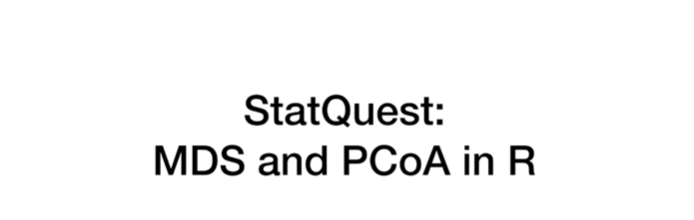
<https://www.youtube.com/watch?v=pGAUHhLYp5Q&list=PLblh5JKOoLUICTaGLRoHQDuF_7q2GfuJF&index=31>

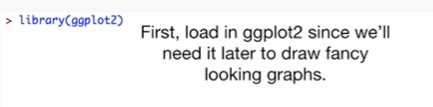


Today we're going to talk about doing multi-dimensional scaling MDS and principle coordinate analysis PCoA in R.

If you don't already know MDS, or classical, or metric MDS is the exact same thing as PCoA.

One last thing before we move on, the code that I use in the stack quest is available on the stack quest website.

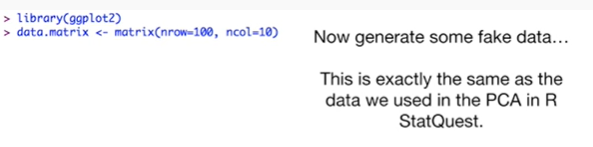
And the link to that code is in the description below.



First, we load in ggplot2 since we'll need it later to draw fancy looking graphs.



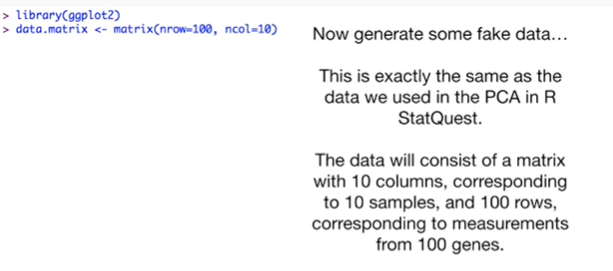
Now we generate some fake data.



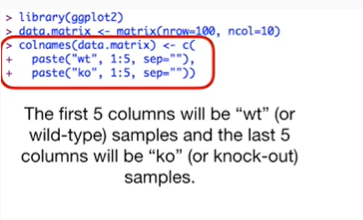
This is exactly the same as the data we used in the PCA in our stat quest.

So I'm going to breeze through this pretty quickly.

If you need more details check out that stat quest.



The data will consist of a matrix with ten columns corresponding to ten samples and 100 rows corresponding to measurements from 100 genes.



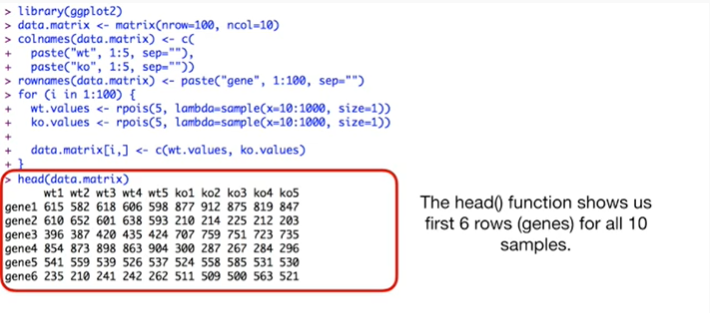
The first five columns will be wt (or wild-type samples) and the last five columns will be ko (or knockout) samples.



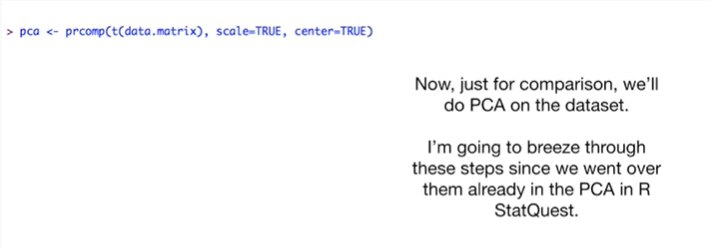
The genes will have really creative names like gene one gene two.



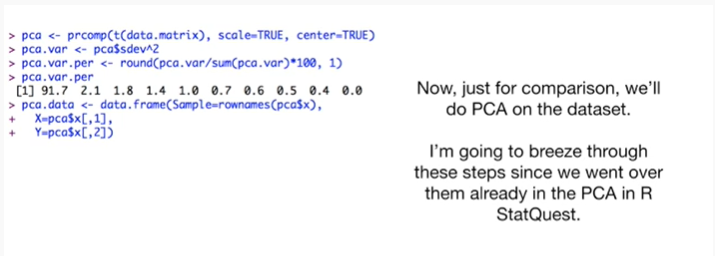
This is where we generate the fake data.



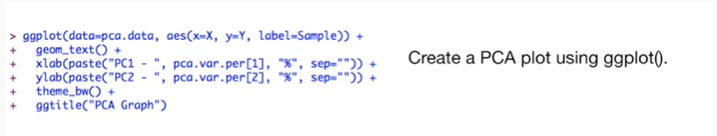
And the head function shows us the first six rows or genes for all ten samples.



Now just for comparison will do PCA.



On the data set I'm going to breeze through these steps since we went over them already in the PCA in our stat quest.

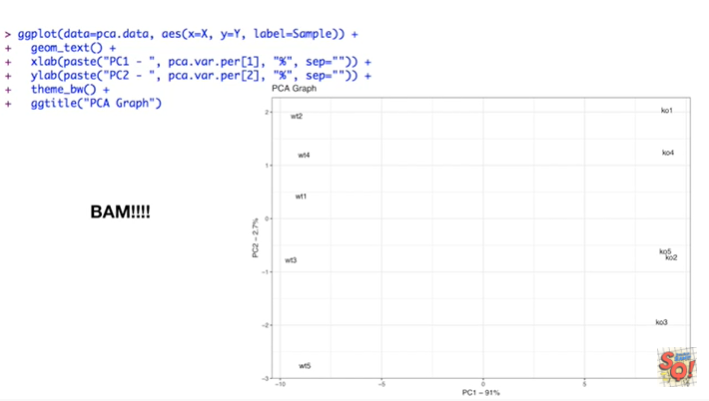


Now we create a PCA plot using ggplot().

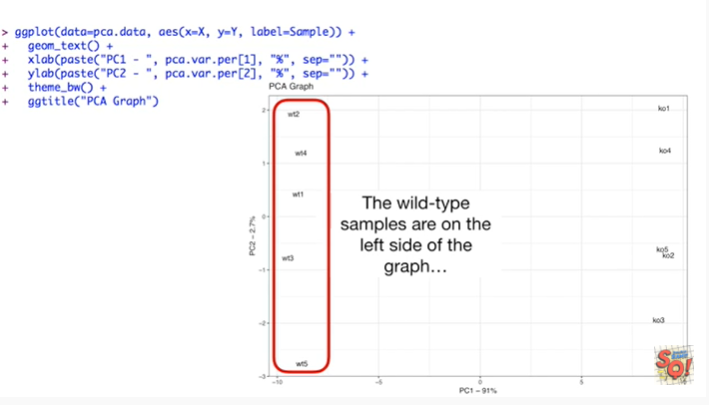


Note : we covered this command in the PCA and our stat quest.

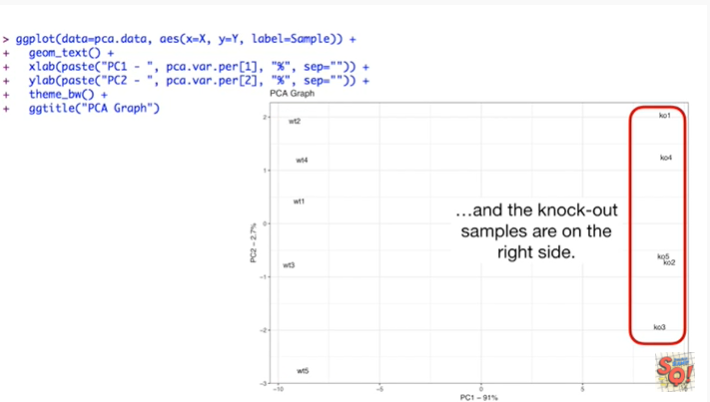
So check that out if this looks totally crazy.



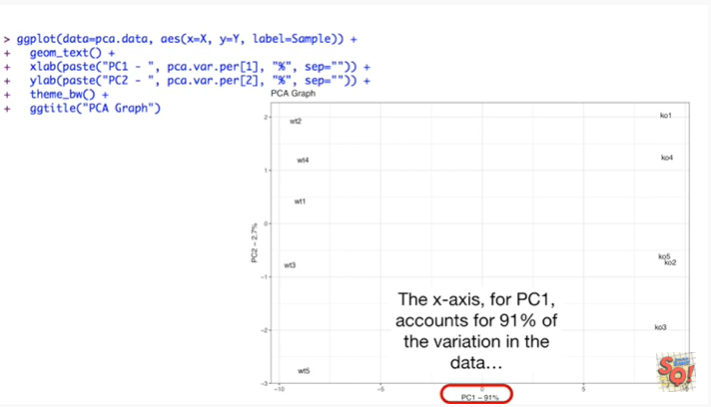
BAM !!!



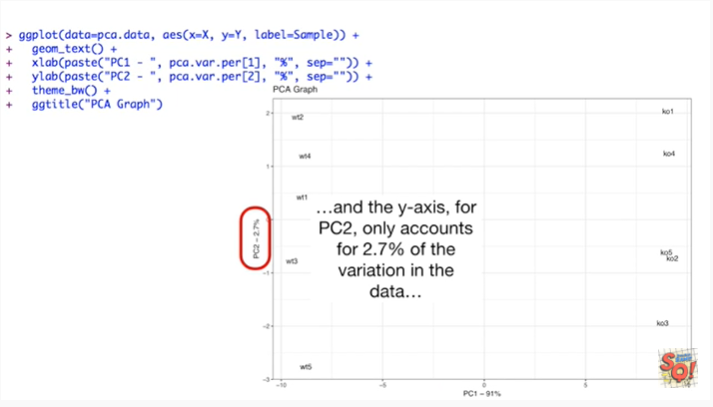
The wild-type samples are on the left side of the graph



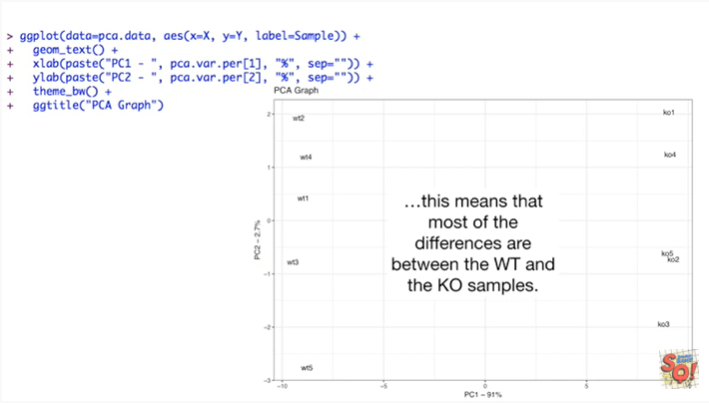
and the knockout samples are on the right side.



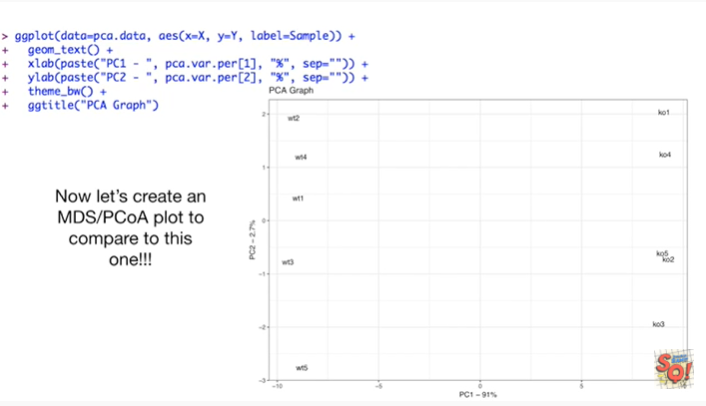
The x-axis PC1, for the first principal component, accounts for 91% of the variation in the data.



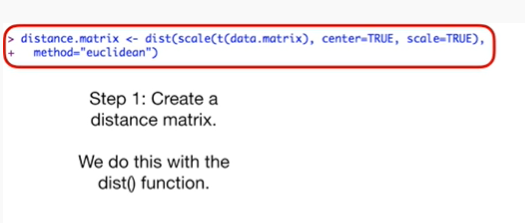
And the y-axis for PC2, for principal component 2, only accounts for 2.7 percent of the variation in the data.



This means that most of the differences were between the wild type and the knockout samples.



Now let's create an MDS or PCoA plot to compare to this one !!!

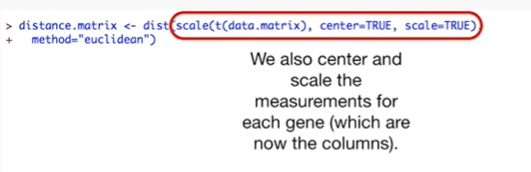


Step one : create a distance matrix.

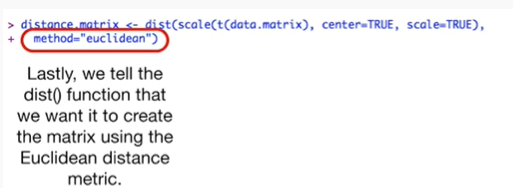
We do this with the dist() function.



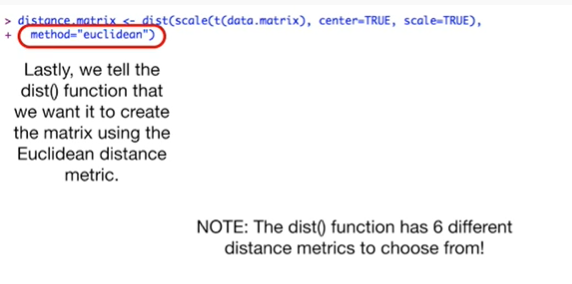
Just like with PCA we transpose the matrix so the samples are rows.



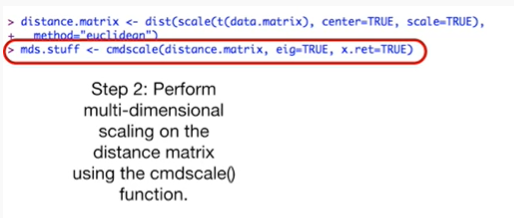
We also Center and scale the measurements for each gene which are now the columns.



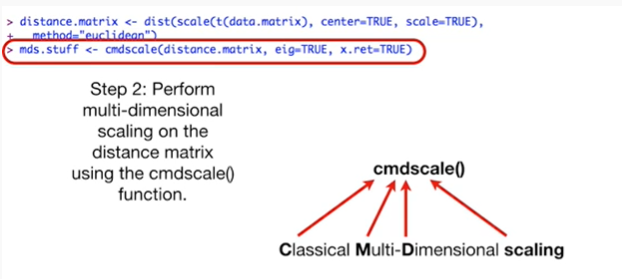
Lastly, we tell the disk function that we want to create the matrix using euclidean distance metric.



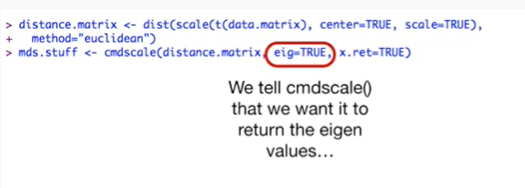
Note : the dist() function has six different distance metrics to choose from.



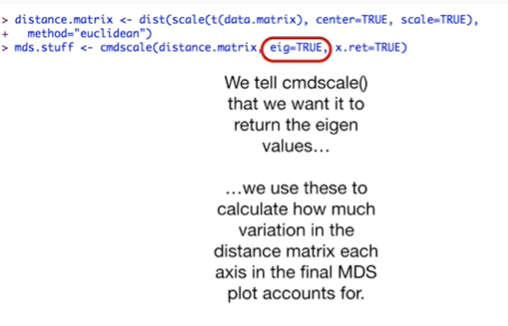
Step 2 : perform multi-dimensional scaling on the distance matrix using the cmdscale() function.



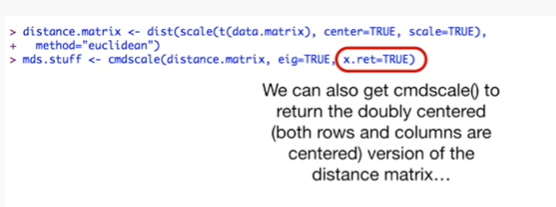
Cmdscale() stands for classical multi-dimensional scaling.



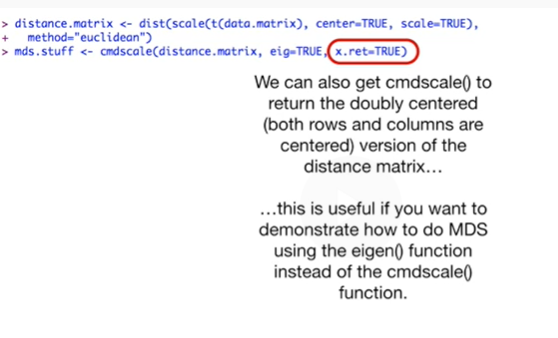
We tell cmdscale() that we want it to return the eigen values



we use these to calculate how much variation in the distance matrix each axis in the final MDS plot accounts for.

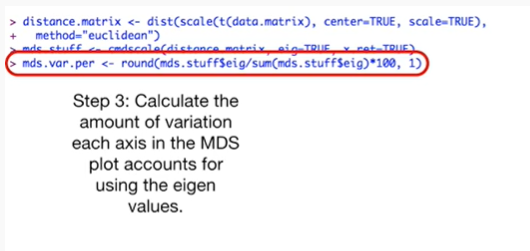


We can also get Cmdscale() to return the doubly centered ie both rows and columns are centered version of the matrix

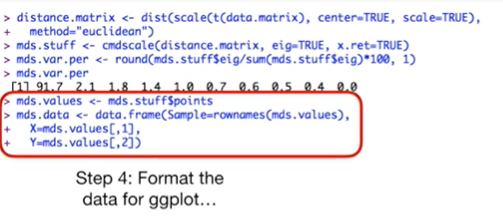


this is useful if you want to demonstrate how to do MDS using the eigen function instead of the Cmdscale() function.

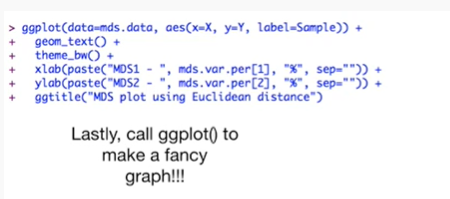
Originally I thought I was going to demonstrate how to use the eigen function to do multi-dimensional scaling, but in the end I really wanted to keep this practical and you're gonna do MDS you're gonna use the Cmdscale() function.



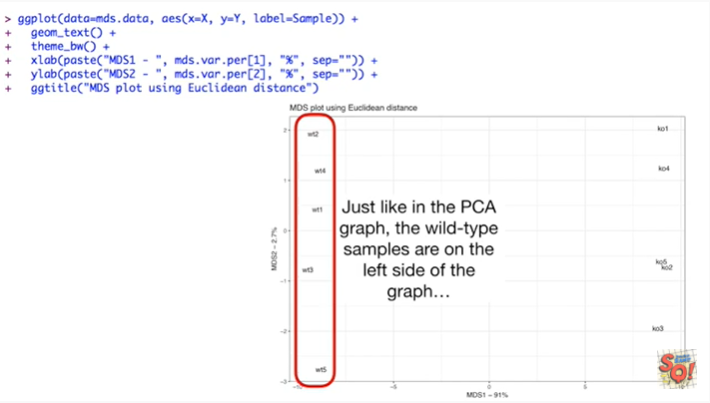
Step 3 : calculate the amount of variation each access in the MDS plot accounts for using the eigenvalues.



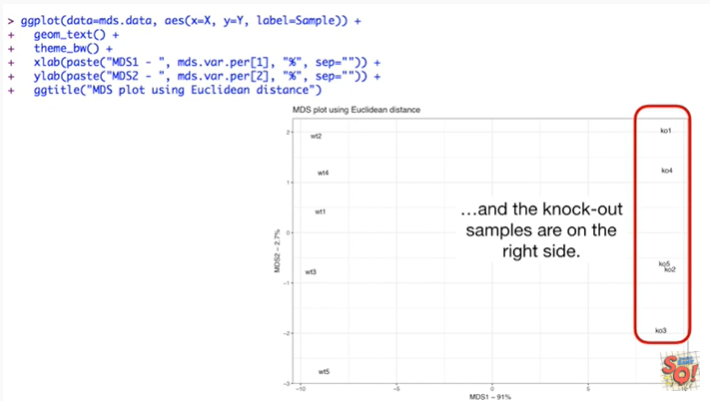
Step 4 : format the data for ggplot().



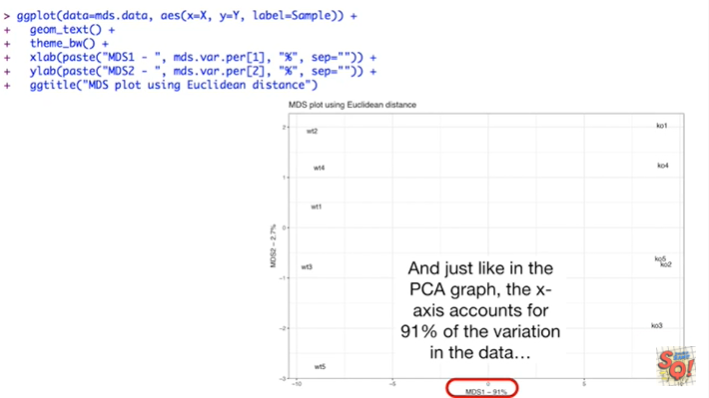
Lastly, call ggplot() to make a fancy graph.



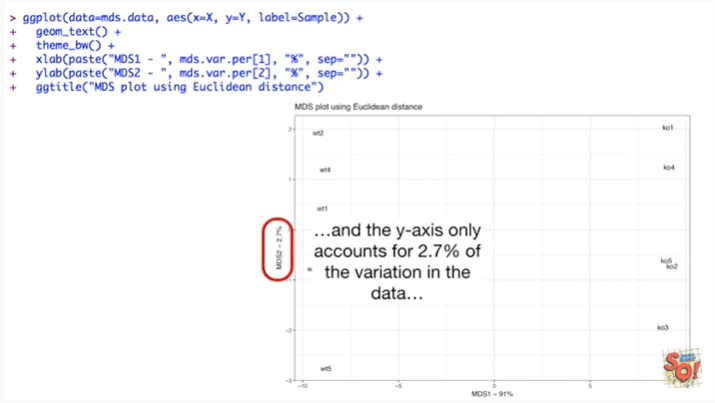
Just like in the PCA graph, the wild-type samples are on the left side of the graph



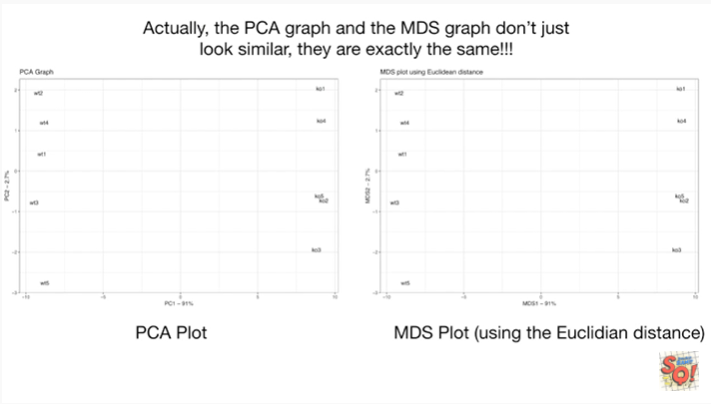
and the knockout samples are on the right side.



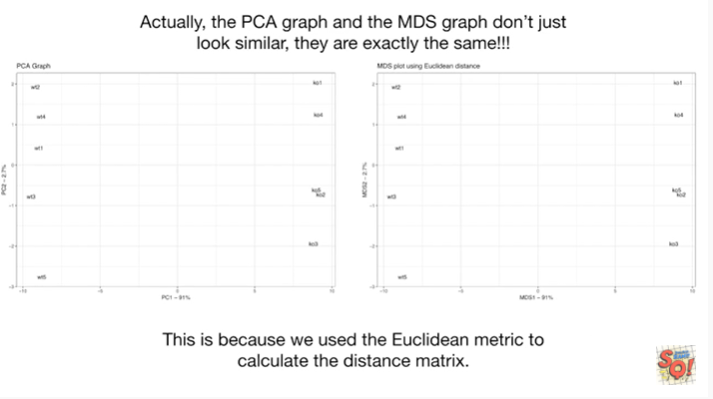
And just like in the PCA graph the X access accounts for 91% of the variation in the data



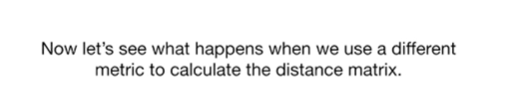
and the y axis only accounts for 2.7 percent of the variation in the data.



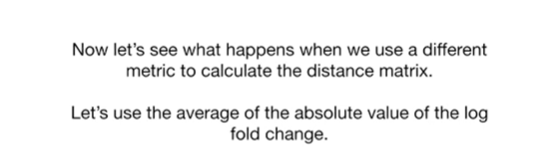
Actually the PCA graph and the MDS graph don't just look similar, they are exactly the same !!!



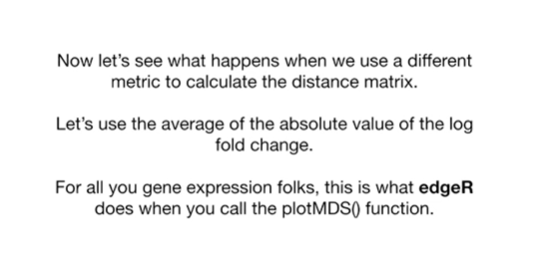
This is because we use the Euclidean metric to calculate the distance matrix.



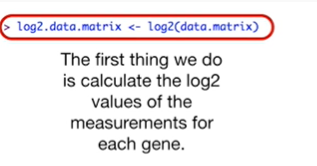
Now let's see what happens when we use a different metric to calculate the distance matrix.



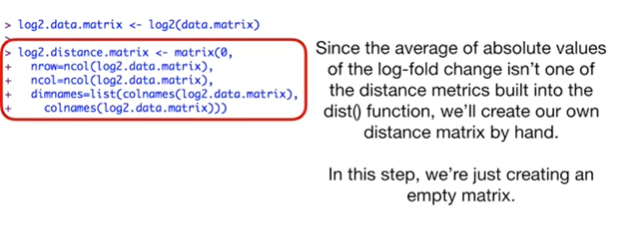
Let's use the average of the absolute value of the log fold change.



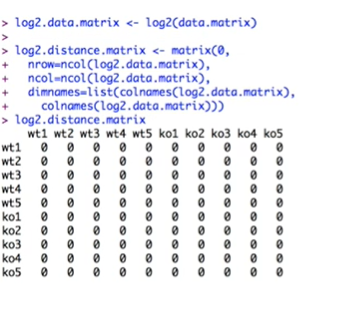
Psst : for all you gene expression folks, this is what edgeR does when you call the plotmds() function.



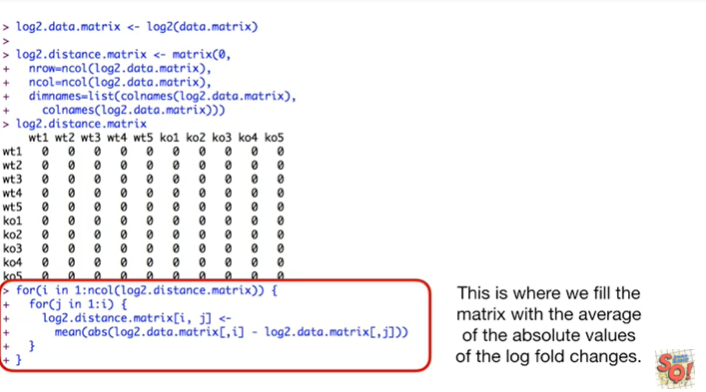
The first thing we do is calculate the log to values of the measurements for each gene.



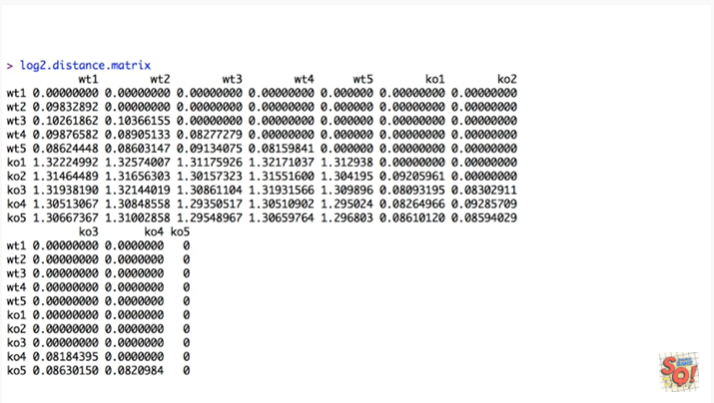
Since the average of absolute values of the log fold change isn't one of the distance metrics built into the dist function, we'll create our own distance matrix by hand.



In this step we're just creating an empty matrix.

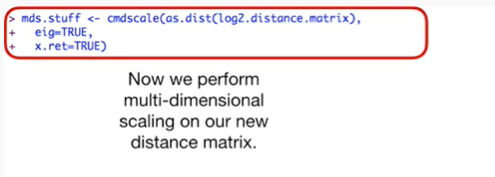


This is where we fill the matrix with the average of the absolute values of the log fold changes.

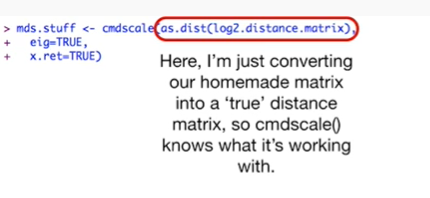


And here's what that matrix looks like.

Because the full matrix would be symmetrical, we only have to calculate the values for the lower triangle.

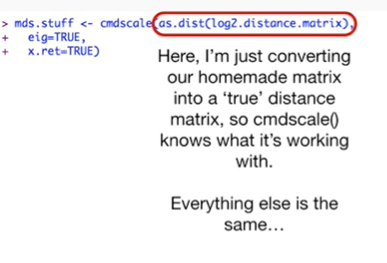


Now we perform multi-dimensional scaling on our new distance matrix.

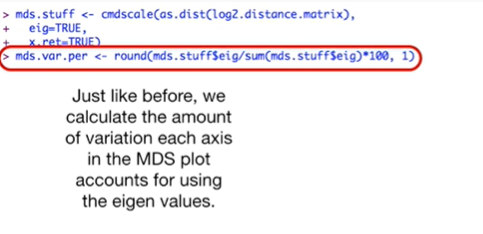


Here I'm just converting our homemade matrix into a true distance matrix so that cmdscale() knows what it's working with.

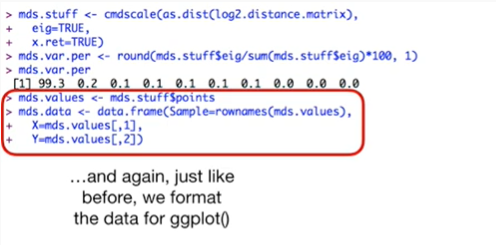
In other words a true distance matrix only needs the bottom triangle to be computed and not the whole thing.



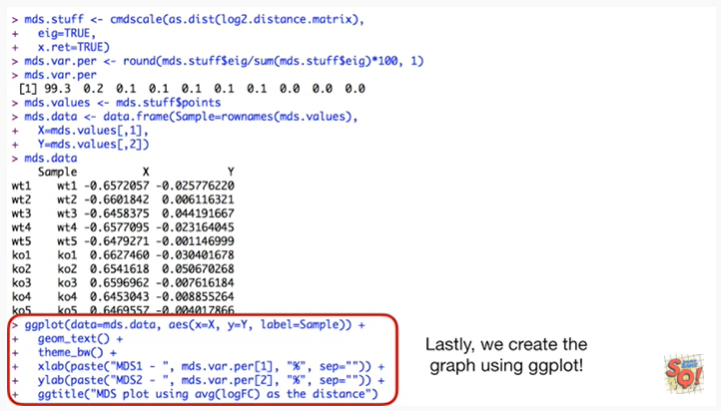
Everything else is the same.



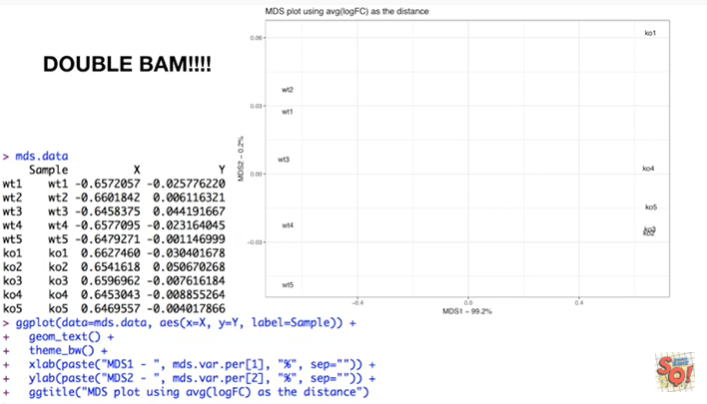
Just like before we calculate the amount of variation each axis in the MDS plot accounts for sing the eigenvalues.



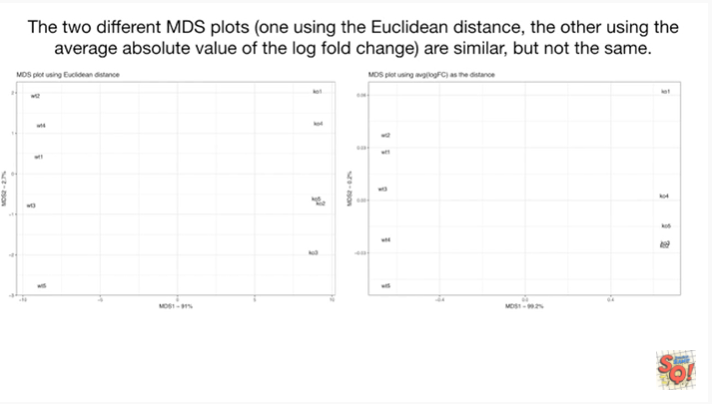
And again just like before we format the data for ggplot().



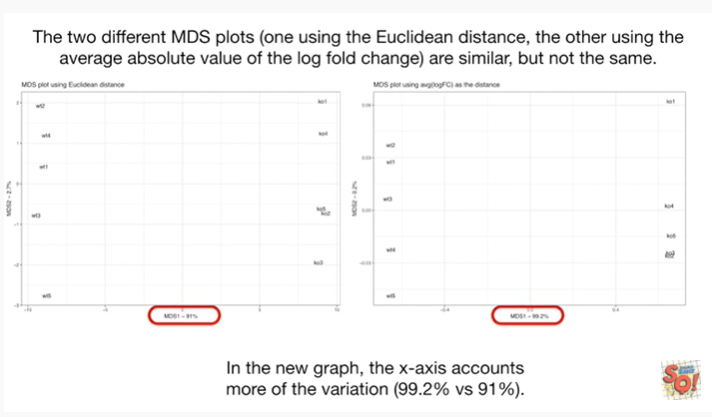
Lastly, we create the graph using ggplot().



Double BAM !!!



The two different MDS plots one using the Euclidean distance and the other using the average of the absolute value of the log fold change are similar, but not the same.



In the new graph the x-axis accounts for more of the variation 99.2% versus 91%.