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" by focusing on using both PCA and PSLR analysis to determine the combined effects of signals on gene activity. The transcription level of a single gene can be affected by multiple cell signals that can independently "turn on" or "turn off" cell activity. The motivation for studying this effect using machine learning algorithms is due to its applicability in many areas of medicine. For instance, in growing proteins it's invaluable to know which conditions - buffer temperature, media, oxygen level - lead to increased protein expression. In drug discovery and research, it can be helpful in predicting what effect small molecules and proteins can have on specific genes without running time-consuming experiments and costly assays. Sanford *et al.*'s model determines that many combined cell signals show additive or multiplicative effects i.e. two cell signals affecting the same gene often result in a response equal to the sum or product of the signals individually. However, it's also interesting to examine overlapping signals with a deleterious or sub additive effect on gene expression.

Problem Definition:

We used Python Jupyter Notebook, Pandas(Python Data Analysis Library), and sklearn to characterize similarities in gene regulation, predictions of gene transcription amount, and most effective treatment combinations. We employed several machine learning techniques to not only uncover data patterns, but also to learn new techniques and their limitations.

PCA: Normal and Log-Normal

Methods:

We first wanted to use PCA to analyse the genetic treatments to see if we could find a pattern between additive and multiplicative genes, or even find a way to use our results to determine which genes were additive and which were multiplicative. With 37 different treatments and 20,000+ genes, dimensionality reduction was necessary to find meaningful relationships within the data.

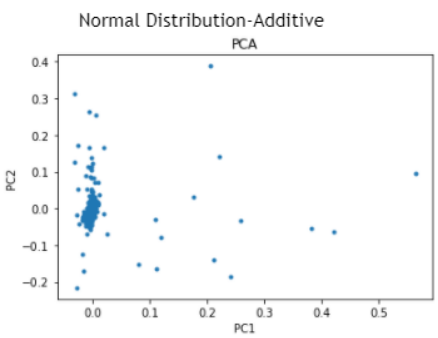
PCA by nature captures additive responses, because principal components are additive combinations of the data. We initially ran PCA with 3 components after using the "pivot_table" function to restrict data to the "tpm", "gene_names", and "sampleIDs" values. To determine the multiplicative effects of genes we log-transformed per the following log relationship:

$$log(x * y) = log(x) + log(y)$$

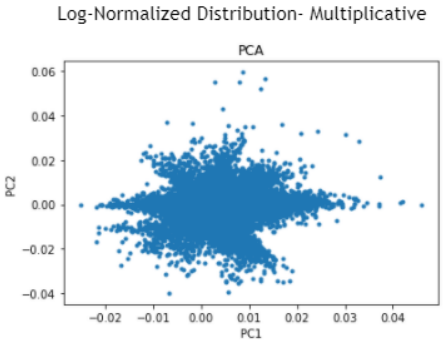
After initially struggling with 0 values in our dataset we removed them altogether as we assumed they held little to no useful information.

Results:

The following are the PCA graphs for both the normal and the log-transformed datasets:



Left: PCA on data. Shows additive genetic responses

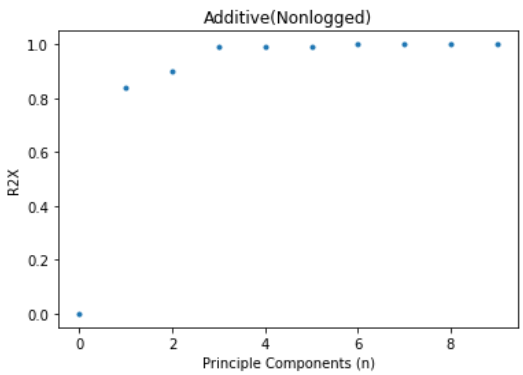


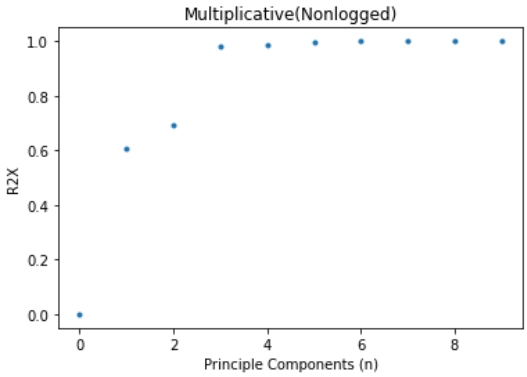
Right: Logged data points, then applied PCA Shows multiplicative genetic responses

The left and right graphs show the additive and multiplicative responses of the genes, respectively. It was difficult to draw concrete conclusions from these graphs. By labeling the genes, we observed which had similar additive and multiplicative responses (i.e., genes that were close together on either graph), and could determine vaguely how the genes behaved in general. However, there weren't many overarching patterns to investigate, or definite ways to determine if one particular gene's response was additive or multiplicative, therefore, we turned to analysing R2X values for differing principal components.

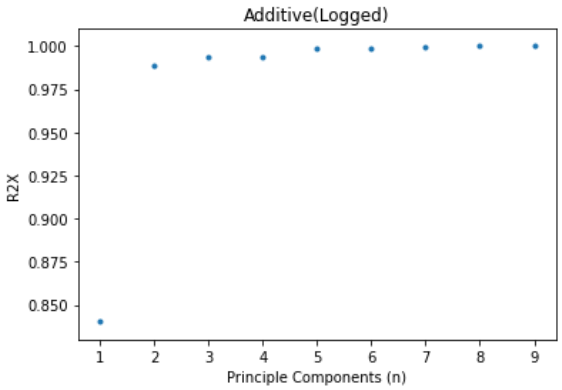
R^2X:

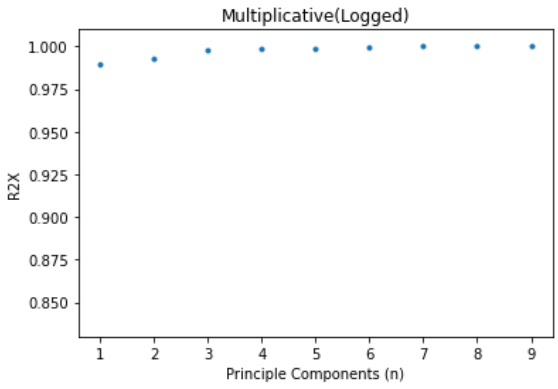
The original 37x27311 dataframe was manipulated into two 9x27311 dataframes. The first new dataframe was put through PCA analysis and the variance (R^2X) of a known additive gene (GPRC5A) and a known multiplicative gene (EPHB2) was plotted against the principal component numbers.





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





This also expectedly shows the manipulated log-transformed data makes the multiplicative combinations look additive, which PCA captures better, as shown by the higher variance for the multiplicative gene.

PLSR(Leon*):

Sample Treatment Scores: To view how treatments were affecting the tpm count we reformatted the dataframe where X=treatments(rows) x genes(columns) where the values='count'(transcription count) and Y=treatments(rows) x genes(columns) where the values='tpm'. We reduced the matrix by averaging the treatments as the experiments were replicated, which polluted our plots by having duplicate treatments with different prefixes i.e. '01-RA-med', '23-RA-med', and '44-RA-med'. Surprisingly, the scores plot shows four distinct clusters for each type of treatment, regardless if they were in low, medium, or high doses. This suggests that specific types of signalling affects behavior more than the amount of signalling.

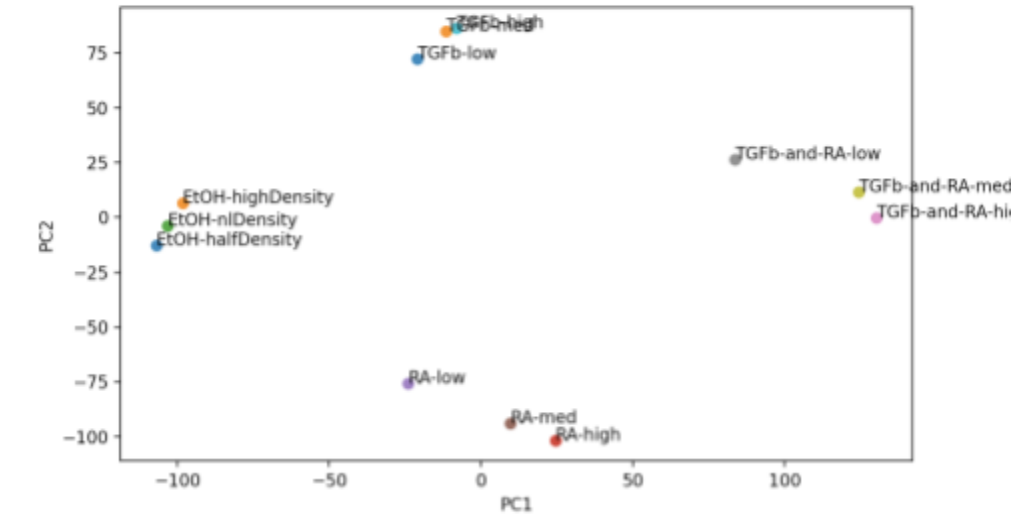


Figure 1

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 2

Prediction: After fitting the model 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 3

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 investigation we discovered that NNs need large samples of data to approximate a solution, and with only 10 samples it likely couldn’t converge. 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 is unworkable as the number of input nodes would 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 wouldn't dramatically improve.

Alternate ML Models(Leon*)

Support Vector Regression: Support Vector Machines are known to be very powerful in their ability to classify high dimensional data. Unfortunately, for Support Vector Regression it cannot fit multi-output models. Fitting the model for a single gene gave us a weak r2 scoring metric and no matter what input was given the output was always the same. Overall, SVR proved to be a poor ML model.

Decision Trees and Random Forest: Decision Trees are a branch-like model that help make choices by using conditional statements. It has a regression counterpart we used to evaluate and predict data similar to the steps for PLSR. The score for both the Decision Tree and Random Forest were exactly 1.0 using the models' default scoring function, suggesting extreme overfit. The cross validation accuracy was in the high negatives, further cementing the notion that our model would not generalize well. In an effort to learn how badly a prediction would deviate we used a 'Leave One Out' method where we removed the highest concentration of TGF-□ and retinoic acid, fitted the model and observed the results. Interestingly, for many of the gene values the predictions 'seemed' fairly close. However, visual inspection of differences between ground truth and prediction cannot override hard data when the official accuracy measurements show poor model fit. Though Random Forests are supposed to provide better results than Decision Trees due to more random subsampling, the results were exactly the same. We can only assume that this was due to a small number of observations.

```
Cross validation scores: [  -13.63055596 -5531.83157602 -7760.67145271  -919.6598702 ]
Decision Tree Regressor score is 1.0
```

Y true

gene_name	TGFb	Retinoic Acid	A1BG	A1CF	A2M	A2ML1	A3GALT2	A4GALT	A4GNT	AAAS	...	ZWILCH	ZWINT	ZXDA	ZXDB	Z
9	10.0	400	0.419643	0.046901	0.0	0.098149	0.376772	10.329859	0.0	40.635957	...	22.338756	57.523884	3.284246	9.952191	36.07

1 rows × 22114 columns

Y predict

	TGFb	Retinoic Acid	A1BG	A1CF	A2M	A2ML1	A3GALT2	A4GALT	A4GNT	AAAS	...	ZWILCH	ZWINT	ZXDA	ZXDB	ZXDC	ZY
0	10	400	0.36281	0.0	0.054184	0.050074	0.676265	9.562037	0.0	42.264973	...	25.416376	66.817892	3.253249	10.199674	36.682645	13.1

1 rows × 22114 columns

```
Cross validation scores: [  -13.63055596 -5531.83157602 -7760.67145271  -919.6598702 ]
Random Forest Regressor score is 1.0


Y true
```

gene_name	TGFb	Retinoic Acid	A1BG	A1CF	A2M	A2ML1	A3GALT2	A4GALT	A4GNT	AAAS	...	ZWILCH	ZWINT	ZXDA	ZXDB	Z
9	10.0	400	0.419643	0.046901	0.0	0.098149	0.376772	10.329859	0.0	40.635957	...	22.338756	57.523884	3.284246	9.952191	36.07

1 rows × 22114 columns

Y predict

	TGFb	Retinoic Acid	A1BG	A1CF	A2M	A2ML1	A3GALT2	A4GALT	A4GNT	AAAS	...	ZWILCH	ZWINT	ZXDA	ZXDB	ZXDC	ZY
0	10	400	0.36281	0.0	0.054184	0.050074	0.676265	9.562037	0.0	42.264973	...	25.416376	66.817892	3.253249	10.199674	36.682645	13.1

1 rows × 22114 columns

Conclusion

We found that PLSR was a really powerful analytical tool that predicted fairly accurate results when comparing the sheer number of outputs versus minimum inputs. We also found that the neural network will not be a useful analytical tool when we have a small amount of data. PCA was found to be very useful in describing data correlation with different treatments before and after log transformation as log transformed PCA can help us interpret multiplicative effects of data. I personally had a lot of fun with this assignment and was extremely grateful it was a project rather than a test as this experience can help me in my future research.

*Sections Leon Aburime mainly worked on

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