K Means Clustering

In this lecture. . .

- Assigning data points to centres
- How to find the best positions for the centres
- How to find the ideal number of clusters

Introduction

K means clustering is an **unsupervised-learning technique**. We have many individual data points each represented by vectors. Each entry in the vector represents a feature. But these data points are not labelled or classified *a priori*.

Our goal is to group these data points in a sensible way. Each group will be associated with its centre of mass. But how many groups are there and where are their centres of mass?

What Is It Used For?

K means clustering is used for

- Grouping together unlabelled data points according to similarities in their features. The features must have meaningful numerical values

- Example: Classification of customers according to their purchase history, each feature might be expenditure on different types of goods

- Example: Optimal placement of car parks in a city

- Example: Optimizing the neck size and arm length of shirts

K means clustering (KMC) is a very simple method for assigning individual data points to a collection of groups or clusters.

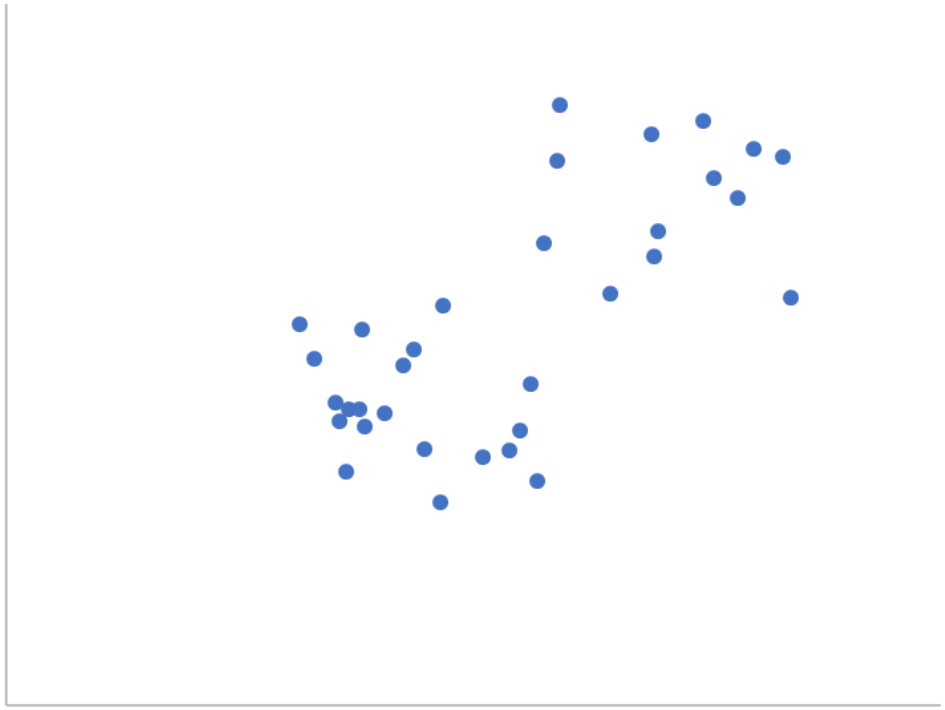
- Which cluster each data point is assigned to is governed simply by its distance from the centres of the clusters

- Since it's the machine that decides on the groups this is an example of unsupervised learning

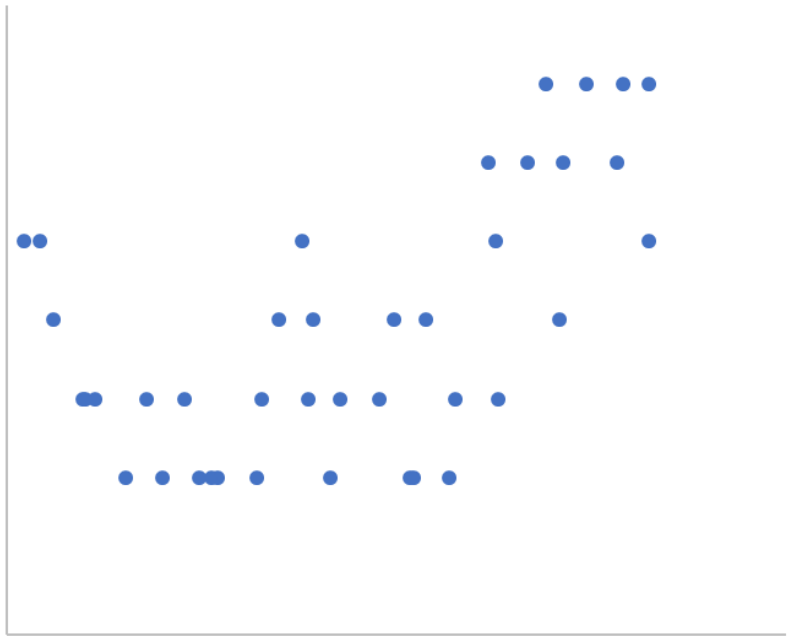
KMC is a very popular clustering technique. It is highly intuitive and visual, and extremely easy to programme.

The technique can be used in a variety of situations. . .

- Looking for structure within datasets. Here there are two obvious distinct groups



- Dividing up data artificially even if there is no obvious grouping. The made-up data in this figure might represent family income on the horizontal axis with the vertical axis being the number of people in the household. A manufacturer of cutlery wants to know how many knives and forks to put into his presentation boxes and how fancy they should be)



What Does K Means Clustering Do?

We have a dataset of N individuals, each having an associated M-dimensional vector representing M features, so $\mathbf{x}^{(n)}$ for $n =$ 1 to N.

Each entry in the vectors represents a different numerical quantity. For example each vector could represent an individual household, with the first element in each vector being an income, the second the number of cars, . . . , the M^{th} being the number of pets.

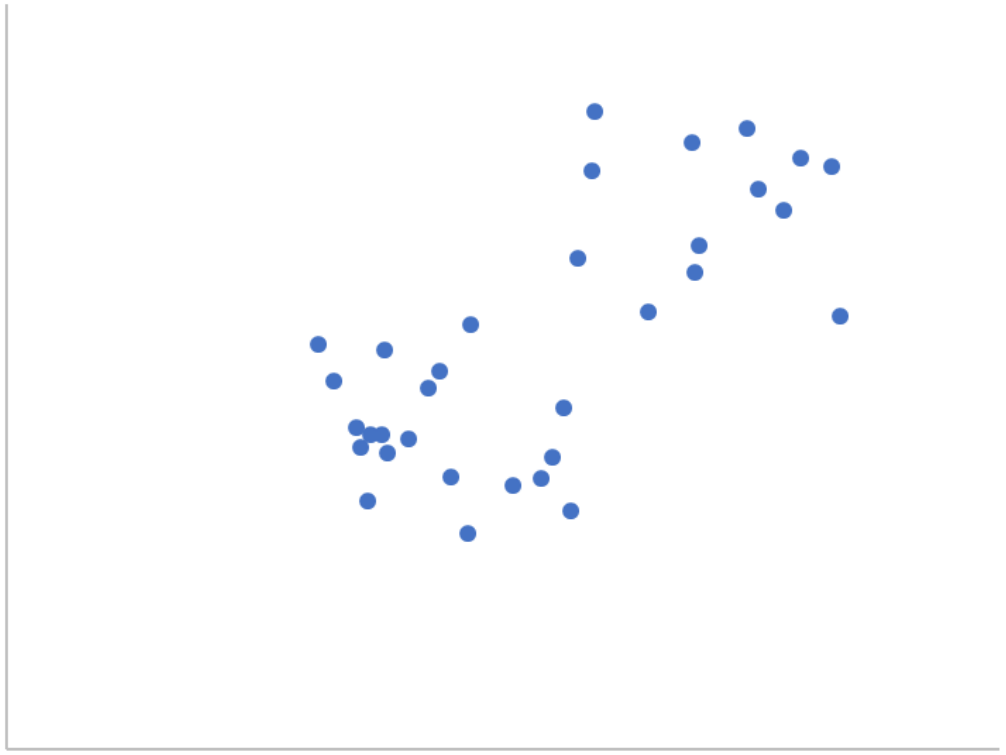
We are going to group these data points into K clusters.

- We will pick a number for K, say three. So we will have three clusters
- Each of these three clusters will have a centre, a point in the M-dimensional feature space
- Each of the N individual data points is associated with the centre that is closest to it.

(Like houses and postboxes. The postbox is like the centre for the cluster. And each house is associated with one postbox.)

The goal of the method is to find the best positions for the centres of these clusters. (And so you could use KMC to tell you where is best to put the postboxes.)

In the figure below there are clearly two distinct groups. If I asked you to put two dots in the picture representing the centres of the groups where would you put them? That is the goal, but the machine will do it for us.



- Mathematically, the idea is to minimize the intra-cluster (or within-cluster) variation. And the intra-cluster variance is just a measure of how far each individual point is to its nearest centre. (How far is the house from the nearest postbox.)

Typically you might then choose a different K and see what effect that has on distances.

The algorithm is really simple. It involves first guessing the centres and then iterating until convergence.

The K Means Algorithm

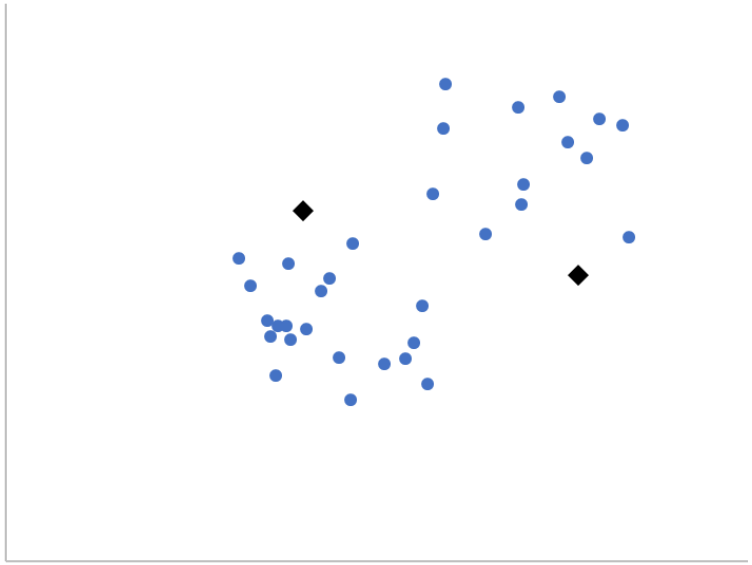
Step 0: Scaling

We first scale our data, since we are going to be measuring distances.

Now we start on the iteration part of the algorithm.

Step 1: Pick some centres

We need to seed the algorithm with centres for the K clusters. Either pick K of the N vectors to start with, or just generate K random vectors. In the latter case they should have the same size properties as the scaled datasets, either in terms of mean and standard deviation or minimum and maximum. Call these centres $\mathbf{c}^{(k)}$ for $k = 1$ to K. See the two diamonds in the figure.



Step 2: Find distances of each data point to the centres

Now for each data point (vector $\mathbf{x}^{(n)}$) measure its distance from the centres of each of the K clusters. The measure we use might be problem dependent. But often we'd use the obvious Euclidean distance:

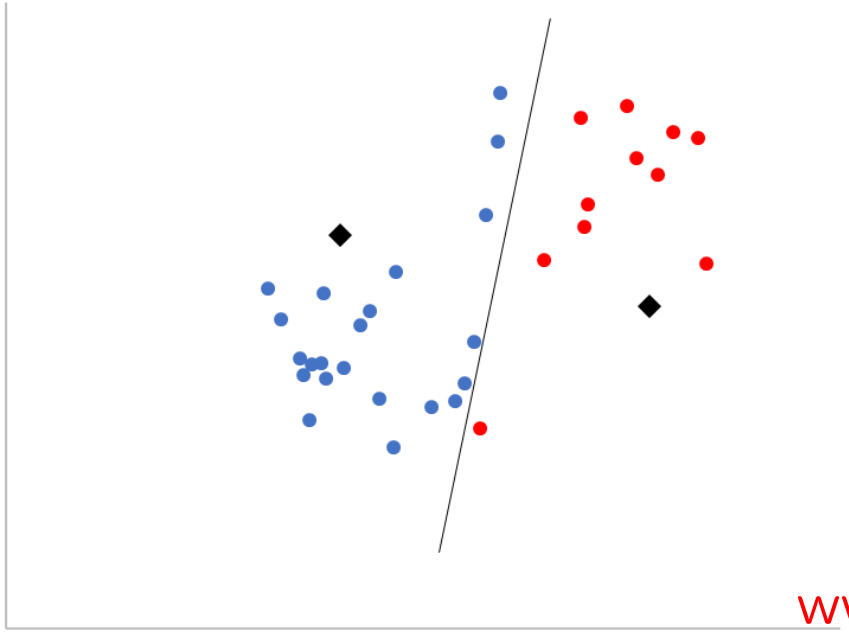
$$\text{Distance}^{(n,k)} = \sqrt{\sum_{m=1}^{M} \left(x_j^{(n)} - c_j^{(k)} \right)^2} \quad \text{for } k = 1 \text{ to } K.$$

Each data point, that is each n, is then associated with the nearest cluster/centre:

$$\underset{k}{\text{argmin}}\, \text{Distance}^{(n,k)}.$$

This is easily visualized as follows. Suppose K is two, there are thus two clusters and two corresponding centres. Let's call them the red cluster and the blue cluster.

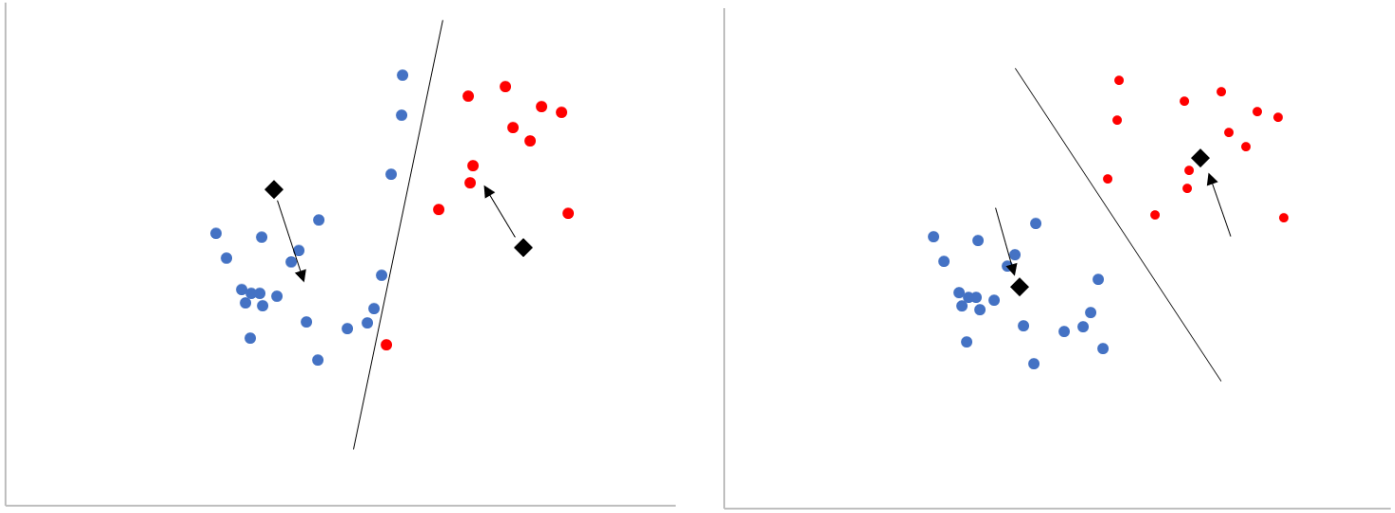
We take the first data point and measure its distance from each of the two centres. That's two distances. The smaller of these turns out to be the distance to the blue centre. So we paint our first data point blue. Repeat for all of the other data points, so each data point gets coloured.



Step 3: Find the K centroids

Now take all of the data points associated with the first centre and calculate the centroid, its centre of mass. In the colourized version, just find the centroid of all the red dots. Do the same with all the blue dots, i.e. find the K centroids. These will be the cluster centres for the next iteration.

Go back to Step 2 and repeat until convergence.



Scree Plots

Adding up all the squared distances to the nearest cluster gives us a measure of total distance or an error. This error is a decreasing function of the number of clusters, K. In the extreme case $K = N$ you have one cluster for each data point and this error will be zero.

If you plot this error against K you will get a scree plot.

It will be one of the two types of plots shown in the figure. If you get the plot with an '**elbow**' where the error falls dramatically and then levels off then you probably have data that falls into nice groupings (the triangles in the figure). The number of clusters is then obvious, it is three in this plot.

If the error only gradually falls then there is no obvious best K using this methodology.

Note that while convergence is typically quite rapid, the algorithm might converge to a local minimum distance. So one would usually repeat several times with other initial centres.

Example: Crime in England, a 13-dimensional example

For our first real example I am going to take data for crime in England. The data I use is for a dozen different categories of crime in each of 342 local authorities, together with population and population density data. The population data is only for scaling the numbers of crimes, and so we shall therefore be working in 13 dimensions. Don't expect many 13-dimensional pictures.

The crimes are:

Burglary in a building other than a dwelling
Burglary in a dwelling
Criminal damage
Drug offences
Fraud and forgery
Offences against vehicles
Other offences
Other theft offences
Robbery
Sexual offences
Violence against the person - with injury
Violence against the person - without injury

The first few lines of the raw data looks like this:

Local Authority	Burglary in a building other than a dwelling	Burglary in a dwelling	Criminal damage	Drug offences	Fraud and forgery	Offences against vehicles	...	Population	Population per Square Mile
Adur	280	120	708	158	68	382	...	58500	3610
Allerdale	323	126	1356	392	79	394	...	96100	198
Alnwick	94	33	215	25	11	71	...	31400	75
Amber Valley	498	367	1296	241	195	716	...	116600	1140
Arun	590	299	1806	471	194	819	...	140800	1651
Ashfield	784	504	1977	352	157	823	...	107900	2543
Ashford	414	226	1144	196	162	608	...	99900	446
Aylesbury Vale	696	377	1490	502	315	833	...	157900	453
Babergh	398	179	991	137	152	448	...	79500	346
Barking & Dagenham	639	1622	2353	1071	1194	3038	...	155600	11862
Barnet	1342	3550	2665	1198	1504	4104	...	331500	9654
Barnsley	1332	860	3450	1220	322	1661	...	228100	1803
Barrow-in-Furness	190	134	1158	179	59	227	...	70400	2339
Basildon	756	1028	1906	680	281	1615	...	164400	3874
Basingstoke & Deane	1728	598	426	930	182	1159	...	147900	605

The crime numbers are first scaled with population in the local authorities. And then there is the second translation and scaling as discussed earlier.

A straightforward application of K-means clustering results in the following scree plot.

There is a moderately convincing elbow, at around three clusters. And those three clusters are particularly interesting.

The results are shown in the following table, in original variables but with the crime numbers scaled per head of population.

	Cluster 1	Cluster 2	Cluster 3
Number in cluster	1	68	273
Burglary in a building other than a dwelling	0.0433	0.0059	0.0046
Burglary in a dwelling	0.0077	0.0079	0.0030
Criminal damage	0.0398	0.0156	0.0114
Drug offences	0.1446	0.0070	0.0029
Fraud and forgery	0.1037	0.0042	0.0020
Offences against vehicles	0.0552	0.0125	0.0060
Other offences	0.0198	0.0018	0.0009
Other theft offences	0.6962	0.0313	0.0154
Robbery	0.0094	0.0033	0.0004
Sexual offences	0.0071	0.0015	0.0008
Violence against the person - with injury	0.0560	0.0098	0.0053
Violence against the person - without injury	0.0796	0.0128	0.0063
Population per Square Mile	4493	10952	1907

Cluster 1 has precisely one point. It is the City of London. In this data it appears as an outlier because the population figures for each local authority are people who *live* there. And not many people live in the City. We thus can't tell from this analysis how safe the City really is, since most of the crimes probably happen to non residents. For example, Burglary from a Dwelling is similar in both the City and in Cluster 2.

The other two clusters clearly represent dangerous local authorities (Cluster 2) and safer ones (Cluster 3).

The above is quite a high-dimensional problem, 13 features meaning 13 dimensions. And relatively few, 342, training points. We might have expected to suffer from the curse of dimensionality. From the results it looks like, luckily, we didn't. The reason for this might be the common one that the different features don't seem to be independent. Theft, robbery, burglary are very similar. Fraud and forgery, and sexual offences are not.

If you wanted to reduce the dimensions early on, before using K means, then you could use Principal Components Analysis. Or, as possibly here, don't use features that common sense says are very closely related.

We are now going to do a few financial/economic examples. There is an abundance of such data, for different financial or economic quantities and in vast amounts.

Warning:

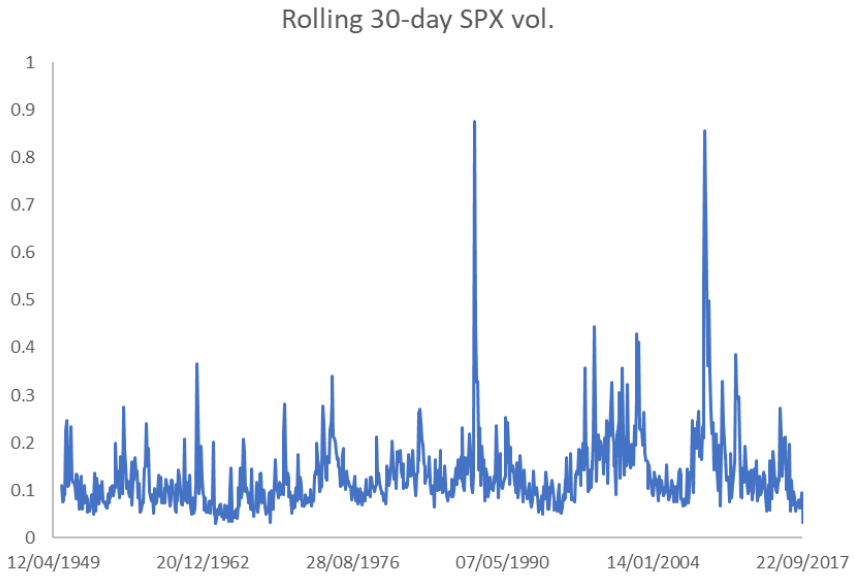
I am going to be using time-series data below. Time-series data is not necessarily suitable for K-means clustering analysis because the time component is lost, or rather it is not utilized.

Nevertheless in the following examples we shall find some interesting results.

Example: Volatility

Of course, we can do problems in any number of dimensions. So let's start our financial examples with a one-dimensional example.

I first downloaded the S&P500 index historical time series going back to 1950. From this I calculated a 30-day volatility.



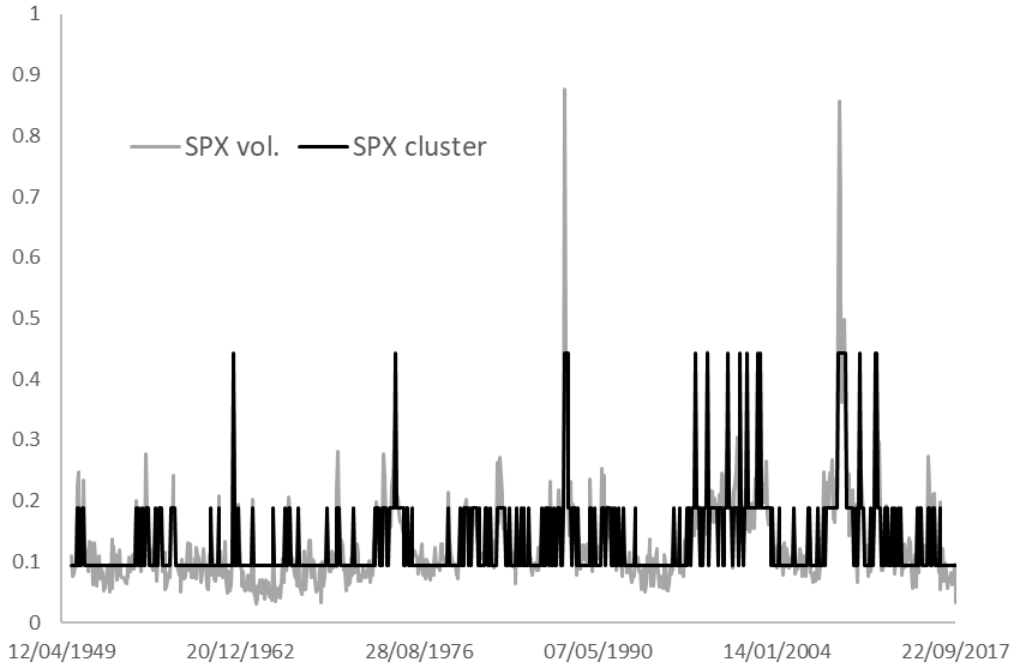
Remember that this KMC analysis completely throws away any time dependence in the behaviour of volatility. However I do have some motivation for using K means on this data and that is that there is a model used in finance in which volatility jumps from one level to another, from regime to regime. And the volatility in this plot seems to have this behaviour.

You see that often the volatility is pretty low, sometimes kind of middling, and occasionally very large. Now in the jump volatility model one has volatility moving between given levels. That's not quite what we have here. Here there is a continuum of volatility levels. But I'm going to pretend that this is because of sampling error, that really there are only three clusters, $K = 3$. Remember this is just an illustrative example. Let's see how it goes. I will find those three clusters and then take this model a little bit further.

The algorithm very quickly finds the following clusters.

	Cluster 1	**Cluster 2**	**Cluster 3**
Number in cluster	586	246	24
SPX volatility	9.5%	18.8%	44.3%

How volatility moves from cluster to cluster:



We can take this a step further and use it to give an idea of the likelihood of jumping from one volatility regime to another. And this is where I cunningly bring back a very simple, weak time dependence.

Rather easily one finds the following matrix of probabilities:

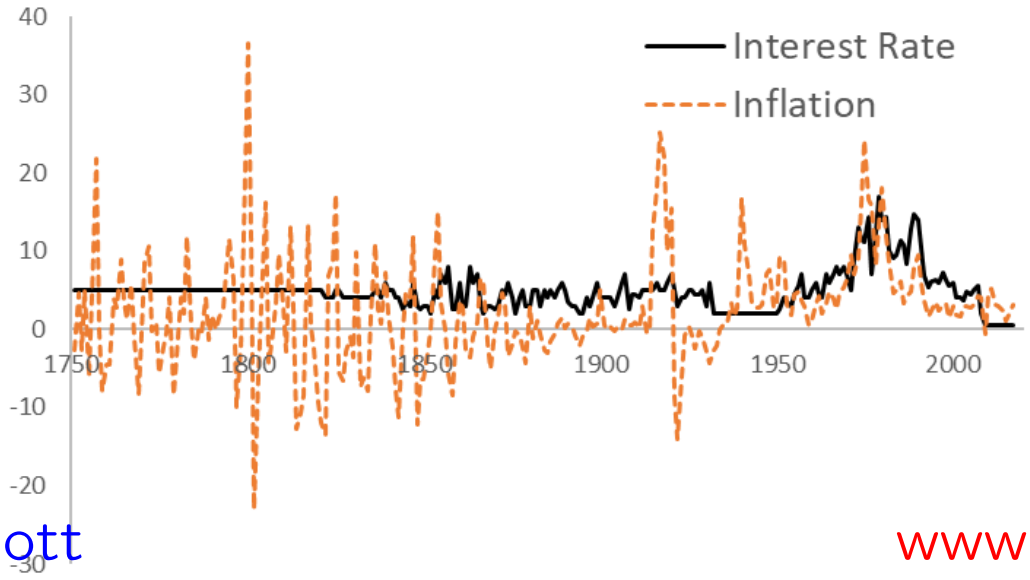
	To:		
From:	**Cluster 1**	**Cluster 2**	**Cluster 3**
Cluster 1	84%	16%	0%
Cluster 2	38%	57%	5%
Cluster 3	0%	54%	46%

We interpret this as, for example, the probability of jumping from Cluster 1 to Cluster 2 is 16% every 30 days.

Example: Interest rates and inflation

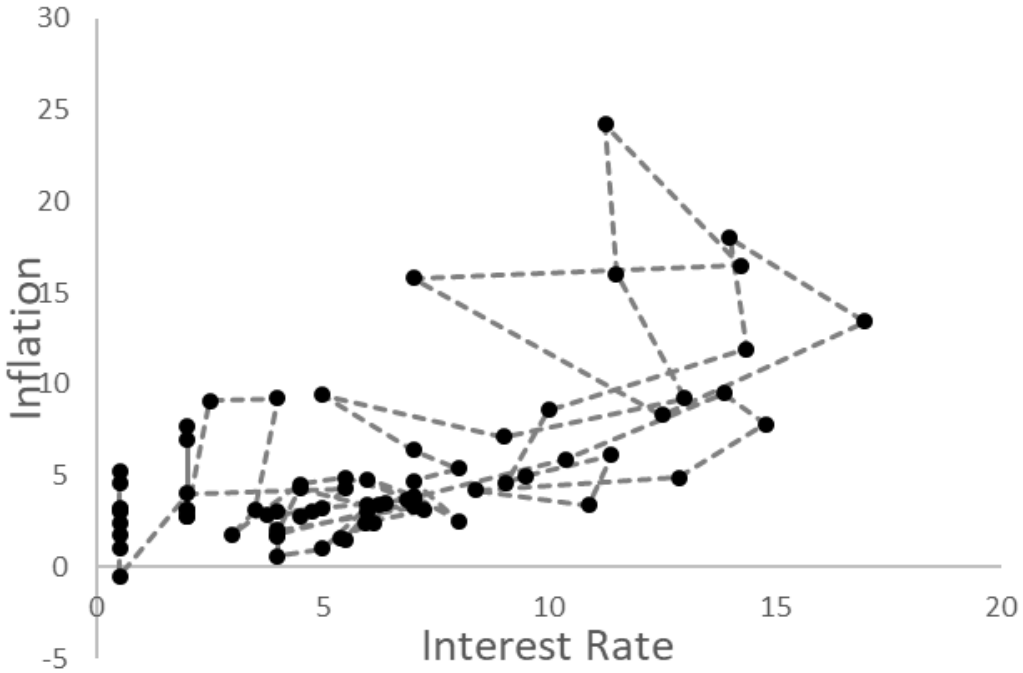
Continuing with a financial theme, or rather economic, let's look at base rates and inflation. This is motivated by how central banks supposedly use base rates to control inflation. There will certainly be an important time component to these two variables, but I'm going to ignore it in this KMC analysis.

I found data for United Kingdom interest rates and inflation going back to 1751.



 www.wilmott.com

It seems unlikely that something simple like K means clustering will do a good job with all of this data so I restricted myself to looking at data since 1945. In this figure I have plotted inflation against the interest rate. I have also joined the dots to show how the data evolves, but, of course, this is not modelled by the simple K means algorithm.

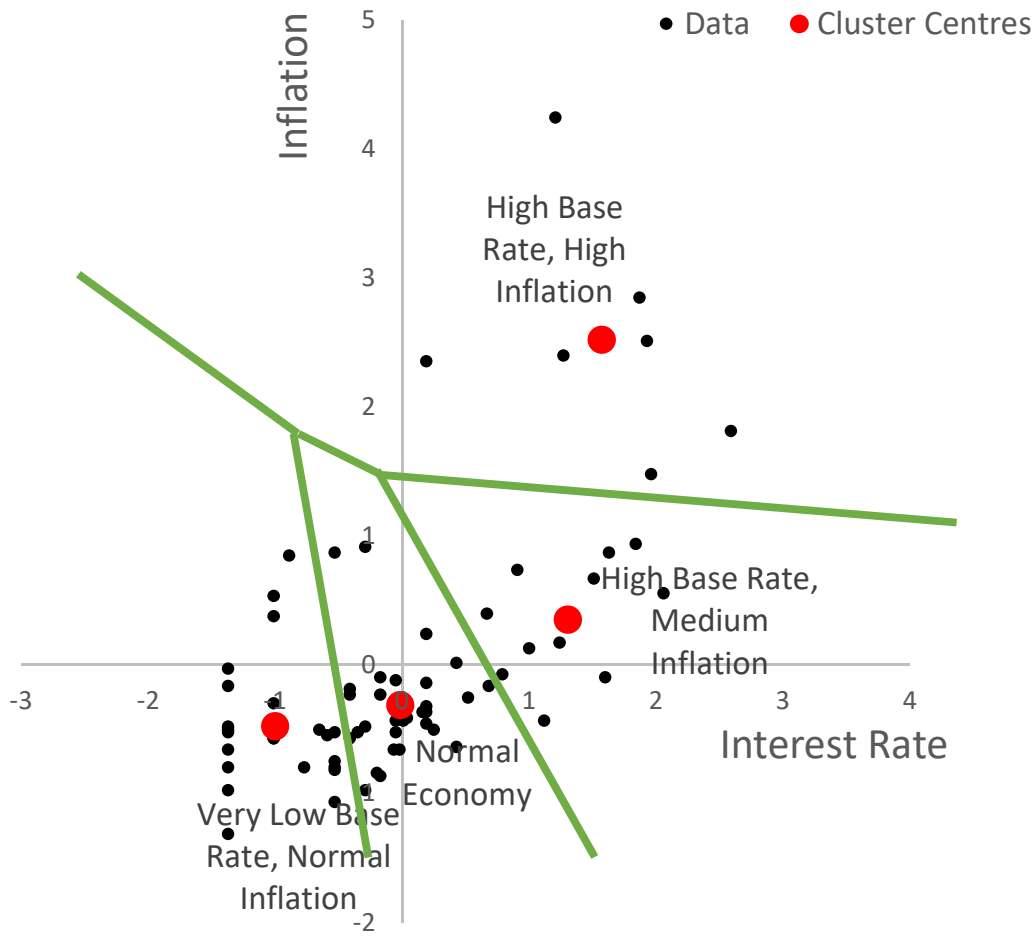


Even though the interest rate and inflation numbers are ballpark the same order of magnitude I still translated and scaled them first.

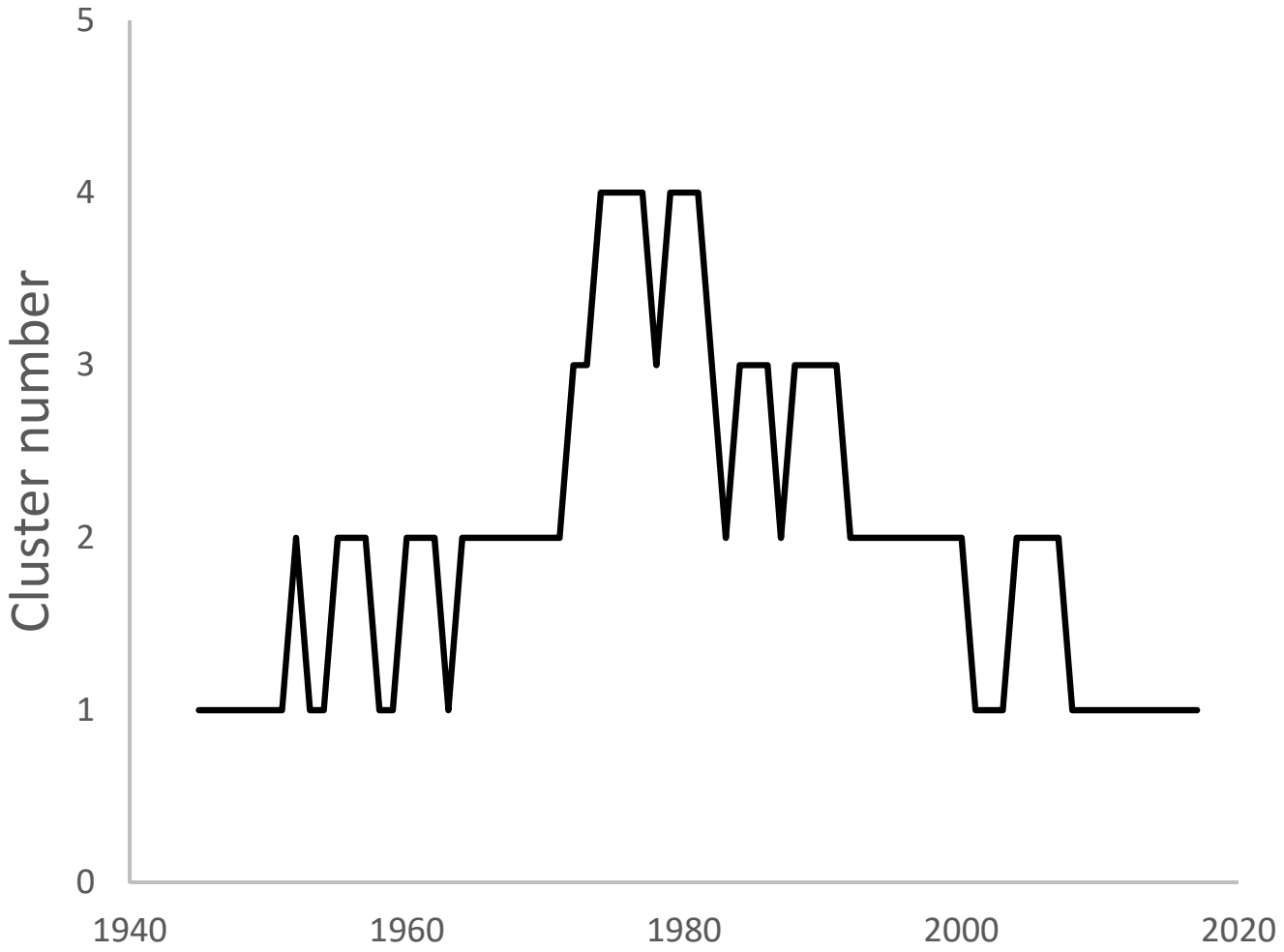
With four clusters we find:

	Cluster 1	**Cluster 2**	**Cluster 3**	**Cluster 4**
No. in cluster	25	30	11	7
Description	V. Low Base Rate, Normal Inflation	'Normal Economy' 'Normal Economy'	High Base Rate, Medium Inflation	High Base Rate, High Inflation
Interest rate	2.05%	6.15%	11.65%	12.77%
Inflation	3.21%	3.94%	6.89%	16.54%

The centres of the clusters and the original data are shown below in the scaled quantities.



How the cluster evolves through time is shown here.

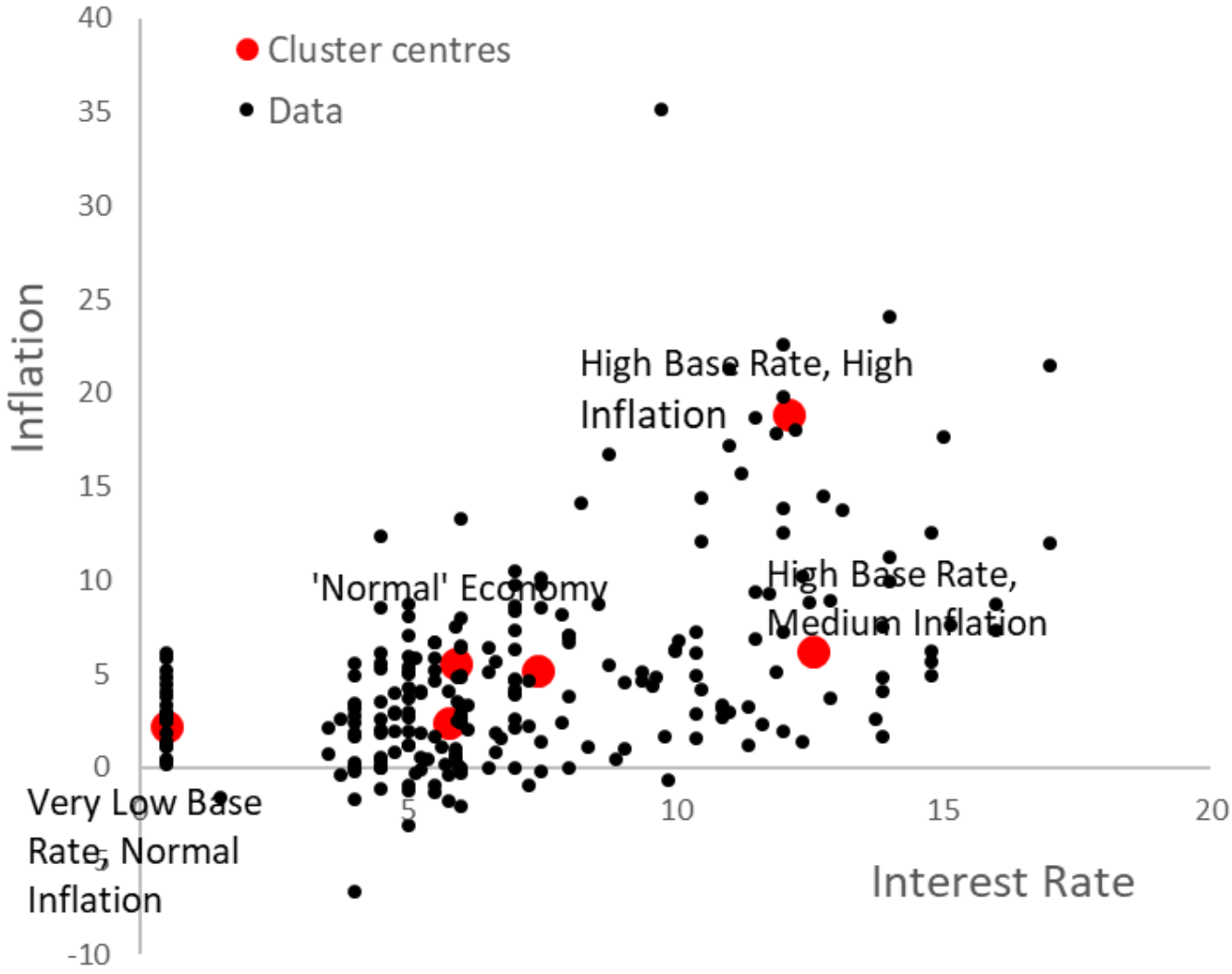


Again we can look at the probability of jumping from one cluster to another.

	To:			
From:	**Cluster 1**	**Cluster 2**	**Cluster 3**	**Cluster 4**
Cluster 1	88.9%	11.1%	0.0%	0.0%
Cluster 2	5.6%	86.1%	8.3%	0.0%
Cluster 3	0.0%	25.0%	58.3%	16.7%
Cluster 4	0.0%	0.0%	33.3%	66.7%

Example: Interest rates, inflation and GDP growth

Take the data in the previous example and add to it data for GDP growth. We now have a three-dimensional problem. I've also used quarterly data to see if that changes the results much. I shall skip straight to the results. With six clusters I found that three of the clusters that I had earlier did not change much but the 'Normal economy' broke up into three separate clusters.



The numbers are:

	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6
Number in cluster	31	107	23	31	17	38
Interest rate	0.50%	5.77%	5.89%	7.44%	12.11%	12.56%
Inflation	2.24%	2.37%	5.59%	5.17%	18.89%	6.18%
GDP growth	0.48%	0.71%	-0.63%	2.25%	-0.51%	0.33%

Summary

Please take away the following important ideas

- K means clustering is an unsupervised-learning technique that groups unlabelled data
- It is simple to implement
- The centres of the clusters are found by iteration