FPP Chapter 8 HW: Exponential Smoothing

Alex Ptacek

```r
library(tidyverse)
library(fpp3)
```

Question 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 and 0, and generate forecasts for the next four months.

Answer: Optimal alpha = 0.36. Optimal level = 95,487.5

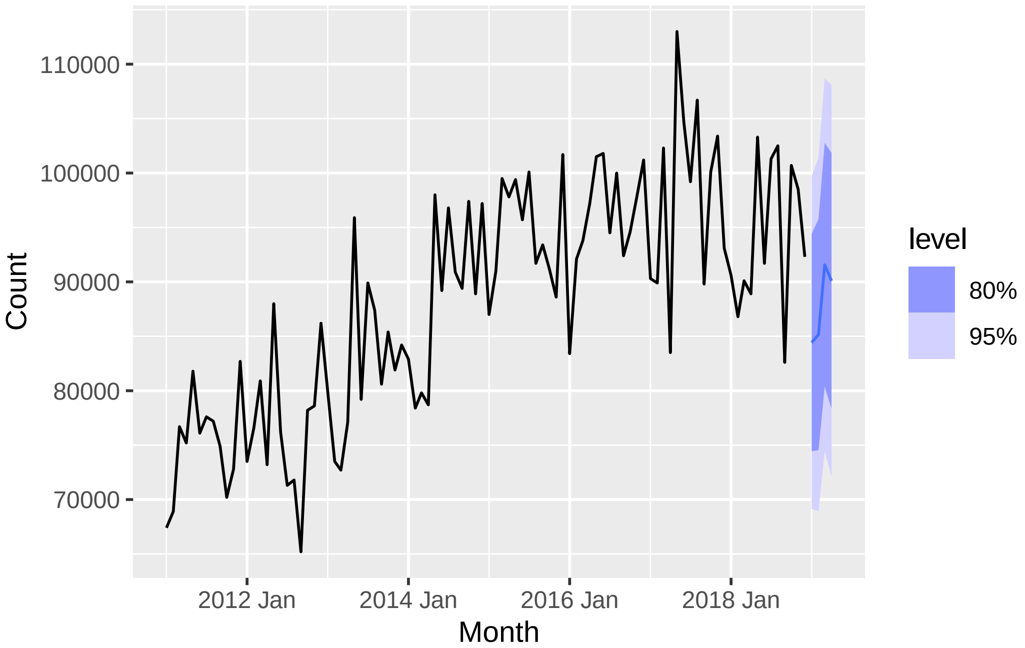
```r
vict_pigs <- aus_livestock |>
  filter(State == "Victoria" & Animal == "Pigs")

#Find our optimal parameters
vict_pigs_fit <- vict_pigs |>
  model(ETS(Count)) |>
  report()
```

```
Series: Count
Model: ETS(A,N,A)
  Smoothing parameters:
    alpha = 0.3579401
    gamma = 0.0001000139

  Initial states:
    l[0]     s[0]    s[-1]     s[-2]     s[-3]     s[-4]     s[-5]   s[-6]
 95487.5 2066.235 6717.311 -2809.562 -1341.299 -7750.173 -8483.418 5610.89
     s[-7]    s[-8]     s[-9]   s[-10]   s[-11]
 -579.8107 1215.464 -2827.091 1739.465 6441.989

  sigma^2:  60742898

     AIC     AICc      BIC 
13545.38 13546.26 13610.24 
```

```
vict_pigs_fit
```

```
# A mable: 1 x 3
# Key:     Animal, State [1]
  Animal State    `ETS(Count)`
  <fct>  <fct>         <model>
1 Pigs   Victoria <ETS(A,N,A)>
```

```
#Generate 4 months of forecasts. Filter year to see forecasts better.
vict_pigs_fit |>
  forecast(h = 4) |>
  autoplot(vict_pigs |> filter(year(Month) > 2010))
```



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

```
#Lower interval
84425 - 1.96*sqrt(60742898)
```

```
[1] 69149.2
```

```
#Upper interval
84425 + 1.96*sqrt(60742898)
```

```
[1] 99700.8
```

```
#Manual calculations match first row of fable calculation
vict_pigs_fit |>
  forecast(h = 4) |>
  hilo(level = 95) |>
  mutate(lower = `95%`$lower, upper = `95%`$upper) |>
  select(lower, .mean, upper)
```

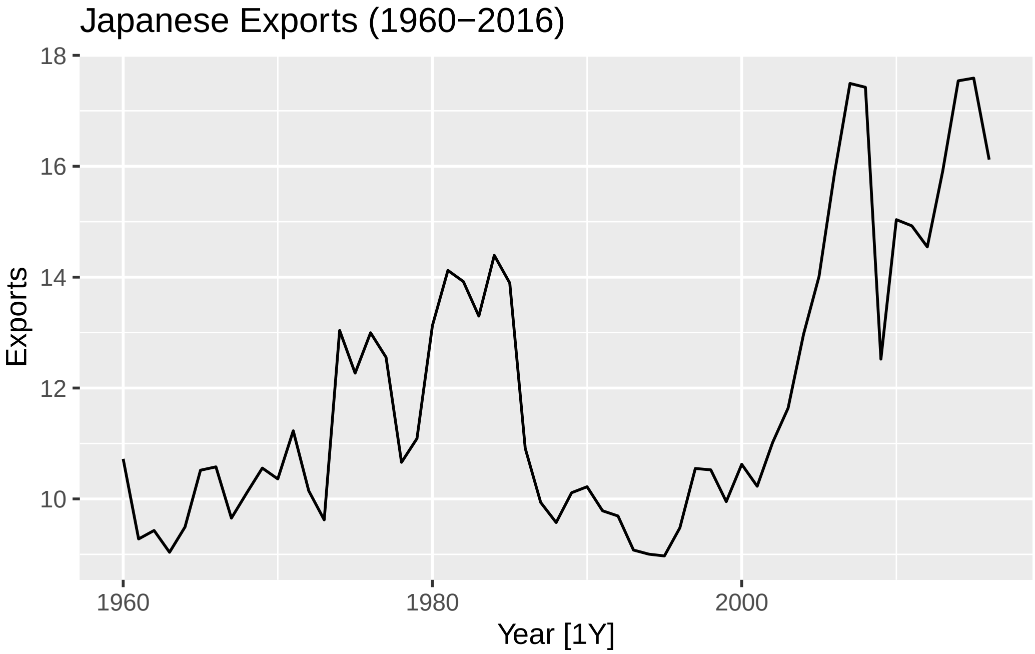
```
# A tsibble: 4 x 4 [1M]
   lower  .mean   upper    Month
   <dbl>  <dbl>   <dbl>    <mth>
1 69149. 84425.  99700. 2019 Jan
2 68933. 85158. 101383. 2019 Feb
3 74446. 91567. 108688. 2019 Mar
4 72125. 90098. 108071. 2019 Apr
```

Question 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.

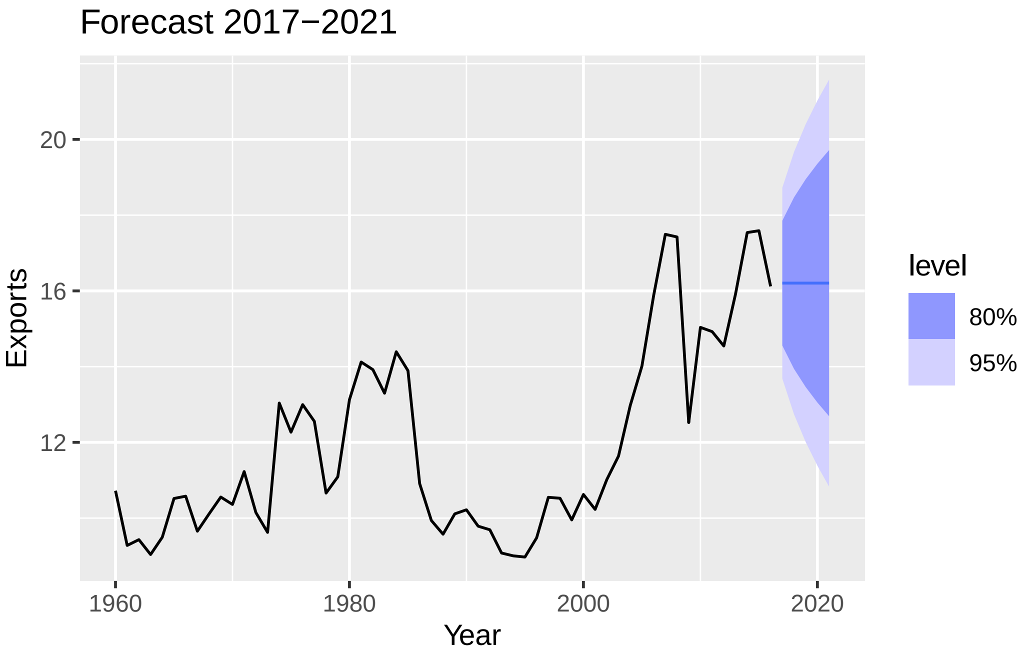
Answer: The Japanese Exports series starts off with an upward trend until 1987, where the Exports troughed for about 15 years before rising rapidly in 2001 and continuing a cyclic pattern. The cyclic pattern seems to be 1-3 years of rapid growth, followed by 2-5 years of plateau, and then 1-3 years of rapid decline.

```
#Removing 2017 because it has NA Exports for Japan
jap_exports <- global_economy |>
  filter(Country == "Japan") |>
  filter(Year != 2017)

jap_exports |>
  autoplot(Exports) +
  labs(title = "Japanese Exports (1960-2016)")
```



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

```
jap_ANN <- jap_exports |>
  model(ETS(Exports ~ error("A") + trend("N") + season("N")))

jap_ANN |>
  forecast(h = 5) |>
  autoplot(jap_exports) +
  ggtitle("Forecast 2017-2021")
```



c. Compute the RMSE values for the training data.

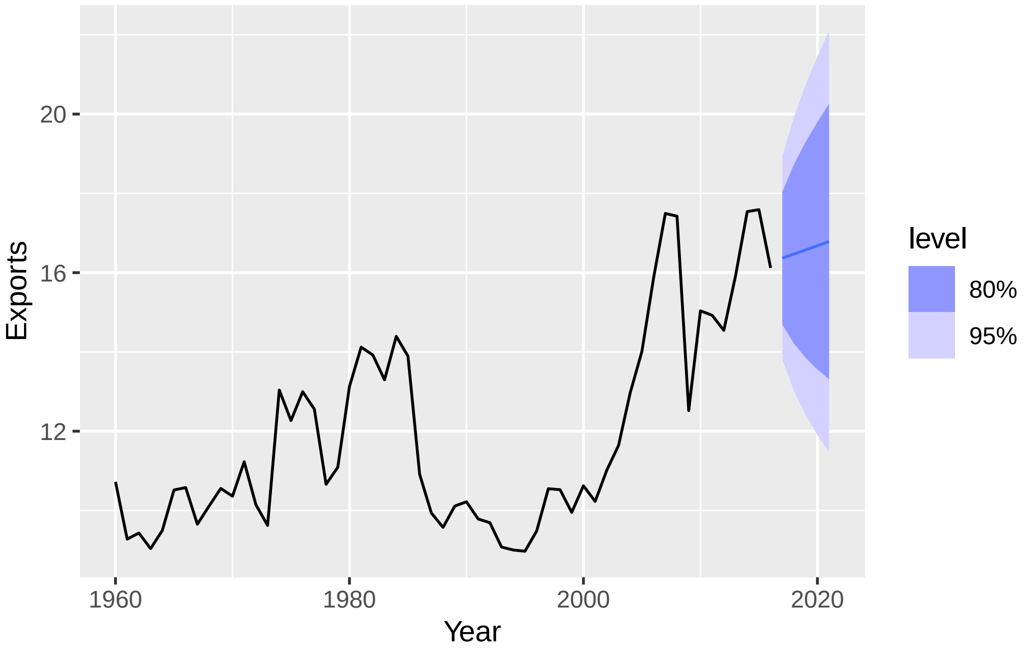
```
ANN_RMSE <- jap_ANN |>
  accuracy() |>
  pull(RMSE)
ANN_RMSE
```

```
[1] 1.263342
```

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

Answer: Plotting the AAN forecast, we can notice an upward trends vs. the ANN model, which had a nearly horizontal line. The RMSE is slightly lower in the ETS(A,A,N) model, but extremely close. The alpha value for AAN is lower than ANN, so the AAN model is giving higher weight to earlier observations. Beta, only given for AAN, is near 0, so it's not as responsive to changes in y, and therefore captures the general upward trend of the time series. The AAN method seems more useful for this data, because we do see an overall upward trend in the series, while the Simple ANN method only accounts for the errors in the data.

```
jap_AAN <- jap_exports |>
  model(ETS(Exports ~ error("A") + trend("A") + season("N")))

#Plot AAN forecast
jap_AAN |>
  forecast(h = 5) |>
  autoplot(jap_exports)
```



```
#Get RMSE fro AAN model
jap_AAN |>
  accuracy() |>
  pull(RMSE)
```

```
[1] 1.259002
```

```
#Compare coefficients of ANN vs AAN
tidy(jap_ANN)
```

```
# A tibble: 2 x 4
  Country .model                                                  term   estimate
  <fct>   <chr>                                                   <chr>     <dbl>
```

```
1 Japan   "ETS(Exports ~ error(\"A\") + trend(\"N\") + season(\"~ alpha    0.941
2 Japan   "ETS(Exports ~ error(\"A\") + trend(\"N\") + season(\"~ l[0]    10.6
```

```
#Compare coefficients of ANN vs AAN
tidy(jap_AAN)
```

```
# A tibble: 4 x 4
  Country .model                                                  term   estimate
  <fct>   <chr>                                                   <chr>     <dbl>
1 Japan   "ETS(Exports ~ error(\"A\") + trend(\"A\") + season(\"~ alpha  9.09e-1
2 Japan   "ETS(Exports ~ error(\"A\") + trend(\"A\") + season(\"~ beta   1.00e-4
3 Japan   "ETS(Exports ~ error(\"A\") + trend(\"A\") + season(\"~ l[0]   1.05e+1
4 Japan   "ETS(Exports ~ error(\"A\") + trend(\"A\") + season(\"~ b[0]   1.05e-1
```

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.

Answer: Overall, the intervals are nearly the same using RMSE vs. standard deviation with fable.

```
#Observe lower, mean, and upper levels of AAN prediction interval
jap_AAN |>
  forecast(h = 5) |>
  hilo(95) |>
  mutate(lower = `95%`$lower, upper = `95%`$upper) |>
  select(lower, .mean, upper)
```

```
# A tsibble: 5 x 4 [1Y]
  lower .mean upper  Year
  <dbl> <dbl> <dbl> <dbl>
1  13.8  16.4  18.9  2017
2  13.0  16.5  19.9  2018
3  12.4  16.6  20.7  2019
4  11.9  16.7  21.5  2020
5  11.5  16.8  22.1  2021
```

```
AAN_RMSE <- accuracy(jap_AAN) |>
  pull(RMSE)

#AAN lower
15.6 - 1.96*AAN_RMSE
```

```
[1] 13.13236
```

```
#AAN upper
15.6 + 1.96*AAN_RMSE
```

```
[1] 18.06764
```

```
#Observe lower, mean, and upper levels of ANN prediction interval
jap_ANN |>
  forecast(h = 5) |>
  hilo(95) |>
  mutate(lower = `95%`$lower, upper = `95%`$upper) |>
  select(lower, .mean, upper)
```

```
# A tsibble: 5 x 4 [1Y]
  lower .mean upper  Year
  <dbl> <dbl> <dbl> <dbl>
1  13.7  16.2  18.7  2017
2  12.7  16.2  19.7  2018
3  12.0  16.2  20.4  2019
4  11.4  16.2  21.0  2020
5  10.8  16.2  21.6  2021
```

```
#ANN lower
16.2 - 1.96*ANN_RMSE
```

```
[1] 13.72385
```

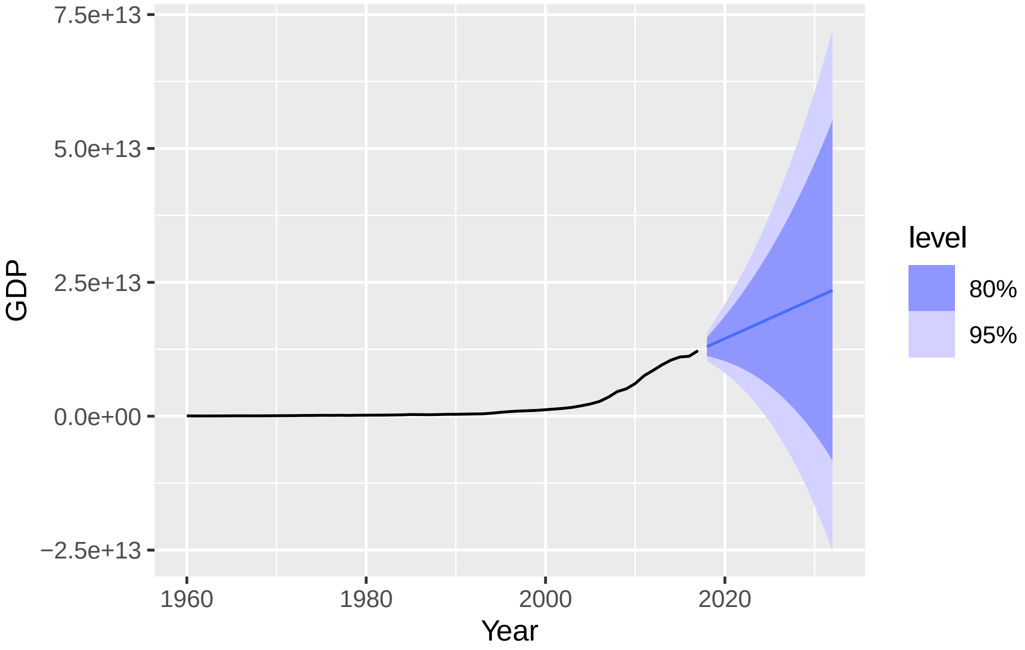
```
#ANN upper
16.2 + 1.96*ANN_RMSE
```

```
[1] 18.67615
```

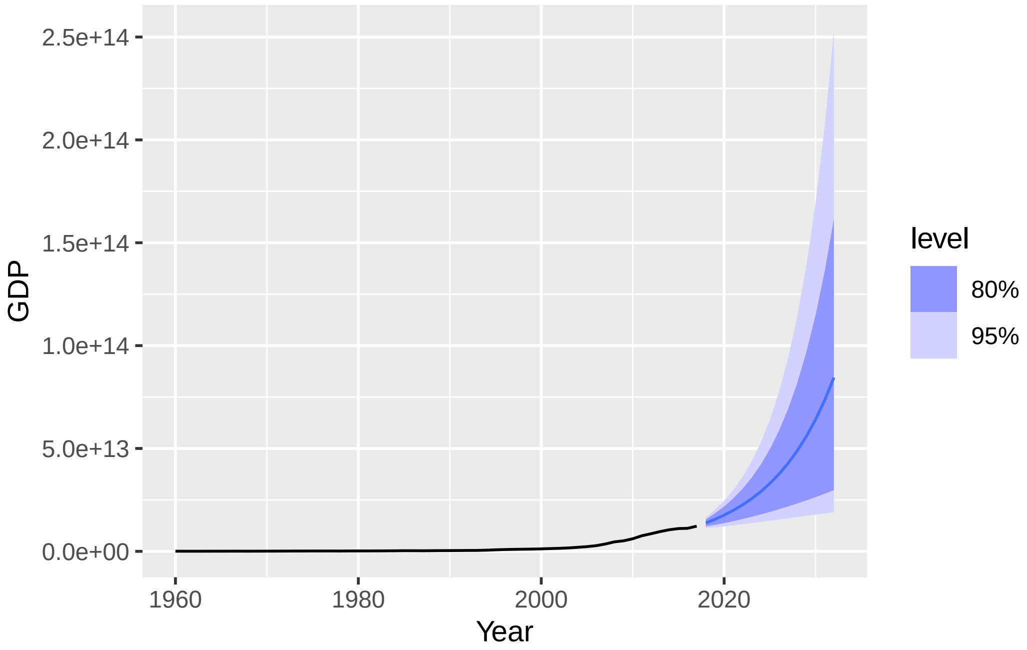
Question 8.6: Forecast the Chinese GDP from the global_economy data set using an ETS model. Experiment with the various options in the ETS() function to see how much the forecasts change with damped trend, or with a Box-Cox transformation. Try to develop an intuition of what each is doing to the forecasts.

[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.]

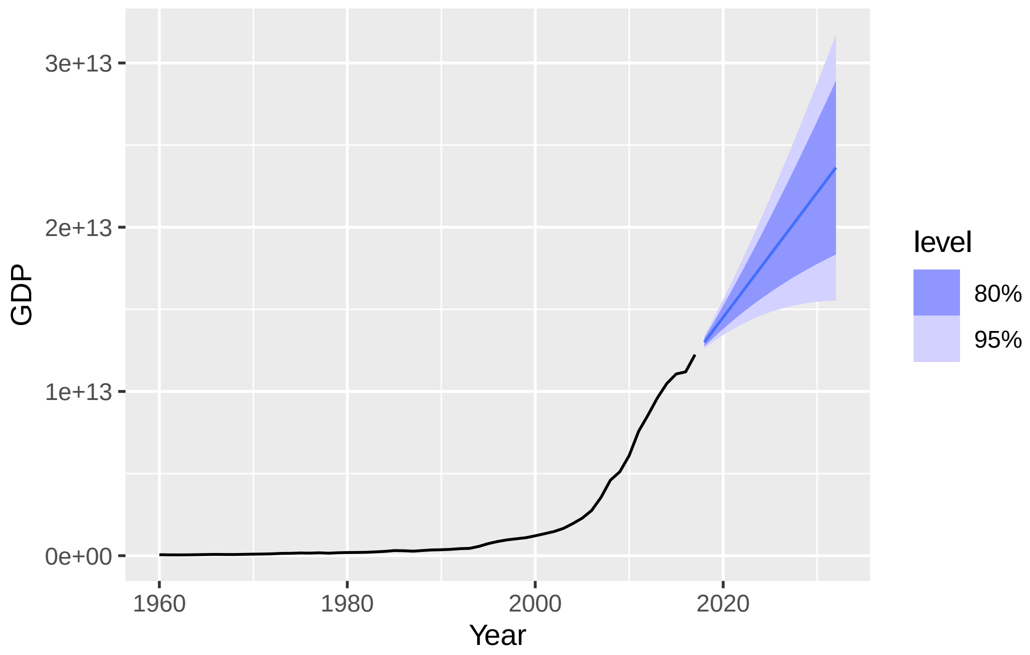
```r
china_gdp <- global_economy |>
  filter(Country == "China")

#MAN model is automatically picked
china_gdp |>
  model(ETS(GDP)) |>
  forecast(h = 15) |>
  autoplot(china_gdp)
```



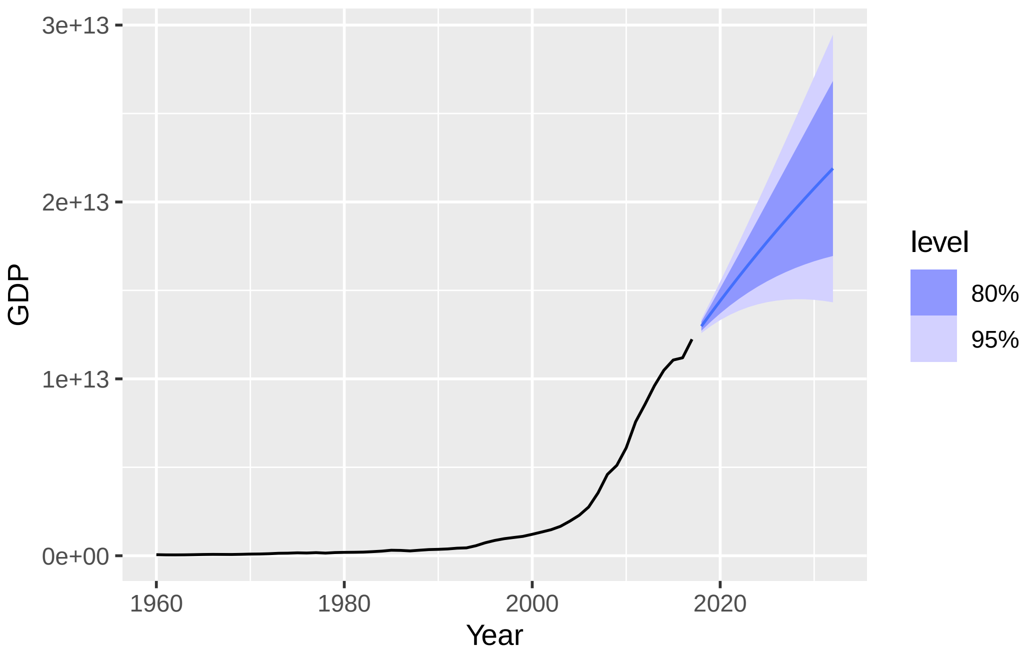
```r
china_gdp |>
  model(ETS(log(GDP))) |>
  forecast(h = 15) |>
  autoplot(china_gdp)
```



```
china_gdp |>
  model(ETS(GDP ~ error("A") + trend("A") + season("N"))) |>
  forecast(h = 15) |>
  autoplot(china_gdp)
```



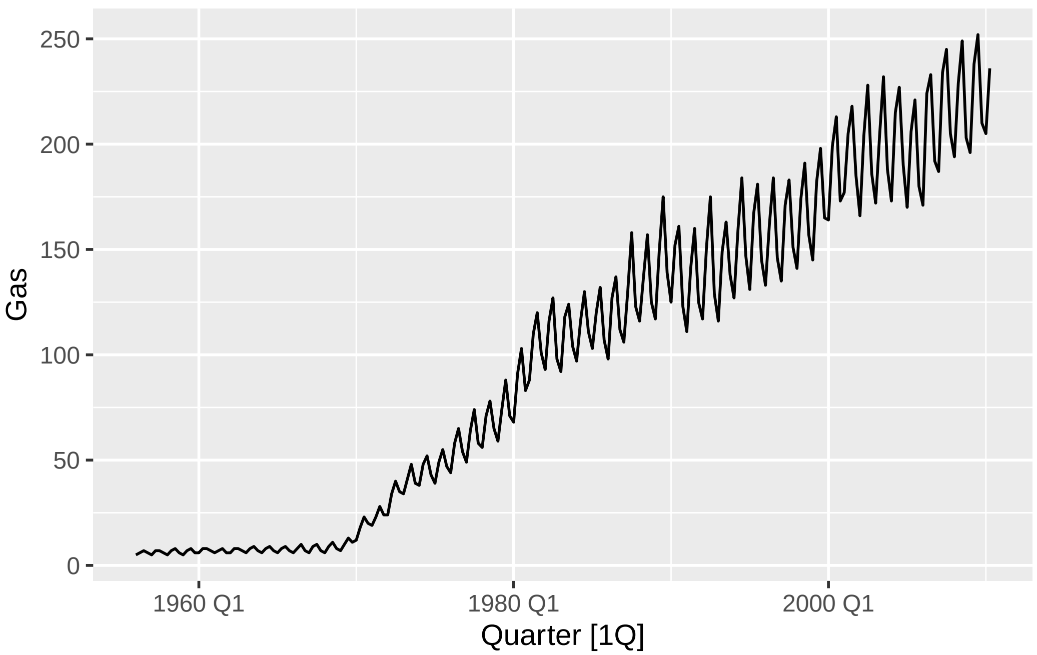
```r
china_gdp |>
  model(ETS(GDP ~ error("A") + trend("Ad") + season("N"))) |>
  forecast(h = 15) |>
  autoplot(china_gdp)
```



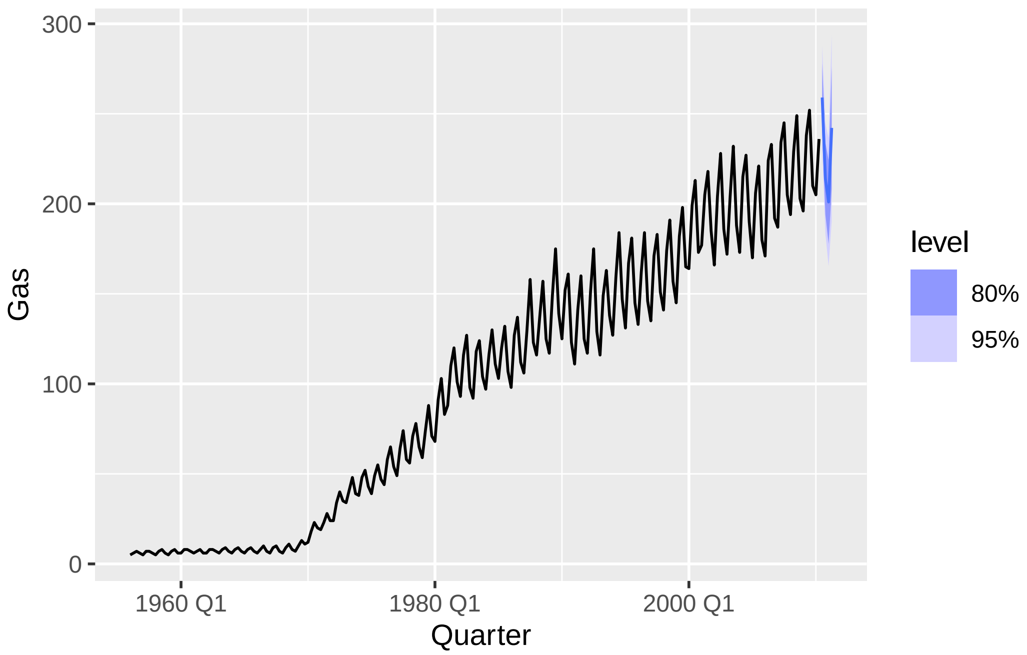
Question 8.7: Find an ETS model for the Gas data from aus_production and forecast the next few years. Why is multiplicative seasonality necessary here? Experiment with making the trend damped. Does it improve the forecasts?

Answer: The ETS function automatical picks an ETS(M,A,M) model. Multiplicative seasonality is necessary here because the variation is increasing relative to the level of Gas production. Additionally, we need to make sure the residuals of our fitted model are normally distributed and homoschodastic. This is achieved (mostly) with the MAM model, but not an AAA model. Based on the RMSE and AICc, the auto-fable-computed model (MAM) has better acuracy than a damped model (M,Ad,M).

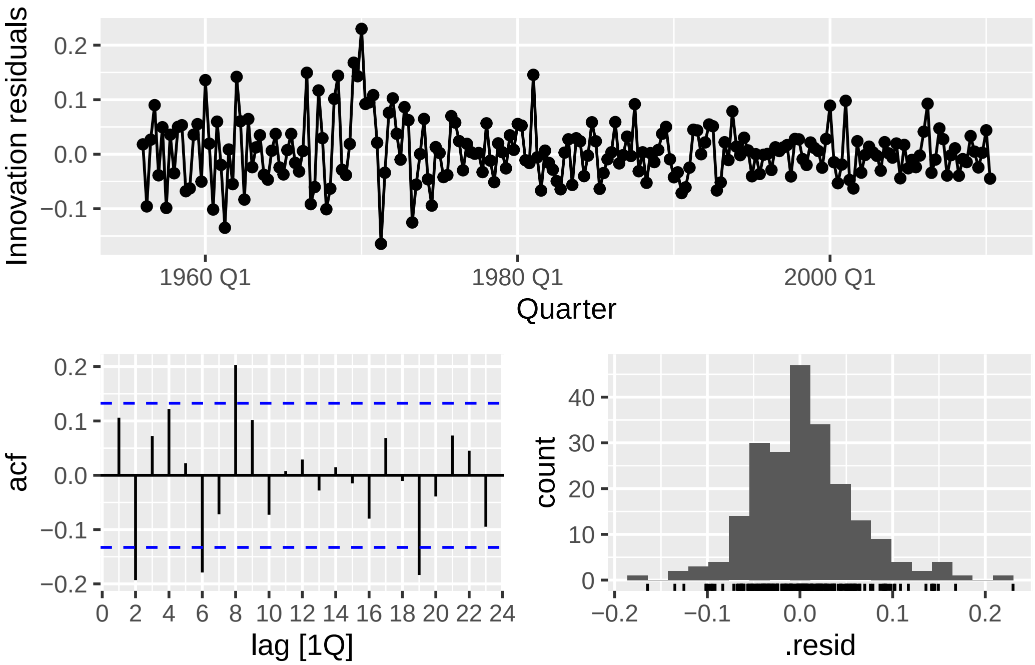
```
#Original Gas Time Series
aus_production |>
  autoplot(Gas)
```



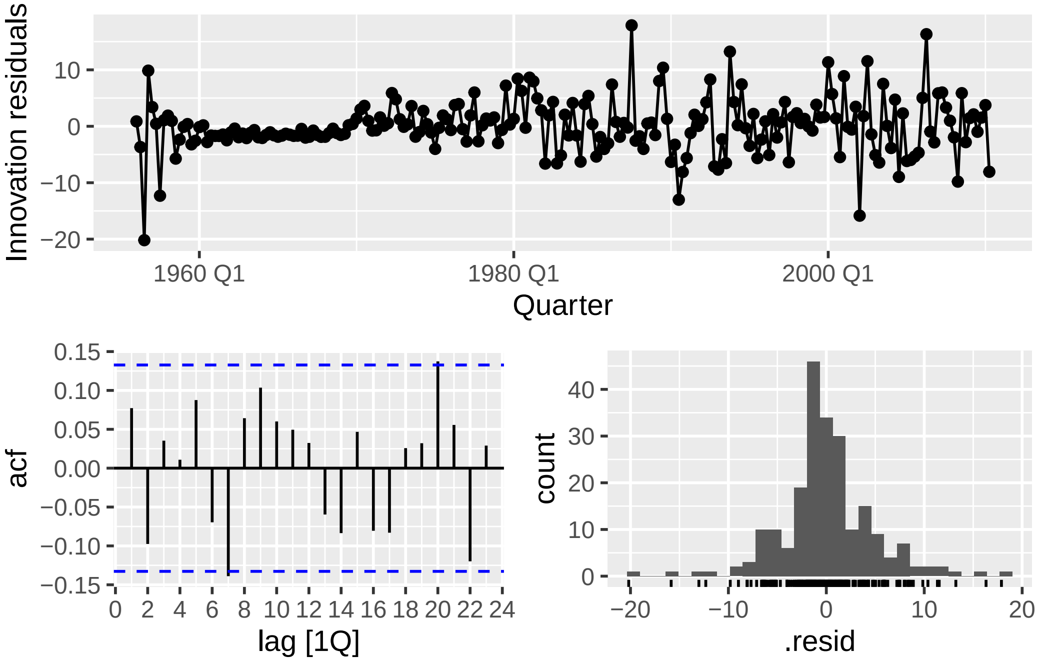
```
#ETS automatically picks MAM model
auto_fit <- aus_production |>
  model(ETS(Gas))
auto_fit |>
  forecast(h = 4) |>
  autoplot(aus_production)
```



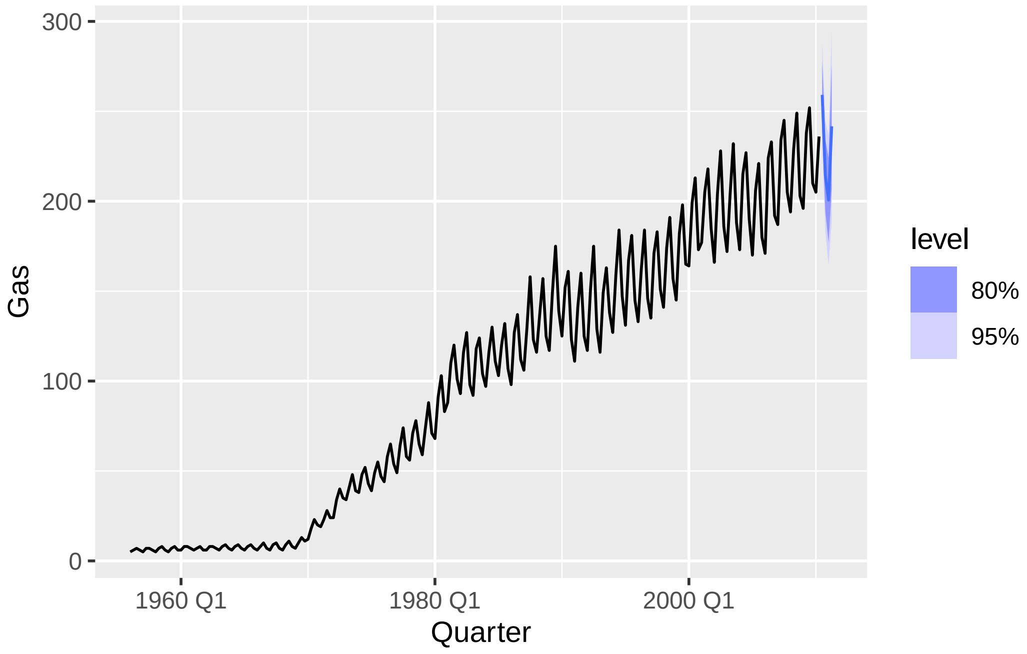
```
#Observe homoschodasticity of MAM model
aus_production |>
  model(ETS(Gas)) |>
  gg_tsresiduals()
```



```r
#Observe heteroschodasticity of AAA model
aus_production |>
  model(ETS(Gas ~ error("A") + trend("A") + season("A"))) |>
  gg_tsresiduals()
```



```
#Plot damped ETS model
damp_fit <- aus_production |>
  model(ETS(Gas ~ error("M") + trend("Ad") + season("M")))
damp_fit |>
  forecast(h = 4) |>
  autoplot(aus_production)
```



```
#Get RMSE for models
accuracy(auto_fit) |>
  pull(RMSE)
```

```
[1] 4.595113
```

```
#Get RMSE for models
accuracy(damp_fit) |>
  pull(RMSE)
```

```
[1] 4.59184
```

```
#Get AICc for models
glance(auto_fit) |>
  pull(AICc)
```

```
[1] 1681.794
```

```
#Get AICc for models
glance(damp_fit) |>
  pull(AICc)
```

```
[1] 1685.091
```

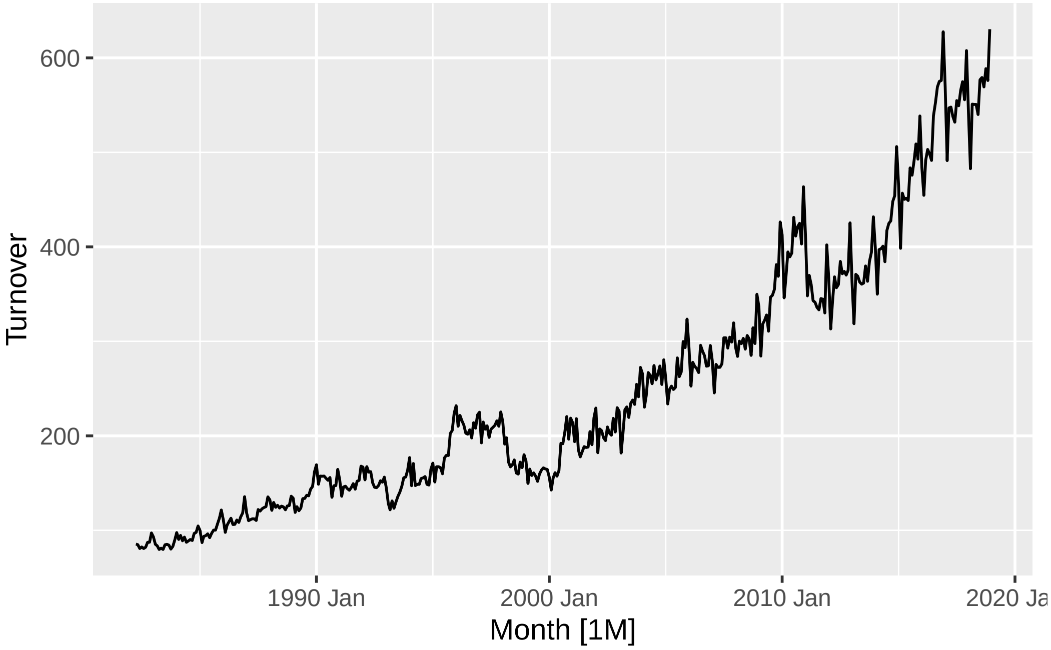
Question 8.8: Recall your retail time series data (from Exercise 7 in Section 2.10).

```
set.seed(624)
myseries <- aus_retail |>
  filter(`Series ID` == sample(aus_retail$`Series ID`, 1))
```

a. Why is multiplicative seasonality necessary for this series?

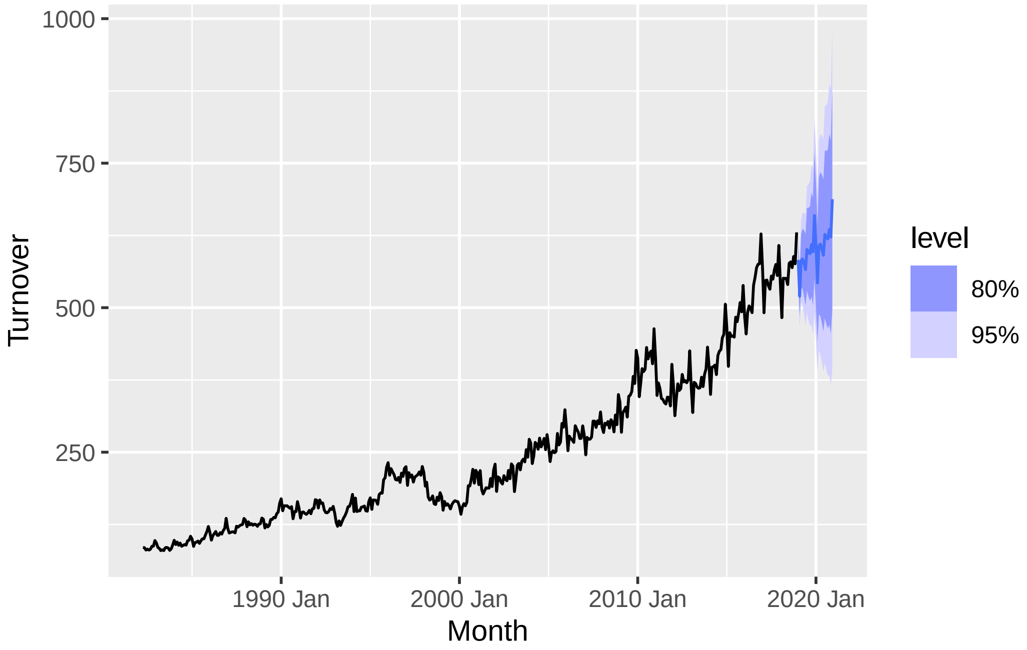
Answer: Multiplicative seasonality is necessary for this series because the variance is increasing relative to the level of Turnover.

```
myseries |>
  autoplot(Turnover)
```

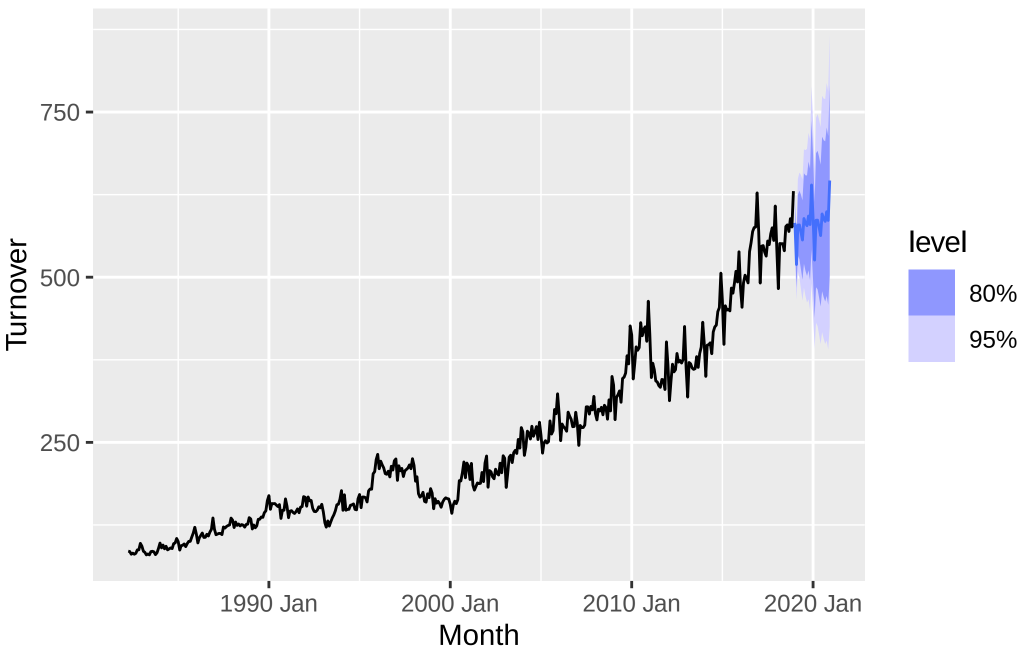


b. Apply Holt-Winters' multiplicative method to the data. Experiment with making the trend damped.

```
#Holt-Winters' Multiplicative method
mam_myseries <- myseries |>
  model(ETS(Turnover ~ error("M") + trend("A") + season("M")))
mam_myseries |>
  forecast() |>
  autoplot(myseries)
```



```
#Damped version
madm_myseries <- myseries |>
  model(ETS(Turnover ~ error("M") + trend("Ad") + season("M")))
madm_myseries |>
  forecast() |>
  autoplot(myseries)
```



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

Answer: The RMSE of one-step cross-validation is equivalent to the AIC, so we can simply check the AIC (or the AICc for small samples). I prefer the Damped version [ETS(M,Ad,M)] because is has lower AICc.

```
glance(mam_myseries) |>
  pull(AICc)
```

```
[1] 4635.968
```

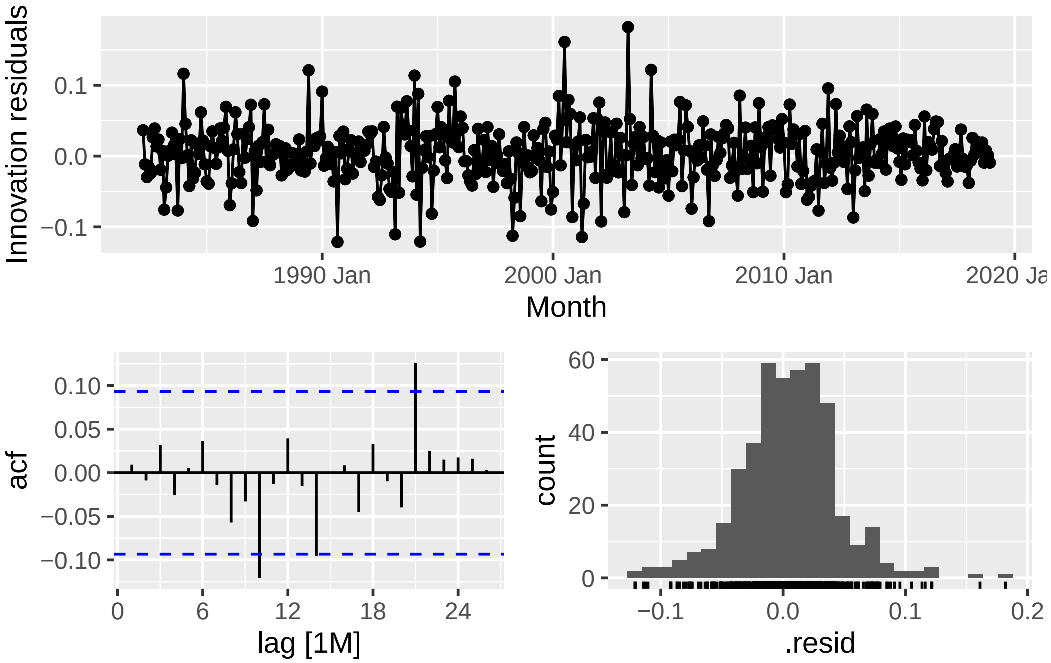
```
glance(madm_myseries) |>
  pull(AICc)
```

```
[1] 4626.859
```

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

Answer: There are a couple spikes in the ACF plot of residuals, but they are not at seasonal spikes, so this looks like white noise. Nonetheless, I opted to run the ljung-box text and I confirmed that the residuals are white noise.

```
gg_tsresiduals(madm_myseries)
```



```
#Set lag = m*2 = 24
augment(madm_myseries) |>
  features(.innov, ljung_box, lag = 24)
```

```
# A tibble: 1 x 5
  State           Industry                 .model                   lb_stat lb_pvalue
  <chr>           <chr>                    <chr>                      <dbl>     <dbl>
1 New South Wales Takeaway food services "ETS(Turnover ~ erro~    25.3     0.392
```

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?

Answer: Yes! By fitting an auto-fable-computed ETS model, we decreased RMSE from 96.8 (SNAIVE) to 70.4 (Exponential Smoothing).

i. Seasonal Naive approach from 5.7

```r
train_myseries <- myseries |> filter(year(Month) <= 2010)

fit_sn <- train_myseries |> model(SNAIVE(Turnover))

fit_sn |>
  forecast(h = "8 years") |>
  accuracy(myseries)
```

```
# A tibble: 1 x 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 SNAIVE(T~ New ~ Takeawa~ Test   48.6  96.8  79.5  7.67  16.3  4.14  3.71 0.964
```

ii. ETS Model

```r
fit_ets <- train_myseries |> model(ETS(Turnover))

fit_ets |>
  forecast(h = "8 years") |>
  accuracy(myseries)
```

```
# A tibble: 1 x 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(Turn~ New ~ Takeawa~ Test  -59.3  70.4  60.9 -15.0  15.3  3.17  2.69 0.905
```

Question 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?

Answer: The resulting RMSE of 88.9 is inbetween our SNAIVE and ETS models' RMSE, but still significantly worse than the ETS. This is likely due to the decomposition model using a seasonal naive model on the seasonal component (effectively, gamma = 1), while using ETS on the seasonally adjusted component. The ETS model is accounting for more of the variation in seasonality and using weighted parameters accordingly. Looking at our gamma coefficient for our ETS model, we can see it's near-zero, so the model is effectively using a mean seasonality model for the seasonal component of `myseries`.

```
#Observe near-zero lambda, which means can take log of Turnover
myseries |>
  features(Turnover, features = guerrero)
```

```
# A tibble: 1 x 3
  State           Industry               lambda_guerrero
  <chr>           <chr>                            <dbl>
1 New South Wales Takeaway food services         0.00214
```

```
fit_decomp <- train_myseries |>
  model(decomposition_model(STL(log(Turnover)), ETS(season_adjust)))
fit_decomp |>
  forecast(h = "8 years") |>
  accuracy(myseries)
```

```
# A tibble: 1 x 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 decompos~ New ~ Takeawa~ Test  -85.1  88.9  85.1 -19.9  19.9  4.43  3.40 0.734
```

```
tidy(fit_ets) |>
  filter(term == "gamma")
```

```
# A tibble: 1 x 5
  State           Industry               .model         term  estimate
  <chr>           <chr>                  <chr>          <chr>    <dbl>
1 New South Wales Takeaway food services ETS(Turnover) gamma 0.000119
```