Lab 6: Regression and MANOVA

In this lab, you'll perform multivariate analyses on gene expression data from the third paper of the course, including MANOVA and linear regression. Then, you'll make some interpretations. The new functions you'll learn in this lab are:

```
lm()
geom_smooth()
cor.test()
boxM()
cronbach.alpha()
manova()
```

There are 11 question checkpoints in **bold**. Make sure you've answered all of them outside of the chunks of your R Markdown file before submitting.

Section 1: Set-up

1. All data today come from NCBI's Gene Expression Omnibus, accession number GSE102016. This is the real data from Shi, et al. (2017), but it has already been pre-processed for each test.
2. You'll need two new packages today, so make sure they're installed by running `install.packages("biotools")` and `install.packages("ltm")` in the console.
3. Open a new R Markdown file. Load in all the necessary packages for this lab in your first chunk.

```
library(ggplot2)
library(tidyr)
library(biotools)
library(ltm)
```

Section 2: Simple Linear Regression

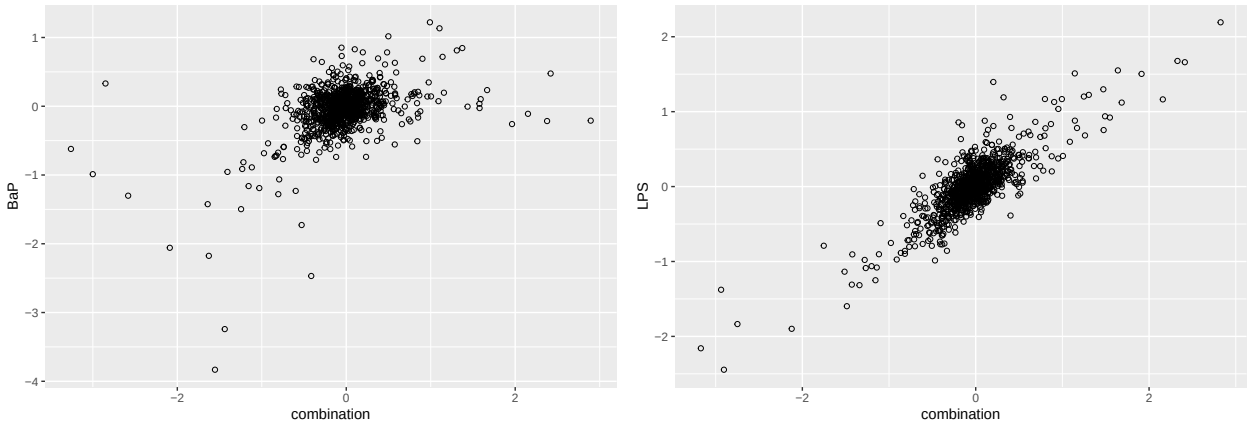
1. Let's start with linear regression. Go ahead and download the dataset `Shi_logFC_subset.csv` from the course website, and read it into R. I'm naming my variable `logFC`.
2. Inspect the imported dataset by either using `View()` in the console or clicking on it in the environment panel in the top right of RStudio. Although you may recall that the full microarray screened for over 20,000 genes, this is a random subset of 1,000 genes that less powerful computers should still be able to handle. The columns represent the four different conditions and the rows represent the different genes denoted by IDs in the leftmost column. Each value is the mean log2(fold-change) gene expression of the replicates for that group, relative to the control. Thus, a value of -1 represents downregulation by half of the control, and a value of 1 represents upregulation by double the control. A value of 0 is the exact same as the control, which explains how the control column consists of only zeroes.

3. Today, you'll run linear regression with the `combination` column as the dependent variable, and the `BaP` and `LPS` columns as the independent variables. The assumptions for linear regression are a little more complicated to check for than the other tests, so I've already done so. For your information however, these are the assumptions that were made:

    - The observations were independent (validated by proper experimental design)
    - The relationship is linear
    - The residuals are normally distributed
    - Variance for the residuals is homoscedastic (i.e., homogeneous)
    - No data point is "pulling" on the regression too much

4. Next, visualize the data. There are ways to plot all three variables together, such as a 3D scatterplot, but let's keep it simple by creating two separate scatter plots: ***`BaP` vs `combination`*** and ***`LPS` vs `combination`***.

    - Use geom_jitter() layers instead of a geom_point() layers to avoid overlaps.
    - For extra measure, change the shape of the points from dots to circles by adding `shape = 1` to both `geom_jitter()` layers.
    - Your plots should look pretty good for now, but you're going to go back and add to them later so assign each one to its own variable, such as `BaP.plot` and `LPS.plot`.



5. Run your code and take a look at your plots. Remember: since you assigned the plot to a variable, you'll need a separate line consisting of only the variable name to actually display the plot.

    - **Q1. Do you think that either of the comparisons in the plots will have some correlation, and if so, which one(s)? If both, does one correlation look stronger than the other? Would they be positive or negative correlations?**

6. For your first linear regression, keep it simple and compare `BaP` to `combination`. Remember that you're trying to see if there's a correlation between the two, with `combination` as the dependent variable and `BaP` as the independent variable.

7. Next, assume an alpha of 0.05 and run the regression. In R, the function is called `lm()`, which stands for Linear Model. `lm()` works very similarly to `aov()`: you need a formula and a data frame to pull from, then you need to use `summary()` to get a good look. The data frame should be the logFC variable you imported earlier, but I'll let you figure out the formula on your own and fill in the blank (copy+paste the code, then replace the underscores with the formula. Replace `logFC` if you named your variable something different.):

```
lm.BaP <- lm(_________________, data = logFC)
summary(lm.BaP)
```

8. In your output, you'll get an estimate, standard error, t value, and p-value (`Pr(>|t|)`) for the intercept of the line and for each of the independent variables. You'll also get the standard error of the residuals, the multiple and adjusted R-squared, the F-statistic, and the overall p-value.

9. Although you can get the correlation coefficient from the `summary( lm() )` output by taking the square root of R-squared, a square root can never give a negative number, so you don't know whether the correlation is positive or negative. To get the proper correlation coefficient, use `cor.test()`.

    - This function also takes a formula and data frame as input. However, since correlation coefficients can't differentiate independent vs dependent variables, we need to set both variables as independent. So they'll both go after the `~`, with a `+` separating them: `~ combination + BaP`
    - `logFC` is still the data frame to use; replace this in the code below if you named your data frame differently.

```
cor.test(~ combination + BaP, data = logFC)
```

```
##
##  Pearson's product-moment correlation
##
## data:  combination and BaP
## t = 16.453, df = 998, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.4117152 0.5093266
## sample estimates:
##       cor
## 0.4619185
```

10. Take a look at both the correlation coefficient and the output from earlier to answer the questions below. Be sure to refer to the values to justify your answers.

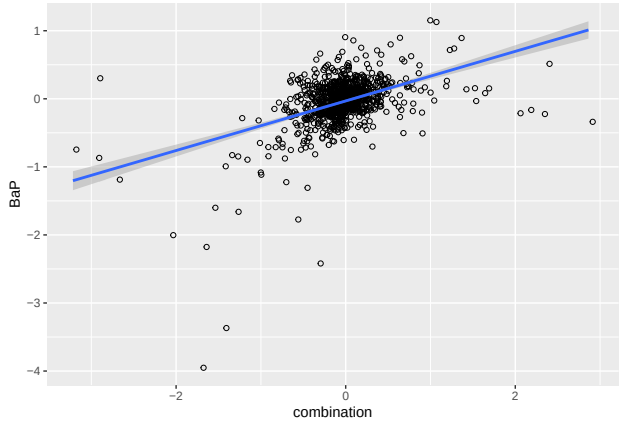
    - **Q2. Is there a correlation? If so, how strong is it (mild, moderate, or strong; just use your best judgment here)?**
    - **Q3. If the `BaP` value were high for a gene, would `combination` be more likely to be high or low for that same gene?**

11. To wrap up simple linear regression, plot the regression on the scatter plots from before. The layer `geom_smooth( method = lm )` will add the regression line based on the function `lm()` that we used earlier. If you used the same variable names that I did, you can copy and paste the below code into your R Markdown file to get the new plots.

    - The darker gray area around the blue line indicates the confidence interval.

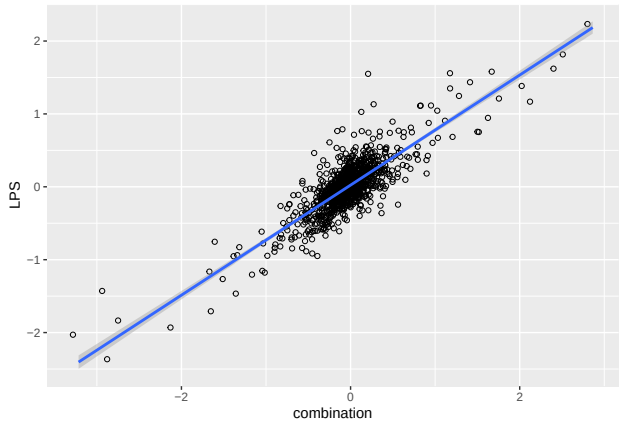
```
BaP.plot +
  geom_smooth( method = lm )
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
LPS.plot +
  geom_smooth( method = lm )
```

```
## `geom_smooth()` using formula 'y ~ x'
```



Section 3: Multiple Linear Regression

12. Although we performed simple linear regression only on `BaP` and `combination`, the model doesn't account for the effect of LPS on gene expression. Therefore, the resulting output may not be the best thing to draw conclusions from. Instead, let's try adding `LPS` into our model using multiple linear regression.

    - **Q4. What is one pair of complementary hypotheses for this multiple linear regression? As always, please label the null and alternative.**

13. Remember that the formula for `lm()` uses the same format as that of `aov()`. Therefore, since we're using two independent variables, we can either separate them with a `+` to exclude analysis of interactions between independent variables or a `*` to include them. Since the interaction could be interesting here, use the asterisk, `*`. The format should therefore be `dependent ~ independent1 * independent2`. Copy and paste the code below into your file, then fill in the blank (replace the underscores) and run it to get a model including all three variables. As usual, replace `logFC` if needed as well.

```
lm.both <- lm(_________________, data = logFC)
summary(lm.both)
```

14. `cor.test()` doesn't work on more than two variables, so we have to get the correlation coefficient from the output of `summary( lm() )` above. The caveat here is that we don't know whether the correlation is positive or negative, but we can at least get the strength of the correlation. Use `sqrt( summary(lm.both)$r.squared )` to take the square root of the `multiple R-squared`.
    - **Q5. How do the simple linear regression from before and this multiple linear regression compare? What do the differences/similarities tell you about the data?**
    - **Q6. Write up a final interpretation of this model that includes all 3 variables. Be sure to justify your interpretation with p-values and the correlation coefficient that you just calculated.**
    - **Q7. What conclusion can you draw about the data based on your interpretation? (Whereas I'm asking for a more numbers-based objective analysis in the last question, here I'm asking you to think about how the results relate to the actual experiment. I'm not looking for anything specific and this will be graded on thoughtful completion, so just do your best.)**

Section 4: MANOVA

16. For this next section, you'll conduct a MANOVA. Download, import, and assign the file `Shi_Cyp1a2_Cyp1b1.csv` to a variable, such as `manova.genes`.

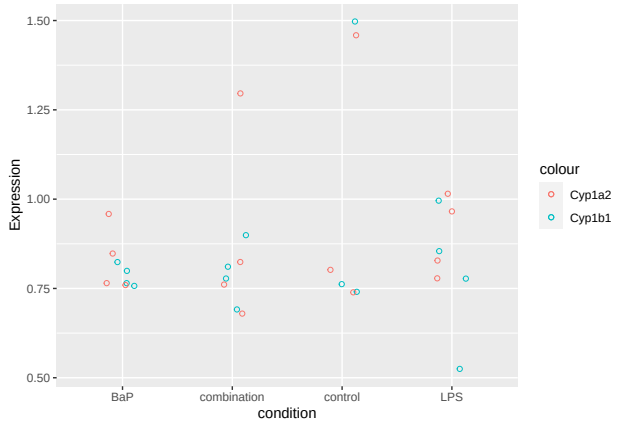
17. Inspect the dataset. These data are expression values for only the genes *Cyp1a2* and *Cyp1b1*, which are featured in figure 1a-b. Each row represents a different sample, and below are details on each column:

    - `condition`: the treatment for each sample
    - `Cyp1a2.raw` and `Cyp1b1.raw`: Processed expression values for the genes *Cyp1a2* and *Cyp1b1* respectively.
    - `Cyp1a2` and `Cyp1b1`: Processed expression values normalized by the means of the control samples. Values below 1 are downregulated relative to the control, while those above 1 are upregulated. Closer to 1 is closer in value to the control mean.
    - `Cyp1a2.transform` and `Cyp1b1.transform`: Normalized expression values transformed for a more normal distribution (more on this later).

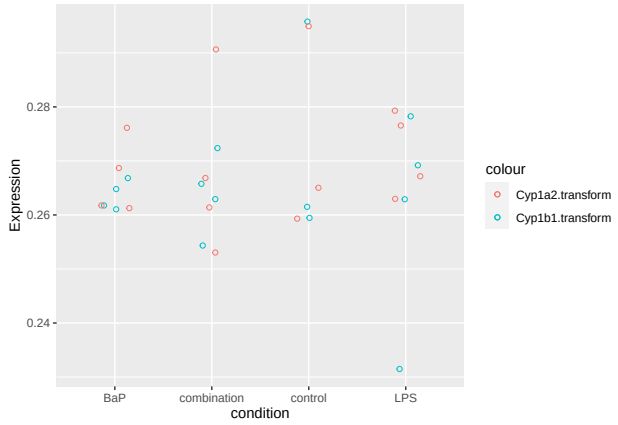
18. You'll use `Cyp1a2.transform` and `Cyp1b1.transform` as the dependent variables in MANOVA. The independent variable will be `condition`.

    - The reason you need to use the transformed data instead of the raw or only-normalized data is because the other columns would violate the assumption of normality for MANOVA. The normalized data was transformed using the Yeo-Johnson transformation. You don't need to know the details, but the idea is that you can transform data by performing certain mathematical functions on them, and this will push the data to a more normal distribution if it's skewed. As long as the same operations are performed on all the data, the relative positions will remain valid for statistical analysis. Common transformations include squaring all the data, taking the logarithm of all the data, taking the square root, and so on. Log transformations in particular are very common in gene expression analysis, as you've already seen in log2(fold-change) values!
    - To see how close the transformed and normalized data are, paste in the following code to create a plot of the data. The first graph represents normalized data, and the second represents normalized+transformed data. Notice how the absolute values change pretty drastically, but the position of most of the dots relative to the others stays roughly the same. Only the extreme values have a more noticeable shift, since they lie on the tails of the distribution.

```
ggplot( data = manova.genes, aes( x = condition )) +
  geom_jitter( aes(y = Cyp1b1, color="Cyp1b1"), width=0.15, height=0, shape=1 ) +
  geom_jitter( aes(y = Cyp1a2, color="Cyp1a2"), width=0.15, height=0, shape=1 ) +
  ylab("Expression")
```




```
ggplot( data = manova.genes, aes( x = condition )) +
  geom_jitter( aes(y = Cyp1b1.transform, color="Cyp1b1.transform"), width=0.15, height=0, shape=1 ) +
  geom_jitter( aes(y = Cyp1a2.transform, color="Cyp1a2.transform"), width=0.15, height=0, shape=1 ) +
  ylab("Expression")
```



19. Use the second graph to draw some preliminary conclusions before running the MANOVA.
    - **Q8. What are the complementary null and alternative hypotheses for this MANOVA? Please be specific.**
    - **Q9. Based on this plot, do you think that the null from your last answer will be rejected? Why or why not?**
20. Now let's check our assumptions. We'll start with the three listed below, so please copy and paste the code to run the tests.
    - Equality of variance/covariance matrices: We'll test this using the Box's M test from the biotools package. Failure to reject the null means the data passed this test. The first input here is a data frame of only the values to be tested, plus a vector indicating the groups of the test.
    - Reliability of variables: An indicator of this would be the Cronbach alpha value, where a higher value is better. The input is a data frame of values to be tested.

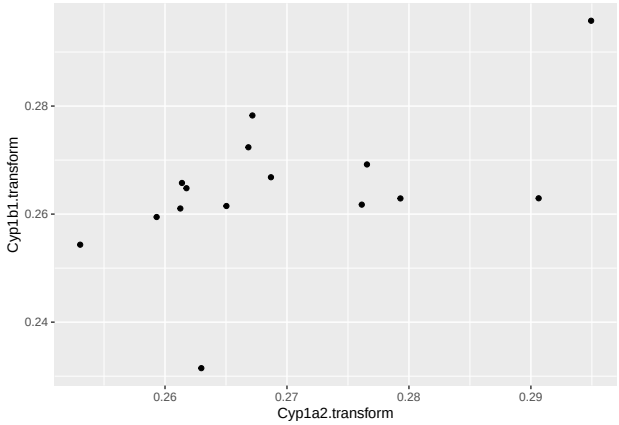
- Correlation: A very low correlation might suggest that regular univariate ANOVAs are a better approach. A very high correlation violates the assumption of no multicollinearity. The inputs are vectors of the values to be tested.

```
# Box's M test
boxM( manova.genes[ , c("Cyp1a2.transform", "Cyp1b1.transform") ], manova.genes$condition )

# Cronbach alpha
cronbach.alpha( manova.genes[ , c("Cyp1a2.transform", "Cyp1b1.transform") ] )

# correlation
cor.test(manova.genes$Cyp1a2.transform, manova.genes$Cyp1b1.transform)
```

21. Next, check for linearity by creating a scatterplot of `Cyp1a2.transform` vs `Cyp1b1.transform`.
    - **Q10. Do the data pass this test? Explain your reasoning.**



22. The last test is for normality. The code is below; you'll notice that one of the results is a little low (p=0.04), so that's something to keep in mind for your final interpretation. The rest of the tests look good, since failure to reject the null suggests that the data is approximately normal.

```
# univariate tests for each condition group
shapiro.test(manova.genes[1:3,6])
shapiro.test(manova.genes[1:3,7])
shapiro.test(manova.genes[4:7,6])
shapiro.test(manova.genes[4:7,7])
shapiro.test(manova.genes[8:11,6])
shapiro.test(manova.genes[8:11,7])
shapiro.test(manova.genes[12:15,6])
shapiro.test(manova.genes[12:15,7])

# tests for all conditions within each gene
shapiro.test(manova.genes$Cyp1a2.transform)
shapiro.test(manova.genes$Cyp1b1.transform)
```

22. Finally, assume an alpha of 0.05 and run the MANOVA. The function to use is `manova()`, and it takes the same inputs that `aov()` did. However, since MANOVAs use two dependent continuous variables, we'll combine the two variables on the left side of the tilde (`~`) using the function `cbind()`. The `condition` will go on the right side of the `~` since it's the single independent variable. See below for how to run the MANOVA and view the results.

```
manova.results <- manova( cbind(Cyp1a2, Cyp1b1) ~ condition, data = manova.genes)
summary(manova.results)
```

```
##           Df  Pillai approx F num Df den Df Pr(>F)
## condition  3 0.19945  0.40615      6     22 0.8669
## Residuals 11
```

23. The output is similar to the ones you've been seeing in the last couple labs. The new addition, `Pillai`, is a test statistic.
    - **Q11. Do you reject the null? What does this mean about the relationship between the treatment condition, *Cyp1a2* expression, and *Cyp1b1* expression?**
24. When you're done, make sure you've answered Q1-Q11 and knit your R Markdown to either pdf or html for submission.

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.