Time Series Analysis

Python for Finance

- It is now time to shift our focus to dealing with time series data!
- A lot of our financial information is going to come in the form of some value plotted against a time series.

Python for Finance

- While the concepts presented in this section of the course are very important, we may not use them often when working directly with our algorithmic trading models.

- In fact, one of our main reasons for covering these topics is so that in future sections of the course we can show why using some of these analysis techniques on stock information is actually NOT a good idea.

Python for Finance

- It can be very tempting to use some of these techniques on financial data, but sometimes it’s actually not a good idea, and to understand why that is, we first need to understand the techniques themselves.

- So as an overall approach to this section, you should try to get a higher level understanding of some of these concepts, but don't get concerned too much with the details (as far as future sections of the course are concerned).

Python for Finance

- In this section we will discuss:
  - Time Series Basics
  - Statsmodels Python Library
  - ETS Models and Decomposition
  - EWMA Models
  - ARIMA Models

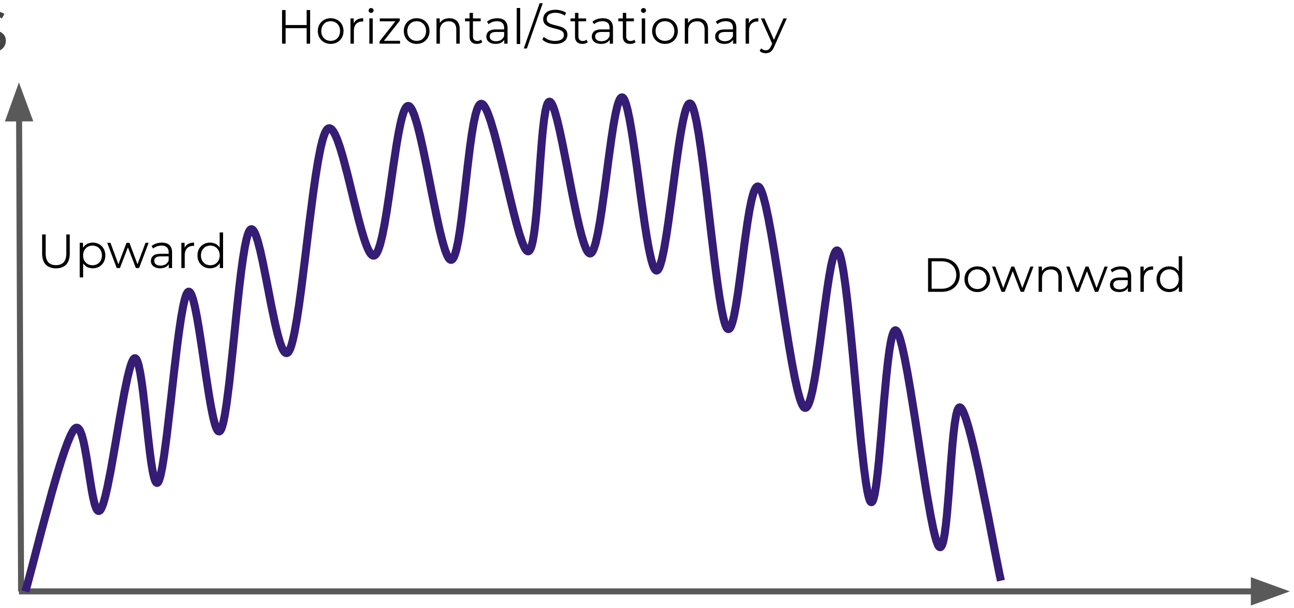
Time Series Basics

Python for Finance

- Let's begin discussing some important Time series concepts.
- Time series data has particular properties, let's take a look at some plots and discuss some important terms!

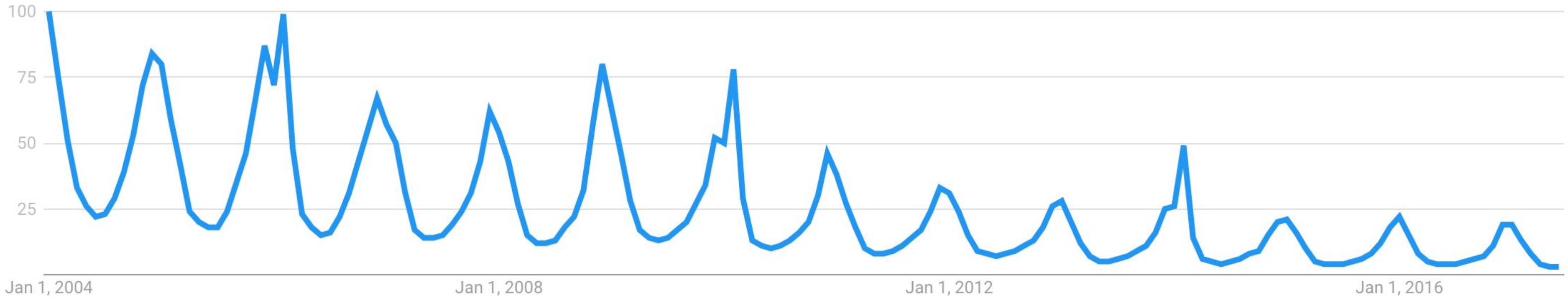
Python for Finance

- Trends



Python for Finance

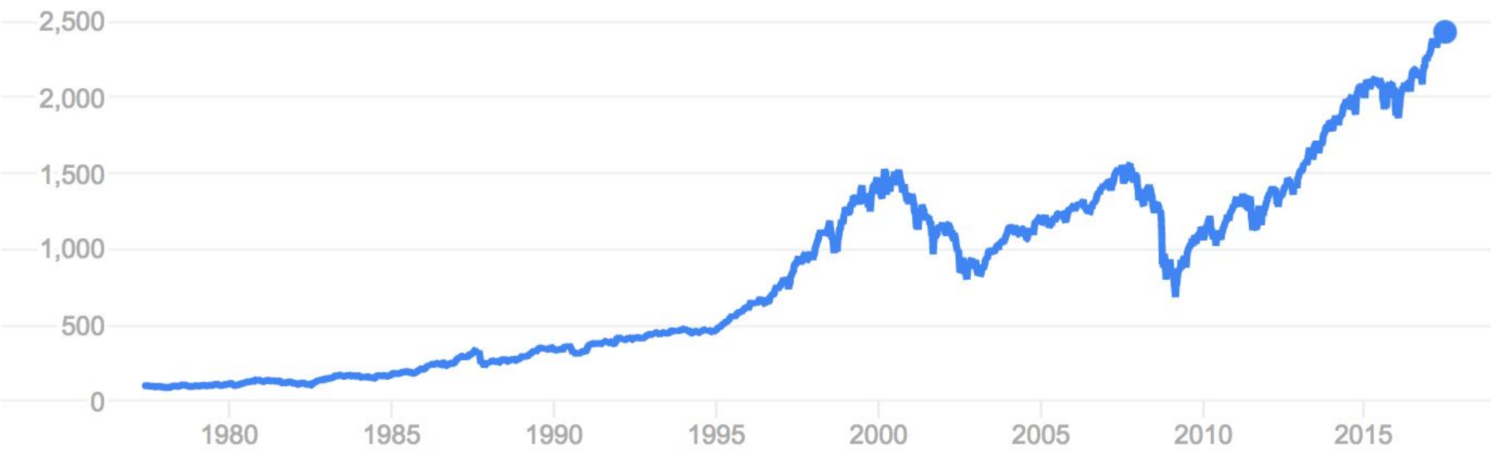
- Seasonality - Repeating trends



Google Trends - “Snowboarding”

Python for Finance

- Cyclical - Trends with no set repetition.



SP500

Python for Finance

- Now that we understand some of the very basics, let's begin to learn about the most popular library in Python for handling time series data, statsmodels!

Statsmodels

Python for Finance

- The most popular library in Python for dealing with Time Series data is the statsmodels library.
- It is heavily inspired by the R statistical programming language.

- Statsmodels is a Python module that allows users to explore data, estimate statistical models, and perform statistical tests.

Python for Finance

- An extensive list of descriptive statistics, statistical tests, plotting functions, and result statistics are available for different types of data and each estimator.

- Statsmodels is already included in the provided environment file.
- To manually install you can use:
    - conda install statsmodels

Python for Finance

- Let’s explore the documentation and then run through a simple demonstration of what we can use statsmodels for in relation to time series data.

ETS Models

Python for Finance

- For the next few lectures, we will discuss topics conceptually in slides.
- Afterwards we will revisit these topics using Python and statsmodels to code through them!

Python for Finance

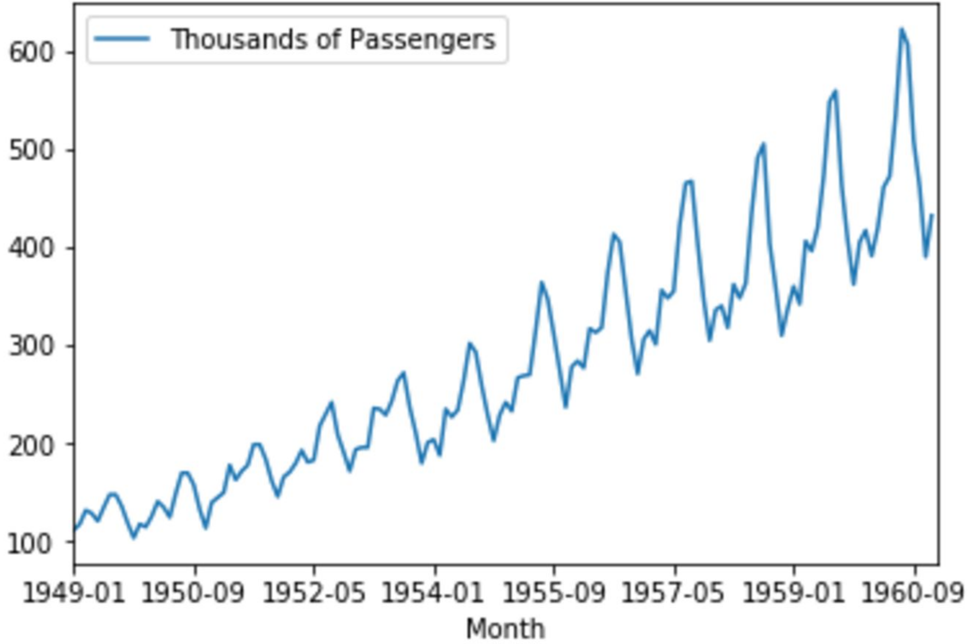
- ETS Models (Error-Trend-Seasonality)
  - Exponential Smoothing
  - Trend Methods Models
  - ETS Decomposition
- We'll work with several of these with the python statsmodels library!

- ETS (Error-Trend-Seasonality) Models will take each of those terms for “smoothing” and may add them, multiply them, or even just leave some of them out.
- Based off these key factors, we can try to create a model to fit our data.

- Time Series Decomposition with ETS (Error-Trend-Seasonality).
- Visualizing the data based off its ETS is a good way to build an understanding of its behaviour.

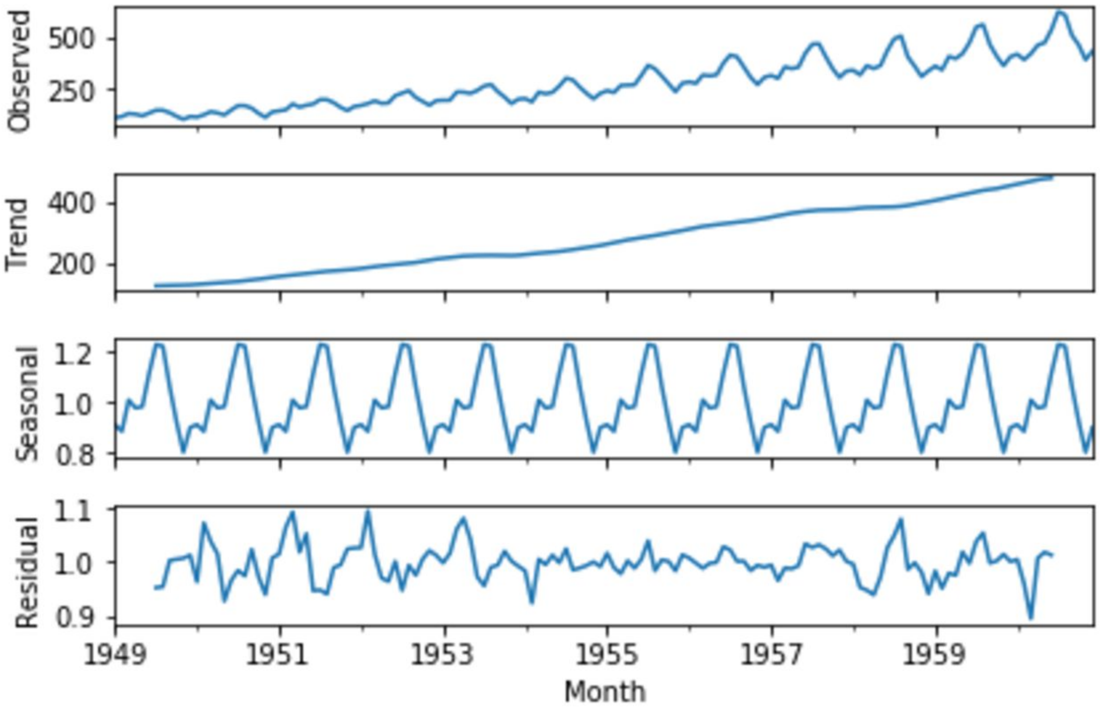
Python for Finance

- ETS Decomposition - Airline Passengers



Python for Finance

- ETS Decomposition - Airline Passengers



Python for Finance

- Time Series Decomposition with ETS (Error-Trend-Seasonality).
- Visualizing the data based off its ETS is a good way to build an understanding of its behaviour.

- We will visit Time Series Decomposition again when we discuss ARIMA models.
- For now let's move on to EWMA models!

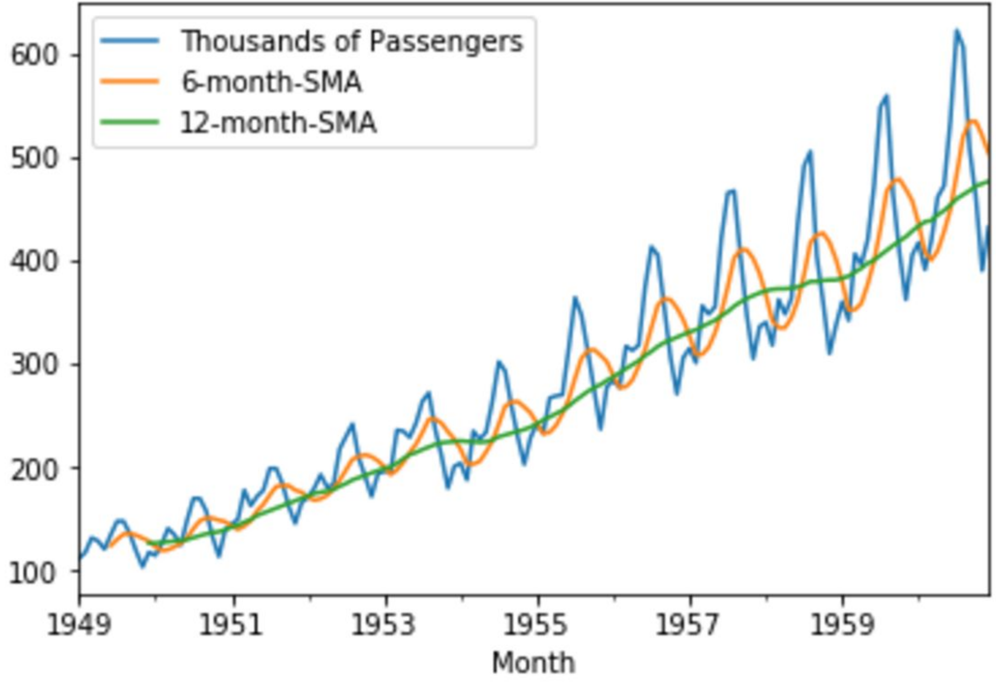
EWMA Models

Python for Finance

- We’ve previously seen how calculating simple moving averages can allow us to create a simple model that describes some trend level behavior of a time series, for example...

Python for Finance

- SMA - Simple Moving Averages



- EWMA- Exponentially Weighted Moving Averages
- Basic SMA has some "weaknesses".
    - Smaller windows will lead to more noise, rather than signal

Python for Finance

- EWMA- Exponentially Weighted Moving Averages
- Basic SMA has some "weaknesses".
    - It will always lag by the size of the window

- EWMA- Exponentially Weighted Moving Averages
- Basic SMA has some "weaknesses".
    - It will never reach to full peak or valley of the data due to the averaging.

- EWMA- Exponentially Weighted Moving Averages
- Basic SMA has some "weaknesses".
    - Does not really inform you about possible future behaviour, all it really does is describe trends in your data.

Python for Finance

- EWMA- Exponentially Weighted Moving Averages
- Basic SMA has some "weaknesses".
    - Extreme historical values can skew your SMA significantly

* EWMA- Exponentially Weighted Moving Averages
* Basic SMA has some "weaknesses".
  * To help fix some of these issues, we can use an EWMA (Exponentially-weighted moving average).

- EWMA will allow us to reduce the lag effect from SMA and it will put more weight on values that occurred more recently (by applying more weight to the more recent values, thus the name).

- The amount of weight applied to the most recent values will depend on the actual parameters used in the EWMA and the number of periods given a window size.

- Let’s code out an example of using pandas to create EWMA in the next lecture!

EWMA Code Along

ETS Decomposition
Code Along

ARIMA Models

- We will now discuss one of the most common time series models, ARIMA.
- Please note, this is an **optional** section of the course.

Python for Finance

- For various reasons we will discover later on, ARIMA models often don’t work well with historical stock data.
- However, they are so fundamental to understanding time series analysis that it is still worth the time to go over them.

Python for Finance

* ARIMA models can be complex!
* Make sure to make full use of the various links and extra resources presented throughout this section if you want to later use ARIMA models for other problems.

- AutoRegressive Integrated Moving Average (ARIMA) model is a generalization of an autoregressive moving average (ARMA) model.

- Both of those models (ARIMA and ARMA) are fitted to time series data either to better understand the data or to predict future points in the series (forecasting).

- ARIMA (Autoregressive Integrated Moving Averages)
  - Non-seasonal ARIMA
  - Seasonal ARIMA

- We will start by discussing non-seasonal ARIMA models and then move on to seasonal ARIMA models.
- The python examples at the end will be using seasonal ARIMA.

- ARIMA models are applied in some cases where data show evidence of non-stationarity, where an initial differencing step (corresponding to the "integrated" part of the model) can be applied one or more times to eliminate the non-stationarity.

Python for Finance

- Differencing is actually a very simple idea, but let’s put it on hold for now, and talk a bit more about ARIMA!
- We’ll touch back on differencing later on.
- Let’s talk about the major components of ARIMA.

- Non-seasonal ARIMA models are generally denoted ARIMA(p,d,q) where parameters p, d, and q are non-negative integers.
- Let's discuss what these three components are!

- Parts of ARIMA model
- AR (p): Autoregression
    - A regression model that utilizes the dependent relationship between a current observation and observations over a previous period

Python for Finance

- Parts of ARIMA model
- I (d): Integrated.
  - Differencing of observations (subtracting an observation from an observation at the previous time step) in order to make the time series stationary.

Python for Finance

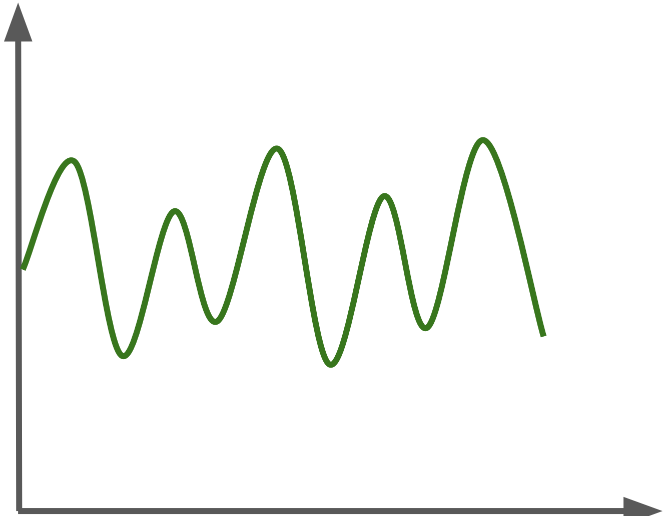
- Parts of ARIMA model
- MA (q): Moving Average.
    - A model that uses the dependency between an observation and a residual error from a moving average model applied to lagged observations.

- Stationary vs Non-Stationary Data
  - To effectively use ARIMA, we need to understand Stationarity in our data.
  - So what makes a data set Stationary?
    - A Stationary series has constant mean and variance over time.

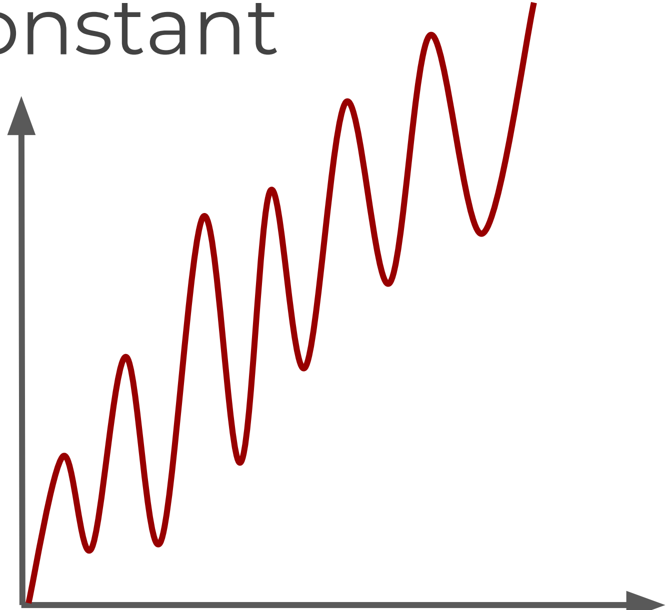
- A Stationary data set will allow our model to predict that the mean and variance will be the same in future periods.
- Let’s take a look at a few examples!

Python for Finance

- Mean needs to be constant



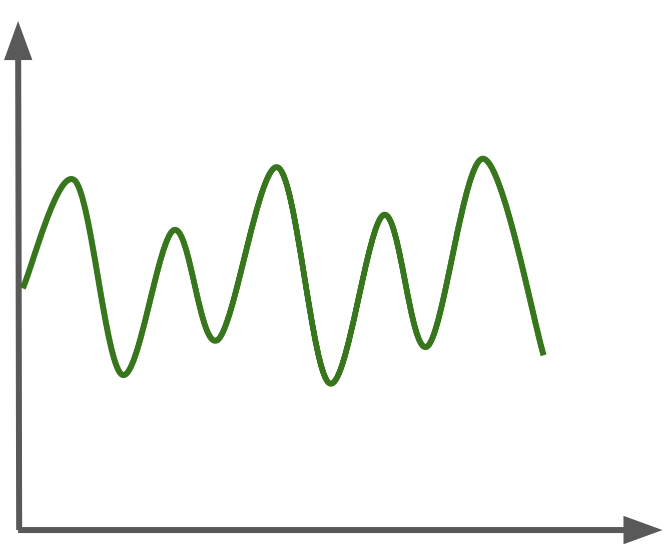
Stationary



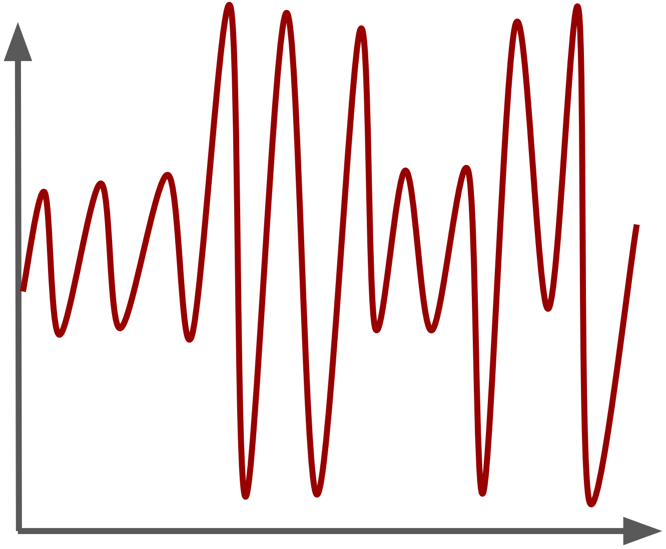
Non-Stationary

Python for Finance

- Variance should not be a function of time

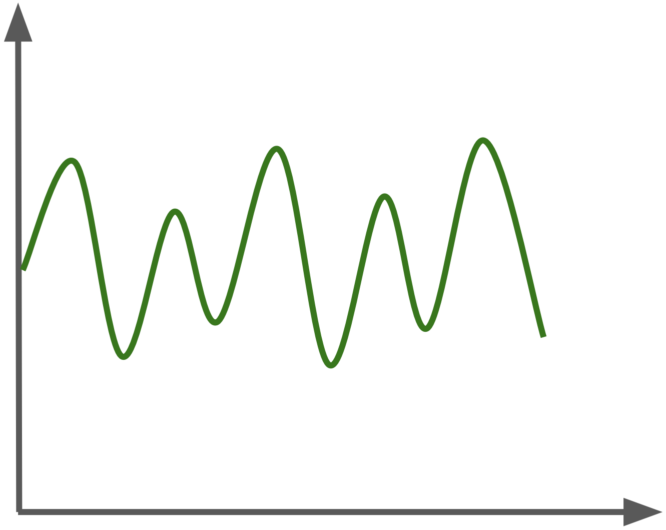


Stationary

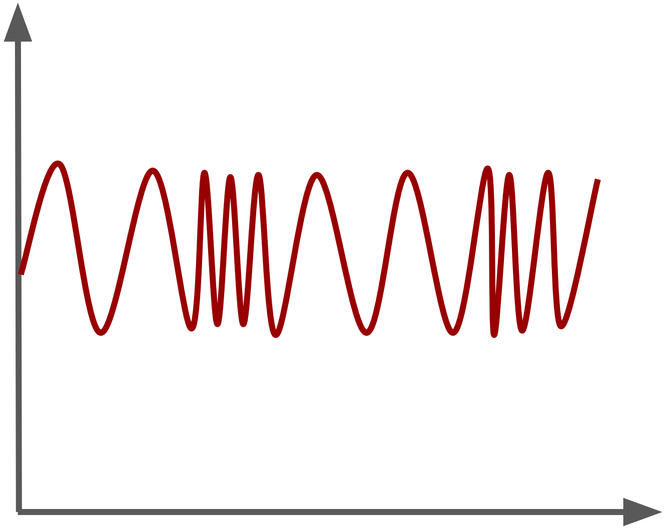


Non-Stationary

Covariance should not be a function of time



Stationary



Non-Stationary

- There are also mathematical tests you can use to test for stationarity in your data.
- A common one is the Augmented Dickey–Fuller test (we will see how to use this with Python's statsmodels)

Python for Finance

- If you've determined your data is not stationary (either visually or mathematically), you will then need to transform it to be stationary in order to evaluate it and what type of ARIMA terms you will use.

- One simple way to do this is through "differencing".
- The idea behind differencing is quite simple, let's see an example...

Python for Finance

Original Data

Time1	10
Time2	12
Time3	8
Time4	14
Time5	7

First Difference

Time1	NA
Time2	2
Time3	-4
Time4	6
Time5	-7

Second Difference

Time1	NA
Time2	NA
Time3	-6
Time4	10
Time5	-13

- You can continue differencing until you reach stationarity (which you can check visually and mathematically)
- Each differencing step comes at the cost of losing a row of data.

- For seasonal data, you can also difference by a season.
- For example, if you had monthly data with yearly seasonality, you could difference by a time unit of 12, instead of just 1.

- Another common technique with seasonal ARIMA models is to combine both methods, taking the seasonal difference of the first difference.

- With your data now stationary it is time to go back and discuss the p,d,q terms and how you choose them.
- A big part of this are AutoCorrelation Plots and Partial AutoCorrelation Plots.
- Let's move on to discuss them!

AutoCorrelation Plots

- An autocorrelation plot (also known as a Correlogram) shows the correlation of the series with itself, lagged by x time units.
- So the y axis is the correlation and the x axis is the number of time units of lag.

- Let's explain this idea of correlation with a simple example.
- We'll start off by trying to imagine how to calculate the plot value for x=1

- Imagine taking your time series of length T, copying it, and deleting the first observation of copy #1 and the last observation of copy #2.

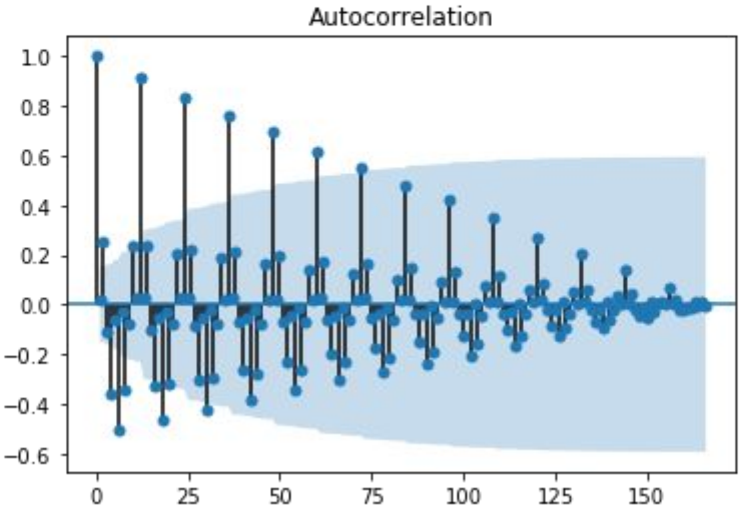
- Now you have two series of length T−1 for which you calculate a correlation coefficient.
- This is the value of the vertical axis at x=1 in your plots.

Python for Finance

- It represents the correlation of the series lagged by one time unit.
- You go on and do this for all possible time lags x and this defines the plot.
- Let's see some typical examples!

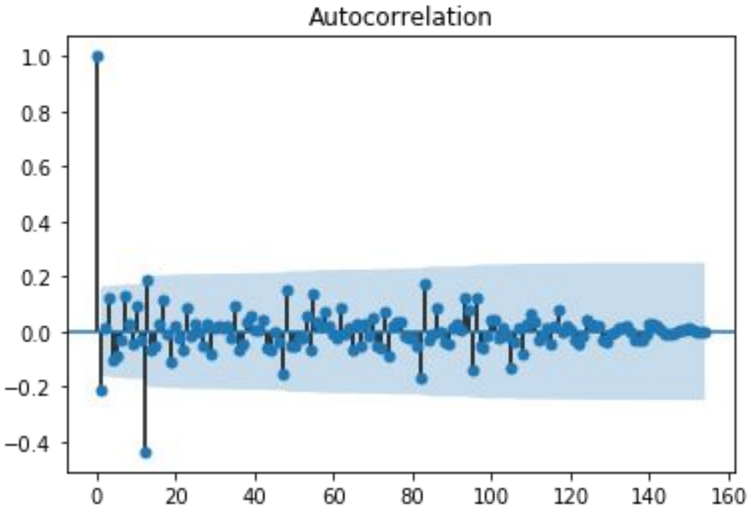
Python for Finance

- Gradual Decline



Python for Finance

- Sharp Drop-off



- The actual interpretation and how it relates to ARIMA models can get a bit complicated, but there are some basic common methods we can use for the ARIMA model.

Python for Finance

* Our main priority here is to try to figure out whether we will use the AR or MA components for the ARIMA model (or both!) as well as how many lags we should use.

- In general you would use either AR or MA, using both is less common.
- When actually applying the AR and MA terms, you will set values of p or q.

- If the autocorrelation plot shows positive autocorrelation at the first lag (lag-1), then it suggests to use the AR terms in relation to the lag

Python for Finance

- If the autocorrelation plot shows negative autocorrelation at the first lag, then it suggests using MA terms.
- This will allow you to decide what actual values of p,d, and q to provide your ARIMA model.

- p: The number of lag observations included in the model.
- d: The number of times that the raw observations are differenced
- q: The size of the moving average window, also called the order of moving average.

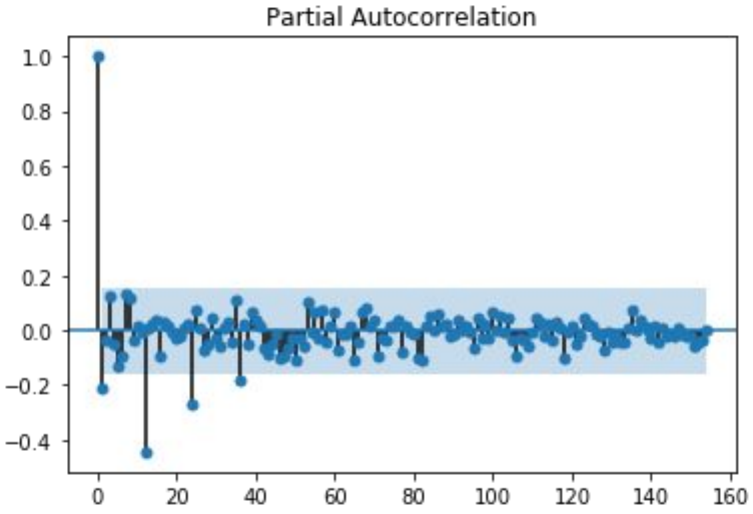
- There are also partial autocorrelation plots!
- These are a little more complicated than autocorrelation plots, but let’s show you the basics.

- In general, a partial correlation is a conditional correlation.
- It is the correlation between two variables under the assumption that we know and take into account the values of some other set of variables.

- For instance, consider a regression context in which y = response variable and x1, x2, and x3 are predictor variables.
- The partial correlation between y and x3 is the correlation between the variables determined taking into account how both y and x3 are related to x1 and x2.

Python for Finance

- Let's see an example of what the plot can look like:



- Typically a sharp drop after lag "k" suggests an AR-k model should be used.
- If there is a gradual decline, it suggests an MA model.

- Identification of an AR model is often best done with the PACF.
- Identification of an MA model is often best done with the ACF rather than the PACF.
- View the notebook and resource links for more details.

- Finally once you've analyzed your data using ACF and PACF you are ready to begin to apply ARIMA or Seasonal ARIMA, depending on your original data.
- You will provide the p,d, and q terms for the model.

- An ARIMA will then take three terms p,d, and q. (We'll see this in the coding example)
- For seasonal ARIMA there will be an additional set of P,D,Q terms that we will see.

Python for Finance

* Alright, now it is time to see all of this in action with Python and statsmodels!
* Let's get started!

ARIMA Code Along

Part Four