Bioinformatics lab: Part 1

This is part 1 of the bioinformatics lab, which will guide you through DEG analysis from the last paper of the course. You'll start with a PCA plot of the expression data, then find and plot DEGs. The next part of the lab will dive into gene ontology and enrichment analysis. Below are the new functions for part 1:

```
BiocManager::install()
head()
t()
prcomp()
model.matrix()
lmFit()
eBayes()
topTable()
```

There are 8 question checkpoints in **bold** throughout this first part of the lab, so make sure you answer all of them outside of the chunks of your R Markdown file. Part 2 will have more questions to answer.

Section 1: Set-up

1. Up until now, we've been using packages from a resource called CRAN. As you know, the function to install from CRAN is `install.packages()`. However, this lab will use packages from another resource called Bioconductor, where all of the packages included are curated for use in bioinformatics. You can't use the usual function to install these packages. First you need to install the package manager using `install.packages("BiocManager")`. Then, anytime you want to isntall a Bioconductor package, use `BiocManager::install("packagename")`. The beginning of that line, `BiocManager::`, calls the package manager, while the second part, `install()` is the function to actually install the package.

2. After you've install BiocManager the usual way, use the Bioconductor method to install a package called limma, like so: `BiocManager::install("limma")`. Remember to run all this in the console of RStudio.

3. Open a new RMarkdown file and load in the necessary packages in the first chunk. Although you need to use a different method to install Bioconductor packages, you can load them in as usual.

```
library("limma")
library("ggplot2")
```

4. Download and import the file `Shi_all_genes.csv` off the course website to a variable, with `row.names=1`. This dataset has over 20,000 genes in it, so instead of using `View()` to look at it, use `head()`. This will print only the first six rows. Notice that each row represents a gene (with its ID as the row name) and each column is a different sample. The values are normalized, log2-transformed values from the microarrays.

```
data <- read.csv("Shi_all_genes.csv", row.names=1)
head(data)
```

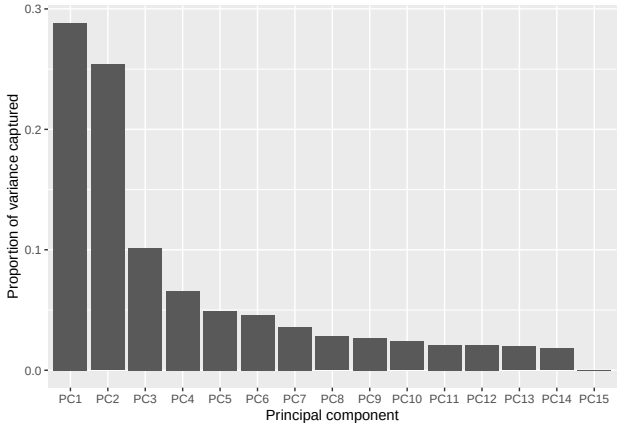
Section 2: PCA (all genes)

5. Start by running a PCA (Principal Component Analysis) to see how well the gene expression profiles differentiate between treatment conditions. To do this, you'll use `prcomp()`. However, this function runs PCA where each row would be a datapoint; we want the datapoints to be samples, which are each shown as a column, not a row. The function `t()` will transpose the data frame so that columns are now rows and rows are now columns, thus yielding the desired results when entered into `prcomp()`. Assign this to `all.PCA`.

```
all.PCA <- prcomp( t(data) )
```

6. Run the code below to make a scree plot for your PCA. A scree plot shows the principal components along the x-axis (so the leftmost bar is PC1, and the second is PC2, and so on), and the variance captured along the y-axis as a proportion of total variance.
   - Notice that the slope changes most drastically between PC2 and PC3. This is known as the *elbow* of the scree plot. After this point, changes in variance between one principal component and the next drop significantly.
   - **Q1. Roughly how much variance is captured in the first two principal components, combined? (Add the variance from PC1 to the variance from PC2)**

```
scree.df <- data.frame( "prop.var" = summary(all.PCA)$importance[2,] ) # prep the dataframe
PCs <- factor(rownames(scree.df), levels=unique(rownames(scree.df))) # reorder x-axis
ggplot(data=scree.df, aes(x = PCs, y = prop.var)) +
  geom_col() +
  ylab("Proportion of variance captured") +
  xlab("Principal component")
```

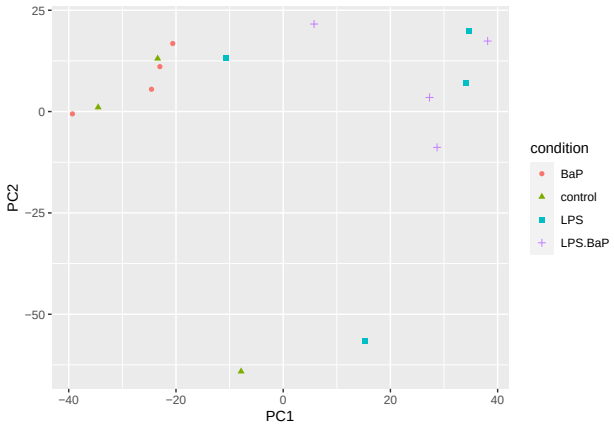


7. Next, use the code below to set up a data frame that you can plot in a PCA scatter plot.

```
all.PCA.df <- as.data.frame(all.PCA$x) # extract PC coordinates for plotting
all.PCA.df$condition <- c(
  rep("control",3),rep("LPS",4),rep("BaP",4),rep("LPS.BaP",4)
  ) # add condition column to PCA.df
```

8. Using the data frame created above, `all.PCA.df`, create a ggplot2 scatter plot with `PC1` as the x-axis and `PC2` as the y-axis. Include `aes( colour = condition, shape = condition )` in the `geom_point()` layer to differentiate between the conditions.

   - **Q2. What's on the x-axis and y-axis of this graph? What exactly do these axes quantify, and how does that relate to the real-world data?**



Section 3: Find DEGs with limma

9. Now you'll use limma (LInear Models for MicroArray data) to find which genes are differentially expressed. limma requires a design model to work, which is basically a matrix that indicates which groups you want to compare. Start by using the code below to create and print an object that lists the condition of each sample, in the same order that they appear in `data`. This is an object of the data type `factor`, which means that any of the contents that are exactly the same will be considered the same level and therefore grouped together. For instance, each `control` is exactly the same so they're all part of the `control` level.

```
factors <- factor(
  c(rep("control",3), rep("LPS",4), rep("BaP",4), rep("LPS.BaP",4)),
  levels=c("control","LPS","BaP","LPS.BaP") )
factors
```

```
##  [1] control control control LPS     LPS     LPS     LPS     BaP     BaP
## [10] BaP     BaP     LPS.BaP LPS.BaP LPS.BaP LPS.BaP
## Levels: control LPS BaP LPS.BaP
```

10. Next, you need to turn `factors` into an actual design model. Use the function `model.matrix()` and the code below to achieve this. Since `control` is the first level listed in `factors`, the function will create a model that compares each of the other conditions to `control`. This is what's called a reference model, in which all conditions are compared to a single reference treatment (usually a control).

```
model <- model.matrix( ~ factors)
```

11. Now that you have a design model, you can fit a preliminary linear model to the data using the limma function `lmFit()`. Upon investigation of the documentation, you should see that this function requires two main arguments: `object`, which in this case contains the log-expression values with columns as

sample and rows as genes, and `design`, which is the design model that you just created. Both of these objects are already formatted as required and ready to go, so run `lmFit()` with `data` as the object and `model` as the design. Assign this to the variable name `pre.fit`.

12. You now have a linear model fit to the expression data, but no DEGs yet. The function `eBayes()` will use a Bayesian method to find moderated t-statistics that can then determine which genes are differentially expressed via significance. Run `eBayes()` with the linear fit from the last step as the only input, and assign this to the variable name `fit`. After you run this, please use `colnames(coef(fit))` to print the coefficients of the fit, which will tell you which comparison you can make. These coefficients were decided based on the design model you used earlier, so they should each represent a condition as compared to the control condition.

```
colnames(coef(fit))
```

```
## [1] "(Intercept)"    "factorsLPS"     "factorsBaP"     "factorsLPS.BaP"
```

13. The code below will finally list the overall DEGs, in a test comparing all conditions and control at once. `fit` is the object you made in the lst step, while `number=Inf` specifies to list any number of genes without limit, `p.value=0.05` sets the alpha, and `lfc=1.2` sets a log fold-change cutoff.

    - **Q3. How many differentially expressed genes did you find in this step, with a max p-value of 0.05 and a minimum log fold-change of 1.2?**

```
results <- topTable(fit, number=Inf, p.value=0.05, lfc=1.2)
```

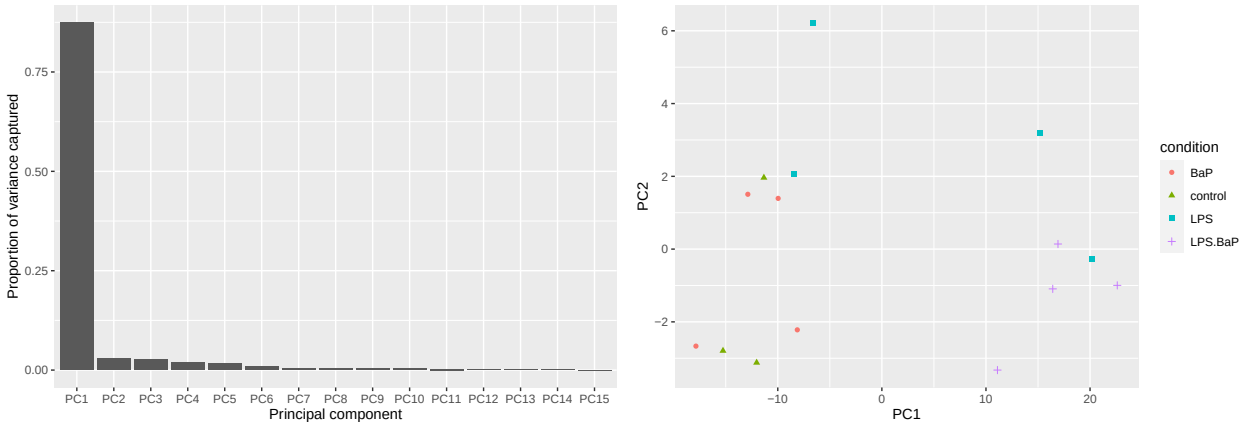
```
## Removing intercept from test coefficients
```

14. For a more detailed analysis, you'll want to compare each condition to the control individually. Use the function `topTable()` again, but this time don't set a p-value or log fold-change cutoff. This way you can get the p-values and fold-changes for genes that aren't differentially expressed too. To specify a specific comparison that you want to make, use the `coef` argument. Set this to the appropriate entry from `colnames(coef(fit))`, which you should have run before. Below is how you would get the results for the LPS-control contrast. Run this line and then write your own lines to get results for the BaP-control and LPS.BaP-control contrasts. Your goal is to get a total of 3 variables named `LPS.results`, `BaP.results`, and `LPS.BaP.results`, respectively.

```
LPS.results <- topTable(fit, coef="factorsLPS", number=Inf)
```

Section 4: PCA with DEGs only

15. For this section, you'll need the overall `results` DEGs from a couple steps ago again. Since `results` only contains DEGs, you can use the row names of this object to select for the appropriate rows of expression values in the original `data` object. With the first PCA as a model, run `prcomp()` on only the rows of `data` that also appear in `results`. Use this new `prcomp()` result in place of `all.PCA` from the original to create a new scree plot and PCA scatter plot from only DEG expression. Your new plots should look the same as those below.



16. Examine the new plots and answer the following question checkpoints:

    - **Q4. Roughly how much variance is captured in the first two principle components of this new PCA, combined? (Add the variance from PC1 to the variance from PC2)**
    - **Q5. Compare the two new plots with the two old plots (scree plots and scatter plots). Which PCA does a better job of differentiating between the sample conditions? Explain your reasoning using both the scree plots and the scatter plots.**
    - **Q6. To the best of your knowledge, why did the model you named in Q5 better differentiate between the conditions? Why would someone expect that this model would perform better in this respect?**

Section 5: Volcano plots

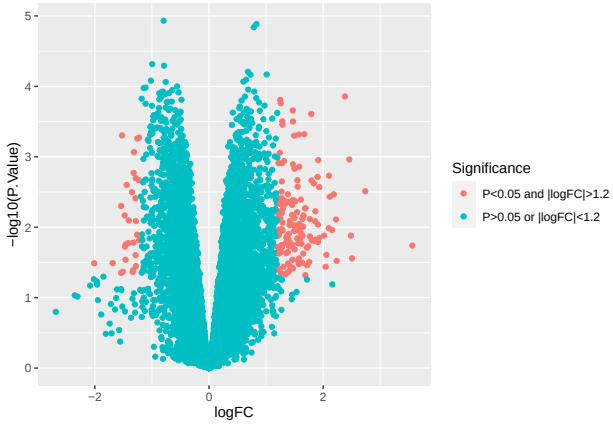
17. For this section, I'll guide you through making a volcano plot for the LPS-control contrast. Then, you'll create the BaP-control and LPS.BaP-control volcano plots on your own. To begin, you'll need to add a column to the results object (`LPS.results`) that indicates both statistical and practical significance. Start with the line below, which will add the column `Significance` and set every row to read "P>0.05 or |logFC|<1.2" in that column.

```
LPS.results$Significance <- "P>0.05 or |logFC|<1.2"
```

18. Next, you need the rows that actually are significant to read as such. Use logical operators to test for two things: `LPS.results$P.Value<0.05` and `abs(LPS.results$logFC)>1.2`. Assign "P<0.05 and |logFC|>1.2" to the `Significance` column of `LPS.results`, but only in the rows that meet these requirements.

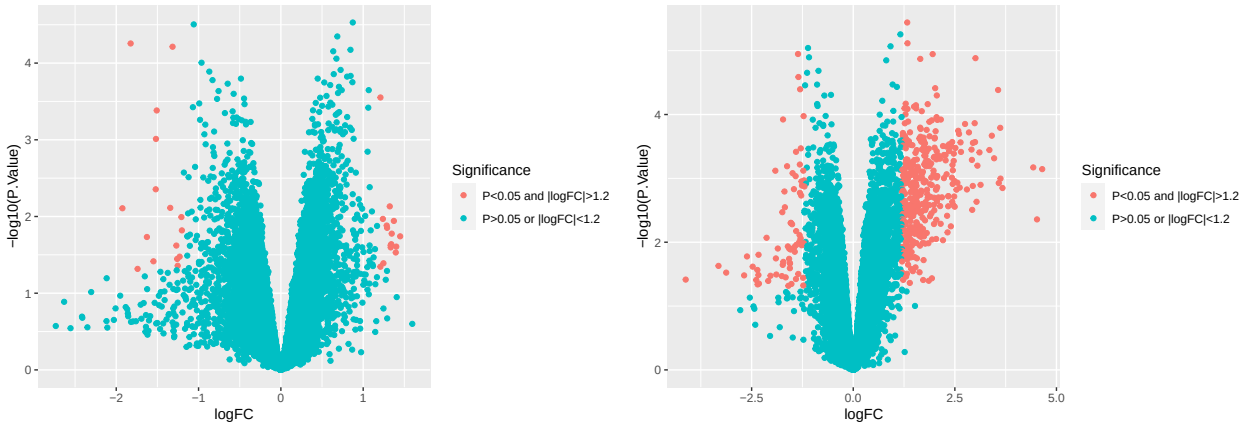
19. Finally, create the plot using ggplot2. The data frame should be `LPS.results` and the `logFC` column should go on the x-axis. For the y-axis, transform the data to make it easier to interpret; set `y = -log10(P.Value)`. Finally, use `aes(color=Significance)` within the `geom_point()` layer so that significant points will be a different color from non-significant points.

    - We're using P-value instead of the adjusted P-value to get a better sense of the overall trend, since not all contrasts have significant genes using the adjusted values, but because of this we're not as confident that these genes are actually correlated with the different treatment conditions.



20. Create volcano plots for BaP.results and LPS.BaP.results using the same method as above, then answer the following question checkpoints:

    - **Q7. Rank the conditions from most DEGs to fewest DEGs as compared to the control condition.**
    - **Q8. Consider the volcano plots and the PCAs from before. Which conditions are most similar, and which conditions are most different? Explain how you came to this conclusion using the volcano plots AND the PCAs; both types of plots should support the same conclusion.**



21. And you're done with part 1 of the bioinformatics lab! Ensure that you've answered all questions Q1-Q8 in bold outside of the chunks, but don't knit or turn in your work yet. You'll submit both parts together at once.

Reference

Shi, Q., Fijten, R.R., Spina, D. et. al. **Altered gene expression profiles in the lungs of benzo[a]pyrene-exposed mice in the presence of lipopolysaccharide-induced pulmonary inflammation.** Toxicol Appl Pharmacol, Dec 2017, 336:8-19. doi: 10.1016/j.taap.2017.09.023. PMID: 28987381; PMCID: PMC5703654.