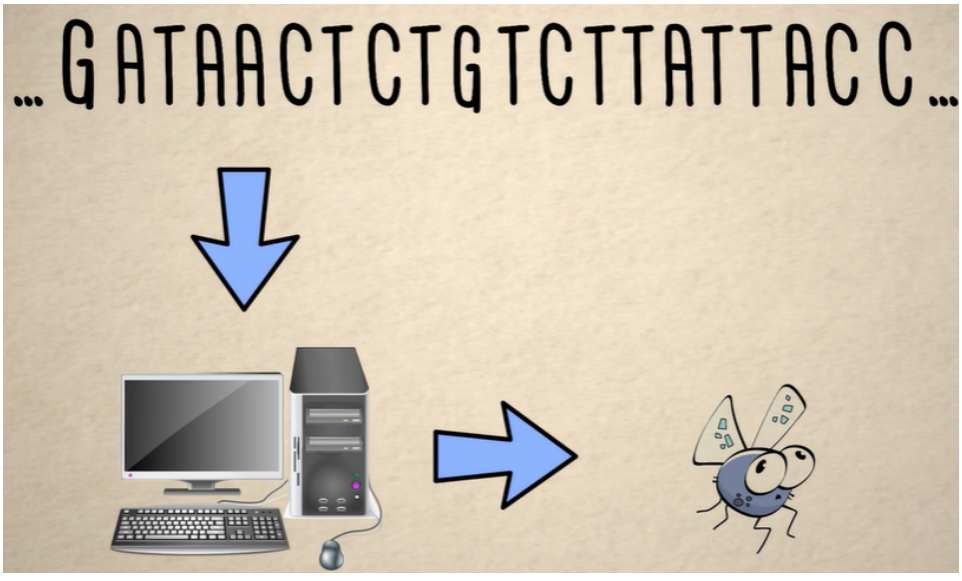
Understanding the Genetic Code

In the previous videos, we've introduced unsupervised learning. We saw that clustering is a particular form of unsupervised learning and that K-means clustering is a particular form of clustering. Now we'll explore K-means clustering in an extended application courtesy of Alexander Gorban and Andrei Zinovyev, which is **Understanding the Genetic Code.**

Recall that DNA is essentially a sequence of letters: A's, C's, Gs, and T's.

You typically learn in a biology class that these letters form a sort of code, a set of instructions for how to grow and maintain a living organism.



But how can we interpret the sequence of letters as a set of instructions?

Well, any English sentence is a sequence of letters and the units of meaning within that sentence are words. So we might hypothesize that a DNA sequence is similarly composed of words.

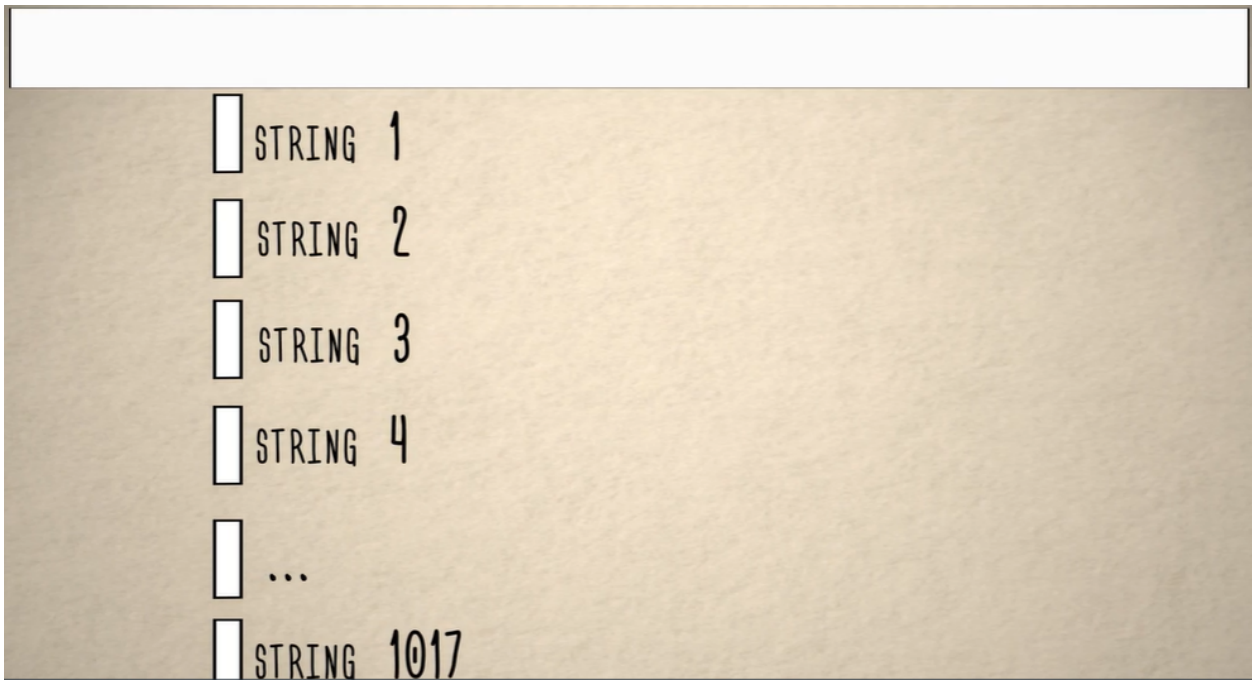
A particularly simple hypothesis is to assume that the words are all of the same lengths. We might first guess that the words in DNA are either 1, or 2, or 3, or 4 letters long. So the question we want to answer is, is it true that DNA breaks down into meaningful words at the same length? And if so, how long are the words? We're going to see how we can use K-means to help answer these questions.

First, we gather up some real DNA, a fragment of the full DNA sequence of a particular bacteria: the name of the bacteria is **Caulobacter Crescentus**

The total fragment is made up of about 300,000 letters (305,100 to be exact).

Next, we break this whole sequence into strings of 300 letters each, and we make sure these strings don't overlap at all. We will think of these strings like a data point.

So in this dataset, we end up with 1017 data points.



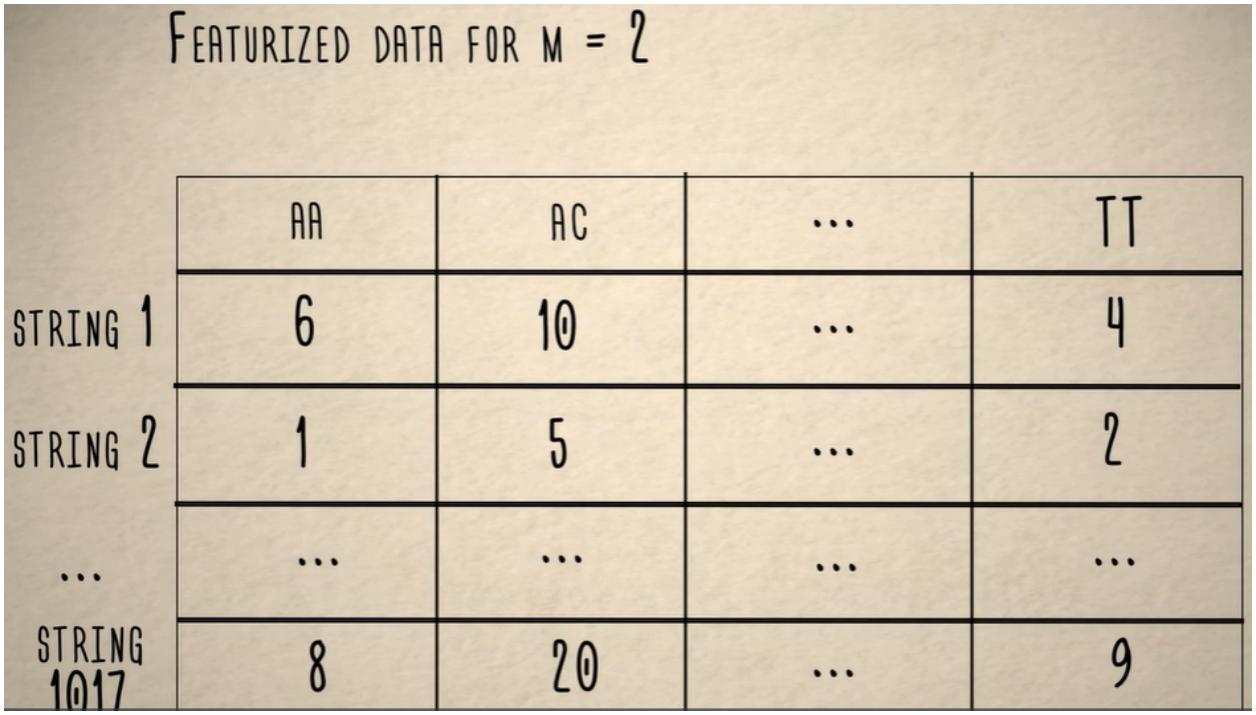
Let, M be the proposed word length. So if M = 2, we expect each word to be two letters long.

For each value of M, we're actually going to make a different dataset. Recall that in order to run K-means, we need to featurize our data i.e we need to come up with a vector of numerical features for each data point. Here's something we can do. For any value of M, we can divide the DNA string up into substrings of that length. For each possible word like 'AC', we will count how many times that word occurs i.e how many substrings are the same as that word?

Consider M = 2 for a moment.
Now, how many possible words are there?

Well, there are four possible letters in the first slot and four possible letters in the second slot. **So there are 4^2 or 16 possible words.**

So for the M = 2 dataset, each data point will have 16 features associated with it.
For instance, the AC feature will tell us how many times the two-letter word 'AC' occurs in the string.



Now, these counts only take integer values. They're not exactly continuous values, but they're close enough.

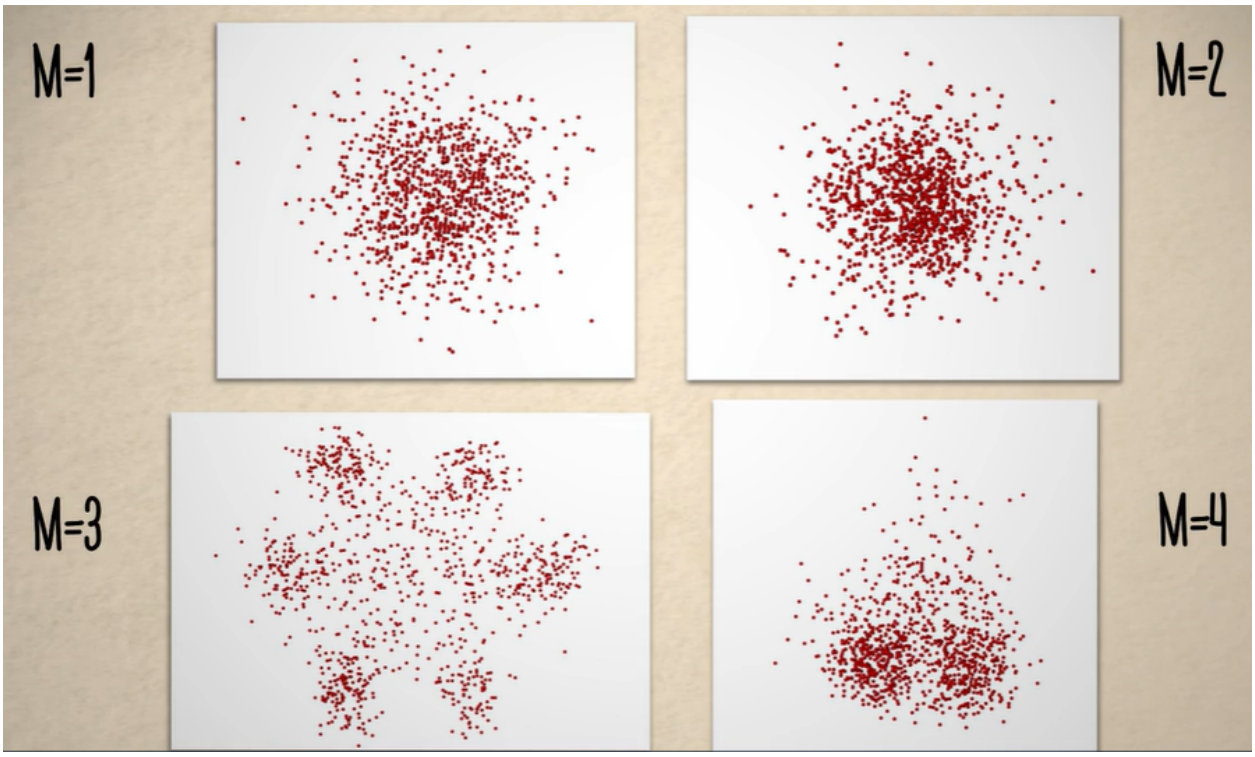
Okay, so if our dataset only had words of length one (M=1), it would only have four features.
Our data set for M=2 has 4^2 or 16 features.
Our dataset for M=3 has 4^3 or 64 features.
And our data set for M=4 has 4^4 or **256 features**. That's a lot of features.

Even for M=1, 4 features are hard to visualize. As humans, we basically only ever look at plots of two dimensions, even trying to visualize in three dimensions is pushing it.

So 4 features is already a lot and it just gets worse as the proposed word lengths get longer.

Since we can't really visualize these datasets as they are, let's run PCA (Principal Component Analysis). We'll pick out the top two features for PCA. Also, remember, these don't have to correspond to any new features in our dataset. They can be wholly new composite features.

Julia Lye, an awesome engineering student, did the coding and plotting for this case study as well as the next. Here Julia plotted these two PCA features for each data set. We immediately see something really interesting.



There's not much going on for M=1 or M=2, the data just form a big blob. And that's also pretty true for M=4.

But there's a lot of clear structure and symmetry for words of length three.

So from here on out, **let's concentrate on M=3.**

And now let's run K-means. Remember, we still need to standardize or normalize our data.

So **let's first standardize each feature** by subtracting the empirical feature mean across the dataset and dividing by the empirical feature standard deviation.

$$x_{1,aaa} = (N_{1,aaa} - \text{mean}_{aaa})/\text{std}_{aaa}$$

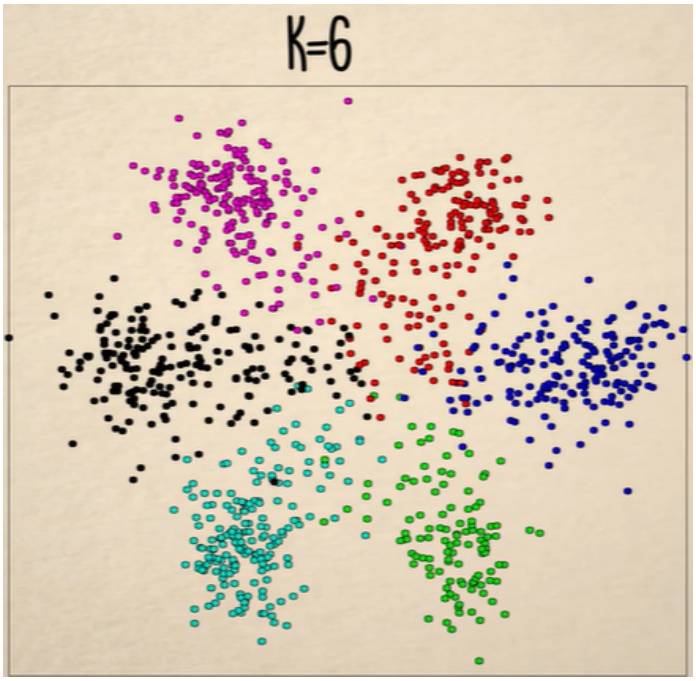
Now we'll take that standardized data and run K-means.

So how should we choose K?
From a visual examination of the PCA plot from earlier, it looks like there are six or seven clusters.

Let's try each value of K, and see what happens.

First, we try K-means with K = 6, so that the 6 clusters are identified in the high-dimensional (64 dimensions because M=3) dataset, and then we visualize the end result using the first two components of PCA.

Remember, **we have run K-means clustering on the full 64 feature dataset,** and only later are we visualizing the clusters in 2 dimensions by using the first two principal components.

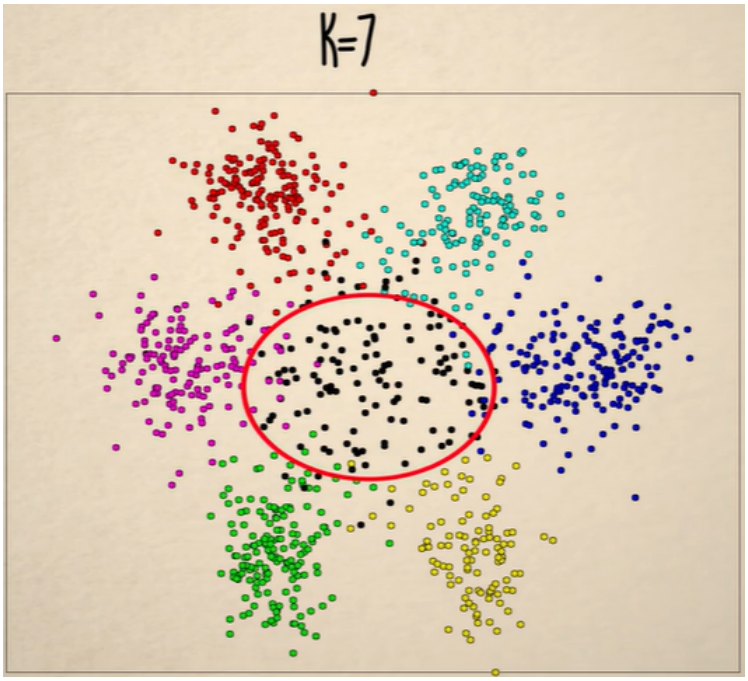


We actually ran K-means clustering on the 64-dimensional dataset, and are only visualizing the clusters here in 2 dimensions using PCA.

The fact that those 64-dimensional clusters are still so cleanly distinguishable even in 2 dimensions is a great result and is much more difficult to come by than just running K-means on the reduced 2-dimensional PCA dataset, where it would have been easier to spot the clusters in 2 dimensions. Remember, we didn't just run K-means on the two dimensions that we were visualizing, but rather, we did it on the 64-dimensional dataset of M=3.

So it's a meaningful and great sign that the K-means results align so well with the structure we're seeing in this visualization.

Now let's try K =7.

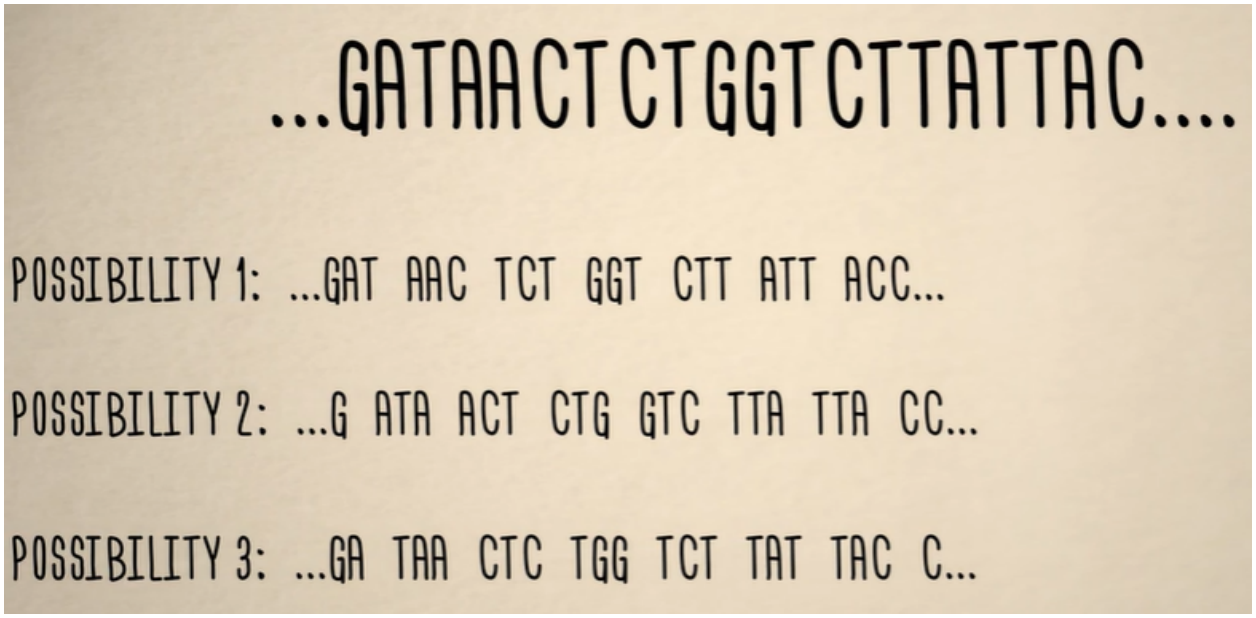


In this case, we pick up even more structure. We're able to almost perfectly separate out the middle part from the six lobes on the outside in the visualization.

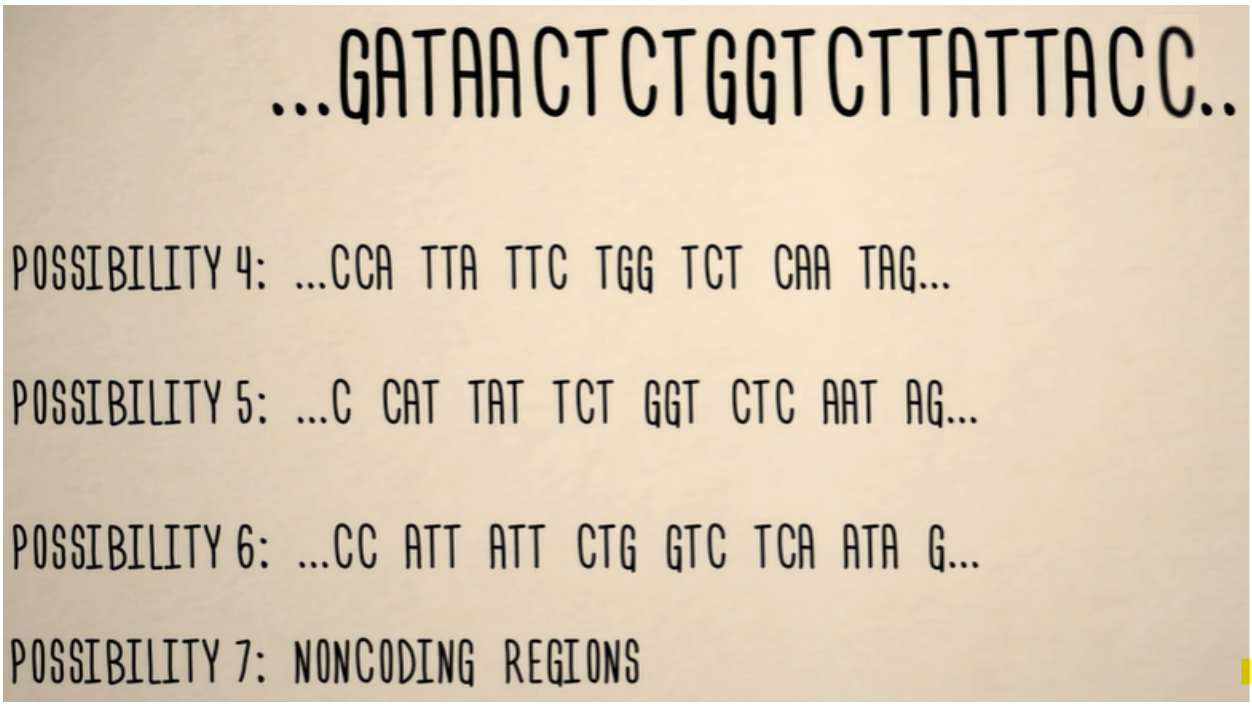
So what's going on here? What does this result mean?

Well, let's talk a little bit more about DNA. Suppose the words are really of length three. As we've been assuming, and running K-means on M = 3 data, then what would we expect to see?

Well, we don't know exactly where these words start. So there are three possible starting points we might pick up. When we divided up DNA into words, we might have started on the first letter of the word, the second letter of the word (discounting the first letter), or the third letter of the word (discounting the first two letters).



And actually, **we don't really know that all the genes read forward for the direction we chose for our strings. Some might read backward, too.** So we might start on the first letter backward, the second letter backward, or the third letter backward.
And finally, **there might be some non-coding regions of DNA**, DNA that doesn't actually have any instructions and acts as a filler.



We've seen from the other M plots like for M = 1 and M = 2, that filler kind of looks like a center blob. And that's exactly what we're seeing here.

We see six lobes, one for each of the possible directions, we might read meaningful words in DNA, **three directions forward, and three directions backward.**

And the seventh blob in the middle represents words in non-coding DNA.

In fact, now that we've run K-means we can go back into each of our data points, each of our DNA string segments, and say whether it's non-coding or coding, and which of the six ships it shares with other strings based on the results of K-means.

So **we've seen strong evidence that DNA is composed of words of three letters each,** as well as non-coding bits. **In fact, these three-letter words are now known as codons, and scientists know that they basically encode amino acids,** which are the building blocks for all the proteins created in living organisms. **We were able to show this with just a dataset of a DNA sequence fragment.** And we're also able to identify which of our strings are likely coding regions and non-coding regions, and which have similar offsets.

In this case study, we've seen how quickly and easily we can use K-means clustering and PCA, and other powerful unsupervised learning to get results that lend deep insights into a scientific problem. And in the next case study, we'll look at another data type that is ubiquitous these days, which is human-generated text.