ESTIMATION OF THEORY-IMPLIED CORRELATION MATRICES

Marcos López de Prado

This version: November 10, 2019

Marcos López de Prado is CIO of True Positive Technologies, LP, in New York, NY, and Professor of Practice at Cornell University's School of Engineering, in Ithaca, NY. E-mail: mldp@truepositive.com. Some of the methods and systems described in this paper are covered and protected under U.S. Patent Application No. 62/899,163. All rights reserved.

ESTIMATION OF THEORY-IMPLIED CORRELATION MATRICES

ABSTRACT

Correlation matrices are ubiquitous in finance. Some key applications include portfolio construction, risk management, and factor/style analysis. Correlation matrices are usually estimated from historical empirical observations or derived from historically estimated factors. It is widely acknowledged that empirical correlation matrices: (a) have poor numerical properties that lead to unreliable estimators; and (b) have poor predictive power. Additionally, factor-based correlation matrices have their own caveats. In particular, estimated factors are typically non-hierarchical and do not allow for interactions at different levels. This contravenes the fact that financial instruments typically exhibit a nested cluster structure (e.g., MSCI's GICS levels 1-4).

This paper introduces a machine learning (ML) algorithm to estimate forward-looking correlation matrices implied by economic theory. Given a particular theoretical representation of the hierarchical structure that governs a universe of securities, the method fits the correlation matrix that complies with that theoretical representation of the future. This particular use case demonstrates how, contrary to popular perception, ML solutions are not black-boxes, and can be applied effectively to develop and test economic theories.

Keywords: Hierarchical clustering, economic classification, correlation estimation, knowledge graph.

JEL Classification: G0, G1, G2, G15, G24, E44.
AMS Classification: 91G10, 91G60, 91G70, 62C, 60E.

1. MOTIVATION

Correlation matrices are ubiquitous in finance. Some key applications include portfolio construction, risk management, and factor/style analysis (López de Prado [2016]). A common approach to building a correlation matrix is to directly estimate the pairwise correlation across the securities that form an investment universe. Empirical correlation matrices pose a couple of problems. First, they tend to be numerically ill-conditioned, as a result of estimating a large number of variables on an insufficient number of observations. These numerical problems have traditionally been addressed by shrinking the correlation matrix (Ledoit and Wolf [2003]). Shrinkage eliminates some of the noise in the correlation matrix, at the cost of diluting much of the signal. A better approach is to shrink only the eigenvalues associated with noise (Potter et al. [2005], López de Prado [2019b]).

A second problem of empirical correlation matrices is that they are purely observation driven, and do not impose a structural view of the investment universe, supported by economic theory. Some practitioners see this theory-agnostic attribute in a positive light. Still, it would be desirable to count with correlation matrices that incorporate a user-defined structure, informed by economic theory. In particular, theory-implied correlation matrices are useful for the development and testing of economic hypothesis (an essential step of the scientific method), and for expressing *forward-looking* views of the world (in contrast to the backward-looking views of empirical correlations).

Academics and practitioners often work with factor-based correlation matrices, where the economic factors impose a theoretical structure. Factor-based correlation matrices pose problems of their own: (a) the factors are not robustly estimated, because they are derived from empirical correlation matrices, subject to the same numerical ill-conditioning; (b) the factors do not represent forward-looking views, because they are estimated from historical series; and (c) factor-based correlation matrices are derived from structural regression models that fail to recognize the complex hierarchical interactions among securities. For example, MSCI's Global Industry Classification Standard (GICS) classifies investment universes in terms of four nested levels, which regression specifications cannot model properly (López de Prado [2019a]).

In summary, empirical correlation matrices are unstable and backward-looking, and factor-based correlation matrices do not fully address these concerns, while failing to grasp the complex interactions among securities. This paper introduces a machine learning (ML) algorithm to estimate forward-looking correlation matrices implied by economic theory. Given a particular theoretical representation of the hierarchical structure that governs a universe of securities, the method fits the correlation matrix that complies with that theoretical representation. As a side note, this particular use case demonstrates how, contrary to popular perception, ML solutions are not necessarily black-boxes, and can be applied effectively to develop and test economic theories (López de Prado [2018]).

2. PROBLEM STATEMENT

We wish to compute the correlation matrix implied by an economic theory that binds the securities within an investment universe. The economic theory provides the fundamental structure that will be fit on the empirical data, akin to the specification or functional form that a regression model fits on observations. Accordingly, the theory-implied correlation matrix will conform to the economic theory, even if that means departing from the empirical correlation matrix. This departure can be justified in terms of: (a) the need to correct for spurious correlations, driven by noise rather than signal; (b) the desire to make a forward-looking statement, rather than the backward-looking views expressed by historical correlations.

2. INPUT DATA

As described in the problem statement, our starting point is: (1) an economic theory; and (2) empirical data. The economic theory is represented is terms of a *tree graph*. The tree can have as many levels as needed, with one or more leaves per branch, and some branches may include more levels than other branches. For instance, GICS classifies stocks in terms of sub-industry, industry, industry group, and sector. Alternative economic classifications may group stocks according to a knowledge graph (Liu et al. [2018]). A knowledge graph links companies on the basis of a wide ranging collection of criteria, such as their business ties, supply chain, competitors, industrial sectors, shared ownership, etc. Exhibit 1 illustrates part of a possible tree graph structure. The columns are ordered bottom-up, with the leftmost column corresponding to the terminal leaves, and the rightmost column corresponding to the tree's root.

[EXHIBIT 1 HERE]

The empirical data is represented in terms of an *empirical correlation matrix*, estimated on historical observations. This empirical correlation matrix must be symmetric, and include a main diagonal of 1s. However, the empirical correlation matrix does not need to be invertible, positive definite or non-singular. This is extremely useful in practice, because it allows us to derive a correlation matrix on N variables even if we count with fewer than $\frac{1}{2}N(N-1)$ observations, or in the case where different variables miss observations at different points in time.

3. THE TIC ALGORITHM

The theory-implied correlation (TIC) algorithm fits a tree graph structure to an empirical correlation matrix in three steps.

3.1. DERIVATION OF THE LINKAGE STRUCTURE

The first step is to fit the theoretical tree graph structure on the evidence presented by the empirical correlation matrix. The tree graph establishes what variables (e.g., stocks) are related to each other, and the empirical correlation matrix informs how distant is that relationship. We can blend both inputs together, by endowing the tree graph with a distance metric, derived from the empirical correlation matrix. The result is a binary tree that sequentially clusters two items together, while measuring how closely together the two items are, until all items are subsumed within the same cluster. This type of agglomerative binary tree is sometimes known as a dendrogram.

Code snippet 1 implements this step. Function `getLinkage_corr` receives two arguments, `tree` (the tree graph) and `corr` (the empirical correlation matrix), and returns a linkage object, `link0`. The linkage object characterizes the dendrogram. It is composed of one row per cluster, and four columns: (1) and (2) are the integer numbers that identify the two items clustered together; (3) reports the distance between those two items; and (4) reports the number of original variables subsumed into that cluster. Integers in the range $[0, N-1]$ identify the original variables, and integers in the range $[N, 2N-1]$ identify the binary agglomerative clusters.

Function `getLinkage_corr` begins by adding a top level to the tree, if that top level is missing. This ensures that all variables end up subsumed into a single cluster. Empirical correlations do not satisfy the properties of a distance metric. A proper distance is derived from the correlation matrix by performing the transformation

$$d_{i,j} = \sqrt{\frac{1}{2}\left(1-\rho_{i,j}\right)}$$

With that information, we can agglomerate items into clusters. Within each level, the tree graph restricts what items can be clustered together. Function `getLinkage_corr` enforces that by clustering together items in close proximity, but only if they belong to the same category. The clustering is performed by function `sch.linkage`.

If all variables depend directly from a single category, then the theory-implied correlation matrix will be the same as the empirical correlation matrix. That does not mean that the more structure is imposed by the economic theory, the more the theory-implied correlation matrix will depart from the empirical correlation matrix. The degree of departure will depend of how much the tree graph forces the clustering of variables that the empirical correlation deems distant.

```
import numpy as np
import scipy.spatial.distance as ssd
import scipy.cluster.hierarchy as sch
#------------------------------------------------------------------------------
def getLinkage_corr(tree,corr):
    if len(np.unique(tree.iloc[:,-1]))>1:tree['All']=0 # add top level
    lnk0=np.empty(shape=(0,4))
    lvls=[[tree.columns[i-1],tree.columns[i]] for i in xrange(1,tree.shape[1])]
    dist0=((1-corr)/2.)**.5 # distance matrix
    items0=dist0.index.tolist() # map lnk0 to dist0
    for cols in lvls:
        grps=tree[cols].drop_duplicates(cols[0]).set_index(cols[0]).groupby(cols[1])
        for cat,items1 in grps:
            items1=items1.index.tolist()
            if len(items1)==1: # single item: rename
                items0[items0.index(items1[0])]=cat
                dist0=dist0.rename({items1[0]:cat},axis=0)
                dist0=dist0.rename({items1[0]:cat},axis=1)
                continue
            dist1=dist0.loc[items1,items1]
            lnk1=sch.linkage(ssd.squareform(dist1,force='tovector',
                checks=(not np.allclose(dist1,dist1.T))),
                optimal_ordering=True) # cluster that cat
            lnk_=linkClusters(lnk0,lnk1,items0,items1)
            lnk0=np.append(lnk0,lnk_,axis=0)
            items0+=range(len(items0),len(items0)+len(lnk_))
            dist0=updateDist(dist0,lnk0,lnk_,items0)
            # Rename last cluster for next level
        items0[-1]=cat
        dist0.columns=dist0.columns[:-1].tolist()+[cat]
        dist0.index=dist0.columns
    lnk0=np.array(map(tuple,lnk0),dtype=[('i0',int),('i1',int), \
        ('dist',float),('num',int)])
    return lnk0
```

Snippet 1 – Derivation of the linkage structure

Because function `sch.linkage` clusters together items within the same category, the resulting linkage object offers only a partial view of the global linkage object. Function `linkClusters` corrects that by transforming the partial linkage object into a subset of the global one. Code snippet 2 performs this operation.

```
def linkClusters(lnk0,lnk1,items0,items1):
    # transform partial link1 (based on dist1) into global link0 (based on dist0)
    nAtoms=len(items0)-lnk0.shape[0]
    lnk_=lnk1.copy()
    for i in xrange(lnk_.shape[0]):
        i3=0
        for j in xrange(2):
            if lnk_[i,j]<len(items1):
                lnk_[i,j]=items0.index(items1[int(lnk_[i,j])])
            else:
                lnk_[i,j]+=-len(items1)+len(items0)
            # update number of items
            if lnk_[i,j]<nAtoms:i3+=1
            else:
                if lnk_[i,j]-nAtoms<lnk0.shape[0]:
                    i3+=lnk0[int(lnk_[i,j])-nAtoms,3]
                else:
                    i3+=lnk_[int(lnk_[i,j])-len(items0),3]
        lnk_[i,3]=i3
    return lnk_
```

Snippet 2 – Transforming a partial link into a global one

Once all variables within a category have been clustered together, we need to update the distance matrix, by adding the new clusters as variables. This is a necessary step, because the upper next level will use these distances to cluster together items within the same higher category. This task is carried out by function `updateDist`, implemented in code snippet 3. The distance column associated with a new cluster is defined by the linkage criterion. By default, function `updateDist` computes that distance column as the weighted average of the two distance columns associated with the clustered items. The weightings are given by the number of original variables subsumed within each item clustered. The user can apply alternative linkage criterions, such as those implemented in Scipy.[1]

```
def updateDist(dist0,lnk0,lnk_,items0,criterion=None):
    # expand dist0 to incorporate newly created clusters
    nAtoms=len(items0)-lnk0.shape[0]
    newItems=items0[-lnk_.shape[0]:]
```

[1] See https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.linkage.html

```
  for i in xrange(lnk_.shape[0]):
    i0,i1=items0[int(lnk_[i,0])],items0[int(lnk_[i,1])]
    if criterion is None:
      if lnk_[i,0]<nAtoms:w0=1.
      else:w0=lnk0[int(lnk_[i,0])-nAtoms,3]
      if lnk_[i,1]<nAtoms:w1=1.
      else:w1=lnk0[int(lnk_[i,1])-nAtoms,3]
      dist1=(dist0[i0]*w0+dist0[i1]*w1)/(w0+w1)
    else:dist1=criterion(dist0[[i0,i1]],axis=1) # linkage criterion
    dist0[newItems[i]]=dist1 # add column
    dist0.loc[newItems[i]]=dist1 # add row
    dist0.loc[newItems[i],newItems[i]]=0. # main diagonal
    dist0=dist0.drop([i0,i1],axis=0)
    dist0=dist0.drop([i0,i1],axis=1)
  return dist0
```
Snippet 3 – Updating the distance matrix

To summarize, function `getLinkage_corr` fits the structure imposed by the economic theory (represented by the tree graph) with the empirical evidence (represented by the empirical correlation matrix). The result is a dendrogram, characterized by a linkage object. Note that the dendrogram is not the tree graph. The tree graph could have one or more leaves per branch, whereas the dendrogram always has two items per cluster. The tree graph could have an unlimited number of levels, whereas the dendrogram always has $N-1$ clusters. The tree graph did not incorporate a notion of distance, whereas the dendrogram does.

3.2. IMPLIED CORRELATION

The second step in the TIC algorithm is to derive a correlation matrix from the linkage object returned by the first step. We know that the main diagonal of the correlation matrix is composed of 1s. The off-diagonal elements are defined by the clusters in the linkage object. For example, in a 2-by-2 correlation matrix, there is only one possible clustering, and the value of the only off-diagonal element is a function of the distance between these two items. In a 3-by-3 correlation matrix, a first cluster is formed by combining two variables, and a second cluster is formed by combining the first cluster with the remaining (third) variable.

Code snippet 4 implements this second step. Function `link2corr` receives the linkage object from the first step, and outputs the correlation matrix associated with that linkage object. For each cluster in the linkage object, function `getAtoms` gets all the original variables subsumed into that item. Then, the off-diagonal correlations between the original variables in the two items is computed as

$$\rho_{i,j} = 1 - 2d_{i,j}^2$$

```
def getAtoms(lnk,item):
  # get all atoms included in an item
  anc=[item]
  while True:
    item_=max(anc)
    if item_<=lnk.shape[0]:break
    else:
      anc.remove(item_)
```

```
            anc.append(lnk['i0'][item_-lnk.shape[0]-1])
            anc.append(lnk['i1'][item_-lnk.shape[0]-1])
    return anc
#------------------------------------------------------------------------------
def link2corr(lnk,lbls):
    # derive the correl matrix associated with a given linkage matrix
    corr=pd.DataFrame(np.eye(lnk.shape[0]+1),index=lbls,columns=lbls,
                  dtype=float)
    for i in range(lnk.shape[0]):
        x=getAtoms(lnk,lnk['i0'][i])
        y=getAtoms(lnk,lnk['i1'][i])
        corr.loc[lbls[x],lbls[y]]=1-2*lnk['dist'][i]**2 # off-diagonal values
        corr.loc[lbls[y],lbls[x]]=1-2*lnk['dist'][i]**2 # symmetry
    return corr
```
Snippet 4 – Derivation of the correlation matrix associated with a linkage object

3.3. DE-NOISING

The correlation matrix derived from the linkage object may not be definite positive, or it may have a high condition number. One possibility is to apply the covariance de-noising procedures explained in Potter et al. [2005]. See López de Prado [2019c] for details, alternative implementations, and examples.

4. EXPERIMENTAL RESULTS

On the constituents of the Standard & Poors 500 as of September 30, 2019, we have downloaded from Bloomberg the most recent five years of daily returns. Exhibit 2 displays the dendrogram associated with the empirical correlation matrix.

[EXHIBIT 2 HERE]

For each of these stocks, we have also collected MSCI's GICS four-level hierarchical classification. GICS classifies companies quantitatively and qualitatively. Each company is assigned a single GICS classification at the sub-industry level according to its principal business activity. MSCI uses revenues as a key factor in determining a firm's principal business activity. Earnings and market perception are also recognized as important and relevant information for classification purposes, and are taken into account during the annual review process.[2]

4.1. TIC ESTIMATION

Code snippet 5 shows how snippets 1 to 4 are instantiated for this example. The definition for function `deNoiseCov` can be found in López de Prado [2019c].

```
import numpy as np,pandas as pd
from marcenkoPastur import deNoiseCov
gics=pd.read_csv(path+'SPX_GICS.csv')
corr=pd.read_csv(path+'SPX_Cov.csv',index_col=0).corr()
lnk0=getLinkage_corr(gics,corr)
corr0=link2corr(lnk0,corr.index)
corr1=deNoiseCov(corr0,q=10.,bWidth=.01,minEVal=0)
```
Snippet 5 – Running the TIC algorithm

[2] Additional details may be found at https://www.msci.com/gics

Exhibit 3 plots the dendrogram associated with the TIC matrix. Exhibit 4 plots the TIC matrix after de-noising and clustering. Exhibit 5 plots the empirical correlation matrix, where its rows and columns follow the same order at the clustered the TIC matrix (to facilitate the comparison between the two).

[EXHIBIT 3 HERE]

[EXHIBIT 4 HERE]

[EXHIBIT 5 HERE]

The values of the TIC matrix appear to change more smoothly and less abruptly across clusters, compared to the values of the empirical correlation matrix. This is a desirable trait, as we would expect that the correlation matrix implied by economic theory is less impacted by noisy observations. At the same time, the TIC matrix strongly resembles the empirical correlation matrix. We can quantify this similarity using the correlation matrix distance introduced by Herdin and Bonek [2004],

$$d[\Sigma_1, \Sigma_2] = 1 - \frac{\mathrm{tr}\{\Sigma_1 \Sigma_2\}}{\|\Sigma_1\|_f \|\Sigma_2\|_f}$$

where $\{\Sigma_1, \Sigma_2\}$ represent the two correlation matrices on which the distance is measured, and $\|.\|_f$ is the Frobenius norm. Code snippet 6 implements the Herdin-Bonek correlation matrix distance.[3]

```
def corrDist(corr0,corr1):
    num=np.trace(np.dot(corr0,corr1))
    den=np.linalg.norm(corr0,ord='fro')
    den*=np.linalg.norm(corr1,ord='fro')
    cmd=1-num/den
    return cmd
```

Snippet 6 – Correlation matrix distance proposed by Herdin and Bonek [2004]

The distance $\mathrm{d}[\Sigma_1, \Sigma_2]$ measures the orthogonality between the considered correlation matrices. It becomes zero if the correlation matrices are equal up to a scaling factor, and one if they differ to a maximum extent. In our particular experiment, $\mathrm{d}[\Sigma_1, \Sigma_2] \approx 0.1352$, where Σ_1 is the TIC matrix and Σ_2 is the empirical correlation matrix. While TIC departs from the empirical correlation matrix ($\mathrm{d}[\Sigma_1, \Sigma_2] \gg 0$), the two are not too far apart. This corroborates that the TIC matrix has blended theory-implied views with empirical evidence.

[3] This implementation tracks the definition of the Herdin-Bonek distance. More numerically efficient implementations exist.

4.2. CONTROLLED EXPERIMENT

We would like to assess to what extent the tree graph structure is consistency with the empirical correlation matrix. In order to do that, we have run a controlled experiment, whereby we have repeated the calculations in the above section after shuffling the tickers (first column) in the GICS file. Code snippet 7 details the operations involved in this experiment.

```
import numpy as np,pandas as pd
from marcenkoPastur import deNoiseCov
gics=pd.read_csv(path+'SPX_GICS.csv')
df0=gics['Ticker'].tolist()
np.random.shuffle(df0)
gics['Ticker']=df0
corr=pd.read_csv(path+'SPX_Cov.csv',index_col=0).corr()
lnk0=getLinkage_corr(gics,corr)
corr0=link2corr(lnk0,corr.index)
corr1=deNoiseCov(corr0,q=10.,bWidth=.01,minEVal=0)
corr=corr.loc[corr1.index,corr1.columns]
print corrDist(corr,corr1)
```

Snippet 7 – Controlled study

As a result of shuffling the tickers, we do not expect to find any consistency between the tree graph structure and the empirical correlation matrix. Now the tree graph structure no longer resembles MSCI's GICS, and the distance between the TIC matrix and the empirical correlation matrix is much higher, $\mathrm{d}[\Sigma_1, \Sigma_2] \approx 0.3250$. Exhibit 6 plots the TIC matrix that results from this controlled experiment.

[EXHIBIT 6 HERE]

Because the stocks that comprise the Standard & Poors 500 share many characteristic (large cap, developed market, common shareholders, etc.), the shuffled tickers are not too dissimilar from the original ones. This places a cap on the maximum value that $\mathrm{d}[\Sigma_1, \Sigma_2]$ can take. In more diversified investment universes, composed heterogeneous stocks (e.g., including small caps, emerging markets, ETFs on commodities, etc.), the distance $\mathrm{d}[\Sigma_1, \Sigma_2]$ will rise even further, because shuffling will lead to economic theories that are more inconsistent with the empirical evidence.

From this controlled experiment, we reach two conclusions: (a) MSCI's GICS resembles the historical correlation structure, since shuffling the tickers materially increases the distance between the TIC matrix and the correlation matrix; and that (b) the TIC matrix preserves that resemblance, while yielding a less noisy correlation matrix.

5. CONCLUSIONS

For over half a century, most asset managers have used historical correlation matrices (empirical or factor-based) to develop investment strategies and build diversified portfolios. When financial variables undergo structural breaks, historical correlation matrices do not reflect the correct state of the financial system, hence leading to wrong investment decisions.

TIC matrices open the door to the development of forward-looking investment strategies and portfolios. These theories are represented by a variety of knowledge graphs, where some graphs link variables in terms of supply-chain, others in terms of shared ownership, others in terms of related product lines, etc. No knowledge graph can reflect the full information set, and different knowledge graphs capture different aspects of the association between financial variables.

For each knowledge graph, investors can derive a unique TIC matrix. Accordingly, an investor can form a multiplicity of investment strategies or portfolios, one associated with each knowledge graph, and then invest in the ensemble of those strategies or portfolios. This ensemble approach: (a) limits the amount of overfitting, as the ensemble portfolio benefits from multiple sources of information, not a single input; (b) is less biased by the past, because knowledge graphs instantly incorporate structural changes; and (c) offers transparency, as users can evaluate the impact on the ensemble strategy or portfolio from controlled changes in the knowledge graphs.

For all the above reasons, investors will benefit from not relying solely on historical covariance matrices, and instead allow theoretical considerations to inform their investment decisions.

REFERENCES

Herdin, M. and E. Bonek (2004): “A MIMO Correlation Matrix based Metric for Characterizing Non-Stationarity.” *Proceedings IST Mobile & Wireless Communications Summit*, Lyon, France, June.

Ledoit, O. and M. Wolf (2003): “A Well-Conditioned Estimator for Large-Dimensional Covariance Matrices.” *Journal of Multivariate Analysis*, Vol. 88, No. 2, pp. 365-411.

Liu Y., Q. Zeng, H. Yang, and A. Carrio (2018): “Stock Price Movement Prediction from Financial News with Deep Learning and Knowledge Graph Embedding.” In: Yoshida K., Lee M. (eds) *Knowledge Management and Acquisition for Intelligent Systems*. PKAW 2018. Lecture Notes in Computer Science, vol 11016. Springer, Cham.

López de Prado, M. (2016): “Building diversified portfolios that outperform out-of-sample.” *The Journal of Portfolio Management*, Vol. 42, No. 4, pp. 59-69. https://jpm.iijournals.com/content/42/4/59

López de Prado, M. (2018): *Advances in financial machine learning*. Wiley, First edition.

López de Prado, M. (2019a): “Beyond econometrics: The path towards financial machine learning.” Working paper. Available at https://ssrn.com/abstract=3365282.

López de Prado, M. (2019b): *Machine Learning for Asset Managers*. Cambridge University Press, First edition (forthcoming).

López de Prado, M. (2019c): “A Robust Estimator of the Efficient Frontier.” Working paper. Available at https://ssrn.com/abstract=3469961

Potter, M., J.P. Bouchaud, L. Laloux (2005): “Financial applications of random matrix theory: Old laces and new pieces.” *Acta Physica Polonica B*, Vol. 36, No. 9, pp. 2767-2784.

EXHIBITS

TICKER	GICS_SUB_INDUSTRY	GICS_INDUSTRY	GICS_INDUSTRY_GROUP	GICS_SECTOR
A UN Equity	35203010	352030	3520	35
AAL UW Equity	20302010	203020	2030	20
AAP UN Equity	25504050	255040	2550	25
AAPL UW Equity	45202030	452020	4520	45
ABBV UN Equity	35201010	352010	3520	35
ABC UN Equity	35102010	351020	3510	35
ABMD UW Equity	35101010	351010	3510	35
ABT UN Equity	35101010	351010	3510	35
ACN UN Equity	45102010	451020	4510	45

Exhibit 1 – Example of a GICS tree graph

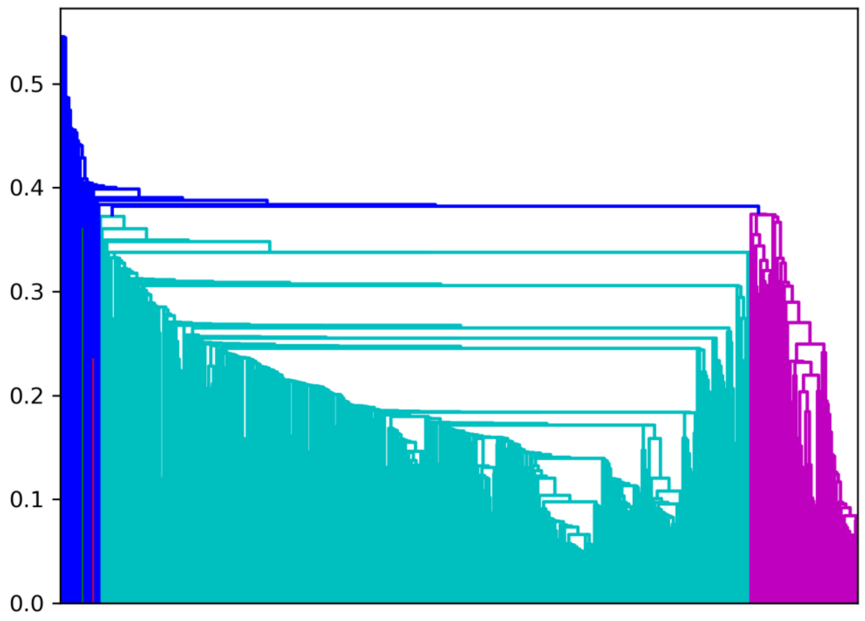

Exhibit 2 – Dendrogram from the empirical correlation matrix

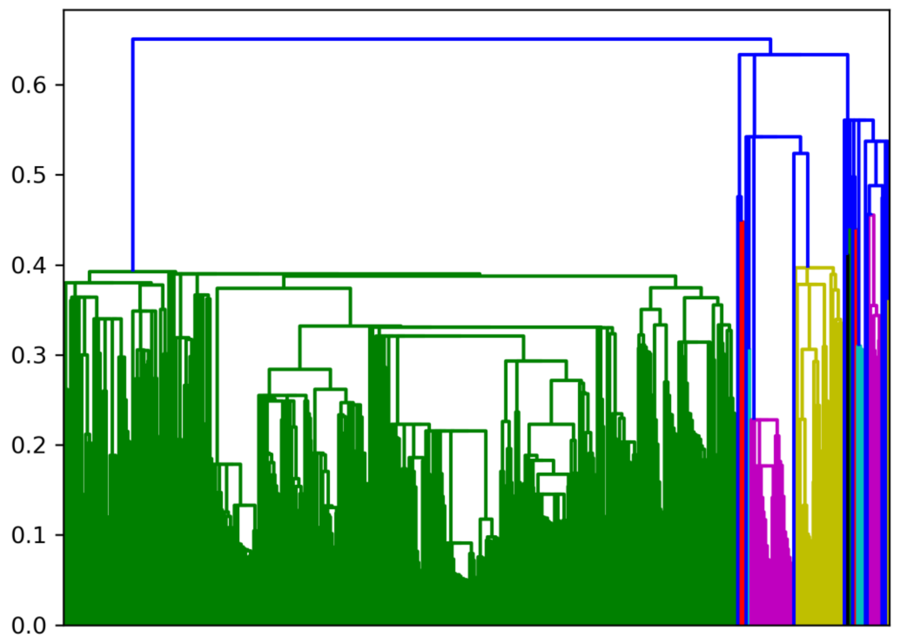

Exhibit 3 – Dendrogram from the TIC matrix

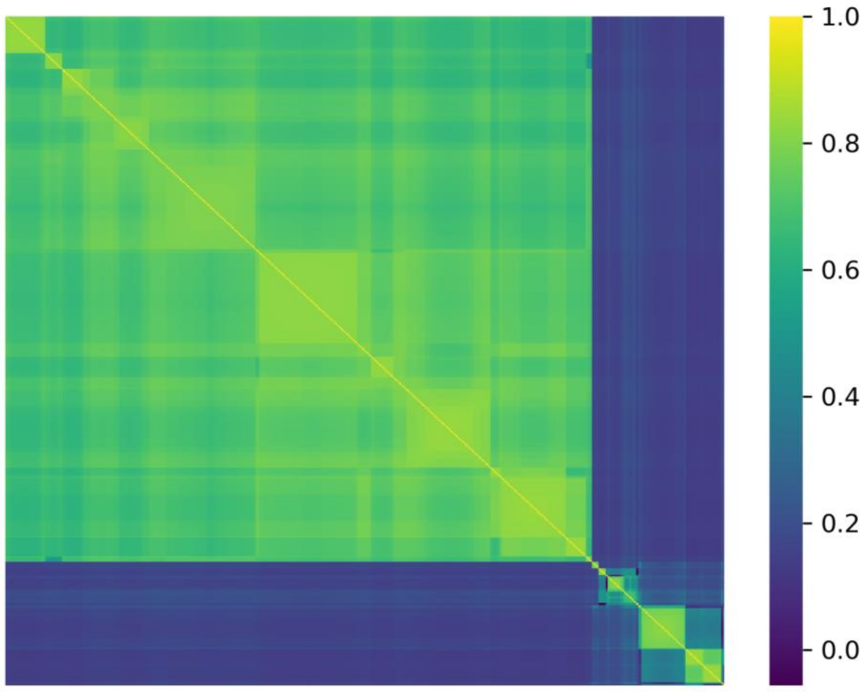

Exhibit 4 – The de-noised TIC matrix

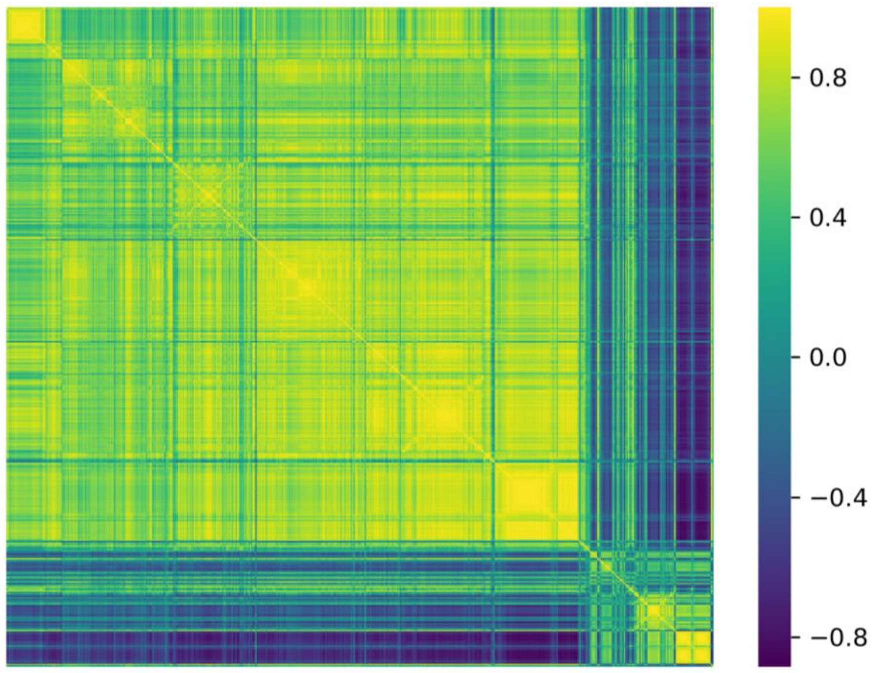

Exhibit 5 – The empirical correlation matrix

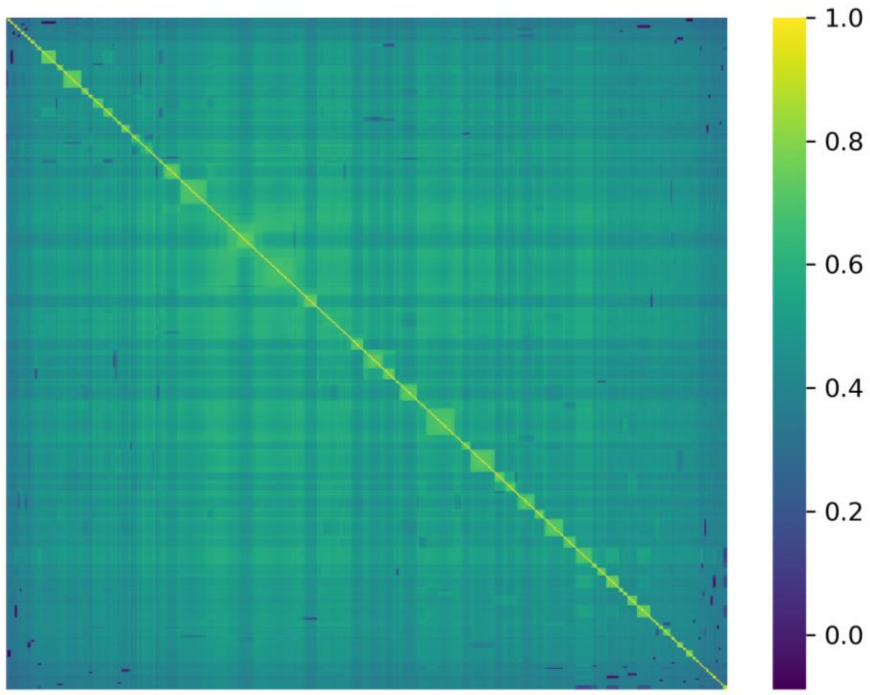

Exhibit 6 – TIC from control experiment