Dennis Pham

Final Report

Introduction/Motivation:

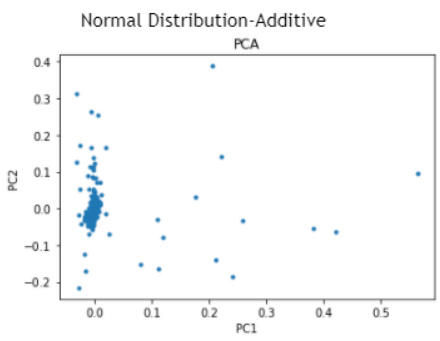
This project focuses on the re-implementation of Sanford *et al.*'s "Gene regulation gravitates toward either addition or multiplication when combining the effects of two signals", using both PCA and PSLR analysis to determine the combined effects of signals on gene activity. The transcription of a single gene can be affected by multiple cell signals that can independently "turn on" or "turn off" cell activity. It is pertinent within biological research to determine the relationship between overlapping cell signals for their potential in technologic or medicinal use. Sanford *et al.*'s model determines that two cell signals that affect the same gene will often result in a response equal to the sum or product of the signals individually. However, it is also interesting to examine overlapping signals with a deleterious or sub additive effect on gene expression. We hope to use PLSR and PCA techniques to further analyse the effect combined gene signals on transcription activity.

Problem Definition:

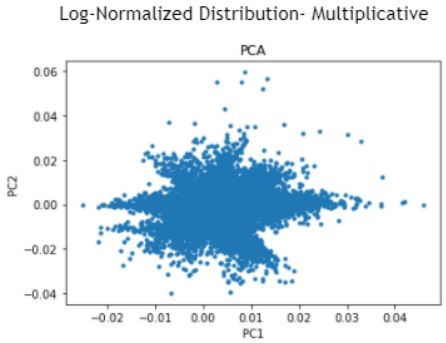
Our project is interesting because it tracks how drugs affect cell signaling, and how transcriptional response can be widely applied to many different aspects and fields of bioengineering. For example, in E. coli protein production, tracking how signals affect protein expression can optimize the production and lower the manufacturing cost of proteins. Most obviously, the motivation behind the project for us was because it showed us how medicines/ drugs can affect cells, whether they are therapeutic or they can uncover previously unseen relationships that can be utilized in other areas of research. Besides, because PCA can assist in determining if the drug is additive or not, this can be used as a primer for more complicated experiments, such as with varied concentrations of drugs or much bigger numbers of samples.

PCA: Normal and Log-Normal

We first wanted to use PCA to analyse the genetic treatments to see which genes were more additive or multiplicative. PCA by nature captures additive responses, because principal components are additive combinations of the data. We surmised that running PCA on the basic dataset would show us which gene/treatment combinations displayed additive responses. We used the pivot_table function to modify our dataset, then ran our PCA with initially 3 components: tpm values, gene names and sample IDs, which is what we're interested in. The following are the PCA graphs for both the normal and the log-transformed datasets:



Left: PCA on data. Shows additive genetic responses

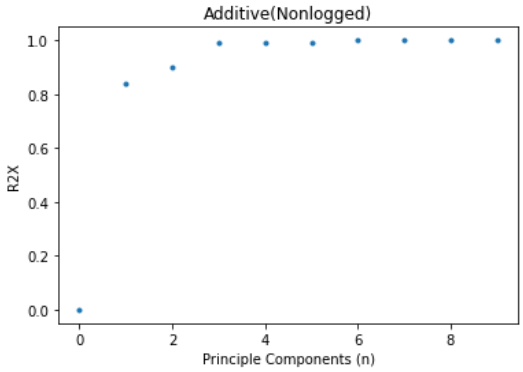


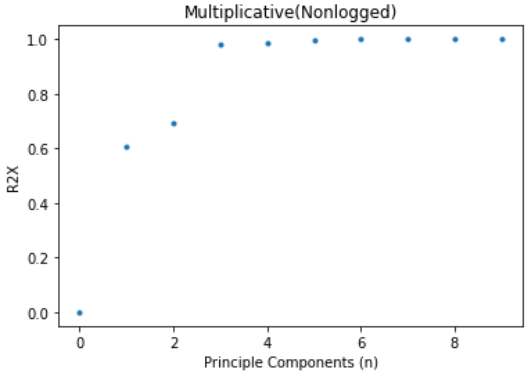
Right: Logged data points, then applied PCA Shows multiplicative genetic responses

It was difficult to draw concrete conclusions from these graphs. There were not very many overarching patterns to go off of, or sure ways to determine if one particular gene has an additive or multiplicative response. To try and get more precise results, we turned to analysing the R^2X values for differing principal components.

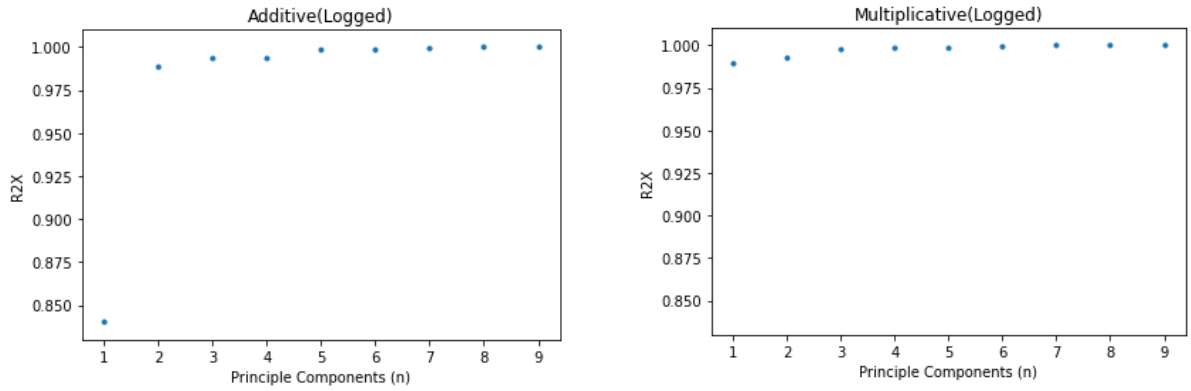
R^2X:

The original 37x27311 dataframe was manipulated into two 9x27311 dataframes. The first new dataframe was put through PCA analysis and the variance (R^2X) of a known additive gene (GPRC5A) and a known multiplicative gene (EPHB2) was plotted against the principal component numbers.





As expected, PCA explains more variance for the additive gene since principal components are additive combinations of the data. Then the second (logged) data frame was put through PCA analysis and GPRC5A and EPHB2 was again plotted for R^2X versus principal component number.

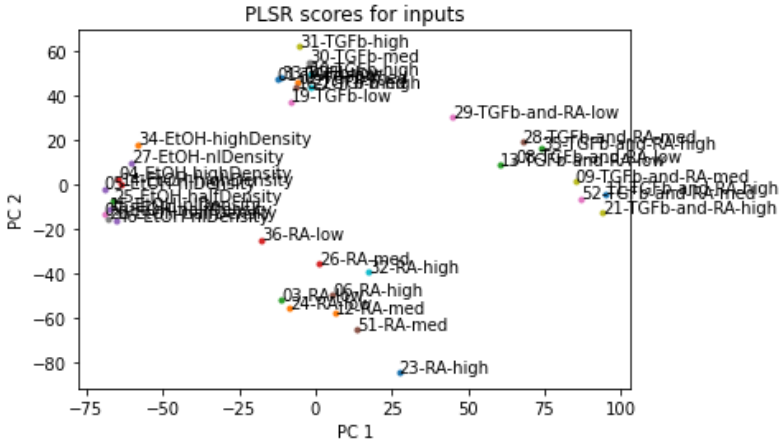


This also expectedly shows what we want because the manipulated logged data frame makes the multiplicative combinations look additive, which PCA captures better, as shown by the higher variance for the multiplicative gene.

PLSR:

Tpm vs counts PLSR scores cluster:

We then performed PLSR with tpm as input and gene count as output. Surprisingly, the scores plot showing the different treatments had four distinct clusters for each type of treatment, regardless if they were in low, medium, or high doses. This shows that specific types of signalling affects the behavior more than the amount of signalling. The following are our PLSR results (notice the four distinct groups of treatments):



PLSR for Prediction:

To predict transcription amount from retinoic acid and TGF-□, the data frame was reformatted by replacing the 'sampleID' string column with numerical 'TGFb' and 'Retinoic Acid' columns. The genes transcription amounts were then averaged over the different treatments. To fit this table for PLSR X=observations(rows) x treatments(columns) where the

values=values of ‘TGFb’ and ‘Retinoic Acid’ and y=matrix of genes. After this was fitted we recorded the score and PLSR captured ~40% of the variance.

gene_name	TGFb	Retinoic Acid	A1BG	A1CF	A2M	A2ML1	A3GALT2	A4GALT	A4GNT
0	0.00	0	0.377997	0.035485	0.087716	0.263830	0.350411	12.057979	0.319075
1	0.00	50	0.345151	0.034716	0.127103	0.137183	0.000000	13.425112	0.000000
2	0.00	200	0.300917	0.000000	0.162882	0.082244	0.345012	13.311664	0.000000
3	0.00	400	0.203307	0.000000	0.211807	0.069556	0.335723	13.785032	0.000000
4	1.25	0	0.256296	0.000000	0.080188	0.125608	0.000000	11.649623	0.000000
5	1.25	50	0.351086	0.000000	0.054925	0.075671	0.385303	8.973253	0.000000
6	5.00	0	0.230624	0.035837	0.053812	0.110743	0.336408	9.336744	0.193978
7	5.00	200	0.362810	0.000000	0.054184	0.050074	0.676265	9.562037	0.000000
8	10.00	0	0.350926	0.032057	0.092482	0.130880	0.358890	10.128333	0.000000
9	10.00	400	0.419643	0.046901	0.000000	0.098149	0.376772	10.329859	0.000000

Figure 1

Prediction:

We were then able to predict the outputs of sample ‘TGFb’ and ‘Retinoic Acid’. The first sample prediction mirrors the last treatment row in Figure 1. We can get a rough estimate of how far off our predictions are for our inputs instead of just looking at the score. The second sample input is an extrapolation which helps to determine the efficacy of additional amounts of protein.

	TGFb	Retinoic Acid	A1BG	A1CF	A2M	A2ML1	A3GALT2	A4GALT	A4GNT	AAAS	...	ZWILCH	ZWIN
0	10	400	0.362298	0.029668	0.054499	0.054887	0.518592	10.323523	-0.043468	40.068333	...	22.720921	52.02081
1	14	300	0.393041	0.042628	0.004124	0.067310	0.545893	8.713564	-0.029093	38.268409	...	20.779741	35.96603

2 rows × 22114 columns

Figure 2

Neural Network

In an effort to compare PLSR prediction capabilities vs those of a neural network(NN) we used Tensorflow to create an NN with 3 input nodes, 2 hidden layers of 64 nodes, and 1 layer of output for regression. The reformatted datatable where X=sample numerical treatments and Y=’tpm’ of genes was fed into the NN and ran for 100 epochs. The results for regression were

incredibly poor. Using the Keras Accuracy metric repeatedly showed accuracy at zero for every single epoch.

To try to improve the results we reduced the gene matrix output data to one single gene column 'AADAC'. Unfortunately the results were the same as before with poor regression estimation and zero accuracy. Upon further inspection we discovered that NNs need large samples of data to approximate a solution, and since we had only 10 samples which we averaged using Pandas Dataframe methods it likely could not converge on a solution. There was also the option of one-hot encoding the genes as columns for the input and genes with their 'tpm' count for the output but this involved X=(observations x genes) where the values are 1 for the gene that was recorded for that observation and zero elsewhere. This ends up being unworkable as the number of input nodes would have to be over 20,000 alone! With that many columns and 600,000+ rows the NN would take too long to converge and because many genes were only tested once or a handful of times we suspected our results would not dramatically improve.

What I did: R^2X

I assessed PCA's ability to capture additive interactions through its R^2X value over multiple principle components, and then see if log transforming the data would let PCA be able to capture multiplicative interactions better. At first it sounds like an easy problem where you run PCA code on the whole data and then PCA code on log-transformed data, but the data itself did not actually show summation or multiplication in the combination treatments, rather through this equation:

$$x_{induced} = x_{baseline} + \Delta_A + \Delta_B + \left(\frac{\Delta_A \times \Delta_B}{x_{baseline}}\right) \times c$$

When c is 0, then the combination is additive, and when c is 1 the combination is multiplicative. It's obvious why the combination treatment row doesn't really show a summation or multiplication of the individual treatments because the equation itself doesn't reflect it. So the data frame must be manipulated in a way that the combination treatments show simple summation or multiplication. In order to do that, the individual treatment rows were subtracted with the baseline values (from the ethanol control group) to get the Δ_A and Δ_B values, then the combination treatments were also subtracted to get the total transcription change increase from the combination treatments, Δ_C:

$$\Delta_C = \Delta_A + \Delta_B$$

This now shows a simple summation of two values for the first dataframe. But if I were to log all of three values, the multiplication would not show since Δ_C is

$$\Delta_C = \Delta_A + \Delta_B + \left(\frac{\Delta_A \times \Delta_B}{x_{baseline}}\right)$$

which simply isn't just Δ_A multiplied by Δ_B. To combat this and get a new dataframe that captures multiplicative data better, I manipulated the equation to get a new Δ_C that just shows a multiplication between Δ_A and Δ_B.

$$\Delta_{Cnew} = \Delta_A \times \Delta_B$$

$$where\ \Delta_{Cnew} = x_{baseline} * (x_{induced} - (\Delta_A + \Delta_B))$$

Then, with log-transformation of both sides:

$$log(\Delta_{Cnew}) = log(\Delta_A \times \Delta_B)$$

$$log(\Delta_{Cnew}) = log(\Delta_A) + log(\Delta_B)$$

Finally, the combination treatment shows the multiplication of two changes and log-transforming turns it into an additive equation. To further filter out the data, the rows corresponding to the ethanol control group were deleted since they don't contribute to additive combinations. The low and medium dosage treatments were also deleted and only the high dosage treatments remained to guarantee seeing a larger difference between genes that either react additively or multiplicatively. So the original 37x27311 dataframe was manipulated into two 9x27311 dataframes.

Git repo: https://github.com/leonaburime/be275_final_project
Name of file: 'si2-si4_RNA-seq-pipeline-output-normalized.tsv'
Dropbox link
https://www.dropbox.com/sh/fhx7huyhhtf8fux/AAAS_jDsRyaIrwBwVwfR2f7oa/replicates1-3_RNA-seq?dl=0&lst=&preview=si2-si4_RNA-seq-pipeline-output-normalized.tsv&subfolder_nav_tracking=1

References

[1] Sanford, E. M., Emert, B. L., Coté, A., & Raj, A. (2020). Gene regulation gravitates toward either addition or multiplication when combining the effects of two signals. eLife, 9, e59388. https://doi.org/10.7554/eLife.59388