In our previous video, we found the distance matrix, which computes the pairwise distances between all the intensity values in the flower vector.

Now we can cluster the intensity values using hierarchical clustering.

So we're going to type "cluster intensity." And then we're going to use the hclust function, which is the hierarchical clustering function in R, which takes as an input the distance matrix.

And then we're going to specify the clustering method to be "word." As a reminder, the "words" method is a minimum variants method, which tries to find compact and spherical clusters.

We can think about it as trying to minimize the variance within each cluster and the distance among clusters.

Now we can plot the cluster dendrogram.

So-- plot(clusterIntensity).

And now we obtain the cluster dendrogram.

Let's have here a little aside or a quick reminder about how to read a dendrogram and make sense of it.

Let us first consider this toy dendrogram example.

The lowest row of nodes represent the data or the individual observations, and the remaining nodes represent the clusters.

The vertical lines depict the distance between two nodes or clusters.

The taller the line, the more dissimilar the clusters are.

For instance, cluster D-E-F is closer to cluster B-C-D-E-F than cluster B-C is.

And this is well depicted by the height of the lines connecting each of clusters B-C and D-E-F to their parent node.

Now cutting the dendrogram at a given level yields a certain partitioning of the data.

For instance, if we cut the tree between levels two and three, we obtain four clusters, A, B-C, D-E, and F.

If we cut the dendrogram between levels three and four, then we obtain three clusters, A, B-C, and D-E-F.

And if we were to cut the dendrogram between levels four and five, then we obtain two clusters, A and B-C-D-E-F.

What to choose, two, three, or four clusters?

Well, the smaller the number of clusters, the coarser the clustering is.

But at the same time, having many clusters may be too much of a stretch.

We should always have this trade-off in mind.

Now the distance information between clusters can guide our choice of the number of clusters.

A good partition belongs to a cut that has a good enough room to move up and down.

For instance, the cut between levels two and three can go up until it reaches cluster D-E-F. The cut between levels three and four has more room to move until it reaches the cluster B-C-D-E-F. And the cut between levels four and five has the least room.

So it seems like choosing three clusters is reasonable in this case.

Going back to our dendrogram, it seems that having two clusters or three clusters is reasonable in our case.

We can actually visualize the cuts by plotting rectangles around the clusters on this tree.

To do so, we can use the rect.hclust function, which takes as an input clusterIntensiy, which is our tree.

And then we can specify the number of clusters that we want.

So let's set k=3.

And we can color the borders of the rectangles.

And let's color them, for instance, in red.

Now going back to our dendrogram, now we can see the three clusters in these red rectangles.

Now let us split the data into these three clusters.

We're going to call our clusters, for instance, flowerClusters.

And then we're going to use the function cut tree.

And literally, this function cuts the dendrogram into however many clusters we want.

The input would be clusterIntensity.

And then we have to specify k=3, because we would like to have three clusters.

Now let us output the flower clusters variable to see how it looks.

So flowerClusters.

And we see that the flower cluster is actually a vector that assigns each intensity value in the flower vector to a cluster.

It actually has the same length, which is 2,005, and has values 1, 2, and 3, which correspond to each cluster.

To find the mean intensity value of each of our clusters, we can use the tapply function and ask R to group the values in the flower vector according to the flower clusters, and then apply the mean statistic to each of the groups.

What we obtain is that the first cluster has a mean intensity value of 0.08, which is closest to zero, and this means that it corresponds to the darkest shape in our image.

And then the third cluster here, which is closest to 1, corresponds to the fairest shade.

And now the fun part.

Let us see how the image was segmented.

To output an image, we can use the image function in R, which takes a matrix as an input.

But the variable flowerClusters, as we just saw, is a vector.

So we need to convert it into a matrix.

We can do this by setting the dimension of this variable by using the dimension function.

So, let's use the dimension function, or dim, which takes as input flowerClusters.

And then we're going to use the combined function, or the c function.

And its first argument will be the number of rows that we want for the matrix, and that would be 50.

And the second argument would be the number of columns.

Why did we use 50?

Simply because we have a 50 by 50 resolution picture.

Now pressing Enter, and flowerClusters looks like a matrix.

Now we can use the function image, which takes as an input the "flower cl clusters" matrix.

And let's turn off the axes by writing axes="false." And now, going back to our graphics window, we can see our segmented image here.

The darkest shade corresponds to the background, and this is actually associated with the first cluster.

The one in the middle is the core of the flower, and this is cluster 2.

And then the petals correspond to cluster 3, which has the fairest shade in our image.

Let us now check how the original image looked.

Let's go back to the console and then maximize it here.

So let's go back to our image function, but now this time the input is the flower matrix.

And then let's keep the axis as false.

But now, how about we add an additional argument regarding the color scheme?

Let's make it grayscale.

So we're going to take the color, and it's going to take the function gray.

And the input to this function is a sequence of values that goes from 0 to 1, which actually is from black to white.

And then we have to also specify its length, and that's specified as 256, because this corresponds to the convention for grayscale.

And now, going back to our image, now we can see our original grayscale image here.

You can see that it has a very, very low resolution.

But in our next video, we will try to segment an MRI image of the brain that has a much, much higher resolution.