Time Series Forecasting Exercises

Souleymane Doumbia

2024-09-29

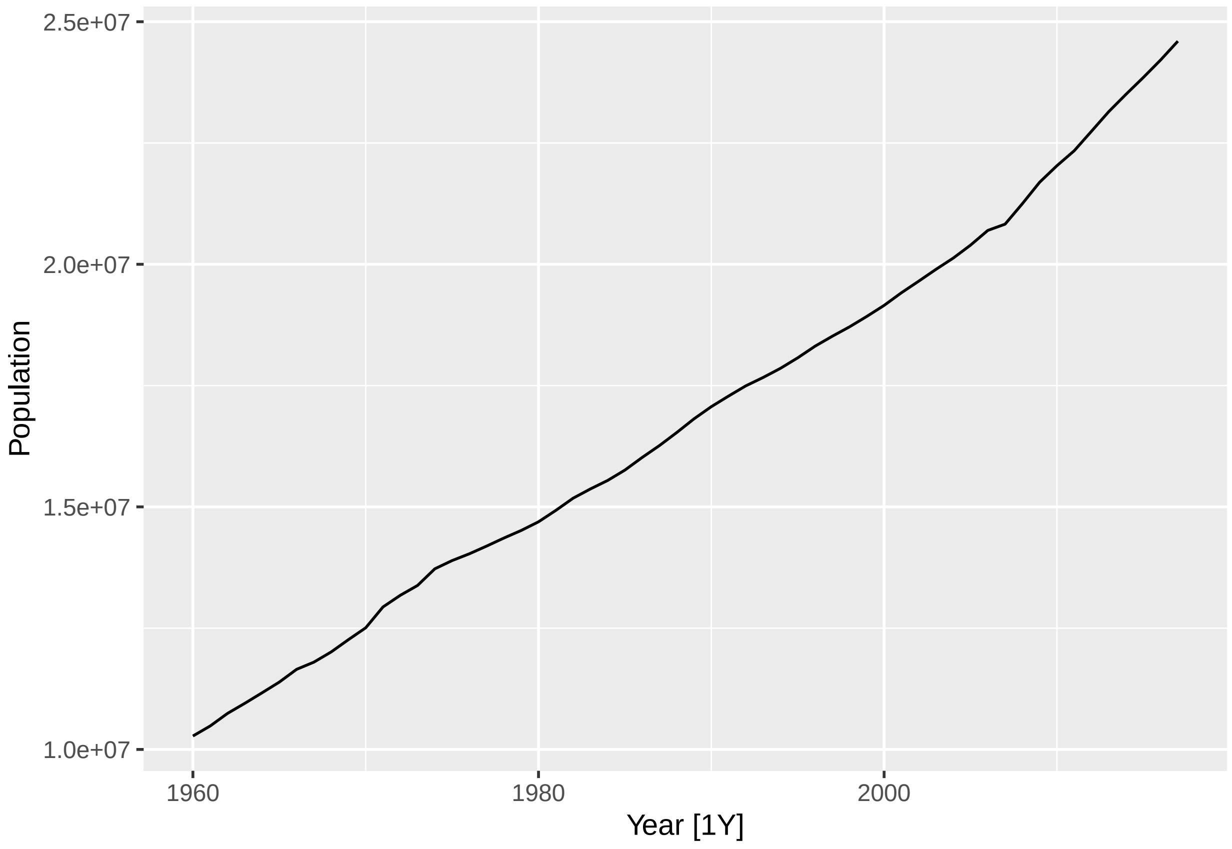
Contents

Exercise 1.: Forecasting Australian Population, Bricks, NSW Lambs, Household Wealth, and Australian Takeaway Food Turnover

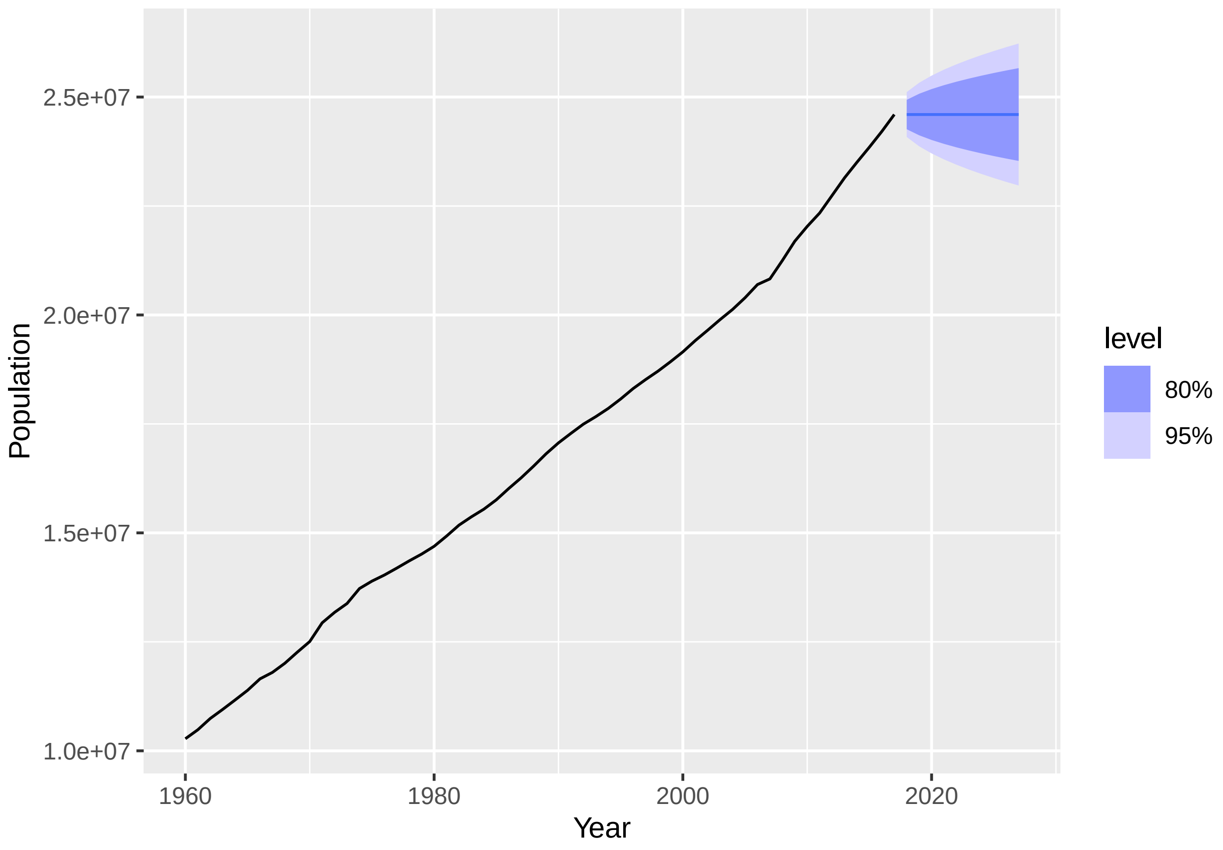
Australian Population Forecast (global_economy)

```r
# Loading global_economy dataset
global_economy <- global_economy
```

```r
aus_pop <- global_economy %>% filter(Country == "Australia") %>% select(Year, Population)

# Plotting Australian population time series
autoplot(aus_pop, Population)
```

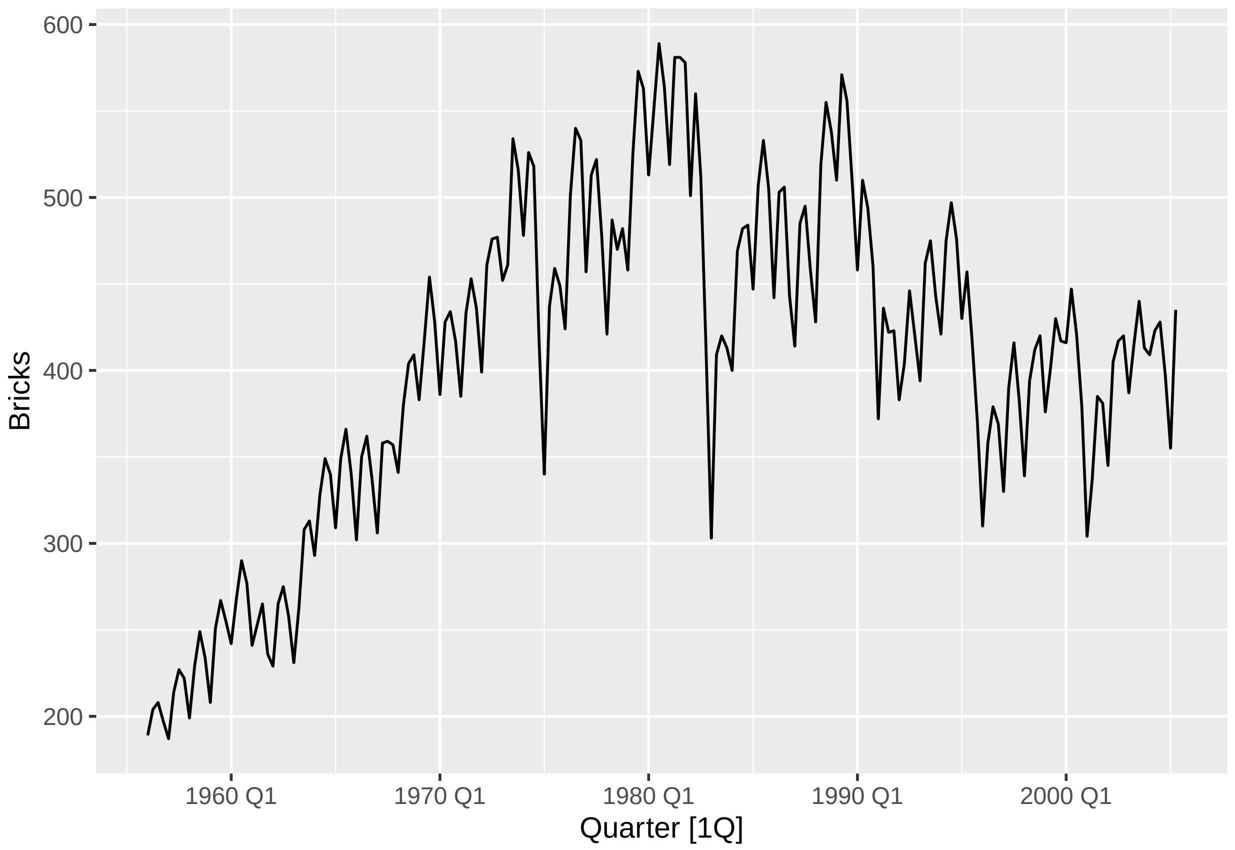


```r
# Applying appropriate forecasting method: NAIVE
fit_naive_pop <- aus_pop %>% model(NAIVE(Population))
fit_naive_pop %>% forecast(h = "10 years") %>% autoplot(aus_pop)
```

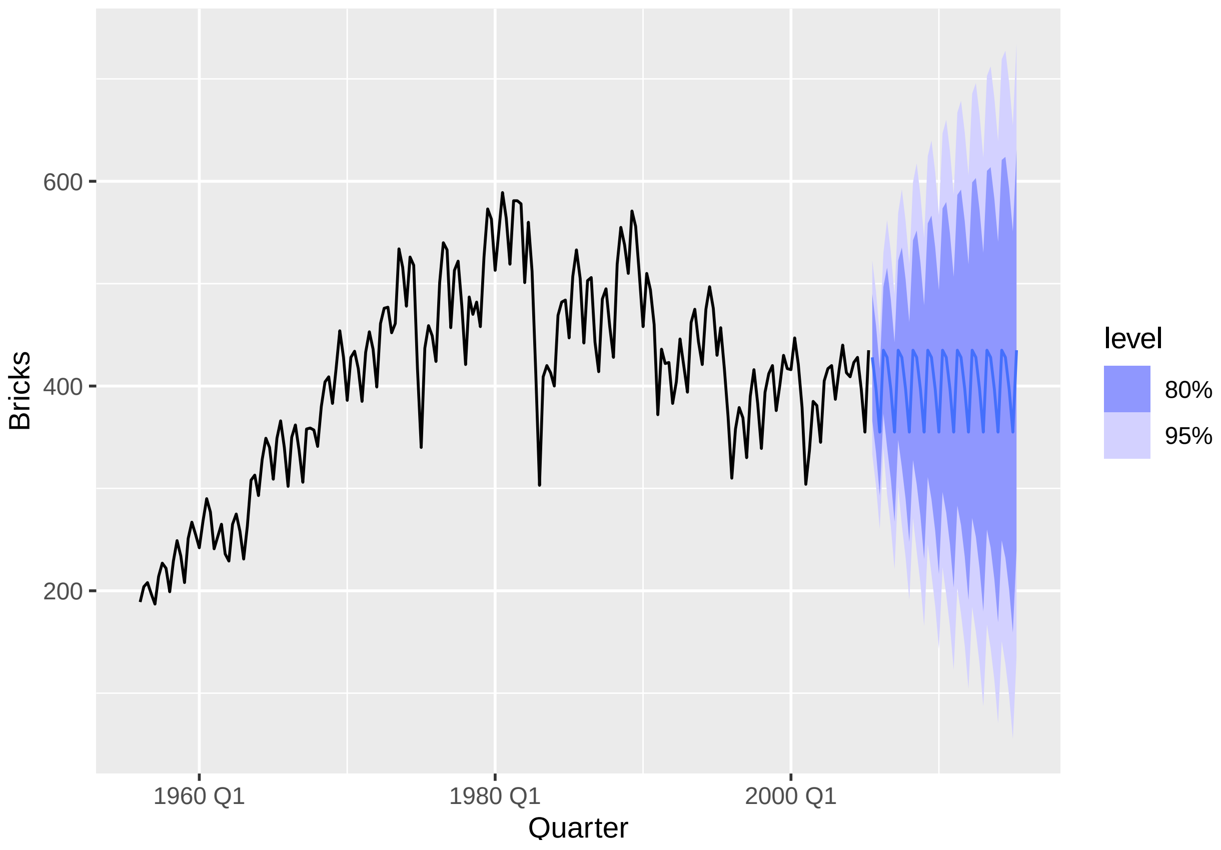


Bricks Forecast (aus_production)

```r
# Removing missing values from Bricks (9% of data)
bricks_clean <- aus_production %>% filter(!is.na(Bricks))

# Plotting Bricks production time series without missing values
autoplot(bricks_clean, Bricks)
```

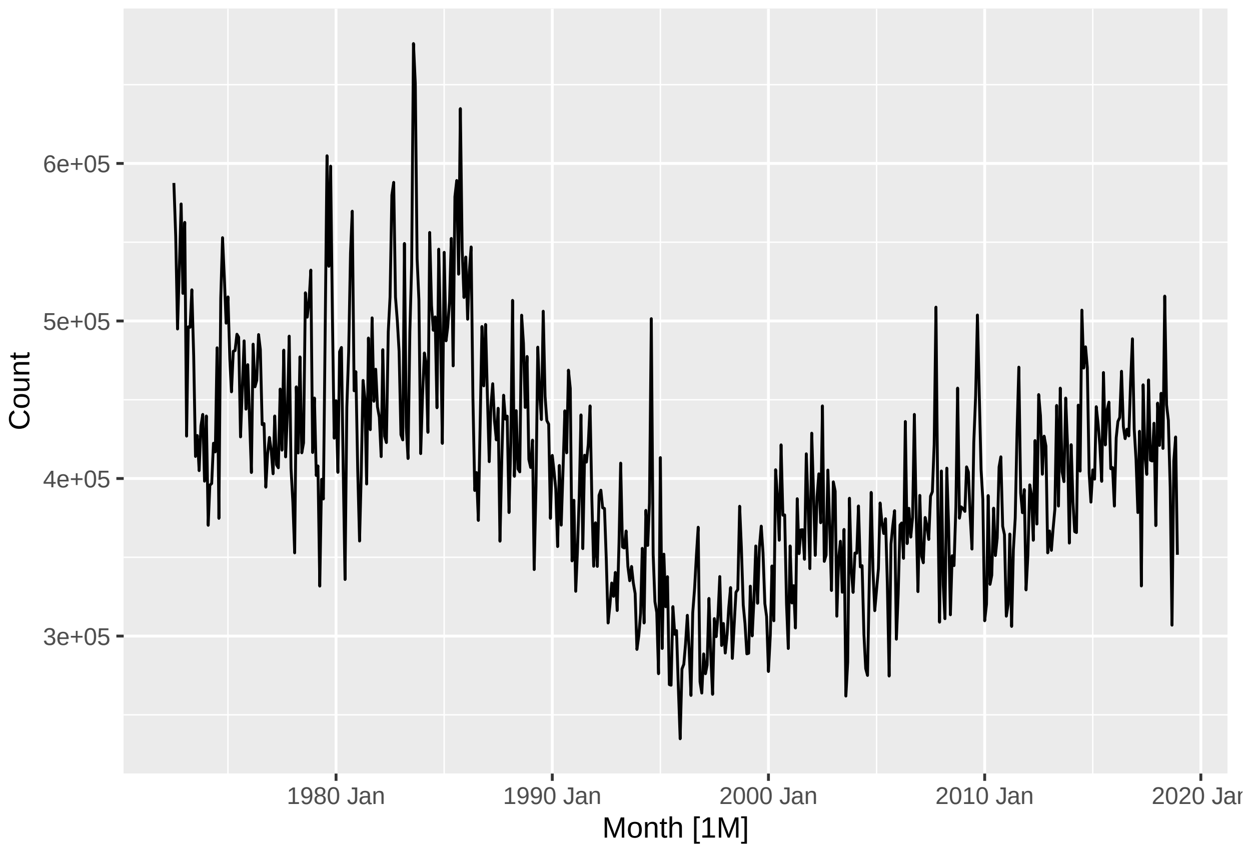


```r
# Applying SNAIVE model after removing missing values
fit_snaive_bricks <- bricks_clean %>% model(SNAIVE(Bricks))
fit_snaive_bricks %>% forecast(h = "10 years") %>% autoplot(bricks_clean)
```

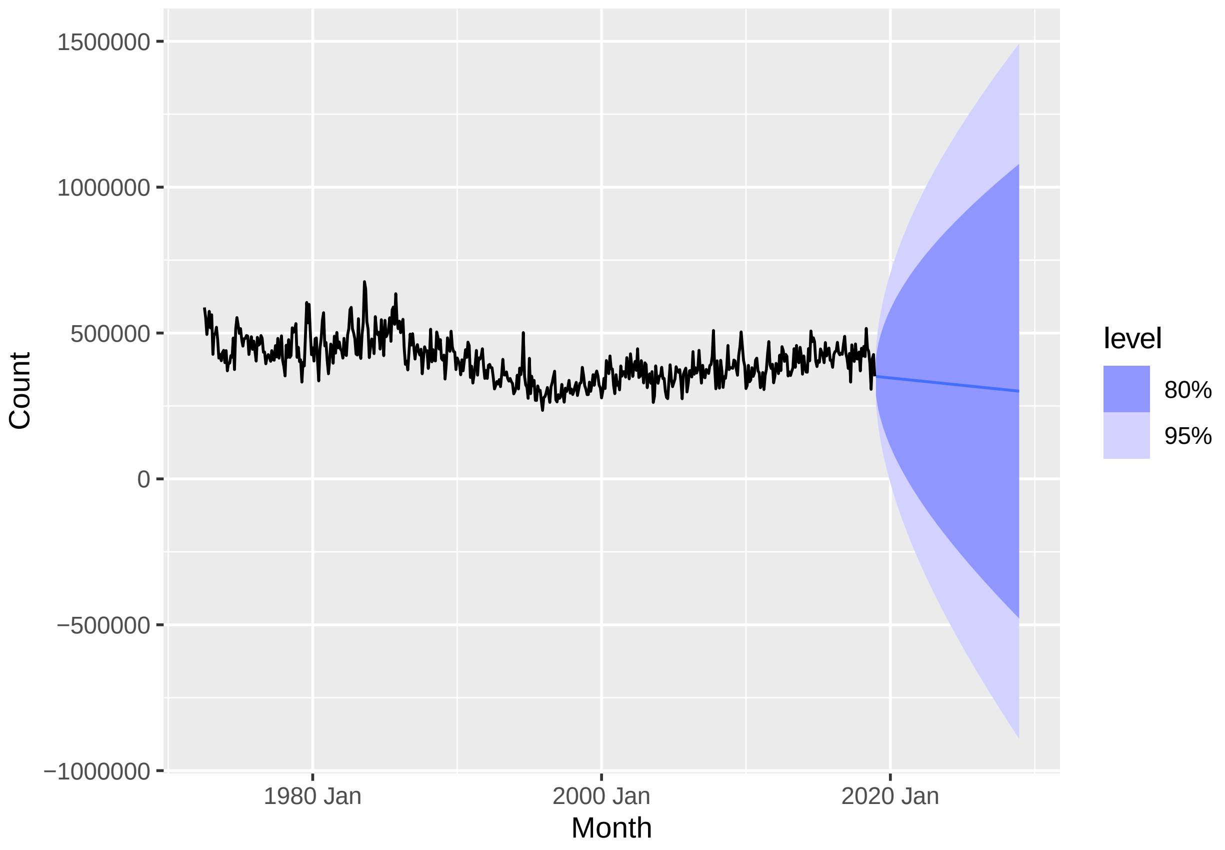


NSW Lambs Forecast (aus_livestock)

```
# Loading aus_livestock dataset
nsw_lambs <- aus_livestock %>% filter(State == "New South Wales", Animal == "Lambs")

# Plotting NSW lambs time series
autoplot(nsw_lambs, Count)
```

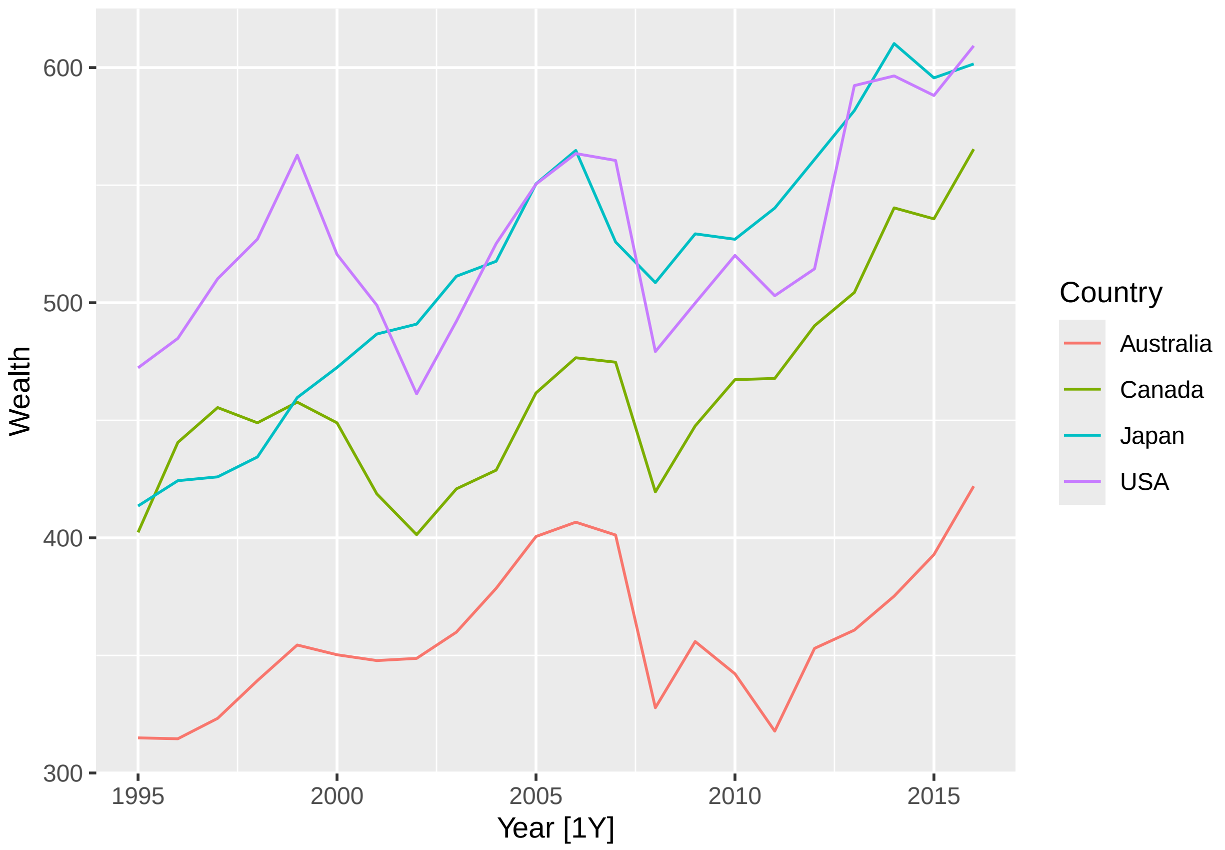


```
# Applying appropriate forecasting method: RW with drift
fit_rw_drift_lambs <- nsw_lambs %>% model(RW(Count ~ drift()))
fit_rw_drift_lambs %>% forecast(h = "10 years") %>% autoplot(nsw_lambs)
```

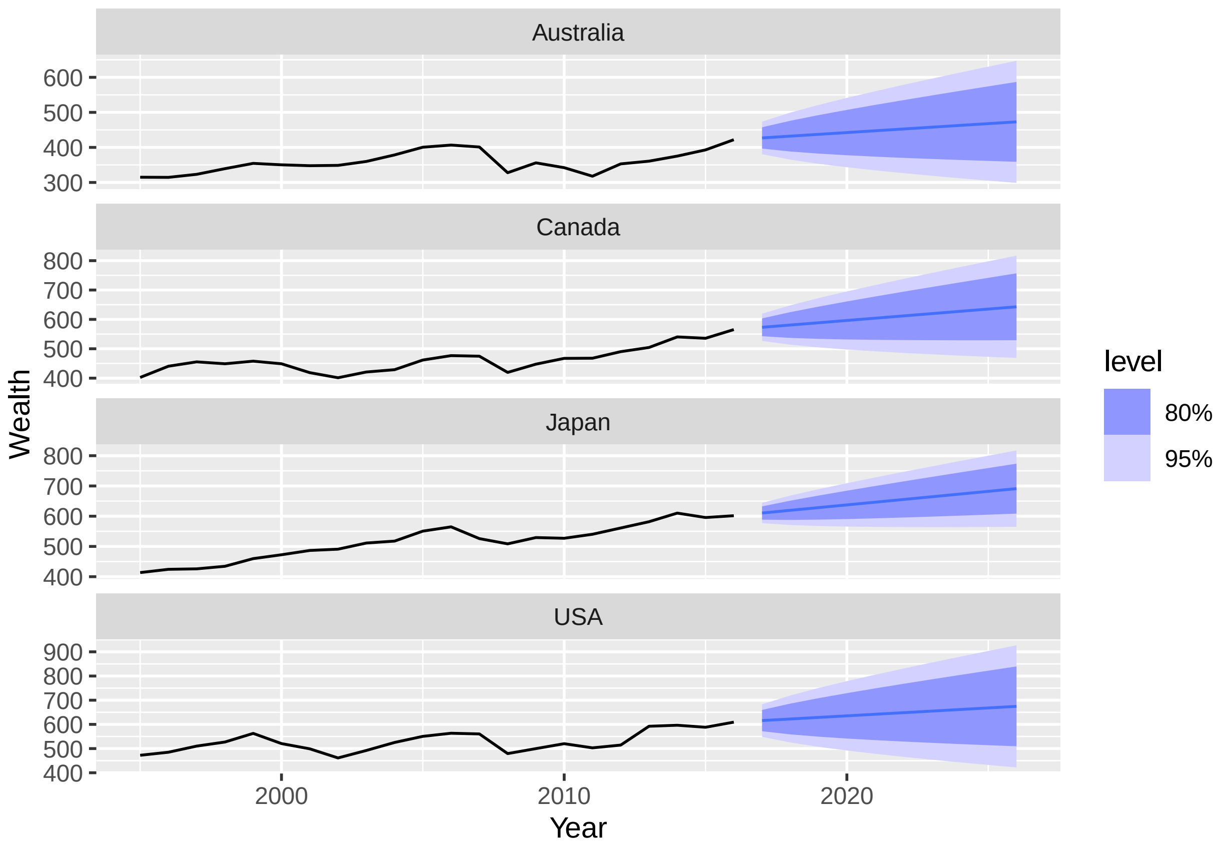


Household Wealth Forecast (hh_budget)

```r
# Loading hh_budget dataset
hh_wealth <- hh_budget %>% select(Year, Wealth)

# Plotting household wealth time series
autoplot(hh_wealth, Wealth)
```

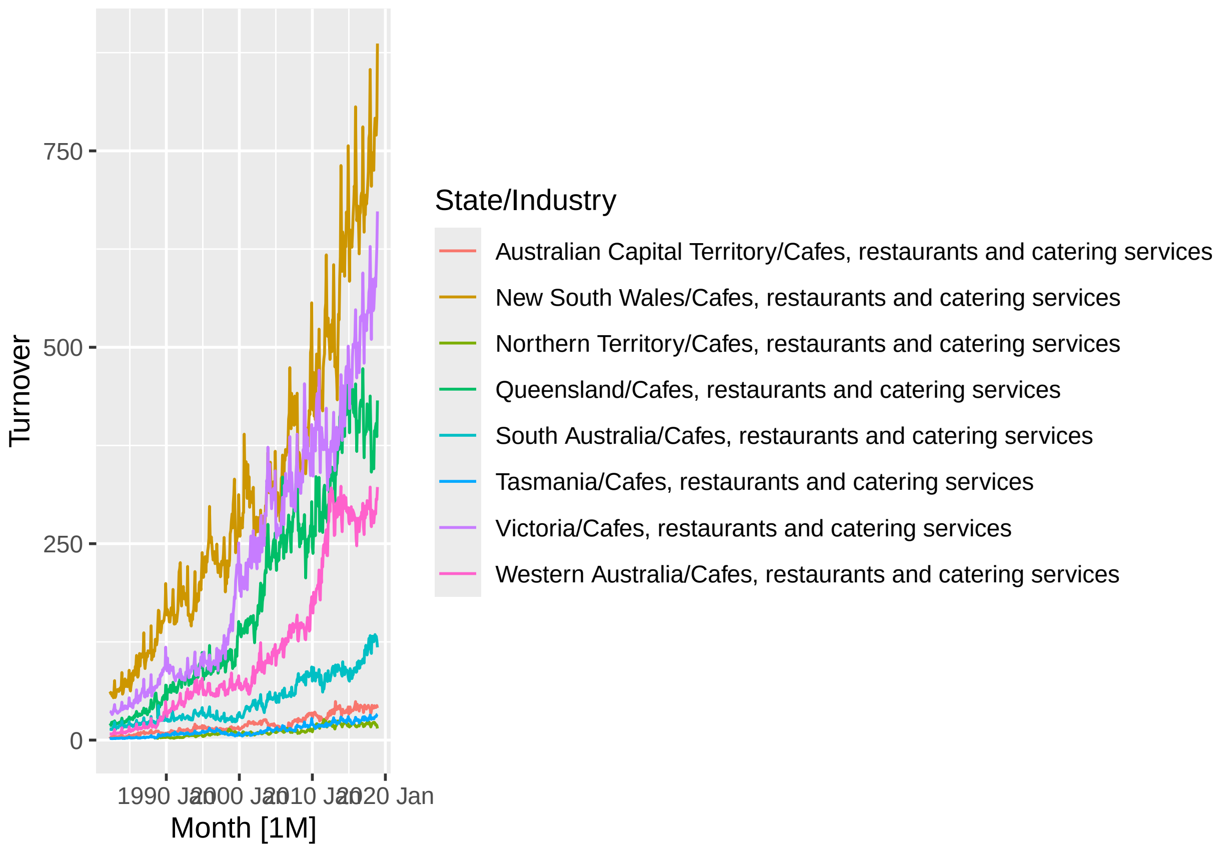


```
# Applying appropriate forecasting method: RW with drift
fit_rw_drift_wealth <- hh_wealth %>% model(RW(Wealth ~ drift()))
fit_rw_drift_wealth %>% forecast(h = "10 years") %>% autoplot(hh_wealth)
```



Australian Takeaway Food Turnover Forecast (aus_retail)

```r
# Loading aus_retail dataset
aus_takeaway <- aus_retail %>% filter(Industry == "Cafes, restaurants and catering services")

# Plotting Australian takeaway food turnover time series
autoplot(aus_takeaway, Turnover)
```

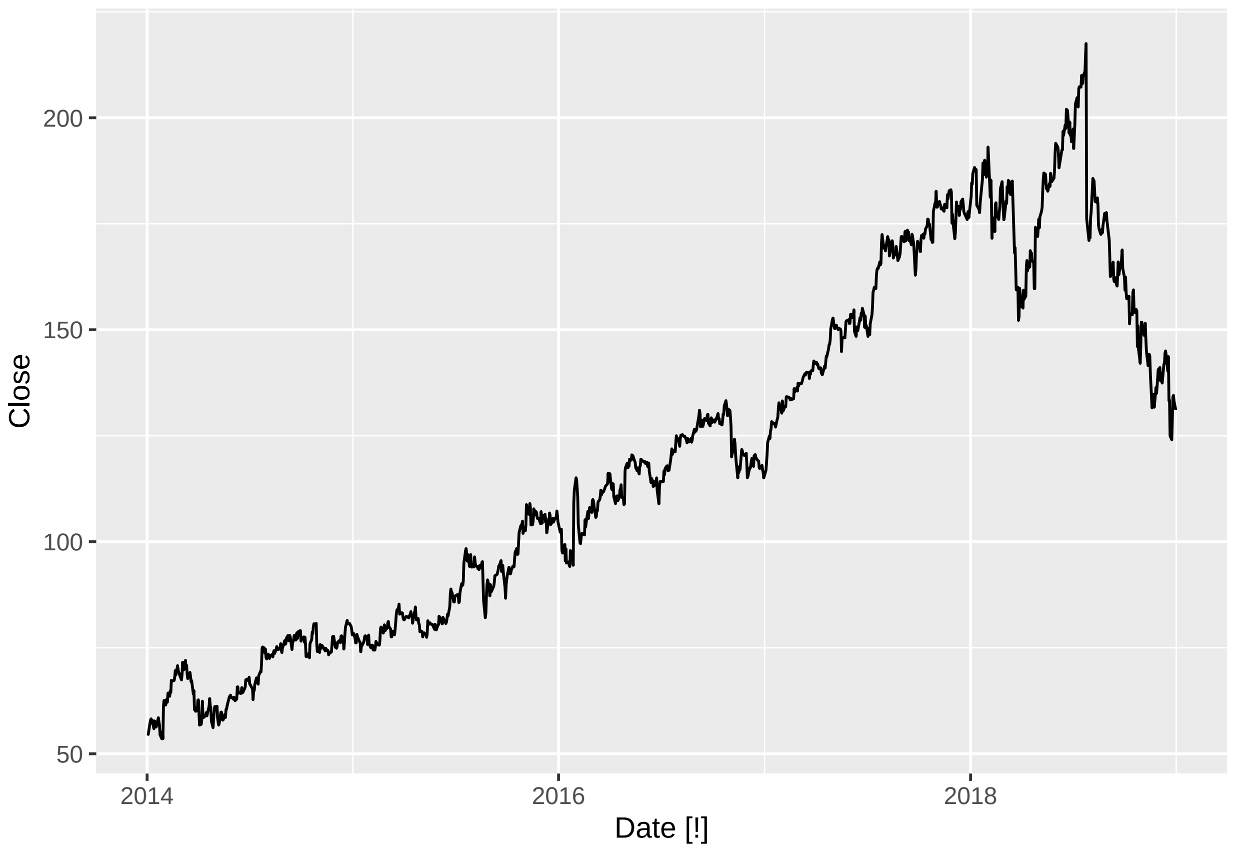


```
# Applying appropriate forecasting method: SNAIVE
fit_snaive_takeaway <- aus_takeaway %>% model(SNAIVE(Turnover))
fit_snaive_takeaway %>% forecast(h = "10 years") %>% autoplot(aus_takeaway)
```

Exercise 2.: Facebook Stock Price (gafa_stock)

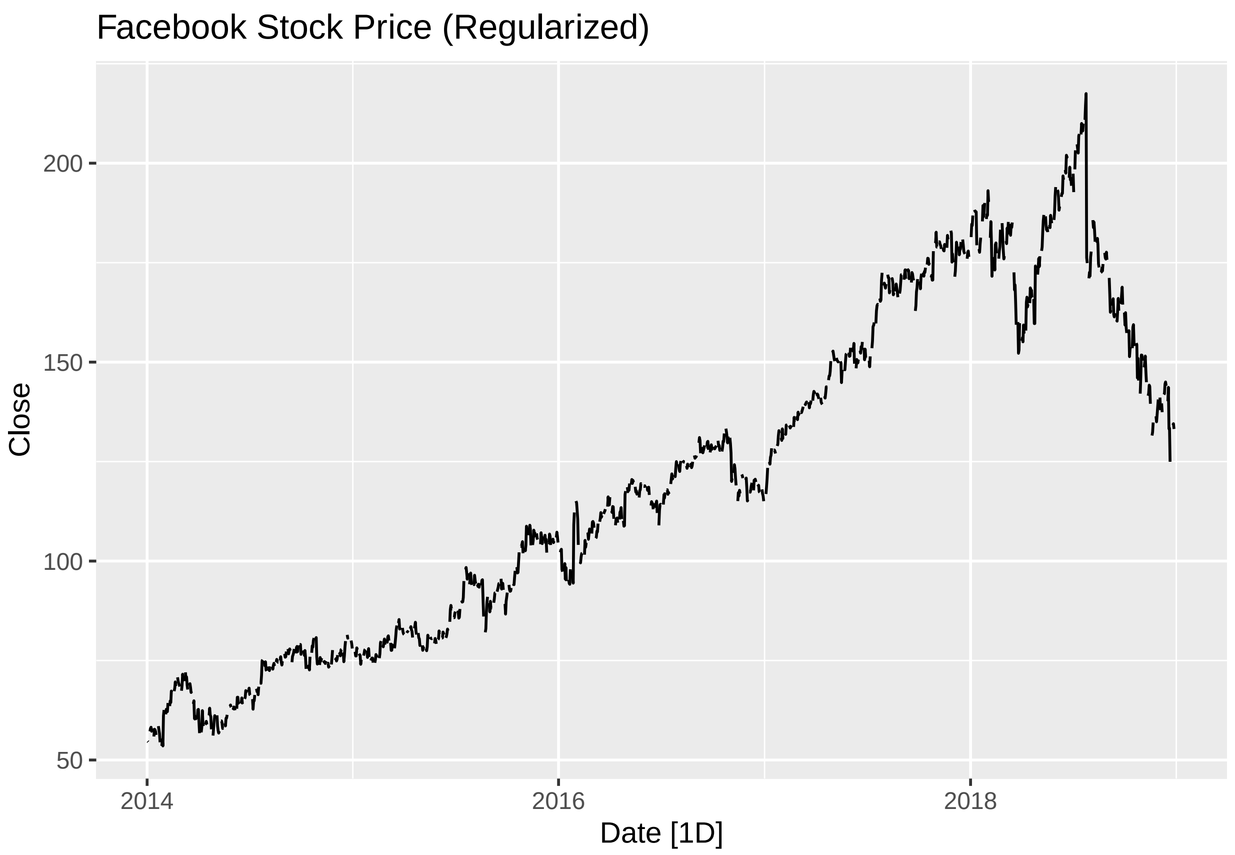
Time Plot of Facebook Stock Price

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

# Plotting time series of Facebook stock price
autoplot(fb_stock, Close)
```

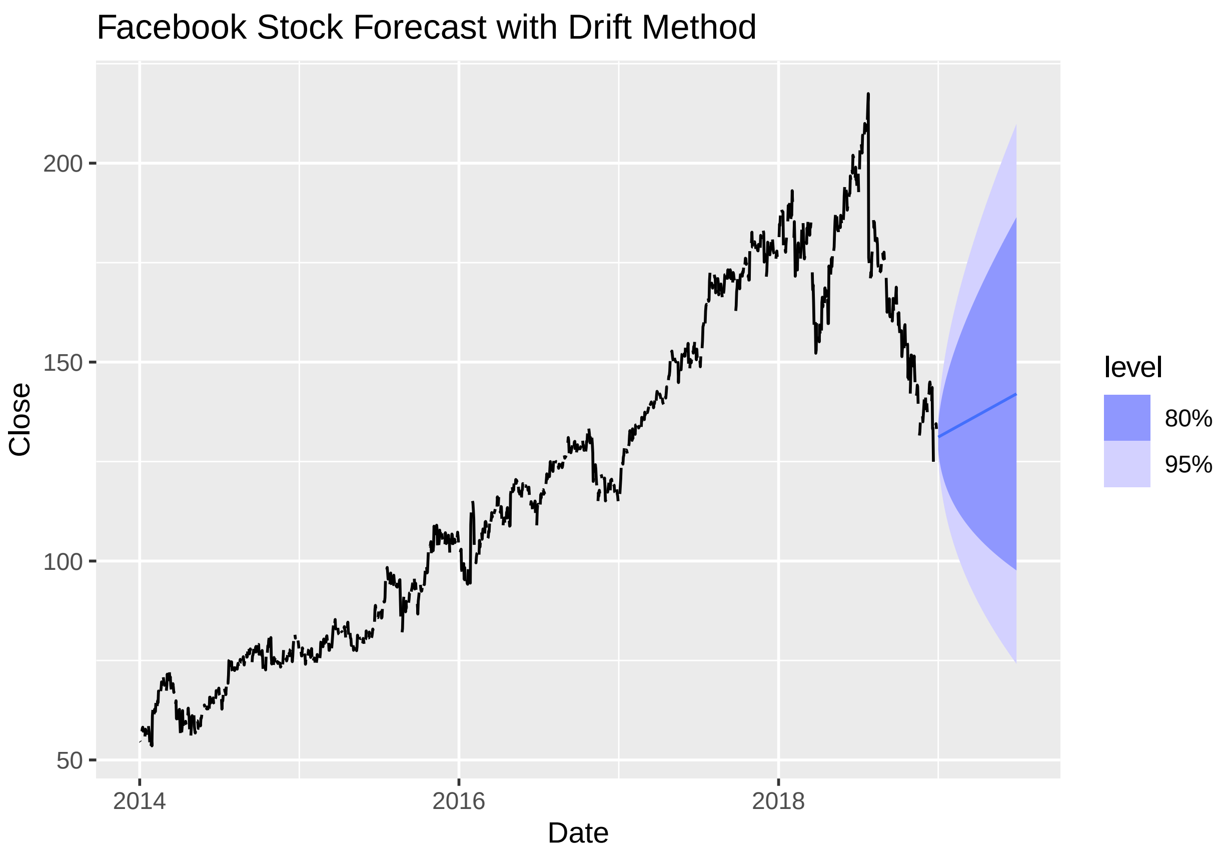


Forecast Using Drift Method

```r
# Making the Facebook stock data regular by filling in missing dates and filling gaps
fb_stock_regular <- gafa_stock %>%
  filter(Symbol == "FB") %>%
  update_tsibble(regular = TRUE) %>%
  fill_gaps(Close = NA)

# Plotting the regular time series with gaps filled
autoplot(fb_stock_regular, Close) +
  ggtitle("Facebook Stock Price (Regularized)")
```

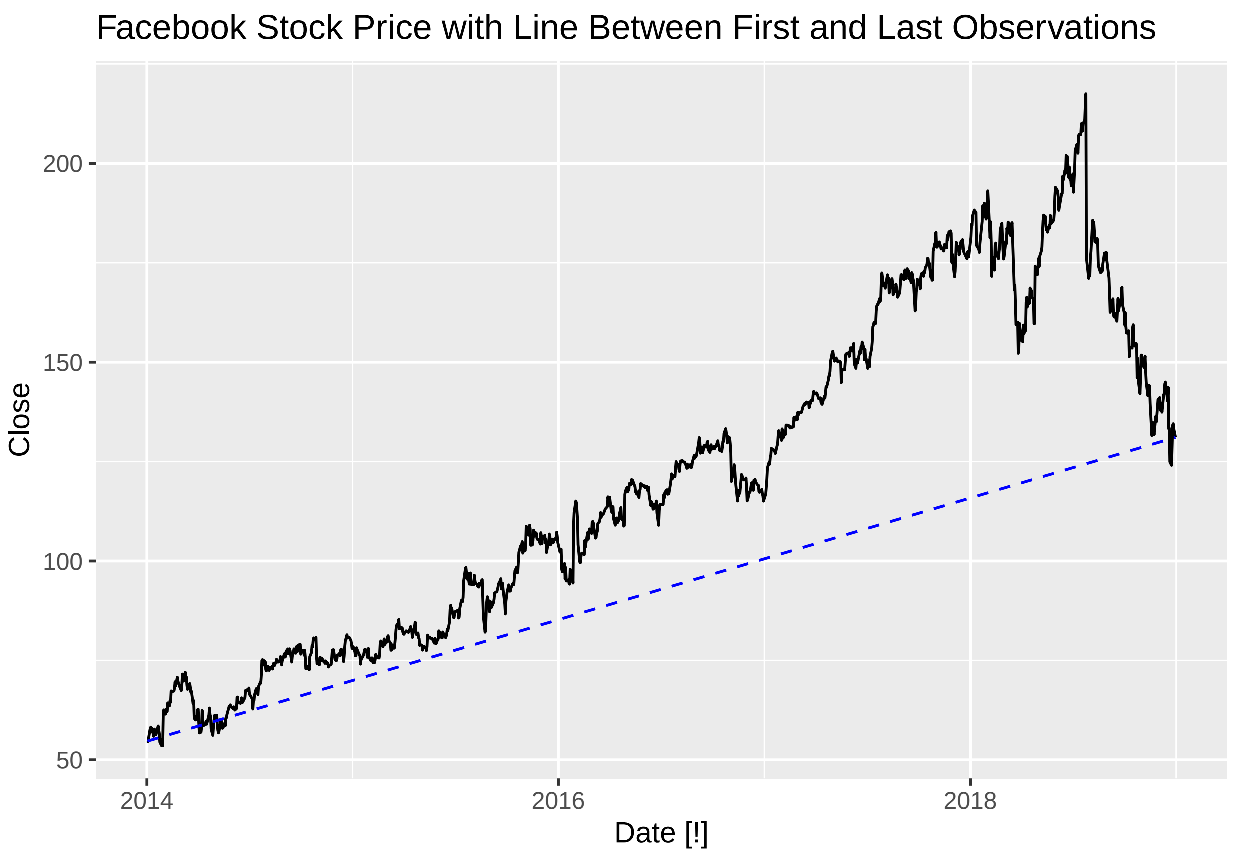


```r
# Applying drift method on regularized data
fit_drift_fb <- fb_stock_regular %>%
  model(RW(Close ~ drift()))

# Forecasting and plotting
fit_drift_fb %>% forecast(h = "180 days") %>%
  autoplot(fb_stock_regular) +
  ggtitle("Facebook Stock Forecast with Drift Method")
```



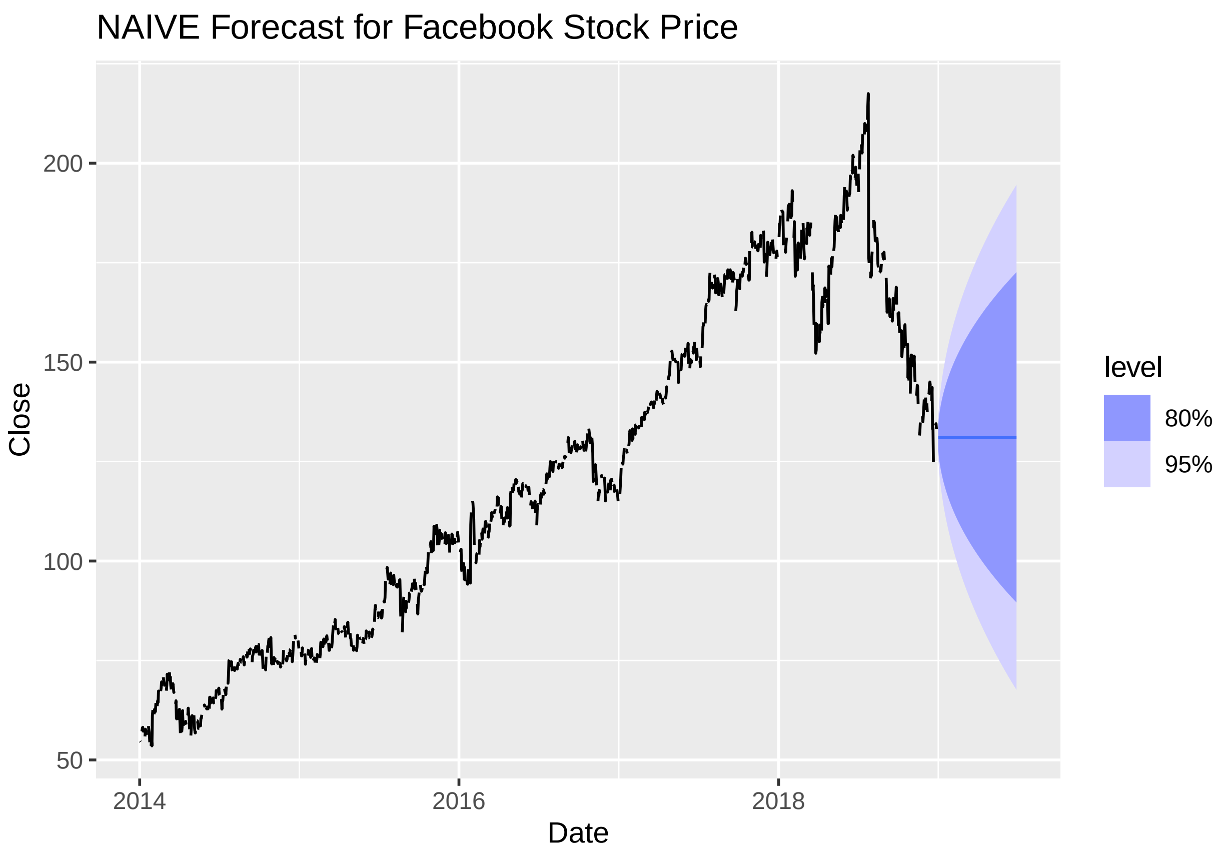
Identical Forecast to Extending Line Between First and Last Observations

```r
# Getting the first and last observations
first_last_line <- fb_stock %>%
  filter(Symbol == "FB") %>%
  reframe(x = c(min(Date), max(Date)), y = c(Close[1], Close[n()]))

# Plotting the time series and extend the line between the first and last observations
autoplot(fb_stock, Close) +
  geom_line(data = first_last_line, aes(x = x, y = y), color = "blue", linetype = "dashed") +
  ggtitle("Facebook Stock Price with Line Between First and Last Observations")
```

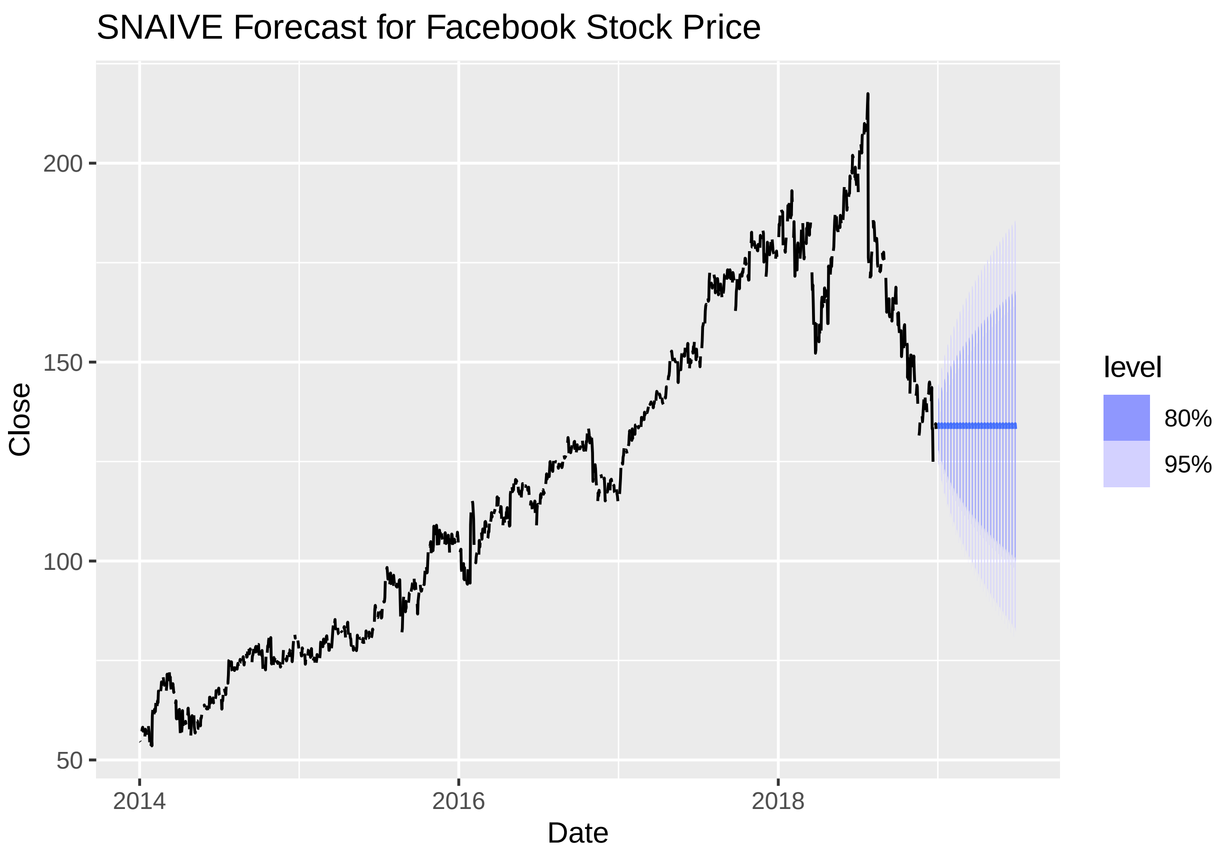


Other Benchmark Functions

```r
# Applying NAIVE model to fb_stock_regular
fit_naive_fb <- fb_stock_regular %>%
  model(NAIVE(Close))

# Applying SNAIVE model to fb_stock_regular
fit_snaive_fb <- fb_stock_regular %>%
  model(SNAIVE(Close))

# Forecasting and plotting for both models
# Avoid repetitive plotting and suppress missing values warning

# Plotting NAIVE forecast
autoplot(fit_naive_fb %>% forecast(h = "180 days"), fb_stock_regular, na.rm = TRUE) +
  ggtitle("NAIVE Forecast for Facebook Stock Price")
```



```r
# Plotting SNAIVE forecast
autoplot(fit_snaive_fb %>% forecast(h = "180 days"), fb_stock_regular, na.rm = TRUE) +
  ggtitle("SNAIVE Forecast for Facebook Stock Price")
```



Which Forecasting Method is Best and Why?

After applying the Drift, Naïve, and Seasonal Naïve forecasting methods to the Facebook stock price data, I compared the resulting forecasts:

- **Drift Method**: This method extends a line between the first and last observations, capturing the overall trend in the data.
- **Naïve Method**: Assumes that the future values will be the same as the last observed value, resulting in a horizontal forecast line.
- **Seasonal Naïve Method**: Repeats the last observed seasonal cycle. As we know, this method is suitable for data with strong seasonal patterns.

Evaluation:

- The **Drift Method** produces forecasts that continue the upward (or downward) trend observed in the historical stock prices, aligning well with the data's trajectory.
- The **Naïve Method** provides a flat forecast, which doesn't capture any trends or movements in the stock price.
- The **Seasonal Naïve Method** is not appropriate here because stock prices typically do not exhibit strong seasonal patterns.

Conclusion:

The **Drift Method** is the best choice for forecasting the Facebook stock price because:

1. **Trend Representation**: It effectively captures and extends the underlying trend in the stock price, which is crucial for financial time series data.

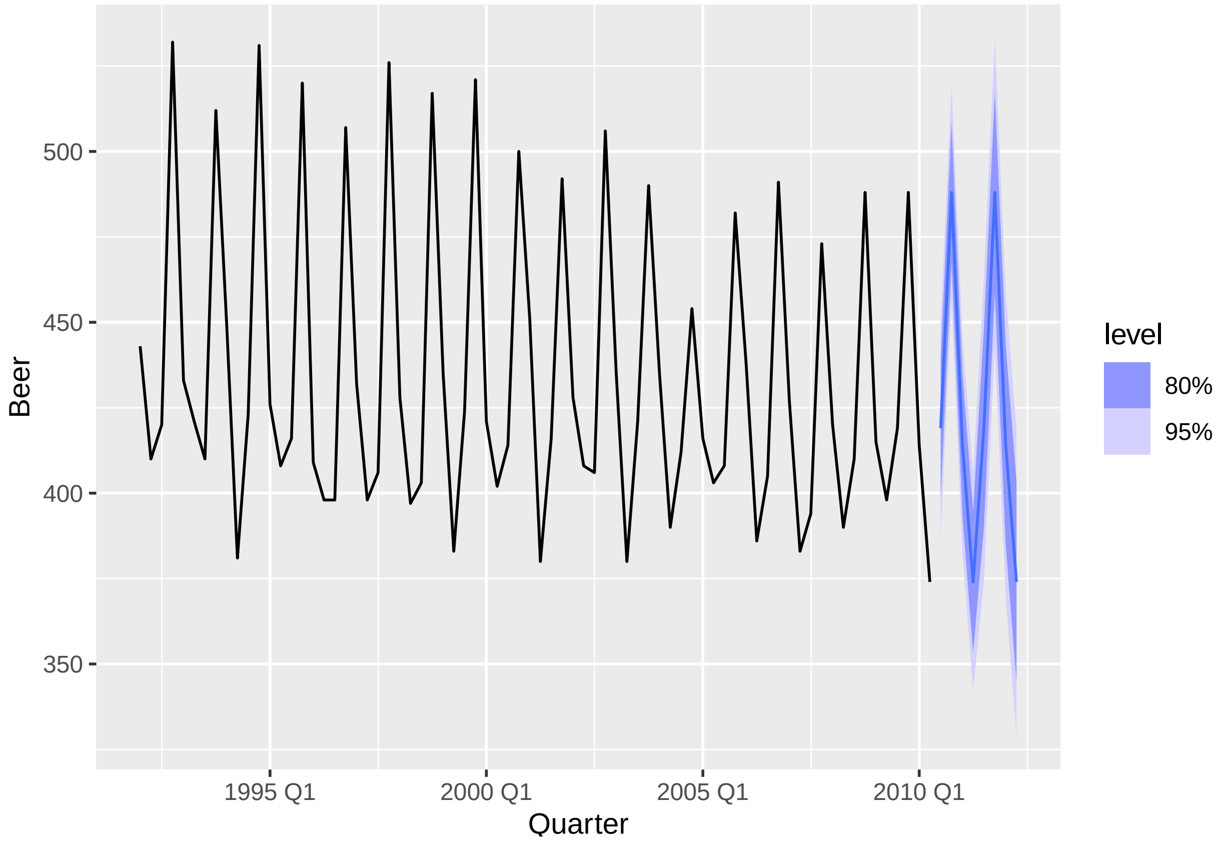
2. **Realistic Forecasts**: By accounting for the observed trend, the forecasts are more realistic compared to the flat predictions of the Naïve Method.
3. **Lack of Seasonality**: Since there is no significant seasonal pattern in the data, the Seasonal Naïve Method is less suitable.

Therefore, the Drift Method provides the most accurate and meaningful forecasts for the Facebook stock price in this context.

Exercise 3.: Seasonal Naïve Method for Australian Beer Production

Forecast Australian Beer Production

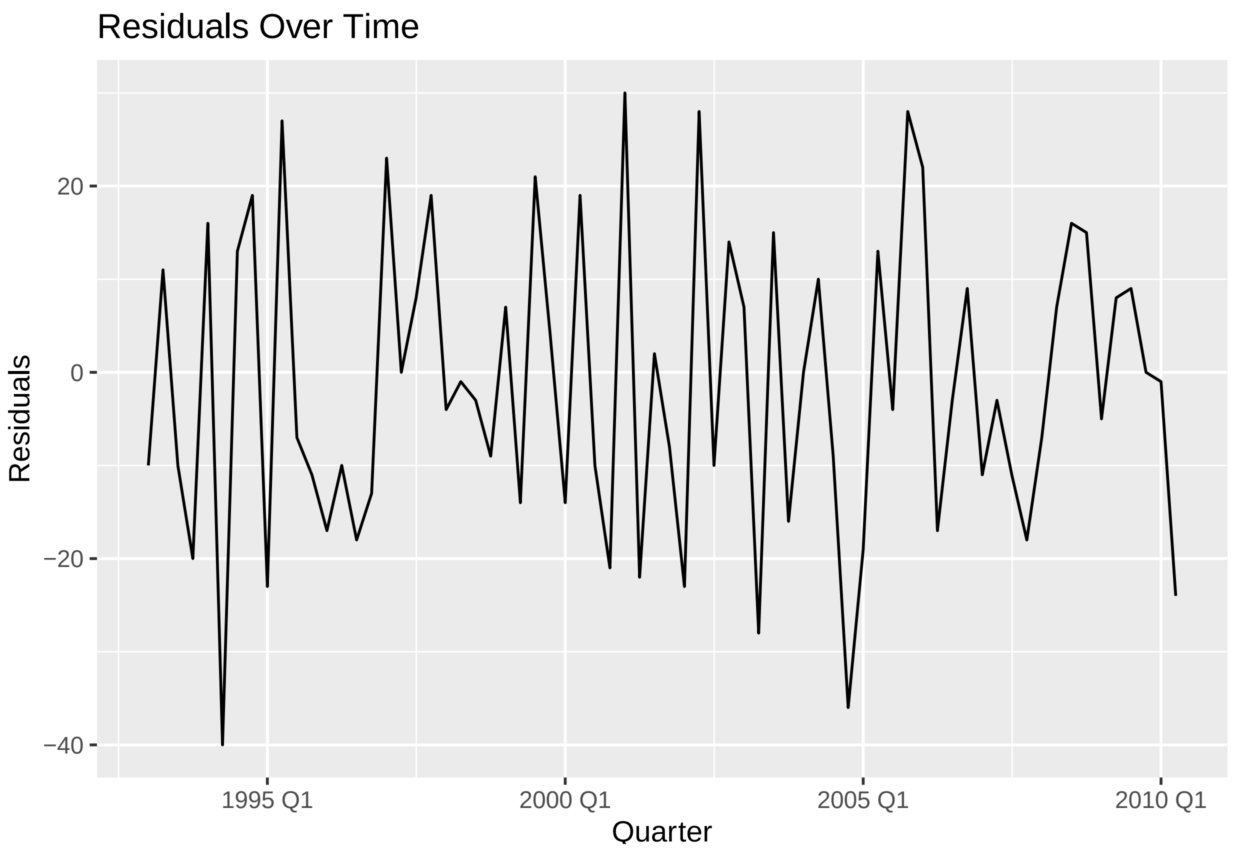
```r
# Filtering data for Australian Beer Production from 1992
recent_production <- aus_production %>% filter(year(Quarter) >= 1992)

# Applying SNAIVE model
fit_snaive_beer <- recent_production %>% model(SNAIVE(Beer))
fit_snaive_beer %>% forecast() %>% autoplot(recent_production)
```



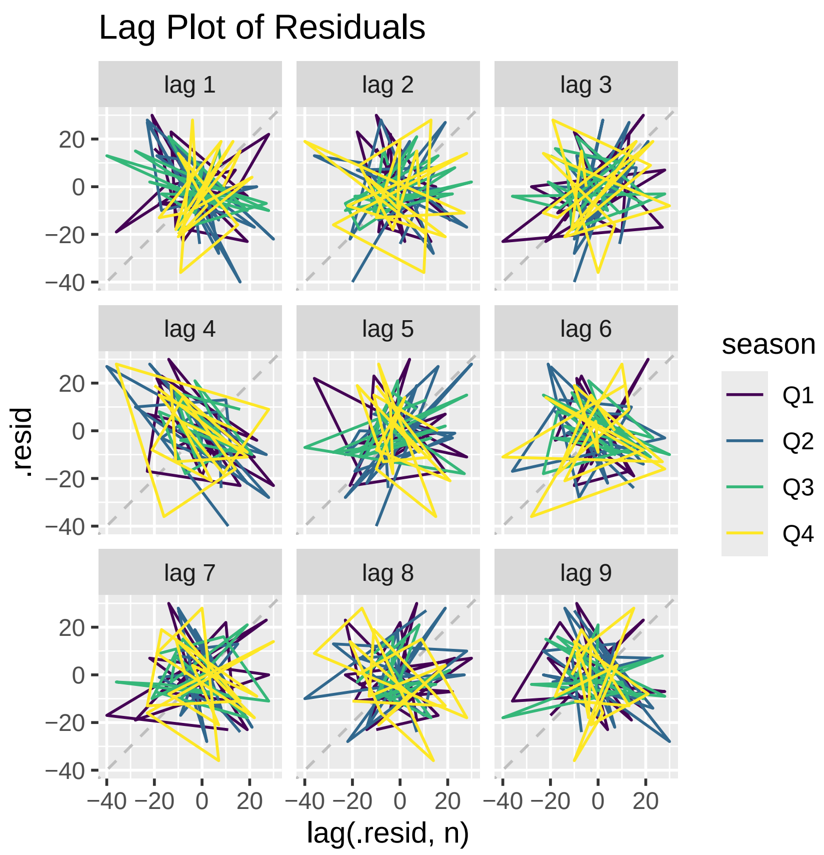
Residuals Check

```r
# Extracting residuals and remove NA values
residuals_beer <- augment(fit_snaive_beer) %>%
  filter(!is.na(.resid))

# 1. Plotting residuals over time
```

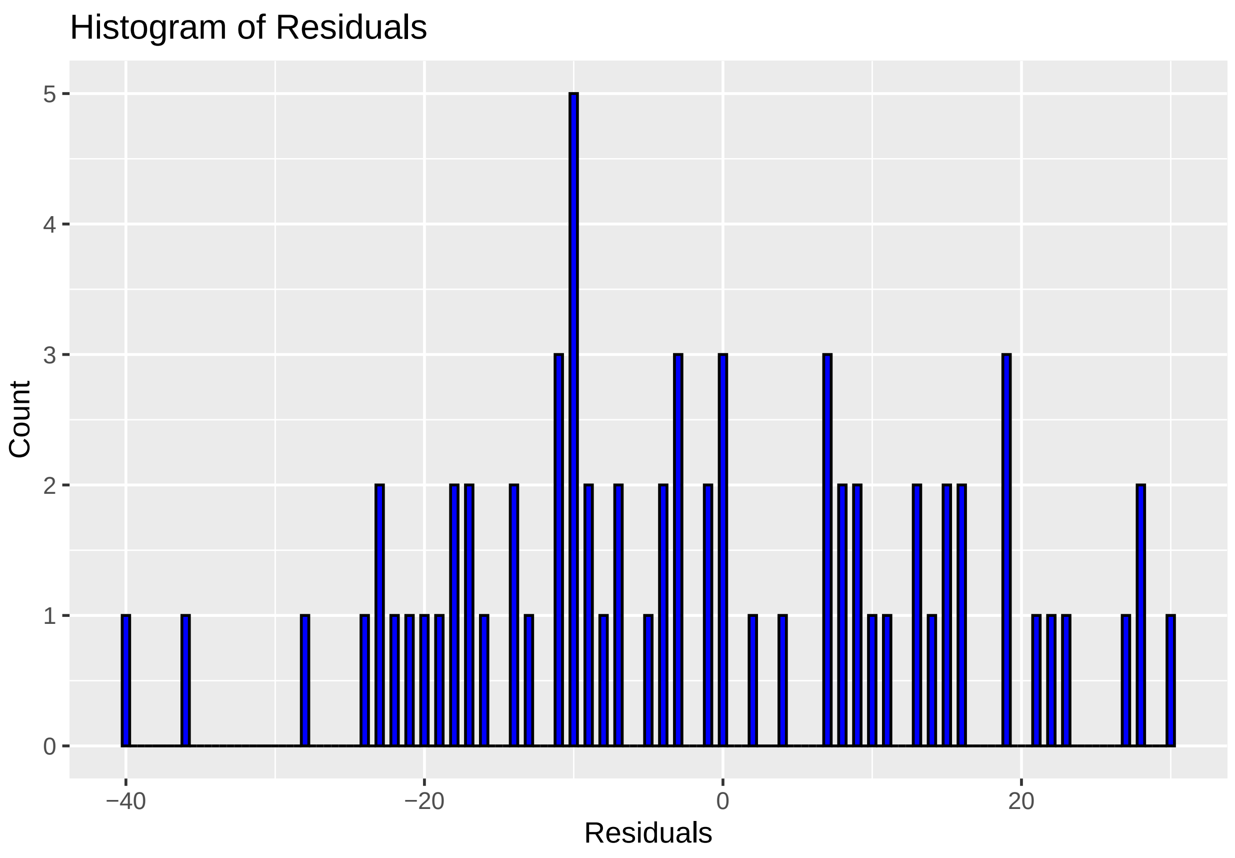
```r
autoplot(residuals_beer, .resid) +
  labs(title = "Residuals Over Time", x = "Quarter", y = "Residuals")
```



```r
# 2. Plotting residuals lag (autocorrelation)
gg_lag(residuals_beer, .resid) +
  labs(title = "Lag Plot of Residuals")
```



```r
# 3. Plotting residuals histogram
ggplot(residuals_beer, aes(x = .resid)) +
  geom_histogram(binwidth = 0.5, fill = "blue", color = "black") +
  labs(title = "Histogram of Residuals", x = "Residuals", y = "Count")
```



Conclusion on Residuals

After examining the residuals from the SNAIVE model applied to the Australian beer production data, we can conclude that the residuals do not fully resemble white noise:

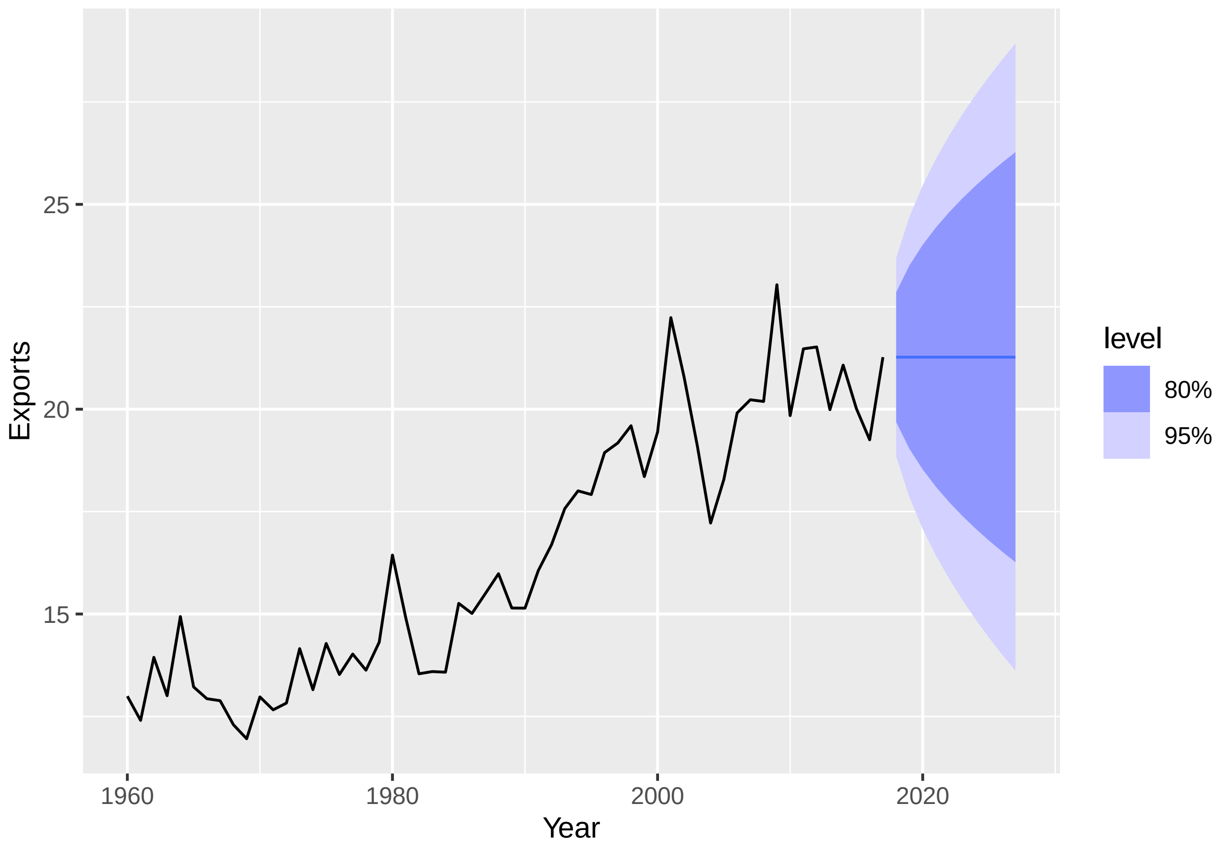
- **Residuals Over Time**: There are noticeable spikes and dips, indicating potential autocorrelation or patterns that are not fully random.
- **Lag Plot**: The presence of clustering and patterns in the lag plots suggests some degree of autocorrelation between residuals at different lags.
- **Histogram**: The residuals do not appear to be symmetrically distributed around zero, hinting at non-normality and systematic behavior.

These observations suggest that the model may not have completely captured the underlying structure of the data, and further refinement may be necessary.

Exercise 4.: Repeat Exercise for Australian Exports and Bricks

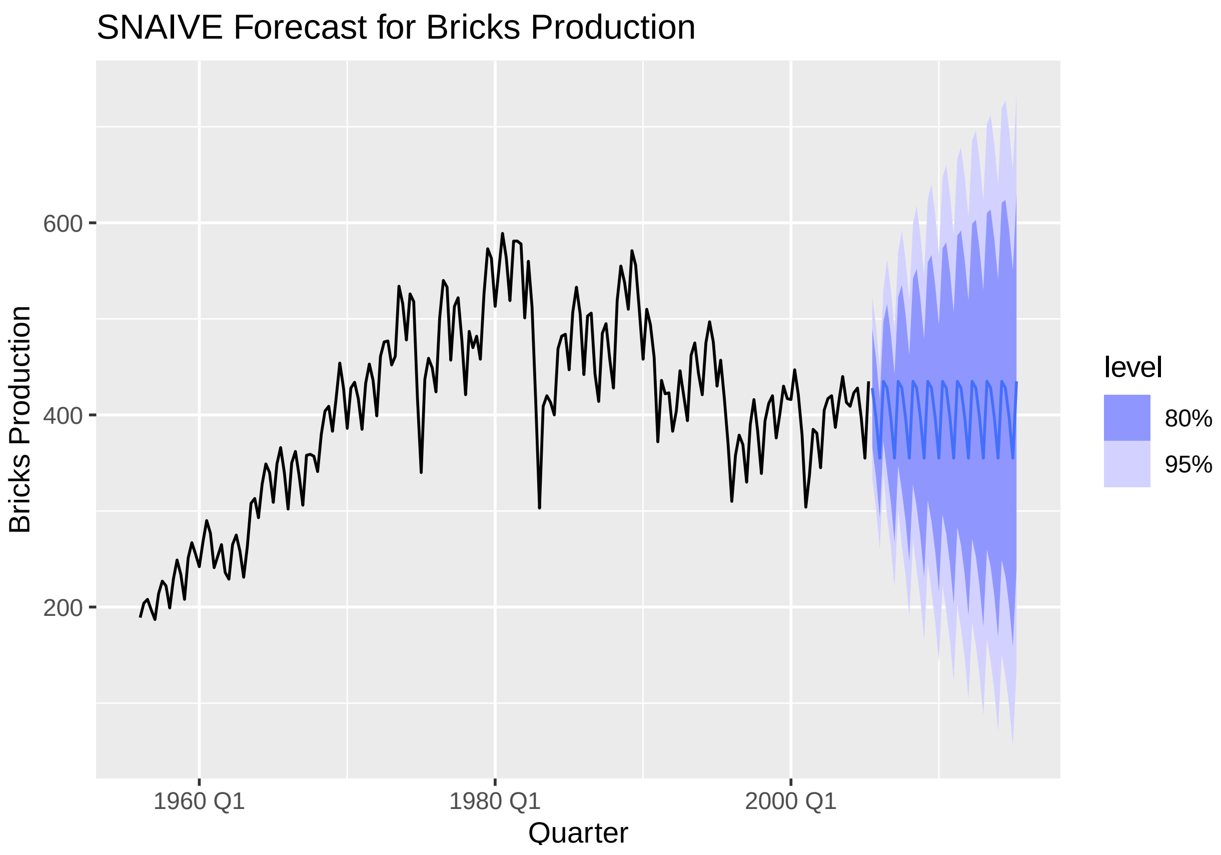
Australian Exports (global_economy)

```
# Filtering data for Australian exports
aus_exports <- global_economy %>% filter(Country == "Australia") %>% select(Year, Exports)

# Applying appropriate forecasting method: NAIVE
fit_naive_exports <- aus_exports %>% model(NAIVE(Exports))
fit_naive_exports %>% forecast(h = "10 years") %>% autoplot(aus_exports)
```



Bricks (aus_production)

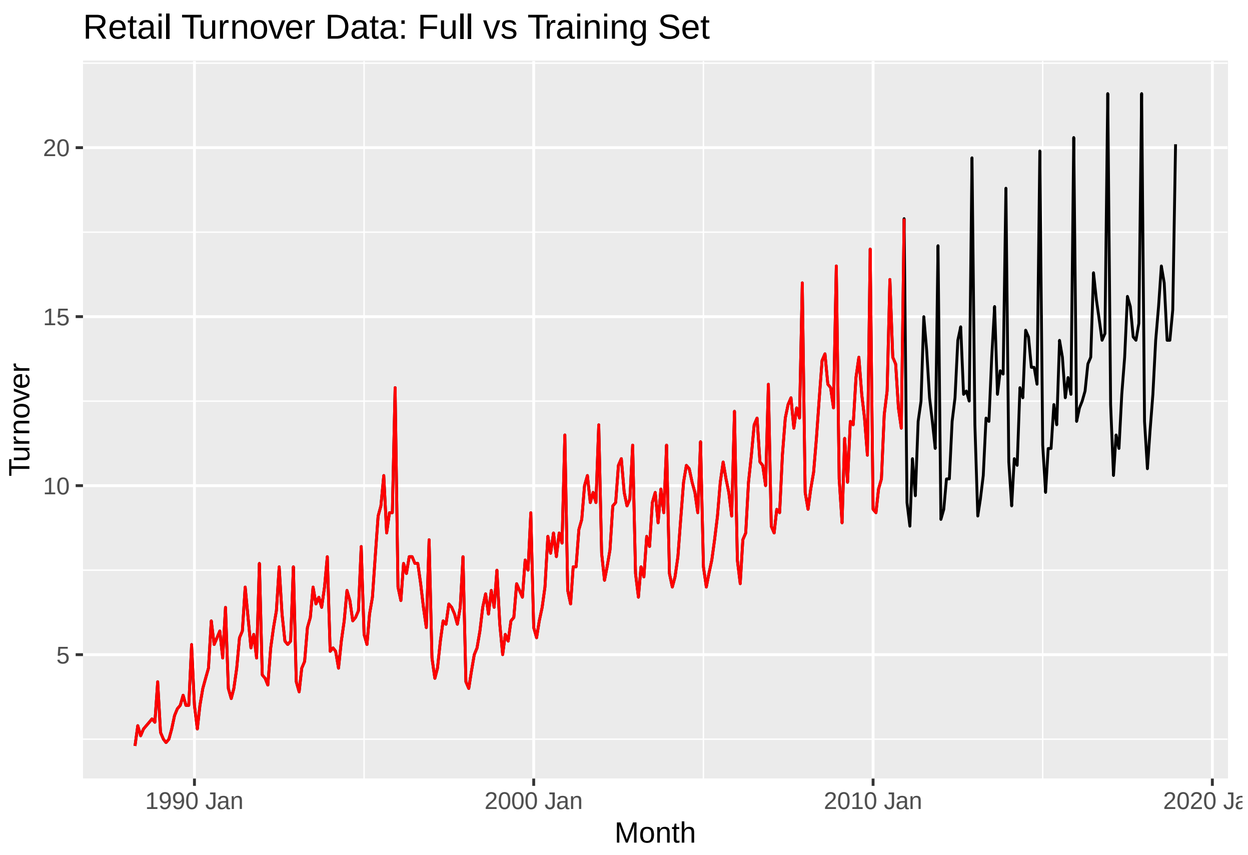
```
bricks_clean <- aus_production %>%
  filter(!is.na(Bricks))

fit_snaive_bricks_repeat <- bricks_clean %>%
  model(SNAIVE(Bricks))

# Forecasting and plot
fit_snaive_bricks_repeat %>%
  forecast(h = "10 years") %>%
  autoplot(bricks_clean) +
  labs(title = "SNAIVE Forecast for Bricks Production",
       x = "Quarter", y = "Bricks Production")
```



Exercise 7.: Retail time series

Splitting the Dataset into Training and Test Sets

```
# Loading aus_retail dataset
set.seed(12345678)
myseries <- aus_retail %>%
  filter(`Series ID` == sample(aus_retail$`Series ID`, 1))

# Splitting the dataset into training data before 2011
myseries_train <- myseries %>%
  filter(year(Month) < 2011)

# Plotting to check that the split is correct
autoplot(myseries, Turnover) +
  autolayer(myseries_train, Turnover, colour = "red") +
  labs(title = "Retail Turnover Data: Full vs Training Set",
       x = "Month", y = "Turnover")
```



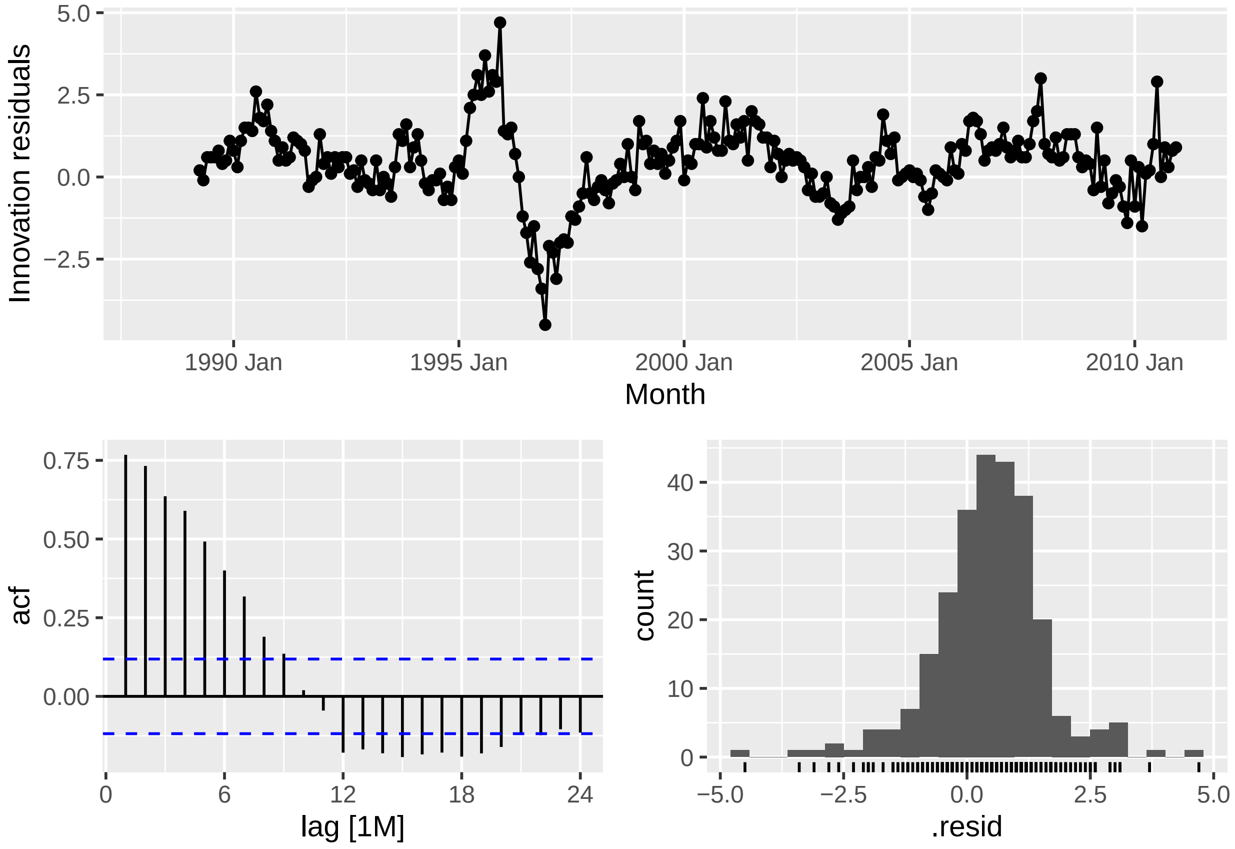
Fit a Seasonal Naïve Model

```r
# Removing missing values from the training dataset
myseries_train_clean <- myseries_train %>%
  filter(!is.na(Turnover))

# Fitting a seasonal naive model to the cleaned training data
fit <- myseries_train_clean %>%
  model(SNAIVE(Turnover))

# Checking 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()`).
```

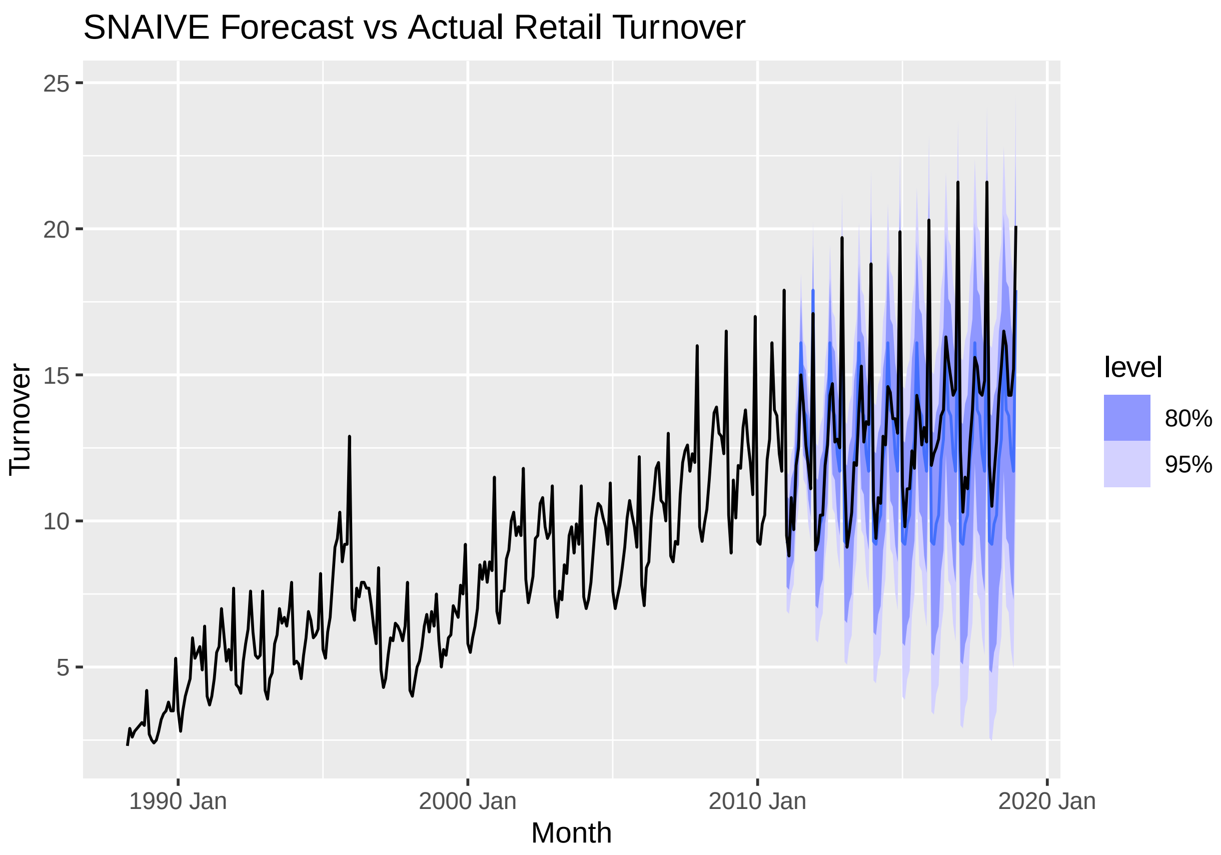


Analyzing Residuals (Above grapgh)

- **Uncorrelated Residuals:** The residuals are **not uncorrelated** due to the presence of significant autocorrelation in the ACF plot.
- **Normally Distributed Residuals:** The residuals **do not follow a perfect normal distribution**. There is some skewness, and the distribution may not be symmetric, as seen in the histogram.

Produce Forecasts for the Test Data

```r
# Producing forecasts for the test data
fc <- fit %>%
  forecast(new_data = anti_join(myseries, myseries_train, by = "Month"))

# Plotting the forecast against the actual data
fc %>% autoplot(myseries) +
  labs(title = "SNAIVE Forecast vs Actual Retail Turnover",
       x = "Month", y = "Turnover")
```



Compare Forecast Accuracy

```
# Calculating accuracy measures for the training data
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
```

```
# Calculating accuracy measures for the test data (using the full dataset for comparison)
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
```

Sensitivity to the Amount of Training Data

- Given the accuracy measures obtained from the training and test data, we have sufficient evidence to conclude that the Seasonal Naive (SNAIVE) model performs reasonably well. The model's accuracy metrics, including RMSE, MAE, MAPE, and ACF1, show acceptable performance with minimal bias (ME close to zero).
- Additionally, the training data provided consistent results when comparing the training and test sets, which suggests that the model is relatively robust.