## Algorithm User Guide: Statistical Arbitrage Trading Strategy

### Introduction

Welcome to the user guide for our statistical arbitrage trading strategy algorithm! This guide aims to provide comprehensive instructions on using the algorithm effectively, including adjusting parameters for different risk profiles and understanding the flexibility in data frequency.

### Requirements

Before getting started, ensure you have Python installed on your system along with the required Python libraries: yfinance, matplotlib, pandas, and numpy. You can install these libraries using pip:

pip install yfinance matplotlib pandas numpy

### Understanding the Algorithm

#### Overview

Our algorithm implements a statistical arbitrage strategy known as pairs trading. This strategy involves identifying two correlated assets and trading them simultaneously based on deviations from their historical price relationship. The algorithm aims to exploit these pricing inefficiencies in the market for potential profit.

#### Components

1. **Data Retrieval**: The algorithm fetches historical price data for the chosen assets from the Yahoo Finance API. Optionally, it can also retrieve data for benchmark indices to provide context for the trading signals.
2. **Signal Generation**: Using the historical price data, the algorithm calculates a ratio between the prices of the two assets. It then computes moving averages and standard deviations of this ratio to generate buy and sell signals.
3. **Trading Strategy**: Simulated trades are executed based on the generated signals. The algorithm calculates the resulting profit or loss and computes risk-adjusted returns metrics such as the Sharpe ratio and standard deviation.

### Using the Algorithm

#### 1. Setting Parameters

The algorithm offers flexibility through various parameters that can be adjusted to customize your trading strategy:

* **Start and End Dates**: Define the start and end dates for the historical data retrieval.
* **Moving Average Windows (window1 and window2)**: Set the window sizes for computing the moving averages of the ratio between the two assets.
* **Data Frequency**: Choose the frequency of data for computing moving averages. Options include daily (‘1d’), weekly (‘1wk’), and 5-day (‘5d’) intervals.
* **Z-score Threshold**: Adjust the threshold for generating buy and sell signals based on z-score deviations.

#### 2. Frequency Flexibility

The algorithm provides flexibility in data frequency, allowing you to tailor your trading strategy to different time horizons:

* **Daily Data (‘1d’)**: Suitable for short-term trading strategies with a focus on intraday price movements.
* **Weekly Data (‘1wk’)**: Ideal for medium-term trading strategies that capture broader market trends.
* **5-Day Data (‘5d’)**: Offers a balance between short-term and medium-term strategies, providing a smoother signal compared to daily data.

Choose the data frequency that aligns with your trading objectives and risk preferences.

#### 3. Risk Profiles

To accommodate different risk preferences, the algorithm supports multiple risk profiles with predefined parameter sets:

* **High-Risk Profile**: Aggressive trading strategy with shorter moving average windows and lower z-score thresholds, suitable for traders seeking high returns with higher volatility.
* **Mid-Risk Profile**: Balanced trading strategy with moderate moving average windows and z-score thresholds, offering a mix of risk and return potential.
* **Low-Risk Profile**: Conservative trading strategy with longer moving average windows and higher z-score thresholds, emphasizing capital preservation and stability.

Select the risk profile that matches your risk tolerance and investment goals.

### Example Usage

Here’s an example of how to use the algorithm with different risk profiles:

from trading\_strategy import trade\_strategy\_freq  
  
# Define risk profiles with parameters  
risk\_profiles = {  
 'high\_risk': {'frequency': '1d', 'window1': 6, 'window2': 180, 'zscore\_threshold': 0.6},  
 'mid\_risk': {'frequency': '5d', 'window1': 8, 'window2': 190, 'zscore\_threshold': 0.8},  
 'low\_risk': {'frequency': '5d', 'window1': 10, 'window2': 200, 'zscore\_threshold': 1.0}  
}  
  
# Run the algorithm for each risk profile  
for profile\_name, params in risk\_profiles.items():  
 print(f"Running strategy for {profile\_name} profile:")  
 profit\_percentage, sharpe\_ratio, std\_dev = trade\_strategy\_freq(start='2018-01-01', end='2023-01-01',   
 window1=params['window1'],   
 window2=params['window2'],   
 frequency=params['frequency'],   
 zscore\_threshold=params['zscore\_threshold'])

### Conclusion

With its flexibility in parameter settings, data frequency options, and support for multiple risk profiles, our algorithm offers a versatile tool for implementing statistical arbitrage trading strategies. By adjusting parameters and selecting the appropriate risk profile, traders can tailor the algorithm to their preferences and objectives, potentially enhancing their trading performance.