# Homework 3

Nick Climaco

February 14, 2024

## Table of contents

# Chapter 5: The Forecaster's Toolbox

```
# data wrangling
import pandas as pd
import numpy as np

# data visuals
import matplotlib.pyplot as plt
import seaborn as sns

# timeseries analysis
from darts import TimeSeries

# ts decomposition
from statsmodels.tsa.seasonal import STL
```

## Exercise 1

Produce forecasts for the following series using whichever of NAIVE(y), SNAIVE(y) or RW(y ~ drift()) is more appropriate in each case:

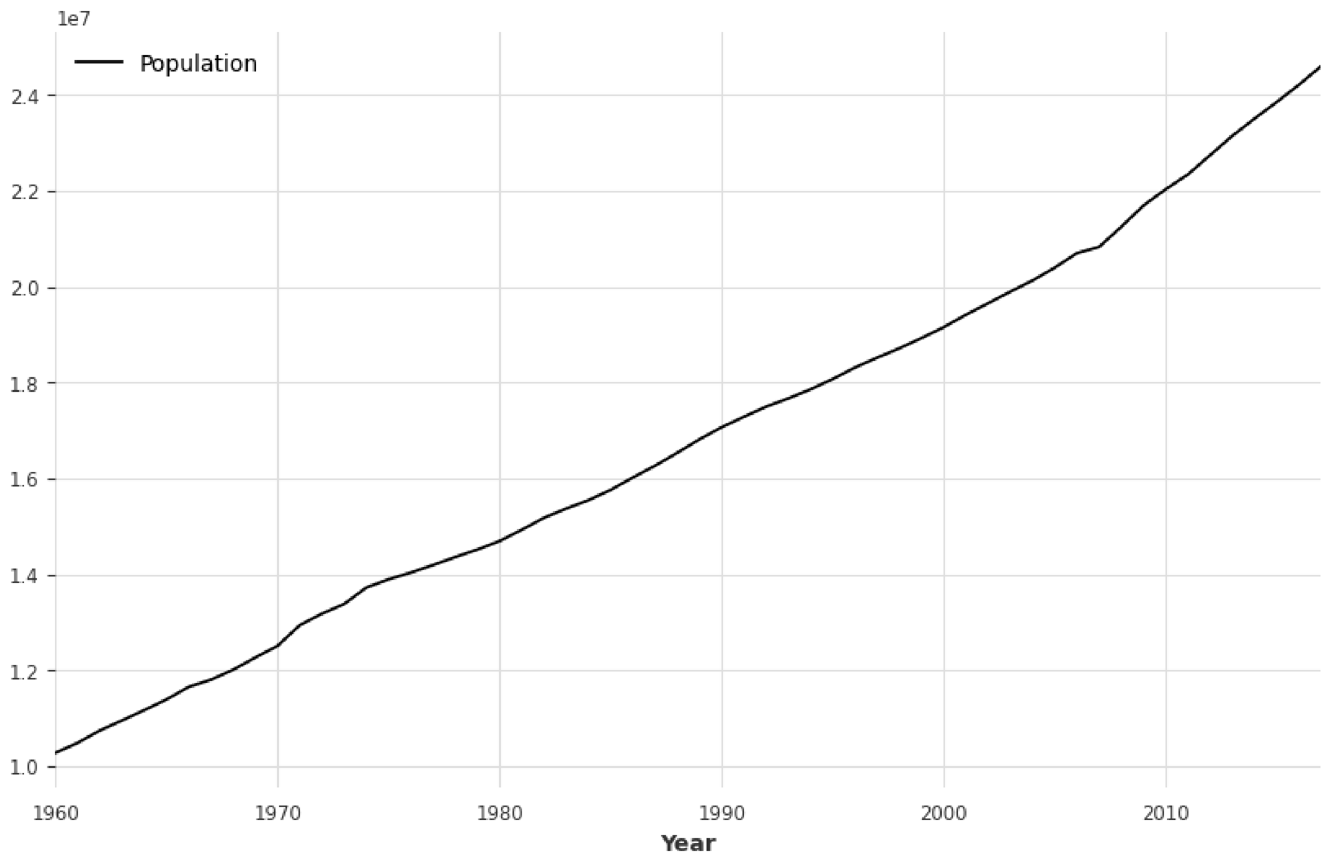
- Australian Population (global_economy)
- Bricks (aus_production)
- NSW Lambs (aus_livestock)
- Household wealth (hh_budget).
- Australian takeaway food turnover (aus_retail).

```
# reading in the data
df_global_economy = pd.read_csv("../rdata/global_economy.csv", parse_dates=['Year'])
df_production= pd.read_csv("../rdata/aus_production.csv", parse_dates=['Quarter'])
df_livestock = pd.read_csv("../rdata/aus_livestock.csv", parse_dates=['Month'])
df_budget = pd.read_csv("../rdata/hh_budget.csv", parse_dates=['Year'])
df_retail = pd.read_csv("../rdata/hh_budget.csv",parse_dates=['Year'])
```

### Australian Population from `global_economy`

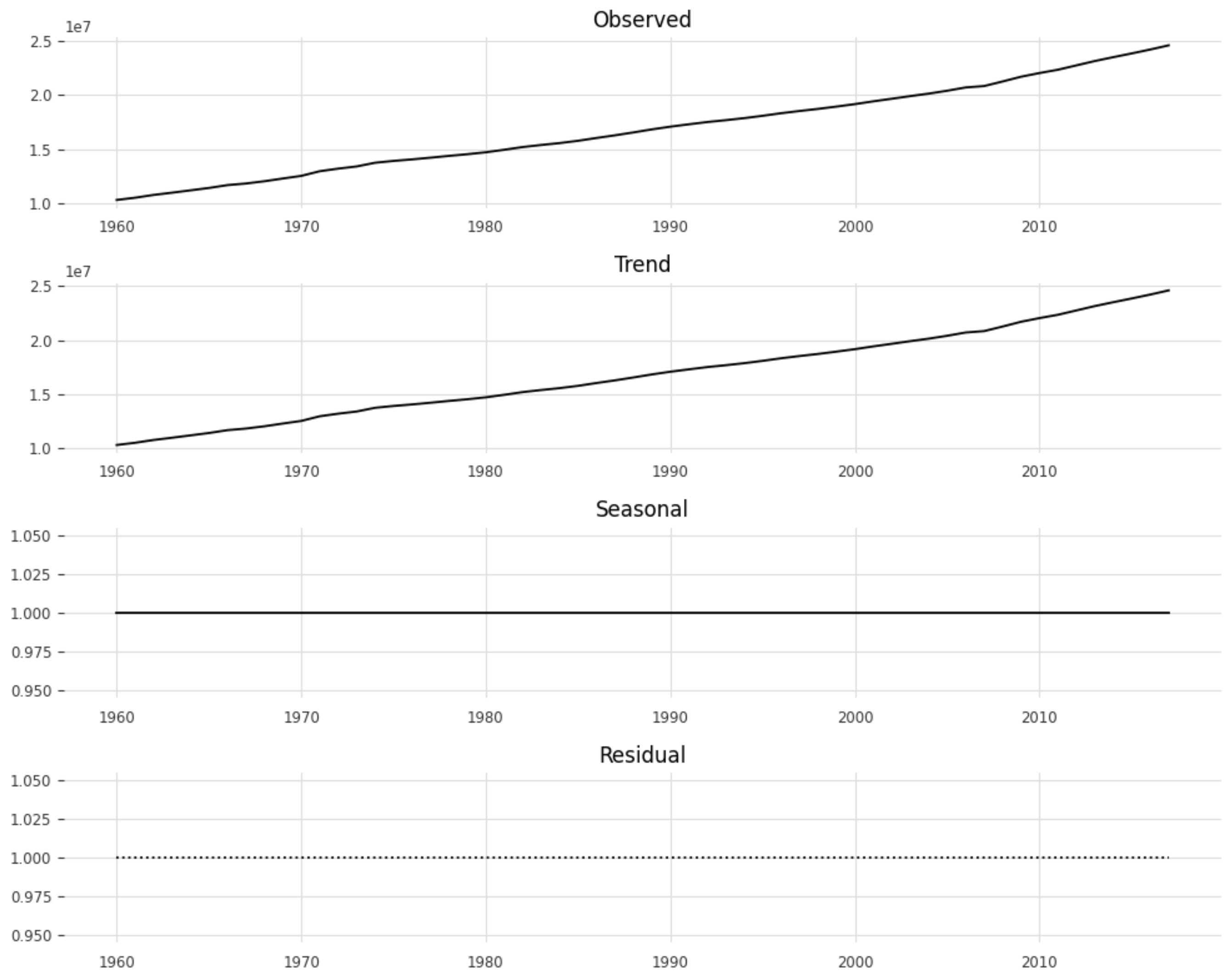
```
# filter australia
df_aus = df_global_economy.query('Country == "Australia"')
```

```
# plot australian timeseries pop
df_aus_pop = df_aus[['Year', 'Population']]
df_aus_pop.set_index('Year', inplace=True)
df_aus_pop.plot()
```



The Australian Populatiion has near perfect linear growth from the 1960 to 2020. Before, choosing which forecasting method to go with. I would like to see the decomposition of this time series just to make sure that my initial observation of no seasonality is correct.

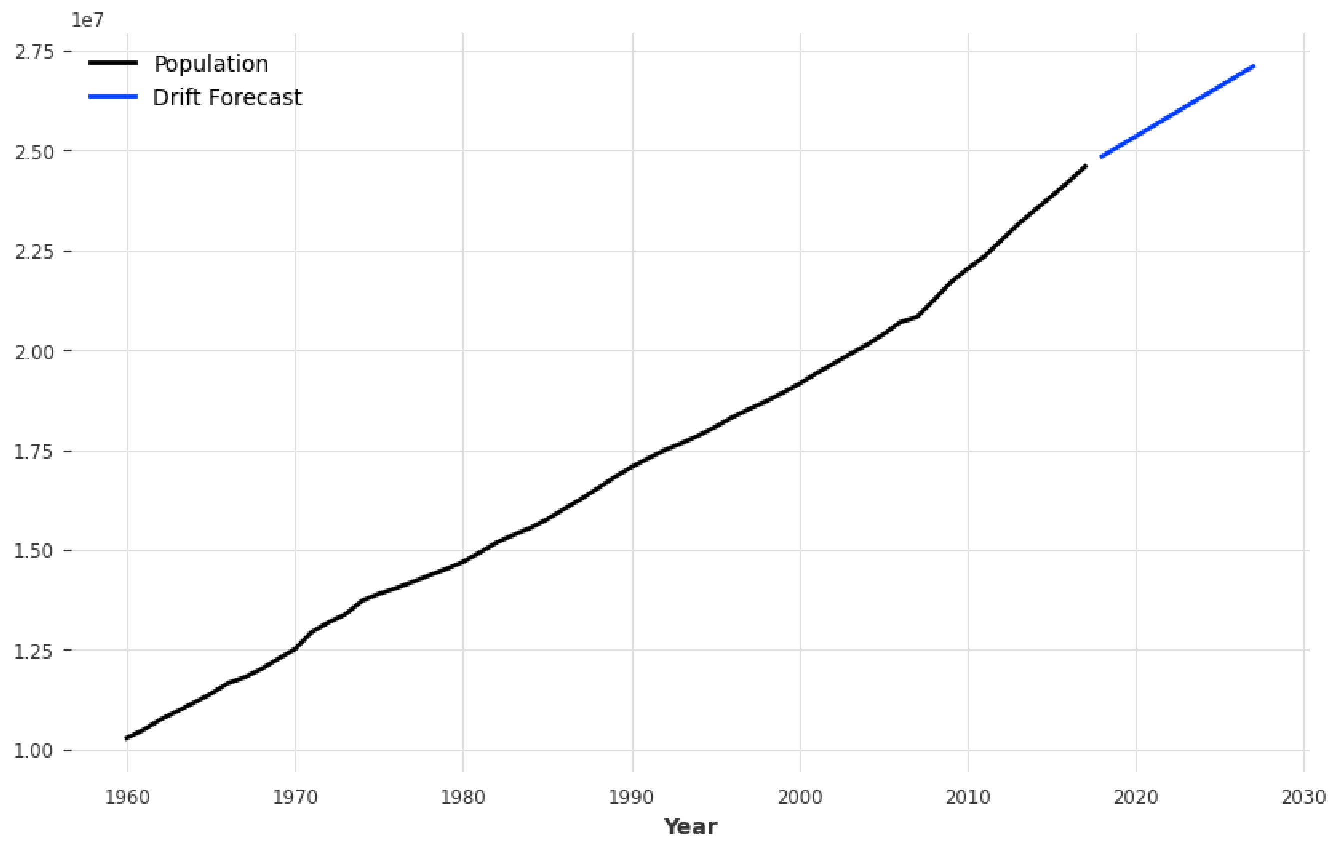
```python
from statsmodels.tsa.seasonal import seasonal_decompose

# decompose this ts
decomposition = seasonal_decompose(df_aus_pop.Population, period = 1,
  model="multiplicative")

fig, (ax1, ax2, ax3, ax4) = plt.subplots(4, 1, figsize=(10, 8))

ax1.plot(decomposition.observed)
ax1.set_title('Observed')

ax2.plot(decomposition.trend)
ax2.set_title('Trend')

ax3.plot(decomposition.seasonal)
ax3.set_title('Seasonal')

ax4.plot(decomposition.resid, linestyle = "dotted", markersize = 10)
ax4.set_title('Residual')

plt.tight_layout()
plt.show()
```



The trend line captures almost all of the observed data indicated that the seasonal component is constant at a value of 1. Thus, I believe that using the Drift method would be the most appropriate for this timeseries since it exhibits no seasonality and a overal upward trend. Wherein our forecast cast would be the average change seen in the data.

```
# convert df to ts
series = TimeSeries.from_dataframe(df_aus,'Year', 'Population')
```

```
from darts.models import NaiveDrift

drift = NaiveDrift()

drift.fit(series)

forecast = drift.predict(10) # 10 timesteps
series.plot()
forecast.plot(label='Drift Forecast', low_quantile = 0.05, high_quantile=0.95)
plt.legend()
```

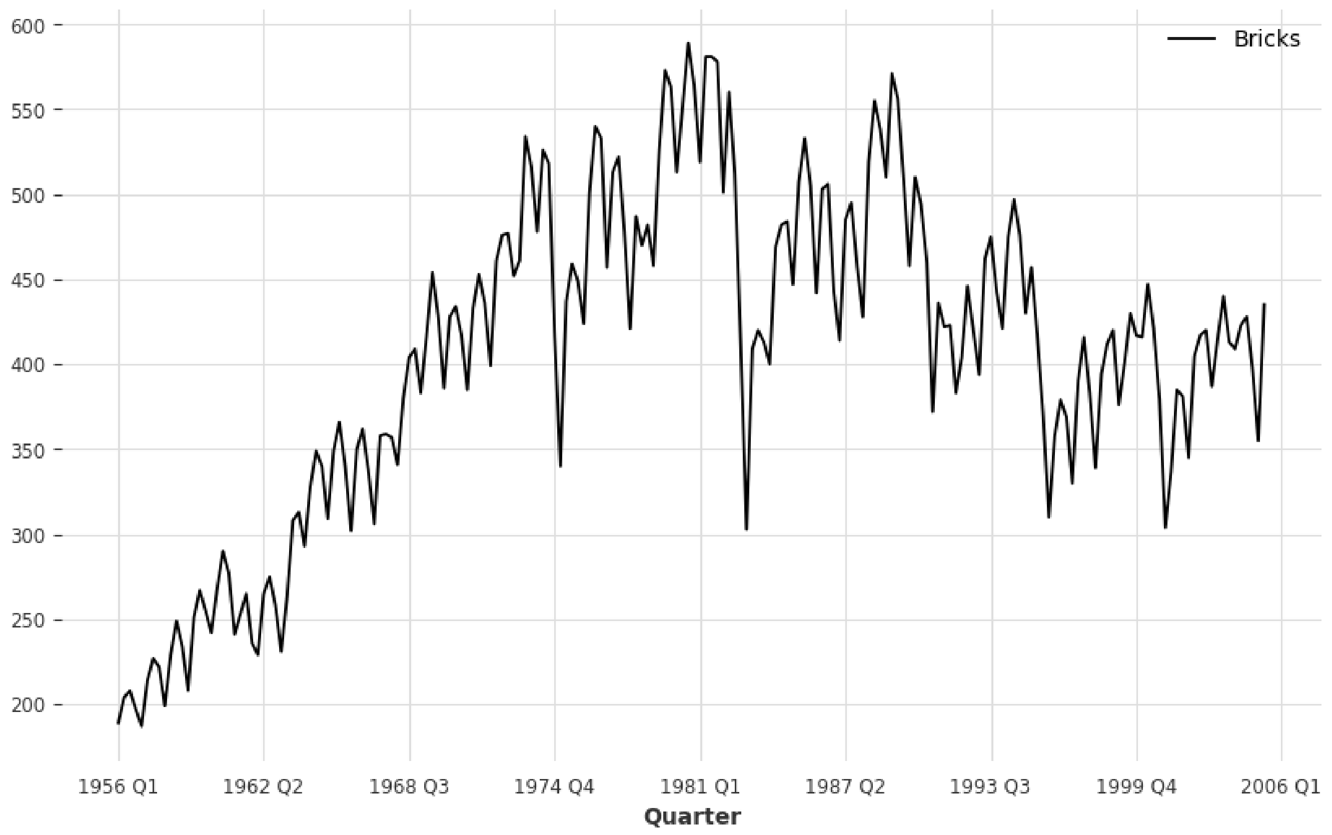


From the eye test, we observe that the Drift forecast is in-line with the data. It reasonable to agree with the forecast since the timeseries demonstrates an linear growth in population.

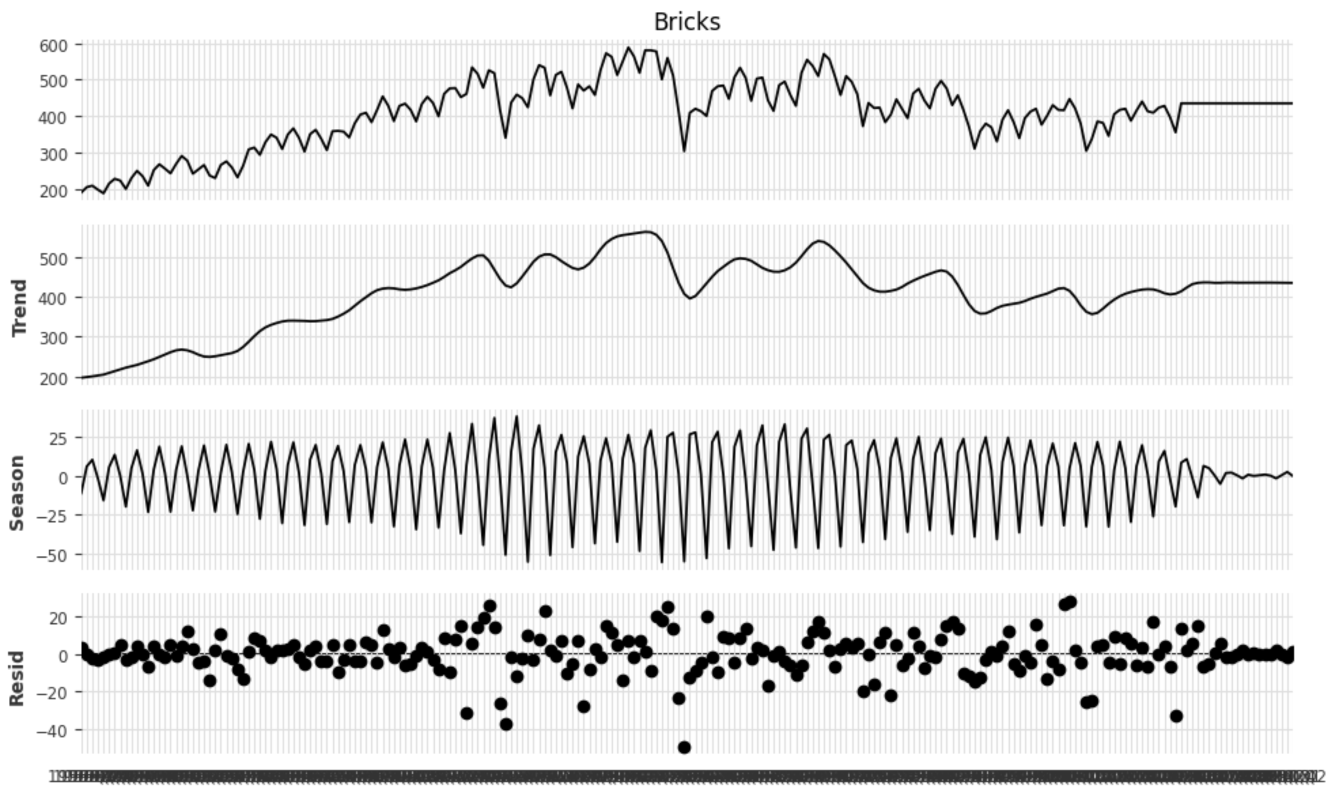
**Bricks from `aus_production`**

```
df_bricks = df_production[['Quarter', 'Bricks']]
```

```
df_bricks.set_index('Quarter').plot()
```



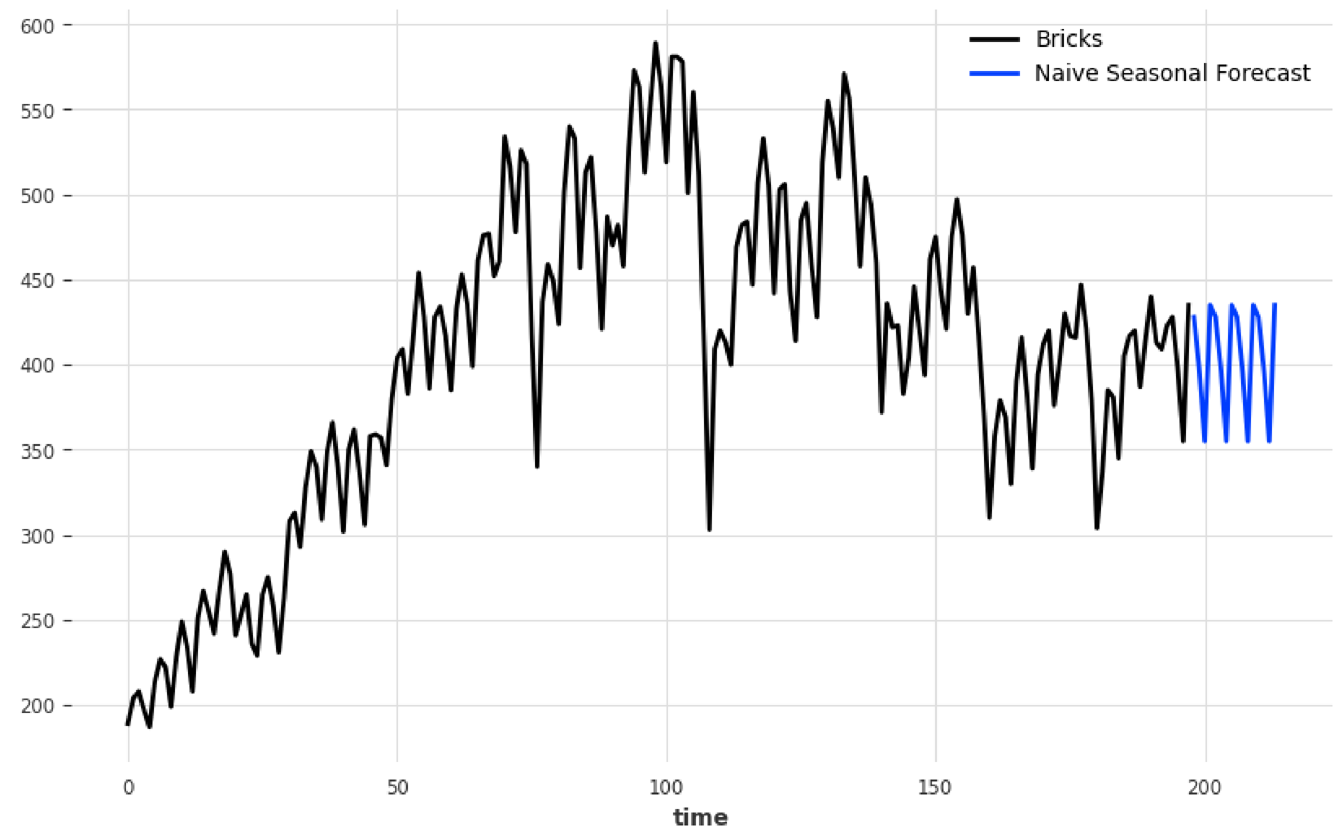
```
bricks_plot = df_bricks.set_index('Quarter')
bricks_plot['Bricks'] = bricks_plot['Bricks'].ffill()

decom = STL(bricks_plot['Bricks'], period = 4).fit()
decom.plot()
plt.show()
```



Since the data

```
df_bricks = df_bricks.dropna() # missing a data at the tail()

# format datetime to accomodate the input for darts library
df_bricks['Quarter'] = pd.to_datetime(df_bricks['Quarter'].astype(str), format='%Y Q%m')

df_bricks.set_index('Quarter')
```

```
from darts.models import NaiveSeasonal

# remove time col since it is set as the index
series = TimeSeries.from_dataframe(df_bricks, value_cols='Bricks',
  fill_missing_dates=True)

seasonal = NaiveSeasonal(K=4) # K=4 for quarterly data

seasonal.fit(series)

forecast = seasonal.predict(16)

series.plot()
forecast.plot(label='Naive Seasonal Forecast')
plt.legend()
```



**NSW Lambs from `aus_livestock`**

```
display(df_livestock.State.unique())
display(df_livestock.Animal.unique())
```

```
array(['Australian Capital Territory', 'New South Wales',
       'Northern Territory', 'Queensland', 'South Australia', 'Tasmania',
       'Victoria', 'Western Australia'], dtype=object)
```

```
array(['Bulls, bullocks and steers', 'Calves', 'Cattle (excl. calves)',
       'Cows and heifers', 'Lambs', 'Pigs', 'Sheep'], dtype=object)
```

Assuming NSW means New South Wales. So, we need to filter `df_livestock` of sheep from New South Wales.

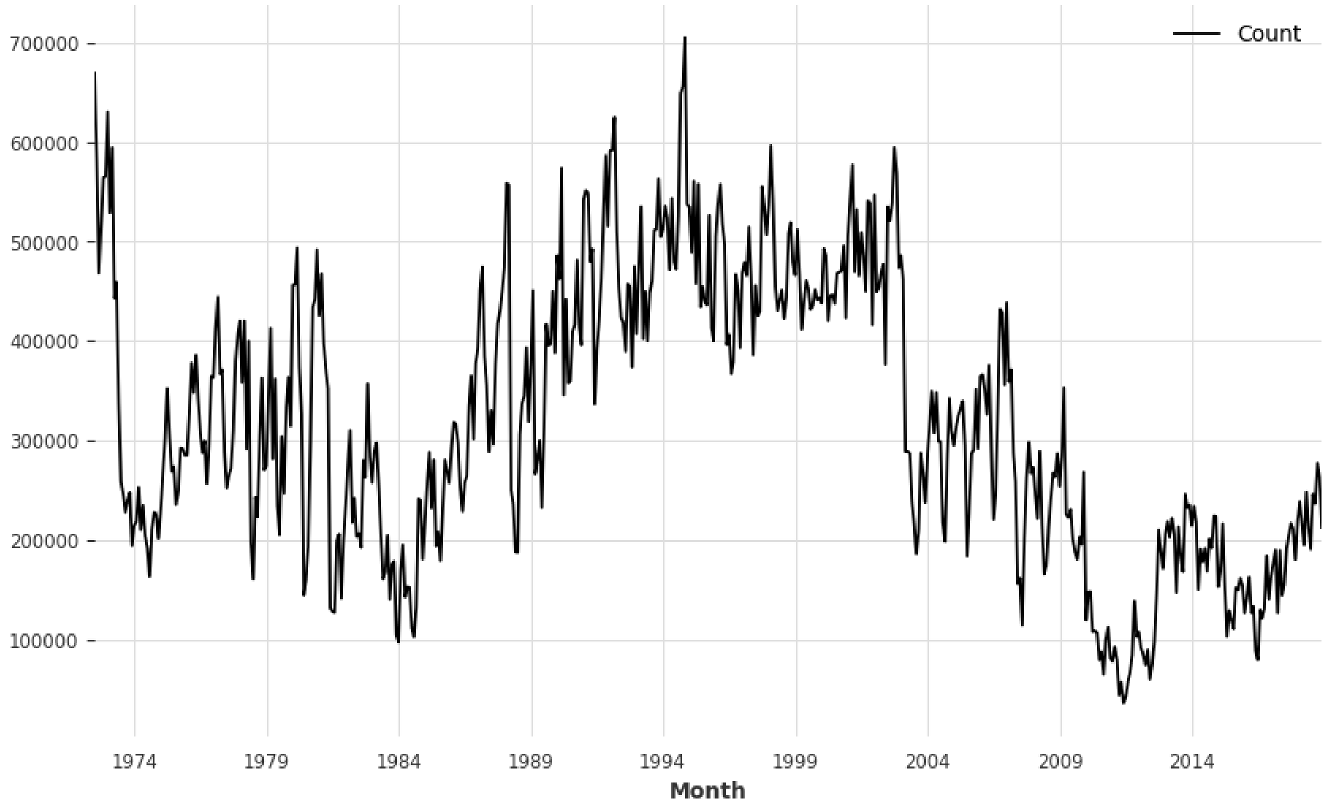
```
df_sheep = df_livestock.query('Animal == "Sheep" & State == "New South Wales"')
df_sheep
```

| | Unnamed: 0 | Month | Animal | State | Count |
|---|---|---|---|---|---|
| 25458 | 25459 | 1972-07-01 | Sheep | New South Wales | 669400.0 |
| 25459 | 25460 | 1972-08-01 | Sheep | New South Wales | 581100.0 |
| 25460 | 25461 | 1972-09-01 | Sheep | New South Wales | 468100.0 |
| 25461 | 25462 | 1972-10-01 | Sheep | New South Wales | 515300.0 |
| 25462 | 25463 | 1972-11-01 | Sheep | New South Wales | 564500.0 |
| … | … | … | … | … | … |
| 26011 | 26012 | 2018-08-01 | Sheep | New South Wales | 245900.0 |

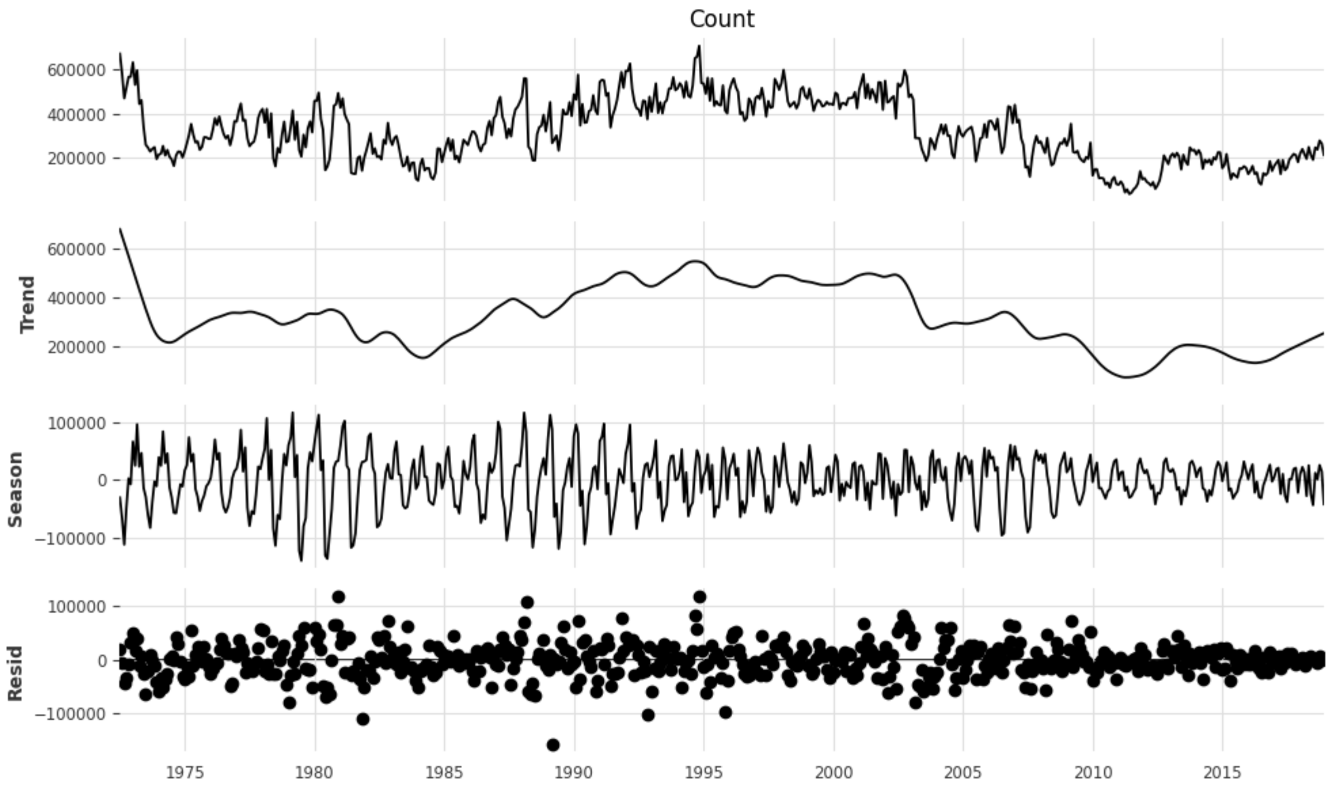
| | Unnamed: 0 | Month | Animal | State | Count |
|---|---|---|---|---|---|
| 26012 | 26013 | 2018-09-01 | Sheep | New South Wales | 236800.0 |
| 26013 | 26014 | 2018-10-01 | Sheep | New South Wales | 277200.0 |
| 26014 | 26015 | 2018-11-01 | Sheep | New South Wales | 263600.0 |
| 26015 | 26016 | 2018-12-01 | Sheep | New South Wales | 212300.0 |

Next, we want to see the timeseries.

```
df_sheep[['Month', 'Count']].set_index(['Month']).plot()
```

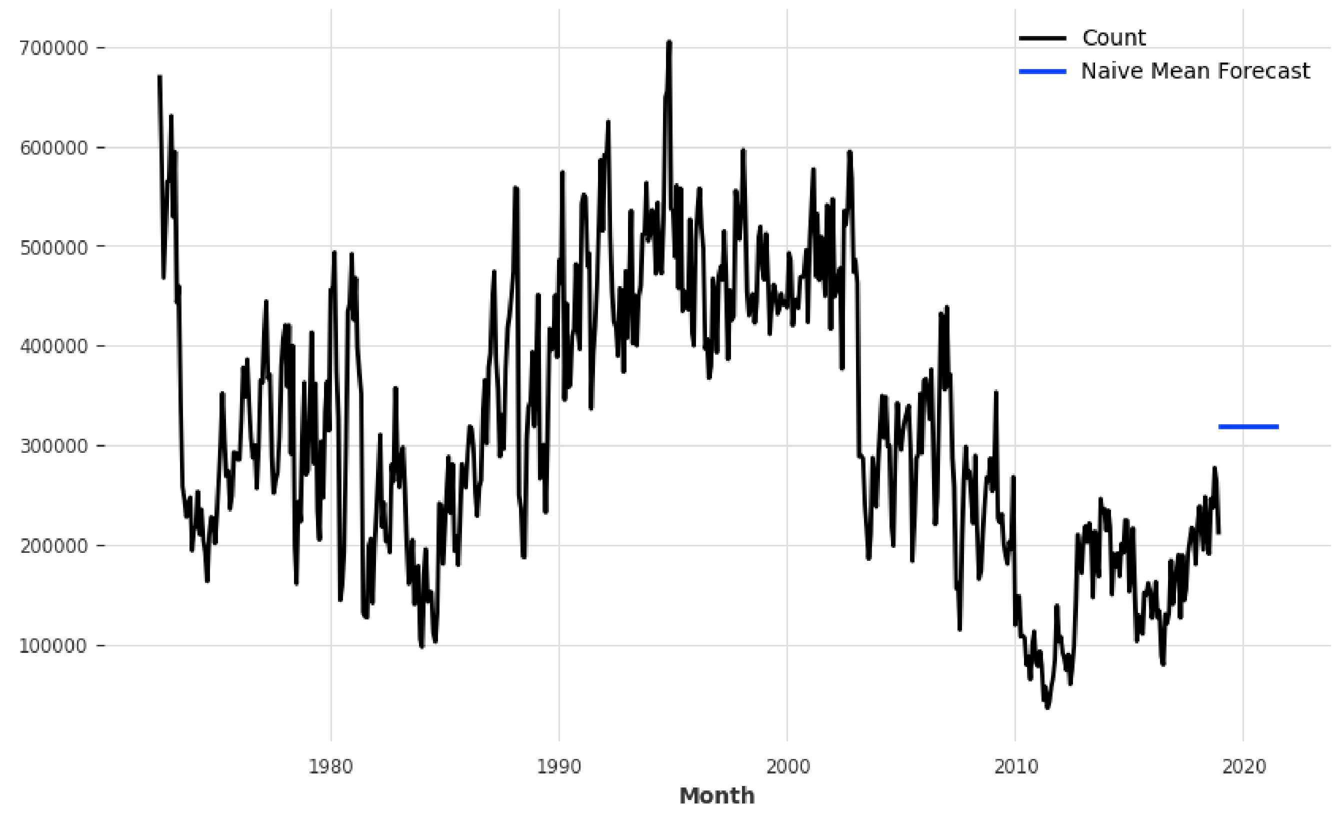


```
sheep = df_sheep.set_index('Month')
decomposition = STL(sheep['Count']).fit()
decomposition.plot()
plt.show()
```



We see that this time seres has no clear trend and inconsistent seasonality component which suggest that we use the Mean Naive method.

```
from darts.models import NaiveMean

series = TimeSeries.from_dataframe(df_sheep, 'Month', 'Count')

model = NaiveMean()
model.fit(series)
forecast = model.predict(30) # 30 timesteps in the future in this case 30 months

series.plot()
forecast.plot(label = 'Naive Mean Forecast')
plt.legend()
```

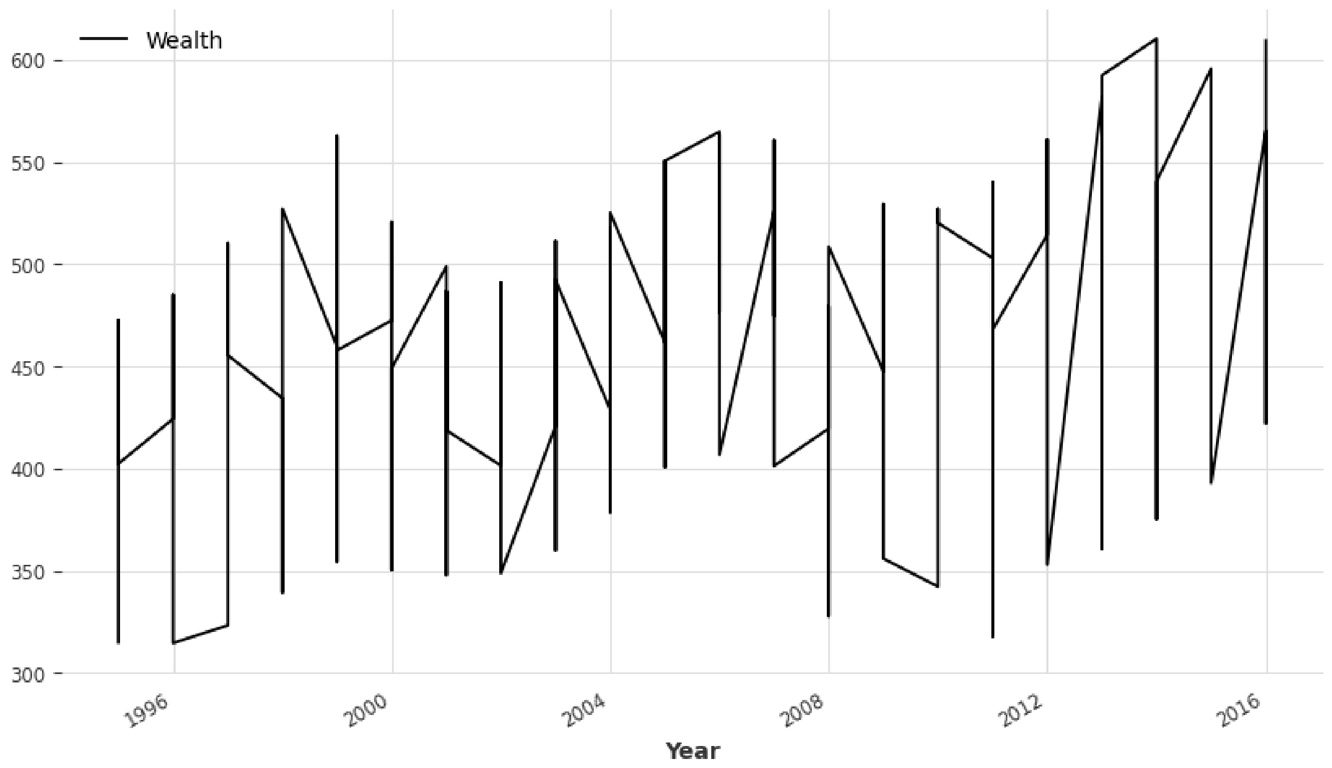


**Household Wealth from `hh_budget`**

```
df_wealth = df_budget[['Year', 'Wealth']]
df_wealth
```

| | Year | Wealth |
|---|---|---|
| 0 | 1995-01-01 | 314.9344 |
| 1 | 1996-01-01 | 314.5559 |
| 2 | 1997-01-01 | 323.2357 |
| 3 | 1998-01-01 | 339.3139 |
| 4 | 1999-01-01 | 354.4382 |
| ... | ... | ... |
| 83 | 2012-01-01 | 514.4276 |
| 84 | 2013-01-01 | 592.3568 |
| 85 | 2014-01-01 | 596.4713 |
| 86 | 2015-01-01 | 588.1454 |
| 87 | 2016-01-01 | 609.2657 |

```
wealth = df_wealth
wealth = wealth.set_index(['Year'])

wealth.plot()
```



**Australian takeaway food turnover from `aus_retail`**

```
Index(['Unnamed: 0', 'Country', 'Year', 'Debt', 'DI', 'Expenditure', 'Savings',
       'Wealth', 'Unemployment'],
      dtype='object')
```

### Exercise 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?

### Exercise 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?

### Exercise 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

### Exercise 7

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

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.

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

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

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

d. Check the residuals.

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

Do the residuals appear to be uncorrelated and normally distributed?

e.Produce forecasts for the test data

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

f. Compare the accuracy of your forecasts against the actual values.

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

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