DATA624: Homework 3

Donald Butler

2023-09-23

Contents

Homework 3

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

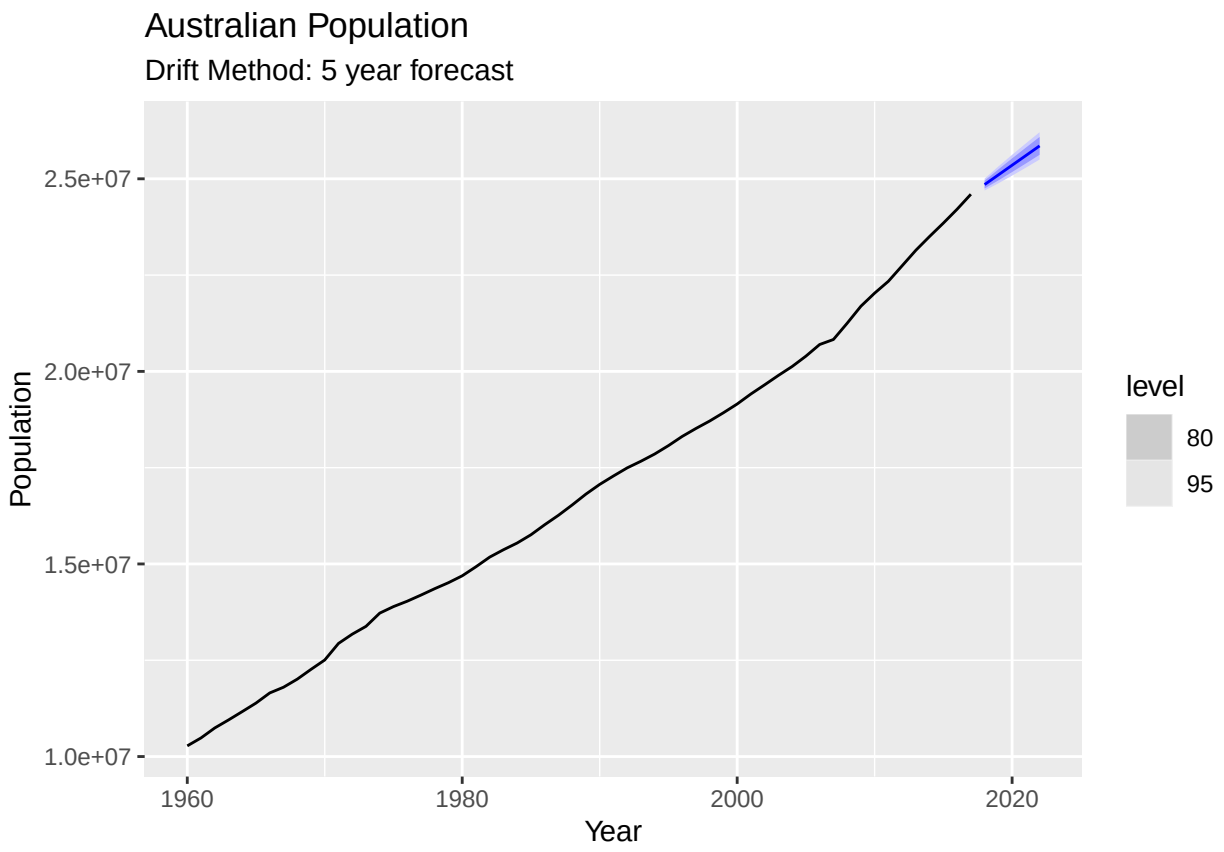
Exercise 5.1

> Produce forecasts for the following series using whichever of `NAIVE(y)`, `SNAIVE(y)` or `RW(y ~ drift())` is more appropriate in each case:

Australian Population (`global_economy`)

Use of the `RW(y ~ drift())` is most appropriate because the data is trending strictly increasing and has no seasonality component.

```
global_economy |>
  filter(Country == 'Australia') |>
  model(RW(Population ~ drift())) |>
  forecast(h = 5) |>
  autoplot(global_economy) +
  labs(title = 'Australian Population',
       subtitle = 'Drift Method: 5 year forecast')
```

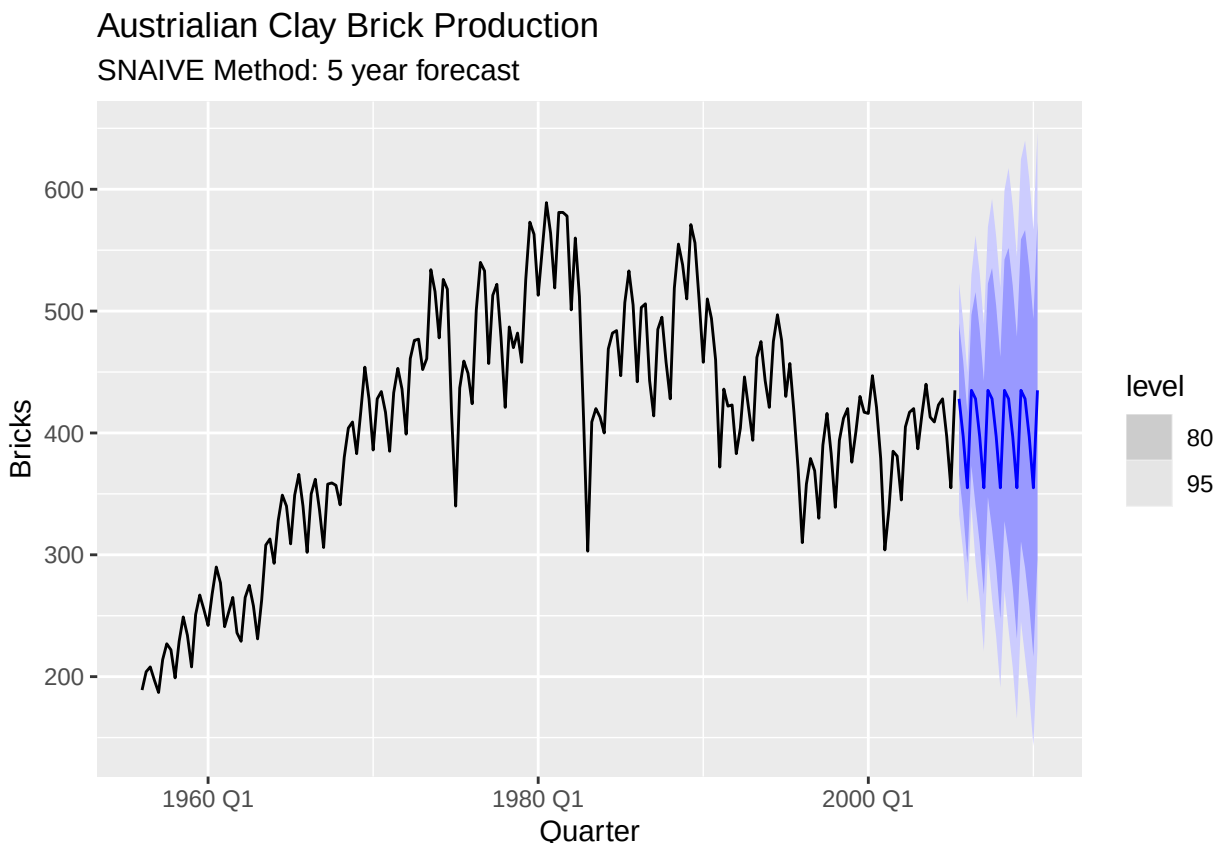


Bricks (`aus_production`)

Use of the `SNAIVE` method is most appropriate because the data is highly seasonal.

```
aus_production |>
  filter(!is.na(Bricks)) |>
  model(SNAIVE(Bricks)) |>
  forecast(h = '5 years') |>
  autoplot(aus_production) +
  labs(title = 'Austrialian Clay Brick Production',
       subtitle = 'SNAIVE Method: 5 year forecast')
```

```
## Warning: Removed 20 rows containing missing values (‘geom_line()‘).
```

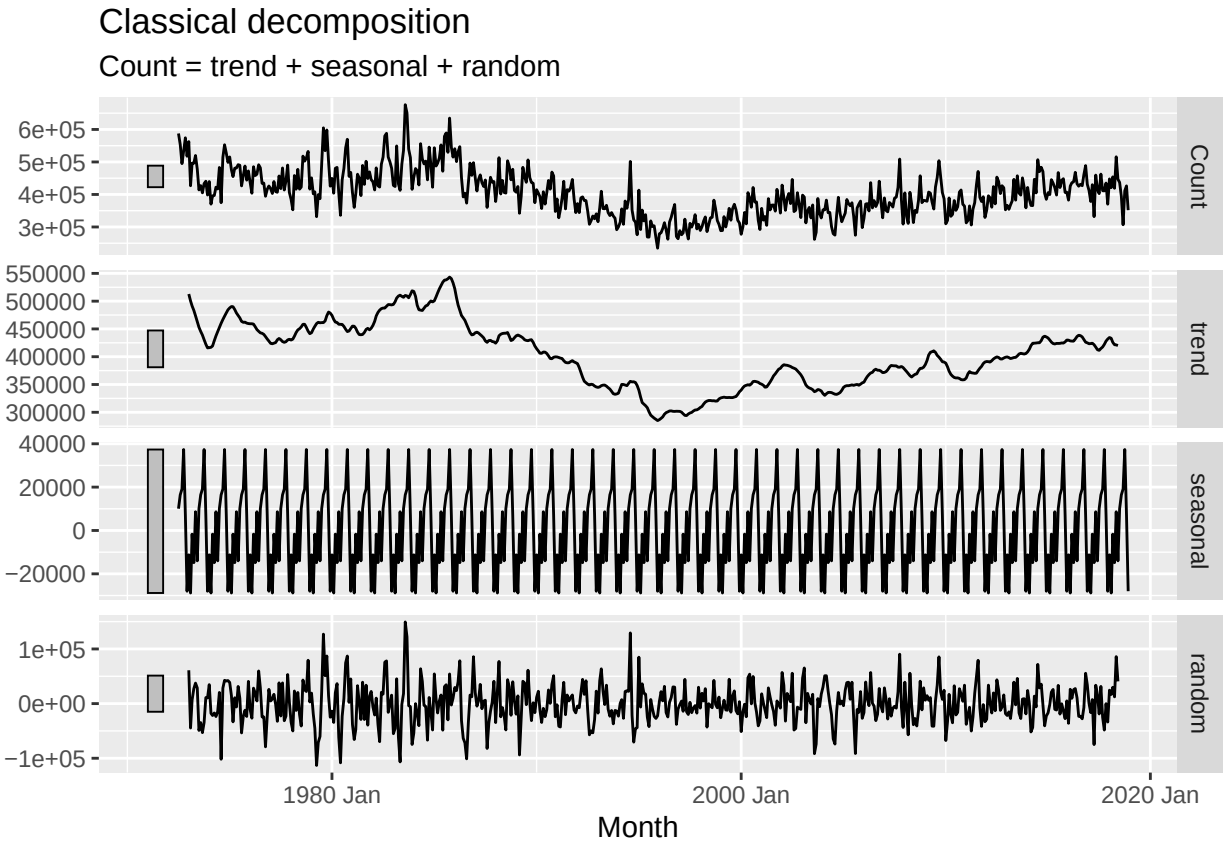


NSW Lambs (`aus_livestock`)

Looking at the classical decomposition, we can see that the seasonal component is insignificant compared to the trend and random components.

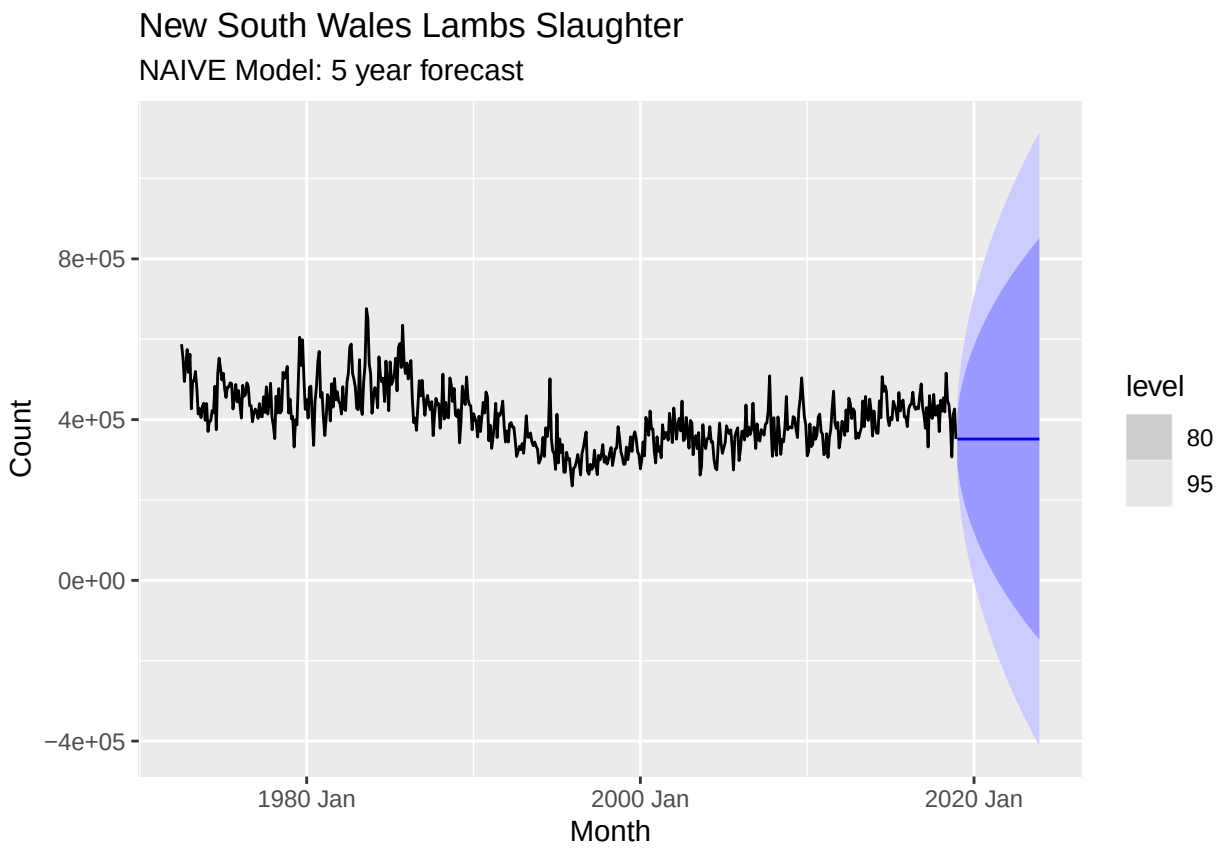
```
aus_livestock |>
  filter(Animal == 'Lambs',
         State == 'New South Wales',
         !is.na(Count)) |>
  model(classical_decomposition(Count, type = 'additive')) |>
  components() |>
  autoplot()
```

```
## Warning: Removed 6 rows containing missing values (`geom_line()`).
```



Use of the `NAIVE` method is appropriate because there is no seasonality to the data and the trend is not linear.

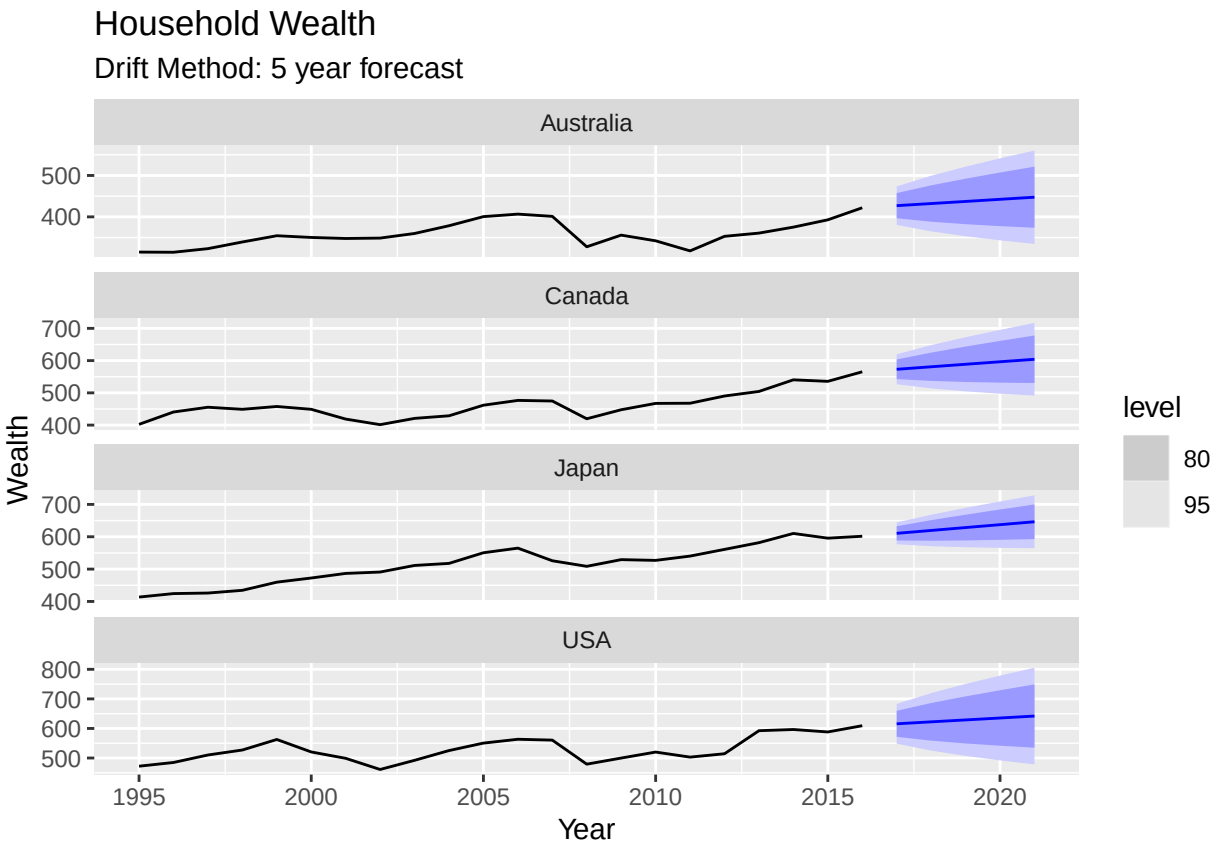
```
aus_livestock |>
  filter(Animal == 'Lambs',
         State == 'New South Wales',
         !is.na(Count)) |>
  model(NAIVE(Count)) |>
  forecast(h = '5 years') |>
  autoplot(aus_livestock) +
  labs(title = 'New South Wales Lambs Slaughter',
       subtitle = 'NAIVE Model: 5 year forecast')
```



Household Wealth (`hh_budget`)

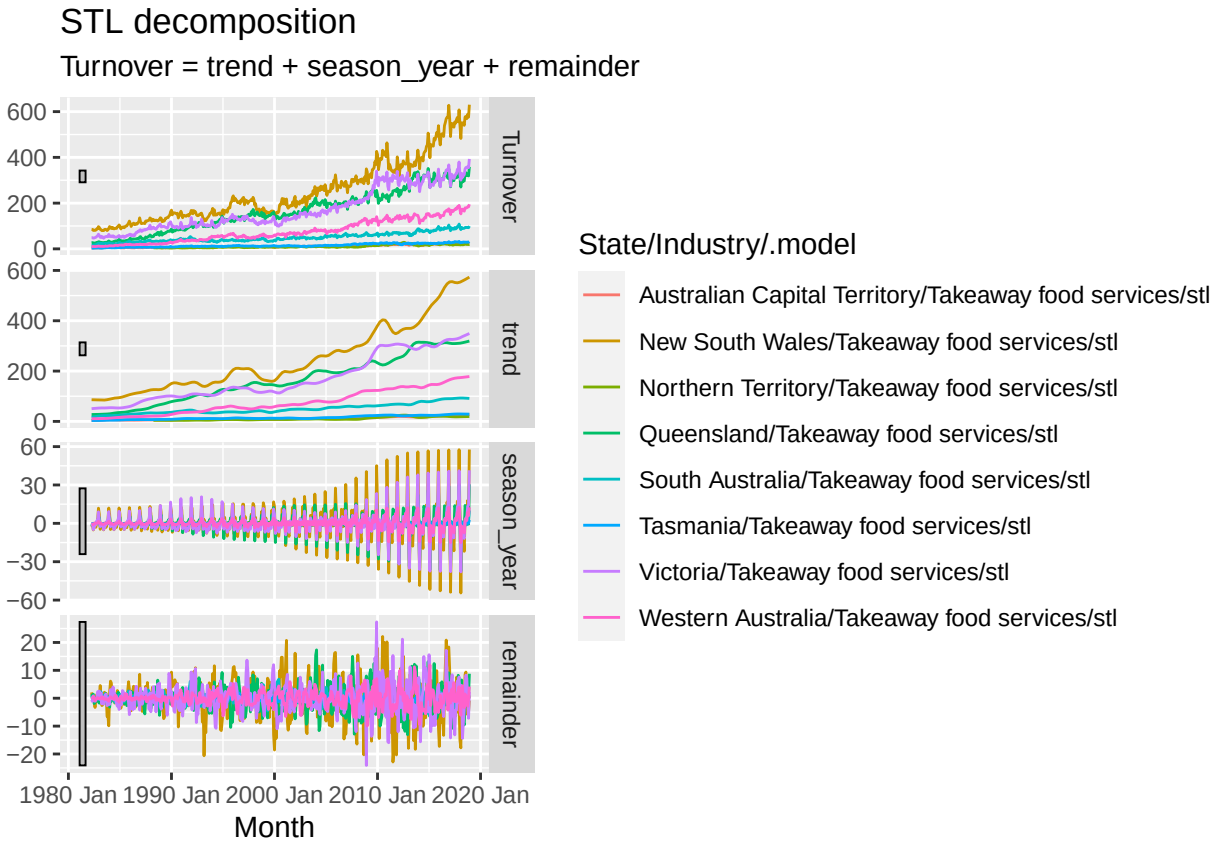
Use of the `RW(y ~ drift())` is most appropriate the data has no seasonality and is generally increasing.

```
hh_budget |>
  filter(!is.na(Wealth)) |>
  model(RW(Wealth ~ drift())) |>
  forecast(h = '5 years') |>
  autoplot(hh_budget) +
  labs(title = 'Household Wealth',
       subtitle = 'Drift Method: 5 year forecast')
```



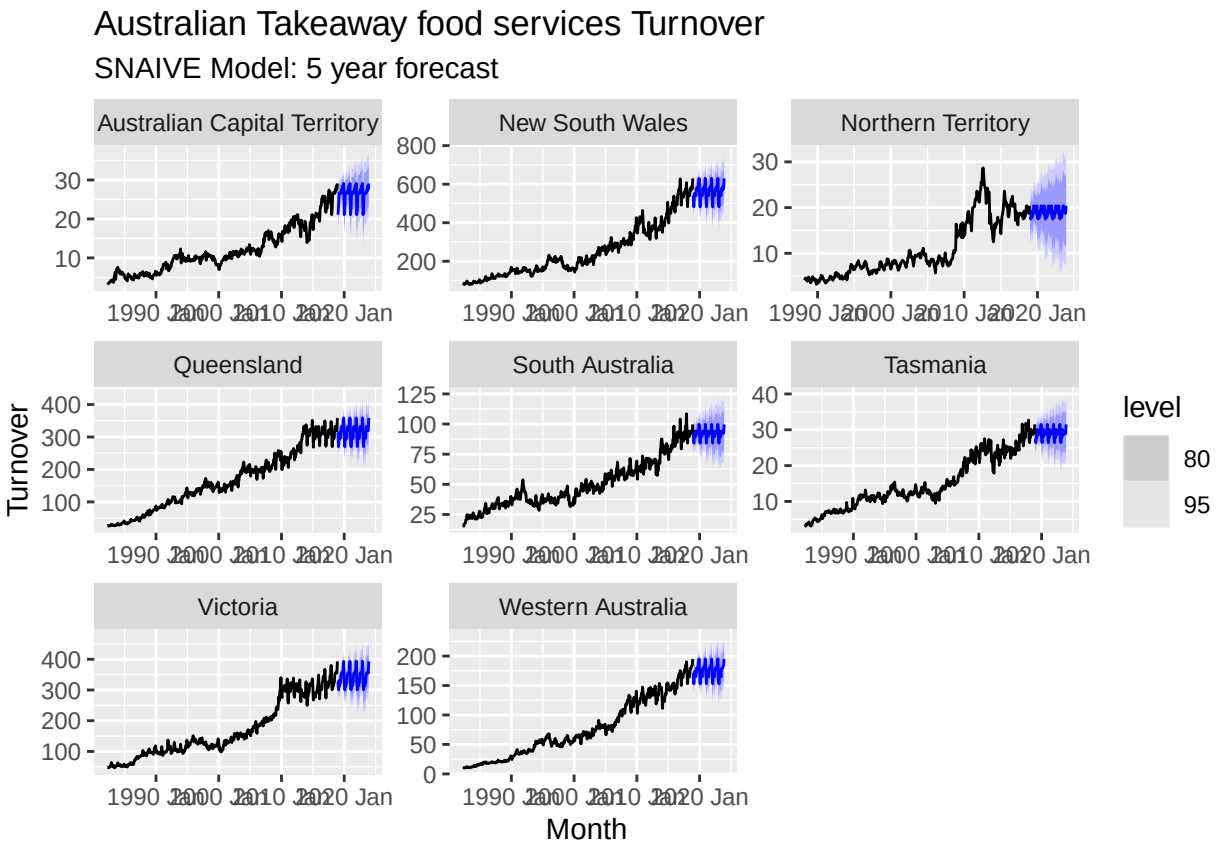
Australian takeaway food turnover (`aus_retail`)

```
aus_retail |>
  filter(Industry == 'Takeaway food services') |>
  model(stl = STL(Turnover)) |>
  components() |>
  autoplot()
```



Use of the `SNAIVE` model is most appropriate because the STL decomposition shows a small seasonal component.

```
aus_retail |>
  filter(Industry == 'Takeaway food services') |>
  model(SNAIVE(Turnover)) |>
  forecast(h = '5 years') |>
  autoplot(aus_retail) +
  labs(title = 'Australian Takeaway food services Turnover',
       subtitle = 'SNAIVE Model: 5 year forecast') +
  facet_wrap(~State, scales = 'free')
```



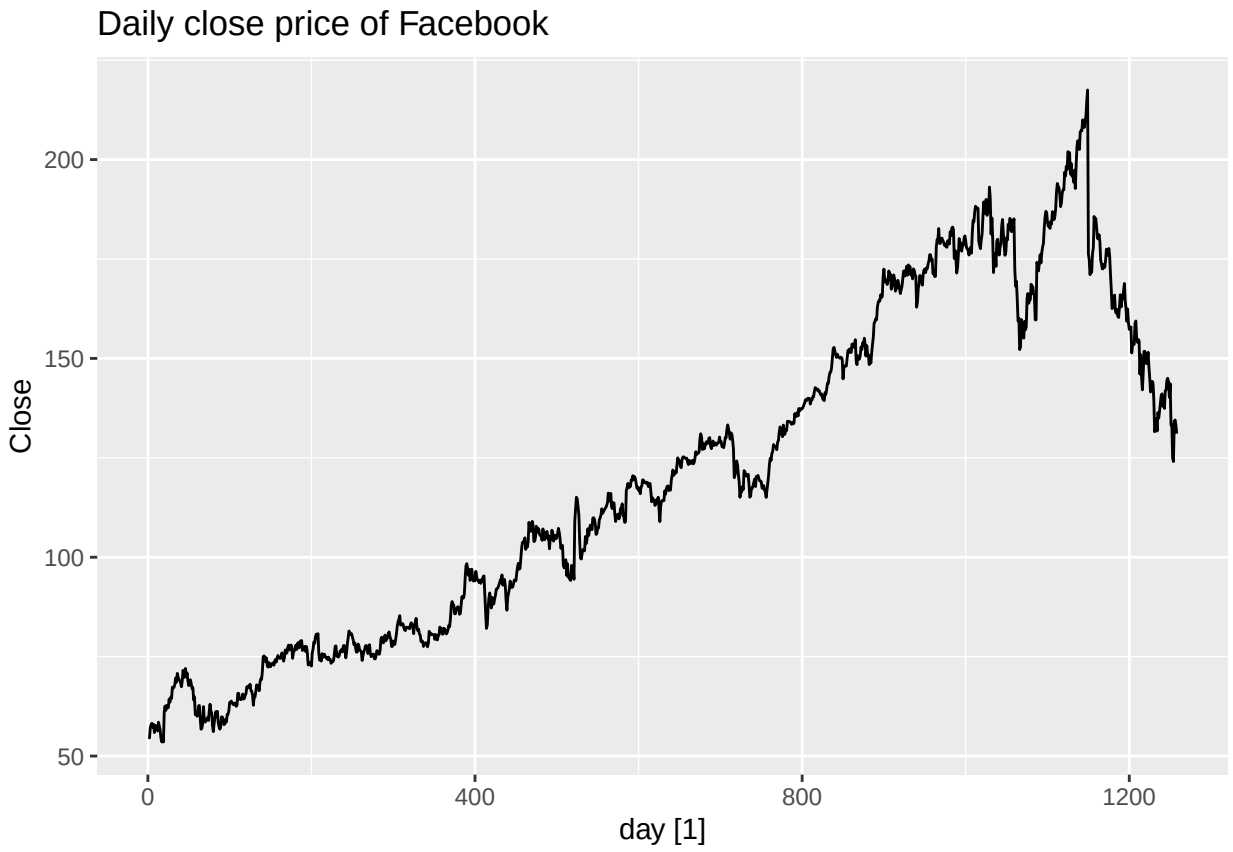
Exercise 5.2

Use the Facebook stock price (`gafa_stock`) to do the following:

a. Produce a time plot of the series

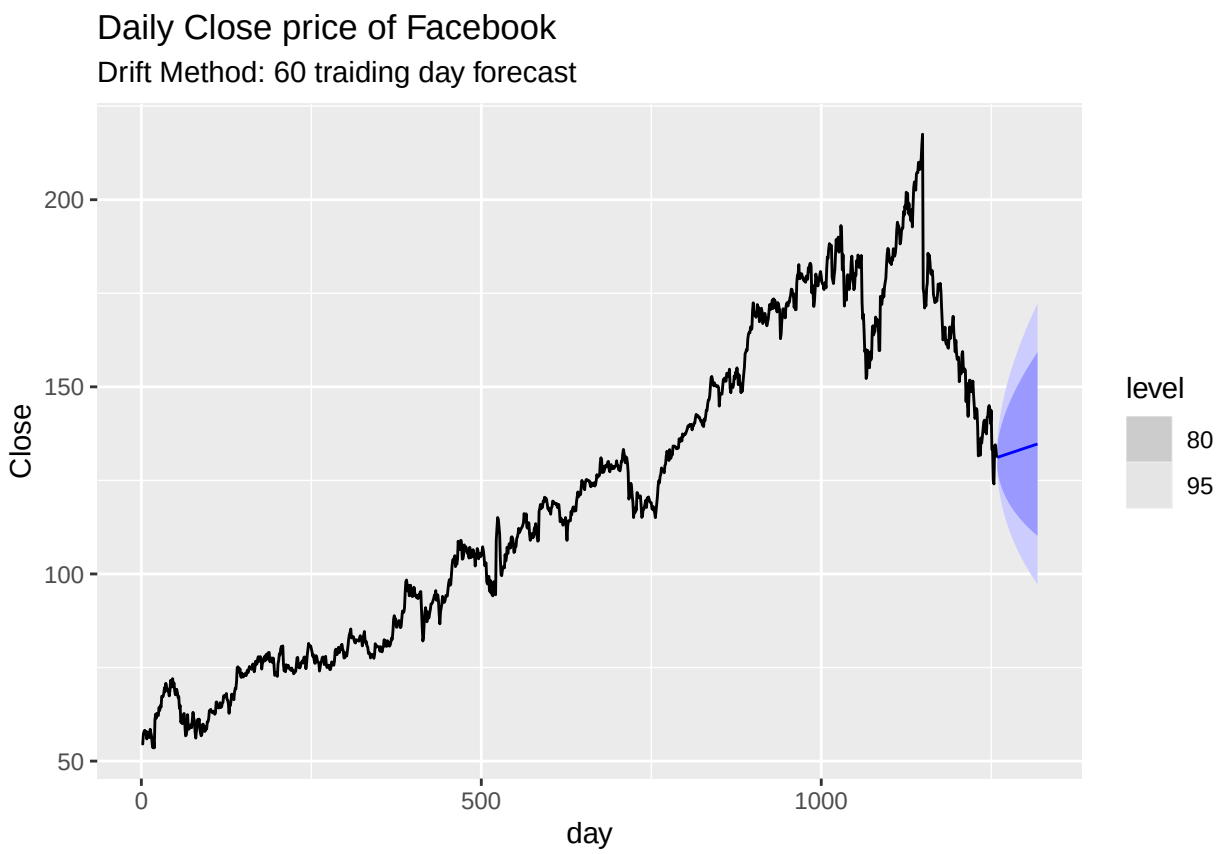
```
fb <- gafa_stock |>
  filter(Symbol == 'FB') |>
  mutate(day = row_number()) |>
  update_tsibble(index = day, regular = TRUE)

fb |>
  autoplot(Close) +
  labs(title = 'Daily close price of Facebook')
```



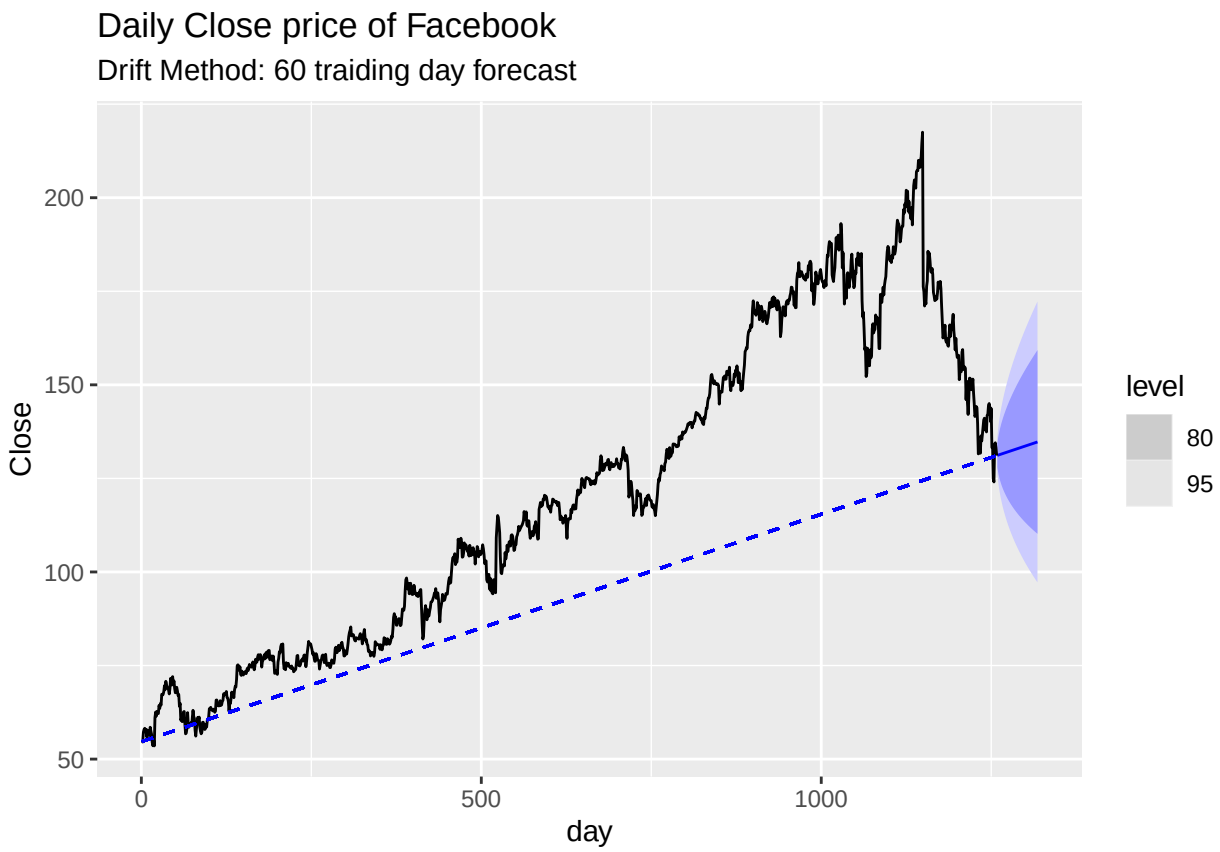
b. Produce forecasts using the drift method and plot them.

```
fb |>
  model(RW(Close ~ drift())) |>
  forecast(h = 60) |>
  autoplot(fb) +
  labs(title = 'Daily Close price of Facebook',
       subtitle = 'Drift Method: 60 traiding day forecast')
```



c. Show that the forecasts are identical to extending the line drawn between the first and last observations.

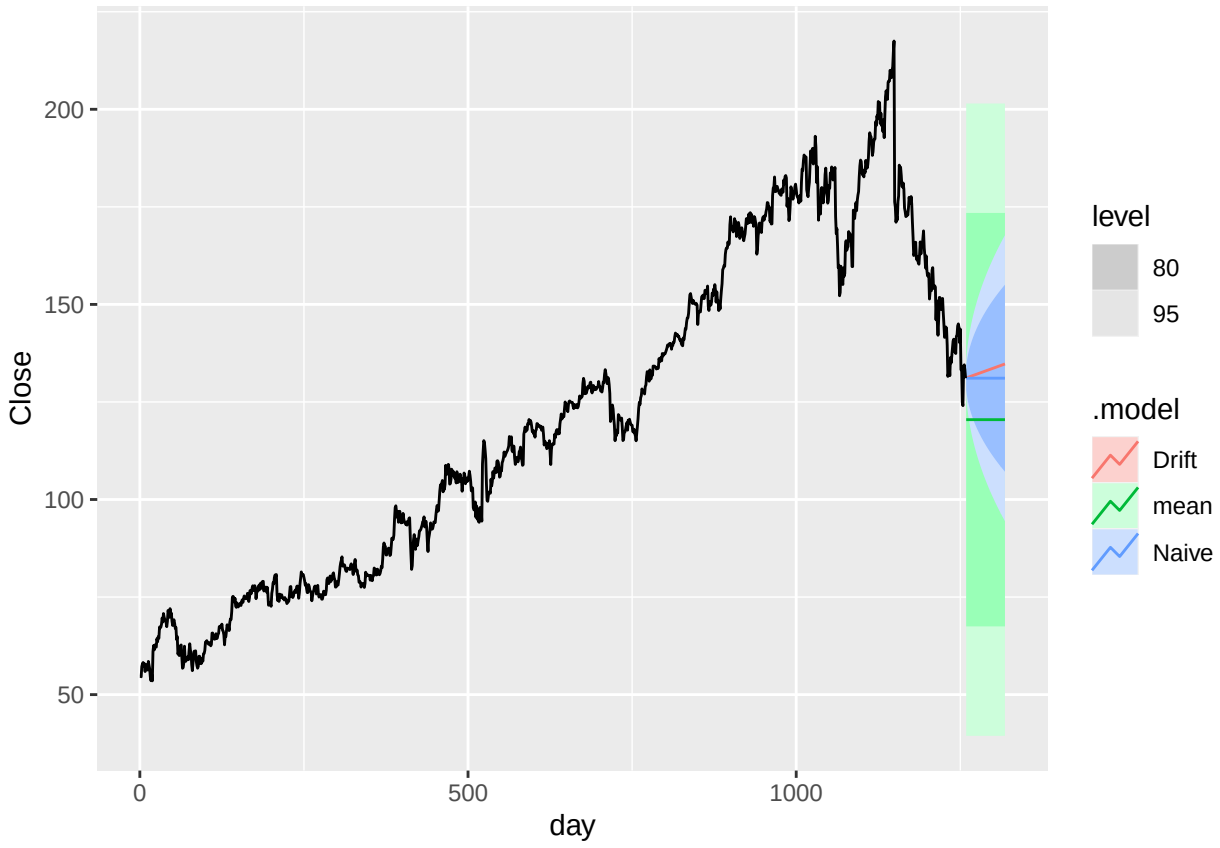
```
fb |>
  model(RW(Close ~ drift())) |>
  forecast(h = 60) |>
  autoplot(fb) +
  labs(title = 'Daily Close price of Facebook',
       subtitle = 'Drift Method: 60 traiding day forecast') +
  geom_segment(x = fb |>
                 filter(day == min(day)) |>
                 pull(day),
               y = fb |>
                 filter(day == min(day)) |>
                 pull(Close),
               xend = fb |>
                 filter(day == max(day)) |>
                 pull(day),
               yend = fb |>
                 filter(day == max(day)) |>
                 pull(Close),
               colour = 'blue',
               linetype = 'dashed')
```



d. Try using some of the other benchmark functions to forecast the same data set. Which do you think is best? Why?

It makes no sense to use the `SNAIVE` method since the data has no seasonality component. Using `MEAN` and `NAIVE` produce a constant forecast which doesn't take the overall trend into account.

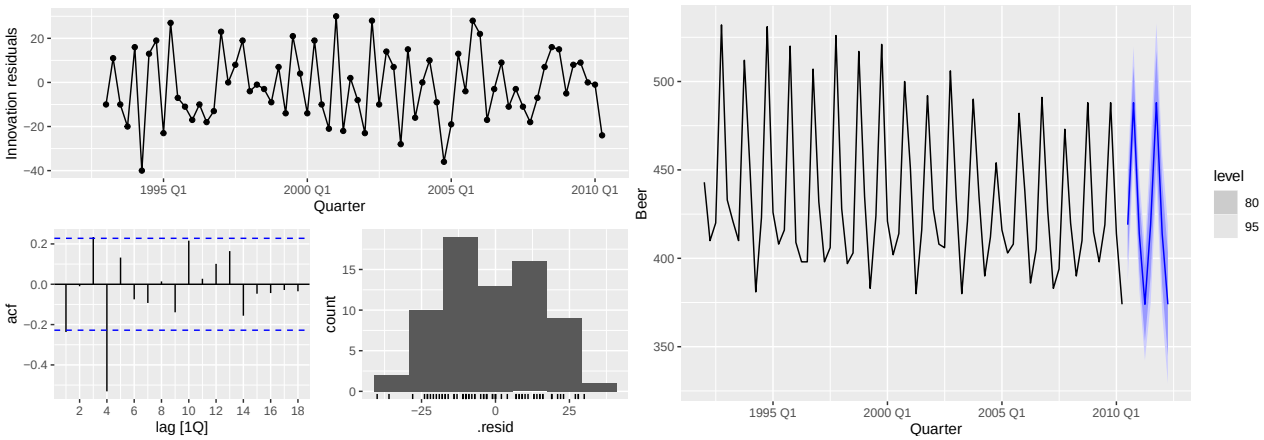
```
fb |>
  model(mean = MEAN(Close),
        Naive = NAIVE(Close),
        Drift = RW(Close ~ drift())) |>
  forecast(h = 60) |>
  autoplot(fb)
```



Exercise 5.3

Apply a seasonal naïve method to the quarterly Australian beer production data from 1992. Check if the residuals look like white noise, and plot the forecasts. The following code will help.

```
# Extract data of interest
recent_production <- aus_production |>
  filter(year(Quarter) >= 1992)
# Define and estimate a model
fit <- recent_production |> model(SNAIVE(Beer))
# Look at the residuals
fit |> gg_tsresiduals()
# Look a some forecasts
fit |> forecast() |> autoplot(recent_production)
```



Box-Pierce Test

```
fit |>
  augment() |>
  features(.innov, box_pierce, lag = 8)
```

```
## # A tibble: 1 x 3
##   .model       bp_stat bp_pvalue
##   <chr>          <dbl>     <dbl>
## 1 SNAIVE(Beer)    29.7  0.000234
```

Ljung-Box Test

```
fit |>
  augment() |>
  features(.innov, ljung_box, lag = 8)
```

```
## # A tibble: 1 x 3
##   .model       lb_stat lb_pvalue
##   <chr>          <dbl>     <dbl>
## 1 SNAIVE(Beer)    32.3 0.0000834
```

Conclusion

The p-value from both tests is less than 0.05, which indicates that the residuals are not explained by white noise.

Exercise 5.4

Repeat the previous exercise using the Australian Exports series from `global_economy` and the Bricks series from `aus_production`. Use whichever of `NAIVE()` or `SNAIVE()` is more appropriate in each case.

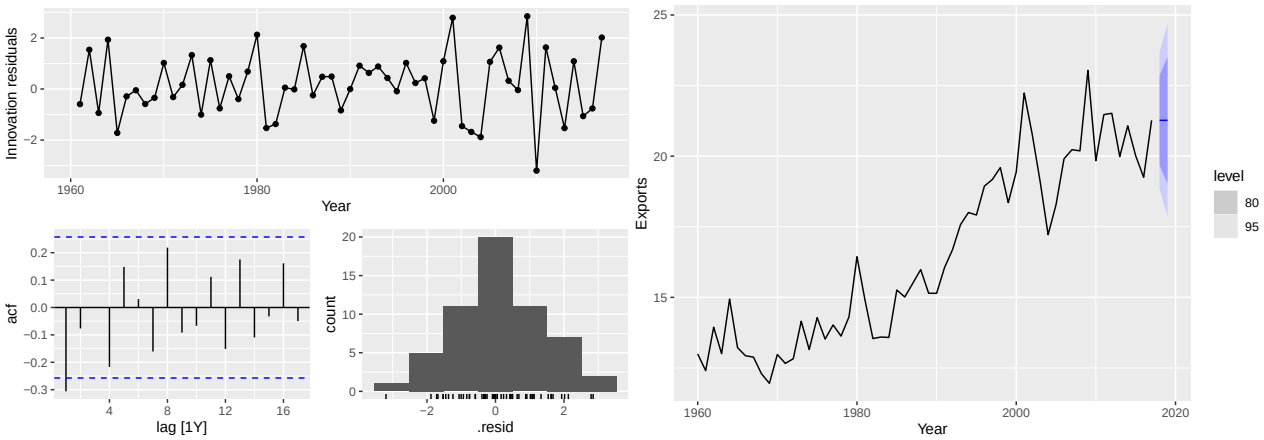
Australian Exports from `global_economy`

```
global_economy |>
  filter(Country == 'Australia') |>
  autoplot(Exports) +
  labs(title = 'Australian Exports')
```

Australian Exports does not appear to have a seasonal component, so we will construct a model using `NAIVE`.

```
model_ae <- global_economy |>
  filter(Country == 'Australia') |>
  model(NAIVE(Exports))

model_ae |> gg_tsresiduals()
model_ae |> forecast() |> autoplot(global_economy)
```



The residuals are centered around zero and appear to be normally distributed.

```
model_ae |>
  augment() |>
  features(.innov, ljung_box, lag = 10)
```

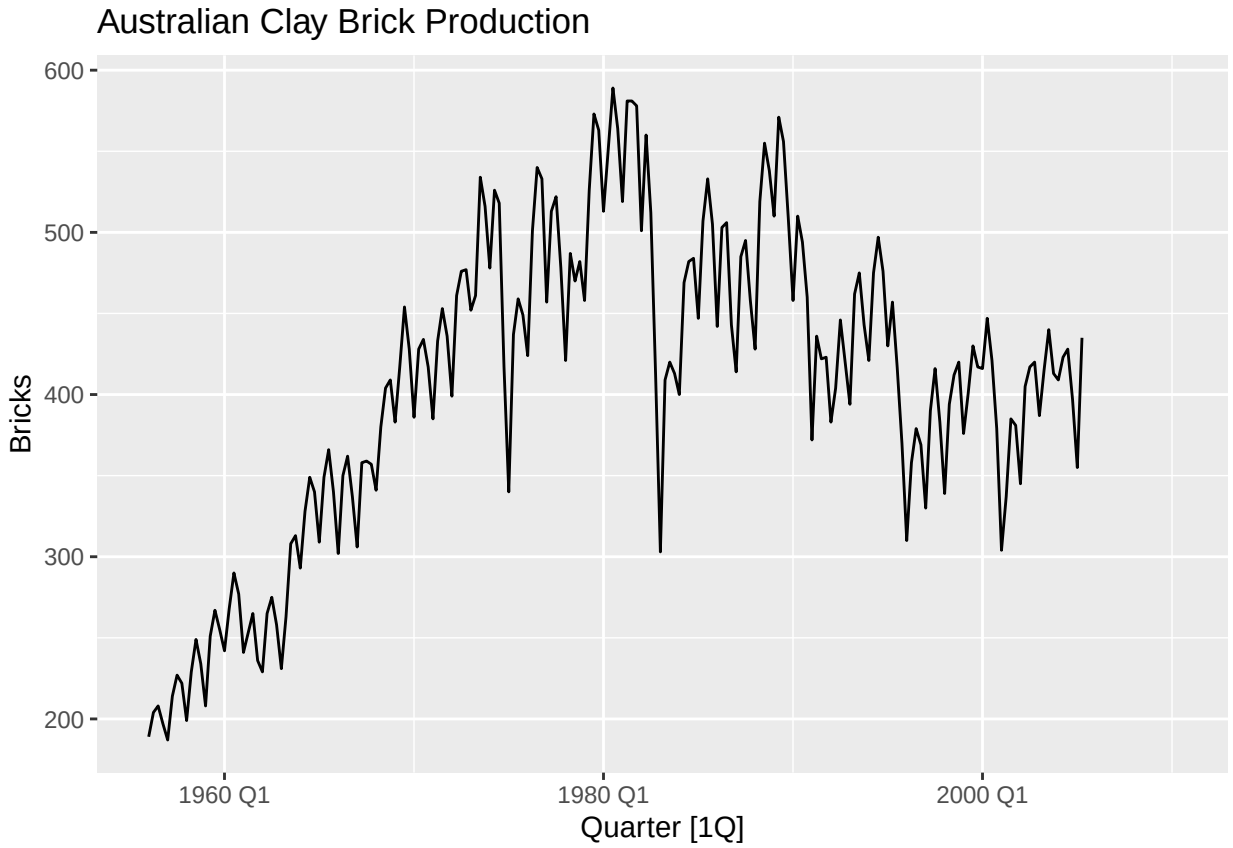
```
## # A tibble: 1 x 4
##   Country   .model          lb_stat lb_pvalue
##   <fct>     <chr>             <dbl>     <dbl>
## 1 Australia NAIVE(Exports)    16.4    0.0896
```

The Ljung-Box test produces a p-value greater than 0.05 which allows us to conclude that the residuals resemble white noise.

Bricks from `aus_production`

```
aus_production |>
  autoplot(Bricks) +
  labs(title = 'Australian Clay Brick Production')
```

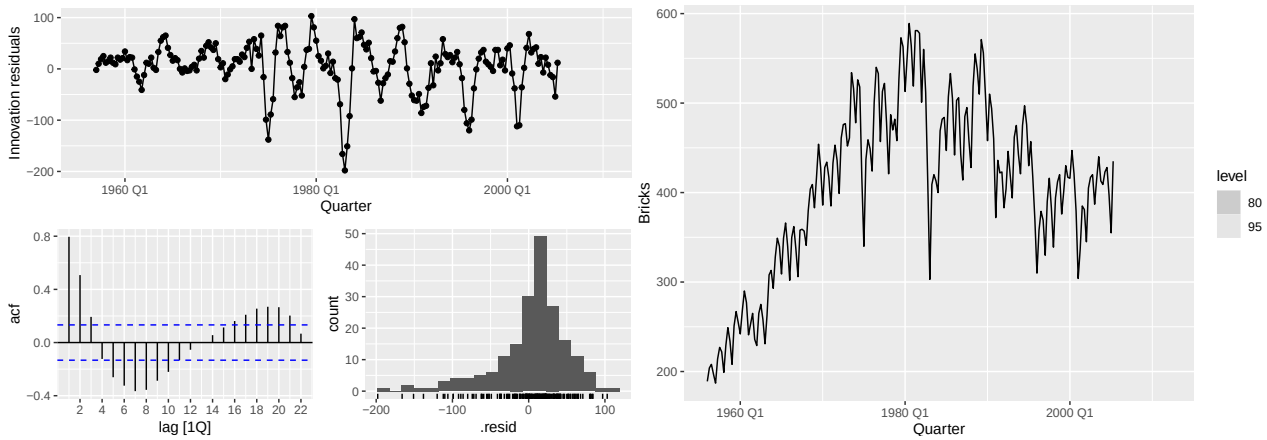
```
## Warning: Removed 20 rows containing missing values (‘geom_line()‘).
```



Bricks production has a strong seasonal component, so we will construct a model using `SNAIVE`.

```
model_b <- aus_production |>
  model(SNAIVE(Bricks))

model_b |> gg_tsresiduals()
model_b |> forecast() |> autoplot(aus_production)
```



The residuals do not seem to be centered at zero. The ACF shows strong seasonality that is not accounted for in the model.

```
model_b |>
  augment() |>
  features(.innov, ljung_box, lag = 8)
```

```
## # A tibble: 1 x 3
##   .model          lb_stat lb_pvalue
##   <chr>             <dbl>     <dbl>
## 1 SNAIVE(Bricks)    274.          0
```

The Ljung-Box test produces a p-value less than 0.05 which allows us to conclude that the residuals do not resemble white noise.

Exercise 5.7

From your retail time series (from Exercise 7 in Section 2.10):

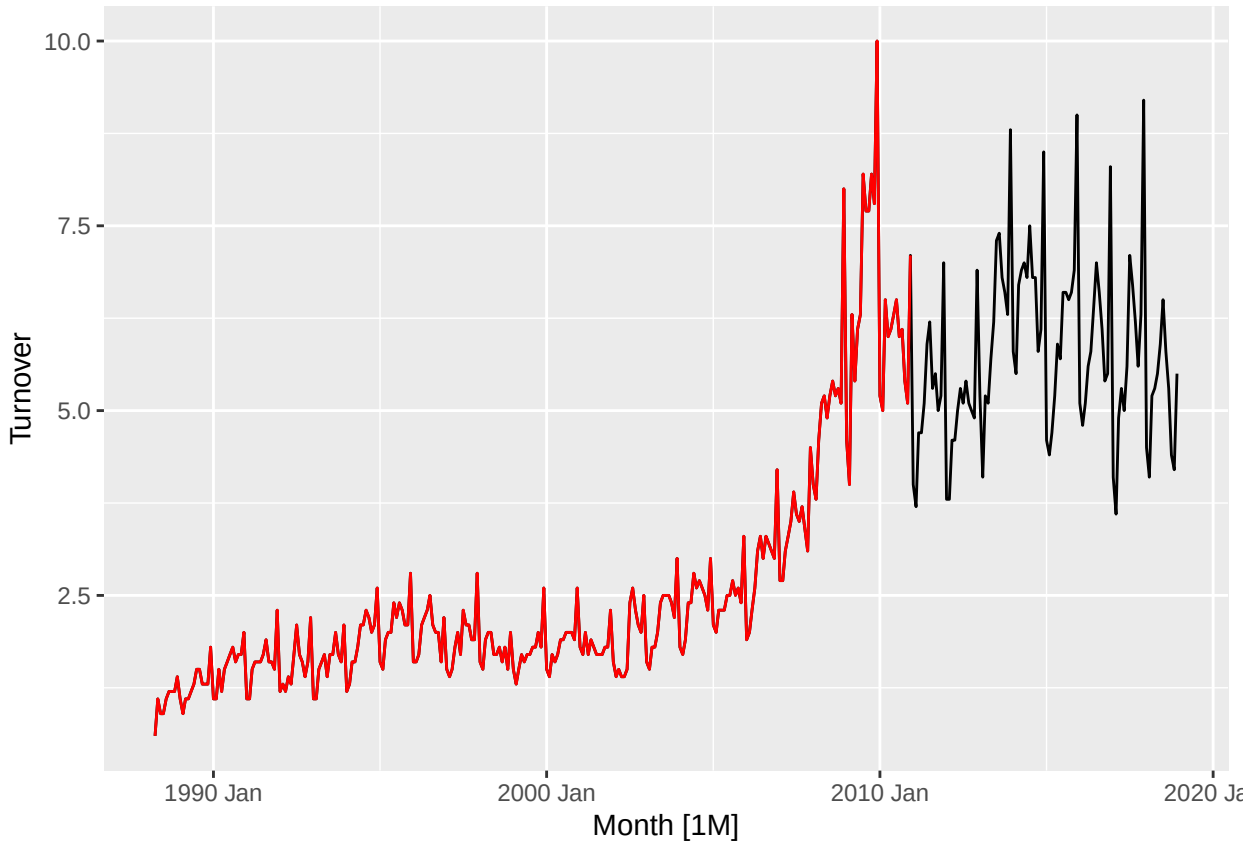
```
set.seed(31415)
myseries <- aus_retail |>
  filter(`Series ID` == sample(aus_retail$`Series ID`, 1))
```

a. Create a training dataset consisting of observations before 2011 using

```
myseries_train <- myseries |> filter(year(Month) < 2011)
```

b. Check that your data have been split appropriately by producing the following plot.

```
autoplot(myseries, Turnover) +
  autolayer(myseries_train, Turnover, colour = "red")
```

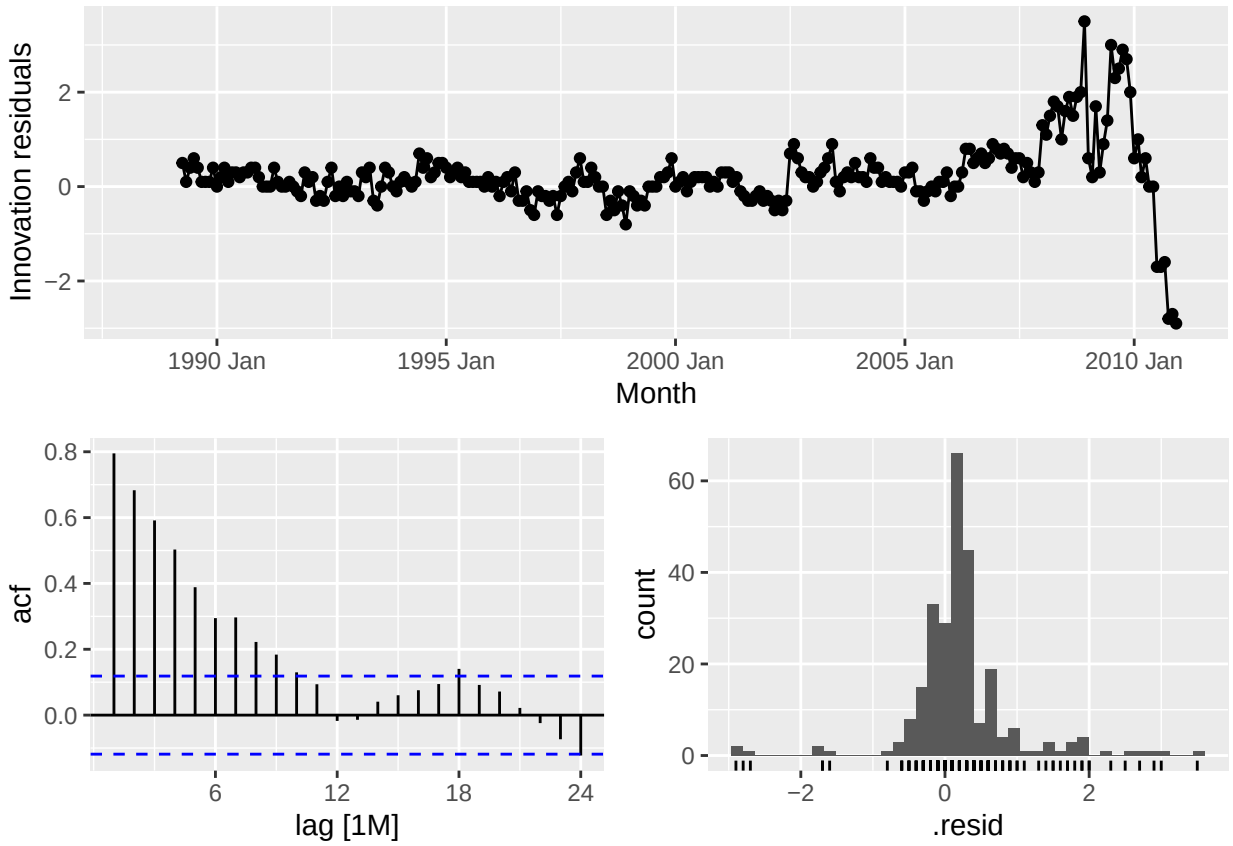


c. Fit a seasonal naïve model using `SNAIVE()` applied to your training data (`myseries_train`).

```
myseries_fit <- myseries_train |>
  model(SNAIVE(Turnover))
```

d. Check the residuals

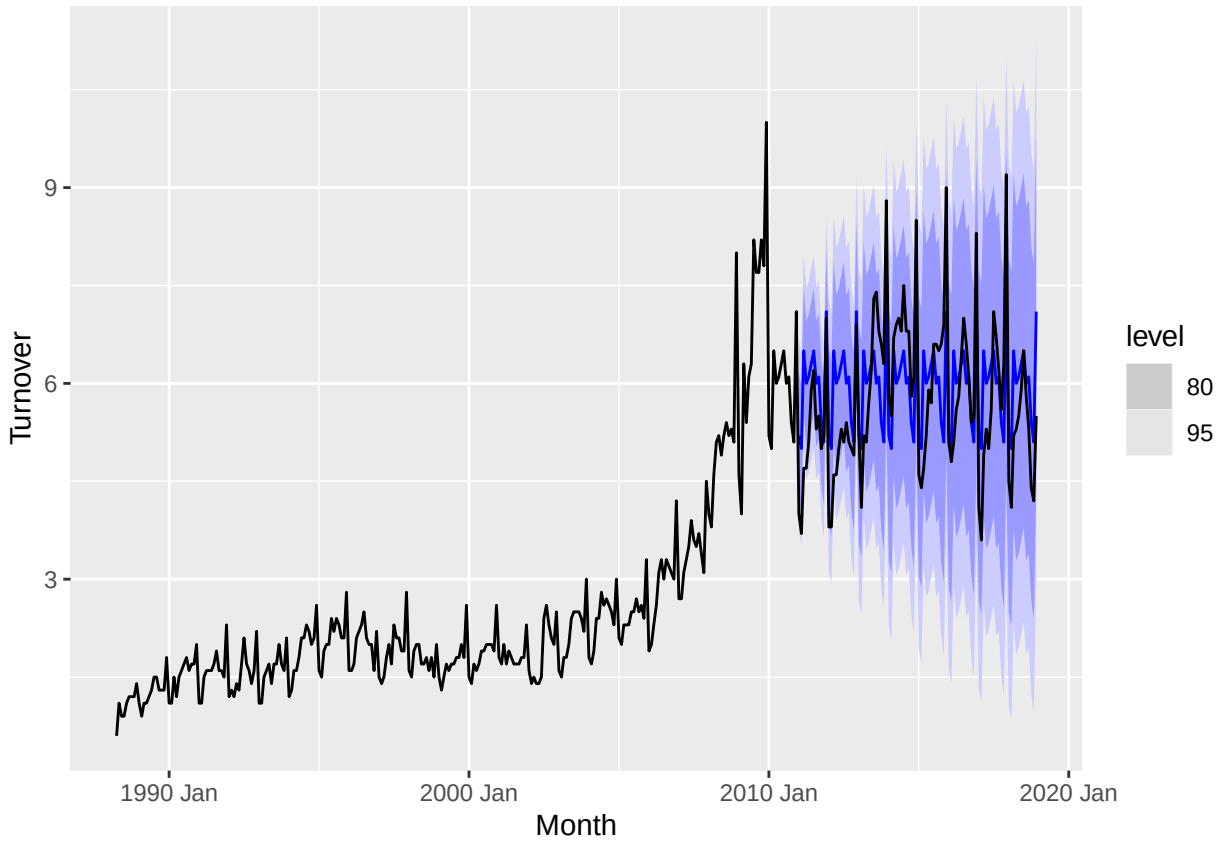
```
myseries_fit |> gg_tsresiduals()
```



The residuals do not appear to be uncorrelated nor normally distributed.

e. Produce forecasts for the test data

```
fc <- myseries_fit |>
  forecast(new_data = anti_join(myseries, myseries_train,
                                by = join_by(State, Industry, `Series ID`, Month, Turnover)))
fc |> autoplot(myseries)
```



f. Compare the accuracy of your forecasts against the actual values.

```
myseries_fit |>
  accuracy() |>
  select(.model, .type, RMSE, MAE, MAPE, MASE, RMSSE)
```

```
## # A tibble: 1 x 7
##   .model           .type     RMSE   MAE  MAPE  MASE RMSSE
##   <chr>            <chr>    <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 SNAIVE(Turnover) Training 0.748 0.446  14.9     1  1.00
```

```
fc |>
  accuracy(myseries) |>
  select(.model, .type, RMSE, MAE, MAPE, MASE, RMSSE)
```

```
## # A tibble: 1 x 7
##   .model           .type  RMSE   MAE  MAPE  MASE RMSSE
##   <chr>            <chr> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 SNAIVE(Turnover) Test  0.953 0.801  14.7  1.80  1.27
```

g. How sensitive are the accuracy measures to the amount of training data used?

Accuracy measures are very sensitive to the amount of training data used. The more training data, the better the accuracy. With time series it is also important to use training data as close to the time periods being forecast.