Hands-on Time Series Analysis with Python

From Basics to Bleeding Edge Techniques

—

B V Vishwas
Ashish Patel

Apress®

Hands-on Time Series Analysis with Python

From Basics to Bleeding Edge Techniques

B V Vishwas
Ashish Patel

Apress®

Hands-on Time Series Analysis with Python: From Basics to Bleeding Edge Techniques

B V Vishwas
Infosys
Bengaluru, India

Ashish Patel
Cygnet Infotech Pvt Ltd
Ahmedabad, India

ISBN-13 (pbk): 978-1-4842-5991-7
ISBN-13 (electronic): 978-1-4842-5992-4
https://doi.org/10.1007/978-1-4842-5992-4

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Celestin Suresh John
Development Editor: James Markham
Coordinating Editor: Aditee Mirashi

Cover designed by eStudioCalamar

Cover image designed by Pixabay

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this Book is available to readers on GitHub via the Book's product page, located at www.apress.com/978-1-4842-5991-7. For more detailed information, please visit www.apress.com/source-code.

Printed on acid-free paper

Table of Contents

About the Authors

B V Vishwas is a Data Scientist, AI researcher and AI Consultant, Currently living in Bengaluru(INDIA). His highest qualification is Master of Technology in Software Engineering from Birla Institute of Technology & Science, Pilani, India and his primary focus and inspiration is Data Warehousing, Big Data, Data Science (Machine Learning, Deep Learning, Timeseries, Natural Language Processing, Reinforcement Learning, and Operation Research). He has over seven years of IT experience currently working at Infosys as Data Scientist & AI Consultant. He has also worked on Data Migration, Data Profiling, ETL & ELT, OWB, Python, PL/SQL, Unix Shell Scripting, Azure ML Studio, Azure Cognitive Services, and AWS.

Ashish Patel is a Senior Data Scientist, AI researcher, and AI Consultant with over seven years of experience in the field of AI, Currently living in Ahmedabad(INDIA). He has a Master of Engineering Degree from Gujarat Technological University and his keen interest and ambition to research in the following domains such as (Machine Learning, Deep Learning, Time series, Natural Language Processing, Reinforcement Learning, Audio Analytics, Signal Processing, Sensor

Technology, IoT, Computer Vision). He is currently working as Senior Data Scientist for Cynet infotech Pvt Ltd. He has published more than 15 + Research papers in the field of Data Science with Reputed Publications such as IEEE. He holds Rank 3 as a kernel master in Kaggle. Ashish has immense experience working on cross-domain projects involving a wide variety of data, platforms, and technologies.

About the Technical Reviewer

Over his nearly three-decade career, **Alexey Panchekha**, **PhD**, **CFA**, has spent 10 years in academia, where he focused on nonlinear and dynamic processes; 10 years in the technology industry, where he specialized in program design and development; and eight years in financial services. In the latter arena, he specialized in applying mathematical techniques and technology to risk management and alpha generation. For example, Panchekha was involved in the equity derivative trading technology platform at Goldman Sachs, and he led the creation of the multi-asset multigeographies portfolio risk management system at Bloomberg. He also served as the head of research at Markov Process International, a leader in portfolio attribution and analytics. Most recently, Panchekha cofounded Turing Technology Associates, Inc., with Vadim Fishman. Turing is a technology and intellectual property company that sits at the intersection of mathematics, machine learning, and innovation. Its solutions typically service the financial technology (fintech) industry. Turing primarily focuses on enabling technology that supports the burgeoning ensemble active management (EAM). Prior to Turing, Panchekha was managing director at Incapital, and head of research at F-Squared Investments, where he designed innovative volatility-based risk-sensitive investment strategies. He is fluent in multiple computer programming languages and software and database programs and is certified in deep learning software. He earned a PhD from Kharkiv Polytechnic University with studies in physics and mathematics as well as an MS in physics. Panchekha is a CFA charterholder.

Acknowledgments

This being my first book, I found transforming idea and real-world experience into its current shape to be a strenuous task. I am grateful to almighty God for blessing and guiding me in all endeavors. I would like to thank my parents (Vijay Kumar and Rathnamma), brother (Shreyas), other family, and friends for helping me sail though the sea of life.

—B V Vishwas

First and foremost, praises and thanks to God, the almighty, for His showers of blessings throughout the book-writing process. I would like to express my deep and sincere gratitude to my parents (Dinesh Kumar N Patel and Javnika ben D Patel), my sister (Nisha Patel), and my family and friends (Shailesh Patel, Sanket Patel, Nikit Patel, Mansi Patel, Khushboo Shah) for their support and valuable prayers.

—Ashish Patel

Special thanks to Celestin Suresh John, Aditee Mirashi, James Markham, Alexey Panchekha, and the Apress team for bringing this book to life.

Introduction

This book explains the concepts of time series from traditional to bleeding-edge techniques with full-fledged examples.

The book begins by covering time-series fundamentals and their characteristics, Structure & Components of time series data, preprocessing, and ways of crafting features through data wrangling. Next, it covers traditional time-series techniques such as the smoothing methods ARMA, ARIMA, SARIMA, SARIMAX, VAR, and VARMA using trending frameworks such as Statsmodels and Pmdarima.

Further covers how to leverage advanced deep learning-based techniques such as ANN, CNN, RNN, LSTM, GRU, and Autoencoder to solve time-series problems using Tensorflow. It concludes by explaining how to use the popular framework fbprophet for modeling time-series analysis.

After completion of the book, the reader will have thorough knowledge of concepts and techniques to solve time-series problems. All the code presented in this book is available in Jupyter Notebooks; this allows readers to do hands-on experiments and enhance them in exciting ways.

CHAPTER 1

Time-Series Characteristics

A *time series* is a collection of data points that are stored with respect to their time. Mathematical and statistical analysis performed on this kind of data to find hidden patterns and meaningful insight is called *time-series analysis*. Time-series modeling techniques are used to understand past patterns from the data and try to forecast future horizons. These techniques and methodologies have been evolving for decades.

Observations with continuous timestamps and target variables are sometimes framed as straightforward regression problems by decomposing dates into minutes, hours, days, weeks, months, years, and so on, which is not the right way to handle such data because the results obtained are poor. In this chapter, you will learn the right approach for handling time-series data.

There are different kinds of data, such as structured, semistructured, and unstructured, and each type should be handled in its own way to gain maximum insight. In this book, we are going to be looking at time-series data that is structured in manner such as data from the stock market, weather, birth rates, traffic, bike-sharing apps, etc.

This chapter is a gentle introduction to the types of time-series data, its components, and ways to decompose it.

B V Vishwas and A. Patel, *Hands-on Time Series Analysis with Python*,
https://doi.org/10.1007/978-1-4842-5992-4_1

Types of Data

Time-series analysis is a statistical technique that measures a sequential set of data points. This is a standard measure in terms of time that comes in three types, as shown in Figure 1-1.

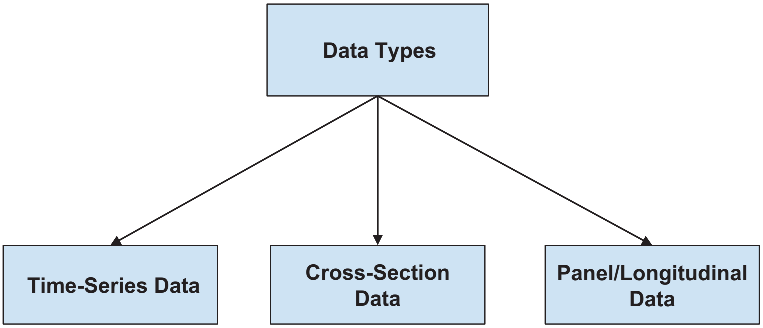

Figure 1-1. *Types of data*

Time-Series Data

A time series contains data points that increase, decrease, or otherwise change in chronological order over a period. A time series that incorporates the records of a single feature or variable is called a *univariate* time series. If the records incorporate more than one feature or variable, the series is called a *multivariate* time series. In addition, a time series can be designated in two ways: continuous or discrete.

In a *continuous* time series, data observation is carried out continuously throughout the period, as with earthquake seismograph magnitude data, speech data, etc. Figure 1-2 illustrates earthquake data measured continuously from 1975 to 2015.

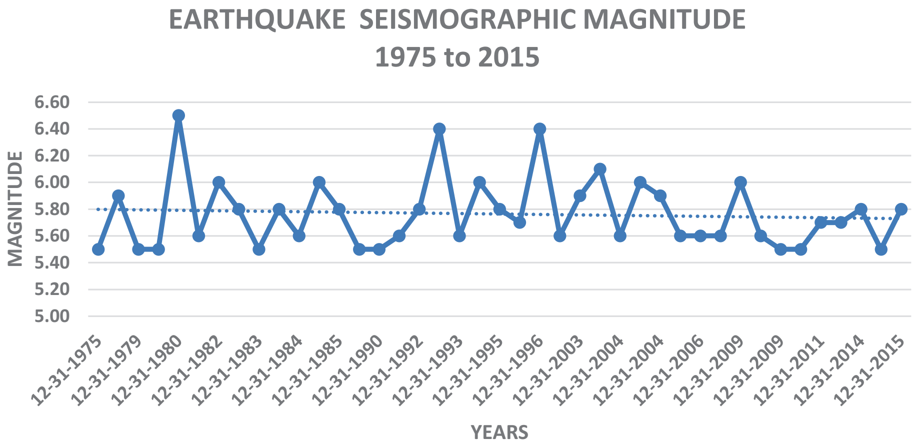

Figure 1-2. *Forty years of earthquake seismograph magnitude data*

Figure 1-3 Illustrates temperature behavior in India over a century and clearly shows that temperature is increasing monotonically.

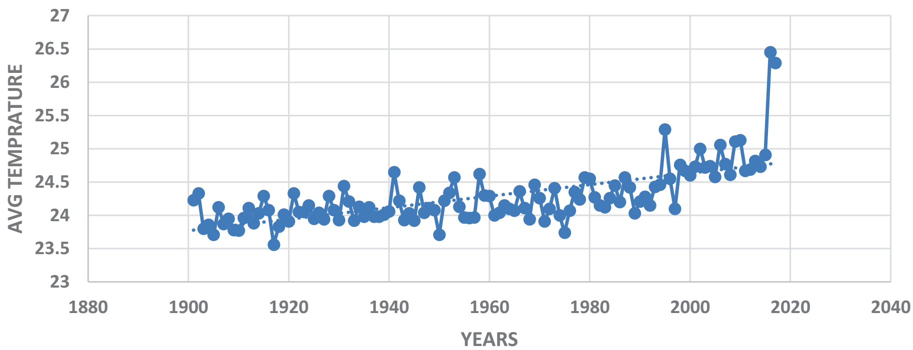

Figure 1-3. *India's temperature data from 1901 to 2017*

In a *discrete* time series, data observation is carried out at a specific time or equally spaced, as with temperature increases or decreases, exchange rates of currencies, air pressure data, etc. Figure 1-2 illustrates the analysis of the average temperature of India from 1901 to 2017, which either increases or decreases with time. This data behavior is discrete.

Cross-Section Data

Cross-section data is data gathered at a specific point of time for several subjects such as closing prices of a particular group of stocks on a specific date, opinion polls of elections, obesity level in population, etc. Cross-section studies are utilized in many research areas such as medical, economics, psychology, etc. For instance, high blood pressure is one of the significant risk factors for cause of death in India according to a 2017 WHO report. WHO has carried out the study of several risk factors (considered various subjects), which reflects cross-section survey data. Figure 1-4 illustrates the cross-section data.

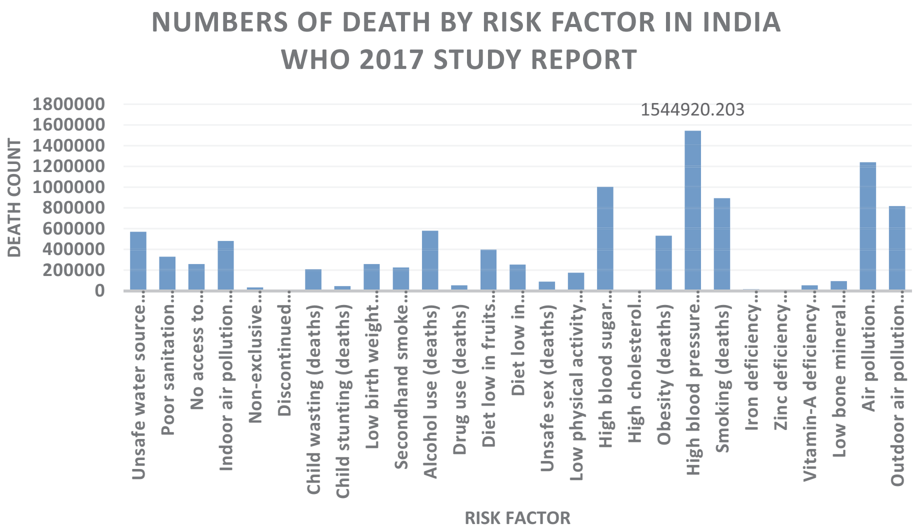

Figure 1-4. *Number of deaths by risk factor in India*

Panel Data/Longitudinal Data

Panel data/longitudinal data contains observations of multiple occurrences collected over various durations of time for the same individuals. It is data that is determined periodically by the number of observations in cross-sectional data units such as individuals, companies, or government agencies. Table 1-1 provides examples of data available for multiple people over the course of a few years where the data gathered comprises income, age, and sex.

Table 1-1. Example of Panel Data

Panel Data A				
Name	**Year**	**Income**	**Age**	**Sex**
Allen	2016	42145	24	Female
Allen	2017	47797	21	Female
Allen	2018	41391	23	Female
Malissa	2016	41100	22	Male
Malissa	2017	25800	23	Male
Malissa	2018	34508	22	Male
Panel Data B				
Name	**Year**	**Income**	**Age**	**Sex**
Malissa	2016	42688	27	Female
Malissa	2017	21219	25	Female
Allen	2016	46340	26	Male
Allen	2017	22715	22	Male
Allen	2018	34653	21	Male
Alicia	2017	31553	29	Female

In Table 1-1, datasets A and B (with the attributes income, age, and sex) gathered throughout the years are for different people. Dataset A is a depiction of two people, Allen and Malissa, who were subject to observation over three years (2016, 2017, 2018); this is known as *balanced panel data*. Dataset B is called *unbalanced panel data* because data does not exist for every individual every year.

Trend

A *trend* is a pattern that is observed over a period of time and represents the mean rate of change with respect to time. A trend usually shows the tendency of the data to increase/uptrend or decrease/downtrend during the long run. It is not always necessary that the increase or decrease is in the same direction throughout the given period of time. A trend line is also drawn using candlestick charts.

For example, you may have heard about an increase or decrease in different market commodities such as gold, silver, stock prices, gas, diesel, etc., or about the rate of interest for banks or home loans increasing or decreasing. These are all commodity market conditions, which may either increase or decrease over time, that show a trend in data.

Detecting Trend Using a Hodrick-Prescott Filter

The Hodrick-Prescott (HP) filter has become a benchmark for getting rid of trend movements in data. This method is broadly employed for econometric methods in applied macroeconomics research. The technique is nonparametric and is used to dissolve a time series into a trend; it is a cyclical component unaided by economic theory or prior trend specification. Like all nonparametric methods, the HP filter is contingent significantly on a tuning parameter that controls the degree of smoothing. This method is broadly employed in applied macroeconomics utilized in central banks, international economics agencies, industry, and government.

With the following example code, you can see how the EXINUS stock changes over a period of time:

```
import pandas as pd
from statsmodels.tsa.filters.hp_filter import hpfilter
df = pd.read_excel(r'\Data\India_Exchange_Rate_Dataset.xls',\
```

```
index_col=0,parse_dates=True)
EXINUS_cycle,EXINUS_trend = hpfilter(df['EXINUS'], lamb=1600)
EXINUS_trend.plot(figsize=(15,6)).autoscale(axis='x',tight=True)
```

Figure 1-5 shows an upward trend over the period.

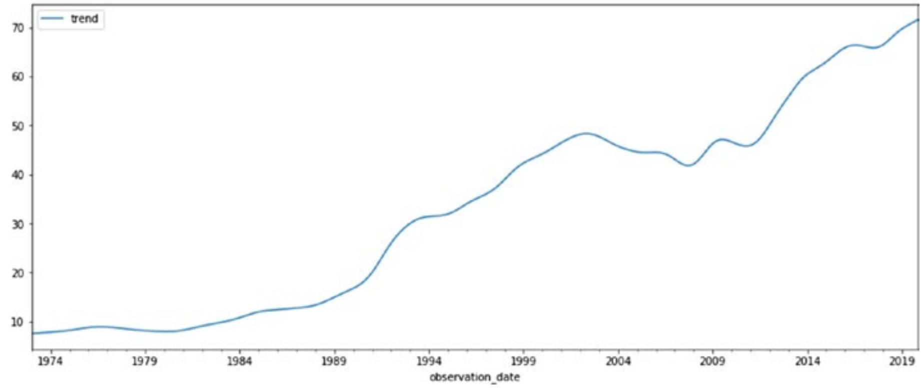

Figure 1-5. *EXINUS stock showing an upward trend*

Detrending a Time Series

Detrending is the process of removing a trend from time-series data, or it mentions a change in the mean over time. It is continuously increasing or decreasing over the duration of time. Identification, modeling, and even removing trend data from time-series datasets can be beneficial. The following are methods to detrend time-series data:

- Pandas differencing
- SciPy signal
- HP filter

Detrending Using Pandas Differencing

The Pandas library has a built-in function to calculate the difference in a dataset. This `diff()` function is used both for series and for DataFrames. It can provide a period value to shift in order to form the difference. The following code is an example of Pandas differencing.

- `Warning` is a built-in module of Python that handles the warning messages.
- `Pyplot` is a submodule of Matplotlib that is used to design the graphical representation of the data.

```
import pandas as pd
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
df = pd.read_excel(r'\Data\India_Exchange_Rate_Dataset.xls',\
index_col=0,parse_dates=True)
diff = df.EXINUS.diff()

plt.figure(figsize=(15,6))
plt.plot(diff)
plt.title('Detrending using Differencing', fontsize=16)
plt.xlabel('Year')
plt.ylabel('EXINUS exchange rate')
plt.show()
```

Figure 1-6 shows the data without a trend by using Pandas differencing.

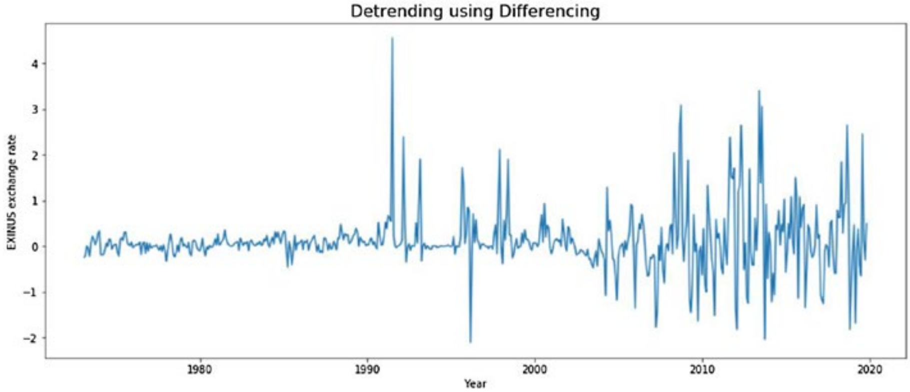

Figure 1-6. *Trend removal using differencing*

Detrending Using a SciPy Signal

A signal is another form of time-series data. Every signal either increases or decreases in a different order. Using the SciPy library, this can be removing the linear trend from the signal data. The following code shows an example of SciPy detrending.

- `Signal.detrend` is a submodule of SciPy that is used to remove a linear trend along an axis from data.

```
import pandas as pd
import matplotlib.pyplot as plt
from scipy import signal
import warnings
warnings.filterwarnings("ignore")
df = pd.read_excel(r'\Data\India_Exchange_Rate_Dataset.xls',\
index_col=0,parse_dates=True)
detrended = signal.detrend(df.EXINUS.values)

plt.figure(figsize=(15,6))
plt.plot(detrended)
plt.xlabel('EXINUS')
```

```
plt.ylabel('Frequency')
plt.title('Detrending using Scipy Signal', fontsize=16)
plt.show()
```

Figure 1-7 shows the detrended data using SciPy.

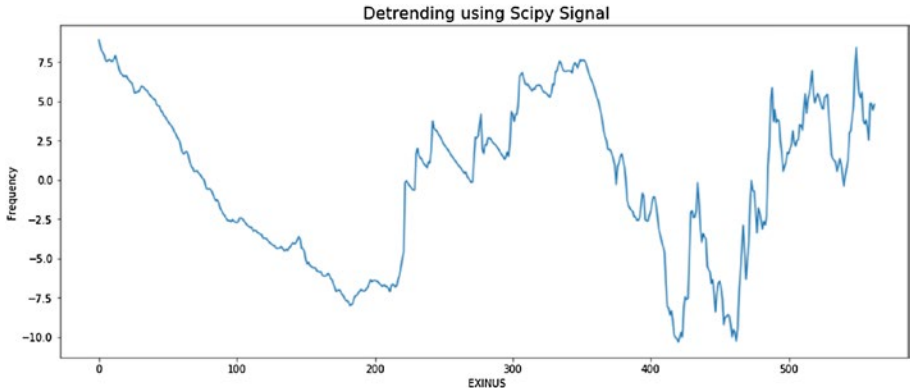

***Figure 1-7.** Removing a linear trend in a signal using SciPy*

Detrend Using an HP Filter

An HP filter is also used to detrend a time series and smooth the data. It's used for removing short-term fluctuations. The following code shows an example of HP filter detrending.

- `Hpfilter` is a submodule of Statmodels that is used to remove a smooth trend.

```
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.filters.hp_filter import hpfilter
import warnings
warnings.filterwarnings("ignore")
df = pd.read_excel(r'\Data\India_Exchange_Rate_Dataset.xls',\
index_col=0,parse_dates=True)
```

```
EXINUS_cycle,EXINUS_trend = hpfilter(df['EXINUS'],
lamb=1600)
df['trend'] = EXINUS_trend
.
detrended = df.EXINUS - df['trend']
plt.figure(figsize=(15,6))
plt.plot(detrended)
plt.title('Detrending using HP Filter', fontsize=16)
plt.xlabel('Year')
plt.ylabel('EXINUS exchange rate')
plt.show()
```

Figure 1-8 shows the data after removing a smooth trend.

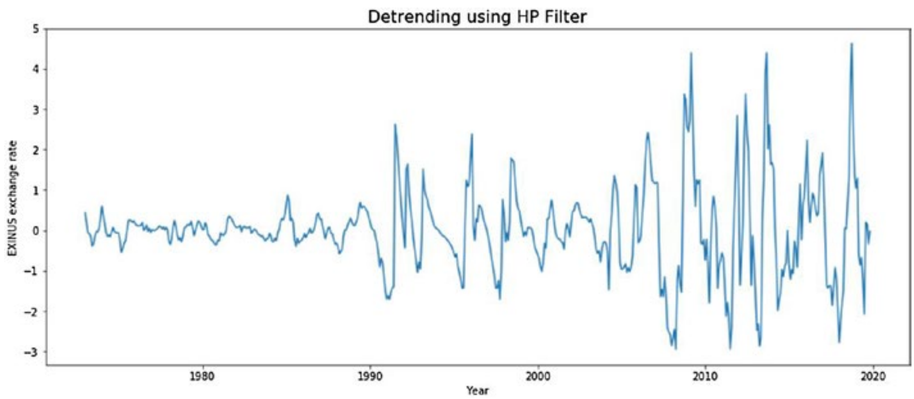

Figure 1-8. *Trend removal using an HP filter*

Seasonality

Seasonality is a periodical fluctuation where the same pattern occurs at a regular interval of time. It is a characteristic of economics, weather, and stock market time-series data; less often, it's observed in scientific data. In other industries, many phenomena are characterized by periodically recurring seasonal effects. For example, retail sales tend to increase during Christmas and decrease afterward.

The following methods can be used to detect seasonality:

- Multiple box plots
- Autocorrelation plots

Multiple Box Plots

A *box plot* is an essential graph to depict data spread out over a range. It is a standard approach to showing the minimum, first quartile, middle, third quartile, and maximum. The following code shows an example of detecting seasonality with the help of multiple box plots. See Figure 1-9.

- Seaborn is a graphical representation package similar to Matplotlib.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
df=pd.read_excel(r'\Data\India_Exchange_Rate_Dataset.
xls',\
parse_dates=True)
df['month'] = df['observation_date'].dt.strftime('%b')
df['year'] = [d.year for d in df.observation_date]
df['month'] = [d.strftime('%b') for d in
df.observation_date]
years = df['year'].unique()
plt.figure(figsize=(15,6))
sns.boxplot(x='month', y='EXINUS', data=df).set_
title("Multi Month-wise Box Plot")
plt.show()
```

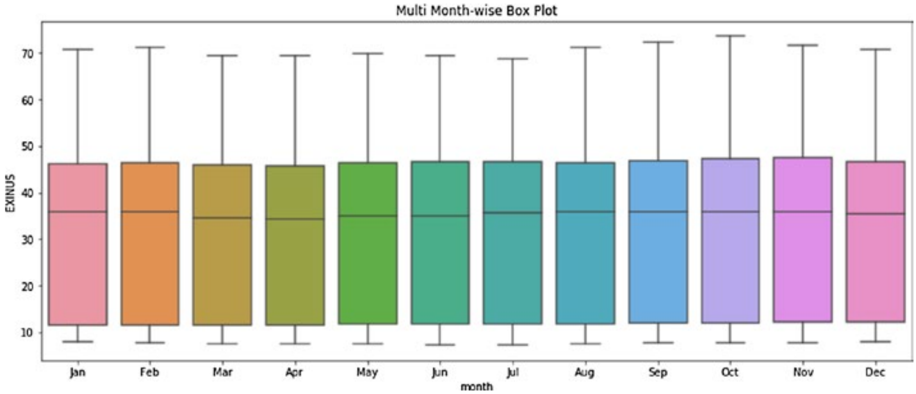

Figure 1-9. *Multiple-box plot to identify seasonality*

Autocorrelation Plot

Autocorrelation is used to check randomness in data. It helps to identify types of data where the period is not known. For instance, for the monthly data, if there is a regular seasonal effect, we would hope to see massive peak lags after every 12 months. Figure 1-10 demonstrates an example of detecting seasonality with the help of an autocorrelation plot.

```
from pandas.plotting import autocorrelation_plot
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_excel(r'\Data\India_Exchange_Rate_Dataset.xls',\
index_col=0,parse_dates=True)
plt.rcParams.update({'figure.figsize':(15,6), 'figure.
dpi':220})
autocorrelation_plot(df.EXINUS.tolist())
```

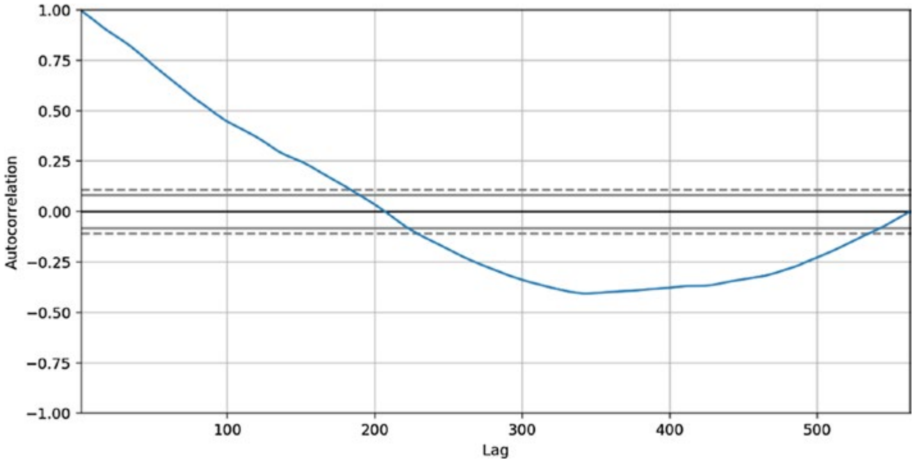

Figure 1-10. *Autocorrelation plot to identify seasonality*

Note Sometimes identifying seasonality is not easy; in that case, we need to evaluate other plots such as sequence or seasonal subseries plots.

Deseasoning of Time-Series Data

Deseasoning means to remove seasonality from time-series data. It is stripped of the pattern of seasonal effect to deseasonalize the impact. Time-series data contains four main components.

- **Level** means the average value of the time-series data.
- **Trend** means an increasing or decreasing value in time-series data.
- **Seasonality** means repeating the pattern of a cycle in the time-series data.
- **Noise** means random variance in time-series data.

Note An *additive model* is when time-series data combines these four components for linear trend and seasonality, and a *multiplicative model* is when components are multiplied to gather for nonlinear trends and seasonality.

Seasonal Decomposition

Decomposition is the process of understanding generalizations and problems related to time-series forecasting. We can leverage seasonal decomposition to remove seasonality from data and check the data only with the trend, cyclic, and irregular variations. Figure 1-11 illustrates data without seasonality.

```
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose
import warnings
warnings.filterwarnings("ignore")
df = pd.read_excel(r'\Data\India_Exchange_Rate_Dataset.xls',
index_col=0,parse_dates=True)
result_mul = seasonal_decompose(df['EXINUS'],
model='multiplicative', extrapolate_trend='freq')
deseason = df['EXINUS'] - result_mul.seasonal

plt.figure(figsize=(15,6))
plt.plot(deseason)
plt.title('Deseasoning using seasonal_decompose', fontsize=16)
plt.xlabel('Year')
plt.ylabel('EXINUS exchange rate')
plt.show()
```