DATA 624 - Homework 3

Richie Rivera

```r
# Importing the library
library("fpp3")
```

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:

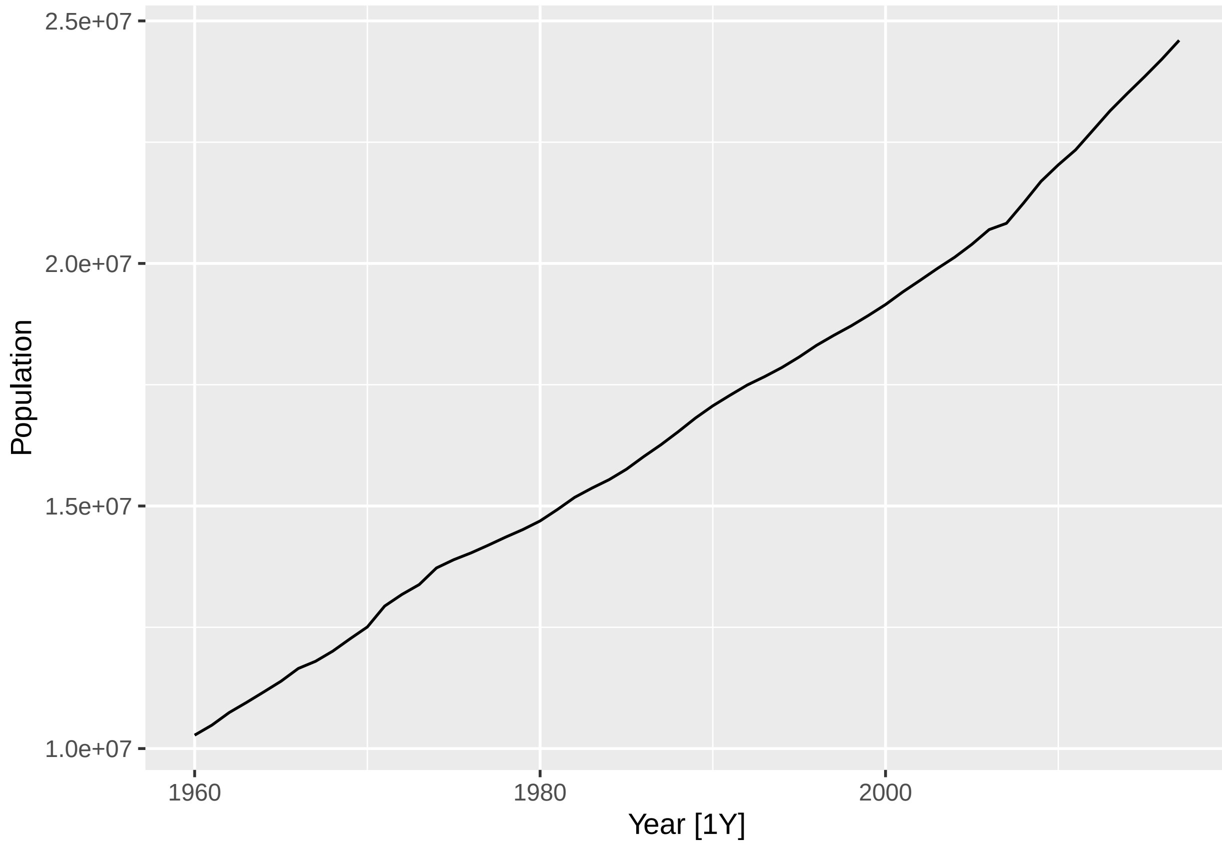
```r
# Getting the global_economy data
data(global_economy)
head(global_economy)
```

A. Australian Population (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
```

```r
# Filter to Australian population
aus_pop <- global_economy |>
  filter(Country == "Australia") |>
  select(Population)

# Plot the data
autoplot(aus_pop)
```

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



Because the data seems to simply grow, I would honestly start with a linear model to see how well it fits the data.

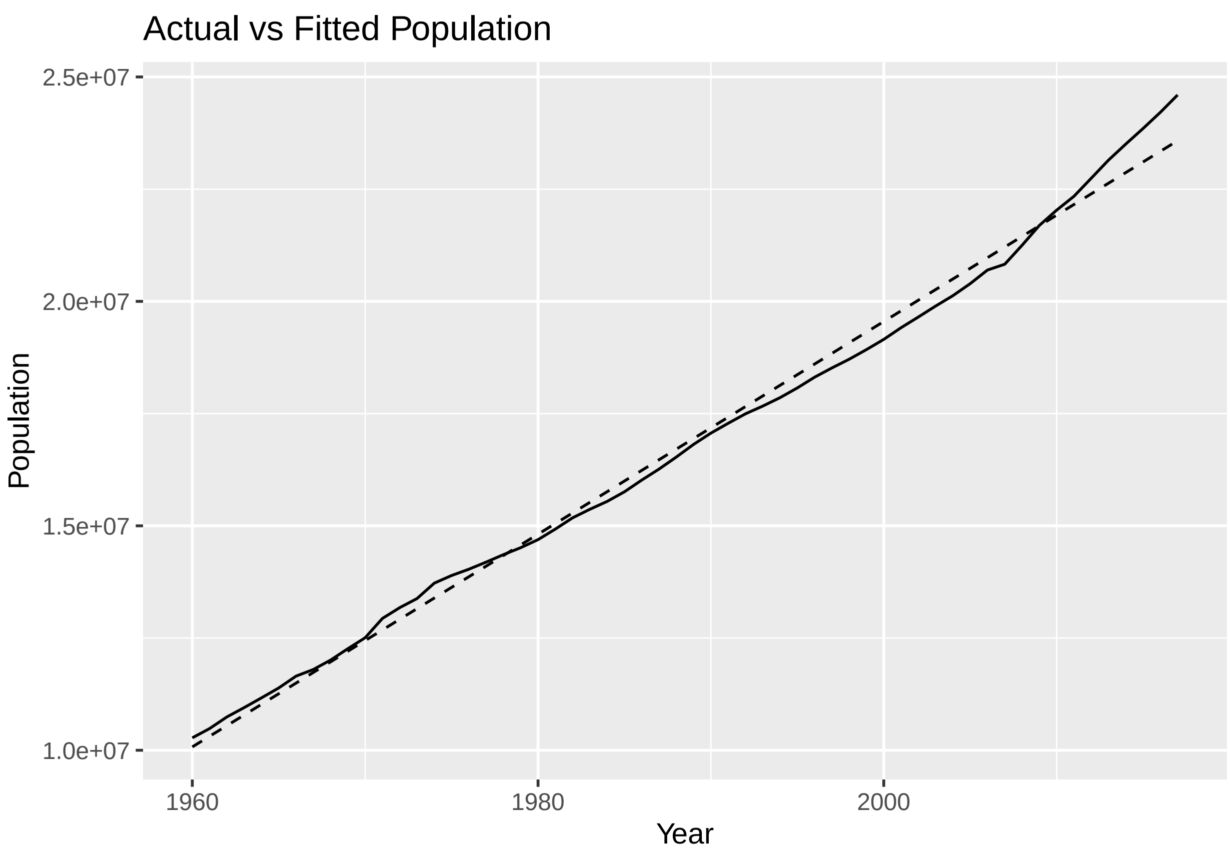
```
aus_pop_lm <- aus_pop |>
  model(TSLM(
    Population ~ Year
  ))

report(aus_pop_lm)
```

```
## Series: Population
## Model: TSLM
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -398673 -236410  -85654  170399 1019552
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept) -454296374    5081649  -89.40   <2e-16 ***
## Year            236924       2555   92.71   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 325800 on 56 degrees of freedom
## Multiple R-squared: 0.9935,  Adjusted R-squared: 0.9934
```

```
## F-statistic:  8596 on 1 and 56 DF, p-value: < 2.22e-16
```

```r
aus_pop %>%
  autoplot(Population) +
  autolayer(
    fitted(aus_pop_lm),
    series = "Fitted",
    linetype = "dashed"
  ) +
  labs(
    title = "Actual vs Fitted Population",
    y = "Population",
    x = "Year"
  ) +
  guides(color = guide_legend(title = "Series"))
```

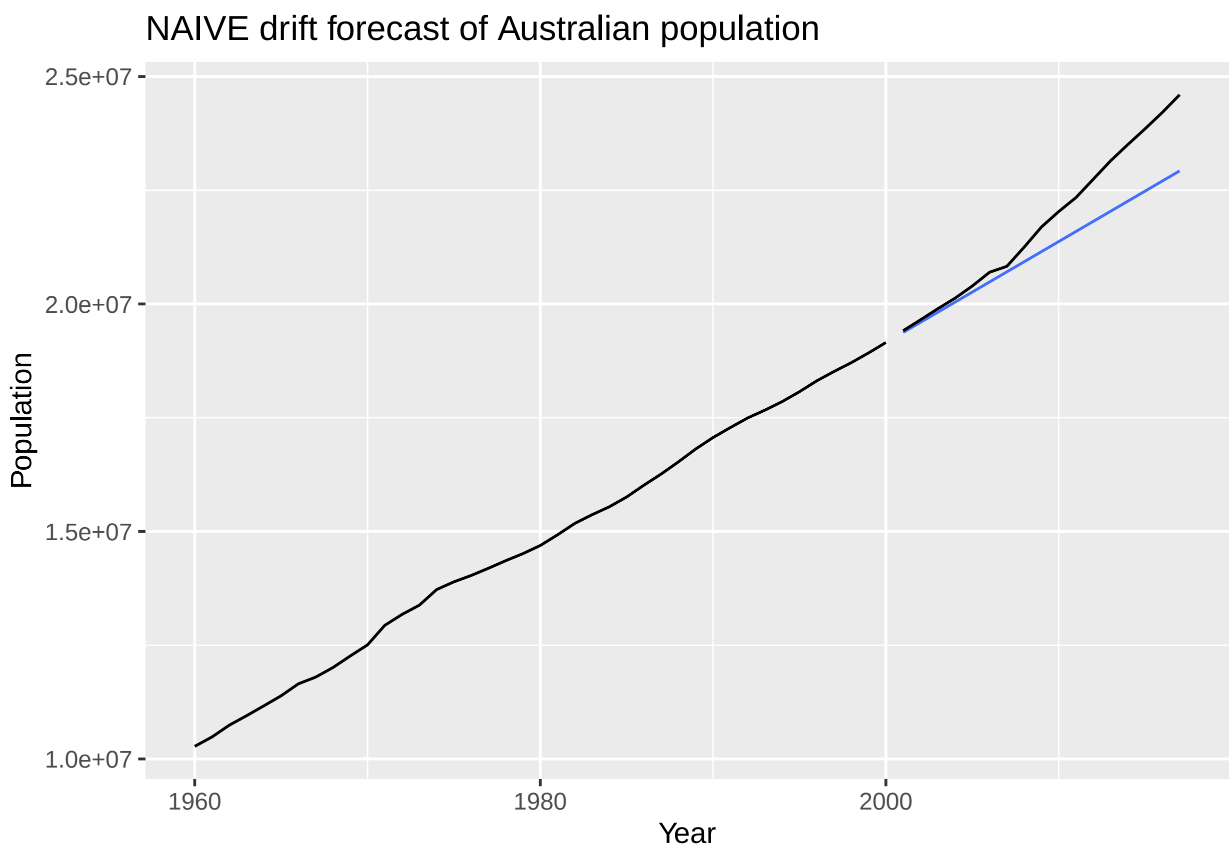
```
## Plot variable not specified, automatically selected `.vars = .fitted`

## Warning in geom_line(eval_tidy(expr(aes(!!!aes_spec))), data = object, ..., :
## Ignoring unknown parameters: `series`
```



From here, we can see that this can be well predicted with a linear model. Although that isn't an option, this does inform me that a `NAIVE()` model with drift would be good here (`RW(~drift)`). This is because: 1. A simple `NAIVE()` would show the value fixed at the last known datapoint. 2. A `SNAIVE(y)` would be seasonal, which the graph above proves that this is simply just growing. 3. This leaves us with a `NAIVE()` model with drift (`RW(y~drift)`) as it allows the forecast to grow/decrease over time.

```r
# Split the data into a train dataset
aus_pop_train <- aus_pop |>
  filter_index("1960" ~ "2000")

# Fit a drift model
aus_pop_drift_fit <- aus_pop_train |>
  model(RW(Population ~ drift()))

# make predictions with the model for the next 17 years
aus_pop_drift_forecast <- aus_pop_drift_fit |>
  forecast(h = 17)

# Plot forecast against the actuals
aus_pop_drift_forecast |>
  autoplot(aus_pop_train, level = NULL) +
  autolayer(
    filter_index(aus_pop, "2001" ~ .),
    colour = "black"
  ) +
  labs(
    y = "Population",
    x = "Year",
    title = "NAIVE drift forecast of Australian population"
  ) +
  guides(
    colour = guide_legend(title = "forecast")
  )
```

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



I'll be using the framework above for the following plots and refer to reasons by 1, 2, or 3.

```r
# load the data
data(aus_production)

# inspect the data
head(aus_production)
```

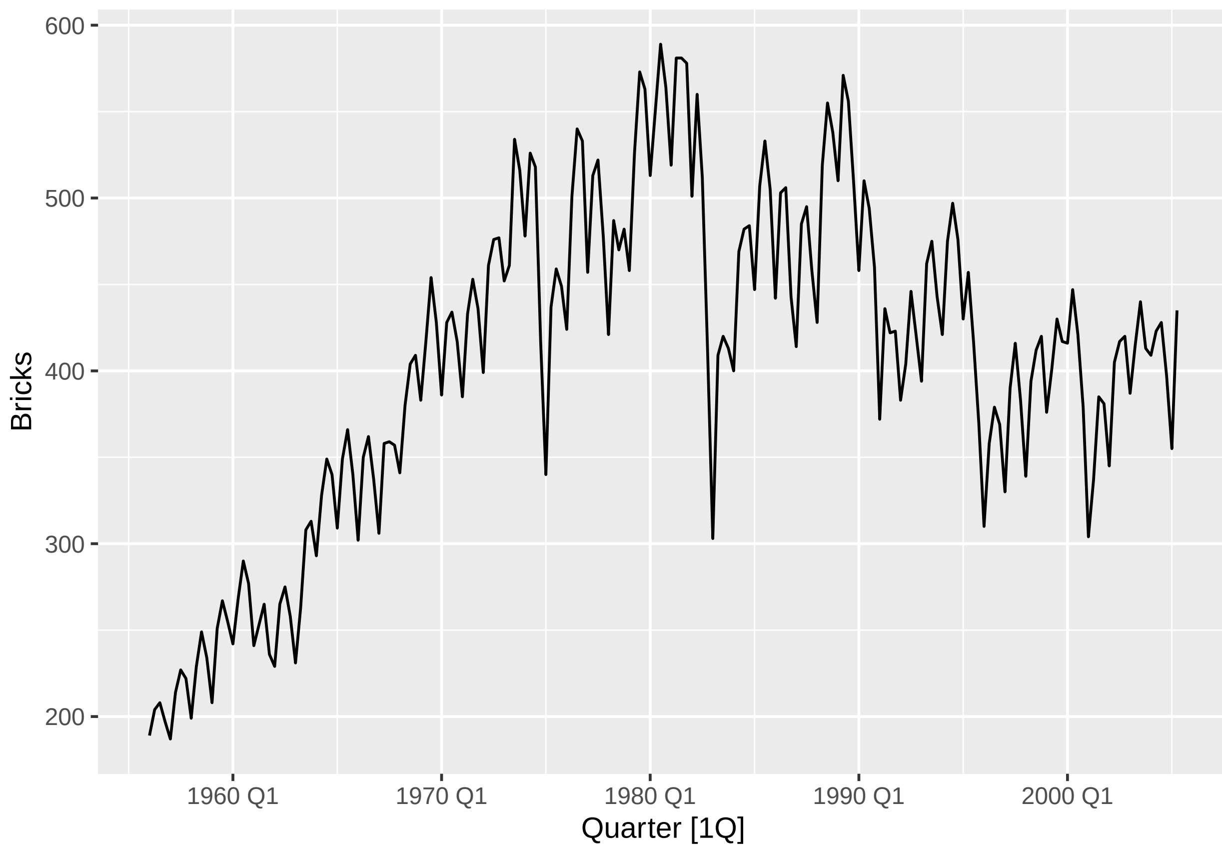
B. Bricks (aus_production)

```
## # 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
```

```r
# Remove unnecessary fields and remove NA datapoints
bricks <- aus_production |>
  select(Bricks) |>
```

```
  drop_na()

autoplot(bricks)
```

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

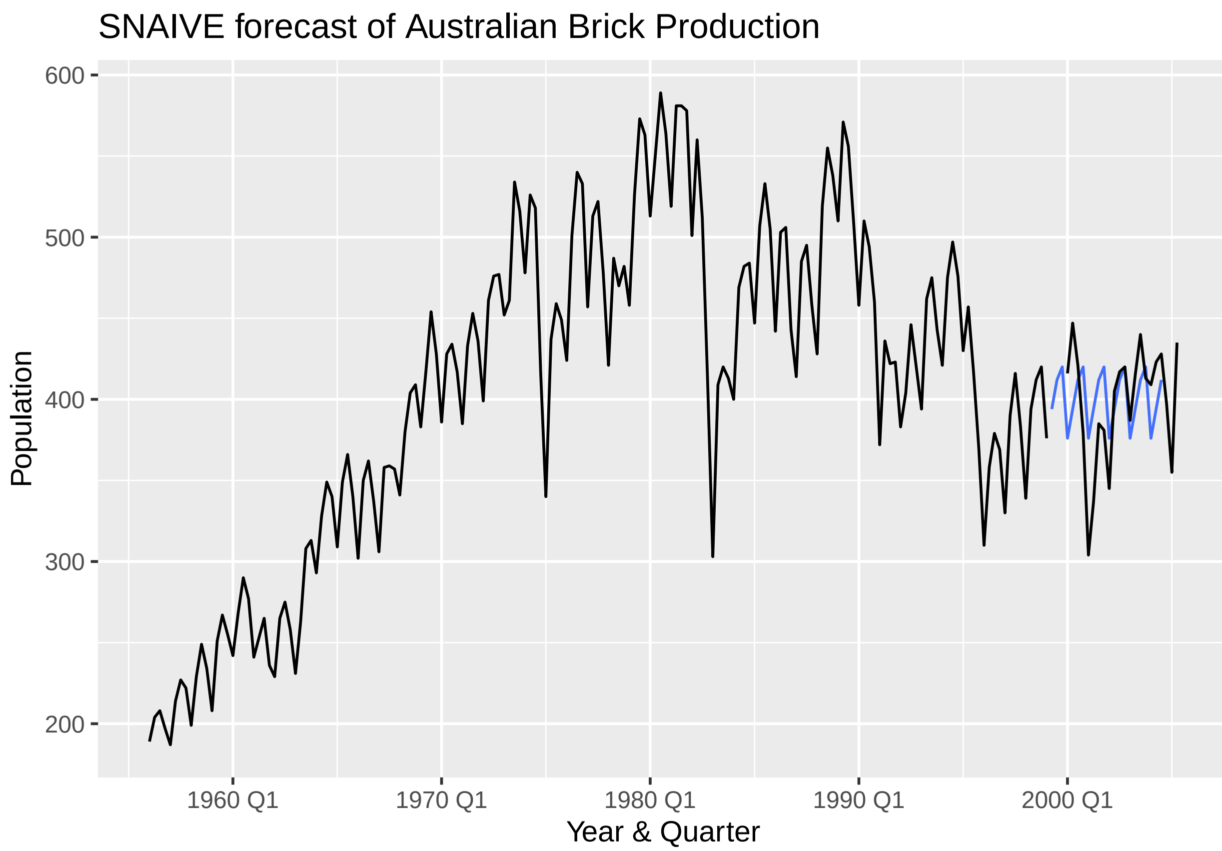


This data looks like a random walk with some seasonality. With that, I think it makes sense to use a `SNAIVE()` model here:

```
# Split the data into a train dataset
bricks_train <- bricks |>
  filter_index("1956" ~ "1999")

# Fit a snaive model
bricks_snaive_fit <- bricks_train |>
  model(SNAIVE(Bricks))

# make predictions with the model for the next 18 quarters
bricks_snaive_forecast <- bricks_snaive_fit |>
  forecast(h = 22)

# Plot forecast against the actuals
bricks_snaive_forecast |>
  autoplot(bricks_train, level = NULL) +
  autolayer(
```

```r
    filter_index(bricks, "2000" ~ .),
    colour = "black"
  ) +
  labs(
    y = "Population",
    x = "Year & Quarter",
    title = "SNAIVE forecast of Australian Brick Production"
  ) +
  guides(
    colour = guide_legend(title = "forecast")
  )
```

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



This model is not perfect but we can see from the blue line that the values don't seem to typically deviate too far from the predictions.

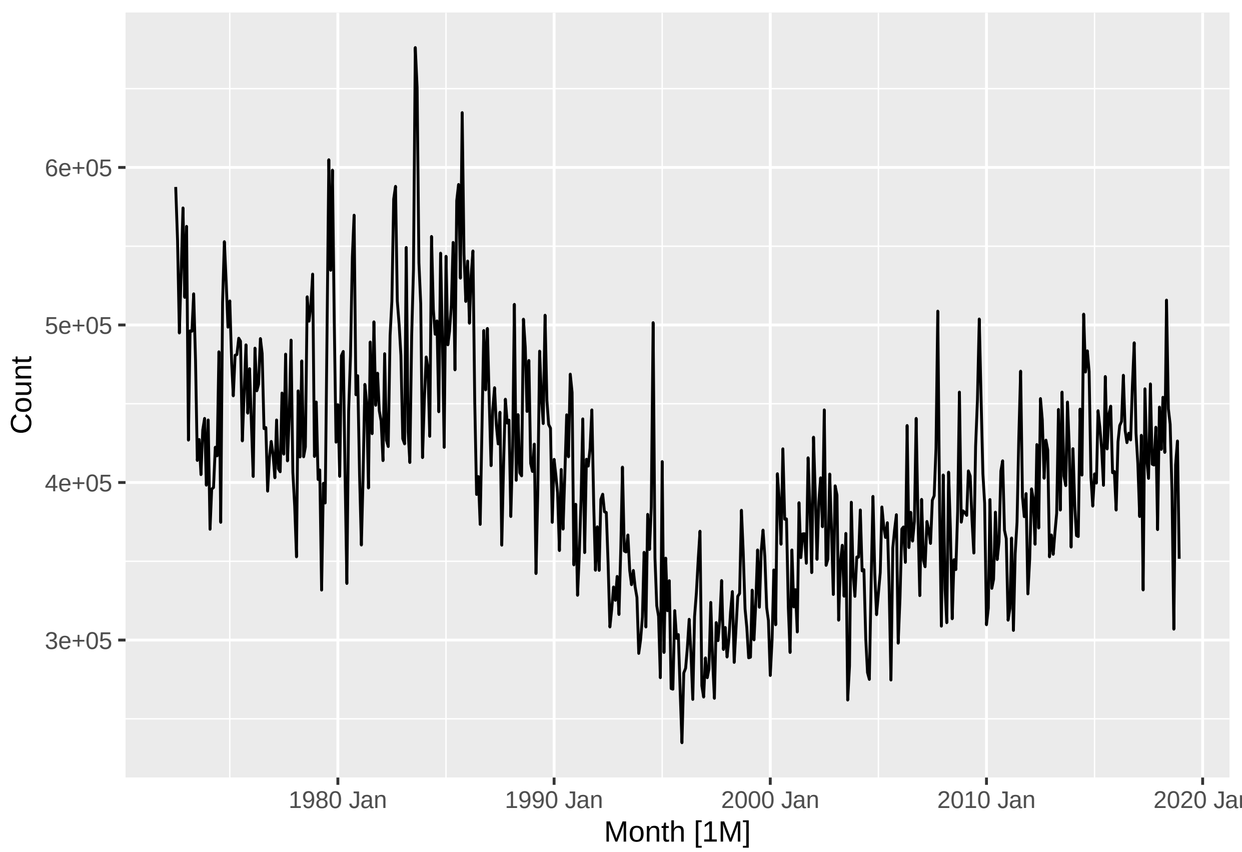
```r
# load the data
data(aus_livestock)

# inspect the data
head(aus_livestock)
```

C. NSW Lambs (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
```

```
# Remove unnecessary fields and remove NA datapoints
lambs <- aus_livestock |>
  filter(
    Animal == "Lambs",
    State == "New South Wales"
  ) |>
  select(Count)

autoplot(lambs)
```

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



This data looks more like a random walk, so I'll continue with a `NAIVE()` model:

```r
# Split the data into a train dataset
lambs_train <- lambs |>
  filter_index("1956" ~ "2015")
```

```
## Warning: There were 2 warnings in `filter()`.
## The first warning was:
## i In argument: `time_in(Month, ...)`.
## Caused by warning:
## ! `yearmonth()` may yield unexpected results.
## i Please use arg `format` to supply formats.
## i Run `dplyr::last_dplyr_warnings()` to see the 1 remaining warning.
```

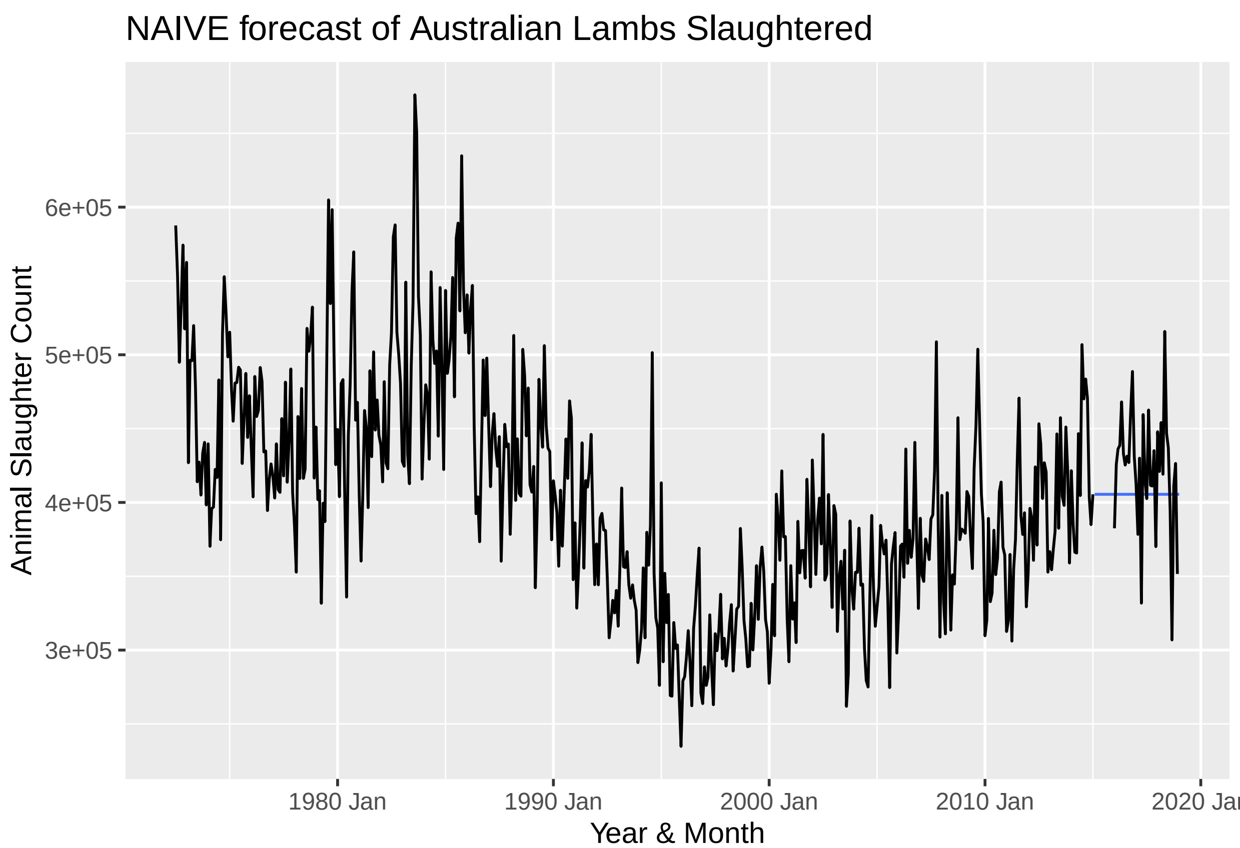
```r
# Fit a naive model
lambs_naive_fit <- lambs_train |>
  model(NAIVE())
```

```
## Model not specified, defaulting to automatic modelling of the `Count` variable.
## Override this using the model formula.
```

```r
# make predictions with the model for the next 48 months (4 years)
lambs_naive_forecast <- lambs_naive_fit |>
  forecast(h = 12 * 4)

# Plot forecast against the actuals
lambs_naive_forecast |>
  autoplot(lambs_train, level = NULL) +
  autolayer(
    filter_index(lambs, "2016" ~ .),
    colour = "black"
  ) +
  labs(
    y = "Animal Slaughter Count",
    x = "Year & Month",
    title = "NAIVE forecast of Australian Lambs Slaughtered"
  ) +
  guides(
    colour = guide_legend(title = "forecast")
  )
```

```
## Warning: There was 1 warning in `filter()`.
## i In argument: `time_in(Month, ...)`.
## Caused by warning:
## ! `yearmonth()` may yield unexpected results.
## i Please use arg `format` to supply formats.
```

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



D. Household wealth (hh_budget). Running a `?hh_budget` I find that this is a percentage and has multiple countries in it. For this model, I've decided to only include Australia:

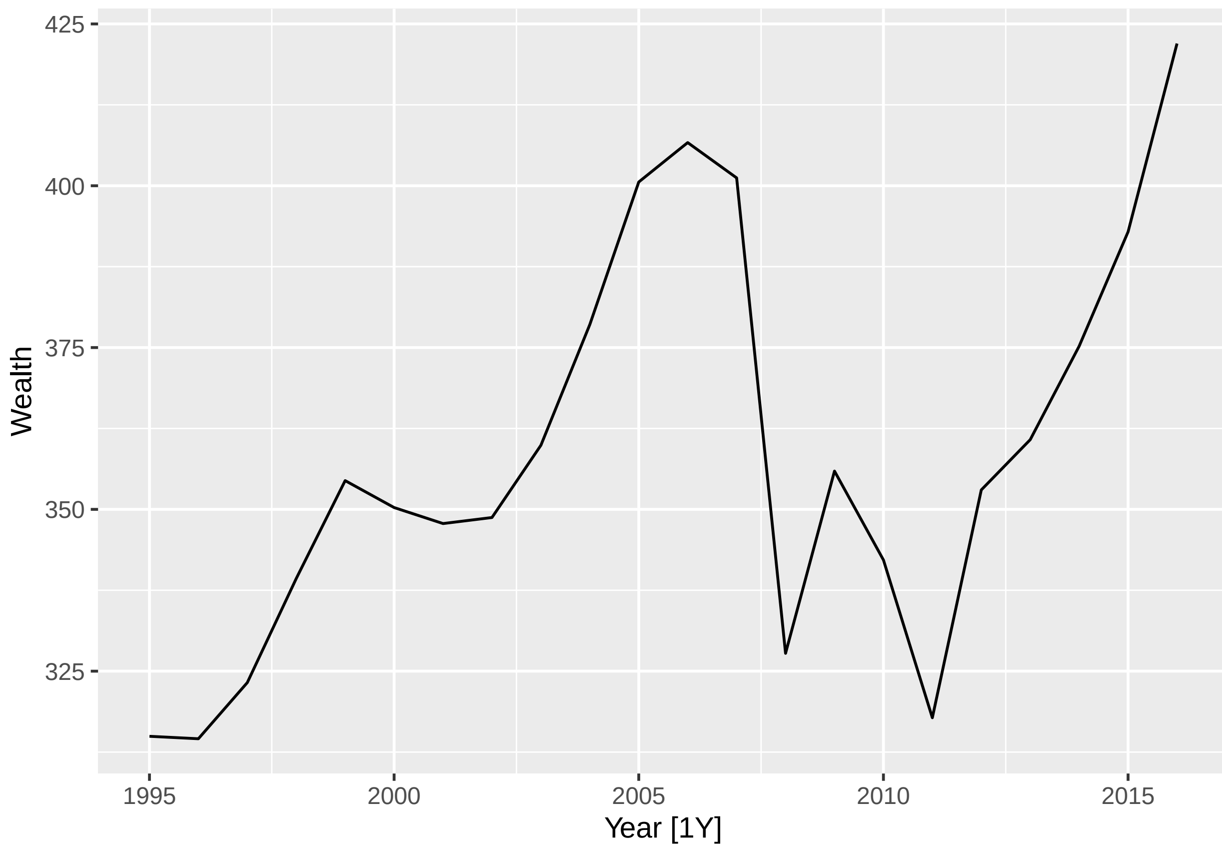
```
# load the data
data(hh_budget)

# inspect the data
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
```

```
# Remove unnecessary fields and remove NA datapoints
hh_wealth <- hh_budget |>
  filter(
    Country == "Australia"
```

```
  ) |>
  select(Wealth)

autoplot(hh_wealth)
```

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

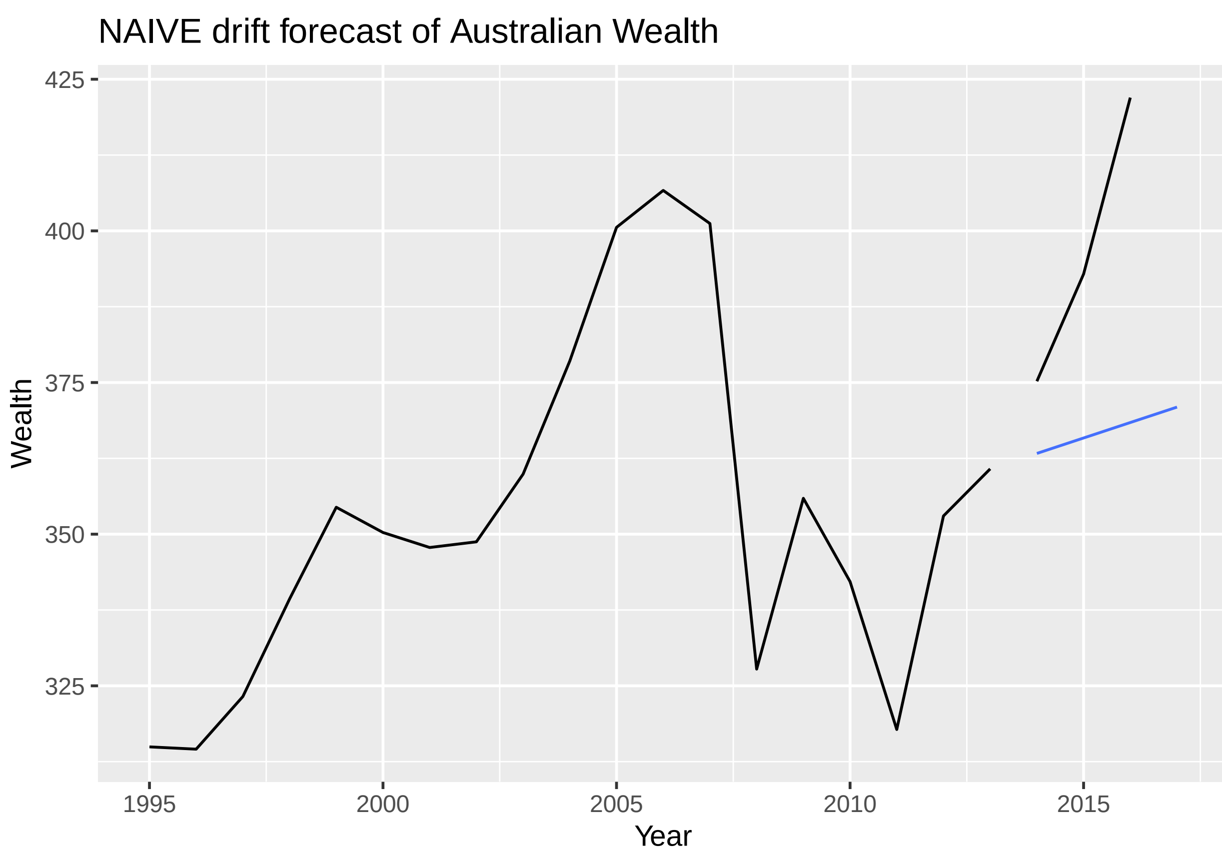


From this graph, I don't see any seasonality but I do see that there are windows where trends stay consistent. Looking at the most recent datapoints, I can see that it's on a growth trajectory. For this reason, I believe a `RW(y ~ drift())` will be most applicable here:

```
# Split the data into a train dataset
hh_wealth_train <- hh_wealth |>
  filter_index("1960" ~ "2013")

# Fit a drift model
hh_wealth_drift_fit <- hh_wealth_train |>
  model(RW(Wealth ~ drift()))

# make predictions with the model for the next 5 years
hh_wealth_drift_forecast <- hh_wealth_drift_fit |>
  forecast(h = 4)

# Plot forecast against the actuals
hh_wealth_drift_forecast |>
```

```r
autoplot(hh_wealth_train, level = NULL) +
  autolayer(
    filter_index(hh_wealth, "2014" ~ .),
    colour = "black"
  ) +
  labs(
    y = "Wealth",
    x = "Year",
    title = "NAIVE drift forecast of Australian Wealth"
  ) +
  guides(
    colour = guide_legend(title = "forecast")
  )
```

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



The model has the right idea, but it's not really good at capturing the magnitude of the growth.

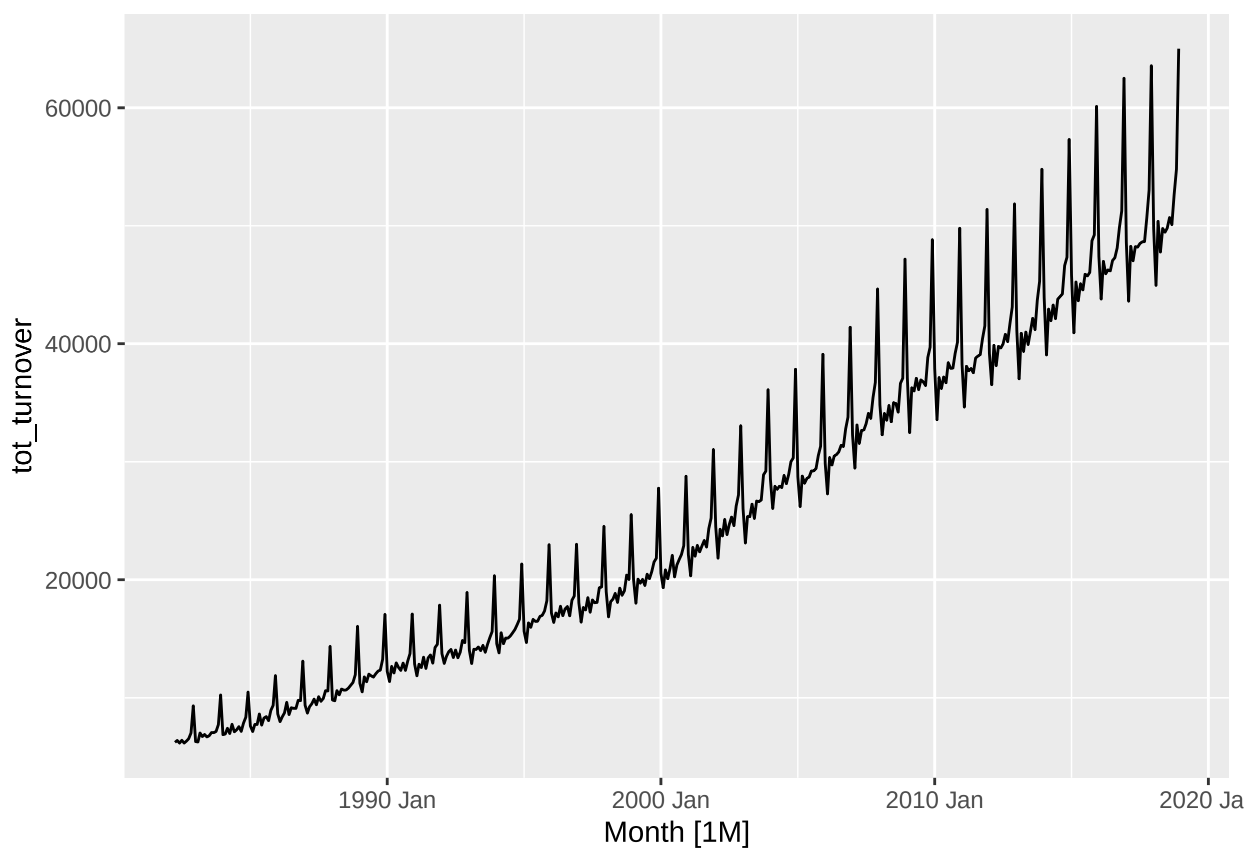
```r
# load the data
data(aus_retail)

# inspect the data
head(aus_retail)
```

E. Australian takeaway food turnover (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
```

```
# Remove unnecessary fields and remove NA datapoints
turnover <- aus_retail |>
  index_by(Month) |>
  summarise(tot_turnover = sum(Turnover)) |>
  ungroup() |>
  select(tot_turnover) |>
  as_tsibble(index = Month)

autoplot(turnover)
```

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



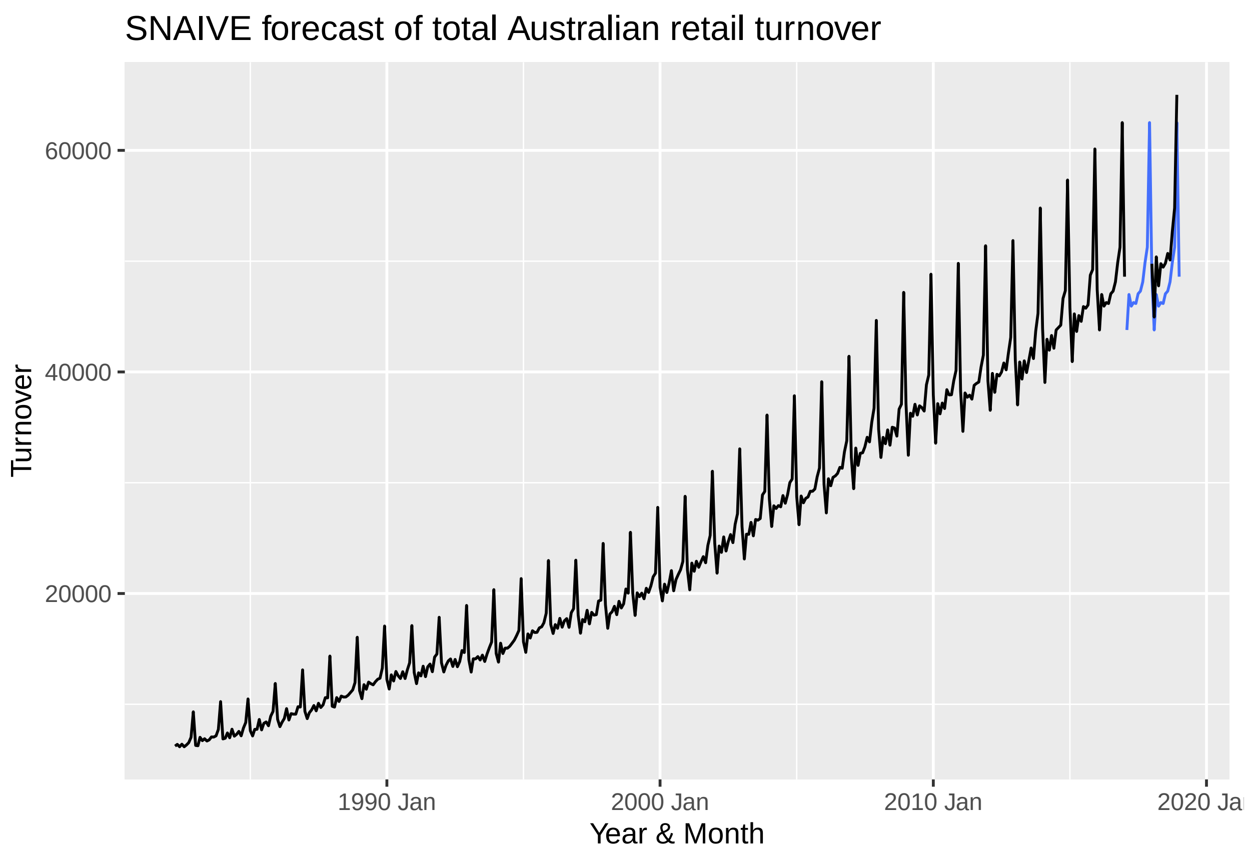
This is seasonal and shows growth. For this it'd be convenient if we can combine `SNAIVE(y)` and a `RW(y ~ drift())`. But given the options we have, I would actually think that a `SNAIVE(y)` would be better for shorter prediction windows. If we were going with a larger prediction window, then it'd be likely better to go with a drift model.

```r
# Split the data into a train dataset
turnover_train <- turnover |>
  filter_index("1982" ~ "2017")
```

```
## Warning: There were 2 warnings in `filter()`.
## The first warning was:
## i In argument: `time_in(Month, ...)`.
## Caused by warning:
## ! `yearmonth()` may yield unexpected results.
## i Please use arg `format` to supply formats.
## i Run `dplyr::last_dplyr_warnings()` to see the 1 remaining warning.
```

```r
# Fit a snaive model
turnover_snaive_fit <- turnover_train |>
  model(SNAIVE(tot_turnover))

# make predictions with the model for the next 24 months
turnover_snaive_forecast <- turnover_snaive_fit |>
  forecast(h = 24)

# Plot forecast against the actuals
turnover_snaive_forecast |>
  autoplot(turnover_train, level = NULL) +
  autolayer(
    filter_index(turnover, "2018" ~ .),
    colour = "black"
  ) +
  labs(
    y = "Turnover",
    x = "Year & Month",
    title = "SNAIVE forecast of total Australian retail turnover"
  ) +
  guides(
    colour = guide_legend(title = "forecast")
  )
```

```
## Warning: There was 1 warning in `filter()`.
## i In argument: `time_in(Month, ...)`.
## Caused by warning:
## ! `yearmonth()` may yield unexpected results.
## i Please use arg `format` to supply formats.

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



Actually, the `SNAIVE()` did a wonderful job at capturing this trend! I believe that if I were to increase the prediction window, it would not perform as well. Here's some code of me testing that;

```
# Split the data into a train dataset
turnover_train <- turnover |>
  filter_index("1982" ~ "2010")
```

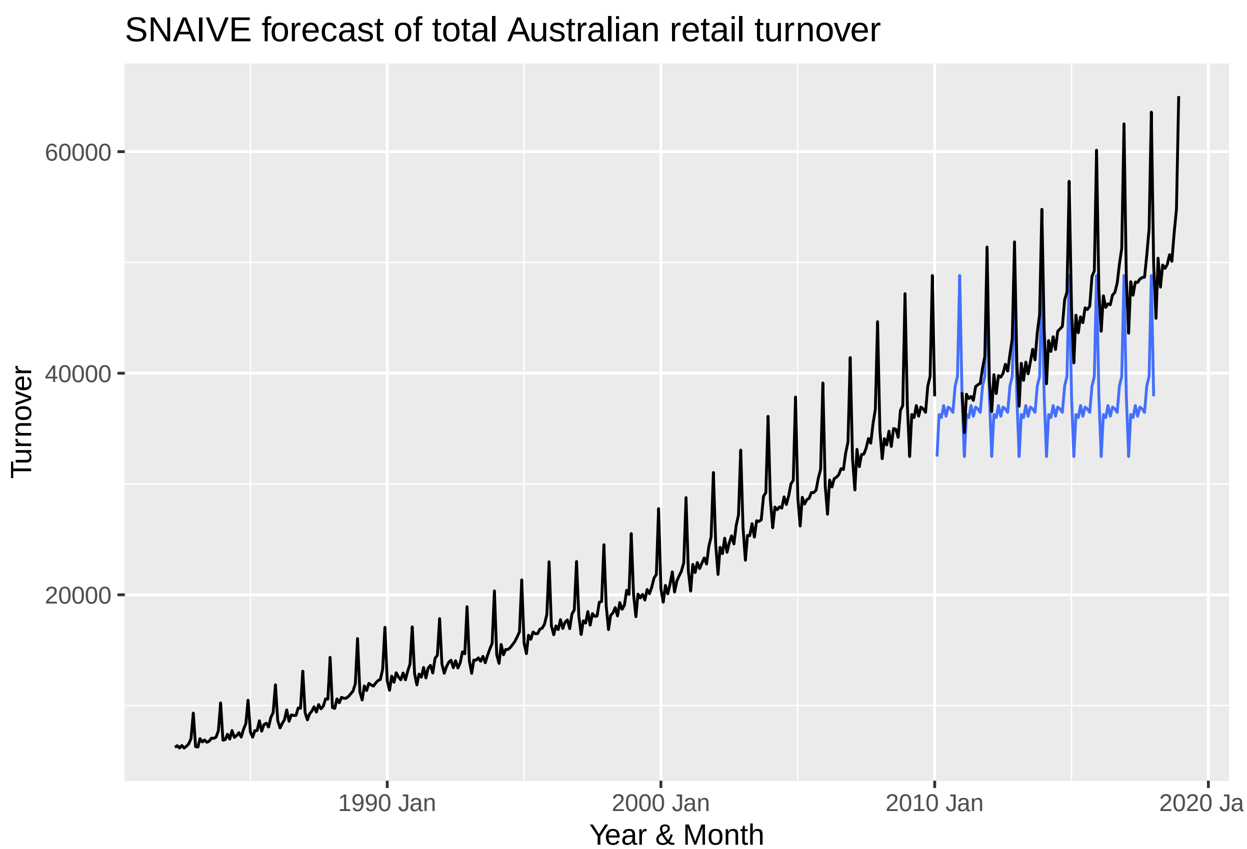
```
## Warning: There were 2 warnings in ‘filter()‘.
## The first warning was:
## i In argument: ‘time_in(Month, ...)‘.
## Caused by warning:
## ! ‘yearmonth()‘ may yield unexpected results.
## i Please use arg ‘format‘ to supply formats.
## i Run ‘dplyr::last_dplyr_warnings()‘ to see the 1 remaining warning.
```

```
# Fit a snaive model
turnover_snaive_fit <- turnover_train |>
  model(SNAIVE(tot_turnover))

# make predictions with the model for the next 8 years
turnover_snaive_forecast <- turnover_snaive_fit |>
  forecast(h = 8 * 12)

# Plot forecast against the actuals
turnover_snaive_forecast |>
  autoplot(turnover_train, level = NULL) +
```

```r
  autolayer(
    filter_index(turnover, "2011" ~ .),
    colour = "black"
  ) +
  labs(
    y = "Turnover",
    x = "Year & Month",
    title = "SNAIVE forecast of total Australian retail turnover"
  ) +
  guides(
    colour = guide_legend(title = "forecast")
  )
```

```
## Warning: There was 1 warning in `filter()`.
## i In argument: `time_in(Month, ...)`.
## Caused by warning:
## ! `yearmonth()` may yield unexpected results.
## i Please use arg `format` to supply formats.
```

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



Just as I thought, this doesn't perform as well for longer prediction windows. But it has extracted the seasonality very well.

Question 5.2

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

A. Produce a time plot of the series. Since no metric was defined, I decided to use the Close price:

```
# importing the data
data(gafa_stock)

# inspecting the data
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
```

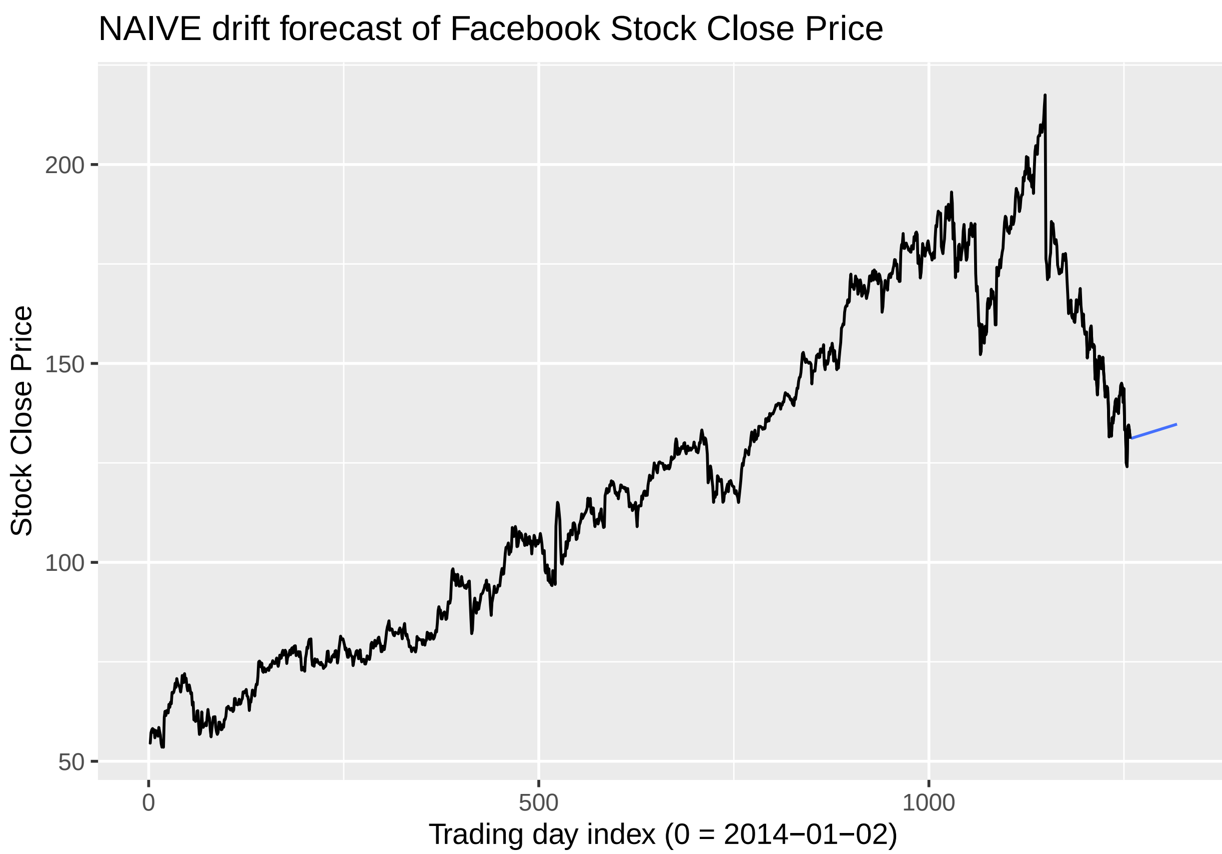
```
# Filtering the data to facebook
fb <- gafa_stock |>
  filter(Symbol == "FB") |>
  select(Close) |>
  mutate(Date = row_number()) |>
  rename(trading_day = Date)

autoplot(fb)
```

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

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

```
# Fit a drift model
fb_drift_fit <- fb |>
  model(RW(Close ~ drift()))

# make predictions with the model for the next 58 trading days
fb_drift_forecast <- fb_drift_fit |>
  forecast(h = 60)

# Plot forecast against the actuals
fb_drift_forecast |>
  autoplot(fb, level = NULL) +
  autolayer(
    filter_index(fb, "1201" ~ .),
    colour = "black"
  ) +
  labs(
    y = "Stock Close Price",
    x = "Trading day index (0 = 2014-01-02)",
    title = "NAIVE drift forecast of Facebook Stock Close Price"
  ) +
  guides(
    colour = guide_legend(title = "forecast")
  )
```

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



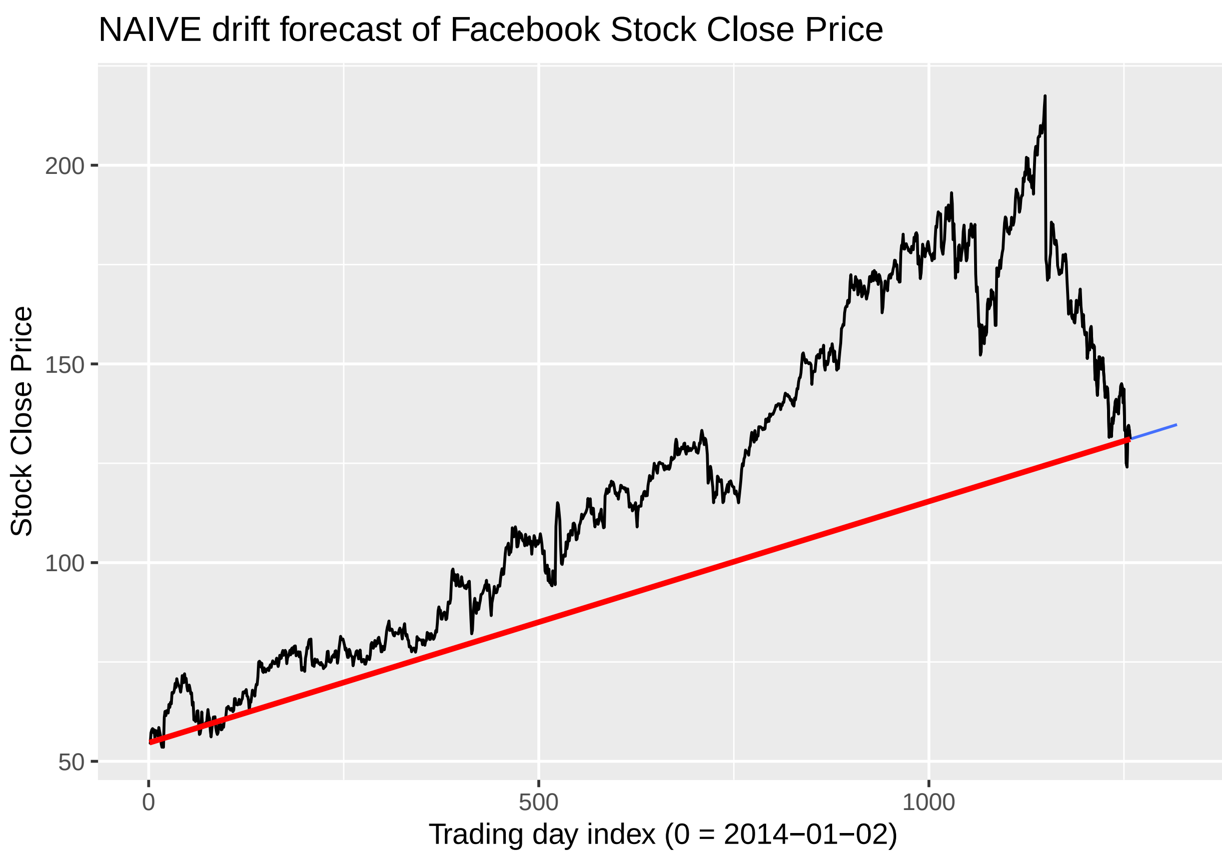

```r
fb_first_last <- data.frame(
  x1 = head(fb, 1)$trading_day,
  x2 = tail(fb, 1)$trading_day,
  y1 = head(fb, 1)$Close,
  y2 = tail(fb, 1)$Close
)

fb_drift_forecast |>
  autoplot(fb, level = NULL) +
  autolayer(
    filter_index(fb, "1201" ~ .),
    colour = "black"
  ) +
  labs(
    y = "Stock Close Price",
    x = "Trading day index (0 = 2014-01-02)",
    title = "NAIVE drift forecast of Facebook Stock Close Price"
  ) +
  guides(
    colour = guide_legend(title = "forecast")
  ) +
  geom_segment(
    data = fb_first_last,
```

```
    aes(x = x1, xend = x2, y = y1, yend = y2),
    color = "red", size = 1
  )
```

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

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

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



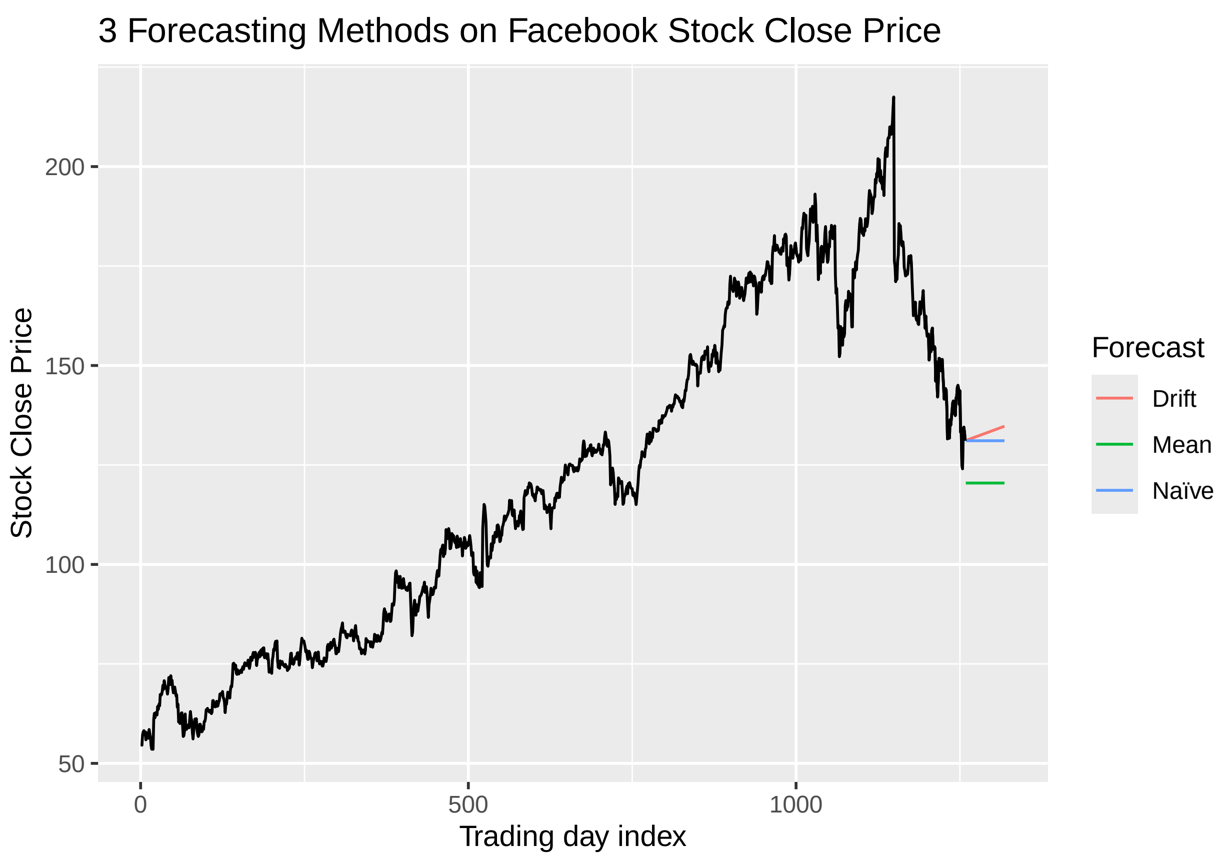
The graph above makes it pretty obvious that the blue line is a continuation of the red line where the red line is the linear interpolation between the first and last point.

D. Try using some of the other benchmark functions to forecast the same data set. Which do you think is best? Why? Using all the methods we've been exposed to:

```
fb_all_fit <- fb |>
  model(
    Mean = MEAN(Close),
```

```
    `Naïve` = NAIVE(Close),
    Drift = NAIVE(Close ~ drift())
  )

fb_all_forecast <- fb_all_fit |>
  forecast(h = 60)

fb_all_forecast |>
  autoplot(fb, level = NULL) +
  autolayer(
    filter_index(fb, "1201" ~ .),
    colour = "black"
  ) +
  labs(
    x = "Trading day index",
    y = "Stock Close Price",
    title = "3 Forecasting Methods on Facebook Stock Close Price"
  ) +
  guides(colour = guide_legend(title = "Forecast"))
```

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



Given these forecasts, I would likely pick the drift one. This is because it seems to have a random walk and it does seem to typically grow over time (we all know that stocks only go up...). For this reason, a drift forecast seems the most applicable.

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?

By looking at these graphs: 1. The mean of the residuals is close to zero 2. The time plot of the residuals show that the variation stays fairly consistent across the historical data. 3. The histogram seems to be a little normal but not very normal. To me it looks like a bimodal distribution.

With this, we can conclude that forecasts built using this method will likely be good.

Question 5.4

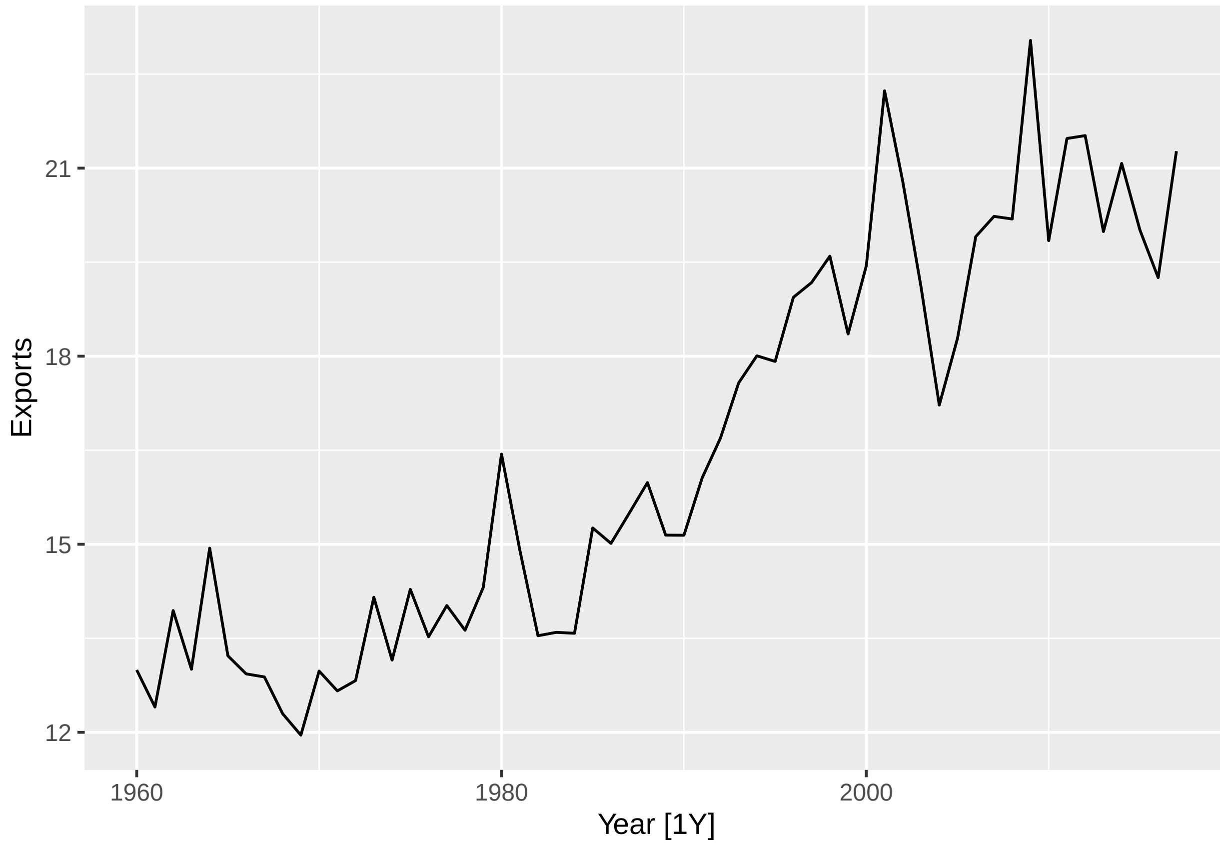
Repeat the previous exercise using the Australian Exports series from `global_economy` and the Bricks series from `aus_production`. Use whichever of `NAIVE()` or `SNAIVE()` is more appropriate in each case.

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

head(aus_exports)
```

```
## # A tsibble: 6 x 2 [1Y]
##   Exports  Year
##     <dbl> <dbl>
## 1    13.0  1960
## 2    12.4  1961
## 3    13.9  1962
## 4    13.0  1963
## 5    14.9  1964
## 6    13.2  1965
```

```
autoplot(aus_exports)
```

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



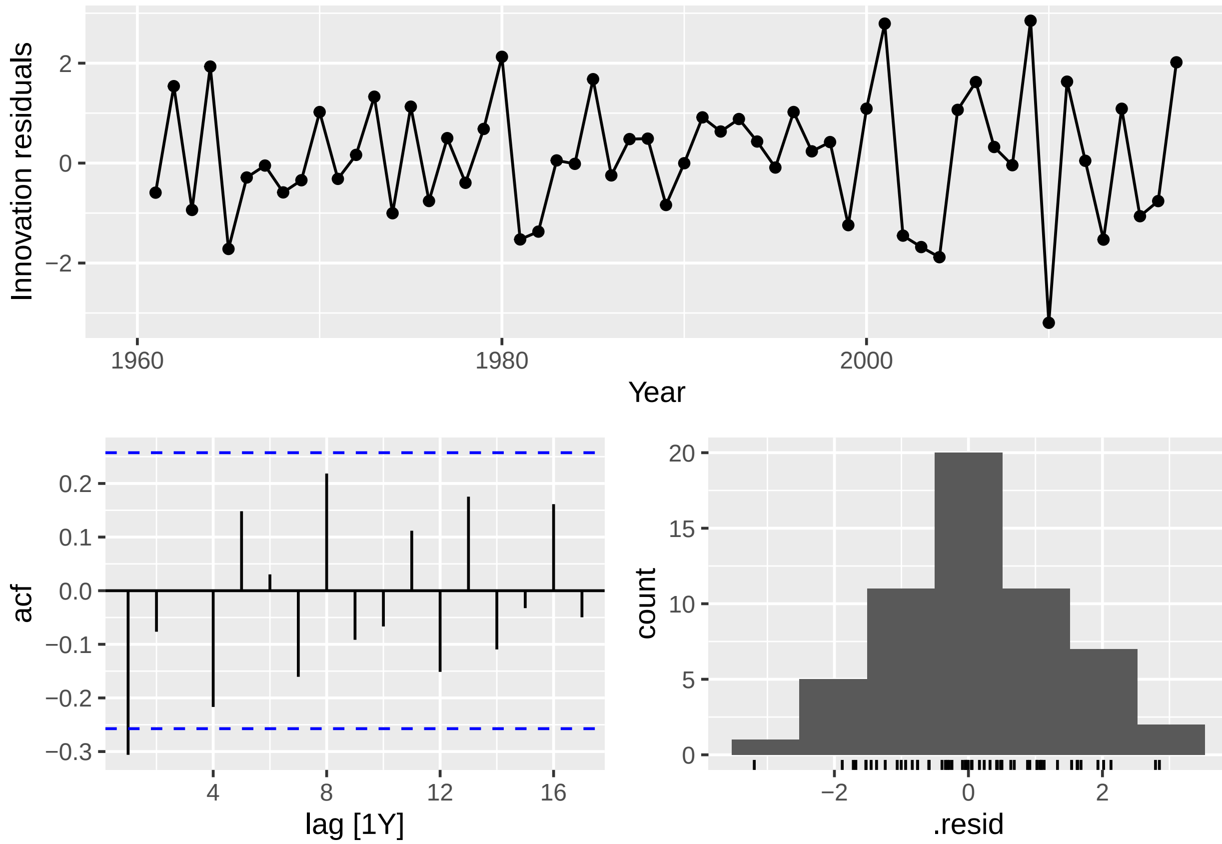
Given the above, Australian Exports from `global_economy` should use `NAIVE()`. From a previous exercise, we determined that using `SNAIVE()` for bricks was a good choice:

```r
# Define and estimate a model
aus_exports_fit <- aus_exports |>
  model(NAIVE(Exports))
# Look at the residuals
aus_exports_fit |>
  gg_tsresiduals()
```

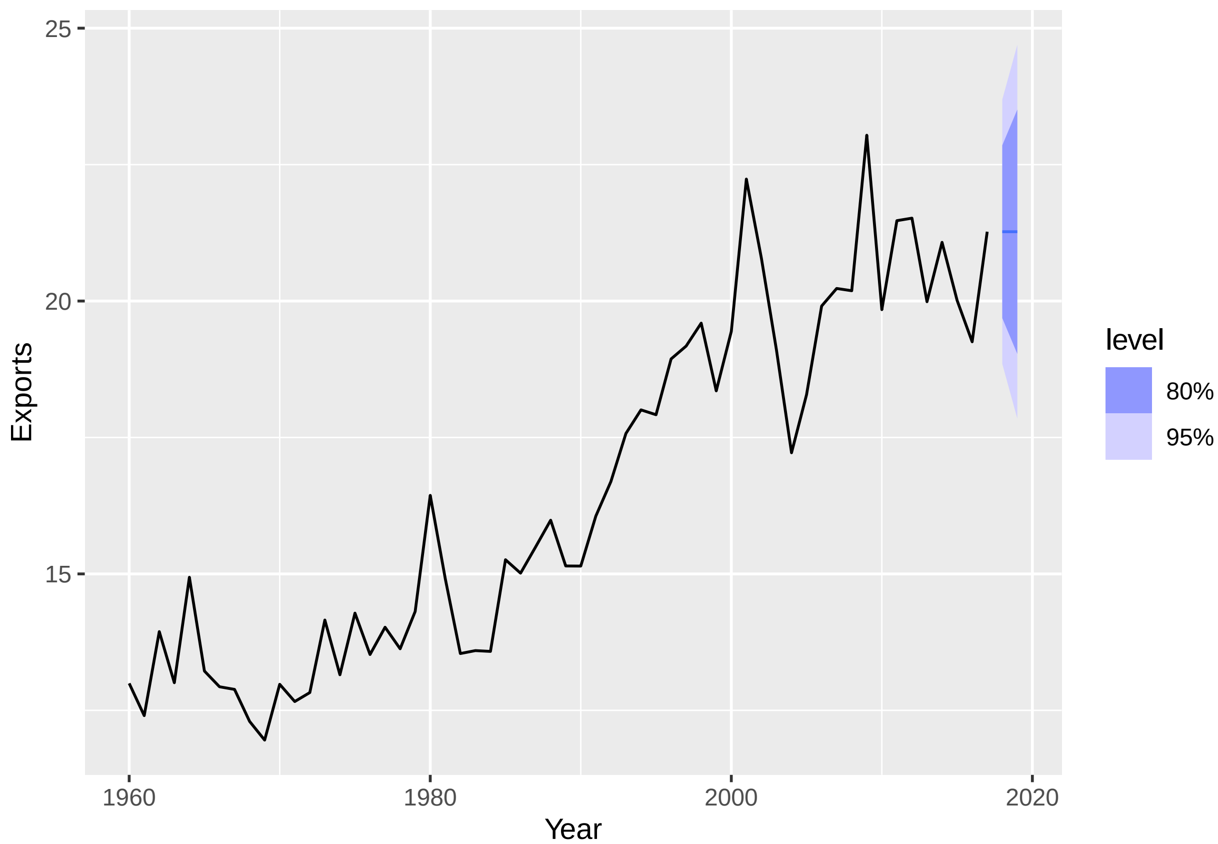
```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).
```

```
## 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
# Look a some forecasts
aus_exports_fit |>
  forecast() |>
  autoplot(aus_exports)
```



Looking at Australian Exports `NAIVE()` model, we can see that: 1. The mean of the residuals is close to 0 2. The time series plot of residuals has a few sharp increases and decreases, specifically in the mid 2000s and 2010 but it typically has fairly consistent variation. 3. The histogram of residuals is very normal

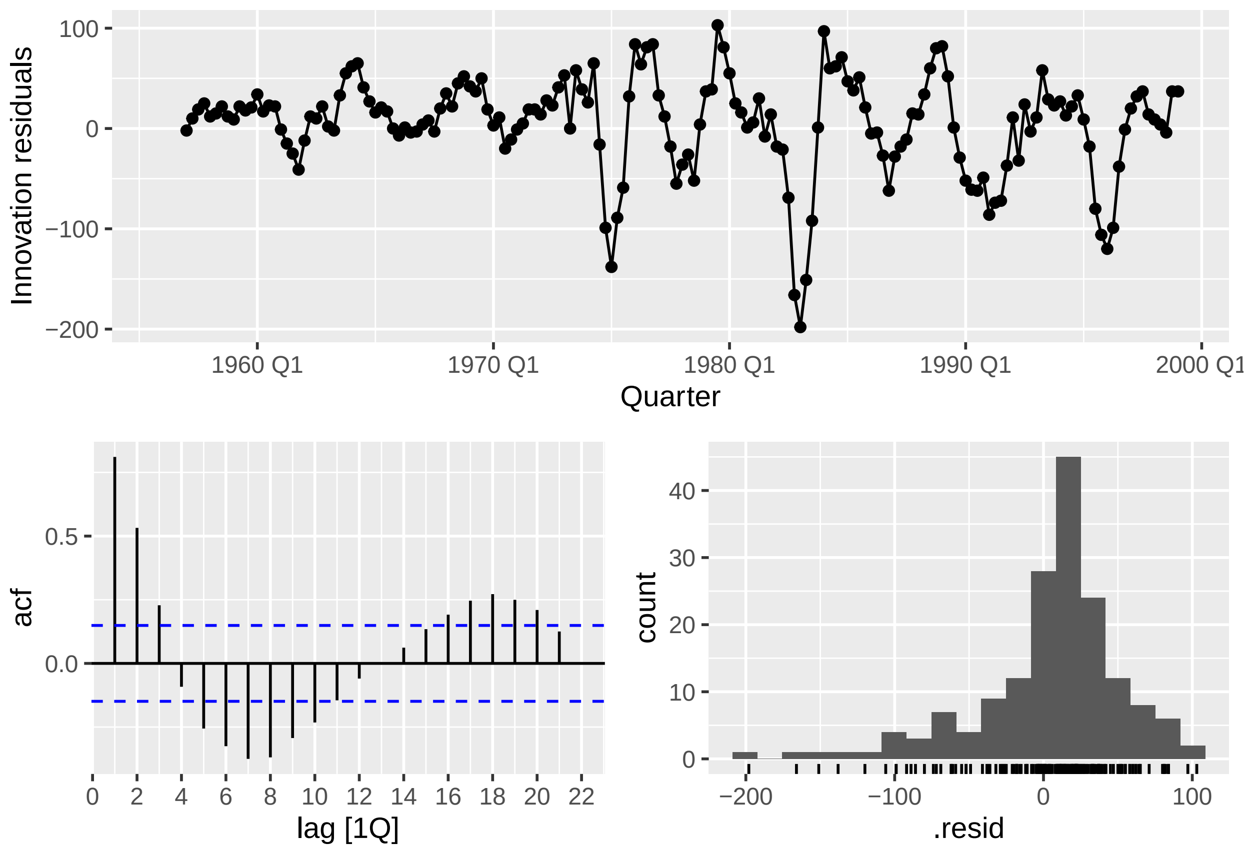
With this, we can conclude that forecasts built using this method will likely be pretty good.

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

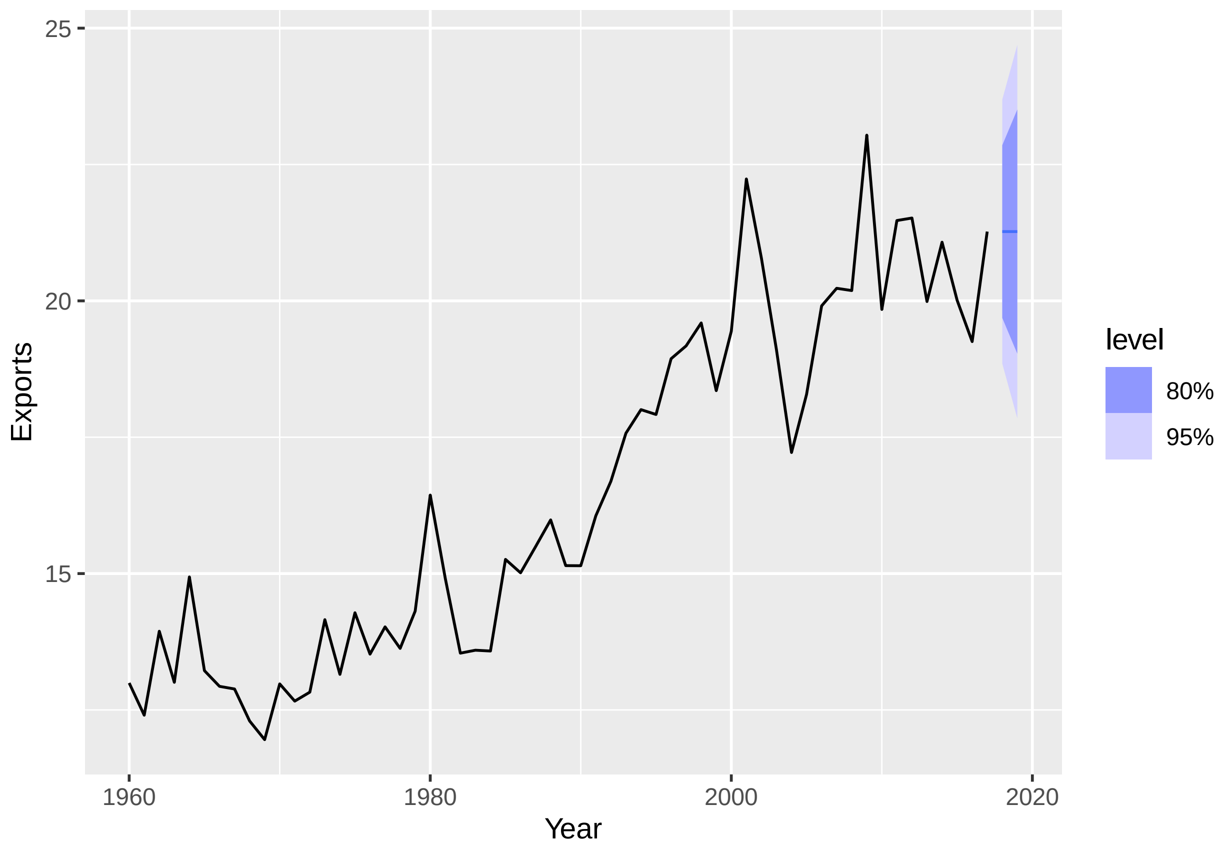
```
## Warning: Removed 4 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

```
## 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
aus_exports_fit |>
  forecast() |>
  autoplot(aus_exports)
```



From the graphs above we can see that the distribution of residuals is left skewed. Additionally, it doesn't seem that our mean of residuals is close to 0. Lastly, the ACF is showing a sinusoidal pattern which indicates that the data is autocorrelated with its lags.

Question 5.7

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

```
set.seed(2111994)

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

head(myseries)

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

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

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

B. Check that your data have been split appropriately by producing the following plot. The data does seem split by a pre and post 2011..

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

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

D. Check the residuals. Do the residuals appear to be uncorrelated and normally distributtionted?

```r
fit |> gg_tsresiduals()
```

These residuals seem highly correlated as we can see from the acf chart and they do have a bit of a normal distribution to them as we can see in the distribution of residuals. Also, the mean of the innovation residuals does not seem to be near 0.

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

fc |> autoplot(myseries)
```

E. Produce forecasts for the test data

```r
fit |> accuracy()
fc |> accuracy(myseries)
```

F. Compare the accuracy of your forecasts against the actual values. Observing the accuracy, we can see that the MAPE on the training dataset is 10.2 while it is 18.8 for the test dataset. This indicates to me that this model is particularly bad on new data. Because the model isn't very good at predicting on new data, it could be overfit to the training dataset.

G. How sensitive are the accuracy measures to the amount of training data used? The accuracy measures are very sensitive to the amount of training data as the more training data will typically increase the accuracy of the model built. Although, especially with timeseries forecasting, there can be a point of reversal as there can be trends that aren't accounted for or larger, more foundational shifts which the model may not pick up. Additionally, and as alluded to in the last question, there is a chance of overfitting which is a greater risk the more data that is input. Conversely, with too little data the model will not have enough of an opportunity to build the model and train itself leading to an underfit model.