Rで始めるシステムトレード

LT

2019/3/2(Sat)

Hioki Ryuji

Introduction

- **twitter: @Liparas1729**

  homepage: hiokiryuji.com

- **Work**

  データ分析・可視化

  インフラ基盤

  フロント(React、Gatsby)、サーバーサイド

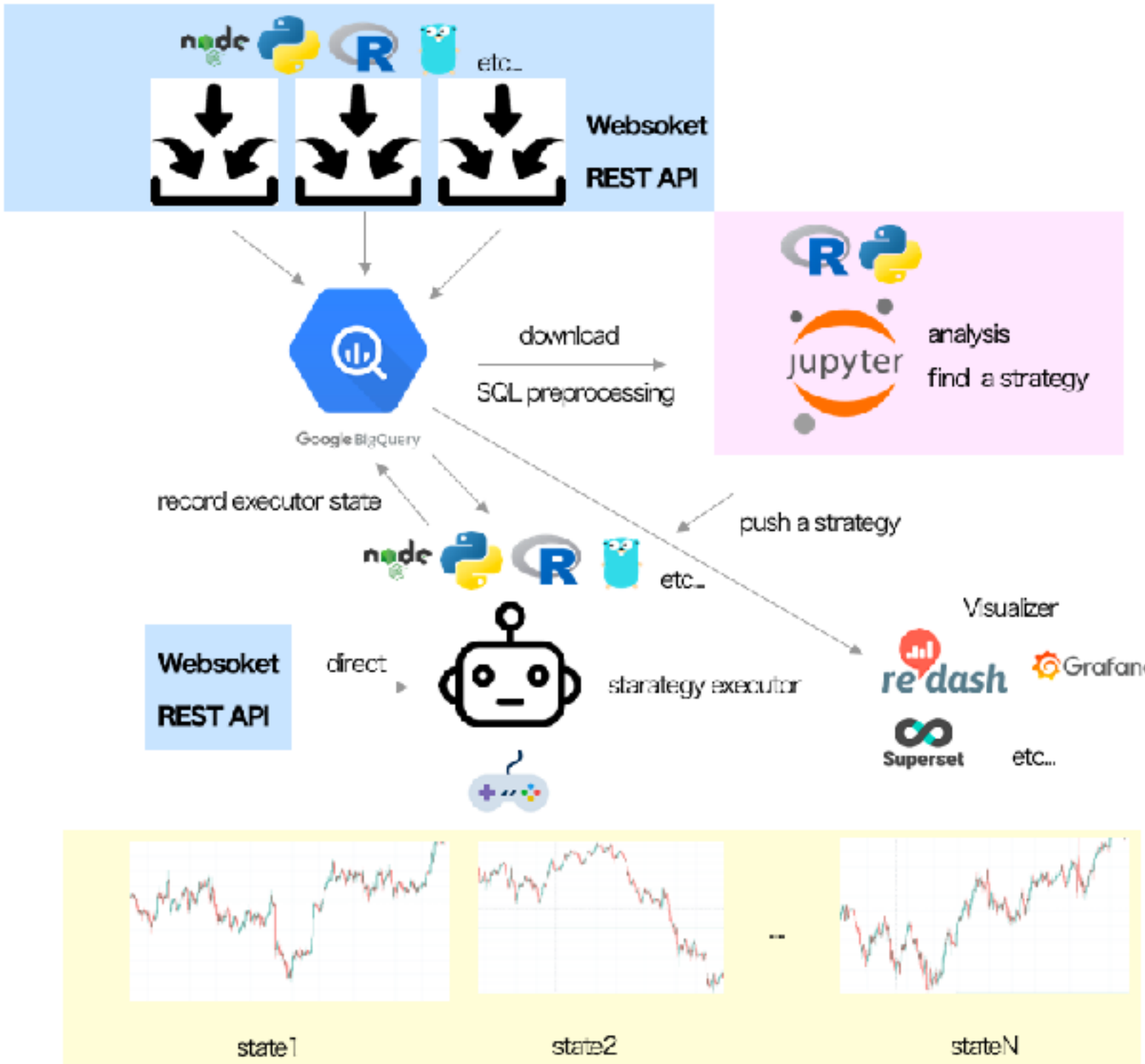
  Python/R/SQL/NodeJS/Golang/CSS

  AWS/GCP

- **Hobby**

  求職、大道芸、Kaggle、自然言語開発、言語学習、データ収集・可視化、サービス開発、読書
  UI/UX、心理学、統計、スマブラ

システム構築の標本


node
R
etc_
Websoket
REST API
Google BigQuery
download
SQL preprocessing
record executor state
push a strategy
jupyter
analysis
find a strategy
node
R
etc_
Visualizer
Websoket
REST API
direct
starategy executor
re dash
Grafana
Superset
etc...
state1
state2
..
stateN

データ収集①

◆ バックテストをして、統計的に戦略の有効性を確かめるには、
ある程度のデータ量が必要(時間軸による)

◆ Realtimeでデータを収集した場合、はじめのノードで、
集計・加工できた方が楽な場合も多いので、
SQL等が使えるデータベースが望ましい

◆ まずはデータ収集から

データ収集②

◆ データベース選抜

InfluxDB

Google BigQuery

GridDB

etc...

データ収集③

Rでwebsocketに接続する

```
library(websocket)

ws <- WebSocket$new("ws://echo.websocket.org/", autoConnect = FALSE)
ws$onOpen(function(event) {
  cat("Connection opened\n")
})
ws$onMessage(function(event) {
  cat("Client got msg: ", event$data, "\n")
})
ws$onClose(function(event) {
  cat("Client disconnected with code ", event$code,
      " and reason ", event$reason, "\n", sep = "")
})
ws$onError(function(event) {
  cat("Client failed to connect: ", event$message, "\n")
})
ws$connect()
```

データ収集④

Rでbigqueryに接続する

```
library(bigrquery)
billing <- bq_test_project() # replace this with your project ID
sql <- "SELECT year, month, day, weight_pounds FROM `publicdata.samples.natality`"

tb <- bq_project_query(billing, sql)
bq_table_download(tb, max_results = 10)
#> # A tibble: 10 x 4
#>     year month   day weight_pounds
#>    <int> <int> <int>         <dbl>
#>  1  1969     1    20          7.87
#>  2  1969     6    27          8.00
#>  3  1969     2    14          6.62
#>  4  1969     2     1          7.56
#>  5  1969     6     9          7.50
#>  6  1969    10    21          6.31
#>  7  1969     1    14          5.69
#>  8  1969     6     5          7.94
#>  9  1969     5     8          7.94
#> 10  1969     1     3          6.31
```

特徴量①

- **基本的なテクニカル指標(EMA、SMA、DMI、RSI、RCI、MACD等)は十分に機能するが、単体では使えない→調整が必要**

- **値そのものは、特徴量として使いづらい、**
  値そのものではなく変化率や差分を使うと良い

- **価格、取引量は板情報、約定データから算出される**

- **取引量は特に重要な指標**

特徴量②

Rでテクニカル指標を計算する

```r
install.packages("TTR")
library(TTR)

# "TTR Composite" (simulated data)
data(ttrc)

# Bollinger Bands
bbands <- BBands( ttrc[,c("High","Low","Close")] )

# Directional Movement Index
adx <- ADX(ttrc[,c("High","Low","Close")])

# Moving Averages
ema <- EMA(ttrc[,"Close"], n=20)
sma <- SMA(ttrc[,"Close"], n=20)

# MACD
macd <- MACD( ttrc[,"Close"] )

# RSI
rsi <- RSI(ttrc[,"Close"])

# Stochastics
stochOsc <- stoch(ttrc[,c("High","Low","Close")])
```

テクニカル指標は、大きく分けて2カテゴリー

・オシレーター系

・トレンドフォロー系

特徴量③

- NLPを使ってnewsなどで取引するものを作れるか

  →newsはoutlier(外れ値)であり、モデリングには不向き?

  →現状のNLPの精度が低いという事情もある

- また、newsを見て反応した人たちの動きはRealtimeデータの反映されうるので、Realtimeデータでも検知できる

戦略①

◆ 基本的な考え方は損小利大

◆ 価格が大きく変動することもあるので、もし負けたとしても、
統計的に優位な戦略において、サンプリングを続行できるように
資金管理を徹底する

◆ 上がるか、下がるかを予測するモデルで、
精度50%を超えることは容易だが、確率だけでは無意味で、
どれだけ価格が動くのかの倍率を管理することが最も大切

戦略②

◆　損失は大きく見積もって、利益は小さく見積もる

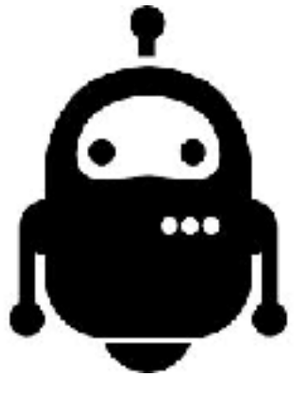
→ 損益の計算において、1rowずらして算出した未来の価格は、

取引執行からの遅延が0という理想的な値で実際の値とは異なる

◆　バックテストの段階で、停止する条件を定める

時間軸が大きくなると、一時的な損失(ドローダウン)も高くなる

◆　バックテストの結果から、損失を出しても統計的に想定の範囲内であれば続行できることが大切



データ可視化①

◆ データの可視化は2通りで

・戦略を決定する際のEDA

・戦略が想定通りに動作しているかの確認

◆ 戦略実行のログは、データベースに溜めておく

◆ 異常が起こった場合は、LINEや、Slack 、DiscordのAPI

を使って、通知できるようにしておくと安心

データ可視化②

- **チームでやる場合は、Redash、Grafanaなどで十分**

- **個人なら、Jupyter notebookやオレオレツールでも良い**

- **現在の損益、取引実行時刻、ポジション推移は**
  確認できるようにしておく

まとめ

・データベース選択→データ加工→分析・戦略の決定→戦略実行→可視化

・確率よりも、倍率をコントロールすることが大切

・相場によっては、大きく損失を出すこともあるので、統計的に
　利益を積み重ねるために、適切に資金を維持していく

・戦略の中には、停止条件をあらかじめ組み込んでおく

・テスト段階での未来の値は、理想的値に過ぎない

・データ収集→データ加工→データ分析→データ可視化
　シストレはデータサイエンスをやる上で、良い題材