Next Generation Sequencing Technologies: Data Analysis and Applications

Differential Gene Expression 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. We have been working on an RNA-Seq dataset, and we have gone through several steps. So, we have started with the count data, and then we have done some preliminary analysis through distance-based clustering and principal component analysis. We have seen how the samples are similar or whether there is an outlier. So, we have found one outlier in our sample.

And then we did the bias correction using the package EDASeq. We corrected for GC content, but of course you can try different ways for this bias correction or within sample normalization. So, now we have the bias-corrected data, and we can move on to the differential expression analysis—the actual differential expression analysis, right? So, in this class, we will be talking about this expression analysis using this tool, DESeq2. So, this is the agenda for this class. We will talk about the first part; we will cover the first part of this differentiation expression analysis. There will be a second part where we will cover this differential expression analysis when you have the spike in the samples, ok? So, the normalization we can do differently, and that we will cover in the next class, ok. So, again, coming back to the steps we have completed up to this bias correction part, we can start with the differential gene expression analysis, and for that, we will use this tool, DESeq2, that I have already introduced, and I have also shown you, right, how we can install this package, ok.

So, we will now use this package to actually run this analysis, and then, of course, we can generate some interesting results that we can interpret and visualize. So, let us now move on to the R console, where we can actually load the data and sample information and proceed with the differential gene expression analysis.

```
rdhar@LAPTOP-3K4C9VBI:/mnt/c/Users/Dhar/Desktop/NGS_Data_Analysis_HandsOn2/Test$ ls
Analyze_RNAseq_data.R                    RUN3_all_S1-S16_analysis.txt  steps
BiasCorrected_RUN3_all_S1-S16_analysis.txt  Rcode_bias_correction.R
Prelim_analysis_Rcode.R                  SampleInformation.txt
rdhar@LAPTOP-3K4C9VBI:/mnt/c/Users/Dhar/Desktop/NGS_Data_Analysis_HandsOn2/Test$
```

So, we will go back to this folder where we have all these files, and as you remember, we have now created this bias-corrected data file, which we will utilize for doing this differential gene expression analysis, ok. So, once we have done this, once we have loaded this, we also need the sample information, which is again stored, ok. So, the sample information has not changed, right? Whether you do this bias correction or not, the sample information has not changed.

```
> library(DESeq2)
```

So, it will remain the same, but we will have to load it and run through DESeq2. So, again, going through the manual of DESeq2, I have collected the commands that we require, right, for doing our analysis, and I have compiled an R script, right, which I will utilize for doing all this analysis. Again, this is very simple: just look at the manual, format this according to their instructions, load the appropriate data, and run the analysis, ok? So, just going back to the Bioconductor DESeq2, here is the pdf file with all the details of the manual, right? And the first thing we need again is this DESeq dataset class, right? So, we need to create this DESeq dataset from the data to the count data, ok, and then we can run this differential expression analysis.

```
> setwd("../Desktop/NGS_Data_Analysis_HandsOn2/Test/")
> data=read.table("BiasCorrected_RUN3_all_S1-S16_analysis.txt",header=T)
> samp=read.table("SampleInformation.txt",header=T)
> head(data)
         S1  S5  S9 S13   S2  S6  S10  S14   S3   S7  S11  S15   S4  S8  S12 S16
YMR056C 695 654 667 672 1017 775 1286 1121 1002 1079 1168 1327 1051 950 1211 631
YBR085W  35  26  74  31   22  16   31   15   15   19   32   17   11  19   26  17
YJR155W 437 245 216 215  270 198  362  254  230  246  274  232  210 218  245 124
YNL331C 380 355 403 349  244 246  389  258  215  268  241  272  245 253  233 224
YOL165C   4   0   5   4    3   3    8    0    0    8    3    4    0   3    3   0
YCR107W 372 322 240 312  263 245  272  235  219  250  264  214  192 209  273 213
> head(samp)
  Sample bin
1     S1  G1
2     S5  G1
3     S9  G1
4    S13  G1
5     S2  G2
6     S6  G2
>
```

Again, you will see there are a lot of functions that are also present in this program that we can also use for plotting our data, and maybe we will use some of them, for example, the plotMA we can use for doing this analysis. So, let us use this. So, let us load the library and also the data, ok? So, the first step is library, which is DESeq2. It might take a bit of time, right? Because it also

needs to load other packages that are required for its run.

Now, once it is loaded, we can now load the data, and remember, right, we need to load the bias-corrected data, right? So, again, we need to change the directory to the directory where we are working. Again, it will depend on your system and how you set this up, right? So, here in my system, I know this is where we have all the data, ok? So, I will now load the data from here, and we have the bias-corrected file that we created in the last class, ok, and the header equals true, ok.

So, we also need to load the sample information file, right? So, this will be the same, and also the header equals true because you have this design information, right? The bin is there. So, we can simply check whether this data and sample information are loaded correctly, and we can then move on to the analysis now, ok? So, for the differential expression analysis, again, as I mentioned, we have created a set of code that I will utilize for doing this in a very short period of time. So, we have loaded the library, we have loaded the data and the sample, we can change the data matrix and sample matrix, right, as data frames from this data and sample, right, and then we can run these steps one after another, ok? So, first step: ok, I will convert this again to a data matrix; this is going to be a data frame. This is required because sometimes some of these tools or the commands expect this data in a certain format, ok. Again, this might also change from one version to another depending on the R version that you are using. You might have to add a few things or write it slightly differently, ok. We have seen these differences between versions sometimes, and if you see any errors, we might have to go back to the manual and see if there are any changes or not.

```
> 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: 6696 16
metadata(1): version
assays(1): counts
rownames(6696): YMR056C YBR085W ... YGR285C YNL241C
rowData names(0):
colnames(16): S1 S5 ... S12 S16
colData names(2): Sample bin
>
```

So, now we will convert these two sample matrices, right? So, the frame and sample are okay. Now, the next step is to create this DESeq dataset, right? So, we have done this before, right, and you remember, right, we use this dds variable to actually assign this new DESeq dataset, right, and the command is DESeq dataset from matrix, we have count data, we have column data, right, and the design. This is a command that we used before when doing the preliminary analysis, and we will use the same here, ok? So, we will copy this command, and I will simply run it here again. It gives some warning messages, but that is ok.

We have seen this warning message; as long as it is not giving any errors, it is alright. We can again write the dds, ok, and it then gives us the summaries of this dataset. So, this is a DESeq dataset. Now, the dimensions are different because these are the bias-corrected datasets. Remember, we took the data for only those genes where there is overlap with this GC content data? So, the GC content data and this data had an overlap of about 6700 genes, ok? Now, once we have done this, right, what we will do is run that DESeq data DESeq analysis. So, the differential expression analysis and the command that we have to use is very simple: this DESeq dds, ok, and we can run that now in the console, right, ok, and ok. What I will do is not run this; I will first clean it off so that you can see the steps. So, once we do this DESeq command, it will go through certain steps that we have discussed, ok? We will see this, right, one after another.

It will take a bit of time. Again, we are running the full thing, ok? So, here it is. You can see that the first step is estimating size factors. We have talked about size factors, and DESeq2 utilizes this median of ratios method for normalization.

So, this estimating size factor is just estimating these normalization factors, and then it is calculating these estimating dispersions, ok? So, again you remember this model, right? So, we have to calculate the gene-wise dispersions, and then this is what we are doing: gene-wise dispersion estimates. It is calculating this mean dispersion relationship. It is calculating the final dispersion estimate.

So, this is the shrinkage of dispersion that we again discussed in the theory class, and finally, it is

fitting the model and testing, right? It is doing this statistical test from there, ok? So, we can actually take some of these steps. So, idea again, we can look at this PRDDS dollar, ok, and say sample. So, these are the samples we have utilized for this analysis, OK, and the bin.

```
> prdds <- DESeq(dds)
estimating size factors
estimating dispersions
gene-wise dispersion estimates
mean-dispersion relationship
final dispersion estimates
fitting model and testing
> prdds$Sample
 [1] "S1"  "S5"  "S9"  "S13" "S2"  "S6"  "S10" "S14" "S3"  "S7"  "S11" "S15" "S4"  "S8"  "S12" "S16"
> prdds$bin
 [1] G1 G1 G1 G1 G2 G2 G2 G2 G3 G3 G3 G3 G4 G4 G4 G4
Levels: G1 G2 G3 G4
> prdds$sizeFactor
       S1        S5        S9       S13        S2        S6       S10       S14        S3        S7       S11
1.1044893 1.0974487 0.8958489 0.9992632 0.9484810 0.9111134 1.1501710 0.9948139 0.9566362 1.0566660 1.0128092
      S15        S4        S8       S12       S16
0.9957956 1.0105482 0.9764742 0.9854391 0.9338109
> prdds
class: DESeqDataSet
dim: 6696 16
metadata(1): version
assays(4): counts mu H cooks
rownames(6696): YMR056C YBR085W ... YGR285C YNL241C
rowData names(30): baseMean baseVar ... deviance maxCooks
colnames(16): S1 S5 ... S12 S16
colData names(3): Sample bin sizeFactor
> |
```

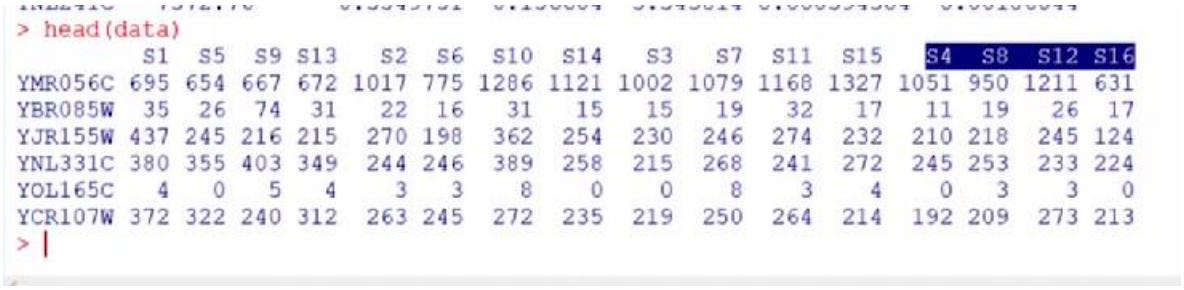
This is the experimental design. It can give you the levels, right? So, it found all the bins, and then it knew the levels were G 1, G 2, G 3, and G 4. So, these are the conditions or the groups that we have created, ok, and it should look at comparison between these groups, ok. And similarly, we can also look at the size factor, and it will give us the values of each of the size factors that are calculated. So, for this sample, the size factor is 1, and so on. So, you remember the size factor, right? So, we have to divide, and rather, this will divide the read data by these size factors, which it does before fitting the model. So, that is it. So, we have the results now. We can actually look at the results, ok? So, again, it says, class DESeqDataSet, the dimensions are given, the gene names, etcetera. Now what you have to do is generate the results, right?

```
> prdds
class: DESeqDataSet
dim: 6696 16
metadata(1): version
assays(4): counts mu H cooks
rownames(6696): YMR056C YBR085W ... YGR285C YNL241C
rowData names(30): baseMean baseVar ... deviance maxCooks
colnames(16): S1 S5 ... S12 S16
colData names(3): Sample bin sizeFactor
> res <- results(prdds)
> res
log2 fold change (MLE): bin G4 vs G1
Wald test p-value: bin G4 vs G1
DataFrame with 6696 rows and 6 columns
         baseMean log2FoldChange     lfcSE      stat      pvalue        padj
        <numeric>      <numeric> <numeric> <numeric>   <numeric>   <numeric>
YMR056C 955.12061       0.568978  0.171921  3.309523 9.34550e-04 0.003882112
YBR085W  25.56582      -1.165487  0.435999 -2.673141 7.51447e-03 0.022812651
YJR155W 246.02707      -0.404657  0.227173 -1.781271 7.48683e-02 0.149262255
YNL331C 284.92367      -0.582315  0.135052 -4.311793 1.61936e-05 0.000122874
YOL165C   2.95522      -1.095091  1.281757 -0.854367 3.92902e-01 0.536182387
...           ...            ...       ...       ...         ...         ...
YLR130C   1692.19     -0.0346775  0.225729 -0.153624 0.877905846  0.92184298
YKL175W   1653.65     -0.3126392  0.108956 -2.869413 0.004112338  0.01365751
YBR046C   4658.16      0.3201982  0.220141  1.454515 0.145803481  0.25478985
YGR285C   2828.64     -0.1280478  0.325748 -0.393089 0.694253627  0.79177260
YNL241C   7372.76     -0.5549751  0.156604 -3.543814 0.000394384  0.00186044
> |
```

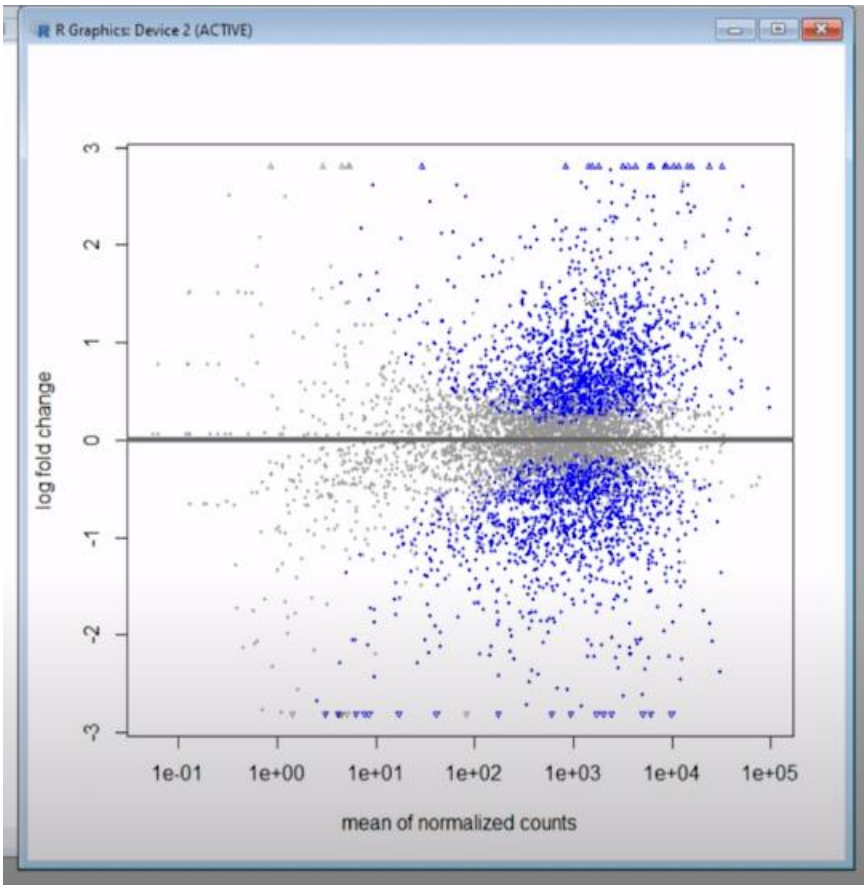
So, we have to kind of do this, right? So, summarize the results from here, and we generate the results here. So, in this variable race, we can simply type race, and it shows you the results of these differential gene expression analyses. So, the first line you can see is the log 2 fold change, and it specifies, ok, we are comparing bin G 4 versus G 1, ok. And the p, while the test p value again bins G 4 versus G 1, the data frame has 6696 rows.

So, 6696 genes and 6 columns, ok? So, these six columns are here. The first column is the base mean; this is the average expression. The second one is the log2 fold change. We have talked about the LFC. Then you have the standard error in LFC, then you have the statistics, then you have the p value and the adjusted p value through the Benjamin Hochberg correction, right? So, this is the one we can then use for FDR correction. We can choose, let's say, a cutoff of 10 percent, and we can identify genes that show this significant p-adjustment. So, this is, these columns are given for all these 6700 genes, approximately. So, this is the result that we have gotten. Now, you might ask why G 4 versus G 1, right? So, why not the other ones? So, by default, because if you look at the original data, ok. So, I will show you the data first, and then I will explain. So, if you look at the original data and if you remember the sample information pile, these S1, S5, S9, and S13 are part

of group 1, ok? Then you have group 2, then you have group 3, and then you have group 4. So, by default, what DESeq2 will do is compare this G 4 versus G 1, write the last sample versus the first sample, and give the results. So, let us imagine you want this G 4 versus G 3, ok? So, what you have to do is maybe just create a file with only G 3, G 4, or reorganize this data so that G 3, G 4 are in the first and last columns, ok?



So, by default, when you have this kind of data, you will get these G 4 versus G 1 results, ok? But then this is what we want in our analysis, right? We simply want this because G 1 was the most interesting group, if you remember from the preliminary analysis. So, this is good. We have generated this data. Now that we can interpret the data, we can look at some of the statistics and see whether the normalization, for example, has been good. So, one of the first things to check for normalization is the MA plot, right? So, we have discussed this MA plot before, and we take these results and generate this MA plot to see whether the data looks good or not. So, let us do that. Let us generate this MA plot now from this results file, ok, or the race variable. So, I will clear this window so that you can see clearly, and I will minimize because we will generate a plot now here, ok? So, plotMA, yes, ok. We can simply write this, and then maybe later on we can specify the limits, ok? So, what you see here is something we have seen before, right? On the x axis, we have the mean of normalized counts, right in the data. So, this is the average expression across all the samples, and then you have the log-fold change.



This is a comparison of G 4 versus G 1, ok? So, again, we have some of these gray points and some of the blue points, right? So, gray points are the points for which we do not see significant results, and the blue points are the genes for which we have a significant difference in expression between G 4 and G 1, right? So, they are simply color-coded based on their significance, ok? And what you also notice is that genes that are low in expression or show a low read count—right, there is a lot of variability, ok? And again, some of these genes might show a very high log-fold change, but these are not significant. Again, because there is a very high level of variability between the technical replicates, we cannot find these results significant.

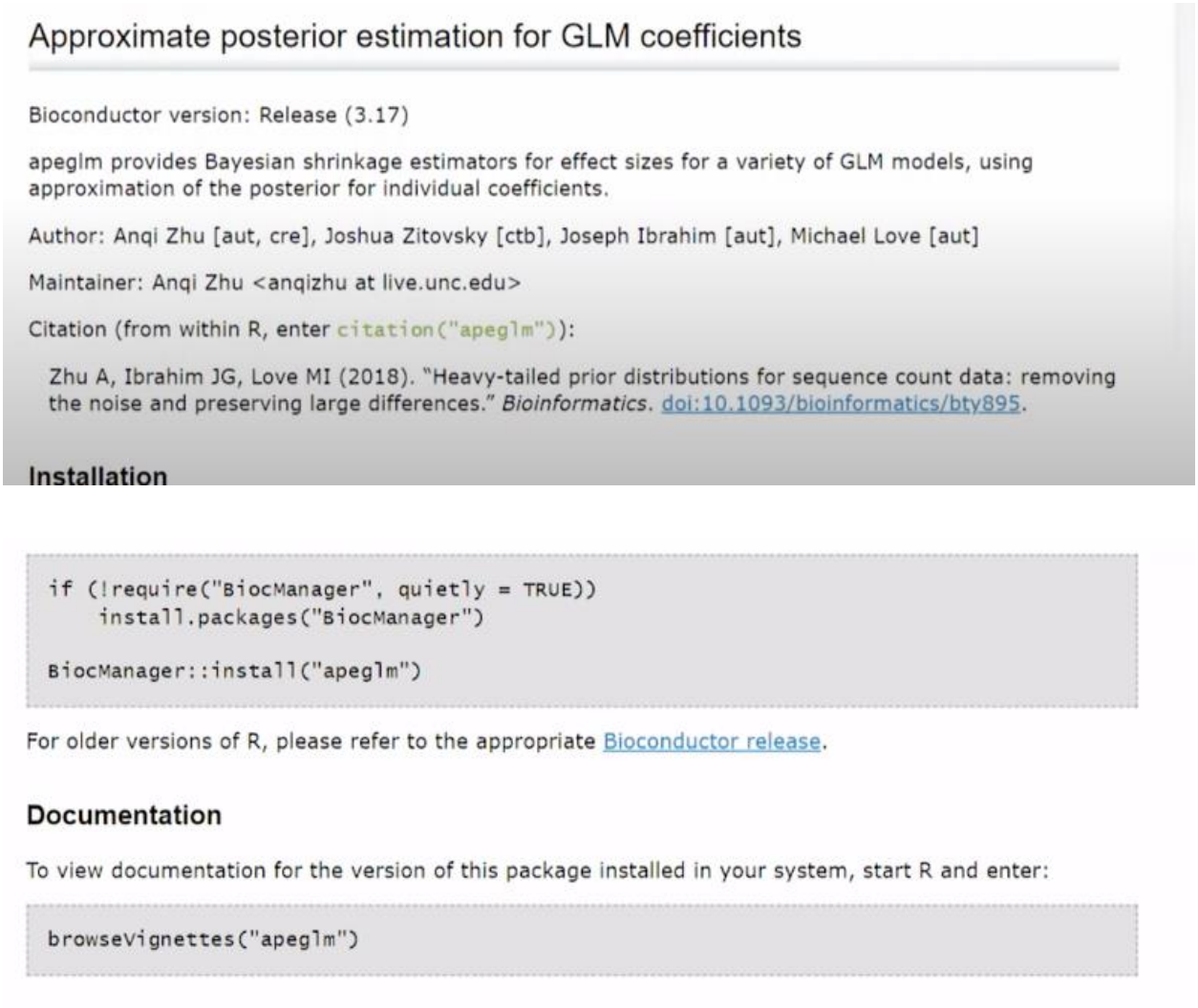
You will not find these results significant, ok? And this analysis is the significance analysis, right? That actually depends on two things. One is the effect size, or how much different this expression is between these two conditions or two types of samples; that is the first thing. And how reproducible or how low the variability is between the technical replicates between these two different conditions, ok? So, for these samples here with very low expression, the lock fold change

is very high, right? So, the effect size is high, but the variance is also very high, right? So, there is a lot of variance between the technical replicates, ok? And that is why these are not significant, ok? So, is there any way to reduce this variance? And we have talked about one method, which is the shrinkage of LFC. And there is a function in DESeq2, right, that actually we will do that, ok? So, let us go to the manual and we will see whether we can find this function LFC shrink, ok, that we can use for this shrinkage, ok. So, here is this function in DESeq2, LFC shrink that we can use. We can go to that page and we can see where we can use this LFC shrink, ok. So, here it is: LFC shrink, shrink lock to fold change, right? So, this is the theory we have discussed: why do you want to estimate this, and why do you want to shrink these estimates for LFC? And you can do this with two different methods, and here is the usage of this command: LFC shrink.

We have to use this data set that you have created, and etcetera, you have contrast. You can use three different types of shrinkage methods, ok? So, again, these are explained later on, right? You can see this, so we can also use the LFC threshold, which will come later on. So, these three methods are explained here, right? ere is this type. So, you have apeglm; this is a student shrinkage. Again, we are not going into the statistics, but these are three different ways you can do this shrinkage. We can use the first one and then you have the ashr shrinkage, then you have the normal shrinkage, ok? So, this is why they are using different types of shrinkage estimators, and that is why they will give you slightly different results, ok? So, let us use this apeglm shrinkage method, ok, and we will use this library apeglm, ok.

So, we need that library to be installed first before we can use this shrinkage, ok. So, this library is apeglm, ok. Let us see if it is installed on my system. If not, again, we can install this very easily

through a bioconductor. If it is a bioconductor package, I think it is a bioconductor package.



Approximate posterior estimation for GLM coefficients

Bioconductor version: Release (3.17)

apeglm provides Bayesian shrinkage estimators for effect sizes for a variety of GLM models, using approximation of the posterior for individual coefficients.

Author: Anqi Zhu [aut, cre], Joshua Zitovsky [ctb], Joseph Ibrahim [aut], Michael Love [aut]

Maintainer: Anqi Zhu <anqizhu at live.unc.edu>

Citation (from within R, enter citation("apeglm")):

Zhu A, Ibrahim JG, Love MI (2018). "Heavy-tailed prior distributions for sequence count data: removing the noise and preserving large differences." *Bioinformatics*. doi:10.1093/bioinformatics/bty895.

Installation

```
if (!require("BiocManager", quietly = TRUE))
    install.packages("BiocManager")

BiocManager::install("apeglm")
```

For older versions of R, please refer to the appropriate Bioconductor release.

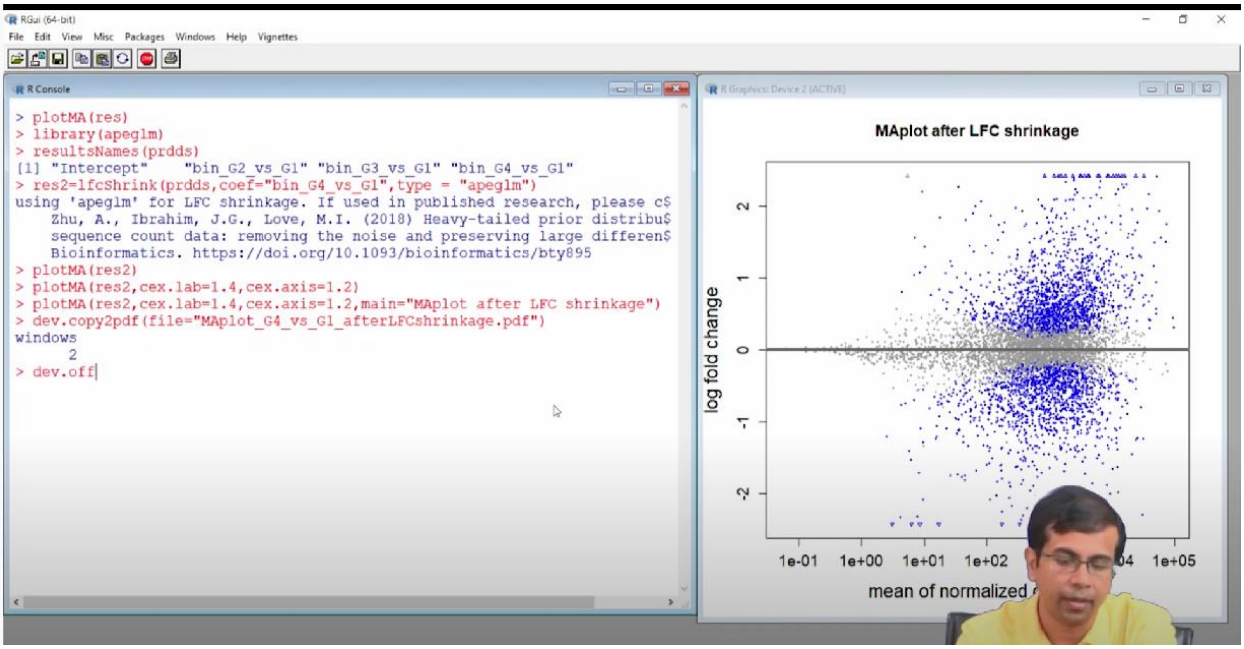
Documentation

To view documentation for the version of this package installed in your system, start R and enter:

```
browseVignettes("apeglm")
```

So, we can simply search here and see whether this appears. So, by-conductor apeglm package, right? So, again, you have the details; you have this reference that you can read that will describe the method; and you also have a manual in a PDF file that will tell you how to use this package, ok, and what are the functions that you can use for this purpose, ok. So, what I will do now is let us check whether we have this, and then we will run this LFC shrink function using this method, and we will generate the final results. So, let us go to R and see if we have library access, ok? So, it is installed; otherwise, it will give me an error, and then we can install it. And the first thing we need to do is check these results. Right now, we have generated the results names. So, we have intercept; we have G 2 versus G 1, G 2 versus G 1, but we need G 4 versus G 1, right? So, this is what we want to check, right? This is what we want to shrink using this LFC shrink function, ok, and that is what we do here. You can see this, ok.

So, we have to do this LFC shrink on this DESeq object, right? So, after we have drawn this DESeq analysis, we will have to use the LFC shrink on that, ok? So, this is what we are doing here. We can see this function in LFC shrink PDDS; this is after the DESeq analysis. We are using this coefficient because we are interested in this comparison, the right G 4 versus G 1 comparison, and then the type of method that we will use is the APEGLM method, ok? Instead of apeglm, you can try different methods and see whether there is much difference or not. So, we will run this command now, ok? So, we will store the results in this res2 variable, which you can look at later on, ok? So, let us run this on the R console, ok? It will take a bit of time, and we can then check the results, ok? It has run it is saying, ok if you are using this for then we can we have to cite this manuscript, ok. So, now that we have generated this res2, we can also check, right through the MA plot, whether this method has worked and whether we see a reduction in variability for these genes, and right the change in this logfold change for these genes that have low expression or a low read count. So, we will see mostly a large amount of changes here around these genes, where the normalize count is quite low, ok? So, we can check this through the MA plot. So, plotMA res res 2, right? This is the result after this LFC shrinkage, ok? And you see now that it looks much nicer, right? So, this variability has decreased because we have used this LFC shrinkage function. So, these logfold change values have decreased and are close to 0 now and for some of them, it is 0; this is actually more appropriate, because they might not be showing very high log4 change because of the variance that is present in the technical replicates, ok? So, this part is now good. Now we can do what you can do: we can actually read the results, we can export the results, and one of the things I will show you now is how we can actually save these plots.



We have not talked about this so far, right? So, we can maybe make it a bit nicer right before we actually export the data and then store it as a PDF file, for example, because we want to sort these plots for later references, or we may want to use them in some presentations in publications, etcetera. So, before we do that I will probably make this plot a bit nicer to look at, ok, and then we can export it. In a PDF file, okay. So, what I will do is we will start with this xlim ylim, which is fine, right? So, we can change the cex.lab 1.4 axis to 1.2, right? So, we are changing the sizes of the labels, right, because I think they are too small, and then what you can do is also use a main, right? So, we want the title. So, MA plot after LFC shrinkage, ok? Ok, so you see this, right? We now have a very nice title here, ok? Now, it looks much nicer than before. So, we can now save this file as a PDF file, ok? So, the command to do that is copy to PDF, oh, sorry, dev copy dev dot copy to PDF, sorry, yeah. So, this is the command: dev dot copy to PDF, right? So, it will copy this output or save this output in a PDF file, ok? So, this is the command we use for the file. Let us say we plot G4 versus G1 after LFC shrinkage, ok dot PDF, and we can simply enter, right, and then we have to use this dev dot off, ok. And we can now check this file to see if it has been saved. Okay, and we can see. You can open this PDF file, and you can see that this is now saved in a PDF file. In R, you also have options for saving this in different other formats, such as JPEG or BMP, and then you can directly insert it into the presentation. Why is it saved in PDF? Because then you can maintain the resolution, and you can also export later on to JPEG or BMP, ok? There are other ways to also save in EPS format, etcetera which are much more useful if you want to

export them to different formats later on. So, this is very useful, right? So, once we have generated this plot, we can now export it into a PDF file, ok? So, we have seen two types of operations, right? Once we have generated some useful data or some results, we can export the results to text files or write them into text files. And similarly, once we have generated some plots that we want to look at later on or maybe use for presentations, etcetera, we can also save those in PDF format.

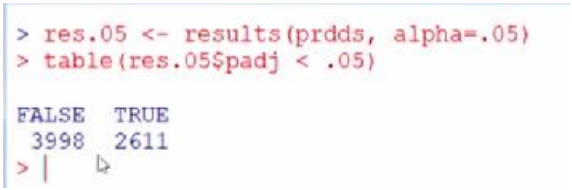
I have shown you the PDF format, but this can also be done in other formats, ok? So, there are a few other things we can do once we have generated these results, ok? We can, for example, filter according to the alpha values and the p-adjusted values. We can also filter based on logfold changes, etcetera. So, this is something we can actually do right now, ok? So, one of the things we can do first is generate summary statistics here, ok? So, by just running summary res to, ok. So, what it is saying is giving us a summary of the results that we have seen, ok? So, we have 6652 with a nonzero total read count, ok? So, we have this many genes, and the adjusted p value cutoff is 0.

```
> summary(res2)

out of 6652 with nonzero total read count
adjusted p-value < 0.1
LFC > 0 (up)       : 1626, 24%
LFC < 0 (down)     : 1378, 21%
outliers [1]       : 43, 0.65%
low counts [2]     : 0, 0%
(mean count < 0)
[1] see 'cooksCutoff' argument of ?results
[2] see 'independentFiltering' argument of ?results
```

1. So, this is the FDR cut-off. So, there is a 10 percent false discovery rate. So, that is quite acceptable. Of course, you can change it or redo the filtering. Then you have LFC greater than 0, up-regulating about 600 genes. So, which is about 24 percent of the whole data set? You have LFC less than 0, that is, down-regulated genes about 1378. So, 21 percent of the data sets. You have some outliers, which is a very small percentage, ok? And this kind of gives us, ok, there are a lot of genes that are showing this up-or-down regulation. And maybe we can now look for only the top genes, right? So, the genes that show a significant difference in lock pull change may be okay. So, LFC greater than 0 does not mean much, to be honest, because LFC could be 0.2 or 0.3. These are very small differences in terms of actual expression levels, but they are significant, right? But then, if you have 1600 genes, you would probably be mostly interested in the genes that show very

high logfold change, right? Those are probably more interesting to look at, ok? So, on top of this data, we can now use our own filtering criteria, ok? And this is what we are going to do, ok? And it is very simple, actually. So, we can create these cutoffs, right? So, for example, we can generate these tables and use these LFC thresholds on top of the data. So, here is this one. The first one is that we are generating these results where we are using this alpha cutoff of 0.05, right? So, it means we are now using the FDR cutoff. Okay, alright. So, what you can do is set our own filters, ok? And we can run this different program to adjust the cutoff. So, we can change the FDR cutoffs, and we can also change the LFC threshold cutoffs, ok? So, we will try that using the results that we have generated, but we then have to run this on the DESeq object, right? So, after this differential gene expression analysis that we generate, we can then run these filters, and then we can see all these results, ok.



So, the first thing is that we are actually running this on, ok. So, we will run this PRDDS at alpha 0.05. So, this is looking at only genes for which the p value is less than 0.05, right? We are filtering those. And then after that, we can also see how many genes are there where p adjusted is less than 0.05, right? So, we have 5 percent FDR, right? So, there are only about 2611 genes that actually satisfy this criteria, ok? So, this is something that is a significant reduction from the earlier value, right? We have seen about 24 percent of genes showing up regulation and 21 percent of genes showing down regulation; in total, there are about 2600 genes, right? So, that is, we are making this more stringent, ok? So, similarly, we can use these LFC thresholds, right, and we can see, right, whether we can generate, ok.

So, we are using this LFC threshold of 1, right? So, a log2 fold change threshold of 1 means we want only those genes that show at least a 2-fold change in expression, ok? So, this is what we are going to do, right, and then we can again have a look at this, right? Out of these genes, what fractions are showing this p adjusted less than 0.1?

```
> resLFC1 <- results(prdds, lfcThreshold=1)
> table(resLFC1$padj < 0.1)

FALSE  TRUE
 6499   110
> table(resLFC1$padj < 0.05)

FALSE  TRUE
 6516    93
> |
```

So, it is about 110, ok? So, that is a significant reduction now. We can make it more stringent, and we see that only 93 genes are showing the LFC threshold; crossing this LFC threshold of 1 means they are showing at least a 2-fold change in expression. In addition, they are showing p adjusted, which is less than 5 percent, ok? So, that is a good thing we have 93 genes, and these are probably the most interesting genes, ok? We can do further analysis, right? We can look at something else, right? For example, we can sort this list, find out the top genes, etcetera, but one thing we can simply do is export this data. Whatever we have generated, we can simply export it, and then we can play around with this data set and identify which top genes you want. For example, maybe we want only the up-regulated or down-regulated genes, and that is something we can do, and we can analyze those data sets, ok? So, again, for exporting, we simply have a look at these two. So, perhaps we can just export this right into some DE gene list, and then we can do our other type of analysis, ok?

```
> res2
log2 fold change (MAP): bin G4 vs G1
Wald test p-value: bin G4 vs G1
DataFrame with 6696 rows and 5 columns
         baseMean log2FoldChange     lfcSE      pvalue        padj
        <numeric>      <numeric> <numeric>   <numeric>   <numeric>
YMR056C 955.12061       0.520180  0.170836 9.34550e-04 0.003882112
YBR085W  25.56582      -0.865976  0.466316 7.51447e-03 0.022812651
YJR155W 246.02707      -0.330967  0.214880 7.48683e-02 0.149262255
YNL331C 284.92367      -0.552777  0.134789 1.61936e-05 0.000122874
YOL165C   2.95522      -0.109865  0.420216 3.92902e-01 0.536182387
...           ...            ...       ...         ...         ...
YLR130C   1692.19     -0.0277865  0.199137 0.877905846  0.92184298
YKL175W   1653.65     -0.2965020  0.107226 0.004112338  0.01365751
YBR046C   4658.16      0.2561617  0.204734 0.145803481  0.25478985
YGR285C   2828.64     -0.0804100  0.260468 0.694253627  0.79177260
YNL241C   7372.76     -0.5142418  0.155676 0.000394384  0.00186044
> write.table(res2,"DE_genes_list.txt")
> write.table(res2,"DE_genes_list.txt",quote=F,sep=)
```

So, if you look at the DE gene list, of course, we have to use these other things that we have used. We do not want code, right, and we want tabs separated by, right, and this should be created in the. So, the DE genes list contains lists of all the genes that are present in the data sets and also the logfold change values. With that data, we can now play around, set our own thresholds, do different sorts of plotting, etcetera in terminal. So, one thing I will just do is finally have a look at the file. We created this DE gene list file, and this is what we get, right? This is the final outcome; all the genes are here. If you can go to the last line, you will see we have 6696 genes, right? o, for a total of 6696, with the header, we have the extra line.

```
rdhar@LAPTOP-3K4C9VBI:/mnt/c/Users/Dhar/Desktop/NGS_Data_Analysis_HandsOn2/Test$ ls
Analyze_RNAseq_data.R                       RUN3_all_S1-S16_analysis.txt
BiasCorrected_RUN3_all_S1-S16_analysis.txt  Rcode_bias_correction.R
DE_genes_list.txt                           SampleInformation.txt
MAplot_G4_vs_G1_afterLFCshrinkage.pdf       steps
Prelim_analysis_Rcode.R
rdhar@LAPTOP-3K4C9VBI:/mnt/c/Users/Dhar/Desktop/NGS_Data_Analysis_HandsOn2/Test$ vi DE_genes_list.txt
```

```
baseMean        log2FoldChange  lfcSE   pvalue  padj
YMR056C 955.120607194351        0.520179644894320.170836336072413       0.000934549791145517
YBR085W 25.565815642665 -0.865976388651881      0.466315842686723       0.00751447130303377     0.022812
YJR155W 246.027065852305        -0.330967060889698      0.214879649306078       0.0748682666104638
YNL331C 284.92366640616 -0.552776652103582      0.134789116061964       1.61935913881358e-05    0.000122
YOL165C 2.95521757462095        -0.109865320556055      0.420216363618564       0.392901542691354
YCR107W 254.968377576616        -0.387054233687017      0.13172993973712        0.00191746607279776
YDL243C 242.722884574902        -0.840548853506414      0.205951302195454       8.20401737878477e-06
YFL056C 160.700173861211        -0.372518921688433      0.181404485330961       0.0228221385077307
YNL141W 490.696828724442        0.051512320515449       0.279331456694578       0.80925463111259
YHR047C 2276.19650159919        0.00228263238922601     0.225304782686553       0.985933260986273
YBL074C 130.406901134153        0.132261509094553       0.129729441736829       0.285044127643592
YKL106W 314.282369851277        1.00671136555084        0.184857005308569       7.51986079700836e-09
YLR027C 9603.562765049  -0.0722340389642158     0.240908702235436       0.714413303294218       0.806139
YBR236C 1981.11713251185        0.0322160709350437      0.0927412970418466      0.721428938933296
YKL112W 2215.78987235199        -0.814681127072458      0.241533390406222       0.000143349004973773
YMR072W 11354.9470568905        -0.211126242739398      0.197514682591335       0.224959553463756
YJR108W 23.103820582033 -0.0167745446738419     0.242257552637552       0.931685151968172       0.956877
YCR088W 4045.8263668076 -0.392266337248338      0.0732007960238405      5.05373298411098e-08    7.743089
YOR239W 3943.3242292661 -0.00669511022976417    0.215709982767522       0.968215116978954       0.979628
YNR033W 1475.86497877305        -0.554636254005403      0.105062660892763       5.04055528726213e-08
YMR289W 898.719673510543        0.540531229555701       0.114355775198941       9.26559255329309e-07
YER045C 352.243373727123        -0.457232695824126      0.196027131029593       0.00890418072984397
YGR037C 1550.2360305652 -0.946359997385729      0.103188002991658       7.89039513502148e-21    1.682181
YNR016C 2929.58664118202        -0.956160939582643      0.192497882224608       1.03220944833316e-07
YLR131C 597.05555376813 -0.818272730148075      0.271941850603095       0.000459738207498256    0.002118
YLR144C 1207.40367836374        -0.0202287578634089     0.0895731020087642      0.816194478659525
:se nowrap                                                                                      1,1           Top
```

We have the average mean expression; this is the first column here, then, of course, after the gene name, you have the log2 FoldChange value, you have lfcse, you have the p value, and the p is adjusted, right? So, these are the two values that we want to work with, for example, the lock-to-pole change value and the p-adjusted value. These two are the most interesting ones for us, and we can do further analysis, plotting etcetera with this data once we have generated this file, ok? So,

this is the first part of the DESeq analysis, right? The next part we will use is that we will utilize a data set where we have the spike present. So, we will have this spike in the mix, with which synthetic RNA molecules have been added to the samples in an equal amount, and based on that, we will do normalization and differential gene expression analysis, and so that will be for the next class. Thank you very much.