**What is a baseline?**

The **baseline** is yesterday’s “normal” ratio between **BTC price** and a given stock’s price (BTC ÷ Stock).  
Today we monitor that ratio minute-by-minute:

* If today’s ratio rises **above baseline × (1 + buy%)**, it signals **BUY**.
* If it falls **below baseline × (1 – sell%)**, it signals **SELL**.

How we calculate the **baseline** matters a lot—different methods weight or filter yesterday’s data differently.

**The five baseline methods**

**1. Equal Mean**

* **What:** Simple arithmetic mean of all minute-by-minute BTC/Stock ratios from the previous day.
* **How:**
* **Pros:** Simple, quick.
* **Cons:** Sensitive to outliers.

**2. Median**

* **What:** The middle value of yesterday’s BTC/Stock ratios.
* **How:** Sort all minute ratios, pick the middle.
* **Pros:** Robust to single-minute spikes or anomalies.
* **Cons:** Ignores magnitude/volume of trades.

**3. VWAP\_RATIO (Volume Weighted Average Price ratio)**

* **What:** Uses **daily VWAP** for BTC and for the stock, then divides them.
* **How:**
* **Pros:** Accounts for where most trading happened.
* **Cons:** Relies on Polygon’s “volume” for BTC, which is sometimes print counts, not actual dollar turnover.

**4. VOL\_WEIGHTED**

* **What:** Weighted average of **minute ratios** using **stock’s share volume** as weights.
* **How:**
* **Pros:** Reflects the price levels where most stock trading occurred.
* **Cons:** Still sensitive if BTC prints are noisy.

**5. WINSORIZED**

* **What:** Trims extreme outliers before averaging.
* **How:** Clip the distribution of ratios to between the 5th and 95th percentile, then take the mean.
* **Pros:** Reduces effect of sudden spikes/drops.
* **Cons:** Throws away tail info (sometimes signal is in the tail).

**How to add to trader\_ui\_v6**

Right now, trader\_ui\_v6 lets you configure **buy%**, **sell%**, and **budget** per symbol. To add these methods:

1. **Database schema:**
   * symbol\_configs already has a method column (we added this earlier).
   * Each symbol row can store "VWAP\_RATIO", "WINSORIZED", etc.
2. **Worker (ratio\_worker\_sql\_v6.py):**
   * Replace the single baseline calc with a dispatcher function (like we built in V7).
   * Example:
   * def compute\_baseline(df, method):
   * if method == "EQUAL\_MEAN":
   * return df["ratio"].mean()
   * elif method == "MEDIAN":
   * return df["ratio"].median()
   * elif method == "VWAP\_RATIO":
   * # compute VWAP for BTC and stock
   * ...
   * elif method == "VOL\_WEIGHTED":
   * ...
   * elif method == "WINSORIZED":
   * ...
   * Before checking today’s ratios, load yesterday’s data for that symbol and compute baseline with the selected method.
3. **UI (trader\_ui\_v6.py):**
   * Add a method column to the editable per-symbol controls.
   * Users pick method per symbol from a dropdown:
   * st.data\_editor(df, column\_config={
   * "method": st.column\_config.SelectboxColumn(
   * options=["EQUAL\_MEAN","MEDIAN","VWAP\_RATIO","VOL\_WEIGHTED","WINSORIZED"]
   * )
   * })

That way, live trading will behave like the backtests: each stock can have its own **method**, **buy%**, **sell%**, and **budget**.