Data 624 - HW3 (Fall 2024)

Khyati Naik

```r
library(fpp3)
```

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

## v tibble      3.2.1     v tsibble     1.1.3
## v dplyr       1.1.4     v tsibbledata 0.4.1
## v tidyr       1.3.0     v feasts      0.3.2
## v lubridate   1.9.2     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()
```

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)

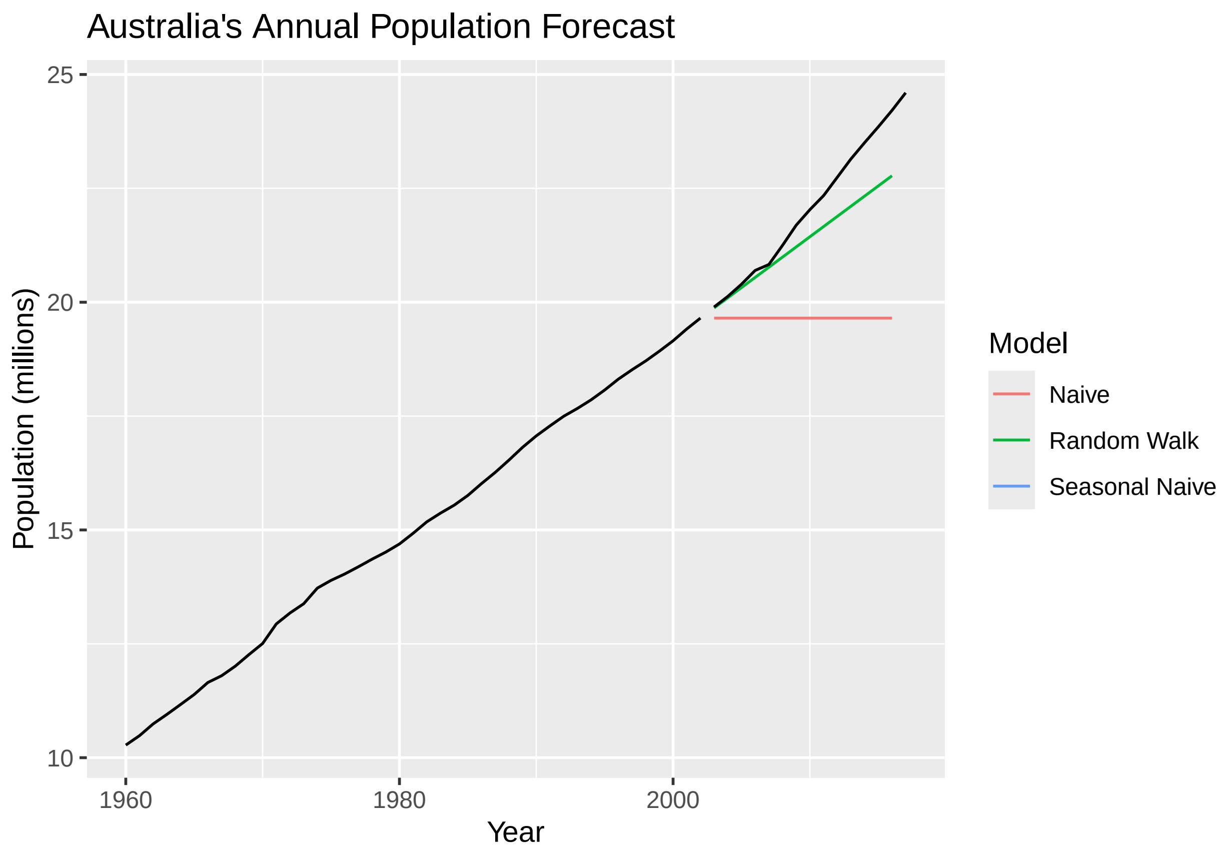
```r
# Data for Australia from 1960 to 2017 (total of 57 years)
aus_population <- global_economy %>%
  filter(Country == "Australia") %>%
  mutate(Population = Population / 1e6) %>%
  select(Country, Code, Year, Population)

# Use data from 1960 to 2002 for training (43 years)
train_data <- aus_population %>%
  filter_index("1960" ~ "2002")

# Train models on the training set
population_model <- train_data %>%
  model(
    Naive = NAIVE(Population),
    `Seasonal Naive` = SNAIVE(Population),
    `Random Walk` = RW(Population ~ drift())
  )
```

```
## Warning: 1 error encountered for Seasonal Naive
## [1] Non-seasonal model specification provided, use RW() or provide a different lag specification.
```

```
# Forecast the next 14 years (2003-2017)
population_forecast <- population_model %>%
  forecast(h = "14 years")

# Plot the forecasts alongside actual population values
population_forecast %>%
  autoplot(train_data, level = NULL) +
  autolayer(
    filter_index(aus_population, "2003" ~ "2017"),
    colour = "black"
  ) +
  labs(
    y = "Population (millions)",
    title = "Australia's Annual Population Forecast"
  ) +
  guides(colour = guide_legend(title = "Model"))
```

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

```
## Warning: Removed 14 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

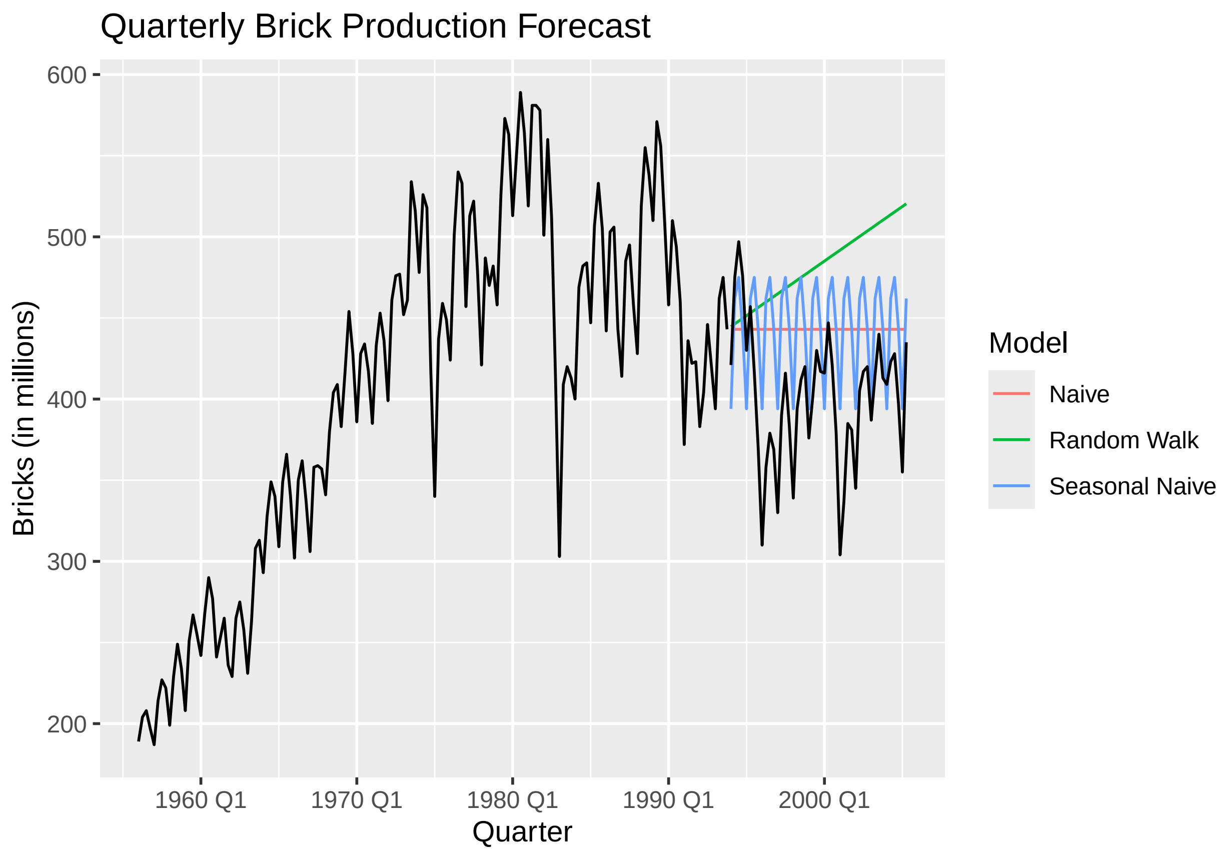


The Random Walk with drift() seems to be the best fit since the plot indicates a clear upward trend. The Naive model isn't suitable because it doesn't account for trends, and the Seasonal Naive model is also inappropriate as there doesn't seem to be any seasonality in the data.

- Bricks (aus_production)

```
# Data from 1956 Q1 to 2005 Q2 (198 total quarters)
aus_brick_data <- aus_production %>%
  select(Quarter, Bricks) %>%
  na.omit()

# Define the training period from 1956 Q1 to 1993 Q4
training_data <- aus_brick_data %>%
  filter_index("1956 Q1" ~ "1993 Q4")

# Train models on the brick production data
brick_models <- training_data %>%
  model(
    `Naive` = NAIVE(Bricks),
    `Seasonal Naive` = SNAIVE(Bricks),
    `Random Walk` = RW(Bricks ~ drift())
  )

# Forecast for the next 46 quarters
brick_forecasts <- brick_models %>%
  forecast(h = 46)

# Visualize the forecasts against actual values
brick_forecasts %>%
  autoplot(training_data, level = NULL) +
  autolayer(
    filter_index(aus_brick_data, "1994 Q1" ~ .),
    colour = "black"
  ) +
  labs(
    y = "Bricks (in millions)",
    title = "Quarterly Brick Production Forecast"
  ) +
  guides(colour = guide_legend(title = "Model"))
```

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



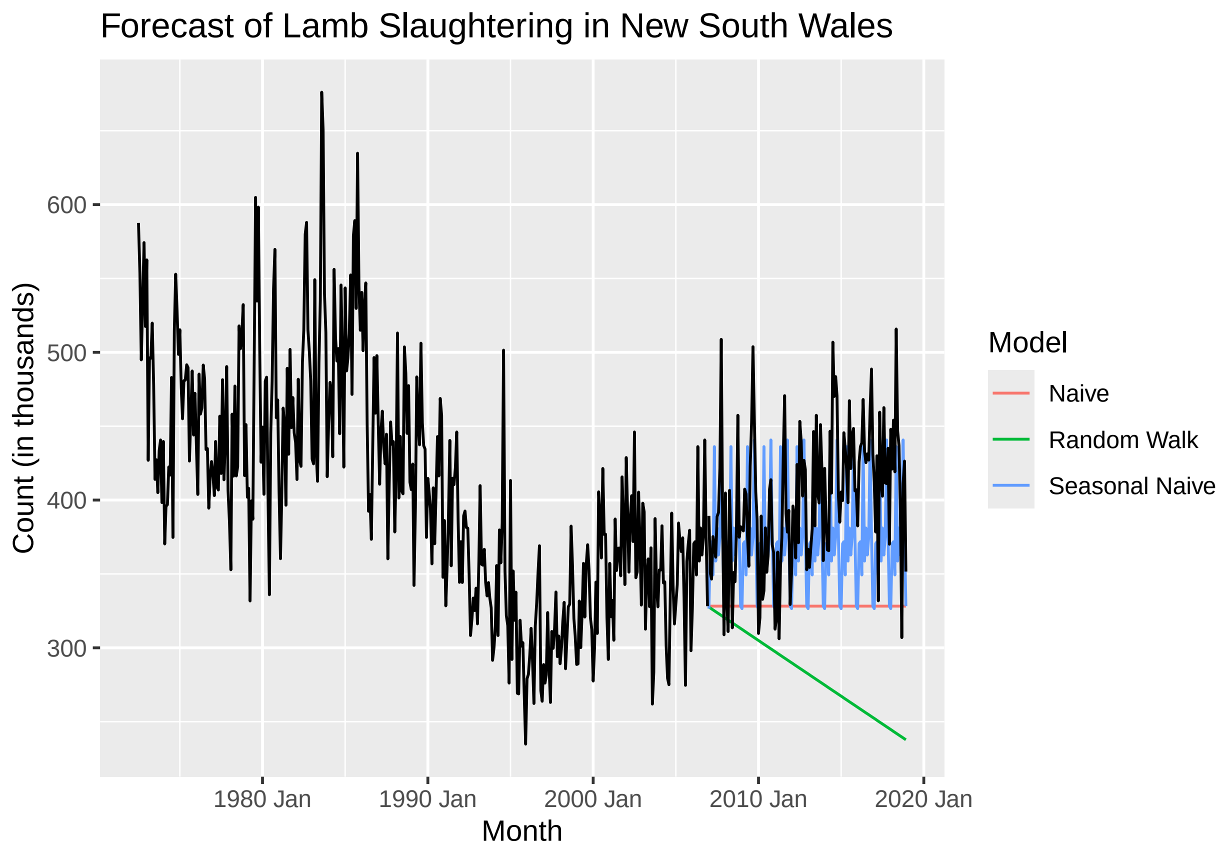
Initially, none of the three models—Naive, Seasonal Naive, or Random Walk with drift—seemed like a perfect fit. However, given the strong seasonal pattern in the data, I would choose the Seasonal Naive model. I did test the Seasonal Naive with drift, but its forecasts showed an upward trend that didn't reflect the recent downward movement in the data. So, despite its limitations, the Seasonal Naive model is preferable as it at least captures the seasonal variations.

- NSW Lambs (aus_livestock)

```r
# Data spanning from July 1972 to December 2018 (558 months)
nsw_lamb_data <- aus_livestock %>%
  filter(State == 'New South Wales' & Animal == 'Lambs') %>%
  mutate(Count = Count / 1000) %>%
  select(Month, Count)

# Define the training period from July 1972 to December 2006
training_set <- nsw_lamb_data %>%
  filter_index("1972 JUL" ~ "2006 DEC")

# Train the models on the lamb slaughter data
lamb_models <- training_set %>%
  model(
    `Naive` = NAIVE(Count),
    `Seasonal Naive` = SNAIVE(Count),
    `Random Walk` = RW(Count ~ drift())
  )
```

```r
# Forecast for the next 144 months
lamb_forecasts <- lamb_models %>%
  forecast(h = 144)

# Visualize the forecasts compared to actual data
lamb_forecasts %>%
  autoplot(training_set, level = NULL) +
  autolayer(
    filter_index(nsw_lamb_data, "2007 JAN" ~ .),
    colour = "black"
  ) +
  labs(
    y = "Count (in thousands)",
    title = "Forecast of Lamb Slaughtering in New South Wales"
  ) +
  guides(colour = guide_legend(title = "Model"))
```

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



I opted for the Seasonal Naive model for forecasting NSW lambs because the data clearly shows seasonal patterns, which align well with the Seasonal Naive forecasting approach. This choice is supported by the observations made in the Brick plot, which highlight the data's seasonal characteristics.

- Household wealth (hh_budget).

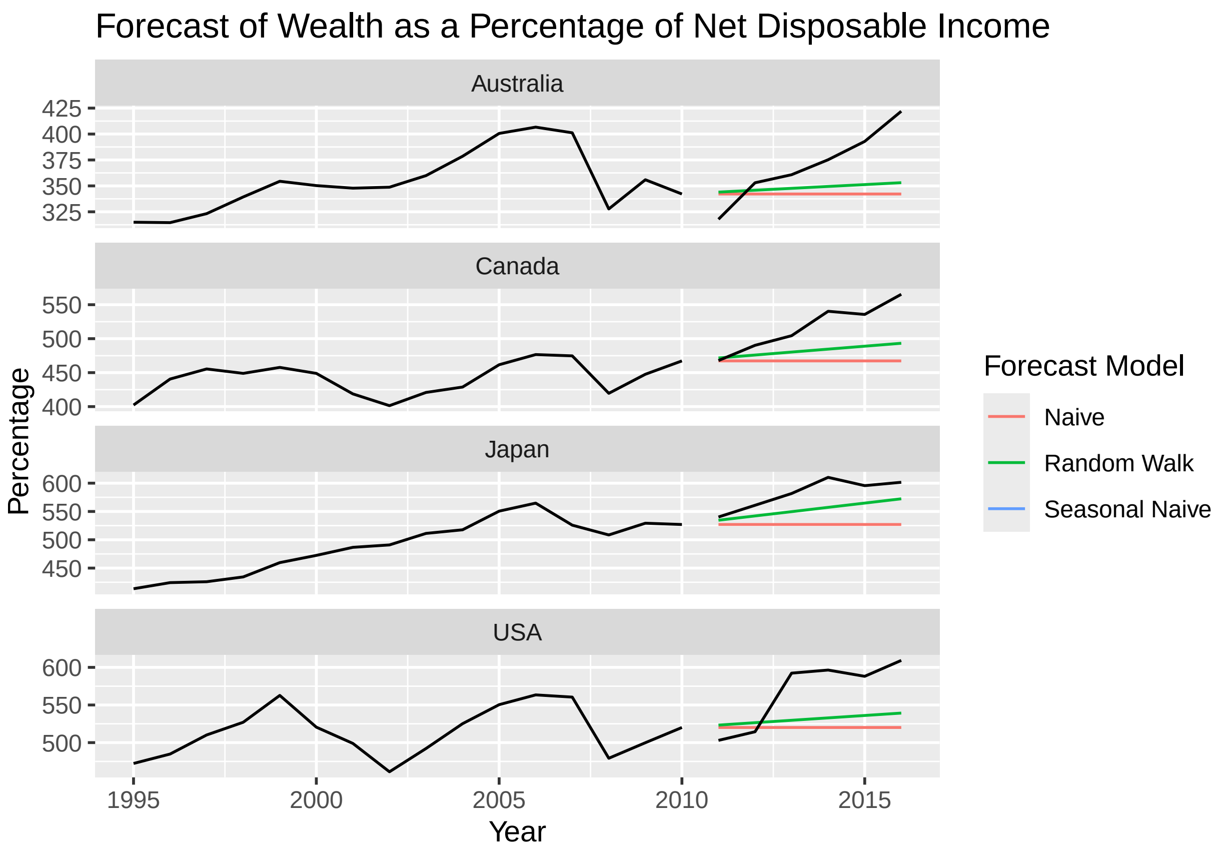
```r
# Wealth represented as a percentage of net disposable income
household_wealth <- hh_budget %>%
  select(Country, Year, Wealth)

# Data range from 1995 to 2016 (total of 22 years)

# Define the training set from 1995 to 2010
training_data <- household_wealth %>%
  filter_index("1995" ~ "2010")

# Fit different forecasting models to the training data
wealth_models <- training_data %>%
  model(
    `Naive` = NAIVE(Wealth),
    `Seasonal Naive` = SNAIVE(Wealth),
    `Random Walk` = RW(Wealth ~ drift())
  )
```

```
## Warning: 4 errors (1 unique) encountered for Seasonal Naive
## [4] Non-seasonal model specification provided, use RW() or provide a different lag specification.
```

```r
# Forecast for the next 6 years
wealth_forecasts <- wealth_models %>%
  forecast(h = 6)

# Visualize the forecasts alongside the actual data
wealth_forecasts %>%
  autoplot(training_data, level = NULL) +
  autolayer(
    filter_index(household_wealth, "2011" ~ .),
    colour = "black"
  ) +
  labs(
    y = "Percentage",
    title = "Forecast of Wealth as a Percentage of Net Disposable Income"
  ) +
  guides(colour = guide_legend(title = "Forecast Model"))
```

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

```
## Warning: Removed 24 rows containing missing values or values outside the scale range
## (`geom_line()`).
```



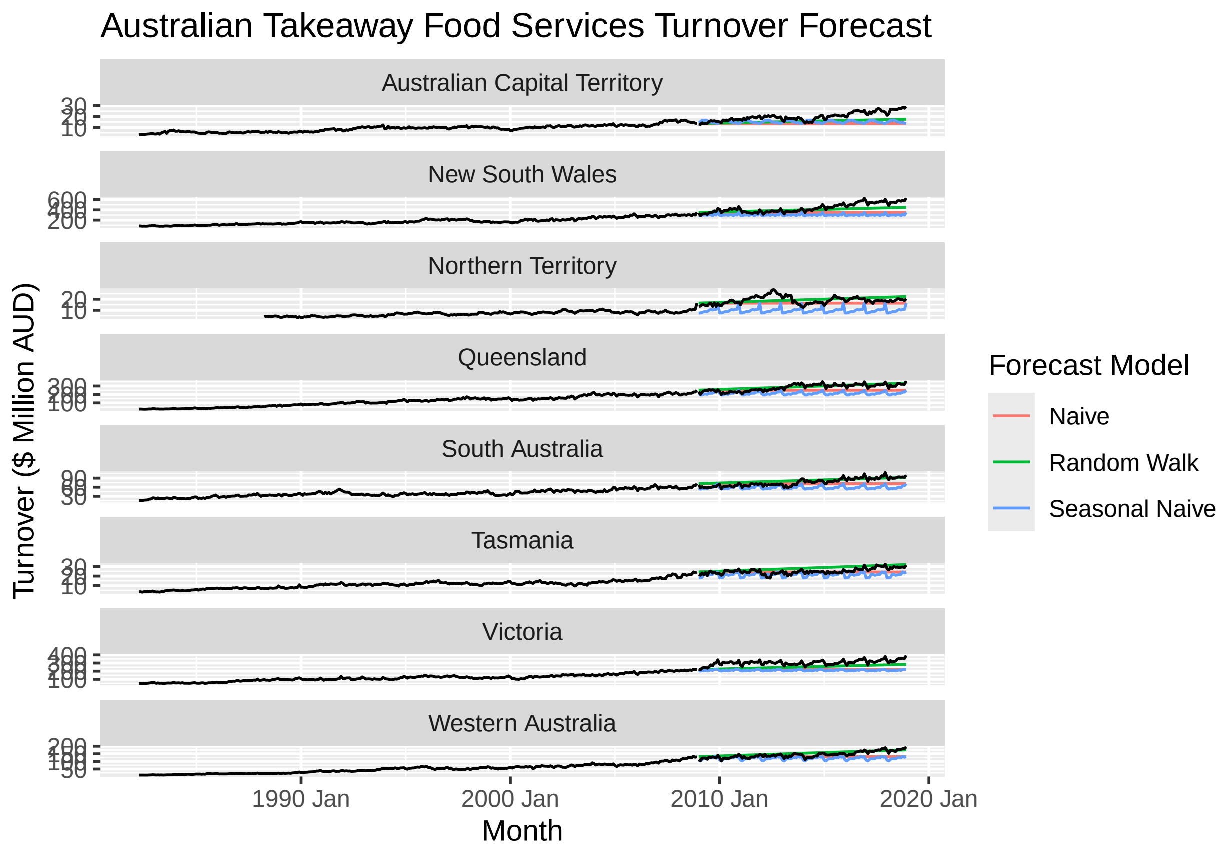
The plot of household wealth displays forecasts for four countries: Australia, Canada, Japan, and the United States. Although I thought about combining these countries into a single forecast, the small number of countries (only four) made it practical to forecast each one separately. Given that all four countries exhibit a rising trend in household wealth, the Random Walk with drift() method seems to be the most suitable forecasting approach.

- Australian takeaway food turnover (aus_retail).

```
# Data for takeaway food services across 8 states
# From April 1982 to December 2018 (total of 36 years)
takeaway_data <- aus_retail %>%
  filter(Industry == "Takeaway food services") %>%
  select(State, Month, Turnover)

# Define the training period from April 1982 to December 2008
training_set <- takeaway_data %>%
  filter_index("1982 Apr" ~ "2008 Dec")

# Fit various forecasting models to the training data
forecast_models <- training_set %>%
  model(
    `Naive` = NAIVE(Turnover),
    `Seasonal Naive` = SNAIVE(Turnover),
    `Random Walk` = RW(Turnover ~ drift())
  )
```

```r
# Generate forecasts for the next 120 months
forecast_results <- forecast_models %>%
  forecast(h = 120)

# Plot the forecast results against the actual data
forecast_results %>%
  autoplot(training_set, level = NULL) +
  autolayer(
    filter_index(takeaway_data, "2009 Jan" ~ .),
    colour = "black"
  ) +
  labs(
    y = "Turnover ($ Million AUD)",
    title = "Australian Takeaway Food Services Turnover Forecast"
  ) +
  guides(colour = guide_legend(title = "Forecast Model"))
```

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

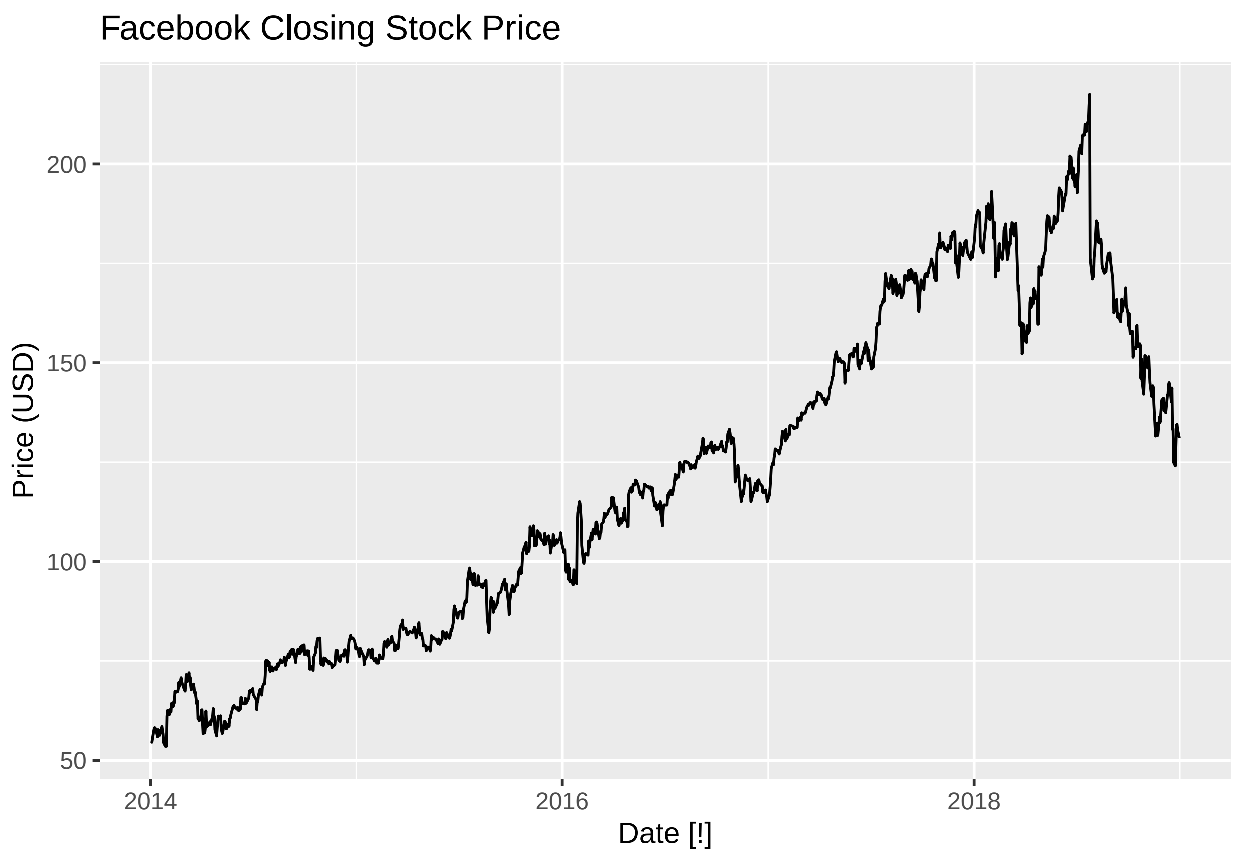


When analyzing each of the eight Australian states separately, it becomes evident that while the Seasonal Naive model effectively captures the seasonal patterns in retail turnover, the Random Walk with drift more accurately reflects the overall increasing trend. Given that the upward trend seems to be a more significant factor in the data, I think the Random Walk with drift is the more suitable forecasting method for this dataset.

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

a. Produce a time plot of the series.

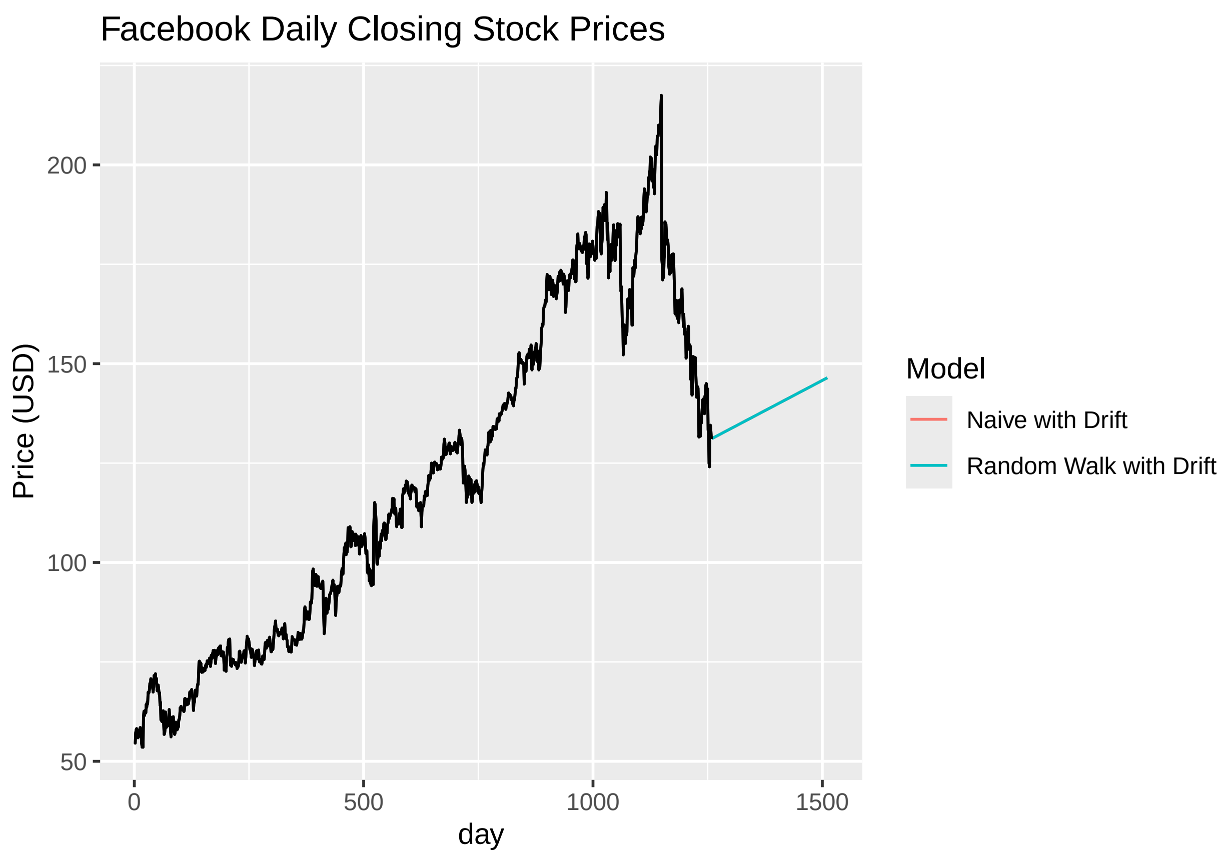
```r
# Last entry in the dataset is December 31, 2018
facebook_stock <- gafa_stock %>%
  filter(Symbol == 'FB')

# Plot the closing stock price of Facebook
facebook_stock %>% autoplot(Close) +
  labs(
    y = "Price (USD)",
    title = "Facebook Closing Stock Price"
  )
```



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

```r
# Re-index the data based on trading days
facebook_stock <- gafa_stock %>%
  filter(Symbol == "FB") %>%
  mutate(day = row_number()) %>%
  update_tsibble(index = day, regular = TRUE)
```

```r
# Fit different forecasting models
model_fit <- facebook_stock %>%
  model(
    `Naive with Drift` = NAIVE(Close ~ drift()),
    `Random Walk with Drift` = RW(Close ~ drift())
  )

# Forecast the next 253 days (approximately one year)
forecast_results <- model_fit %>%
  forecast(h = 253)

# Plot the forecasted values alongside the actual data
forecast_results %>%
  autoplot(facebook_stock, level = NULL) +
  autolayer(facebook_stock, Close, colour = "black") +
  labs(
    y = "Price (USD)",
    title = "Facebook Daily Closing Stock Prices"
  ) +
  guides(colour = guide_legend(title = "Model"))
```



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

```r
# Enhanced plot with a dashed line showing the trend
forecast_results %>%
  autoplot(facebook_stock, level = NULL) +
  autolayer(facebook_stock, Close, colour = "black") +
  labs(
    y = "Price (USD)",
    title = "Facebook Daily Closing Stock Prices"
  ) +
  guides(colour = guide_legend(title = "Model")) +
  geom_segment(aes(x = first(facebook_stock$day), y = first(facebook_stock$Close),
                   xend = last(facebook_stock$day), yend = last(facebook_stock$Close)),
               linetype = 'dashed')
```

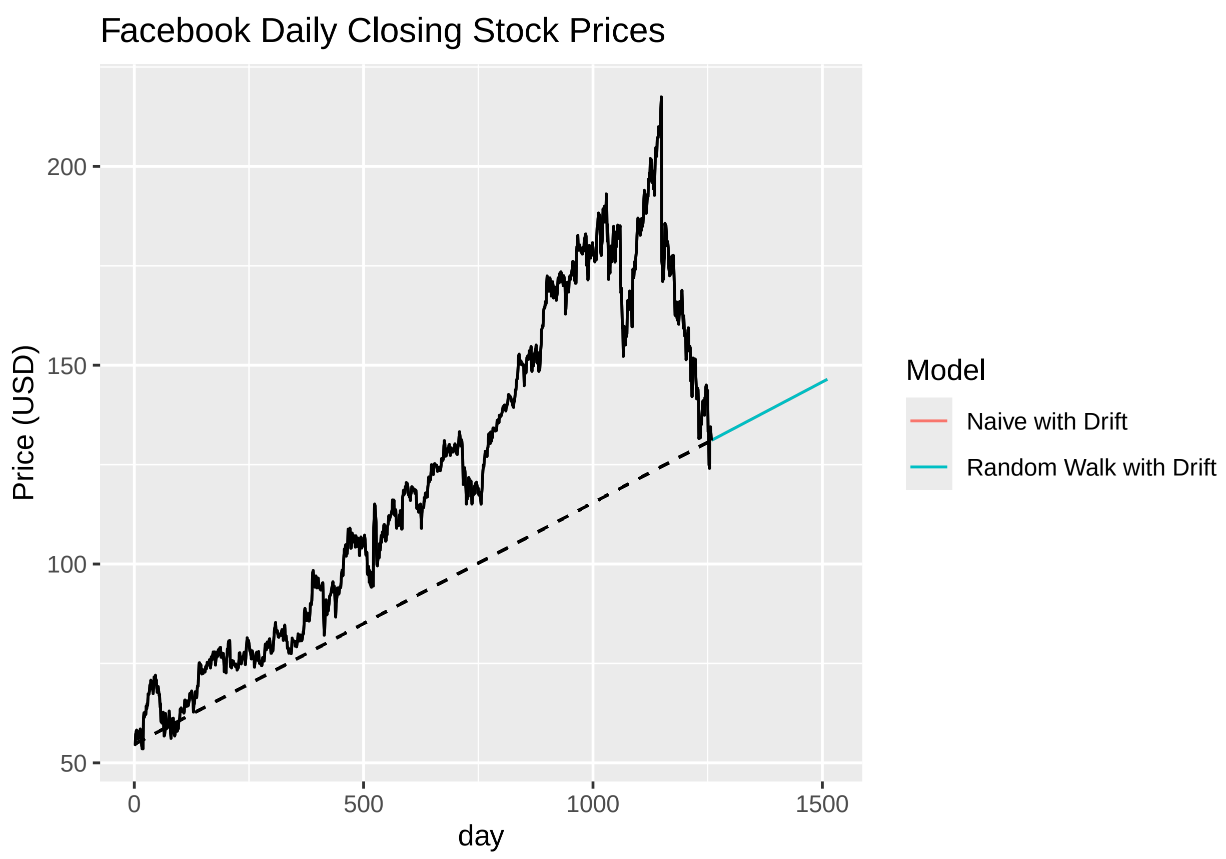
```
## Warning: Use of `facebook_stock$day` is discouraged.
## i Use `day` instead.
```

```
## Warning: Use of `facebook_stock$Close` is discouraged.
## i Use `Close` instead.
```

```
## Warning: Use of `facebook_stock$day` is discouraged.
## i Use `day` instead.
```

```
## Warning: Use of `facebook_stock$Close` is discouraged.
## i Use `Close` instead.
```

```
## Warning in geom_segment(aes(x = first(facebook_stock$day), y = first(facebook_stock$Close), : All ae
## i Please consider using `annotate()` or provide this layer with data containing
##   a single row.
```



Drawing a straight, dashed line from the initial to the final data point shows that this line intersects with the forecasted areas of both the Naive Drift and Random Walk with Drift models.

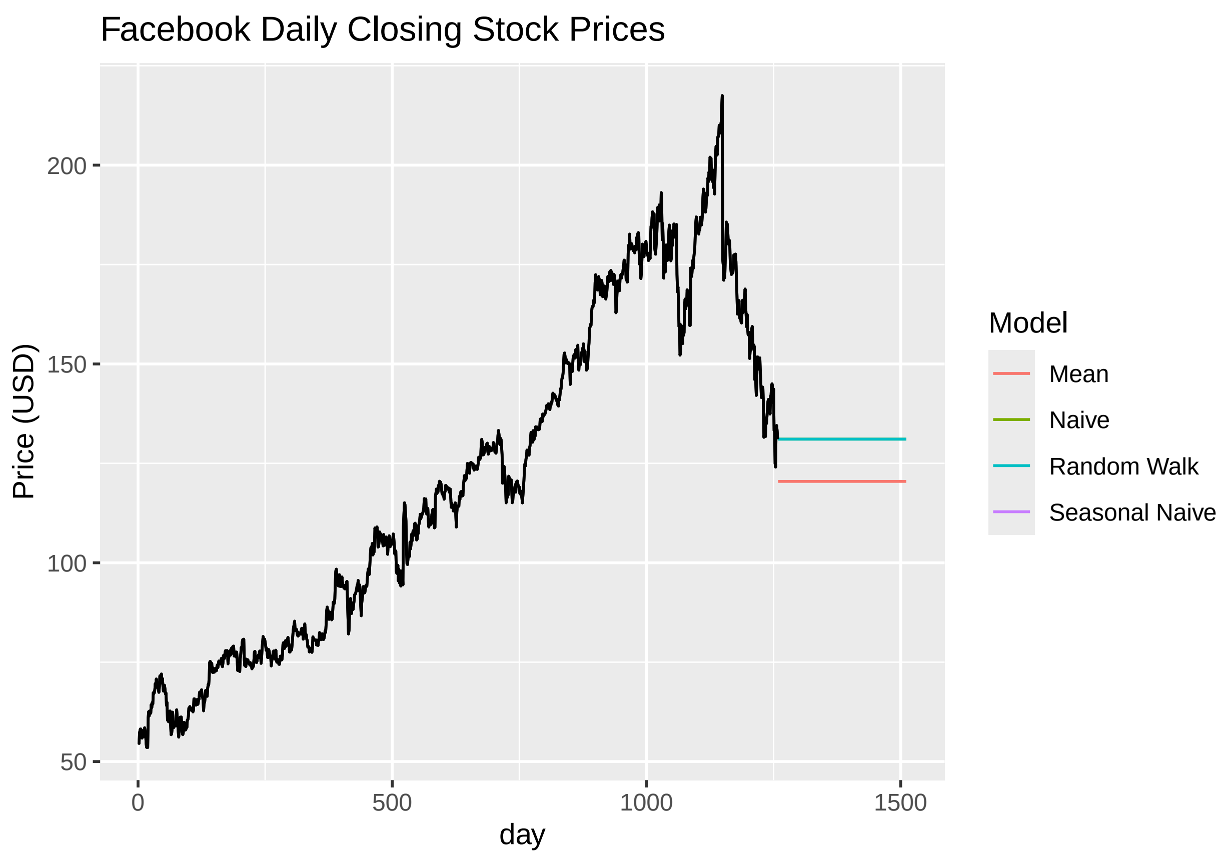
d. Try using some of the other benchmark functions to forecast the same data set. Which do you think is best? Why?

```r
# Fit additional models
model_fit_alternative <- facebook_stock %>%
  model(
    Mean = MEAN(Close),
    Naive = NAIVE(Close),
    `Seasonal Naive` = SNAIVE(Close),
    `Random Walk` = RW(Close)
  )
```

```
## Warning: 1 error encountered for Seasonal Naive
## [1] Non-seasonal model specification provided, use RW() or provide a different lag specification.
```

```r
# Generate forecasts using the alternative models
forecast_results_alternative <- model_fit_alternative %>%
  forecast(h = 253)

# Plot forecasts from the alternative models
forecast_results_alternative %>%
  autoplot(facebook_stock, level = NULL) +
  autolayer(facebook_stock, Close, colour = "black") +
  labs(
    y = "Price (USD)",
    title = "Facebook Daily Closing Stock Prices"
  ) +
  guides(colour = guide_legend(title = "Model"))
```

```
## Warning: Removed 253 rows containing missing values or values outside the scale range
## (`geom_line()`).
```



In exploring alternative forecasting methods, I tested MEAN, NAIVE, SNAIVE, and RW without drift. From the initial assessment of the plots, it's clear that none of these methods effectively capture the observed trend. Although Facebook's stock has been on a long-term uptrend, it seems to be shifting into a downtrend now, with no significant seasonal pattern present. Although the idea of "regression to the mean" suggests that MEAN might be a reasonable choice, there isn't enough historical data for Facebook to fully support this approach. Given these observations, I would recommend using the NAIVE method. Based on the trends visible in the plots, a continued decrease in stock price seems likely, but none of the tested methods reflect this trend accurately. Therefore, considering both the historical uptrend and the recent downtrend, forecasting based on the most recent observations appears to be the most appropriate strategy.

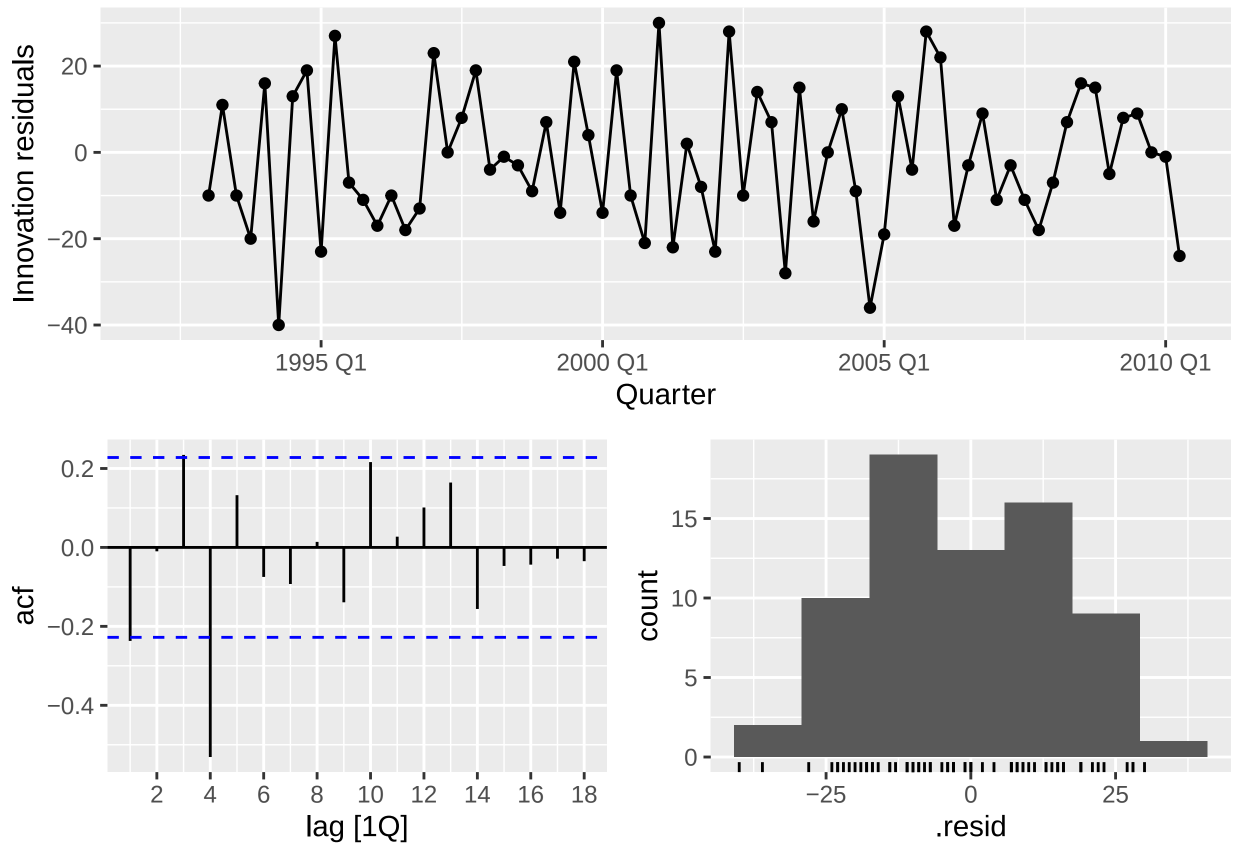
5.3 Apply a seasonal naïve method to the quarterly Australian beer production data from 1992. Check if the residuals look like white noise, and plot the forecasts. The following code will help. What do you conclude?

```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()
```

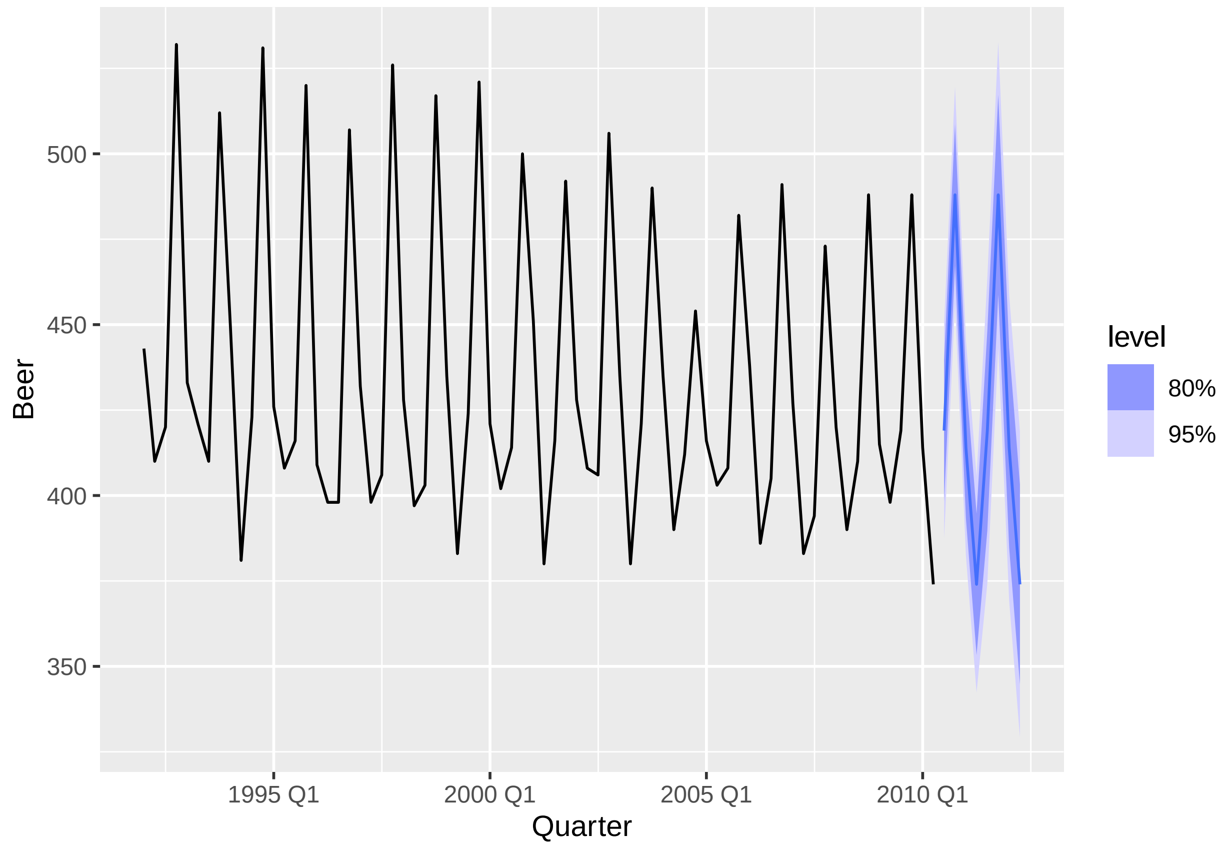
```
## Warning: Removed 4 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

```
## 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)
```



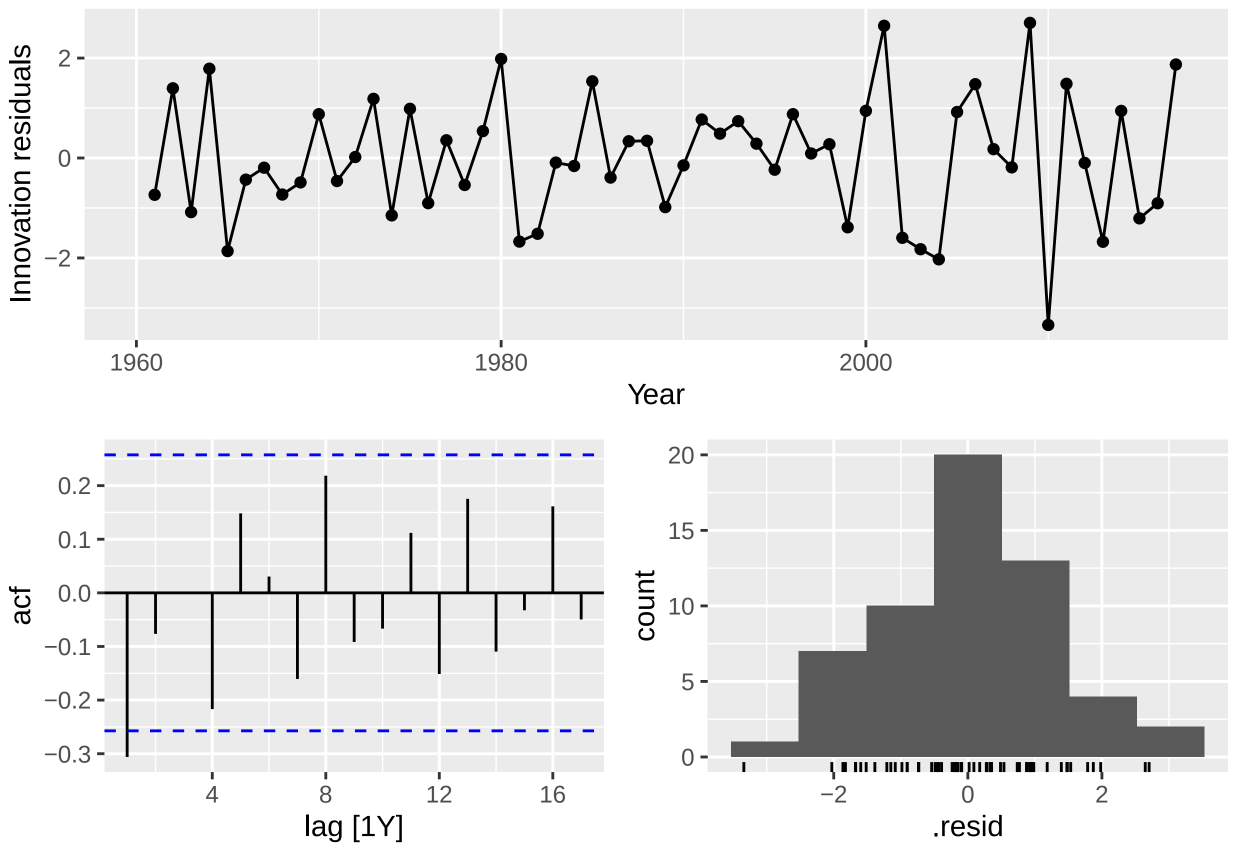
The residuals seem to resemble white noise. The plot of innovation residuals shows a mean close to zero with consistent variance around that mean. The autocorrelation function (ACF) plot indicates a significant correlation at the fourth lag, but no other lags show meaningful correlations. Although the residuals' histogram is not perfectly normal—featuring a central dip—the overall distribution approximates normality and lacks outliers. Based on this analysis, it appears that the seasonal naive method effectively incorporates nearly all available information. While there may be potential to refine the seasonal forecast further, the current seasonal naive forecast remains valid.

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.

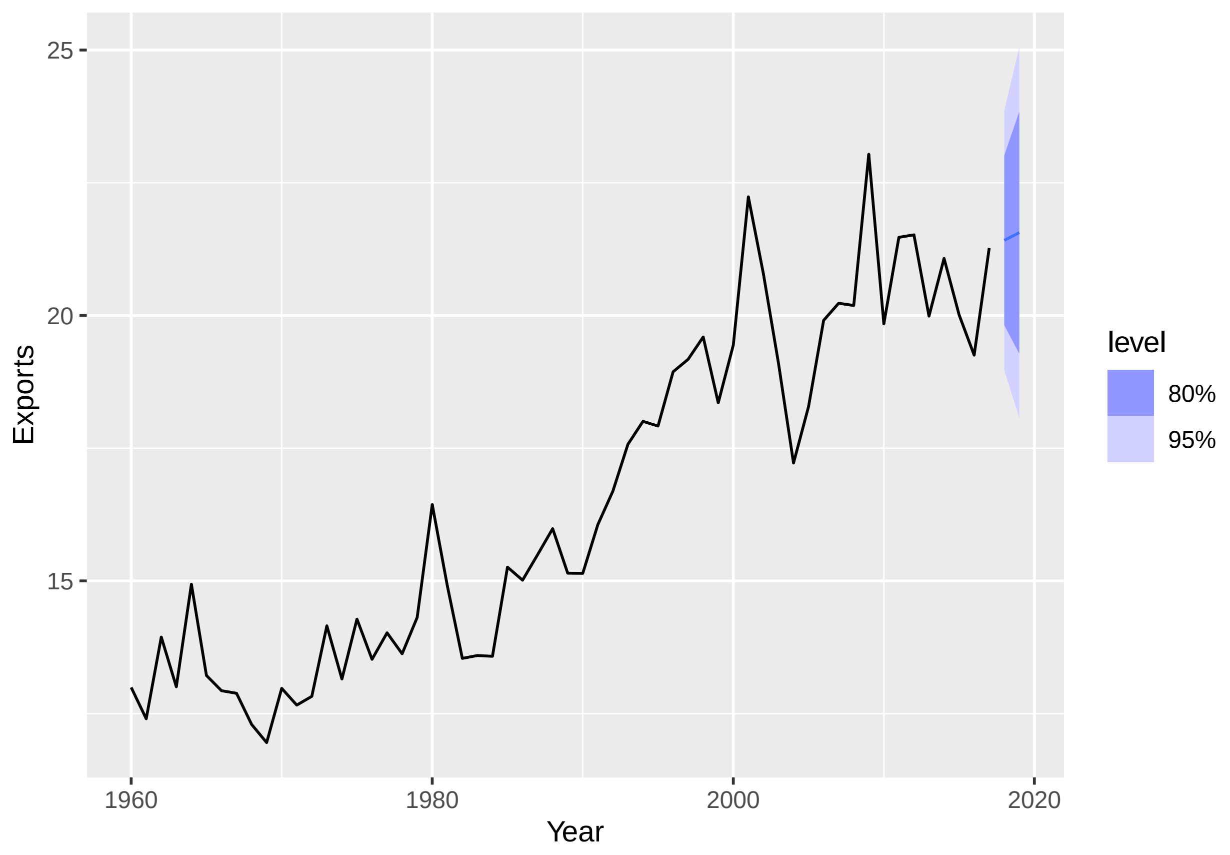
```r
# Filter the dataset for Australia and focus on export data
# Data spans from 1960 to 2017, covering 57 years
australia_exports <- global_economy %>%
  filter(Country == 'Australia')

# Specify and fit the forecasting model
model_fit <- australia_exports %>%
  model(NAIVE(Exports ~ drift()))
```

```r
# Examine the residuals of the fitted model
model_fit %>% 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()`).
```



```r
# Generate and visualize forecasts
model_fit %>% forecast() %>% autoplot(australia_exports)
```

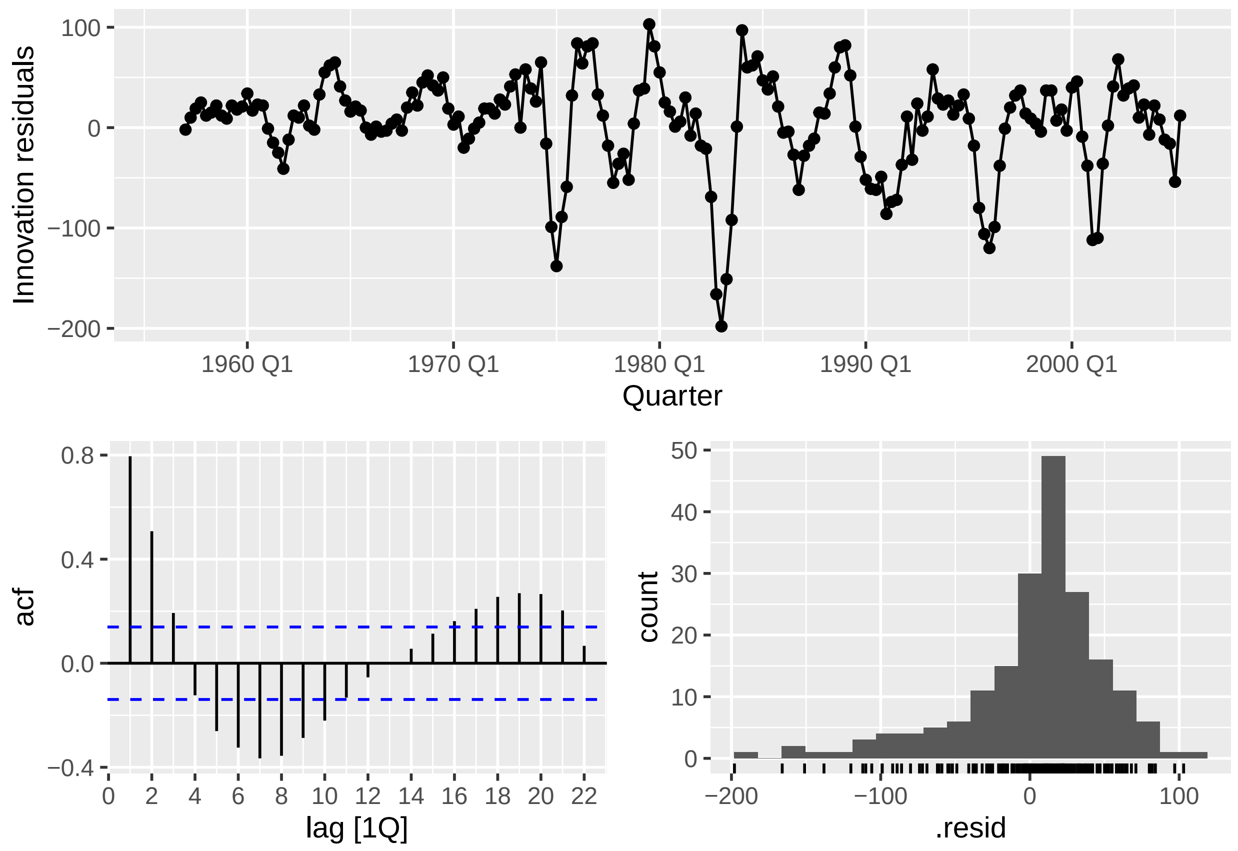


The analysis of the innovation residuals plot reveals that the residuals have a mean close to zero and exhibit near-constant variance, indicating they resemble white noise. The autocorrelation function (ACF) confirms that there are no significant correlations among the residuals, suggesting they are uncorrelated. Additionally, the histogram of residuals approximates a normal distribution, supporting the assumption of normally distributed residuals. Given the absence of a clear seasonal pattern in the data, the NAIVE model was chosen over the Seasonal Naive (SNAIVE) model. Despite its simplicity, the NAIVE model effectively captures the essential characteristics of the data. Incorporating drift() into the NAIVE model accounts for the observed upward trend and provides a slightly better forecast compared to a NAIVE model without drift().

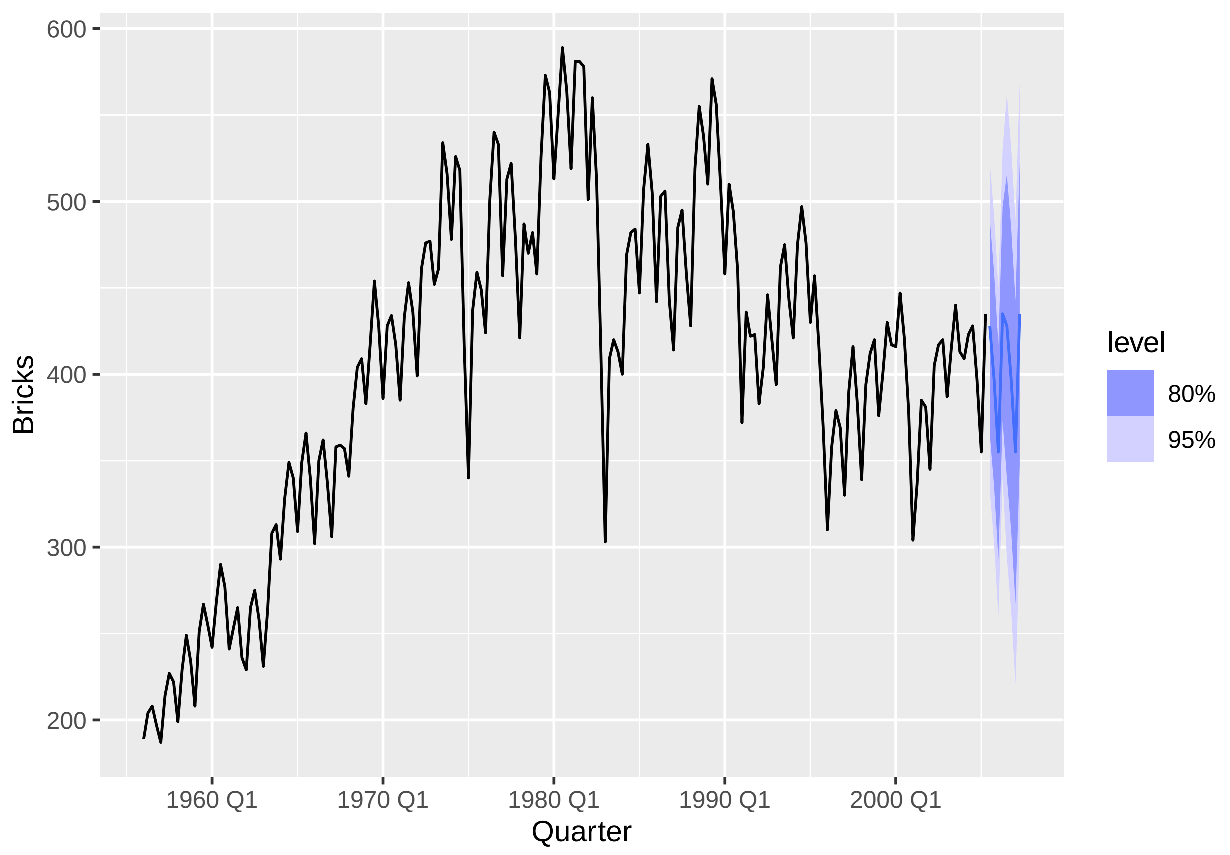
```r
# Data covers from Q1 1956 to Q2 2005 (198 quarters)
aus_bricks <- aus_production %>%
  select(Quarter, Bricks) %>% na.omit()

# Specify and fit the Seasonal Naive model
model_fit <- aus_bricks %>% model(SNAIVE(Bricks))

# Examine the residuals of the fitted model
model_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
# Generate and visualize forecasts
model_fit %>% forecast() %>% autoplot(aus_bricks)
```



The analysis of the residuals suggests that the model does not fully capture the data's characteristics. The innovation residuals indicate that the residuals are not white noise, as evidenced by the clear correlations seen in the ACF plot and the non-normal distribution of the histogram. Given these issues, the Seasonal Naive model was chosen out of the two available options, primarily because the NAIVE model showed significant quarterly correlations in the ACF. The forecast plot does align with the observed seasonal patterns in the data, indicating that the Seasonal Naive method captures seasonal variations. However, the model seems to lack a cyclic component, which might be contributing to the less-than-ideal residual performance. While the Seasonal Naive model is a better choice compared to the NAIVE method in this case, it still does not fully address all aspects of the data, such as cyclic patterns, making it a suboptimal choice for this dataset.

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

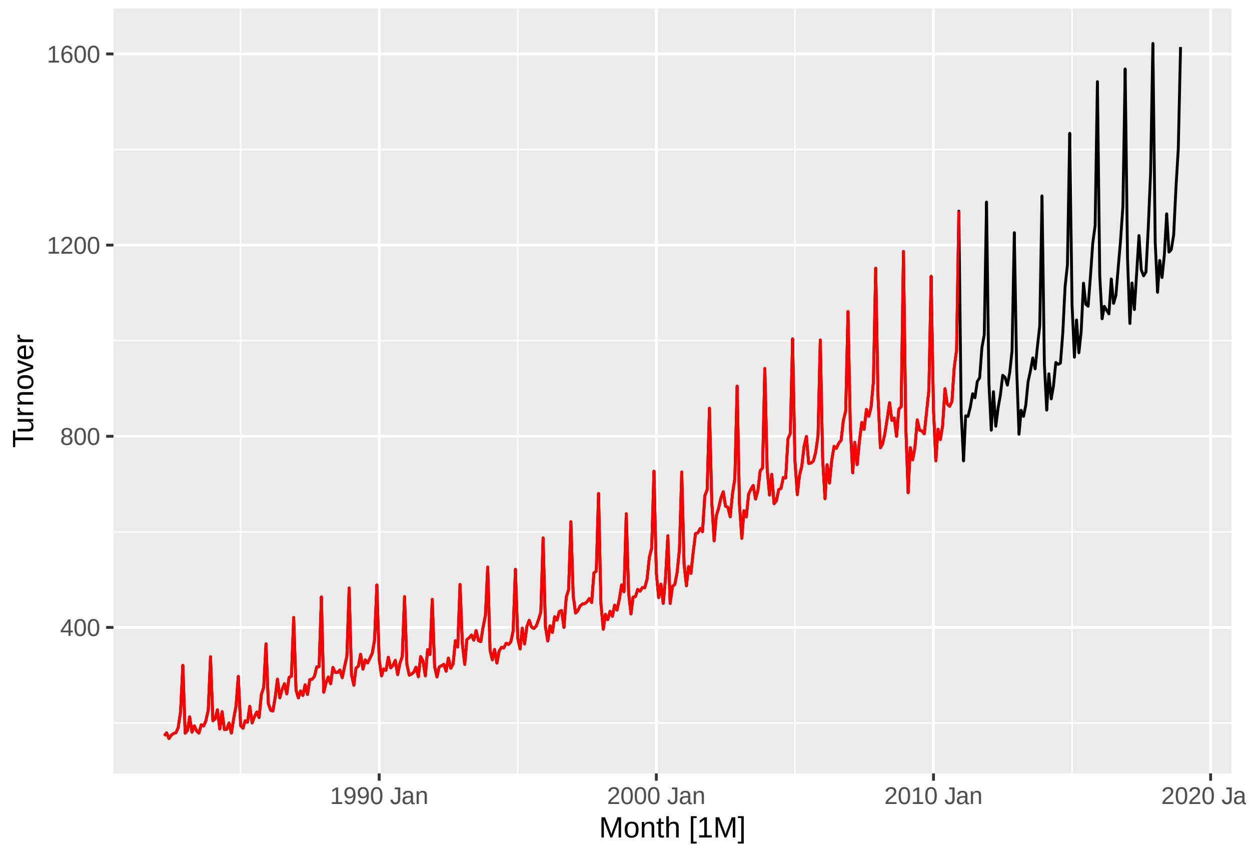
```
set.seed(123)
myseries <- aus_retail %>%
  filter(`Series ID` == sample(aus_retail$`Series ID`,1))
```

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.

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



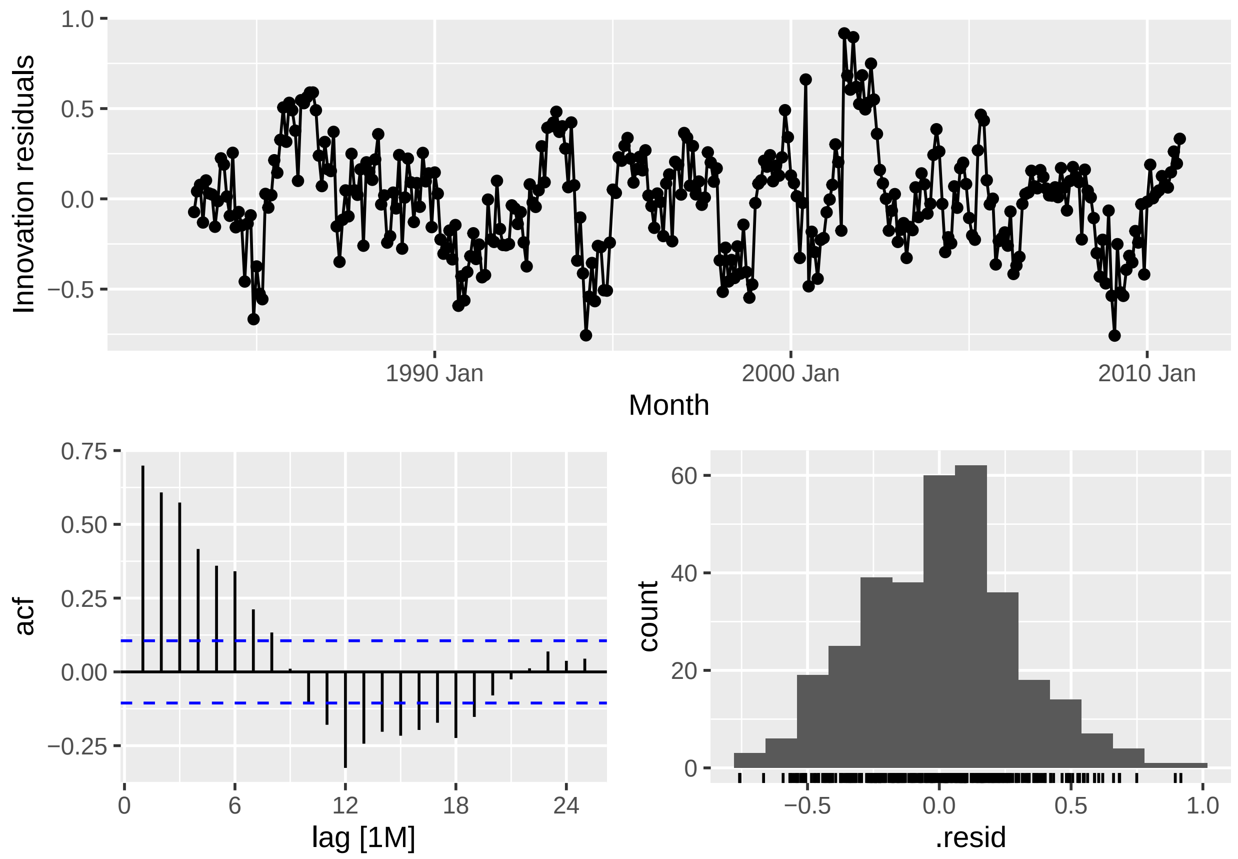
c. Fit a seasonal naïve model using SNAIVE() applied to your training data (myseries_train).

```r
# lambda from Assignment 2
lambda <- 0.22

fit <- myseries_train %>%
  model(SNAIVE(box_cox(Turnover, lambda) ~ drift()))

fit_3 <- myseries_train %>%
  model(
    `SNAIVE` = SNAIVE(Turnover),
    `SNAIVE with drift` = SNAIVE(Turnover ~ drift()),
    `SNAIVE with drift and Box-Cox` = SNAIVE(box_cox(Turnover, lambda) ~ drift()))
```

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

```
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()`).
```



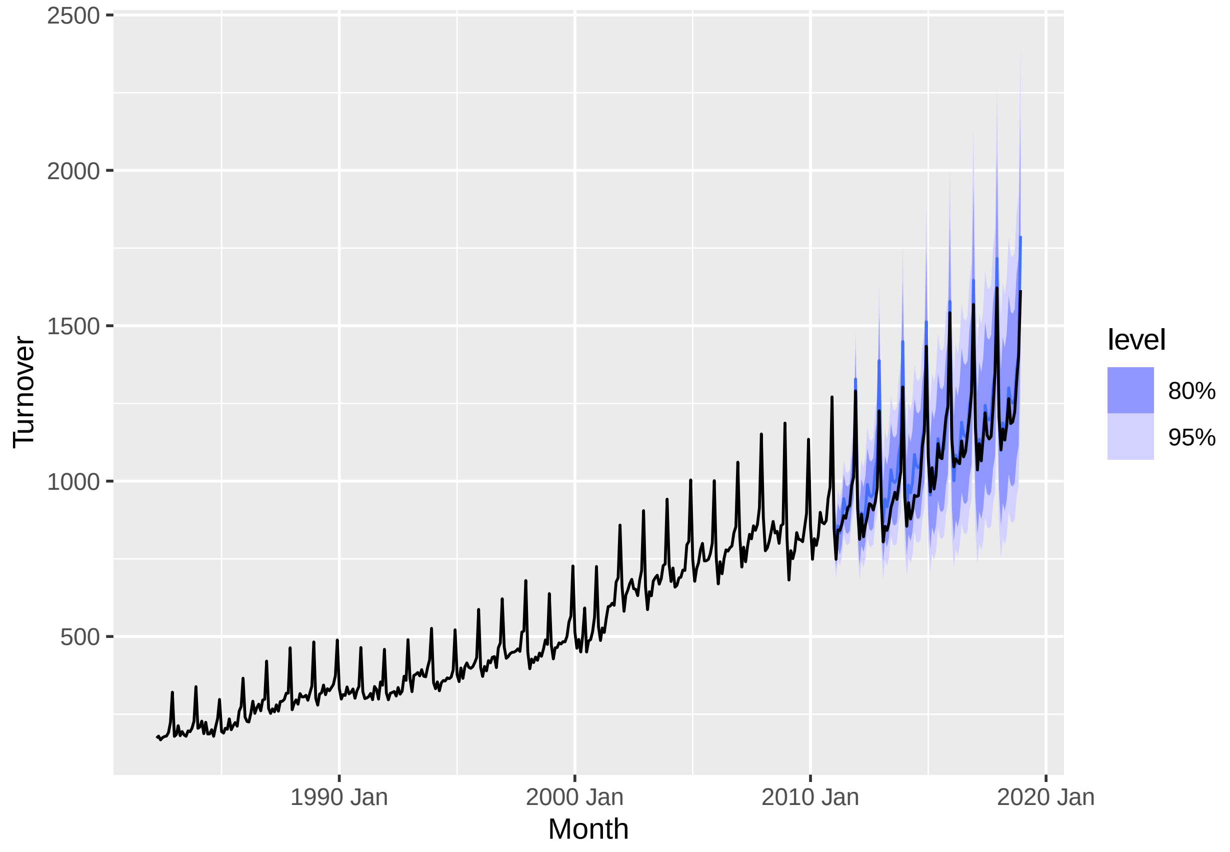
The residuals display a right skew and are not centered around zero. Additionally, the ACF plot reveals some autocorrelation in the dataset, suggesting that the residuals do not exhibit consistent variation.

e. Produce forecasts for the test data

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

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

```
fc %>% autoplot(myseries)
```



The forecast using the Box-Cox transformation follows the overall direction of the time series, though it doesn't fully capture the increasing seasonal pattern. However, the actual values falling within the 80% confidence interval suggest a decent fit. On the other hand, the drift model effectively incorporates both the seasonal variation and the upward trend, while the basic seasonal naive method only accounts for seasonality, overlooking any trends in the data.

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 Victo~ Househo~ SNAIV~ Trai~ -0.662  38.2  28.7 -0.256  6.01 0.809 0.839 0.695
```

```
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(b~ Vict~ Househo~ Test  -43.2  59.8  46.5 -4.16  4.47  1.31  1.31 0.474
```

The error is significantly lower on the training set when compared to the test set.

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

I believe the accuracy of the model is heavily influenced by the amount of training data used. As the book suggests, about 80% of the total data should be allocated for training, leaving 20% for testing. The size of the training set directly affects the model's calculation, which in turn impacts the fitted values and forecast outcomes. If the training data is insufficient, the model may struggle to perform well on the test set. Even models that fit the training data perfectly can fail on the test set due to overfitting. This relationship between training data size and model performance highlights that forecasting is both an art and a science.