THE CHINESE UNIVERSITY OF HONG KONG

Cardinality Constrained Portfolio Selection

A thesis statement in partial fulfillment of the requirements of an Engineering Degree in Elite Programme

ZHOU Yunxiu

25/11/2016

Student ID: 1155046976

Cardinality constraint portfolio optimization problem is of popular concern in recent years in the area of portfolio optimization. Transaction costs such as brokerage fees make the diversification proposed by Markowitz not feasible in the real world. Thus there is a genie interest in solving the Cardinality constraint portfolio optimization problem (CCMV).

Table of Contents

Introduction

Portfolio optimization refers to the process of allocating proper weights of various assets classes to be held in a portfolio, in a way such that the portfolio is better than others according to certain criteria. Most of these criteria take both the expected return of the portfolio and the portfolio risk into consideration in determining the optimization objective, although there are differences among various risk measures.
Harry Markowitz, well-known as the father of modern portfolio theory, proposed mean-variance framework in the 1950s.[1] He stated in the model that investors generally want to maximize a portfolio's expected return with respect to different risk level, and the risk is defined to be the standard deviation of the portfolio's rate of return.
Although the variance might not be a perfect measure for the risk (see Markowitz 1959) and there are a lot of revised and advanced models developed during the last decades, Markowitz's Mean-Variance Model laid the foundation of Modern Portfolio Theory (MPT) and nurture the widely-accepted practice in achieving risk-return trade off discussed above in portfolio selection process.
One important implication of Markowitz's framework is that investors always allocate in all risky assets available in the market to fully diversify away risks. This situation, however, is ideal and is only attainable in a frictionless world and can hardly achieved in real life due to the presence of various forms of market friction, such as transaction costs and management fees.[2] Such limitations motivates me to investigate in cardinality constrained mean-variance (CCMV) portfolio selection problem. The cardinality constraint here refers to the limitation in the total number of different assets in the optimal portfolio due to the transaction cost. In consistency with the former research community, the problem of interest of this project is thus to identify a small number of risky assets achieving a performance as close as possible to the market portfolio.
Generalized Markowitz's model can be formulated into the optimization problem as follows(P_1):

$$\min_x x'\boldsymbol{Q}x$$
$$\text{s.t. } \mathbf{r}'\text{x} \geq \overline{r}$$
$$\mathbf{1}'x = 1$$

where $\mathbf{r} = (r_1, r_2, \dots, r_n)'$ is the expected return vector of then risky assets, $\mathbf{Q}$ is the covariance matrix of these n risky securities which is positive semi- definite, $\mathbf{x} = (x_1, x_2, \dots, x_n)'$ is portfolio weight vector, $\overline{\boldsymbol{r}}$ is the targeted return, $\mathbf{1}$ is the n-dimensional all-one vector. In this generalized portfolio optimization problem, the objective is to minimize portfolio risk measured by $x'\boldsymbol{Q}x$, i.e, the variance of the portfolio return. There is no restriction on x_i, implying that shorting is allowed in P.
Putting one more constraint to limit shorting position, the Long-only portfolio optimization problem is, P_2:

$$\min_x x'Qx$$
$$\text{s.t. } \text{r}'\text{x} \geq \overline{r}$$
$$\mathbf{1}'x = 1$$
$$0 \leq x_i \leq 1$$
$$for\ i\ from\ 1\ to\ n$$

By introducing an n-dimensional binary vector $\mathbf{b} = (b_1, b_2, \dots, b_n)'$, cardinality constrained mean-variance portfolio optimization (CCMV) is formulated as:

$$\min_x x'Qx$$
$$\text{s.t. } \text{r}'\text{x} \geq \overline{r}$$
$$\mathbf{1}'x = 1$$
$$0 \leq x_i \leq 1$$
$$\sum_{i=1}^{n} b_i = k$$

[1] Markowitz, H.M. (March 1952).
[2] Gao and Li, 2013.

$$b_i = \begin{cases} 1, if\ x_i > 0 \\ 0, if\ x_i = 0 \end{cases}$$
$$for\ i\ from\ 1\ to\ n$$

Solving problem (CCMV) with targeted return $\bar{r}$ varying from its minimum level (the return level corresponding to the global minimum variance portfolio) to maximum return level of the components yields the efficient frontier in the mean-variance plane under the cardinality constraint.

Review of methodology used in literature

The literature in tackling CCMV in the last two decades can be roughly classified into two categories, exact and heuristic algorithms.

Reviewing the existing literature on cardinality constrained portfolio selection in last two decades; I summarize the two main categories of investigation methods, direct and heuristic algorithms.[3]

Direct methods refer to those who directly tackle CCMV, provide possible solutions from a pure mathematical or theoretical angle and stay largely within optimization regime. For instance, although adopting different relaxation schemes, almost all exact algorithms invoke branch-and-bound algorithms to attain optimality. (See Jiang K, Li D, Gao J, 2014)

Heuristic methods, on the other hand, utilize structural information from the market in the selection process to help choose candidate assets into the portfolio. For example, Jiang proposed a scientific-based heuristic algorithm that integrates factor models in finance, clustering analysis in computer science and mixed integer programming models in operations research to solve CCMV problem. (See Jiang K, Li D, Gao J, 2014)

Heuristics appeal more to me than direct approaches at the first insight, because by incorporating financial analysis with structural information available in the market, heuristics such as Jiang's method does not only reduces computational burden of the original optimization problem, but also yields more meaningful outcomes with an understanding of the financial market.

Methodology and process

In this paper, I will present both the heuristics and direct methods I applied in solving CCMV problem in the following sections.

Investigation of market data

I choose components stocks of Hang Sang Index as the entire asset pool for duplication purpose, although smaller in asset size, Hang Sang Index components are widely agreed as the benchmark of Hong Kong stock market.

I got prices of Hang Sang Index component stocks from Yahoo Finance since 2000 and computed the expected annual return and volatility of each of 49 stocks. Here is a need to clarify the reason for 49 rather than 50 stocks finalized in this project: to compute covariance matrix among risky assets, complete trading information over a continuous time window is essential. Additionally, longer the time period, more conceivable the covariance matrix output is and more meaningful is the result. However, the real situation is far from ideal. In special, 1113.HK equity possess a unusually short time period available for available price information. There are three common practices in solving such data problems, either choose the longest time window for which price information on all 50 stocks are available, or auto fill the missed elements in covariance matrix by linear regression or filter individual assets to get more consistent

[3] Jiang, Li and Gao, 2014.

data points. For the sake of a sufficient length of observations, I choose the third solution, i.e. not to include 1113.HK in the following analysis.

Symbol	Return	Risk
0001.HK	9.3%	23.8%
0002.HK	7.3%	11.1%
0003.HK	9.9%	17.4%
0004.HK	9.2%	29.2%
0005.HK	-0.3%	23.1%
0006.HK	12.1%	16.0%
0011.HK	10.9%	18.2%
0012.HK	8.2%	25.6%
0016.HK	2.2%	26.4%
0017.HK	3.6%	37.1%
0019.HK	0.1%	20.3%
0023.HK	5.6%	24.7%
0027.HK	47.7%	50.9%
0066.HK	8.7%	14.4%
0083.HK	3.2%	27.4%
0101.HK	-5.7%	29.4%
0135.HK	-2.5%	30.8%
0144.HK	1.0%	25.8%
0151.HK	-2.6%	24.7%
0267.HK	-1.1%	32.8%
0293.HK	-5.0%	26.0%
0322.HK	-7.7%	31.3%
0386.HK	6.8%	25.4%
0388.HK	11.8%	35.7%
0494.HK	-14.7%	32.1%
0688.HK	14.5%	35.6%
0700.HK	39.5%	30.4%
0762.HK	2.6%	29.9%
0823.HK	20.5%	18.3%
0836.HK	5.0%	28.5%
0857.HK	-3.0%	25.9%
0883.HK	-0.4%	28.4%
0939.HK	3.0%	23.9%
0941.HK	6.4%	17.3%
0992.HK	8.2%	34.3%
1038.HK	17.3%	17.2%
1044.HK	0.9%	20.2%
1088.HK	-5.4%	28.1%
1109.HK	12.9%	37.3%
1299.HK	16.6%	20.9%
1398.HK	3.9%	29.2%
1880.HK	-11.1%	29.8%
1928.HK	27.6%	39.8%
2318.HK	13.8%	49.6%
2319.HK	24.1%	62.2%
2388.HK	10.3%	23.4%
2628.HK	-2.3%	31.1%
3328.HK	3.7%	29.9%
3988.HK	4.5%	27.2%

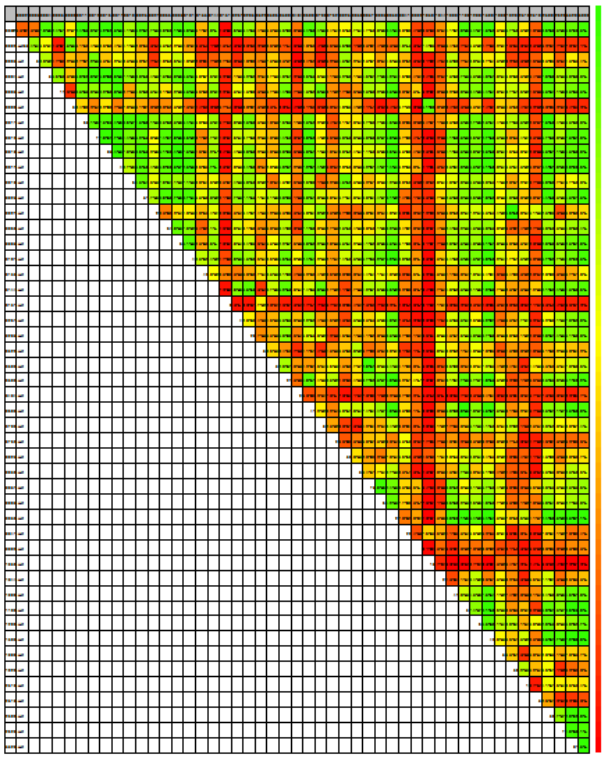


Figure 1 Market structure

Figure 1 shows the expected return and volatility table of each of 49 stocks over the time period. Correlation relationship among each stock is also shown. The color bar on the right corner represents the magnitude to which the corresponding stocks are related. Red color indicates relatively negative correlation, while green represents rather positive correlation.

Heuristics

With this first understanding of the market, I start by following Jiang's method to solve CCMV problem and then extend the scope by applying different factor model and various clustering algorithms.

I follow Jiang's major steps in dealing with CCMV, i.e. i) characterize different risky assets by factor models using market data, ii) cluster similar risky assets into groups in accordance with their loading coefficients in the factor model, and iii) select representative(s) within individual groups according to different criteria to form a portfolio.

Factor model

Introduction

Luenberger introduced factor models in his famous book, *investment science*, to characterize risky assets.[4] Under the arbitrage pricing theory (APT), the market movement is driven by a set of different factors. In addition, risky asset returns are believed to be solely pinned down by a linear combination of these factors. Thus, loading coefficients of these factors could capture almost all the characteristics of each asset. In other words, market performance of that asset could be largely determined once we derive the loading coefficients.

[4] Luenberger,1997.

In this thesis, I calculate all the coefficients in various factor models introduced in the following by linear regression which is feasible and logical.

Factor model across industry

With an initial grasp of the data source and understanding about factor model, I start to examine the factor model across industries proposed in Jiang's article. Specifically, Jiang proposed to use the Hang Seng Composite Industrial Index as factors in their analysis, including Hang Seng Comp. Energy Index, Hang Seng Comp. Materials Index, Hang Seng Comp. Industrial Goods Index, Hang Seng Comp.Consumer Goods Index, Hang Seng Comp. Services Index,Hang Seng Comp. Telecommunications Index, Hang Seng Comp. Utilities Index, Hang Seng Comp. Financials Index, Hang Seng Comp. Properties & Construction Index, Hang Seng Comp. Information Technology Index, and Hang Seng Comp. conglomerates index, the total 11 factors.

This model is believed to work well with components stocks of Hang Sang Index because of its easiness and meaningfulness. First, , data source of these market indexes are credible, authorized and easy to obtain ; second, each of the 11 indexes serves as a typical representation of the corresponding market industry, and such grouping across industries is definitely meaningful to consider. Based on this reasons, I realize this factor model on the data set and present the results below.

	intercept	HSCIE	HSCIM	HSCIIG	HSCICG	HSCIS	HSCIT	HSCIU	HSCIF	HSCIPC	HSCIIT	HSCIC
0001_HKW	0.00280539	-0.173410575	0.027975366	-0.038763195	-0.079299651	0.076551875	-0.039381577	-0.015679225	0.170546978	0.176970668	-0.112011009	1.033062355
0002_HKW	-0.000512195	0.003886528	-0.052383626	-0.069401313	-0.179620575	-0.000588051	0.147554758	0.579760105	0.051426572	0.071097542	-0.032802855	0.149037036
0003_HKW	-0.000913946	0.027311403	-0.121341106	-0.021809377	-0.101949764	0.020841268	0.08418749	0.69074844	0.156566272	-0.000257485	-0.038175704	0.135500015
0004_HKW	-0.001077679	0.072371848	-0.041612894	-0.235016424	0.005848378	0.155144896	0.154930805	0.428159702	-0.175021525	0.829197007	0.000850592	-0.050580701
0005_HKW	-0.002165349	-0.026865938	-0.119468566	-0.099563982	-0.132818517	0.129720038	0.040510033	0.033357557	0.835568957	-0.053864609	0.025327504	0.036045313
0006_HKW	0.000264588	-0.127308818	0.053928415	-0.189238275	0.176492695	-0.148910512	0.127777774	0.784540523	0.019313331	0.046294784	-0.147778222	0.340298187
0011_HKW	0.000418255	-0.065907216	-0.159406943	-0.04531036	-0.055489332	0.090125733	0.051561146	0.158911776	0.484333751	-0.069161018	-0.007145648	0.348109923
0012_HKW	0.002315546	0.101529071	-3.95E-05	-0.068286352	-0.23088963	0.01214863	0.096076096	0.053247279	-0.166906172	0.87730133	-0.086277455	0.303829344
0016_HKW	-0.001082159	-0.115134829	0.014524403	-0.17855566	-0.034320638	0.080699478	0.00200733	0.180919602	0.090888855	0.688377842	-0.045767171	0.151182955
0017_HKW	-0.002743878	-0.005007539	-0.026182425	-0.178385623	-0.334940835	0.135636034	0.064729831	0.111691222	0.144598366	0.890908234	0.143014442	0.063313869
0019_HKW	-0.000714024	-0.034961592	-0.018931781	-0.099335478	0.069661071	0.009276619	0.109927397	0.097474905	0.06922989	0.163285789	-0.05683562	0.45704398
0023_HKW	0.00020923	0.155509015	-0.038699893	-0.19005073	0.154279865	0.026699002	0.019522015	0.196472459	0.33783242	0.158088726	0.008658562	0.156009548
0027_HKW	0.003704172	0.247654949	-0.166191105	-0.104321515	-0.22116182	1.853930606	-0.275479005	-0.027593118	0.116488061	-0.130076692	0.029610521	-0.209100451
0066_HKW	0.000993306	-0.12096619	0.0836649	0.005632612	0.232749429	-0.053384215	0.113094476	0.061230127	0.140265091	-0.0460304	-0.023773164	0.32937143
0083_HKW	-0.001041785	0.067646172	-0.03261529	-0.234791922	0.047745466	-0.155704697	-0.010160783	0.181421304	-0.116182979	0.98145213	0.06494496	0.223907989
0101_HKW	-0.002324932	0.115442328	-0.148155514	-0.180644834	0.227606294	0.005299311	0.123948686	0.369111928	0.005994226	0.552065507	-0.076957921	0.124567786
0135_HKW	-0.003074062	0.721408903	-0.183387656	0.242265168	-0.252479016	-0.009461757	-0.093691107	0.436556339	0.053140155	0.148002063	0.023957654	0.000910765
0144_HKW	-0.000539012	0.017505666	-0.019060913	0.002335322	0.500358023	-0.08478226	0.193958515	0.162615762	0.493874332	-0.219994477	0.142229359	0.074432826
0151_HKW	-0.001919551	-0.337903476	0.018704223	-0.64663018	2.061619898	-0.169958186	0.047963846	0.382459565	0.336851667	-0.215702412	-0.292564641	0.012759639
0267_HKW	0.001977176	0.075923176	0.360423745	0.0707516	-0.059130446	-0.129196882	0.036486436	-0.056341799	0.115953529	-0.022709251	-0.138649063	0.707437872
0293_HKW	-0.003257417	-0.394121973	-0.024827331	0.014633377	0.138821983	0.041367212	-0.03737648	0.090654413	0.811054635	-0.059470704	0.243853235	0.265250772
0322_HKW	-0.001635953	0.136026078	0.054715127	-0.481131087	1.232393759	0.032899079	0.113639678	0.2123268	0.418475369	-0.518207883	-0.065062436	-0.055333427
0386_HKW	0.001661895	0.822971454	0.063147185	-0.150307742	-0.183450354	-0.034026587	-0.050232949	0.140565544	0.264249403	0.029560957	-0.017784287	0.224713296
0388_HKW	0.004265966	-0.095110358	0.580107092	0.096934096	0.031481669	-0.036380773	0.049813359	0.043837855	0.825171746	-0.240660402	-0.060896563	0.019420978
0494_HKW	-0.002009891	0.301656481	-0.034843877	-0.360042156	1.497758126	-0.181406303	-0.118909379	-0.407047207	-0.470220701	0.402793433	0.075799193	-0.012270121
0688_HKW	0.002330371	-0.021465799	0.042825619	-0.026419291	0.124764017	-0.128404565	0.057436787	-0.405318208	-0.144777905	1.746047865	0.11643251	-0.173167019
0700_HKW	9.13E-05	0.005403229	-0.117902607	-0.05254078	-0.134756911	-0.019227408	0.02492486	-0.059597321	0.180485829	-0.064253424	1.3003719	-0.031590325
0762_HKW	0.000176945	0.107698824	0.098541698	0.092998784	0.008484013	0.03821698	0.953441307	-0.227258089	-0.103562343	0.166294841	-0.039164871	-0.003959855
0823_HKW	0.000767124	-0.003621746	-0.108234107	-0.256526172	0.02884137	-0.043415369	0.083033758	0.429296578	0.131944464	0.290807342	-0.004861646	0.060440191
0836_HKW	0.000219109	-0.125579422	0.069365492	-0.590131289	0.929609096	-0.07948763	0.035538241	1.461392239	-0.219786865	0.575061969	-0.194283537	-0.325329988
0857_HKW	0.000412059	1.154897491	-0.071580641	-0.056747661	-0.113256581	-0.062385198	0.095235876	0.193782706	0.06354891	-0.051655649	-0.014225679	-0.067267234
0883_HKW	0.000949371	1.335263475	-0.215769793	0.09132787	0.112516116	-0.024471776	0.099214977	-0.186957785	-0.331300685	0.026744005	0.08288559	-0.029879333
0939_HKW	-0.000394056	0.112143204	-0.015036961	-0.043610829	0.040886334	-0.084608034	0.026068782	-0.061921749	1.130745797	0.028221943	-0.010179366	-0.035950826
0941_HKW	-0.000120617	-0.013979478	-0.024314062	-0.003510143	-0.026563446	-0.01016885	1.065460086	0.038474526	-0.043895844	-0.00096196	0.017149967	0.019955421
0992_HKW	0.001756246	0.017052091	0.414916125	0.006134133	0.119624354	0.024784626	0.000894879	-0.077823506	-0.312919261	0.50383862	0.329329623	-0.169048659
1038_HKW	0.000990232	-0.176595562	0.050929947	-0.079905745	-0.070687845	-0.00767825	0.192245513	0.66434861	0.01615328	-0.142424496	0.066069615	0.071269489
1044_HKW	-0.001205656	-0.081944528	-0.482458935	0.051460781	0.781408359	0.071109719	0.255868059	0.057177943	0.062565773	0.197282069	0.117824027	0.098263882
1088_HKW	-0.001038105	0.60059203	0.233972687	-0.01546936	0.134100097	0.076200733	-0.121372294	-0.1483668	0.605455236	-0.142370949	-0.215121753	-0.14781678
1109_HKW	0.002711426	0.081183841	0.05458732	-0.303670156	0.0389286	-0.065578912	-0.014086087	-0.218381765	-0.117227257	1.938485389	0.002730431	-0.042169414
1299_HKW	0.002465048	0.030997887	-0.173481363	0.09356787	-0.100929058	0.155004304	-0.005984191	0.37955732	0.511664824	-0.020593667	-0.009665361	0.116774626
1398_HKW	-0.000846014	0.051295366	-0.056020112	-0.061244142	0.259421007	-0.135330868	0.005397507	-0.060946343	1.27857296	0.042899961	0.049376297	-0.279896642
1880_HKW	-0.004224542	0.251775813	-0.470362745	-0.005855163	1.36436904	0.088681713	0.376115283	0.050079186	0.206161389	0.226657155	0.029193095	-0.393729829
1928_HKW	0.003277978	0.27478797	-0.228869493	-0.111588514	-0.431180367	1.750527251	-0.058971563	0.191939158	-0.234109862	0.216821122	-0.094056003	-0.311879292
2318_HKW	0.003071003	-0.039178063	-0.002027642	0.134218379	0.192004868	-0.058653156	-0.196660092	-0.386871372	1.53442414	0.086402254	-0.129025307	0.07460975
2319_HKW	0.002706894	-0.054882694	-0.042282423	0.115239155	0.835832919	0.209018523	0.271967194	0.103578685	-0.243148271	-0.20605609	-0.112343775	0.23596834
2388_HKW	0.000745988	0.087938232	-0.127123599	-0.213251838	0.346187582	0.17896141	-0.108591271	0.099835714	0.568162256	0.018616578	-0.035707647	0.07407948
2628_HKW	0.0009141	0.099811427	0.120575064	0.199521078	-0.162540261	-0.125021903	-0.112562942	-0.146888666	1.199524787	0.069435295	0.009738246	0.15633476
3328_HKW	-0.000221908	-0.091609296	0.040831183	0.034288918	-0.304622514	-0.020699803	-0.060462813	-0.073476277	1.575040223	0.097588363	-0.039334859	-0.186939825
3988_HKW	0.00018562	-0.019787686	0.023478799	-0.016262399	0.140138443	-0.083088621	0.095541617	-0.029679561	1.223414191	-0.060059788	-0.013386813	-0.037745364

Table 1 regression coefficients of industrial factors

The table presents the result of applying linear regression on observations over the 15 years time window of stock returns with respect to industrial factor returns for each stock. Column 1 is the intercept, or alpha, and column 2 to 12 shows the loading coefficients of each factors. By applying the factor model across industries, each of 49 risky assets is transformed to an 12 = 11 + 1 dimensional vector feature space with the intercept and 11 factor loadings as its point coordinates.
As XXX points out, other than statistical factors, such as industrial indexes presented above, there are two more kinds of factors worth to consider, namely, macroeconomic factors, such as Gross National Product(GNP)and unemployment rate; and traditional value factors [5], as known as fundamental factors, such as Price-Earnings ratio, Price-Sales ratio, Price-Book ratio and Dividend Yield.
Hence, I incorporate Fama-French 3 factor model including a momentum factor to develop a more comprehensive and reasonable characterization of risky assets.

Fama-French factor model

Eugene Fama and Kenneth French proposed the Fama–French three-factor model in 1993 to describe stock returns.
Compared with the traditional capital asset pricing model (CAPM), which uses only one variable to describe the returns of individual asset or portfolio with those of the entire market, Fama and French use three. They added another two factors to the traditional model to reflect portfolio's return with respect to SMB and HML. (See Fama and French, 1993)
The Carhart four-factor model is an extension of the Fama–French three-factor model including a momentum factor, also known in the industry as the MOM factor (monthly momentum).[6] Momentum of a stock is defined as the tendency for its price to continue rising if it is going up and to continue declining if it is going down.
There are a large number of studies on the explanatory ability of these four factors in Fama-French model in different market regions(Global, North America, Europe, Japan and Asia Pacific). Based on the decent results of such studies, Fama-French model are believed to have strong power in explaining stock returns. Thus, I commit to realize this model on the data set.
I load factor data from the Kenneth R French: Data Library[7], compare and analyze the relationships between returns of Hang Sang Index component stocks and factor returns from both global market and from Asia Pacific excluding Japan market. Theoretically, Asian Pacific data fits my project better based on the fact that Hong Kong is a featured Asian Pacific financial center. In addition, Griffin points out that the Fama - French factors are country specific and concludes that the local factors provide a better explanation of time-series variation in stock returns than the global factors in his 2002 paper. To test the explanatory of Fama and French factors on Hang Seng index component stocks, I conduct statistical testing in the hoping of deriving the same result as Griffin. As expected, statistical report of regression further supports the notion in applying local data rather than global one by showing a higher R square (0.61) than that (0.29) from global market, which implicates that Asia Pacific excluding Japan market factors explain around 60% of historical performance of Hang Sang Index component stocks whereas global factors only explain less than 30%. Thus, Asia Pacific excluding Japan market factors data is used in the following analysis.
Similar to factor model across industries, Fama–French factor model transform each of these 49 risky assets to a 5 (4 + 1) -dimensional vector with respect to the intercept and 4 factor loadings.
The result of linear regression on stock returns with respect to four factors in Carhart four-factor model is presented. Note however, the table has 6 columns, including an extra column signifying risk free rate, which is needed to compute excess return. Although assuming risk free rate to be 0 does not affect the relative magnitude of loading coefficients, I include it in the analysis of Fama-French factor model for the purpose of completeness and logicality.

---

[5] It refers to the fundamental factors described in the CSFB Alpha Factor Framework.
[6] Carhart, 1997.
[7] See data source 1.

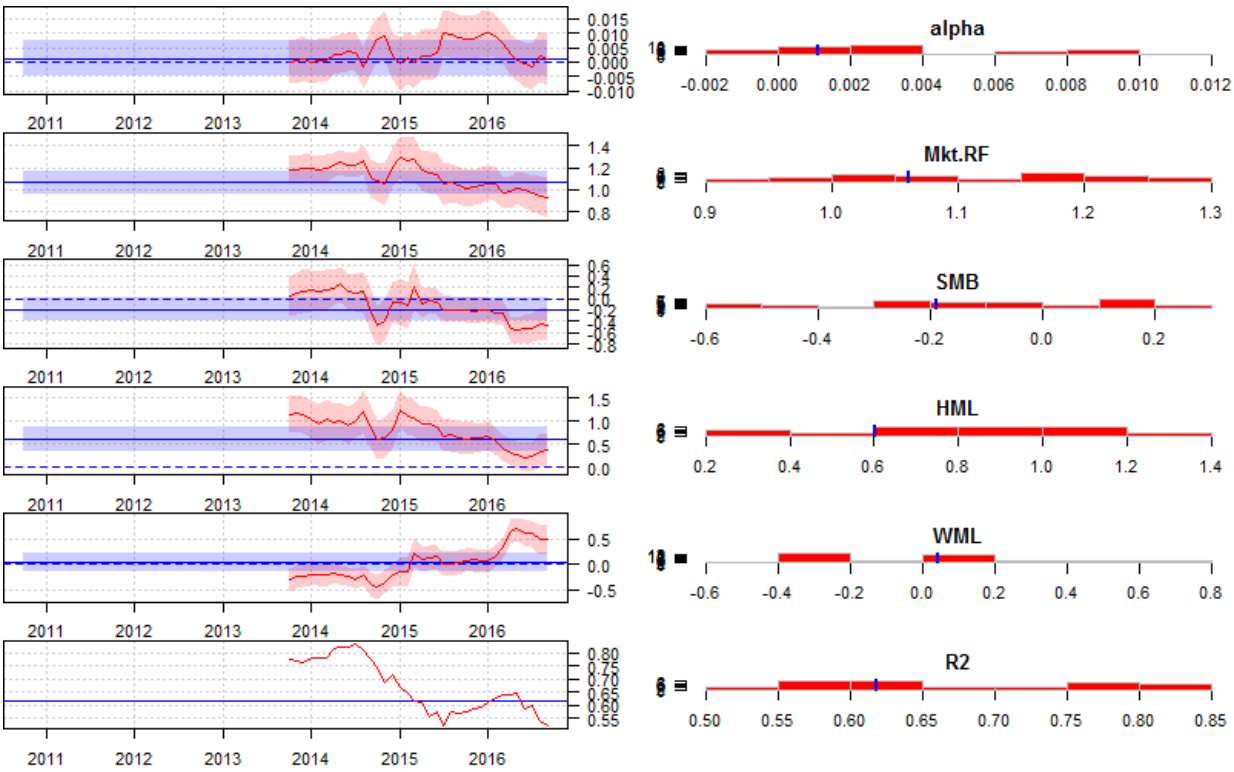


Figure 2 Statistical testing in Asia Pacific excluding Japan market

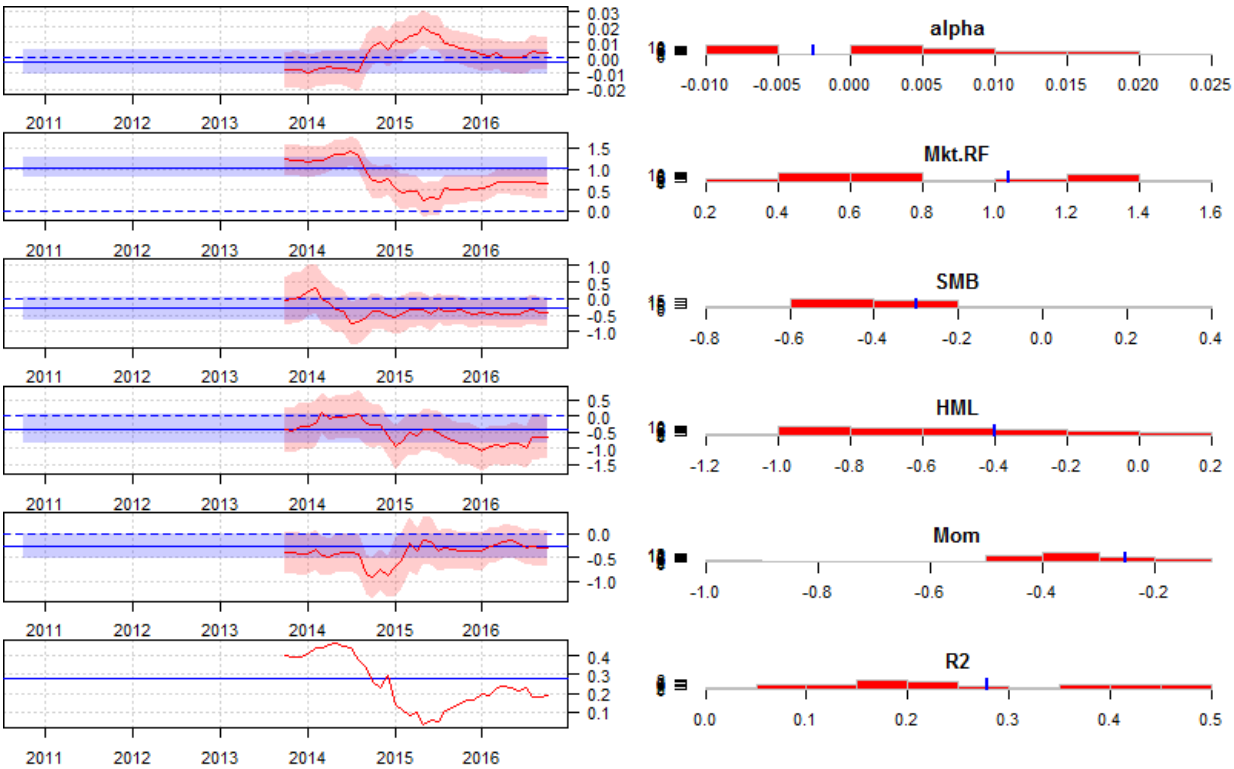


Figure 3 Statistical testing in global market

	alpha	Mkt.RF	SMB	HML	RF	WML
0001.HK	-0.00756	1.205127	-0.31902	0.871204233	0.041904	0.678549
0002.HK	0.0026	0.132299	-0.06166	0.123003559	0.136637	0.044677
0003.HK	0.001653	0.437313	-0.03272	0.456028277	0.049889	0.203604
0004.HK	-0.0026	1.285279	-0.49307	1.104412822	-0.26856	0.634605
0005.HK	-0.00554	0.774743	0.10272	0.363527833	-0.2419	0.44955
0006.HK	0.004098	0.147135	-0.28245	-0.03864989	0.153335	0.070674
0011.HK	-0.00174	0.736789	-0.1599	0.512768	-0.08815	0.484398
0012.HK	-0.0093	1.251505	-0.31561	1.436122797	-0.14629	0.651374
0016.HK	-0.01068	1.28951	-0.52289	1.305734744	-0.04134	0.698076
0017.HK	-0.01185	1.802074	-0.09975	1.97627423	-0.44935	0.689231
0019.HK	-0.0017	0.895887	-0.20153	0.775346795	-0.27174	0.594689
0023.HK	-0.00391	1.140767	-0.00026	0.555289095	-0.07553	0.576377
0027.HK	0.020407	1.702097	0.554019	0.53270429	0.057159	0.286859
0066.HK	-0.00024	0.785553	-0.27003	0.568420659	-0.07068	0.527629
0083.HK	-0.00028	1.496222	-0.35717	1.303607924	-0.504	0.634673
0101.HK	-0.00353	1.092324	-0.58576	0.969312796	-0.04577	0.479788
0135.HK	0.009193	1.049426	0.594554	0.128427486	0.030552	0.281064
0144.HK	-0.00043	1.103851	0.078888	0.636227369	-0.07924	0.482266
0151.HK	0.006704	0.380132	-0.41709	-0.24682322	0.063381	0.111668
0267.HK	-0.00461	1.482569	0.541803	0.419842175	-0.38747	0.54264
0293.HK	-0.00527	0.919047	-0.05761	0.753528754	-0.29987	0.455923
0322.HK	0.009467	0.566624	0.038794	1.120322329	-0.03132	0.149352
0386.HK	0.003788	0.922779	0.066959	0.096940594	0.242193	0.29553
0388.HK	0.010742	1.387892	0.312557	0.68694357	-0.2226	0.449939
0494.HK	0.007375	0.757857	-0.07292	0.103239732	-0.44447	0.241201
0688.HK	0.004454	1.317748	-0.02586	0.912761737	0.283291	0.371431
0700.HK	0.033682	1.134236	-0.00944	0.099316123	0.064001	0.3441
0762.HK	-0.00261	0.883034	-0.24682	0.138594922	-0.27761	0.25787
0823.HK	0.011203	0.458018	-0.52526	0.120286081	0.407254	0.267456
0836.HK	0.002574	0.844133	-0.37191	-0.01934385	0.563245	0.26373
0857.HK	0.006537	0.97752	0.091177	-0.13055679	-0.04187	0.330975
0883.HK	0.003416	1.184808	-0.03983	-0.26999045	0.108219	0.518346
0939.HK	0.002591	0.982069	-0.11784	0.239646347	0.267658	0.45879
0941.HK	0.001514	0.769705	-0.15433	-0.17732624	0.166166	0.262296
0992.HK	-0.0003	1.146634	0.831257	0.180580318	-0.32999	0.34784
1038.HK	0.003466	0.260223	-0.11771	0.394956974	0.199823	0.086662
1044.HK	0.012927	0.671311	0.192481	0.356005824	0.125677	0.225889
1088.HK	0.002066	1.177754	0.004169	0.110017043	0.038955	0.503772
1109.HK	0.007422	1.403384	-0.27664	1.014148098	0.166031	0.313031
1299.HK	0.010131	0.90941	0.03344	0.090836996	0.04901	0.569379
1398.HK	0.002079	1.082645	-0.31657	0.138522573	0.071281	0.479212
1880.HK	0.008544	1.021166	0.566655	0.246354626	0.047121	0.27502
1928.HK	0.029925	1.155495	-0.59515	-0.77875718	-0.65509	0.404168
2318.HK	0.013825	1.330427	-0.71224	-0.36451132	-0.13884	0.339355
2319.HK	0.014596	1.149514	-0.21213	0.407845782	-0.23284	0.196905
2388.HK	-0.00137	1.064657	0.020469	0.583729148	0.001764	0.588225
2628.HK	0.000214	1.057428	-0.23685	0.229815226	0.292212	0.342634
3328.HK	0.001197	1.233896	-0.24107	0.392376047	0.272915	0.45148
3988.HK	-0.00112	0.979807	-0.0806	0.294393158	0.137534	0.472078

Table 2 Loading coefficients of Fama French factors

Clustering algorithm

Up to this point, each risky asset has been characterized into a feature vector by factor model, next task is to cluster the entire asset pools into different groups based on their similarity. Degree of similarity between two items is defined as the negative relationship between the norm of the difference between this pair of feature vectors in Jiang's paper.[8]The smaller the norm of the difference between a pair of feature vectors, the higher degree of similarity of these two assets. (See Jiang, 2014)

Determination of clustering algorithms

There are three most common clustering algorithms currently used in practice, namely partitioning clustering, density-based clustering and hierarchical clustering (see, e.g., Han et al. (2011)).

Density-based clustering is the first to be eliminated from consideration because its unrealistic assumption on a uniform density distribution within genuine clusters.
Next, I experimented both hierarchical clustering and partitioning clustering algorithm on Hang Seng Index component stocks to observe trend of the optimal number of groups across history and also as a comparison among different clustering algorithms. Optimal number of clusters is generally difficult to determine in clustering algorithms, as different data points possess different structure across time, even tiny changes in the correlation matrix may alter the cluster result a lot. It is an inevitable result since most partitioning algorithms such as optimization routine aim to maximize inter-cluster dissimilarity and intra-cluster similarity. Thus, stability and consistency of number of clusters becomes significantly important in evaluating various clustering methods.

Here are the results of comparing the following two methods:

- Minimum number of clusters that explain at least 90% of variance
- Hierarchical clustering tree cut at 1/3 height

[8] Jiang, Li and Gao, 2014.

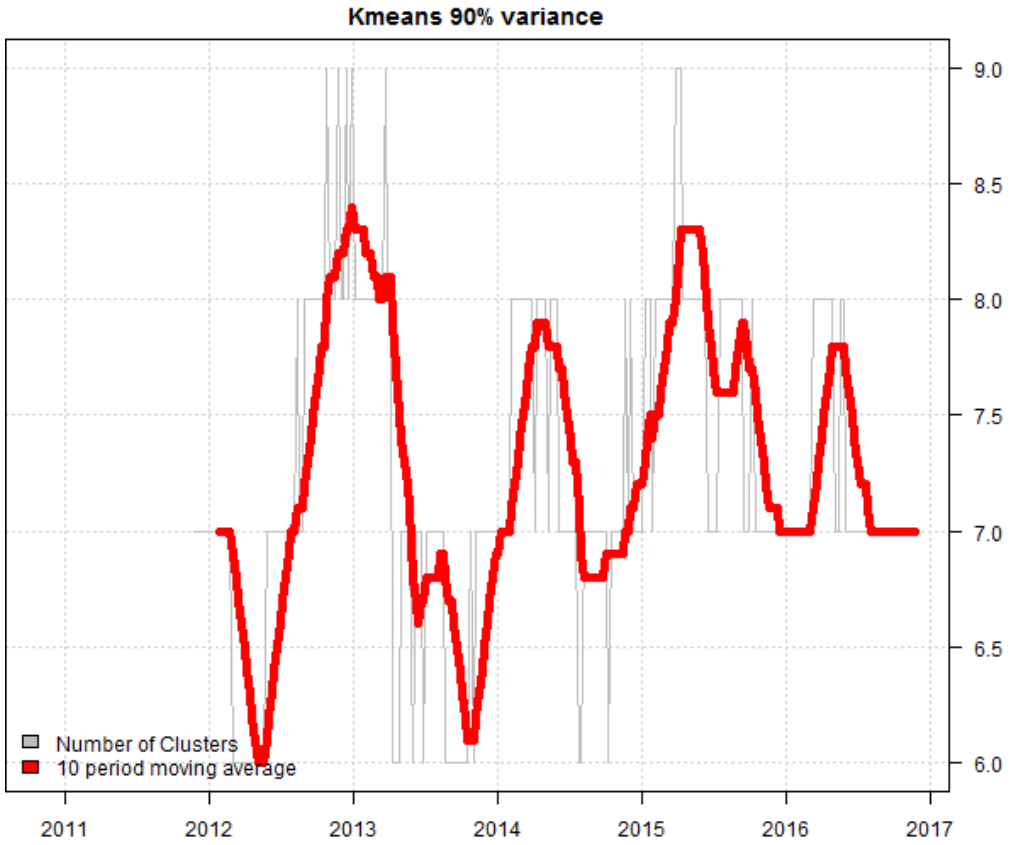


Figure 4 Number of clusters by K-means 90% correlation

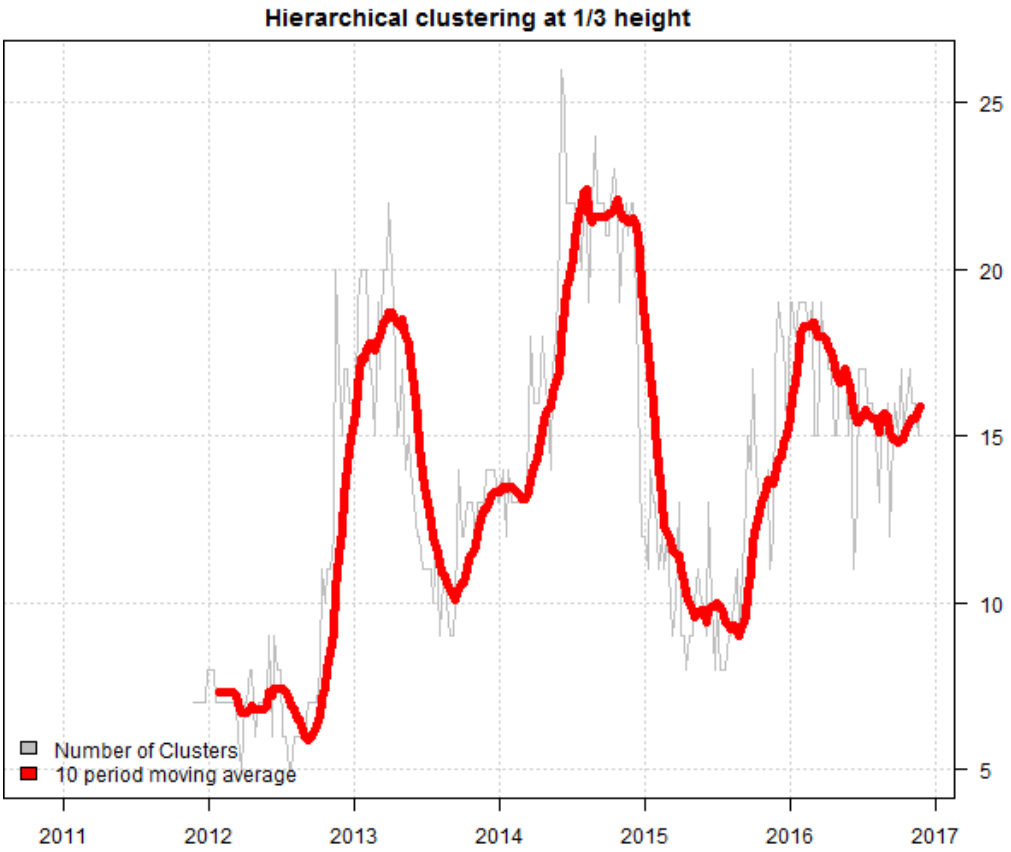


Figure5. Number of clusters by hierarchical clustering at 1/3 height

The figures show that with K-means analysis, the most typical and reasonable number of groups among Hang Seng Index component stocks is in the range of seven to nine. Hierarchical clustering, on the other hand, generates more unstable clusters across time, with the minimum at around 7 and hitting a maximum of nearly 25 groups, which is much more than rationale would tend to dictate. Such inconsistency and illegitimacy makes hierarchical clustering less favorable for this project.

As a result, taking both the benefits and drawbacks of all these three clustering algorithms into consideration, combining with the fact that dimensions of the object in this project is relatively low, I finally chose to mainly focus on partitioning clustering algorithm

Partitioning clustering

Introduction of partitioning clustering

Partitioning clustering, also known as flat or unnested clustering[9], decomposes a data set into a set of disjoint clusters. Given a data set of N points, a partitioning method constructs K ($N \geq K$) partitions of the data, with each partition representing a cluster. K-means clustering, quality threshold clustering and expectation maximization clustering etc. are representative algorithms for partitioned data clustering. (See Jing, 2011)

[9] Jing, 2011.

Examples

K-means

The most common and well-known partitioning method is the K-means cluster analysis. Conceptually, it can be formulated as following steps:

1. Selects K observations randomly and set them to be the initial centroids
2. Assigns each data point to its closest centroid
3. Recalculates the centroids as the average of all data points in a cluster
4. Assigns data points to their closest centroids
5. Continues steps 3 and 4 until observations are not reassigned or the maximum number of iterations is reached.

FTCA

Another clustering algorithm, the Fast Threshold Clustering Algorithm (FTCA) created by David Varadi, attracts me a lot for some of its desirable properties that traditional clustering algorithms do not have. Specifically, FTCA uses the average correlation of each asset to all other assets as an indicator of how closely or distantly related an asset is to the universe of assets chosen. (Varadi, 2016) the graph below vividly presents the logic of how FTCA creates clusters:

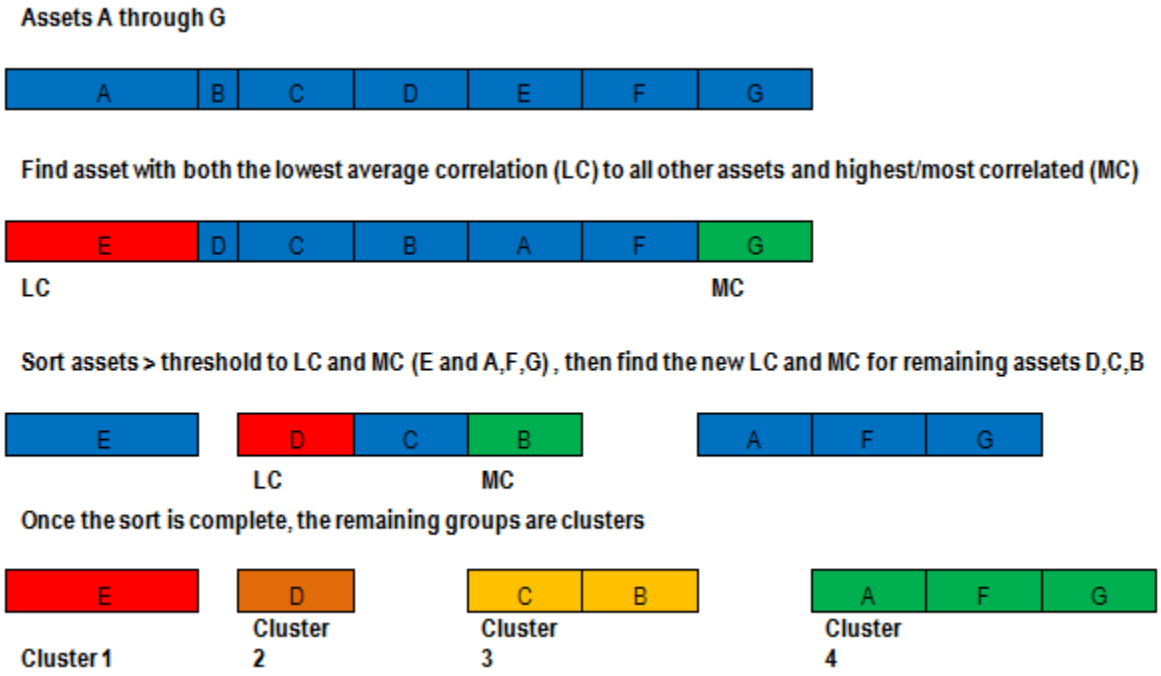


Figure 5 FTCA clustering process

For better understand, I also state the pseudo code for FTCA[10] as follows:

While there are assets that have not been assigned to a cluster

If only one asset remaining then

Add a new cluster

[10]Varadi, d. (2016). Fast Threshold Clustering Algorithm (FTCA). [online] CSSA. Available at: https://cssanalytics.wordpress.com/2013/11/26/fast-threshold-clustering-algorithm-ftca/ [Accessed 25 Nov. 2016].

Only member is the remaining asset

Else

Find the asset with the Highest Average Correlation (HC) to all assets not yet been assigned to a Cluster

Find the asset with the Lowest Average Correlation (LC) to all assets not yet assigned to a Cluster

If Correlation between HC and LC > Threshold

Add a new Cluster made of HC and LC

Add to Cluster all other assets that have yet been assigned to a Cluster and have an Average Correlation to HC and LC > Threshold

Else

Add a Cluster made of HC

Add to Cluster all other assets that have yet been assigned to a Cluster and have a Correlation to HC > Threshold

Add a Cluster made of LC

Add to Cluster all other assets that have yet been assigned to a Cluster and have Correlation to LC > Threshold

End if

End if
End While

Now, I present the results of applying both K-means clustering and FTCA clustering methods on the

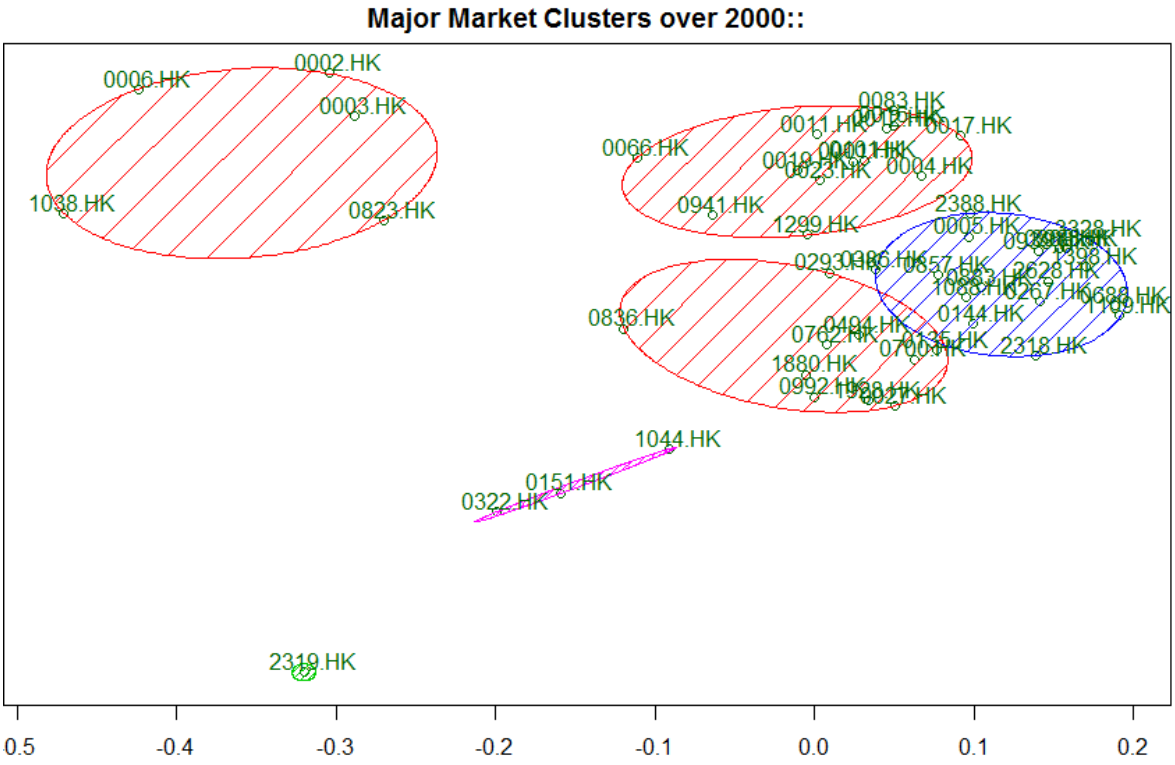


Figure 6 K-means market clusters

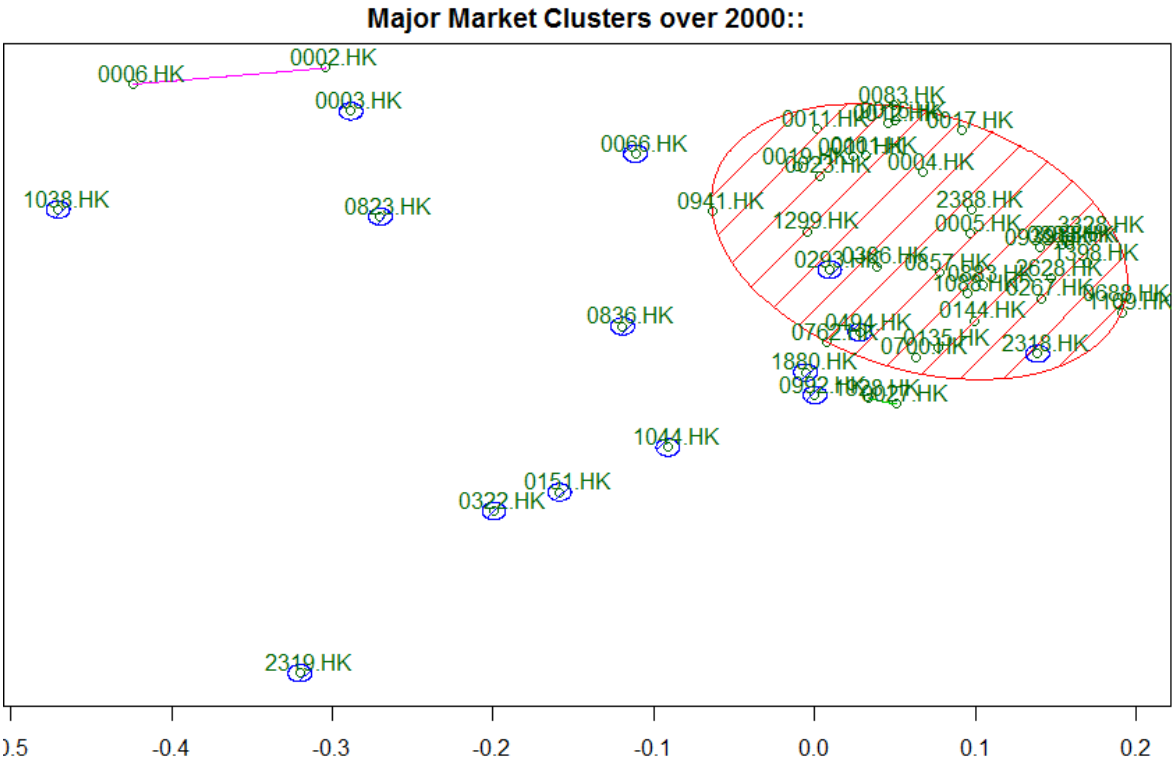
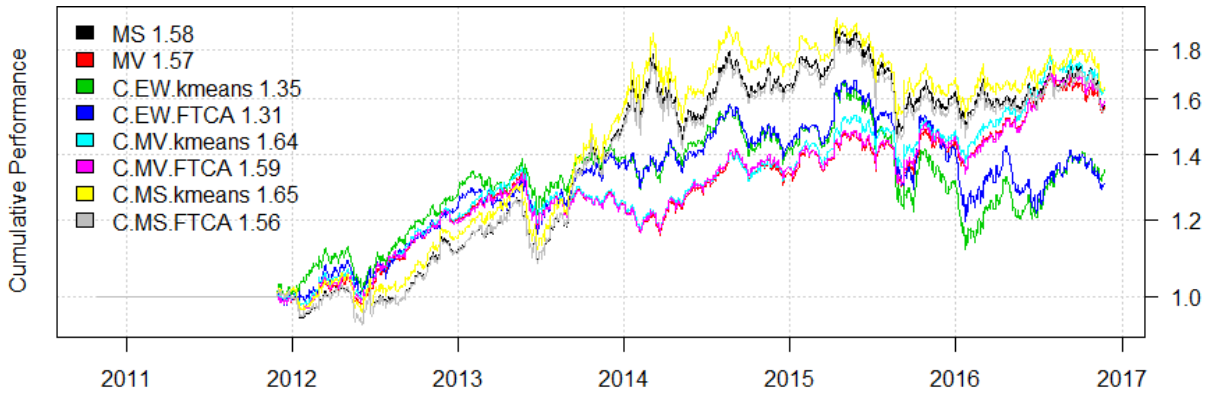


Figure 7 K-means by FTCA



System	MS	MV	C.EW.kmeans	C.EW.FTCA	C.MV.kmeans	C.MV.FTCA	C.MS.kmeans	C.MS.FTCA
Period	Oct2010 - Nov2016	Oct2010 - Nov2016	Oct2010 - Nov2016	Oct2010 - Nov2016	Oct2010 - Nov2016	Oct2010 - Nov2016	Oct2010 - Nov2016	Oct2010 - Nov2016
Cagr	7.79	7.68	5.08	4.51	8.45	7.93	8.54	7.61
Sharpe	0.55	0.77	0.42	0.38	0.85	0.79	0.6	0.54
DVR	0.45	0.7	0.21	0.25	0.78	0.73	0.48	0.44
Volatility	15.51	10.14	14.08	14.15	10.04	10.1	15.46	15.45
MaxDD	-22.46	-14.02	-32.59	-28.49	-14.32	-13.17	-21.42	-22.57
AvgDD	-3.75	-2.18	-2.65	-2.48	-1.8	-2.03	-3.17	-3.4
VaR	-1.64	-1	-1.46	-1.36	-0.99	-0.99	-1.65	-1.63
CVaR	-2.39	-1.55	-2.1	-2.1	-1.55	-1.54	-2.38	-2.39
Exposure	83.54	83.54	83.54	83.54	83.54	83.54	83.54	83.54

Figure 8 Cumulative performance time trend plot

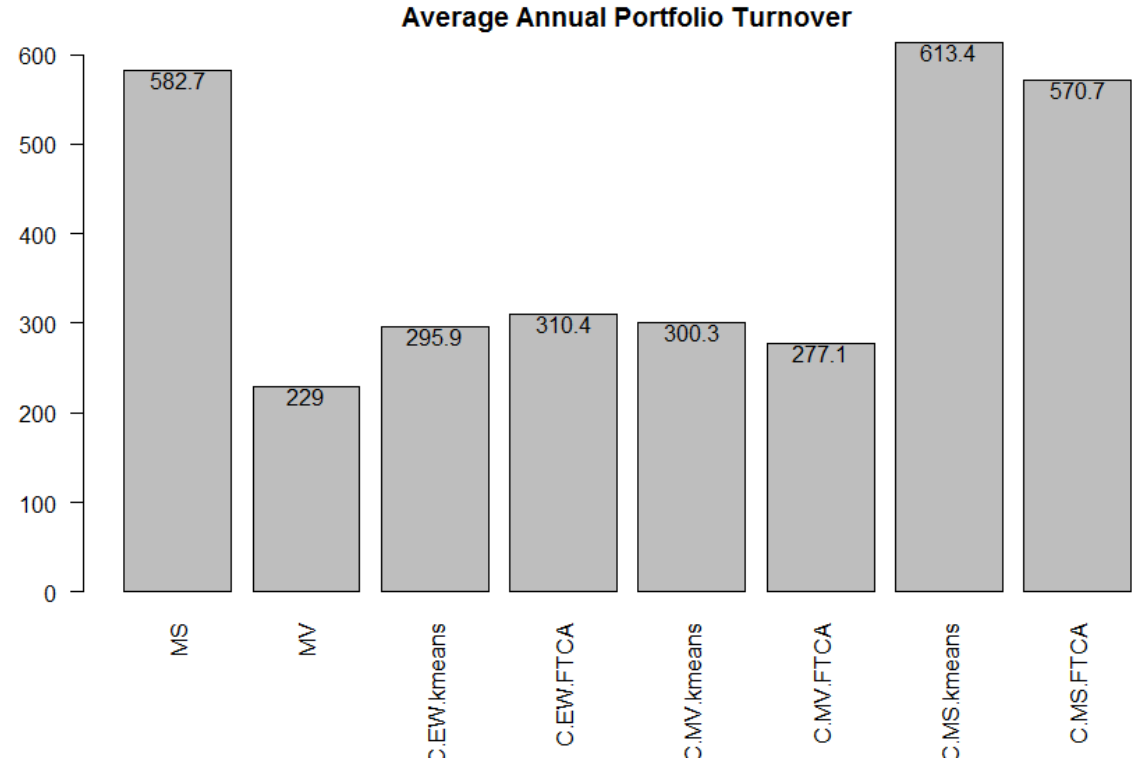


Figure 9 Average Annual Portfolio Turnover bar shart

First, in terms of correlation, which both clustering methods use as a measure of distance between points in a feature space, k-means generates a more well-distributed clusters whereas FTCA produces an

outcome where large number of clusters contain only one individual asset.From this observation, I then compare the results of back-testing two investment strategies based on the clusters results suggested by these two algorithms to have a close-up investigation. The average annual portfolio turnover bar chart suggests that, portfolio turnover is much higher if investors target to maximizize Sharpe ratio, i.e. risk adjusted return, compared with other strategies such as equal weight portfolio or portfolio risk (in terms of volatility) minimization. The cumulative performance time trend shows the differences among various strategies from a different angle: holding investment objective to be the same, k-means clustering outperforms FTCA during most of the time window.

Such observations guides me to focus on k-means clustering at the first stage. By now, I how show the investigation process of narrowing down the choices of various clustering algorithms and how I finally determine to use combination of factor model with clustering algorithms.

The following two graphs visualize the Euclidean distance by scaling multi-dimensional vectors to a plane. [11]

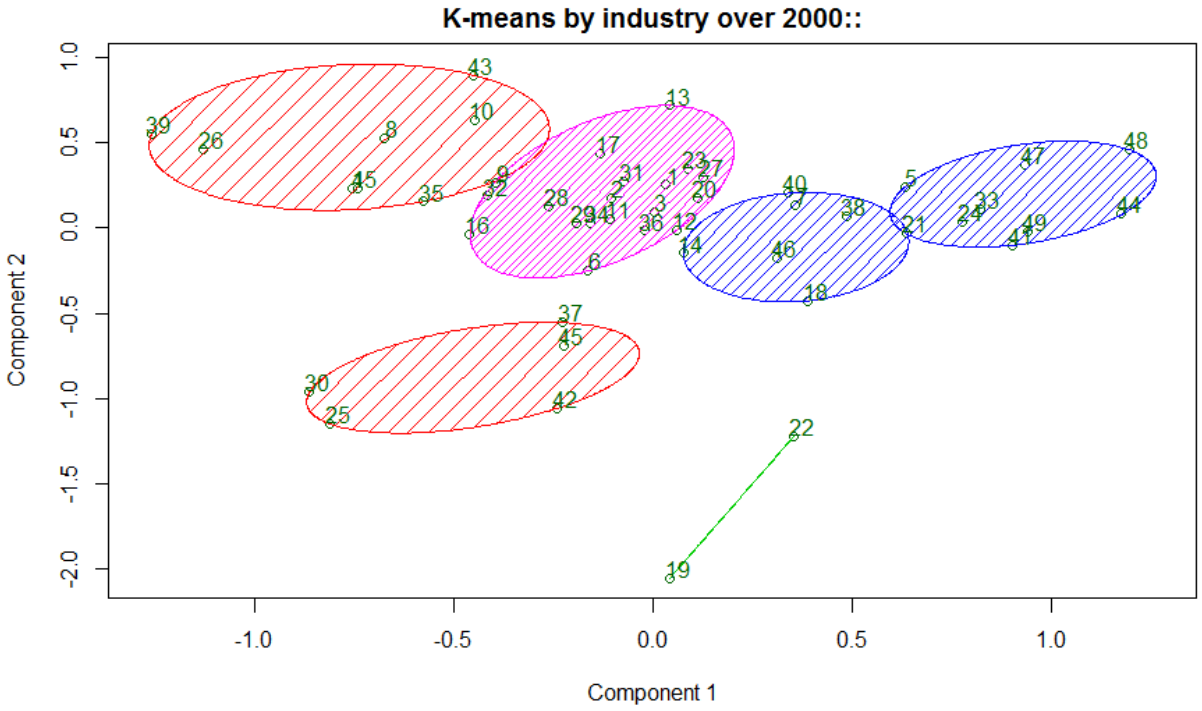


Figure 10 Group visualization by Industral factors

[11] See Appendix A for more information on ***Classical Multidimensional Scaling.***

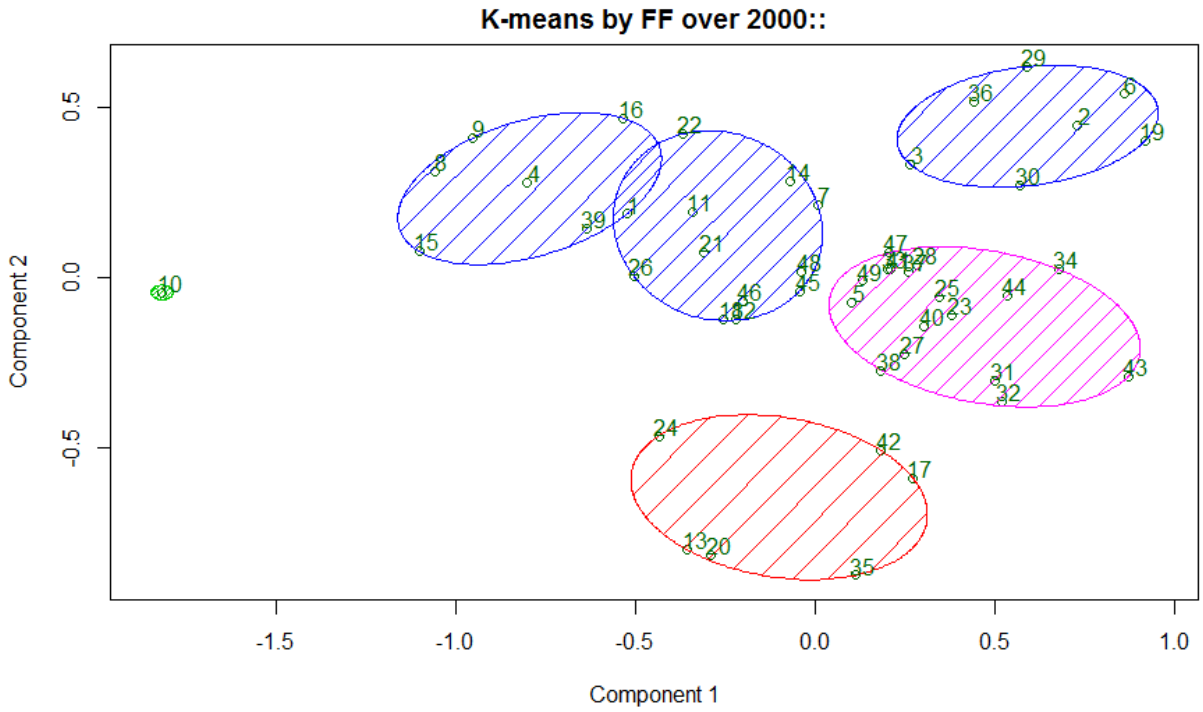


Figure 11 Group visualization by Fama-French factors

Combination with optimization procedure

After partitioning the entire pool into k different groups by clustering algorithms, the remaining problem is to select representatives from each group in order to form the best portfolio that satisfies the cardinality constraint.(Jiang, 2011) In Jiang's paper, "best" is defined in terms of portfolio mean and variance trade-off. There are other measures which are sometimes considered to be even better than traditional Markowitz' model, such as mean-absolute deviation Portfolio Optimization and minimizing conditional value-at-risk of portfolio. These criteria may be applied in this project in the future, but I merely show mean-variance optimization results at current stage to keep consistent with the literature.

The pre-grouping outcome of clustering algorithms empowers a large decrease in the number of possible combinations of candidate stocks and thus significantly relieving computational burden. Recall that the original CCMV problem tries to find k numbers of stocks directly from the assets universe where there are $\binom{n}{k}$ number of choices. In this project with n = 49 and k = 3, there are 49C3 = 18424 combinations to be considered! After partitioning the entire pool into k different groups, we now only need to consider picking the representatives within each group. More specifically, the revised problem is stated as follows:

$$\min_x x'Qx$$

$$\text{s.t. } r'x \geq \bar{r}$$

$$\mathbf{1}'x = 1$$

$$\alpha_j \leq \sum_{i \in I_j}^{n} b_i \leq \beta_j$$

$$\sum_{i=1}^{n} b_i = k$$

$$b_i = \begin{cases} 1, if\ x_i > 0 \\ 0, if\ x_i = 0 \end{cases}$$

$$i = 1,2, \dots, n.$$
$$j = 1,2, \dots, k.$$

Where I_j represents the k groups of risky assets partitioned by clustering algorithm, for instance, I_1 represents the binary variables of those assets who belong to group 1, I_2 represents the binary variables of those assets who belong to group 2, etc. Thus, by imposing both a lower and upper bound on the total number of candidate stocks that investors could pick from each group, the revised problem has only $\prod_{j=1}^{k} \left(\beta_j - \alpha_j + 1\right)$possible options, which is much smaller than the combinatorial number $\binom{n}{k}$.

Direct Optimization

With deeper understandings about both the market structure and the CCMV problem as the project goes by, I also devote to provide direct solutions to CCMV based on some interesting yet meaningful findings during the process.

1. With long-only constraint, many of the weight elements are spontaneously tend to 0, which largely reduces dimensions of CCMV problem when comparing with the case when shorting is allowed. Thus, optimization method such as Branch and Bound are computational feasible.
2. By carefully pick an appropriate lower and upper bound on each group, optimization can achieve a result where the factor model and clustering algorithm does not matter at all.

Now, I show the result of both static and dynamic results of direct optimization and optimization combining with heuristics results.

First, it comes to the comparison of the number of Branch and Bound methods used in various optimization problem.[12] Intuitively, the direct optimization requires the most number of calls while the differences among different subset optimization problems are not severe.
Second, each portfolio result is visualized in terms of portfolio weight allocation and number of assets contained in the portfolio. And here are some of the meaningful findings:

- The number of assets contained in unconstrained portfolio decreases when the risk level increases.
- The cardinality constraints on both direct optimization and subset optimization problems are effective as the maximum number assets contained in these portfolios are k. (here k is pre-defined to be 3)
- Comparing results of subset optimization problems, when setting lower bound to be 0 and upper bound to be 1, all of the subset optimization problems have cases when none of the assets are selected into the portfolio, meaning that the problem has no solution. While setting the upper bound to be k, which is a less strict constraint, such cases are eliminated.

[12] See Appendix B.

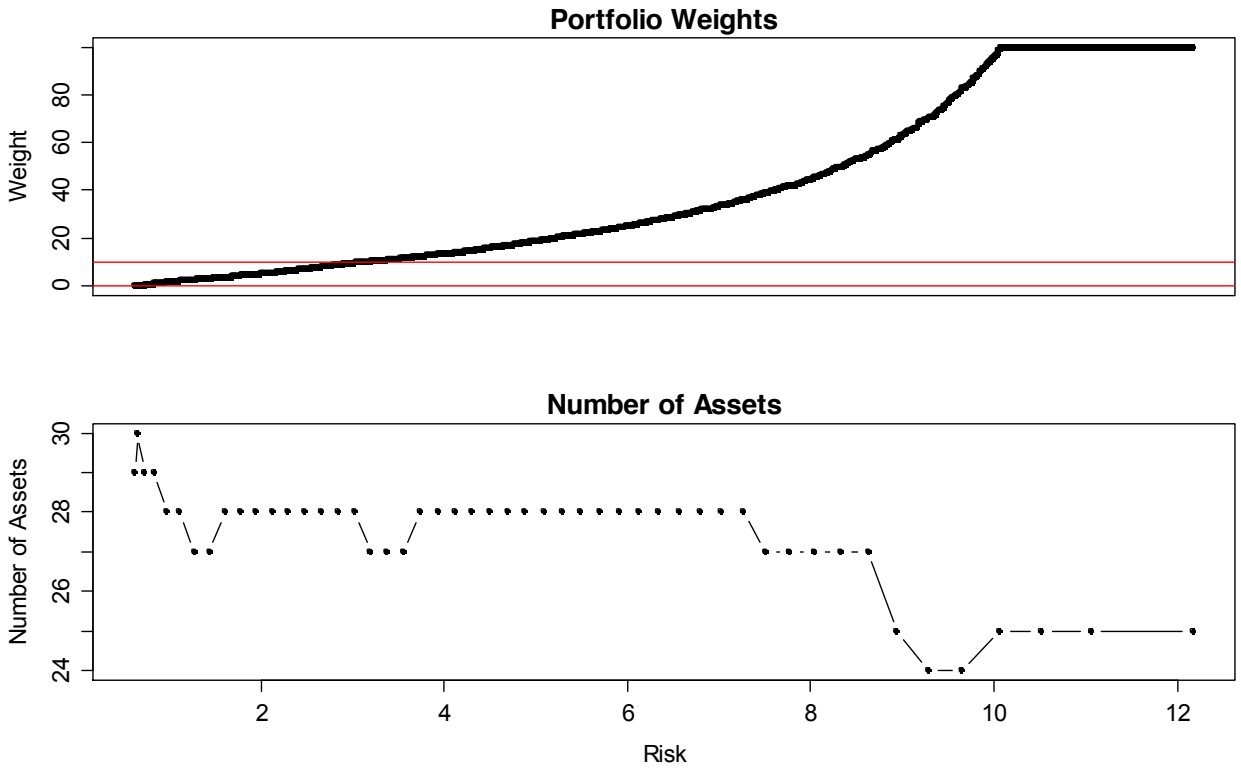


Figure 12 Long & Short Portfolio

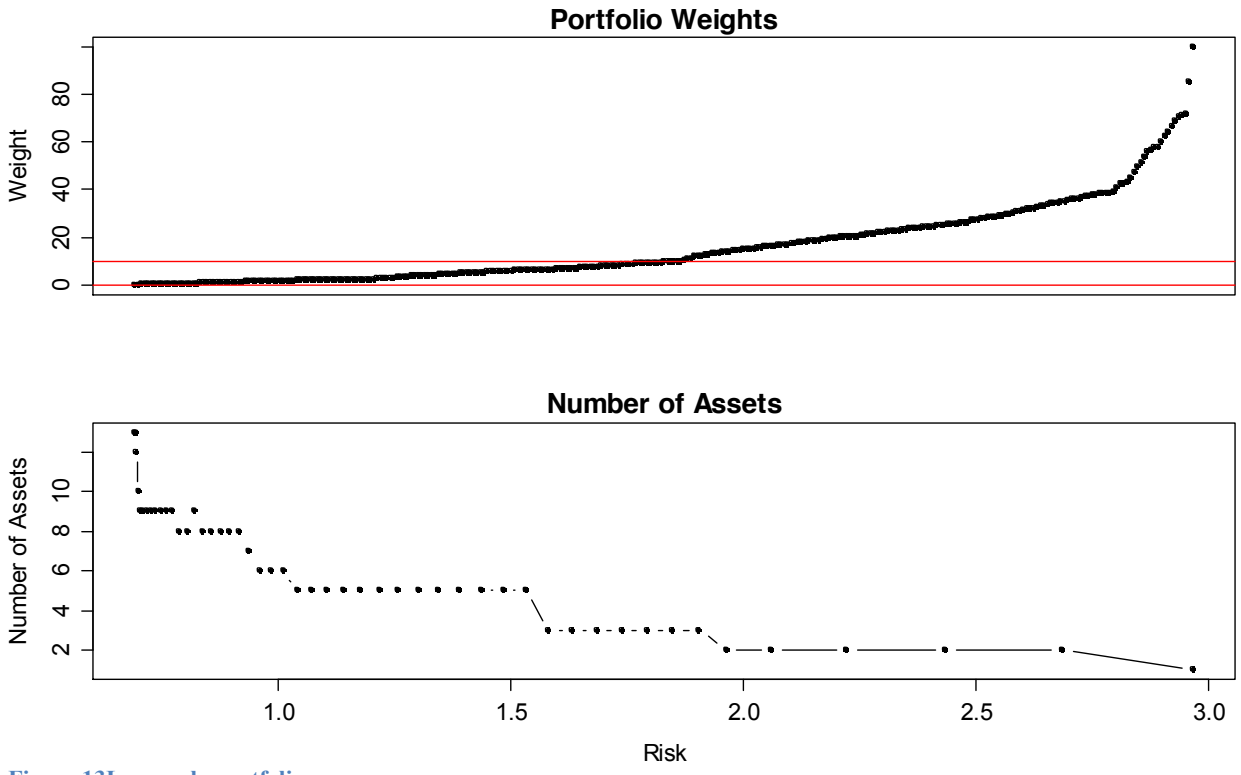


Figure 13Long only portfolio

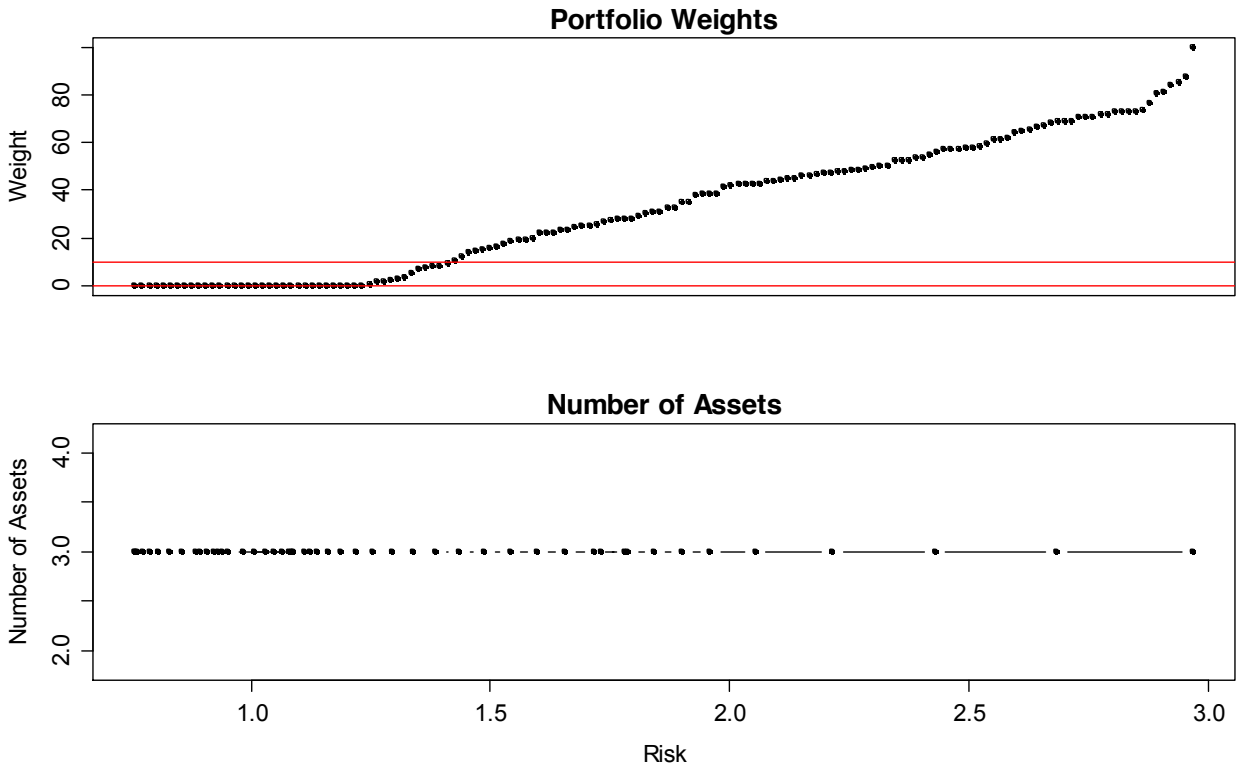


Figure 14 Cardinality optimization

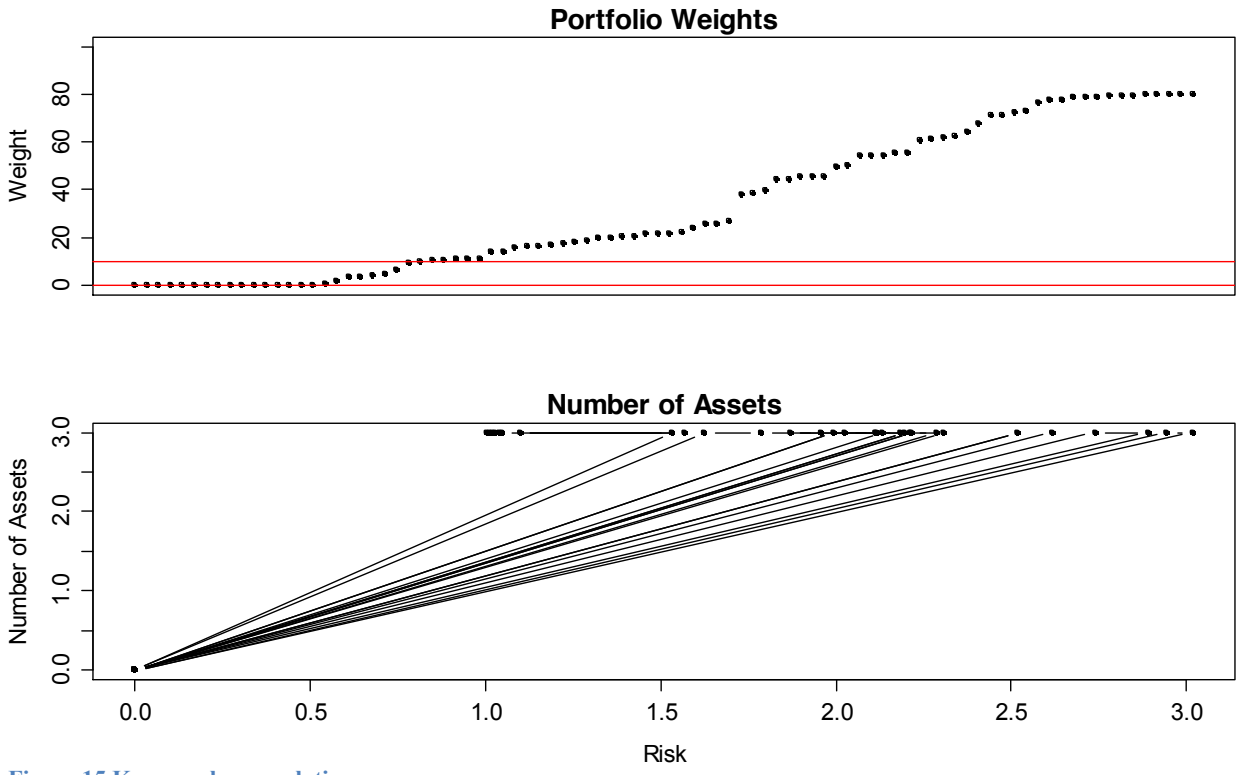


Figure 15 K-means by correlation

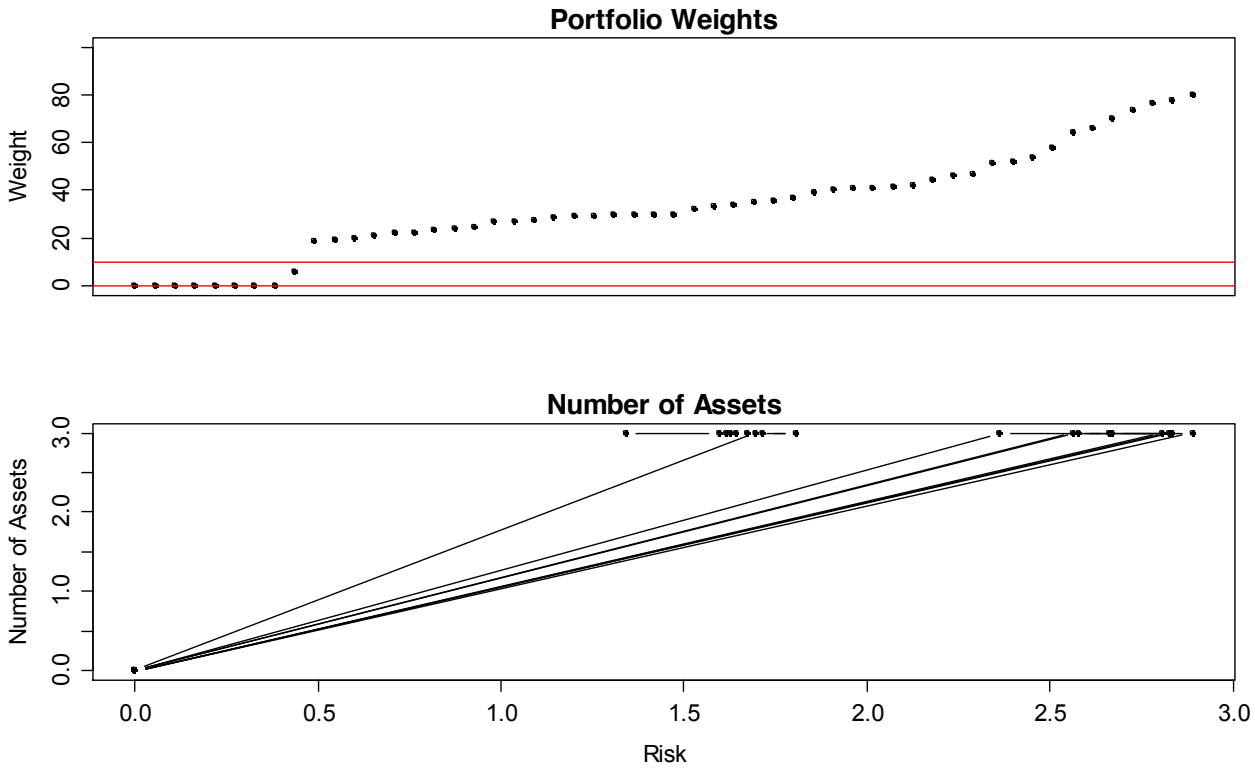


Figure 16 K-means Industry

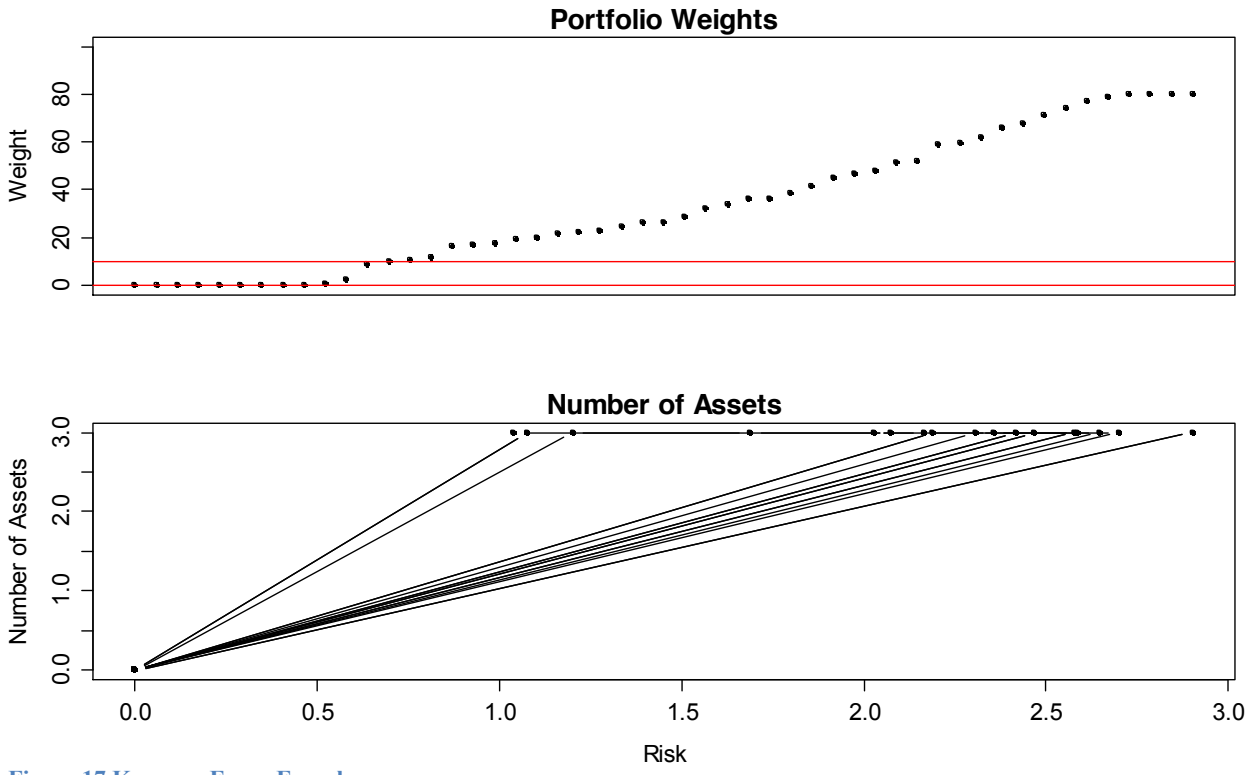


Figure 17 K-means Fama-French

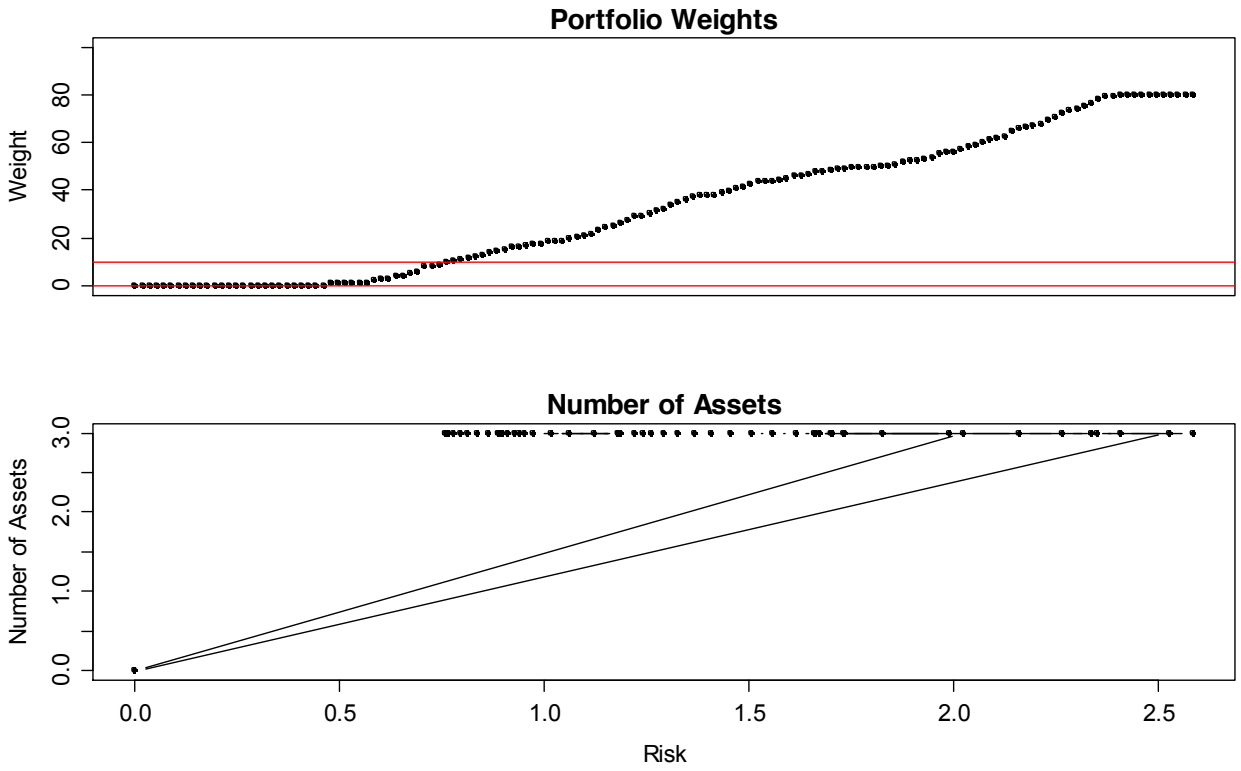


Figure 18 K-means FTCA

Third, the efficient frontier is plotted as a static comparison of portfolio performance.

- Efficient frontier of randomly picked k(=3) assets from n (=49) assets pool are shown in gray curves. By repeating sufficient number of sampling of the numeration, I am confident to ensure the efficient frontier computed by my algorithm is generally better than other choices. [Figure 19]
- When setting upper bound$\beta_j = 1$, various subletting strategies differ from each other at different risk level.
- When setting upper bound$\beta_j = 3$, there is no difference among the direct cardinality optimization and sub setting optimization problem, and also the clustering algorithm and factor model does not really matter.
- MD portfolio, which represent maximize diversification shows a special pattern among all of the strategies, and I would like to investigate more in the future work.

Forth, the transformation map of various strategies help to have a better understranding about the portfolio structure. For example, with shorting allowed, the weight for each asset is not bounded, whereas when shorting is limited, many assets weights are spontaneously force to be 0. Additionally, there are areas remained white for subsetting optimization portfolios, implying that there are cases when none of the assets is selected, which is in consistant with the observations above.

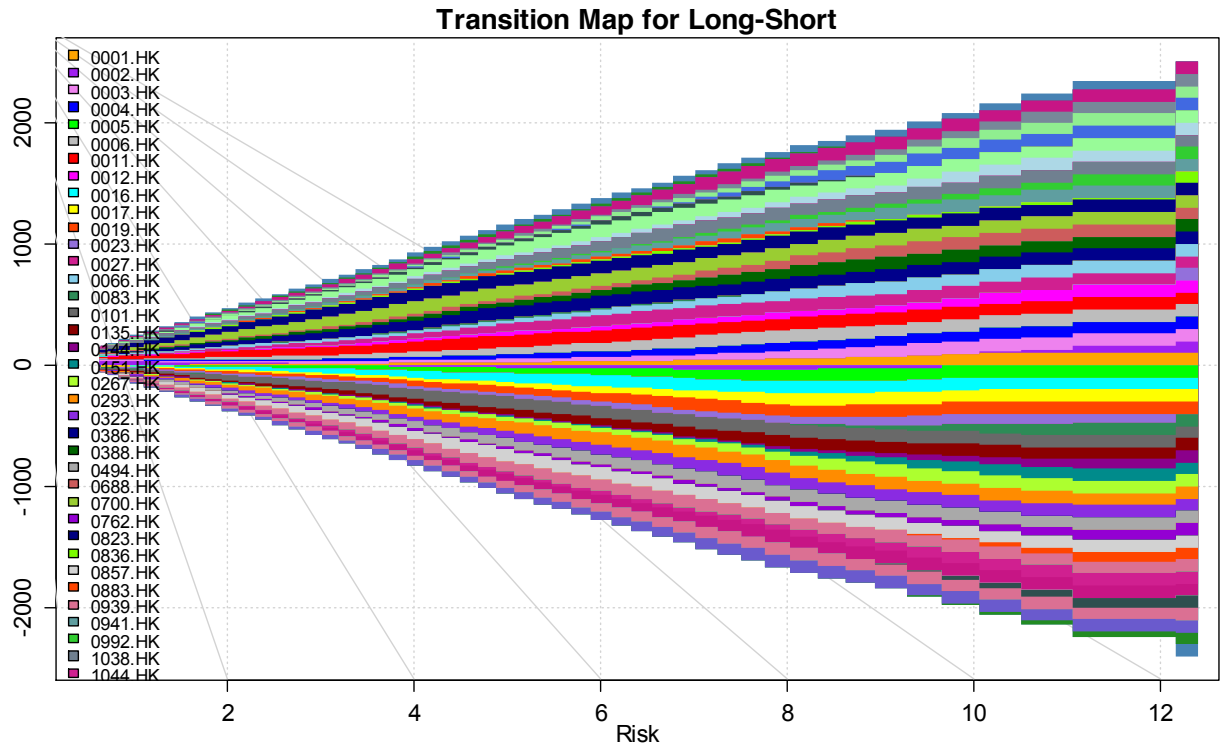


Figure 19 Transition map Long short

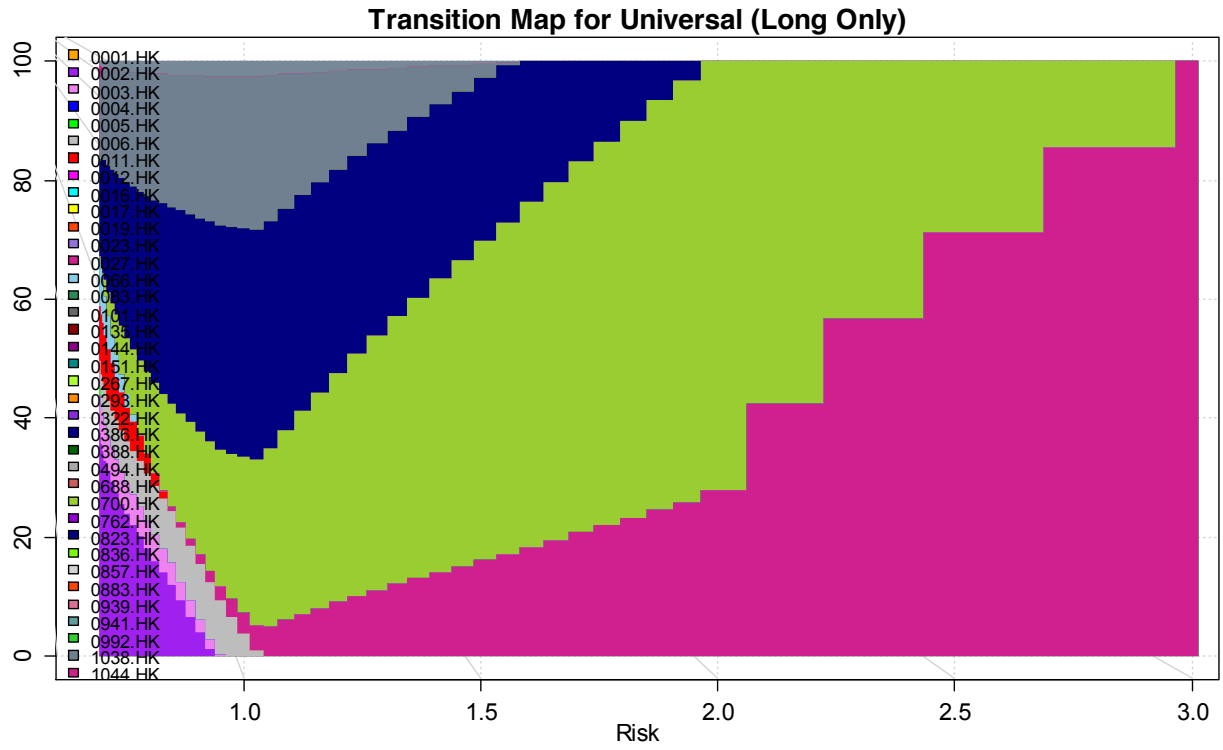


Figure 20Transition map Long only

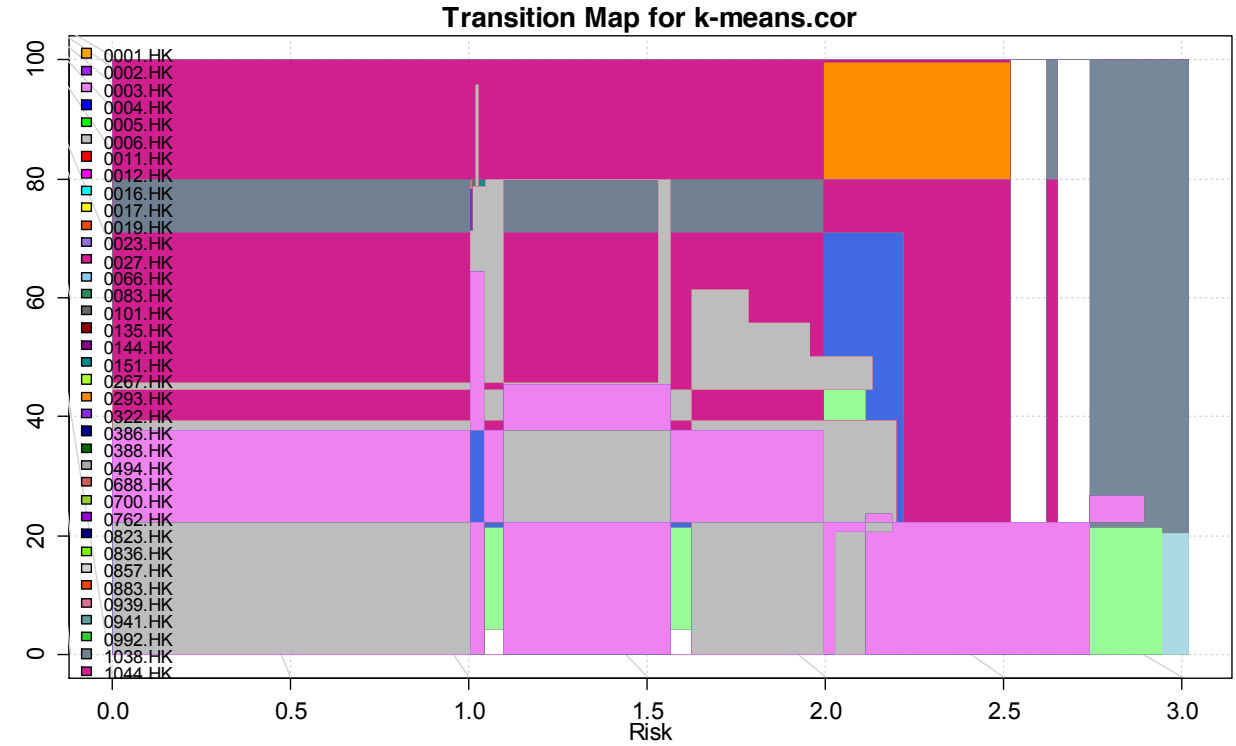


Figure 21 transition map for subsetting optimization

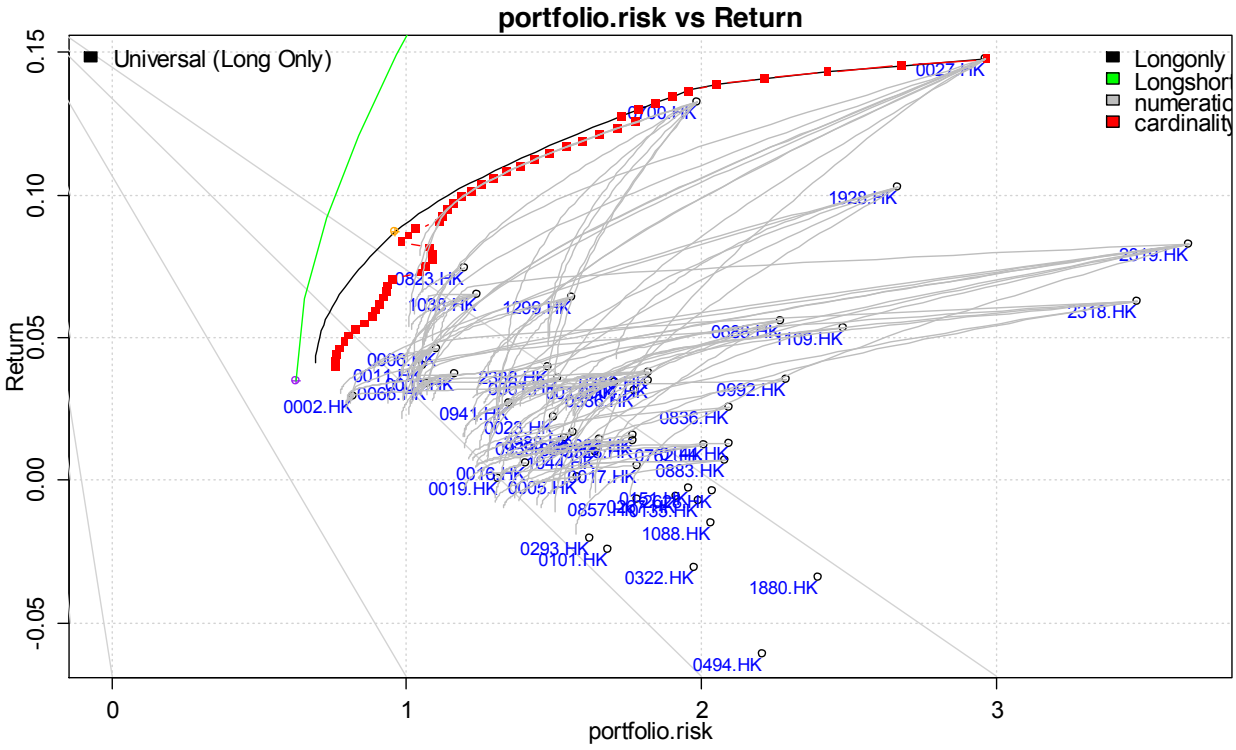


Figure 22 Efficient Frontier with random numeration comparison

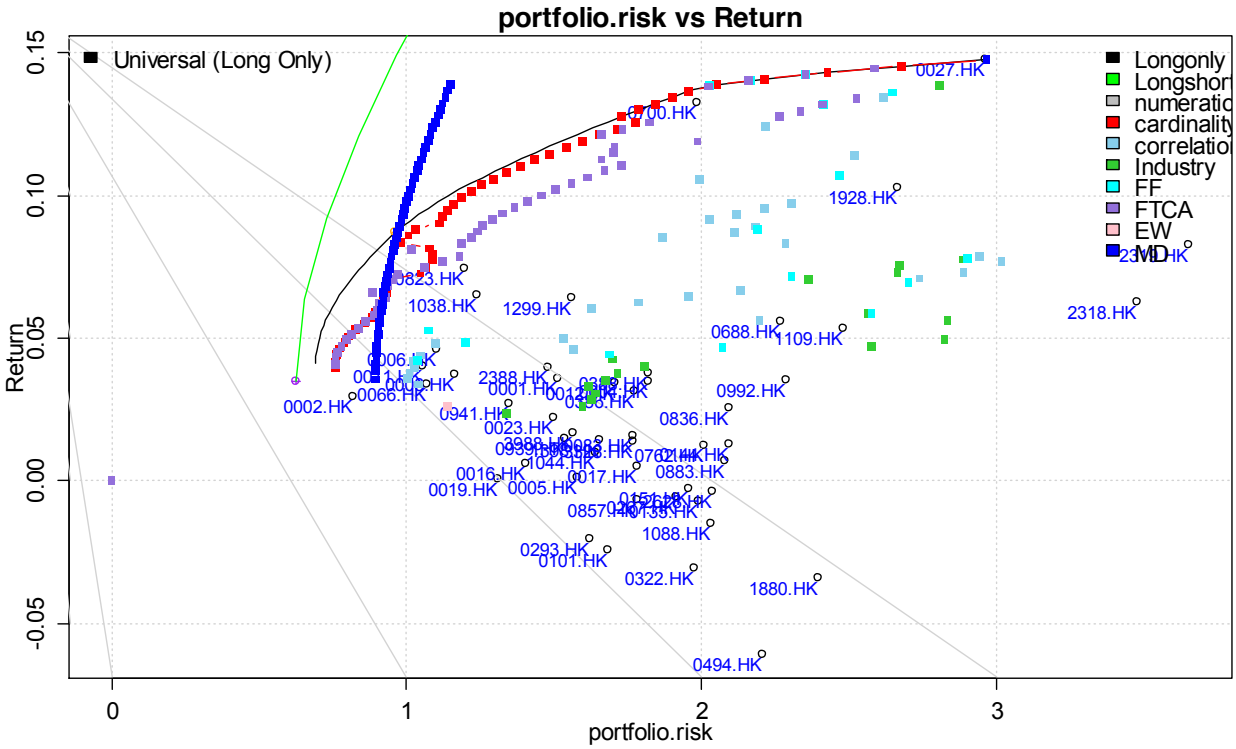


Figure 23. Efficient Frontier comparison for different optimization strategies when upper bound is 1

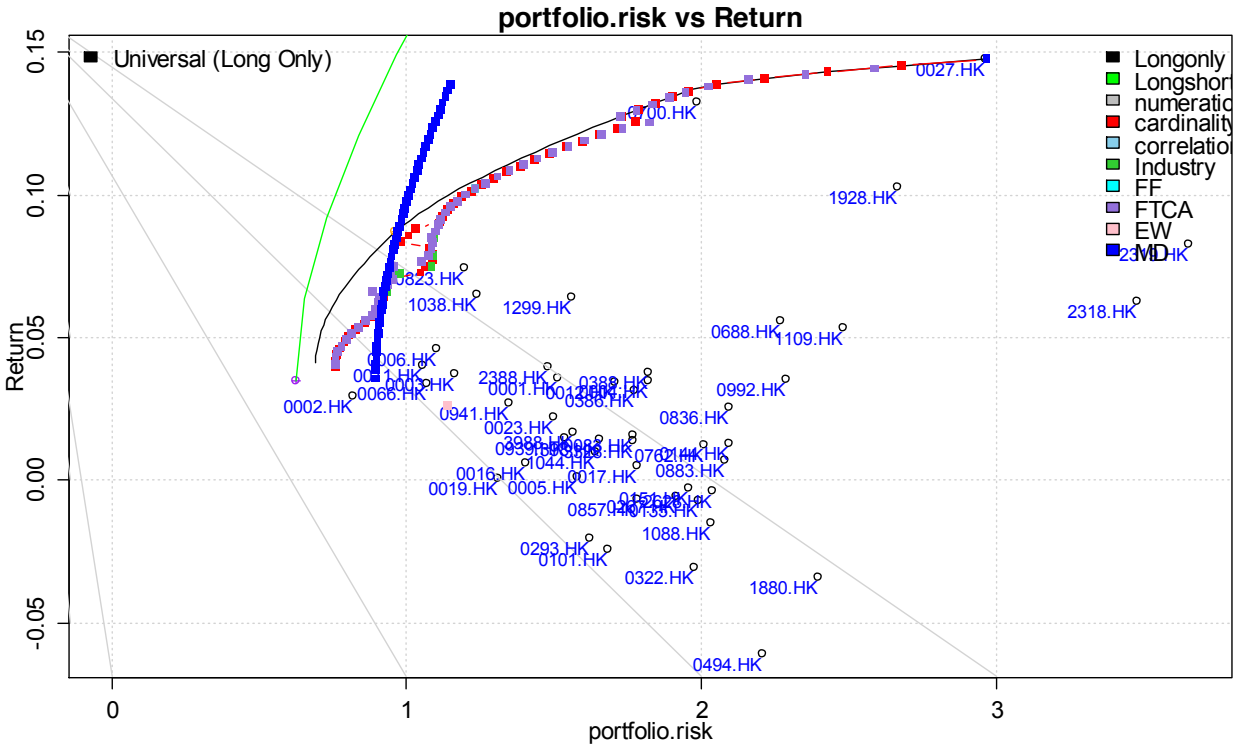


Figure 24Efficient Frontier comparison for different optimization strategies when upper bound is 3

Lastly, I present back testing of different strategies in comparison with equal weight portfolio as benchmark.

EW stands for equal weight portfolio, MvLS represents the strategy to minimize portfolio variance when shorting is allowed, MVL is similar to MvLS but shorting is limited. max.sharpe.Cardinality refers to the portfolio when imposing cardinality constraints and the objective is to maximize sharpe ratio. Cor, ind, FF and FTCA refers to the sub setting optimization problem in terms of k-means by correlation, by industrial factor model, by Fama-French Factor Model and by Threshold correlation respectively.

Figure 25Cumulative performance comparison

Appendix A

Simple note on Classical Multidimensional Scaling

Multidimensional scaling takes a set of dissimilarities and returns a set of points such that the distances between the points are approximately equal to the dissimilarities. Following the analysis of Mardia(1978), a set of Euclidean distances on n points can be represented exactly in at most n - 1 dimensions and returns the best-fitting k-dimensional representation, where k may be less than the argument k. Here, I set k equal to 2 to enable a 2-D visualization of multi-dimensional vectors. More specifically, as discussed above, industrial factor model and Fama-French factor model characterize risky assets into 12 and 5 dimensional vectors respectively. Classical Multidimensional Scaling empowers visual presentation of the cluster results for better understanding.

Appendix B

```
> cardinality= portopt(ia, constraints.cardinality,50, 'Cardinality')
173 QP calls made to solve problem with 49 binary variables using Branch&Bound
130 QP calls made to solve problem with 49 binary variables using Branch&Bound
121 QP calls made to solve problem with 49 binary variables using Branch&Bound
79 QP calls made to solve problem with 49 binary variables using Branch&Bound
133 QP calls made to solve problem with 49 binary variables using Branch&Bound
143 QP calls made to solve problem with 49 binary variables using Branch&Bound
169 QP calls made to solve problem with 49 binary variables using Branch&Bound
203 QP calls made to solve problem with 49 binary variables using Branch&Bound
331 QP calls made to solve problem with 49 binary variables using Branch&Bound
253 QP calls made to solve problem with 49 binary variables using Branch&Bound
219 QP calls made to solve problem with 49 binary variables using Branch&Bound
193 QP calls made to solve problem with 49 binary variables using Branch&Bound
190 QP calls made to solve problem with 49 binary variables using Branch&Bound
149 QP calls made to solve problem with 49 binary variables using Branch&Bound
127 QP calls made to solve problem with 49 binary variables using Branch&Bound
311 QP calls made to solve problem with 49 binary variables using Branch&Bound
201 QP calls made to solve problem with 49 binary variables using Branch&Bound
224 QP calls made to solve problem with 49 binary variables using Branch&Bound
168 QP calls made to solve problem with 49 binary variables using Branch&Bound
111 QP calls made to solve problem with 49 binary variables using Branch&Bound
11 QP calls made to solve problem with 49 binary variables using Branch&Bound
11 QP calls made to solve problem with 49 binary variables using Branch&Bound
9 QP calls made to solve problem with 49 binary variables using Branch&Bound
81 QP calls made to solve problem with 49 binary variables using Branch&Bound
83 QP calls made to solve problem with 49 binary variables using Branch&Bound
89 QP calls made to solve problem with 49 binary variables using Branch&Bound
85 QP calls made to solve problem with 49 binary variables using Branch&Bound
95 QP calls made to solve problem with 49 binary variables using Branch&Bound
95 QP calls made to solve problem with 49 binary variables using Branch&Bound
95 QP calls made to solve problem with 49 binary variables using Branch&Bound
95 QP calls made to solve problem with 49 binary variables using Branch&Bound
95 QP calls made to solve problem with 49 binary variables using Branch&Bound
95 QP calls made to solve problem with 49 binary variables using Branch&Bound
87 QP calls made to solve problem with 49 binary variables using Branch&Bound
93 QP calls made to solve problem with 49 binary variables using Branch&Bound
97 QP calls made to solve problem with 49 binary variables using Branch&Bound
105 QP calls made to solve problem with 49 binary variables using Branch&Bound
67 QP calls made to solve problem with 49 binary variables using Branch&Bound
61 QP calls made to solve problem with 49 binary variables using Branch&Bound
145 QP calls made to solve problem with 49 binary variables using Branch&Bound
45 QP calls made to solve problem with 49 binary variables using Branch&Bound
45 QP calls made to solve problem with 49 binary variables using Branch&Bound
37 QP calls made to solve problem with 49 binary variables using Branch&Bound
33 QP calls made to solve problem with 49 binary variables using Branch&Bound
33 QP calls made to solve problem with 49 binary variables using Branch&Bound
3 QP calls made to solve problem with 49 binary variables using Branch&Bound
3 QP calls made to solve problem with 49 binary variables using Branch&Bound
1 QP calls made to solve problem with 49 binary variables using Branch&Bound
1 QP calls made to solve problem with 49 binary variables using Branch&Bound
```

>
> w0 = portopt(ia,constraints.cluster.cor.90, 50, 'k-means.cor')
155 QP calls made to solve problem with 49 binary variables using Branch&Bound
119 QP calls made to solve problem with 49 binary variables using Branch&Bound
75 QP calls made to solve problem with 49 binary variables using Branch&Bound
109 QP calls made to solve problem with 49 binary variables using Branch&Bound
135 QP calls made to solve problem with 49 binary variables using Branch&Bound
149 QP calls made to solve problem with 49 binary variables using Branch&Bound
171 QP calls made to solve problem with 49 binary variables using Branch&Bound
229 QP calls made to solve problem with 49 binary variables using Branch&Bound
223 QP calls made to solve problem with 49 binary variables using Branch&Bound
243 QP calls made to solve problem with 49 binary variables using Branch&Bound
185 QP calls made to solve problem with 49 binary variables using Branch&Bound
149 QP calls made to solve problem with 49 binary variables using Branch&Bound
57 QP calls made to solve problem with 49 binary variables using Branch&Bound
145 QP calls made to solve problem with 49 binary variables using Branch&Bound
93 QP calls made to solve problem with 49 binary variables using Branch&Bound
39 QP calls made to solve problem with 49 binary variables using Branch&Bound
33 QP calls made to solve problem with 49 binary variables using Branch&Bound
113 QP calls made to solve problem with 49 binary variables using Branch&Bound
131 QP calls made to solve problem with 49 binary variables using Branch&Bound
121 QP calls made to solve problem with 49 binary variables using Branch&Bound
107 QP calls made to solve problem with 49 binary variables using Branch&Bound
69 QP calls made to solve problem with 49 binary variables using Branch&Bound
81 QP calls made to solve problem with 49 binary variables using Branch&Bound
75 QP calls made to solve problem with 49 binary variables using Branch&Bound
75 QP calls made to solve problem with 49 binary variables using Branch&Bound
95 QP calls made to solve problem with 49 binary variables using Branch&Bound
85 QP calls made to solve problem with 49 binary variables using Branch&Bound
85 QP calls made to solve problem with 49 binary variables using Branch&Bound
95 QP calls made to solve problem with 49 binary variables using Branch&Bound
95 QP calls made to solve problem with 49 binary variables using Branch&Bound
95 QP calls made to solve problem with 49 binary variables using Branch&Bound
95 QP calls made to solve problem with 49 binary variables using Branch&Bound
85 QP calls made to solve problem with 49 binary variables using Branch&Bound
95 QP calls made to solve problem with 49 binary variables using Branch&Bound
87 QP calls made to solve problem with 49 binary variables using Branch&Bound
97 QP calls made to solve problem with 49 binary variables using Branch&Bound
77 QP calls made to solve problem with 49 binary variables using Branch&Bound
77 QP calls made to solve problem with 49 binary variables using Branch&Bound
17 QP calls made to solve problem with 49 binary variables using Branch&Bound
37 QP calls made to solve problem with 49 binary variables using Branch&Bound
71 QP calls made to solve problem with 49 binary variables using Branch&Bound
41 QP calls made to solve problem with 49 binary variables using Branch&Bound
41 QP calls made to solve problem with 49 binary variables using Branch&Bound
37 QP calls made to solve problem with 49 binary variables using Branch&Bound
27 QP calls made to solve problem with 49 binary variables using Branch&Bound
29 QP calls made to solve problem with 49 binary variables using Branch&Bound

3 QP calls made to solve problem with 49 binary variables using Branch&Bound
3 QP calls made to solve problem with 49 binary variables using Branch&Bound
3 QP calls made to solve problem with 49 binary variables using Branch&Bound

> w1 = portopt(ia,constraints.cluster.Industry, 50, 'k-means.Industry')
155 QP calls made to solve problem with 49 binary variables using Branch&Bound
119 QP calls made to solve problem with 49 binary variables using Branch&Bound
75 QP calls made to solve problem with 49 binary variables using Branch&Bound
109 QP calls made to solve problem with 49 binary variables using Branch&Bound
135 QP calls made to solve problem with 49 binary variables using Branch&Bound
149 QP calls made to solve problem with 49 binary variables using Branch&Bound
171 QP calls made to solve problem with 49 binary variables using Branch&Bound
229 QP calls made to solve problem with 49 binary variables using Branch&Bound
223 QP calls made to solve problem with 49 binary variables using Branch&Bound
243 QP calls made to solve problem with 49 binary variables using Branch&Bound
239 QP calls made to solve problem with 49 binary variables using Branch&Bound
149 QP calls made to solve problem with 49 binary variables using Branch&Bound
175 QP calls made to solve problem with 49 binary variables using Branch&Bound
145 QP calls made to solve problem with 49 binary variables using Branch&Bound
93 QP calls made to solve problem with 49 binary variables using Branch&Bound
47 QP calls made to solve problem with 49 binary variables using Branch&Bound
324 QP calls made to solve problem with 49 binary variables using Branch&Bound
113 QP calls made to solve problem with 49 binary variables using Branch&Bound
176 QP calls made to solve problem with 49 binary variables using Branch&Bound
121 QP calls made to solve problem with 49 binary variables using Branch&Bound
107 QP calls made to solve problem with 49 binary variables using Branch&Bound
105 QP calls made to solve problem with 49 binary variables using Branch&Bound
81 QP calls made to solve problem with 49 binary variables using Branch&Bound
75 QP calls made to solve problem with 49 binary variables using Branch&Bound
75 QP calls made to solve problem with 49 binary variables using Branch&Bound
95 QP calls made to solve problem with 49 binary variables using Branch&Bound
85 QP calls made to solve problem with 49 binary variables using Branch&Bound
95 QP calls made to solve problem with 49 binary variables using Branch&Bound
95 QP calls made to solve problem with 49 binary variables using Branch&Bound
95 QP calls made to solve problem with 49 binary variables using Branch&Bound
95 QP calls made to solve problem with 49 binary variables using Branch&Bound
95 QP calls made to solve problem with 49 binary variables using Branch&Bound
95 QP calls made to solve problem with 49 binary variables using Branch&Bound
95 QP calls made to solve problem with 49 binary variables using Branch&Bound
87 QP calls made to solve problem with 49 binary variables using Branch&Bound
97 QP calls made to solve problem with 49 binary variables using Branch&Bound
77 QP calls made to solve problem with 49 binary variables using Branch&Bound
77 QP calls made to solve problem with 49 binary variables using Branch&Bound
17 QP calls made to solve problem with 49 binary variables using Branch&Bound
37 QP calls made to solve problem with 49 binary variables using Branch&Bound
71 QP calls made to solve problem with 49 binary variables using Branch&Bound
41 QP calls made to solve problem with 49 binary variables using Branch&Bound
41 QP calls made to solve problem with 49 binary variables using Branch&Bound
37 QP calls made to solve problem with 49 binary variables using Branch&Bound
27 QP calls made to solve problem with 49 binary variables using Branch&Bound
29 QP calls made to solve problem with 49 binary variables using Branch&Bound

3 QP calls made to solve problem with 49 binary variables using Branch&Bound
3 QP calls made to solve problem with 49 binary variables using Branch&Bound
3 QP calls made to solve problem with 49 binary variables using Branch&Bound

> w2 = portopt(ia,constraints.cluster.FF, 50, 'k-means.FF')
155 QP calls made to solve problem with 49 binary variables using Branch&Bound
119 QP calls made to solve problem with 49 binary variables using Branch&Bound
75 QP calls made to solve problem with 49 binary variables using Branch&Bound
109 QP calls made to solve problem with 49 binary variables using Branch&Bound
135 QP calls made to solve problem with 49 binary variables using Branch&Bound
149 QP calls made to solve problem with 49 binary variables using Branch&Bound
171 QP calls made to solve problem with 49 binary variables using Branch&Bound
229 QP calls made to solve problem with 49 binary variables using Branch&Bound
223 QP calls made to solve problem with 49 binary variables using Branch&Bound
243 QP calls made to solve problem with 49 binary variables using Branch&Bound
185 QP calls made to solve problem with 49 binary variables using Branch&Bound
149 QP calls made to solve problem with 49 binary variables using Branch&Bound
57 QP calls made to solve problem with 49 binary variables using Branch&Bound
145 QP calls made to solve problem with 49 binary variables using Branch&Bound
93 QP calls made to solve problem with 49 binary variables using Branch&Bound
39 QP calls made to solve problem with 49 binary variables using Branch&Bound
33 QP calls made to solve problem with 49 binary variables using Branch&Bound
113 QP calls made to solve problem with 49 binary variables using Branch&Bound
131 QP calls made to solve problem with 49 binary variables using Branch&Bound
121 QP calls made to solve problem with 49 binary variables using Branch&Bound
107 QP calls made to solve problem with 49 binary variables using Branch&Bound
69 QP calls made to solve problem with 49 binary variables using Branch&Bound
81 QP calls made to solve problem with 49 binary variables using Branch&Bound
75 QP calls made to solve problem with 49 binary variables using Branch&Bound
75 QP calls made to solve problem with 49 binary variables using Branch&Bound
95 QP calls made to solve problem with 49 binary variables using Branch&Bound
85 QP calls made to solve problem with 49 binary variables using Branch&Bound
85 QP calls made to solve problem with 49 binary variables using Branch&Bound
95 QP calls made to solve problem with 49 binary variables using Branch&Bound
95 QP calls made to solve problem with 49 binary variables using Branch&Bound
95 QP calls made to solve problem with 49 binary variables using Branch&Bound
95 QP calls made to solve problem with 49 binary variables using Branch&Bound
85 QP calls made to solve problem with 49 binary variables using Branch&Bound
95 QP calls made to solve problem with 49 binary variables using Branch&Bound
87 QP calls made to solve problem with 49 binary variables using Branch&Bound
97 QP calls made to solve problem with 49 binary variables using Branch&Bound
77 QP calls made to solve problem with 49 binary variables using Branch&Bound
77 QP calls made to solve problem with 49 binary variables using Branch&Bound
17 QP calls made to solve problem with 49 binary variables using Branch&Bound
37 QP calls made to solve problem with 49 binary variables using Branch&Bound
71 QP calls made to solve problem with 49 binary variables using Branch&Bound
41 QP calls made to solve problem with 49 binary variables using Branch&Bound
41 QP calls made to solve problem with 49 binary variables using Branch&Bound
37 QP calls made to solve problem with 49 binary variables using Branch&Bound

27 QP calls made to solve problem with 49 binary variables using Branch&Bound
29 QP calls made to solve problem with 49 binary variables using Branch&Bound
3 QP calls made to solve problem with 49 binary variables using Branch&Bound
3 QP calls made to solve problem with 49 binary variables using Branch&Bound
3 QP calls made to solve problem with 49 binary variables using Branch&Bound

w3 = portopt(ia,constraints.cluster.FTCA, 50, 'FTCA')
155 QP calls made to solve problem with 49 binary variables using Branch&Bound
119 QP calls made to solve problem with 49 binary variables using Branch&Bound
75 QP calls made to solve problem with 49 binary variables using Branch&Bound
109 QP calls made to solve problem with 49 binary variables using Branch&Bound
135 QP calls made to solve problem with 49 binary variables using Branch&Bound
149 QP calls made to solve problem with 49 binary variables using Branch&Bound
171 QP calls made to solve problem with 49 binary variables using Branch&Bound
229 QP calls made to solve problem with 49 binary variables using Branch&Bound
223 QP calls made to solve problem with 49 binary variables using Branch&Bound
243 QP calls made to solve problem with 49 binary variables using Branch&Bound
185 QP calls made to solve problem with 49 binary variables using Branch&Bound
149 QP calls made to solve problem with 49 binary variables using Branch&Bound
57 QP calls made to solve problem with 49 binary variables using Branch&Bound
145 QP calls made to solve problem with 49 binary variables using Branch&Bound
93 QP calls made to solve problem with 49 binary variables using Branch&Bound
39 QP calls made to solve problem with 49 binary variables using Branch&Bound
33 QP calls made to solve problem with 49 binary variables using Branch&Bound
113 QP calls made to solve problem with 49 binary variables using Branch&Bound
131 QP calls made to solve problem with 49 binary variables using Branch&Bound
121 QP calls made to solve problem with 49 binary variables using Branch&Bound
107 QP calls made to solve problem with 49 binary variables using Branch&Bound
69 QP calls made to solve problem with 49 binary variables using Branch&Bound
81 QP calls made to solve problem with 49 binary variables using Branch&Bound
75 QP calls made to solve problem with 49 binary variables using Branch&Bound
75 QP calls made to solve problem with 49 binary variables using Branch&Bound
95 QP calls made to solve problem with 49 binary variables using Branch&Bound
85 QP calls made to solve problem with 49 binary variables using Branch&Bound
85 QP calls made to solve problem with 49 binary variables using Branch&Bound
95 QP calls made to solve problem with 49 binary variables using Branch&Bound
95 QP calls made to solve problem with 49 binary variables using Branch&Bound
95 QP calls made to solve problem with 49 binary variables using Branch&Bound
95 QP calls made to solve problem with 49 binary variables using Branch&Bound
85 QP calls made to solve problem with 49 binary variables using Branch&Bound
95 QP calls made to solve problem with 49 binary variables using Branch&Bound
87 QP calls made to solve problem with 49 binary variables using Branch&Bound
97 QP calls made to solve problem with 49 binary variables using Branch&Bound
77 QP calls made to solve problem with 49 binary variables using Branch&Bound
77 QP calls made to solve problem with 49 binary variables using Branch&Bound
17 QP calls made to solve problem with 49 binary variables using Branch&Bound
37 QP calls made to solve problem with 49 binary variables using Branch&Bound
71 QP calls made to solve problem with 49 binary variables using Branch&Bound
41 QP calls made to solve problem with 49 binary variables using Branch&Bound
41 QP calls made to solve problem with 49 binary variables using Branch&Bound
37 QP calls made to solve problem with 49 binary variables using Branch&Bound

27 QP calls made to solve problem with 49 binary variables using Branch&Bound
29 QP calls made to solve problem with 49 binary variables using Branch&Bound
3 QP calls made to solve problem with 49 binary variables using Branch&Bound
3 QP calls made to solve problem with 49 binary variables using Branch&Bound
3 QP calls made to solve problem with 49 binary variables using Branch&Bound

Reference

Sun, X., Zheng, X., & Li, D. (2013). Recent advances in mathematical programming with semi-continuous variables and cardinality constraint. Journal of the Operations Research Society of China, 1(1), 55-77.

Markowitz H. Portfolio selection[J]. The journal of finance, 1952, 7(1): 77-91.

Gao J, Li D. Optimal cardinality constrained portfolio selection[J]. Operations research, 2013, 61(3): 745-761.

Jiang K, Li D, Gao J, et al. Factor Model Based Clustering Approach for Cardinality Constrained Portfolio Selection[J]. IFAC Proceedings Volumes, 2014, 47(3): 10713-10718.

Luenberger D G. Investment science[J]. OUP Catalogue, 1997.

Fama E F, French K R. Common risk factors in the returns on stocks and bonds[J]. Journal of financial economics, 1993, 33(1): 3-56.

Carhart M M. On persistence in mutual fund performance[J]. The Journal of finance, 1997, 52(1): 57-82.

Griffin J M. Are the Fama and French factors global or country specific?[J]. Review of Financial Studies, 2002, 15(3): 783-803.

Han J, Pei J, Kamber M. Data mining: concepts and techniques[M]. Elsevier, 2011.

Jin X, Han J. Partitional clustering[M]//Encyclopedia of Machine Learning. Springer US, 2011: 766-766.

Varadi, d. (2016). *Fast Threshold Clustering Algorithm (FTCA)*. [online] CSSA. Available at: https://cssanalytics.wordpress.com/2013/11/26/fast-threshold-clustering-algorithm-ftca/ [Accessed 25 Nov. 2016].

Mardia, K.V. (1978) Some properties of classical multidimensional scaling. Communications on Statistics – Theory and Methods, A7, 1233–41.

Data source

1. Fama French factors, http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html
2. Hang Seng Index component stocks
   https://finance.yahoo.com/quote/%5EHSI/components?p=%5EHSI
3. Complementary data, such as risk free rate
   http://thomsonreuters.com/en.html