# Retail Project

### 2023-05-18

```
library(fpp3)
```

```
## ── Attaching packages ─────────────────────────────────────── fpp3 0.5 ──
```

```
## ✔ tibble      3.2.1     ✔ tsibble     1.1.3
## ✔ dplyr       1.1.2     ✔ tsibbledata 0.4.1
## ✔ tidyr       1.3.0     ✔ feasts      0.3.1
## ✔ lubridate   1.9.2     ✔ fable       0.3.3
## ✔ ggplot2     3.4.2     ✔ fabletools  0.3.2
```

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

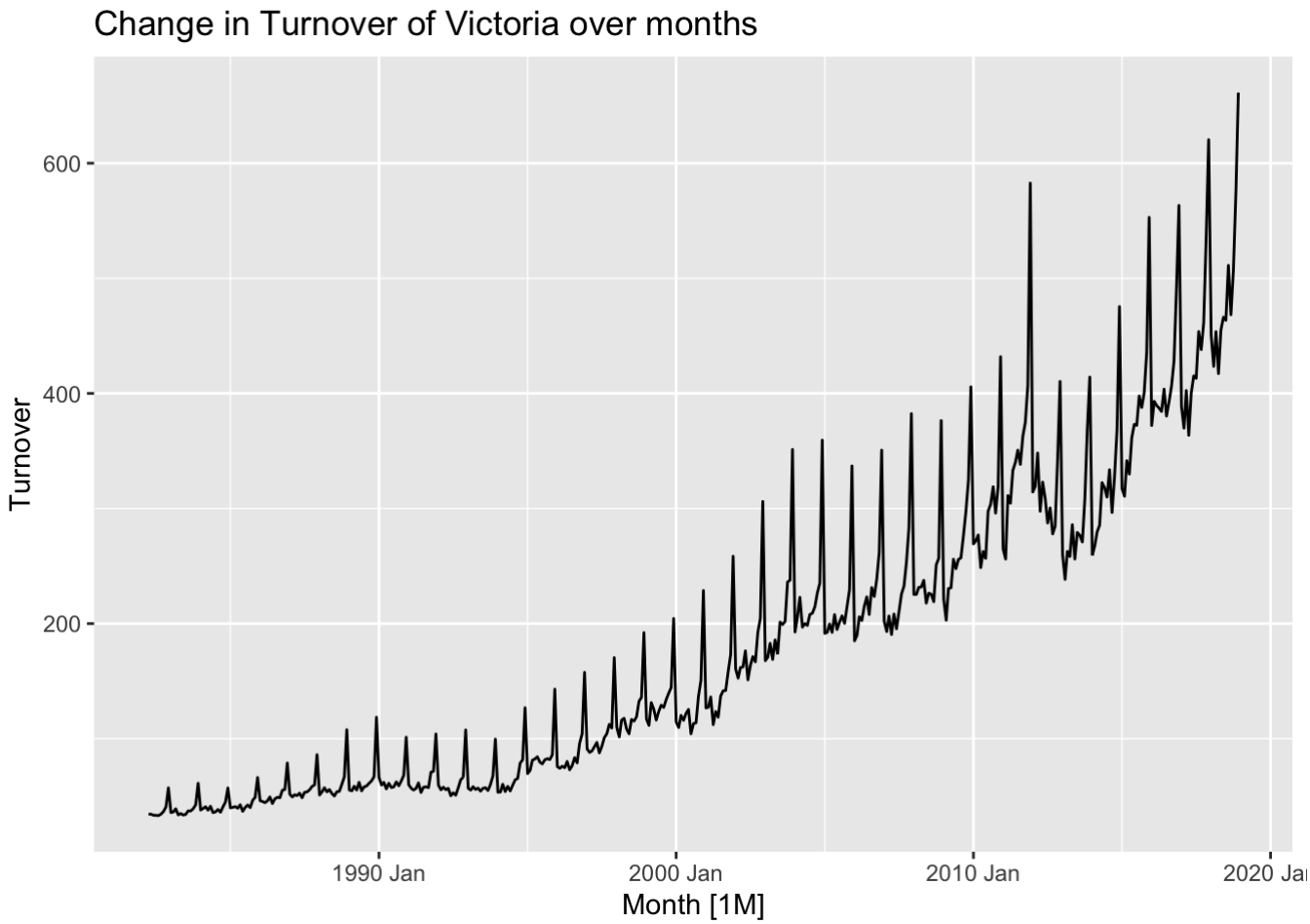
```
library(fable)
library(tsibble)
```

# Load the data

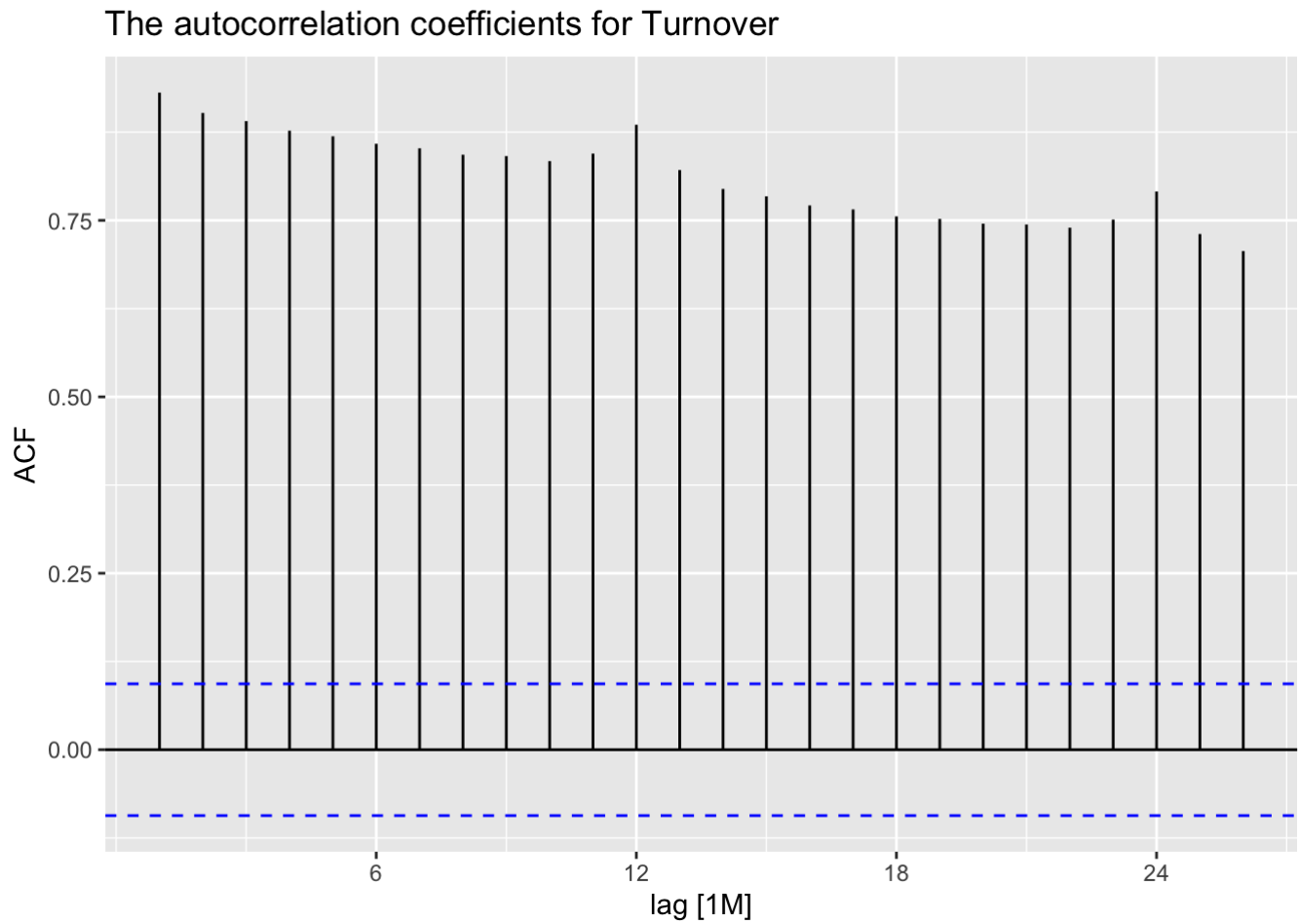
```
# Use your student ID as the seed
set.seed(30806933)
myseries <- aus_retail |>
  # Remove discontinued series
  filter(!(`Series ID` %in% c("A3349561R","A3349883F","A3349499L","A3349902A",
                        "A3349588R","A3349763L","A3349372C","A3349450X",
                        "A3349679W","A3349378T","A3349767W","A3349451A"))) |>
  # Select a series at random
  filter(`Series ID` == sample(`Series ID`,1))
```

# Question 1: A discussion of the statistical features of the original data. [4 marks]

```
myseries |>
  autoplot(Turnover) +
  labs(y = "Turnover",
       title = "Change in Turnover of Victoria over months")
```




```
myseries |>
  ACF(Turnover) |>
  autoplot() +
  labs(y = "ACF",
       title = "The autocorrelation coefficients for Turnover")
```




Explanation:

As we can see from the time plot, there is an upward trend in Turnover over months and years. It also appears to be seasonality, and the variance along the plot is not constant. We can also look at the ACF plot of this time series, that the ACFs slowly decays , and r1 is the highest (close to 1). All of these features show that this time serie is not stationary, and we need to perform transformations and differencing in order to make forecasts based on this data.

# Question 2: Explanation of transformations and differencing used. You should use a unit-root test as part of the discussion. [5 marks]

Transformations help to stabilize the variance and differencing helps to stabilize the mean.

```
# Perform KPSS test
myseries |>
  features(Turnover, unitroot_kpss)
```

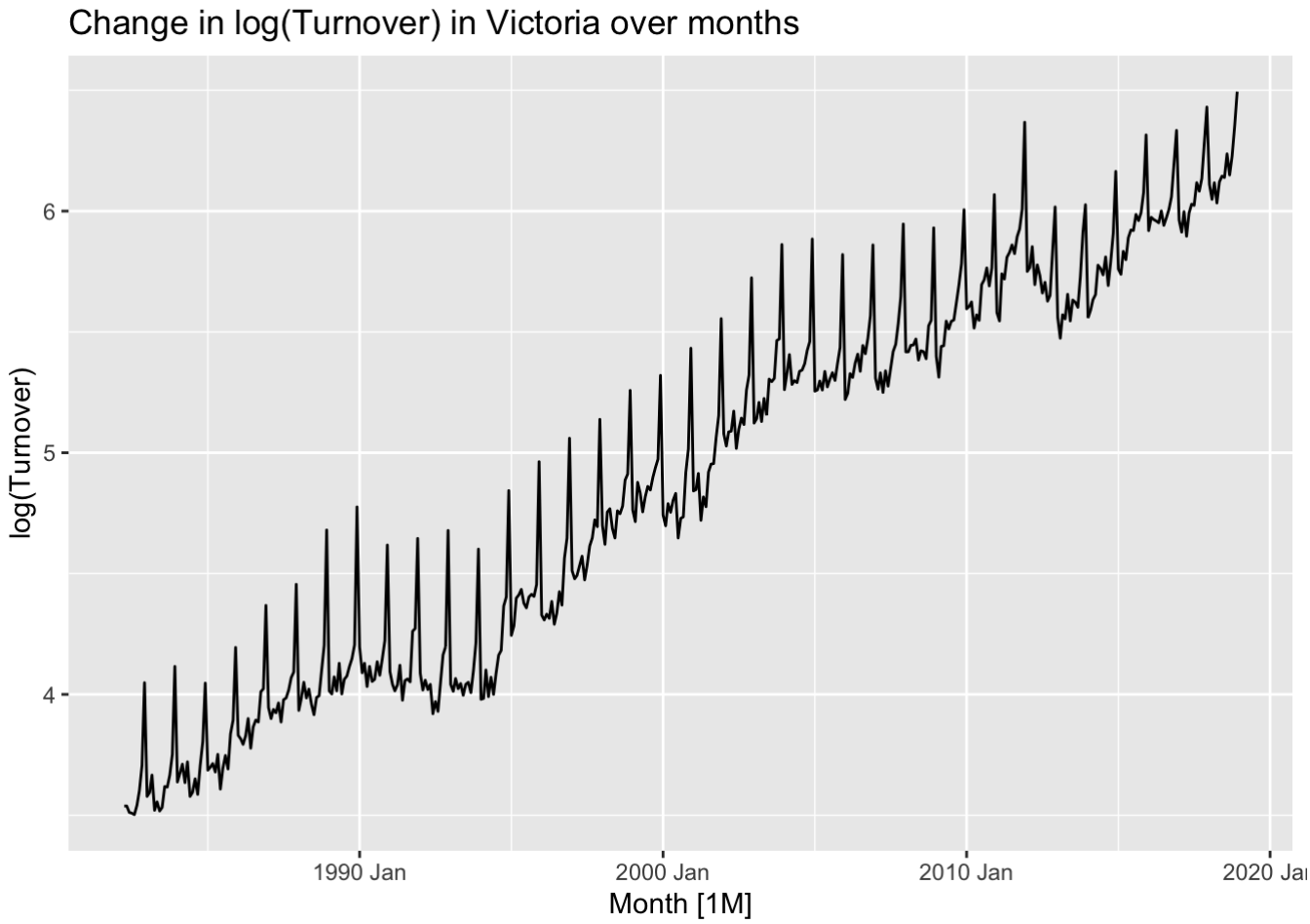
```
## # A tibble: 1 × 4
##   State    Industry               kpss_stat kpss_pvalue
##   <chr>    <chr>                      <dbl>       <dbl>
## 1 Victoria Other retailing n.e.c.      6.85        0.01
```

As we use KPSS test, we have: Null hypothesis: the data is stationary and non-seasonal. In this case, the kpss_pvalue = 0.01, which is less than p-value = 0.05. Therefore, we reject the null hypothesis that the data is stationary and non-seasonal. Since the data is not stationary and non-seasonal, we need to perform

transformation and differencing.

First, we take the log transformation to make the variance constant (one of conditions to make data stationary).

```
# Transformation of the data to stabilize the constant
myseries |>
  autoplot(log(Turnover)) +
  labs(y = "log(Turnover)",
       title = "Change in log(Turnover) in Victoria over months")
```




Next, we will consider number of differencing to make data non-seasonal. We can use unitroot_nsdiffs and unitroot_ndiffs to identify how many differences we should take.

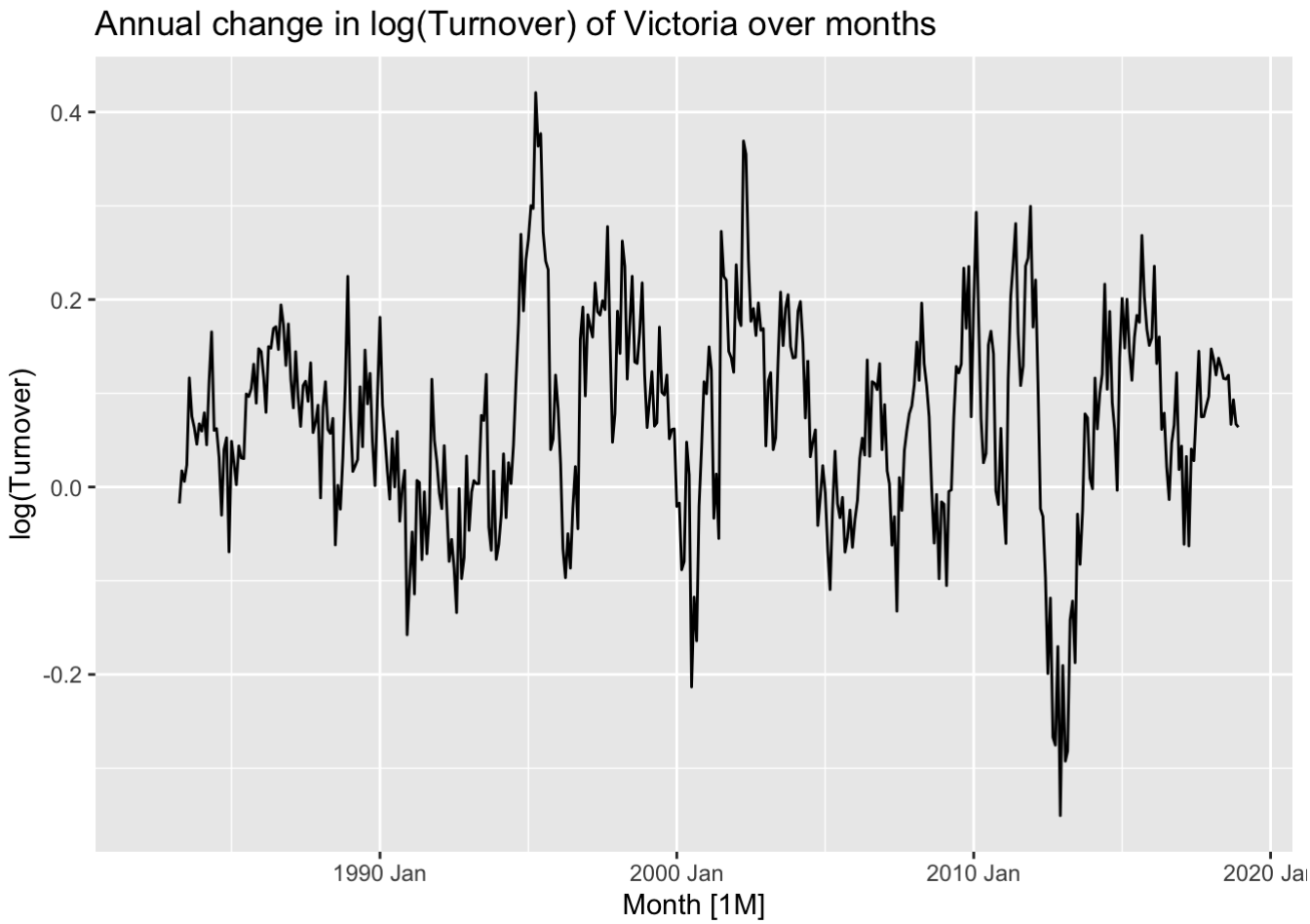
```
# KPSS test to check now many seasonal differencing we need to use
myseries |>
  features(log(Turnover), unitroot_nsdiffs)
```

```
## # A tibble: 1 × 3
##   State    Industry               nsdiffs
##   <chr>    <chr>                    <int>
## 1 Victoria Other retailing n.e.c.       1
```

Since the graph have a strong seasonal pattern, we will take the seasonal differencing be done first, and then consider whether to take the first differencing or not. This recommends that we need to take 1 seasonal differencing to make the data stationary.

```
# After seasonal differencing
myseries |>
  autoplot(log(Turnover) |>
  difference(12)) +
  labs(y = "log(Turnover)",
       title ="Annual change in log(Turnover) of Victoria over months")
```

```
## Warning: Removed 12 rows containing missing values (`geom_line()`).
```




It is clearly shown in the graph that the plot is not stationary, that the variance is not constant along time. Therefore, we could use unitroos_ndiffs to check whether we need to take the first differencing.

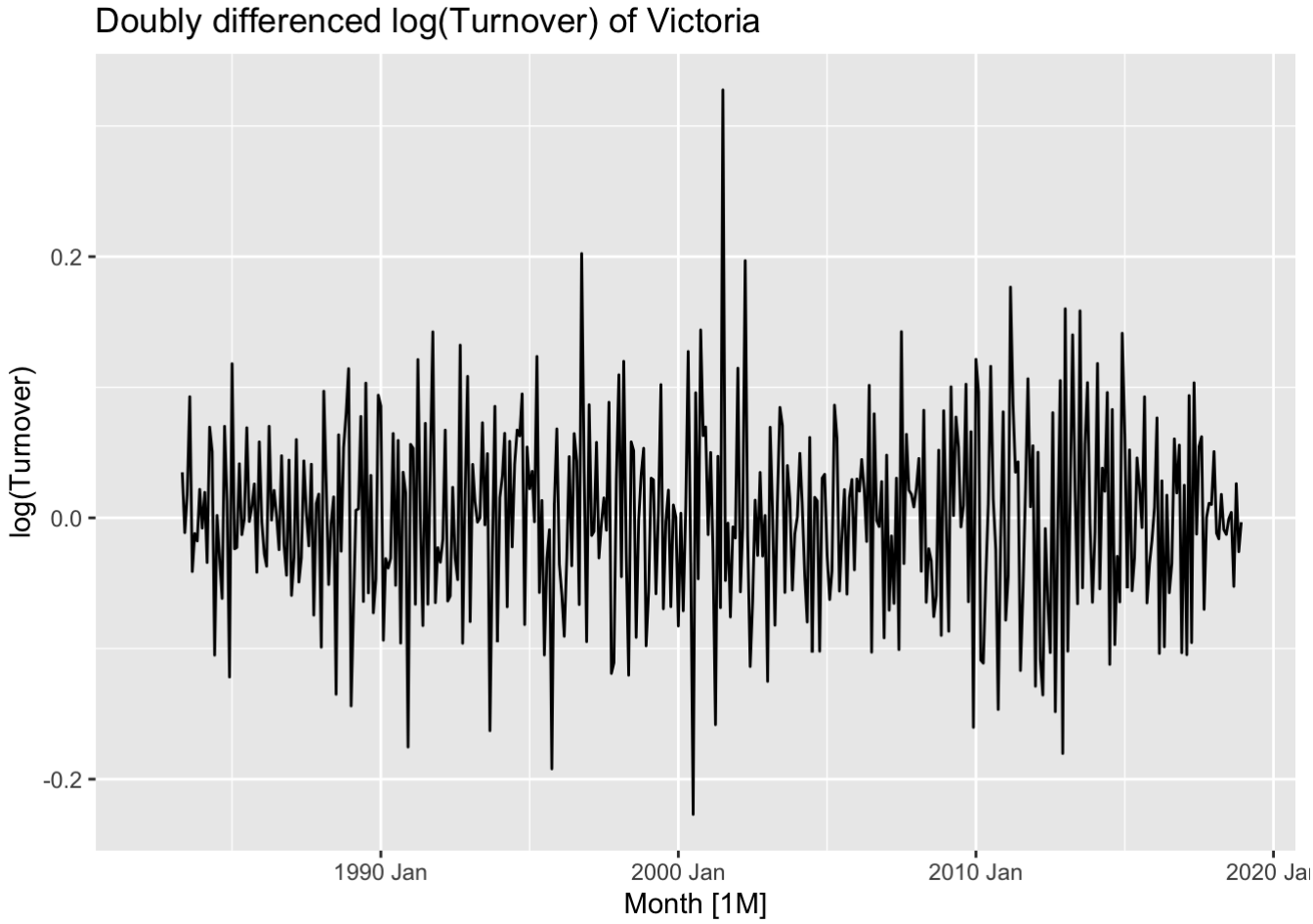
```
# KPSS test to check how many normal differencing we need
myseries |>
  features(log(Turnover), unitroot_ndiffs)
```

```
## # A tibble: 1 × 3
##   State    Industry               ndiffs
##   <chr>    <chr>                   <int>
## 1 Victoria Other retailing n.e.c.      1
```

Since the result of ndiffs is 1, we should take the first differencing to make data stationary.

```
# Double differenced graph
myseries |>
  autoplot(log(Turnover) |>
  difference(12) |>
    difference(1)) +
  labs(y = "log(Turnover)",
       title = "Doubly differenced log(Turnover) of Victoria")
```

```
## Warning: Removed 13 rows containing missing values (`geom_line()`).
```




# Question 3: A description of the methodology used to create a short-list of appropriate ARIMA models and ETS models. Include discussion of AIC values as well as results from applying the models to a test-set consisting of the last 24 months of the data provided. [6 marks]

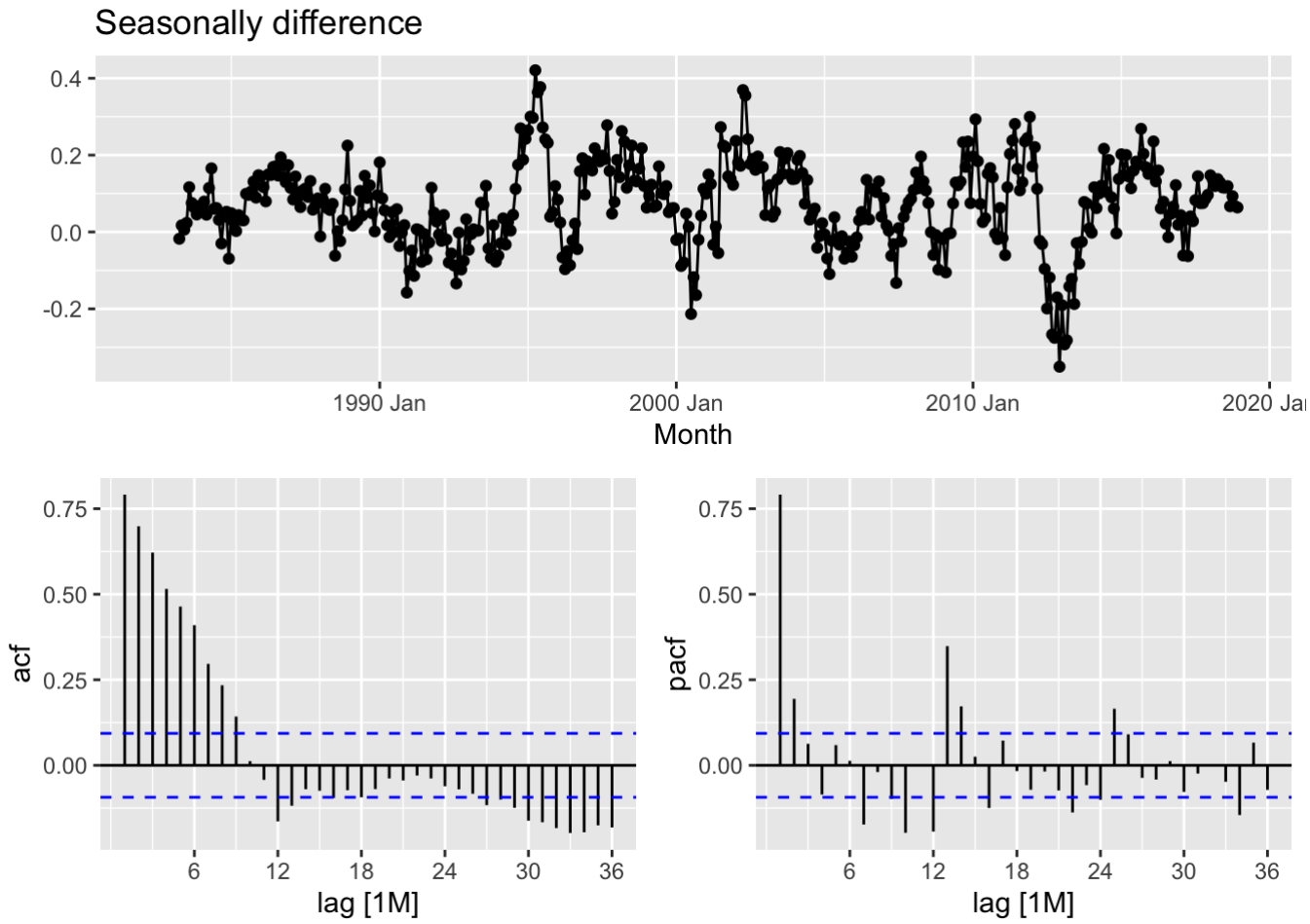
```
# Split data into training set and testing set
test_data <- myseries |>
  slice_tail(n = 24)
train_data <- myseries |>
  slice_head(n = 417)
```

# Short-list of ARIMA model

```
# choose the first ARIMA model based on seasonal differencing
myseries |>
  gg_tsdisplay(difference(log(Turnover), 12), plot_type = "partial", lag = 36) +
  labs(title = "Seasonally difference", y = "")
```

```
## Warning: Removed 12 rows containing missing values (`geom_line()`).
```

```
## Warning: Removed 12 rows containing missing values (`geom_point()`).
```

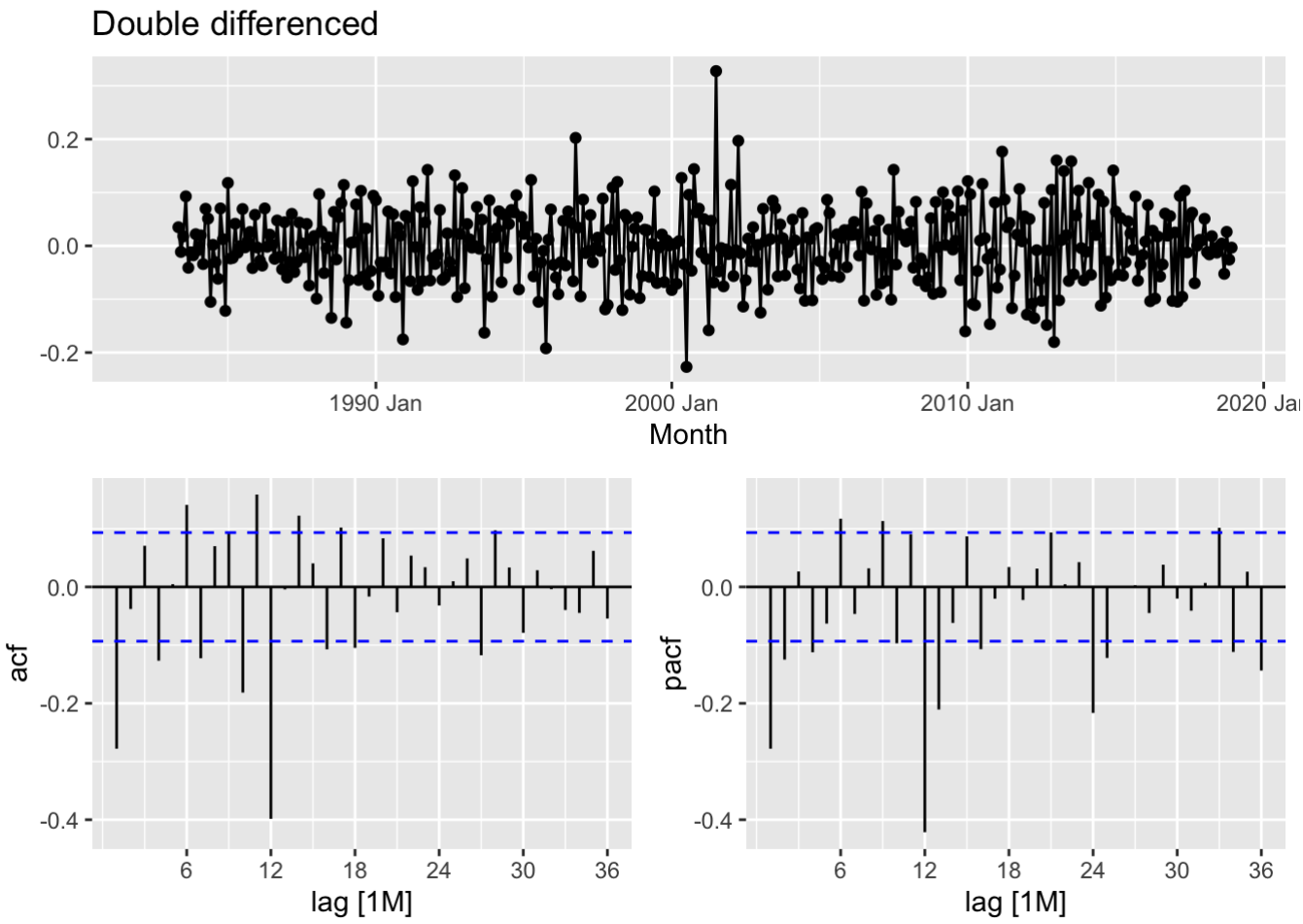



As we take the seasonal differencing, then d = 1. - Spikes in PACF at lag 12 suggests seasonal AR(1) term -> P = 1 - Spikes in PACF suggests non-seasonal AR(2) term -> p = 1. This is because I only chose the significant spikes that large and I do not want to make model too complicated, so I ignore the spike at lag 13. - The model I choose based on PACF plot is: ARIMA(2,0,0)(1,1,0)[12]

```
# choose the second ARIMA model based on doubled differenced
myseries |>
  gg_tsdisplay(difference(log(Turnover), 12) |> difference(), plot_type = "partial",
lag = 36) +
  labs(title = "Double differenced", y = "")
```

```
## Warning: Removed 13 rows containing missing values (`geom_line()`).
```

```
## Warning: Removed 13 rows containing missing values (`geom_point()`).
```




As we take the seasonal differencing and first differencing, then d = 1 and D = 1 - Spikes in PACF at lag 12 suggests seasonal AR(2) term -> P = 2. I ignored lag 36 because it is not much significant compared to other seasonal lags - Spikes in PACF suggests non-seasonal AR(6) term -> p = 6. This is because I only chose the significant spikes that large and I do not want to make model too complex. - Spikes in ACF at lag 12 suggests seasonal MA(1) term -> Q = 1. - The model I choose based on PACF plot is: ARIMA(6,1,0)(2,1,1)[12]

```
# Short-list of ARIMA model
arima_fit <- train_data |>
model (arima1 = ARIMA(log (Turnover) ~ 0 + pdq(2,0,0) + PDQ(1,1,0)), # first ARIMA mo
del
       arima2 = ARIMA(log(Turnover) ~ 0 + pdq (6,1,0) + PDQ(2,1,1)), # second ARIMA m
odel
       auto_arima = ARIMA(log(Turnover), stepwise = FALSE, approximation = FALSE)) #
third ARIMA model chosen by R
```

My ARIMA model short-list has 3 models, the first and the second I have chosen from ACF and PACF plots, and the last one was chosen by algorithm in R, which I only pass the log(Turnover) into the function and set stepwise = FALSE, approximation = FALSE, to make R work harder to choose the best model. All three models were trained by the training data set, which I created from previous step.

```
# Using pivot longer to make models easier to compare
arima_fit |>
  pivot_longer(!c(State, Industry), names_to = "Model name",
               values_to = "Orders")
```

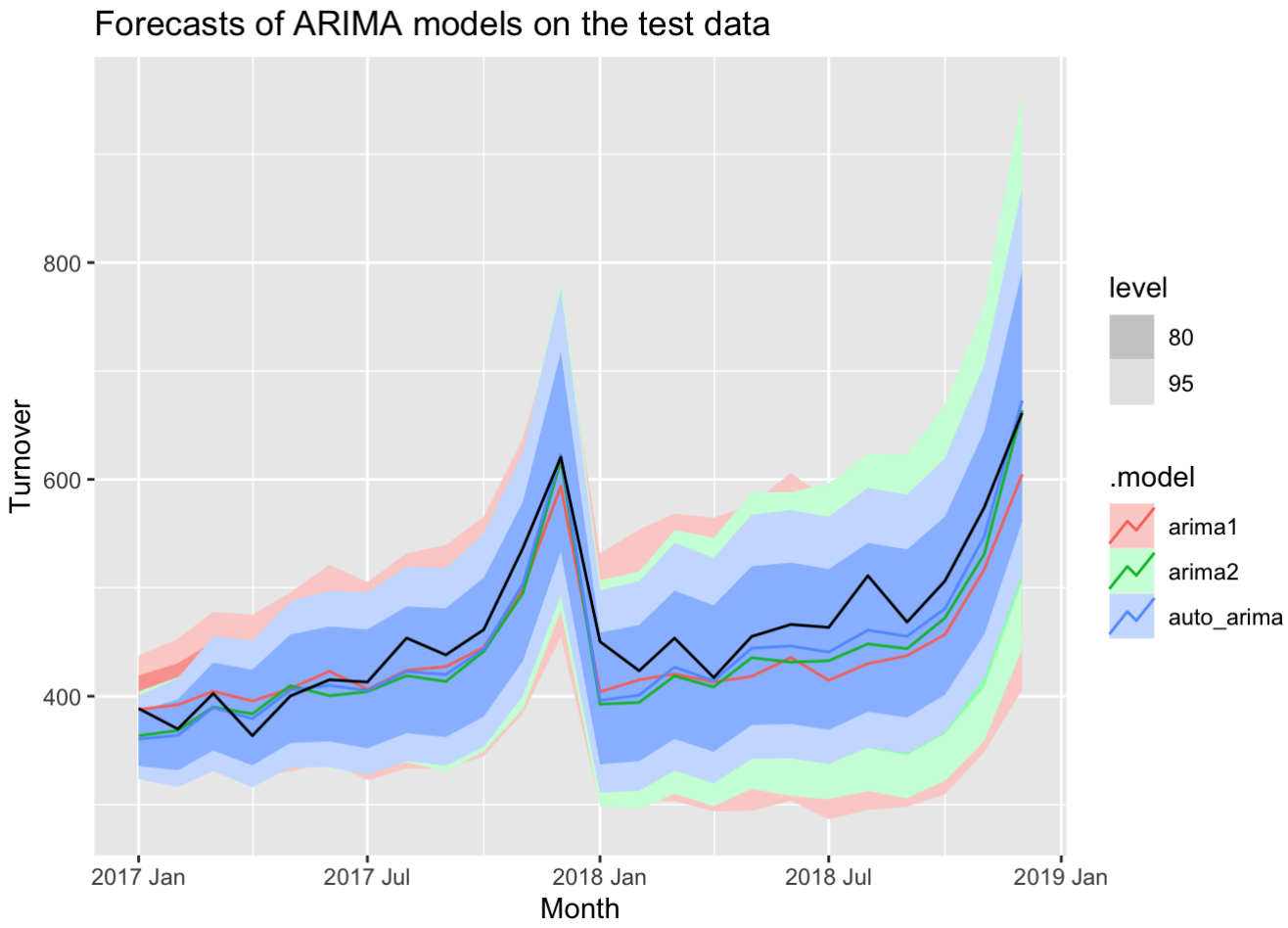
```
## # A mable: 3 x 4
## # Key:     State, Industry, Model name [3]
##   State    Industry              `Model name`                                Orders
##   <chr>    <chr>                 <chr>                                      <model>
## 1 Victoria Other retailing n.e.… arima1               <ARIMA(2,0,0)(1,1,0)[12]>
## 2 Victoria Other retailing n.e.… arima2               <ARIMA(6,1,0)(2,1,1)[12]>
## 3 Victoria Other retailing n.e.… auto_arima   <ARIMA(3,0,2)(0,1,1)[12] w/ drift>
```

```
# Extract the AICc values to compare models
glance(arima_fit) |> arrange(AICc) |> select(.model:BIC)
```

```
## # A tibble: 3 × 6
##   .model      sigma2 log_lik    AIC   AICc    BIC
##   <chr>        <dbl>   <dbl>  <dbl>  <dbl>  <dbl>
## 1 auto_arima 0.00297    602. -1188. -1188. -1156.
## 2 arima2     0.00304    597. -1174. -1173. -1134.
## 3 arima1     0.00390    548. -1089. -1088. -1073.
```

We can compare how well the models perform by looking at their AICc values, the smaller the AICc value is, the better the model performs. As we can see from the table, the model that was chosen by R performs best (AICc value = -1187.613), followed by arima2 (AICc value = -1173.456) and then arima1 (AICc value = -1088.496). Model ‘auto_arima’ and ‘arima2’ have close AICc value, so their performance in forecasting should not be really different.

```
# Apply the forecast on the test_data
arima_fit |>
  forecast(.model = "auto") |>
  autoplot(test_data) +
  labs(title = "Forecasts of ARIMA models on the test data")
```




This graph shows how well these models forecast in the test data. As we can see, the blue line, which is model 'auto_arima', is the closest line to the black line. Therefore, in this case, 'auto_arima' is performing the best.

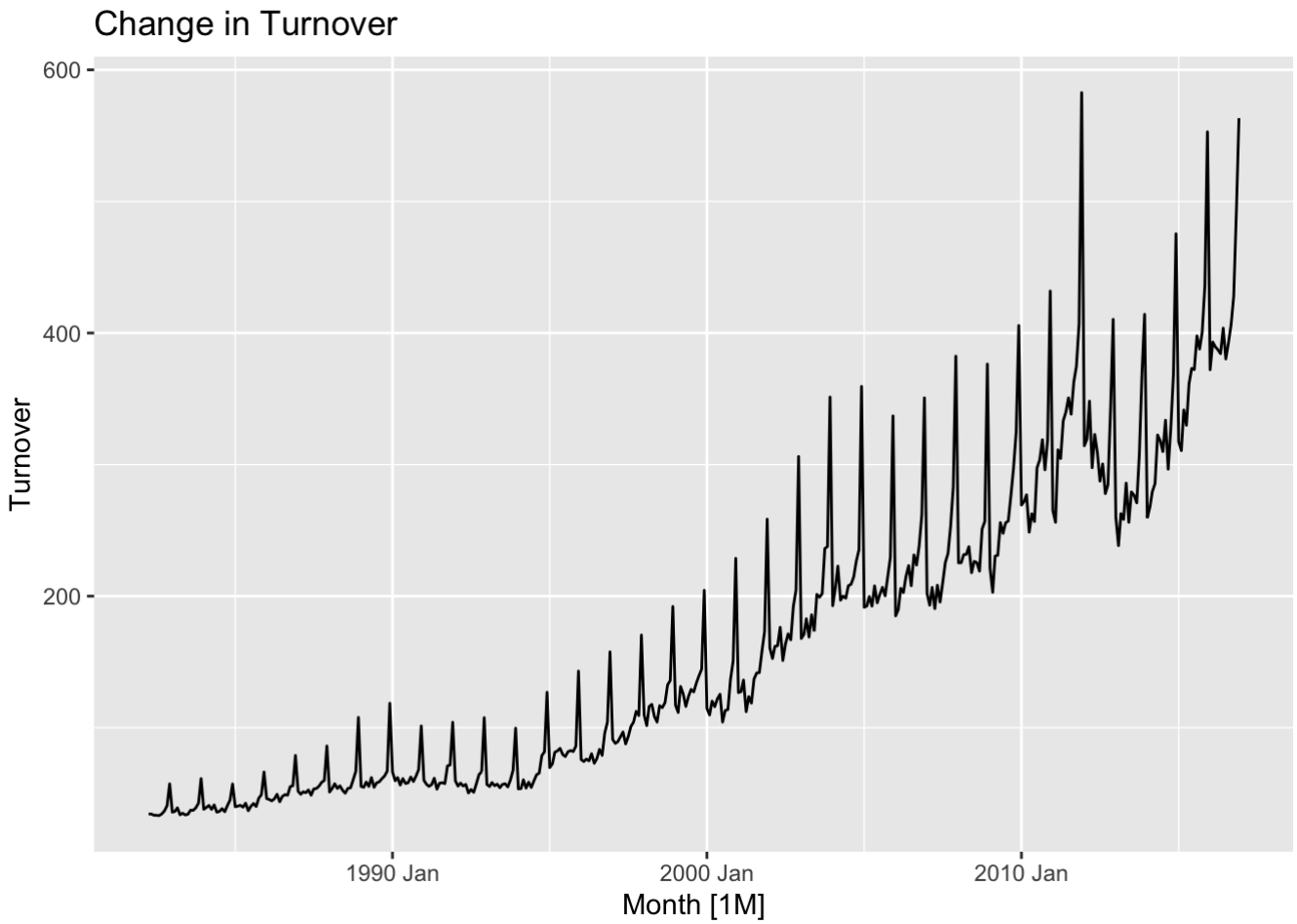
```
# Accuracy metrics to evaluate the performance of models
arima_fit |>
  forecast(h = 24) |>
  accuracy(test_data)
```

```
## # A tibble: 3 × 12
##   .model    State Industry .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE   ACF1
##   <chr>     <chr> <chr>    <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>
## 1 arima1    Vict… Other r… Test   22.5  35.1  28.5  4.30  5.88   NaN   NaN 0.581
## 2 arima2    Vict… Other r… Test   22.3  29.7  25.0  4.70  5.39   NaN   NaN 0.160
## 3 auto_ar… Vict… Other r… Test   16.7  23.7  19.6  3.59  4.24   NaN   NaN 0.0138
```

When applying to the test set, we can see that model 'auto_arima' performs best, compared to the others, since its ME, RMSE, MAE, is the lowest among all models. The error in forecasting of that model is the lowest. Therefore, I choose it to be my best arima model: ARIMA(3,0,2)(0,1,1)[12] w/ drift

# Short-list of ETS model

```
# Plot of original training data
train_data |>
  autoplot(Turnover) +
  labs(title = "Change in Turnover",
       y = "Turnover")
```




As we can see, there are both trend and seasonality appear in the data. Therefore we need to specify whether to use multiplicative/additive trend and multiplicative/additive seasonality in the ETS model, as well as the error terms. Since the variance of data is increase proportional with the level of the series, I would consider the following cases: 1. ETS(M,A,A) 2. ETS(M,Ad,M)

```
# Short-list of ETS model
ets_fit <- train_data |>
  model(ets1 = ETS(Turnover ~ error("M") + trend("A") + season("A")),
        ets2 = ETS(Turnover ~ error("M") + trend("Ad") + season("M")),
        auto_ets = ETS(Turnover))
```

My ETS model short-list has 3 models, the first and the second I have chosen based on the plot, and the last one was chosen by algorithm in R, which I only pass the Turnover into the function. All three models were trained by the training data set, which I created from previous step.

```
# Using pivot longer to make models easier to compare
ets_fit |>
  pivot_longer(!c(State, Industry), names_to = "Model name",
               values_to = "Orders")
```

```
## # A mable: 3 x 4
## # Key:     State, Industry, Model name [3]
##   State    Industry               `Model name`        Orders
##   <chr>    <chr>                  <chr>              <model>
## 1 Victoria Other retailing n.e.c. ets1          <ETS(M,A,A)>
## 2 Victoria Other retailing n.e.c. ets2         <ETS(M,Ad,M)>
## 3 Victoria Other retailing n.e.c. auto_ets      <ETS(M,A,M)>
```

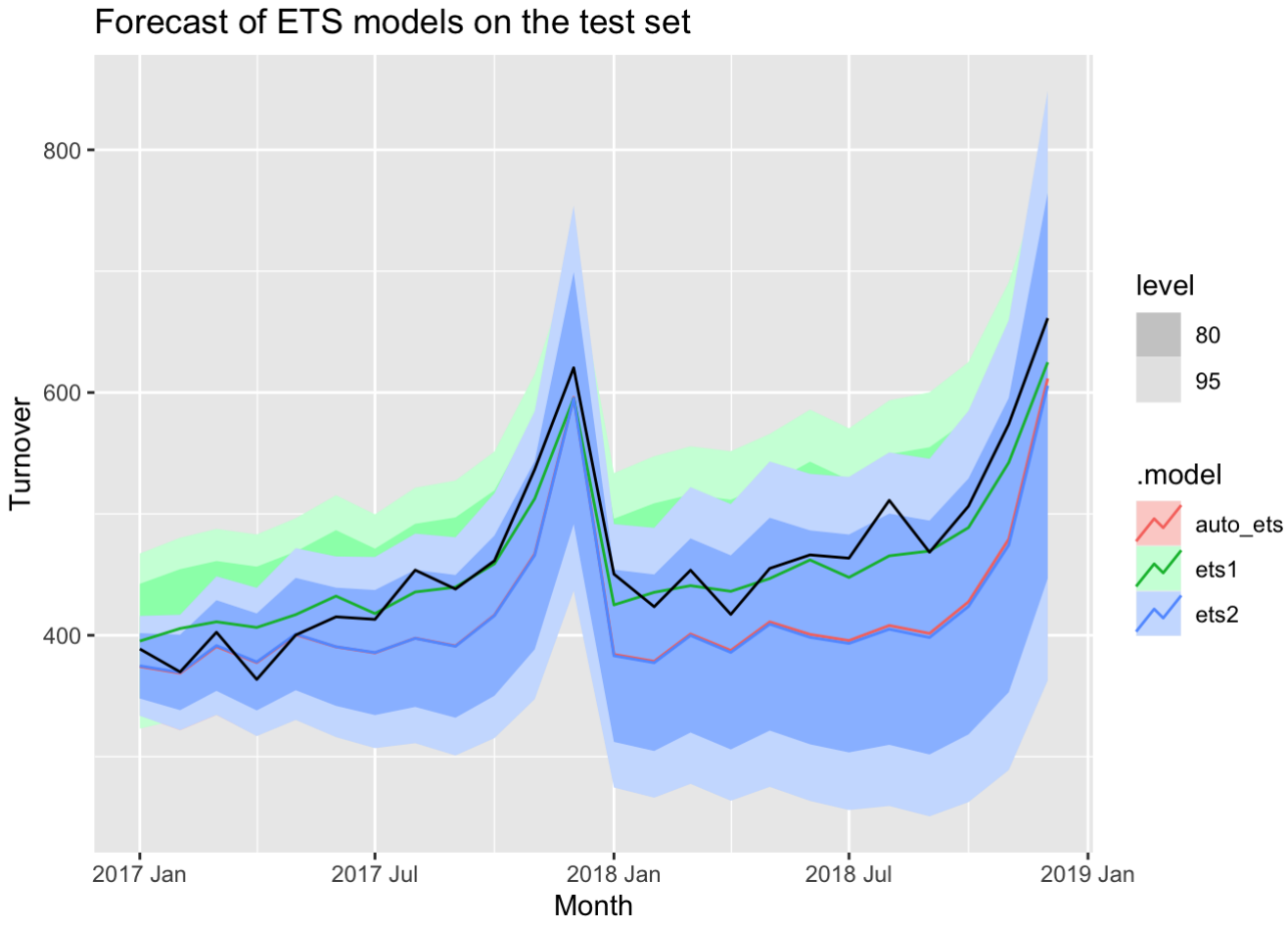
As we can see, the model 2 that I have chosen is not really different from the model that R chose, except the trend is Damped trend.

```
# Extract the AICc values to compare models
glance(ets_fit) |> arrange(AICc) |> select(.model:BIC)
```

```
## # A tibble: 3 × 6
##   .model    sigma2 log_lik   AIC  AICc   BIC
##   <chr>      <dbl>   <dbl> <dbl> <dbl> <dbl>
## 1 auto_ets 0.00307  -2051. 4136. 4137. 4204.
## 2 ets2     0.00314  -2054. 4143. 4145. 4216.
## 3 ets1     0.00867  -2270. 4575. 4576. 4643.
```

We can compare how well the models perform by looking at their AICc values, the smaller the AICc value is, the better the model performs. As we can see from the table, the model that was chosen by R performs best (AICc value = 4135.816), followed by ets2 (AICc value = 4143.241) and then ets1 (AICc value = 4574.776). Model 'auto_ets' and 'ets2' have close AICc value, so their performance in forecasting should not be really different.

```
# Apply the forecast on the test data
ets_fit |>
  forecast(.model = "auto") |>
  autoplot(test_data) +
  labs(title = "Forecast of ETS models on the test set")
```




This graph shows different result to the AICc values. That the model which has highest AICc performs best (ets1), since its green line is really close to the black line. The other two are quite far from the actual data.

```
# Accuracy metrics to evaluate the performance of models
ets_fit |>
  forecast(h = 24) |>
  accuracy(test_data)
```

```
## # A tibble: 3 × 12
##   .model   State  Industry .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1
##   <chr>    <chr>  <chr>    <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 auto_ets Victo… Other r… Test  44.7   53.4  45.8 9.24   9.57   NaN   NaN 0.642
## 2 ets1     Victo… Other r… Test   4.25  22.1  18.1 0.352  3.91   NaN   NaN 0.396
## 3 ets2     Victo… Other r… Test  46.0   55.2  47.3 9.50   9.84   NaN   NaN 0.666
```

As we use different metrics to compare models, we could see that the 'auto_ets' model does not perform the best anymore, since its metrics are really high, almost double the metrics of model ets1. Therefore, I would choose the 'ets1' as my best model, based on metrics such as ME, RMSE, MAE, etc.

# Question 4: Choose one ARIMA model and one ETS model based on this analysis and show parameter estimates, residual diagnostics, forecasts and prediction intervals for both models. Diagnostic checking for both models should include ACF graphs and the Ljung-Box test. [8 marks]

# ARIMA model

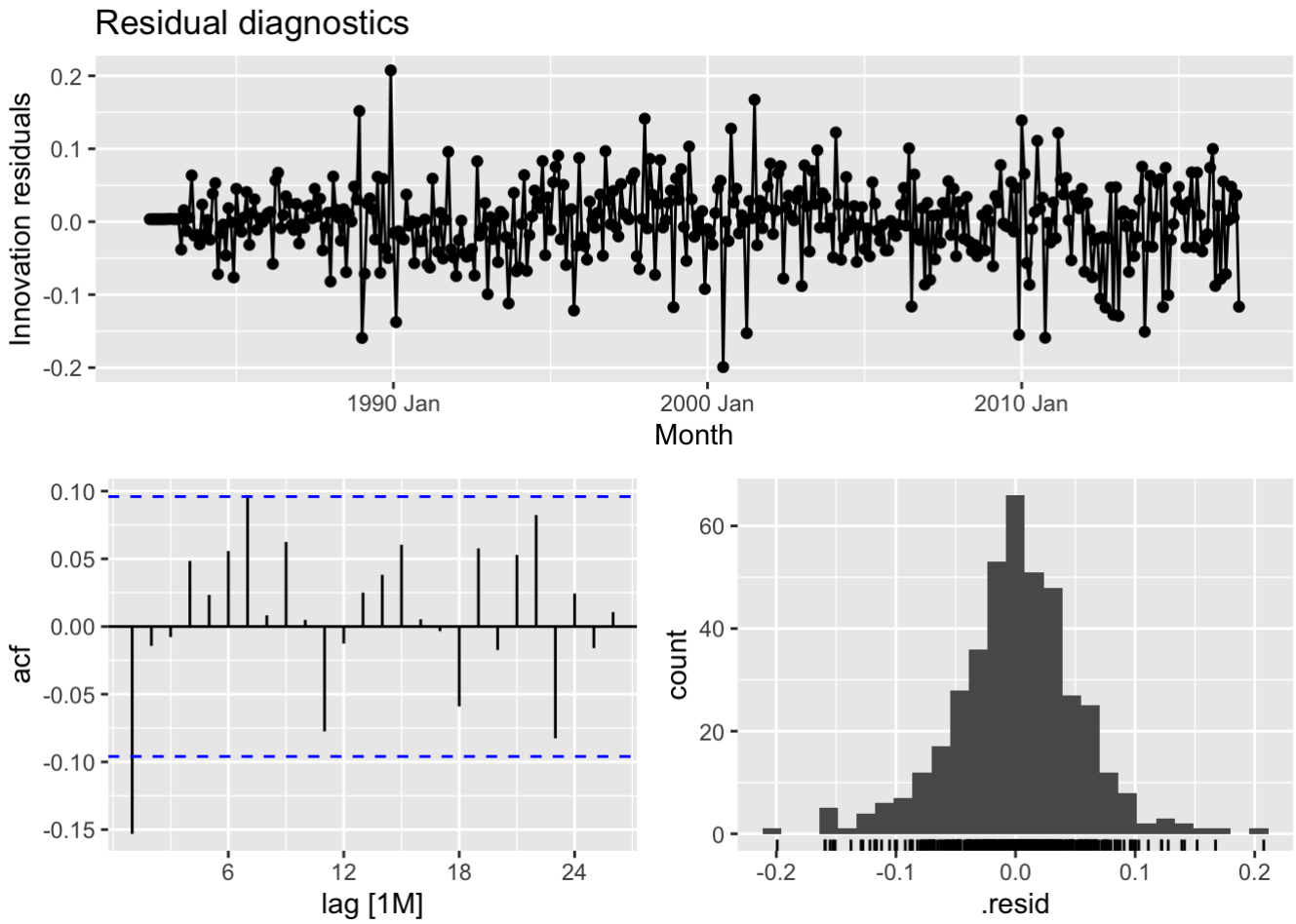
```
# parameter estimates
arima_best <- arima_fit |>
  select(auto_arima) |>
report(arima_best)
```

```
## Series: Turnover
## Model: ARIMA(3,0,2)(0,1,1)[12] w/ drift
## Transformation: log(Turnover)
##
## Coefficients:
##           ar1     ar2     ar3     ma1     ma2     sma1  constant
##       -0.2307  0.0564  0.8778  1.0431  0.8820  -0.7003    0.0209
## s.e.   0.0486  0.0464  0.0407  0.1014  0.1222   0.0516    0.0025
##
## sigma^2 estimated as 0.002968:  log likelihood=601.99
## AIC=-1187.98   AICc=-1187.61   BIC=-1155.95
```

Coefficients: - ar1, ar2, ar3: These coefficients represent the autoregressive parameters for the non-seasonal component of the ARIMA model. In this case, the model includes lag 1, lag 2, and lag 3. - ma1, ma2: These coefficients represent the moving average parameters for the non-seasonal component of the ARIMA model.

In this case, the model includes lag 1 and lag 2. - sma1: This coefficient represents the seasonal moving average parameter for the seasonal component of the ARIMA model. - constant: This coefficient represents the drift term in the ARIMA model, which captures any systematic linear trend or bias in the data.

```
# Residual diagnostics
gg_tsresiduals(arima_best) +
  labs(title = "Residual diagnostics")
```



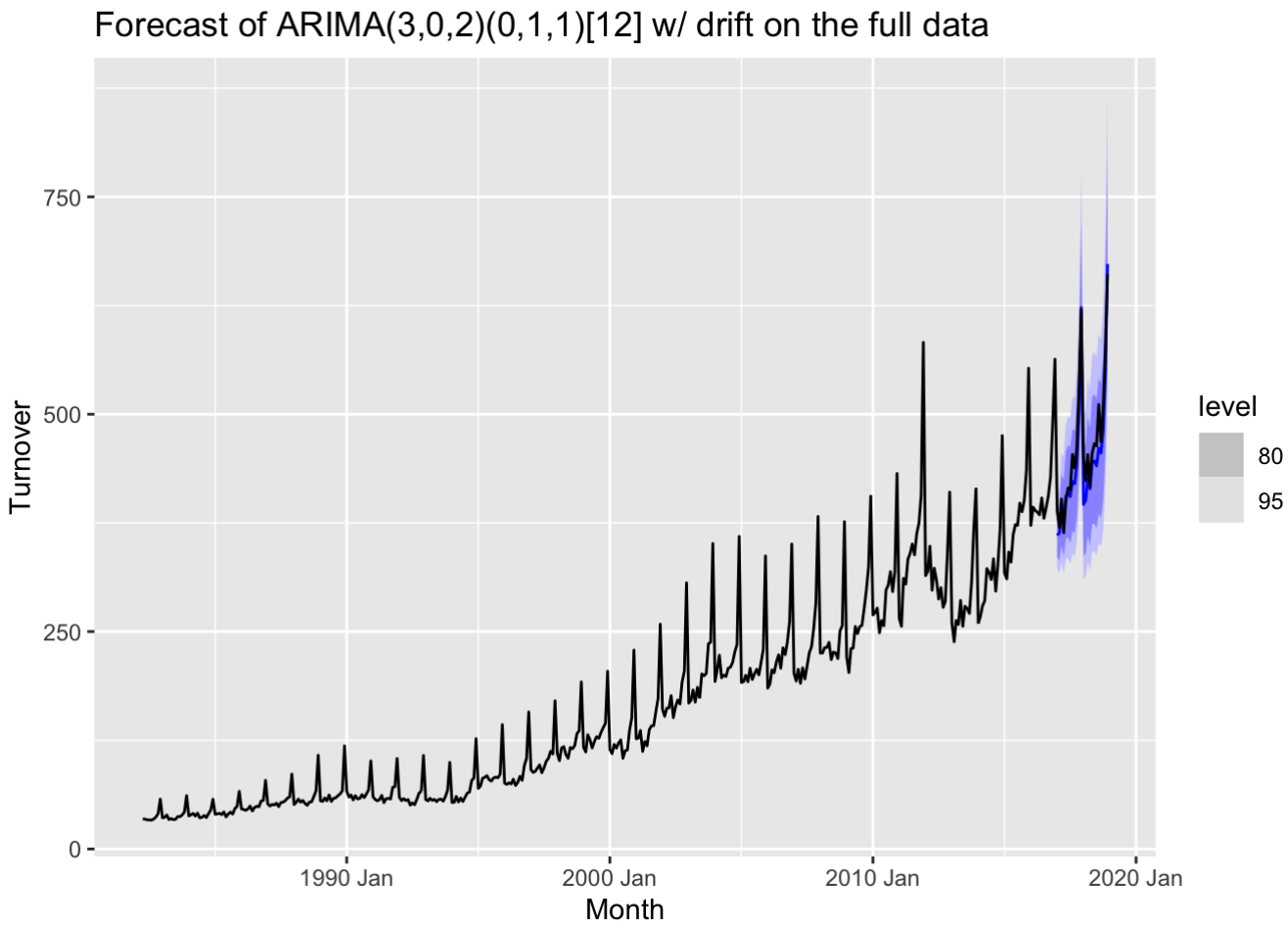

As we can see from the graphs above, I would conclude that the error of this model is stationary and also white-noise, since there's no autocorrelation the ACF plot, even when there is one significant lag. The innovative residuals also shows a constant variance in different level of the serie. And the .resid histogram show that the error is Normal distribution. So the error in this model is white-noise.

```
# Ljung-box test
augment(arima_best) |>
  features(.innov, ljung_box, lag = 36, dof = 6)
```

```
## # A tibble: 1 × 3
##   .model     lb_stat lb_pvalue
##   <chr>        <dbl>     <dbl>
## 1 auto_arima    39.9     0.107
```

As the lb_pvalue is 0.107, which is > 0.05. Therefore, we fail to reject the null hypothesis and conclude that there's no autocorrelation in the residuals. Therefore, the model is reliable to be used to forecast.

```
# Forecast of best arima model
best_arima <- train_data |>
  model(arima = ARIMA(log(Turnover), stepwise = FALSE, approximation = FALSE)) |>
  forecast(h = 24) |>
  autoplot(myseries) +
  labs(title = "Forecast of ARIMA(3,0,2)(0,1,1)[12] w/ drift on the full data")
best_arima
```




As we can see from the graph, the forecast of the best arima model is similar to the actual data. The forecasting line and the actual data line almost match each other.

```
# Prediction interval of the forecast
arima_interval <- train_data |>
  model(arima = ARIMA(log(Turnover), stepwise = FALSE, approximation = FALSE)) |>
  forecast(h = 24) |>
  hilo()
arima_interval
```

```
## # A tsibble: 24 x 8 [1M]
## # Key:       State, Industry, .model [1]
##    State Industry .model    Month          Turnover .mean                  `80%`
##    <chr> <chr>    <chr>     <mth>            <dist> <dbl>                 <hilo>
##  1 Vict… Other r… arima  2017 Jan  t(N(5.9, 0.003))  361. [335.7562, 386.0744]80
##  2 Vict… Other r… arima  2017 Feb t(N(5.9, 0.0049))  364. [331.7455, 397.1396]80
##  3 Vict… Other r… arima  2017 Mar   t(N(6, 0.0066))  390. [349.9405, 430.9603]80
##  4 Vict… Other r… arima  2017 Apr t(N(5.9, 0.0083))  379. [336.2101, 424.4786]80
##  5 Vict… Other r… arima  2017 May   t(N(6, 0.0093))  406. [356.9247, 456.8825]80
##  6 Vict… Other r… arima  2017 Jun     t(N(6, 0.01))  410. [358.2905, 464.3493]80
##  7 Vict… Other r… arima  2017 Jul    t(N(6, 0.011))  405. [352.0385, 461.5511]80
##  8 Vict… Other r… arima  2017 Aug    t(N(6, 0.012))  423. [365.9950, 482.7830]80
##  9 Vict… Other r… arima  2017 Sep    t(N(6, 0.012))  420. [362.3087, 481.0624]80
## 10 Vict… Other r… arima  2017 Oct  t(N(6.1, 0.013))  444. [381.3352, 509.3061]80
## # ℹ 14 more rows
## # ℹ 1 more variable: `95%` <hilo>
```

Since the residual is white-noise, the estimated mean of the time series is likely to be an unbiased estimate of the true mean and the prediction intervals are likely to be symmetric and capture the random variability of the future observations. This allows for more reliable and accurate estimation of the uncertainty associated with the forecasted values. In this case, even though the residual is white-noise, there's still some uncertainty in the forecast, as we can see the width of prediction intervals are quite large in some months.

# ETS model

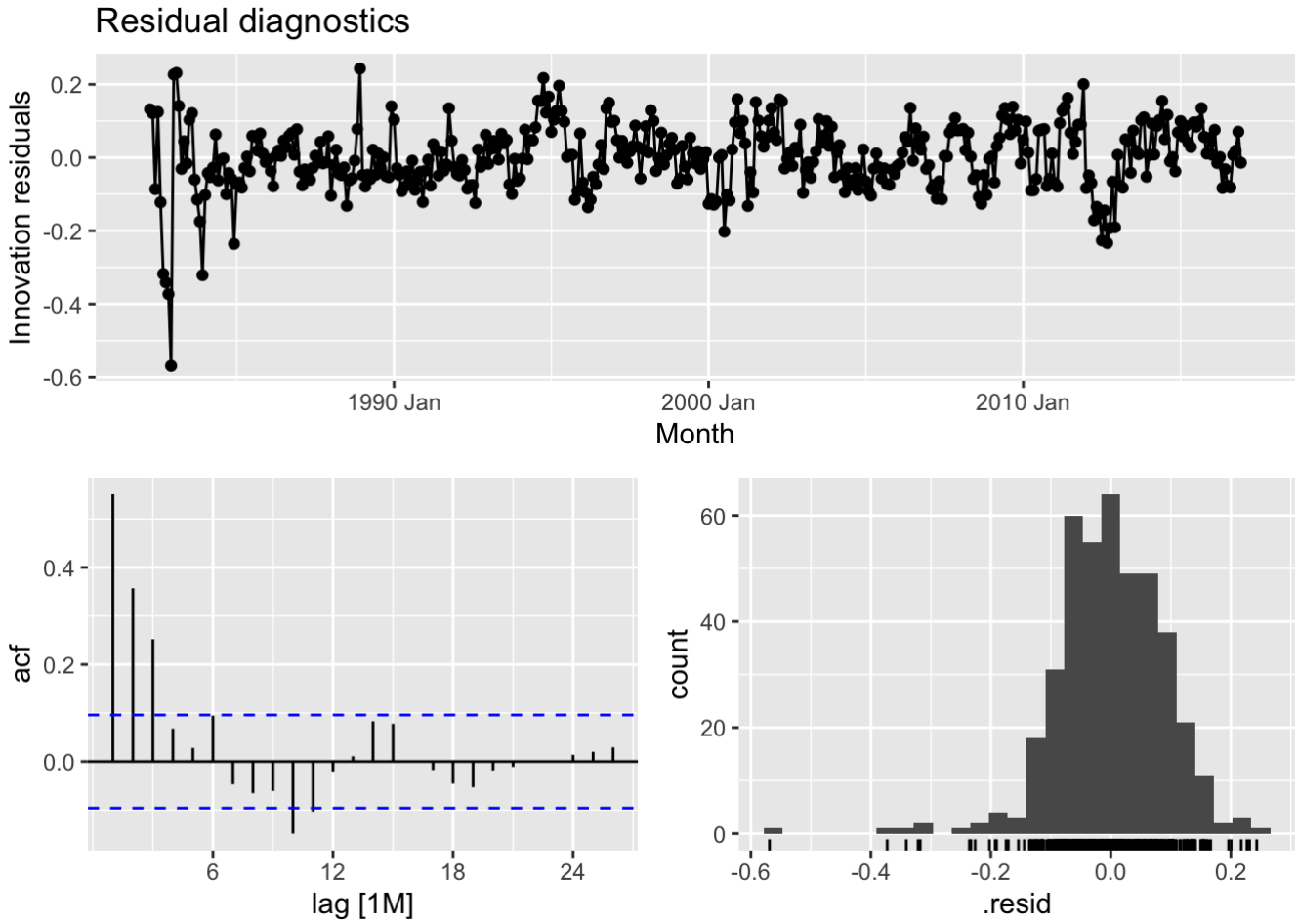
```
# parameter estimates
ets_best <- ets_fit |>
  select(ets1) |>
report(ets_best)
```

```
## Series: Turnover
## Model: ETS(M,A,A)
##   Smoothing parameters:
##     alpha = 0.1366287
##     beta  = 0.006594895
##     gamma = 0.6022353
##
##   Initial states:
##      l[0]     b[0]      s[0]     s[-1]     s[-2]    s[-3]  s[-4]    s[-5]
##  40.98503 1.764697 -8.866856 -11.64871 -10.18724 84.15005 14.188 3.932928
##      s[-6]     s[-7]     s[-8]     s[-9]    s[-10]    s[-11]
##  -1.690347 -13.27808 -19.07684 -10.75455 -14.41419 -12.35417
##
##   sigma^2:  0.0087
##
##      AIC     AICc      BIC
## 4574.776 4576.310 4643.339
```

The model estimates the smoothing parameters that control the weight given to the most recent observations and the rate at which the model adapts to changes. The values reported here are: - Alpha: The smoothing parameter for the level component (trend). - Beta: The smoothing parameter for the trend component. - Gamma: The smoothing parameter for the seasonal component.

Initial States: These are the initial values for the level (l), trend (b), and seasonal (s) components. The initial states are essential for initializing the model. The values reported here are the initial states for each component at time t = 0 and for several preceding time points (s[-1], s[-2], etc.). These values are estimated based on the data and play a role in the model's forecasting.

```
# Residual plot
gg_tsresiduals(ets_best) +
  labs(title = "Residual diagnostics")
```



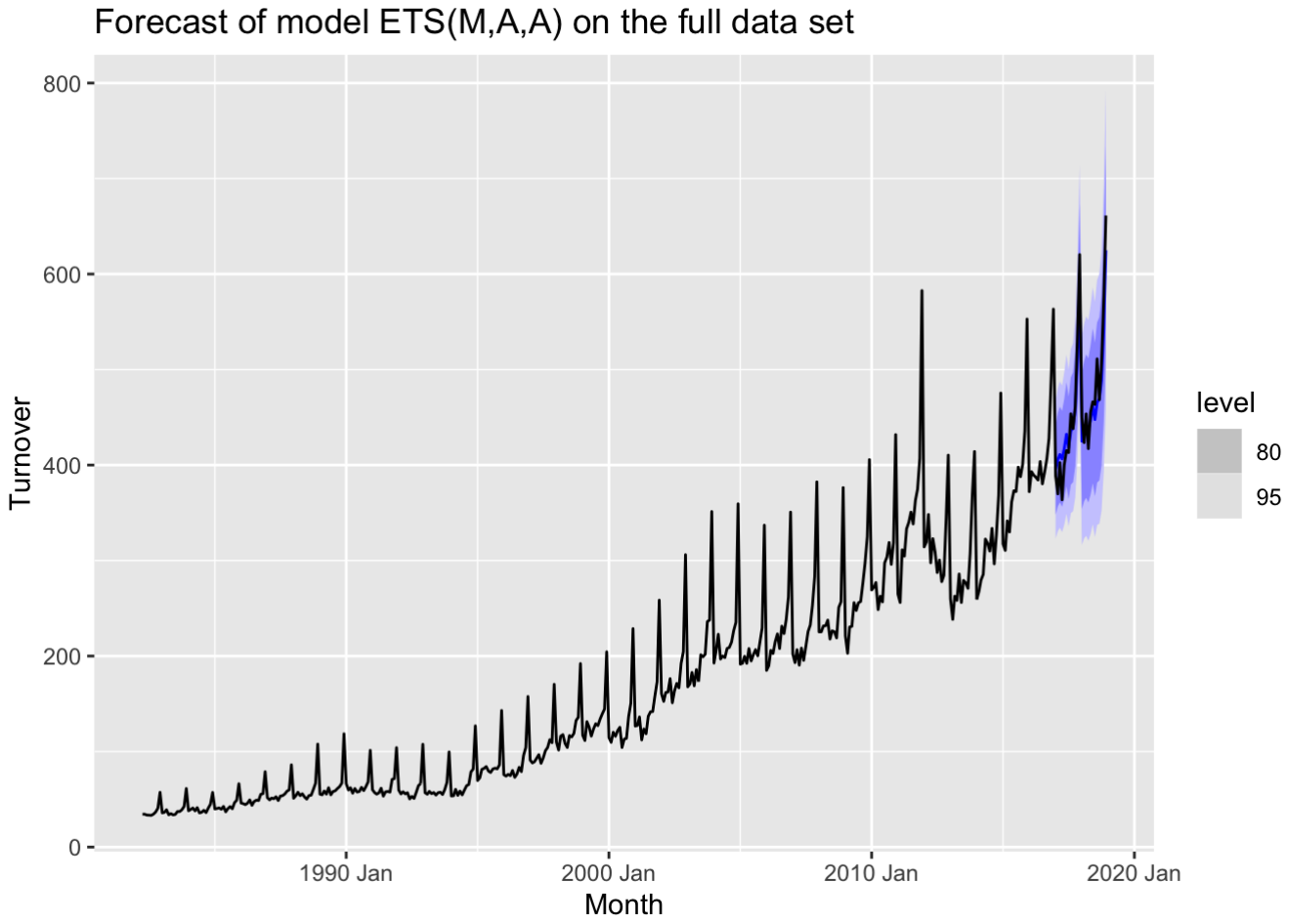

The ACF plot suggests that there's autocorrelation appears in the data, as the correlations are sinusoidal. There's a constant variance in different level of the series, except the beginning of the graph.

```
# Ljung-box test
augment(ets_best) |>
  features(.innov, ljung_box, lag = 10)
```

```
## # A tibble: 1 × 3
##   .model lb_stat lb_pvalue
##   <chr>    <dbl>     <dbl>
## 1 ets1      228.         0
```

Since the graphs above suggest the autocorrelations, therefore when we do the Ljung-box test, we would reject the null hypothesis and conclude that there's autocorrelation in the residuals, since the lb_value is 0, which is < 0.05.

```
# Forecast of best ETS model
best_ets <- train_data |>
  model(ets = ETS(Turnover ~ error("M") + trend("A") + season("A"))) |>
  forecast(h = 24) |>
  autoplot(myseries) +
  labs(title = "Forecast of model ETS(M,A,A) on the full data set")
best_ets
```



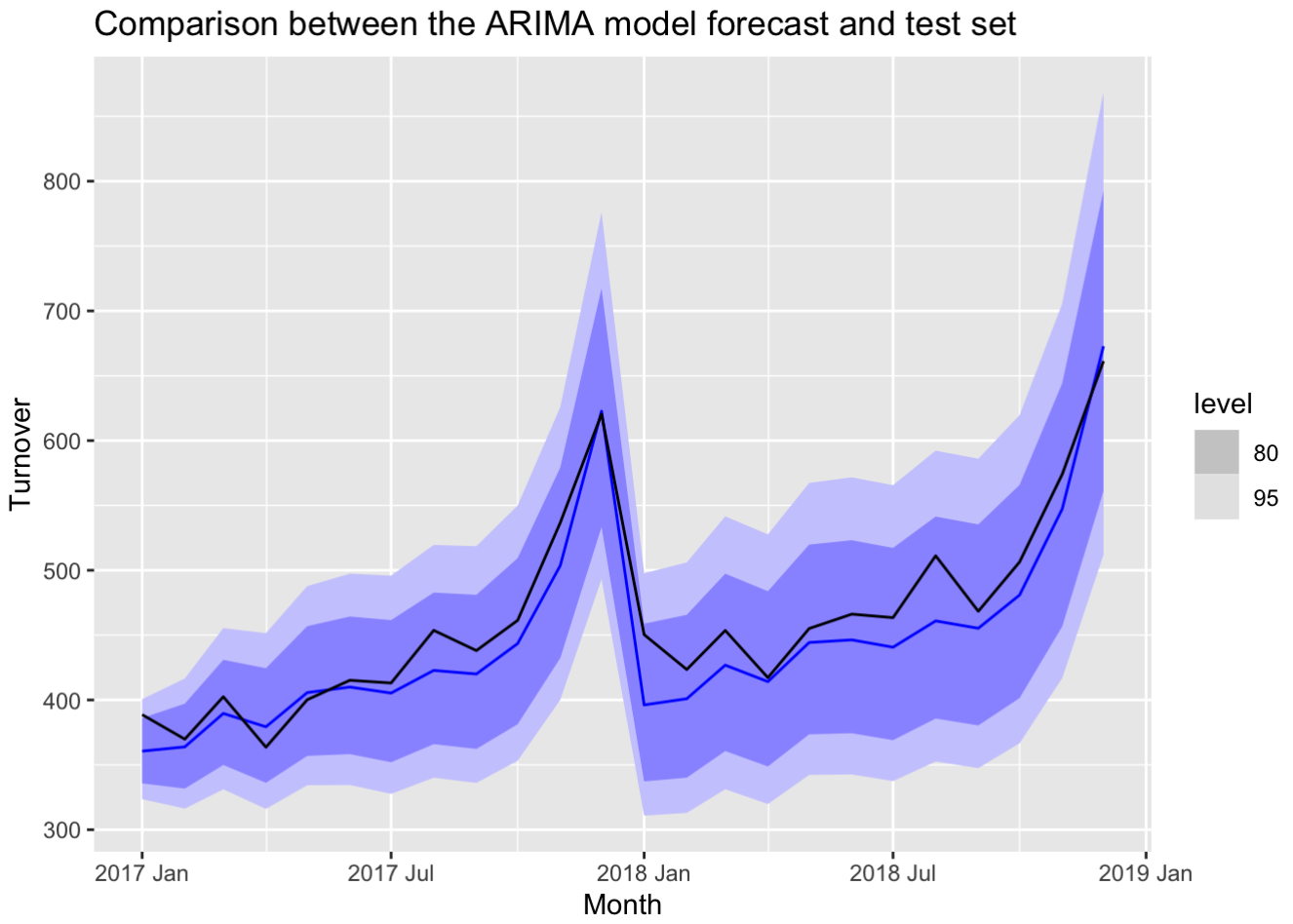

```
# Prediction interval of the forecast
ets_interval <- train_data |>
  model(ets = ETS(Turnover ~ error("M") + trend("A") + season("A"))) |>
  forecast(h = 24) |>
  hilo()
ets_interval
```

```
## # A tsibble: 24 x 8 [1M]
## # Key:       State, Industry, .model [1]
##    State    Industry   .model    Month     Turnover .mean                 `80%`
##    <chr>    <chr>      <chr>     <mth>       <dist> <dbl>                <hilo>
##  1 Victoria Other ret… ets    2017 Jan N(395, 1355)  395. [348.0567, 442.3931]80
##  2 Victoria Other ret… ets    2017 Feb N(406, 1455)  406. [356.7132, 454.4704]80
##  3 Victoria Other ret… ets    2017 Mar N(411, 1526)  411. [361.0334, 461.1519]80
##  4 Victoria Other ret… ets    2017 Apr N(406, 1529)  406. [356.3306, 456.5431]80
##  5 Victoria Other ret… ets    2017 May N(417, 1642)  417. [365.0591, 468.9317]80
##  6 Victoria Other ret… ets    2017 Jun N(432, 1798)  432. [377.9288, 486.6083]80
##  7 Victoria Other ret… ets    2017 Jul N(418, 1741)  418. [364.4190, 471.3551]80
##  8 Victoria Other ret… ets    2017 Aug N(436, 1923)  436. [379.4646, 491.8510]80
##  9 Victoria Other ret… ets    2017 Sep N(440, 2010)  440. [382.1789, 497.0809]80
## 10 Victoria Other ret… ets    2017 Oct N(459, 2222)  459. [398.5448, 519.3612]80
## # ℹ 14 more rows
## # ℹ 1 more variable: `95%` <hilo>
```

Even when there's autocorrelations in the data, but ETS model can handle the non-stationary data. It can still perform well with the autocorrelations. The graph above shows that the actual line and the forecasting line is really close to each other. Therefore, the performance of this ETS model is not too bad in my opinion, even when it does not pass the Ljung-box test, since its really hard for a model to satisfy every conditions. There's still some uncertainty in the forecast, as we can see the width of prediction intervals are quite large in some months.

# Question 5: Comparison of the results from each of your preferred models. Which method do you think gives the better forecasts? Explain with reference to the test-set. [2 marks]

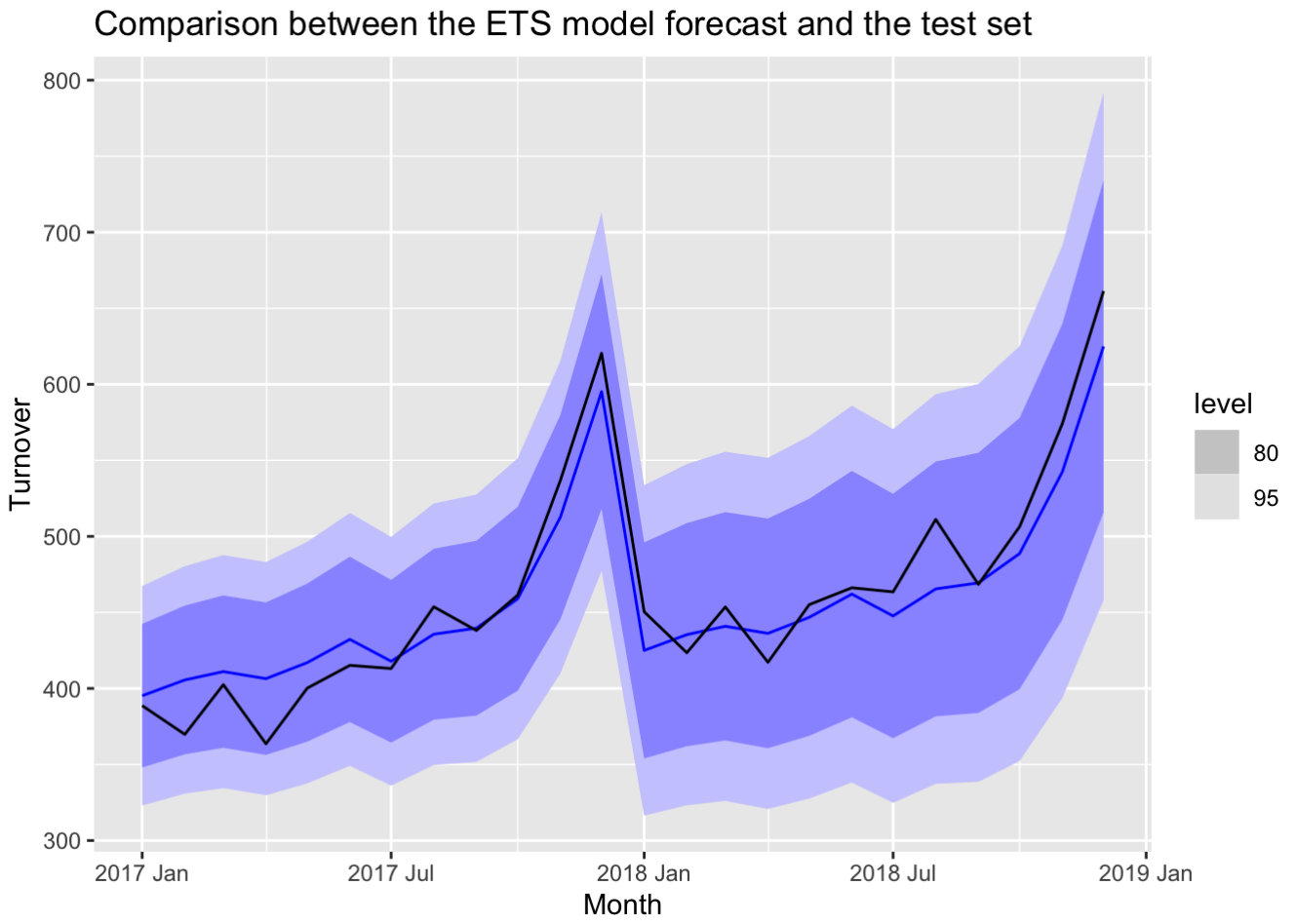
```
# Plot the forecast the the best ARIMA model against the test set
best_arima <- train_data |>
  model(arima = ARIMA(log(Turnover), stepwise = FALSE, approximation = FALSE)) |>
  forecast(h = 24) |>
  autoplot(test_data) +
  labs(title = "Comparison between the ARIMA model forecast and test set")
best_arima
```




```
# Check the accuracy in forecasting of ARIMA model using the test set
arima_accuracy <- arima_fit |>
  forecast(h = 24) |>
  accuracy(test_data)
arima_accuracy |>
  filter(.model == "auto_arima")
```

```
## # A tibble: 1 × 12
##   .model    State Industry .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE   ACF1
##   <chr>     <chr> <chr>    <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>
## 1 auto_ar… Vict… Other r… Test   16.7  23.7  19.6  3.59  4.24   NaN   NaN 0.0138
```

```
# Plot the forecast the the best ETS model against the test set
best_ets <- train_data |>
  model(ets = ETS(Turnover ~ error("M") + trend("A") + season("A"))) |>
  forecast(h = 24) |>
  autoplot(test_data) +
  labs(title = "Comparison between the ETS model forecast and the test set")
best_ets
```




```
# Check the accuracy in forecasting of ETS model using the test set
ets_accuracy <- ets_fit |>
  forecast(h = 24) |>
  accuracy(test_data)
ets_accuracy |>
  filter(.model == "ets1")
```

```
## # A tibble: 1 × 12
##   .model State    Industry .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1
##   <chr>  <chr>    <chr>    <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 ets1   Victoria Other r… Test   4.25  22.1  18.1 0.352  3.91   NaN   NaN 0.396
```

As these graphs show us, the forecast of ETS model seems to be closer to the actual data. The accuracy metrics used of these two models are really similar, that the values for ARIMA model are always higher than those of ETS model a little bit, except the Mean Error.The ME represents the average error or bias of the forecasted values. It is calculated as the mean of the forecast errors (residuals). Usually we want ME to be close to 0. The ME of model ‘ets1’ is 4.2 (really close to 0), while the figure for arima model is 16.9. There is a large gap between these numbers. Therefore, I would say that ETS model performs better than ARIMA model, but not too much.

# Question 6: Apply your two chosen models to the full data set, re-estimating the parameters but not changing the model structure. Produce out-of-sample point

# forecasts and 80% prediction intervals for each model for two years past the end of the data provided. [4 marks]

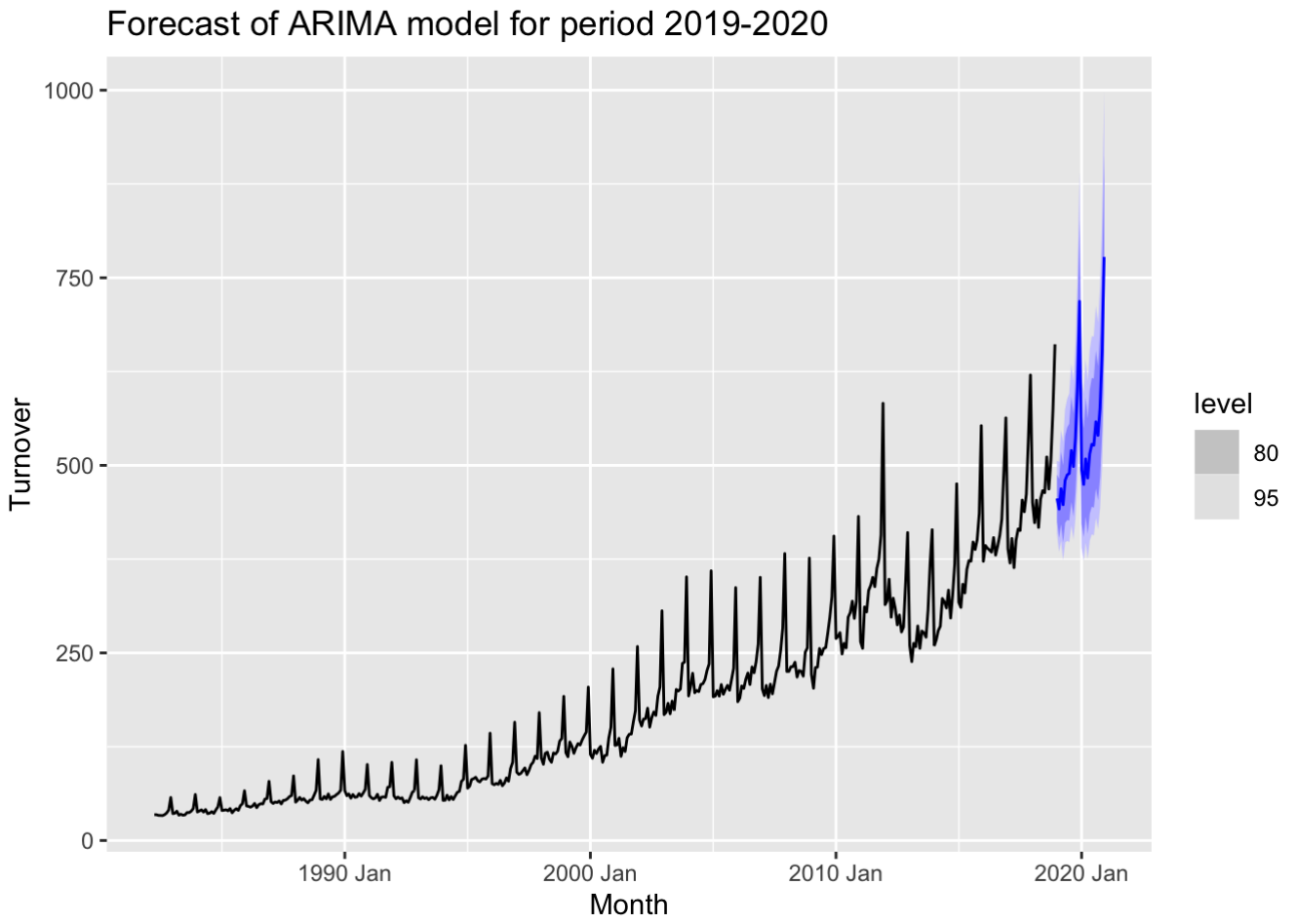
```
# Re-estimate the parameters of the model with the full data set
arima_model <- myseries |>
  model(arima = ARIMA(log (Turnover) ~ 1 + pdq(3,0,2) + PDQ(0,1,1))) |>
report(arima_model)
```

```
## Series: Turnover
## Model: ARIMA(3,0,2)(0,1,1)[12] w/ drift
## Transformation: log(Turnover)
##
## Coefficients:
##           ar1     ar2     ar3     ma1     ma2     sma1  constant
##       -0.2375  0.0507  0.8722  1.0420  0.8849  -0.6728    0.0224
## s.e.   0.0384  0.0372  0.0322  0.0727  0.0812   0.0439    0.0026
##
## sigma^2 estimated as 0.002905:  log likelihood=642.78
## AIC=-1269.56   AICc=-1269.22   BIC=-1237.07
```

```
# 80% prediction intervals and point forecast
arima_model |>
  forecast(h = 24, level = 80) |>
  hilo() |>
  select(c("80%", ".mean"))
```

```
## # A tsibble: 24 x 3 [1M]
##                     `80%` .mean    Month
##                    <hilo> <dbl>    <mth>
##  1 [424.8229, 487.7617]80  456. 2019 Jan
##  2 [403.2457, 481.4751]80  442. 2019 Feb
##  3 [421.8012, 517.7631]80  469. 2019 Mar
##  4 [397.5076, 499.7162]80  447. 2019 Apr
##  5 [423.4264, 539.2247]80  480. 2019 May
##  6 [427.1473, 550.4026]80  487. 2019 Jun
##  7 [426.4204, 555.3915]80  489. 2019 Jul
##  8 [451.7355, 591.6082]80  520. 2019 Aug
##  9 [431.7207, 568.8322]80  498. 2019 Sep
## 10 [463.2528, 613.5942]80  536. 2019 Oct
## # ℹ 14 more rows
```

```
# Forecast of ARIMA model for the next 2 years
arima_model |>
  forecast(h = 24, level = 80) |>
  autoplot(myseries) +
  labs(title = "Forecast of ARIMA model for period 2019-2020")
```



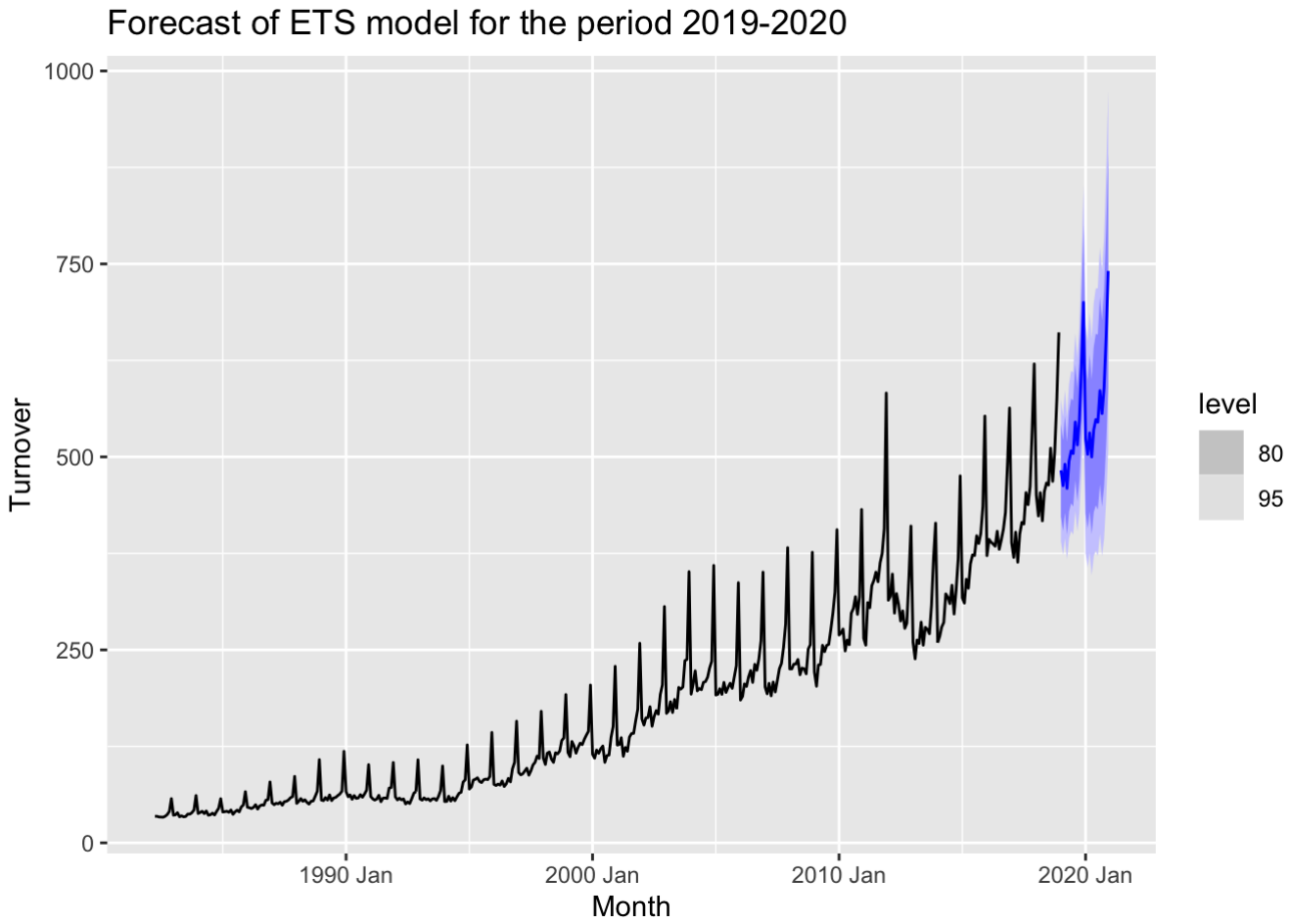

```
# Re-estimate the parameters of the model with the full data set
ets_model <- myseries |>
  model(ets = ETS(Turnover ~ error("M") + trend("A") + season("A"))) |>
report(ets_model)
```

```
## Series: Turnover
## Model: ETS(M,A,A)
##   Smoothing parameters:
##     alpha = 0.1445363
##     beta  = 0.01270171
##     gamma = 0.570657
##
##   Initial states:
##      l[0]     b[0]      s[0]     s[-1]     s[-2]    s[-3]    s[-4]    s[-5]
##  37.71262 3.514404 -4.020015 -18.31448 -16.23792 79.60884 21.58226 11.69103
##      s[-6]     s[-7]     s[-8]     s[-9]    s[-10]    s[-11]
##  -8.211316 -6.738291 -17.53319 -14.91276 -11.47286 -15.44127
##
##   sigma^2:  0.0094
##
##      AIC     AICc      BIC
## 4962.516 4963.963 5032.030
```

```
# 80% prediction intervals and point forecast
ets_model |>
  forecast(h = 24, level = 80) |>
  hilo() |>
  select(c("80%", ".mean"))
```

```
## # A tsibble: 24 x 3 [1M]
##                     `80%` .mean    Month
##                    <hilo> <dbl>    <mth>
##  1 [422.6156, 542.6116]80  483. 2019 Jan
##  2 [404.3815, 520.9719]80  463. 2019 Feb
##  3 [427.9964, 552.9890]80  490. 2019 Mar
##  4 [399.4883, 518.9875]80  459. 2019 Apr
##  5 [429.5123, 559.4146]80  494. 2019 May
##  6 [440.1070, 575.5342]80  508. 2019 Jun
##  7 [435.6902, 573.0398]80  504. 2019 Jul
##  8 [470.4547, 619.9168]80  545. 2019 Aug
##  9 [442.3946, 588.6083]80  516. 2019 Sep
## 10 [469.9682, 626.7756]80  548. 2019 Oct
## # ℹ 14 more rows
```

```
# Forecast of ETS model for the next 2years
ets_model |>
  forecast(h = 24, level = 80) |>
  autoplot(myseries) +
  labs(title = "Forecast of ETS model for the period 2019-2020")
```



# Question 7: Obtain up-to-date data from the ABS website (https://www.abs.gov.au/statistics/industry/retail-and-wholesale-trade/retail-trade-australia (https://www.abs.gov.au/statistics/industry/retail-and-wholesale-trade/retail-trade-australia) Table 11). You may need to use the previous release of data, rather than the latest release. Compare your forecasts with the actual numbers. How well did you do? [5 marks]

# ARIMA model

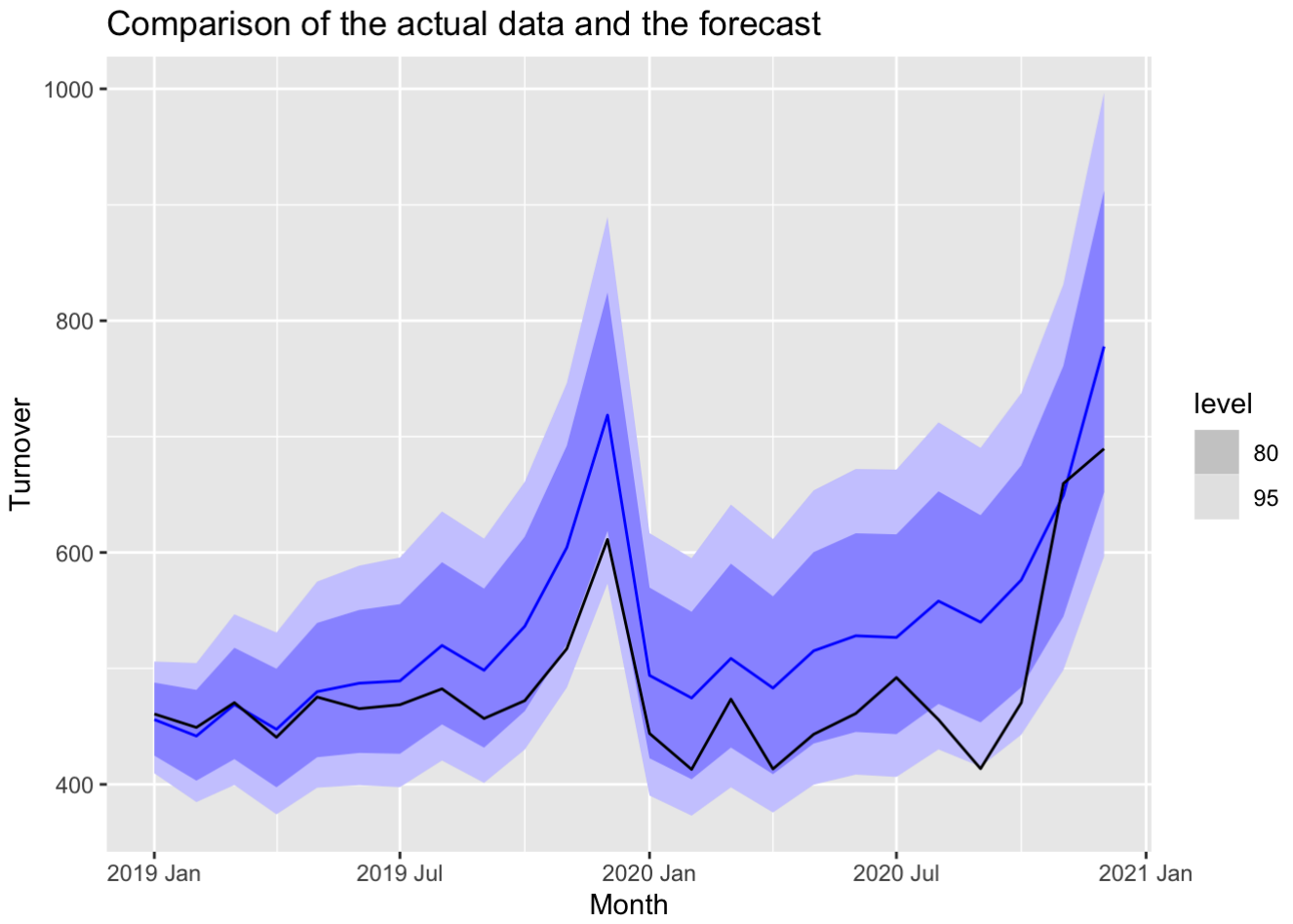
```
# read the new data
new_data <- readxl::read_excel("8501011.xls", sheet = "Data1", skip = 9) |>
  select(Month = `Series ID`, Turnover = myseries$`Series ID`[1]) |>
  mutate(
    Month = yearmonth(Month),
    State = myseries$State[1],
    Industry = myseries$Industry[1]
  ) |>
  as_tsibble(index = Month, key = c(State, Industry))
```

```
# Get data of 2019-2020 in order to compare
data_compare <- new_data |>
  slice_tail(n = 24)
```

```
# Check the accuracy of ARIMA model using the new data
arima_model |>
  forecast(h = 24, level = 80) |>
  accuracy(data_compare)
```

```
## # A tibble: 1 × 12
##   .model State    Industry .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1
##   <chr>  <chr>    <chr>    <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 arima  Victoria Other r… Test  -49.2  63.3  51.2 -10.4  10.7   NaN   NaN 0.453
```

```
# Plot the difference between actual data and forecast
arima_model |>
  forecast(h = 24, level = 80) |>
  autoplot(data_compare) +
  labs(title = "Comparison of the actual data and the forecast")
```




```
# Get the point forecast of ARIMA model
arima_mean <- arima_model |>
  forecast(h = 24, level = 80) |>
  hilo() |>
  select(.mean)
```

```
# Compare the actual numbers and the point forecast
full_join(data_compare, arima_mean, by = "Month") |>
  select(-c(State, Industry))
```

```
## # A tsibble: 24 x 3 [1M]
##       Month Turnover .mean
##       <mth>    <dbl> <dbl>
##  1 2019 Jan     461.  456.
##  2 2019 Feb     449.  442.
##  3 2019 Mar     470.  469.
##  4 2019 Apr     441.  447.
##  5 2019 May     475.  480.
##  6 2019 Jun     465.  487.
##  7 2019 Jul     469.  489.
##  8 2019 Aug     482.  520.
##  9 2019 Sep     457.  498.
## 10 2019 Oct     472.  536.
## # ℹ 14 more rows
```

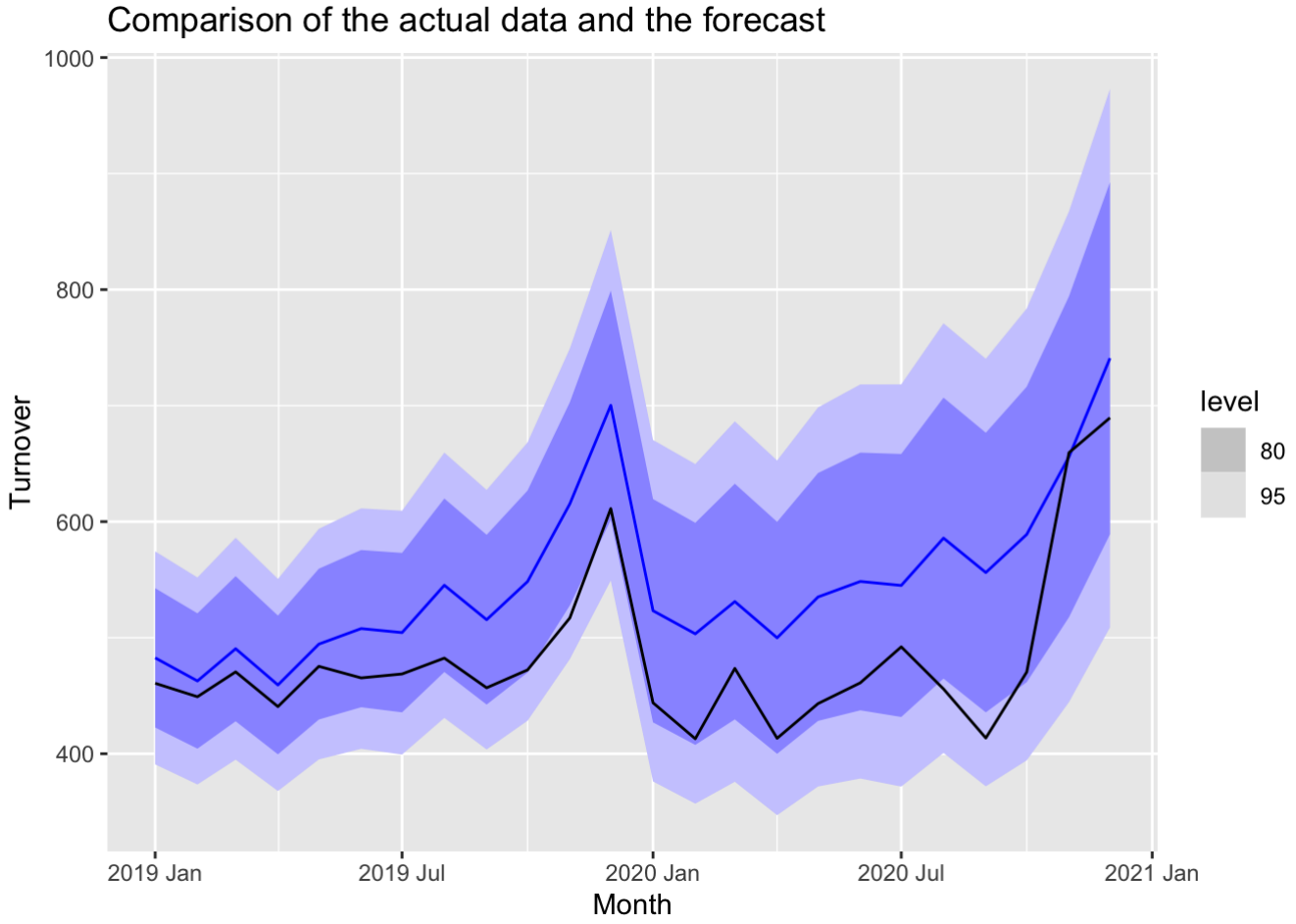
We will compare the mean (point forecast) and the actual numbers. As we can see from the graph and also the numbers, the ARIMA model is not doing well in forecasting the next 2 years, since the actual numbers and arima model's numbers are quite different, that the forecast is higher than the actual value a lot. There are only few months in early 2019 that the forecast is accurate. The metrics also show that there's a lot of difference in the forecast compared to the actual data.

# ETS model

```
# Check the accuracy in forecasting of ETS model using new data
ets_model |>
  forecast(h = 24, level = 80) |>
  accuracy(data_compare)
```

```
## # A tibble: 1 × 12
##   .model State    Industry .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1
##   <chr>  <chr>    <chr>    <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 ets    Victoria Other r… Test  -64.2  74.7  64.5 -13.8  13.8   NaN   NaN 0.536
```

```
# Plot a graph the show the difference between actual data and the forecast
ets_model |>
  forecast(h = 24, level = 80) |>
  autoplot(data_compare) +
  labs(title = "Comparison of the actual data and the forecast")
```



```
# Select the point forecast of the ETS model
ets_mean <- ets_model |>
  forecast(h = 24, level = 80) |>
  hilo() |>
  select(.mean)
```

```
# # Compare the actual numbers and the point forecast
full_join(data_compare, ets_mean, by = "Month") |>
  select(-c(State, Industry))
```

```
## # A tsibble: 24 x 3 [1M]
##       Month Turnover .mean
##       <mth>    <dbl> <dbl>
##  1 2019 Jan     461.  483.
##  2 2019 Feb     449.  463.
##  3 2019 Mar     470.  490.
##  4 2019 Apr     441.  459.
##  5 2019 May     475.  494.
##  6 2019 Jun     465.  508.
##  7 2019 Jul     469.  504.
##  8 2019 Aug     482.  545.
##  9 2019 Sep     457.  516.
## 10 2019 Oct     472.  548.
## # ℹ 14 more rows
```

We will compare the mean (point forecast) and the actual numbers. ETS model also does not perform well in this forecast, since the graph and also the table above show a large difference between the forecast and the actual numbers. The forecast numbers are higher than the actual numbers. There is a big gap between two lines in the graph. The accuracy metrics also determine that this model is not doing well in forecasting, since the errors are quite high.

Despite the accuracy these two model have in previous forecast, both of these does not perform really well in forecasting the next 2 years.

# Question 8: A discussion of benefits and limitations of the models for your data. [3 marks]

Despite the accuracy these two model have in previous forecast, both of these does not perform really well in forecasting the next 2 years.

1. ARIMA(3,0,2)(0,1,1)[12] with drift: Benefits: The ARIMA model allows for the inclusion of autoregressive (AR) and moving average (MA) components, as well as seasonal differencing. This flexibility allows the model to capture different patterns and dependencies in the data. The inclusion of seasonal parameters in the model ([12] in this case) helps capture and forecast seasonal patterns in the data. A drift term accounts for a constant trend or long-term growth/decline in the data. In this case, both seasonality, (constant) trend of the data are captured in the structure of the model.

Limitations: This ARIMA model is complex with multiple AR and MA terms. Interpreting the coefficients and understanding their impact on the data can be challenging.

2. ETS(M,A,A): Benefits: The ETS model is based on exponential smoothing methods, which provide a simple and intuitive approach to capturing trends and seasonality in the data. The model automatically

adjusts the smoothing parameters (alpha, beta, gamma) to the changing patterns in the data, allowing for adaptability to different time series characteristics.

Limitations: The performance of ETS models relies heavily on accurate estimation of the smoothing parameters. In order to have a good model, we need to estimate the parameters appropriately, and this would be hard and time-consuming. This ETS models assume that the seasonality is additive. In cases where the seasonality is multiplicative, ETS models may not capture it effectively.