**TRC calculation code：**

filename = "C:\Users\13683\Desktop\TOP100.xlsx";

num\_iterations = 943; % number of iterations

num\_stocks = 100;

weights\_matrix = zeros(num\_stocks, num\_iterations);

% Iterate one by one, read 30 rows of data every other row

for i = 1:num\_iterations

start\_row = 1 + (i - 1) \* 31;

end\_row = start\_row + 29;

data = xlsread(filename, 'Sheet2', sprintf('A%d:CZ%d', start\_row, end\_row))';

target\_rc = 1 / num\_stocks;

% Calculate the standard deviation (risk) and average return for each stock

risk = std(data, 0, 2);

mean\_returns = mean(data, 2);

fun = @(weights) sum((weights .\* risk / sum(weights .\* risk) - target\_rc).^2);

initial\_weights = ones(num\_stocks, 1) / num\_stocks;

Aeq = ones(1, num\_stocks);

beq = 1;

lb = zeros(num\_stocks, 1);

ub = ones(num\_stocks, 1);

options = optimoptions('fmincon', 'Algorithm', 'interior-point', 'Display', 'off');

weights = fmincon(fun, initial\_weights, [], [], Aeq, beq, lb, ub, [], options);

weights\_matrix(:, i) = weights;

disp(['current iteration count：', num2str(i)]);

end

weights\_matrix=weights\_matrix';

**ICIR code：**

import numpy as np

from scipy.stats import pearsonr

import pandas as pd

file1\_path = "C:/Users/13683/Desktop/外盘内盘比(6).xlsx"

file2\_path = "C:/Users/13683/Desktop/开盘价收益率(1)(2).xlsx"

factor\_data = pd.read\_excel(file1\_path)

returns\_data = pd.read\_excel(file2\_path)

if factor\_data.shape == (972, 2224) and returns\_data.shape == (972, 2224):

print("The data is read successfully, and the data size of the two Excel files is the same。")

else:

print("Data reading failed, please check the data size of the Excel file。")

print("Factor Data Shape:", factor\_data.shape)

print("Returns Data Shape:", returns\_data.shape)

num\_stocks = 2224

def calculate\_ic(factor\_data, returns\_data):

factor\_data = np.array(factor\_data) # Convert factor\_data to NumPy array

returns\_data = np.array(returns\_data) # Convert returns\_data to NumPy array

num\_stocks = factor\_data.shape[1]

ic\_values = []

for i in range(num\_stocks):

factor\_values = factor\_data[:, i]

stock\_returns\_data = returns\_data[:, i]

ic, \_ = pearsonr(factor\_values, stock\_returns\_data)

ic\_values.append(ic)

return ic\_values

def calculate\_ir(returns\_data):

avg\_excess\_returns = np.nanmean(returns\_data)

excess\_returns\_std = np.nanstd(returns\_data)

ir = avg\_excess\_returns / excess\_returns\_std

return ir

def calculate\_ic\_and\_ir\_for\_stocks(factor\_data, returns\_data):

ic\_values = []

ir\_values = []

for i in range(num\_stocks):

stock\_factor\_data = factor\_data.iloc[:, i].values # Convert to NumPy array

stock\_returns\_data = returns\_data.iloc[:, i].values # Convert to NumPy array

ic = calculate\_ic(stock\_factor\_data, stock\_returns\_data)

ir = calculate\_ir(stock\_returns\_data)

ic\_values.append(ic[0])

ir\_values.append(ir)

return ic\_values, ir\_values

ic\_values, ir\_values = calculate\_ic\_and\_ir\_for\_stocks(factor\_data, returns\_data)

def calculate\_effective\_factor\_proportion(ic\_values, ir\_values, ic\_threshold=0.0005, ir\_threshold=0.005):

num\_stocks = len(ic\_values)

num\_effective\_factors = 0

ic\_values\_flat = np.array(ic\_values).flatten()

ir\_values\_flat = np.array(ir\_values).flatten()

for ic, ir in zip(ic\_values\_flat, ir\_values\_flat):

if abs(ic) > ic\_threshold and ir > ir\_threshold:

num\_effective\_factors += 1

proportion\_effective\_factors = num\_effective\_factors / num\_stocks

return proportion\_effective\_factors

proportion\_effective\_factors = calculate\_effective\_factor\_proportion(ic\_values, ir\_values)

print(f"Of all stocks, there are {proportion\_effective\_factors:.2%} stocks for which the factor is effective (meets IC and IR thresholds）")

**Add the content of the first row to the middle of each row in excel：**

（After calculating the scores of each stock at different times, insert the stock name into excel so that we can check the names of the top 100 stocks and the last 100 stocks.）

import openpyxl

def insert\_rows\_with\_header(input\_file, output\_file):

source\_workbook = openpyxl.load\_workbook(input\_file)

target\_workbook = openpyxl.Workbook()

source\_sheet = source\_workbook.active

target\_sheet = target\_workbook.active

first\_row\_values = [cell.value for cell in source\_sheet[1]]

target\_sheet.append(first\_row\_values)

total\_rows = source\_sheet.max\_row - 1 # Subtract 1 to exclude the header row

for idx, row in enumerate(source\_sheet.iter\_rows(min\_row=2, values\_only=True), start=2):

target\_sheet.append([])

target\_sheet.append(first\_row\_values)

target\_sheet.append(row) #

progress = (idx - 1) / total\_rows \* 100

print(f"Progress: {progress:.2f}% - Processing row {idx} of {total\_rows}")

target\_workbook.save(output\_file)

source\_workbook.close()

target\_workbook.close()

input\_file\_name = "C:/Users/13683/Desktop/最终结果.xlsx"

output\_file\_name = "C:/Users/13683/Desktop/结果1.xlsx"

insert\_rows\_with\_header(input\_file\_name, output\_file\_name)

Process the second excel in the same way as the first excel to delete unlisted stocks：

（According to the guideline that a score of 0 means not listed. Delete unlisted stocks in excel to facilitate all subsequent operations.）

import pandas as pd

def remove\_zero\_values(file\_path):

df = pd.read\_excel(file\_path)

none\_positions = [] # Used to record the position that needs to be set to None

for i in range(1, df.shape[0], 3):

for j in range(1, df.shape[1]):

if df.iloc[i, j] == 0:

df.iat[i, j] = None

df.iat[i-1, j] = None

none\_positions.append((i, j))

none\_positions.append((i-1, j))

print(f"Processing row {i}")

return df, none\_positions

# Read the first Excel file and record the position that needs to be set to None

input\_file\_path = "C:/Users/13683/Desktop/EI新.xlsx"

result\_df, none\_positions = remove\_zero\_values(input\_file\_path)

#output\_file\_path = "C:/Users/13683/Desktop/EI用.xlsx"

#result\_df.to\_excel(output\_file\_path, index=False)

input\_file\_path2 = "C:/Users/13683/Desktop/RSI新.xlsx"

result\_df2 = pd.read\_excel(input\_file\_path2)

for i, j in none\_positions:

result\_df2.iat[i, j] = None

output\_file\_path2 = "C:/Users/13683/Desktop/RSI用.xlsx"

result\_df2.to\_excel(output\_file\_path2, index=False)

input\_file\_path2 = "C:/Users/13683/Desktop/PEG新.xlsx"

result\_df2 = pd.read\_excel(input\_file\_path2)

for i, j in none\_positions:

result\_df2.iat[i, j] = None

output\_file\_path2 = "C:/Users/13683/Desktop/PRG用.xlsx"

result\_df2.to\_excel(output\_file\_path2, index=False)