SOC 2025 - Algorithmic Trading in Python
Week 1

Revanth Manepalli

May 2025

Mean Reversion

- Mean reversion is a financial theory that suggests asset prices and historical returns eventually return to their long-term average (mean).
- This concept assumes that extreme deviations from the mean are temporary and that prices will revert over time.
- This strategy works well for stocks that are bounded within a certain range and are not highly volatile.
- **Code Example:**

```
mean_price = prices[-20:].mean()
threshold = 1.05
if current_price > mean_price * threshold:
    action = "sell_short"
elif current_price < mean_price / threshold:
    action = "buy"
else:
    action = "hold"
```

- The threshold and the number '20' are parameters that we optimize during backtesting.
- Threshold is self-explanatory, while the number '20' can mean a lot of different things like 20 seconds, 20 minutes, 20 hours or 20 days etc depending on the asset, market, and our algorithm.
- Threshold is not a constant value. Mean reversion can be combined with Bollinger Bands, RSI, and Z-scores to have thresholds that change with market conditions.
- It's not good to have a fixed threshold like in the above pseudo-code
  - **Reason:** Markets are dynamic, and a fixed threshold may not adapt to changing market conditions.

Market Making

- Market making is a trading strategy where a trader continuously buys and sells a financial asset, quoting both bid and ask prices to provide liquidity to the market.
- **Bid Price:** The price at which a trader is willing to buy an asset.
- **Ask Price:** The price at which a trader is willing to sell an asset.
- **Liquidity:** Liquidity refers to how easily an asset can be bought or sold in the market without affecting its price.
- **Code Example:**

```
mid_price = (best_bid + best_ask) / 2
spread = best_ask - best_bid
if spread > min_desired_spread:
    buy_price = mid_price - spread/2
    sell_price = mid_price + spread/2
    place_limit_order("buy", price=buy_price, amount=lot_size)
    place_limit_order("sell", price=sell_price, amount=lot_size)
```

- **Best Bid and Best Ask:** The best bid is the highest price a buyer is willing to pay, while the best ask is the lowest price a seller is willing to accept.
- **Mid Price** is the average of the best bid and best ask prices and we take it to be the **fair price** of the asset.
- Our buy order is placed at a price slightly below the mid price, and our sell order is placed at a price slightly above the mid price.

Stop Loss & Take Profit

- These aren't trading strategies, they are the rules to manage exits.
- **Stop Loss:** A stop loss is an order placed to sell a security when it reaches a certain price, limiting the potential loss on a trade.
- **Take Profit:** A take profit order is placed to sell a security when it reaches a certain price, locking in profits on a trade.
- **Code Example:**

```
entry_price = purchase_price
stop_loss = entry_price * 0.9   # 10% drop
take_profit = entry_price * 1.2 # 20% rise

if current_price <= stop_loss:
    action = "sell_and_stop_loss"
elif current_price >= take_profit:
    action = "sell_and_take_profit"
else:
    action = "hold"
```

- **why do we need a take-profit? Why can't we remove it and collect as much profit as possible?**
    1. **Markets are unpredictable:** Prices can fluctuate rapidly, and without a take-profit, you might miss out on gains if the price reverses.
    2. **Greed kills profits:** Traders often hold onto positions too long, hoping for more profit, which can lead to losses if the market turns against them.
    3. **Discipline over Emotion:** A take-profit order helps maintain discipline in trading by automating the exit at a predetermined profit level, reducing emotional decision-making.
    4. **Optimizing Win rate and risk: Reward:** A take-profit order can help optimize the win rate and risk-reward ratio of a trading strategy by ensuring that profits are realized before the market can reverse.
    5. **Captial Efficiency:** By locking in profits, traders can free up capital to reinvest in other opportunities, enhancing overall portfolio performance.
- Stop loss thresholds are determined by backtesting our algorithm
- It can't be too tight, otherwise we will be stopped out too often, and it can't be too loose, otherwise we will lose a lot of money.
- Stop-loss thresholds are determined by backtesting your algorithm. It cannot be too tight because then it'll detect noise and exit your position, and it can't be too high, or you'll suffer bigger losses before exiting a bad position. We want to stay in a profitable position for as long as possible and we want to exit a bad position as quickly as possible without getting distracted by noise.

Bollinger Bands

- They are volatile-based technical indicator consisting of three lines plotted around a price chart.
- The middle band is s 20-day simple moving average (SMA).
- The upper and lower bands are bound by adding and subtracting two standard deviations from the middle band.
- According to guassian distribution, 95% of the data points lie within two standard deviations from the mean - one of the reasons why we use additional indicators along with Bollinger Bands.
- As Volatility increases, standard deviation increases, and the Bollinger Bands expand, and vice versa if Volatility is low.
- These bands serve as thresholds for mean reversion.
- **Misconception:**
    - Some traders believe:
        * If the price touches the upper band, it must reverse and go down (so I should short), and if it touches the lower band, it must bounce back up (so I should go long).

  - But this is not always true.
  - When the price touches the upper band, it simply means **Price is high relative to recent history**

- During strong uptrends, the price can stay above the upper band for a long time, and during strong downtrends, the price can stay below the lower band for a long time.

- **Code Example:**

```
def calculate_bollinger_bands(prices, period=20, std_multiplier=2.0):
    # Assume you defined a SMA function somewhere
    middle_band = SMA(prices, period)

    # Calculate standard deviation
    std_deviation = np.std(prices[-period:], ddof=1)

    # Calculate upper and lower bands
    upper_band = middle_band + (std_multiplier * std_deviation)
    lower_band = middle_band - (std_multiplier * std_deviation)

    return upper_band, middle_band, lower_band
```

- **We want you to look into why 'np.std()' has the argument 'ddof' and why we used it here.**
  - 'ddof' stands for "Delta Degrees of Freedom" and is used to adjust the divisor during the calculation of the standard deviation.
  - In the context of sample standard deviation, 'ddof=1' is used to apply Bessel's correction, which corrects the bias in the estimation of the population standard deviation from a sample.
  - When calculating the standard deviation of a sample, we divide by $n - ddof$ (where n is the number of data points).

- **Also learn how to implement SMA in code yourself.**

```
def SMA(prices, period):
    if len(prices) < period:
        return None  # Not enough data to calculate SMA
    return sum(prices[-period:]) / period
```

Relative Strength Index (RSI)

- RSI is a momentum oscillator that measures the speed and change of price movements.
- It measures the magnitude of recent price changes(typically 14 days) to evaluate overbought or oversold conditions in a market.
- It ranges from 0 to 100 and is typically used to identify overbought or oversold conditions in a market.

```
def calculate_rsi(prices, period=14):
    gains = []
    losses = []

    # Calculate **daily** price changes
    for i in range(1, len(prices)):
        change = prices[i] - prices[i-1]
        if change > 0:
            gains.append(change)
            losses.append(0) # periods with price increase, we assume
                loss to be 0
        else:
            gains.append(0) # periods with price decrease, we assume
                profit to be 0
            losses.append(abs(change))

    # Calculate average gain and loss
    avg_gain = SMA(gains, period)
    avg_loss = SMA(losses, period)

    # Calculate RS and RSI
    if avg_loss == 0:
        return 100

    rs = avg_gain / avg_loss
    rsi = 100 - (100 / (1 + rs))

    return rsi
```

- RSI + Bollinger Bands form a goog trading strategy for mean reversion or even trend following and volatile stocks.
- RSI gives us the momentum of the price, while Bollinger Bands give us the volatility of the price.
- Although, during strong trends, RSI can stay in the overbought or oversold region for a long time, so we can't just rely on RSI alone.
- **From the code you should be able to understand what the RSI is but it's still not clear how we can use RSI. Try to think of what it means when RSI is 100 and when RSI is 0, what does that say for the avg_gain and avg_loss ?**
  - When RSI is 100, it indicates that the average gain over the specified period is significantly higher than the average loss, suggesting a strong upward momentum in the asset's price.
  - Conversely, when RSI is 0, it indicates that the average loss is significantly higher than the average gain, suggesting a strong downward momentum in the asset's price.
  - In both cases, the RSI values indicate extreme conditions in the market, either overbought (RSI near 100) or oversold (RSI near 0).
  - This can help traders identify potential reversal points or confirm the strength of a trend.

- typically RSI values above 70 indicate that an asset is overbought, while values below 30 indicate that it is oversold.

Momentum

- Momentum is an idea that assets that have performed well in the past will continue to perform well in the future, and vice versa for poorly performing assets.
- To measure momentum, we can compare the current price to the price a certain number of time unit (e.g., days, hours, minutes) ago.
- If the current price is significantly higher than the price a certain number of time unit ago, you may go long, and if the current price is significantly lower than the price a certain number of time unit ago, you may go short.
- Momentum traders often use technical indicators like the Moving Average Convergence Divergence (MACD) or the Rate of Change (ROC) to identify momentum.
- Momentum traders don't try to predict the future price of an asset, they simply try to identify the current trend and ride it until it reverses.
- **Code Example:**

```
lookback = 5
momentum = prices[-1] - prices[-1-lookback]
if momentum > 0:
    action = "buy"   # price is higher than 'lookback' time units ago
else:
    action = "sell"  # price is lower, downtrend may continue
```

- ***There is difference between selling and shorting. Find out what it is.**
  - **Selling:**
    - Selling means closing a position that you already own.
    - you're exiting a trade or investment by selling shares or assets that you previously bought.
  - **Shorting:**
    - Shorting means you're betting that the price will fall. You borrow shares from a broker, sell them immediately, and later buy them back at a lower price (hopefully), returning them to the broker.

Selling	**Shorting**
You own the asset.	You don't own the asset (you borrow it).
Profit if price goes up, then you sell.	Profit if price goes down after you short.
This is called a "long" position.	Higher risk — potential for unlimited losses if price rises.

Table 1: Difference Between Selling and Shorting

Breakout

- A Breakout occurs when the price of an asset moves outside a defined support or resistance level with increased volume.
- It's a signal that the price may start tredning strongly in the direction of the breakout.
  - **Resistence:** A ceiling price where the asset struggles to move higher.
  - **Support:** A floor price where the asset struggles to move lower.
- A breakout means the price breaks above resistence or below support, suggesting a new trend is forming.
- **Code Example:**

```
lookback = 20
recent_high = max(prices[-lookback:])
recent_low = min(prices[-lookback:])

if current_price > recent_high:
    action = "buy_breakout"
elif current_price < recent_low:
    action = "sell_breakout"
else:
    action = "hold"
```

Z-Score

- The Z-score is a statistical measure that tells you how far a data point is from the mean, in terms of standard deviations.
- It helps traders understand wether a price unusually high or low compared to its historical average.
- **Code Example:**

```
def calculate_zscore(prices, window=20):
    moving_avg = SMA(prices, window)
    moving_std = np.std(prices[-window:], ddof=1)
    zscore = (prices - moving_avg) / moving_std
    return zscore
```

- A Z-score of 0 or close to 0 (like < 0.3) means there are no significant price changes happening.

- **Rolling Z-Score:**

```
import pandas as pd
import numpy as np

def calculate_rolling_zscore(prices, window=20):
    prices_series = pd.Series(prices)
    moving_avg = prices_series.rolling(window=window).mean()
    moving_std = prices_series.rolling(window=window).std(ddof=1)
    zscore = (prices_series - moving_avg) / moving_std
    return zscore
```