



bitflyer_lightning_fxにおけるmarket_make_bot(gava)に関する設計案

2019年3月

株式会社バーニングミート システムトレード部

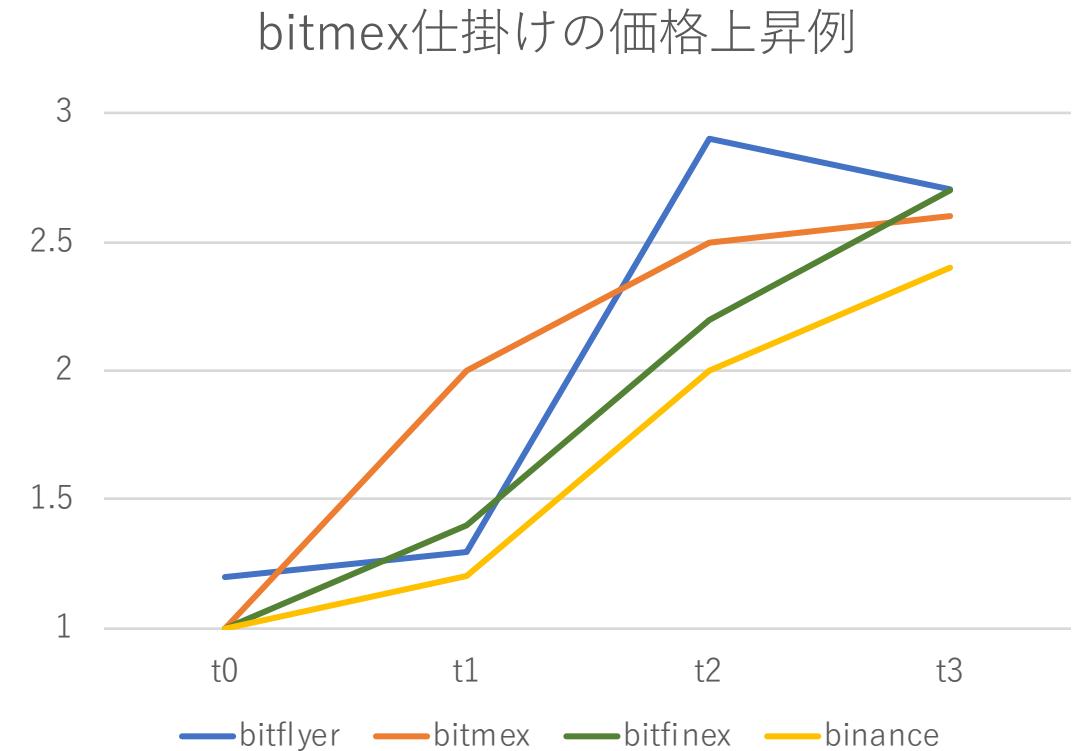
1. gava_botとは

他取引所での価格変動がどのようにbitflyer_lightning_fxで再現されるかを予測

bitflyerにおける価格変動が主要海外取引所に追従していると仮定、マーケットメイク戦略をとる

価格追従可能性

- 主要海外取引所で観測された価格変動が bitflyer 上ではどのように再現されるのかを分析
- bitcoin価格の取引所間での偏りが存在した場合は、十分な裁定がなされていないと判断
- 注文板の構成(指値注文)・約定履歴(成行注文)を基本情報とし、取引所をモデル化
- 参照されやすい数値(最頻値等)や、恒常的な価格変動を所与とし、bitflyer上で起こるであろう価格変動を予測

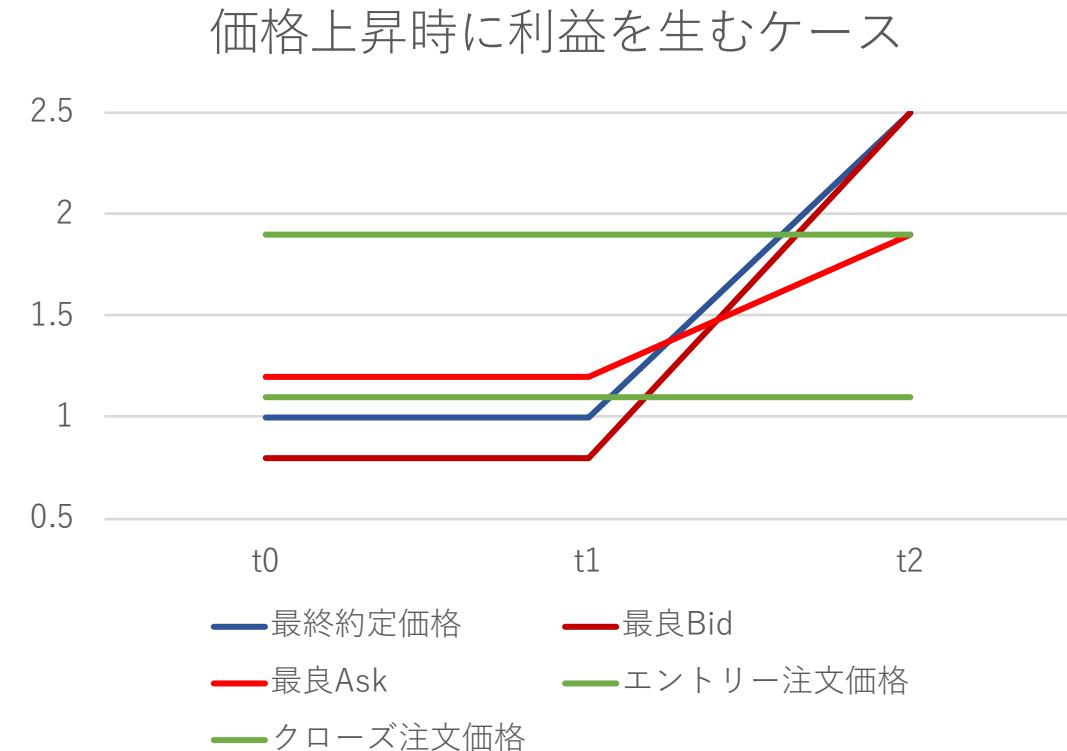


他取引所での価格変動がどのようにbitflyer_lightning_fxで再現されるかを予測

予想される価格変動の上限と下限の幅(スプレッド)を利益の源泉とし、予め指値をさしとく

マーケットメイク戦略

- bitflyer上で起こるであろう価格変動の幅を予測し、変動上限・下限それぞれ付近に指値をさす
- 注文のエントリーからポジションのクローズまでの1サイクルでもつ注文は売り・買いの一対とし、そのどちらも約定されることで、エントリー注文とクローズ注文の値幅がそのまま利益となる
- $| \text{エントリー注文価格} P_e - \text{クローズ注文価格} P_c |$ (注文の値幅 B_o)
 $\text{注文値幅} B_o \times \text{注文数量} S_o = \text{利益}$



2. botアーキテクチャー

プログラムの基本要件:高速・冗長・柔軟

速度低下を避ける/落ちやすいbiflyerのwebsocketを冗長化/投資戦略の更新・追加を容易に

高速

- appendを安易に使わない
- 一括データ入力はpandas
- 関数を使い回す
- コメントアウトは一行で
- for文を出来るだけ避ける
 - forじゃなくとも実装できないか調べる

冗長

- websocketの二重化
 - 落ちた場合の再接続処理も書く
- マルチプロセスでの管理にも気を使う
- 二重注文を避ける

柔軟

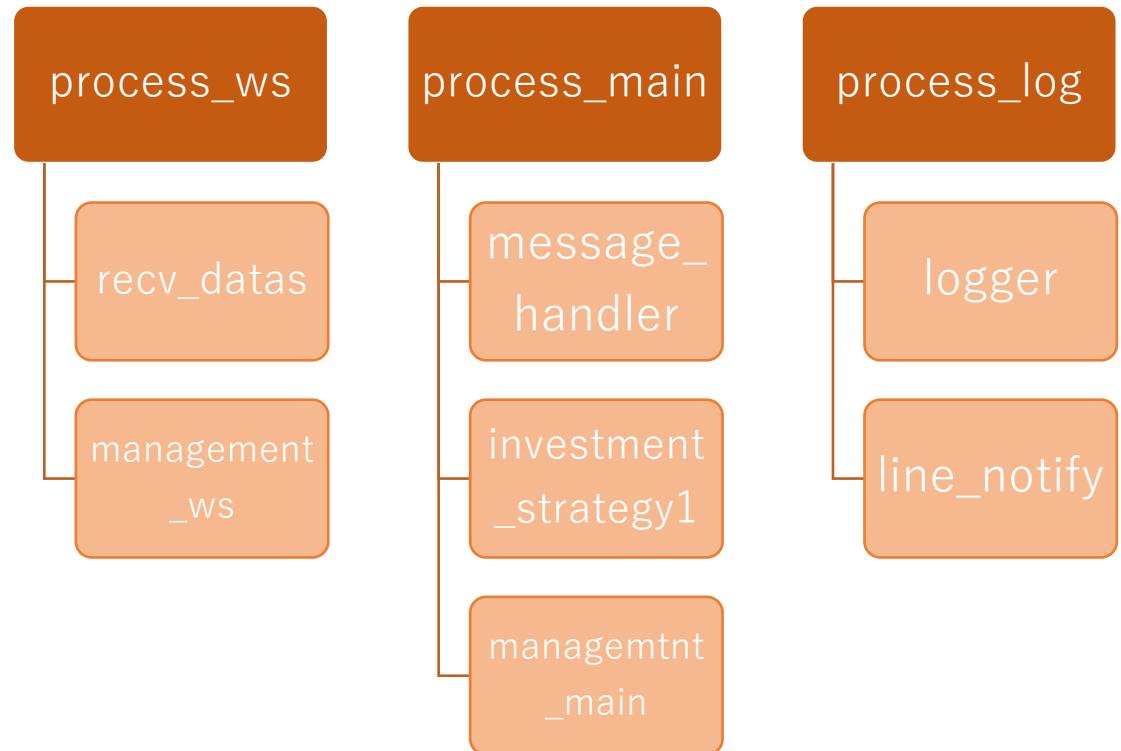
- これからのボットの基本的な枠組みとするべく開発
- 後から分かりにくくなるような書き方はしない
- 投資戦略あたりは変数を明確に定義
- 実行環境が変わっても動かしやすいように使うモジュールは最低限に

websocket/main/logの3プロセスでシステムを管理

各プロセス内はマルチスレッドで管理・リソースを使い切ると同時に可用性を確保

各プロセスの役割

- process_ws
 - 各取引所websocketを受信するプロセス
 - recv_datas
 - management_ws
- process_main
 - 取引に関するメインプロセス
 - message_handler
 - investment_strategy1
 - management_main
- process_log
 - ログを出力するプロセス



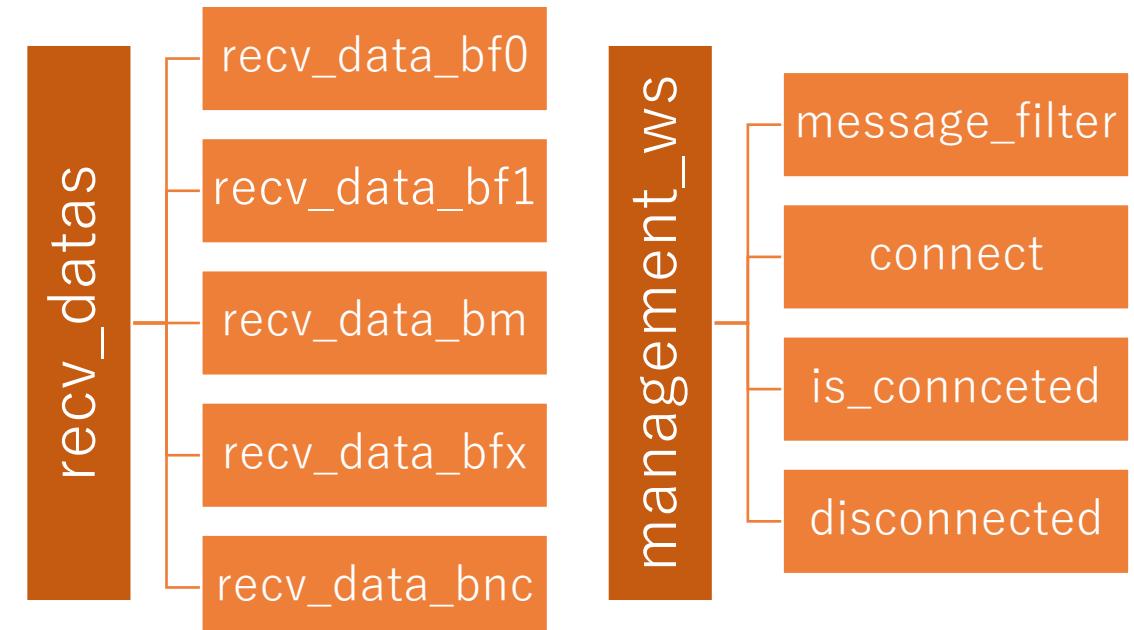
2.1 process_ws

各websocketのメッセージをprocess_mainにqueueを用いて渡す

recv_datasをmanagement_wsで管理・recv_datasの各関数はマルチスレッド化

process_ws内の処理

- recv_datas (各取引所に合わせたコールバック関数を用意)
 - recv_data_bf0 :bitflyer用(メイン)
 - recv_data_bf1 :bitflyer用(サブ)
 - recv_data_bm :bitmex用
 - recv_data_bfx :bitfinex用
 - recv_data_bnc :binance用
- management_ws (process_wsを管理)
 - message_filter :冗長化したメッセージをフィルタ
 - connect :websocket接続処理
 - is_connected :わからん
 - disconnected :websocket切断処理

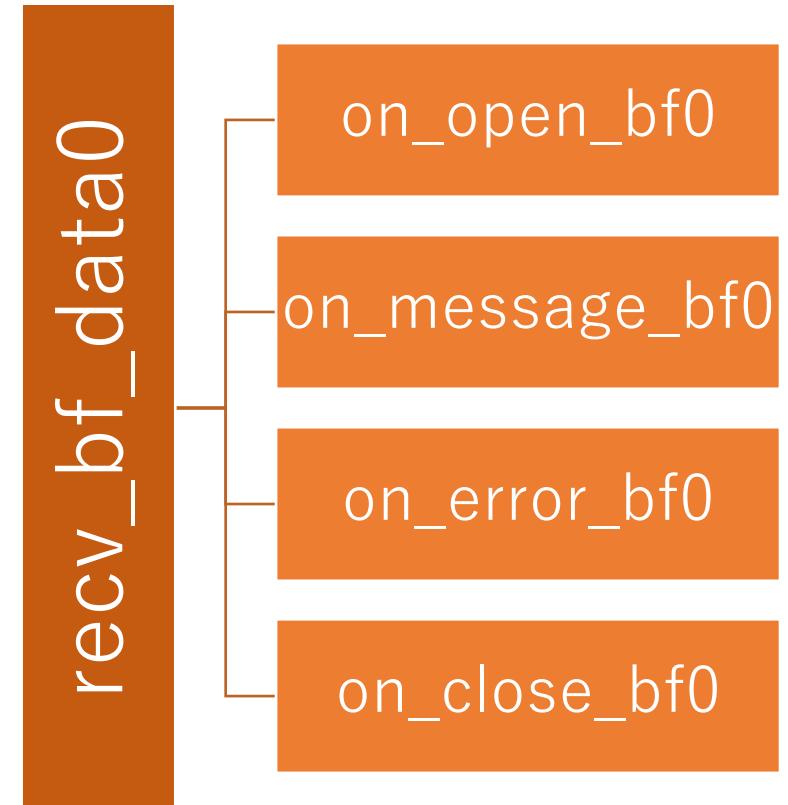


2.1.1 recv_datas

各取引所のコールバック関数

on_open/on_message/on_error/on_closeを明記・on_messageは短く
ex)recv_bf_data0内の処理

- on_open_bf0
 - 購読チャネルを指定
- on_message_bf0
 - messageをjsonとして受け取り、
message_filterにパス
 - board_snapshotだけは初回受信後に
unsubscribe
- on_error_bf0
 - 再接続処理
 - disconnectを実行し、time.sleep後、connect
- on_close_bf0
 - 終了操作



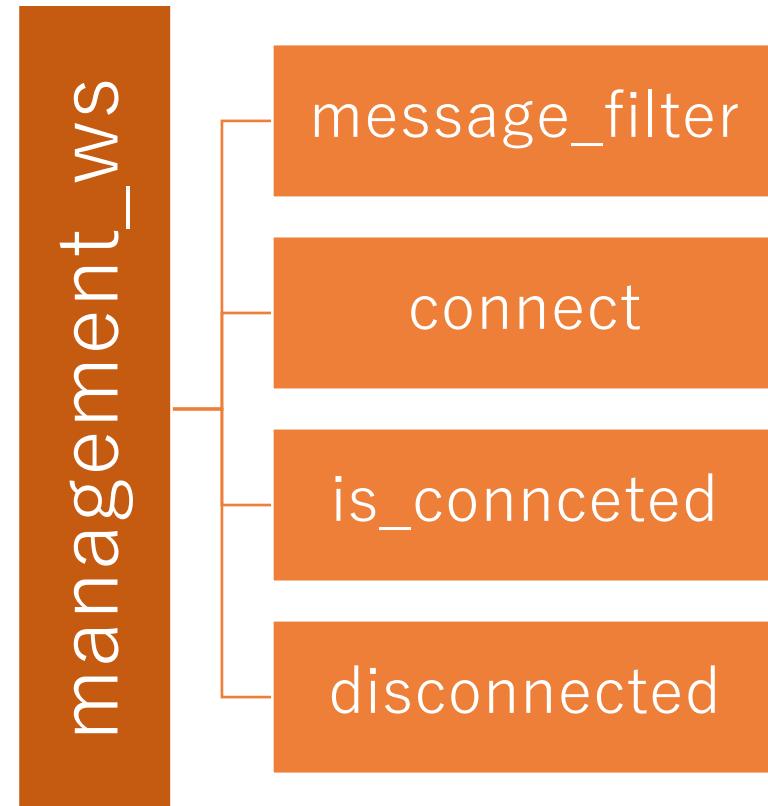
2.1.2 management_ws

process_wsを管理

メッセージのフィルタリング・接続、切断処理

management_ws内の処理

- message_filter
 - 二重にしている場合は早い方
 - メッセージ内容の相違チェック
 - 各チャネルごとに分解し、process_mainへ
- connect
 - 接続処理
 - websocketApp自体の設定
- is_connect
 - わからん
- disconnect
 - 切断処理
 - websocketApp自体の設定変更
- 各スレッドの立ち上げ



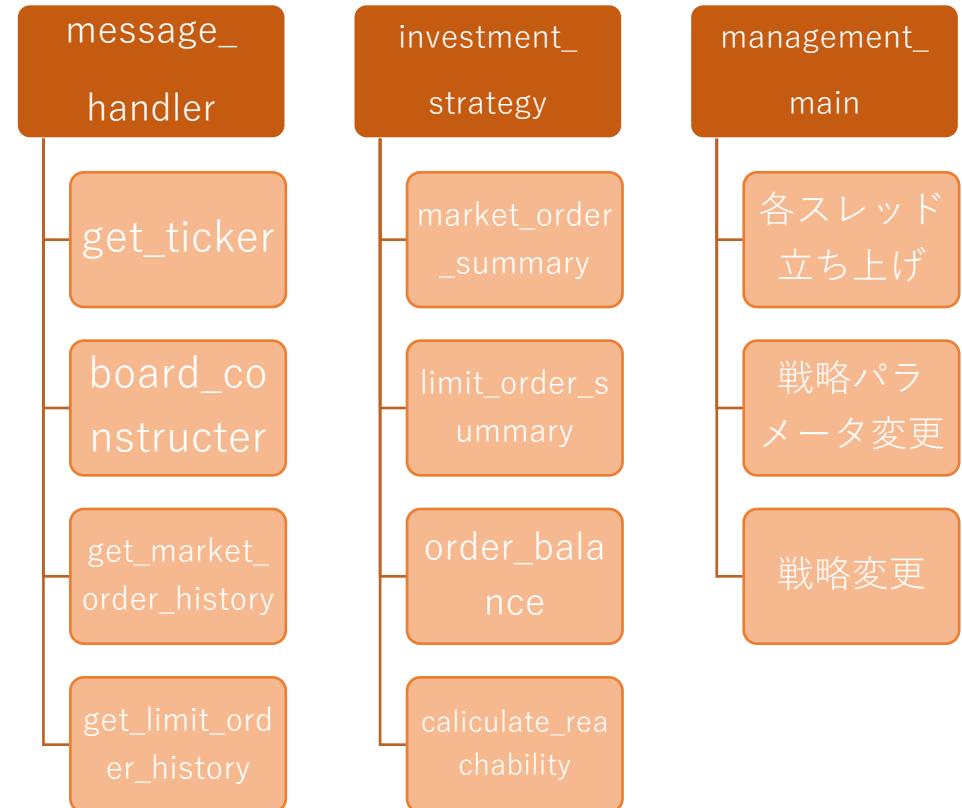
2.2 process_main

受け取ったメッセージから各取引所の現状を把握、裁定・注文発注

データ管理はpythonの基本的な配列・計算処理はpandas・注文はccxt・for文に注意

process_main内の処理

- message_handler(queueで受け取ったメッセージを分類)
 - get_ticker :ticker情報
 - board_constructor :注文板作成
 - get_market_order_history :約定履歴(成行注文)
 - get_limit_order_history :新規注文履歴(指値注文)
- investment_strategy
 - market_order_summary :成行注文の統計情報
 - limit_order_summary :指値注文の統計情報
 - order_balance :成行/指値の比率から需給バランスを計算
 - calculate_reachability :注文板ベースで価格変動予測
- management_main
 - 各スレッド立ち上げ
 - 戰略パラメータ変更
 - 戰略変更



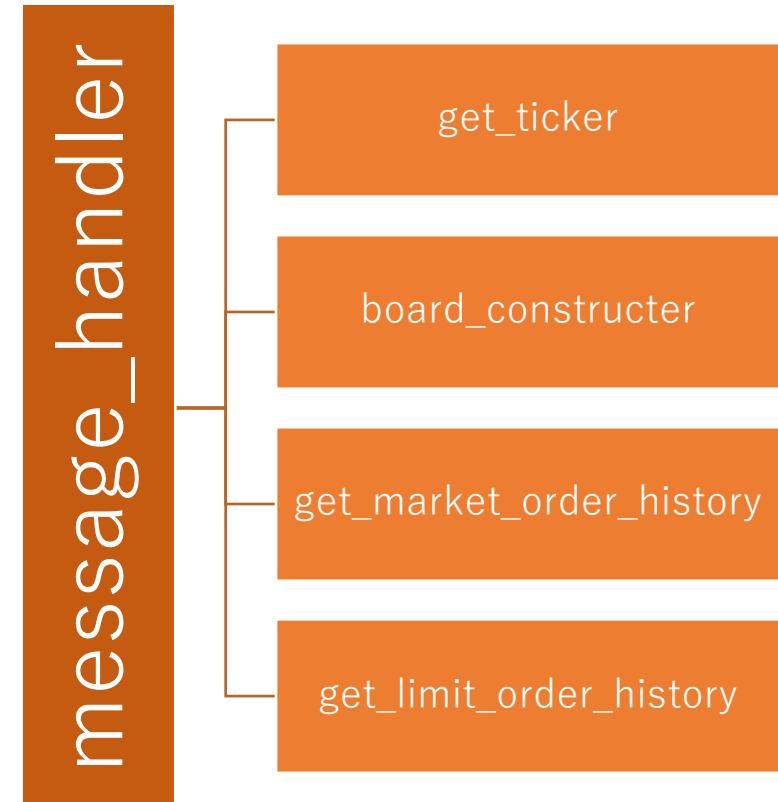
2.2.1 message_handler

recv_datasからのデータを元に各種データの取り出し・処理

データの取り出しへ内包表記を使うかpandasで一括で取り込む

message_handler内の処理

- get_ticker
 - ticker情報取得
 - micro_price等を計算
- board_constructer
 - 受け取った情報から注文板を構築
 - bids/asksの二つ管理し、price,size,timestampを明記
- get_market_order_history
 - 約定履歴作成
 - 成行注文に関する統計データを処理
- get_limit_order_summary
 - 指値履歴作成
 - 指値注文に関する統計データを処理



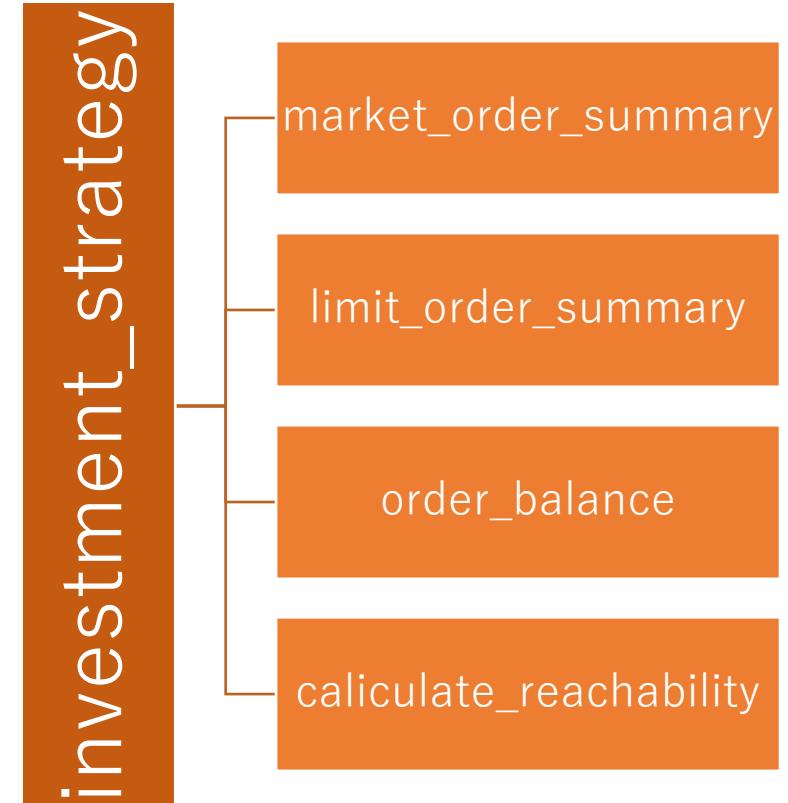
2.2.2 investment_strategy

投資戦略に基づき、注文の発注

これまで得られたデータをもとに裁定・注文発注を行う

investment_strategy内の処理

- market_order_summary
 - market_order_historyが処理したデータを受け取り、最頻値や成行価格・数量等を算出
- limit_order_summary
 - limit_order_historyが処理したデータを受け取り、最頻値や指値価格・数量等を算出
- order_balance
 - market_order_summaryとlimit_order_summaryが処理したデータを元に注文の需給バランスを算出
 - 現在のトレンドを判断(sliderの方向を決定)
- calculate_reachability
 - order_balanceが処理したデータを元に、変動しうる価格幅を注文板のデータと照合
 - 注文の値幅 B_o 、数量 S_o を決定し、注文を行う



2.2.3 management_main

process_mainを管理

戦略自体の変更や戦略パラメータをここに記述し、investment_strategyを扱いやすく

management_main内の処理

- 各スレッド管理
 - 立ち上げた各スレッドの立ち上げ・監視
- 戦略パラメータ変更
 - 現在の戦略で用いられているパラメータを変更できるように
 - パラメータを変数で置いとく
- 戦略変更
 - そもそも戦略の差し替え等を行う
 - 将来的に複数の戦略を市場の状況に応じて使い分けができるようにしておく
 - 我が社のAI戦略につながる基盤コード

