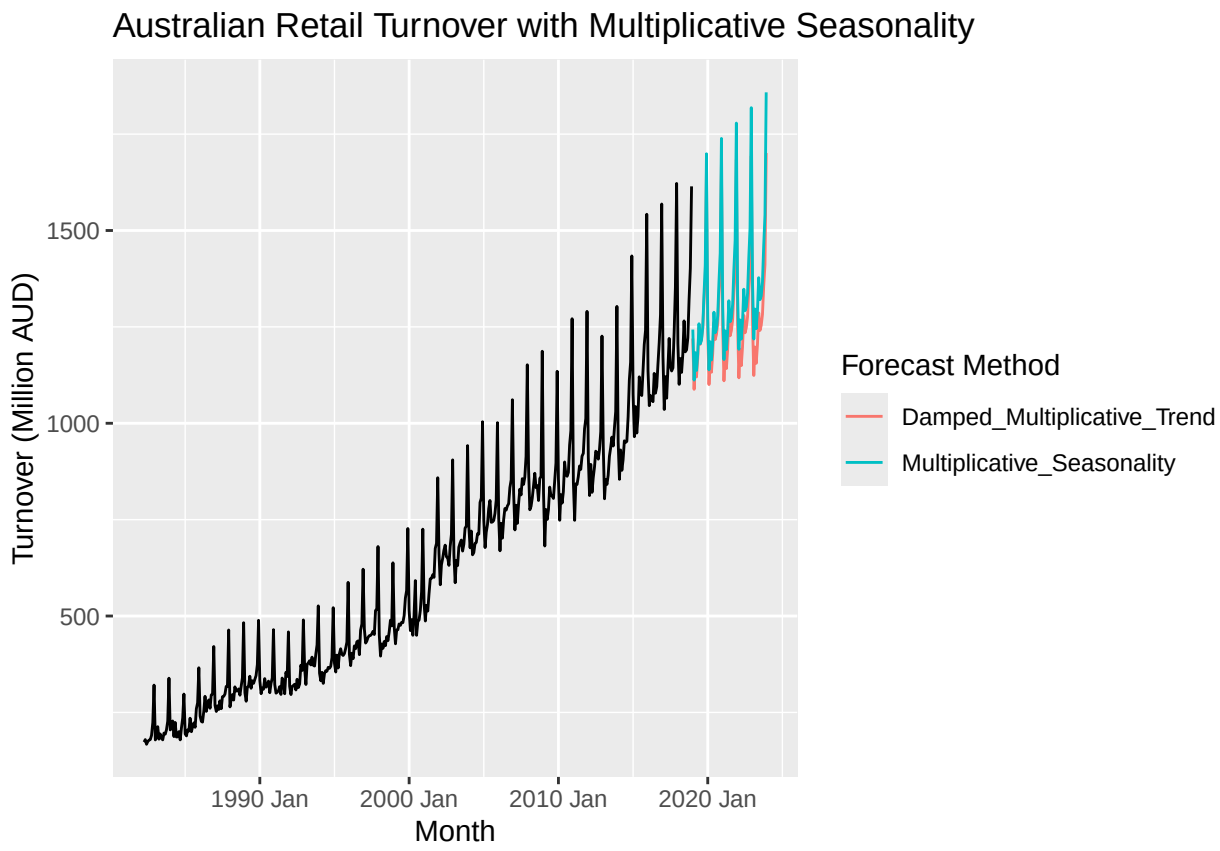
```
## Plot variable not specified, automatically selected `.vars = Turnover`
```



When looking at the retail data, we observe that both the overall trend and the size of the seasonal fluctuations are increasing over time. This suggests that the seasonal variation grows in proportion to the level of the time series. In other words, the higher the retail turnover, the larger the seasonal swings. This makes a multiplicative seasonal model more appropriate, as it allows the seasonal pattern to scale with the changing level of the series, rather than staying constant as it would in an additive model.

b. Apply Holt-Winters' multiplicative method to the data. Experiment with making the trend damped.

```r
# Apply Holt-Winters' multiplicative method with and without damped trend
fit_multiplicative <- myseries %>%
  model(
    Multiplicative_Seasonality = ETS(Turnover ~ error("M") + trend("A") + season("M")),
    Damped_Multiplicative_Trend = ETS(Turnover ~ error("M") + trend("Ad") + season("M"))
  )

# Forecast for the next 5 years (monthly frequency)
forecast_multiplicative <- fit_multiplicative %>% forecast(h = "5 years")

# Plot the forecasts along with the original data
forecast_multiplicative %>%
  autoplot(myseries, level = NULL) +
  labs(y = "Turnover (Million AUD)", title = "Australian Retail Turnover with Multiplicative Seasonality
  guides(colour = guide_legend(title = "Forecast Method"))
```



c. Compare the RMSE of the one-step forecasts from the two methods. Which do you prefer?

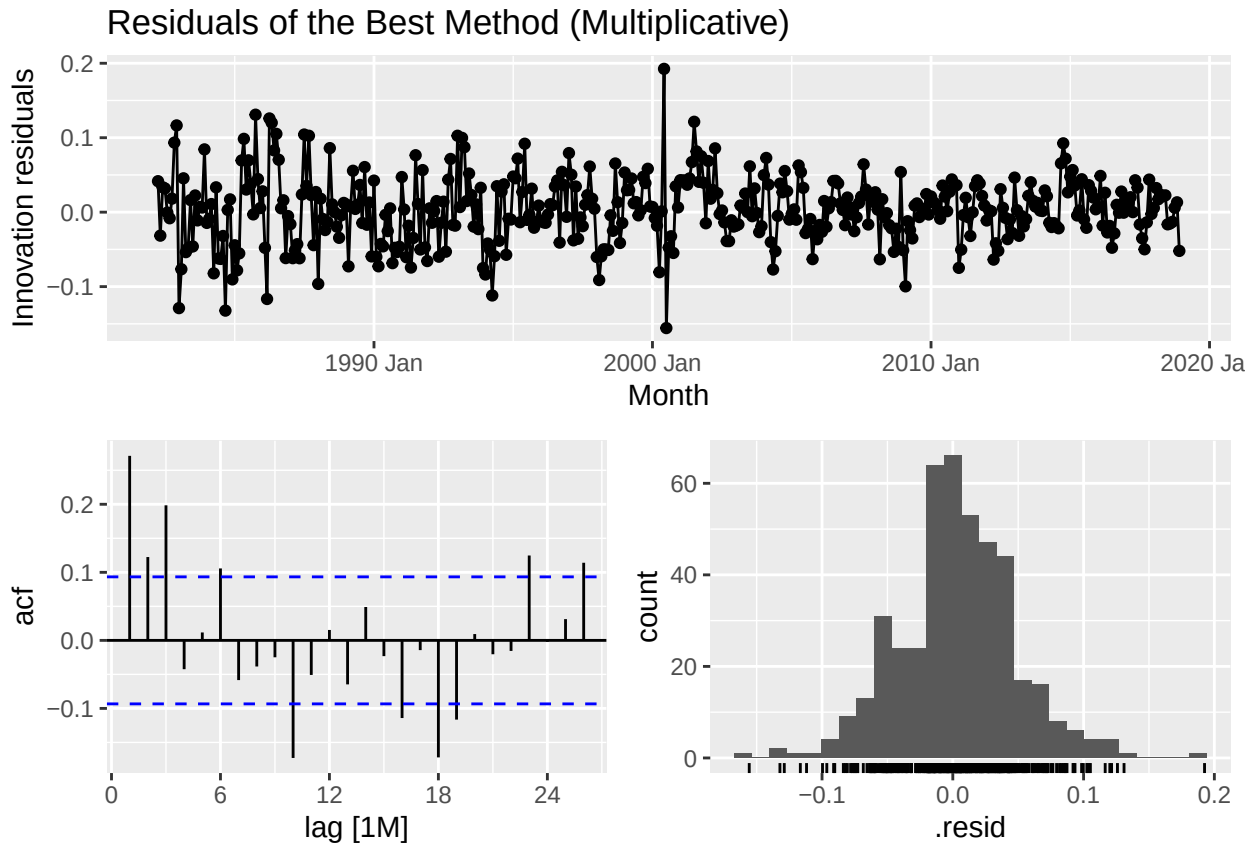
```
# Calculate accuracy for one-step-ahead forecasts and extract RMSE
accuracy_comparison <- fit_multiplicative %>%
  accuracy() %>%
  select(.model, RMSE)

# Display RMSE comparison
print(accuracy_comparison)
```

```
## # A tibble: 2 x 2
##   .model                        RMSE
##   <chr>                        <dbl>
## 1 Multiplicative_Seasonality   24.3
## 2 Damped_Multiplicative_Trend  23.7
```

d. Check that the residuals from the best method look like white noise.

```
# Fit the model with the best method (multiplicative without damped trend)
best_fit <- myseries %>%
  model(best_method = ETS(Turnover ~ error("M") + trend("A") + season("M")))
```

```
# Check if residuals resemble white noise
best_fit %>%
  gg_tsresiduals() +
  ggtitle("Residuals of the Best Method (Multiplicative)")
```




```
# Box-Pierce test
box_pierce_result <- myseries %>%
  model(multiplicative = ETS(Turnover ~ error("M") + trend("A") + season("M"))) %>%
  augment() %>%
  features(.innov, box_pierce, lag = 24, dof = 0)

print(box_pierce_result)
```

```
## # A tibble: 1 x 5
##   State    Industry                  .model         bp_stat bp_pvalue
##   <chr>    <chr>                     <chr>            <dbl>     <dbl>
## 1 Victoria Household goods retailing multiplicative    114.  1.03e-13
```

```
# Ljung-Box test
ljung_box_result <- myseries %>%
  model(multiplicative = ETS(Turnover ~ error("M") + trend("A") + season("M"))) %>%
  augment() %>%
  features(.innov, ljung_box, lag = 24, dof = 0)

print(ljung_box_result)
```

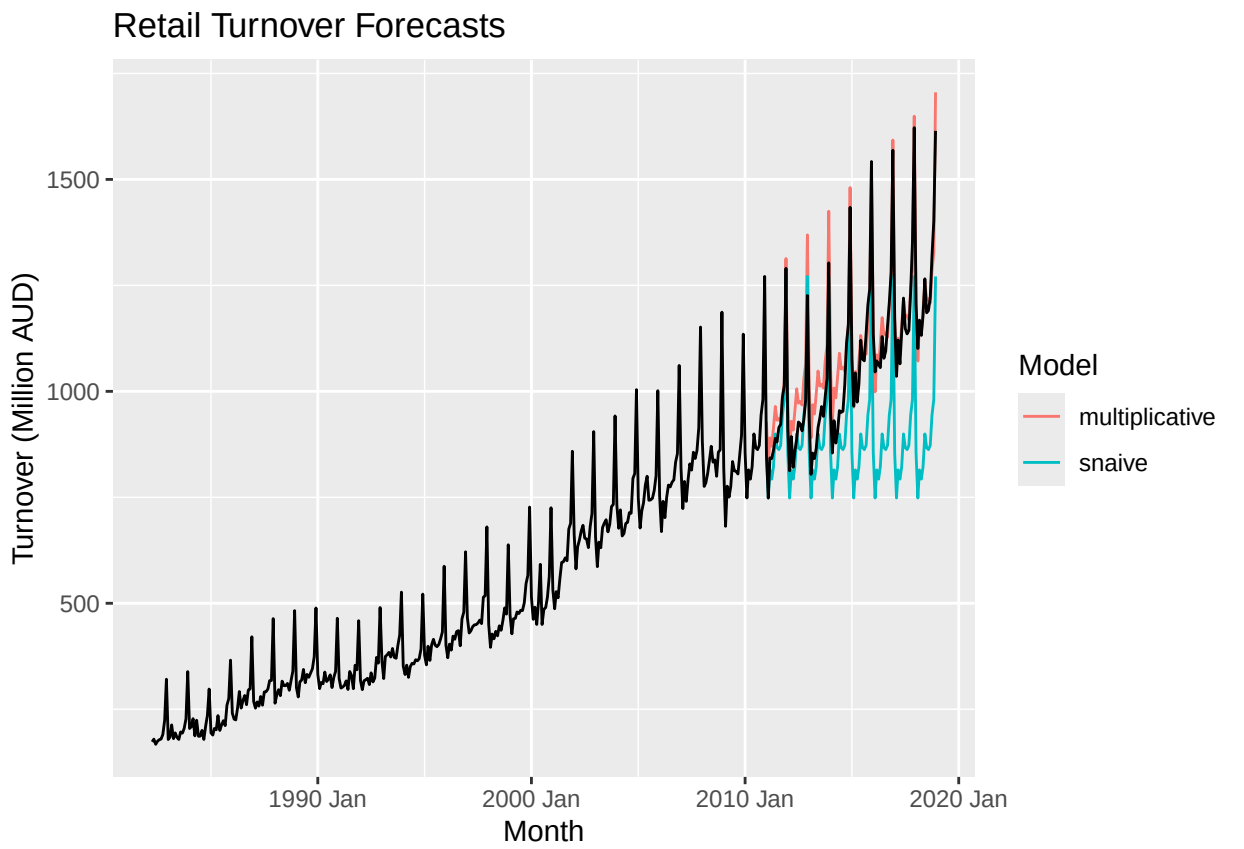
```
## # A tibble: 1 x 5
##   State    Industry                  .model         lb_stat lb_pvalue
##   <chr>    <chr>                     <chr>            <dbl>     <dbl>
## 1 Victoria Household goods retailing multiplicative    117.  3.36e-14
```

e. Now find the test set RMSE, while training the model to the end of 2010. Can you beat the seasonal naïve approach from Exercise 7 in Section 5.11?

```
# Split data: Training set (up to 2010)
myseries_train <- myseries %>%
  filter(year(Month) < 2011)

# Fit models: ETS multiplicative and Seasonal Naive
fit_train <- myseries_train %>%
  model(
    multiplicative = ETS(Turnover ~ error("M") + trend("A") + season("M")),
    snaive = SNAIVE(Turnover)
  )

# Forecasts: Generate forecasts for the test period (2011 onwards)
fc <- fit_train %>%
  forecast(new_data = anti_join(myseries, myseries_train))
```

```
## Joining with `by = join_by(State, Industry, `Series ID`, Month, Turnover)`
```

```
# Plot the forecasts along with the original data
fc %>%
  autoplot(myseries, level = NULL) +
  labs(y = "Turnover (Million AUD)", title = "Retail Turnover Forecasts") +
  guides(colour = guide_legend(title = "Model"))
```




```
# Calculate accuracy on the training set
train_accuracy <- accuracy(fit_train) %>%
  select(.type, .model, RMSE)

# Calculate accuracy on the test set (2011 onwards)
test_accuracy <- fc %>%
  accuracy(myseries) %>%
  select(.type, .model, RMSE)

# Display training and test set RMSE
train_accuracy
```

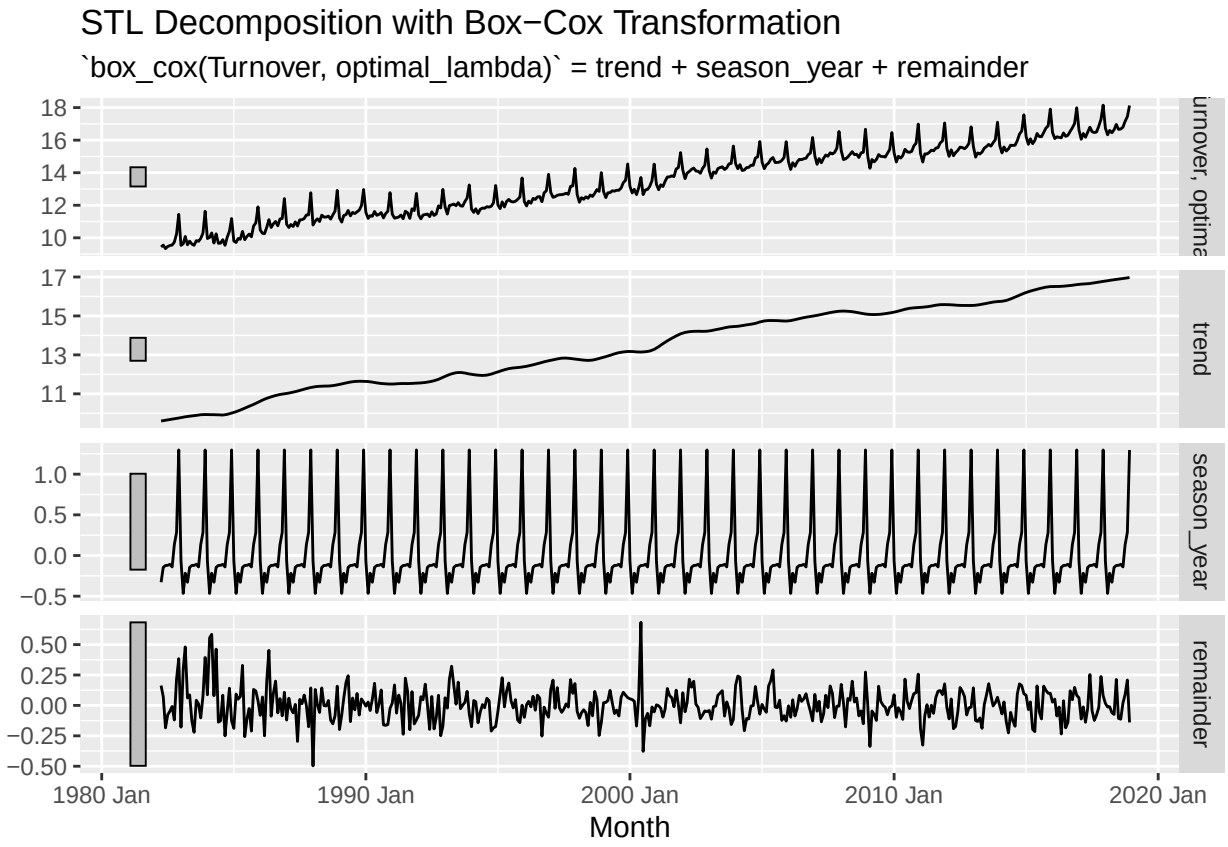
```
## # A tibble: 2 x 3
##   .type    .model          RMSE
##   <chr>    <chr>          <dbl>
## 1 Training multiplicative  20.9
## 2 Training snaive          45.6
```

```
test_accuracy
```

```
## # A tibble: 2 x 3
##   .type .model          RMSE
##   <chr> <chr>          <dbl>
## 1 Test  multiplicative  62.2
## 2 Test  snaive         213.
```

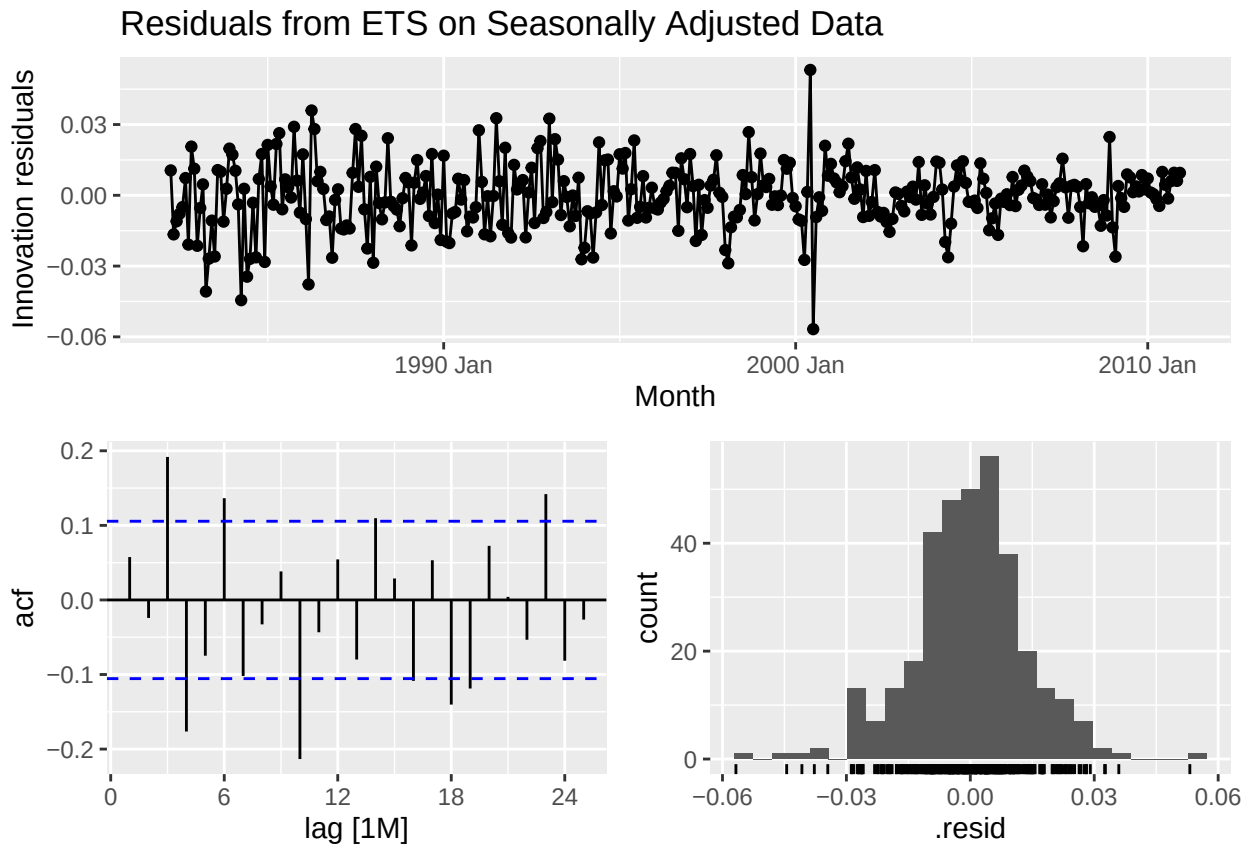
8.9 For the same retail data, try an STL decomposition applied to the Box-Cox transformed series, followed by ETS on the seasonally adjusted data. How does that compare with your best previous forecasts on the test set?

```
# Calculate the optimal lambda for Box-Cox transformation using the Guerrero method
optimal_lambda <- myseries %>%
  features(Turnover, features = guerrero) %>%
  pull(lambda_guerrero)

# Perform STL decomposition on the Box-Cox transformed series
myseries %>%
  model(STL_decomp = STL(box_cox(Turnover, optimal_lambda) ~ season(window = "periodic"), robust = TRUE)
  components() %>%
  autoplot() +
  ggtitle("STL Decomposition with Box-Cox Transformation")
```




```
# Extract the components from the STL decomposition
decomposed_components <- myseries %>%
  model(STL_box = STL(box_cox(Turnover, optimal_lambda) ~ season(window = "periodic"), robust = TRUE)) %
  components()

# Create a new column for seasonally adjusted turnover
myseries <- myseries %>%
  mutate(Turnover_adjusted = decomposed_components$season_adjust)
```

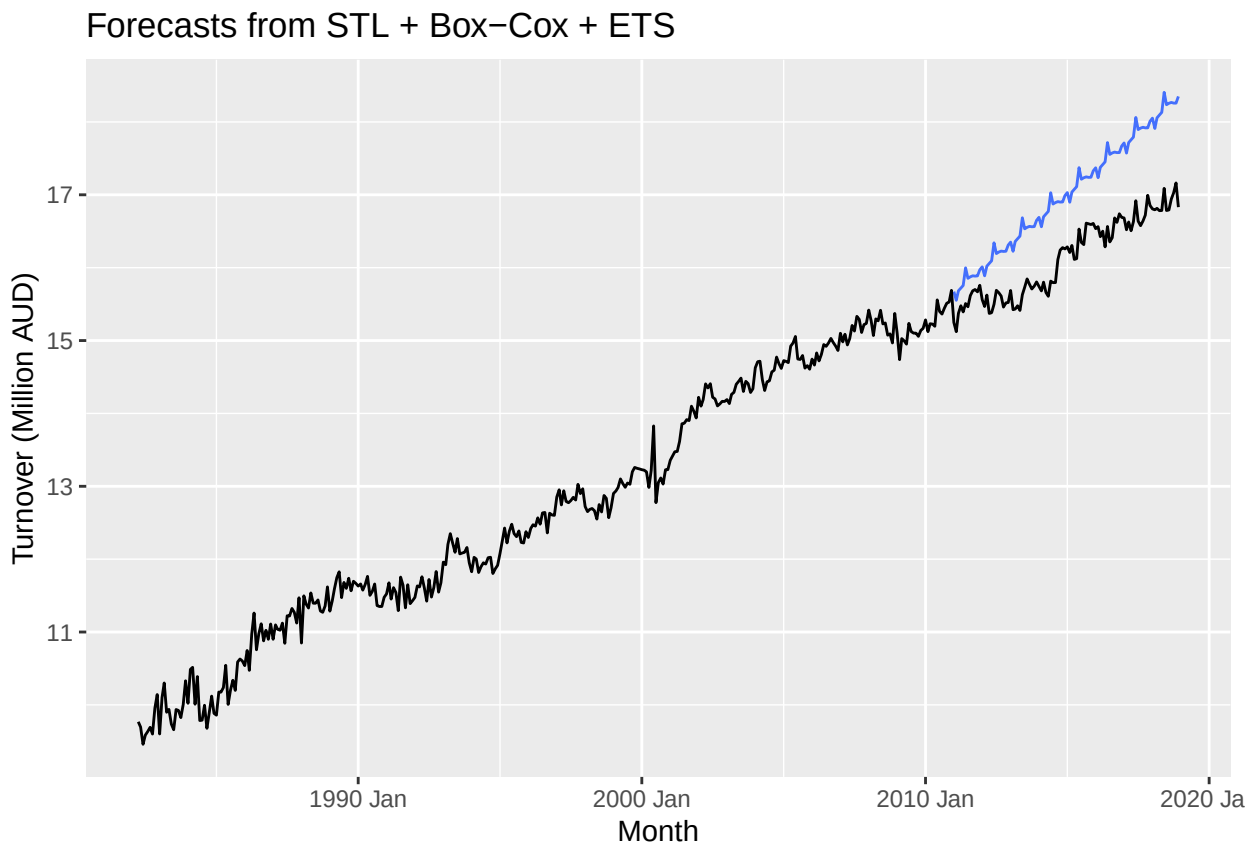
```r
# Filter the training data to include records up to 2010
training_data <- myseries %>%
  filter(year(Month) < 2011)

# Fit the ETS model to the seasonally adjusted data
ets_model <- training_data %>%
  model(ETS_model = ETS(Turnover_adjusted ~ error("M") + trend("A") + season("M")))

# Visualize the residuals from the fitted ETS model
ets_model %>%
  gg_tsresiduals() +
  ggtitle("Residuals from ETS on Seasonally Adjusted Data")
```




```r
# Generate forecasts for the test data (2011 and later)
forecasted_values <- ets_model %>%
  forecast(new_data = anti_join(myseries, training_data))
```

```
## Joining with `by = join_by(State, Industry, `Series ID`, Month, Turnover,
## Turnover_adjusted)`
```

```r
# Plot the forecasts alongside the original data
forecasted_values %>%
  autoplot(myseries, level = NULL) +
  labs(y = "Turnover (Million AUD)", title = "Forecasts from STL + Box-Cox + ETS")
```




```
# Evaluate RMSE on the training dataset
training_accuracy <- ets_model %>%
  accuracy() %>%
  select(.model, .type, RMSE)

# Calculate RMSE for the test dataset (2011 onward)
test_accuracy <- forecasted_values %>%
  accuracy(myseries) %>%
  select(.model, .type, RMSE)

# Display the accuracy results
training_accuracy
```

```
## # A tibble: 1 x 3
##   .model    .type     RMSE
##   <chr>     <chr>    <dbl>
## 1 ETS_model Training 0.165
```

```
test_accuracy
```

```
## # A tibble: 1 x 3
##   .model    .type  RMSE
##   <chr>     <chr> <dbl>
## 1 ETS_model Test  0.924
```