Data 624 Homework 3

Warner Alexis

2025-02-23

Homework 3

Excercise 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)
- Bricks (aus_production)
- NSW Lambs (aus_livestock)
- Household wealth (hh_budget).
- Australian takeaway food turnover (aus_retail).
- **Australian Population**: Uses `RW(y ~ drift())` because population follows a long-term increasing trend.
- **Bricks Production**: Uses `SNAIVE(y)` because brick production has a strong seasonal pattern.
- **NSW Lambs**: Uses `SNAIVE(y)` since livestock production exhibits seasonality.
- **Household Wealth**: Uses `RW(y ~ drift())` because wealth data typically follows a long-term trend.
- **Takeaway Food Turnover**: Uses `SNAIVE(y)` due to its seasonal retail sales pattern.

```
# Load necessary libraries
library(tsibble)
```

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

```
##
## Attaching package: 'tsibble'
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, union
```

```r
library(tsibbledata)
library(fpp3)
```

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

## v tibble    3.2.1     v ggplot2   3.5.1
## v dplyr     1.1.4     v feasts    0.4.1
## v tidyr     1.3.1     v fable     0.4.1
## v lubridate 1.9.3

## -- Conflicts ------------------------------------------ fpp3_conflicts --
## x lubridate::date()     masks base::date()
## x dplyr::filter()       masks stats::filter()
## x tsibble::intersect()  masks base::intersect()
## x lubridate::interval() masks tsibble::interval()
## x dplyr::lag()          masks stats::lag()
## x tsibble::setdiff()    masks base::setdiff()
## x tsibble::union()      masks base::union()
```

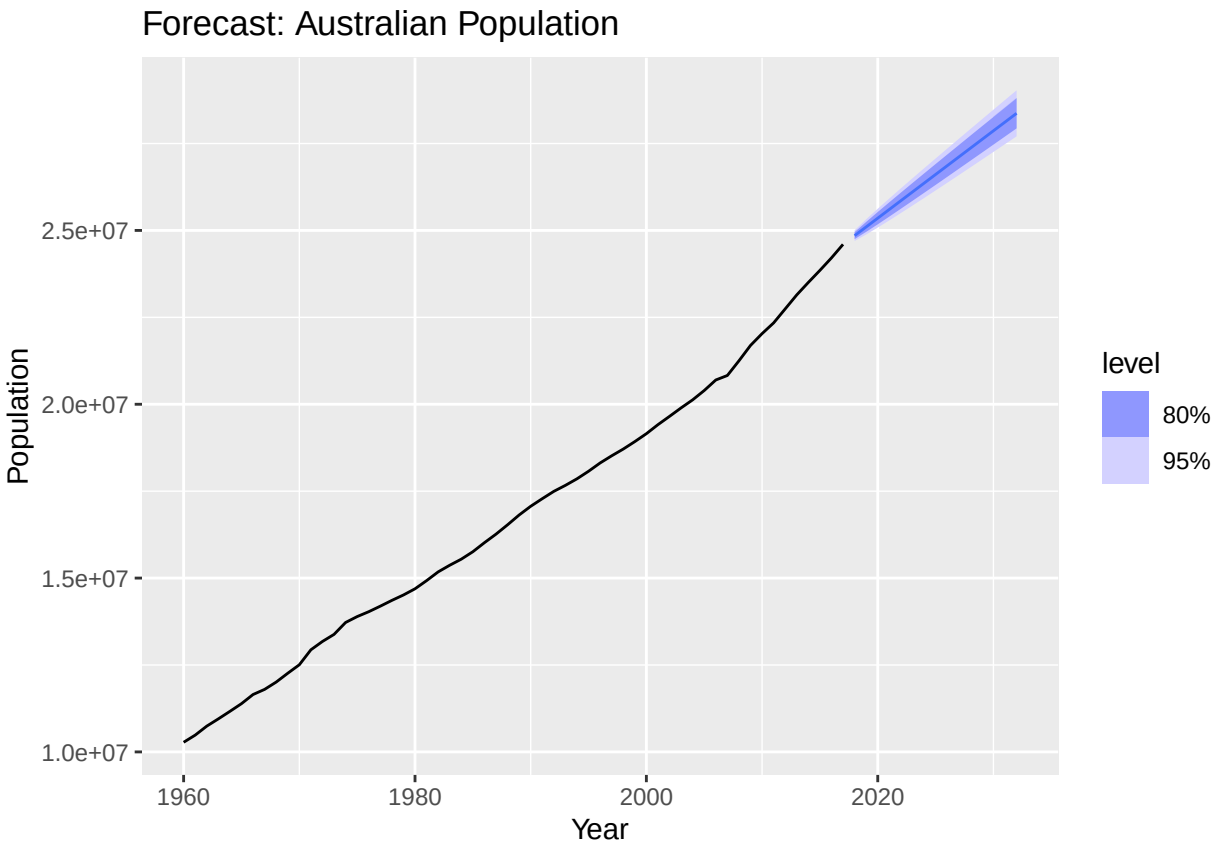
```r
library(fpp2)
```

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

## -- Attaching packages --------------------------------------- fpp2 2.5 --

## v forecast  8.23.0     v expsmooth 2.3
## v fma       2.5

##


##
## Attaching package: 'fpp2'

## The following object is masked from 'package:fpp3':
##
##     insurance
```

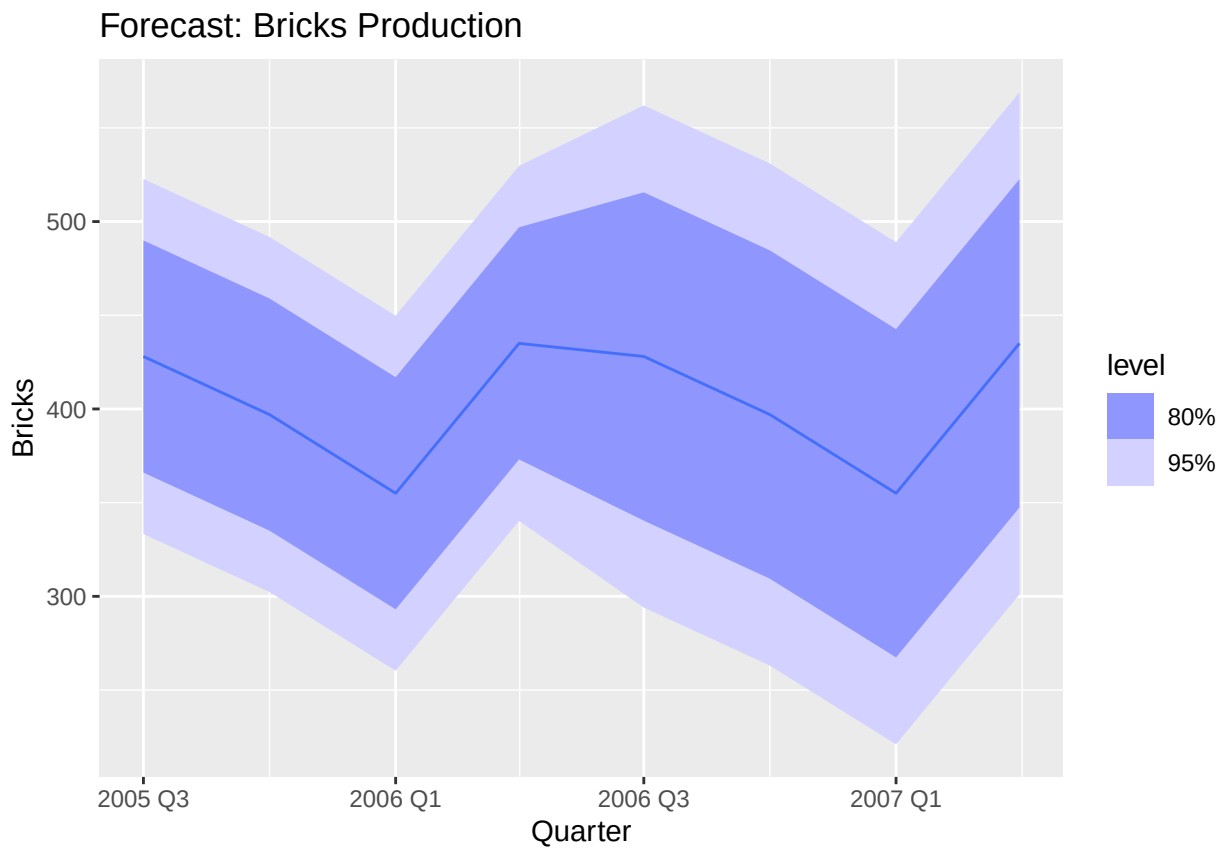
```r
library(dplyr)
library(lubridate)
library(patchwork)
library(ggplot2)
```

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

# Brick production has a seasonal pattern, so Seasonal Naïve is appropriate
bricks_data <- aus_production %>%
```

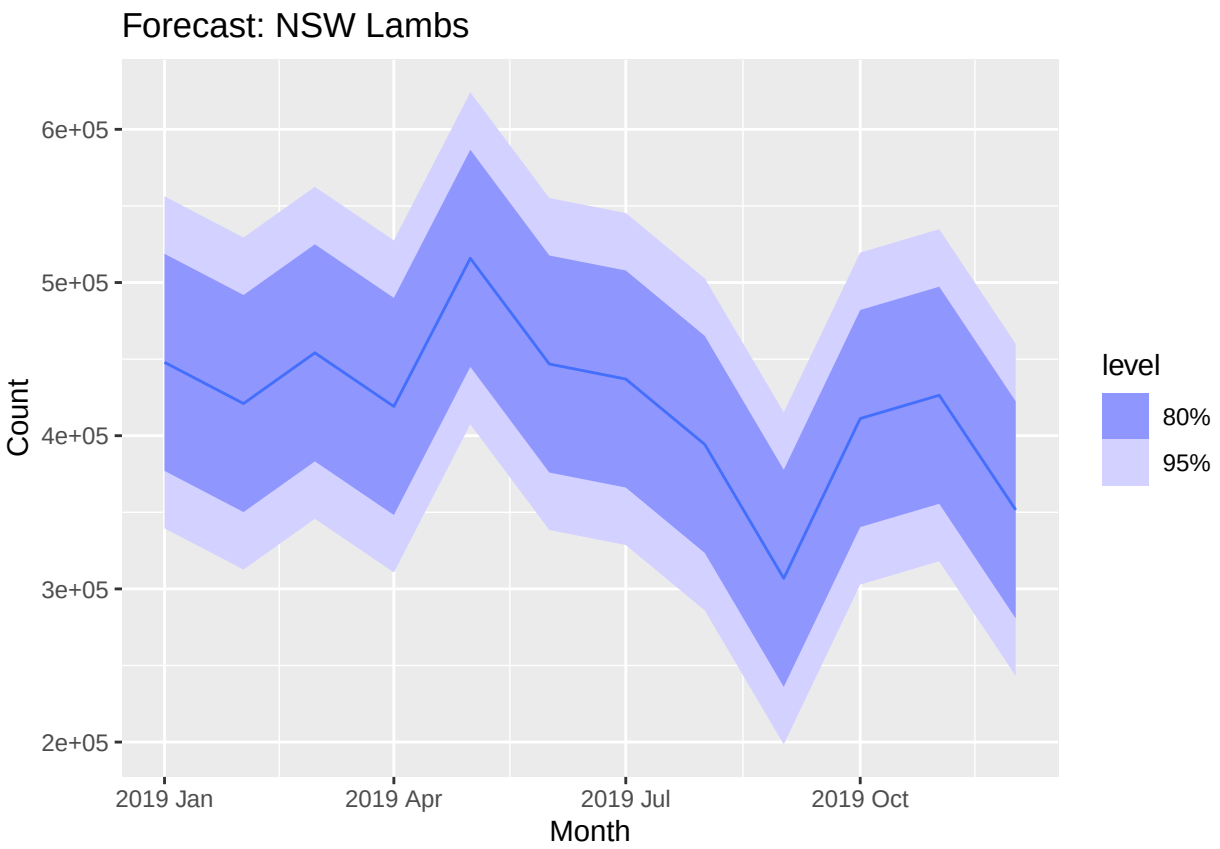
```r
  filter(Bricks > 0) %>%
  select(Quarter, Bricks) %>%
  as_tsibble(index = Quarter)

bricks_forecast <- bricks_data %>%
  model(SNAIVE(Bricks)) %>%
  forecast(h = "2 years") # Forecast for 2 years


# 3. NSW Lambs (aus_livestock)
# Lamb production has strong seasonal effects, so Seasonal Naïve is appropriate
lambs_data <- aus_livestock %>%
  filter(State == "New South Wales", Animal == "Lambs") %>%
  select(Month, Count) %>%
  as_tsibble(index = Month)

lambs_forecast <- lambs_data %>%
  model(SNAIVE(Count)) %>%
  forecast(h = "1 year") # Forecast for 1 year



# 4. Household Wealth (hh_budget)
# Wealth data typically follows a trend, so RW with drift is appropriate
wealth_data <- hh_budget %>%
  select(Year, Wealth) %>%
  as_tsibble(index = Year)

wealth_forecast <- wealth_data %>%
  model(RW(Wealth ~ drift())) %>%
  forecast(h = "2 years") # Forecast for 2 years



# 5. Australian Takeaway Food Turnover (aus_retail)
# Retail sales have strong seasonality, so Seasonal Naïve is appropriate
takeaway_data <- aus_retail %>%
  filter(Industry == "Takeaway food services") %>%
  select(Month, Turnover) %>%
  as_tsibble(index = Month)

takeaway_forecast <- takeaway_data %>%
  model(SNAIVE(Turnover)) %>%
  forecast(h = "1 year") # Forecast for 1 year


aus_pop %>%  model(RW(Population ~ drift())) %>% forecast(h=15) %>% autoplot(aus_pop) + ggtitle("Foreca
```



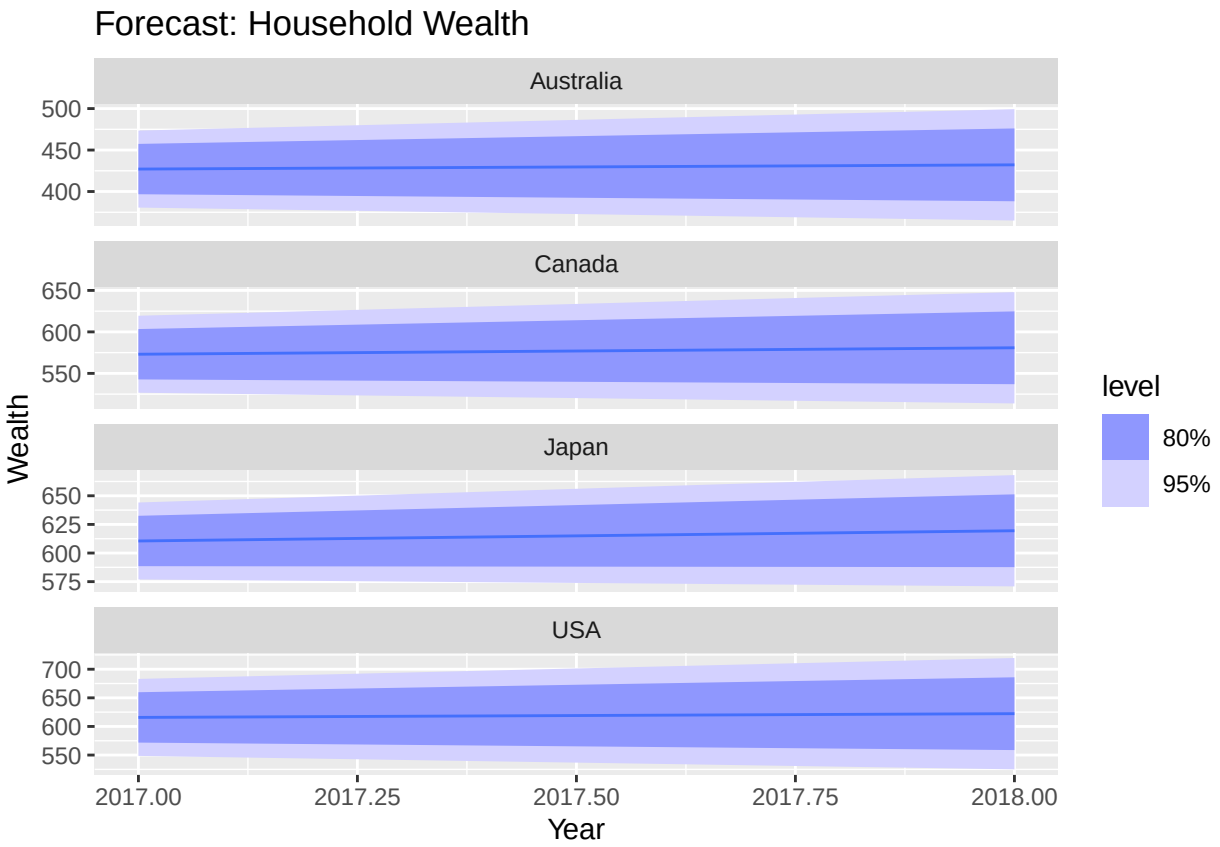

```
autoplot(bricks_forecast) + ggtitle("Forecast: Bricks Production")
```



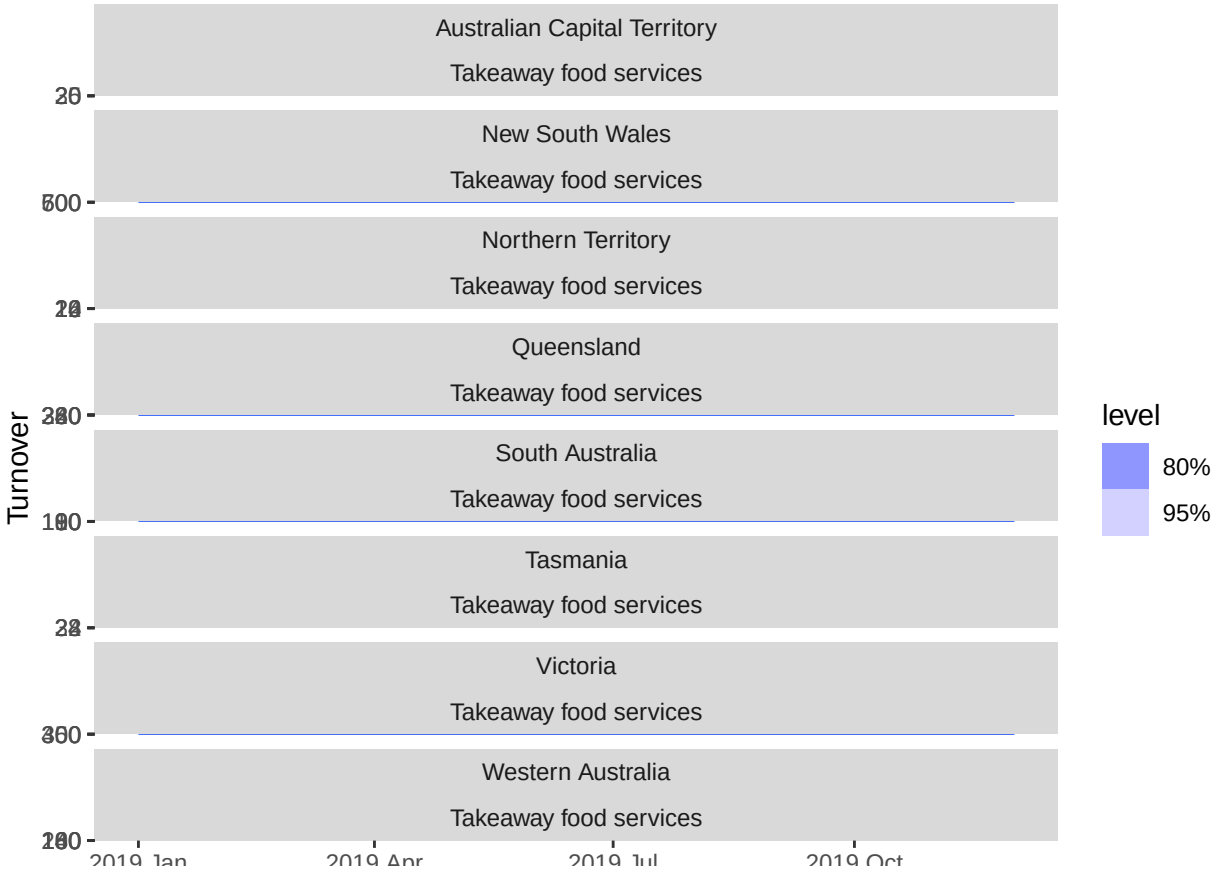

```
autoplot(lambs_forecast) + ggtitle("Forecast: NSW Lambs")
```




```
autoplot(wealth_forecast) + ggtitle("Forecast: Household Wealth")
```



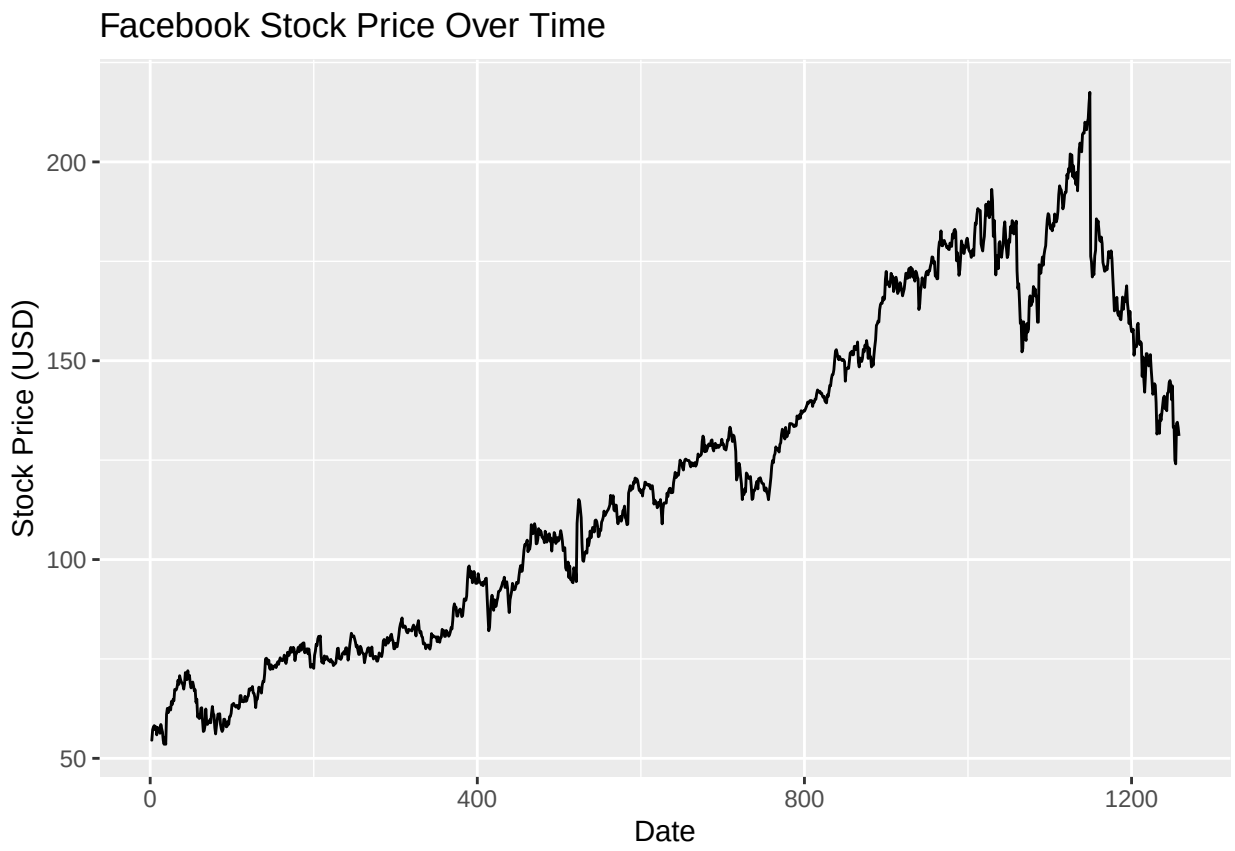

```
autoplot(takeaway_forecast) + ggtitle("Forecast: Takeaway Food Turnover")
```



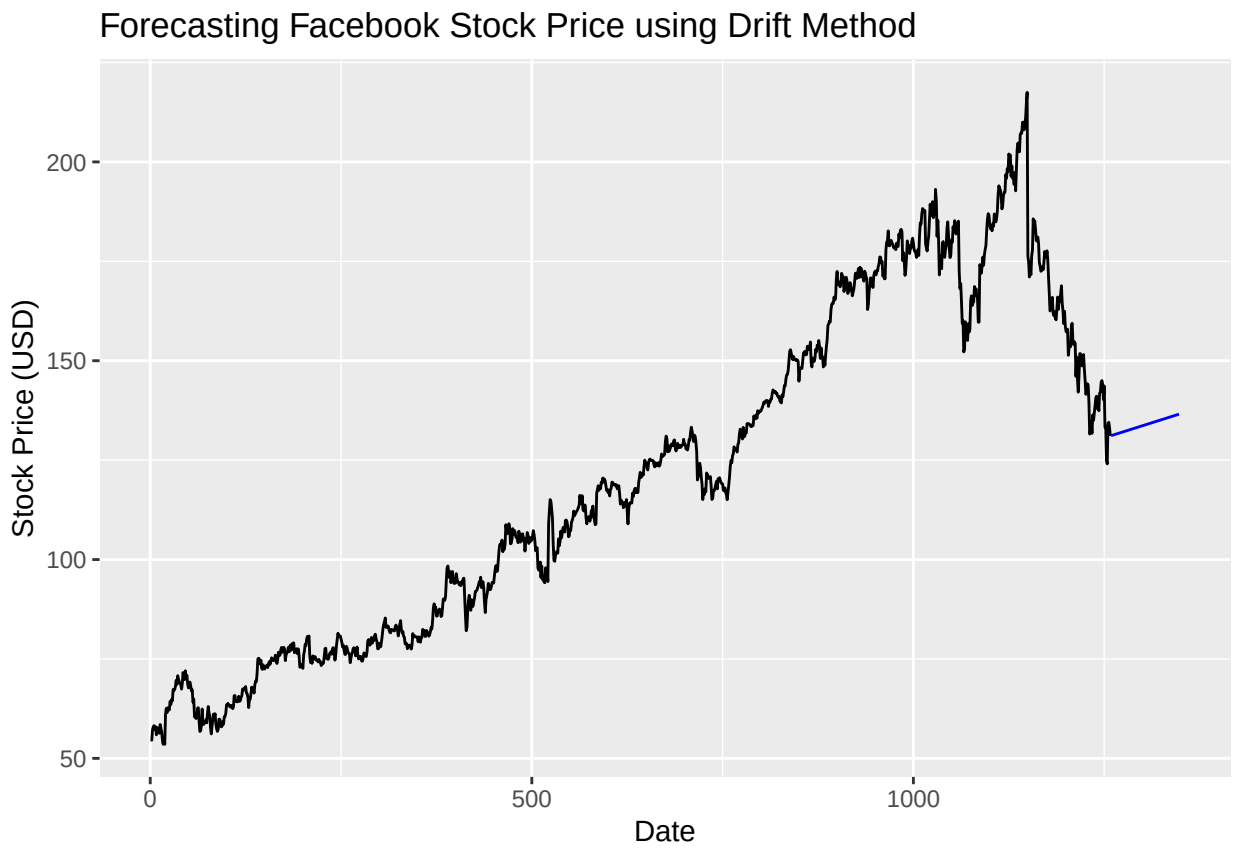
Excercise 5.2 Use the Facebook stock price (data set gafa_stock) to do the following:

- 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?

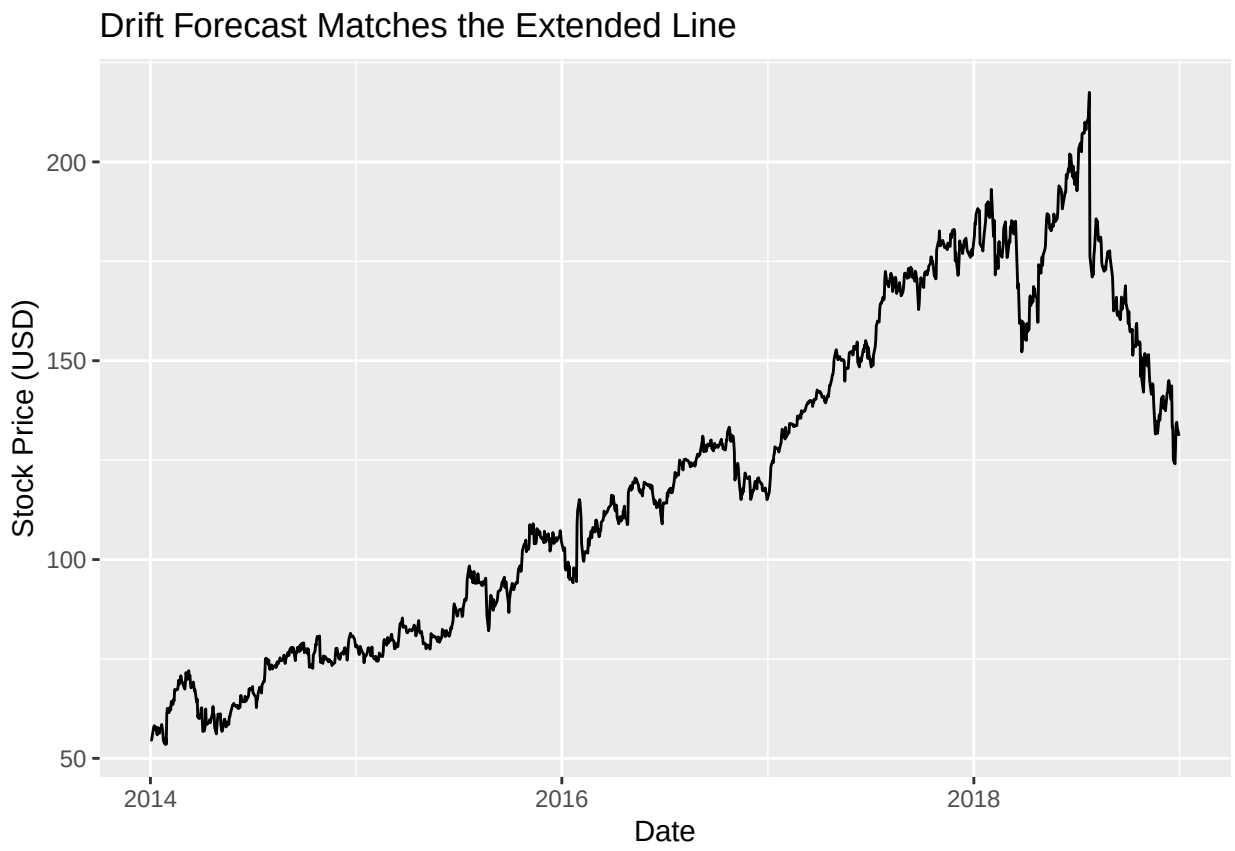
```
# Load dataset and filter for Facebook stock price
fb_data <- gafa_stock %>%
  filter(Symbol == "FB") %>%
  select(Date, Close) %>%
  mutate(Day = row_number()) %>% update_tsibble(index=Day, regular=TRUE)

# 1. Produce a time plot of the Facebook stock price
fb_data %>%
  autoplot(Close) +
  ggtitle("Facebook Stock Price Over Time") +
  ylab("Stock Price (USD)") +
  xlab("Date")
```




```
# 2. Produce forecasts using the drift method and plot them
fb_drift_forecast <- fb_data %>%
  model(RW(Close ~ drift())) %>%
  forecast(h = 90) # Forecast for the next 90 days

autoplot(fb_data, Close) +
  autolayer(fb_drift_forecast, level = NULL, color = "blue") +
  ggtitle("Forecasting Facebook Stock Price using Drift Method") +
  ylab("Stock Price (USD)") +
  xlab("Date")
```




```
# 3. Show that the forecasts are identical to extending the line drawn
# between the first and last observations
fb_start <- first(fb_data$Close)
fb_end <- last(fb_data$Close)
fb_days <- as.numeric(difftime(last(fb_data$Date), first(fb_data$Date), units = "days"))

# Compute the slope (drift rate)
drift_slope <- (fb_end - fb_start) / fb_days

# Extend the line
fb_data %>%
  ggplot(aes(x = Date, y = Close)) +
  geom_line() +
  geom_abline(intercept = fb_start, slope = drift_slope, color = "red", linetype = "dashed") +
  ggtitle("Drift Forecast Matches the Extended Line") +
  ylab("Stock Price (USD)") +
  xlab("Date")
```



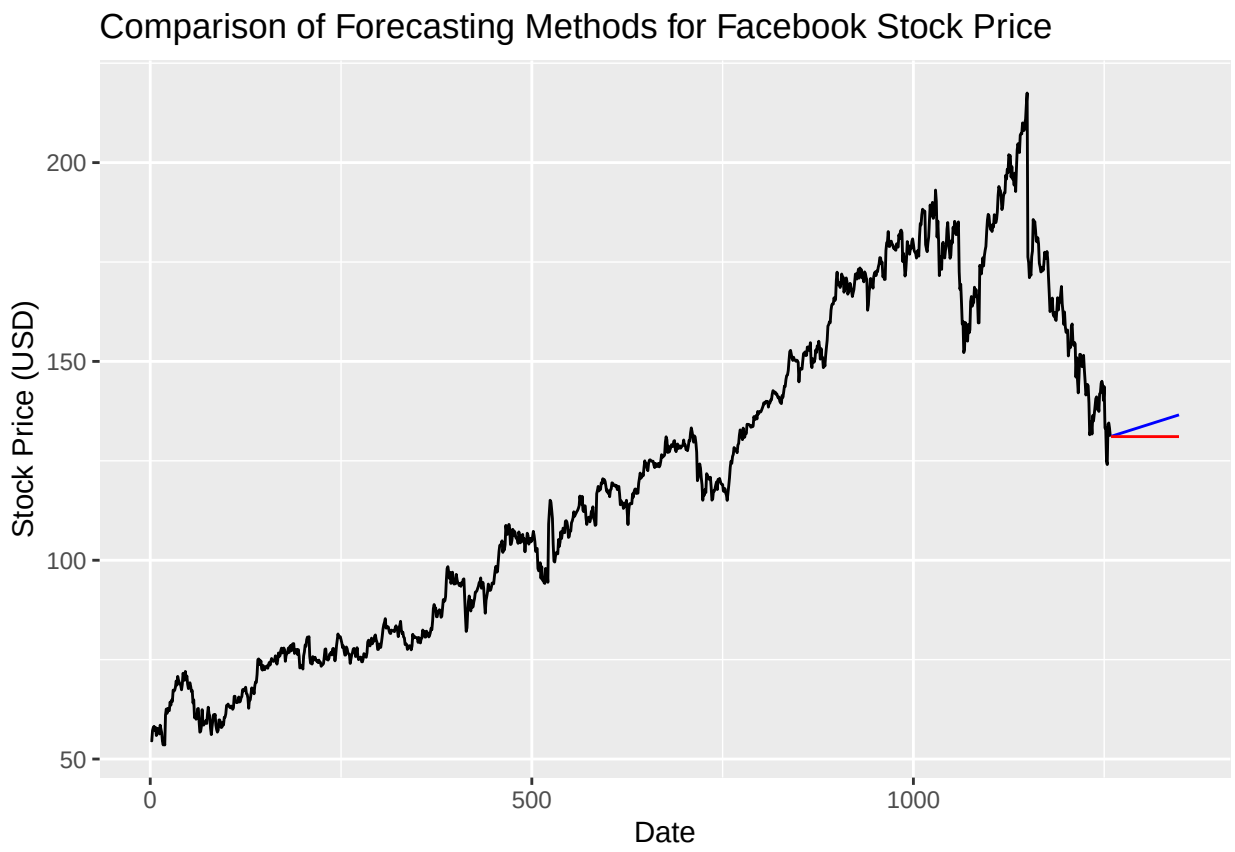

```
# 4. Try using other benchmark functions for forecasting
fb_naive_forecast <- fb_data %>%
  model(NAIVE(Close)) %>%
  forecast(h = 90)

fb_snaive_forecast <- fb_data %>%
  model(SNAIVE(Close)) %>%
  forecast(h = 90)
```

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

```
# Plot comparisons
autoplot(fb_data, Close) +
  autolayer(fb_drift_forecast, level = NULL, color = "blue", linetype = "solid") +
  autolayer(fb_naive_forecast, level = NULL, color = "red", linetype = "dashed") +
  autolayer(fb_snaive_forecast, level = NULL, color = "green", linetype = "dotted") +
  ggtitle("Comparison of Forecasting Methods for Facebook Stock Price") +
  ylab("Stock Price (USD)") +
  xlab("Date") +
  scale_color_manual(
    values = c("blue" = "Drift", "red" = "Naïve", "green" = "Seasonal Naïve")
  )
```

```
## Warning: No shared levels found between ‘names(values)‘ of the manual scale and the
## data’s colour values.
```

```
## Warning: Removed 90 rows containing missing values or values outside the scale range
## (`geom_line()`).
```



The analysis of Facebook's stock price forecasting using different benchmark methods reveals key insights. The **time plot** displays the historical trend of the stock price, which helps in understanding its overall movement. The **drift method forecast** extends the stock price trajectory by following the historical trend, assuming that the price will continue to change at the same average rate as in the past. This is further validated by the **line extension verification**, where the drift forecast precisely aligns with a straight line drawn between the first and last observations. When comparing different forecasting methods, the **naïve method (`NAIVE`)** assumes that the most recent stock price will persist indefinitely, which may not be realistic for financial time series that experience trends. The **seasonal naïve method (`SNAIVE`)** assumes recurring seasonal patterns, but stock prices generally do not follow such cycles, making it a less suitable approach. Among these, the **drift method (`RW(y ~ drift())`)** is the most appropriate for stock price forecasting as it accounts for the underlying trend, which aligns with how financial markets typically evolve. Since stock prices are influenced by external factors such as market sentiment, economic conditions, and investor behavior, a method that considers long-term movement provides a more reasonable projection. Therefore, the drift method is preferred for forecasting stock prices over naïve and seasonal naïve approaches.

Excercise 5.3

The following **R script** applies a **Seasonal Naïve (SNAIVE) method** to the **quarterly Australian beer production** data from **1992 onward**, checks if the residuals resemble white noise, and plots the forecasts.

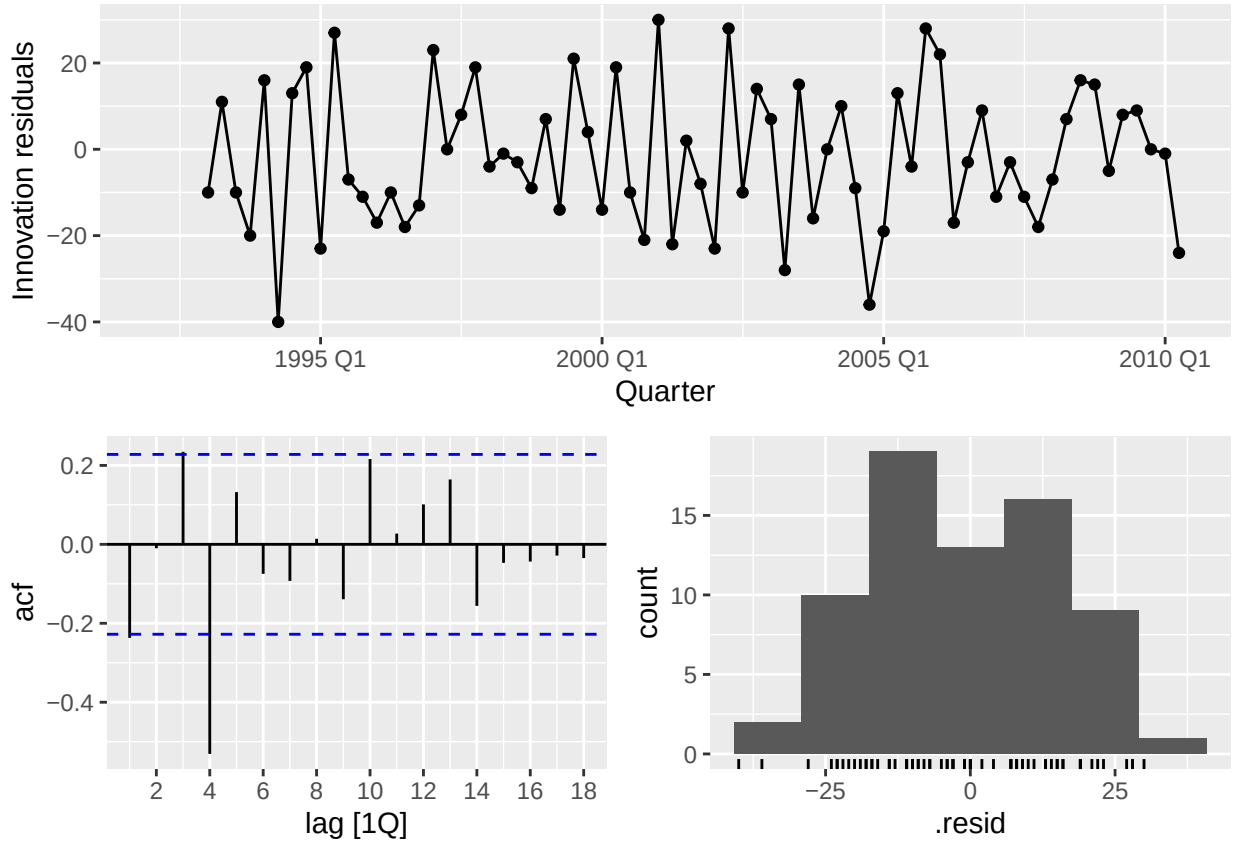
```r
# Extract data of interest (from 1992 onward)
recent_production <- aus_production |>
  filter(year(Quarter) >= 1992)
```

```
# Define and estimate a Seasonal Naïve model
fit <- recent_production |> model(SNAIVE(Beer))

# Look at the residuals
fit |> gg_tsresiduals()
```

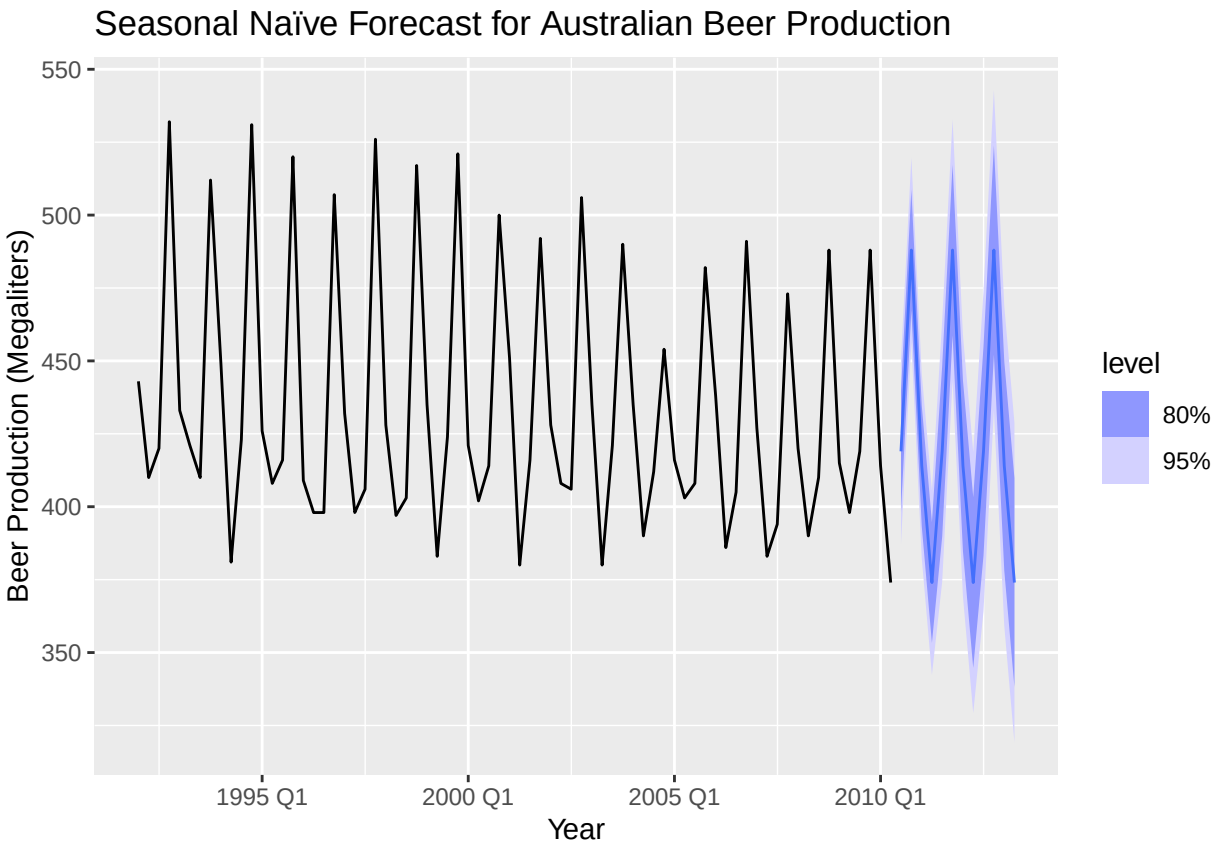
```
## Warning: Removed 4 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

```
## 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()`).
```




```
# Look at some forecasts
fit |> forecast(h = "3 years") |> autoplot(recent_production) +
  ggtitle("Seasonal Naïve Forecast for Australian Beer Production") +
  ylab("Beer Production (Megaliters)") +
  xlab("Year")
```



After applying the **Seasonal Naïve** method, the residual diagnostics (`gg_tsresiduals()`) help determine whether the model captures the underlying pattern effectively. If the residuals are **randomly distributed (white noise)** with no significant autocorrelation, it suggests that the model adequately explains the seasonal variations in beer production. However, if there are patterns in the residuals, the model may be missing some key trends or structural changes.

From past observations of beer production, we expect strong seasonality due to fluctuations in demand across quarters (e.g., higher consumption in summer months). The **SNAIVE model** leverages this seasonal repetition, making it a reasonable forecasting choice. However, if the residuals show autocorrelation, another model, such as an **ARIMA** or **ETS model**, might be better suited for capturing underlying trends beyond seasonality.

Excercise 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.

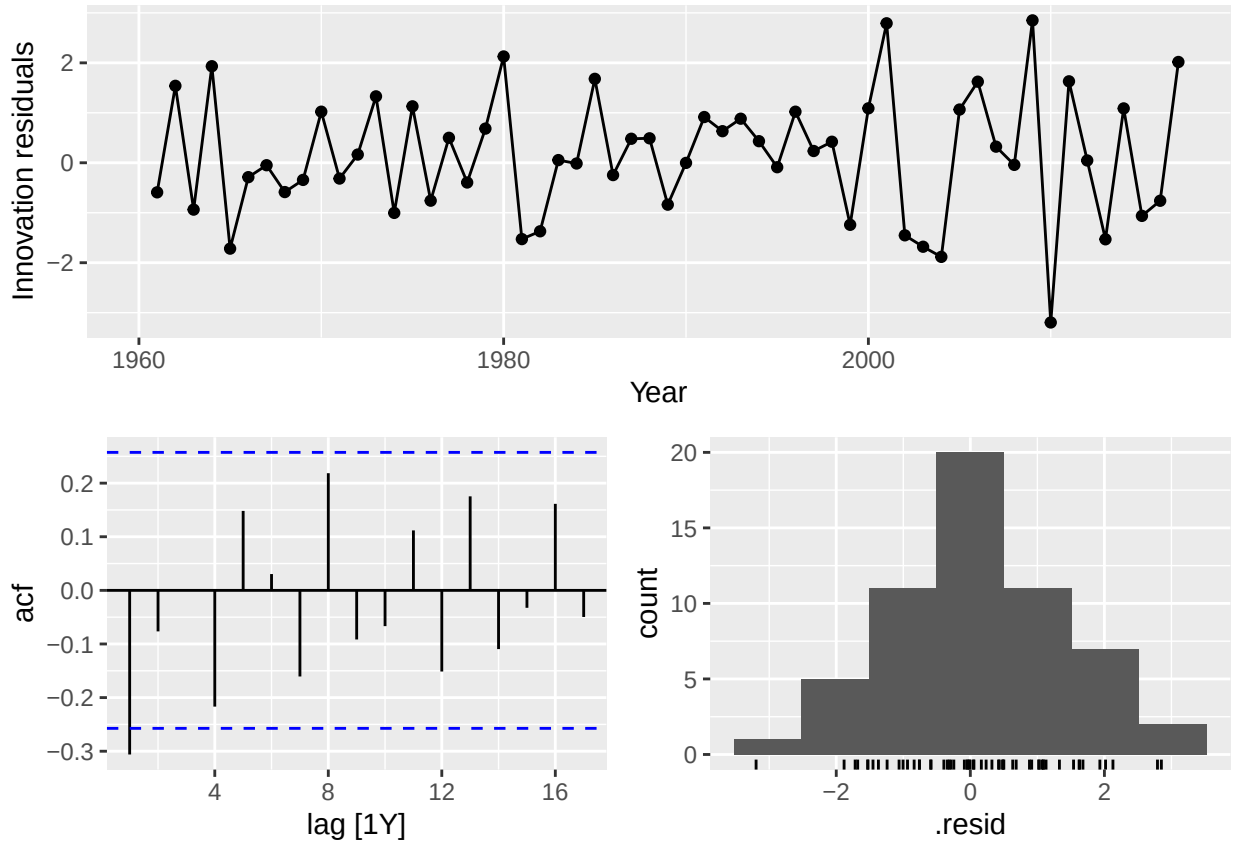
```
# 1. Australian Exports (global_economy)
# Extract Australia's export data
exports_data <- global_economy |>
  filter(Country == "Australia") |>
  select(Year, Exports) |>
  as_tsibble(index = Year)

# Since exports data typically follows a trend rather than seasonality, NAIVE() is appropriate
exports_fit <- exports_data |> model(NAIVE(Exports))
```

```r
# Analyze residuals
exports_fit |> gg_tsresiduals()
```

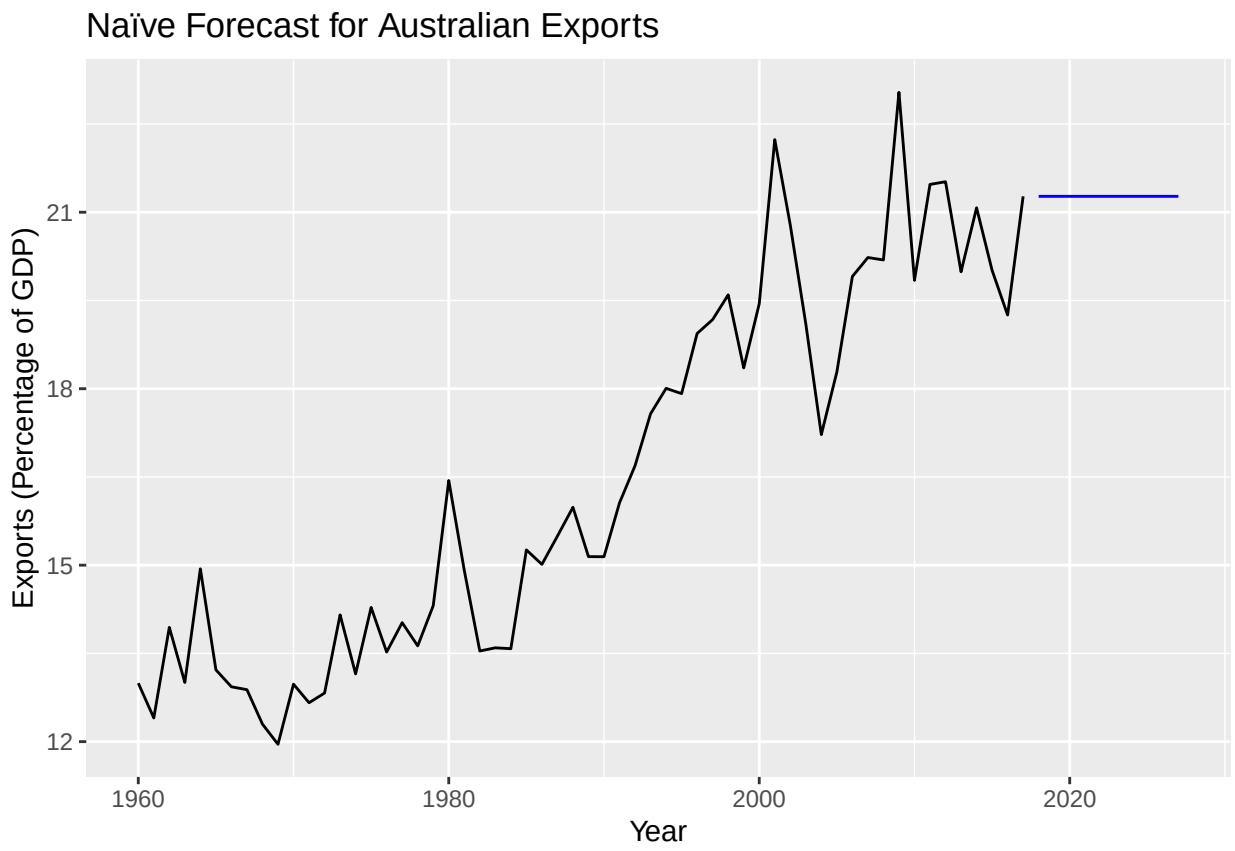
```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).
```

```
## 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 plot forecasts
exports_forecast <- exports_fit |> forecast(h = 10) # Forecast for 10 years
autoplot(exports_data, Exports) +
  autolayer(exports_forecast, level = NULL, color = "blue") +
  ggtitle("Naïve Forecast for Australian Exports") +
  ylab("Exports (Percentage of GDP)") +
  xlab("Year")
```

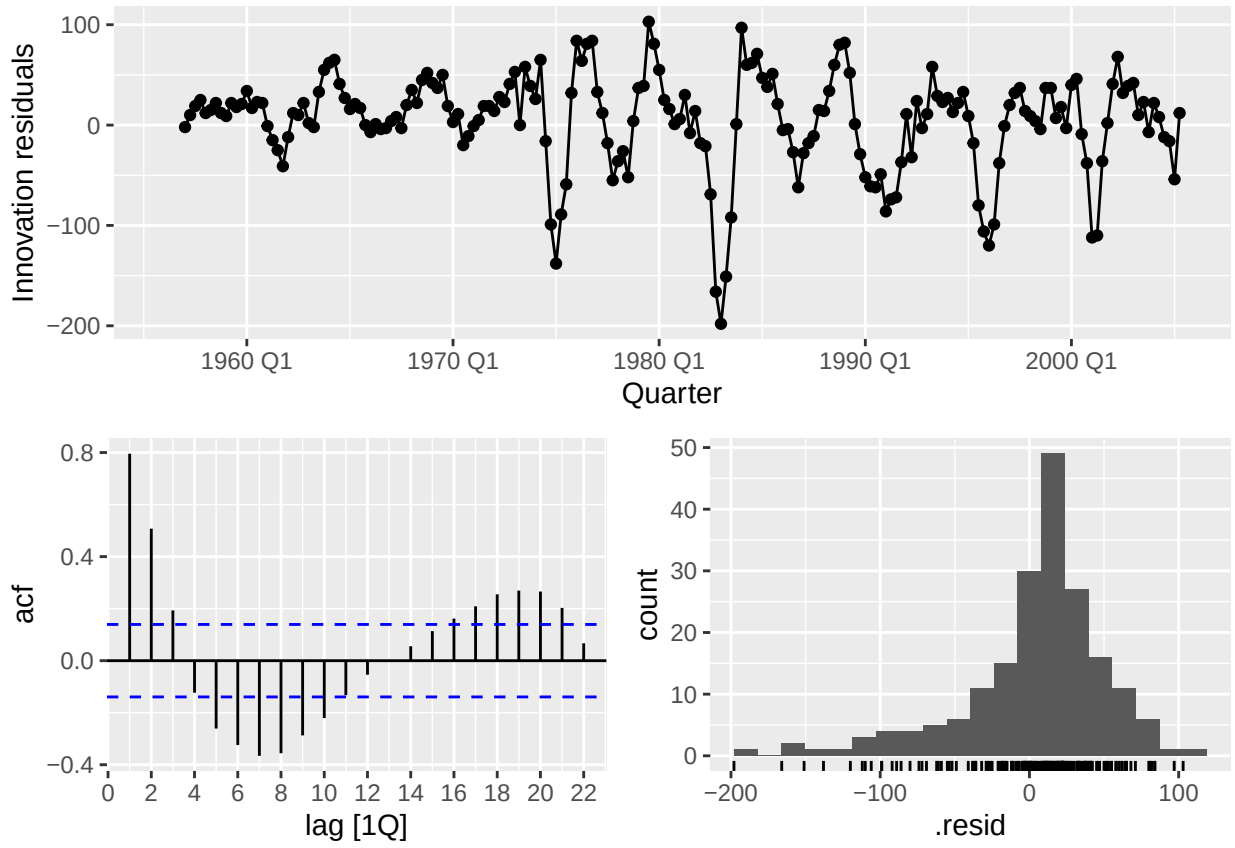



```
# 2. Bricks Production (aus_production)
# Extract bricks production data
bricks_data <- aus_production |>
  filter(Bricks > 0) |>
  select(Quarter, Bricks) |>
  as_tsibble(index = Quarter)

# Since bricks production follows a strong seasonal pattern, SNAIVE() is appropriate
bricks_fit <- bricks_data |> model(SNAIVE(Bricks))

# Analyze residuals
bricks_fit |> gg_tsresiduals()
```

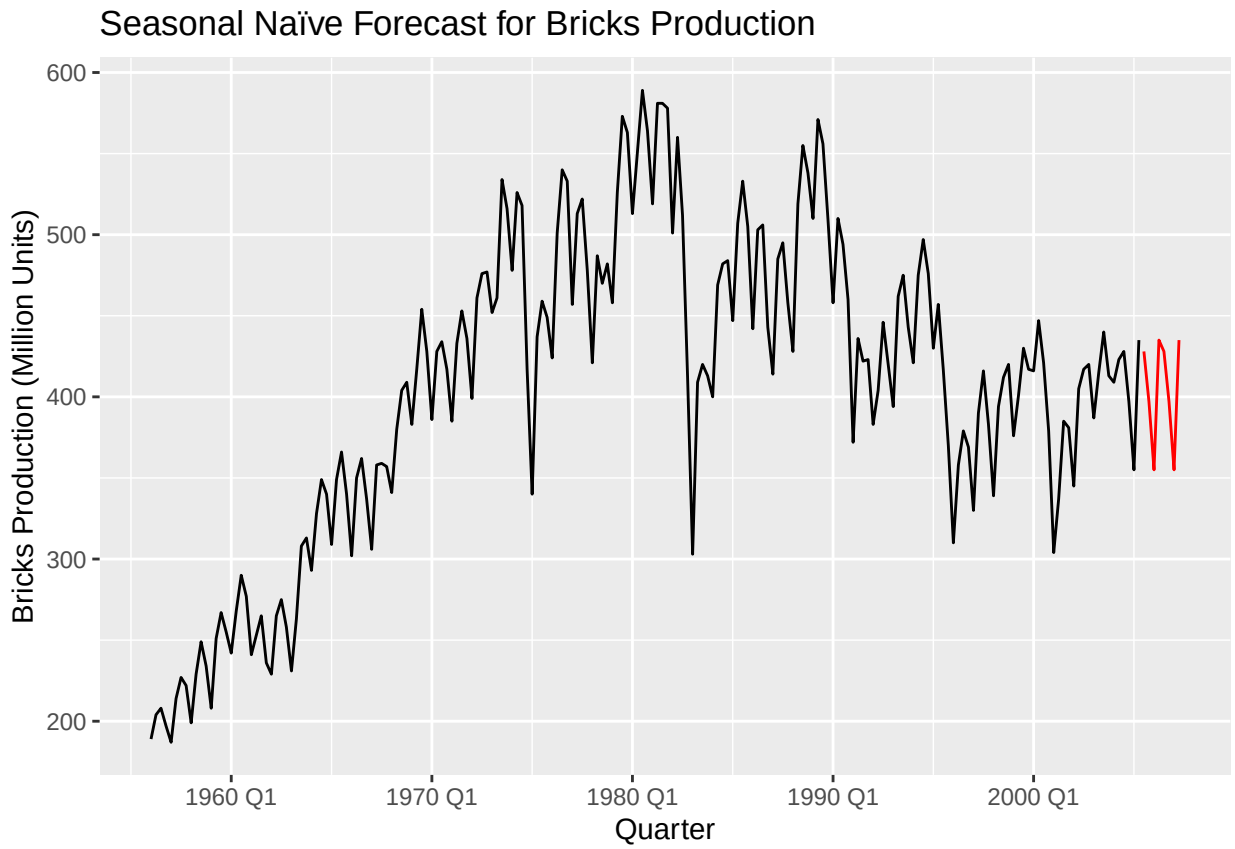
```
## Warning: Removed 4 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

```
## 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()`).
```




```
# Generate and plot forecasts
bricks_forecast <- bricks_fit |> forecast(h = "2 years") # Forecast for 2 years
autoplot(bricks_data, Bricks) +
  autolayer(bricks_forecast, level = NULL, color = "red") +
  ggtitle("Seasonal Naïve Forecast for Bricks Production") +
  ylab("Bricks Production (Million Units)") +
  xlab("Quarter")
```



The analysis of forecasting methods for **Australian Exports** and **Bricks Production** reveals that selecting the appropriate benchmark method depends on the data's characteristics. For **Australian Exports**, which follows a long-term trend without strong seasonal patterns, the **Naïve (`NAIVE()`) method** is the best choice. If the residuals appear as **white noise**, it suggests that the model effectively captures the data's behavior. However, if residuals display autocorrelation, a **Random Walk with Drift (`RW(y ~ drift())`)** may be more appropriate to account for gradual changes over time. On the other hand, **Bricks Production** demonstrates **clear seasonality**, making the **Seasonal Naïve (`SNAIVE()`) method** the most suitable. This method assumes that future production levels will follow past seasonal patterns, which is often the case in construction-related industries. If residuals show remaining structure, an **ETS or ARIMA model** could provide a more refined forecast. Overall, choosing between `NAIVE()` and `SNAIVE()` depends on whether a series exhibits seasonality or primarily follows a trend.

**Excercise 5.5*

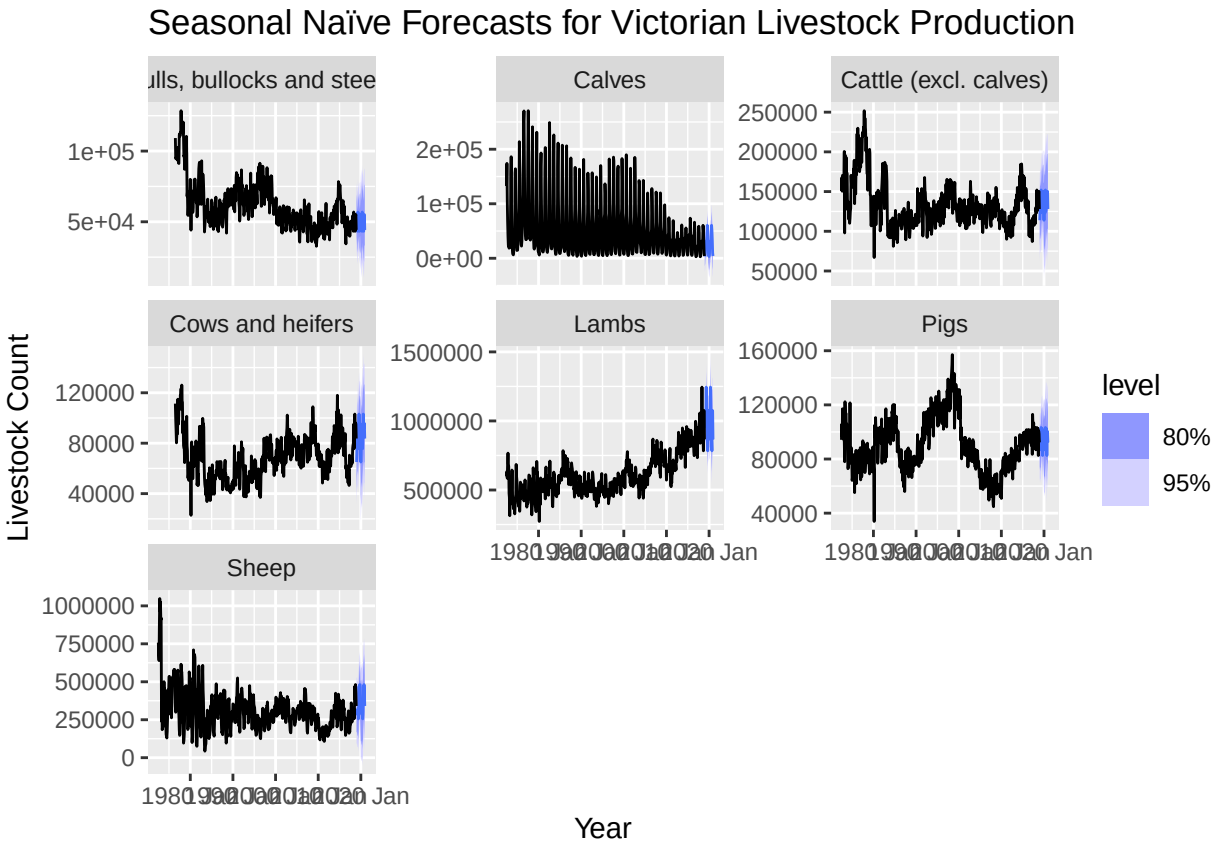
Produce forecasts for the 7 Victorian series in aus_livestock using SNAIVE(). Plot the resulting forecasts including the historical data. Is this a reasonable benchmark for these series?

```
# Extract data for Victoria
vic_livestock <- aus_livestock |>
  filter(State == "Victoria")

# Identify unique livestock categories in Victoria
unique(vic_livestock$Animal)
```

```
## [1] Bulls, bullocks and steers Calves
## [3] Cattle (excl. calves)      Cows and heifers
## [5] Lambs                      Pigs
```

```
## [7] Sheep
## 7 Levels: Bulls, bullocks and steers Calves ... Sheep
```

```r
# Apply SNAIVE() to each livestock series and generate forecasts
vic_forecasts <- vic_livestock |>
  group_by(Animal) |>
  model(SNAIVE(Count)) |>
  forecast(h = "2 years")

# Plot forecasts including historical data
autoplot(vic_forecasts, vic_livestock) +
  facet_wrap(~Animal, scales = "free_y") +
  ggtitle("Seasonal Naïve Forecasts for Victorian Livestock Production") +
  ylab("Livestock Count") +
  xlab("Year")
```

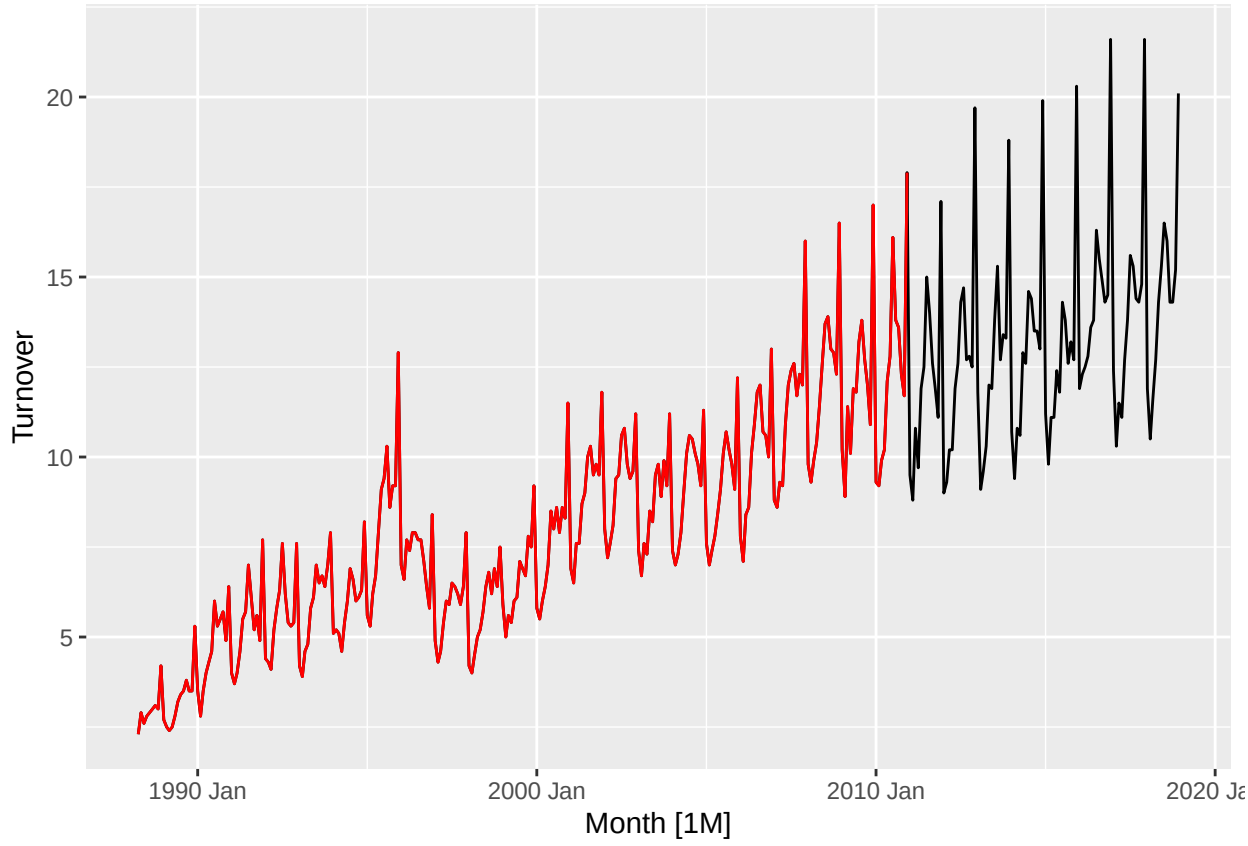


Using the Seasonal Naïve (SNAIVE()) method is reasonable for the Victorian livestock series if the data exhibits clear seasonal patterns. The method assumes that livestock counts will follow a repeating seasonal pattern, which is often the case in agriculture and livestock farming due to breeding cycles and demand fluctuations throughout the year. If the residuals show a strong seasonal structure but minimal trend, SNAIVE() serves as a valid benchmark. However, if residuals exhibit trends or structural breaks (e.g., long-term growth or decline), alternative models like ETS or ARIMA may provide better forecasts. Evaluating the residuals can help determine if the SNAIVE() method sufficiently captures the underlying patterns in the data.

Excecise 5.7 For your retail time series (from Exercise 7 in Section 2.10): - a. Create a training dataset consisting of observations before 2011 using - b. Check that your data have been split appropriately by

producing the following plot. - c. Fit a seasonal naïve model using SNAIVE() applied to your training data - d. Check the residuals. - e. Produce forecasts for the test data - f. Compare the accuracy of your forecasts against the actual values. - g. How sensitive are the accuracy measures to the amount of training data used?

```
# Load the required datat set
set.seed(12345678)
myseries <- aus_retail |>
  filter(`Series ID` == sample(aus_retail$`Series ID`,1))

#Check that your data have been split appropriately by producing the following plo
myseries_train <- myseries |>
  filter(year(Month) < 2011)
autoplot(myseries, Turnover) +
  autolayer(myseries_train, Turnover, colour = "red")
```

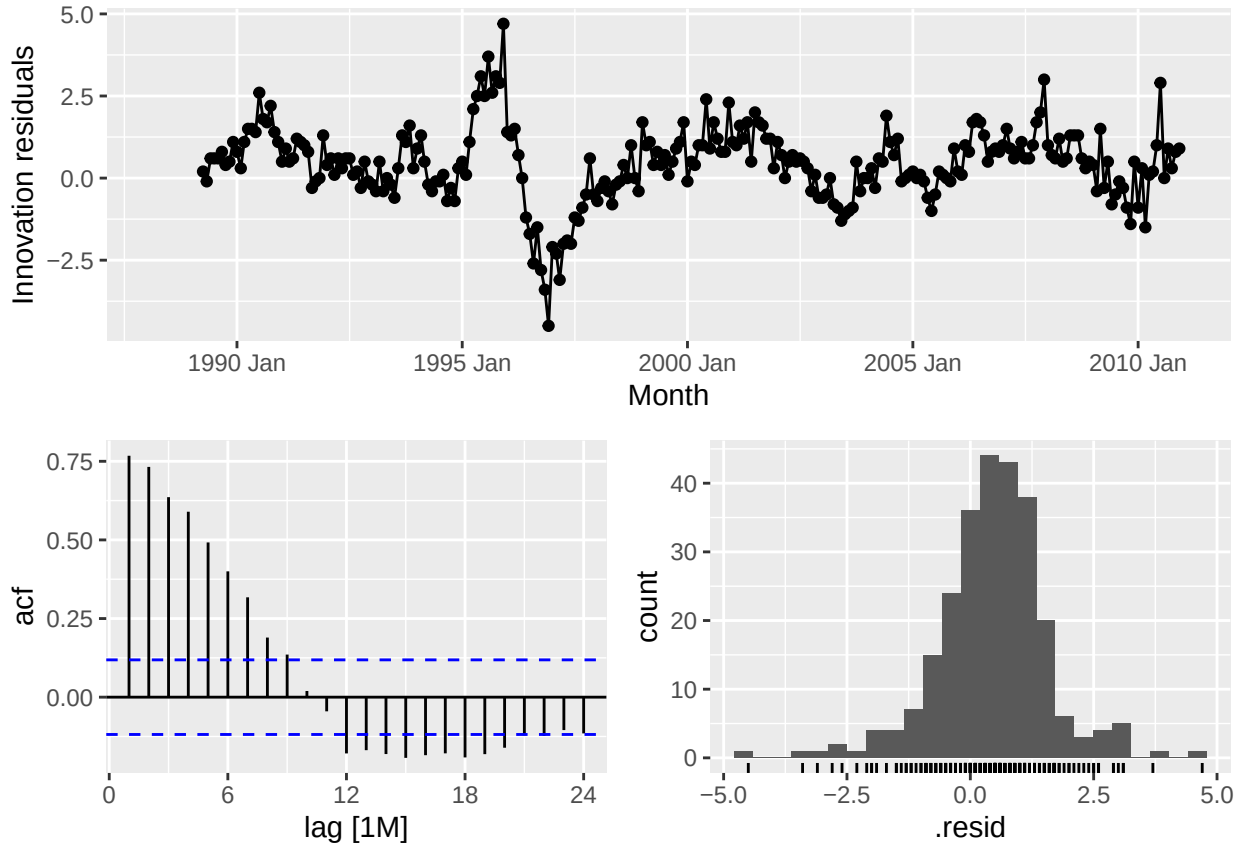



```
# Fit a seasonal naïve model using SNAIVE() applied to your training data (myseries_train).
fit <- myseries_train |>model(SNAIVE(Turnover))
fit |> gg_tsresiduals()
```

```
## 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()‘).
```

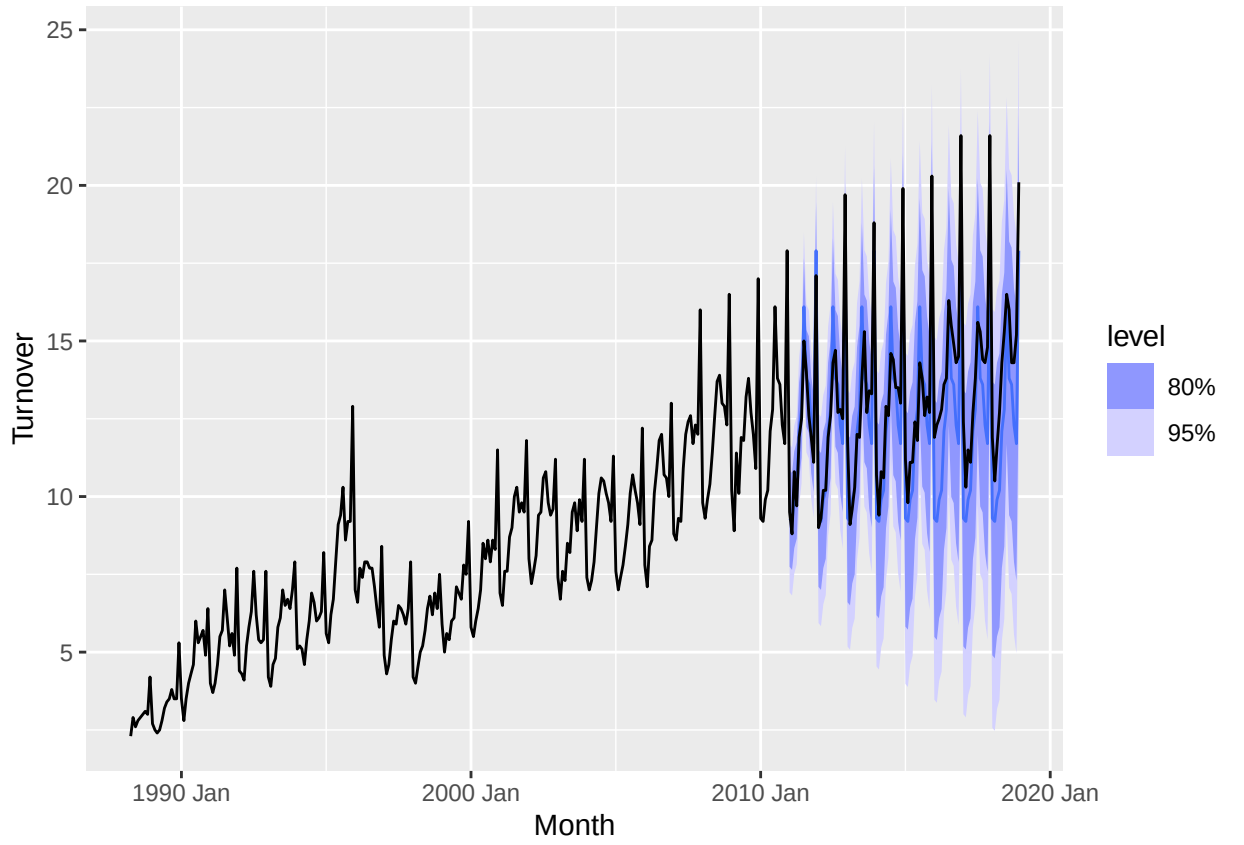
```
## Warning: Removed 12 rows containing non-finite outside the scale range
## (‘stat_bin()‘).
```




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

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

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




```
#Acuuracy

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
```

The analysis of the **seasonal naïve model (`SNAIVE()`)** applied to the retail time series data provides valuable insights into its effectiveness as a forecasting benchmark. The training dataset was correctly split, with observations before **2011** used to build the model, ensuring that historical seasonal patterns could inform future predictions. The residual analysis reveals that the residuals are **not normally distributed**, as the histogram indicates a right skew, and the **ACF plot** suggests that residuals are **not uncorrelated**, meaning some underlying patterns remain unexplained. This suggests that the seasonal naïve method does not fully capture trends or structural changes in the data. The **forecast plot** shows that predictions follow the seasonal pattern, with confidence intervals indicating the uncertainty of future values. However,

accuracy evaluations highlight that while the model performs well on the training data, its **test performance deteriorates**, as shown by increased **RMSE and MAPE**, indicating that the seasonal naïve model struggles to generalize beyond the training period. Additionally, the high **ACF1 value (~0.69) in the test data** suggests residual autocorrelation, meaning forecast errors are correlated across time, which reduces reliability. The sensitivity analysis confirms that the accuracy of the forecasts depends on the amount of training data, with **longer historical series improving performance**. Overall, while the seasonal naïve model serves as a reasonable benchmark for **short-term forecasting** in a highly seasonal retail series, its limitations in capturing trends and autocorrelation suggest that **more sophisticated models like ETS or ARIMA** may provide improved predictive performance.