UNIVERSITY OF UTAH

MATH 6010

MULTILINEAR MODELS

Stock Data Clustering

Author:
Curtis MILLER

Instructor:
Dr. Braxton OSTING

April 25, 2016

UNIVERSITY OF UTAH DEPARTMENT OF MATHEMATICS

MATH 6020 Project

Curtis Miller

April 25, 2016

1 INTRODUCTION

How can one identify, from a collection of stocks, which behave similarly? One may initially try to use information about stocks for forming clusters. For example, an investor may decide that a natural division of stocks that allows for easy clustering is industry. One would expect stocks from similar industries to be affected by the same market forces (shocks, trends, technological developments, etc.), and therefore stocks from the same industry should behave similarly. Thus, industry may form a basis for clusters.

However, grouping stocks is not this simple. First, there are alternative and competing groupings that should be considered, such as the size of companies or where they are based. Thus one must wonder whether this particular means of grouping is truly the most important. Second, interrelationships among financial assets have become more complicated with the increasing sophistication of the finance industry. Strategies employed by hedge funds, mutual funds, investment banks, etc., can all lead to a tangled web of relationships and correlations that may not be obvious at first. Strategies such as hedging and diversification may make seemingly unrelated financial assets correlated, and navigating this web of relationships can be difficult to do.

The increasing complexity of financial assets makes utilizing good clustering techniques more imperative. In this paper, I examine various clustering techniques' performance on S&P 500 stocks from January 6 to February 22, 2016, using stock data downloaded from Yahoo! Finance (Yahoo! 2016). In addition to using and examining well-understood clustering algorithms such as the Gonzalez algorithm, the k-means algorithm, and the EM algorithm for the Gaussian mixture model (GMM), I also introduce model-based clustering using time series models, particularly the AR(1) model, deriving the EM algorithm for performing this clustering, developing an R function to perform this type of clustering, and comparing the results of this algorithm to the other techniques. I conclude with a final comparison and discussion.

2 Data

The data used in this study was the stocks that composed the S&P 500 stock index from January 6 to February 22, 2016, obtained from Yahoo! Finance (Yahoo! 2016). In addition to being the dataset I could most easily obtain, the stock data during this period appears to be well-behaved and thus good for initial exploration of clustering techniques. I used daily adjusted closing prices, which account for stock splits, dividends, and other activities that are not considered changing the value of a stock.

All analysis on the data is not done on the prices themselves, but rather on the first difference of the log prices of the stocks, $\log(x_{it}) - \log(x_{i,t-1})$. This difference can be interpreted as the percentage change in the price of a stock i between day $t-1$ and day t (which, despite weekends and holidays, are all considered consecutive). This is easily interpreted and ensures that all data is in effectively the same units (otherwise, problems could emerge since some stocks are individually priced higher than others, which may have no practical impact on the return other than to make investing in higher-priced stocks more difficult). This is important for algorithms such as k-means that assume all variables are in the same units.

3 Clustering

Four techniques for clustering were used: Gonzalez k-center clustering, k-means clustering, model-based clustering via the EM algorithm using a Gaussian mixture model (GMM), and model-based clustering using AR(1) processes. The latter three clustering methods used the results of the Gonzalez clustering centers for initialization.

Throughout this analysis I search for four clusters. I did examine scree plots to try and find good clusters, but they were not very conclusive. The decision ultimately resulted from the inability of the default method of `kmeans` (which is *not* Lloyd's algorithm) to find clusters using Gonzalez algorithm centers for more than four groups; too few points were assigned to some clusters, producing an error. Rather than use different methods, I decided that if having more than four clusters produced pathologies, perhaps four would be a good number of clusters. Later analysis will show that four clusters produce nicely interpretable results.

3.1 Gonzalez k-Center Clustering

The first clustering method I used was the Gonzalez k-center clustering algorithm (Gonzalez 1985). This is a greedy algorithm for finding a set of clusters and cluster centers that minimize the k-center cost, $\max_{x \in X} \|\phi_C(x) - x\|$, where x is a point, X the set of all points, and $\phi_C(x)$ the center of the cluster to which x is assigned. The algorithm tries to find centers that are as distant from each other as possible. The algorithm (as described by (Phillips 2016)) is:

```
Choose starting center $c_1 \in X$ arbitrarily. Let $C_1 = \{c_1\}$.
**for** $i = 2$ to $k$ **do**
```

```
    $c_i \leftarrow \text{argmax}_{x \in X} \|x - \phi_{C_{i-1}}(x)\|$
    $C_i = \{c_1, ..., c_i\}$
**end for**
```

A common problem with later algorithms used in this paper is the possibility of getting "stuck" in solutions that, while locally optimal, are not globally optimal, which can be attributed to a bad choice of initial cluster centers. The Gonzalez clustering algorithm chooses a set of cluster centers that are very resistant to this phenomenon, and so these centers/clusters are very useful as initial centers/clusters in other algorithms. It is capable of finding these centers in $O(kn)$ time, with k being the number of clusters and n the number of data points (Gonzalez 1985).

To my knowledge, there is no R package that implements Gonzalez clustering. However, I have written a Python implementation of the Gonzalez algorithm, and using the **rPython** package (Bellosta 2015) I was able to write an R interface for this implementation. Thus, I found Gonzalez centers for the stock data:

```
gzcluster(stock_mat_norm, 4, start_point = 10) -> gz_clust
```

`stock_mat_norm` is the object storing the stock data. The option `start_point = 10` instructs the algorithm to use the tenth data point (in this case, the stock with symbol "ADBE") as the first center (otherwise a center would have been chosen at random). This symbol was chosen for no particular reason.

The Gonzalez algorithm found four centers for four clusters with respective sizes of 82, 134, 158, and 130. The k-center cost of the result is 0.61.

Figure 3.1 visualizes the clusters found by the Gonzalez algorithm. A stock is represented by its return since the start of the period, with a red line indicating a \$1 return (i.e. no loss or gain). There appear to be no particularly interesting patterns in the Gonzalez clusters. Nevertheless, this is a useful step since these clusters can be used for other clustering algorithms.

3.2 k-MEANS CLUSTERING

The next clustering examined was the k-means clustering. The `kmeans` function in R performs k-means clustering. The default algorithm used by `kmeans` is `not` Lloyd's algorithm; instead, R uses the Hartigan-Wong clustering algorithm by default, which is believed by the authors of the function to produce the best results (R Core Team 2016). This algorithm does require that enough points be allocated to each initial cluster, and if this is not the case, an exception will be thrown. It appears that using five or more Gonzalez cluster centers results in this phenomenon. Rather than try a different clustering algorithm, I decided that this phenomenon indicated pathologies associated with seeking more than four clusters, thus resulting in my decision to use only four.

k-means clustering was performed with the following command (notice the initialization with Gonzalez centers):

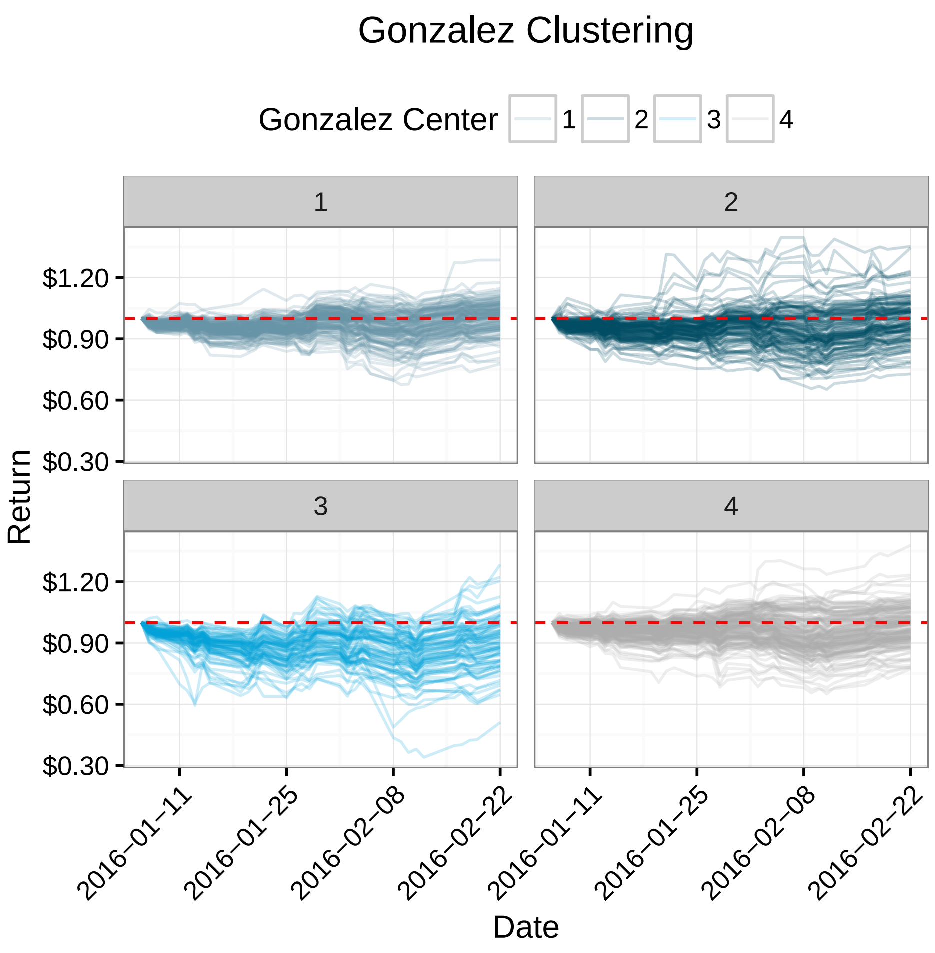

Figure 3.1: Gonzalez algorithm clustering solution, with the return of each stock since the start of the period shown. The red line indicates one dollar return per one dollar invested (i.e. no loss or gain).

```r
kmeans(stock_mat_norm, gz_clust$centers, iter.max = 20) ->
  stock_clusters
```

The total within-cluster sum-of-squares of the k-means solution is 6.48. Each of the four clusters have respectively 132, 222, 27, and 123 symbols each.

Figure 3.2 visualizes the k-means clustering solution. It is not entirely evident what the k-means clusters represent; all clusters appear to have assets with similar movement patterns over the period in question, though it does appear that this shape is slightly different from cluster to cluster. The third cluster is the most sparse and also the most volatile, while the other three clusters have roughly the same volatility in their assets. The fourth cluster appears to consist of stocks that dropped at the beginning of February and did not recover by the end

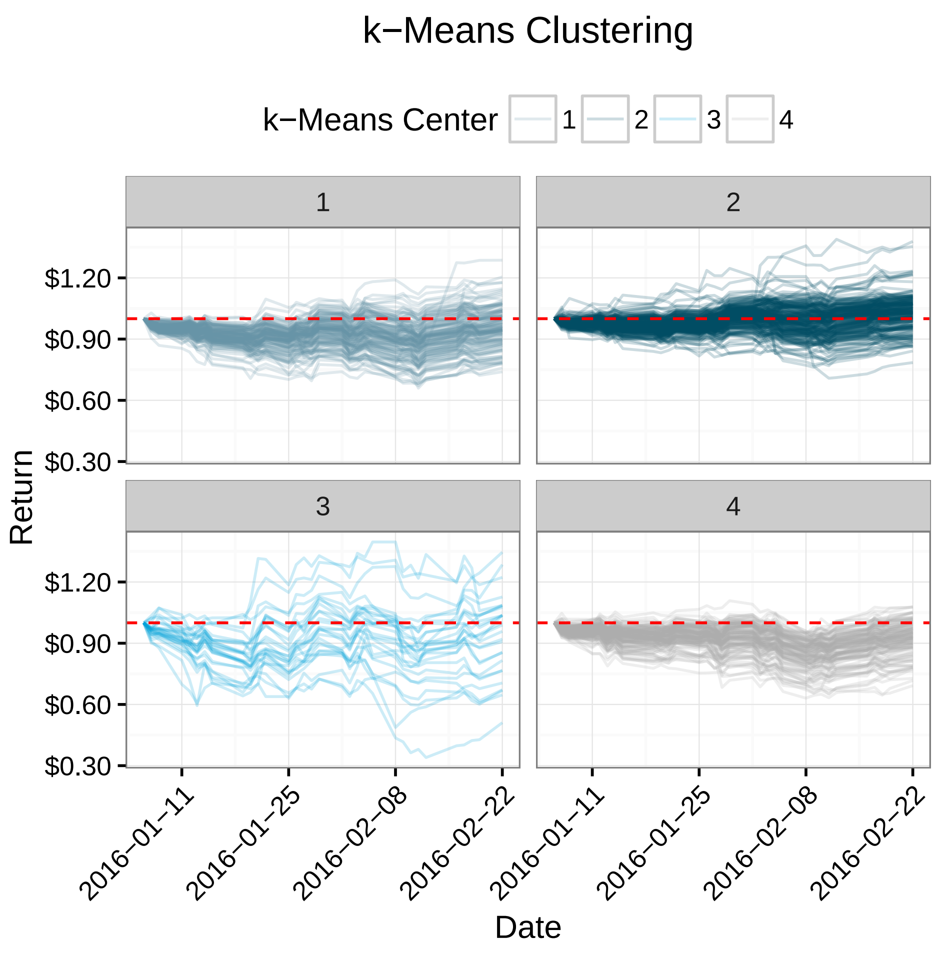

Figure 3.2: k-means clustering solution, with the return of each stock since the start of the period shown. The red line indicates one dollar return per one dollar invested (i.e. no loss or gain).

of the period, while stocks in other clusters have about an equal likelihood of either doing well or doing poorly.

3.3 Gaussian Mixture Model (GMM) Clustering

I investigated two model-based clustering solutions, the first of which being the well-known Gaussian mixture model (GMM) clustering method. There are a multitude of different permutations of GMM clustering largely depending on whether the data is univariate or multivariate and to what degree the covariance matrix of each cluster can differ and what "shape" it may take.

The R package **mclust** performs GMM clustering. Its a fairly sophisticated package with the main function, `Mclust`, making lots of decisions and taking preliminary steps in order to find good clusters (Fraley, Raftery, and Scrucca 2015). For instance, a heirarchical clustering step is performed to initialize parameters used by the EM algorithm for clustering.

The following code finds GMM clusters:

```r
# Use centers found via the Gonzalez algorithm to help
# initialization
hcVVV(stock_mat_norm, partition = gz_clust$cluster) ->
  stock_hc
# Perform GMM model-based clustering
Mclust(stock_mat_norm, G=4,
       initialization = list("hcPairs" = stock_hc)) ->
  em_clust
```

`Mclust` decided to fit the data using a VEI GMM model. Each cluster contains 36, 235, 99, and 134 symbols each. In total, 161 parameters had to be estimated by the algorithm, and it estimated the mixing proportions for each cluster being 7.05%, 46.63%, 19.83%, and 26.49%, respectively. Of all clustering methods investigated, this one took the longest to complete.

The clustering solution found via GMM clusters is shown in Figure 3.3. This is the first clustering solution of those considered so far that actually can be interpreted easily: GMM clustered the stocks based on their volatility. Cluster 1 consists of highly volatile stocks, and cluster 3 stocks with low volatility. The other two clusters consist of stocks with volatility between these two extremes. This is an unexpected result; I anticipated that GMM would cluster stocks based on shape rather than volatility. Nevertheless, this appears to be not only a valid but useful clustering solution.

3.4 AR(1) Model-Based Clustering

The final clustering method used was again a model-based clustering algorithm, but this time modelling each stock as an auto-regressive process with one lag component (commonly known as an AR(1) process). Stocks are a classic example of processes with time dependence, and none of the methods considered so far (save for GMM, in some sense) take time dependence into account, a critical ommission.

An AR(1) process represents perhaps the simplest time-series model. Let x_{it} represent the value of series i at time t. An AR(1) representation of x_{it} would be:

$$x_{it} = \phi x_{i,t-1} + \epsilon_t$$

where ϵ_t is an error term (typically Gaussian) with $\mathbb{E}[\epsilon_t] = 0$ and $\mathrm{Var}(\epsilon_t) = \sigma^2$, and ϕ is the auto-regressive coefficient.

Let $Z_i \in \{0,1\}^k$ be a random vector such that $\sum_{k=1}^{K} (Z_i)_k = 1$ and $\mathbb{P}((Z_i)_k = 1) = \pi_k$. This is effectively a vector of indicator variables (seen as a latent variable) signalling which cluster an observation i belongs to. In this context, each series in the dataset is considered to be a realization of one of K AR(1) models. Let T denote the time for which each process runs. The probability that a series $x_{i1}, \ldots, x_{iT}$ was generated by the process $x_{it} = \phi_k x_{i,t-1} + \epsilon_{kt}$ (with $\mathrm{Var}(\epsilon_{kt}) = \sigma_k^2$) is π_k. Our objective is to recover Z_i as well as estimate π_k, ϕ_k, and σ_k^2.

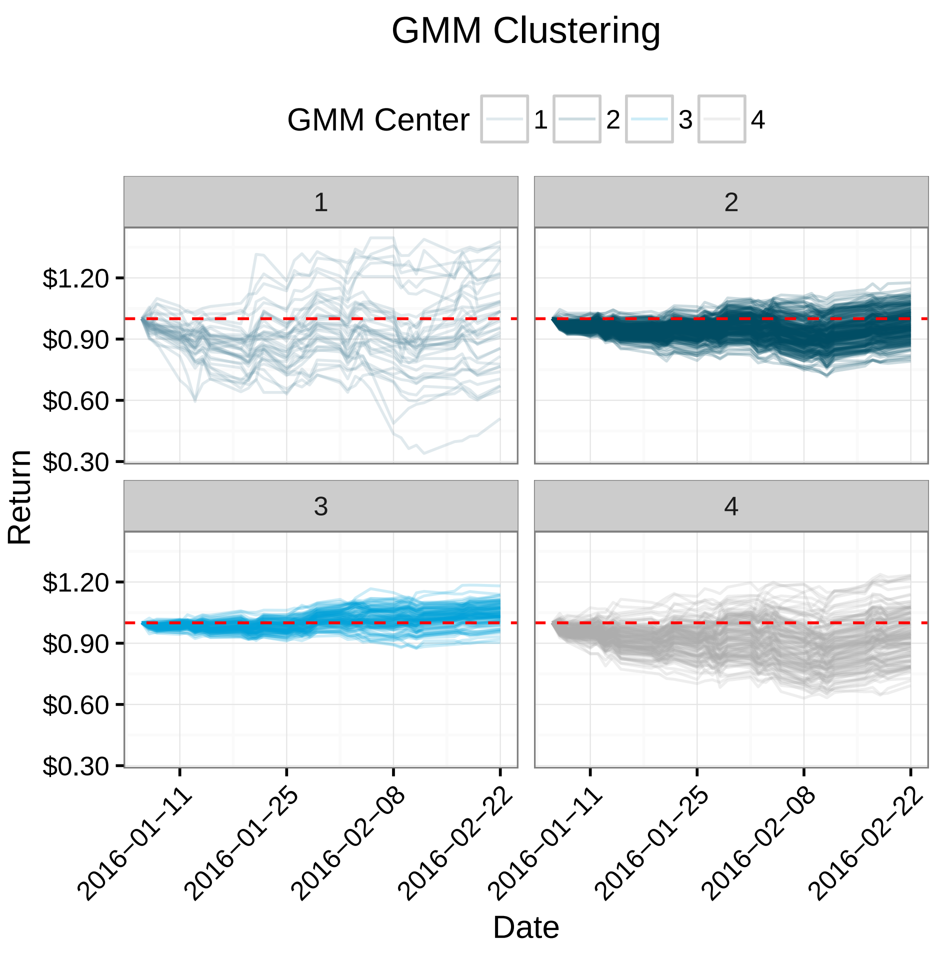

Figure 3.3: GMM clustering solution, with the return of each stock since the start of the period shown. The red line indicates one dollar return per one dollar invested (i.e. no loss or gain).

The EM algorithm is by far the most common way to solve this problem. For this particular problem, when one assumes that the error terms in the AR(1) models follow Gaussian distributions, the EM algorithm takes the following form:

E-Step In general, the E-step is to compute $Q(\Theta|\Theta^{(j)}) = \mathbb{E}\left[\log L(\Theta; X, Z)\middle| X, \Theta^{(j)}\right]$. Let:

$$p_{ik} = \frac{\pi_k^{(j)}\left(2\pi\left(\sigma_k^2\right)^{(j)}\right)^{-\frac{T-1}{2}}\exp\left(-\frac{1}{2\left(\sigma_k^2\right)^{(j)}}\sum_{t=2}^{T}\left(x_{it}-\phi_k^{(j)}x_{i,t-1}\right)^2\right)}{\sum_{l=1}^{K}\pi_l^{(j)}\left(2\pi\left(\sigma_l^2\right)^{(j)}\right)^{-\frac{T-1}{2}}\exp\left(-\frac{1}{2\left(\sigma_l^2\right)^{(j)}}\sum_{t=2}^{T}\left(x_{it}-\phi_l^{(j)}x_{i,t-1}\right)^2\right)}$$

where $\Theta^{(j)}$ is the estimated values of the parameters π_k, σ_k^2, and ϕ_k at step j. Then the

E-step is to compute:

$$Q(\Theta|\Theta^{(j)}) = \sum_{i=1}^{N}\sum_{k=1}^{K} p_{ik}\log(\pi_k) + \sum_{i=1}^{N}\sum_{k=1}^{K} p_{ik}\left(-\frac{T-1}{2}\log\left(2\pi\sigma_k^2\right) - \frac{1}{2\sigma_k^2}\sum_{t=2}^{T}\left(x_{it} - \phi_k x_{i,t-1}\right)^2\right) \tag{3.1}$$

where N is the number of series in the dataset. (In practice, only the computation of p_{ik} is done.)

M-Step In general, the M-step is to compute $\Theta^{(j+1)} = \text{argmax}_\theta\, Q(\Theta|\Theta^{(j)})$. In this context, the updated values of the parameters are:

$$\begin{aligned}
\pi_k^{(j+1)} &= \frac{\sum_{i=1}^{N} p_{ik}}{\sum_{i=1}^{N}\sum_{l=1}^{K} p_{il}} \\
\phi_k^{(j+1)} &= \frac{\sum_{i=1}^{N}\sum_{t=2}^{T} p_{ik} x_{it} x_{i,t-1}}{\sum_{i=1}^{N}\sum_{t=2}^{T} p_{ik} x_{i,t-1}^2} \\
(\sigma_k)^{(j+1)} &= \frac{\sum_{i=1}^{N}\sum_{t=2}^{T} p_{ik}\left(x_{it} - \phi_k^{(j+1)} x_{i,t-1}\right)^2}{(T-1)\sum_{i=1}^{N} p_{ik}}
\end{aligned} \tag{3.2}$$

One then iterates through the EM algorithm until convergence of the parameters is attained. One can assign a hard classification to the i^{th} series with $\text{argmax}_k\, p_{ik}$.

To my knowledge, there is no R implementation for any form of model-based clustering of time series, even for the AR(1) process. I wrote my own R function for performing this type of clustering, `kar1`, which is shown in the Appendix. The function is used to find clusters below:

```r
# First, generate a matrix that assigns each point to its
# Gonzalez cluster
idx <- matrix(c(1:nrow(stock_mat_norm),gz_clust$cluster),
              ncol=2)
clust_assign <- matrix(0, nrow=nrow(stock_mat_norm),
                       ncol=4)
clust_assign[idx] <- 1
# Use this as a prior clustering for AR(1) clustering, then
# find cluster solutions
kar1(stock_mat_norm, G=4, gamma = clust_assign) -> ar_clust
```

Details about the AR(1) clustering solution are shown in the table below.

Like the GMM clustering solution, the AR(1) clustering solution appears to cluster stocks with similar volatility, although the ordering of the clusters is different. Stocks in cluster 3 has the highest volatility, cluster 4 the lowest, and clusters 1 and 2 in-between these two extremes. This clustering solution, again, is both valid and useful.

k	Size	$(100 \times \pi_k)\%$	σ_k	ϕ_k
1	228	43.98	0.02	-0.03
2	136	28.48	0.03	-0.02
3	36	7.22	0.06	0
4	104	20.31	0.01	-0.08

4 Conclusion

I have demonstrated four clustering methods for stock data. Which method should be preferred? The Gonzalez clustering algorithm does not give a good `final` solution, but would be a good way to initialize other algorithms. The two best solutions found were with GMM-based clusters and AR(1)-based clusters, both of these solutions actually appearing very similar. I see little incentive to use k-means clustering on this data.

I argue that given the choice between GMM-based clustering and time series model-based clustering (of which AR(1) clustering is only one instance), one should use the latter. The arguments in favor of time series mode-based clustering are as follows:

1. For GMM-based clustering, depending on the variant of GMM chosen, one could estimate from KT to $K\left(2T + \binom{T}{2}\right)$ parameters. If one were to view the series as realizations of one of K ARMA(p,q) models with drift terms and differing error term standard deviations, then the number of parameters estimated would be $K(p+q+2)$, which in general will be much smaller than even KT, let alone $K\left(2T + \binom{T}{2}\right)$. One could add complexity to the time series models and still find themselves estimating far fewer parameters than the GMM-based clustering algorithms, while attaining similar-quality clusters. In this paper, I was able to obtain a good clustering solution from the AR(1)-based clustering scheme that estimated twelve parameters total, while the GMM-based clustering solution had to estimate 161 parameters in order to find a solution that differed little from the clustering solution found by AR(1)-based clustering. Thus there are savings in both time and complexity.

2. Time series models, unlike the Gaussian distributions found in GMM-based clsutering, are generative. Thus, one could use add elements such as cluster forecasts to the clustering solution, along with anything else one can do with time series models.

3. A time series model is the "correct" model for this type of data, since it accounts for time dependence. While GMM-based clustering can account for autocorrelation and other features characteristic of time dependence, it does not do so as naturally as a time series model. This means that the result of a time series based clustering can be more easily presented, interpreted, and understood.

In this paper, an AR(1)-based clustering scheme performed remarkably well for clustering, but one would naturally want to extend this to clustering ARMA(p,q) models[1]. Some papers

[1]I actually began this project by trying to develop the EM algorithm for ARMA(p,q) processes, and ran into

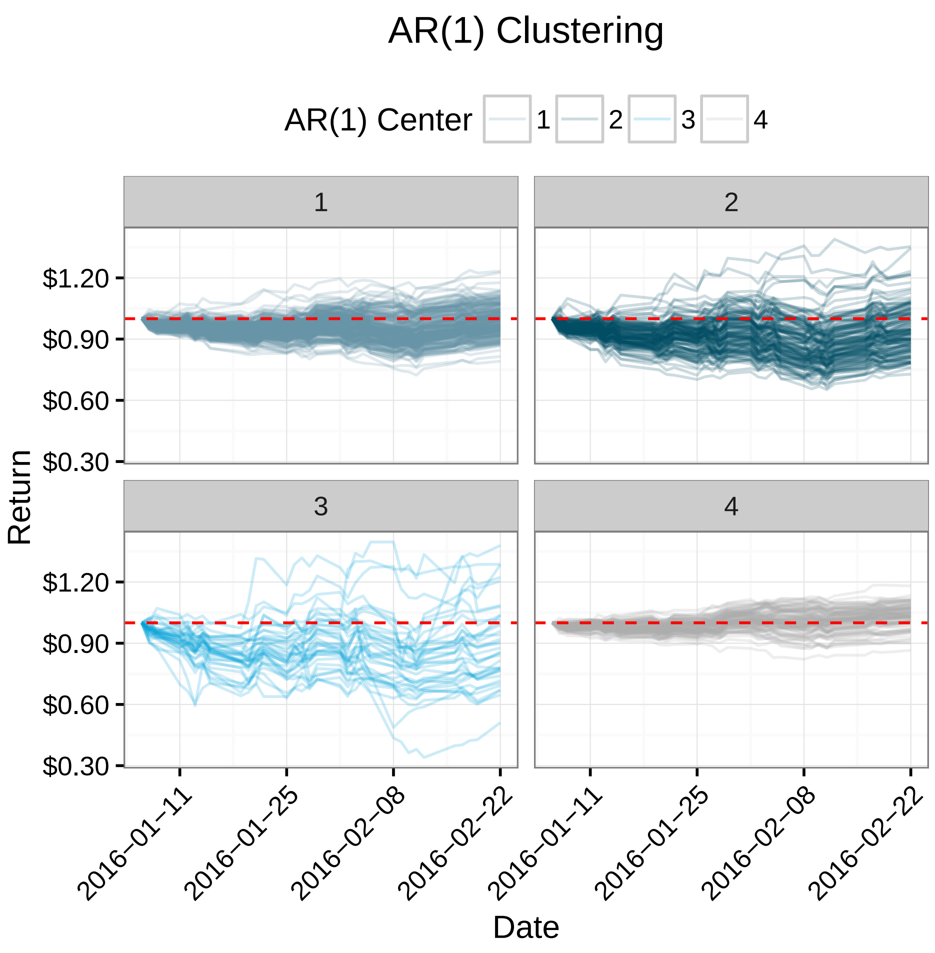

Figure 3.4: AR(1) model-based clustering solution, with the return of each stock since the start of the period shown. The red line indicates one dollar return per one dollar invested (i.e. no loss or gain).

have been written for ARMA(p,q)-based clustering (Xiong and Yeung 2004), but the methods do not appear to be well-developed and to my knowledge there is presently no R implementation for them. I believe more can be done to make these methods accessible and useful.

The dataset use here was for a limited time frame and one may wonder how well ARMA(p,q) performs on stock data when extended to a longer period of time. The assumption of Gaussian "noise" is well-known to be grossly inaccurate for stocks (Mandelbrot and Hudson 2005) and an ARMA(p,q) would not likely do this data justice when all the pathologies of stock data are given a chance to appear. I do believe that for clustering purposes ARMA(p,q) models will still perform very well since most of the time stocks are Gaussian-like; nevertheless, someone

problems with the M-step that eventually led me to abandon the endeavor and settle for AR(1) processes, which are much more tractable.

should investigate the performance of clustering methods on stock data over extended periods of time and also how more advanced models like GARCH or ARMA-GARCH models perform, along with deriving the appropriate instantiation of the EM algorithm to perform such a clustering. Developing R implementations of these clustering methods would also be useful.

Additionally, one may question what would result if stocks other than those that compose the S&P 500 are used. After all, those stocks are selected for creating a good stock index for tracking the market trend; perhaps if a more "natural" pool of stocks were chosen, different patterns would dominate. This may be worth a future investigation.

Overall, while the clusters found were not what I expected initially, I am pleased with the results. The clusters I was able to find look like clusters financial advisors would find useful for portfolio building, since one has a division of stocks into low, moderate, and high volatility. This could then allow for easier portfolio diversification and building depending on investor needs and demands. Hopefully clustering methods for this style of data will be further developed so that more interesting patterns can be discovered.

References

Arnold, Jeffrey B. (2016). *ggthemes: Extra Themes, Scales and Geoms for 'ggplot2'.* R package version 3.0.2. URL: https://CRAN.R-project.org/package=ggthemes.

Bache, Stefan Milton and Hadley Wickham (2014). *magrittr: A Forward-Pipe Operator for R.* R package version 1.5. URL: https://CRAN.R-project.org/package=magrittr.

Bellosta, Carlos J. Gil (2015). *rPython: Package Allowing R to Call Python.* R package version 0.0-6. URL: https://CRAN.R-project.org/package=rPython.

Fraley, Chris, Adrian E. Raftery, and Luca Scrucca (2015). *mclust: Normal Mixture Modelling for Model-Based Clustering, Classification, and Density Estimation.* R package version 5.1. URL: https://CRAN.R-project.org/package=mclust.

Gonzalez, Teofilo F. (1985). “Clustering to minimize the maximum intercluster distance”. In: *Theoretical Computer Science* 38, pp. 293–306.

Mandelbrot, B.B. and R.L. Hudson (2005). *Misbehavior of Markets.* Basic Books. ISBN: 9780465043576. URL: https://books.google.com/books?id=DPwBTj99a7UC.

Phillips, Jeff (2016). *Assignment-based Clustering.* URL: https://www.cs.utah.edu/~jeffp/teaching/cs5140/L9-kmeans.pdf.

R Core Team (2016). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing. Vienna, Austria. URL: https://www.R-project.org/.

Wickham, Hadley (2007). “Reshaping data with the reshape package”. In: *Journal of Statistical Software* 21.12. URL: http://www.jstatsoft.org/v21/i12/paper.

– (2009). *ggplot2: Elegant Graphics for Data Analysis.* Springer-Verlag New York. ISBN: 978-0-387-98140-6. URL: http://had.co.nz/ggplot2/book.

– (2015). *stringr: Simple, Consistent Wrappers for Common String Operations.* R package version 1.0.0. URL: https://CRAN.R-project.org/package=stringr.

Wickham, Hadley and Romain Francois (2015). *dplyr: A Grammar of Data Manipulation.* R package version 0.4.3. URL: https://CRAN.R-project.org/package=dplyr.

Xiong, Yimin and Dit-Yan Yeung (2004). “Time series clustering with {ARMA} mixtures”. In: *Pattern Recognition* 37.8, pp. 1675 –1689. ISSN: 0031-3203. DOI: http://dx.doi.org/10.1016/j.patcog.2003.12.018. URL: http://www.sciencedirect.com/science/article/pii/S0031320304000585.

Yahoo! (2016). *Yahoo! Finance*. URL: http://finance.yahoo.com/.

Appendix

I wrote an R function for performing AR(1)-based clustering, kar1. This function takes a dataset and the number of clusters, along with optionally the number of iterations and the initial responsibilities of each cluster for each datapoint, gamma. (It is not recommended to use the default for gamma, since this may result in "stationary" clusters.) The comments in the code block provide more details on use and the kar1-class object returned.

```r
kar1 <- function(data, G, gamma = NULL, iteration=100) {
  ## kar1
  ##
  ## A function perfoming AR(1)-based clustering on a
  ## dataset
  ##
  ## :param data: data.frame; The dataset to cluster
  ## :param G: integer; The number of clusters to find
  ## :param gamma: matrix; A matrix with nrow(data) rows
  ##     and G columns corresponding to the
  ##     responsibilities each cluster has initially for
  ##     each data point. If not initialized, each cluster
  ##     will have equal responsibility (i.e. 1/G).
  ##     WARNING: The default may result in non-moving
  ##     clusters! It is not recommended to use the default
  ##     now.
  ## :param iteration: integer; The number of iterations to
  ##     cluster through
  ##
  ## :return: kar1; A list with the following components:
  ##     proportions: A vector of estimated mixing
  ##         proportions
  ##     coef: A vector of the AR coefficients for each
  ##         cluster
  ##     sd: A vector of standard deviation parameters
  ##         estimated for each cluster
  ##     responsibility: A matrix corresponding to each
  ##         cluster's responsibility for each data point
  ##     classification: The most likely cluster each point
  ##         would be assigned to
  ##
  ## This function performs model-based clustering, with
  ## the generative model being an AR(1) process. With the
  ## EM algorithm, clusters are found and a list object is
  ## returned with estimated parameters and cluster
  ## assignments.

  data_next <- data[,-1]
  data_lag <- data[,-ncol(data)]
  if (is.null(gamma)) {
    gamma <- matrix(1/G, nrow=nrow(data), ncol=G)
  }
  for (i in 1:iteration) {
    mix_prop <- colSums(gamma)/sum(gamma)
    phi <- colSums(gamma*rowSums(data_next*data_lag))/
      colSums(gamma*rowSums(data_lag^2))
    sig_num <- sapply(phi, function(p) rowSums(
      (data_next - p*data_lag)^2))
    sigma_sq <- colSums(gamma*sig_num)/
      ((ncol(data_next)-1)*colSums(gamma))
    gnum <- t(mix_prop*(2*pi*sigma_sq)^(
      -(ncol(data_lag)-1)/2)*exp(-t(sig_num)/(2*sigma_sq)))
    gamma <- gnum/rowSums(gnum)
  }

  return_obj <- list()

  return_obj$proportions <- mix_prop
  return_obj$coef <- phi
  return_obj$sd <- sqrt(sigma_sq)
  return_obj$responsibility <- gamma
  return_obj$classification <- t(apply(gamma, 1,
                    function(s) min(which(s == max(s)))))

  class(return_obj) <- "kar1"
  return(return_obj)
}
```