Documentation of FXVOLAUTOMAT v1.0.py

The following code can be used to upload, clean and perform data pre-processing steps for the ML FX Vol trading automat.

Input Files (in same directory as code)

filename - **"DataTables.xlsx"** (Daily price data for Predictors dataset)

The functions used in this script are:

def euro_vanilla(S, K, T, rd,rf, sigma, option = 'call')

def getData()

def getDependentVariable(method = "BS")

def traintestsplit(X,Y,split)

def standardize(trainX,testX)

def plot_pca_components(x,y)

def model_pca(variance)

The structure of the code is as follows:

1. Read in the predictors from the **"DataTables.xlsx"** using function **getData().** The aim is to try and model the predictors based on the JPM FX Vol Automat research paper. The predictor list as defined by the paper is shown as follows for reference:

Table 1: Our dataset consists of 377 indicators from FX and cross-asset markets as well as macro data

A total of 377 market indicators are formed across various asset classes. We use a 10Y history of these indicators, with daily sampling.

Market data type	Market data	Level	1 week change	1M change	Count
FX realised vols	2M realised vols in USD vs G10, MXN, BRL, ZAR, TRY, NR and KRW	X	X	X	45
FX ATM vols	1M, 3M and 1Y ATM in same pairs	X	X	X	135
FX skews	3M 25D RRs	X	X	X	45
FX spots	FX Spots in 15 G10 and EM USD pairs		X	X	30
Depo rates / Basis	3M FX Forward drops		X	X	30
Interest Rates	10Y Gov yields: US, Japan, UK, Germany, France, Italy, Spain, Australia		X	X	16
Equity Indices	S&P, Nikkei, FTSE, E-Stoxx, ASX, Mexbol, Bovespa, KOSPI, Hang Seng		X	X	18
Commodities	Gold and Brent spot		X	X	4
Credit spreads	CDX IG and HY, iTraxx spread indices		X	X	6
EASI indices	Global, US, CAD, EU, UK, CHF, NOK, SEK, Japan, AU, NZ, China		X	X	24
IMM positions	USD, EUR, JPY, GBP, CHF, CAD, AUD, NZD, MXN, RUB, Gold		X	X	24

Source: J.P. Morgan.

   a. The predictor data is stored in the above xlsx file under the following sheet names:

i. FX SPOT, ATM VOLS, 3M 25D RR, 3M DEPOSIT RATES, 10Y YIELD, EQUITY INDICES, COMDTY, CREDIT SPREADS, JPM EASI, IMM POSITIONING

b. We store the daily levels for the following predictors: FX 2M REALIZED VOLS, FX ATM VOLS, and FX SKEWS

c. For all the remaining predictors we also alculate both the 1 week change and 1M change and store these.

d. The dataframe **df_main** stores all the modified 373 predictors to be further used for data manipulation and input to ML algorithms

2. We use 2 methods to calculate the dependent variable namely – Black Scholes Calculation method and the Realized – Implied Volatility measure. We use the function **getDependentVariable(method = "BS")** to pre-process the dependent variable and store this in the dataframe **Y**

   a. When method is "BS" -

      i. We calculate the value of the ATM Put and ATM Call Options on the daily dataset using the Black Scholes option pricing formula defined for FX options via function **euro_vanilla(S, K, T, rd,rf, sigma, option = 'call')**

      ii. The call and put option prices are added to get the price of the ATM Straddle and stored in dataframe **option**

      iii. We calculate % gain in the price from the 1M ahead spot to current Strike price to calculate in the moneyness of the ATM straddle, subtract this from the cost of the ATM straddle from the above step and store this in **option["Payoff"]**

      iv. We segregate this data set to find a uniformly distributed percentages of positive payoffs and negative payoffs and store these as +1 and -1 respectively in dataframe **Y**

   b. When method is "RV- IV" –

      i. We calculate the 1M ahead realized volatility for EURUSD using the daily returns for EURUSD spot stored in **df_returns["EURUSD CURNCY"]** and store this in dataframe - **df_1MRV**

      ii. **RV_IV** dataframe then holds the realized – Implied vols difference

      iii. We segregate this data set to find a uniformly distributed percentages of positive payoffs and negative payoffs and store these as +1 and -1 respectively in dataframe **Y**

3. Once the dependent variable stored in Y and predictors stored in **df_main** are calculated, we resample both these to Business Month data sample only from period 2007 to 2016 as done in the JPM paper.
4. We use an 80% and 20% split ratio for the training and test data set and split these using the function **traintestsplit(X,Y,split)**
5. We standardize the predictors as the next step in data processing using the function **standardize(trainX,testX)**
6. The next step towards data processing is to get the reduced dimensionality predictors/features using Principal Component Analysis for 86.1%, 91.9% and 98.6% explained variance of the dataset using the function **model_pca(variance)**