Data 624_Exercise 5.11_HW3

Jiaxin Zheng

2025-02-17

5.11 Exercises:

```r
library(fpp3)
```

```
## Warning: package 'fpp3' was built under R version 4.4.2

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

## -- Attaching packages -------------------------------------------- fpp3 1.0.1 --

## v tibble      3.2.1     v tsibble     1.1.6
## v dplyr       1.1.4     v tsibbledata 0.4.1
## v tidyr       1.3.1     v feasts      0.4.1
## v lubridate   1.9.4     v fable       0.4.1
## v ggplot2     3.5.1

## Warning: package 'dplyr' was built under R version 4.4.3

## Warning: package 'ggplot2' was built under R version 4.4.2

## Warning: package 'tsibbledata' was built under R version 4.4.2

## Warning: package 'feasts' was built under R version 4.4.2

## Warning: package 'fabletools' was built under R version 4.4.2

## Warning: package 'fable' was built under R version 4.4.2

## -- 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()
```

```r
library(dplyr)
```

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)
- b.Bricks (aus_production)
- c.NSW Lambs (aus_livestock)
- d.Household wealth (hh_budget).
- e.Australian takeaway food turnover (aus_retail).

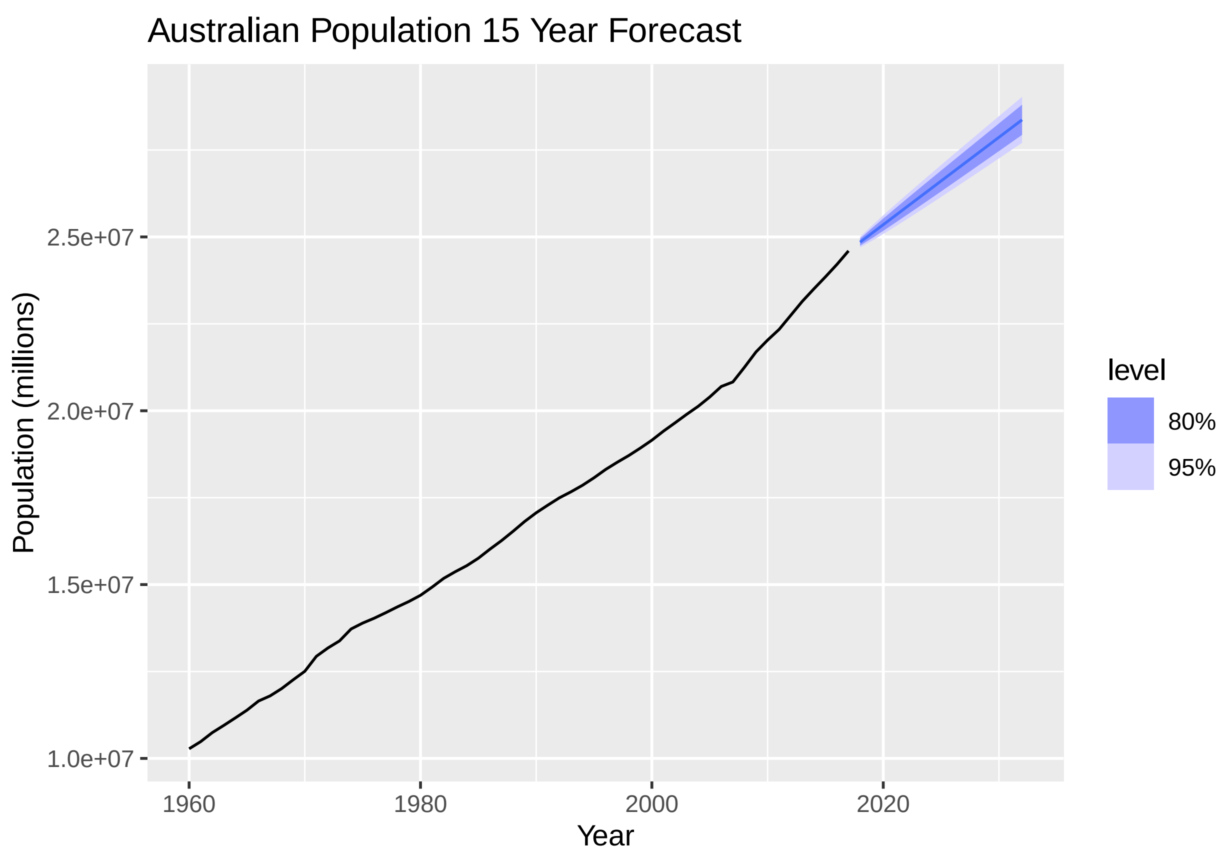
a. Australian Population (global_economy)

```r
head(global_economy)
```

```
## # A tsibble: 6 x 9 [1Y]
## # Key:       Country [1]
##   Country     Code   Year         GDP Growth   CPI Imports Exports Population
##   <fct>       <fct> <dbl>       <dbl>  <dbl> <dbl>   <dbl>   <dbl>      <dbl>
## 1 Afghanistan AFG    1960  537777811.     NA    NA    7.02    4.13    8996351
## 2 Afghanistan AFG    1961  548888896.     NA    NA    8.10    4.45    9166764
## 3 Afghanistan AFG    1962  546666678.     NA    NA    9.35    4.88    9345868
## 4 Afghanistan AFG    1963  751111191.     NA    NA   16.9     9.17    9533954
## 5 Afghanistan AFG    1964  800000044.     NA    NA   18.1     8.89    9731361
## 6 Afghanistan AFG    1965 1006666638.     NA    NA   21.4    11.3     9938414
```

For the Australian Population data, the RW(~drift()) model is more appropriate for this case. In this case we forecast the population in Australia for next 15 years. It captures the average increase over the years.

```r
aus_pop_fc <- global_economy %>%
  filter(Country == 'Australia') %>%
  model(RW(Population~drift())) %>%
  forecast(h = 15)

aus_pop_fc %>%
  autoplot(global_economy) +
  labs(title = "Australian Population 15 Year Forecast", x = "Year", y = "Population (millions)")
```



b.Bricks (aus_production)

```
head(aus_production)
```

```
## # A tsibble: 6 x 7 [1Q]
##   Quarter  Beer Tobacco Bricks Cement Electricity   Gas
##     <qtr> <dbl>   <dbl>  <dbl>  <dbl>       <dbl> <dbl>
## 1 1956 Q1   284    5225    189    465        3923     5
## 2 1956 Q2   213    5178    204    532        4436     6
## 3 1956 Q3   227    5297    208    561        4806     7
## 4 1956 Q4   308    5681    197    570        4418     6
## 5 1957 Q1   262    5577    187    529        4339     5
## 6 1957 Q2   228    5651    214    604        4811     7
```

```
?aus_production
```

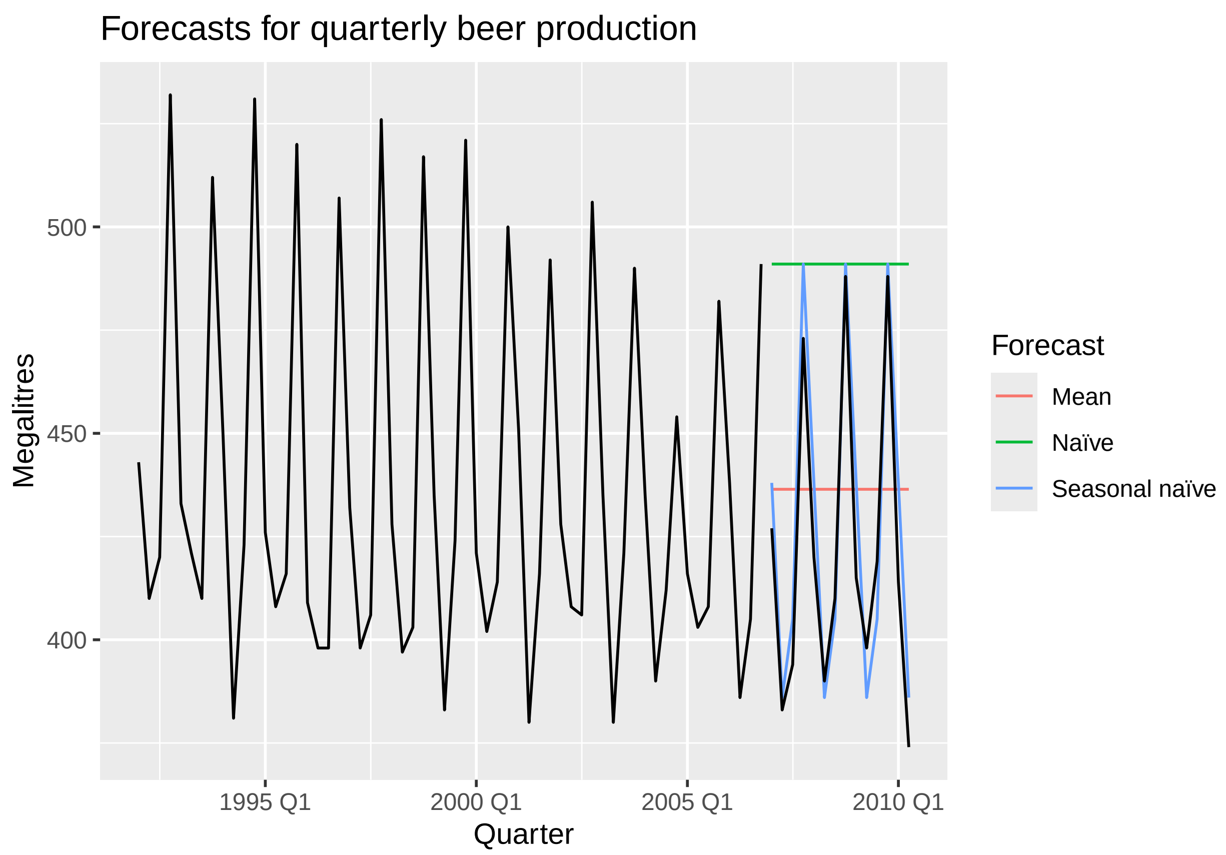
```
## starting httpd help server ... done
```

For the aus_production data, the mean(), NAIVE(), SNAIVE() are all appropriate in this case.

```
train <- aus_production %>%
  filter_index("1992 Q1" ~ "2006 Q4")
```

```r
beer_fit <- train %>%
  model(
    Mean = MEAN(Beer),
    `Naïve` = NAIVE(Beer),
    `Seasonal naïve` = SNAIVE(Beer)
  )

beer_fc <- beer_fit %>%  forecast(h = 14)

beer_fc %>%
  autoplot(train, level = NULL) +
  autolayer(
    filter_index(aus_production, "2007 Q1" ~ .),
    colour = "black"
  ) +
  labs(
    y = "Megalitres",
    title = "Forecasts for quarterly beer production"
  ) +
  guides(colour = guide_legend(title = "Forecast"))
```

```
## Plot variable not specified, automatically selected '.vars = Beer'
```



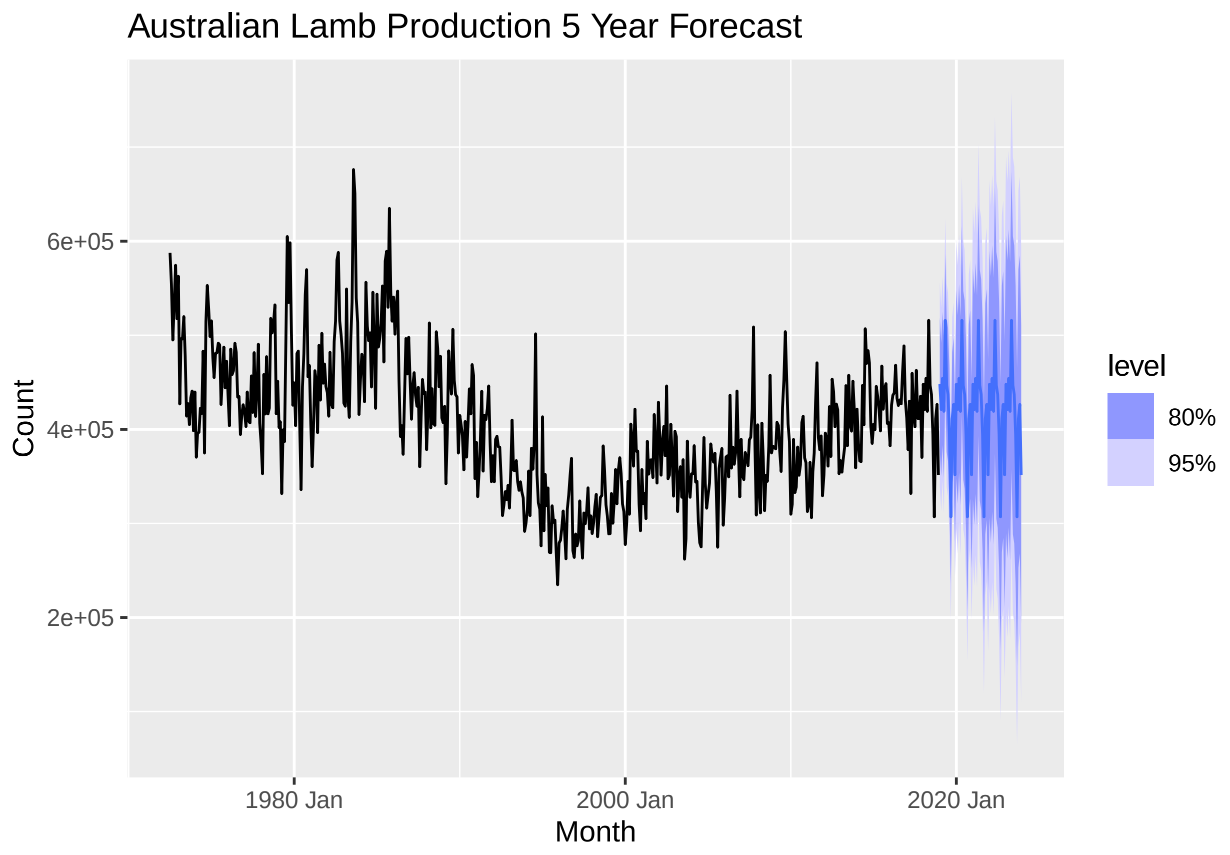
### c.NSW Lambs (aus_livestock)

```r
?aus_livestock
head(aus_livestock)
```

```
## # A tsibble: 6 x 4 [1M]
## # Key:       Animal, State [1]
##      Month Animal                     State                        Count
##      <mth> <fct>                      <fct>                        <dbl>
## 1 1976 Jul Bulls, bullocks and steers Australian Capital Territory  2300
## 2 1976 Aug Bulls, bullocks and steers Australian Capital Territory  2100
## 3 1976 Sep Bulls, bullocks and steers Australian Capital Territory  2100
## 4 1976 Oct Bulls, bullocks and steers Australian Capital Territory  1900
## 5 1976 Nov Bulls, bullocks and steers Australian Capital Territory  2100
## 6 1976 Dec Bulls, bullocks and steers Australian Capital Territory  1800
```

I considered the NAIVE(), But in this case, SNAIVE() is more appropriate.

```r
NSW_fc <- aus_livestock %>%
  filter(Animal == "Lambs", State == "New South Wales") %>%
  model(SNAIVE(Count)) %>%
  forecast(h = "5 years")

NSW_fc %>%
  autoplot(aus_livestock) +
  labs(title = "Australian Lamb Production 5 Year Forecast")
```



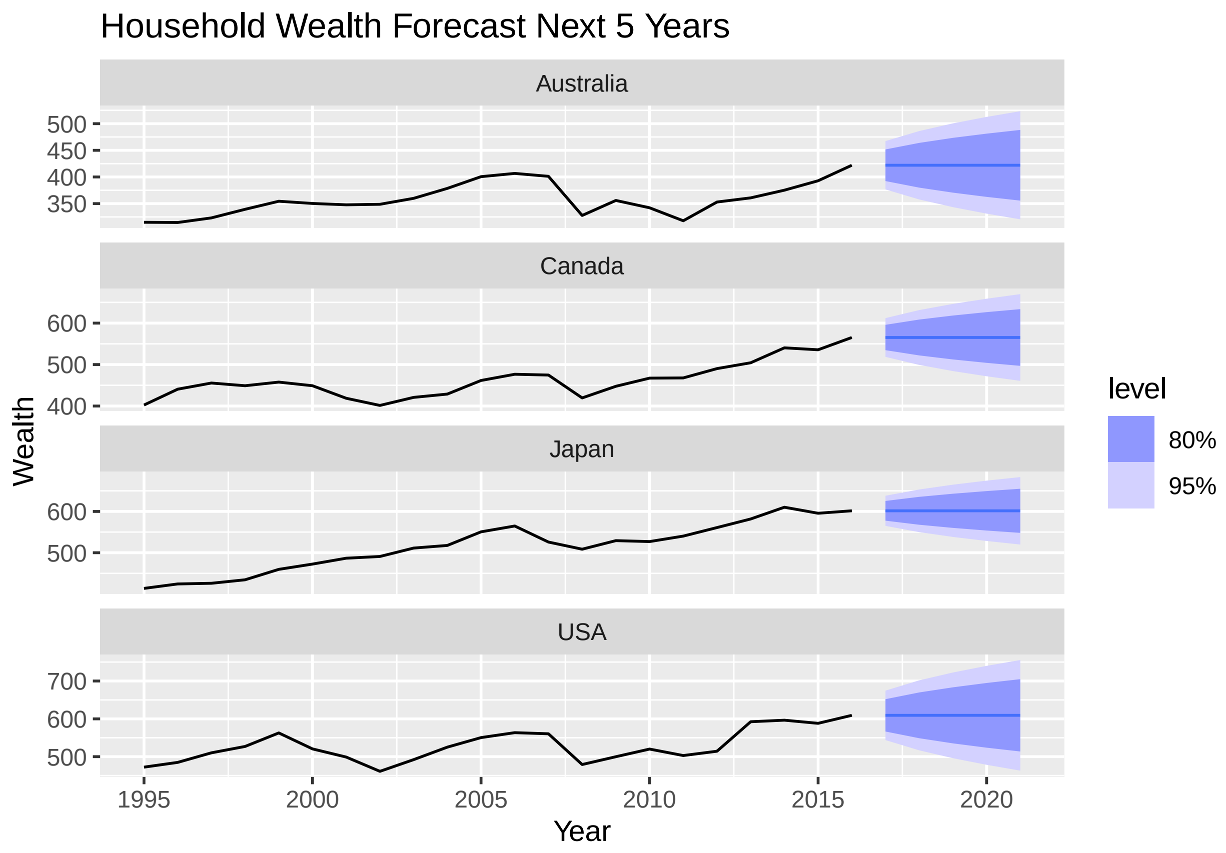
d.Household wealth (hh_budget).

```
?hh_budget
head(hh_budget)
```

```
## # A tsibble: 6 x 8 [1Y]
## # Key:       Country [1]
##   Country    Year  Debt    DI Expenditure Savings Wealth Unemployment
##   <chr>     <dbl> <dbl> <dbl>       <dbl>   <dbl>  <dbl>        <dbl>
## 1 Australia  1995  95.7  3.72        3.40   5.24    315.         8.47
## 2 Australia  1996  99.5  3.98        2.97   6.47    315.         8.51
## 3 Australia  1997 108.   2.52        4.95   3.74    323.         8.36
## 4 Australia  1998 115.   4.02        5.73   1.29    339.         7.68
## 5 Australia  1999 121.   3.84        4.26   0.638   354.         6.87
## 6 Australia  2000 126.   3.77        3.18   1.99    350.         6.29
```

I considered about SNAIVE(), but I think NAIVE() is more appropriate, because the data is in time series.

```
household_wealth_fc <- hh_budget %>%
  model(NAIVE(Wealth)) %>%
  forecast(h = "5 years")

household_wealth_fc %>%
  autoplot(hh_budget) +
  labs(title = "Household Wealth Forecast Next 5 Years")
```



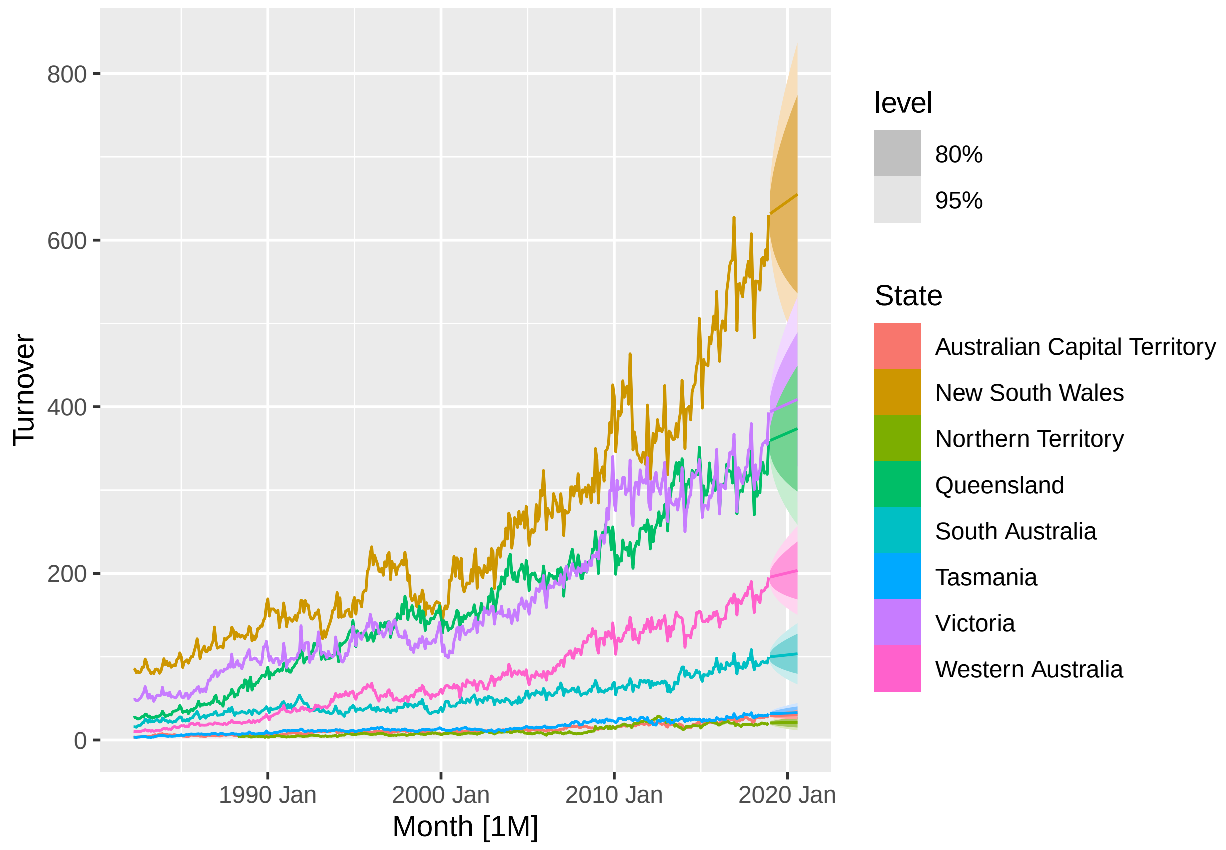
e.Australian takeaway food turnover (aus_retail).

```r
?aus_retail
head(aus_retail)
```

```
## # A tsibble: 6 x 5 [1M]
## # Key:       State, Industry [1]
##   State                        Industry            `Series ID`    Month Turnover
##   <chr>                        <chr>               <chr>          <mth>    <dbl>
## 1 Australian Capital Territory Cafes, restaurants~ A3349849A   1982 Apr      4.4
## 2 Australian Capital Territory Cafes, restaurants~ A3349849A   1982 May      3.4
## 3 Australian Capital Territory Cafes, restaurants~ A3349849A   1982 Jun      3.6
## 4 Australian Capital Territory Cafes, restaurants~ A3349849A   1982 Jul      4
## 5 Australian Capital Territory Cafes, restaurants~ A3349849A   1982 Aug      3.6
## 6 Australian Capital Territory Cafes, restaurants~ A3349849A   1982 Sep      4.2
```

This case, I use RW(y ~ drift()).

```r
aus_takeaway <- aus_retail |>
  filter(Industry == "Takeaway food services") %>%
  select(State, Month, Turnover)

aus_takeaway_fc <- aus_takeaway %>%
  model(RW(Turnover~drift())) %>%
  forecast(h = 20)

autoplot(aus_takeaway, Turnover) +
  autolayer(aus_takeaway_fc)
```



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

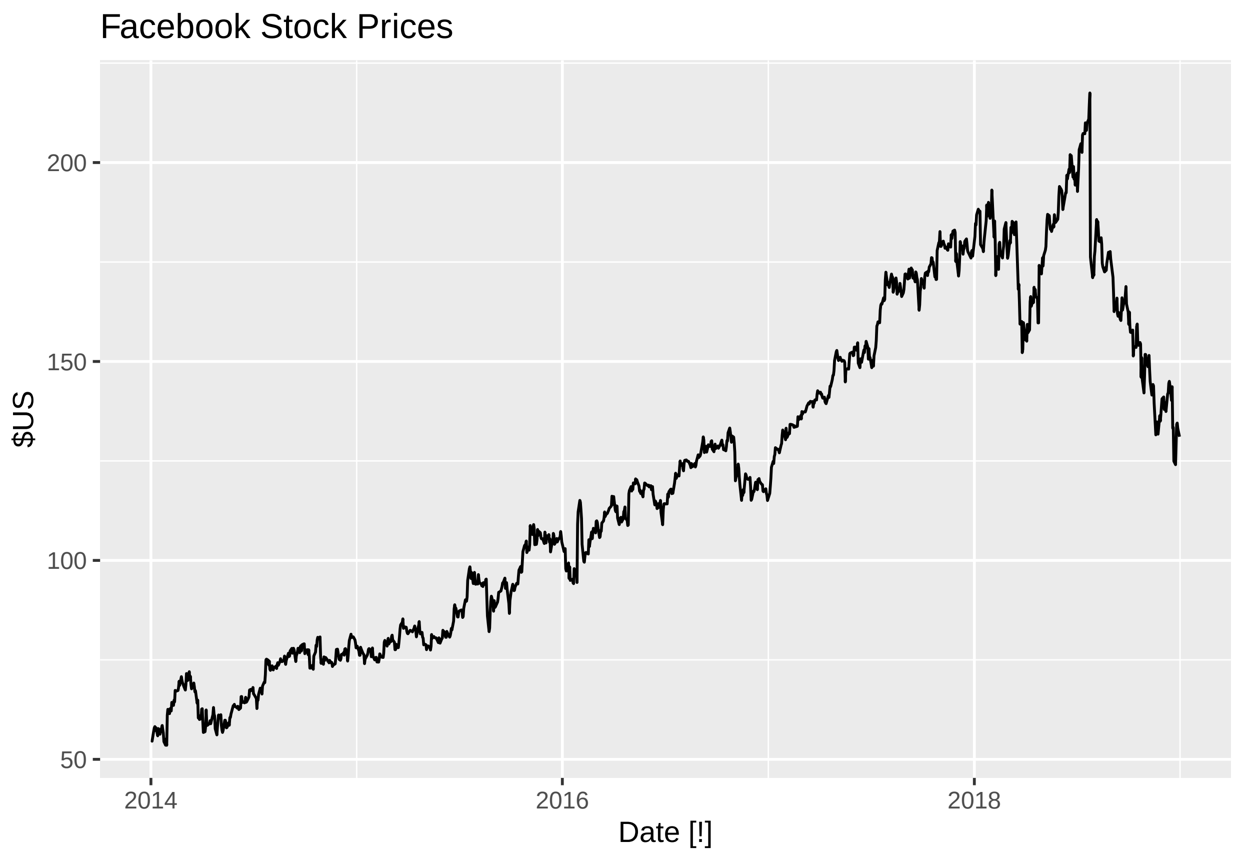
- a.Produce a time plot of the series.
- b.Produce forecasts using the drift method and plot them.
- c.Show that the forecasts are identical to extending the line drawn between the first and last observations.
- d.Try using some of the other benchmark functions to forecast the same data set. Which do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

a.Produce a time plot of the series.

```
?gafa_stock
head(gafa_stock)
```

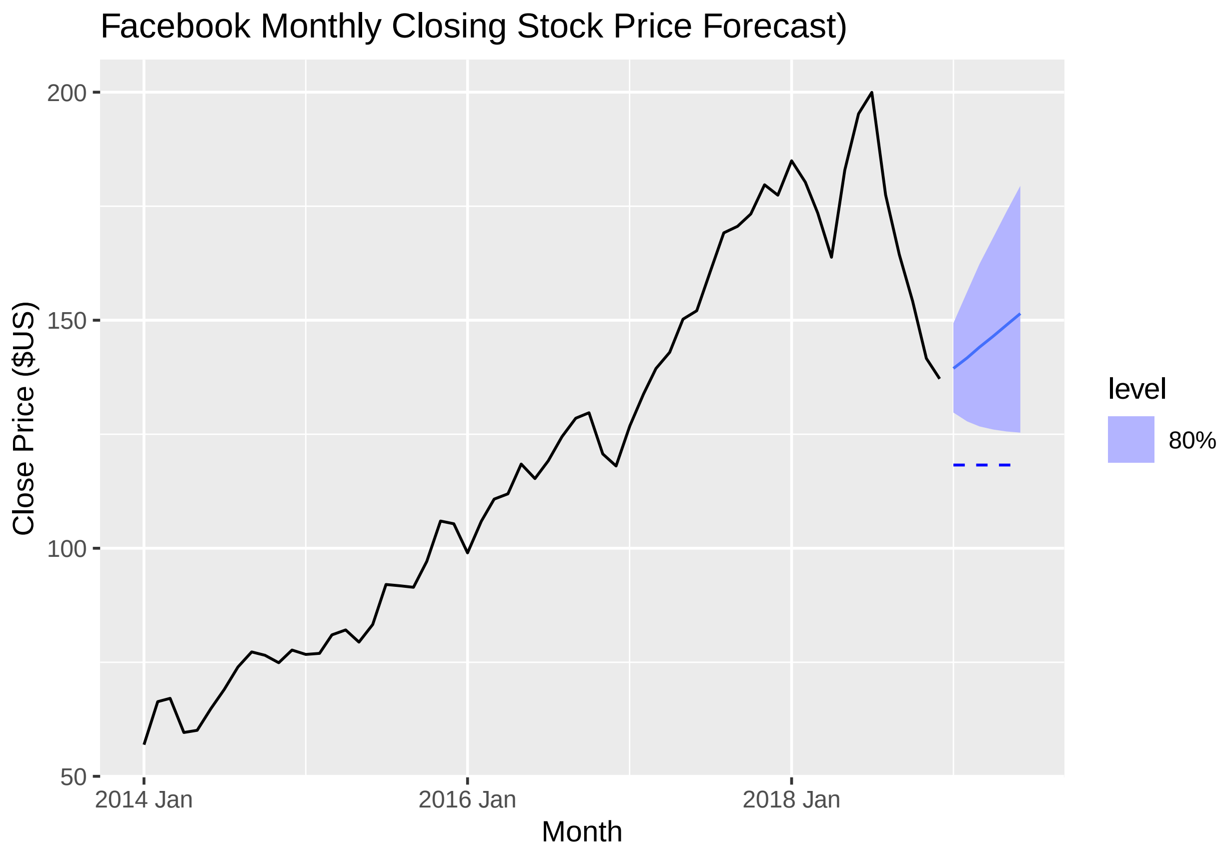
```
## # A tsibble: 6 x 8 [!]
## # Key:       Symbol [1]
##   Symbol Date        Open  High   Low Close Adj_Close    Volume
##   <chr>  <date>     <dbl> <dbl> <dbl> <dbl>     <dbl>     <dbl>
## 1 AAPL   2014-01-02  79.4  79.6  78.9  79.0      67.0  58671200
## 2 AAPL   2014-01-03  79.0  79.1  77.2  77.3      65.5  98116900
## 3 AAPL   2014-01-06  76.8  78.1  76.2  77.7      65.9 103152700
## 4 AAPL   2014-01-07  77.8  78.0  76.8  77.1      65.4  79302300
## 5 AAPL   2014-01-08  77.0  77.9  77.0  77.6      65.8  64632400
## 6 AAPL   2014-01-09  78.1  78.1  76.5  76.6      65.0  69787200
```

```r
fb_stock <- gafa_stock %>%
  filter(Symbol == "FB")

autoplot(fb_stock, Close) +
  labs(y = "$US",
       title = "Facebook Stock Prices")
```



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

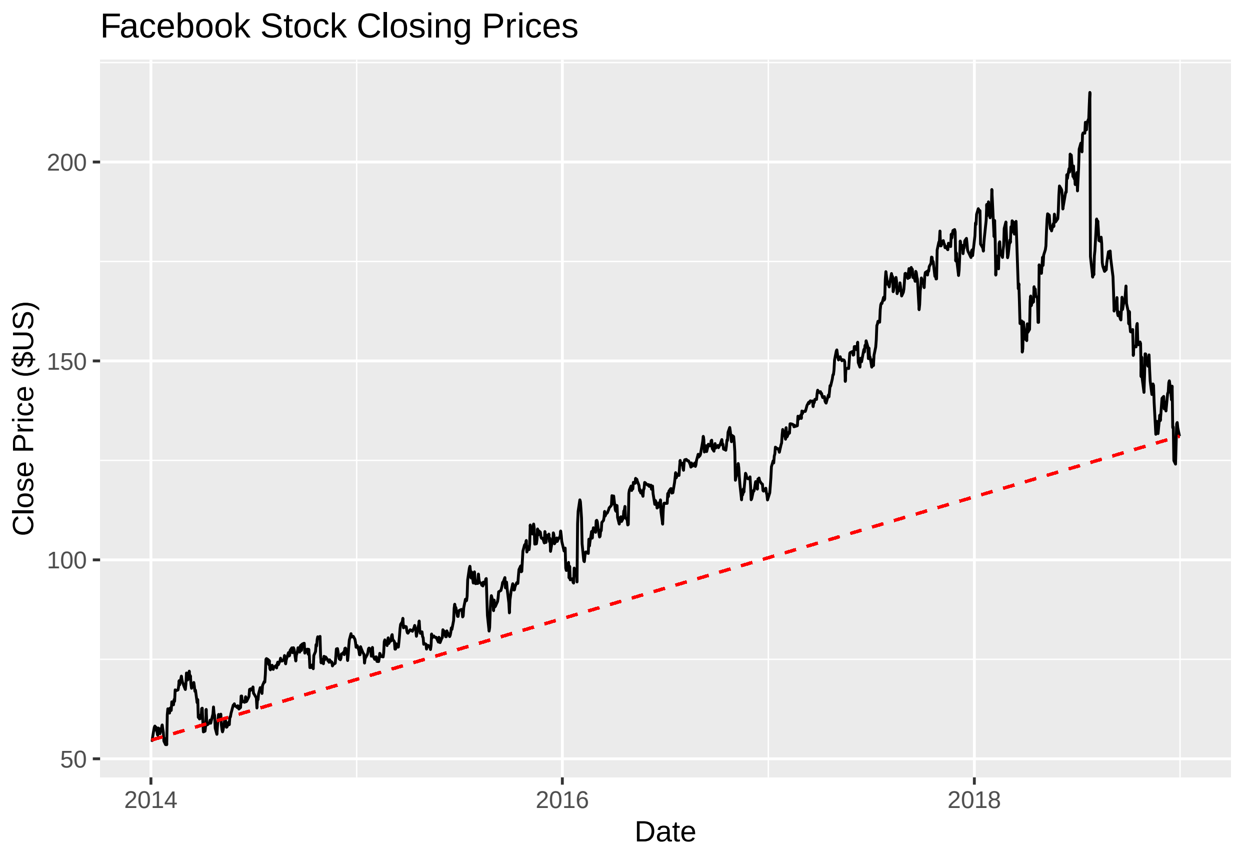
```r
fb_monthly <- gafa_stock %>%
  filter(Symbol == "FB", !is.na(Close)) %>%
  index_by(Month = yearmonth(Date)) %>%
  summarise(Close = mean(Close))

# Fit drift model
fb_fc <- fb_monthly %>%
  model(Drift = RW(log(Close) ~ drift())) %>%
  forecast(h = 6) %>%
  mutate(.median = median(fb_monthly$Close, na.rm = TRUE))

# Plot
fb_fc %>%
  autoplot(fb_monthly, level = 80) +
  geom_line(aes(y = .median), data = fb_fc, linetype = 2, color = "blue") +
  labs(
    title = "Facebook Monthly Closing Stock Price Forecast)",
    y = "Close Price ($US)"
  )
```



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

```
ggplot(fb_stock, aes(x = Date)) +
  geom_line(aes(y = Close)) +
  geom_segment(
    aes(x = min(Date), y = first(Close),
        xend = max(Date), yend = last(Close)),
    color = "red", linetype = "dashed"
  ) +
  labs(
    title = "Facebook Stock Closing Prices",
    y = "Close Price ($US)",
    x = "Date"
  )
```

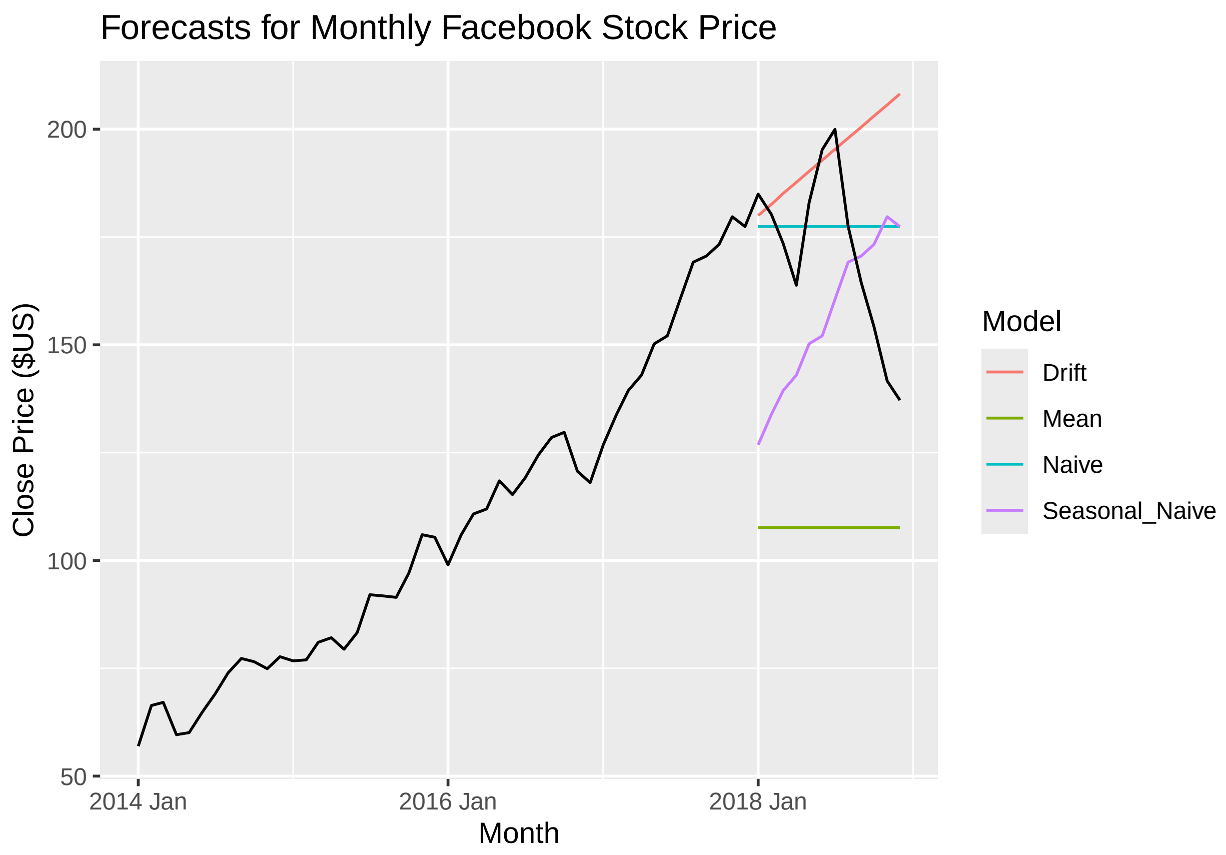
```
## Warning in geom_segment(aes(x = min(Date), y = first(Close), xend = max(Date), : All aesthetics have
## i Please consider using `annotate()` or provide this layer with data containing
##   a single row.
```



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

- Because Facebook stock price are follow a random patterns. I think NAIVE() model is the best. It has lowest RMSE and MAE, that means the forecast is closest to the true.

```r
fb_train <- fb_monthly %>%
  filter_index("2014 Jan" ~ "2017 Dec")

fb_fit <- fb_train %>%
  model(
    Mean = MEAN(Close),
    Naive = NAIVE(Close),
    Seasonal_Naive = SNAIVE(Close),
    Drift = RW(Close ~ drift())
  )

# Forecast 12 months ahead
fb_fc <- fb_fit %>% forecast(h = 12)

fb_fc %>%
  autoplot(fb_monthly, level = NULL) +
  labs(
    title = "Forecasts for Monthly Facebook Stock Price",
    y = "Close Price ($US)"
  ) +
  guides(colour = guide_legend(title = "Model"))
```



```
accuracy(fb_fit)
```

```
## # A tibble: 4 x 10
##   .model         .type           ME  RMSE   MAE     MPE  MAPE  MASE RMSSE  ACF1
##   <chr>          <chr>        <dbl> <dbl> <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Mean           Training  7.47e-16 34.8  29.4  -10.9   30.2  1.01  1.11  0.926
## 2 Naive          Training  2.56e+ 0  5.06  4.19   2.28   4.06 0.143 0.161 0.106
## 3 Seasonal_Naive Training  2.92e+ 1 31.4  29.2   23.8   23.8  1     1     0.730
## 4 Drift          Training -4.23e-15  4.37  3.57  -0.319  3.62 0.122 0.139 0.106
```

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. What do you conclude?

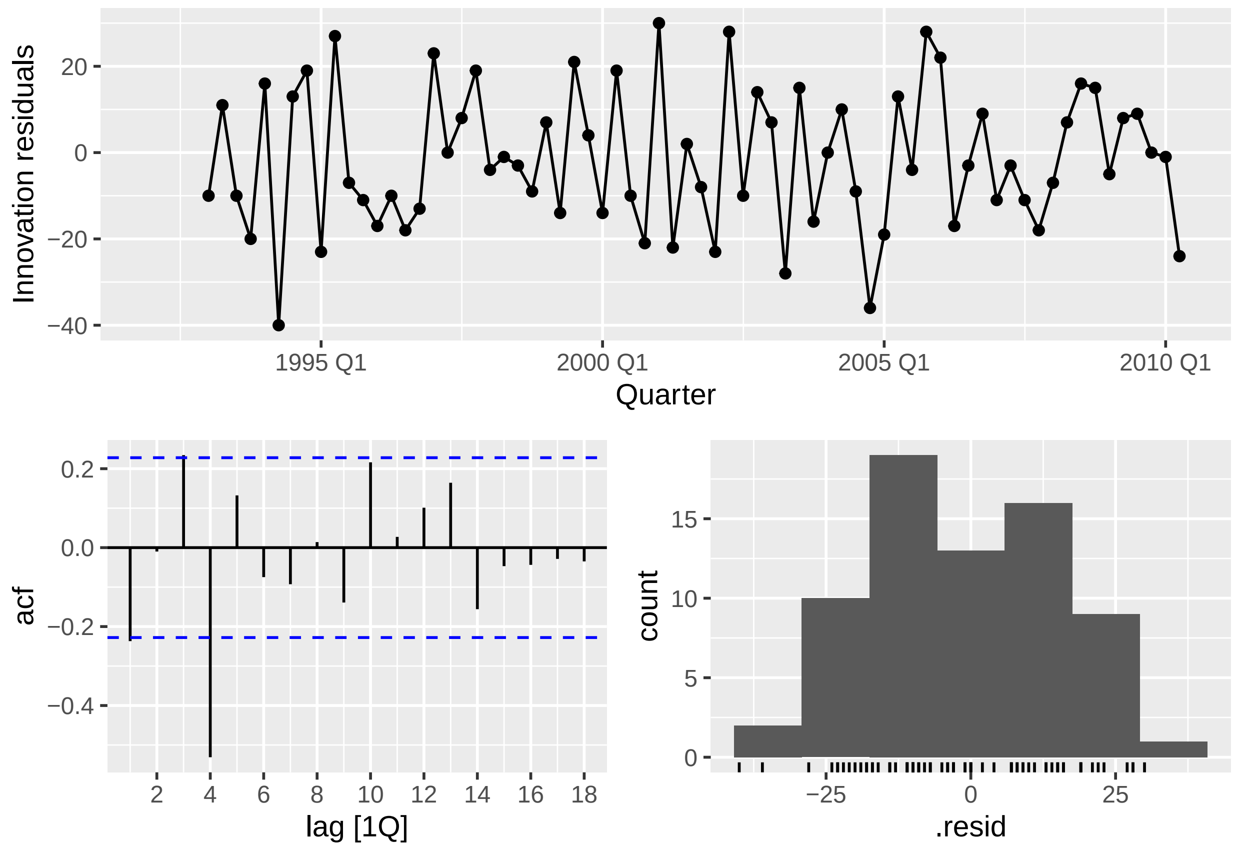
- In the time plot, there is no clear pattern, or trends.
- In ACF plot most autocorrelation bars are within the dashed blue line.
- The residuals is a normal distributed.

```r
# 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()
```

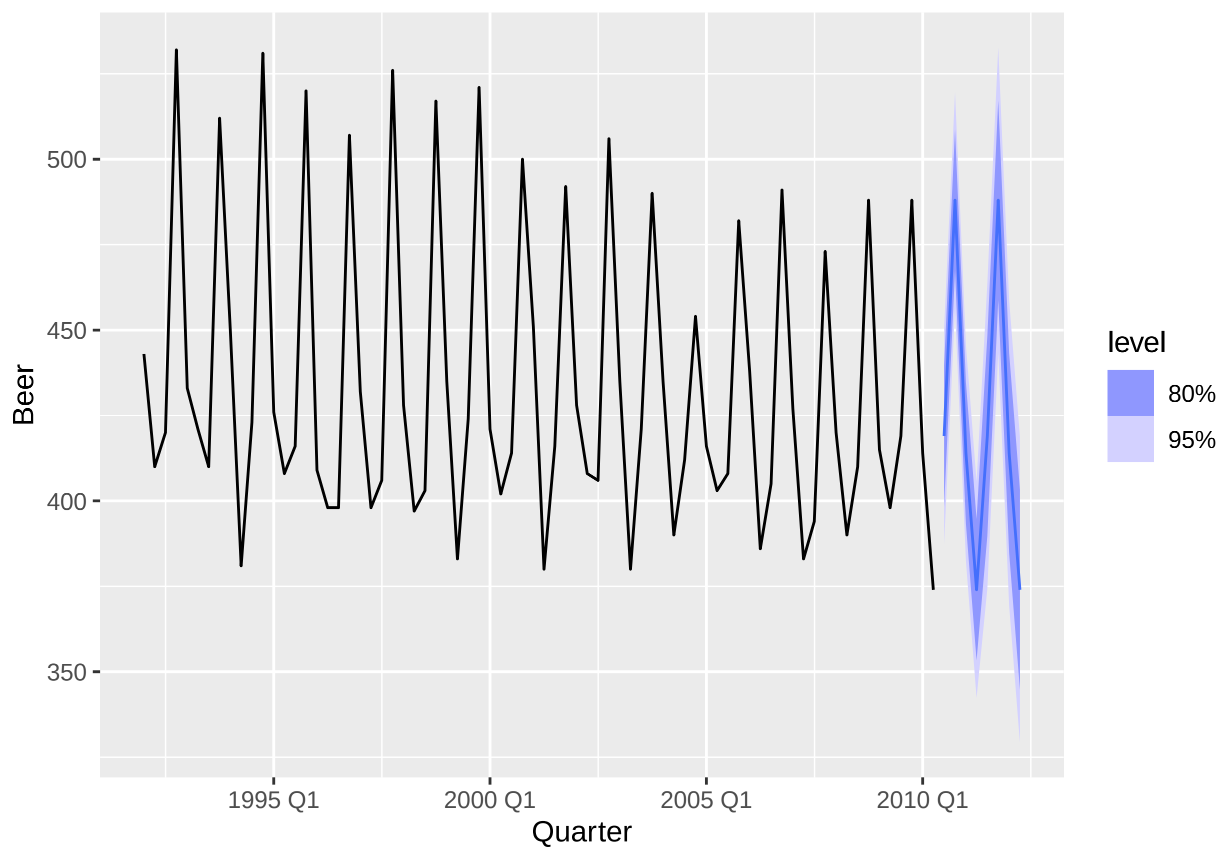
```
## Warning: Removed 4 rows containing missing values or values outside the scale range
## (‘geom_line()‘).
```

```
## 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()‘).
```



```r
# Look a some forecasts
fit |> forecast() |> autoplot(recent_production)
```



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.

- I don't think NAIVE() or SNAIVE() will work in this case, the residuals is no patterns, no trends, or any seasonal patterns.

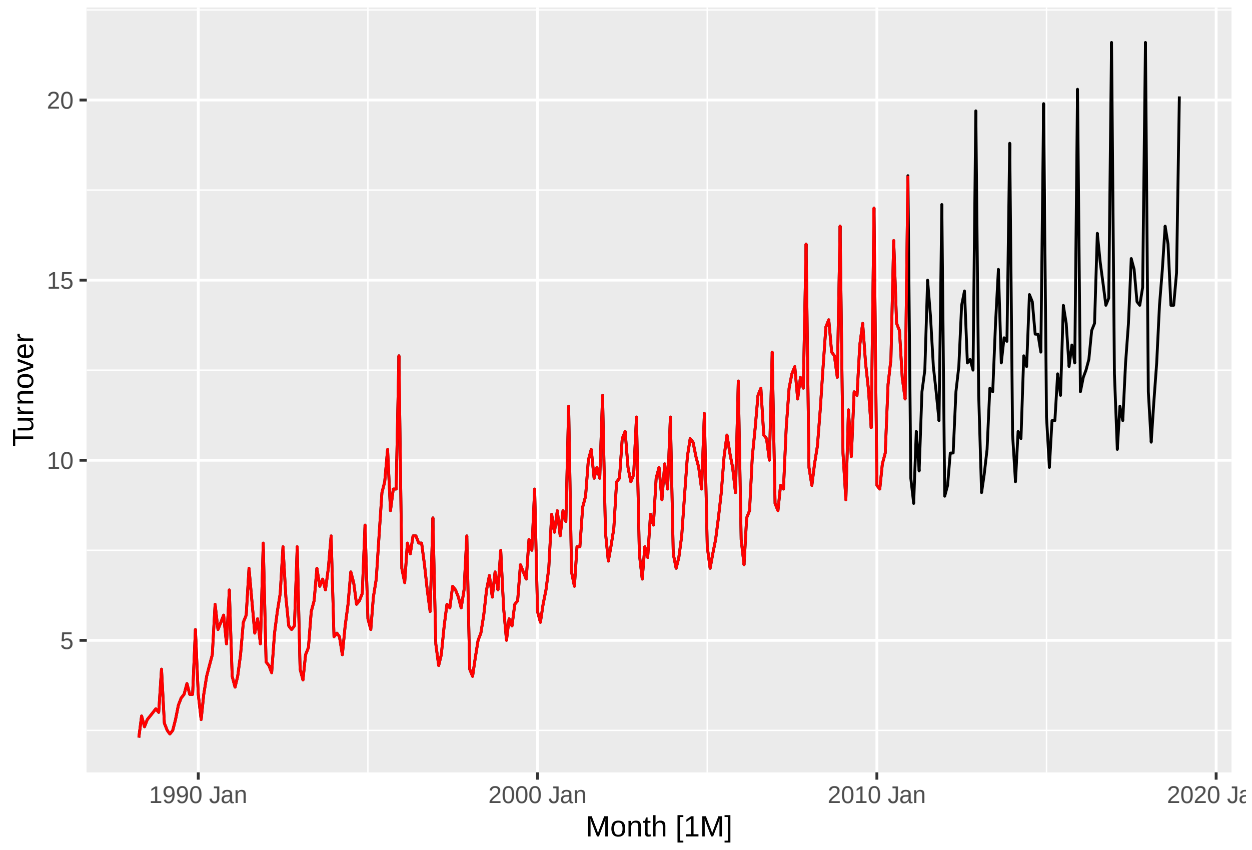
7.For your retail time series (from Exercise 7 in Section 2.10):

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

```r
set.seed(12345678)
myseries <- aus_retail |>
  filter(`Series ID` == sample(aus_retail$`Series ID`,1))

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

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

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



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

```
head(myseries_train)
```

```
## # A tsibble: 6 x 5 [1M]
## # Key:       State, Industry [1]
##   State              Industry                          `Series ID`    Month Turnover
##   <chr>              <chr>                             <chr>          <mth>    <dbl>
## 1 Northern Territory Clothing, footwear and perso~ A3349767W   1988 Apr      2.3
## 2 Northern Territory Clothing, footwear and perso~ A3349767W   1988 May      2.9
## 3 Northern Territory Clothing, footwear and perso~ A3349767W   1988 Jun      2.6
## 4 Northern Territory Clothing, footwear and perso~ A3349767W   1988 Jul      2.8
## 5 Northern Territory Clothing, footwear and perso~ A3349767W   1988 Aug      2.9
## 6 Northern Territory Clothing, footwear and perso~ A3349767W   1988 Sep      3
```

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

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

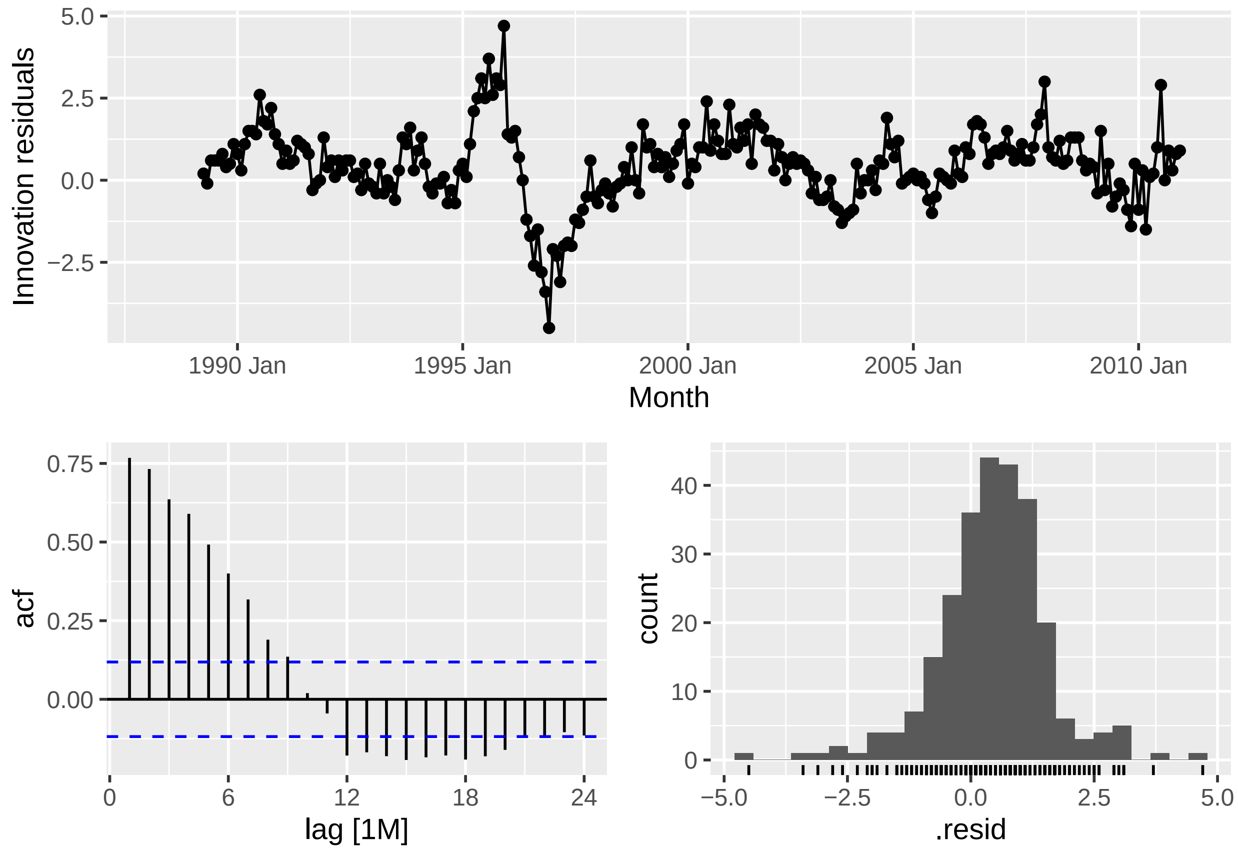
- I don't think residuals appear to be uncorrelated. Time plot shows some parterres. The histogram of residuals shows approximately normal distribution.

```r
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()‘).
```

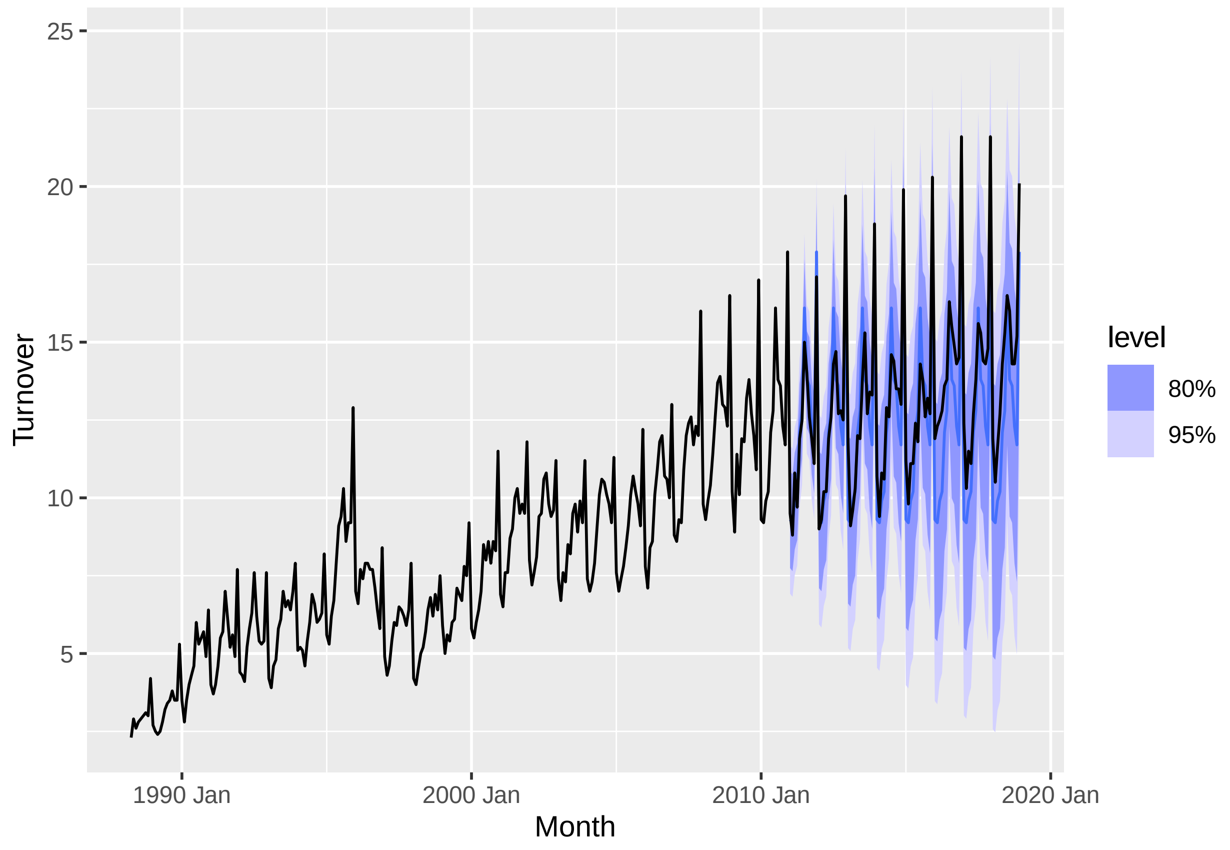


e. Produce forecasts for the test data

```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 Norther~ Clothin~ SNAIV~ Trai~ 0.439  1.21 0.915  5.23  12.4     1     1 0.768
```

```
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~ Nort~ Clothin~ Test  0.836  1.55  1.24  5.94  9.06  1.36  1.28 0.601
```

g.How sensitive are the accuracy measures to the amount of training data used? - The accuracy measures are highly sensitive to the amount of training data used and to the sample size. If the sample is too small or has less training data, the model fails to generalize well to new data. If the sample is too large, it is possible overfitting to the training data.