FPP Chapter 5 HW: The Forecaster's Toolbox

Alex Ptacek

Load Packages

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

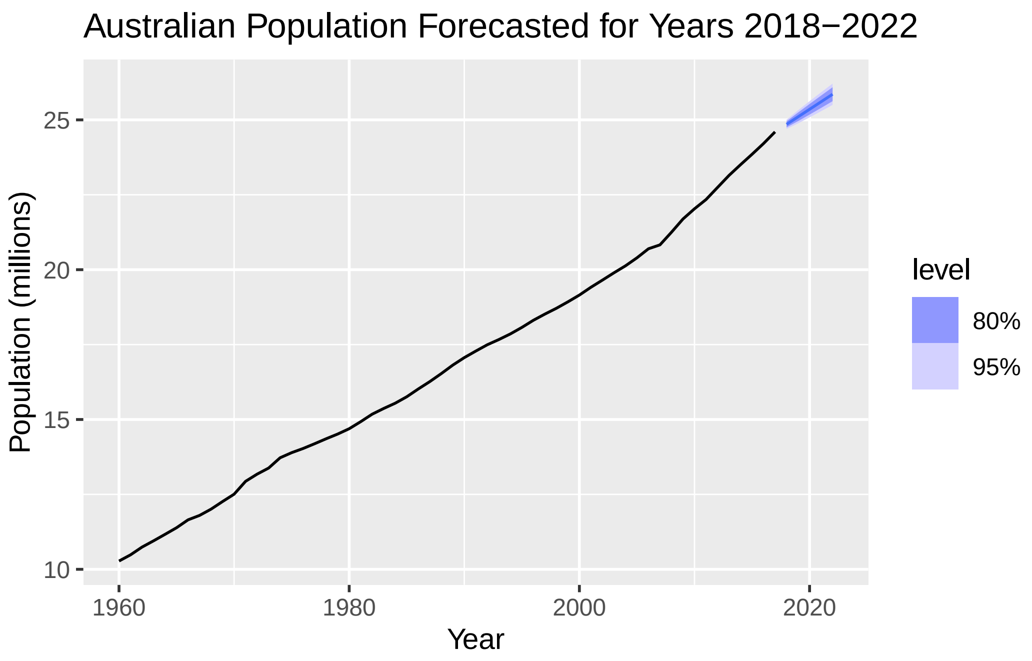
Question 5.1

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

a. Australian Population (global_economy)

Answer: I observed a strong linear trend in the Australian population time series, so it makes the most sense to use the drift method, because it creates a simple linear forecast based on the first and last observation.

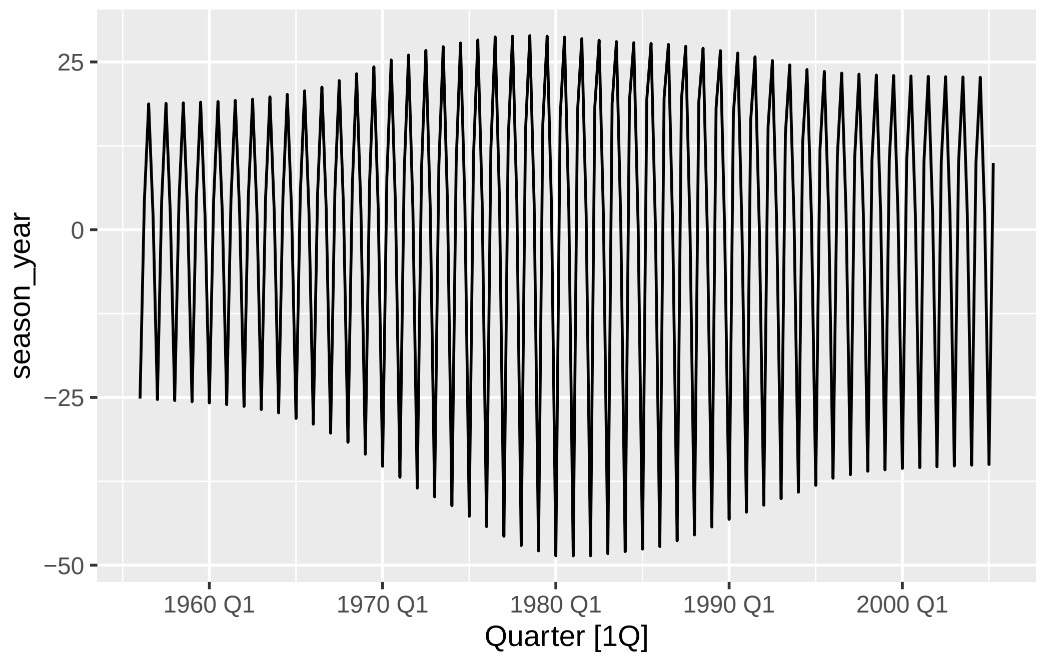
```
aus_population <- global_economy |>
  filter(Country == "Australia") |>
  mutate(Population = Population/1e6) |>
  select(1:3, Population)

aus_population |>
  model(Drift = RW(Population ~ drift())) |>
  forecast(h = 5) |>
  autoplot(aus_population) +
  ylab('Population (millions)') +
  ggtitle("Australian Population Forecasted for Years 2018-2022")
```



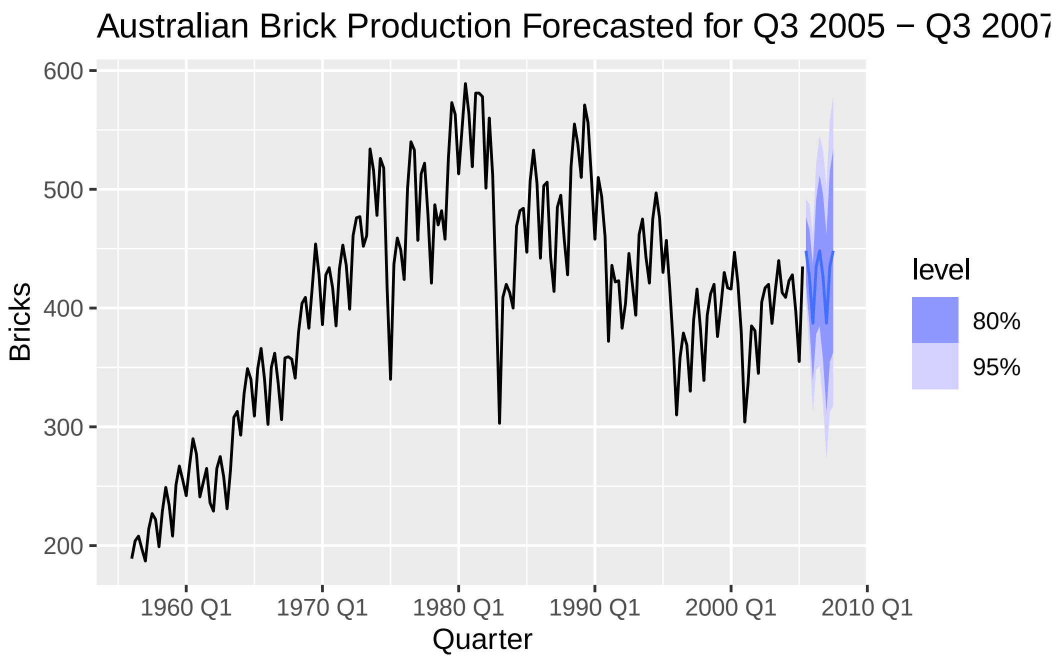
b. Bricks (aus_production)

Answer: There was a clear seasonal pattern in the data, so I ran a decomposed forecast to forecast season seperately from the rest of the data before adding the forecasts.

```
aus_bricks <- aus_production |>
  select(1, Bricks) |>
  filter(!is.na(Bricks))

# Observe seasonal pattern
aus_bricks |>
  model(STL(Bricks ~ season(window = 20))) |>
  components() |>
  select(-Bricks, -trend, -remainder, -season_adjust) |>
  autoplot()
```

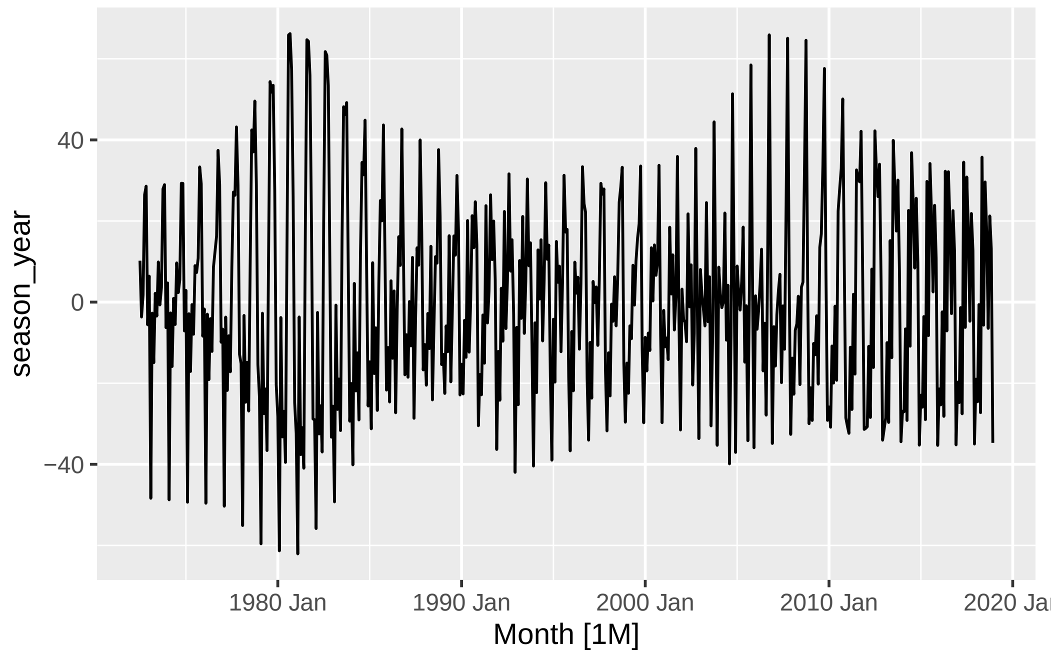


```
# Forecast season and seasonally adjusted data separately and add together
aus_bricks |>
  model(stlf = decomposition_model(STL(Bricks ~ season(window = "periodic")), NAIVE(season_ac
  forecast(h = 9) |>
  autoplot(aus_bricks) +
  ggtitle("Australian Brick Production Forecasted for Q3 2005 - Q3 2007")
```

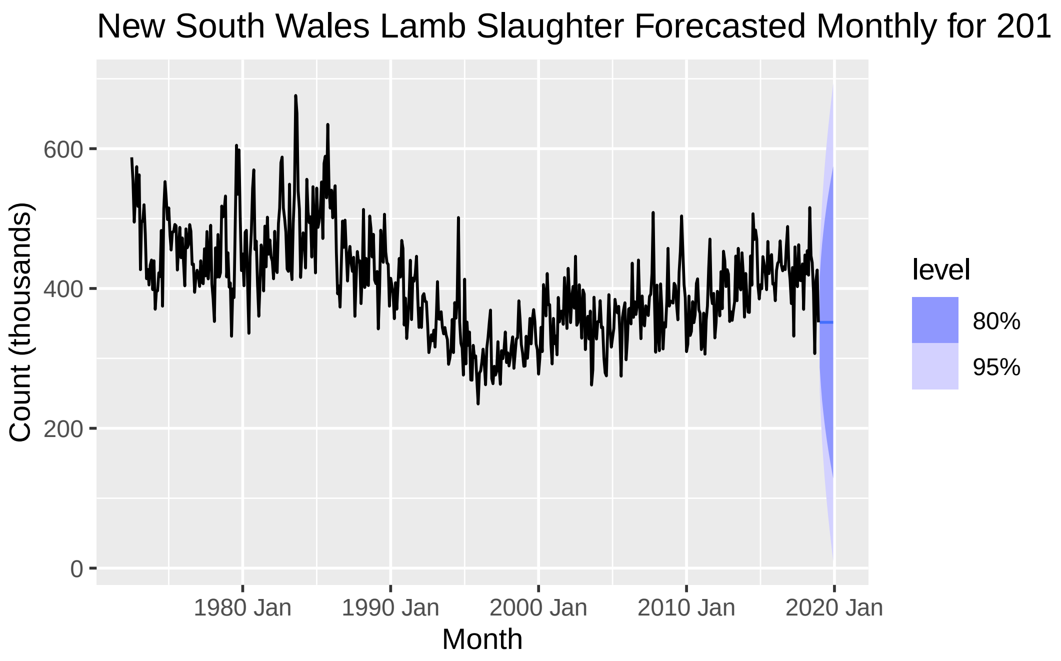


c. NSW Lambs (aus_livestock)

```
nsw_lambs <- aus_livestock |>
  filter(State == "New South Wales" & Animal == "Lambs") |>
  mutate(Count = Count/1e3)

# Check for seasonality
nsw_lambs |>
  model(STL(Count)) |>
  components() |>
  select(-Count, -trend, -remainder, -season_adjust) |>
  autoplot()
```

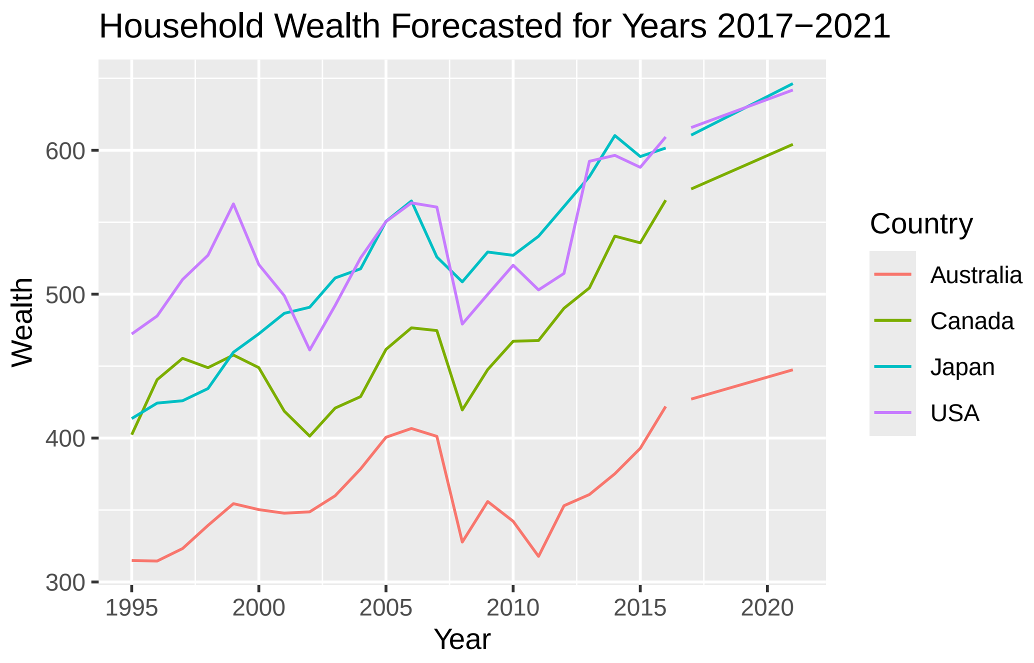


```
nsw_lambs |>
  model(naive = NAIVE(Count)) |>
  forecast(h = 12) |>
  autoplot(nsw_lambs) +
  labs(title = "New South Wales Lamb Slaughter Forecasted Monthly for 2019",
       y = "Count (thousands)")
```



d. Household wealth (hh_budget)

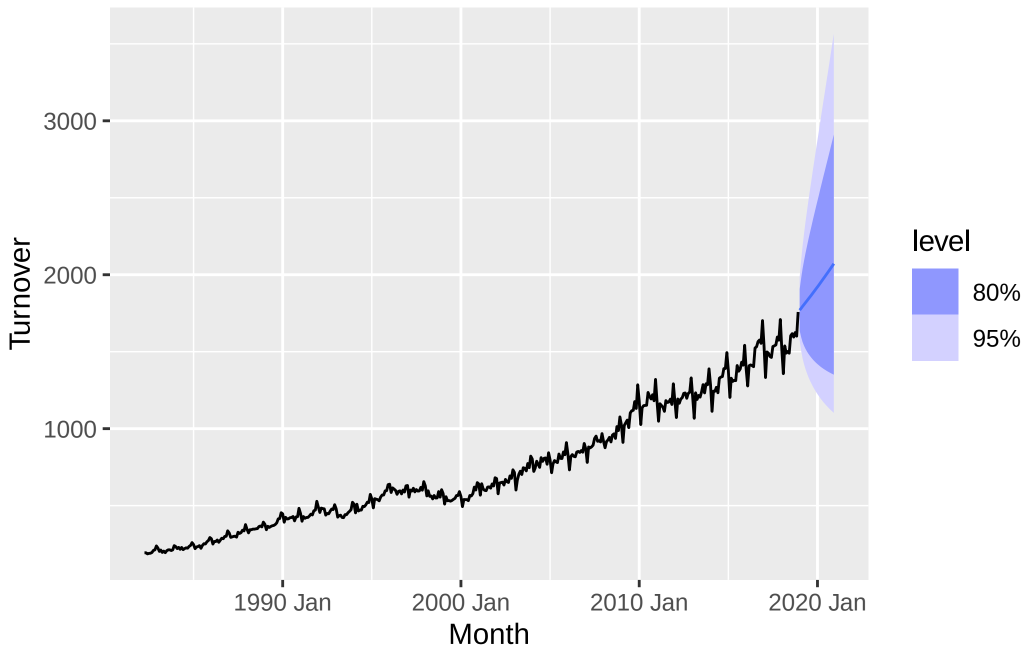
```
hh_wealth <- hh_budget |>
  select(1, 2, Wealth)

hh_wealth |>
  model(drift = RW(Wealth ~ drift())) |>
  forecast(h = 5) |>
  ggplot(aes(x = Year, y = .mean, color = Country)) +
  geom_line() +
  geom_line(data = hh_wealth, aes(x = Year, y = Wealth, color = Country)) +
  labs(title = "Household Wealth Forecasted for Years 2017-2021",
       y = "Wealth")
```



e. Australian takeaway food turnover (aus_retail)

This data needed a log transform. Thankfully, the forecast function back transforms and removes bias for us.

```
aus_food <- aus_retail |>
  filter(Industry == "Takeaway food services") |>
  tibble() |>
  group_by(Industry, Month) |>
  summarise(Turnover = sum(Turnover)) |>
  tsibble(index = Month, key = Industry)

aus_food |>
  model(drift = RW(log(Turnover) ~ drift())) |>
  forecast() |>
  autoplot(aus_food)
```

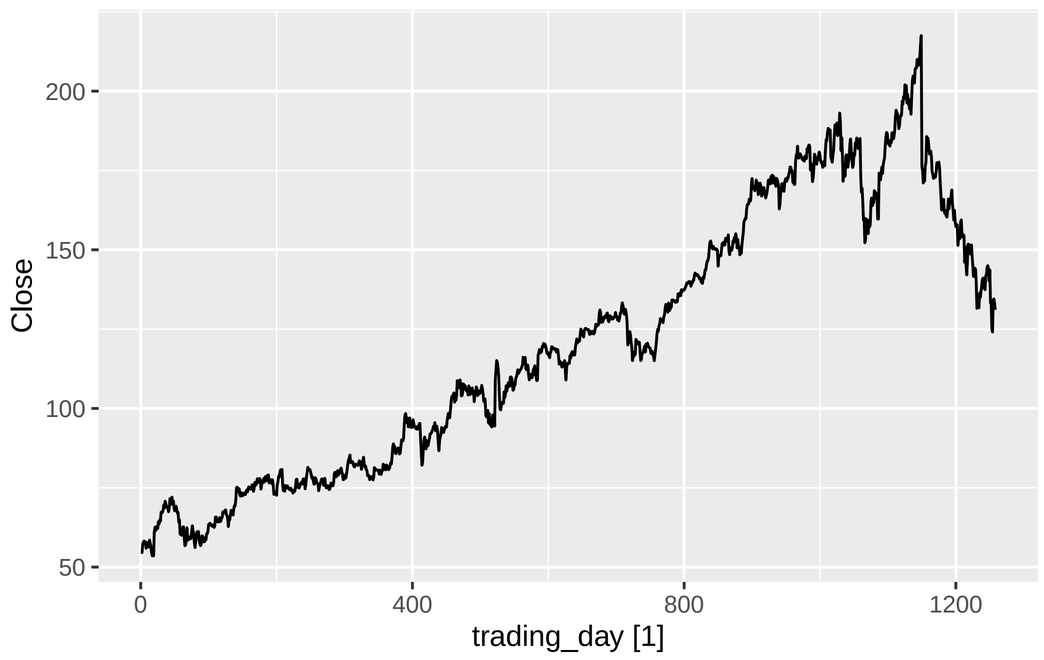


Question 5.2

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

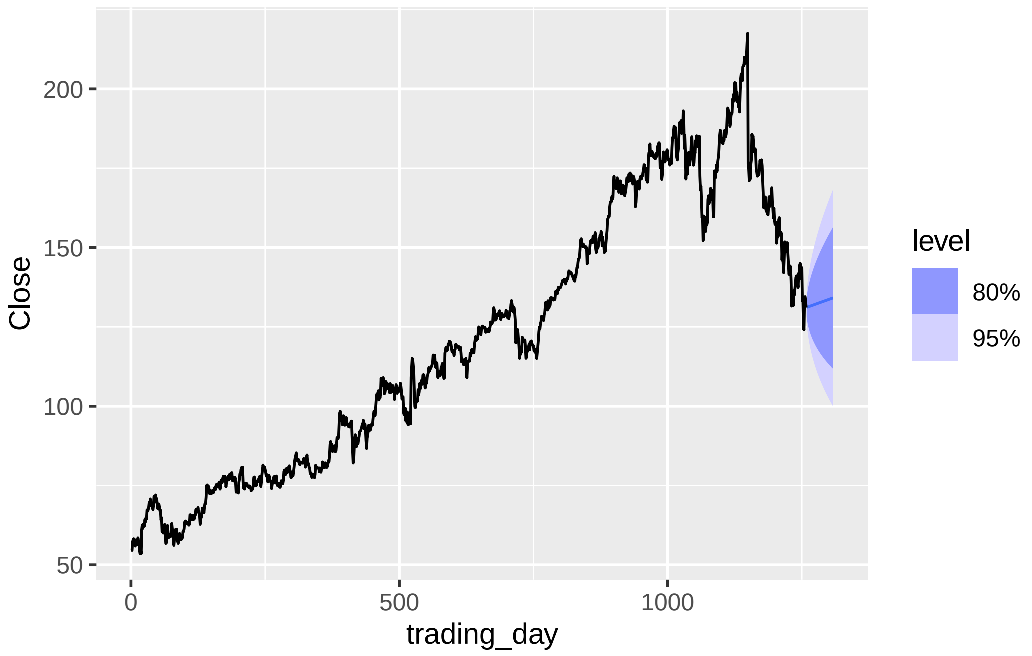
a. Produce a time plot of the series.

```r
fb_stock <- gafa_stock |>
  filter(Symbol == "FB") |>
  mutate(trading_day = row_number()) |>
  update_tsibble(index = trading_day, regular = TRUE) |>
  select(Symbol, Close)

fb_stock |>
  autoplot(Close)
```



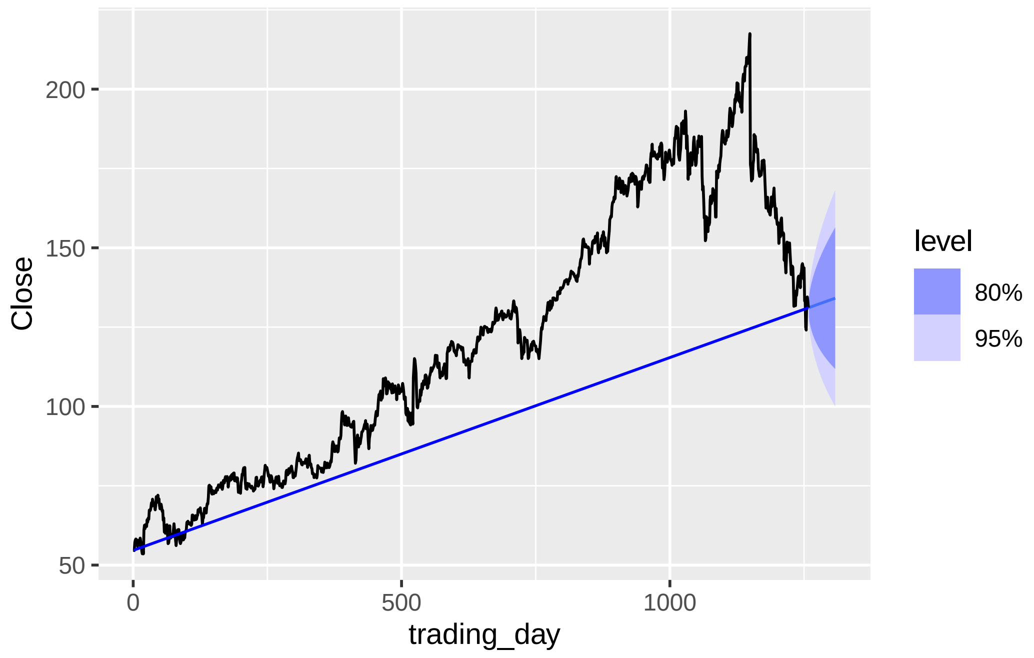
b. Produce forecasts using the drift method and plot them.

```
fb_drift_forecast <- fb_stock |>
  model(Drift = RW(Close ~ drift())) |>
  forecast(h = 50) |>
  autoplot(fb_stock)
fb_drift_forecast
```



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

```r
# Filter original data set to only min and max values
fb_max_min <- fb_stock |>
  filter(trading_day == min(trading_day) | trading_day == max(trading_day))

# Plot stock forecast and add straight line from min to max
fb_drift_forecast +
  geom_line(data = fb_max_min, aes(x = trading_day, y = Close), color = "blue")
```



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

The Naive benchmark model is the best for this this time series, because it has the highest skill score using the CRPS method for evaluating model accuracy. However, we must consider that the conditions for our residuals are not met, which will greatly affect our prediction intervals. Combined with the fact that the drift method is nearly as accurate as the naive method, I am not certain that the Naive model is the best. As of yet, we do not have the tools to correct data with residuals which do not satisfy the conditions for modelling.

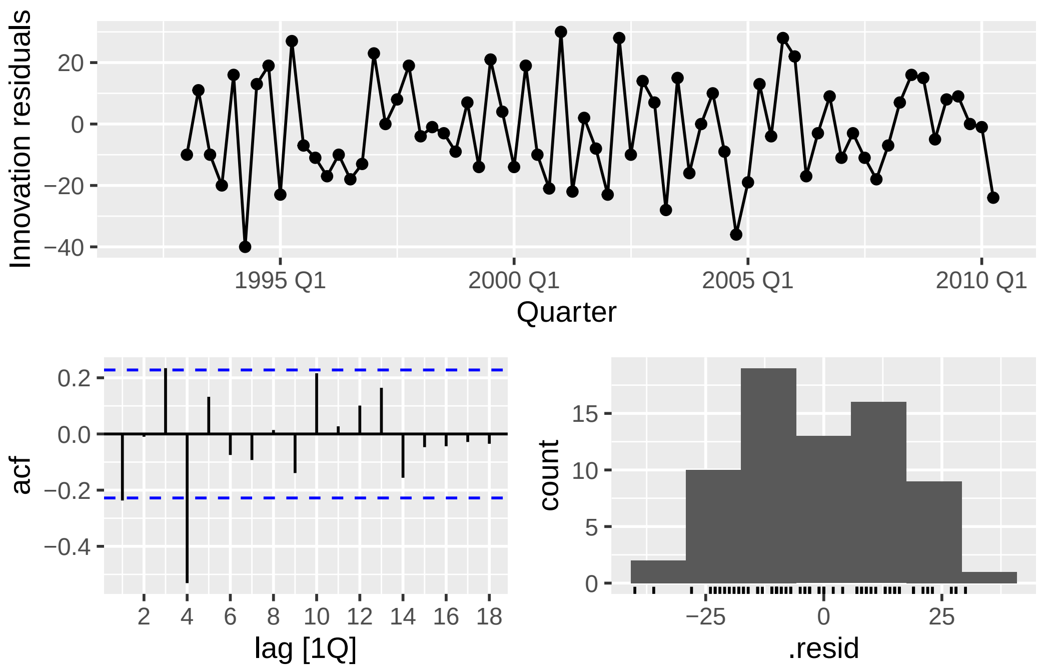
```
fb_many_train_sets <- fb_stock |>
  stretch_tsibble(.init = 10, .step = 1) |>
  filter(.id != max(.id))

fb_many_train_sets |>
  model(naive = NAIVE(Close),
        mean = MEAN(Close),
        drift = RW(Close ~ drift())) |>
  forecast(h = 50) |>
  accuracy(fb_stock, list(skill = skill_score(CRPS)))
```

```
# A tibble: 3 x 4
  .model Symbol .type skill
  <chr>  <chr>  <chr> <dbl>
1 drift  FB     Test  0.947
2 mean   FB     Test  0.730
3 naive  FB     Test  0.948
```

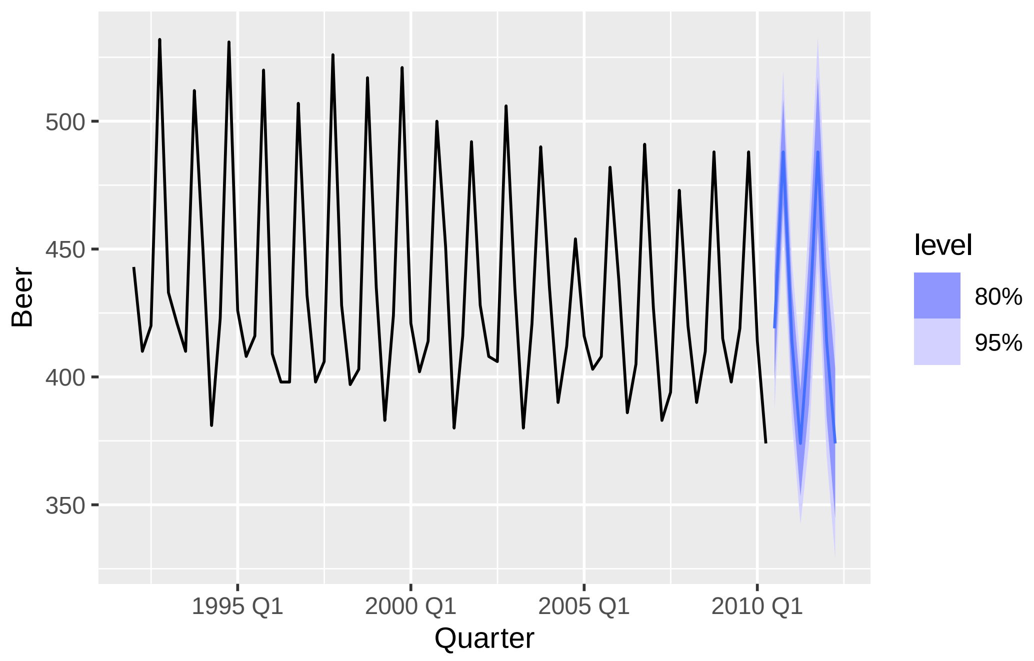
Question 5.3

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)
```



What do you conclude?

Answer: Based on the residual diagnostic plots, there is high residual autocorrelation at lag 4. Even though there is only one lag with autocorrelation, which is often acceptable, I decided to run an autocorrelation chi-squared test with the ljung-box method, to test its significance. As we can see from the p-value below, the likelihood of our stat appearing coincidentally is extremely low (about 0.1%). Therefore, there is likely autocorrelation in our residuals, which will heavily affect our prediction intervals. Additionally, there is likely "information" in our residuals that can be used to improve our forecast model.

```
augment(fit) |>
  features(.innov, ljung_box, lag = 20)
```

```
# A tibble: 1 x 3
  .model       lb_stat lb_pvalue
  <chr>          <dbl>     <dbl>
1 SNAIVE(Beer)    45.4  0.000973
```

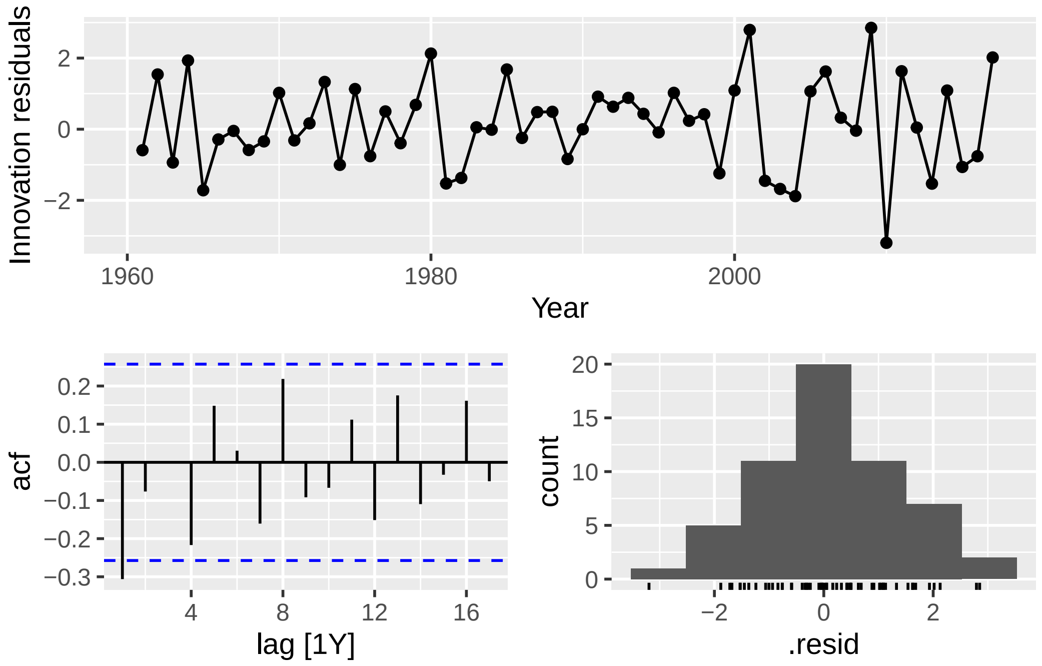
Question 5.4

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.

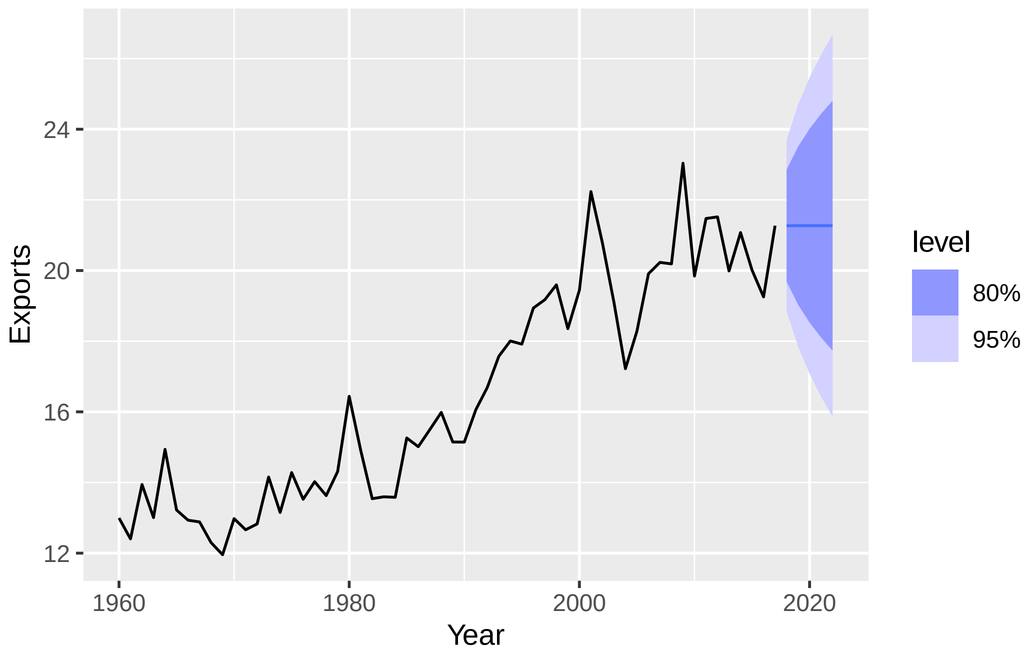
a. Australian Exports (global_economy)

Answer: The Naive model is more appropriate here because yearly time series do not have seasonality. Based on the residual diagnostics plots, autocorrelation and normal distribution are likely met. However, the variance does not appear constant. Residuals' variance is greater towards the end of the time series. This will heavily affect our prediction parameters.

```r
aus_exports <- global_economy |>
  filter(Country == "Australia") |>
  select(Country, Year, Exports)

# Fit naive model
exports_fit <- aus_exports |>
  model(naive = NAIVE(Exports))

exports_fit |> gg_tsresiduals()
```



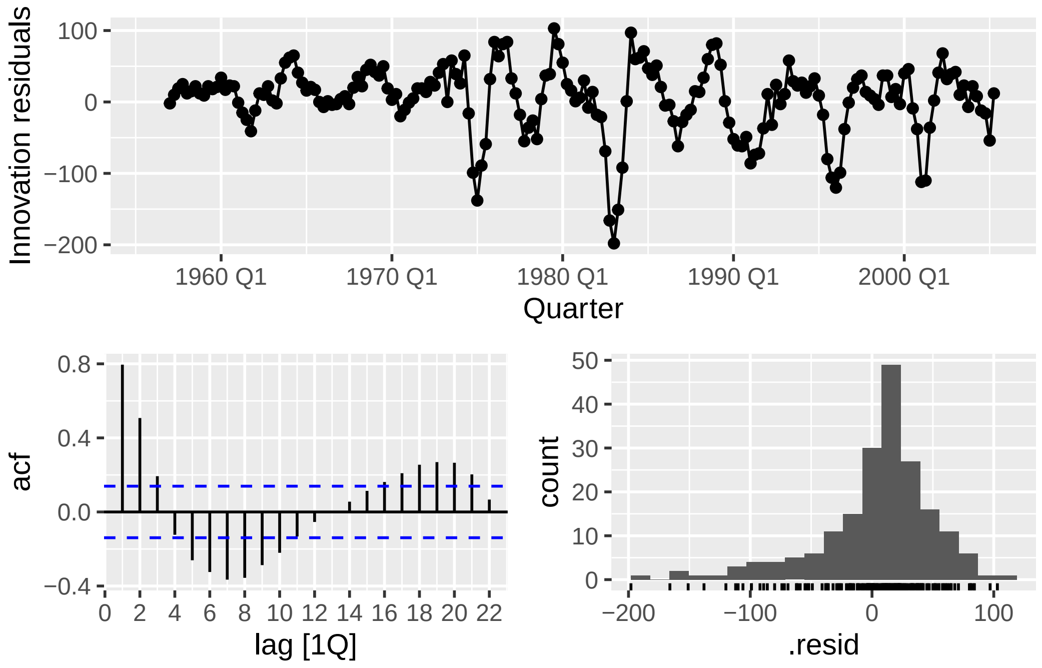
```r
exports_fit |> forecast(h = 5) |> autoplot(aus_exports)
```



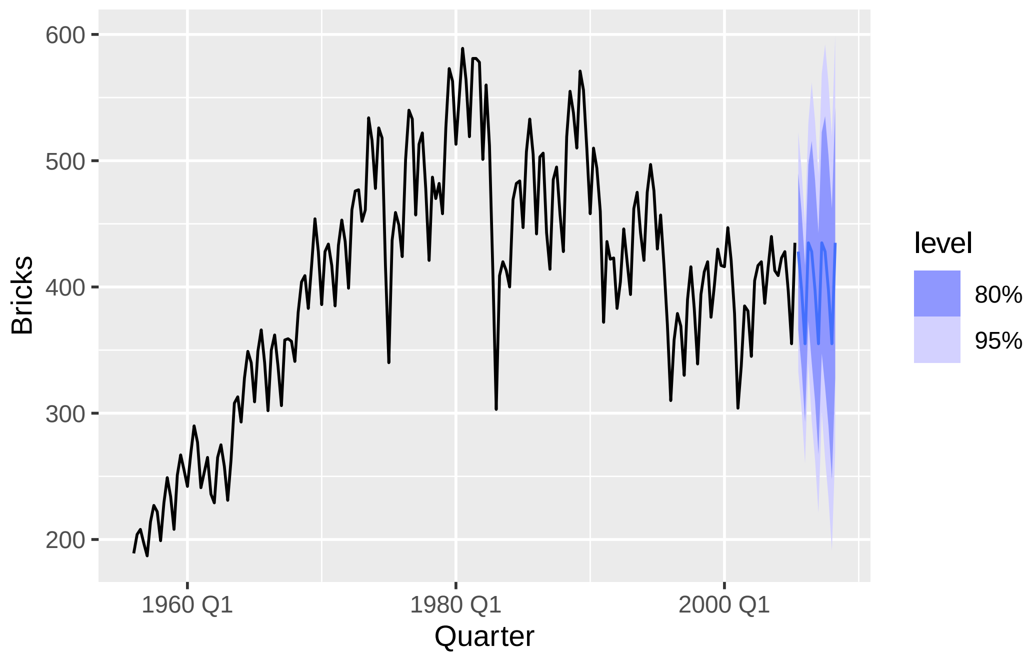
b. Bricks (aus_production)

Answer: The Seasonal Naive model is more appropriate here because the data is seasonal. Based on the residuals diagnostics plots, variance is not constant, distribution is not normal, and there is autocorrelation. Therefore, conditions have not been met and our prediction intervals will be heavily impacted.

```r
bricks_fit <- aus_bricks |>
  model(snaive = SNAIVE(Bricks))

bricks_fit |> gg_tsresiduals()
```



```r
bricks_fit |> forecast(h = "3 years") |> autoplot(aus_bricks)
```



```
augment(bricks_fit) |>
  features(.innov, ljung_box, lag = 12)
```

```
# A tibble: 1 x 3
  .model lb_stat lb_pvalue
  <chr>    <dbl>     <dbl>
1 snaive    305.         0
```

Question 5.7

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

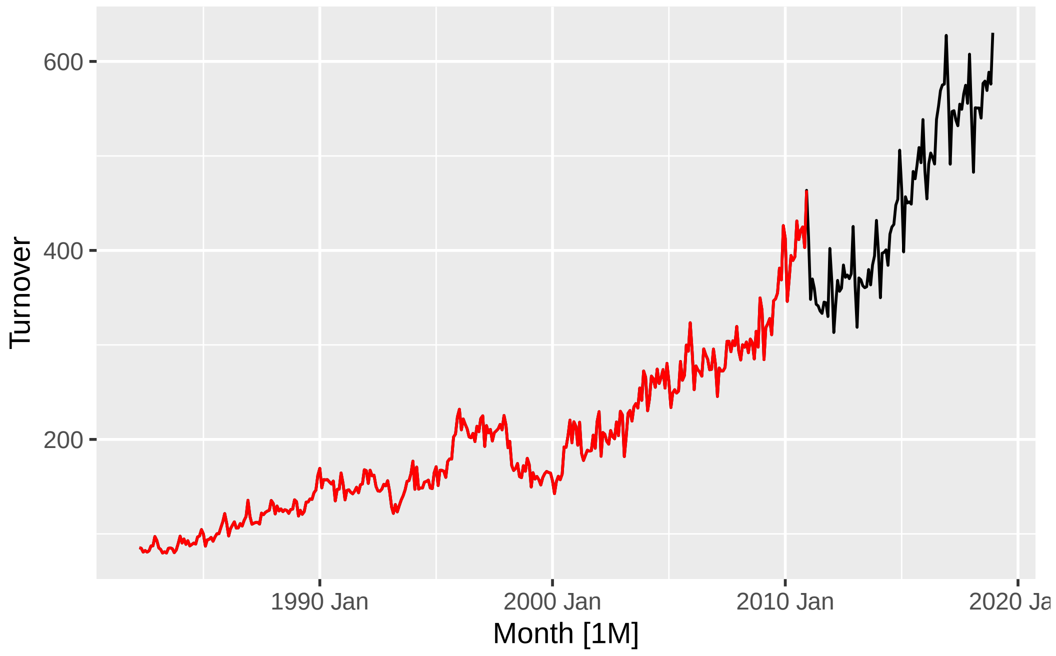
```
set.seed(624)
myseries <- aus_retail |>
  filter(`Series ID` == sample(aus_retail$`Series ID`, 1))
```

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).

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

d. Check the residuals. Do the residuals appear to be uncorrelated and normally distributed?

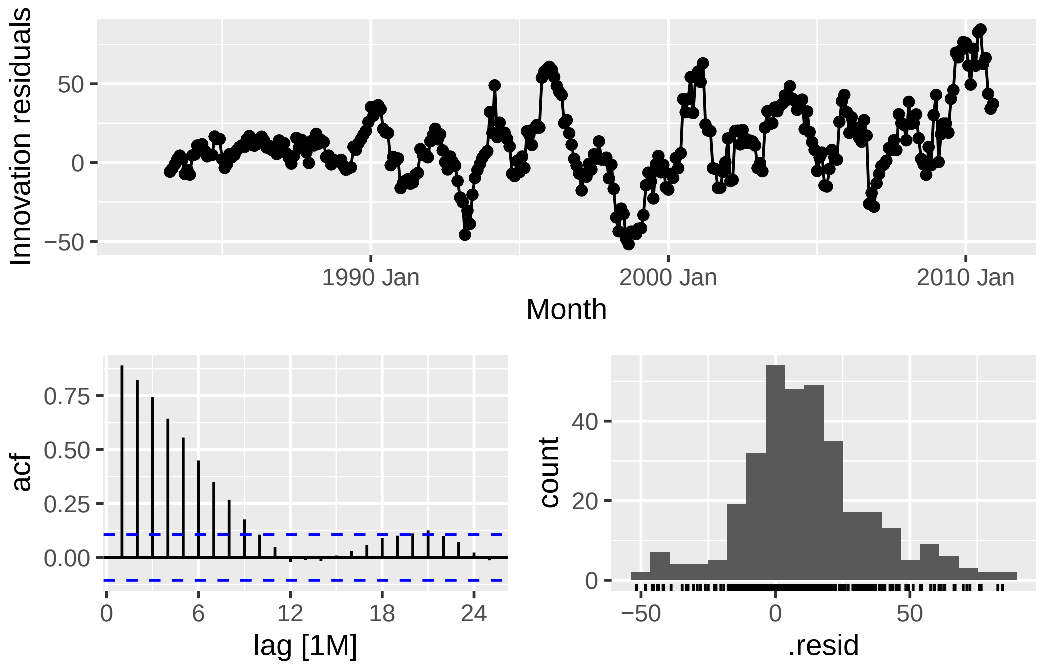
Answer: The residuals do not appear to be normal distributed, although it looks somewhat normal.

```
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()`).
```

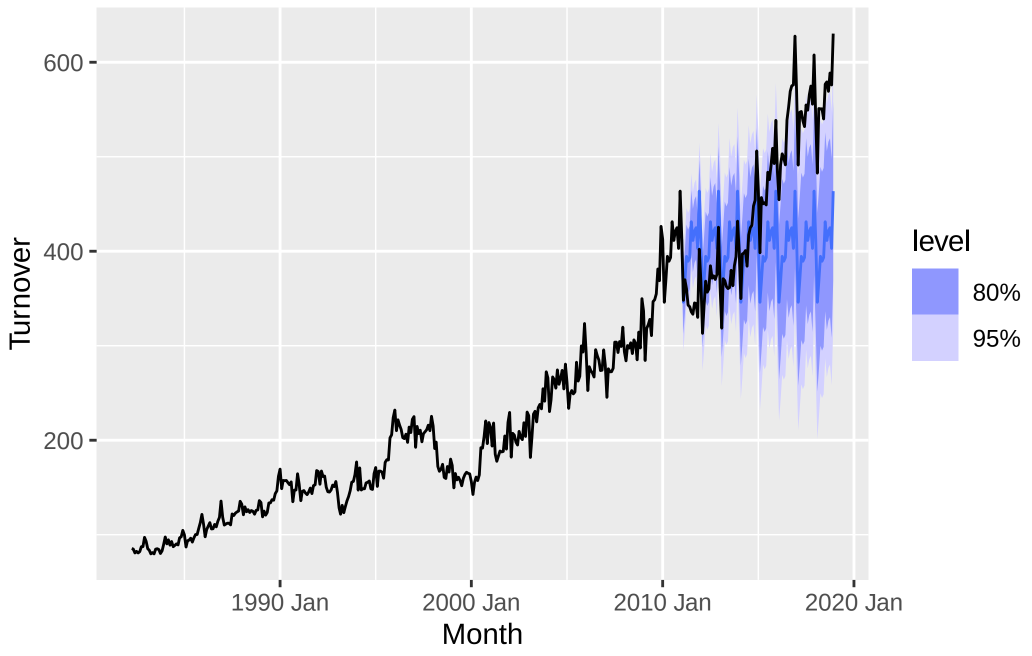


e. Produce forecasts for the test data

```r
fc <- fit |> forecast(new_data = anti_join(myseries, myseries_train))
```

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

```r
fc |> autoplot(myseries)
```



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

```
fit |> accuracy()
```

```
# A tibble: 1 x 12
  State    Industry .model .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1
  <chr>    <chr>    <chr>  <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 New Sou~ Takeawa~ SNAIV~ Trai~  11.5  26.1  19.2  4.81  9.59     1     1 0.890
```

```
fc |> 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
```

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

Answer: Based on the below table, we can see that the accuracy measures are extremely sensitive to the amount of training data used, because even one year less of data results in much higher error. Interestingly, increasing the amount of training data used also greatly increases error.

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

fit <- myseries_train |> model(SNAIVE(Turnover))

fc <- fit |> forecast(new_data = anti_join(myseries, myseries_train))

fc |> 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
```