Time Series with Pandas

PIERIAN DATA

Time Series with Pandas

- It is now time to shift our focus to dealing with time series data!
- Pandas has many tools specifically built for working with time stamped data.

Time Series with Pandas

- In this section we will discuss:
  - DateTime Index Basics
  - Time Resampling
  - Time Shifting
  - Rolling and Expanding
  - Time Series Visualization
  - Time Series Project Exercise

Datetime Index

Time Resampling

Time Shifting

Rolling and Expanding

Visualizing Time Series Data with Pandas

PIERIAN DATA

EWMA Code Along

ETS Decomposition Code Along

ARIMA Models

PIERIAN DATA

Python for Finance

- 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.

Python for Finance

- 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).

Python for Finance

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

Python for Finance

- 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.

Python for Finance

- 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.

Python for Finance

- 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!

Python for Finance

- 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.

Python for Finance

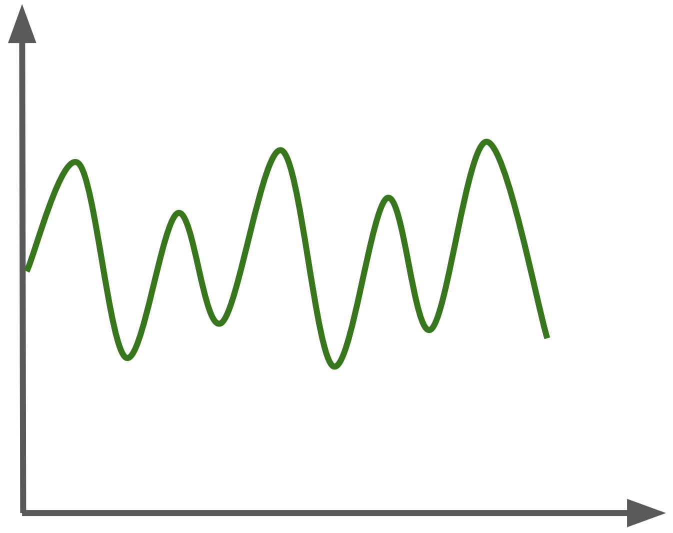
- Stationary vs Non-Stationary Data
    - To effectively use ARIMA, we need to understand Stationarity in our data.
    - So what makes a data set Stationary?
        - A Stationary series has constant mean and variance over time.

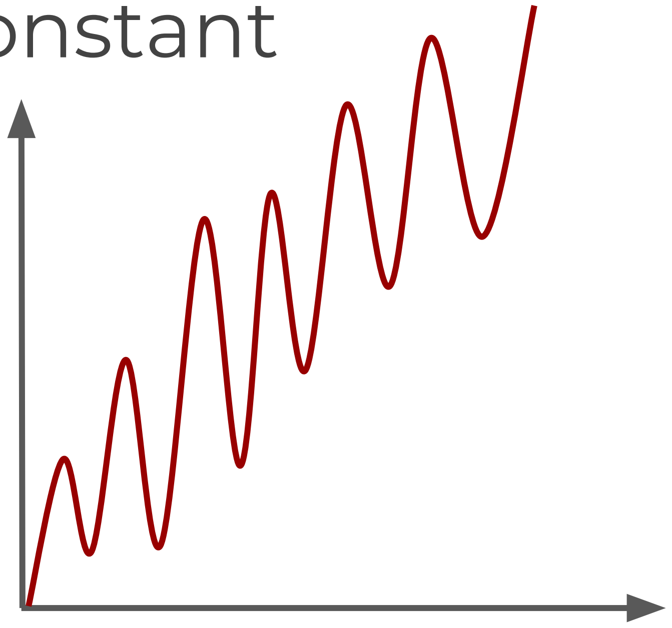
Python for Finance

- 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

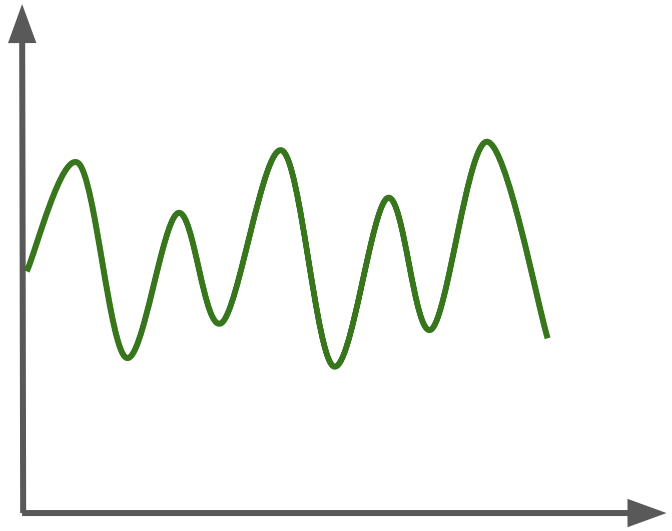
- Mean needs to be constant


Stationary

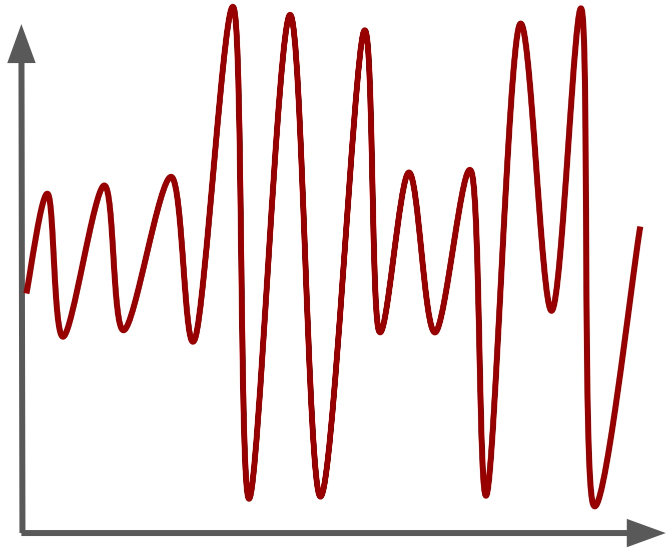

Non-Stationary

Python for Finance

- Variance should not be a function of time



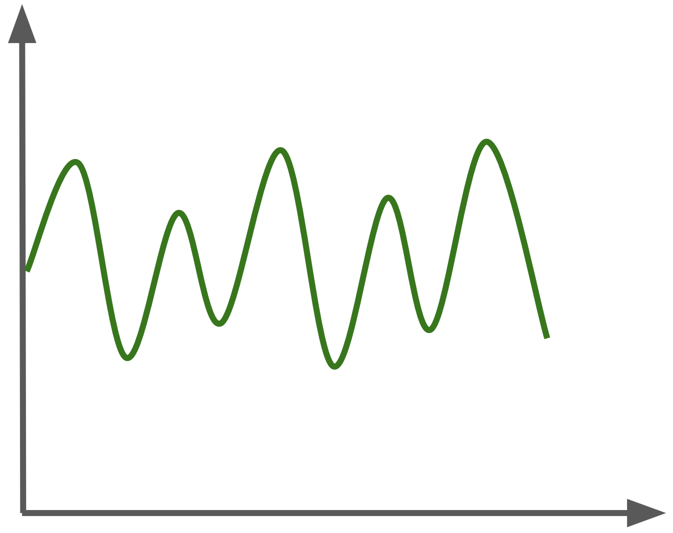
Stationary

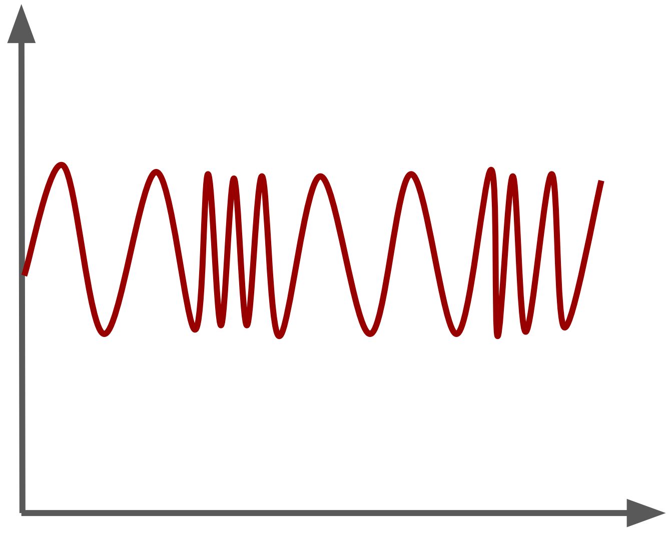


Non-Stationary

PIERIAN DATA

Covariance should not be a function of time


Stationary


Non-Stationary

Python for Finance

- 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.

Python for Finance

- 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

Python for Finance

- 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.

PIERIAN DATA

Python for Finance

- 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.

Python for Finance

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

Python for Finance

- 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

Python for Finance

- 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.

Python for Finance

- 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

Python for Finance

- 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.

Python for Finance

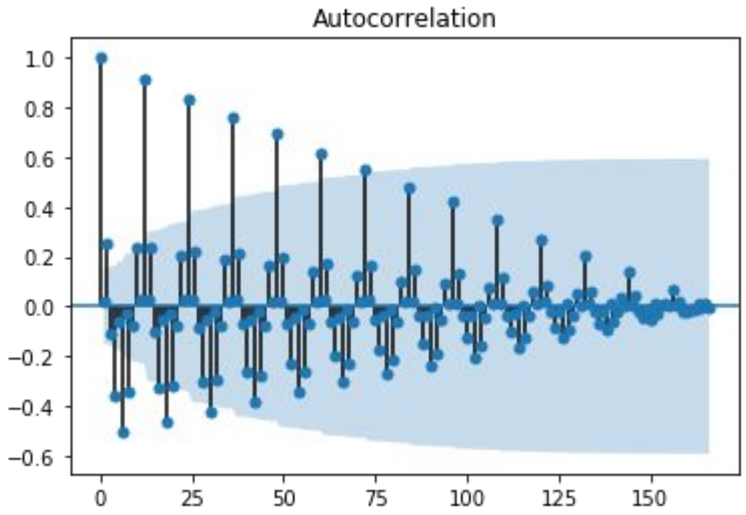
- Now you have two series of length T–1 for which you calculate a correlation coefficient.
- This is the value of the vertical axis at x=1 in your plots.

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!

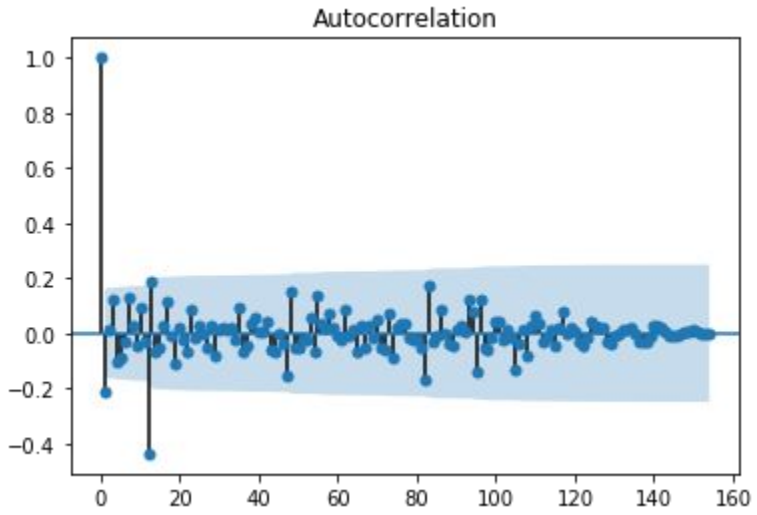
Python for Finance

- Gradual Decline



Python for Finance

- Sharp Drop-off



Python for Finance

- 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.

Python for Finance

- 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.

Python for Finance

- 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

- 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.

Python for Finance

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

Python for Finance

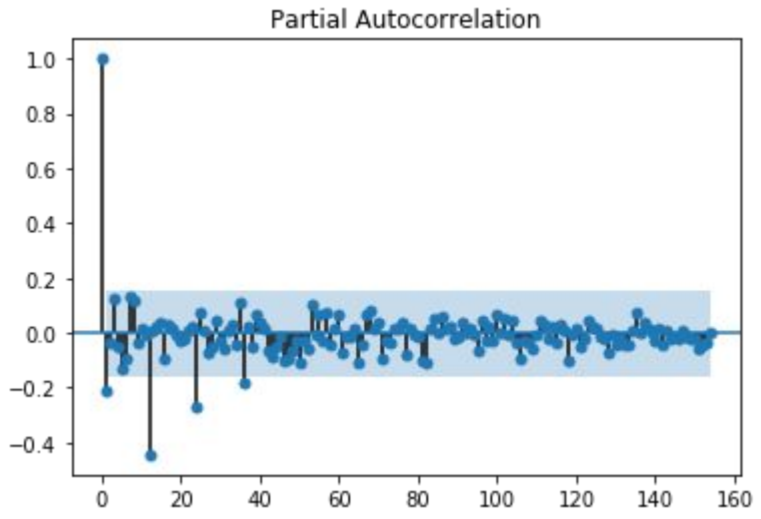
- In general, a partial correlation is a conditional correlation.
- It is the correlation between two variables under the assumption that we know and take into account the values of some other set of variables.

Python for Finance

- 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:



Python for Finance

- 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.

Python for Finance

- 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.

Python for Finance

- 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.

Python for Finance

- 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!

Pandas Time Series Exercise

SET TWO SOLUTIONS

PIERIAN DATA