Next Generation Sequencing Technologies: Data Analysis and Applications

Hands-on 2 Preliminary Data Analysis

Dr. Riddhiman Dhar, **Department of Biotechnology**

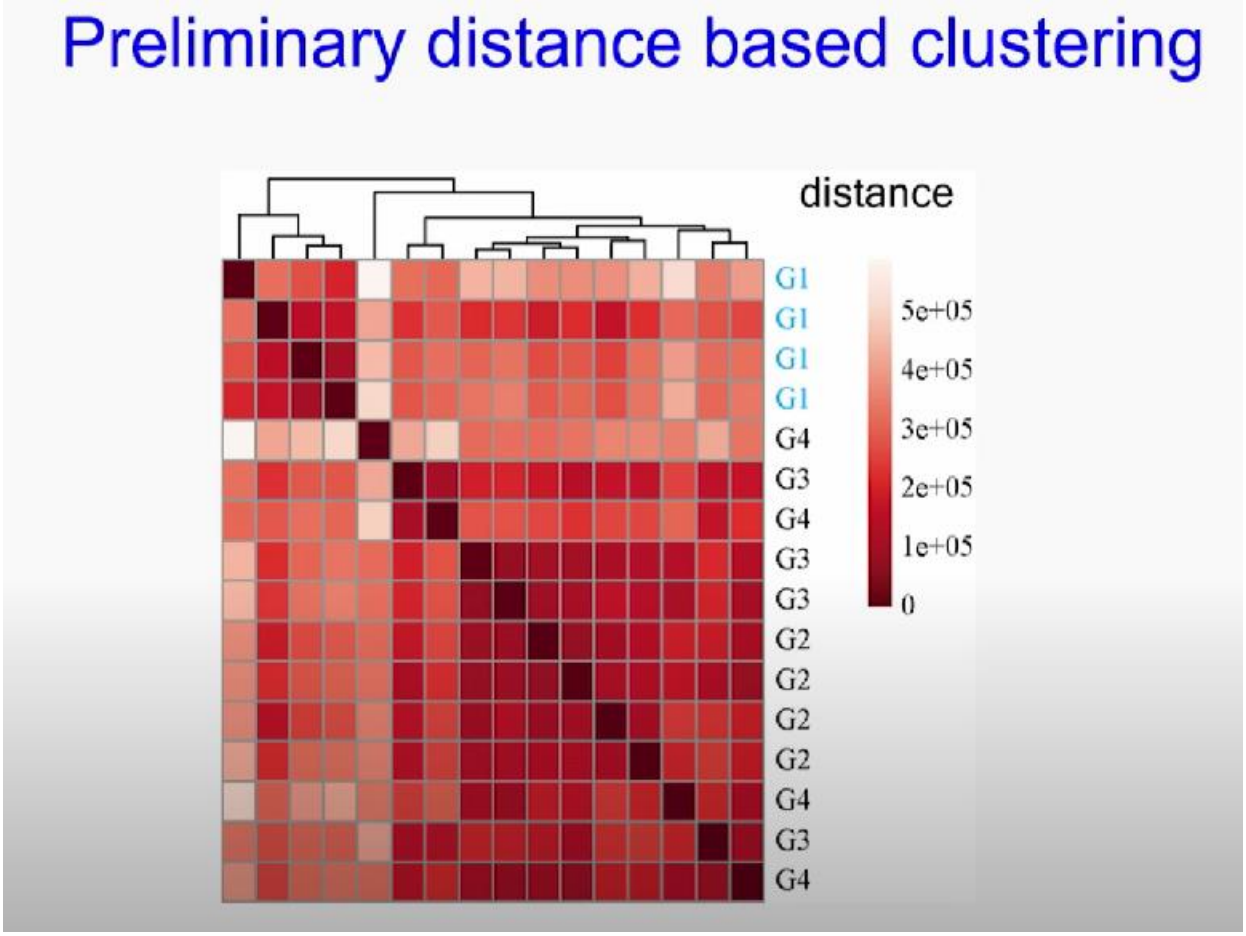
Indian Institute of Technology, Kharagpur

Good day everyone. Welcome to the course on next generation sequencing technologies, data analysis and applications. In the last two classes, we have started the hands-on and we have set up the system in R. And very briefly, we discussed about the common commands and we have looked into some of the plotting tools, how we can plot data and generate very nice plots of the data. We have also seen very basic statistical tests that we can do with R. So, there is a lot of things to explore which we cannot possibly cover in very few classes, right.

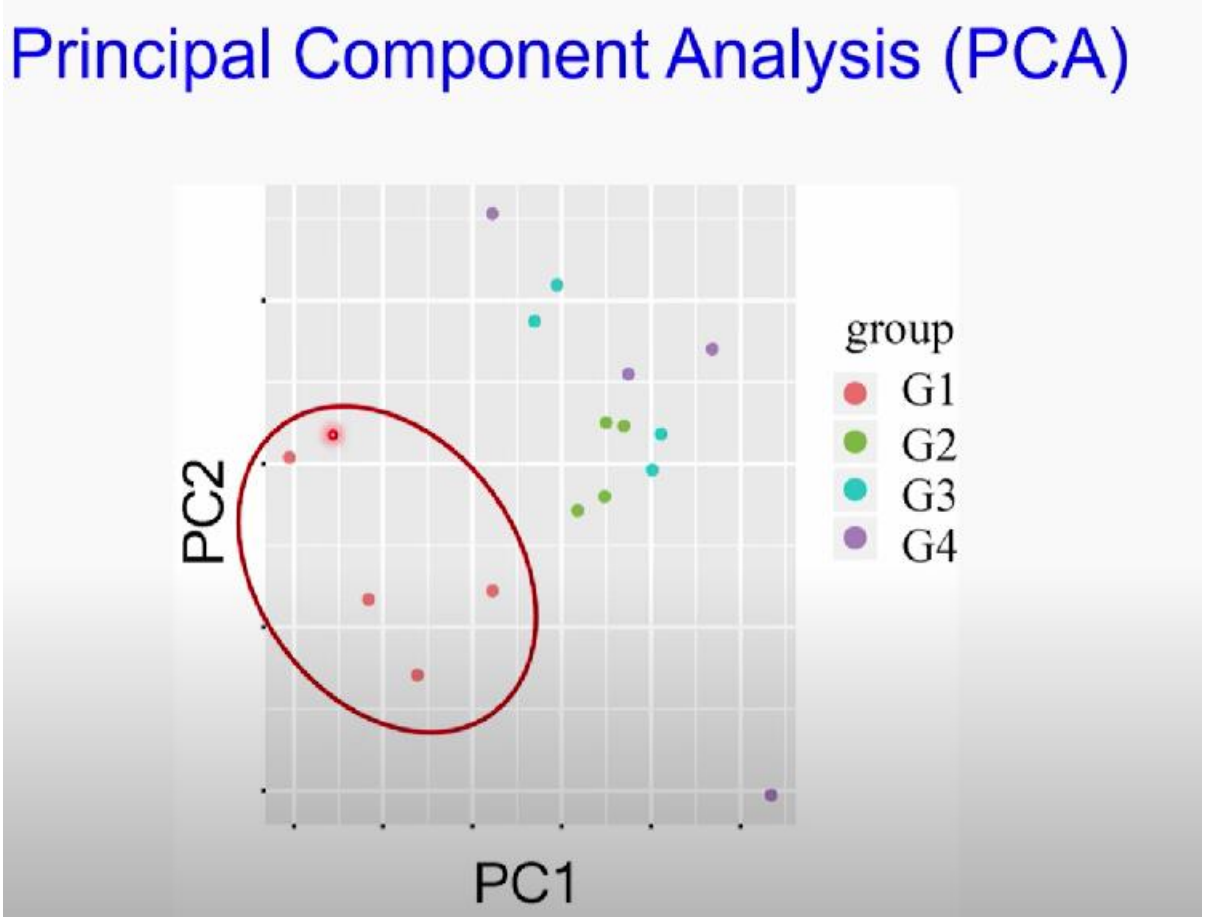
So, I will urge you to explore yourself, right. And you will see like this is a very powerful programming environment where you can generate a very interesting plots, very complex ones. Also, you can organize the data or organize the plots in certain ways that you the ways you like and you can perform all sorts of statistical test. So, coming back to our analysis, the agenda for today's class is that we will go into the preliminary data analysis part, ok.

So, this is what we are going to discuss today. So, again reminding you all the steps, we have set up the system in R. We have looked at the commands very briefly. We have loaded the count data, right, with 16 samples and about 6800 genes. And we are now going to go into the preliminary data analysis, ok.

Just to remind you, we talked about two types of preliminary data analysis when we were talking about this in theory, right. So, we have this distance-based clustering we generated. Again if you remember, right, we have these 16 samples that are present in four groups, ok. So, the groups are given here G1, G2, G3 and G4, ok. And the distance are shown by this color codes here, right.



So, if the distance is zero, you will see this very dark red color and lighter color or white color means they are far apart in terms of their gene expression pattern, ok. So, this kind of helps us visualize the whole thing very nicely, ok. The second type of analysis that we did or that we mentioned in the theoretical theory part, right, is that principal component analysis. So, then this is a very briefly this is a dimensionary reduction method, right. So, which actually generating these principal components from this 6800 genes or it can be even more depending on your data and then it kind of calculates the variance, right.



So, that are explained by each of these principal components, ok. So, here we have these two principal components we can see and we saw like we can isolate these four group one samples, right. They are separate from the rest of the groups, ok. And we also notice that one of the group four sample, right, is actually stays far apart from all the data that we have, ok. So, maybe this there are some issue with preparing the library or during the sequencing with this sample.

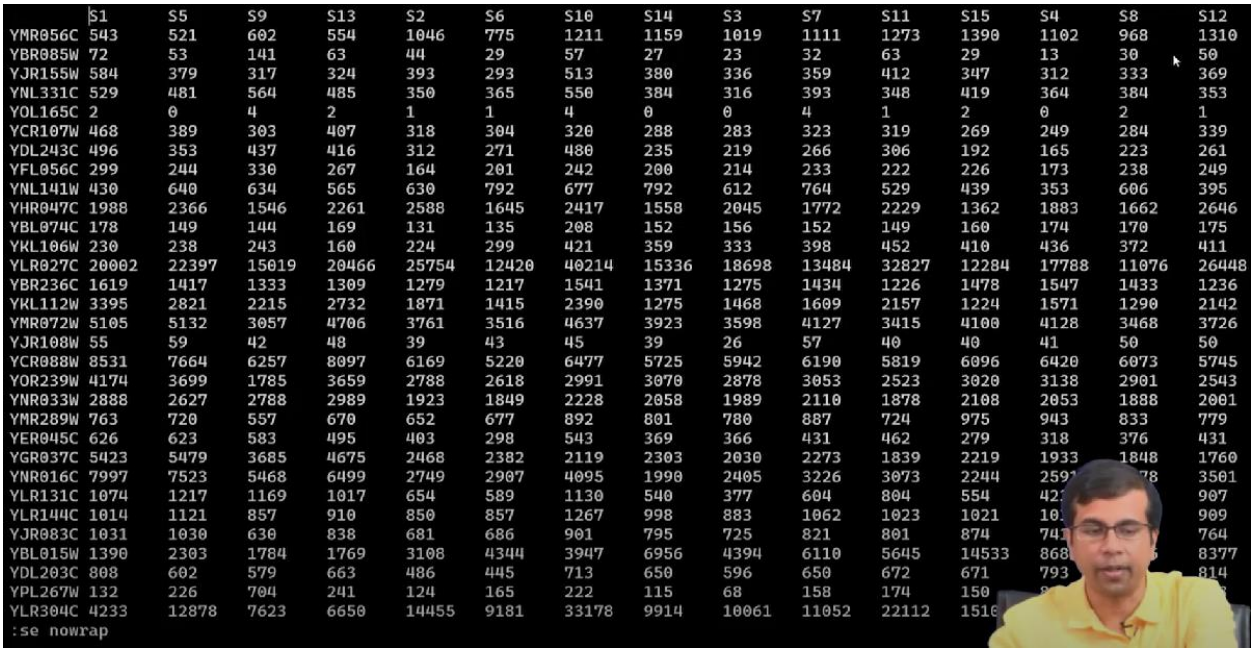
So, what we want to do now is given our data we want to generate this kind of plots, ok. We want to do this distance based clustering, right and then also we want to do this principal component analysis with R and generate this kind of plots that will help us visualize the data and then will help us, right, if you want to maybe remove this replicate of G 4 from our analysis because this is an outlier there are some issue with this sample, ok. So, let us now move on to R and then write the codes to actually do this kind of analysis, ok, alright. So, we move on to R here, right. So, we have the data to do this we need a lot of packages, ok and there is a pipeline that will have to follow, ok.

So, this is something again what I will do is I have collected all the codes I have developed the scripts that I will show you how this works. I will also mention the packages that we are going to use. The question you might have is that how do you find this out, ok. So, this is something that is actually well known if you are working with these packages for a long time you know which package can do what, right and then of course, you can learn, right. You can go through these packages when new packages are coming out or they are published, right.

You will see in certain publications they are discussed, right. You will actually learn about these packages and then you can utilize those packages for your analysis, ok. So, what I have done is I have compiled a code, right and I will go through that code one line by one line, right. So, that we actually understand what we are doing, right. I will also show you these packages very briefly and what they can do, ok.

So, I will clear this plot. So, to clear this plot is called dev dot off, right. So, it will remove or it will kind of close this window from here, right because this is not relevant anymore. We will generate new plots that will appear here, ok. So, you can see this actually is gone, right and you can clear the window and we can start a fresh, ok.

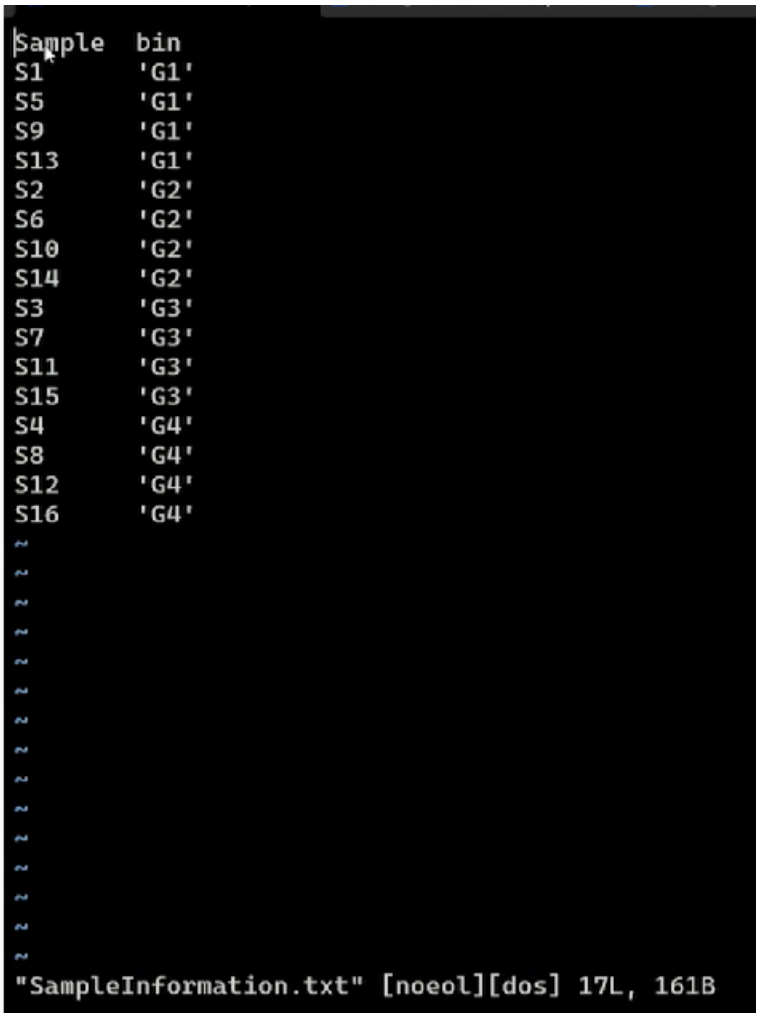
So now, we move on to the count data, right. So, that is the data we will work with before we actually do the DESeq2 etcetera, right. So, the count data is inside this test directory, right and we are here, ok and inside this we have this run three all S 1, S 16 analysis, ok. Just to very briefly just open it again and you can see we have these samples S 1 to S 16, ok. S 16 you cannot see here, it is at the end, right.



	S1	S5	S9	S13	S2	S6	S10	S14	S3	S7	S11	S15	S4	S8	S12
YMR056C	543	521	602	554	1046	775	1211	1159	1019	1111	1273	1390	1102	968	1310
YBR085W	72	53	141	63	44	29	57	27	23	32	63	29	13	30	50
YJR155W	584	379	317	324	393	293	513	380	336	359	412	347	312	333	369
YNL331C	529	481	564	485	350	365	550	384	316	393	348	419	364	384	353
YOL165C	2	0	4	2	1	1	4	0	0	4	1	2	0	2	1
YCR107W	468	389	303	407	318	304	320	288	283	323	319	269	249	284	339
YDL243C	496	353	437	416	312	271	480	235	219	266	306	192	165	223	261
YFL056C	299	244	330	267	164	201	242	200	214	233	222	226	173	238	249
YNL141W	430	640	634	565	630	792	677	792	612	764	529	439	353	606	395
YHR047C	1988	2366	1546	2261	2588	1645	2417	1558	2045	1772	2229	1362	1883	1662	2646
YBL074C	178	149	144	169	131	135	208	152	156	152	149	160	174	170	175
YKL106W	230	238	243	160	224	299	421	359	333	398	452	410	436	372	411
YLR027C	20002	22397	15019	20466	25754	12420	40214	15336	18698	13484	32827	12284	17788	11076	26448
YBR236C	1619	1417	1333	1309	1279	1217	1541	1371	1275	1434	1226	1478	1547	1433	1236
YKL112W	3395	2821	2215	2732	1871	1415	2390	1275	1468	1609	2157	1224	1571	1290	2142
YMR072W	5105	5132	3057	4706	3761	3516	4637	3923	3598	4127	3415	4100	4128	3468	3726
YJR108W	55	59	42	48	39	43	45	39	26	57	40	40	41	50	50
YCR088W	8531	7664	6257	8097	6169	5220	6477	5725	5942	6190	5819	6096	6420	6073	5745
YOR239W	4174	3699	1785	3659	2788	2618	2991	3070	2878	3053	2523	3020	3138	2901	2543
YNR033W	2888	2627	2788	2989	1923	1849	2228	2058	1989	2110	1878	2108	2053	1888	2001
YMR289W	763	720	557	670	652	677	892	801	780	887	724	975	943	833	779
YER045C	626	623	583	495	403	298	543	369	366	431	462	279	318	376	431
YGR037C	5423	5479	3685	4675	2468	2382	2119	2303	2030	2273	1839	2219	1933	1848	1760
YNR016C	7997	7523	5468	6499	2749	2907	4095	1990	2405	3226	3073	2244	259	78	3501
YLR131C	1074	1217	1169	1017	654	589	1130	540	377	604	804	554	42		907
YLR144C	1014	1121	857	910	850	857	1267	998	883	1062	1023	1021	10		909
YJR083C	1031	1030	630	838	681	686	901	795	725	821	801	874	74		764
YBL015W	1390	2303	1784	1769	3108	4344	3947	6956	4394	6110	5645	14533	868		8377
YDL203C	808	602	579	663	486	445	713	650	596	650	672	671	793		814
YPL267W	132	226	704	241	124	165	222	115	68	158	174	150			
YLR304C	4233	12878	7623	6650	14455	9181	33178	9914	10061	11052	22112	151			

:se nowrap

So, we have 16 samples and we have this 6800 genes. Now, the question is where do these samples belong, right? What are these replicates? Where what do they mean etcetera, right. So, the experimental design would also have to be mentioned before we actually go to any differential expression analysis, right. So, the program DESeq2 would need to know, ok, which sample belongs to which group or which condition, right and which are the technical replicates of one another, ok. So, is S 5 a technical replica of S 1, is S 9 a technical replica of S 1 or S 13 is a technical replica of S 2, right that information would have to be given to the program, right.



So, to do that we also have another file which is called this sample information file here. We can see this sample information file, ok and we will load this sample information file, ok and we will see this very easily now, ok. So, in the sample information file you have two values, right two column names these are headers again, ok. So, the first one is sample, ok. This is the sample name we have mentioned in the count data file, ok.

So, we have S1 to S16 again they are arranged in certain order because of their groupings which are given in this column B, ok. So, in the B column we have these groupings given. So, the S 1, S 5, S 9, S 13 the first four samples they belong to group 1, ok. So, this is mentioned here. The next four S 2, S 6, S 10, S 14 they belong to group 2.

Then we have S 3, S 7, S 11, S 15 they belong to group 3 and the last four S 4, S 8, S 12 and S 16 they belong to group 4, ok. So, this bin actually gives the information about the experimental

design, ok and we will see we will use this information when you go for differential expression analysis, right. So, this is important because now the program will know, ok these are the replicates technical replicates of each other and these are the different groups that we want to compare against each other, ok. This information is very important, ok. So, the bin part is very important.

Of course, you can have multiple columns, you can have other components of this experimental design as we have discussed, right when you are discussing the theory of this differential expression analysis. You can have for example, age of the population of the sample, right and you can have other parameters, right and you can combine those in the linear model, ok. But for now let us focus on this, right where we have this very simple design four groups, ok and each group has four replicate measurements, ok. So, this information would also have to load in the R program, right in R console we need also this need this sample information, ok. So, for this preliminary analysis what I have done is I have developed this R code as you can see preliminary analysis R code dot R.

So, inside that we have this code which will run and of course, I will explain the commands that we are using, ok. So, to see this I will just simply run this. Of course, we can run this R code as a script which I want to because I want to explain all the steps that we are going through, ok. So, do not pay attention to everything that is there I will go one by one, ok. So, the first library here that we need is the DESeq2, ok.

So, we have probably loaded this, but I will again load this here library DESeq2. We can put it inside double code or we can simply say library DESeq2 it is ok, ok. We do not have to, ok. So, it is loaded now because it was already there.

So, it is there now, ok. So, we have to load this data and sample files, right. So, this sample information files in R console, ok and we will do that one after the another, right. Data equals to read table, ok and header equals to true, right because we have the header which is the sample name, right and the data is loaded now. So, we can simply check so that the data looks fine after loading you can always check this, right when you are loading any data etcetera. So, you can

see this here that this is fine.

```
> library(DESeq2)
> data=read.table("RUN3_all_S1-S16_analysis.txt",header=T)
> head(data)
         S1  S5  S9 S13   S2  S6  S10  S14   S3   S7  S11  S15   S4  S8
YMR056C 543 521 602 554 1046 775 1211 1159 1019 1111 1273 1390 1102 968
YBR085W  72  53 141  63   44  29   57   27   23   32   63   29   13  30
YJR155W 584 379 317 324  393 293  513  380  336  359  412  347  312 333
YNL331C 529 481 564 485  350 365  550  384  316  393  348  419  364 384
YOL165C   2   0   4   2    1   1    4    0    0    4    1    2    0   2
YCR107W 468 389 303 407  318 304  320  288  283  323  319  269  249 284
         S12 S16
YMR056C 1310 480
YBR085W   50  30
YJR155W  369 171
YNL331C  353 304
YOL165C    1   0
YCR107W  339 255
> samp=read.table("SampleInformation.txt",header=T)
> head(samp)
  Sample bin
1     S1  G1
2     S5  G1
3     S9  G1
4    S13  G1
5     S2  G2
6     S6  G2
>
```

We also will load this sample information file that I just showed you, ok and it is loaded by this sample data equals to true, ok. So, again we have the header. Remember, right we have the sample and the bin as the header, right and you can again check head sample and we have this column sample and bin. We can again explore right, right sample and bin. So, it is telling me the information that is present in this column, right and we can also try maybe summary and see if this gives us any information, right.

```
> samp$bin
 [1] "G1" "G1" "G1" "G1" "G2" "G2" "G2" "G2" "G3" "G3" "G3" "G3" "G4"
[14] "G4" "G4" "G4"
> summary(samp$bin)
   Length     Class      Mode
       16 character character
> levels(samp$bin)
NULL
>
```

It just gives you some statistics that is really not important. Let us see if we can get the label information from here. Now, it is not, ok. So, this requires a specific data format that will give you this label. Anyway, so we have loaded this sample information now.

The next step is to convert this as data frames, right. So, keep this as data frames because DESeq2

actually requires them as data frames, ok. So, to convert that we simply write as data frame, right. So, in R this is actually possible to convert one type of data into another, ok and this is what we are doing here.

You can see this in the command. So, I am just writing data matrix, right and this is this will be utilized later on, ok. So, data matrix is as data frame, sorry, there is a mistake here, right. So, as data frame, ok and simply just say data and similarly we can write, we can convert this sample to also sampmatrix, ok. So, these are now converted to data frame because they will be required, ok for this analysis, ok. So, what we are going to do now is we are going to convert this data into a something called a DESeqDataSet, ok.

```
> datamatrix=as.data.frame(data)
> sampmatrix=as.data.frame(samp)
```

This is very important. This is the first step for doing this differential expression analysis using DESeq2, ok. So, we can now check this DESeq2 package, ok and this is what you will see, ok. This is again the bioconductor page for DESeq2. We have the manual here, ok, and just opening the manual, we will get all this information. So, before you actually go about doing this differential expression analysis, you have to create something called a DESeqDataSet, ok.

Now DESeqDataSet, it combines this count data along with the sample information, right. So, this information is important for all these comparisons, right. So, that because it contains this technical replicating information, the condition information, right the design of the experiments, ok. So, that combination we have to create, ok. So, that part we have to do through this DESeqDataSet class, ok.

So, this is where we have to create this data set from our count data that we have loaded as well as the sample information. So, this is something we will see in different R packages that they require certain data format before they can process the data, ok. And we have to first convert this count and all this information into this specific data format and here it is, right. So, you can see this is how we can convert this and the command is DESeqDataSet from matrix, right. So, the matrix that we have created, we can now convert these to this DESeqDataSet, ok.

So, that is what we do. This is the command and as you probably notice that there are multiple

commands that you can run from this DESeqDataSet DESeq2 package and these commands, right, these are given here, ok. You can see these commands that are given here. There are lot of things that are here. We will not discuss everything right now, ok.

But you can also just click on this link. So, these are also links, right, with the page numbers. So, you can simply click on these links and you will go to this page, ok, where all this information will be there. The description of the function, the usage, right, how it will be used, what are the parameters it will take, what are the arguments it will take, right. So, and the description of the arguments, right. So, this will help us understand the kind of data that we need to feed to this program or this command, ok.

And then of course, there are lot of details and sometimes also some examples, ok. So, here are some examples, right, how you can actually use this command here, ok. So, for now, it is enough to know that we need to create this DESeqDataSet from the file, ok. So, from the matrix we will create this data set and this command is this, ok.

So, again following the manual, ok. So, we have to create this DESeqDataset. We store this in this variable dds DESeqDataSet from matrix that is the command inside this DESeq2 package. We first need the count data, right. This is present in the data matrix, right. Then we have the column data, right, that is in the sample matrix, right.

```
dds <- DESeqDataSetFromMatrix(countData = datamatrix, colData=sampmatrix, design=~bin)
dds
```

This gives us the design of the experiment and then finally, it tells us, ok, the design, right, how this experiment has to be interpreted, ok. So, design is the bin column, right. This bin column gives the information about different conditions or the different types of samples that we have, ok, as well as the technical replicates, ok. So, this is the command that will run, ok.

So, let us run in R now, ok, dds. So, let us clear this so that you can see clearly. So, even though we clear the window, this information is present, right. It is not just gone, ok.

Everything is there. It is the package is loaded now, ok. So, dds is the DESeqDataSet from, again you have to be careful about this capital letters and the small letters, right. They are all mixed up

here, again. So, that is why you have to carefully look at the commands, ok. The first thing we need is the counts data, right, that is present in data matrix.

Then we, I think we need the column data, right. So, again we have to carefully look into how we should specify, right, sample matrix and design equals to, I think, right, bin, ok. Again, I have to be careful whether the bin is in capital letter or in small letters, ok. Again, it will look up in the file, right. So, the sample information file you can very quickly check.

```
> dds <- DESeqDataSetFromMatrix(countData = datamatrix, colData=sampmatrix, design = ~bin)
Warning message:
In DESeqDataSet(se, design = design, ignoreRank) :
  some variables in design formula are characters, converting to factors
>
```

```
> dds
class: DESeqDataSet
dim: 6805 16
metadata(1): version
assays(1): counts
rownames(6805): YMR056C YBR085W ... YGR285C YNL241C
rowData names(0):
colnames(16): S1 S5 ... S12 S16
colData names(2): Sample bin
>
```

The bin is in small letter, right. So, we can simply change that now, ok. So, we have now loaded this, right. So, we have, we are specifying the count data. This is experimental count data that we got.

We have the column data. This is containing information about the experimental design and the final design equals to bin, right. This is telling, interpret the design by using this bin column, ok, that contains information about the replicates and the conditions, ok. So, it is giving me some error, right.

So, let us see, ok. What we have, ok, let us see. Oh yes, so I think I have made a mistake. So, one of the commands is not count, it is actually count data, ok. So, this part again you have to be very careful and this is again we have to pay attention to this package manuals, right. So, where you have this DESeq2 from data set, right.

So, let us again have a look, right. This is something that will be mentioned here, ok. You can

see this is counts, right. So, this is cnts. So, if you have to mention this as count data, ok.

So, that is I will change that and then it should be fine. It is not just count. So, in some package it will say it will count. So, again you might get confused and now it is fine, ok. It is giving me a warning message, but that is ok, right.

So, it is saying like some design formula characters. So, it is converting them to factors. So, it is ok. So, the warning is fine. It is able to read the data. I can check, ok, whether it has converted this matrix data into desec data set format and it has, right.

So, you can see this if you just type dds, the class is DESeqDataSet, right. So, this is the data set format that has been created. So, dim, this is the dimension. So, 6805 rows and 16 columns, ok. Then you have the SS that is the count data it has, right, only the row count data.

The row names are the gene names. So, this is mentioned, right. Of course, it will not give me the all the gene names, but you can see these are the gene names. So, row names are the gene names. Then you have the column names. So, we have 16 columns from S1 to S16, ok. And then the column data names too because these are the part of the design of the experiment, ok.

So, this is fine now. So, we have converted this data into this DESeqDataSet. The next step, ok, is to convert or to transform this dds data through this rlog transformation, ok. So, again without going into lot of details, again this is a transformation that will convert that count data, ok, into certain format, right. So, it will take a log and etcetera, it will transform. So, this rlog transformation, ok, will then use for doing this preliminary analysis, ok.

```
rld <- rlog(dds, blind=FALSE)
head(assay(rld),3)
lims <- c(-2, 20)
```

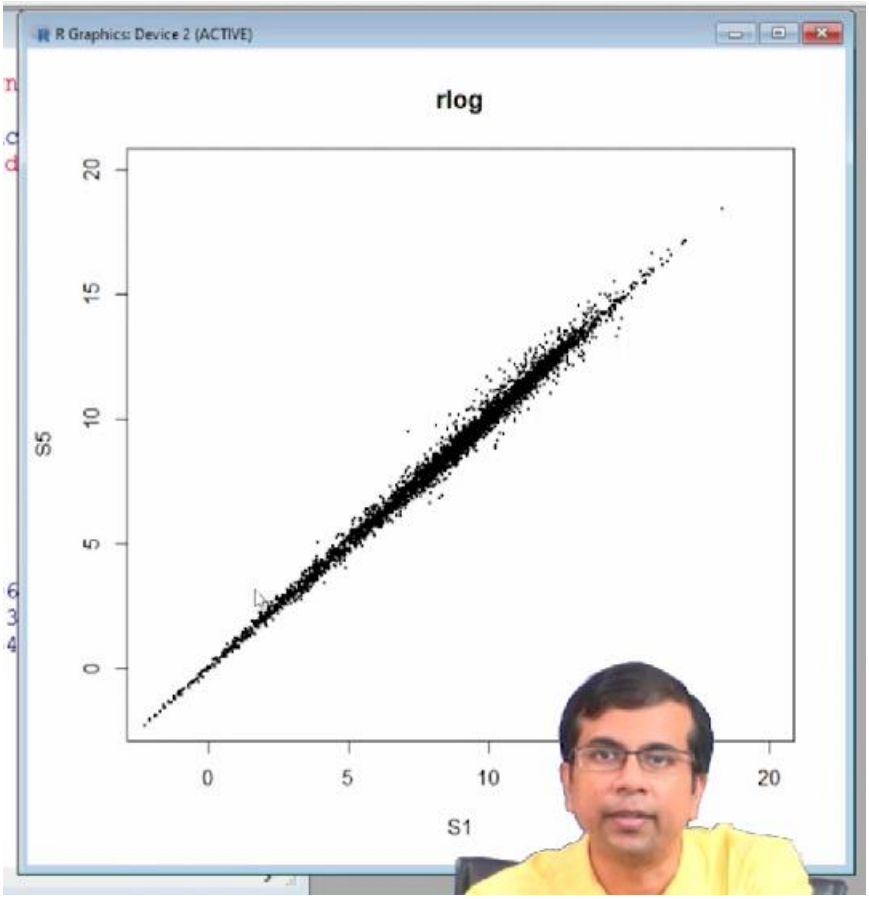
```
plot(assay(rld)[,1:2], pch=16, cex=0.3, main="rlog", xlim=lims, ylim=lims)
```

So, let us run this. So, rld rlog dds line equals to false, ok. So, again you can check where this what this rlog means, what are the commands that we have, but for this preliminary analysis we need this rlog commands. And this comes from default package of R, ok. So, this is this has run

now. You can see this rlog is done. So, this transformation, you can again check this head gives is the, sorry, so head again using this head command, right, you can check this the data, ok, the rld data or the rlog transform data.

```
> rld <- rlog(dds, blind=FALSE)
> head(assay(rld),3)
              S1       S5       S9      S13        S2       S6      S10       S14       S3
YMR056C 9.224381 9.206181 9.509720 9.343487 10.016504 9.762703 9.951441 10.065673 9.975685
YBR085W 5.717044 5.499268 6.481869 5.696690  5.478786 5.232954 5.514174  5.132827 5.068319
YJR155W 8.834755 8.437765 8.445762 8.375966  8.623323 8.385049 8.676296  8.540196 8.460235
              S7       S11       S15       S4       S8       S12       S16
YMR056C 9.969752 10.143643 10.235001 9.992753 9.907222 10.192751 9.304800
YBR085W 5.198446  5.696075  5.165196 4.783283 5.206655  5.535539 5.247097
YJR155W 8.435521  8.601929  8.441224 8.334360 8.434969  8.515824 7.901277
> lims <- c(-2, 20)
> plot(assay(rld)[,1:2], pch=16, cex=0.3, main="rlog", xlim=lims, ylim=lims)
```

You can see the counts have changed, right, because it has been they are smaller because there is a log transformation, right, that has been done on the data, ok. So, this is the data and now we want to do this limits, ok. And why we are setting this limit because we want to check the correlation between this different technical replicates, ok. And you probably have seen this already this correlation and here it simply says plot assay, right.



Some of these part are probably familiar to you, right. You can see assay rld. So, this is actually looking at the count data only, ok. rld has you can you can check what are the components of rld by just writing rld. Then it is saying 1, 2, again now we have talked about this is matrix, right. So, we are accessing the first two columns of this matrix 1 and 2. So, starting from 1 to 2 pch we have talked about, this is the point type, then cex this is the point size, we are making it smaller because less than 1, main is the main title, we are changing Xlim and Ylim, we are setting the limits from minus 2 to 20.

```
> rld
class: DESeqTransform
dim: 6805 16
metadata(1): version
assays(1): ''
rownames(6805): YMR056C YBR085W ... YGR285C YNL241C
rowData names(7): baseMean baseVar ... dispFit rlogIntercept
colnames(16): S1 S5 ... S12 S16
colData names(3): Sample bin sizeFactor
>
```

We have used Xlim command, right, before. So, this should be familiar to you, ok. Once we plot this is what we see, ok. So, what rlog does it not only takes a log transformation of the data, it also does something called the variance kind of stabilization kind of analysis, ok. So, what is this variance stabilization analysis is that it is kind of minimizes this higher variability of this data of the genes that have low read count or low expression value, ok. So, it is very similar to variance stabilization analysis, we will not go into that detail, but you can see now this variability among these reads or genes that show low read count is gone, ok.

So, it looks very nice now, there is a very good correlation in the data, ok. And similarly now we can do a plot and we can add this Y equals to X line, we can do a lot of things, ok, but we will not go into that, we have done this already. So, just checking rld we can simply type rld, it is a transformation that has been done. Again the dimensions are given, right, the row names are given and then you have the column names that are given. And what has been done also is this size factor calculation. So, that kind of normalizes this data and this part of this high variability part is gone, ok.

So, without going into a lot of details, right, we can now see this very good correlation between technical replicates, ok. So, this by the way correlation between S 1 and S 5, right. So, we are looking at the first two columns of this rld data. So, this is between S 1 and S 5.

Now, what we will do is now we want to generate this distance based correlation, right. So, this is what we will do in the next part. So, you can see this code, this we need these two packages, we will see if we have these packages installed, if not we will do them and then we need to calculate these distances, right. So, distances between these transcriptomes or the expression patterns using Euclidean distance, ok, and then we will plot this using a heat map function, ok. So, we will use this library pheatmap to actually generate this distance based correlation and we will get also get a hierarchical clustering in that, ok. So, let us see if we have these libraries installed, library pheatmap and library R color brew.

```
library("pheatmap")
library("RColorBrewer")

sampleDists <- dist(t(assay(rld)))
sampleDists
```

So, pheatmap is the library that we need for generating this heat map and the color brewr is a just giving us combination of colors, ok, for defining this color palettes etcetera, ok. So, let us see if we have these libraries installed or not, right, in there. pheatmap, let us see if it is installed, if not, ok, so it is saying that is ok and also R color, ok. So, these are now loaded here, ok. What I will do is I will clear the window, I will also close this window, right, and now we can generate these new plots, ok.

So, let us look at the code again, right. So, what we will do first in this part is we will generate this distance matrix, ok. So, between these different genes expression patterns we first want to calculate this distance, ok. For doing that we will use this command. So, we will store this distance in this sample distance variable, right.

So, that is why this assignment to this variable. We are calculating distance, so using this command dist, ok. So, this will calculate distance and t is the transpose of this matrix assay rld, ok. So, again if you look into how this distance will work, distance function works, we need to take the transpose. So, if you are working with this raw count data, right, so or normalized count data, we use this Euclidean distance which is the distance matrix, ok.

```
> sampleDists <- dist(t(assay(rld)))
> sampleDists
          S1       S5       S9      S13       S2       S6      S10      S14       S3       S7
S5  21.09609
S9  31.11634 28.69661
S13 15.59883 17.65152 25.69207
S2  33.00852 22.89155 35.86548 29.09306
S6  38.60127 26.74370 36.04236 34.42231 16.68499
S10 35.52095 26.21136 32.44483 31.32798 17.07643 20.10539
S14 41.39019 32.41549 39.37125 37.66091 19.62648 15.86100 22.95719
S3  38.96351 29.68307 38.86080 35.34571 15.11971 14.20891 20.63399 12.09997
S7  41.59373 30.76628 37.79259 37.42451 20.71488 11.87149 22.35658 13.53267 13.40377
S11 38.93646 29.86310 35.19547 35.16026 19.13543 19.85171 14.71740 18.36408 17.53283 19.21834
S15 49.57889 40.85475 43.13164 45.56200 32.14852 25.04848 31.65101 18.99106 23.08026 19.93156
S4  47.56328 38.33849 43.14156 43.55345 28.24581 22.16237 29.11176 17.51409 17.78410 16.32219
S8  46.51830 37.08729 41.52695 42.89015 28.15897 19.59997 28.37966 17.17910 18.54706 13.73279
S12 43.66254 34.60238 38.77623 39.51371 24.95428 23.53151 22.30085 20.12736 20.22283 20.35358
S16 44.19509 38.19090 46.51420 41.47117 34.59210 33.10913 35.38431 37.86134 35.72754 34.50004
         S11      S15       S4       S8      S12
S5
S9
S13
S2
S6
S10
S14
S3
S7
```

```
S2  33.00852 22.89155 35.86548 29.09306
S6  38.60127 26.74370 36.04236 34.42231 16.68499
S10 35.52095 26.21136 32.44483 31.32798 17.07643 20.10539
S14 41.39019 32.41549 39.37125 37.66091 19.62648 15.86100 22.95719
S3  38.96351 29.68307 38.86080 35.34571 15.11971 14.20891 20.63399 12.09997
S7  41.59373 30.76628 37.79259 37.42451 20.71488 11.87149 22.35658 13.53267 13.40377
S11 38.93646 29.86310 35.19547 35.16026 19.13543 19.85171 14.71740 18.36408 17.53283 19.21834
S15 49.57889 40.85475 43.13164 45.56200 32.14852 25.04848 31.65101 18.99106 23.08026 19.93156
S4  47.56328 38.33849 43.14156 43.55345 28.24581 22.16237 29.11176 17.51409 17.78410 16.32219
S8  46.51830 37.08729 41.52695 42.89015 28.15897 19.59997 28.37966 17.17910 18.54706 13.73279
S12 43.66254 34.60238 38.77623 39.51371 24.95428 23.53151 22.30085 20.12736 20.22283 20.35358
S16 44.19509 38.19090 46.51420 41.47117 34.59210 33.10913 35.38431 37.86134 35.72754 34.50004
         S11      S15       S4       S8      S12
S5
S9
S13
S2
S6
S10
S14
S3
S7
S11
S15 25.16353
S4  22.53238 13.68443
S8  22.54705 15.09594 12.05658
S12 13.03074 21.69473 19.07175 20.34930
S16 37.50585 46.00518 42.13643 39.37065 41.26165
>
```

So, let us now run this in R, sorry not here, ok and see if we got this here. So, the distance will be stored in this sample distance variable, ok. So, you can simply write and you can see this pairwise distance data, ok. So, this distance between S 1 and S 5, distance between S 1 and S 9,

right and you can you can see this matrix, only the lower part is filled, ok. The upper part is not filled and the diagonal is of course 0, because distance between S 1 and S 1 is 0, ok. So, once we have generated this distance matrix, now we can use the heat map to plot the data, ok and we can actually use some certain color combinations, right.

So, you can see we use this color combinations to actually plot the heat map, ok. So, you have, we will get this row names and column names, right. So, again this will be used later on, right. So, we can get this row names, column names, ok and sample this matrix is not found, ok. We will see if we can. So, first we actually calculate, so we have calculated the distance, but we need to convert this to a matrix, right.

```
sampleDistMatrix <- as.matrix( sampleDists )
rownames(sampleDistMatrix) <- paste( rld$bin, rld$Sample, sep="-" )
colnames(sampleDistMatrix) <- NULL
colors <- colorRampPalette( rev(brewer.pal(9, "Blues")) )(255)
pheatmap(sampleDistMatrix, clustering_distance_rows=sampleDists, clustering_distance_cols=sampleDists, c
ol=colors)
```

First we need to run this command. We have not run this, we have not created this sample distance matrix. Once we create that, then we can set the row names and column names of this sample distance matrix, ok. So, we will use this sample distance matrix for the heat map generation, right, as a matrix form. So, pheatmap will take the data in a matrix form and will generate this distance based clustering plot, ok.

```
> sampleDistMatrix <- as.matrix( sampleDists )
> rownames(sampleDistMatrix) <- paste( rld$bin, rld$Sample, sep="-" )
> colnames(sampleDistMatrix) <- NULL
>
```

So, that is why we are getting this error here. So, first we generate the sample distance matrix here. Again, we are converting this data into matrix form. So, as I said that in R you can do this, right. You can interconvert between different data types and the this part, the row name sample distance matrix is we are setting the row names of sample distance matrix from this rld dollar bin, right, and rld dollar sample and with this separator, ok. So, we will see when we generate this correlation plot or the distance plot, we will get this bin and the sample name as the labels, ok.

We will see this in a moment. And the column names of the sample distance matrix are null. So, we just setting as null, ok. We do not want any specific name there, ok. So, now we can set the colors, right. We can again this is for some very nice colors.

```
> colors <- colorRampPalette( rev(brewer.pal(9, "Blues")) )(255)
> |
```

If you want to change the colors, we are using this colorramppalette from this RcolorBrewer package, ok. And we are using let us say the blue colors, right. So, we are using these blues, ok. You can check this. We are not explaining everything, right, how this works, but this you can check that this is from the RcolorBrewer package and this is simply for setting the color that we will use in the heat map, ok.

So, now we can actually use this pheatmap command, ok. So, I can go back and I can show you, right, rpheatmap, ok. So, where so rheatmap in cran, right. So, where you will have this manual, right.

So, cran package pheatmap and you will have this reference manual here pheatmap.pdf. And inside you will have this command. So, this is this pheatmap command and it the arguments that it can take, ok. So, again you have this description of the arguments.

Usage

```
pheatmap(mat, color = colorRampPalette(rev(brewer.pal(n = 7, name =
  "RdYlBu")))(100), kmeans_k = NA, breaks = NA, border_color = "grey60",
  cellwidth = NA, cellheight = NA, scale = "none", cluster_rows = TRUE,
  cluster_cols = TRUE, clustering_distance_rows = "euclidean",
  clustering_distance_cols = "euclidean", clustering_method = "complete",
  clustering_callback = identity2, cutree_rows = NA, cutree_cols = NA,
  treeheight_row = ifelse((class(cluster_rows) == "hclust") || cluster_rows,
  50, 0), treeheight_col = ifelse((class(cluster_cols) == "hclust") ||
  cluster_cols, 50, 0), legend = TRUE, legend_breaks = NA,
  legend_labels = NA, annotation_row = NA, annotation_col = NA,
  annotation = NA, annotation_colors = NA, annotation_legend = TRUE,
  annotation_names_row = TRUE, annotation_names_col = TRUE,
  drop_levels = TRUE, show_rownames = T, show_colnames = T, main = NA,
  fontsize = 10, fontsize_row = fontsize, fontsize_col = fontsize,
  angle_col = c("270", "0", "45", "90", "315"), display_numbers = F,
  number_format = "%.2f", number_color = "grey30", fontsize_number = 0.8
  * fontsize, gaps_row = NULL, gaps_col = NULL, labels_row = NULL,
  labels_col = NULL, filename = NA, width = NA, height = NA,
  silent = FALSE, na_col = "#DDDDDD", ...)
```

You need a numeric matrix, of course. You need also a color argument, right, that you will give. And then other different types of arguments, again, you can give them default or you can use them, ok. So, what I have done is I have given this matrix data, right. This is the matrix data that we have to supply. And then we are saying like how you should cluster the samples and which colors it should use, ok.

```
pheatmap(sampleDistMatrix, clustering_distance_rows=sampleDists, clustering_distance_cols=sampleDists, c
ol=colors)
```

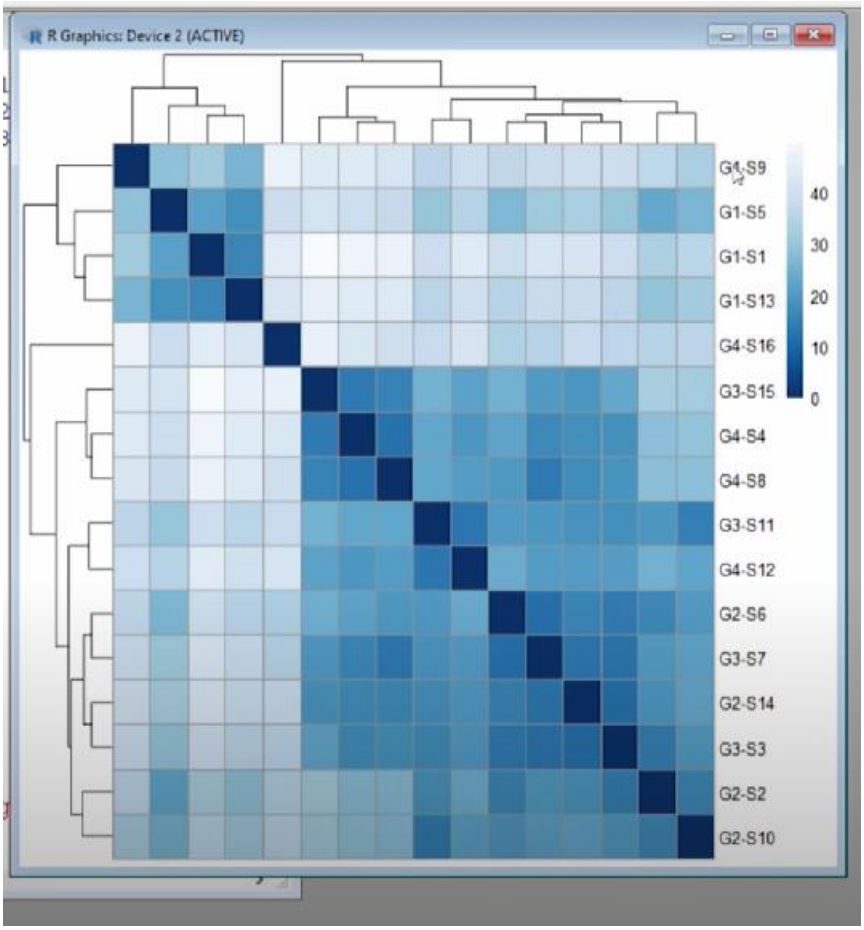
So, the color equals to colors argument, right, this actually you say it is the color from this rcolor here, right. For other variables it will just simply take the default value, ok. So, we can now use

this command, ok. So, we have set color equals to colors. We have given the sample distance matrix, right. And we have said like use this distance rows and also the distance columns from the sample distance list, right, and then generate the plot.

And you can see this very nice plot now, ok, very similar to what we have seen earlier. And what simply tells us, ok, these G1 samples, right, and these are the replicate or the sample names, right. So, S9, S5, S1 and S13, right. So, these are all part of this group 1, ok, and that they are present here. And there is a very nice correlation between them, ok.

So, these group 1 samples there are very nicely correlated with each other. Similarly, for this group 3, group 4 and group 2 samples you can see very nice correlation again among each other. But then you also notice that there is one sample here G4 S16 that seems like an outlier, ok. So, this is what we wanted, right, what we wanted to generate in the first place, ok. So, this is how we can do this distance based clustering, right, which can tell us, ok, which samples are close to each other and which sample might be an outlier in our data, ok. So, in the next class we will talk about the principal component analysis, right, and we will generate this principal component analysis plot. And once we are done, then we can move into the bias correction part. Thank you very much.

You need a numeric matrix, of course. You also need a color argument, right? And then other different types of arguments, again, you can give them default or you can use them, ok? So, what I have done is given this matrix data, right? This is the matrix data that we have to supply. And then we are saying, like, how you should cluster the samples and which colors it should use, ok? So, the color equals the color argument, right? This is actually the color from this rcolor here, right? For other variables it will simply take the default value, ok? So, we can now use this command, ok? So, we have set color equals to colors. We have given the sample distance matrix, right? And we have said to use these distance rows and also the distance columns from the sample distance list, right, and then generate the plot.



And you can see this very nice plot now, which is, ok, very similar to what we have seen earlier. And what simply tells us, ok, these G1 samples, right, and these are the replicate or the sample names, right? So, S9, S5, S1, and S13, right? So, these are all part of Group 1, ok, and they are present here. And there is a very nice correlation between them, ok? So, these group 1 samples are very nicely correlated with each other. Similarly, for this group 3, group 4, and group 2 samples, you can see a very nice correlation again among each other. But then you also notice that there is one sample here, G4 S16, that seems like an outlier, ok? So, this is what we wanted—right, what we wanted to generate in the first place, ok? So, this is how we can do this distance-based clustering, which can tell us, ok, which samples are close to each other and which sample might be an outlier in our data. So, in the next class, we will talk about the principal component analysis, and we will generate this principal component analysis plot. And once we are done, we can move on to the bias correction part. Thank you very much.