**K-Means Clustering EViews add-in**

Erhard Menker

June 6, 2017

**Document Overview**

This document explains the theory & motivation behind k-means clustering, and presents the application of the algorithm using the EViews add-in. The algorithm’s implementation adheres to that given by Dr. Andrew Ng’s Stanford machine learning course.[[1]](#footnote-1) However, any implementation deviations from the pseudocode are of course the responsibility of the add-in’s author. Inquiries of any kind are warmly welcomed; the add-in’s author, Erhard Menker, can be reached via email ([ejmenker@gmail.com](mailto:ejmenker@gmail.com)), and the working version of the project will continue to be developed on Github.[[2]](#footnote-2)

**Cluster Analysis Motivation**

Cluster analysis is an unsupervised machine learning algorithm, meaning it is applied to “unlabeled data” (the classification of observations is not given in the dataset). Given a dataset of m observations, k-means clustering assigns each observation in the dataset to belong to one of cluster centroid 1 or 2 or … or k, conditioned that the # of centroids is less than m. A cluster’s centroid is the coordinates of the arithmetic average of each series for its associated observations. This assignment is done by finding the centroids that minimizes the cost function, a measure that values centroids that have close proximity to their associated points. Therefore, clustering algorithms are a useful form of exploratory data analysis and can be used in EViews with time series data (e.g. classifying macroeconomic regimes over a country’s history) & cross section data (e.g. customer segmentation).

**User Arguments**

* *Mandatory*
  + **k** – the # of centroids that the dataset with n observations will be partitioned into (constrained to be less than n)
* *Optional* 
  + **quiet** – shut off the add-in’s log messages
    - defaults to presenting log messages
  + **inits** – the # of random initializations & solves of the cluster centroids to occur
    - defaults to 3
  + **max\_iters** – the maximum # of times the cluster centroids are allowed to move based on its associated observations for a given solve of centroids
    - defaults to 10
    - if set to “NONE”, will continue until convergence occurs
  + **series** – a space delimited string of the series on the workfile page to be included in the cluster analysis
    - defaults to including all series on the workfile page
  + **smpl** – observations to be included in the clustering algorithm
    - defaults to the sample at time of function call
    - if smpl = @all or smpl = all, then set sample to equal the range
  + **impute** | **interpolate** – standalone argument that interpolates a series’ missing values
    - defaults to not interpolating
    - for time series pages, linearly interpolate series
    - for unstructured workfiles, impute with the series’ median

**K-Means Algorithm Summary[[3]](#footnote-3)**

|  |
| --- |
| ' preprocess series  ' for each random initialization:  ' initialize k random obs as cluster centroids  ' while the cluster centroids continue to converge OR the max # of cluster moves is NOT reached:  ' find the closest centroid to each obs  ' set each cluster centroid equal to the mean of its associated obs for all series  ' find the closest centroid to each obs  ' calculate the cost function of the cluster solve  ' if the cost function is the smallest yet for all solutions:  ' store the centroids & associated obs as the optimal clustering thus far |

**Add-in Output**

* 2 objects are returned to the add-in page:
  + **obs\_cluster** – a series object where each observation states the cluster # to which the observation belongs
  + **kmeans\_results** – a text object declaring, for each centroid, how the mean of each of its series compares to the overall series mean (both nominal & percentage differences)

**Example Application**

***Program:***

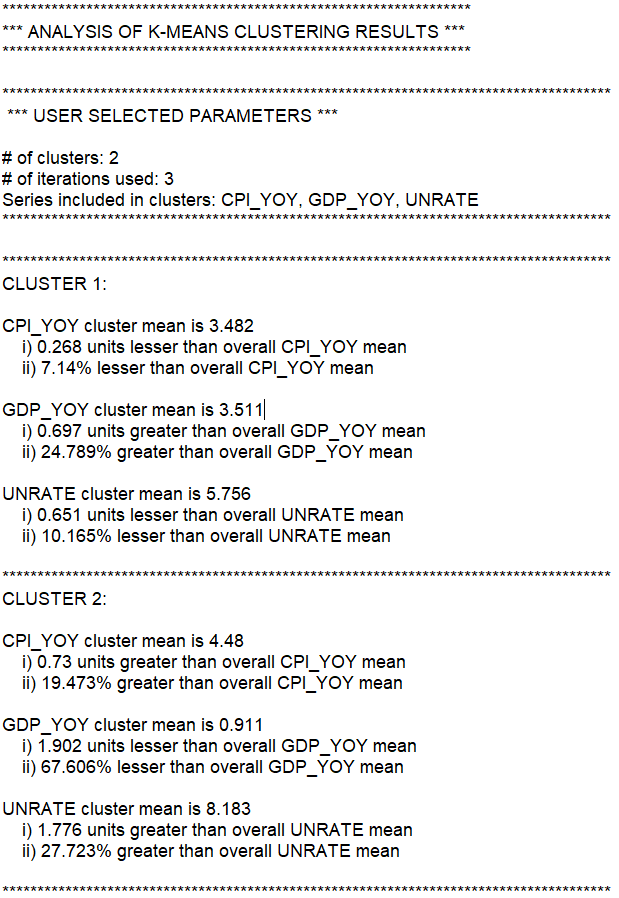
|  |
| --- |
| ' create an annual workfile  wfcreate(wf=k\_means\_example, page=DATA\_A) a 1975 2016  ' fetch series (accessed via FRED API)  ' a) unemployment rate  fetch fred::unrate  ' b) gdp yoy  fetch fred::a191ro1q156nbea  rename a191ro1q156nbea gdp\_yoy  ' c) cpi yoy  fetch fred::cpiaucns  series cpi\_yoy = @pcy(cpiaucns)  delete cpiaucns  ' calculate 2 cluster centroids between 1975 & 2016  exec .\..\kmeans.prg(k = 2) |

***Output:***

a. series object (*obs\_cluster*) indicating which observations correspond to the 2 centroids



b. text file (*kmeans\_results01*) stating how mean of each series for each centroid compares to the series’ aggregate mean



***Analysis:***

This k-means application looks at annual U.S. macroeconomic data between 1975 & 2016. Level unemployment & year-over-year percentage changes in real GDP & the unemployment rate capture the state of the US macroeconomy over the past few decades. Choosing 2 clusters centroids means that there are going to be 2 states of the economy captured. Looking at the text file, we can see there is cluster 1 (higher GDP growth and lower unemployment level & CPI growth) and cluster 2 (lower GDP growth and higher unemployment level & CPI growth), with these comparison to the overall mean derived from the text output.

These economic regimes correspond to understood macroeconomic U.S. history; observations from cluster 2 appear during periods of U.S. recession including in the early 1980s & 1990s, the bursting of the tech bubble, and the financial crisis (with more 2s in the financial crisis than the rest of the recessions here combined, highlighting the severity & slow recovery).

This serves as a vindication of accuracy; more interesting applications require increased sophistication of problem setup. For example, the economist could consider adding more macroeconomic series & higher frequency and increasing k to detect for more complicated regimes (e.g. high GDP growth, lower unemployment & higher inflation). Recessions are defined by GDP, so another application would be to remove GDP & see if different economic indicators, like manufacturing/trade data, could be used to pick up recessions (or even serve as leading indicators of recessions when the series are considered together).

To do analysis on 1 centroid of the data, conditional sample definitions can be called:

|  |
| --- |
| ' constrain the sample to the recession values  smpl @all if obs\_cluster = 2 |

**Frequently Asked Questions**

1. ***Isn’t there a deterministic algorithm to get the optimal clustering!?***

There is! It’s called hierarchical clustering.[[4]](#footnote-4) Hierarchical clustering requires only 1 solve since the returned solve is deterministic & thus guaranteed to minimize the cost function. The reason it is not implemented is because of algorithmic inefficiency as the inputted series become arbitrarily large. Hierarchical clustering requires, for each unmerged point & centroid of merged points, merging into clusters based on the minimal pairwise distance across all these points & merged cluster centroids. A superficial hierarchical clustering implementation would run in quadratic time, and while divide-and-conquer & dynamic programming techniques can speed up execution, k-means clustering is implemented because it runs in linear time, so executing multiple solves of a linear algorithm is preferred to executing one solve of a quadratic algorithm for a sufficiently large dataset.

1. ***K is the only mandatory argument…how should I choose it?***

As K increases (from 1 to the # of observations), the cost function constantly decreases (if K were to equal the # of observations, the cost would be 0 as each observation would be explained perfectly with its very own centroid equal to itself). This means that the simplistic selection of K, if algorithmic, would have to implement an accurate measure that assesses when increasing K marginally reduces the cost function by an insufficient amount. In addition to the difficulty of setting accurate parameters based on the problem’s context, the algorithm could be computationally taxing because the k-means cluster would have to be solved for each value of K.

Another approach suggested is to not algorithmically choose K, but rather judge the usefulness of K based on the ability to serve purposes downstream with which the classification is used. For example, if an economist is using the add-in to classify macroeconomic regimes, she may find that an insufficient value of K misses common regimes in the macroeconomy, while too large a value of K creates different clusters that substantively reflect a similar macroeconomy, and are thus based in inconsequential variations of series values.

1. ***How are the series preprocessed?***
2. Eliminate series that have all NAs or no variability over k-means sample
3. Normalize each series
4. Interpolate (if this argument is passed in to overwrite the default of no interpolation)
5. Disregard observations that have at least 1 series with an NA
6. Add a small # to an observation’s 1st series value if the observation is a duplicate of a prior observation
7. ***What does it mean for the series to be normalized? Does this distort results?***

Series are normalized by converting its values into their z-scores (the series’ value is equal to the original series’ value minus the series’ mean, all divided by the series’ standard deviation). This is essential; normalization removes the distortionary effects of comparing series with different magnitudes & measures of dispersion. In the absence of normalization, relatively slight differences in one series could dwarf relatively major differences in another series if the magnitude of the former series is substantially larger than the latter. Normalization does not distort results – the normalized series are only used to determine how the observations should be clustered. Absolute & percentage changes presented in the outputted text file are all calculated based on the original series’ values.

1. ***Can a cluster centroid wind up with no associated observations?***

Protections are put in place to protect against this in the implementation. Centroids are initialized to equal random observations, so each centroid is identical to a point that it will be associated with initially. An observation-less centroid could occur if multiple points are identical, but protections are taken against this: a small add factor (0.000001) is given to the 1st normalized series in case of point duplication, because if multiple centroids shared identical coordinate points during initialization, then all observations at that point would be assigned to just 1 of the centroids. Giving the small add factor assures that initial assignment is maintained for at least the 1st iteration given duplicates. The interpretation of the outputted cluster only becomes problematic for contrived examples that probably shouldn’t be clustered anyways (e.g. all the initial points are identical, so the clustering is solely contingent based on the minor add factor when there is in fact no real difference between the data points).

1. ***Why eliminate series with no variability?***

If a series has no variability (all observations are the same), it adds no explanatory power to the dataset’s observations. Considering the n other series, adding the series to the cluster is like mapping into n + 1 space, but the (n + 1)st dimension value is just a constant real number. This added dimension adds nothing in differentiating the dataset’s observations from one another because it’s a constant attribute across points. Also, constant series cannot be normalized since normalized series are divided by the series’ standard deviation, which is 0 in this case.

1. <https://www.coursera.org/learn/machine-learning/lecture/93VPG/k-means-algorithm> [↑](#footnote-ref-1)
2. <https://github.com/ErhardMenker/kMeans4EViews> [↑](#footnote-ref-2)
3. For a detailed description with formulas & visuals, see: <http://cs229.stanford.edu/notes/cs229-notes7a.pdf> [↑](#footnote-ref-3)
4. <https://www.coursera.org/learn/exploratory-data-analysis/lecture/68OwU/hierarchical-clustering-part-1> [↑](#footnote-ref-4)