Mega Publishing Limited
Journal of Risk & Control, 3(1), 17-65 | December 1, 2016

Statistical Industry Classification

Zura Kakushadze[1] **and Willie Yu**[2]

Abstract

We give complete algorithms and source code for constructing (multi-level) statistical industry classifications, including methods for fixing the number of clusters at each level (and the number of levels). Under the hood there are clustering algorithms (e.g., k-means). However, what should we cluster? Correlations? Returns? The answer turns out to be neither and our backtests suggest that these details make a sizable difference. We also give an algorithm and source code for building "hybrid" industry classifications by improving off-the-shelf "fundamental" industry classifications by applying our statistical industry classification methods to them. The presentation is intended to be pedagogical and geared toward practical applications in quantitative trading.

JEL Classification numbers: G00
Keywords: industry classification; clustering; cluster numbers; machine learning; statistical risk models; industry risk factors; optimization; regression; mean-reversion; correlation matrix; factor loadings; principal components; hierarchical agglomerative clustering; k-means; statistical methods; multilevel

[1] Ph.D., is the President of Quantigic® Solutions LLC, 1127 High Ridge Road #135, Stamford, CT 06905
Full Professor at Free University of Tbilisi, Business School & School of Physics, David Agmashenebeli Alley, Tbilisi, 0159, Georgia. E-mail: zura@quantigic.com
[2] Ph.D., is a Research Fellow at Centre for Computational Biology, Duke-NUS Medical School, College Road, Singapore 169857. E-mail: willie.yu@duke-nus.edu.sg

Article Info: *Received* : July 11, 2016. *Published online* : December 1, 2016.
Invited Editorial.

1 Introduction and Summary

Industry classifications such as GICS, BICS, ICB, NAICS, SIC, etc.[3] are widely used in quantitative trading. They group stocks into baskets, e.g., industries, i.e., based on some kind of a similarity criterion. On general grounds one then expects (or hopes) that stocks within such baskets on average should be relatively highly correlated. This is valuable information and can be used in various ways. E.g., one can build a simple mean-reversion statistical arbitrage strategy whereby one assumes that stocks in a given industry move together, cross-sectionally demeans stock returns within said industry, shorts stocks with positive residual returns and goes long stocks with negative residual returns, with some generally nonuniform weights.[4] Industries can also be used as risk factors in multifactor risk models.[5].

The aforementioned "fundamental" industry classifications are based on grouping companies together based on fundamental/economic data (see Section 2), which is expected to add value on longer holding horizons. What about shorter holding horizons relevant to quantitative trading strategies? Other than a large number of market players using such industry classifications to arbitrage mispricings,[6] how do we know that they are competitive with purely statistical methods at short horizons?

It is no secret that modern quantitative trading heavily relies on statistical methods such as data mining, machine learning, clustering algorithms, etc. However, after all, quantitative trading is a secretive field and resources on how things are done in practice are at best scarce.[7] The purpose of these notes is to discuss a systematic quantitative framework – in what is intended to be a "pedagogical" fashion – for building what we refer to as *statistical industry classifications*, solely based on stock returns and no additional extraneous data. Under the hood we have clustering algorithms. However, picking a clustering algorithm – and we will see that some work better than others – is insufficient. E.g., what should we cluster? Correlations? Returns? The answer turns out to be neither and stems from quantitative trading intuition, which is not something one expects to find in machine learning books. We discuss various nuances in constructing statistical

[3] Hereinafter we will refer to these as "fundamental" industry classifications (see below).

[4] More generally, one employs a weighted regression instead of demeaning, and there are various ways of fixing the aforesaid weights. For a pedagogical discussion, see, e.g., [Kakushadze, 2015a].

[5] For a discussion and literature on multifactor risk models, see, e.g., [Grinold and Kahn, 2000]

[6] This very relevant reason should not to be underestimated, despite its "behavioral" nature.

[7] Thus, we are unaware of another paper discussing the material herein at short horizons.

industry classifications, and it is those nuances that make a sizable difference. Quant trading is all about detail.

One motivation for considering statistical industry classifications – apart from the evident, to wit, the fact that they differ from "fundamental" industry classifications and are widely used in quant trading – is scenarios where "fundamental" industry classifications are unavailable (or are of subpar quality). This could be in emerging or smaller markets, or even in the U.S. if the underlying trading portfolios are relatively small and a "fundamental" industry classification produces too fragmented a grouping. However, perhaps an equally – if not more – important motivation is application of these methods to returns for "instruments" other than stocks, e.g., quantitative trading alphas, for which there is no analog of a "fundamental" industry classification [Kakushadze and Yu, 2016c]. We will keep this in mind below.[8]

In Section 2 we briefly review some generalities of (binary) "fundamental" industry classifications to set up the framework for further discussion. Next, in Section 3 we address the issue of what to cluster. We discuss why clustering correlations is suboptimal, and why so is directly clustering returns. We argue that returns should be normalized before clustering and give an explicit prescription for such normalization. We then discuss how to construct single-level and multilevel (hierarchical – e.g., BICS has 3 levels: sectors, industries and subindustries) statistical industry classifications together with some tweaks (e.g., cross-sectionally demeaning returns at less granular levels). Many clustering algorithms such as k-means are not deterministic. This can be a nuisance. We give an explicit prescription for aggregating classifications from multiple samplings, which in fact improves stability and performance. We discuss algorithms for "bottom-up" (most granular to least granular level), "top-down" (least granular to most granular level) and "relaxation" (hierarchical agglomerative) clustering, together with their "pros" and "cons".

In Section 4 we discuss detailed backtests of the various algorithms in Section 3 and subsequent sections utilizing the intraday alphas and backtesting procedure described in [Kakushadze, 2015b] by using the resultant multilevel statistical industry classifications for building heterotic risk models. The backtests unequivocally suggest that there is structure in the return time series beyond what is captured by simple principal component analysis and clustering adds

[8] Optimizing weights in alpha portfolios has its own nuances [Kakushadze and Yu, 2016c]; however, the methods we discuss here are readily portable to alpha returns as they are purely statistical. Here we backtests them (see below) on stock returns as the historical data is readily available. Alpha return time series are highly proprietary, so publishing backtests is not feasible.

value. However, clustering still cannot compete with "fundamental" industry classifications in terms of performance due to inherent out-of-sample instabilities in any purely statistical algorithm.

In Section 5 we take it a step further and give a prescription for fixing the number of clusters at each level using the methods discussed in [Kakushadze and Yu, 2016b], including eRank (effective rank) defined in [Roy and Vetterli, 2007]. We also discuss a heuristic for fixing the number of levels, albeit we empirically observe that the number of levels is not as influential as the number of clusters, at least in our backtests. We take this even further in Section 6, where we give an algorithm for improving a "fundamental" industry classification via further clustering large sub-industries (using BICS nomenclature) at the most granular level via statistical industry classification algorithms we discuss here thereby increasing granularity and improving performance. We briefly conclude in Section 7 and outline some ideas.

We give the R source code for our algorithms in Appendix A (multilevel "bottom-up" clustering, dynamical cluster numbers), Appendix B (multilevel "top-down" clustering) and Appendix C ("relaxation" clustering). Appendix D contains legalese.

2 Industry Classification

An industry classification is based on a similarity criterion: stocks' membership in "groups" or "clusters" such as sectors, industries, sub-industries, etc. – the nomenclature varies from one industry classification scheme to another. Commonly used industry classifications such as GICS, BICS, ICB, NAICS, SIC, etc., are based on fundamental/economic data (such as companies' products and services and more generally their revenue sources, suppliers, competitors, partners, etc.). Such industry classifications are essentially independent of the pricing data and, if well-built, tend to be rather stable out-of-sample as companies seldom jump industries.[9]

An industry classification can consist of a single level: N tickers labeled by $i = 1, \ldots, N$ are grouped into K "groups" – let us generically call them "clusters" – labeled by $A = 1, \ldots, K$. So, we have a map $G : \{1, \ldots, N\} \mapsto \{1, \ldots, K\}$ between stocks and "clusters".[10] More generally, we can have a hierarchy with

[9] However, there is variability in the performance of different industry classifications.

[10] Here we are assuming that each stock belongs to one and only one "cluster". Generally, this assumption can be relaxed thereby allowing for "conglomerates" that belong to multiple sub-industries, industries, sectors, etc. However, this is not required for our purposes here.

multiple levels. We can schematically represent this via: Stocks $\rightarrow$ Level-1 "Clusters" $\rightarrow$ Level-2 "Clusters" $\rightarrow \cdots \rightarrow$ Level-P "Clusters". Let us label these P levels by $\mu = 1, \ldots, P$. Level-1 is the most granular level with N stocks grouped into K_1 "clusters". The Level-1 "clusters" are in turn grouped into K_2 Level-2 "clusters", where $K_2 < K_1$, and so on, Level-P being least granular.[11] Thus, consider BICS[12] as an illustrative example, which has a 3-level hierarchy: Stocks $\rightarrow$ Sub-industries $\rightarrow$ Industries $\rightarrow$ Sectors. (Here "Sub-industries" is the most granular level, while "Sectors" is the least granular level.) So, we have: N stocks labeled by $i = 1, \ldots, N$; K sub-industries labeled by $A = 1, \ldots, K$; F industries labeled by $a = 1, \ldots, F$; and L sectors labeled by $\alpha = 1, \ldots, L$. Let G be the map between stocks and sub-industries, S be the map between sub-industries and industries, and W be the map between industries and sectors:

$$G : \{1, \ldots, N\} \mapsto \{1, \ldots, K\} \tag{1}$$

$$S : \{1, \ldots, K\} \mapsto \{1, \ldots, F\} \tag{2}$$

$$W : \{1, \ldots, F\} \mapsto \{1, \ldots, L\} \tag{3}$$

The beauty of such "binary" industry classifications (generally, with P levels) is that the "clusters" (in the case of BICS, sub-industries, industries and sectors) can be used to identify blocks (sub-matrices) in the sample correlation matrix Ψ_{ij} of stock returns.[13] E.g., for sub-industries the binary matrix $\delta_{G(i),A}$ defines such blocks.

3 Statistical Clustering

What if we do not have access to industry classifications based on fundamental data[14] or one is unavailable for the stock universe we wish to trade? Can we build an industry classification from pricing data, i.e., directly from stock returns? After all, intuitively, the time series of returns contains information about how

[11] The branches in this hierarchy tree are assumed to have equal lengths. More generally, we can have branches of nonuniform lengths. However, shorter branches can always be extended to the length of the longest branch(es) by allowing single-element (including single-stock) "clusters".

[12] Bloomberg Industry Classification System.

[13] And this is useful in constructing risk models for portfolio optimization [Kakushadze, 2015b].

[14] Commercially available industry classifications such as GICS and ICB come at nontrivial cost. The underlying SIC data is available from SEC for free, albeit only by company names, not by ticker symbols. It takes considerable effort to download this data and transform it into an actual industry classification. Alternatively, it can be purchased from commercial providers.

correlated the stocks are. Can we extract it and transform it into an industry classification?

The answer is yes, but it is tricky. The key issue is that correlations between stocks typically are highly unstable out-of-sample. A naive attempt at constructing an industry classification based on stock returns may produce an industry classification with subpar performance. Our goal here is to discuss how to mitigate the out-of-sample instability by building statistical industry classifications based on clustering quantities other than returns. But first let us discuss clustering itself.

3.1 K-means

A popular clustering algorithm is k-means [Steinhaus, 1957], [Lloyd, 1957], [Forgy, 1965], MacQueen, 1967], [Hartigan, 1975], [Hartigan and Wong, 1979], [LLoyd, 1982]. The basic idea behind k-means is to partition N observations into K clusters such that each observation belongs to the cluster with the nearest mean. Each of the N observations is actually a d-vector, so we have an $N \times d$ matrix X_{is}, $i = 1, \ldots, N$, $s = 1, \ldots, d$. Let C_a be the K clusters, $C_a = \{i | i \in C_a\}$, $a = 1, \ldots, K$. Then k-means attempts to minimize

$$g = \sum_{a=1}^{K} \sum_{i \in C_a} \sum_{s=1}^{d} (X_{is} - Y_{as})^2 \tag{4}$$

where

$$Y_{as} = \frac{1}{n_a} \sum_{i \in C_a} X_{is} \tag{5}$$

are the cluster centers (i.e., cross-sectional means),[15] and $n_a = |C_a|$ is the number of elements in the cluster C_a. In (4) the measure of "closeness" is chosen to be the Euclidean distance between points in $\mathbf{R}^d$, albeit other measures are possible.

One "drawback" of k-means is that it is not a deterministic algorithm. Generically, there are copious local minima of g in (4) and the algorithm only guarantees that it will converge to a local minimum, not the global one. Being an iterative algorithm, k-means starts with a random or user-defined set of the centers Y_{as} at the initial iteration. However, as we will see, this "drawback" actually adds value.

[15] Throughout this paper "cross-sectional" refers to "over the index i".

3.2 What to Cluster?

So, what should we cluster to construct statistical industry classifications? I.e., what should we pick as our matrix X_{is} in (4)? It is tempting to somehow use pair-wise stock correlations. However, the sample correlation matrix Ψ_{ij} computed based on the time series of stock returns is highly unstable out-of-sample.[16] So, what if we identify X_{is} with the time series of the underlying stock returns? Let R_{is} be these stock returns, where $s = 1, \ldots, d$ now is interpreted as labeling the observations in the time series (e.g., trading days). Further, for definiteness, let $s = 1$ correspond to the most recent observation. Now we can build a statistical industry classification by applying k-means to $X_{is} = R_{is}$. Intuitively this makes sense: we are clustering stocks based on how close the returns are to the centers (i.e., within-cluster cross-sectional means) of the clusters they belong to. However, this is a suboptimal choice.

Indeed, this can be understood by observing that, in the context of stock returns, a priori there is no reason why the centers Y_{as} in (5) should be computed with equal weights. We can think of the clusters C_a as portfolios of stocks, and Y_{as} as the returns for these portfolios. Therefore, based on financial intuition, we may wish to construct these portfolios with nonuniform weights. Furthermore, upon further reflection, it become evident that clustering returns make less sense than it might have appeared at first. Indeed, stock volatility is highly variable, and its cross-sectional distribution is not even quasi-normal but highly skewed, with a long tail at the higher end – it is roughly log-normal. Clustering returns does not take this skewness into account and inadvertently we might be clustering together returns that are not at all highly correlated solely due to the skewed volatility factor.

A simple solution is to cluster the normalized returns $\widetilde{R}_{is} = R_{is}/\sigma_i$, where $\sigma_i^2 = \mathrm{Var}(R_{is})$ is the serial variance. This way we factor out the skewed volatility factor. Indeed, $\mathrm{Cov}(\widetilde{R}_i, \widetilde{R}_j) = \mathrm{Cor}(\widetilde{R}_i, \widetilde{R}_j) = \Psi_{ij}$ (we suppress the index s in the serial covariance Cov and correlation Cor) is the sample correlation matrix with $|\Psi_{ij}| \leq 1$. However, as we will see below, clustering $\widetilde{R}_{is}$, while producing better results than clustering R_{is}, is also suboptimal. Here are two simple arguments why this is so.

Clusters C_a define K portfolios whose weights are determined by what we cluster. When we cluster $X_{is} = R_{is}$, the centers are $Y_{as} = \mathrm{Mean}(R_{is}|i \in C_a)$, i.e., we have equal weights $\omega_i \equiv 1$ for the aforesaid K portfolios, and we group

[16] The sample correlation matrix contains less information than the underlying time series of returns. Thus, it knows nothing about serial means of returns, only deviations from these means.

R_{is} (at each iterative step in the k-means algorithm) by how close these returns are to these equally-weighted portfolios. However, equally-weighted portfolios themselves are suboptimal. So are portfolios weighted by $\omega_i \equiv 1/\sigma_i$, which is what we get if we cluster $X_{is} = \widetilde{R}_{is}$, where the centers are $Y_{as} = \text{Mean}(R_{is}/\sigma_i | i \in C_a)$. Thus, portfolios that maximize the Sharpe ratio [Sharpe, 1994] are weighted by inverse variances:[17] $\omega_i = 1/\sigma_i^2$. We get such portfolios if we cluster $X_{is} = \widehat{R}_{is}$, where $\widehat{R}_{is} = R_{is}/\sigma_i^2$, so the centers are $Y_{as} = \text{Mean}(R_{is}/\sigma_i^2 | i \in C_a)$. Clustering $\widehat{R}_{is}$, as we will see, indeed outperforms clustering $\widetilde{R}_{is}$. Can we understand this in a simple, intuitive fashion?

By clustering $\widetilde{R}_{is} = R_{is}/\sigma_i$, we already factor out the volatility dependence. So, why would clustering $\widehat{R}_{is} = R_{is}/\sigma_i^2$ work better? Clustering $\widetilde{R}_{is}$ essentially groups together stocks that are (to varying degrees) highly correlated in-sample. However, there is no guarantee that they will remain as highly correlated out-of-sample. Intuitively, it is evident that higher volatility stocks are more likely to get uncorrelated with their respective clusters. This is essentially why suppressing by another factor or σ_i in $\widehat{R}_{is}$ (as compared with $\widetilde{R}_{is}$) leads to better performance: inter alia, it suppresses contributions of those volatile stocks into the cluster centers Y_{is}.

3.2.1 A Minor Tweak

So, we wish to cluster $\widehat{R}_{is} = R_{is}/\sigma_i^2$. There is a potential hiccup with this in practice. If some stocks have very low volatilities, we could have large $\widehat{R}_{is}$ for such stocks. To avoid any potential issues with computations, we can "smooth" this out via (MAD = mean absolute deviation):[18]

$$\widehat{R}_{is} = \frac{R_{is}}{\sigma_i u_i} \tag{6}$$

$$u_i = \frac{\sigma_i}{v_i} \tag{7}$$

$$v_i = \exp(\text{Median}(\ln(\sigma_i)) - 3\ \text{MAD}(\ln(\sigma_i))) \tag{8}$$

and for all $v_i < 1$ we set $v_i \equiv 1$. This is the definition of $\widehat{R}_{is}$ we use below (unless stated otherwise). Furthermore, Median$(\cdot)$ and MAD$(\cdot)$ above are cross-sectional.

[17] More precisely, this is the case in the approximation where the sample covariance matrix is taken to be diagonal. In the context of clustering it makes sense to take the diagonal part of the sample covariance matrix as the full sample covariance matrix is singular for clusters with $n_a > d - 1$. Even for $n_a \leq d - 1$ the sample covariance matrix, while invertible, has highly out-of-sample unstable off-diagonal elements. In contrast, the diagonal elements, i.e., sample variances σ_i^2, are much more stable, even for short lookbacks. So it makes sense to use them in defining ω_i.

[18] This is one possible tweak. Others produce similar results.

3.3 Multilevel Clustering

If we wish to construct a single-level statistical industry classification, we can simply cluster $\widehat{R}_{is}$ defined in (6) into K clusters via k-means. What if we wish to construct a multilevel statistical industry classification (see Section 2)? We discuss two approaches here, which we can refer to as "bottom-up" and "top-down".[19]

3.3.1 Bottom-Up Clustering

Say we wish to construct a P-level classification. We can construct it as a sequence: $K_1 \to K_2 \to \cdots \to K_P$ ($K_1 > K_2 > \cdots > K_P$), where we first construct the most granular level with K_1 clusters, then we cluster these K_1 clusters into fewer K_2 clusters and so on, until we reach the last and least granular level with K_P clusters. Given[20] the integers $K_1, \ldots, K_P$, the question is what to use as the returns at each step. Let these returns be $[R(\mu)]_{i(\mu),s}$ (i.e., we cluster $[R(\mu)]_{i(\mu),s}$ into K_μ clusters via k-means), where $\mu = 1, \ldots, P$, $i(\mu) = 1, \ldots, K_{\mu-1}$, and we have conveniently defined $K_0 = N$, so $i(1)$ is the same index as i. As above, we can take $[R(1)]_{is} = \widehat{R}_{is}$. What about $[R(\mu)]_{i(\mu),s}$ at higher levels $\mu > 1$? We have some choices here. Let $C_{a(\mu)} = \{i(\mu) | i(\mu) \in C_{a(\mu)}\}$, $a(\mu) = 1, \ldots, K_\mu$ be the clusters at each level μ. I.e., the index $a(\mu)$ is the same as the index $i(\mu+1)$ for $0 < \mu < P$. Then we can take (in the second line below $2 < \mu \leq P$)

$$[R(2)]_{i(2),s} = \text{Mean}(R'_{is} | i \in \{1, \ldots, N\}) \tag{9}$$

$$[R(\mu)]_{i(\mu),s} = \text{Mean}([R'(\mu-1)]_{i(\mu-1),s} | i(\mu-1) \in C_{a(\mu-1)}) \tag{10}$$

where we can take (i) $R'_{is} = R_{is}$ and $[R'(\mu)]_{i(\mu),s} = [R(\mu)]_{i(\mu),s}$, or (ii) $R'_{is} = \widehat{R}_{is}$ and $[R'(\mu)]_{i(\mu),s} = [\widehat{R}(\mu)]_{i(\mu),s}$, where (Var($\cdot$) below is the serial variance)

$$[\widehat{R}(\mu)]_{i(\mu),s} = \frac{[R(\mu)]_{i(\mu),s}}{\sigma^2_{i(\mu)}} \tag{11}$$

$$\sigma^2_{i(\mu)} = \text{Var}([R(\mu)]_{i(\mu),s}) \tag{12}$$

These two definitions produce very similar results in our backtests (see below).

3.3.2 Another Minor Tweak

In the bottom-up clustering approach we just discussed above, the higher level clusters tend to be highly correlated with each other. I.e., the corresponding clus-

[19] W.r.t. classification levels; "bottom-up" should not be confused with agglomerative clustering.

[20] We will discuss what these cluster number "should" be below.

ter returns have a prominent "market" (or "overall") mode[21] component in them. That is, averages of pair-wise ($i(\mu) \neq j(\mu)$) serial correlations $[\Psi(\mu)]_{i(\mu),j(\mu)} = \text{Cor}([R(\mu)]_{i(\mu),s}, [R(\mu)]_{j(\mu),s})$ at higher levels $\mu > 1$ are substantial.[22] To circumvent this, we can simply cross-sectionally demean the returns at higher levels, i.e., for $\mu > 1$ we substitute $[R(\mu)]_{i(\mu),s}$ by $[R(\mu)]_{i(\mu),s} - \text{Mean}([R(\mu)]_{i(\mu),s}|i(\mu) \in C_a(\mu))$. However, cross-sectional demeaning at level-1 ($\mu = 1$) leads to worse performance. Intuitively, we can understand this as follows. Demeaning at the most granular level removes the "market" mode.[23] Unlike higher-level returns $[R(\mu)]_{i(\mu),s}$, $\mu > 1$, the level-1 returns are not all that highly correlated with each other, so it pays to keep the "market" mode intact as, e.g., high-beta stocks statistically are expected to cluster together, while low-beta stocks are expected to cluster differently. So, the upshot is that we demean the returns at higher levels, but not level-1 returns.

3.3.3 Aggregating Multiple Samplings

As mentioned above, k-means is not a deterministic algorithm. Unless the initial centers are preset, the algorithm starts with random initial centers and converges to a different local minimum in each run. There is no magic bullet here: trying to "guess" the initial centers is not any easier than "guessing" where, e.g., the global minimum is. So, what is one to do? One possibility is to simply live with the fact that every run produces a different answer. The question then one must address in a given context is whether the performance in an actual application is stable from one such random run to another, or if it is all over the place. As we will see below, in our backtests, happily, the performance is extremely stable notwithstanding the fact that each time k-means produces a different looking industry classification.

So, this could be the end of the story here. However, one can do better. The idea is simple. What if we *aggregate* different industry classifications from multiple runs (or samplings) into one? The question is how. Suppose we have M runs ($M \gg 1$). Each run produces an industry classification with K clusters. Let $\Omega^r_{ia} = \delta_{G^r(i),a}$, $i = 1, \ldots, N$, $a = 1, \ldots, K$ (here $G^r : \{1, \ldots, N\} \mapsto \{1, \ldots, K\}$ is the map between the stocks and the clusters),[24] be the binary loadings matrix

[21] See, e.g., [Bouchaud and Potters, 2011], [Kakushadze and Yu, 2016c].

[22] Consequently, there is a large gap between the first $[\lambda(\mu)]^{(1)}$ and higher $[\lambda(\mu)]^{(p)}$, $p > 1$, eigenvalues of $[\Psi(\mu)]_{i(\mu),j(\mu)}$; the eigenvalues are ordered decreasingly: $[\lambda(\mu)]^{(1)} > [\lambda(\mu)]^{(2)} > \ldots$

[23] This essentially drops the 1st principal component from the spectral decomposition of Ψ_{ij}.

[24] For terminological definiteness here we focus on the level-1 clusters; it all straightforwardly applies to all levels. Also, the superscript r in Ω^r_{ia} and $G^r(i)$ is an index, not a power.

from each run labeled by $r = 1, \ldots, M$. Here we are assuming that somehow we know how to properly order (i.e., align) the K clusters from each run. This is a nontrivial assumption, which we will come back to momentarily. However, assuming, for a second, that we know how to do this, we can aggregate the loadings matrices Ω^r_{ia} into a single matrix $\widetilde{\Omega}_{ia} = \sum_{r=1}^{M} \Omega^r_{is}$. Now, this matrix does not look like a binary loadings matrix. Instead, it is a matrix of occurrence counts, i.e., it counts how many times a given stock was assigned to a given cluster in the process of M samplings. What we need to construct is a map G such that one and only one stock belongs to each of the K clusters. The simplest criterion is to map a given stock to the cluster in which $\widetilde{\Omega}_{ia}$ is maximal, i.e., where said stock occurs most frequently. A caveat is that there may be more than one such clusters. A simple criterion to resolve such an ambiguity is to assign said stock to the cluster with most cumulative occurrences (i.e., we take $q_a = \sum_{i=1}^{N} \widetilde{\Omega}_{ia}$ and assign this stock to the cluster with the largest q_a, if the aforesaid ambiguity occurs). In the unlikely event that there is still an ambiguity, we can try to do more complicated things, or we can simply assign such a stock to the cluster with the lowest value of the index a – typically, there is so much noise in the system that dwelling on such minutiae simply does not pay off.

However, we still need to tie up a loose end, to wit, our assumption that the clusters from different runs were somehow all aligned. In practice each run produces K clusters, but i) they are not the same clusters and there is no foolproof way of mapping them, especially when we have a large number of runs; and ii) even if the clusters were the same or similar, they would not be ordered, i.e., the clusters from one run generally would be in a different order than clusters from another run.

So, we need a way to "match" clusters from different samplings. Again, there is no magic bullet here either. We can do a lot of complicated and contrived things with not much to show for it at the end. A simple pragmatic solution is to use k-means to align the clusters from different runs. Each run labeled by $r = 1, \ldots, M$, among other things, produces a set of cluster centers Y^r_{as}. We can "bootstrap" them by row into a $(KM) \times d$ matrix $\widetilde{Y}_{\widetilde{a}s} = Y^r_{as}$, where $\widetilde{a} = a + (r-1)K$ takes values $\widetilde{a} = 1, \ldots, (KM)$. We can now cluster $\widetilde{Y}_{\widetilde{a}s}$ into K clusters via k-means. This will map each value of $\widetilde{a}$ to $\{1, \ldots, K\}$ thereby mapping the K clusters from each of the M runs to $\{1, \ldots, K\}$. So, this way we can align all clusters. The "catch" is that there is no guarantee that each of the K clusters from each of the M runs will be uniquely mapped to one value in $\{1, \ldots, K\}$, i.e., we may have some empty clusters at the end of the day. However, this is fine, we can simply drop such empty clusters and aggregate (via the above procedure) the smaller

number of $K' < K$ clusters. I.e., at the end we will end up with an industry classification with K' clusters, which might be fewer than the target number of clusters K. This is not necessarily a bad thing. The dropped clusters might have been redundant in the first place. Another evident "catch" is that even the number of resulting clusters K' is not deterministic. If we run this algorithm multiple times, we will get varying values of K'. However, as we will see below, the aggregation procedure improves performance in our backtests and despite the variability in K' is also very stable from run to run. In Appendix A we give the R source code for bottom-up clustering with various features we discuss above, including multilevel industry classification, the tweaks, and aggregation.[25]

3.3.4 Top-Down Clustering

Above we discussed bottom-up clustering. We can go the other way around and do top-down clustering. I.e., we can construct a P-level classification as a sequence $K_P \to K_{P-1} \to \cdots \to K_2 \to K_1$ (as before, $K_1 > K_2 > \cdots > K_P$). More conveniently, we start with the entire universe of stocks and cluster $\widehat{R}_{is}$, $i = 1, \ldots, N$, into $L_P = K_P$ clusters. At level-$(P-1)$, we cluster each level-P cluster $C_{a(P)} = \{i | i \in C_{a(P)}\}$, $a(P) = 1, \ldots, K_P$, into L_{P-1} clusters. We do this by clustering the returns $\widehat{R}_{is}$, $i \in C_{a(P)}$ via k-means into L_{P-1} clusters.[26] At level-$(P-2)$, we cluster each level-$(P-1)$ cluster $C_{a(P-1)} = \{i | i \in C_{a(P-1)}\}$, $a(P-1) = 1, \ldots, K_{P-1}$, into L_{P-2} clusters. We do this by clustering the returns $\widehat{R}_{is}$, $i \in C_{a(P-1)}$ via k-means into L_{P-2} clusters. And so on.[27] In the zeroth approximation, $K_{P-1} = L_{P-1}K_P$, $K_{P-2} = L_{P-2}K_{P-1}$, and so on, so $K_1 = K_* = \prod_{\mu=1}^{P} L_\mu$. However, if at some level-μ we have some cluster $C_{a(\mu)}$ with $n_a(\mu) \leq L_\mu$, then we leave this cluster intact and do not cluster it, i.e., we "roll it" forward unchanged. Therefore, we can have $K_1 < K_*$ at the most granular level-1. Also, instead of simply clustering via a single-sampling k-means, as above we can aggregate multiple samplings. Then at any level-μ we can end up clustering a given cluster $C_{a(\mu)}$ into L_μ or fewer clusters. Note, since here we work directly with the returns $\widehat{R}_{is}$, in contrast to the bottom-up approach, no

[25] The source code in Appendix A, Appendix B and Appendix C hereof is not written to be "fancy" or optimized for speed or in any other way. Its sole purpose is to illustrate the algorithms described in the main text in a simple-to-understand fashion. See Appendix D for some legalese.

[26] More generally, we can nonuniformly cluster each level-P cluster with its own $[L(a(P))]_{P-1}$.

[27] Note that, in contrast to bottom-up clustering, because here we are going "backwards", it is convenient to label the elements of each cluster at each level using the index i, which labels stocks.

cross-sectional demeaning is warranted at any level. In Appendix B we give the R source code for top-down clustering, including with aggregation over multiple samplings.

3.3.5 Relaxation Clustering

Instead of k-means, which is nondeterministic, we can use other types of clustering, e.g., hierarchical agglomerative clustering. Let us focus on a 1-level classification here as we can always generalize it to multilevel cases as above. So, we have N stocks, and we wish to cluster them into K clusters. If K is not preset, we can use SLINK [Sibson, 1973], etc. (see, e.g., [Murtagh and Contreras, 2011]). If we wish to preset K, then we can use a similar approach, except that it must be tweaked such that all observations are somehow squeezed into K clusters. We give the R code for one such algorithm in Appendix C. Basically, it is a relaxation algorithm which, as above, clusters $\widehat{R}_{is}$ (not R_{is}). The distance $D(i,j)$ between two d-vectors $\widehat{R}_{is}$ and $\widehat{R}_{js}$ is simply the Euclidean distance in $\mathbf{R}^d$. The initial cluster contains i_1 and j_1 with the smallest distance. If some i_2 and j_2 (such that $i_2 \neq i_1$, $i_2 \neq j_1$, $j_2 \neq i_1$ and $j_2 \neq j_1$) are such that $D(i_2,j_2)$ is smaller than the lesser of $D(i_1,\ell)$ and $D(j_1,\ell)$ for all ℓ ($\ell \neq i_1$ and $\ell \neq j_1$), then i_2 and j_2 form the second cluster. Otherwise ℓ that minimizes $D(i_1,\ell)$ or $D(j_1,\ell)$ is added to the first cluster. This is continued until there are K clusters. Once we have K clusters, we can only add to these clusters.[28]

4 Backtests

Let us backtest the above algorithms for constructing statistical industry classification by utilizing the same backtesting procedure as in [Kakushadze, 2015b]. The remainder of this subsection very closely follows most parts of Section 6 thereof.[29]

4.1 Notations

Let P_{is} be the time series of stock prices, where $i = 1,\ldots,N$ labels the stocks, and $s = 1,2,\ldots$ labels the trading dates, with $s = 1$ corresponding to the most

[28] A brute force algorithm where at each step rows and columns are deleted from the matrix $D(i,j)$ is too slow. The R source code we give in Appendix C is substantially more efficient than that. However, it is still substantially slower than the k-means based algorithms we discuss above.

[29] We "rehash" it here not to be repetitive but so that our presentation here is self-contained.

recent date in the time series. The superscripts O and C (unadjusted open and close prices) and AO and AC (open and close prices fully adjusted for splits and dividends) will distinguish the corresponding prices, so, e.g., P_{is}^{C} is the unadjusted close price. V_{is} is the unadjusted daily volume (in shares). Also, for each date s we define the overnight return as the previous-close-to-open return:

$$E_{is} = \ln\left(P_{is}^{AO}/P_{i,s+1}^{AC}\right) \tag{13}$$

This return will be used in the definition of the expected return in our mean-reversion alpha. We will also need the close-to-close return

$$R_{is} = \ln\left(P_{is}^{AC}/P_{i,s+1}^{AC}\right) \tag{14}$$

An out-of-sample (see below) time series of these returns will be used in constructing the risk models. All prices in the definitions of E_{is} and R_{is} are fully adjusted.

We assume that: i) the portfolio is established at the open[30] with fills at the open prices P_{is}^{O}; ii) it is liquidated at the close on the same day – so this is a purely intraday alpha – with fills at the close prices P_{is}^{C}; and iii) there are no transaction costs or slippage – our aim here is not to build a realistic trading strategy, but to test relative performance of various statistical industry classifications. The P&L for each stock

$$\Pi_{is} = H_{is}\left[\frac{P_{is}^{C}}{P_{is}^{O}} - 1\right] \tag{15}$$

where H_{is} are the *dollar* holdings. The shares bought plus sold (establishing plus liquidating trades) for each stock on each day are computed via $Q_{is} = 2|H_{is}|/P_{is}^{O}$.

4.2 Universe Selection

For the sake of simplicity,[31] we select our universe based on the average daily dollar volume (ADDV) defined via (note that A_{is} is out-of-sample for each date s):

$$A_{is} = \frac{1}{m}\sum_{r=1}^{m} V_{i,s+r}\ P_{i,s+r}^{C} \tag{16}$$

[30] This is a so-called "delay-0" alpha: the same price, P_{is}^{O} (or adjusted P_{is}^{AO}), is used in computing the expected return (via E_{is}) and as the establishing fill price.

[31] In practical applications, the trading universe of liquid stocks typically is selected based on market cap, liquidity (ADDV), price and other (proprietary) criteria.

We take $m = 21$ (i.e., one month), and then take our universe to be the top 2000 tickers by ADDV. To ensure that we do not inadvertently introduce a universe selection bias, we rebalance monthly (every 21 trading days, to be precise). I.e., we break our 5-year backtest period (see below) into 21-day intervals, we compute the universe using ADDV (which, in turn, is computed based on the 21-day period immediately preceding such interval), and use this universe during the entire such interval. We do have the survivorship bias as we take the data for the universe of tickers as of 9/6/2014 that have historical pricing data on http://finance.yahoo.com (accessed on 9/6/2014) for the period 8/1/2008 through 9/5/2014. We restrict this universe to include only U.S. listed common stocks and class shares (no OTCs, preferred shares, etc.) with BICS (Bloomberg Industry Classification System) sector assignments as of 9/6/2014.[32] However, as discussed in detail in Section 7 of [Kakushadze, 2015a], the survivorship bias is not a leading effect in such backtests.[33]

4.3 Backtesting

We run our simulations over a period of 5 years (more precisely, 1260 trading days going back from 9/5/2014, inclusive). The annualized return-on-capital (ROC) is computed as the average daily P&L divided by the intraday investment level I (with no leverage) and multiplied by 252. The annualized Sharpe Ratio (SR) is computed as the daily Sharpe ratio multiplied by $\sqrt{252}$. Cents-per-share (CPS) is computed as the total P&L in cents (not dollars) divided by the total shares traded.

4.4 Optimized Alphas

The optimized alphas are based on the expected returns E_{is} optimized via Sharpe ratio maximization using heterotic risk models [Kakushadze, 2015b] based on statistical industry classifications we are testing.[34] We compute the heterotic risk

[32] The choice of the backtesting window is intentionally taken to be exactly the same as in [Kakushadze, 2015b] to simplify various comparisons, which include the results therefrom.

[33] Here we are after the *relative outperformance*, and it is reasonable to assume that, to the leading order, individual performances are affected by the survivorship bias approximately equally as the construction of all alphas and risk models is "statistical" and oblivious to the universe.

[34] In [Kakushadze, 2015b] BICS is used for the industry classification. Here we simply plug in the statistical industry classification instead of BICS. In the case of a single-level industry classification, we can either add the second level consisting of the "market" with the $N \times 1$ unit matrix as the loadings matrix; or, equivalently, we can use the option `mkt.fac = T` in

model covariance matrix Γ_{ij} every 21 trading days (same as for the universe). For each date (we omit the index s) we maximize the Sharpe ratio subject to the dollar neutrality constraint:

$$\mathcal{S} = \frac{\sum_{i=1}^{N} H_i \; E_i}{\sqrt{\sum_{i,j=1}^{N} \Gamma_{ij} \; H_i \; H_j}} \to \max \tag{17}$$

$$\sum_{i=1}^{N} H_i = 0 \tag{18}$$

In the absence of bounds, the solution is given by

$$H_i = -\eta \left[\sum_{j=1}^{N} \Gamma_{ij}^{-1} \; E_j - \sum_{j=1}^{N} \Gamma_{ij}^{-1} \; \frac{\sum_{k,l=1}^{N} \Gamma_{kl}^{-1} \; E_l}{\sum_{k,l=1}^{N} \Gamma_{kl}^{-1}} \right] \tag{19}$$

where Γ^{-1} is the inverse of Γ, and $\eta > 0$ (mean-reversion alpha) is fixed via (we set the investment level I to \$20M in our backtests)

$$\sum_{i=1}^{N} |H_i| = I \tag{20}$$

Note that (19) satisfies the dollar neutrality constraint (18).

In our backtests we impose position bounds (which in this case are the same as trading bounds as the strategy is purely intraday) in the Sharpe ratio maximization:

$$|H_{is}| \leq 0.01 \; A_{is} \tag{21}$$

where A_{is} is ADDV defined in (16). In the presence of bounds computing H_i requires an iterative procedure and we use the R code in Appendix C of [Kakushadze, 2015b].

4.5 Simulation Results

Table 1 summarizes simulation results for 11 independent runs for the "bottom-up" 3-level statistical industry classification with $K_1 = 100$, $K_2 = 30$ and $K_3 = 10$ (see Subsection 3.3.1). Despite the nondeterministic nature of the underlying k-means algorithm, pleasantly, the backtest results are very stable. Table 2 summarizes simulation results for 11 independent runs for the "bottom-up" single-level statistical industry classification with the target number of clusters $K = 100$

the R function `qrm.het()` in Appendix B of [Kakushadze, 2015b], which accomplishes this internally.

based on aggregating 100 samplings (so the actual number of resultant clusters K' can be smaller than K – see Subsection 3.3.3). Again, the backtest results are very stable. Table 3 summarizes simulation results for 23 independent runs for the "bottom-up" 3-level statistical industry classification with the target number of clusters $K_1 = 100$, $K_2 = 30$ and $K_3 = 10$ based on aggregating 100 samplings (so the actual number of resultant clusters K'_μ can be smaller than K_μ, $\mu = 1, 2, 3$ – see Subsection 3.3.3). The first 15 (out of 23) runs correspond to `norm.cl.ret = F` (this corresponds to choice (i) after Equation (10) in Subsection 3.3.1), while the other 8 runs correspond to `norm.cl.ret = T` (this corresponds to choice (ii) after said Equation); see the function `qrm.stat.ind.class.all()` in Appendix A. The aforesaid stability persists to these cases as well. Table 4 summarizes the number of actual clusters in a statistical industry classification obtained via aggregating 100 samplings. The target numbers of clusters in a 3-level hierarchy are $K_1 = 100$, $K_2 = 30$ and $K_3 = 10$, as in Table 3.

Table 5 summarizes simulation results for "top-down" 3-level statistical industry classifications obtained via a single sampling in each run, with 3 runs for each L_μ. The 3-vector L_μ, $\mu = 1, 2, 3$, is defined in Subsection 3.3.4. Recall that in the zeroth approximation the number of clusters at the most granular level-1 is $K_1 = L_1 L_2 L_3$; however, the actual value can be lower due to the reasons explained in Subsection 3.3.4. Here too we observe substantial stability. Table 6 summarizes simulation results for "top-down" 3-level statistical industry classifications obtained via aggregating 100 samplings in each run, with 3 runs for each L_μ. Stability persists.

From the above results it is evident that aggregating multiple samplings on average improves both performance and stability. Furthermore, not surprisingly, decreasing granularity worsens the Sharpe ratio. 3-level classifications outperform single-level classifications.[35] Above we mentioned that clustering $\widehat{R}_{is} = R_{is}/\sigma_i^2$ outperforms clustering $\widetilde{R}_{is} = R_{is}/\sigma_i$, which in turn outperforms clustering R_{is}. Thus, a random run for the "bottom-up" 3-level classification with $K_1 = 100$, $K_2 = 30$ and $K_3 = 10$ based on clustering R_{is} using a single sampling produced a typical performance with ROC = 41.885%, SR = 15.265 and CPS = 1.889 (cf. Table 1). A random run for the "bottom-up" 3-level classification with $K_1 = 100$, $K_2 = 30$ and $K_3 = 10$ based on clustering $\widetilde{R}_{is}$ using a single sampling produced a typical performance with ROC = 42.072%, SR = 15.840 and CPS = 1.973 (cf. Table 1).[36]

[35] Also, "bottom-up" by construction uses more information than and outperforms "top-down".

[36] Table 1 is based on clustering $\widehat{R}_{is}$ defined via (6). However, clustering $\widehat{R}^*_{is} = R_{is}/\sigma_i^2$

In contrast to nondeterministic k-means based algorithms, the relaxation algorithm (Subsection 3.3.5) is completely deterministic. We run it using the code in Appendix C to obtain a 3-level classification with the target numbers of clusters $K_1 = 100$, $K_2 = 30$ and $K_3 = 10$ (as in the "bottom-up" cases, we cross-sectionally demean the level-2 and level-3 returns, but not the level-1 returns). The simulation results are sizably worse than for k-means based algorithms: ROC = 41.279%, SR = 15.487 and CPS = 1.936. How come? Intuitively, this is not surprising. All such relaxation mechanisms (hierarchical agglomerative algorithms) start with a "seed", i.e., the initial cluster picked based on some criterion. In Subsection 3.3.5 this is the first cluster containing the pair (i_1, j_1) that minimized the Euclidean distance. However, generally this choice is highly unstable out-of-sample, hence underperformance. In contrast, k-means is much more "statistical", especially with aggregation.

5 How to Fix Cluster Numbers?

Thus far we have picked the number of clusters K_μ as well as the number of levels P "ad hoc".[37] Can we fix them "dynamically"? If we so choose, here we can do a lot of complicated things. Instead, our approach will be based on pragmatism (rooted in financial considerations) and simplicity.[38] As can be surmised from Tables 2 and 3, the number of levels does not make it or break it in our context. What is more important is the number of clusters. So, suppose we have a given number of levels $P > 1$. Let us start by asking, what should K_1 (most granular level) and K_P (least granular level) be? In practice, the number of stocks $N > d - 1$, so the sample correlation matrix Ψ_{ij} is singular. (In fact, in most practical applications $N \gg d - 1$.) We can model it via statistical risk models [Kakushadze and Yu, 2016b]. These are factor models obtained by truncating the spectral decomposition of Ψ_{ij}

$$\Psi_{ij} = \sum_{a=1}^{d-1} \lambda^{(a)} \; V_i^{(a)} \; V_j^{(a)} \tag{22}$$

produces essentially the same results. Thus, a random run for the "bottom-up" 3-level classification with $K_1 = 100$, $K_2 = 30$ and $K_3 = 10$ based on clustering $\widehat{R}^*_{is}$ via aggregating 100 samplings produced a typical performance with ROC = 41.707%, SR = 16.220 and CPS = 2.091 (cf. Table 3).

[37] Here we focus on the k-means based "bottom-up" and "top-down" algorithms. As discussed above, the relaxation algorithm underperforms the k-means based algorithms.

[38] A variety of methods for fixing the number of clusters have been discussed in other contexts. See, e.g., [Rousseeuw, 1987], [Goutte *et al*, 2001], [Sugar and James, 2003], [Lleiti *et al*, 2004], [De Amorim and Hennig, 2015].

via the first $d-1$ principal components $V_i^{(a)}$ (only $d-1$ eigenvalues $\lambda^{(a)}$ are positive, $\lambda^{(1)} > \lambda^{(2)} > \ldots, \lambda^{(d-1)} > 0$, while the rest of the eigenvalues $\lambda^{(a)} \equiv 0$, $a \geq d$) to the first F principal components ($F < d-1$) and compensating the deficit on the diagonal (as $\Psi_{ii} \equiv 1$) by adding diagonal specific (idiosyncratic) variance ξ_i^2:

$$\Gamma_{ij} = \xi_i^2 \ \delta_{ij} + \sum_{a=1}^{F} \lambda^{(a)} \ V_i^{(a)} \ V_j^{(a)} \tag{23}$$

I.e., we approximate Ψ_{ij} (which is singular) via Γ_{ij} (which is positive-definite as all $\xi_i^2 > 0$ and are fixed from the requirement that $\Gamma_{ii} \equiv 1$). The question then is, what should F be? One simple ("minimization" based) algorithm for fixing F is given in [Kakushadze, 2015b]. Another, even simpler algorithm recently proposed in [Kakushadze and Yu, 2016b], is based on eRank (effective rank) defined below.[39]

5.1 Effective Rank

Thus, we simply set (here Round($\cdot$) can be replaced by floor($\cdot$) = $\lfloor \cdot \rfloor$)

$$F = \text{Round}(\text{eRank}(\Psi)) \tag{24}$$

Here eRank(Z) is the effective rank [Roy and Vetterli, 2007] of a symmetric semi-positive-definite (which suffices for our purposes here) matrix Z. It is defined as

$$\text{eRank}(Z) = \exp(H) \tag{25}$$

$$H = -\sum_{a=1}^{L} p_a \ \ln(p_a) \tag{26}$$

$$p_a = \frac{\lambda^{(a)}}{\sum_{b=1}^{L} \lambda^{(b)}} \tag{27}$$

where $\lambda^{(a)}$ are the L *positive* eigenvalues of Z, and H has the meaning of the (Shannon a.k.a. spectral) entropy [Campbell, 1960], [Yang *et al*, 2005].

The meaning of eRank(Z) is that it is a measure of the effective dimensionality of the matrix Z, which is not necessarily the same as the number L of its positive eigenvalues, but often is lower. This is due to the fact that many returns can be highly correlated (which manifests itself by a large gap in the eigenvalues) thereby further reducing the effective dimensionality of the correlation matrix.

[39] For prior works on fixing F, see, e.g., [Connor and Korajczyk, 1993] and [Bai and Ng, 2002].

5.2 Fixing K_μ

There is no magic bullet here. It just has to make sense. Intuitively, it is natural to identify the number of clusters K_P at the least granular level with the number of factors F in the context of statistical risk models.[40] In the following, we will therefore simply take

$$K_P = \text{Round}(\text{eRank}(\Psi)) \tag{28}$$

Adding more granular levels explores deeper substructures in the time series of returns based on the closeness criterion. In this regard, we can fix the number of clusters K_1 at the most granular level as follows. The average number of stocks per cluster at level-1 is $N_1 = N/K_1$ (we are being cavalier with rounding). Assume for a second that the number of stocks in each cluster is the same and equal N_1. If $N_1 > d-1$, then the sub-matrices Ψ_{ij}, $i,j \in C_{a(1)}$ (recall that $C_{a(1)}$, $a(1) = 1,\ldots,K_1$, are the level-1 clusters) are singular. For $N_1 \leq d-1$ they are nonsingular. Therefore, intuitively, it is natural to fix K_1 by requiring that $N_1 = d-1$. Restoring rounding, in the following we will set

$$K_1 = \text{Round}(N/(d-1)) \tag{29}$$

What about K_μ, $1 < \mu < P$? Doing anything overly complicated here would be overkill. Here is a simple prescription (assuming $K_1 > K_P$):[41]

$$K_\mu = \left[K_1^{P-\mu}\, K_P^{\mu-1}\right]^{\frac{1}{P-1}}, \quad \mu = 1,\ldots,P \tag{30}$$

We give the R source code for building "bottom-up" statistical industry classifications using this prescription in Appendix A. Table 7 summarized simulation results for $P = 2,3,4,5$. It is evident that the number of levels is not a driver here. The results are essentially the same as for $K_1 = 100$ (recall that $N = 2000$ and $d = 21$ in our case) in Tables 2 and 3. Table 8 isolates the K dependence and suggests that the performance peaks around $K = 100$. Again, there is no magic bullet here.[42]

[40] The number of factors F essentially measures the effective number of degrees of freedom in the underlying time series of returns R_{is}. Hence identification of K_P with this number.

[41] I.e., K_μ are (up to rounding) equidistant on the log scale. For $P = 3$ the "midpoint" $K_2 = \sqrt{K_1 K_P}$ is simply the geometric mean. With this prescription, we can further fix P via some heuristic, e.g., take maximal P such that the difference $K_{P-1} - K_P \geq \Delta$, where Δ is preset, say, $\Delta = K_P$. For $K_1 = 100$ and $K_P = 10$, this would give us $P = 4$ with $K_2 = 46$ and $K_3 = 22$.

[42] Note from Table 8 that too little granularity lowers the Sharpe ratio due to insufficient coverage of the risk space, while too much granularity lowers cents-per-share due to overtrading.

5.3 Comparisons

Let us compare the (very stable) results we obtained for statistical industry classifications with two "benchmarks": statistical risk models [Kakushadze and Yu, 2016b] and heterotic risk models with BICS used as the industry classification [Kakushadze, 2015b]. More precisely, statistical risk models in [Kakushadze and Yu, 2016b] were built based on the sample correlation matrix Ψ_{ij}, which is equivalent to basing them on normalized returns

$$\widetilde{R}_{is} = R_{is}/\sigma_i.$$

If we use the eRank based algorithm for fixing the number of statistical risk factors F, then the performance is ROC = 40.777%, SR = 14.015 and CPS = 1.957 [Kakushadze and Yu, 2016b].[43] However, as we argued above, it makes more sense to build models using $\widehat{R}_{is} = R_{is}/\sigma_i^2$. So, we should compare our results here with the statistical risk models based on $\widehat{R}_{is}$. To achieve this, we can simply replace the line `tr <- apply(ret, 1, sd)` in the R function `qrm.erank.pc(ret, use.cor = T)` in Appendix A of [Kakushadze and Yu, 2016b] by `tr <- apply(ret, 1, sd) / apply(qrm.calc.norm.ret(ret), 1, sd)`, where the R function `qrm.calc.norm.ret()` is given in Appendix A hereof. The performance is indeed better: ROC = 40.878%, SR = 14.437 and CPS = 2.018. So, the k-means based clustering algorithms still outperform statistical risk models, which implies that going beyond the F statistical factors adds value, i.e., there is more structure in the data than is captured by the principal components alone. However, statistical industry classifications still sizably underperform heterotic risk models based on BICS [Kakushadze, 2015b]:[44] ROC = 49.005%, SR = 19.230 and CPS = 2.365. Clearly, statistical industry classifications are not quite on par with industry classifications such as BICS, which are based on fundamental/economic data (such as companies' products and services and more generally their revenue sources, suppliers, competitors, partners, etc.). Such industry classifications are essentially independent of the pricing data and, if well-built, tend to be rather stable out-of-sample as companies seldom jump industries. In contrast, statistical industry classifications by nature are less stable out-of-sample. However, they can add substantial value when "fundamental" industry classifications are unavailable, including for returns other than for stocks, e.g., quantitative trading alphas [Kakushadze and Yu, 2016c].

[43] In [Kakushadze and Yu, 2016b] rounding is to 2 decimals, while here we round to 3 decimals.

[44] Here we use the results from [Kakushadze and Yu, 2016a], which slightly differ from those in [Kakushadze, 2015b], where rounding down (as opposed to simply rounding) was employed.

Finally, before we close this section, let us discuss the "top-down" classifications with dynamically determined numbers of clusters K_μ. More precisely, recall that in this case we work with the vector L_μ (see Subsection 3.3.4). The code we used in the "bottom-up" case (Appendix A) can be used in this case as well (via a parameter choice). A random (and typical) run with $P = 3$ gives ROC = 41.657%, SR = 15.897 and CPS = 2.079, while another such run with $P = 4$ gives ROC = 41.683%, SR = 15.697 and 2.073. These results are in line with our results in Table 6.

6 Hybrid Industry Classification

One application of a statistical industry classification is to use it as a means for improving a "fundamental" industry classification such as BICS, GICS, etc. Thus, a "fundamental" classification at the most granular level can have overly large sub-industries, using the BICS nomenclature for definiteness. One way to deal with such large sub-industries is to further cluster them using statistical industry classification methods discussed above. Let us illustrate this using BICS as an example.

Table 9 summarizes top 10 most populous (by stock counts) sub-industries in one of our 2000 stock backtesting portfolios. For comparison, the stock count summary across all 165 sub-industries in this sample is Min = 1, 1st Qu. = 3, Median = 8, Mean = 12.12, 3rd Qu. = 15, Max = 94, StDev = 14.755, MAD = 8.896 (see Table 4 for notations). So, we have some "large" sub-industries, which are outliers.

We can further split these large sub-industries into smaller clusters using our "bottom-up" clustering algorithm. In fact, it suffices to split them using a single-level algorithm. We give the R code for improving an existing "fundamental" industry classification using our statistical industry classification algorithm in Appendix A. The idea is simple. Let us label the sub-industries (the most granular level) in the "fundamental" industry classification via $A = 1, \ldots, K_*$. Let N_A be the corresponding stock counts. Let

$$\kappa_A = \text{Round}(N_A/(d-1)) \tag{31}$$

We then split each sub-industry with $\kappa_A > 1$ into κ_A clusters. Table 10 summarizes the simulation results for 14 runs. This evidently improves performance. Table 11 gives summaries of top 10 most populous sub-industries before and after statistical clustering based on 60 datapoints at the end of each 21-trading-day interval in our backtests (recall that we have $1260 = 60 \times 21$ trading days – see

Section 4). The average numbers of sub-industries are 165.45 before and 184.1 after clustering.

7 Concluding Remarks

In this paper we discuss all sorts of nuances in constructing statistical industry classifications. Under the hood we have clustering algorithms. However, it is precisely those nuances that make a sizable difference. E.g., if we naively cluster R_{is}, we get a highly suboptimal result compared with clustering $\widetilde{R}_{is} = R_{is}/\sigma_i$, which in turn underperforms clustering $\widehat{R}_{is} = R_{is}/\sigma_i^2$. In this regard, let us tie up a "loose end" here: what if we cluster $\overline{R}_{is} = R_{is}/\sigma_i^3$? It underperforms clustering $\widehat{R}_{is}$. Thus, a typical run for a 3-level "bottom-up" classification with target cluster numbers $K_1 = 100$, $K_2 = 30$ and $K_3 = 10$ based on clustering $\overline{R}_{is}$ and aggregating 100 samplings produces the following: ROC = 40.686, SR = 15.789 and CPS = 2.075.

So, suppressing returns R_{is} by σ_i^2 indeed appears to be optimal – for the intuitive reasons we discussed above. We saw the same in the context of statistical risk models. In this regard, it would be interesting to explore this avenue in the context of heterotic risk models [Kakushadze, 2015b] and the more general (heterotic CAPM) construction of [Kakushadze and Yu, 2016a]. In the latter framework, it would be interesting to utilize an aggregated (non-binary) matrix $\widetilde{\Omega}_{ia}$ (see Subsection 3.3.3). These ideas are outside of the scope hereof and we hope to return to them elsewhere.

Appendices

A R Code for Bottom-Up Clustering

A.1 Code for Single-Level Clustering

In this subsection we give the R source code (R Package for Statistical Computing, http://www.r-project.org) for single-level "bottom-up" clustering (see Subsection 3.3.1). The code is straightforward and self-explanatory as it simply follows the formulas and logic in Section 3. The main function is `qrm.stat.ind.class(ret, k, iter.max = 10, num.try = 100, demean.ret = F)`, which internally calls two auxiliary functions. The function `qrm.calc.norm.ret(ret)` normalizes the $N \times d$ matrix `ret` (the return time series R_{is}, $i = 1, \ldots, N$, $s = 1, \ldots, d$) following Subsection 3.2.1 and outputs $\widehat{R}_{is}$ (see Eq. (6)). The inputs of the function `qrm.calc.kmeans.ind(x, centers, iter.max)` are the same as in the built-in R function `kmeans()`, and it outputs a list: `res$ind` is the $N \times K$ binary industry classification matrix $\Omega_{ia} = \delta_{G(i),a}$, where $G : \{1, \ldots, N\} \mapsto \{1, \ldots, K\}$ maps stocks to K clusters labeled by $a = 1, \ldots, K$; `res$centers` is the $K \times d$ matrix Y_{as} of the cluster centers; `res$cluster` is an N-vector $G(i)$; and the number of clusters K is passed into this function via the argument `centers` as in `kmeans()`. The inputs of the function `qrm.stat.ind.class()` are: `ret` defined above; the target number of clusters `k`; the maximum number of k-means iterations `iter.max` (same as in `kmeans()`) with the default `iter.max = 10`, however, in all our backtests we set `iter.max = 100` (with 100% convergence rate); `num.try = 100` (default), which is the number of independent k-means samplings to be aggregated (see Subsection 3.3.3), with `num.try = 1` corresponding to no aggregation (i.e., a single k-means sampling); `demean.ret = F` (default) corresponds to taking vanilla R_{is}, while `demean.ret = T` corresponds to demeaning it cross-sectionally before running the rest of the code (see Subsection 3.3.2). The main function outputs the $N \times K'$ binary industry classification matrix ($K' \leq K$).[45]

```
qrm.calc.norm.ret <- function (ret)
{
   s <- apply(ret, 1, sd)
   u <- log(s)
   u <- u - (median(u) - 3 * mad(u))
   u <- exp(u)
```

[45] Recall from Subsection 3.3.3 that K' can be less than K unless `num.try = 1`.

```
   take <- u > 1
   u[!take] <- 1
   x <- ret / s / u
   return(x)
}

qrm.calc.kmeans.ind <- function (x, centers, iter.max)
{
   res <- new.env()
   y <- kmeans(x, centers, iter.max = iter.max)
   x <- y$cluster
   k <- nrow(y$centers)
   z <- matrix(NA, length(x), k)
   for(j in 1:k)
      z[, j] <- as.numeric(x == j)
   z <- z[, colSums(z) > 0]
   res$ind <- z
   res$centers <- y$centers
   res$cluster <- y$cluster
   return(res)
}

qrm.stat.ind.class <- function (ret, k,
   iter.max = 10, num.try = 100, demean.ret = F)
{
   if(demean.ret)
      ret <- t(t(ret) - colSums(ret))

   norm.ret <- qrm.calc.norm.ret(ret)

   for(i in 1:num.try)
   {
      res <- qrm.calc.kmeans.ind(norm.ret, k, iter.max)

      if(num.try == 1)
         return(res$ind)

      if(i == 1)
```

```
      {
         comb.cent <- res$centers
         comb.ind <- res$ind
      }
      else
      {
         comb.cent <- rbind(comb.cent, res$centers)
         comb.ind <- cbind(comb.ind, res$ind)
      }
   }

   res <- qrm.calc.kmeans.ind(comb.cent, k, iter.max)
   cl <- res$cluster
   z <- matrix(0, nrow(ret), k)

   for(i in 1:length(cl))
      z[, cl[i]] <- z[, cl[i]] + comb.ind[, i]

   q <- colSums(z)
   for(i in 1:nrow(z))
   {
      take <- z[i, ] == max(z[i, ])
      take <- take & q == max(q[take])
      ix <- 1:ncol(z)
      ix <- min(ix[take])
      z[i, ] <- 0
      z[i, ix] <- 1
   }
   z <- z[, colSums(z) > 0]
   return(z)
}
```

A.2 Code for Multilevel Clustering

In this subsection we give the R source code for building multilevel "bottom-up" statistical industry classifications (see Subsection 3.3.1).
There is only one function `qrm.stat.ind.class.all(ret, k, iter.max = 10, num.try = 100, do.demean = rep(F, length(k)), norm.cl.ret = F)`, which internally calls the main function `qrm.stat.ind.class()` from Subsection A.1 with

the same inputs `ret`, `iter.max` and `num.try`, and the following new inputs: `k` is a P-vector K_μ, $\mu = 1, \ldots, P$, where P is the number of levels (see Subsection 3.3.1); `do.demean = rep(F, length(k))` is a Boolean P-vector, which sets the input `demean.ret` in `qrm.stat.ind.class()` (in our backtests we set all elements of `do.demean` to `TRUE` except for the first one); `norm.cl.ret = F` corresponds to choice (i) right after Eq. (10), and `norm.cl.ret = T` corresponds to choice (ii) (we mostly use choice (i) in our backtest – see Section 4). The output is a list: `ind.list[[i]]` is the $N \times K_\mu$ (*not* $K_{\mu-1} \times K_\mu$) binary industry classification matrix at level `i` $= \mu$, i.e., it maps stocks to the level-μ clusters $C_{a(\mu)}$.

```
qrm.stat.ind.class.all <- function (ret, k,
   iter.max = 10, num.try = 100,
      do.demean = rep(F, length(k)), norm.cl.ret = F)
{
   ind.list <- list()

   for(i in 1:length(k))
   {
      ind.list[[i]] <- qrm.stat.ind.class(ret, k[i],
         iter.max = iter.max, num.try = num.try,
            demean.ret = do.demean[i])
      if(norm.cl.ret)
         ret <- t(ind.list[[i]]) %*% qrm.calc.norm.ret(ret)
      else
         ret <- t(ind.list[[i]]) %*% ret

      if(i > 1)
      {
         ind.list[[i]] <- ind.list[[i - 1]] %*% ind.list[[i]]
         take <- ind.list[[i]] > 0
         ind.list[[i]][take] <- 1
      }
   }
   return(ind.list)
}
```

A.3 Code for Dynamically Fixing Cluster Numbers

In this subsection we give the R source code for building multilevel "bottom-up" statistical industry classifications with the numbers of clusters fixed dynamically (see Section 5 and Subsection 5.2). The main function `qrm.stat.ind.class.dyn (ret, p, iter.max = 10, num.try = 100, top.down = F)` has the same inputs as above except: `p` is the number of levels, and when `top.down = F` it internally calls the function `qrm.stat.ind.class.all()` from Subsection A.2, while when `top.down = T` it internally calls the function `qrm.stat.class()` from Appendix B. The main function internally also calls the function `qrm.eigen(ret, calc.cor = T)`, which provides a more efficient way of computing eigenpairs of the sample covariance (when `calc.cor = F`) or correlation (when `calc.cor = T`) matrix based on `ret` than the built-in R function `eigen()` by internally calling the R function `qrm.calc.eigen.eff(ret, calc.cor = F)` from Appendix C of [Kakushadze and Yu, 2016b] (when $d \leq N + 1$). It also internally calls the within R function `qrm.calc.cov.mat(x, calc.cor = F)` (when $d > N + 1$). The output is a list `ind.list`, same as in Subsection A.2.

```
qrm.stat.ind.class.dyn <- function (ret, p,
   iter.max = 10, num.try = 100, top.down = F)
{
   k1 <- round(nrow(ret) / (ncol(ret) - 1))
   if(p > 1)
   {
      y <- qrm.eigen(ret, calc.cor = T)$values
      kp <- round(qrm.calc.erank(y, F))

      if(k1 < kp)
         p <- 1
   }

   if(p == 1)
      k <- k1
   else
   {
      q <- ((p - 1):0) / (p - 1)
      k <- round(k1^q * kp^(1 - q))
   }

   if(k[p] == 1)
      k <- k[-p]
```

```
   do.demean <- rep(T, length(k))
   do.demean[1] <- F

   if(top.down)
   {
      k1 <- c(k[-1], 1)
      k <- round(k / k1)[length(k):1]
      ind.list <- qrm.stat.class(ret, k,
         iter.max = iter.max, num.try = num.try)
   }
   else
      ind.list <- qrm.stat.ind.class.all(ret, k,
         iter.max = iter.max, num.try = num.try, do.demean = do.demean)

   return(ind.list)
}

qrm.eigen <- function (ret, calc.cor = F)
{
   if(ncol(ret) - 1 <= nrow(ret))
      return(qrm.calc.eigen.eff(ret, calc.cor = calc.cor))

   return(eigen(qrm.calc.cov.mat(ret, calc.cor = calc.cor)))
}

qrm.calc.cov.mat <- function(x, calc.cor = F)
{
   tr <- apply(x, 1, sd)
   x <- x / tr
   x <- x - rowMeans(x)
   y <- x %*% t(x) / (ncol(x) - 1)
   return(y)
}
```

A.4 Code for Hybrid Industry Classification

In this subsection we give the R source code for hybrid industry classifications discussed in Section 6. There is only one function `qrm.improve.ind.class(ret, ind, iter.max = 10, num.try = 100)`, which internally calls the main function from Subsection A.1 `qrm.stat.ind.class()` with the same inputs `ret`, `iter.max` and `num.try`,

and the following new input: `ind` is an $N \times K_*$ binary industry classification matrix corresponding to the most granular level of a "fundamental" industry classification (e.g., sub-industries in BICS). The output is an $N \times K'_*$ binary industry classification matrix `ind1`. Here $K'_* \geq K_*$. Typically $K'_* > K_*$, so we get a more granular industry classification after clustering. If $K'_* = K_*$, then `ind1` is the same as `ind`.

```
qrm.improve.ind.class <- function (ret, ind,iter.max = 10, num.try = 100)
{
   ind1 <- rep(NA, nrow(ret))
   for(i in 1:ncol(ind))
   {
      k <- round(sum(ind[, i]) / (ncol(ret)-1))
      if(k < 2)
      {
         ind1 <- cbind(ind1, ind[, i])
         next
      }
      take <- ind[, i] > 0
      x <- ret[take, ]
      y <- qrm.stat.ind.class(x, k,
         iter.max = iter.max, num.try = num.try)
      if(length(y) > sum(take))
         tmp <- matrix(0, nrow(ret), ncol(y))
      else
         tmp <- matrix(0, nrow(ret), 1)

      tmp[take, ] <- y
      ind1 <- cbind(ind1, tmp)
   }
   ind1 <- ind1[, -1]
   return(ind1)
}
```

B R Code for Top-Down Clustering

In this Appendix we give the R source code for building multilevel "top-down" statistical industry classifications (see Subsection 3.3.4): `qrm.stat.class(ret, k, iter.max = 10, num.try = 100)` internally calls `qrm.stat.ind.class()` defined in Subsection A.1 with the same inputs `ret`, `iter.max` and `num.try`, and the following new input: `k` $= (L_P, L_{P-1}, \ldots, L_2, L_1)$ is a *reversed* P-vector L_μ, $\mu = 1, \ldots, P$, defined in Subsection

3.3.4, and P is the number of levels. The output is a list `ind.list` with P members, same as in Subsection A.2.

```
qrm.stat.class <- function (ret, k, iter.max = 10, num.try = 100)
{
   k <- c(1, k)
   n <- nrow(ret)
   p <- length(k)
   ind <- list()
   ind.list <- list()
   for(lvl in 1:p)
      ind[[lvl]] <- matrix(1, n, 1)

   for(lvl in 2:p)
   {
      for(a in 1:ncol(ind[[lvl - 1]]))
      {
         take <- ind[[lvl - 1]][, a] > 0
         tmp.k <- sum(take)
         if(tmp.k <= k[lvl])
         {
            ind[[lvl]] <- cbind(ind[[lvl]], as.numeric(take))
            next
         }
         x <- matrix(ret[take, ], tmp.k, ncol(ret))
         norm.x <- qrm.calc.norm.ret(x)
         tmp.ind <- qrm.stat.ind.class(x, k[lvl],
            iter.max, num.try = num.try)
         if(length(tmp.ind) > tmp.k)
            tmp <- matrix(0, n, ncol(tmp.ind))
         else
            tmp <- matrix(0, n, 1)

         tmp[take, ] <- tmp.ind
         ind[[lvl]] <- cbind(ind[[lvl]], tmp)
      }
      ind[[lvl]] <- ind[[lvl]][, -1]
   }

   for(lvl in p:2)
      ind.list[[p - lvl + 1]] <- ind[[lvl]]
```

```
   return(ind.list)
}
```

C R Code for Relaxation Clustering

In this Appendix we give the R source code for building relaxation algorithm based multilevel statistical industry classifications (see Subsection 3.3.5). The first function `qrm.stat.clust.all(ret, k, iter.max = 10, num.try = 100, do.demean = rep(F, length(k)), norm.cl.ret = F)` is essentially the same as the function `qrm.stat.ind.class.all()` in Subsection A.2, except internally it calls the within function `qrm.stat.clust(ret, k, demean.ret = F, return.clust = F)`. The latter builds a relaxation based single-level classification with `k` clusters. The additional input is `return.clust`: when set to `TRUE`, this function outputs the N-vector $G(i)$ as opposed to the $N \times K$ binary industry classification matrix (as for the default value). Recall that $G : \{1, \ldots, N\} \mapsto \{1, \ldots, K\}$ maps stocks to clusters.

```
qrm.stat.clust.all <- function (ret, k,
   do.demean = rep(F, length(k)), norm.cl.ret = F)
{
   ind.list <- list()

   for(i in 1:length(k))
   {
      ind.list[[i]] <- qrm.stat.clust(ret, k[i],
         demean.ret = do.demean[i])
      if(norm.cl.ret)
         ret <- t(ind.list[[i]]) %*% qrm.calc.norm.ret(ret)
      else
         ret <- t(ind.list[[i]]) %*% ret

      if(i > 1)
      {
         ind.list[[i]] <- ind.list[[i - 1]] %*% ind.list[[i]]
         take <- ind.list[[i]] > 0
         ind.list[[i]][take] <- 1
      }
   }

   return(ind.list)
```

```
}

qrm.stat.clust <- function (ret, k,
   demean.ret = F, return.clust = F)
{
   calc.take <- function(n, ix, q)
   {
      q1 <- q[ix > q]
      q2 <- q[ix < q]
      take1 <- ix + (q1 - 1) * n
      take2 <- q2 + (ix - 1) * n
      take <- c(take1, take2)
      return(take)
   }

   calc.dist.mat <- function(x)
   {
      if(is.matrix(x))
         n <- nrow(x)
      else
         n <- length(x)

      y <- x %*% t(x)
      z <- matrix(diag(y), n, n)
      y <- z + t(z) - 2 * y
      take <- upper.tri(y, T)
      y[take] <- NA
      return(y)
   }

   extract.ix <- function(y)
   {
      k <- as.numeric(y[1])
      j <- trunc(k / n)
      if(j == k / n)
         i <- n
      else
      {
         i <- k - j * n
         j <- j + 1
      }
```

```
    return(c(i, j))
}

if(demean.ret)
    ret <- t(t(ret) - colSums(ret))

n <- nrow(ret)
v <- 1:n

ret <- qrm.calc.norm.ret(ret)
x <- calc.dist.mat(ret)
x <- as.vector(x)
names(x) <- as.character(1:length(x))
x <- sort(x)
y <- as.numeric(names(x))

m <- 0
count <- 0
w <- rep(0, n)
set.y <- F

while(count < n)
{
    if(m < k)
        y1 <- y
    else if(!set.y)
    {
        set.y <- T
        q <- v[w == 0]
        n1 <- length(q)
        u <- q + matrix((q - 1) * n, n1, n1, byrow = T)
        take <- upper.tri(u, T)
        u <- as.vector(u[!take])
        take <- !(y %in% u)
        y1 <- y[take]
    }
    else
    {
        q <- v[w == 0]
        take <- calc.take(n, p, q)
        take <- !(u %in% take)
```

```
      u <- u[take]
      take <- !(y %in% u)
      y1 <- y[take]
   }

   ix <- extract.ix(y1)
   q <- v[w > 0]

   if(w[ix[1]] > 0)
   {
      count <- count + 1
      w[p <- ix[2]] <- w[ix[1]]
      take <- calc.take(n, p, q)
   }
   else if(w[ix[2]] > 0)
   {
      count <- count + 1
      w[p <- ix[1]] <- w[ix[2]]
      take <- calc.take(n, p, q)
   }
   else
   {
      m <- m + 1
      count <- count + 2
      w[ix] <- m
      take <- c(calc.take(n, ix[1], q), calc.take(n, ix[2], q))
   }

   take <- c(take, ix[1] + (ix[2] - 1) * n)
   take <- !(y %in% take)
   y <- y[take]
}

if(return.clust)
   return(w)

k <- min(k, m)
z <- matrix(NA, n, k)
for(j in 1:k)
   z[, j] <- as.numeric(w == j)
```

```
    return(z)
}
```

D DISCLAIMERS

Wherever the context so requires, the masculine gender includes the feminine and/or neuter, and the singular form includes the plural and *vice versa*. The author of this paper ("Author") and his affiliates including without limitation Quantigic® Solutions LLC ("Author's Affiliates" or "his Affiliates") make no implied or express warranties or any other representations whatsoever, including without limitation implied warranties of merchantability and fitness for a particular purpose, in connection with or with regard to the content of this paper including without limitation any code or algorithms contained herein ("Content").

The reader may use the Content solely at his/her/its own risk and the reader shall have no claims whatsoever against the Author or his Affiliates and the Author and his Affiliates shall have no liability whatsoever to the reader or any third party whatsoever for any loss, expense, opportunity cost, damages or any other adverse effects whatsoever relating to or arising from the use of the Content by the reader including without any limitation whatsoever: any direct, indirect, incidental, special, consequential or any other damages incurred by the reader, however caused and under any theory of liability; any loss of profit (whether incurred directly or indirectly), any loss of goodwill or reputation, any loss of data suffered, cost of procurement of substitute goods or services, or any other tangible or intangible loss; any reliance placed by the reader on the completeness, accuracy or existence of the Content or any other effect of using the Content; and any and all other adversities or negative effects the reader might encounter in using the Content irrespective of whether the Author or his Affiliates is or are or should have been aware of such adversities or negative effects.

The R code included in Appendix A, Appendix B and Appendix C hereof is part of the copyrighted R code of Quantigic® Solutions LLC and is provided herein with the express permission of Quantigic® Solutions LLC. The copyright owner retains all rights, title and interest in and to its copyrighted source code included in Appendix A, Appendix B and Appendix C hereof and any and all copyrights therefor.

References

Bai, J. and Ng, S. (2002) Determining the number of factors in approximate factor models. *Econometrica* 70(1): 191-221.

Bouchaud, J.-P. and Potters, M. (2011) Financial applications of random matrix theory: a short review. In: Akemann, G., Baik, J. and Di Francesco, P. (eds.) *The*

Oxford Handbook of Random Matrix Theory. Oxford, United Kingdom: Oxford University Press.

Campbell, L.L. (1960) Minimum coefficient rate for stationary random processes. *Information and Control* 3(4): 360-371.

Connor, G. and Korajczyk, R.A. (1993) A Test for the Number of Factors in an Approximate Factor Model. *The Journal of Finance* 48(4): 1263-1291.

De Amorim, R.C. and Hennig, C. (2015) Recovering the number of clusters in data sets with noise features using feature rescaling factors. *Information Sciences* 324: 126-145.

Forgy, E.W. (1965) Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *Biometrics* 21(3): 768-769.

Goutte, C., Hansen, L.K., Liptrot, M.G. and Rostrup, E. (2001) Feature-Space Clustering for fMRI Meta-Analysis. *Human Brain Mapping* 13(3): 165-183.

Grinold, R.C. and Kahn, R.N. (2000) *Active Portfolio Management.* New York, NY: McGraw-Hill.

Hartigan, J.A. (1975) Clustering algorithms. New York, NY: John Wiley & Sons, Inc.

Hartigan, J.A. and Wong, M.A. (1979) Algorithm AS 136: A K-Means Clustering Algorithm. *Journal of the Royal Statistical Society, Series C (Applied Statistics)* 28(1): 100-108.

Kakushadze, Z. (2015a) Mean-Reversion and Optimization. *Journal of Asset Management* 16(1): 14-40.
Available online: http://ssrn.com/abstract=2478345.

Kakushadze, Z. (2015b) Heterotic Risk Models. *Wilmott Magazine* 2015(80): 40-55. Available online: http://ssrn.com/abstract=2600798.

Kakushadze, Z. and Yu, W. (2016a) Multifactor Risk Models and Heterotic CAPM. *The Journal of Investment Strategies* 5(4) (forthcoming). Available online: http://ssrn.com/abstract=2722093.

Kakushadze, Z. and Yu, W. (2016b) Statistical Risk Models. *The Journal of Investment Strategies* (forthcoming).
Available online: http://ssrn.com/abstract=2732453.

Kakushadze, Z. and Yu, W. (2016c) How to Combine a Billion Alphas. *Journal of Asset Management* (forthcoming).
Available online: http://ssrn.com/abstract=2739219.

Lleiti, R, Ortiz, M.C., Sarabia, L.A. and Sánchez, M.S. (2004) Selecting Variables for k-Means Cluster Analysis by Using a Genetic Algorithm that Optimises the Silhouettes. *Analytica Chimica Acta* 515(1): 87-100.

Lloyd, S.P. (1957) Least square quantization in PCM. Working Paper. Bell Telephone Laboratories, Murray Hill, NJ.

Lloyd, S.P. (1982) Least square quantization in PCM. *IEEE Transactions on Information Theory* 28(2): 129-137.

MacQueen, J.B. (1967) Some Methods for classification and Analysis of Multivariate Observations. In: LeCam, L. and Neyman, J. (eds.) *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability.* Berkeley, CA: University of California Press, pp. 281-297.

Murtagh, F. and Contreras, P. (2011) Algorithms for hierarchical clustering: An overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 2(1): 86-97.

Rousseeuw, P.J. (1987) Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis. *Journal of Computational and Applied Mathematics* 20(1): 53-65.

Roy, O. and Vetterli, M. (2007) The effective rank: A measure of effective dimensionality. In: *European Signal Processing Conference (EUSIPCO).* Poznań, Poland (September 3-7, 2007), pp. 606-610.

Sharpe, W.F. (1994) The Sharpe Ratio. *The Journal of Portfolio Management* 21(1): 49-58.

Sibson, R. (1973) SLINK: an optimally efficient algorithm for the single-link cluster method. *The Computer Journal (British Computer Society)* 16(1): 30-34.

Steinhaus, H. (1957) Sur la division des corps matériels en parties. *Bull. Acad. Polon. Sci.* 4(12): 801-804.

Sugar, C.A. and James, G.M. (2003) Finding the number of clusters in a data set: An information theoretic approach. *Journal of the American Statistical Association* 98(463): 750-763.

Yang, W., Gibson, J.D. and He, T. (2005) Coefficient rate and lossy source coding. *IEEE Transactions on Information Theory* 51(1): 381-386.

Table 1: Simulation results (11 runs) for the optimized alphas with bounds using heterotic risk models based on "bottom-up" statistical industry classifications obtained via a single sampling in each run. The numbers of clusters in a 3-level hierarchy are 100, 30 and 10. See Subsection 3.3.1 and Section 4 for details.

Run	ROC	SR	CPS
1	41.396%	16.195	2.060
2	41.572%	16.091	2.065
3	41.666%	16.318	2.070
4	41.544%	16.300	2.065
5	41.455%	16.238	2.058
6	41.731%	16.251	2.074
7	41.391%	16.238	2.057
8	41.567%	16.293	2.065
9	41.755%	16.135	2.075
10	41.627%	16.122	2.068
11	41.569%	16.260	2.065

Table 2: Simulation results (11 runs) for the optimized alphas with bounds using heterotic risk models based on "bottom-up" statistical industry classifications obtained via aggregating 100 samplings in each run. The target number of clusters for a single level is 100. See Subsection 3.3.3 and Section 4 for details.

Run	ROC	SR	CPS
1	41.907%	16.427	2.103
2	41.912%	16.210	2.100
3	41.774%	16.227	2.091
4	41.811%	16.295	2.094
5	41.832%	16.263	2.092
6	42.047%	16.102	2.109
7	41.839%	16.242	2.098
8	41.966%	16.027	2.104
9	41.841%	15.941	2.096
10	41.755%	16.131	2.093
11	41.775%	16.284	2.093

Table 3: Simulation results (23 runs) for the optimized alphas with bounds using heterotic risk models based on statistical industry classifications obtained via aggregating 100 samplings in each run. The target numbers of clusters in a 3-level hierarchy are 100, 30 and 10. See Subsection 3.3.3 and Section 4 for details. The first 15 runs correspond to `norm.cl.ret = F`, the other 8 runs correspond to `norm.cl.ret = T`; see the function `qrm.stat.ind.class.all()` in Appendix A.

Run	ROC	SR	CPS
1	42.181%	16.565	2.113
2	41.728%	16.314	2.092
3	41.895%	16.419	2.097
4	41.958%	16.350	2.103
5	42.034%	16.373	2.106
6	41.700%	16.149	2.093
7	42.134%	16.055	2.112
8	42.113%	16.150	2.109
9	41.586%	16.288	2.083
10	41.808%	16.267	2.094
11	41.925%	16.168	2.099
12	41.861%	16.228	2.096
13	41.766%	16.223	2.093
14	41.877%	16.331	2.095
15	42.148%	16.217	2.112
16	41.895%	16.240	2.099
17	41.857%	16.252	2.099
18	41.777%	16.169	2.092
19	41.886%	16.341	2.101
20	41.851%	16.207	2.094
21	42.266%	16.144	2.119
22	41.769%	16.205	2.093
23	42.083%	16.095	2.110

Table 4: Summaries of the actual numbers of clusters in a statistical industry classification obtained via aggregating 100 samplings. The target numbers of clusters in a 3-level hierarchy are 100, 30 and 10. The summaries are based on 60 data points corresponding to sixty 21-trading-day intervals in the 1,260 trading-day backtesting period. See Subsection 3.3.3 and Section 4 for details. 1st Qu. = 1st Quartile, 3rd Qu. = 3rd Quartile, StDev = standard deviation, MAD = mean absolute deviation. The 100 samplings correspond to the run reported in the last row of Table 3.

Level	Min	1st Qu.	Median	Mean	3rd Qu.	Max	StDev	MAD
1	87	93	94	93.95	96	99	2.33	1.48
2	20	24	25	24.93	26	28	1.91	1.48
3	6	8	9	8.58	9	10	0.93	1.48

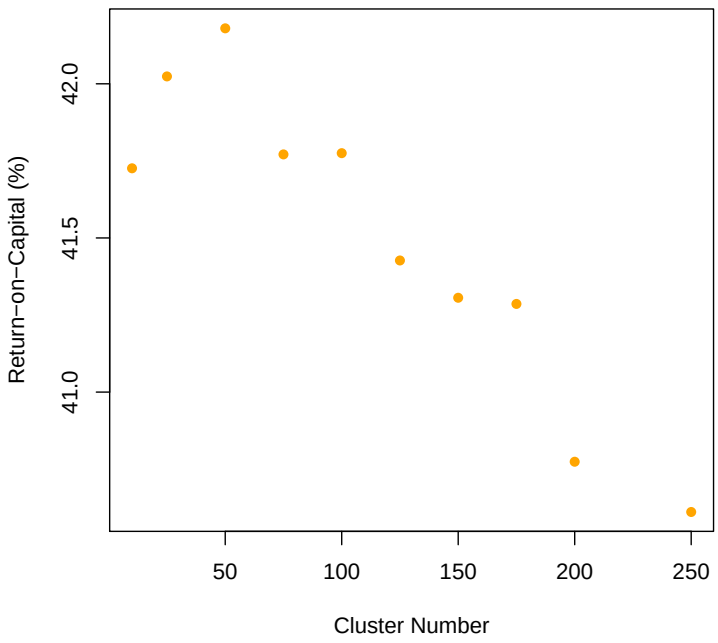

Figure 1. Graph of the values of the return-on-capital (ROC) in percent from Table 8 vs. the target number of clusters K (as defined in said table).

Table 5: Simulation results for the optimized alphas with bounds using heterotic risk models based on "top-down" 3-level statistical industry classifications obtained via a single sampling in each run, with 3 runs for each choice of the 3-vector $\widetilde{L}_\mu = (L_3, L_2, L_1)$, which is the *reverse* of the 3-vector L_μ, $\mu = 1, 2, 3$, defined in Subsection 3.3.4. Also see Section 4 for details.

Run	$\widetilde{L}_\mu$	ROC	SR	CPS
1	(10,5,5)	40.637%	16.502	2.055
2	(10,5,5)	40.880%	16.511	2.070
3	(10,5,5)	40.902%	16.684	2.075
1	(10,5,3)	41.278%	16.274	2.077
2	(10,5,3)	41.274%	16.342	2.076
3	(10,5,3)	41.044%	16.248	2.063
1	(10,3,4)	41.334%	16.046	2.071
2	(10,3,4)	41.442%	16.236	2.077
3	(10,3,4)	41.343%	16.130	2.071
1	(10,3,3)	41.590%	16.102	2.074
2	(10,3,3)	41.620%	16.029	2.072
3	(10,3,3)	41.654%	16.048	2.076
1	(10,2,3)	41.553%	15.724	2.054
2	(10,2,3)	42.046%	16.027	2.081
3	(10,2,3)	41.765%	15.665	2.066
1	(10,2,2)	42.144%	15.598	2.076
2	(10,2,2)	41.925%	15.516	2.063
3	(10,2,2)	42.007%	15.553	2.066

Table 6: Simulation results for the optimized alphas with bounds using heterotic risk models based on "top-down" 3-level statistical industry classifications obtained via aggregating 100 samplings in each run, with 3 runs for each $\widetilde{L}_\mu = (L_3, L_2, L_1)$, which is the *reverse* of the 3-vector L_μ, $\mu = 1, 2, 3$, defined in Subsection 3.3.4. Also see Section 4 for details.

Run	$\widetilde{L}_\mu$	ROC	SR	CPS
1	(10,5,5)	41.412%	16.550	2.098
2	(10,5,5)	41.478%	16.413	2.097
3	(10,5,5)	41.251%	16.401	2.092
1	(10,5,3)	41.696%	16.057	2.095
2	(10,5,3)	41.597%	16.157	2.093
3	(10,5,3)	41.730%	15.975	2.100
1	(10,3,4)	41.680%	15.979	2.085
2	(10,3,4)	41.643%	15.903	2.078
3	(10,3,4)	41.794%	16.023	2.092
1	(10,3,3)	42.078%	15.975	2.090
2	(10,3,3)	41.897%	15.962	2.083
3	(10,3,3)	41.785%	15.904	2.078
1	(10,2,3)	41.817%	15.618	2.063
2	(10,2,3)	41.964%	15.693	2.071
3	(10,2,3)	41.705%	15.598	2.062
1	(10,2,2)	42.080%	15.489	2.065
2	(10,2,2)	41.865%	15.433	2.059
3	(10,2,2)	41.987%	15.468	2.063

Table 7: Simulation results for the optimized alphas with bounds using heterotic risk models based on "bottom-up" P-level statistical industry classifications obtained via aggregating 100 samplings in each run, with multiple (3 or 4) runs for each P. The cluster numbers K_μ, $\mu = 1, \ldots, P$, are determined dynamically via the algorithm of Subsection 5.2. Also see Section 4 for backtesting details.

Run	P	ROC	SR	CPS
1	2	41.746%	16.152	2.093
2	2	41.745%	16.004	2.091
3	2	42.029%	16.007	2.104
1	3	41.921%	16.309	2.103
2	3	41.911%	16.090	2.098
3	3	41.813%	16.455	2.094
1	4	41.887%	16.317	2.096
2	4	42.273%	16.168	2.117
3	4	41.850%	16.115	2.099
1	5	42.095%	16.359	2.112
2	5	41.891%	16.178	2.102
3	5	41.961%	16.278	2.101
4	5	42.152%	16.237	2.111

Table 8: Simulation results for the optimized alphas with bounds using heterotic risk models based on "bottom-up" statistical industry classifications obtained via aggregating 100 samplings in each run. K is the target number of clusters for a single level. The $K = 100$ entry is the same as the last row in Table 2. See Subsection 3.3.3 and Section 4 for details. Also see Figures 1, 2 and 3.

K	ROC	SR	CPS
10	41.726%	14.853	2.027
25	42.024%	15.395	2.065
50	42.180%	15.941	2.094
75	41.771%	16.115	2.085
100	41.775%	16.284	2.093
125	41.427%	16.205	2.080
150	41.306%	16.337	2.073
175	41.286%	16.456	2.076
200	40.774%	16.276	2.047
250	40.611%	16.248	2.032

Table 9: Summary of stock counts (first column) for the 10 (out of 165 in this sample) most populous BICS sub-industries (most granular level, second column) for 2000 stocks in our backtests for a randomly chose date. We also show the corresponding BICS industries (less granular level, third column) and BICS sectors (least granular level, fourth column). The nomenclature is shown as it appears in BICS.

#(stocks)	BICS Sub-industry	BICS Industry	BICS Sector
94	Banks	Banking	Financials
94	REIT	Real Estate	Financials
74	Exploration & Production	Oil, Gas & Coal	Energy
52	Semiconductor Devices	Semiconductors	Technology
50	Application Software	Software	Technology
47	Utility Networks	Utilities	Utilities
46	Telecom Carriers	Telecom	Communications
45	Oil & Gas Services & Equip	Oil, Gas & Coal	Energy
44	Biotech	Biotech & Pharma	Health Care
38	Specialty Pharma	Biotech & Pharma	Health Care

Table 10: Simulation results (14 runs) for the optimized alphas with bounds using heterotic risk models based on hybrid industry classifications (see Section 6) using statistical industry classifications based on aggregating 100 samplings in each run.

Run	ROC	SR	CPS
1	49.214%	19.447	2.380
2	49.260%	19.571	2.387
3	49.224%	19.528	2.386
4	49.126%	19.522	2.379
5	49.217%	19.506	2.384
6	49.163%	19.547	2.382
7	49.204%	19.517	2.384
8	49.138%	19.482	2.381
9	49.247%	19.529	2.385
10	49.195%	19.504	2.384
11	49.191%	19.550	2.383
12	49.216%	19.578	2.383
13	49.212%	19.519	2.385
14	49.307%	19.537	2.389

Table 11: Summaries of the numbers of top 10 most populous: (i) BICS sub-industries before clustering (first 10 rows); and (ii) resultant clusters at the same level in a hybrid industry classification after clustering (last 10 rows). Each summary is over 60 datapoints (see Section 6).

Order	Min	1st Qu.	Median	Mean	3rd Qu.	Max	StDev	MAD
1	89	93	94	93.8	95	98	1.964	1.483
2	81	89	91	90.37	92	96	3.103	2.965
3	63	69	73	72.28	75	81	4.388	4.448
4	50	54	56	56.5	59	65	3.327	3.706
5	49	50	51	51.65	53	56	1.745	1.483
6	46	49	49	49.18	50	51	1.255	1.483
7	44	45.75	47	47.22	49	50	1.869	2.965
8	41	44	45.5	45.58	47	50	1.977	2.224
9	36	38.75	40	40.18	41	46	2.318	1.483
10	34	37	37	37.77	39	45	1.925	1.483
1	44	51.75	57	58.82	64	85	8.981	8.896
2	33	45	49	49	53.25	69	6.857	5.93
3	32	39.75	44	43.28	46.25	56	4.854	4.448
4	31	37	40	40.22	44	47	4.030	4.448
5	31	36	37	37.85	40.25	46	3.473	2.965
6	29	33.75	35	35.42	37	45	3.285	2.965
7	29	31.75	34	33.78	36	41	2.964	2.965
8	28	30	32	31.85	34	38	2.543	2.965
9	26	29	30.5	30.67	32.25	35	2.252	2.224
10	25	28	29	29.2	31	35	2.122	2.965

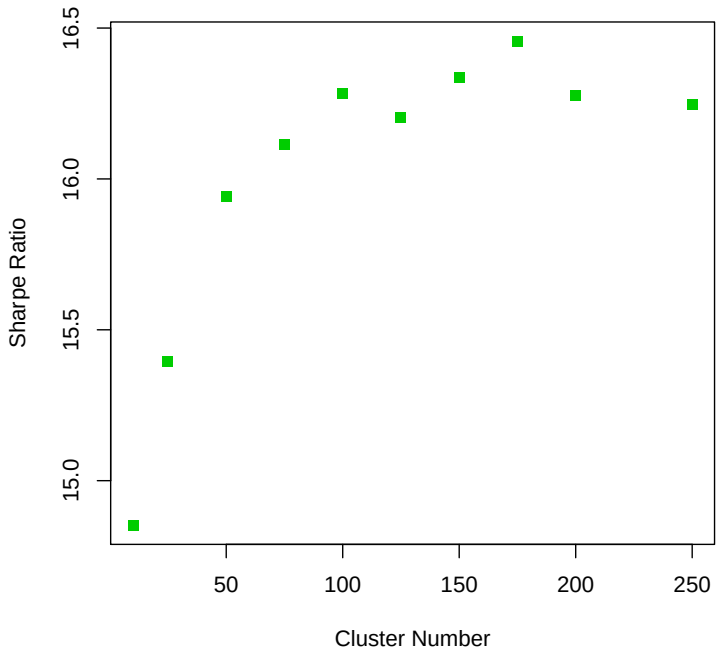

Figure 2. Graph of the values of the Sharpe ratio (SR) from Table 8 vs. the target number of clusters K (as defined in said table).

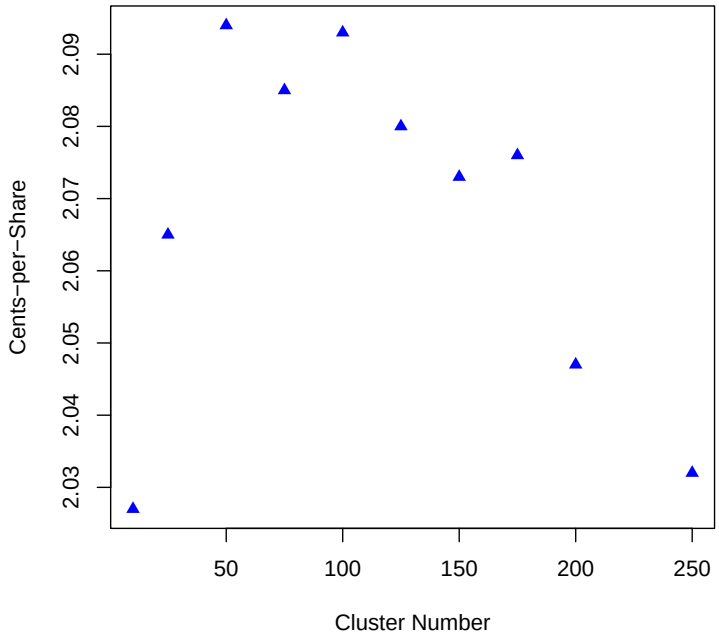

Figure 3. Graph of the values of cents-per-share from Table 8 vs. the target number of clusters K (as defined in said table).