Pairs Trading Strategy Design & Back test (TS)

CQF June 2023 Cohort

Rahul Yadav

Table of Contents

1 Introduction

This project explains the concept of pairs trading and implements the strategy using stocks in Python. In most basic form pairs trading tries to identify two financial securities that are co-integrated and tries to exploit the relative mispricing. This is a market-neutral trading strategy as it involves taking a long position in one security and a short position in another security. Then it focuses on the spread between the prices of these two securities and how it evolves over a period of time. If this spread goes beyond predefined bounds in a positive or negative direction then long/short positions are taken in the two identified securities. When this spread reverts within the predefined bounds positions are reversed.

Python is used for the implementation of the project. The following numerical methods are re-coded from first principles and standard implementation in Python of these numerical methods is **not** used.

- Linear Regression
- Engle-Granger procedure
- ADF tests
- Error correction model (ECM)
- Calibration of Ornstein-Uhlenbeck process
- Backtesting
- Backtesting performance

2 Statistical Methods for Pairs Trading

This section provides a summary of the data and the statistical approaches applied in pairs trading. In addition to describing the implementation process, it will delve into the outcomes achieved through the specific statistical test.

2.1 Data

To identify the pair of stocks which can be used for pairs trading constituents of Nifty 50 are used. Nifty 50 is an Indian benchmark equity index consisting of 50 representative companies from different sectors in India. Data of daily closing prices from 1st Jan 2020 to 30 Dec 2023 is used in the project. This data is further split into train and test data. Training data consists of closing prices from 1st Jan 2020 to 30 Jun 2021. Similarly, test data consists of closing prices from 1st Jul 2021 to 30 Dec 2023. Essentially total of three years of closing price time series is used and it is split into one and half years for training the model and the remaining one and a half is used to backtest the model out of the sample. This data is sourced from Yahoo Finance using Python. However, data for only 48 stocks was available, hence only 48 stocks are studied further.

The daily closing value of the Nifty 50 Index is sourced from Yahoo Finance using Python. It is used to calculate the beta of the returns from the pairs trading strategy with respect to benchmark index.

To calculate the sharpe ratio of the pair's trading strategy, a risk-free rate is needed. The yield of the 10-year treasury bonds issued by Indian Government is used as a proxy for risk-free rates. These yields are downloaded from a website (https://in.investing.com/rates-bonds/india-10-year-bond-yield-historical-data) and saved in CSV file. This CSV file is later imported into the code and used further.

2.2 Linear Regression

Linear regression is a statistical method used to model the relationship between a dependent variable and one or more independent variables. It assumes that there is a linear relationship between the variables, meaning that changes in the dependent variable are associated with constant proportional changes in the independent variable(s).

The general form of a simple linear regression model with n independent variable can be expressed as:

$$Y = \beta_1 + \beta_2 X_1 + \beta_3 X_2 + \cdots + \beta_n X_n + \epsilon$$

Where,

- Y is dependent variable
- $X_1, X_2, \ldots, X_n$ are independent variables
- $\beta_1, \beta_2, \beta_2, \ldots, \beta_n$ *are the parameters which are needed to be estimated from data*
- ϵ is an error term

A dataset containing k values of independent and dependent variables can be represented as below in matrix form.

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ \vdots \\ Y_n \end{bmatrix}_{n\times 1} = \begin{bmatrix} 1 & X_{11} & X_{21} & \ldots & X_{k1} \\ 1 & X_{12} & X_{22} & \ldots & X_{k2} \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ 1 & X_{1n} & X_{2n} & \ldots & X_{kn} \end{bmatrix}_{n\times k} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \vdots \\ \beta_n \end{bmatrix}_{k\times 1} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \vdots \\ \epsilon_n \end{bmatrix}_{n\times 1}$$

This can be rewritten more simply as:

$$y = X\beta + \epsilon$$

This is assumed to be an accurate reflection of the real world. The model has a systematic component $(X\beta)$ and a stochastic component ϵ. Our goal is to obtain estimates of the population parameters in the β vector.

These can be estimated using the below equation.

$$\widehat{\beta} = (X^T X)^{-1} X^T Y$$

In the above equation X^T is the transpose of the matrix X. Derivation of this equation can be found here[1].

[1] https://web.stanford.edu/~mrosenfe/soc_meth_proj3/matrix_OLS_NYU_notes.pdf

2.2.1 Implementation Details

Implementation of linear regression is done independently in Python file **LinearRegresion.py** in the function **ols_linear_regression.** Details are below.

ols_linear_regression(y, x, constant='Y')

Parameters:

- **y :** pandas series
  vector of k*1 of independent variable Y, having the name of the Y variable as the header
- **x** : dataframe
  vectors of k*n of dependent variable X1,X2, …, Xn having name of the all variables as the header
- **constant :** str, optional
  The default is 'Y' which means it will add a vector of k*1 at the start of the above dataframe X so that constant regression parameter β_0 is considered in the calculation. To not include contant you can pass any other string than 'Y'.

Returns:

- **List**

List containing three elements.

1) A list containing the main regression result X variable names, their corresponding betas, standard error, t statistic and p-value of the beta.
2) A list containing dependent variable name, R-squared, Adj. R-squared, Log Likelihood, AIC and BIC.
3) A list containing residuals ϵ_t

   This implementation is further used in the Augmented Dickey-Fuller (ADF) test, Engle-Granger procedure, etc. Its user acceptance testing is done by comparing the results with the standard function available in Python.

2.3 Stationarity of time series

It is essentially for time series to be stationary so that we can develop models and forecasts. A time series is strictly stationary if its joint probability distribution remains constant over any n observations. Let X_t be a time series then it is stationary if its joint probability distributions over any n observations i.e. $\{\{X_{t_1}, X_{t_2}, X_{t_3}, \ldots.., X_{t_n}\}$ (i.e. realizations at times $\{t_1, t_2, t_3, \ldots\ldots, t_{n\}})$ is same as $\{X_{t_{1+k}}, X_{t_{2+k}}, X_{t_{3+k}}, \ldots.., X_{t_{n+k}}\}$ (i.e. realizations at times $\{t_{1+k}, t_{2+k}, t_{3+k}, \ldots\ldots, t_{n+k}\})$.

This is very strong condition and difficult to access in practice. Hence, we have another version of stationarity which is called weak stationarity. A time series is called weakly stationary if its statistical properties do not vary with time and remain constant. Let X_t be a time series then it is stationary if the following three properties remain constant at any time t.

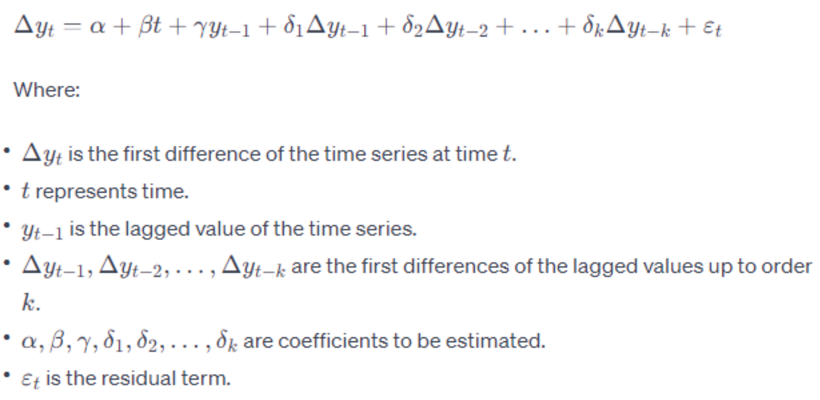
- $E(X_t) = \mu$
- $V(X_t) = \sigma$
- $Cov(X_t, X_{t+k})$ *depends only on lag k*

Henceforth, I will use the term **stationary** to mean weak stationarity.

2.3.1 Augmented Dickey-Fuller (ADF) test

The Augmented Dickey-Fuller (ADF) test is a statistical test commonly used in econometrics and time series analysis to determine whether a unit root is present in a time series dataset or not. A unit root implies that a time series is non-stationary, meaning its statistical properties (such as mean and variance) change over time.

The ADF test is typically conducted using the following regression equation. Note that this equation includes time trend also which is optional and not implemented in this project.



$$\Delta y_t = \alpha + \beta t + \gamma y_{t-1} + \delta_1 \Delta y_{t-1} + \delta_2 \Delta y_{t-2} + \ldots + \delta_k \Delta y_{t-k} + \varepsilon_t$$

Where:

- Δy_t is the first difference of the time series at time t.
- t represents time.
- y_{t-1} is the lagged value of the time series.
- $\Delta y_{t-1}, \Delta y_{t-2}, \ldots, \Delta y_{t-k}$ are the first differences of the lagged values up to order k.
- $\alpha, \beta, \gamma, \delta_1, \delta_2, \ldots, \delta_k$ are coefficients to be estimated.
- ε_t is the residual term.

Hypothesis Testing:

The ADF test involves testing the null hypothesis (H0) that a unit root is present against the alternative hypothesis (H1) that the time series is stationary. The test statistic from the ADF test is compared to critical values to determine statistical significance.

Null Hypothesis (H0): The time series has a unit root; it is non-stationary.

Alternative Hypothesis (H1): The time series is stationary (no unit root).

Key Interpretation:

- If the test statistic is less than the critical value, you reject the null hypothesis in favour of stationarity.
- If the test statistic is greater than the critical value, you fail to reject the null hypothesis, suggesting non-stationarity.

2.3.2 Implementation Details:

Implementation of ADF test is done independently in Python file **LinearRegresion.py** in the function **ADF_test.** Details are below.

ADF_test(df, lags)

Parameters:

- **df : names series or Dataframe column**
- **lags : int**
  number of lags to be used in the test

Returns:

- **List**

List containing six elements.

1) test statistic
2) Critical values against important p-values as per the Dickey-Fuller as tabulated by MacKinnon (2010 update)
3) AIC
4) BIC
5) Number of lags used

This implementation is used to check the stationarity of the stocks. Its user acceptance testing is done by comparing the results with the standard function available in Python.

2.3.3 Results of the Stationarity Test

The stationarity of all the available 48 stocks of the Nift50 benchmark index is checked using the above function. Closing prices of the stocks and 1 lag term are used. A test statistic is compared with a **5% critical value**. If the critical value is less than the test statistic then stock is stationary otherwise stock is not stationary. Critical values are {'1% critical value:': -3.448196541708585, '2.5% critical value:': -3.1298565198541763, **'5% critical value:': -2.869404683789669**, '10% critical value:': -2.5709597356805545}.

The results of these tests are tabulated in Table 1 and shown as a heatmap in Figure 2. Out of 48 stocks, 4 are stationary and 44 are non-stationary. Please note that only test period data is used for this.

Table 1 Stationarity results

Ticker	Test Statistic	AIC	BIC	No of lags Used	Stationary
ADANIPORTS.NS	-3.61167	3184.65793	3196.39032	1	Yes
EICHERMOT.NS	-3.43491	3883.15507	3894.88746	1	Yes
HDFCBANK.NS	-2.96089	3408.58982	3420.32221	1	Yes
AXISBANK.NS	-2.87599	2991.09077	3002.82316	1	Yes
KOTAKBANK.NS	-2.6986	3608.99445	3620.72684	1	No
UPL.NS	-2.62544	3025.53379	3037.26618	1	No
GRASIM.NS	-2.59442	3524.07149	3535.80388	1	No
MARUTI.NS	-2.5699	4684.29753	4696.02992	1	No
SBIN.NS	-2.56877	2629.591	2641.32339	1	No
CIPLA.NS	-2.46397	3005.30688	3017.03927	1	No

ICICIBANK.NS	-2.39301	2908.56855	2920.30094	1	No
BAJAJ-AUTO.NS	-2.3886	4080.27342	4092.00581	1	No
TATAMOTORS.NS	-2.27371	2848.33272	2860.06511	1	No
INDUSINDBK.NS	-2.1847	3394.52352	3406.25591	1	No
BRITANNIA.NS	-2.13611	3858.27243	3870.00482	1	No
DRREDDY.NS	-2.09505	4245.11331	4256.8457	1	No
JSWSTEEL.NS	-2.07382	3068.67173	3080.40412	1	No
TCS.NS	-2.03827	3918.22965	3929.96204	1	No
LT.NS	-1.98089	3481.30637	3493.03876	1	No
IOC.NS	-1.97552	1264.71267	1276.44506	1	No
HINDUNILVR.NS	-1.94837	3627.59548	3639.32787	1	No
NESTLEIND.NS	-1.85396	5010.7023	5022.43469	1	No
HINDALCO.NS	-1.85213	2850.82863	2862.56102	1	No
ULTRACEMCO.NS	-1.84521	4553.65331	4565.3857	1	No
TATACONSUM.NS	-1.83668	2881.45595	2893.18834	1	No
COALINDIA.NS	-1.80904	2040.40388	2052.13627	1	No
POWERGRID.NS	-1.80284	1745.36404	1757.09643	1	No
ASIANPAINT.NS	-1.79575	3967.15511	3978.8875	1	No
SBILIFE.NS	-1.76096	3145.43639	3157.16878	1	No
TATASTEEL.NS	-1.7441	1866.82574	1878.55813	1	No
INFY.NS	-1.70846	3409.79356	3421.52595	1	No
ONGC.NS	-1.66477	1960.19038	1971.92277	1	No
HEROMOTOCO.NS	-1.65927	3915.93433	3927.66672	1	No
RELIANCE.NS	-1.65283	3776.11118	3787.84357	1	No
BHARTIARTL.NS	-1.65003	2816.78167	2828.51406	1	No
HCLTECH.NS	-1.6138	3209.13922	3220.87161	1	No
BAJFINANCE.NS	-1.50694	4664.65894	4676.39133	1	No
SUNPHARMA.NS	-1.49945	2896.03704	2907.76943	1	No
NTPC.NS	-1.49903	1662.97959	1674.71198	1	No
BAJAJFINSV.NS	-1.40547	3563.07724	3574.80963	1	No
TITAN.NS	-1.30097	3777.12945	3788.86184	1	No
WIPRO.NS	-1.29172	2744.88177	2756.61416	1	No
DIVISLAB.NS	-1.2681	4252.03248	4263.76487	1	No
ITC.NS	-1.13862	2009.63296	2021.36535	1	No
TECHM.NS	-1.08337	3437.86908	3449.60147	1	No
M&M.NS	-1.07509	3126.75968	3138.49207	1	No
HDFCLIFE.NS	-1.02909	2775.55558	2787.28797	1	No
BPCL.NS	-0.87755	2522.99394	2534.72633	1	No

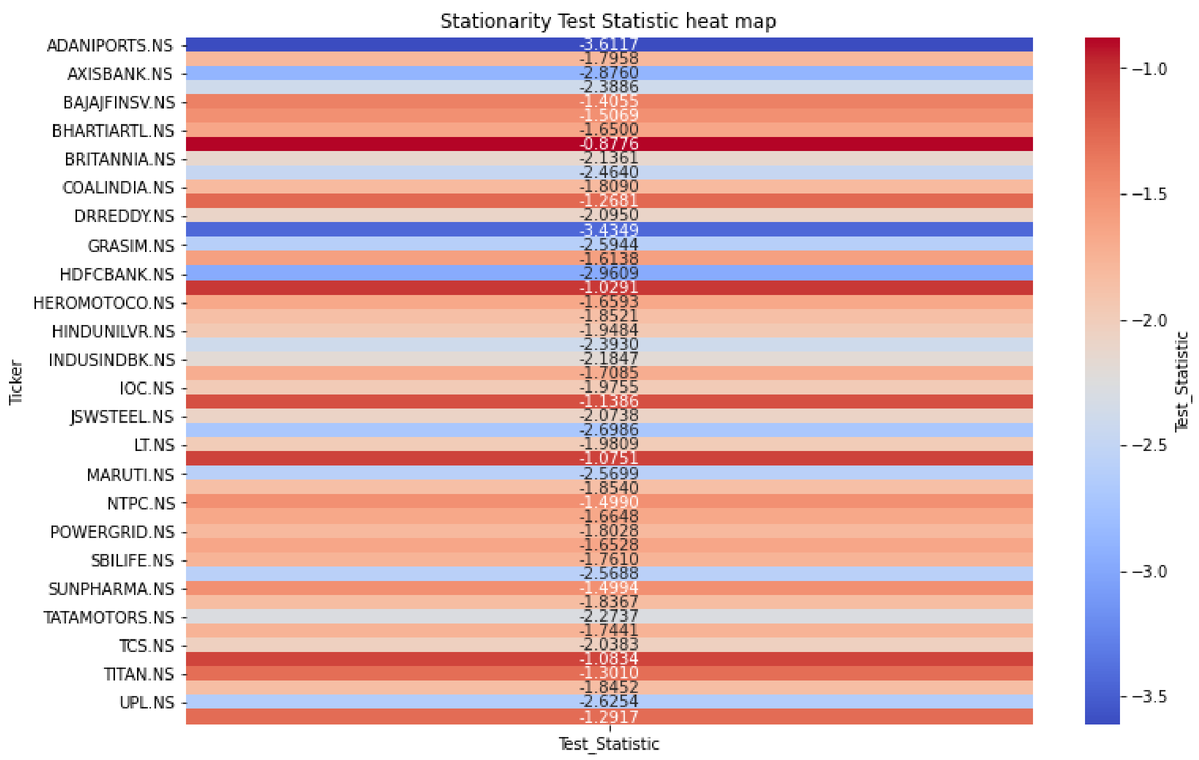


Figure 1 Stationarity results

2.4 Cointegration

Cointegration is a statistical property that occurs when two or more non-stationary time series variables move together over time, even though each of them individually may not be stationary. In other words, cointegration suggests a long-term relationship between variables, indicating that they share a common stochastic trend.

Consider two non-stationary time series variables, Y_t and X_t. If they are cointegrated, there exists a linear combination of them that is stationary. The cointegration relationship can be expressed as:

$$\Delta Y_t = \alpha + \beta \Delta X_t + \varepsilon_t$$

Where:

- ΔY_t is the first difference of Y_t.
- ΔX_t is the first difference of X_t.
- α and β are coefficients to be estimated.
- ε_t is a stationary error term.

The key aspect here is that the linear combination $\alpha + \beta \Delta X_t$ results in a stationary series. This implies that, in the long run, Y_t and X_t move together, and any deviations from their long term relationship (captured by ϵ_t) are stationary.

Implications:

Long-Run Relationship: Cointegration suggests a stable long-run relationship between variables. For example, in finance, it might indicate that two financial instruments move together over the long term.

Error Correction Mechanism: Cointegrated variables often exhibit an error correction mechanism. If there is a short-term deviation from the long-run relationship, adjustments occur to bring the variables back to their equilibrium.

2.4.1 Implementation of Cointegration Test:

The **Engle-Granger** two-step procedure is commonly used to test for cointegration:

1) **Estimate Regression:** Regress one variable on the other and obtain the residuals.
2) **Test Residuals:** Apply a unit root test (e.g., ADF test) to the residuals. If the residuals are stationary, the variables are cointegrated.

To do the first step **ols_linear_regression** function explained in **section 2.2** is used. For the second step, **ADF_test** function explained in section 2.3 is used. In the second step, residuals of the regression obtained in step 1 are sent for stationarity test.

2.4.2 Results of Cointegration Test:

In the previous section, we found out that out of 48 stocks 44 are non-stationary. All possible pairs of 44 stocks are made (976) and a cointegration test is applied on them to check for stationarity. Please note that only test data is used for this. '5% critical value:': -2.869404683789669 is used in the test. Out of 976 pairs spreads 145 were found to be stationary and they are further used in the analysis. Table 2 shows the result of the top 10 pairs for illustration purposes.

TickerPairs	Test Statistic	AIC	BIC	No of lags Used	Stationary
('BRITANNIA.NS', 'HINDUNILVR.NS')	-5.28336	3795.15756	3806.88995	1	Yes
('NTPC.NS', 'RELIANCE.NS')	-4.24587	1835.20409	1846.93648	1	Yes
('NESTLEIND.NS', 'WIPRO.NS')	-4.04506	4976.09361	4987.826	1	Yes
('BAJFINANCE.NS', 'BAJAJFINSV.NS')	-4.03858	4312.77269	4324.50508	1	Yes
('COALINDIA.NS', 'ITC.NS')	-3.9477	2065.07055	2076.80294	1	Yes
('ICICIBANK.NS', 'SBILIFE.NS')	-3.86039	2944.0113	2955.74369	1	Yes
('ONGC.NS', 'TITAN.NS')	-3.83884	2036.25143	2047.98382	1	Yes
('COALINDIA.NS', 'RELIANCE.NS')	-3.80185	2011.20003	2022.93242	1	Yes
('ICICIBANK.NS', 'SBIN.NS')	-3.74411	2719.07657	2730.80896	1	Yes
('CIPLA.NS', 'HEROMOTOCO.NS')	-3.72848	3103.56506	3115.29745	1	Yes

Table 2 Cointegration results (Stationarity of residuals)

2.5 Error correction model (ECM):

When cointegration is found, an error correction mechanism (ECM) can be derived from the estimated cointegrating regression. The ECM represents the short-term adjustments that bring the variables back to their long-term equilibrium after a deviation. The ECM equation is typically expressed as follows:

$$\Delta Y_t = \phi_0 + \phi_1 \Delta X_t - (1-\alpha)\widehat{\epsilon_{t-1}}$$

$\widehat{\epsilon_{t-1}}$ are the stationary fitted residuals from the above step in section 2.4.1. Hence above equation becomes.

$$\Delta Y_t = \phi_0 + \phi_1 \Delta X_t - (1-\alpha)(Y_{t-1} - \beta_0 - \beta_1 X_{t-1})$$

Here, it is required to check the significance of the $(1-\alpha)$ coefficient to confirm the error correction mechanism works.

2.5.1 Implementation of Error Correction Model (ECM):

Implementation of ECM is done independently in Python file **LinearRegresion.py** in the function **vece.** Details are below.

vece(stock1,stock2)

Parameters:

- **stock1 : Dataframe column or named series**
- **stock2 : Dataframe column or named series**

Returns:

- **List**
  It has 4 elements.
  1) Based on significance level of $(1-\alpha)$ dependent variable name
  2) Similarly based on significance level of $(1-\alpha)$ independent variable name
  3) Regression output when dependent variable is stock1
  4) Regression output when dependent variable is stock2

Basically, it runs ECM model in both directions then selects Y(independent variable) and X(dependent variable) and also gives details about the regression results of the ECM model.

2.5.3 Results of Error Correction Model (ECM):

For the each of the 145 pairs whose spread is stationary ECM model is run. The statistical significance of the coefficient of the error term is checked at 5% level. If it is significant then then error correction i.e., mean reversion of the spread for that pair is happening. For illustration purpose results of only one pair is shown below.

Ticker Pair is (ASIANPAINT.NS, BAJFINANCE.NS).

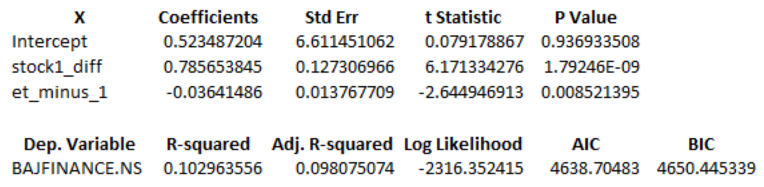
As we can see from Table 3 and 4 t-statistic of ECM is more significant when ASIANPAINT.NS is dependent variable. Hence function outputs mean reversion of spread is happening and ASIANPAINT.NS should be dependent variable.

Table 3 ECM result when dependent variable is ASIANPAINT.NS

X	Coefficients	Std Err	t Statistic	P Value
Intercept	-0.256503369	2.572084402	-0.099725876	0.920616409
stock2_diff	0.118662905	0.019281338	6.154287847	1.97619E-09
et_minus_1	-0.051376298	0.016412515	-3.130312334	0.001886122

Dep. Variable	R-squared	Adj. R-squared	Log Likelihood	AIC	BIC
ASIANPAINT.NS	0.109636859	0.104784743	-1967.043242	3940.086484	3951.826993

Table 4 ECM result when dependent variable is BAJFINANCE.NS



X	Coefficients	Std Err	t Statistic	P Value
Intercept	0.523487204	6.611451062	0.079178867	0.936933508
stock1_diff	0.785653845	0.127306966	6.171334276	1.79246E-09
et_minus_1	-0.03641486	0.013767709	-2.644946913	0.008521395

Dep. Variable	R-squared	Adj. R-squared	Log Likelihood	AIC	BIC
BAJFINANCE.NS	0.102963556	0.098075074	-2316.352415	4638.70483	4650.445339

At this stage all 145 pairs found to have mean reversion of the spreads and all will be considered in the further steps.

2.6 Fitting Ornstein-Uhlenbeck(OU) Process to Spread

The OU process is represented by the following stochastic differential equation:

$$dX_t = \theta(\mu - X_t)dt + \sigma dW_t$$

Where:

- X_t is the price of the asset at time t.
- θ is the speed of mean reversion.
- μ is the long-term mean or equilibrium price.
- σ is the volatility of the process.
- W_t is a Wiener process (Brownian motion).
- dt is the infinitesimal time increment.

In the context of pairs trading, the OU process is often used to model the spread between two co-integrated assets. If Y_t and X_t are the prices of two assets, the spread S_t can be modelled using OU process.

$$dS_t = \theta(\mu - S_t)dt + \sigma dW_t$$

Here, $S_t = Y_t - \beta_0 - \beta_1 X_t$ and β_1 is the co-integration coefficient.

Above SDE needs to be calibrated so that it can be used further to get the mean and volatility of the spreads. To do this we can run the following regression on the spreads.

$$S_t = C + B * S_{t-1} + \epsilon_{t,\tau}$$

And now we can get the parameters that are required to fit OU process as below. In our analysis $\tau = 1$ as our observations are daily.

$$\theta = -\frac{\ln B}{\tau}$$

$$\mu = \frac{C}{1 - B}$$

$$\sigma_{eq} = \sqrt{\frac{SSE * \tau}{1 - e^{-2\theta\tau}}}$$

Where, SSE is sum of squared residuals of your regression for S_t , for this AR (1) process. Above volatility equation is for only creating the trading signals and this volatility is different from the volatility of O-U process. We will generate signals and positions for assets Y and X.

Bounds $S_t = \mu \pm Z\,\sigma_{eq}$ give entry signal, exit at while $S_t \approx \mu$.

Z is a constant can be varied around 1 e.g. [0.7, 1,3]

In the context of the O-U process, the half-life refers to the time it takes for the variable to revert halfway back to its long-term mean or equilibrium value.

The half-life of the Ornstein-Uhlenbeck process can be calculated using the formula:

$$Half - Life = -\frac{\ln{(0.5)}}{\theta}$$

All the above formulas are taken from the CQF project tutorial for pairs trading notes.

2.6.1 Implementation of Fitting Ornstein-Uhlenbeck(OU) Process to Spread

Implementation of calibration of is done independently in Python file **LinearRegresion.py** in the function **fit_ou_process.** Details are below.

fit_ou_process(y,x,delta_t = 1)

Parameters:

- **y : Dataframe column or named series**
- **x : Dataframe column or named series**
- **delta_t : int**
  Default value is 1

Returns:

- **Dictionary**
  1) 'y_ticker' name
  2) 'x_ticker' name
  3) 'intercept' this gives the intercept of the regression of y & x
  4) 'b_coint' this gives the beata of the regression of y & x

  All the below parameters are calculated for the spread AR(1) i.e. calibration of O-U process and their meaning is explained in the above section 2.6.1.

  5) 'C'
  6) 'B'
  7) 'theta'
  8) 'mu_e'
  9) 'sigma_ou'
  10) 'sigma_eq'
  11) 'half_life'

In above implementation following steps are taken.

1) Regression of y & x using ols_linear_regression and get the result of the regression
2) Get the residuals i.e. spread St between two series.
3) Run AR(1) on St using ols_linear_regression
4) Calculate all the calibration parameters of the O-U process

2.6.2 Results Ornstein-Uhlenbeck(OU) Process to Spread

For all the selected 145 pairs, spread is calculated and O-U process is fitted to the spread and the results of all the pairs are saved.

For illustration purposes result of only 15 pairs are shown in Table 5.

Sr.No.	y_ticker	x_ticker	intercept	b_coint	C	B	theta	mu_e	sigma_ou	sigma_eq	half_life
1	ASIANPAINT.NS	BAJFINANCE.NS	1194.8383	0.2768	-0.3081	0.9392	0.0628	-5.0647	55.5171	156.7016	11.0446
2	ASIANPAINT.NS	BAJAJFINSV.NS	1791.8957	0.8402	-0.6827	0.939	0.063	-11.1875	52.2091	147.1182	11.0077
3	GRASIM.NS	ASIANPAINT.NS	-42.6877	0.5258	1.1685	0.9491	0.0522	22.9586	32.2917	99.9059	13.2696
4	ASIANPAINT.NS	HCLTECH.NS	728.7906	2.0628	-0.3387	0.9523	0.0489	-7.0965	55.2629	176.7091	14.1744
5	HINDALCO.NS	ASIANPAINT.NS	-183.1275	0.2056	0.3188	0.9652	0.0354	9.165	14.9352	56.13	19.5804
6	IOC.NS	ASIANPAINT.NS	16.7036	0.0197	0.0396	0.9579	0.043	0.9394	1.5782	5.3798	16.1093
7	ASIANPAINT.NS	INFY.NS	887.9926	1.3095	-0.9151	0.948	0.0534	-17.6029	52.4768	160.6004	12.9842
8	NESTLEIND.NS	ASIANPAINT.NS	9642.1709	2.8355	-2.0605	0.9488	0.0526	-40.2282	211.3727	651.8241	13.1831
9	SBILIFE.NS	ASIANPAINT.NS	98.4755	0.3284	0.5584	0.9389	0.0631	9.1364	20.8135	58.6057	10.9912
10	TCS.NS	ASIANPAINT.NS	1193.2689	0.7557	1.0733	0.9339	0.0684	16.2333	51.058	138.0389	10.1328
11	ASIANPAINT.NS	TATASTEEL.NS	1854.5257	9.8848	-0.8133	0.9517	0.0495	-16.8409	59.6271	189.5087	14.0032
12	ASIANPAINT.NS	TECHM.NS	1717.052	0.98	-0.2753	0.9541	0.047	-5.9985	52.4167	170.9861	14.7515
13	ASIANPAINT.NS	WIPRO.NS	1426.6718	2.7831	-0.4438	0.945	0.0565	-8.0755	52.1115	154.9835	12.262
14	BAJAJ-AUTO.NS	BAJAJFINSV.NS	4222.8045	-0.3509	0.8191	0.9595	0.0413	20.2226	65.4016	227.4366	16.7648
15	BAJAJ-AUTO.NS	BHARTIARTL.NS	4782.572	-1.6397	1.3551	0.9497	0.0516	26.9282	69.0496	214.8712	13.4242

Table 5 calibration results of Fitting O-U process to spreads

‘intercept’ and ‘b_coint’ are the results of regression of ‘x_ticker’ on ‘y_ticker’. Residuals from this regression are fitted to O-U process and remaining columns give those results.

3. Backtesting

Backtesting is a crucial step in evaluating the effectiveness of a pairs trading strategy. It involves applying the strategy to historical data to simulate how it would have performed in the past. However, when backtesting pairs trading strategies, it's essential to avoid look-ahead bias to ensure realistic and unbiased results. Backtesting is performed on the test data which is separate and out of sample from the training data.

Look-Ahead Bias:

Look-ahead bias occurs when information that was not available at the time of making a trading decision is used in the backtesting process. In the context of pairs trading, avoiding look-ahead bias means that you should only use information available up to a specific point in time to make decisions that would have been made at that time. Hence, while implementing backtesting, a decision to enter/exit/status-quo is taken based on yesterday’s signal.

Position Sizing:

Initial capital of 100,000 is taken. Number of positions in each stock is calculated based on the b_{coint} and the closing prices at the start date of test data.

These equations represent the calculation of the holdings for y_ticker and x_ticker based on the initial capital, the cointegration coefficient (b_{coint}) and the initial prices of x_ticker and y_ticker at time t=0 i.e., first day of test data. The round function is used to round down the calculated holdings to the nearest whole number.

1. **Calculation of $y_{\text{ticker\_holdings}}$:**

$$y_{\text{ticker\_holdings}} = \text{round}\left(\frac{\text{initial\_capital}}{b_{\text{coint}} \cdot x_{\text{ticker\_0}} + y_{\text{ticker\_0}}}\right)$$

2. **Calculation of $x_{\text{ticker\_holdings}}$:**

$$x_{\text{ticker\_holdings}} = \text{round}\left(b_{\text{coint}} \cdot y_{\text{ticker\_holdings}}\right)$$

Also, total assets at any time is equal to the sum of holdings in stock y, holdings in stock x and cash. Cash will be the 0 at time t=0.

After that, it will be updated as follows.

If the trading signal is generated today then

$$\text{Cash}_i = \text{Cash}_{i-1} - y_{\text{ticker\_holdings}} - x_{\text{ticker\_holdings}}$$

Else if the trading signal generated is opposite of the previous signal e.g., log to short the spread. To do that you need to close current positions and take fresh positions in opposite directions on the same day.

$$\text{Cash}_i = \text{Total\_Assets}_{i-1} + \text{y}_{\text{stock\_holdings}} + \text{x}_{\text{stock\_holdings}}$$

Else there is no trading signal

$$\text{Cash}_i = \text{Cash}_{i-1}$$

Signal generation:

Depending on a spread and a trading signal yesterday generate a trading signal for today using the below criteria. On the first day of the test data, we calculate the spread and assign the trading signal as not applicable (NA). From the second day onwards rules as per Table 6 are applied to generate the trading signals.

Spread Yesterday	Trading Signal Yesterday	Trading Signal Today
> upper threshold	NA	Short the spread
< lower threshold	NA	Long the spread
upper threshold <= spread <= lower threshold	NA	No signal
> upper threshold	Short the spread	No signal
< lower threshold	Short the spread	Long the spread
<= mean and >= below threshold	Short the spread	No signal
> upper threshold	Long the spread	Short the spread
< lower threshold	Long the spread	Long the spread
>= mean and < upper threshold	Long the spread	No signal
> upper threshold	No signal	Short the spread
< lower threshold	No signal	Long the spread
upper threshold <= spread <= lower threshold	No signal	No signal

Table 6 Signal generation

3.1 Implementation of Backtesting and Signal Generation:

A function backtest is defined in Backtesting.py Python file. Function details are given below.

backtest (data_test, y_ticker, x_ticker, intercept, b_coint, mu_e, sigma_eq, k, capital, save_loc_chart)

Parameters:

- data_test : DataFrame
  containing test data for two tickets closing prices
- y_ticker : str
  name of y ticker
- x_ticker : str
  name of x ticker
- intercept, b_coint, mu_e, sigma_eq : results of O-U fitting process
- k : float
  is the threshold which will be used to calculate upper and lower threshold of spreads
- capital : float
  Initial capital for trading stratergy
- save_loc_chart : string
  save location for saving the charts

Returns:

DataFrame

A DataFrame containing backtesting trading signals and changes in Total Assets.

Also, it saves a chart containing the closing price and spread trend over the test data period.

3.2 Results of Backtesting and Signal Generation:

Table 7 gives an example of the dataframe produced by the above function. As you can see shifted spread is used to generate the signal. All the holdings and total assets change according to the trading signals. Figure 2 gives the trend of the closing prices of the tickers along with it shows the mean reverting nature of the spread between the tickers.

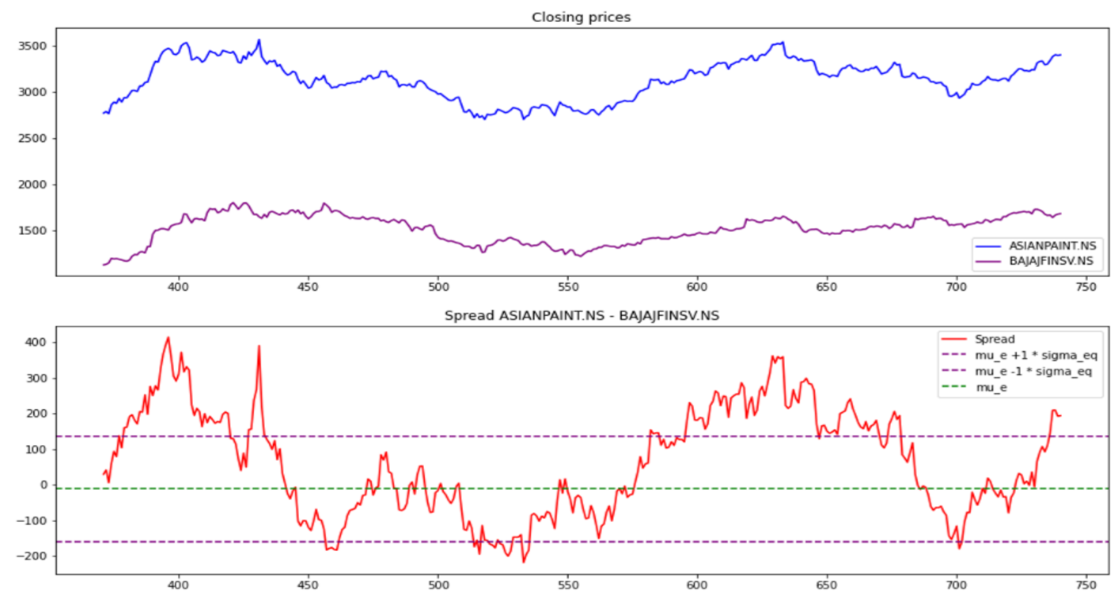
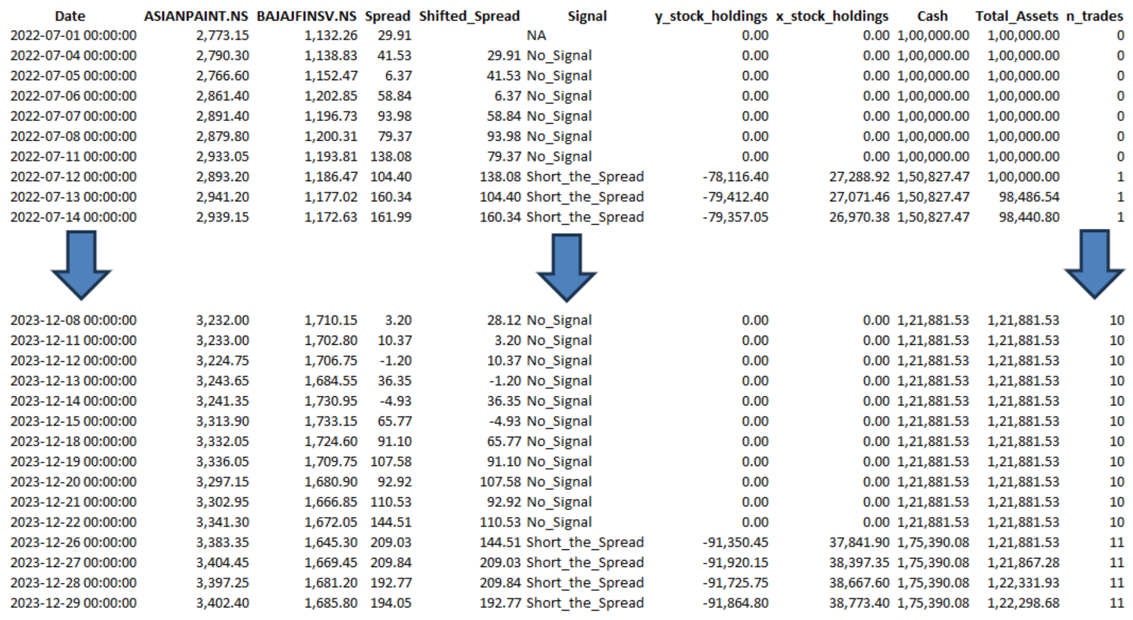


Figure 2 Spread and Closing prices ASIANPAINT.NS and BAJAJFINSV.NS



Date	ASIANPAINT.NS	BAJAJFINSV.NS	Spread	Shifted_Spread	Signal	y_stock_holdings	x_stock_holdings	Cash	Total_Assets	n_trades
2022-07-01 00:00:00	2,773.15	1,132.26	29.91		NA	0.00	0.00	1,00,000.00	1,00,000.00	0
2022-07-04 00:00:00	2,790.30	1,138.83	41.53	29.91	No_Signal	0.00	0.00	1,00,000.00	1,00,000.00	0
2022-07-05 00:00:00	2,766.60	1,152.47	6.37	41.53	No_Signal	0.00	0.00	1,00,000.00	1,00,000.00	0
2022-07-06 00:00:00	2,861.40	1,202.85	58.84	6.37	No_Signal	0.00	0.00	1,00,000.00	1,00,000.00	0
2022-07-07 00:00:00	2,891.40	1,196.73	93.98	58.84	No_Signal	0.00	0.00	1,00,000.00	1,00,000.00	0
2022-07-08 00:00:00	2,879.80	1,200.31	79.37	93.98	No_Signal	0.00	0.00	1,00,000.00	1,00,000.00	0
2022-07-11 00:00:00	2,933.05	1,193.81	138.08	79.37	No_Signal	0.00	0.00	1,00,000.00	1,00,000.00	0
2022-07-12 00:00:00	2,893.20	1,186.47	104.40	138.08	Short_the_Spread	-78,116.40	27,288.92	1,50,827.47	1,00,000.00	1
2022-07-13 00:00:00	2,941.20	1,177.02	160.34	104.40	Short_the_Spread	-79,412.40	27,071.46	1,50,827.47	98,486.54	1
2022-07-14 00:00:00	2,939.15	1,172.63	161.99	160.34	Short_the_Spread	-79,357.05	26,970.38	1,50,827.47	98,440.80	1
2023-12-08 00:00:00	3,232.00	1,710.15	3.20	28.12	No_Signal	0.00	0.00	1,21,881.53	1,21,881.53	10
2023-12-11 00:00:00	3,233.00	1,702.80	10.37	3.20	No_Signal	0.00	0.00	1,21,881.53	1,21,881.53	10
2023-12-12 00:00:00	3,224.75	1,706.75	-1.20	10.37	No_Signal	0.00	0.00	1,21,881.53	1,21,881.53	10
2023-12-13 00:00:00	3,243.65	1,684.55	36.35	-1.20	No_Signal	0.00	0.00	1,21,881.53	1,21,881.53	10
2023-12-14 00:00:00	3,241.35	1,730.95	-4.93	36.35	No_Signal	0.00	0.00	1,21,881.53	1,21,881.53	10
2023-12-15 00:00:00	3,313.90	1,733.15	65.77	-4.93	No_Signal	0.00	0.00	1,21,881.53	1,21,881.53	10
2023-12-18 00:00:00	3,332.05	1,724.60	91.10	65.77	No_Signal	0.00	0.00	1,21,881.53	1,21,881.53	10
2023-12-19 00:00:00	3,336.05	1,709.75	107.58	91.10	No_Signal	0.00	0.00	1,21,881.53	1,21,881.53	10
2023-12-20 00:00:00	3,297.15	1,680.90	92.92	107.58	No_Signal	0.00	0.00	1,21,881.53	1,21,881.53	10
2023-12-21 00:00:00	3,302.95	1,666.85	110.53	92.92	No_Signal	0.00	0.00	1,21,881.53	1,21,881.53	10
2023-12-22 00:00:00	3,341.30	1,672.05	144.51	110.53	No_Signal	0.00	0.00	1,21,881.53	1,21,881.53	10
2023-12-26 00:00:00	3,383.35	1,645.30	209.03	144.51	Short_the_Spread	-91,350.45	37,841.90	1,75,390.08	1,21,881.53	11
2023-12-27 00:00:00	3,404.45	1,669.45	209.84	209.03	Short_the_Spread	-91,920.15	38,397.35	1,75,390.08	1,21,867.28	11
2023-12-28 00:00:00	3,397.25	1,681.20	192.77	209.84	Short_the_Spread	-91,725.75	38,667.60	1,75,390.08	1,22,331.93	11
2023-12-29 00:00:00	3,402.40	1,685.80	194.05	192.77	Short_the_Spread	-91,864.80	38,773.40	1,75,390.08	1,22,298.68	11

Table 7 Backtesting signals for ASIANPAINT and BAJAJFINSV pair

4. Backtesting Performance

Backtesting is a crucial step in evaluating the performance of a trading strategy using historical data. It involves applying a strategy to historical market data to assess how it would have performed over a specific period. Various performance metrics are used to analyse the results. Let's explore some key metrics commonly used in backtesting:

- **Cumulative Return:**

Cumulative return measures the total return generated by a strategy over a specific period. It is calculated as the ratio of the final value to the initial investment, minus 1

- **Drawdown:**

Drawdown represents the peak-to-trough decline during a specific period. It measures the largest loss from a peak to a subsequent trough.

- **Annualized Return:**

Annualized return is the geometric mean annual return over a period. It provides a comparable annualized measure of performance.

- **Rolling Annual Volatility:**

Rolling annual volatility calculates the annual volatility over a rolling window of time. It helps visualize how volatility changes over time.

- **Sharpe Ratio:**

The Sharpe ratio measures the risk-adjusted performance of a strategy by considering the excess return per unit of risk (volatility). It is calculated as the ratio of the average excess return to the standard deviation of excess returns. The risk-free rate is taken as the yield of a 10-year Indian GOV bond.

- **Rolling Sharpe Ratio:**

Similar to rolling annual volatility, the rolling Sharpe ratio is calculated over a moving window, providing insights into how the risk-adjusted performance changes over time.

- **Value at Risk (VaR):**

VaR is a measure of the maximum potential loss at a certain confidence level over a specified time horizon.

- **Beta:**

Beta measures the sensitivity of a strategy's returns to market movements. A beta of 1 indicates that the strategy moves in line with the market.

- **Rolling Beta:**

Rolling beta calculates the beta over a rolling window, helping to visualize how the strategy's sensitivity to the market changes over time.

These metrics collectively provide a comprehensive view of a trading strategy's historical performance, risk-adjusted returns, and sensitivity to market movements.

4.1 Implementation of Backtesting Performance:

A function create_tear_sheet is defined in Backtesting.py Python file. Function details are given below.

create_tear_sheet(returns, save_loc_chart, y_ticker, x_ticker)

Parameters:

- returns : DataFrame
  containing backtesting signals and changes in total assets generated by the previous section
- save_loc_chart : string
  save location for saving the charts
- y_ticker : str
  name of y ticker
- x_ticker : str
  name of x ticker

Returns: It returns two objects and saves a chart.

- **Dictionary**
  Containing a summary of the backtesting performance of the trading strategy.

{ Start Date','End Date','Total Days','Annual Return','Cumulative Return','Annual Volatility','Sharp Ratio','Beta with the benchmark index','Rolling Months','Max Drawdown','Skew','Kurtosis','Numbers trades done ','Daily VaR at 99th Percentile'}

- **DataFrame**

  A DataFrame containing returns along with all the things required to calculate the above parameters and to generate charts.

  Also, it saves a chart containing trends of cumulative return, drawdowns, rolling annualized returns, rolling sharp ratio, rolling annual volatility, histogram of daily returns, daily returns, rolling beta with index over a test data time.

4.2 Results of Backtesting Performance.

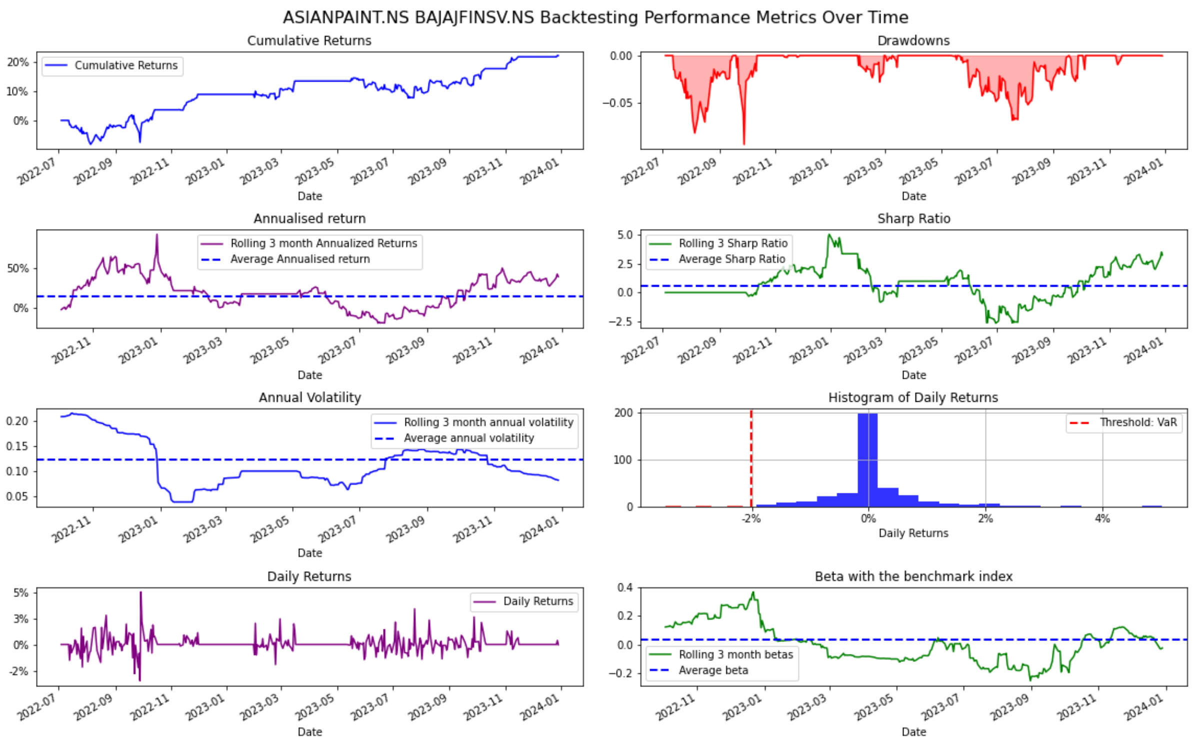
For illustrative purposes, only the performance of 1 pair is shown here. The results of all the other pairs (144) are saved by the script at a given location.

The backtesting result of the pairs Asian Paint and Bajaj Finance over a period of one and a half years shows an annual return of 17.23%. with annual volatility of returns is around 10.36%. Sharpe Ratio of 0.9106 (or 91.06%) generally suggests a moderate level of risk-adjusted performance. It indicates that the investment or portfolio is generating returns above the risk-free rate, but the excess return is not extremely high relative to the risk taken.

Stock Pair	**ASIANPAINT & BAJFINANCE**
Start Date	04-Jul-2022
End Date	29-Dec-2023
Total Days	369
Annual Return	17.23%
Cumulative Return	126.21%
Annual Volatility	10.36%

Sharp Ratio	91.06%
Beta with the benchmark index	10.84%
Rolling Months	3
Max Drawdown	-7.86%
Skew	0.7664
Kurtosis	7.5992
Numbers trades done	13
Daily VaR at 99th Percentile	-1.93%

Table 8 Backtesting performance of ASIAN PAINT and BAJAJ FINANCE



5 Final Selection of a Stock Pair:

The backtesting performance of all 145 stock pairs was analysed. The following filters were applied to finalize a trading pair.

- Pairs having <10% annual return removed. After applying this filter only 21 pairs remained.
- All pairs having <3% of VaR are removed. Only 1 pair got removed and 20 pairs were still in contention.
- After qualitatively analysing the pairs, it was observed that only 3 pairs belong to the same industry sector. The remaining 17 pairs belong to different industry sectors and their relationship might be temporary hence results may be spurious.
- The final three pairs were
    1) TCS and TECHM companies belonging to the software industry

2) BAJFINANCE and BAJAJFINSV companies belonging to the finance industry
3) NTPC and COALINDIA companies belonging to the power sector

Bajaj Finance Limited (BAJFINANCE) and Bajaj Finserv Limited (BAJAJFINSV) are related companies within the Bajaj Group, but they serve different roles and functions in the financial sector. Both companies are part of the larger conglomerate known as Bajaj Holdings & Investments Limited.

Finally, BAJFINANCE and BAJAJFINSV are selected for pairs trading further optimization of the strategy due to the following reasons.

Common Ownership:

Both Bajaj Finance Limited and Bajaj Finserv Limited have common ownership through the Bajaj Group. The Bajaj Group is a well-known conglomerate in India with diverse business interests, including financial services, automobiles, and more.

Bajaj Finserv Limited:

Bajaj Finserv is a holding company that operates as a financial services company. It owns and operates various subsidiaries and divisions, including Bajaj Finance. Bajaj Finserv is involved in insurance, lending, wealth advisory, and other financial services.

Bajaj Finance Limited:

Bajaj Finance is a non-banking financial company (NBFC) and is one of the subsidiaries under Bajaj Finserv. Bajaj Finance is primarily focused on providing a wide range of financial products and services, including consumer finance, personal loans, home loans, business loans, and more.

Business Relationship:

Bajaj Finance operates as a lending and investment arm, providing loans and financial solutions to consumers and businesses. Bajaj Finserv, as the holding company, oversees and manages the diverse financial services provided by its subsidiaries, including Bajaj Finance.

Synergies:

The relationship between Bajaj Finance and Bajaj Finserv creates synergies within the Bajaj Group. Bajaj Finserv acts as an umbrella organization overseeing the financial services sector, and Bajaj Finance plays a significant role in the lending and financing aspects of the group.

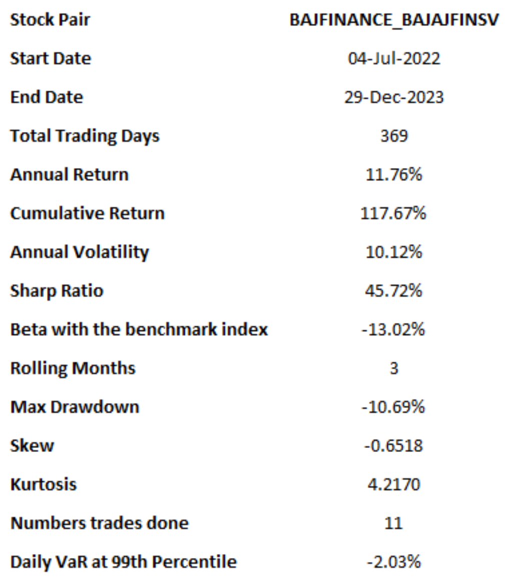
Stock Market Listings:

Both Bajaj Finance Limited (BAJFINANCE) and Bajaj Finserv Limited (BAJAJFINSV) are separately listed on stock exchanges, and their shares are traded independently. Investors can buy and sell shares of each company based on their specific investment preferences.

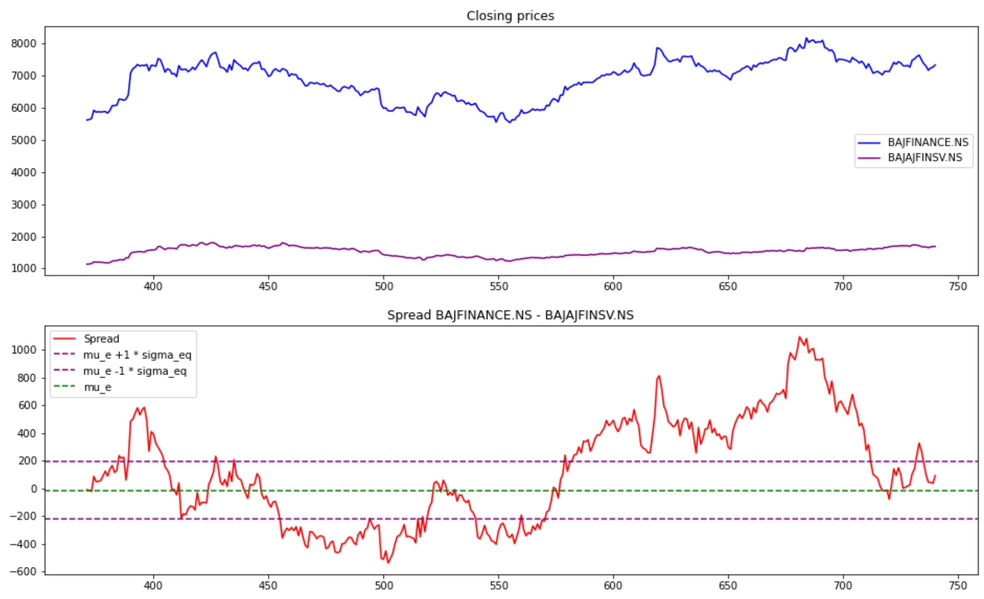
While they are related entities within the Bajaj Group, Bajaj Finance and Bajaj Finserv operate as distinct companies with their own business operations, management structures, and financial objectives. The relationship allows for diversified financial offerings and a comprehensive approach to serving the needs of consumers and businesses in the financial services sector.

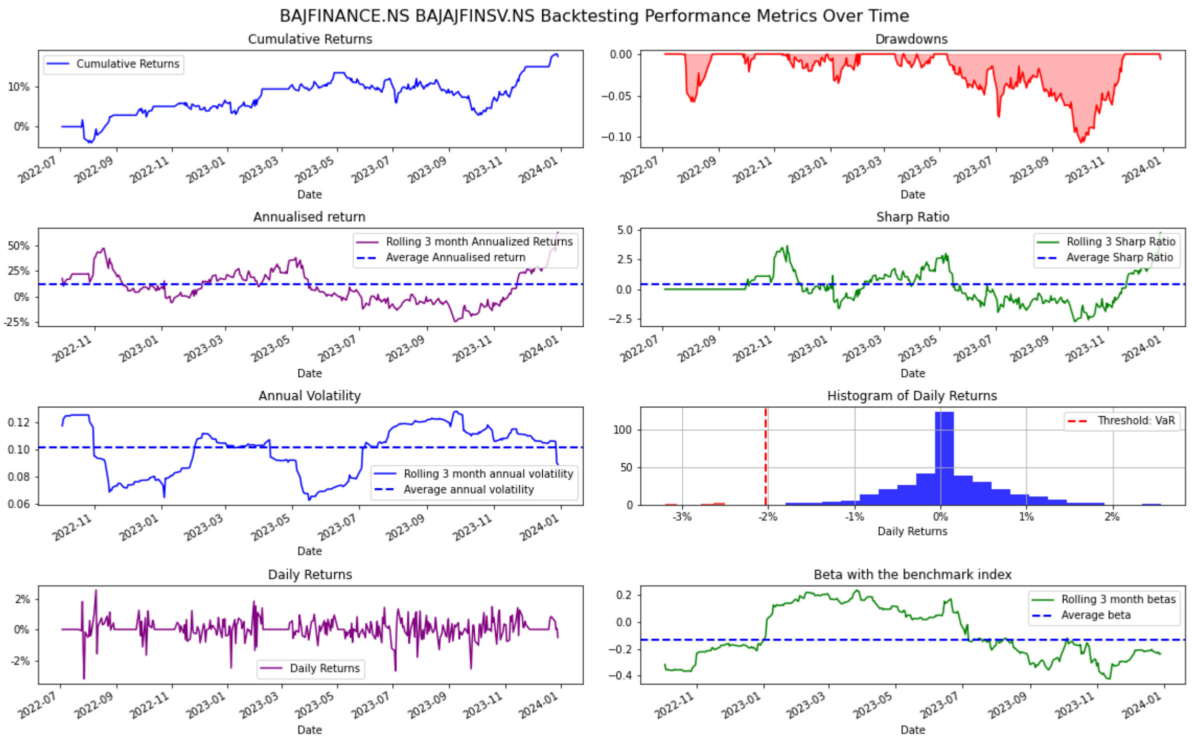
4.1 Backtesting performance of trading pair BAJFINANCE and BAJAJFINSV:

The table and charts give the backtesting performance of the BAJFINANCE and BAJAJFINSV. The trading performance for the period from July 4, 2022, to December 29, 2023, spans 369 trading days. The strategy demonstrated positive results with an annual return of 11.76%, leading to a cumulative return of 117.67%. The risk-adjusted performance was notably strong, as reflected in the high Sharp Ratio of 45.72%. Despite the unusual negative beta of -13.02%, indicating an inverse relationship with the benchmark index, the strategy managed to mitigate risk effectively, as evidenced by the relatively low annual volatility of 10.12%. The maximum drawdown, representing the largest observed loss, was -10.69%. The skewness of -0.6518 indicates a distribution with a longer left tail, suggesting more frequent small positive returns and fewer large negative returns. The kurtosis of 4.2170 implies fatter tails in the return distribution. Over the course of 369 trading days, the strategy executed a total of 11 trades. Furthermore, the daily Value at Risk (VaR) at the 99th percentile was calculated at -2.03%, providing an estimate of the maximum expected daily loss at a 99% confidence level. In summary, the trading performance showcases positive returns, effective risk management, and notable risk-adjusted performance, though the unusual negative beta warrants further scrutiny for a comprehensive understanding. Note that this performance is when Z = 1 in equation $\mu \pm Z\,\sigma_{eq}$.



Stock Pair	**BAJFINANCE_BAJAJFINSV**
Start Date	04-Jul-2022
End Date	29-Dec-2023
Total Trading Days	369
Annual Return	11.76%
Cumulative Return	117.67%
Annual Volatility	10.12%
Sharp Ratio	45.72%
Beta with the benchmark index	-13.02%
Rolling Months	3
Max Drawdown	-10.69%
Skew	-0.6518
Kurtosis	4.2170
Numbers trades done	11
Daily VaR at 99th Percentile	-2.03%





Rolling beta can be a useful metric for certain investment strategies, but its application in the context of statistical arbitrage and market-making may be limited due to the dynamic, high-frequency, and short-term nature of these trading approaches. Traders in these domains typically rely on more immediate and adaptive measures to capture the nuances of rapidly changing market conditions.

Similarly, evaluating the Sharpe Ratio for a risk-neutral trading strategy is challenging due to the departure from real-world risk considerations and the assumptions underlying the Sharpe Ratio calculation. Real-world investors typically exhibit risk aversion, making the application of risk-neutral concepts less relevant for assessing the risk-adjusted performance of trading strategies.

4.2 Optimization of the Trading Bounds:

The above backtesting performance is when Z = 1 in the equation for trading bounds $\mu \pm Z\ \sigma_{eq}$.

Optimizing trading bounds, represented as $\mu \pm Z\ \sigma_{eq}$, involves finding the optimal values for parameters such as Z that maximize certain performance metrics, considering factors like profitability (P&L) and the number of trades. This approach is common in trading and quantitative finance to define entry and exit points for trades.

Here's an overview of the key components and considerations in optimizing trading bounds:

Definition of Trading Bounds:

The trading bounds are defined by the mean (μ) and standard deviation (σ_{eq}) of a relevant metric, often the spread between two assets in pairs trading or any other relevant financial indicator. Z represents the number of standard deviations from the mean that defines the upper and lower bounds.

Optimization Objective:

The optimization process typically involves defining an objective function to be maximized or minimized. Common objectives include maximizing profitability (P&L), minimizing risk, or optimizing a combination of both.

Parameters to Optimize:

The key parameter to optimize in this context is often Z. By varying Z, traders can explore different levels of risk tolerance and profit potential. The goal is to find the optimal value for Z that aligns with the trader's risk-return preferences.

Performance Metrics:

Traders may choose performance metrics to evaluate the effectiveness of different Z values. Common metrics include total P&L, risk-adjusted return measures (e.g., Sharpe Ratio), the number of trades executed, and other relevant indicators.

Backtesting and Simulation:

To assess the impact of different Z values, traders often conduct backtesting or simulation. This involves applying historical data to the trading strategy under various Z scenarios to evaluate how each parameter setting would have performed in the past.

Trade-Offs between P&L and Number of Trades:

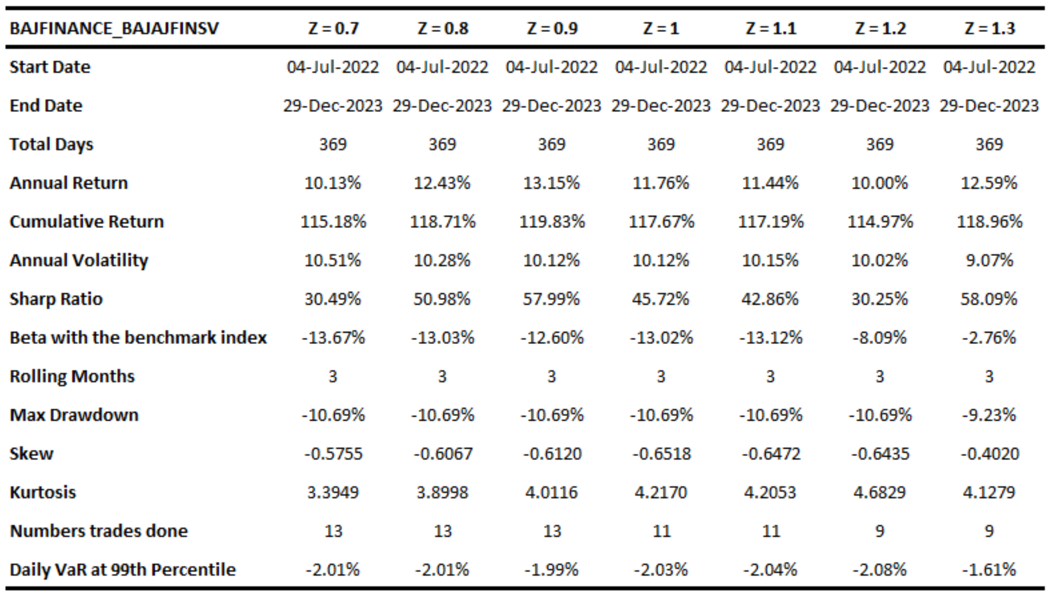
We need to consider the trade-offs between maximizing P&L and the number of trades. Aggressive trading bounds may lead to more frequent trades but might also increase transaction costs and potentially reduce profitability per trade. Balancing these factors is crucial in finding an optimal solution.

Risk Management:

Optimizing trading bounds should also consider risk management. A too-aggressive approach might expose the trader to excessive risk. VaR, help in evaluating the efficiency of the strategy while accounting for risk.

In summary, optimizing trading bounds involves finding the optimal values, especially Z, that maximize profitability and other relevant performance metrics while considering the trade-offs between P&L and the number of trades. This process often requires thorough backtesting, simulation, and a careful consideration of risk and transaction costs.

Backtesting performance is assessed by varying trading bounds by changing Z from 0.7 to 1.3 in steps of 0.1. Below table shows the backtesting performance



BAJFINANCE_BAJAJFINSV	Z = 0.7	Z = 0.8	Z = 0.9	Z = 1	Z = 1.1	Z = 1.2	Z = 1.3
Start Date	04-Jul-2022	04-Jul-2022	04-Jul-2022	04-Jul-2022	04-Jul-2022	04-Jul-2022	04-Jul-2022
End Date	29-Dec-2023	29-Dec-2023	29-Dec-2023	29-Dec-2023	29-Dec-2023	29-Dec-2023	29-Dec-2023
Total Days	369	369	369	369	369	369	369
Annual Return	10.13%	12.43%	13.15%	11.76%	11.44%	10.00%	12.59%
Cumulative Return	115.18%	118.71%	119.83%	117.67%	117.19%	114.97%	118.96%
Annual Volatility	10.51%	10.28%	10.12%	10.12%	10.15%	10.02%	9.07%
Sharp Ratio	30.49%	50.98%	57.99%	45.72%	42.86%	30.25%	58.09%
Beta with the benchmark index	-13.67%	-13.03%	-12.60%	-13.02%	-13.12%	-8.09%	-2.76%
Rolling Months	3	3	3	3	3	3	3
Max Drawdown	-10.69%	-10.69%	-10.69%	-10.69%	-10.69%	-10.69%	-9.23%
Skew	-0.5755	-0.6067	-0.6120	-0.6518	-0.6472	-0.6435	-0.4020
Kurtosis	3.3949	3.8998	4.0116	4.2170	4.2053	4.6829	4.1279
Numbers trades done	13	13	13	11	11	9	9
Daily VaR at 99th Percentile	-2.01%	-2.01%	-1.99%	-2.03%	-2.04%	-2.08%	-1.61%

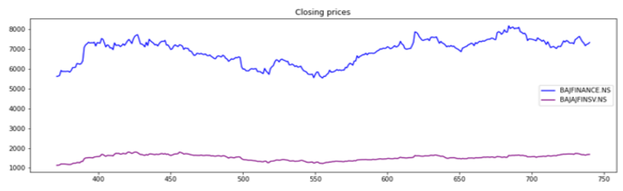
The pairs trading strategy for the BAJFINANCE_BAJAJFINSV pair was assessed across different values of Z (standard deviations for trading bounds). Among the various Z values, Z = 1.3 stands out as a favourable choice, striking a balance between risk and return. At Z = 1.3, the strategy exhibits a Sharpe Ratio of 58.09%, indicating robust risk-adjusted performance. The cumulative return (118.96%) and annual return (12.59%) at Z = 1.3 are among the highest, showcasing attractive profitability.

The strategy maintains consistent annual volatility ranging from 9.07% to 10.51%, reflecting performance stability. The beta with the benchmark index remains negative, indicating an inverse relationship with market movements. Despite variations in Z, the maximum drawdown consistently hovers around -10.69%, demonstrating resilience against significant losses.

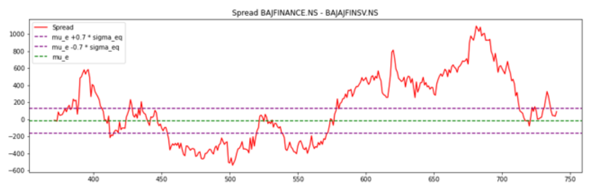
Skewness and kurtosis, providing insights into the return distribution, exhibit consistent patterns across different Z values. The number of trades decreases with higher Z values, reflecting a more conservative approach and potentially reducing the frequency of trades.

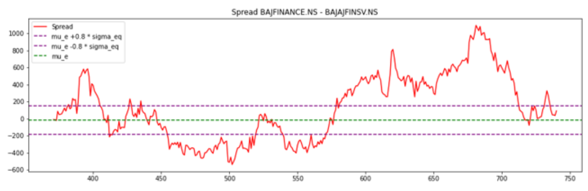
Considering the balance between risk management and return potential, Z = 1.3 emerges as a preferred choice. This comprehensive evaluation ensures a well-informed and prudent choice of Z for the pairs trading strategy.

Charts showing trend of closing prices and variation of trading bounds which result in trading signal generation.

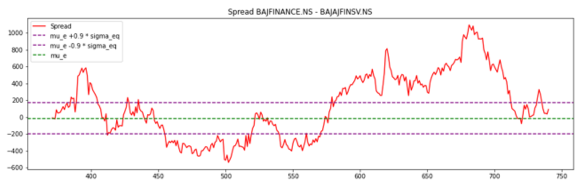


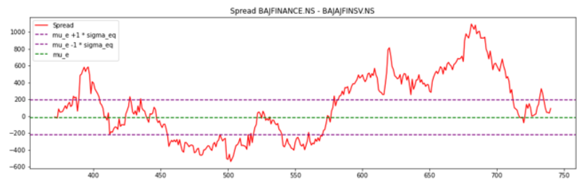
Z= 0.7 **Z= 0.8**



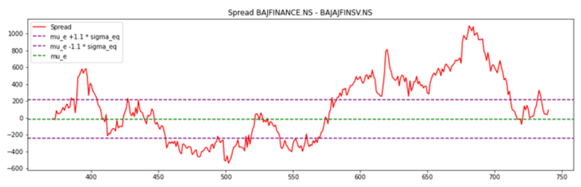


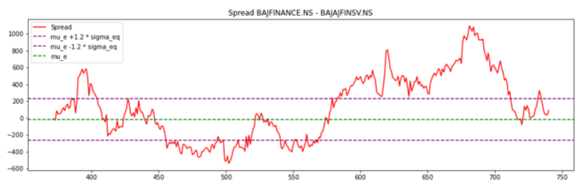
Z= 0.9 **Z= 1**



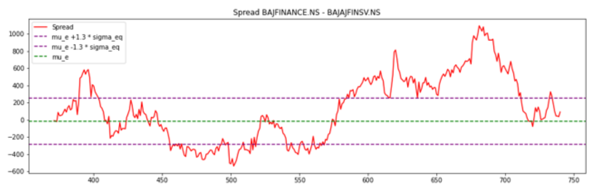


Z= 1.1 **Z= 1.2**



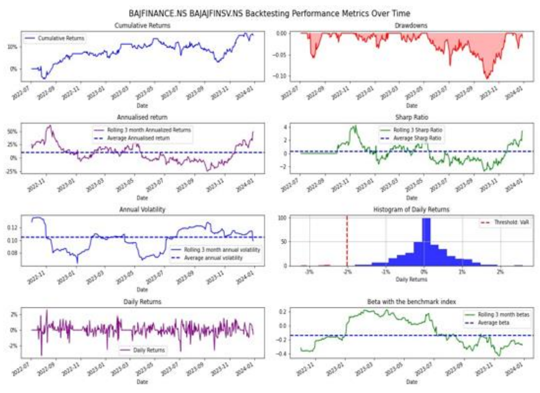


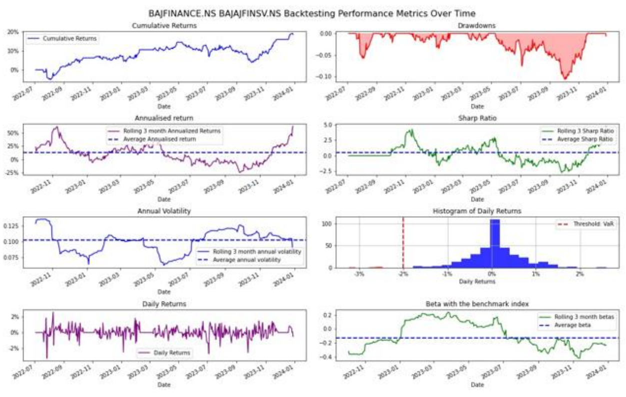
Z= 1.3



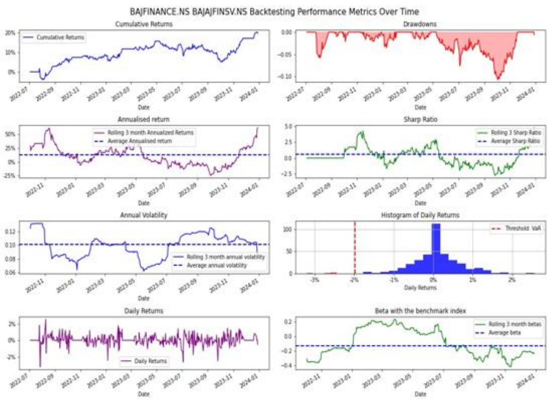
Charts showing backtesting performance of returns when you vary the trading bounds.

Z = 0.7 Z = 0.8

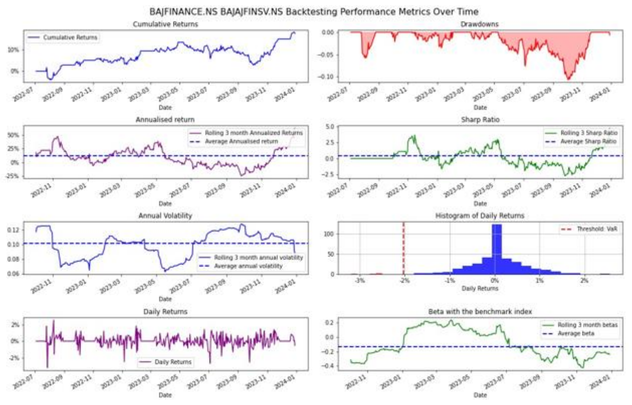




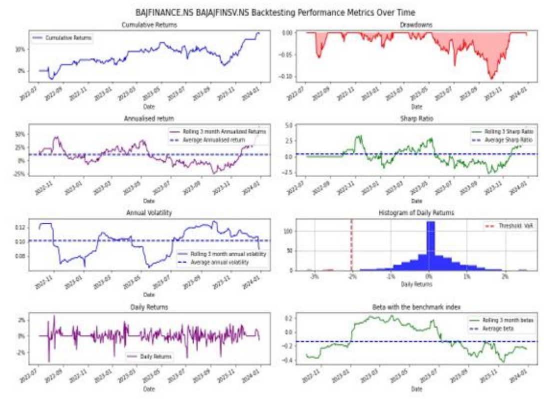
Z = 0.9



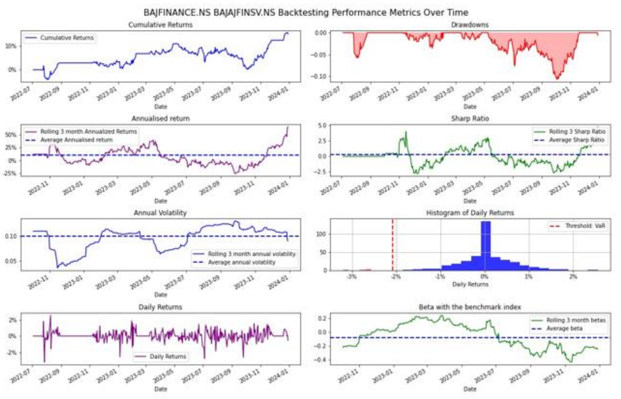
Z = 1



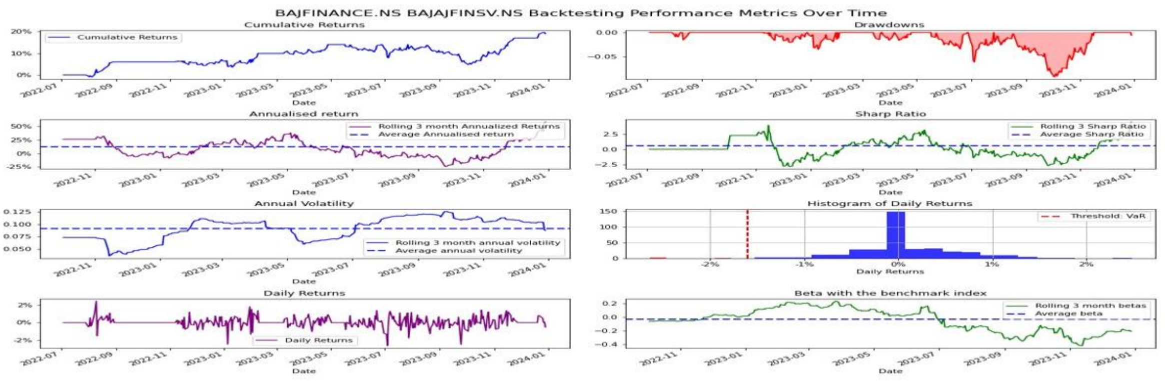
Z = 1.1



Z = 1.2



Z = 1.3



6. Conclusion

The project tries to explain various statistical techniques used in pairs trading such as stationarity of time series, cointegration, Engle-Granger procedure, O-U process, backtesting, etc. All these techniques were coded independently and readymade Python functions are not used. The project started with 50 constituent stocks from the Indian benchmark equity index Nifty50. Sufficient historical data was available for 48 stocks and these are used in further analysis.

Stationarity of the time series of closing prices of stocks was assessed by the ADF test using lag = 1. Out of 48 stocks, 4 are stationary and 44 are non-stationary. 976 pairs of stocks were made from 44 stocks. A cointegration test was performed for all the stock pairs. 145 pairs were found to be cointegrated. Using the Error Correction Model, it is checked that spread is mean reverting or not for these 145 series. All 145 series were found to be mean reverting. After that O-U process is fitted to the spreads and parameters required for trading bounds which are used to generate backtesting signals were calibrated. All these tests and calibrations were done with training data.

Now using the above parameters and test data backtesting performance of all the 145 pairs was assessed. After quantitatively analysing risk-return reward from backtesting performance as well as qualitatively analysing relationship stock pairs one pair is chosen (Bajaj Finance and Bajaj FinServ).

On the selected stock pair trading bonds were varied by changing the value of Z from 0.7 to 1.3 in steps of 0.1. Finally, based on the backtesting performance across the different trading bounds value of Z is chosen as 1.3.

7. Bibliography

Bisgaard, S. (2011). TIME SERIES ANALYSIS AND FORECASTING BY EXAMPLES. *Wiley*.

Diamond, R. (2013). Learning and Trusting Cointegration in Statistical Arbitrage. *WILMOTT*.

Diamond, R. (2023, June). CQF_June_2023_M6L8. *CQF FitchLearning*.

Diamond, R. (2023). FINAL PROJECT TUTORIAL Pairs Trading. *CQF Fitch Learning*.

MacKinnon, J. G. (2010). Critical Values for Cointegration Tests. *Queen's Economics Department Working Paper No. 1227*.

Vidyamurthy, G. (n.d.). *Pairs Trading, Quantitative Methods and Analysis.* Wiley Finance.