Introduction to Time Series Forecasting with Python

How to Prepare Data and Develop Models to Predict the Future

Jason Brownlee

MACHINE
LEARNING
MASTERY

Disclaimer

The information contained within this eBook is strictly for educational purposes. If you wish to apply ideas contained in this eBook, you are taking full responsibility for your actions.
The author has made every effort to ensure the accuracy of the information within this book was correct at time of publication. The author does not assume and hereby disclaims any liability to any party for any loss, damage, or disruption caused by errors or omissions, whether such errors or omissions result from accident, negligence, or any other cause.

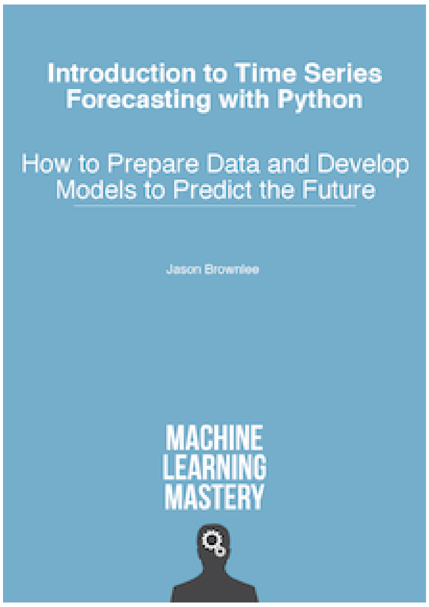
Copyright

Introduction to Time Series Forecasting with Python

Edition: v1.5

This is Just a Sample

Thank-you for your interest in **Introduction to Time Series Forecasting with Python**. This is just a sample of the full text. You can purchase the complete book online from:
https://machinelearningmastery.com/
introduction-to-time-series-forecasting-with-python/



Introduction to Time Series Forecasting with Python

How to Prepare Data and Develop Models to Predict the Future

Jason Brownlee

MACHINE
LEARNING
MASTERY

Contents

Welcome

Welcome to the *Introduction to Time Series Forecasting with Python.* You are one of those rare people that have decided to invest in your education and in your future and I am honored that I can help.

This book will show you how to make predictions on univariate time series problems using the standard tools in the Python ecosystem. Time series is an important and under served topic in applied machine learning, Python is the growing platform for machine learning and predictive modeling, and this book unlocks time series for Python. But, you have to do the work. I will lay out all of the topics, the tools, the code and the templates, but it is up to you to put them into practice on your projects and get results.

Before we dive in, it is important to first dial in to the goals, limitations and structure of this book to ensure that you can get the most out of it, quickly, efficiently and also have a lot of fun doing it. Let's start with the goals of the book.

Goals

The goal of this book is to *show you how to get results on univariate time series forecasting problems using the Python ecosystem.* This goal cannot be achieved until you apply the lessons from this book on your own projects and get results. This is a guidebook or a cookbook designed for immediate use.

Principles

This book was developed using the following five principles:

- **Application**: The focus is on the application of forecasting rather than the theory.
- **Lessons**: The book is broken down into short lessons, each focused on a specific topic.
- **Value**: Lessons focus on the most used and most useful aspects of a forecasting project.
- **Results**: Each lesson provides a path to a usable and reproducible result.
- **Speed**: Each lesson is designed to provide the shortest path to a result.

These principles shape the structure and organization of the book.

Prerequisites

This book makes some assumptions of you the reader. They are:

- **You're a Developer**: This is a book for developers. You are a developer of some sort. You know how to read and write code. You know how to develop and debug a program.
- **You know Python**: This is a book for Python people. You know the Python programming language, or you're a skilled enough developer that you can pick it up as you go along.
- **You know some Machine Learning**: This is a book for novice machine learning practitioners. You know some basic practical machine learning, or you can figure it out quickly.

Don't panic if you're not a perfect match. I will provide resources to help out including help on setting up a Python environment and resources for learning more about machine learning.

How to Read

The book was designed to fit three different reading styles. They are:

- **End-to-End**: The material is ordered such that you may read the book end-to-end, cover-to-cover and discover how you may work through time series forecasting problems.
- **Piecewise**: The lessons are standalone so that you can dip in to the topics you require piecewise for your forecasting project.
- **Template**: The projects are structured to provide a template that can be copied and used to work through your own forecasting projects.

Pick the reading style that is right for you to get the most out of the book as quickly as possible.

Reading Outcomes

If you choose to work through all of the lessons and projects of this book, you can set some reasonable expectations on your new found capabilities. They are:

- **Time Series Foundations**: You will be able to identify time series forecasting problems as distinct from other predictive modeling problems and how time series can be framed as supervised learning.
- **Transform Data For Modeling**: You will be able to transform, rescale, smooth and engineer features from time series data in order to best expose the underlying inherent properties of the problem (the signal) to learning algorithms for forecasting.

- **Harness Temporal Structure**: You will be able to analyze time series data and understand the temporal structure inherent in it such as trends and seasonality and how these structures may be addressed, removed and harnessed when forecasting.
- **Evaluate Models**: You will be able to devise a model test harness for a univariate forecasting problem and estimate the baseline skill and expected model performance on unseen data with various performance measures.
- **Apply Classical Methods**: You will be able to select, apply and interpret the results from classical linear methods such as Autoregression, Moving Average and ARIMA models on univariate time series forecasting problems.

You will be a capable predictive modeler for univariate time series forecasting problems using the Python ecosystem.

Limitations

This book is not all things to all people. Specifically:

- **Time Series Analysis**: This is a book of time series forecasting, not time series analysis. The primary concern of this book is using historical data to predict the future, not to understand the past.
- **Advanced Time Series**: This is not a treatment of complex time series problems. It does not provide tutorials on advanced topics like multi-step sequence forecasts, multivariate time series problems or spatial-temporal prediction problems.
- **Algorithm Textbook**: This is a practical book full of rules of application and code recipes. It is not a textbook, it does not describe why algorithms work or the equations that govern them.
- **Python Language Book**: This is not a treatment of the Python programming language. It does not provide an introduction to the history, syntax or semantics of Python code.

It is important to calibrate your expectations to the goals and principles of what this book is designed to achieve.

Structure

This book is presented as a series of lessons and projects.

- **Lessons**: Lessons take a tutorial format and focus on teaching one specific topic with just enough background to provide a context for the topic with a focus on examples and working code to demonstrate the topic in practice.
- **Projects**: Projects take a tutorial format and focus on the application of a suite of topics presented in lessons focused on getting an outcome on a real world predictive modeling dataset.

This book is primarily comprised of tutorial lessons. Lessons are organized into 5 main topics or themes. These are as follows:

- **Fundamentals**: This part contains introductory material such as how to setup a SciPy environment, examples of time series problems and what differentiates them from other types of predictive modeling and important concepts like random walk and white noise.
- **Data Preparation**: This part contains lessons on how to prepare data for forecasting. This includes methods such as scaling, transforms and feature engineering designed to better expose the inherent structure of the problem to modeling algorithms.
- **Temporal Structure**: This part focuses on the temporal structure of forecasting problems. This includes understanding systematic changes in the data over time such as trends and seasonality and the concept of stationarity.
- **Evaluate Models**: This part focuses on how to evaluate the performance of forecasts. This includes methods to back-test an algorithm, evaluate the skill of a model, perform diagnostics on the predictions from a model and reframe a problem.
- **Forecast Models**: This part focuses on classical linear models for time series forecasting. This includes autoregression, moving average and autoregressive integrated moving average or ARIMA models.

The lessons of the book culminate in three projects. The projects are designed to demonstrate both a structure for working through a forecasting project and the key techniques demonstrated in the book. These projects include:

- **Project 1**: Monthly Armed Robberies in Boston.
- **Project 2**: Annual Water Usage in Baltimore.
- **Project 3**: Monthly Sales of French Champagne.

Summary

You now have an idea of how the book is organized. You also now know the principles behind why the lessons are laser focused on fast repeatable results to help you deliver value as quickly as possible on your on time series forecasting projects.

Next

Next begins the Fundamentals part and the lessons focused on how time series forecasting problems are structured and how they are different to other types of predictive modeling problems. The first lesson gives you some background on how to setup your Python development environment.

Chapter 1

What is Time Series Forecasting?

Time series forecasting is an important area of machine learning that is often neglected. It is important because there are so many prediction problems that involve a time component. These problems are neglected because it is this time component that makes time series problems more difficult to handle. In this lesson, you will discover time series forecasting. After reading this lesson, you will know:

- Standard definitions of time series, time series analysis, and time series forecasting.
- The important components to consider in time series data.
- Examples of time series to make your understanding concrete.

Let's get started.

1.1 Time Series

A normal machine learning dataset is a collection of observations. For example:

```
observation #1
observation #2
observation #3
```

Listing 1.1: Example of a collection of observations.

Time does play a role in normal machine learning datasets. Predictions are made for new data when the actual outcome may not be known until some future date. The future is being predicted, but all prior observations are treated equally. Perhaps with some very minor temporal dynamics to overcome the idea of *concept drift* such as only using the last year of observations rather than all data available.

A time series dataset is different. Time series adds an explicit order dependence between observations: a time dimension. This additional dimension is both a constraint and a structure that provides a source of additional information.

> A time series is a sequence of observations taken sequentially in time.
>
> — Page 1, *Time Series Analysis: Forecasting and Control.*

For example:

```
Time #1, observation
Time #2, observation
Time #3, observation
```

Listing 1.2: Example of time series observations.

1.2 Time Series Nomenclature

Before we move on, it is important to quickly establish the standard terms used when describing time series data. The current time is defined as `t`, an observation at the current time is defined as `obs(t)`.

We are often interested in the observations made at prior times, called lag times or lags. Times in the past are negative relative to the current time. For example the previous time is `t-1` and the time before that is `t-2`. The observations at these times are `obs(t-1)` and `obs(t-2)` respectively. Times in the future are what we are interested in forecasting and are positive relative to the current time. For example the next time is `t+1` and the time after that is `t+2`. The observations at these times are `obs(t+1)` and `obs(t+2)` respectively.

For simplicity, we often drop the `obs(t)` notation and use `t+1` instead and assume we are talking about observations at times rather than the time indexes themselves. Additionally, we can refer to an observation at a lag by shorthand such as *a lag of 10* or `lag=10` which would be the same as `t-10`. To summarize:

- `t-n`: A prior or lag time (e.g. `t-1` for the previous time).
- `t`: A current time and point of reference.
- `t+n`: A future or forecast time (e.g. `t+1` for the next time).

1.3 Describing vs. Predicting

We have different goals depending on whether we are interested in understanding a dataset or making predictions. Understanding a dataset, called time series analysis, can help to make better predictions, but is not required and can result in a large technical investment in time and expertise not directly aligned with the desired outcome, which is forecasting the future.

> In descriptive modeling, or time series analysis, a time series is modeled to determine its components in terms of seasonal patterns, trends, relation to external factors, and the like. [...] In contrast, time series forecasting uses the information in a time series (perhaps with additional information) to forecast future values of that series
>
> — Page 18-19, *Practical Time Series Forecasting with R: A Hands-On Guide.*

1.3.1 Time Series Analysis

When using classical statistics, the primary concern is the analysis of time series. Time series analysis involves developing models that best capture or describe an observed time series in order to understand the underlying causes. This field of study seeks the *why* behind a time series dataset. This often involves making assumptions about the form of the data and decomposing the time series into constitution components. The quality of a descriptive model is determined by how well it describes all available data and the interpretation it provides to better inform the problem domain.

> The primary objective of time series analysis is to develop mathematical models that provide plausible descriptions from sample data

— Page 11, *Time Series Analysis and Its Applications: With R Examples.*

1.3.2 Time Series Forecasting

Making predictions about the future is called extrapolation in the classical statistical handling of time series data. More modern fields focus on the topic and refer to it as time series forecasting. Forecasting involves taking models fit on historical data and using them to predict future observations. Descriptive models can borrow from the future (i.e. to smooth or remove noise), they only seek to best describe the data. An important distinction in forecasting is that the future is completely unavailable and must only be estimated from what has already happened.

The skill of a time series forecasting model is determined by its performance at predicting the future. This is often at the expense of being able to explain why a specific prediction was made, confidence intervals and even better understanding the underlying causes behind the problem.

1.4 Components of Time Series

Time series analysis provides a body of techniques to better understand a dataset. Perhaps the most useful of these is the decomposition of a time series into 4 constituent parts:

- **Level**. The baseline value for the series if it were a straight line.
- **Trend**. The optional and often linear increasing or decreasing behavior of the series over time.
- **Seasonality**. The optional repeating patterns or cycles of behavior over time.
- **Noise**. The optional variability in the observations that cannot be explained by the model.

All time series have a level, most have noise, and the trend and seasonality are optional.

> The main features of many time series are trends and seasonal variations [...] another important feature of most time series is that observations close together in time tend to be correlated (serially dependent)

— Page 2, *Introductory Time Series with R.*

These constituent components can be thought to combine in some way to provide the observed time series. Assumptions can be made about these components both in behavior and in how they are combined, which allows them to be modeled using traditional statistical methods. These components may also be the most effective way to make predictions about future values, but not always. In cases where these classical methods do not result in effective performance, these components may still be useful concepts, and even input to alternate methods. The decomposition of time series into level, trend, seasonality and noise is covered more in Chapter **??**.

1.5 Concerns of Forecasting

When forecasting, it is important to understand your goal. Use the Socratic method and ask lots of questions to help zoom in on the specifics of your predictive modeling problem. For example:

1. **How much data do you have available and are you able to gather it all together?** More data is often more helpful, offering greater opportunity for exploratory data analysis, model testing and tuning, and model fidelity.

2. **What is the time horizon of predictions that is required?** Short, medium or long term? Shorter time horizons are often easier to predict with higher confidence.

3. **Can forecasts be updated frequently over time or must they be made once and remain static?** Updating forecasts as new information becomes available often results in more accurate predictions.

4. **At what temporal frequency are forecasts required?** Often forecasts can be made at a lower or higher frequencies, allowing you to harness down-sampling, and up-sampling of data, which in turn can offer benefits while modeling.

Time series data often requires cleaning, scaling, and even transformation. For example:

- **Frequency**. Perhaps data is provided at a frequency that is too high to model or is unevenly spaced through time requiring resampling for use in some models.

- **Outliers**. Perhaps there are corrupt or extreme outlier values that need to be identified and handled.

- **Missing**. Perhaps there are gaps or missing data that need to be interpolated or imputed.

Often time series problems are real-time, continually providing new opportunities for prediction. This adds an honesty to time series forecasting that quickly fleshes out bad assumptions, errors in modeling and all the other ways that we may be able to fool ourselves.

1.6 Examples of Time Series Forecasting

There is almost an endless supply of time series forecasting problems. Below are 10 examples from a range of industries to make the notions of time series analysis and forecasting more concrete.

- Forecasting the corn yield in tons by state each year.
- Forecasting whether an EEG trace in seconds indicates a patient is having a seizure or not.
- Forecasting the closing price of a stock each day.
- Forecasting the birth rate at all hospitals in a city each year.
- Forecasting product sales in units sold each day for a store.
- Forecasting the number of passengers through a train station each day.
- Forecasting unemployment for a state each quarter.
- Forecasting utilization demand on a server each hour.
- Forecasting the size of the rabbit population in a state each breeding season.
- Forecasting the average price of gasoline in a city each day.

I expect that you will be able to relate one or more of these examples to your own time series forecasting problems that you would like to address.

1.7 Summary

In this lesson, you discovered time series forecasting. Specifically, you learned:

- About time series data and the difference between time series analysis and time series forecasting.
- The constituent components that a time series may be decomposed into when performing an analysis.
- Examples of time series forecasting problems to make these ideas concrete.

1.7.1 Next

In the next lesson you will discover how to frame time series forecasting as a supervised learning problem.

Chapter 2

Load and Explore Time Series Data

The Pandas library in Python provides excellent, built-in support for time series data. Once loaded, Pandas also provides tools to explore and better understand your dataset. In this lesson, you will discover how to load and explore your time series dataset. After completing this tutorial, you will know:

- How to load your time series dataset from a CSV file using Pandas.
- How to peek at the loaded data and query using date-times.
- How to calculate and review summary statistics.

Let's get started.

2.1 Daily Female Births Dataset

In this lesson, we will use the Daily Female Births Dataset as an example. This dataset describes the number of daily female births in California in 1959. You can learn more about the dataset in Appendix ??. Place the dataset in your current working directory with the filename `daily-total-female-births.csv`.

2.2 Load Time Series Data

Pandas represented time series datasets as a Series. A `Series`[1] is a one-dimensional array with a time label for each row. The main function for loading CSV data in Pandas is the `read_csv()` function[2]. We can use this to load the time series as a `Series` object, instead of a `DataFrame`, as follows:

```
# load dataset using read_csv()
from pandas import read_csv
series = read_csv('daily-total-female-births.csv', header=0, index_col=0, parse_dates=True,
    squeeze=True)
print(type(series))
print(series.head())
```

[1] http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.html

[2] http://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html

Listing 2.1: Example of loading a CSV using `read_csv()`.

Note the arguments to the `read_csv()` function. We provide it a number of hints to ensure the data is loaded as a `Series`.

- `header=0`: We must specify the header information at row 0.
- `parse_dates=True`: We give the function a hint that data in the first column contains dates that need to be parsed.
- `index_col=0`: We hint that the first column contains the index information for the time series.
- `squeeze=True`: We hint that we only have one data column and that we are interested in a `Series` and not a `DataFrame`.

One more argument you may need to use for your own data is `date_parser` to specify the function to parse date-time values. In this example, the date format has been inferred, and this works in most cases. In those few cases where it does not, specify your own date parsing function and use the `date_parser` argument. Running the example above prints the same output, but also confirms that the time series was indeed loaded as a `Series` object.

```
<class 'pandas.core.series.Series'>
Date
1959-01-01   35
1959-01-02   32
1959-01-03   30
1959-01-04   31
1959-01-05   44
Name: Births, dtype: int64
```

Listing 2.2: Output of loading a CSV using `read_csv()`.

2.3 Exploring Time Series Data

Pandas also provides tools to explore and summarize your time series data. In this section, we'll take a look at a few, common operations to explore and summarize your loaded time series data.

2.3.1 Peek at the Data

It is a good idea to take a peek at your loaded data to confirm that the types, dates, and data loaded as you intended. You can use the `head()` function to peek at the first 5 records or specify the first `n` number of records to review. For example, you can print the first 10 rows of data as follows.

```
# summarize first few lines of a file
from pandas import read_csv
series = read_csv('daily-total-female-births.csv', header=0, index_col=0, parse_dates=True,
    squeeze=True)
print(series.head(10))
```

Listing 2.3: Example of printing the first 10 rows of a time series.

Running the example prints the following:

```
Date
1959-01-01   35
1959-01-02   32
1959-01-03   30
1959-01-04   31
1959-01-05   44
1959-01-06   29
1959-01-07   45
1959-01-08   43
1959-01-09   38
1959-01-10   27
Name: Births, dtype: int64
```

Listing 2.4: Output of printing the first 10 rows of a dataset.

You can also use the `tail()` function to get the last `n` records of the dataset.

2.3.2 Number of Observations

Another quick check to perform on your data is the number of loaded observations. This can help flush out issues with column headers not being handled as intended, and to get an idea on how to effectively divide up data later for use with supervised learning algorithms. You can get the dimensionality of your `Series` using the `size` parameter.

```
# summarize the dimensions of a time series
from pandas import read_csv
series = read_csv('daily-total-female-births.csv', header=0, index_col=0, parse_dates=True,
    squeeze=True)
print(series.size)
```

Listing 2.5: Example of printing the dimensions of a time series.

Running this example we can see that as we would expect, there are 365 observations, one for each day of the year in 1959.

```
365
```

Listing 2.6: Output of printing the dimensions of a dataset.

2.3.3 Querying By Time

You can slice, dice, and query your series using the time index. For example, you can access all observations in January as follows:

```
# query a dataset using a date-time index
from pandas import read_csv
series = read_csv('daily-total-female-births.csv', header=0, index_col=0, parse_dates=True,
    squeeze=True)
print(series['1959-01'])
```

Listing 2.7: Example of querying a time series by date-time.

Running this displays the 31 observations for the month of January in 1959.

```
Date
1959-01-01   35
1959-01-02   32
1959-01-03   30
1959-01-04   31
1959-01-05   44
1959-01-06   29
1959-01-07   45
1959-01-08   43
1959-01-09   38
1959-01-10   27
1959-01-11   38
1959-01-12   33
1959-01-13   55
1959-01-14   47
1959-01-15   45
1959-01-16   37
1959-01-17   50
1959-01-18   43
1959-01-19   41
1959-01-20   52
1959-01-21   34
1959-01-22   53
1959-01-23   39
1959-01-24   32
1959-01-25   37
1959-01-26   43
1959-01-27   39
1959-01-28   35
1959-01-29   44
1959-01-30   38
1959-01-31   24
Name: Births, dtype: int64
```

Listing 2.8: Output of querying a time series dataset.

This type of index-based querying can help to prepare summary statistics and plots while exploring the dataset.

2.3.4 Descriptive Statistics

Calculating descriptive statistics on your time series can help get an idea of the distribution and spread of values. This may help with ideas of data scaling and even data cleaning that you can perform later as part of preparing your dataset for modeling. The `describe()` function creates a 7 number summary of the loaded time series including mean, standard deviation, median, minimum, and maximum of the observations.

```
# calculate descriptive statistics
from pandas import read_csv
series = read_csv('daily-total-female-births.csv', header=0, index_col=0, parse_dates=True,
    squeeze=True)
print(series.describe())
```

Listing 2.9: Example of printing the descriptive stats of a time series.

Running this example prints a summary of the birth rate dataset.

```
count   365.000000
mean     41.980822
std       7.348257
min      23.000000
25%      37.000000
50%      42.000000
75%      46.000000
max      73.000000
Name: Births, dtype: float64
```

Listing 2.10: Output of printing the descriptive stats of a times series.

2.4 Summary

In this lesson, you discovered how to load and handle time series data using the Pandas Python library. Specifically, you learned:

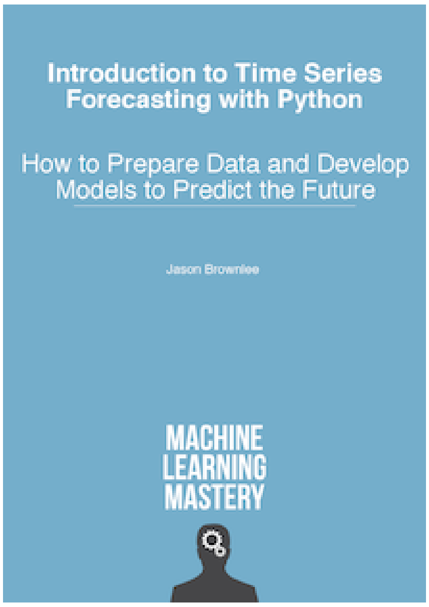
- How to load your time series data as a Pandas `Series`.
- How to peek at your time series data and query it by date-time.
- How to calculate and review summary statistics on time series data.

2.4.1 Next

In the next lesson you will discover how to perform feature engineering with time series data.

This is Just a Sample

Thank-you for your interest in **Introduction to Time Series Forecasting with Python**. This is just a sample of the full text. You can purchase the complete book online from:
https://machinelearningmastery.com/introduction-to-time-series-forecasting-with-python/



Introduction to Time Series Forecasting with Python

How to Prepare Data and Develop Models to Predict the Future

Jason Brownlee

MACHINE LEARNING MASTERY