Big Data Analysis

Fama Friends

Robert LYU
Evan HU
Trista FANG
Veronica PENG
Antoine SCHULER

Jan 13th 2019

OUR OBJECTIVES

1. **The Project**
2. **Data Cleaning**
3. **Alpha Calculation & Normalization**
4. **Single-Factor Test**
4. **Multi-Factor Model**
5. **Live Demo**

The Project

Multi-Factor

Financial investments deal with big data everyday

All market information posses the 3V features

Quantitative investments are the most common and effective use of Big Data in the field

Investment opportunities are written in water

Day trading allows big data to express itself to its full potential

In brief

Establish α-factors

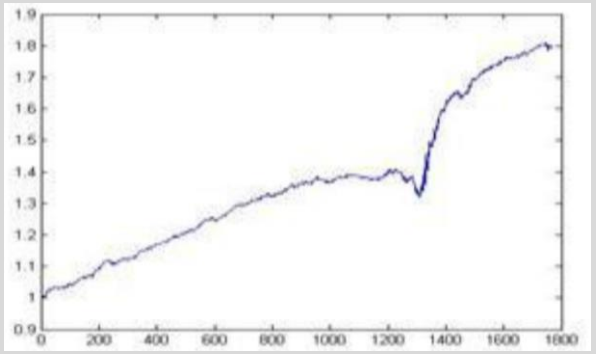
BARRA multifactor model

Form our Portfolio

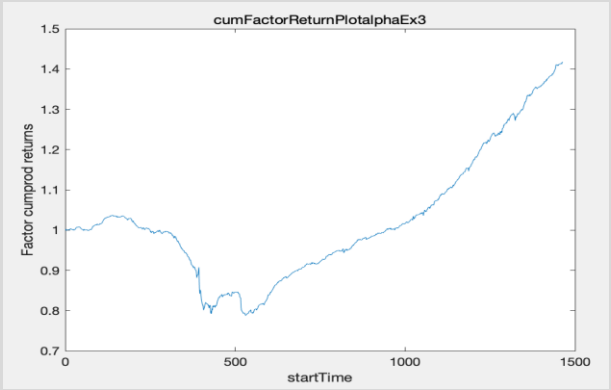
How do we make money?——an example of abnormal trading volume

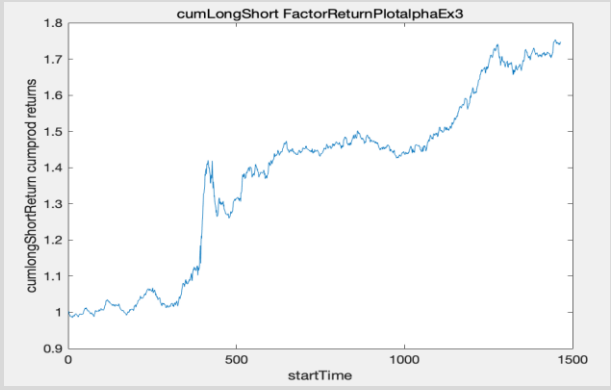
The research

Guotai Junan Securities



Replication





Data & Database

The Data

Volume

3842 Stocks (A-share)

2166 trade days: 04 Jan. 2011 – 28 Nov. 2019

Velocity

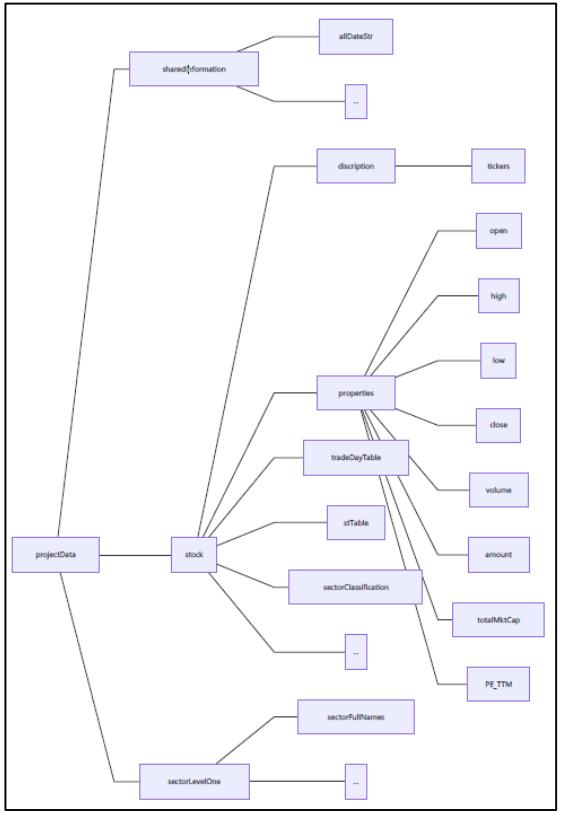
New day **everyday**

Variety

Sources

WIN.D

SHENWAN HONGYUAN Securities



Tickers

OHLC prices

P/E ratio (12m trail)

Special treatment

Volumes

Amount

Total capital market

Industries

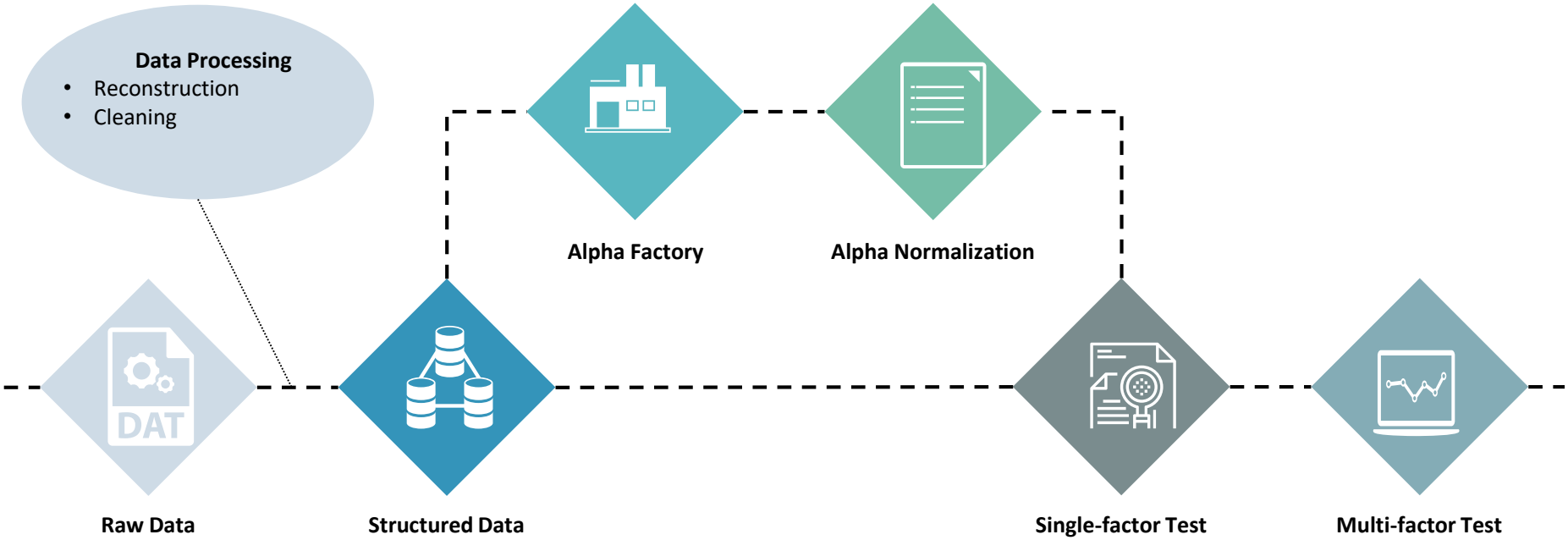
Storage

MATLAB as processing language

Stored in a struct

- Flexible
- Multi-layered
- Compact

Workflow


Data Processing
• Reconstruction
• Cleaning
DAT
Raw Data
Structured Data
Alpha Factory
Alpha Normalization
Single-factor Test
Multi-factor Test

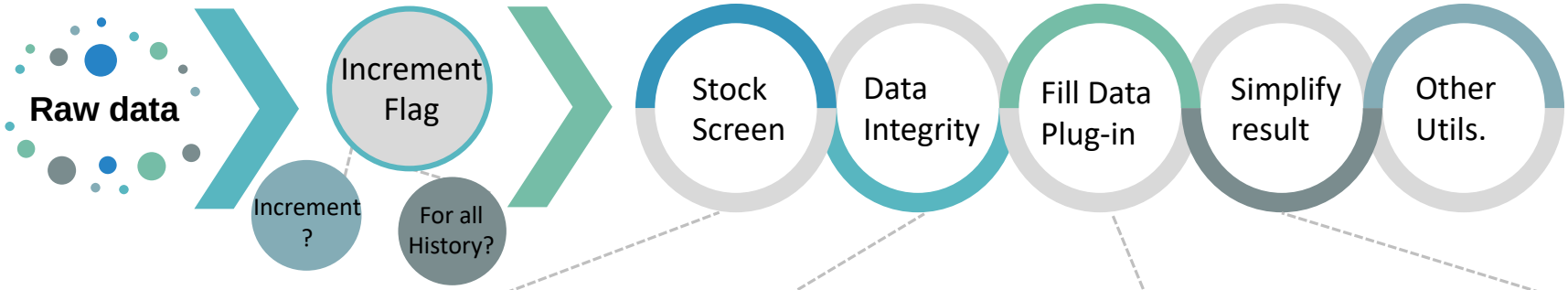
Data Cleaning

Multi-Factor

Data Cleaning Module

The module

- Controlled by 3 json files in Clean Data Config Dictionary



3 Rules

- 1- cumulative non-actionable days in a given past history
- 2- consecutive non-actionable days in a given past history
- 3- no tolerance non-actionable days in a given past history

Data integrity

- Count how many times a data point is used.
- Throw warning if unexpected missing exists

Fill Data

- If NaN detected: Initialize data imputation method
- must notify some imputation method will introduce future information.

Simplify

- Simplify key-value pair according to json file.

Clean Data Module: a example

The json Files

Display for the controlling json files

① fillDataMethod.json

```
{
    "fillHead":["nearest"],
    "fillCommon":["previous"]
}
```

② tableNamesToSelect.json

```
{
    "stock.properties.close":[],
    "stock.properties.high":[],
    "stock.properties.low":[],
    "stock.properties.open":[],
    "stock.properties.amount":[],
    "stock.properties.volume":[],
    "stock.properties.PETTM":[],
    "stock.properties.PE":[],
    "stock.totalReturn":[],
    "stock.properties.totalMktCap":[],
    "stock.properties.floatingShares":[],
    "stock.tradeDayTable":[],
    "stock.stTable":[]
}
```

③ tradeableStocksSelectionCriteria.json

```
{
    "settingClean01":{
                      "maxConsecutiveInvalidLength":[0,0],
                      "maxConsecutiveRollingSize":[0,0],
                      "maxCumulativeInvalidLength":[60,100],
                      "maxCumulativeRollingSize":[200,200],
                      "noToleranceRollingSize":[100,100],
                      "flag":0
                      },
    "settingRefer01Table":["stock.tradeDayTable",
                           "stock.stTable"],
    "settingValidIndicator":[1,0]
}
```

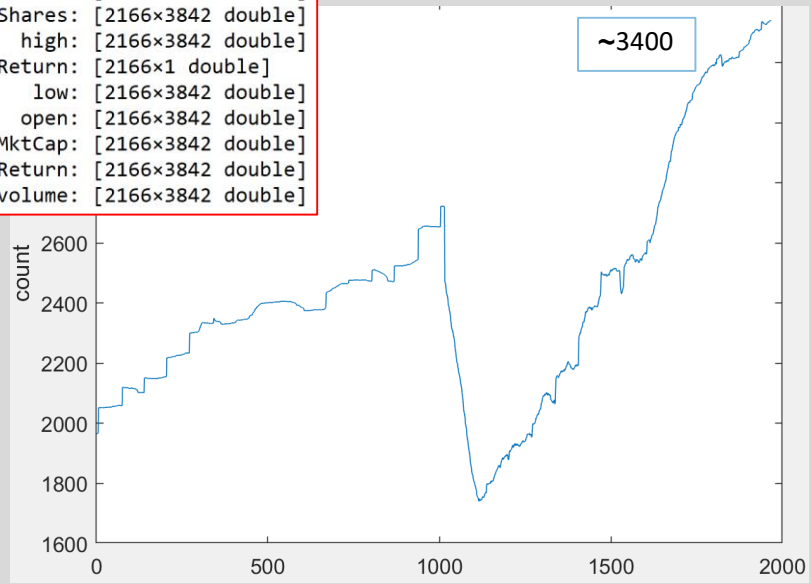
The output

```
cleanData =

  struct with fields:

                PE: [2166×3842 double]
             PETTM: [2166×3842 double]
            amount: [2166×3842 double]
             close: [2166×3842 double]
     floatingShares: [2166×3842 double]
              high: [2166×3842 double]
   indexTotalReturn: [2166×1 double]
               low: [2166×3842 double]
              open: [2166×3842 double]
       totalMktCap: [2166×3842 double]
       totalReturn: [2166×3842 double]
            volume: [2166×3842 double]
```

Count of tradable stocks by trading days



Alpha Calculation & Normalization

Multi-Factor

Alpha Factory

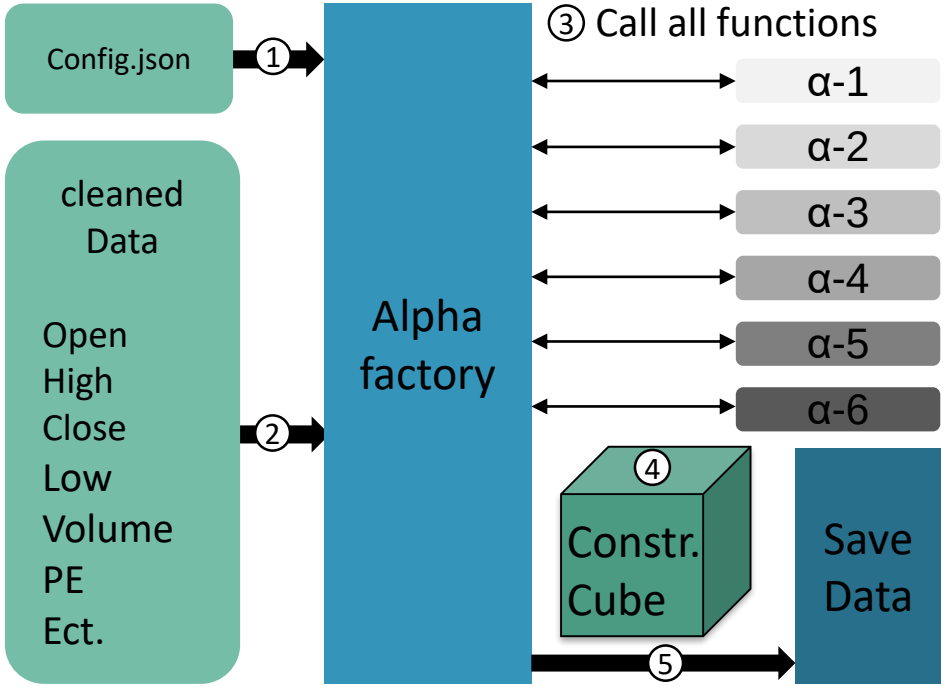
The module

- Calculate all factor exposures
- In batches
- Leveraging cleaned data from latter phase

Process

- Import from config file
- Import cleaned data
- Call all functions
- Construct cube (alphaLoadings)
- Save data

Alpha Factory Workflow



Alpha factory: in depth

Fct. alpha = getAlpha():

```
function alpha = getAlpha(alphaName, alphaPara)

    if isstruct(alphaPara)
        alpha = feval(alphaName, alphaPara);
    end
end
```

Fct. saveALLAlphaHistory():

```
function saveAllAlphaHistory(obj,filePrefix ,saveStucture)
```

%preparation of save

```
 save as 3 dim mat
exposure = [];
alphaNameList = {};

%alphaNameList = [];
for k=1:length(targetAlphas)
    alphaName=targetAlphas{k};
    disp("start process:"+ alphaName);
exposure = cat(3,exposure,obj.getAlphaHistory(alphaName));
alphaNameList{end+1} = alphaName;
%alphaNameList = [alphaNameList alphaName];
disp("success")
```

%save data

Normalizing Exposures

The module

- Uses alpha loadings from previous phase for 2 steps:
  - Process extreme values
  - Normalize

Winsorizing

1

- Factor loadings can be **large** & **away** from the median
- compress the values whose absolute values are bigger than a setting value to certain ranges with the Winsor method

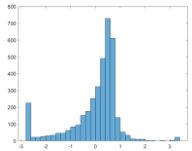
$$\widetilde{x_i} = \begin{cases} x_M + n \times D_{MAD}, & if \quad x_i > x_M + n \times D_{MAD} \\ x_M - n \times D_{MAD}, & if \quad x_i < x_M - n \times D_{MAD} \\ x_i, & else \end{cases}$$

x_i is the loading on stock i of a factor on a single date;

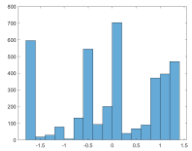
x_M is the median of cross-sectional loading over stocks of the factor;

D_{MAD} is the median of the sequence $|x_i - x_M|$.

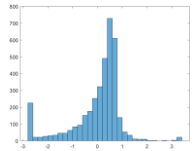
Extreme processing figures



α-factor #4



α-factor #14



α-factor #22

Normalization

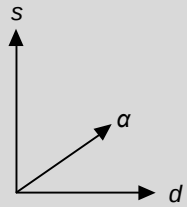
2

- Using **z-score** to normalize the processed factor loadings.
- Save **mean**, **median**, **skewness** and **kurtosis** of the distribution for further examination.

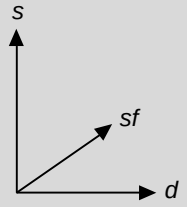
Class: factor Normalization

3

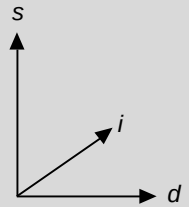
- Following 3 variables as input:



factorCube



styleFactorCube



industryFactorCube

- Returns: Normalized factor loadings

Orthogonalization

4

- Use the normalized factor loadings to do **regression** with the **style** and **industry factors** to get the residuals.

Return:

Orthogonalized Factor Loadings

Single Factor Testing

BARRA Multi-Factor

BARRA – The model & CNE5 Industry Factor

The BARRA model

- Commonly used multifactor model
- Positive AR for daily investments (T+1)
- Provide countless new α-factors

BARRA & Factor Testing

- Factor: Element to explain a certain characteristic of a stock
  - Factor exposure
  - Factor return

$$k = 1, 2, \ldots, N$$

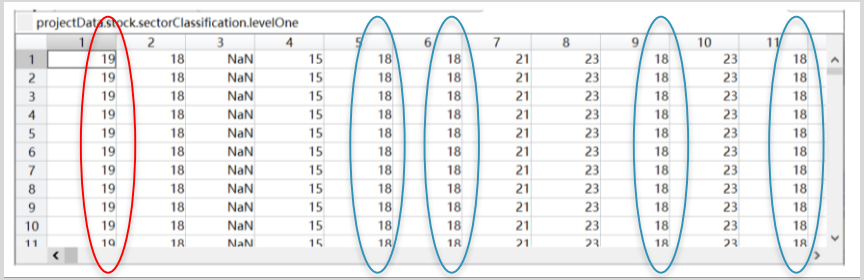
$$r_k^{t+1} = f_{industry}^{(t)} X_{industry}^{(t)} + f_{style}^{(t)} X_{style}^{(t)} + f_{factorReturn}^{(t)} \varepsilon_k^{(t)}$$

Process

- regression on factor exposure
- acquire each factor return
- test the factor's significance
  - Effectiveness
  - Stability.

Industry factor in the Chinese market

- 34 factors
- Binary representation:
  - 1 – in the industry
  - 0 – not in the industry
- The above is a **specificity** to the Level 1 classification of *SHENWAN HONGYUAN* Securities
- Made to represents the **core business** of a company



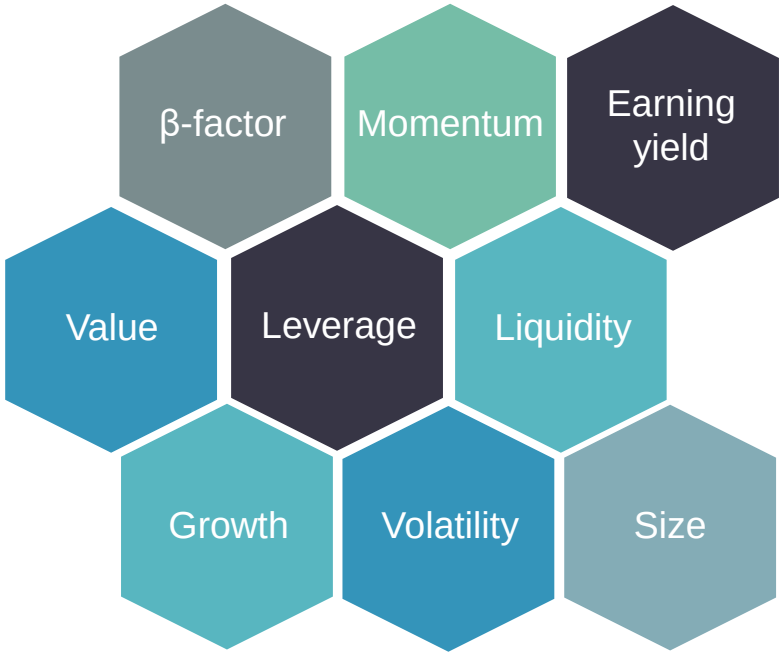
	1	2	3	4	5	6	7	8	9	10	11
1	19	18	NaN	15	18	18	21	23	18	23	18
2	19	18	NaN	15	18	18	21	23	18	23	18
3	19	18	NaN	15	18	18	21	23	18	23	18
4	19	18	NaN	15	18	18	21	23	18	23	18
5	19	18	NaN	15	18	18	21	23	18	23	18
6	19	18	NaN	15	18	18	21	23	18	23	18
7	19	18	NaN	15	18	18	21	23	18	23	18
8	19	18	NaN	15	18	18	21	23	18	23	18
9	19	18	NaN	15	18	18	21	23	18	23	18
10	19	18	NaN	15	18	18	21	23	18	23	18
11	19	18	NaN	15	18	18	21	23	18	23	18

- E.g.:
  - 18: real estate industry
  - 19: financial services industry

BARRA –Style Factors

The BARRA & style factor

- Use of style factor for risk decomposition



Industry factor in the Chinese market

- We use the same method as alpha calculation to construct the style factor.

- BETA.m
- DASTD.m
- ETOP.m
- HSIGMA.m
- LNCAP.m
- RSTR.m
- STOA.m
- STOM.m
- STOQ.m

Size： LNCAP
Natural log of market cap

Beta: beta

$$r_i - r_{ft} = \alpha + \beta R_t + \epsilon_t$$

Momentum: RSTR

$$\sum_{t=L}^{T+L} w_i [ln(1 + r_t) - ln(1 + r_{ft})]$$

Single Factor Testing

Purpose

```
data    _ 00 description
       |_ 01 cleaneData
       |_ 02 styleFactor
       |_ 03 factorExposure
       |_ 04 factorNormalization
       |_ 05 singleFactorTest
```

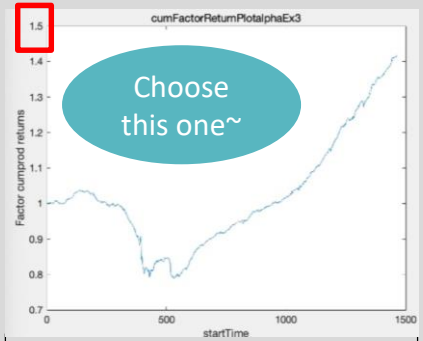
- For each time ***t***, the alpha k has its unique f_k .
- For a given period of time, we can get time series of f_k
- To test the validity of factors, each factor need to pass a series of statistical tests.

The factor **must** provide explanatory power to portfolio returns and have delivered a premium.

- **Persistent** , **Robust**, **Investable**, **Intuitive**, **Pervasive**

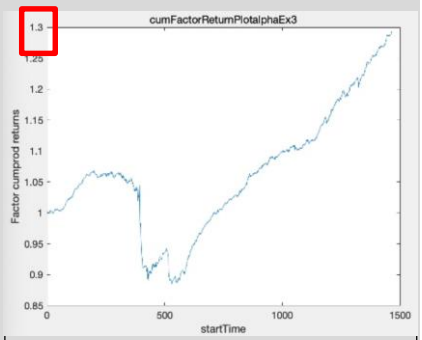
Orthogonal with style factor or not?

- In Barra's risk model, industry factor and style factor are **added**.
- The **style factor** explains **most** of the stock returns.



Non orthogonal with style

Factor cumprod returns: **~1.5**



Orthogonal with style

Factor cumprod returns: **~1.3**

- From the perspective of investable, the alpha we use is the result of normalization, orthogonal to industry and not orthogonal to style.

Single Factor testing: application

T-stat

- test the significance and stability of **factor return' coefficient** in regression

  - t Signaficance

    $H0 : mean(|t_{f_k}(T)|) = 0$

  - t Stationarity

    $H0 : |t| > 2$ or $ADFtest$: H0: the series is not stationary.

F-stat

- test the significance and stability of **factor returns**

  - f_k Significance

    $H0 : mean(f_k) = 0$ or $H0 : |mean(f_k)| = 0$

  - f_k Stationarity

    $std(f_k) = 0$ or $ADFtest$

1x31 struct 包含 7 个字段

...	tSignaficance	tStationarity	fkSignaficance	fkStationarity
1	1	0	[0,1]	0
2	1	0	[0,1]	0
3	1	0	[0,1]	0
4	1	0	[0,1]	0
5	1	0	[0,1]	0
6	1	0	[0,1]	0
7	1	0	[0,1]	0
8	1	0	[0,1]	0
9	1	0	[0,1]	0
10	1	0	[0,1]	0
11	1	0	[0,1]	0
12	1	0	[0,1]	0
13	1	0	[0,1]	0
14	1	0	[0,1]	0
15	1	0	[0,1]	0
16	1	0	[0,1]	0
17	1	0	[0,1]	0
18	1	0	[0,1]	0
19	1	0	[0,1]	0
20	1	0	[0,1]	0
21	1	0	[0,1]	0

Single Factor testing: application (cont.)

Information Coefficient

- test the significance and stability of factors to the future expectation ability.

- Normal IC

$$IC_{\alpha}^{t} = corr(\mathbb{E}(\epsilon_{t+1}), \epsilon_{t+1}).$$

- Rank IC

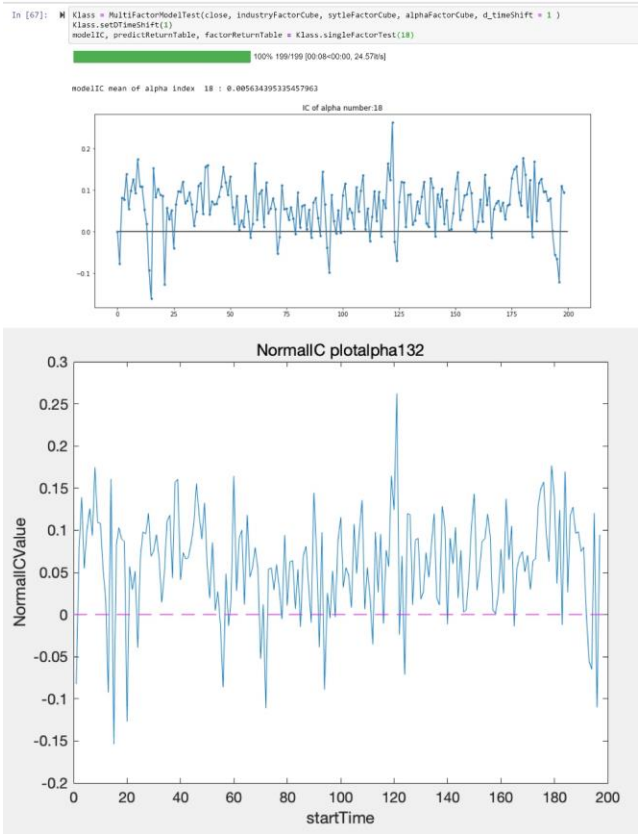
$$IC_{\alpha}^{t} = spearmanCorr(\mathbb{E}(\epsilon_{t+1}), \epsilon_{t+1}).$$

- IC significance

$$H0 : mean(IC) = 0 \text{ or } H0 : |mean(IC)| = 0$$

- IC stationary

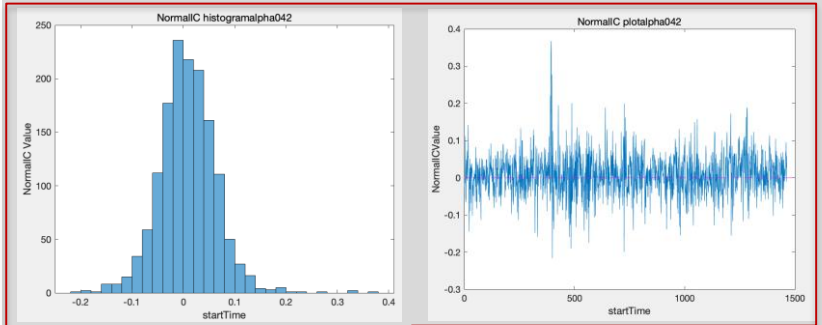
$$std(IC) = 0 \text{ or } ADFtest$$



Single Factor testing: an example

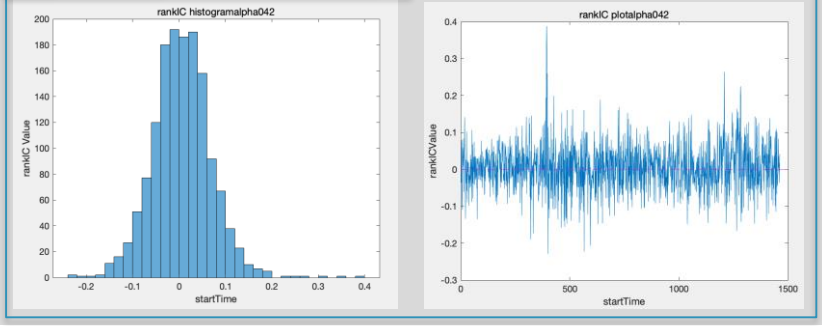
Alpha042

Alpha42 *((-1 * RANK(STD(HIGH, 10))) * CORR(HIGH, VOLUME, 10))*



Normal IC	Normal ICIR
0.0099	0.1139

Rank IC	Rank ICIR
0.0085	0.0874

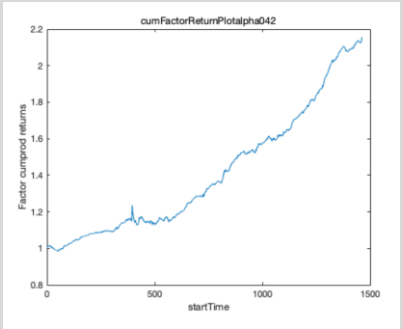


Summary Single Factor Testing

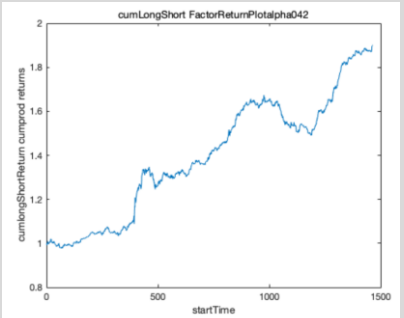
- After one factor done over the test, we can get a summary result of this factor. After whole factors done over the statistics test, we can save the whole result

tSignificance	tStationarity	fkSignificance	fkStationarity	ICSignificance	ICStationarity	totalNumber
1	0	[0,1]	1	[1,1]	[1,0,0]	6
1	0	[0,1]	1	[1,1]	[1,0,0]	6

- Plot cumprod factor return



- Plot Long-Short cumprod Portfolio



Feature selection

BARRA Multi-Factor

Alpha selection

Correlation

- Plot everyday correlation heatmap
- VIF Judge > 5 to delete the alpha

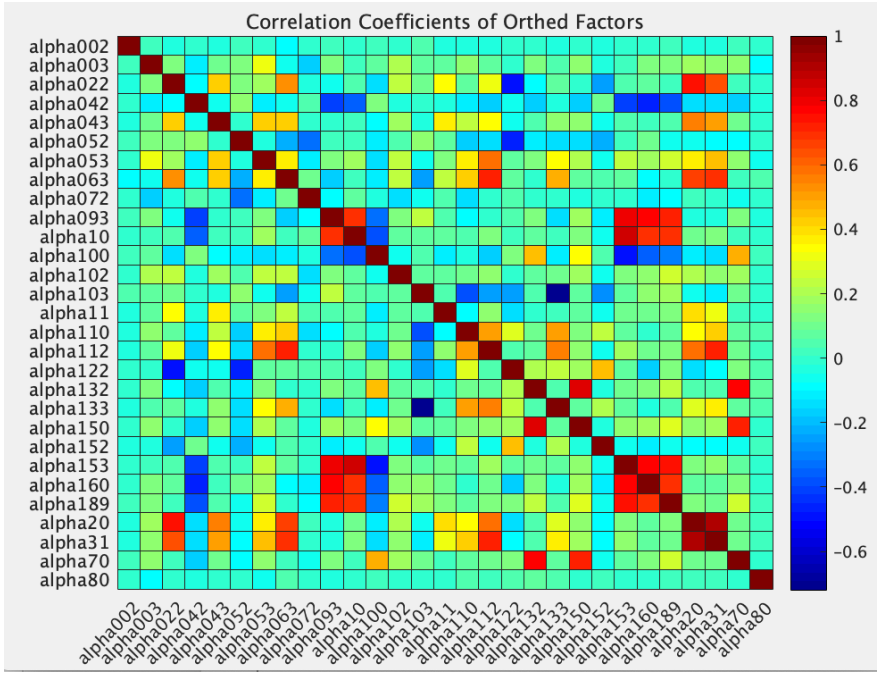
$$VIF = 1/(1 - {R_i}^2)$$

- Save everyday pass alpha results

OR/AND

Ridge & Lasso regression

- Lasso is more suitable for variable selection
- Lasso will delete more variables than ridge's threshold



saveSelectAlpha{1, 1}

	1	2	3	4	5	6	7	8
1	alpha002	alpha003	alpha022	alpha042	alpha043	alpha052	alpha053	alpha072

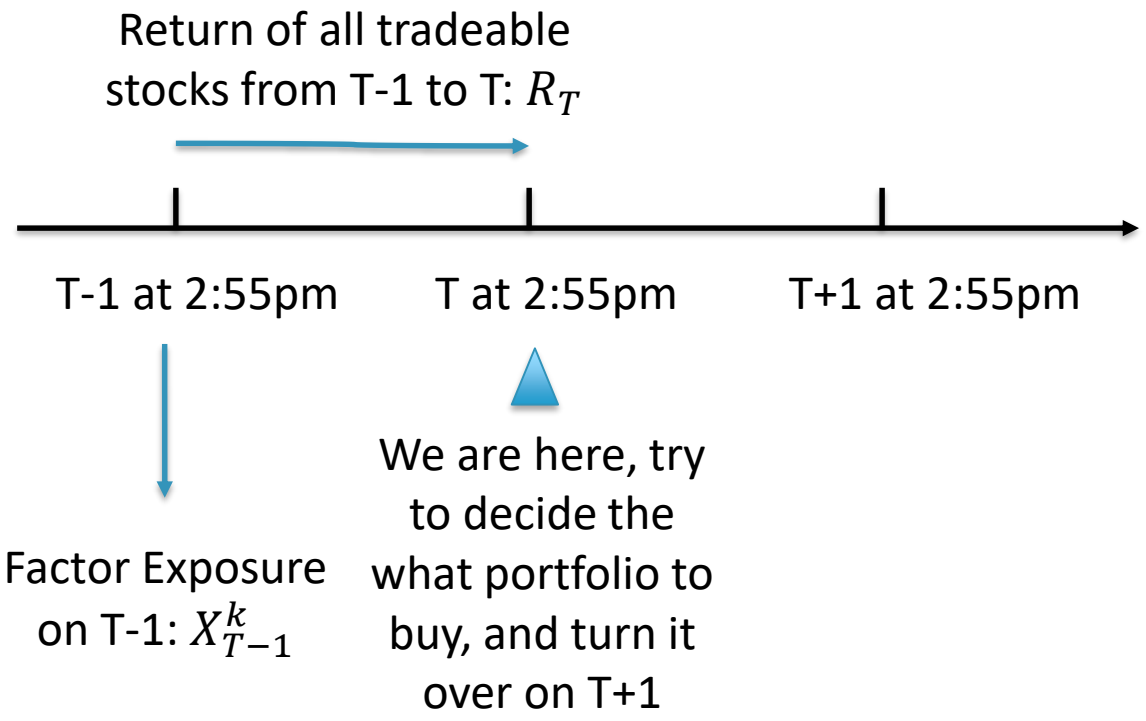
saveSelectAlpha

saveSelectAlph...

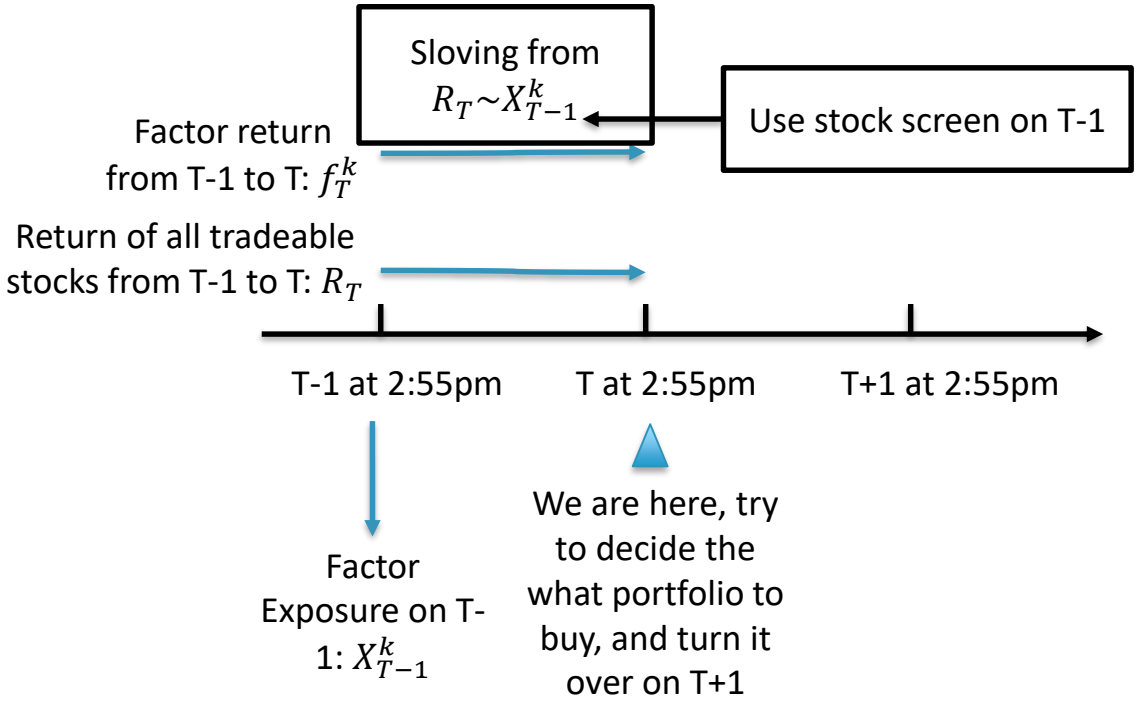
Multi-Factor Model

Multi-Factor

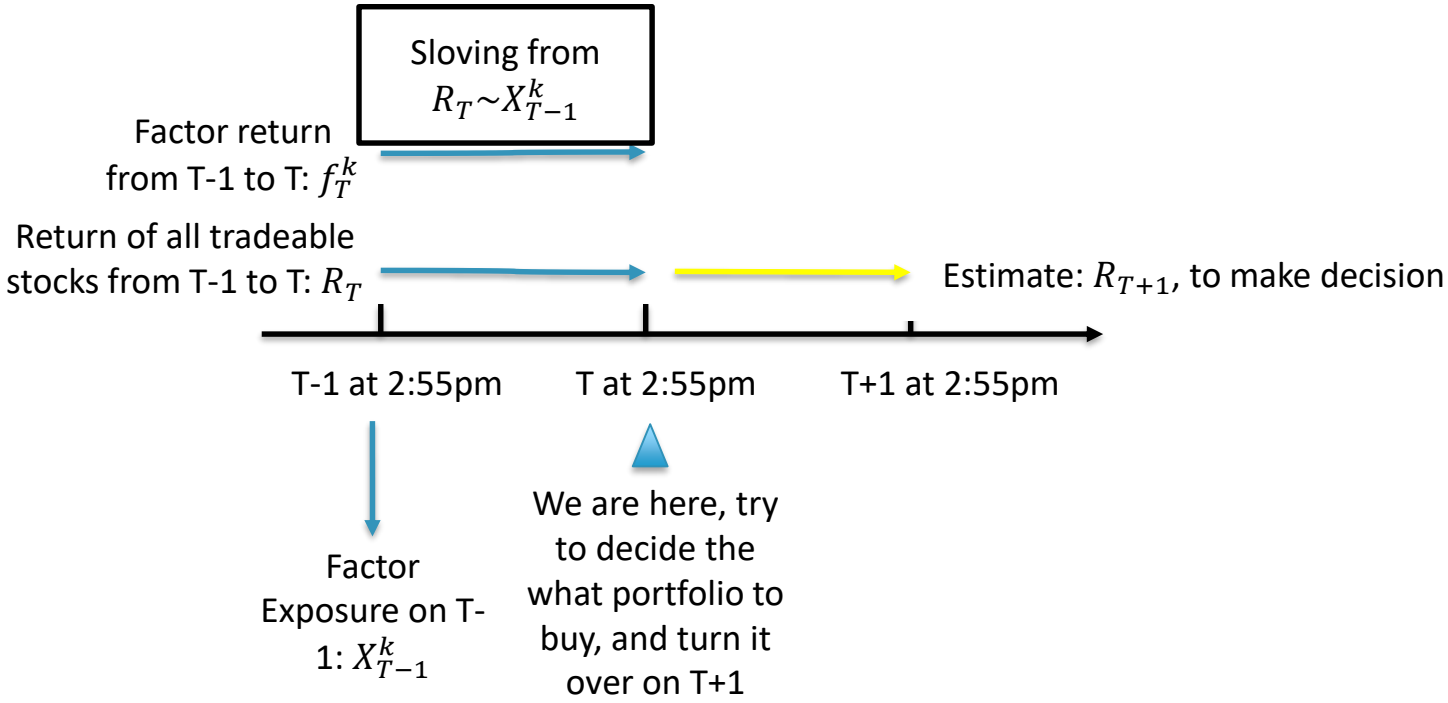
Multi-Factor Model: scenario



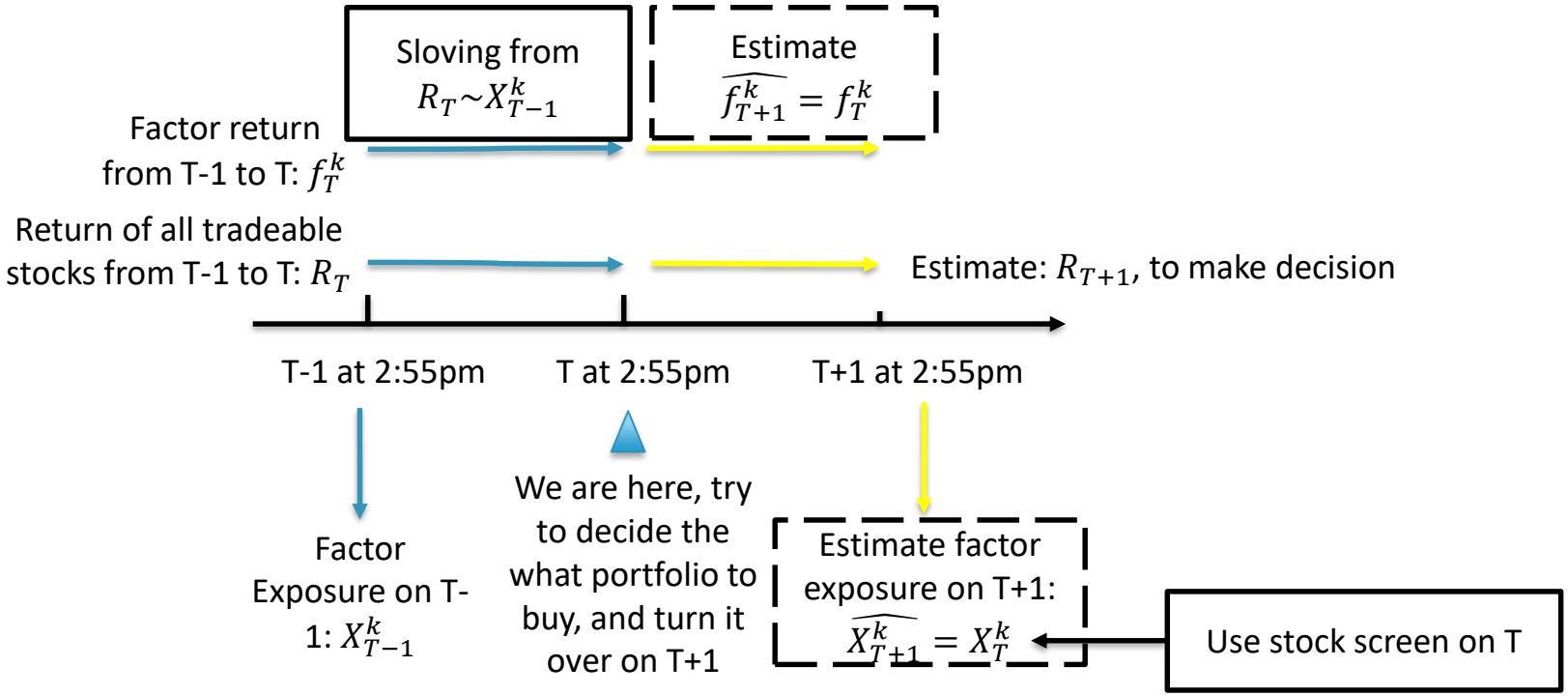
Multi-Factor Model: scenario



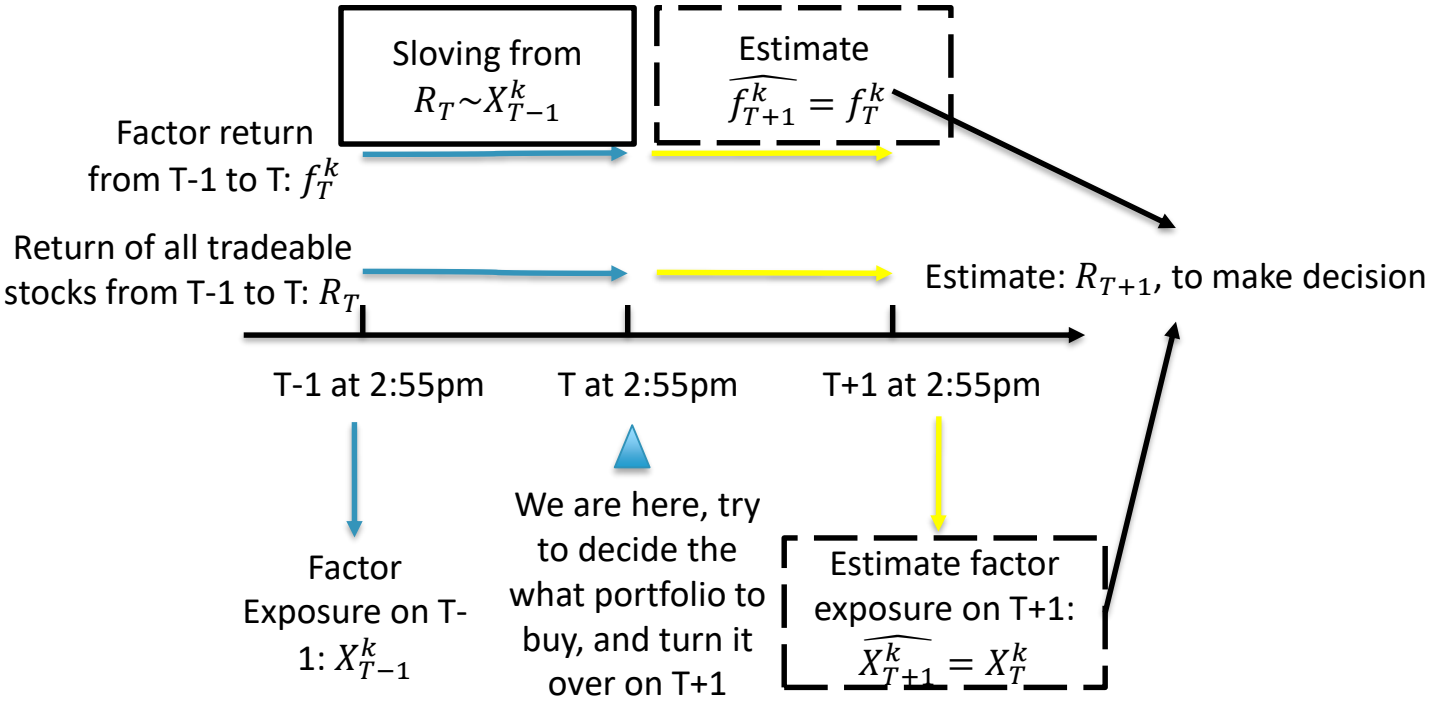
Multi-Factor Model: scenario



Multi-Factor Model: scenario



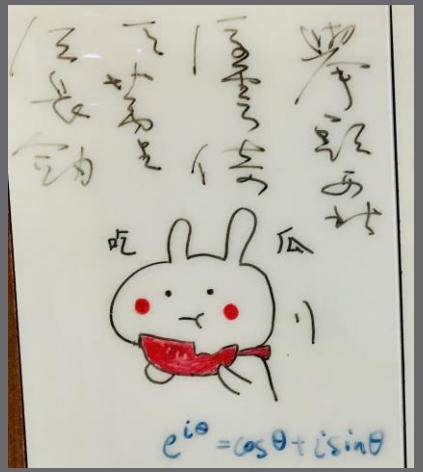
Multi-Factor Model: scenario



Live Demo

Multi-Factor

Acknowledgement



Bonus scenes