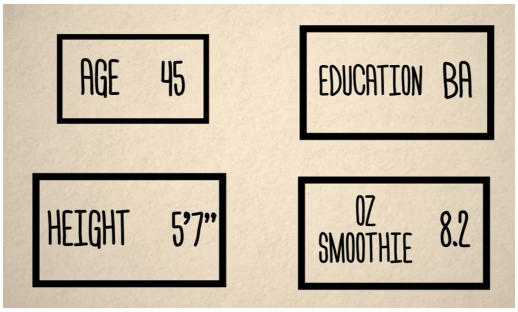
Beyond K-Means: Data and Preprocessing

We have discussed different ways of troubleshooting the K-Means algorithm.

Now we will take a closer look at our **Data**.
In the archaeology example that we have discussed before, there we imagined our example dataset as the locations of the archaeology artifacts.In particular, for each data point we recorded a distance in east from a marker and the distance in north from the same marker.
But there are a lot of other types of data out there that we might collect. So, how could we know when the data is ready to run K-Means?
Before running K-Means on our data there are a bunch of questions that we should ask ourselves which are:

1. **Is our data featurized?**

Example: Suppose we have made a fitness app and we have collected a lot of data about our users. Like their age, height, education and post frequency on the app forum.
But K-Means needs a vector of data. So, the first thing that we have to do is to turn all of this information into an orderly vector. Each entry in the vector will have a particular meaning. Each of these entries is called a feature.



We can imagine this as a big table or matrix that collects these features across all of the users of the app. Once we organize our data like this, then it becomes featurized.

FEATURIZED	AGE	HEIGHT	EDUCATION	OZ SMOOTHIE
PERSON 1	45	5'7"	BA	8.2
PERSON 2				
...				
PERSON N				

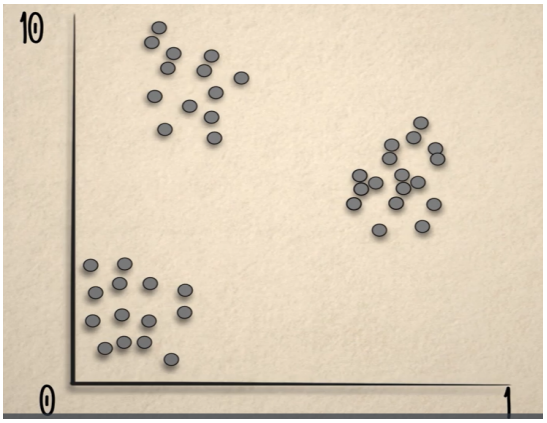
2. We check in the matrix is each of our features a continuous number?

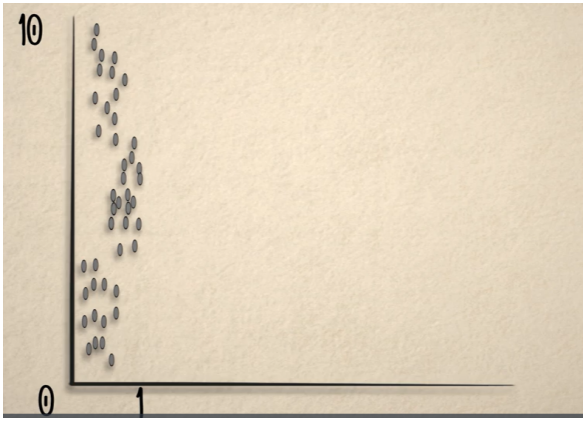
Now K-Means also needs its features to be valued as continuous numbers. So, the next thing we check in the matrix is each of our features a continuous number?
In our example, age is in years, so it should not be changed. We can convert heights into inches. We can express education as years of education(for example, we write 16 in the education column for Bachelors Degree) and smoothie quantity type also need not to be changed.

	AGE	HEIGHT	EDUCATION	OZ SMOOTHIE
PERSON 1	45	67	16	8.2
PERSON 2				
...				
PERSON N				

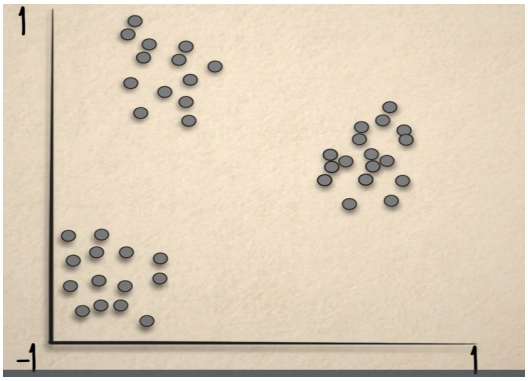
3. Are the numbers in the feature matrix commensurate?

As K-Means assumes that we have spherical clusters. So, the clusters have basically the same width in all directions. Consider the figure in right for a given dataset. Here the Y-axis ranges from 0 to 10 and X axis ranges from 0 to 1. So, if we plot the axes on the same scale, then the data look like the below figure, to the K-Means algorithm.



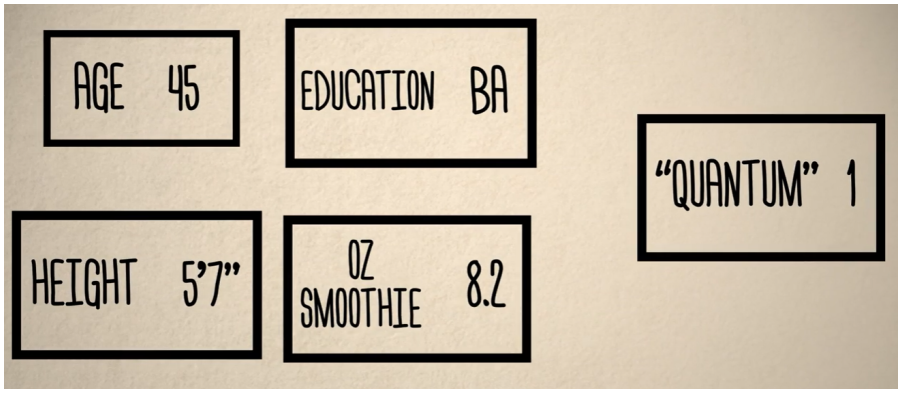


So we want to put all the data on the same scale and for this typically a standardization or a normalization is used. For instance all data on each feature can be separately normalized to lie between -1 and 1.
A common alternative is to standardize the data in each feature to have a standard deviation of 1.

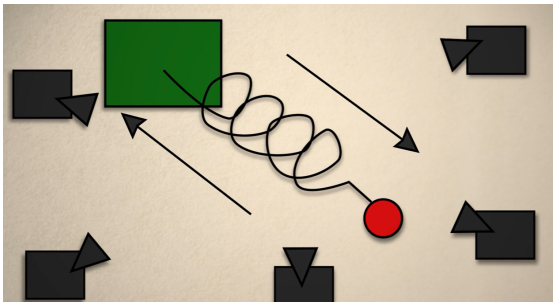


4. Are there too many features?

For instance, if we have a fitness app, then we might want to find different groups of individuals who have similar health and fitness needs. But suppose we have added a bunch of useless features, like for every word in the English dictionary, we add a feature for how many times this particular person uses that word in the forums.



A lot of times we have columns in our data that are not entirely useful to the purpose at hand. When we know which features are not useful because of some domain knowledge, then we want to get rid of them beforehand. But sometimes we don't know which are the useful and useless features. In that case, there are some algorithms that let us reduce the dimension of our feature space.
So we basically reduce the number of features to get the most important features out.



The most widely used algorithm for this is, **Principal Component Analysis(PCA).**

Example:

Suppose, we have a spring and a ball is moving back and forth on that spring. And we record this motion with a bunch of different cameras (Suppose we have 5 cameras). But the motion of the spring is just one dimensional. So, here we have 4 extra features. PCA will help us pick out the feature that really matters, even though it doesn't necessarily correspond to just one camera.

So, it's always a good idea to run PCA on our data before running K-Means. So, PCA is a preprocessing step for K-Means.

5. Can we use domain knowledge?

So, we need to check if there are any domain specific reasons to change the features. If we already know, it's best to get rid of any feature that we know in advance which won't be helpful for our final problem.

Any domain specific feature change should be done before running PCA or K-Means or any other algorithm. It's also best to change any features that might not satisfy the assumptions of K-Means. For instance, we discussed previously that there might be transformations of our features to something that better fits the assumption of K-Means.

It's always good to automate everything using Machine Learning and Statistics. But if there is any domain specific knowledge that we can use specific to our problem area then sometimes it can make a big difference in result.

So, in summary, in this article we have discussed preprocessing the dataset, so we can run the K-Means algorithm with the best possible result.