Table of Contents

[References 1](#_Toc51848778)

[Time Series Decomposition 1](#_Toc51848779)

[Trend 2](#_Toc51848780)

[Seasonality 2](#_Toc51848781)

[Cyclic Behaviour 2](#_Toc51848782)

[Classical Decomposition of a Time Series 3](#_Toc51848783)

[Multiplicative Decomposition 3](#_Toc51848784)

[Additive Decomposition 5](#_Toc51848785)

[Comparing Additive and Multiplicative Decomposition 6](#_Toc51848786)

[Seasonal Detail 7](#_Toc51848787)

[De-Trending 9](#_Toc51848788)

[De-Seasonalizing Data 10](#_Toc51848789)

# References

Some of the content in this document is based on the following articles.

<https://otexts.com/fpp2/classical-decomposition.html>

<https://www.machinelearningplus.com/time-series/time-series-analysis-python/>

# Time Series Decomposition

Over the next few weeks we will be building univariate time series models. Time series data will often exhibit trend, seasonal and cyclic behaviour. However, recurrent components cannot always be modelled as easily with the types of models that we will be building.  With that in mind, today we will break down time series data into recurrent and error components to better understand how to analyze and prepare the data for modelling.

The statsmodels library provides an implementation of the naive, or classical, decomposition method in a function called seasonal\_decompose(). It requires that you specify whether the model is additive or multiplicative.

**Caution:**

Healthy skepticism is needed when using automated decomposition methods. It is important to treat decomposition as a potentially useful analysis tool, but also to consider exploring the data in many alternate ways.

### Trend

A trend is observed when there is an increasing or decreasing slope in the time series.

### Seasonality

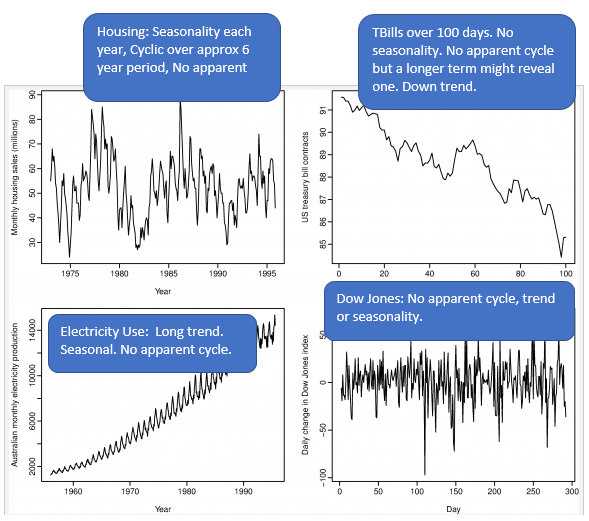
When seasonality is observed there is a distinct repeated pattern observed between regular calendar intervals that occur due to seasonal factors.

### Cyclic Behaviour

Cyclic behaviour occurs when a rise and fall pattern occurs at fixed calendar-based intervals. Care should be taken to not confuse ‘cyclic’ effect with ‘seasonal’ effect. Unlike the seasonality, cyclic effects are typically influenced by the business and other socio-economic factors.

Figure 1 illustrates four different combinations of cyclic, seasonal and trend behaviour.

Figure : Varying Combinations of Cyclic, Seasonal, and Trend



## Classical Decomposition of a Time Series

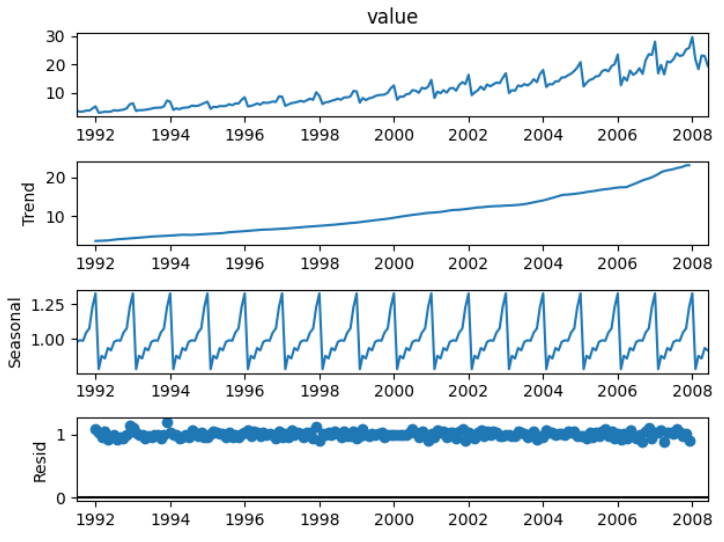
There are two ways to decompose a time series:

1. Multiplicative Decomposition.
2. Additive decomposition.

### Multiplicative Decomposition

This section shows how to decompose time series data into separate components by using multiplicative decomposition. Figure 2 shows visualizations of the separate components.

Figure : Multiplicative Decomposition of Drug Sales



Better for multiplicative because there is no pattern. Eliminating for residiuals is better!

Example : Multiplicative Decomposition Visualization

This code is used to plot the multiplicative series that is shown in Figure 2.

|  |
| --- |
| from statsmodels.tsa.seasonal import seasonal\_decompose  import pandas as pd  import matplotlib.pyplot as plt  # Import data.  PATH = "/Users/pm/Desktop/DayDocs/data/"  FILE = "drugSales.csv"  df = pd.read\_csv(PATH + FILE, parse\_dates=['date'], index\_col='date')  type(df.index)  # Perform decomposition using multiplicative decomposition.  tseries = seasonal\_decompose(df['value'], model='multiplicative', extrapolate\_trend="freq")  tseries.plot()  plt.show() |

Example : Numeric Multiplicative Decomposition

This example helps to explain how the different time steps are represented through multiplicative decomposition. To build this example add this code to the end of Example 1.

|  |
| --- |
| # Extract the Components ----  # Actual Values = Product of (Seasonal \* Trend \* Resid)  dfComponents = pd.concat([tseries.seasonal, tseries.trend,  tseries.resid, tseries.observed], axis=1)  dfComponents.columns = ['seas', 'trend', 'resid', 'actual\_values']  print(dfComponents.head()) |

The output in Table 1identifies the different numeric components of the actual drug sales totals for each time step. All cell values in each row can be multiplied together to calculate the actual value.

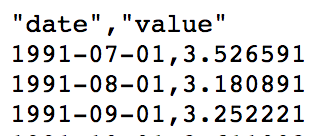
**0.987845\*3.060085\*1.166629 = 3*.526591***

Table : Seasonal, Trend, and Error Components for Multiplicative Decomposition

|  |
| --- |
| seas trend resid actual\_values  date  1991-07-01 0.987845 3.060085 1.166629 3.526591  1991-08-01 0.990481 3.124765 1.027745 **3.180891**  1991-09-01 0.987476 3.189445 1.032615 3.252221  1991-10-01 1.048329 3.254125 1.058513 3.611003  1991-11-01 1.074527 3.318805 0.999923 3.565869 |

Exercise (2 marks)

Write a tiny Python program to multiply the seasonal, trend and residual (error) components to calculate and print the second value in the time series of **3.180891**. Show your program here:



|  |
| --- |
| from statsmodels.tsa.seasonal import seasonal\_decompose  import pandas as pd  import matplotlib.pyplot as plt  # Import data.  PATH = "/Users/hyerimshin/PycharmProjects/BigData/datasets/"  FILE = "drugSales.csv"  df = pd.read\_csv(PATH + FILE, parse\_dates=['date'], index\_col='date')  type(df.index)  # Perform decomposition using multiplicative decomposition.  tseries = seasonal\_decompose(df['value'], model='multiplicative', extrapolate\_trend="freq")  tseries.plot()  plt.show()  # Extract the Components ----  # Actual Values = Product of (Seasonal \* Trend \* Resid)  dfComponents = pd.concat([tseries.observed], axis=1)  dfComponents.columns = ['actual\_values']  print(dfComponents.head()) |

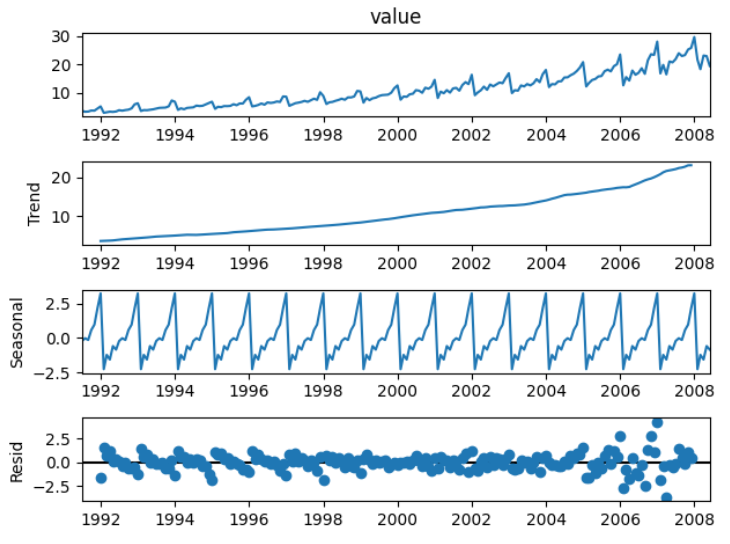
## Additive Decomposition

The process for implementing additive decomposition is almost identical to multiplicative decomposition except the seasonal, trend and error components are added together to form the

Example : Additive Decomposition

Here are the seasonal, trend and error components that are extracted using additive decomposition. Figure 3 shows the additive components.

Figure : Additive Decomposition Visualization



To build this example change the parameter in Example 1 to additive.

|  |
| --- |
| tseries = seasonal\_decompose(df['value'], model='additive', extrapolate\_trend="freq") |

The actual drug sales values are calculated by adding the seasonal, trend and error components together.

**-0.140765+3.060085+0.607271= 3*.526591***

|  |
| --- |
| seas trend resid actual\_values  date  1991-07-01 -0.140765 3.060085 0.607271 3.526591  1991-08-01 0.027747 3.124765 0.028379 3.180891  1991-09-01 -0.090361 3.189445 0.153137 3.252221  1991-10-01 0.602876 3.254125 -0.245998 3.611003  1991-11-01 0.970698 3.318805 -0.723634 3.565869 |

### Comparing Additive and Multiplicative Decomposition

The multiplicative and additive diagrams are similar. However, the error component in the additive decomposition graph at the right of Figure 4 shows some pattern remains. The remaining pattern in the residual component of additive decomposition suggests that multiplicative decomposition is slightly better at decomposing the series. Ideally, we want to make the residual component completely random.

Figure : Comparing Multiplicative and Additive Decomposition

|  |  |
| --- | --- |
| **Multiplicative Decomposition** | **Additive Decomposition** |
|  |  |

## Seasonal Detail

It is further possible to examine seasonal date in detail. After decomposing a series, you can easily isolate any of the different decomposed components:

trend = tseries.trend

seasonal = tseries.seasonal

error = tseries.resid

Example : Plotting Seasonal Detail

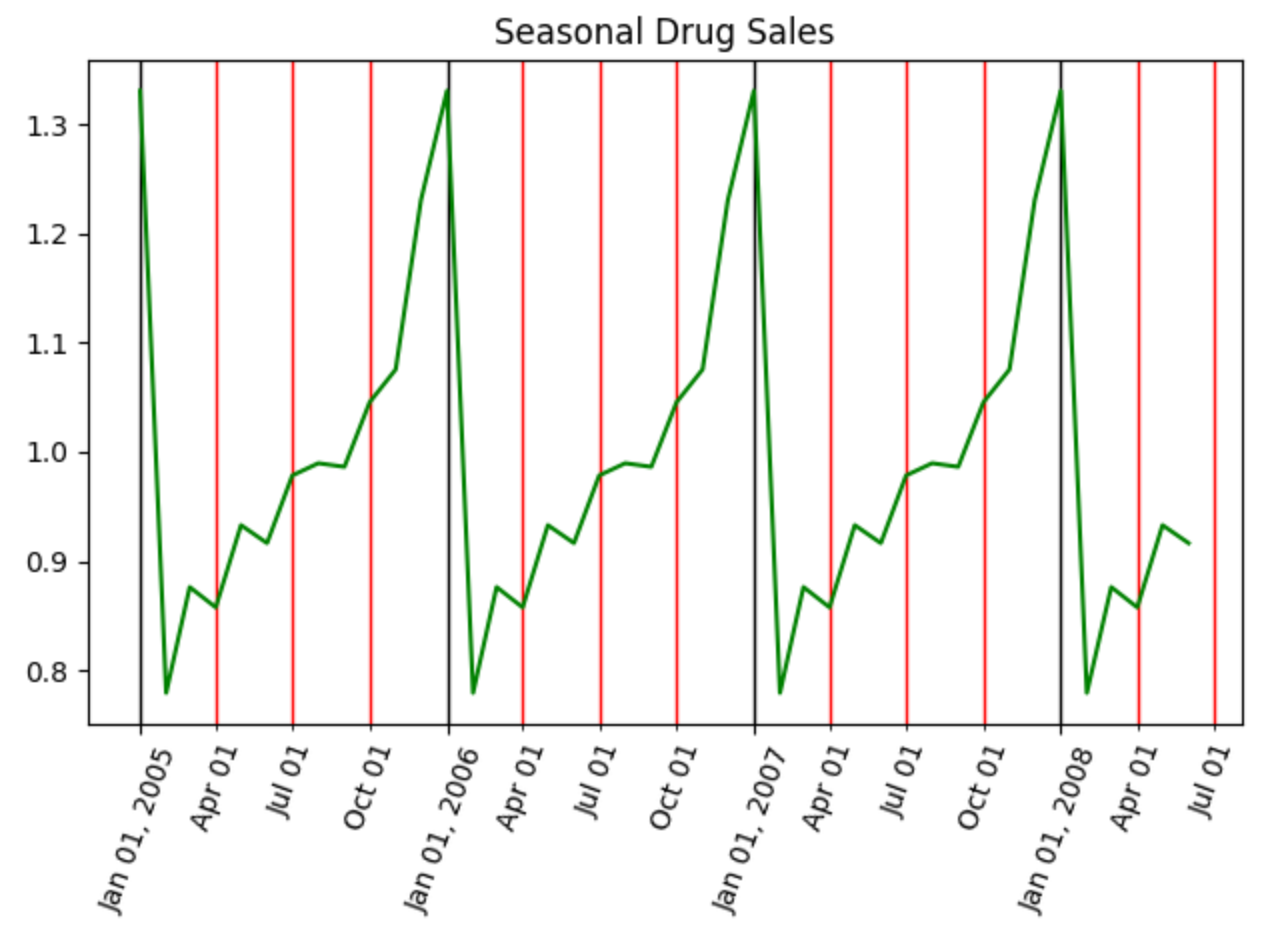
To avoid too much clutter, this example plots seasonal detail starting from January 2005:

start, end = '2005-01', '2009-12'

ax.plot(seasonal.loc[start:end], color='green')

Techniques from the lab are used to plot the seasonal data in Figure 5. It is interesting to note how drug sales spike in the December holiday season. Try not to read too much into this though since I believe the data is fictitious.

Figure : Seasonal Drug Sales Between January 1, 2005 to July 2008.



|  |
| --- |
| from statsmodels.tsa.seasonal import seasonal\_decompose  import pandas as pd  import matplotlib.pyplot as plt  import matplotlib.dates as mdates  # Import data.  PATH = "/Users/pm/Desktop/DayDocs/data/"  FILE = "drugSales.csv"  df = pd.read\_csv(PATH + FILE, parse\_dates=['date'], index\_col='date')  type(df.index)  fig, ax = plt.subplots()  # Perform decomposition using multiplicative decomposition.  tseries = seasonal\_decompose(df['value'], model='multiplicative',  extrapolate\_trend='freq')  trend = tseries.trend  seasonal = tseries.seasonal  # Set vertical major grid.  ax.xaxis.set\_major\_locator(mdates.YearLocator(day=1))  ax.xaxis.grid(True, which = 'major', linewidth = 1, color = 'black')  # Set vertical minor grid.  ax.xaxis.set\_minor\_locator(mdates.MonthLocator(bymonth=(1,4,7,10),bymonthday=1))  ax.xaxis.grid(True, which = 'minor', linewidth = 1, color = 'red')  start, end = '2005-01', '2009-12'  ax.plot(seasonal.loc[start:end], color='green')  plt.setp(ax.xaxis.get\_majorticklabels(), rotation=70)  plt.setp(ax.xaxis.get\_minorticklabels(), rotation=70)  ax.xaxis.set\_minor\_formatter(mdates.DateFormatter('%b %d'))  ax.xaxis.set\_major\_formatter(mdates.DateFormatter('%b %d, %Y'))  plt.title("Seasonal Drug Sales")  plt.show() |

Exercise (3 marks)

Perform additive and multiplicative decomposition with the **AirPassengers.csv** file.Show graphs of additive and multiplicative components.

|  |
| --- |
| Addictive  Chart  Description automatically generatedChart  Description automatically generated  Multiplicative |

Which type of decomposition does a better job at breaking down all of the patterns and why?

|  |
| --- |
| Multiplicative is better since it is less fluctuated |

## De-Trending

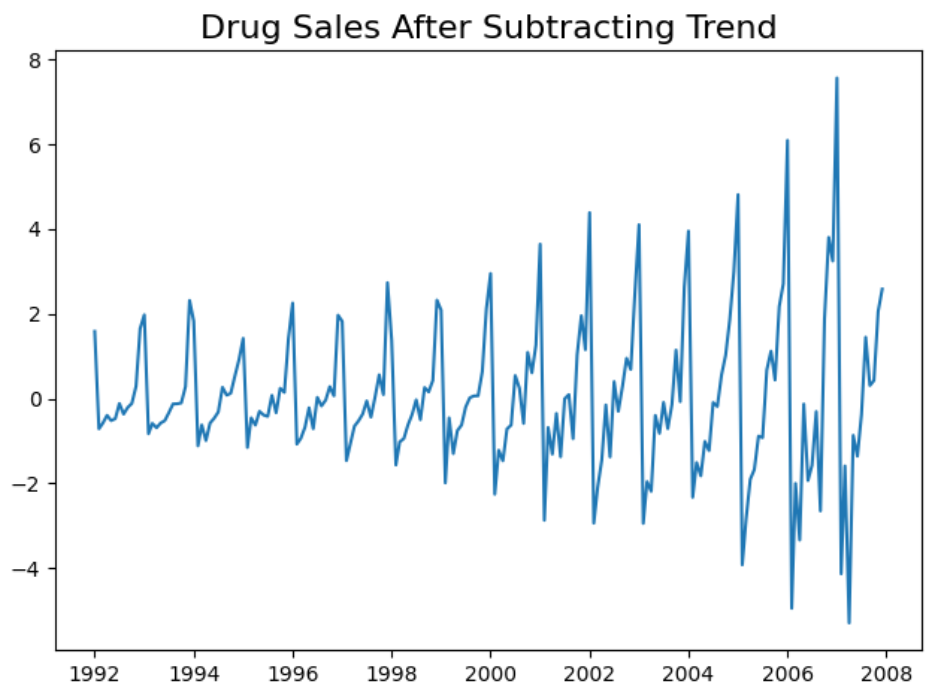
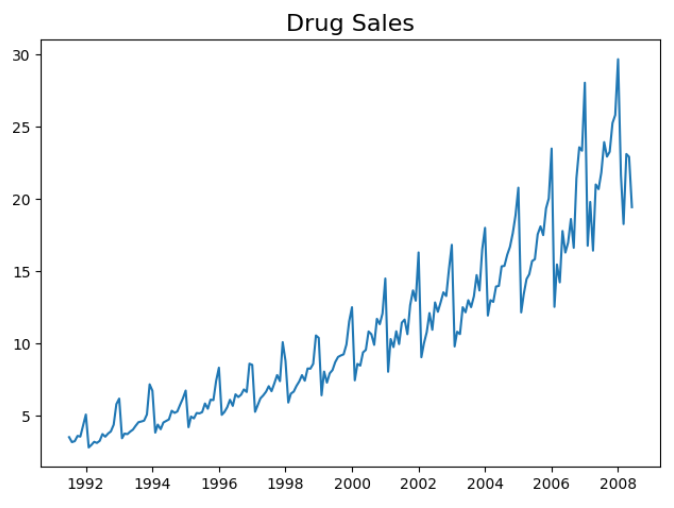
It is possible to remove the trend component from a series. Removing the trend may help to improve the results of a model but not necessarily. Removing a trend may also help to identify relationships between input and target variables. We can use several methods to remove the trend. These methods may include:

1. Subtracting the best fit line from the series.
2. Subtracting the trend component from the series.
3. Other alternatives.

Example : De-Trending by Removing Subtracting the Trend

Subtracting the trend component offers one of the cleanest ways to isolate the seasonal and error data for drug sales. This example shows how to isolate the drug sale data so it can be presented without the trend like the plot at the right of Figure 6.

Figure : Drug Sales with Trend and Without Trend Data



Here is the code which draws plots for drug sale data with and without the trend component.

|  |
| --- |
| from statsmodels.tsa.seasonal import seasonal\_decompose  import pandas as pd  import matplotlib.pyplot as plt  # Import Data  PATH = "/Users/pm/Desktop/DayDocs/data/"  FILE = "drugSales.csv"  df = pd.read\_csv(PATH + FILE, parse\_dates=['date'], index\_col='date')  tseries = seasonal\_decompose(df['value'], model='additive',  extrapolate\_trend='freq')  plt.plot(df['value'])  plt.title("Drug Sales", fontsize=16)  plt.show()  detrended = df['value'] - tseries.trend  plt.plot(detrended)  plt.title('Drug Sales After Subtracting Trend', fontsize=16)  plt.show() |

Exercise (3 marks)

Plot the airline passenger data set output before and after the trend data has been removed.

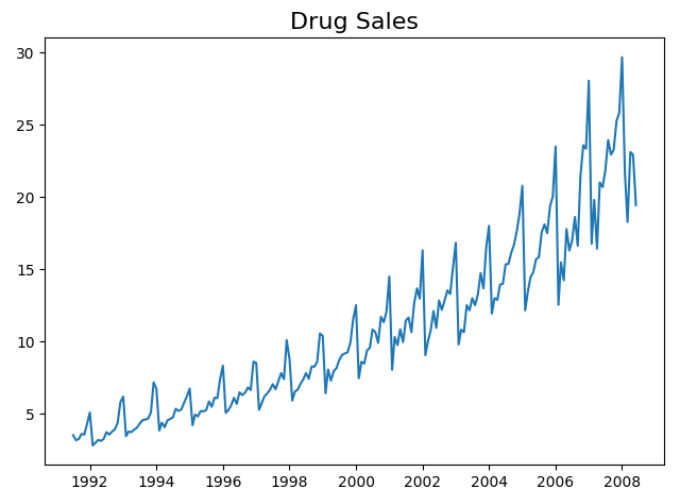
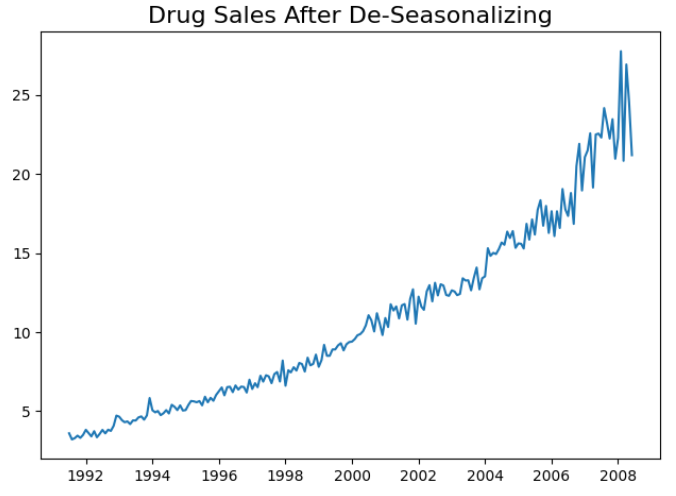
|  |
| --- |
| Chart, line chart, histogram  Description automatically generated |

## De-Seasonalizing Data

As you might expect there are several ways to remove seasonal information from a series. Removing the seasonal component may help to visualize the performance of inputs and outputs along with trend with random environmental shocks.

Example : De-Seasonalizing

This example shows how to remove seasonal data by dividing the drug sales data by the seasonal component vector. The result shows drug sale data without the seasonal fluctuation.

Here is the code:

|  |
| --- |
| from statsmodels.tsa.seasonal import seasonal\_decompose  import pandas as pd  import matplotlib.pyplot as plt  # Import Data  PATH = "/Users/pm/Desktop/DayDocs/data/"  FILE = "drugSales.csv"  df = pd.read\_csv(PATH + FILE, parse\_dates=['date'], index\_col='date')  tseries = seasonal\_decompose(df['value'], model='multiplicative',  extrapolate\_trend='freq')  plt.plot(df['value'])  plt.title("Drug Sales", fontsize=16)  plt.show()  deseasonalized = df.value.values / tseries.seasonal  plt.plot(deseasonalized)  plt.title('Drug Sales After De-Seasonalizing', fontsize=16)  plt.show() |

Exercise (3 marks)

Show the airline passenger data before and after seasonal data has been removed.

|  |
| --- |
| Chart, line chart  Description automatically generated |