lesson 11

Time Series Analysis

Python for Financial Analysis

Rajah Chacko

|elvtr|

Syllabus Review

1. Introduction to Python: Python in Finance
2. Python Basic Syntax: Importing Libraries
3. Working with Pandas
4. Pandas Underneath the Hood: Working with NumPy
5. Data Wrangling and Visualization
6. Extracting Financial Insights from Charts and Graphs
7. Financial Calculations with Python: Part 1
8. Financial Calculations with Python: Part 2
9. CAPM and Portfolio Management
10. Linear Regression
11. **Time Series Analysis**
12. Algorithmic Trading

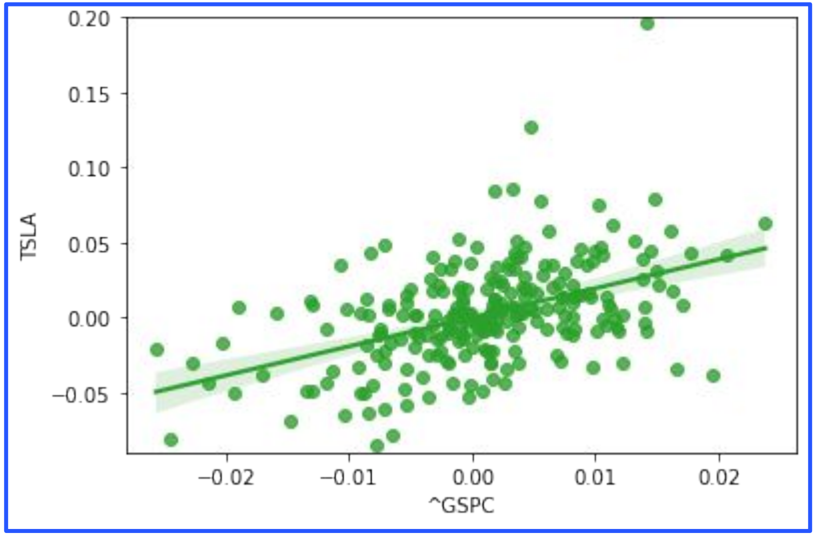
\+ **Bonus Class:** Cryptocurrency Beyond the Basics with a Fintech Guest Speaker

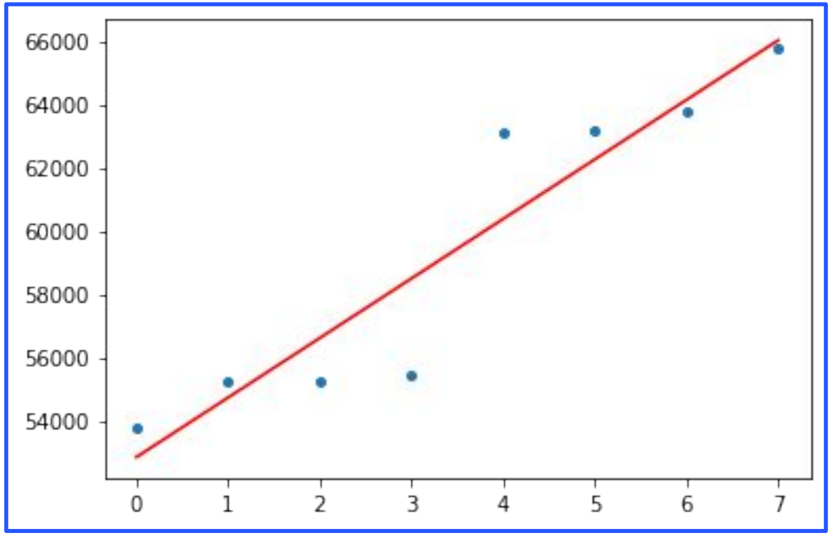
Class agenda

- Differences between OLS and time series
- How Pandas can support time series
- Calculating SMA and EMA and defining trends
- Using statsmodels to find the trend and cycles
- ARIMA
- Graphing monthly prices with daily data
- Pythonic: Financial APIs, PyPi, and Financial libraries

OLS vs. time series

- Time series always has (equally spaced) time on the x-axis
- Recall from chapter 10. SPX returns on x-axis (left). Years on x-axis (right)





Differences between OLS and time series

- We'll look at USDA Milk Production
- Pandas
  A. Simple Moving Average (SMA)
     a. df['6-month-SMA'] = df['price'].rolling(window=6).mean()
  B. Exponentially weighted Moving Average (EMA)
     a. df['EMA-12'] = df['price'].ewm(span=12).mean()
     b. Follows faster, fewer missing startups

Time series - statsmodels

- Statsmodels - we apply tools to extract several items
   A. Hodrick-Prescott filter separates a timeseries into
      a. Trend
      b. Cycle
   B. Will also look at an ETS model (Error - Trend - Seasonality)

Calculating SMA and EMA and defining trends

- Simple Moving Average
  A. For 14-day SMA, calculate average of last 14 days
  B. Roll the 14-day frame forward with each subsequent day
  C. Has a lag
  D. Example: milk_df['12-month-SMA'] = milk_df['milk_B_lbs'].rolling(window=12).mean()
- Exponentially-weighted Moving Average
  A. Last day more heavily weighted
  B. No lag
  C. Example: milk_df['EMA-12'] = milk_df['milk_B_lbs'].ewm(span=12).mean()

Time series - ARIMA

- AutoRegressive Integrated Moving Average - think AR + I + MA
  A. Non-seasonal ARIMA
  B. Seasonal
- ARIMA, indicated by 3 non-negative integers (p, d, q)
  A. AutoRegression – AR (p) – performs a regression on self (“auto”)
     i. p = number of lag observations
  B. Integrated – I (d) – takes differences to make the time series stationary
     i. d = number of differences
  C. Moving Average – MA(q) – tries to minimize residual error on a moving avg
     i. q = size of MA window
  D. Generally set AR (p) or MA (q), but not both

Time series - ARIMA

- AutoCorrelations
  A. Autocorrelation (ACF) - Better at identifying MA models
     a. How much does it autocorrelate when when lagged by p units?
     b. Gradual decline
     c. Sharp drop-off
  B. Partial Autocorrelation (PACF) - Better at identifying AR models
     a. Takes into account partial correlation between RHS variables
     b. Gradual decline suggests MA
     c. Sharp drop after lag “k” means use an AR with p=k
- More about stationary = constant mean and variance
  A. Detrend to get a constant mean
  B. ADF (Augmented Dickey-Fuller) test helps us
  C. If not stationary, we transform it with the p, d, q

What ARIMA is used for (and what it's not)

- Used for macroeconomic variables
- Used for many of the climate change models
- Not great for stocks and securities
    A. Time is not the only driver.
    B. Market (=trader's) sentiment exogenous to time

Adjusting your time windows

- Can turn hourly into daily data, daily into monthly, quarterly, or annual
- See references (or module) for resample
- Can interpolate on smaller periods

Financial APIs, PyPi, and Financial libraries

- Financial APIs
  A. These are for getting data off a server and into Python
  B. We have used tiingo for this
  C. Top 7 Best Stock Market APIs (for Developers): https://rapidapi.com/blog/best-stock-api/
- PyPi
  A. Everything Python, good and bad
  B. Try “candlestick” or “stock API”
  C. I have used edgar 5.4.1 for accessing the SEC Edgar database
  D. Watch out for unloved packages!
- Financial libraries
  A. Quick search for “best python financial libraries”
  B. I have used backtrader to test trading systems

Assignment #11

Download several stock prices, and plot the prices and a simple moving average.

Go Deeper: Install a financial library or API of your choice and test out some features that interest you.

Resources

- Simple Moving Average (SMA)

  Reference: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.rolling.html

- Exponentially weighted Moving Average (EMA)

  Reference: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.ewm.html

  Formula for EMA: https://www.investopedia.com/terms/e/ema.asp

- Statsmodels

  Reference: https://www.statsmodels.org/stable/index.html

  Hodrick-Prescott (HP) filter: https://www.statsmodels.org/stable/examples/notebooks/generated/statespace_cycles.html

Resources

- ARIMA

  Reference:
  https://stats.stackexchange.com/questions/44992/what-are-the-values-p-d-q-in-arima

- Time Series

  https://www.kaggle.com/prashant111/complete-guide-on-time-series-analysis-in-python

  https://people.duke.edu/~rnau/411l696.htm

- ACF and PACF interpretation

  https://people.duke.edu/~rnau/411arim3.htm

  https://towardsdatascience.com/identifying-ar-and-ma-terms-using-acf-and-pacf-plots-in-time-series-forecasting-ccb9fd073db8

- Resample

  Reference:
  https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.resample.htm

  https://www.geeksforgeeks.org/python-pandas-dataframe-resample/

Q&A