Homework Three for Fall 2024 DATA 624 at CUNY School of Professional Studies

Kevin Kirby

2024-09-16

Overview

The below are answers to exercises 5.1, 5.2, 5.3, 5.4 and 5.11 from section 3.7 of the [Forecasting: Principles and Practice (3rd ed)] (https://otexts.com/fpp3/graphics-exercises.html). This is homework one of the DATA 624 class "Predictive Analytics." Unless otherwise noted, all datasets used below are from the fpp3 package that's owned and maintained by the book's authors.

First, I'll load the required libraries:

```
library(fpp3)
```

```
## Registered S3 method overwritten by 'tsibble':
##   method               from
##   as_tibble.grouped_df dplyr
```

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

```
## v tibble      3.2.1     v tsibble     1.1.5
## v dplyr       1.1.4     v tsibbledata 0.4.1
## v tidyr       1.3.1     v feasts      0.3.2
## v lubridate   1.9.3     v fable       0.3.4
## v ggplot2     3.5.1     v fabletools  0.4.2
```

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

```
library(fable)
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(dplyr)
library(ggplot2)
library(feasts)
```

5.1: Produce forecasts with simple methods

Thhe first exercise asks me to produce forecasts using the most appropriat of the following methods:

- Naive
- Seaonal Naive (SNaive)
- Random Walk (RW) with Dift

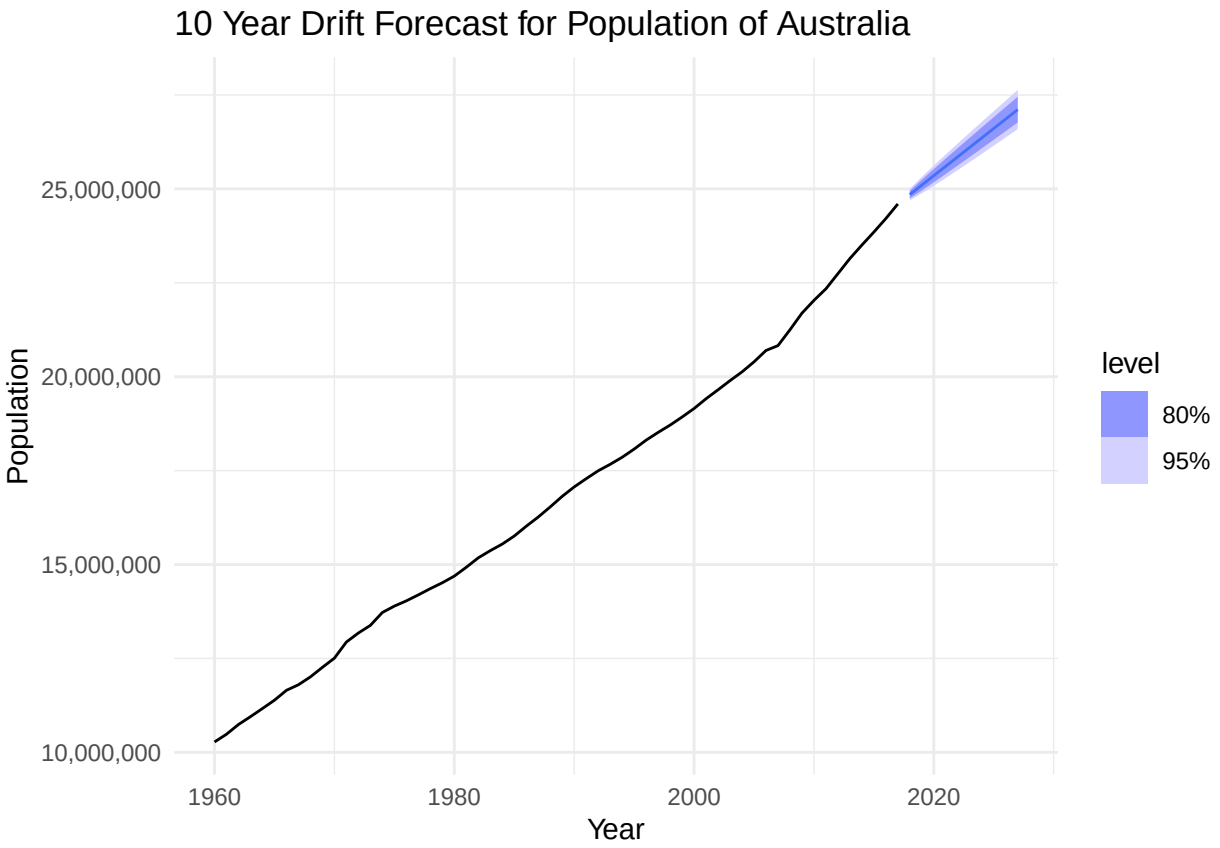
For the following data sets:

- Australian Population from global_economy
- Bricks from aus_production
- NSW Lambs aus_livestock
- Household wealth hh_budget
- Australian takeaway food turnover aus_retail

Forecast for Australian Population (global_economy) - RW with drift

The RW is a technique where the changes between consecutive forecasted values are random. When you combine it with the drift technique, which is where the amount of increase or decrease in the forecast equals the average change seen in historical data, you get predictions that are randomly in the direction of where it could reasonably go. Australian population has been on a steady and consistent growth pattern for the last 60 years so a random forecast in the proper direction based on historical data in a reasonable approach.

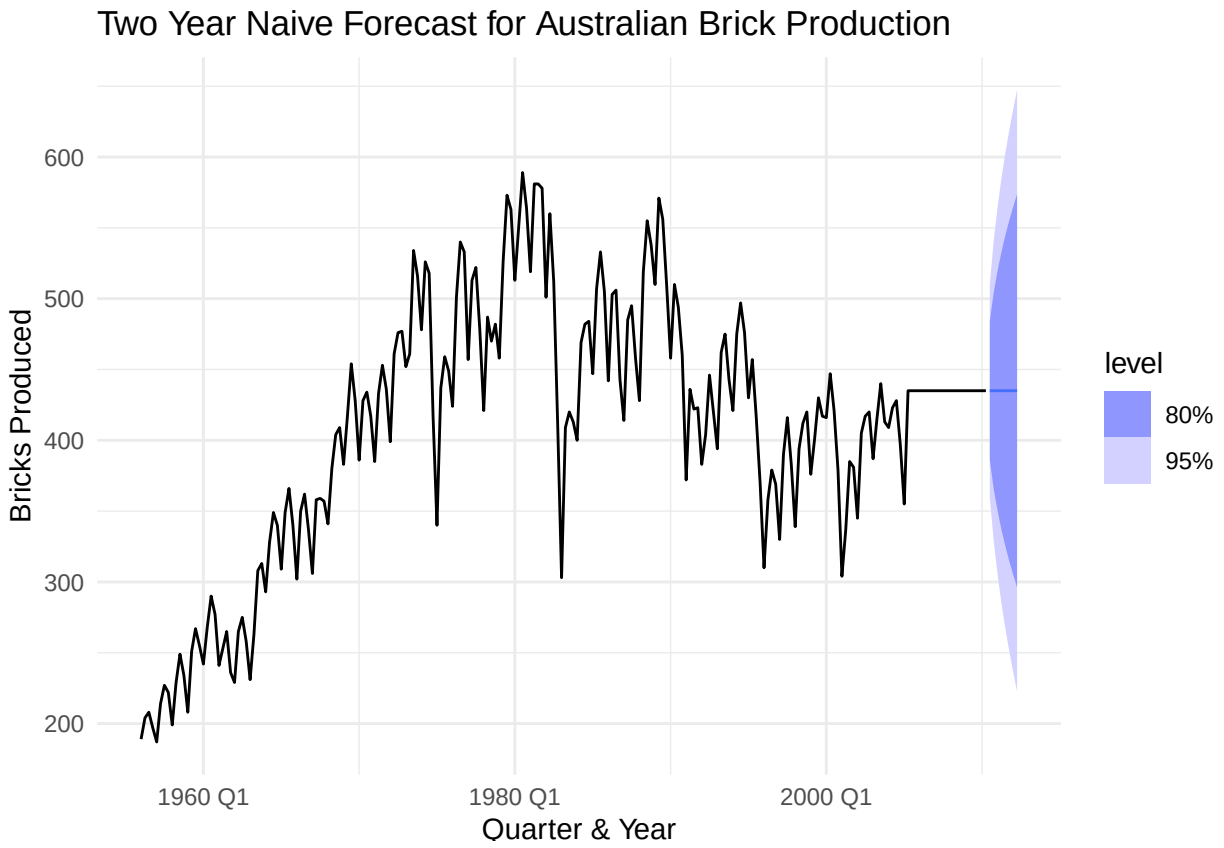
```
aussie_drift <- global_economy %>%
  filter(Country == "Australia") %>%
  model(RW(Population ~ drift())) %>%
  forecast(h = "10 years")

aussie_drift %>%
  autoplot(global_economy) +
    labs(title = "10 Year Drift Forecast for Population of Australia",
       x = "Year",
       y = "Population") +
    scale_y_continuous(labels = scales::comma_format()) +  # Changes to regular number format with comm
    theme_minimal()
```



Forecast for Bricks (aus_production) - Naive method

Naive is where all forecasted values are equal to the last observed value. While brick production has been all over the place between 1970 and 2010, it's started and ended at the same place . Additionally, the data has calmed down in the last few quarters available, making naive a decent choice.

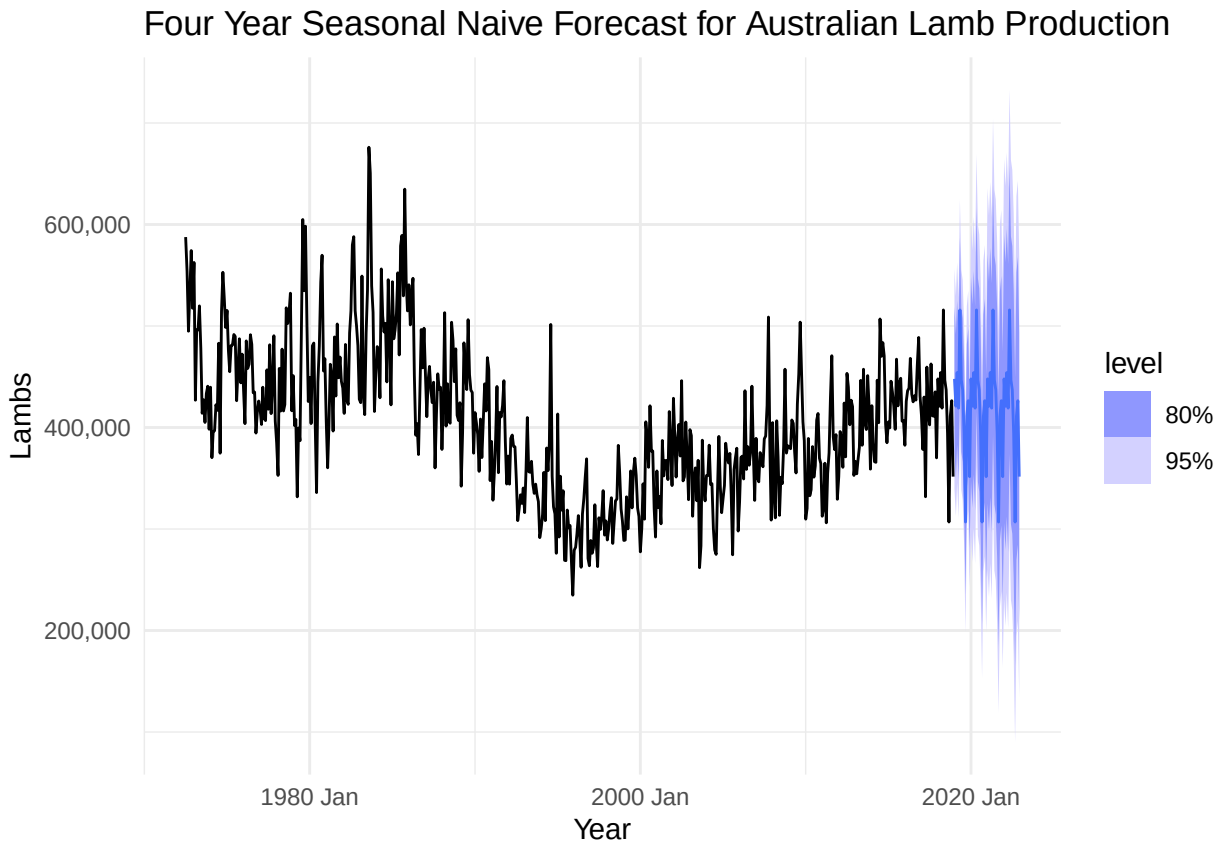
```
aus_bricks <- aus_production %>%
  mutate(Bricks = na.interp(Bricks))

bricks_naive <- aus_bricks %>%
  model(NAIVE(Bricks)) %>%
  forecast(h = "2 years")

bricks_naive %>%
  autoplot(aus_bricks) +
    labs(title = "Two Year Naive Forecast for Australian Brick Production",
         x = "Quarter & Year",
         y = "Bricks Produced") +
    scale_y_continuous(labels = scales::comma_format()) +
    theme_minimal()
```



Forecast for NSW lambs (aus_livestock) - SNaive

SNaive is similar to the naive method, with the difference being that the forecasted value equals the last value from the same season. A first quarter value would equal the last first quarter value. Lamb production has had the same pattern in a narrow seaonal band for the last fifteen year and it's reasonable to assume it will continue the same pattern into the future.

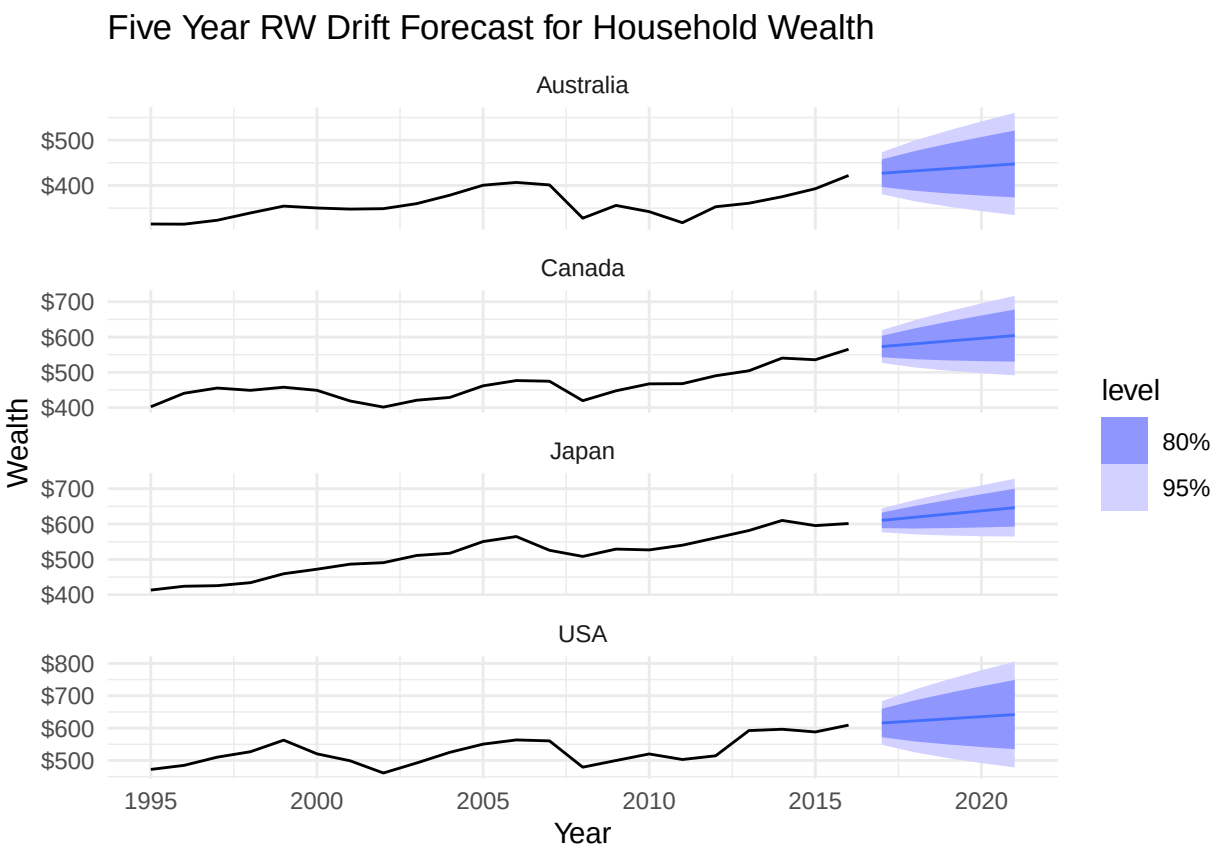
```
lambs_snaive <- aus_livestock %>%
  filter(Animal == "Lambs", State == "New South Wales") %>%
  model(SNAIVE(Count)) %>%
  forecast(h = "4 years")

lambs_snaive %>%
  autoplot(aus_livestock) +
    labs(title = "Four Year Seasonal Naive Forecast for Australian Lamb Production",
         x = "Year",
         y = "Lambs") +
    scale_y_continuous(labels = scales::comma_format()) +
    theme_minimal()
```



Forecast for household wealth (hh_budget) - RW with drift

For the same reasons as the bricks forecast earlier, I chose RW with drift for household wealth. The drift component is key as it sends in the same direction as where it came from. Japan's forecast is a bit aggressive, though. I would wager on their household wealth starting to trend down due to aging population.

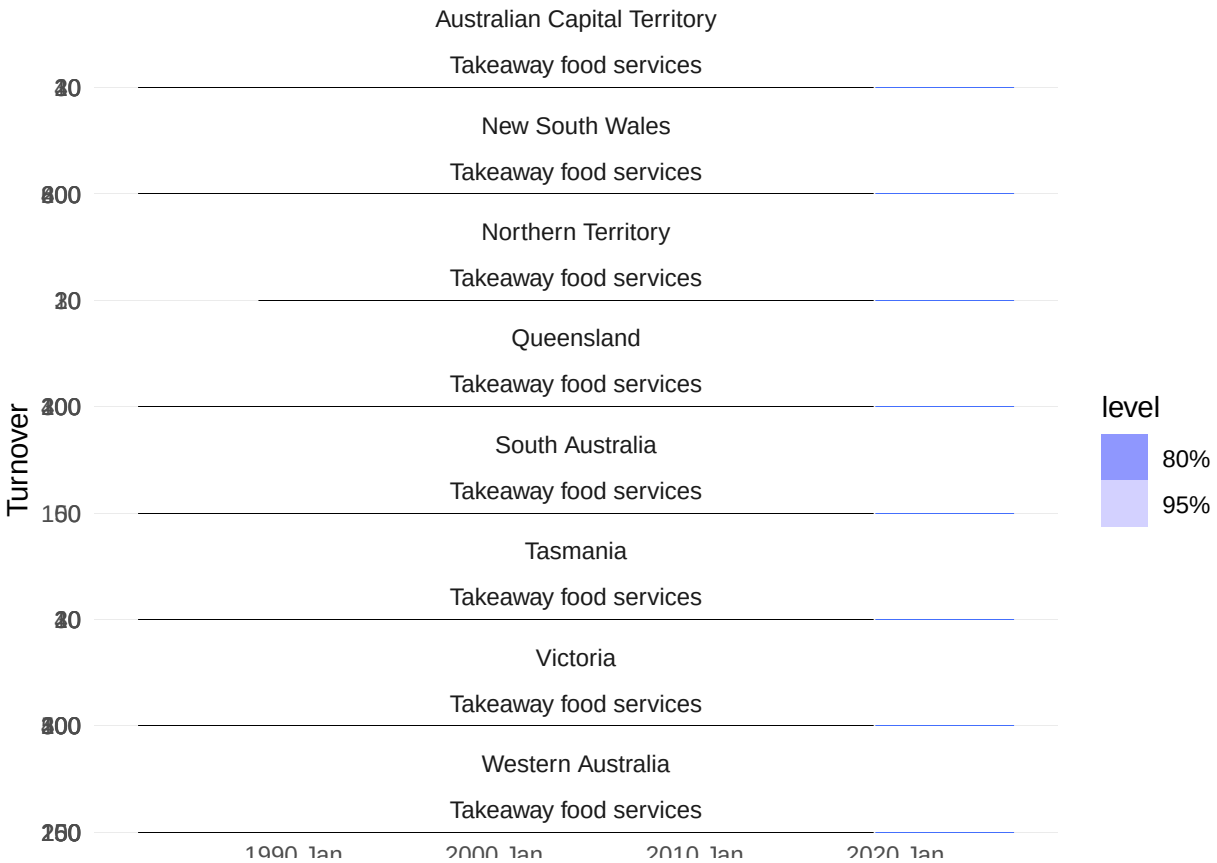
```
wealth_rw <- hh_budget %>%
  model(RW(Wealth ~ drift())) %>%
  forecast(h = "5 years")

wealth_rw %>%
  autoplot(hh_budget) +
    labs(title = "Five Year RW Drift Forecast for Household Wealth",
         x = "Year",
         y = "Wealth") +
    scale_y_continuous(labels = scales::dollar_format(scale = 1, prefix = "$", big.mark = ",")) +
    theme_minimal()
```



Forecast for Australian takeaway food turnover (aus_retail) - SNAIVE method

Turnover is a term for how much money is spent in that sector in a given month, with the data provided by Australian state and sector. I chose Snaive and kept it at the state level because, while takeaway food sales are seaonanal in nature, it's also dependent on the region.

```
takeaway_snaive <- aus_retail %>%
  filter(Industry == "Takeaway food services") %>%
  model(SNAIVE(Turnover)) %>%
  forecast(h = "7 years")

takeaway_snaive %>%
  autoplot(aus_retail) +
    labs(title = "Seven Year Seasonal Naive Forecast for Takeaway Food Turnover",
         x = "Year",
         y = "Turnover") +
    scale_y_continuous(labels = scales::comma_format()) +
    theme_minimal()
```



5.2 Forecast and review of Facebook stock price from gafa_stock

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

The next exercise is to answer the following questions about Faceboock's stock price:

- Produce a time plot of the series
- Produce forecasts using the drift method and plot them
- Show that the forecasts are identical to extending the line drawn between the first and last observations
- Try using some of the other benchmark functions to forecast the same data set. Which do you think is best? Why?

First, I'll setup the data by filtering for the stock and dropping NA values for the closing price:

```
fb_stock <- gafa_stock %>%
  filter(Symbol == "FB") %>%
  update_tsibble(index = Date, regular = TRUE) %>%
  fill_gaps()
```

A. time plot

```
fb_stock %>%
  autoplot(Close) +
  labs(title = "Facebook Stock Price - 2014-2018",
       x = "Year",
       y = "Closing Price") +
  scale_y_continuous(labels = scales::comma_format()) +
  theme_minimal()
```

B and C. Forecast and plot using RW Drift method Show that the forecasts are identical to extending the line drawn between the first and last observations

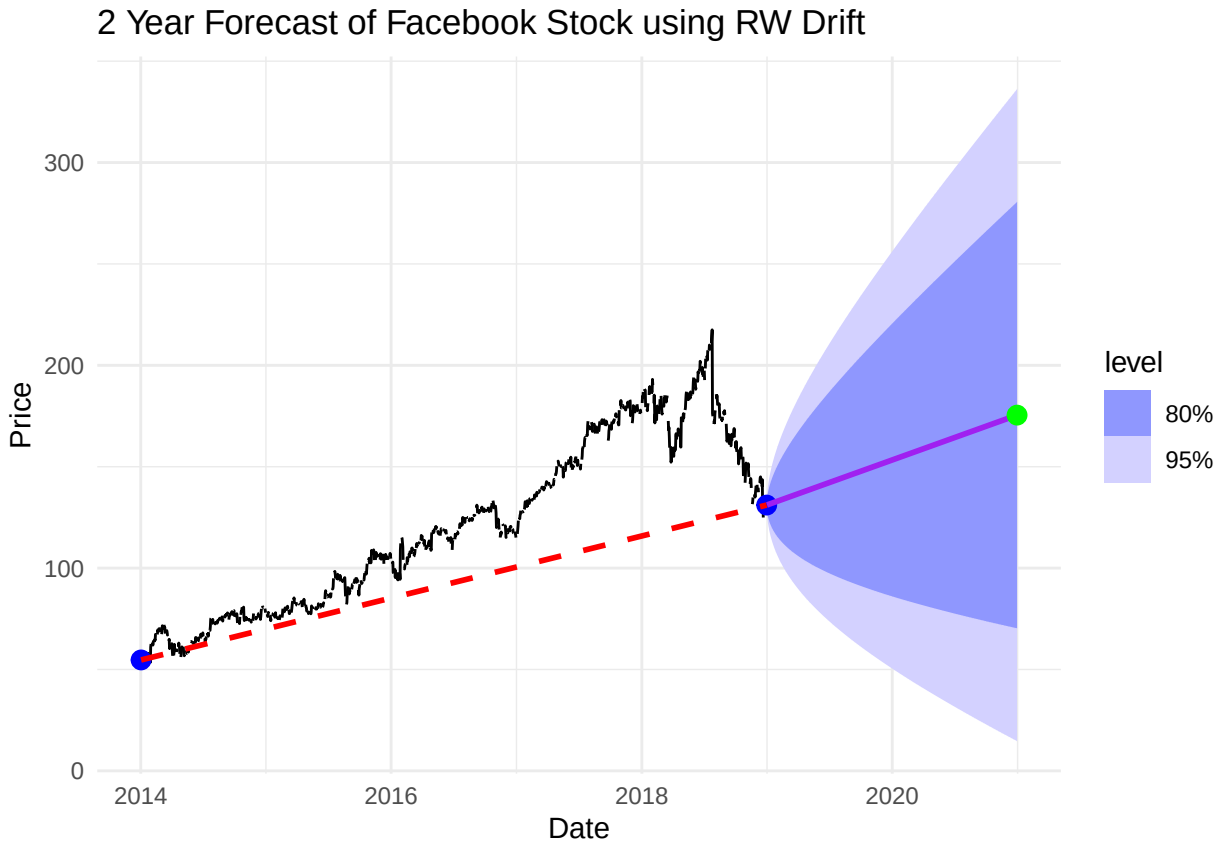
This chunk:

- Creates and draws the forecast using RW drift
- Plots as blue dots the first and last points of the historical data and draws a red dashed line between them
- Plots as a green dot the last point in the forecast, using the mean as the plotted point
- Plots a solid purple line from the last historical point to the edge of the graph at the same slope as the historical line
  - This shows that extending the line is the same as the forecast since it falls within the predicted range

```
fb_first_last <- fb_stock %>%
  slice(c(1, n()))

fb_drift <- fb_stock %>%
  model(Drift = RW(Close ~ drift())) %>%
  forecast(h = "2 years")

fb_last_f <- fb_drift %>%
  slice(n())

fb_drift %>%
  autoplot(fb_stock) +
  geom_point(data = fb_first_last, aes(x = Date, y = Close), color = "blue", size = 3) +
  annotate("segment",
```

```
          x = first(fb_first_last$Date), y = first(fb_first_last$Close),
          xend = last(fb_first_last$Date), yend = last(fb_first_last$Close),
          color = "red", linetype = "dashed", size = 1) +
  geom_line(data = fb_drift, aes(x = Date, y = .mean), color = "purple", size = 1) +
  geom_point(data = fb_last_f, aes(x = Date, y = .mean), color = "green", size = 3) +
  labs(title = "2 Year Forecast of Facebook Stock using RW Drift",
       x = "Date",
       y = "Price") +
  theme_minimal()
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



D Trying out other benchmarks + analysis

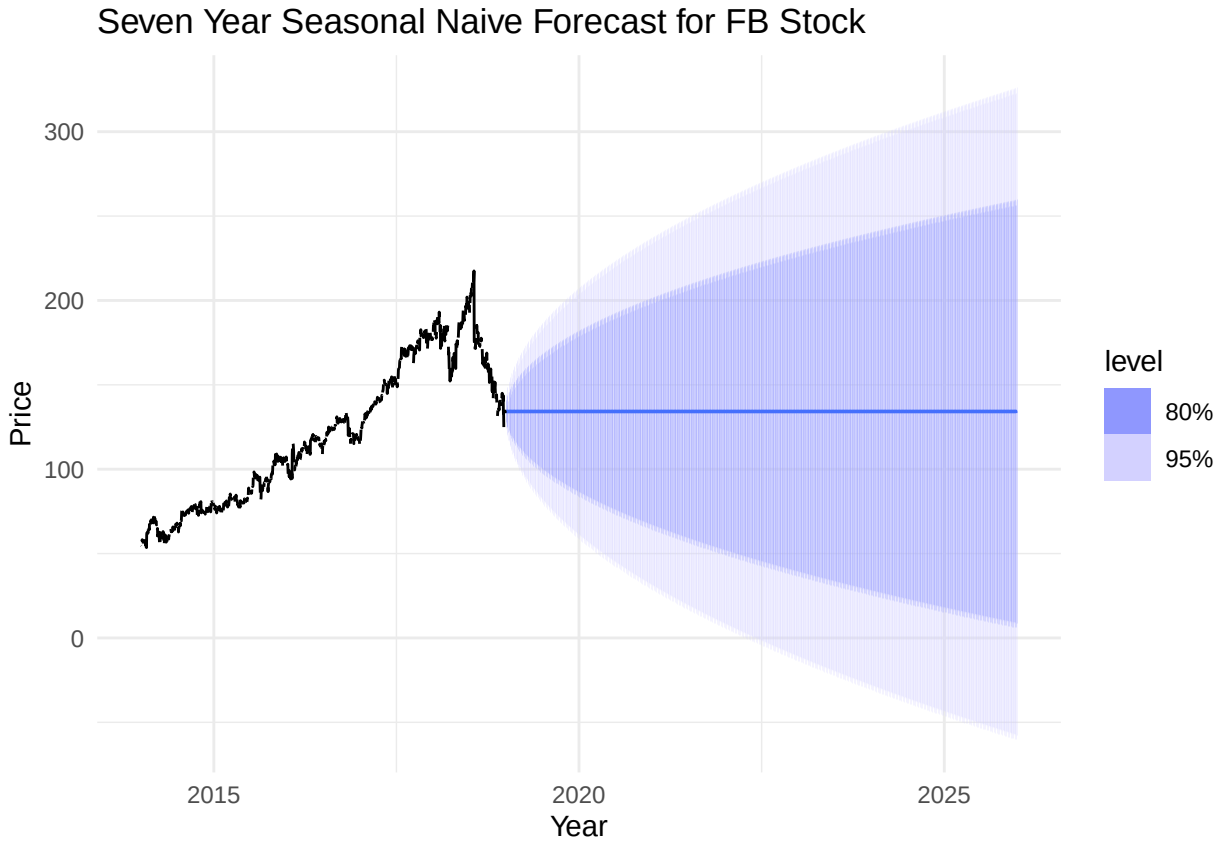
Below are SNaive and Naive forecasts for the same dataset as the chart above. The SNaive is the most accurate of the two because it includes more room for upward growth

Here's a seasonal naive forecast:

```
fb_snaive <- fb_stock %>%
  model(SNAIVE(Close)) %>%
  forecast(h = "7 years")

fb_snaive %>%
```

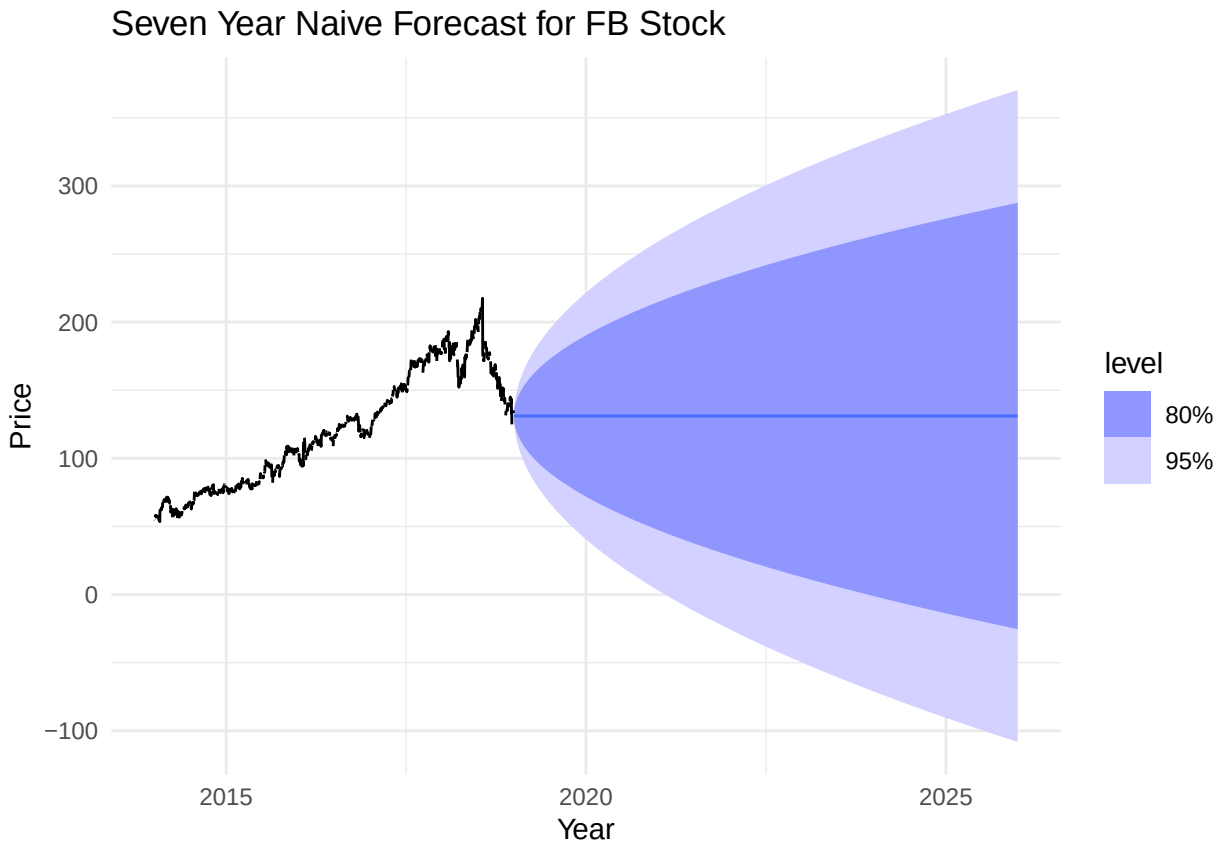
```
  autoplot(fb_stock) +
    labs(title = "Seven Year Seasonal Naive Forecast for FB Stock",
         x = "Year",
         y = "Price") +
    scale_y_continuous(labels = scales::comma_format()) +
    theme_minimal()
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).
```



Here's a regular naive forecast:

```
fb_naive <- fb_stock %>%
  model(NAIVE(Close)) %>%
  forecast(h = "7 years")

fb_naive %>%
  autoplot(fb_stock) +
    labs(title = "Seven Year Naive Forecast for FB Stock",
         x = "Year",
         y = "Price") +
    scale_y_continuous(labels = scales::comma_format()) +
    theme_minimal()
```



5.3: SNaive on Australian beer production

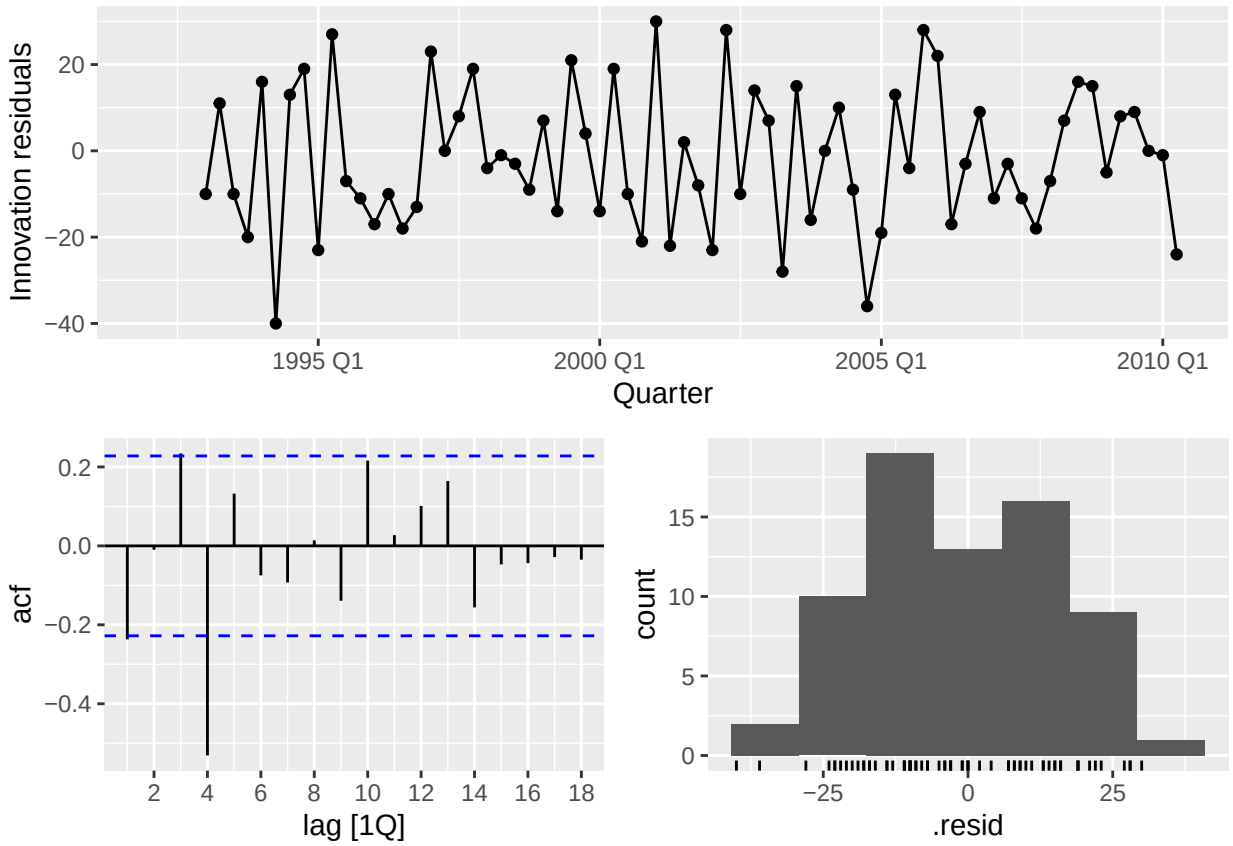
Based on the forecast graphed below, I conclude that this is a very predictable dataset, as it's been doing the same thing within almost the same seasonal bands since 1992. The only flaw in the forecast is that it assumes wide bands relative to what recent trends have been.

```
beer_cut <- aus_production %>%
  filter(year(Quarter) >= 1992)

fit <- beer_cut %>%
  model(SNAIVE(Beer))

fit %>%
  gg_tsresiduals()
```

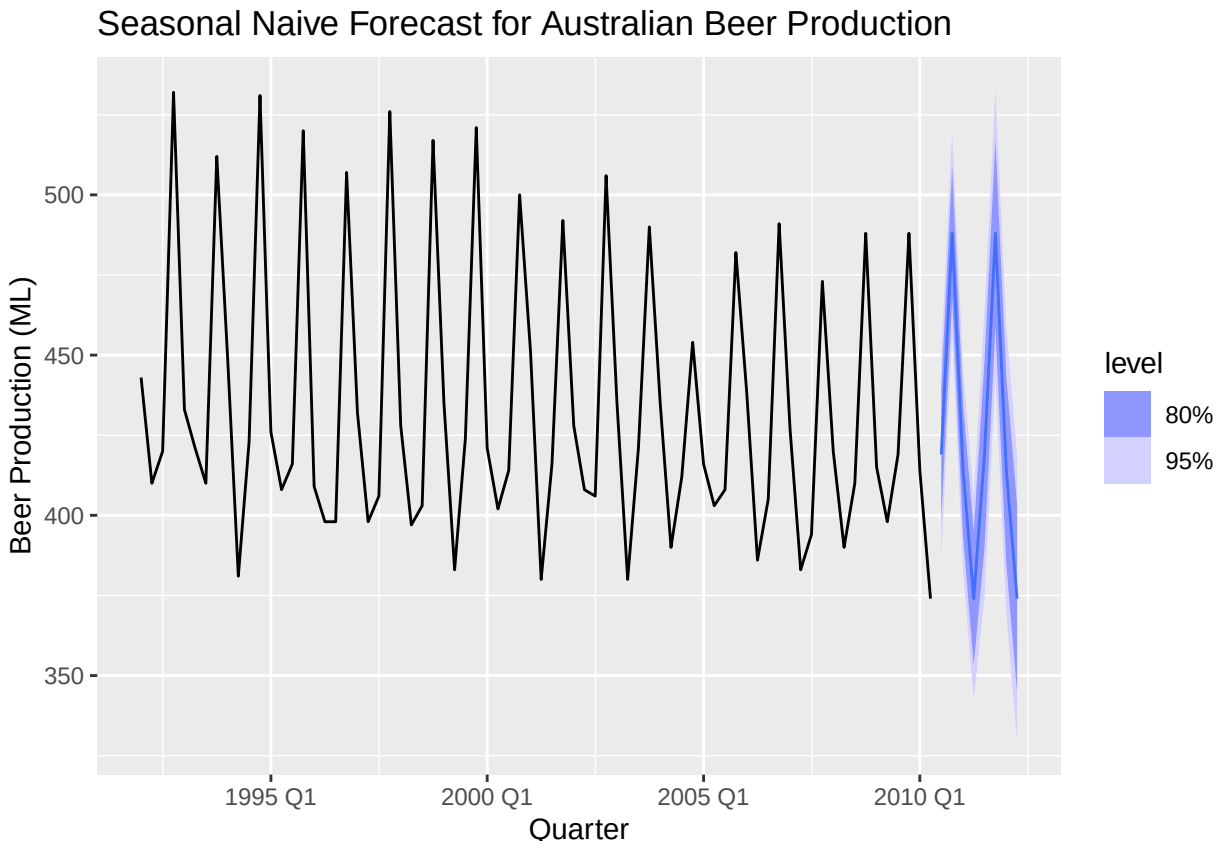
```
## Warning: Removed 4 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

```
## Warning: Removed 4 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

```
## Warning: Removed 4 rows containing non-finite outside the scale range
## (`stat_bin()`).
```




```
fit %>%
  forecast() %>%
  autoplot(beer_cut) +
  labs(title = "Seasonal Naive Forecast for Australian Beer Production",
       x = "Quarter",
      y = "Beer Production (ML)")
```



5.4: Previous exercise for Australian Exports series from global_economy and the Bricks series from aus_production

This questions says to repeat 5.3 for two more datasets:

Naive forecast for Australian exports

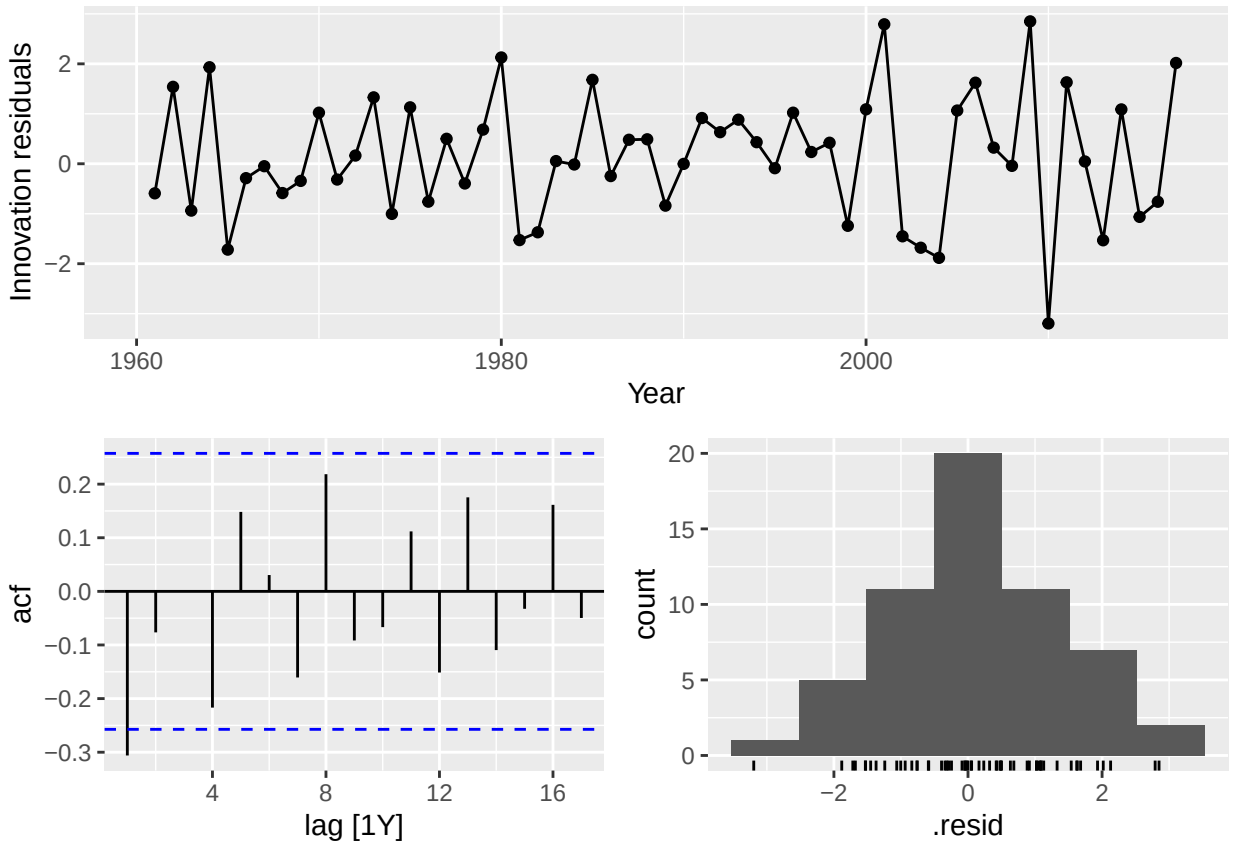
A Naive forecast ksn't that helpful for this dataset and the high residuals show that. The exports have been on a steady upward trend and a straight line out from the last point doesn'capture it meaningfully.

```
aussie_exports <- global_economy %>%
  filter(Country == "Australia")

fit_exports <- aussie_exports %>%
  model(NAIVE(Exports))

fit_exports %>%
  gg_tsresiduals()
```

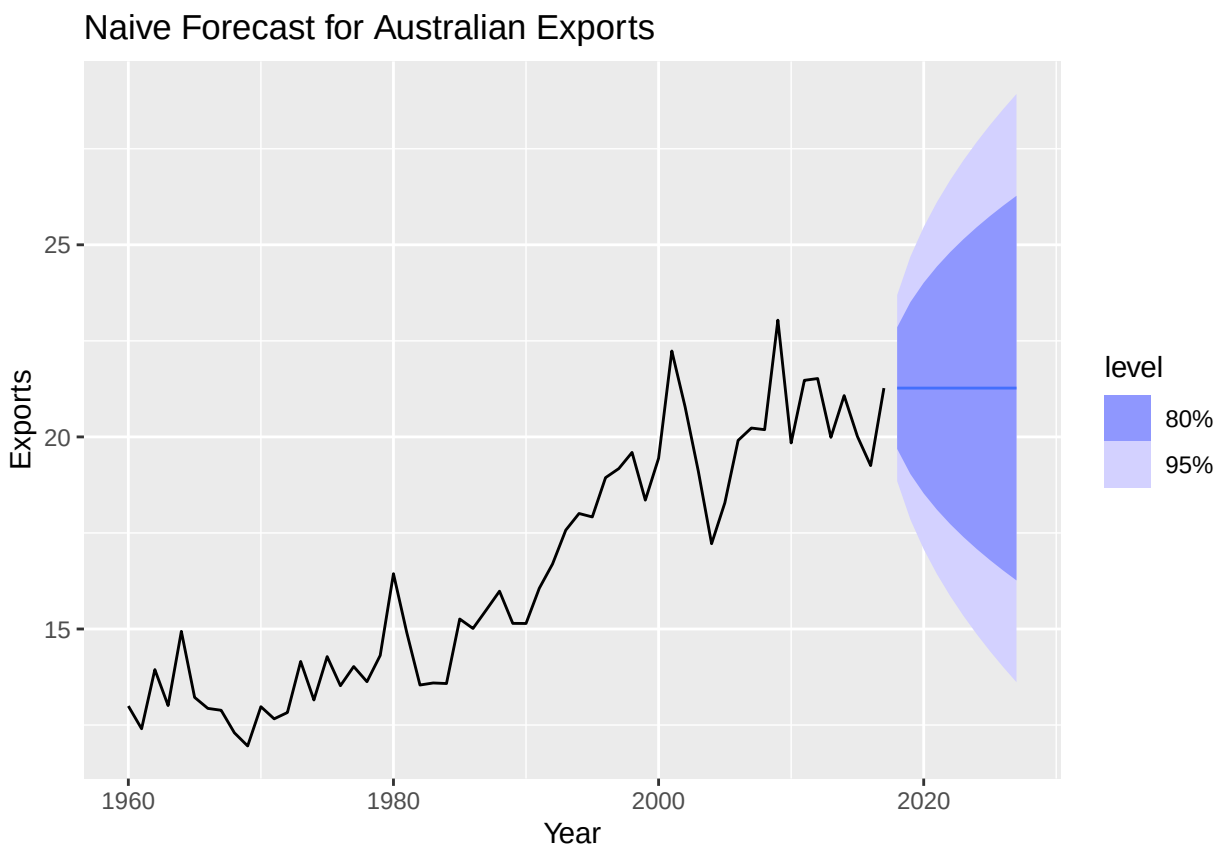
```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_point()`).
```

```
## Warning: Removed 1 row containing non-finite outside the scale range
## (`stat_bin()`).
```




```
fit_exports %>%
  forecast(h = 10) %>%
  autoplot(aussie_exports) +
  labs(title = "Naive Forecast for Australian Exports",
       x = "Year",
       y = "Exports")
```



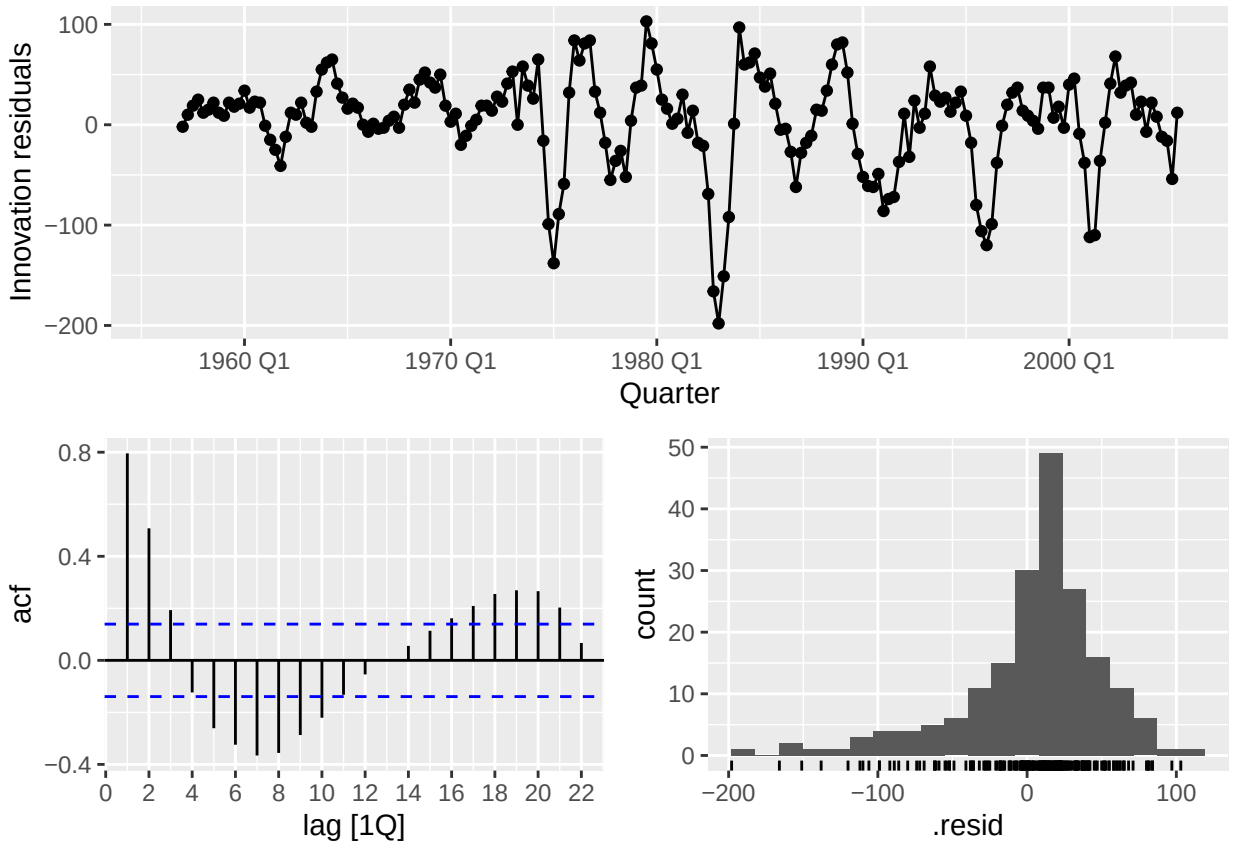
SNaive forecast for Bricks from aus_production SNaive works better here because the data has a cycliccal nature of some sort. The low resisduants relative to dataset size shows it working better.

```
aussie_bricks<- aus_production %>%
  filter(!is.na(Bricks))

fit_bricks <- aussie_bricks %>%
  model(SNAIVE(Bricks))

fit_bricks %>%
  gg_tsresiduals()
```

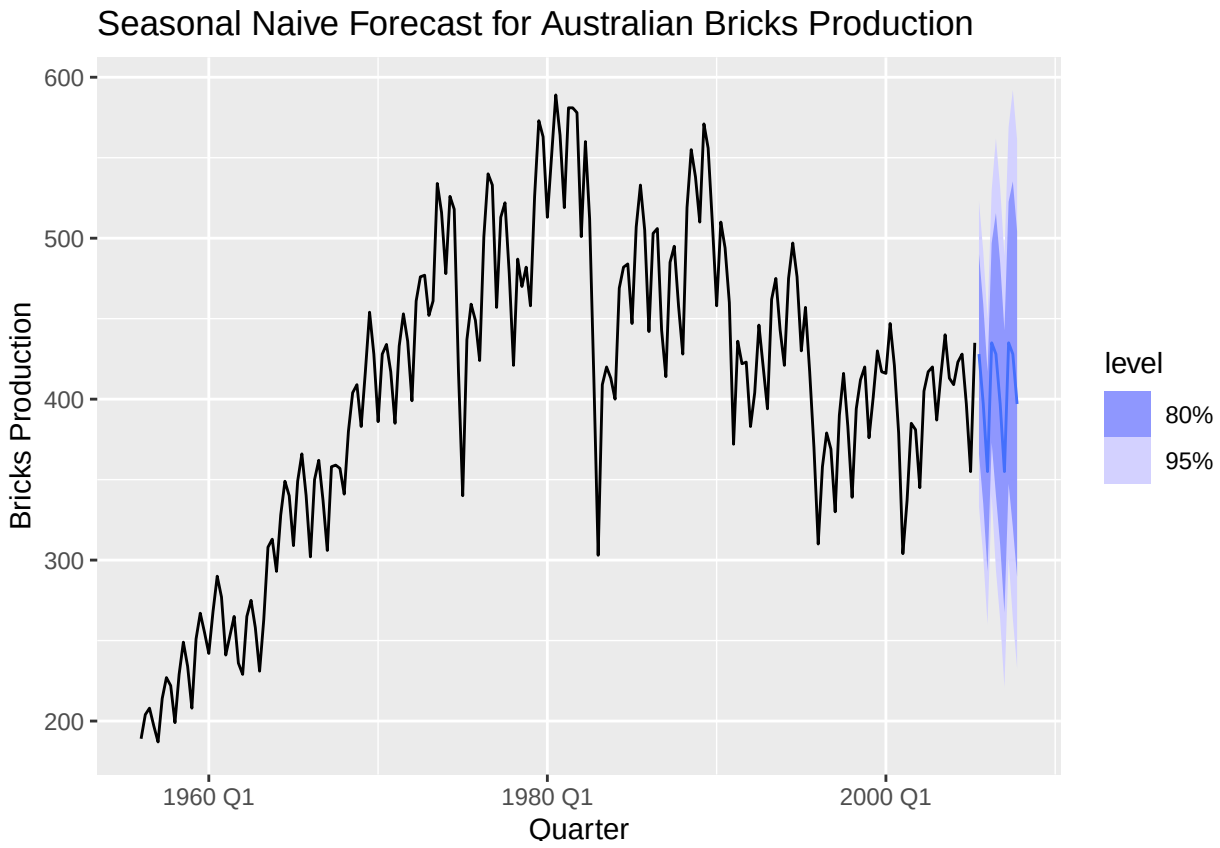
```
## Warning: Removed 4 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

```
## Warning: Removed 4 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

```
## Warning: Removed 4 rows containing non-finite outside the scale range
## (`stat_bin()`).
```




```
fit_bricks %>%
  forecast(h = 10) %>%
  autoplot(aussie_bricks) +
  labs(title = "Seasonal Naive Forecast for Australian Bricks Production",
       x = "Quarter",
       y = "Bricks Production")
```



5.7 A series of questions about aus_retail

This exercise has components A through G, with A through F using code provided by the exercise. My task was to take care to swap in aus_retail data.

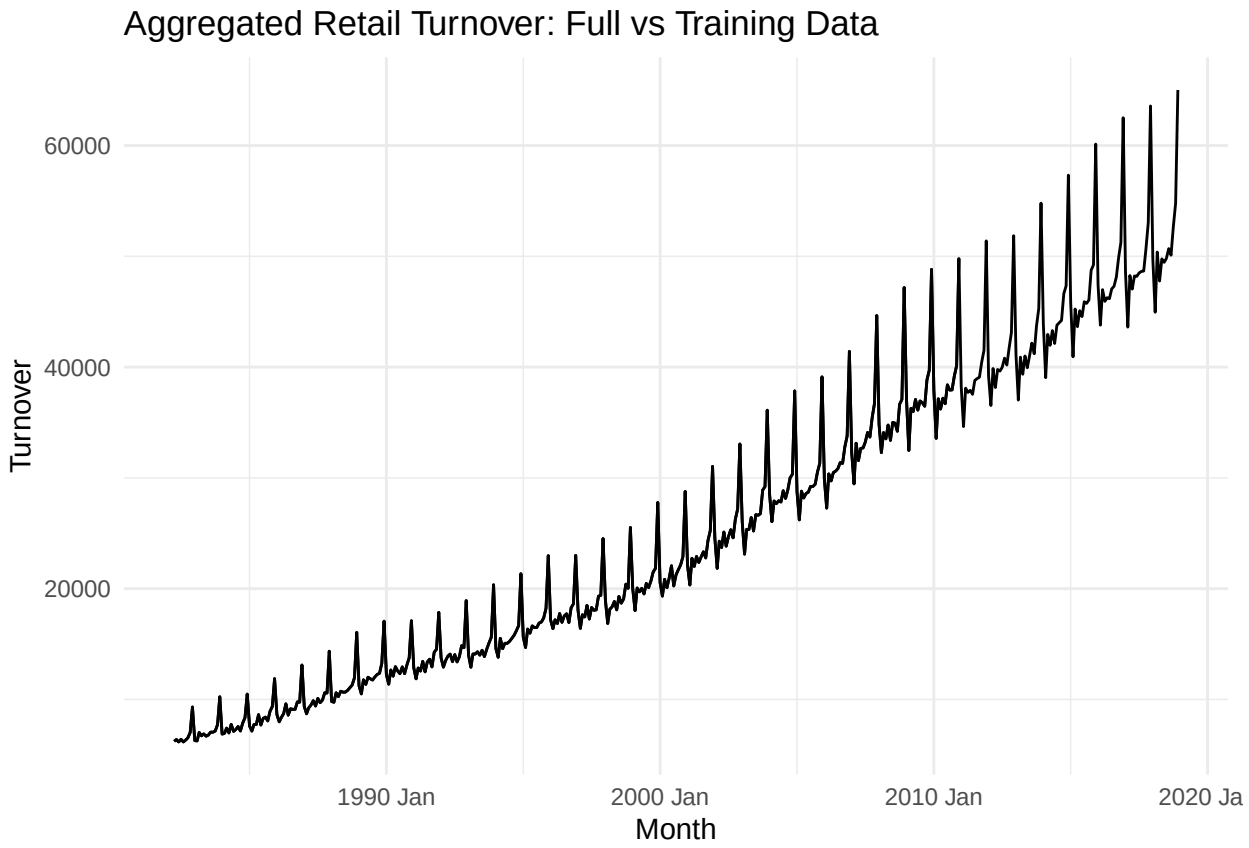
A: Create a training dataset consisting of observations before 2011 using provided code:

Here's the R provided by the exercise. I swapped in aus_retail data but otherwise kept it the same.

```
aus_agg <- aus_retail |>
  index_by(Month) |>
  summarise(Turnover = sum(Turnover))

myseries_train <- aus_retail |>
  filter(year(Month) < 2011)

myseries_train_agg <- myseries_train |>
  index_by(Month) |>
  summarise(Turnover = sum(Turnover))
```

B: Check that your data have been split appropriately by producing the following plot

```
autoplot(aus_agg, Turnover) +
  autolayer(myseries_train_agg, Turnover) +
  labs(title = "Aggregated Retail Turnover: Full vs Training Data", x = "Month", y = "Turnover") +
  theme_minimal()
```



C: # Fit a seasonal naive model to the training data

```
fit <- myseries_train_agg |>
  model(SNAIVE(Turnover))
```

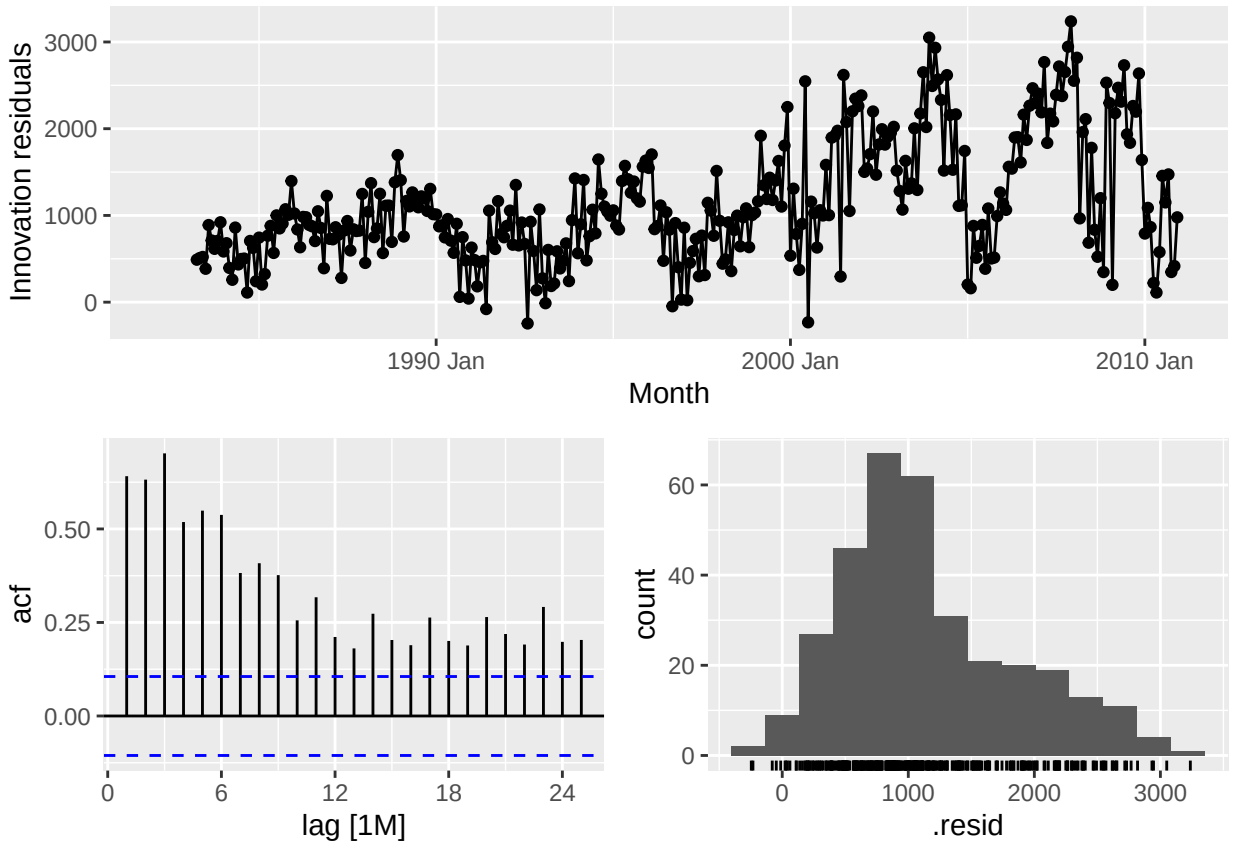
D: Check the residuals

```
fit |> gg_tsresiduals()
```

```
## Warning: Removed 12 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

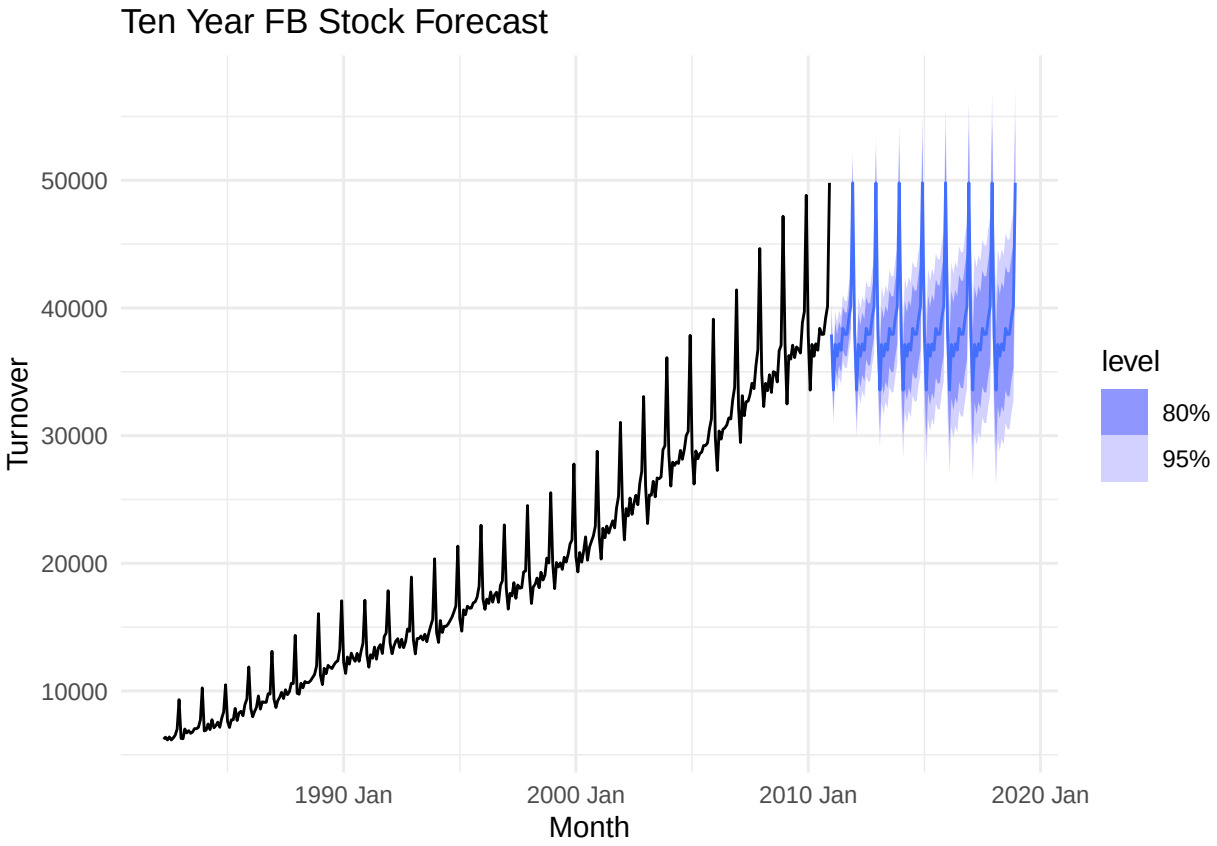
```
## Warning: Removed 12 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

```
## Warning: Removed 12 rows containing non-finite outside the scale range
## (`stat_bin()`).
```



E: Produce forecasts for the test data

```
new_aus <- aus_agg |>
  filter(year(Month) >= 2011)

fc <- fit |>
  forecast(new_data = new_aus)

autoplot(myseries_train_agg, Turnover) +
  autolayer(fc, Turnover) +
  labs(title = "Ten Year FB Stock Forecast", x = "Month", y = "Turnover") +
  theme_minimal()
```



F: Compare the accuracy of the forecasts

```
fc_accuracy <- fc |>
  accuracy(aus_agg)

fc_accuracy
```

```
## # A tibble: 1 x 10
##   .model           .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1
##   <chr>            <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 SNAIVE(Turnover) Test  6962. 8089. 6962.  14.6  14.6  6.05  6.04 0.945
```

G: How sensitive are the accuracy measures to the amount of training data used?

More training data would generally produce tighter accuracy rates. The model needs a sufficient baseline to be able to detect patterns and adapt to the seasonality trends. I did group by month as opposed to forecasts by state or industry and that reduced the dataset down to around 300 points.

The biggest risk os more training data is overfitting, where the model understands the training data very well but can't generalize well. When I train models in Python I use gradient descent and monitor loss errors to try and determine the appropriate model parameters and hyperparameters.