Predicting Cell Age from Synthetic scRNAseq data

Thomas Kerby

April 28, 2022

1 Introduction

Single cell RNAseq (scRNAseq) data is known for being noisy and high dimensional. Cells begin as embryonic stem cells and eventually develop into different types of cells. Examples of cell types that an embryonic stem cell will differentiate into includes nerve, muscle, blood, and immune cells among others. Scientists are often interested in determining when a cell begins to differentiate and what genes are responsible for determining the final state of the cell [2].

Because determining when a cell has begun to differentiate is difficult it is often easier to learn the expected age that a cell begins to differentiate and use age as a proxy for differentiation progress. It is fairly easy to keep track of cell ages and track their progression that way. In this paper, the author aims to create models that predict cell age accurately while being able to draw inference from the model to determine the structure of the data.

2 Data

2.1 RNA Sequencing Data

RNAseq data is used to see the presence and quantity of RNA transcripts for a group of cells. It is generally used to estimate gene expression. If there are lots of RNA transcripts associated with a given gene it is likely the RNA transcripts are being translated into proteins; thus, RNAseq is used as a proxy for gene expresssion. scRNAseq is the same process but the measurements are associated with a single cell and not a group of cells. This allows finer grain detail and opens the possibility to comparing different cells from one another.

The difficulty with scRNAseq data is that anywhere to 10 - 50% of the transcipts are collected for a given cell, which means all of the gene counts are under counted. This has many implications and many algorithms have been designed to impute missing or undercounted values. Because of this, accurate synthetic data is attractive since it gives a true model to then compare with the under counted data. There are a variety of tools developed to simulate scRNAseq data, but the chosen for this analysis is dyngen.

2.2 Dyngen Overview

Dyngen [1] simulates scRNAseq data by first creating a directed graph of gene to gene relationships that model a system of interest. The user is able to define not only the relationships but the type and strength of each relationship. To create the raw data, cells have a starting state and are randomly assigned a time. Then, iteratively, the cell samples values for new expression levels given the gene regulatory network defined earlier until the cell reaches the predetermined time. Its gene expression values are then stored with the associated time.

2.3 Synthetic Data

For the experiments, dyngen was used to generate a dataset with 2,000 cells and 5,031 genes. The gene regulatory network used was the default bifurcating backbone which uses 31 genes to drive the cell differentiation. The remaining 5,000 genes were split up such that 1,000 were transcription factors (genes that turn on other genes and are more closely related to the 31 important genes) and 4,000 house keeping

genes (genes that are necessary for life and are not related to cell differentiation).

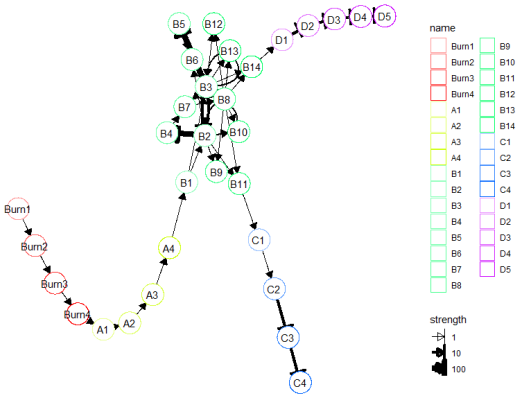

Figure 1: Driving genes in gene regulatory network.

When examining Table 1, we see that B genes are often regulating the genes with the highest correlations with cell age. Of note, B2 was a regulator 3 times, B3 4 times, B6 4 times, and B8 3 times. This is interesting because B6 is regulated by B2 and B8 is regulated by both B2 and B3 which suggests that they are both very important. This is validated by the fact that the B2 gene is responsible for sending cells toward the D pathway and the B3 gene is responsible for sending cells down the C pathway. B2 and B3 repress one another such that a cell will only go down one pathway. It is also not surprising that genes that are turned on toward the end of a cells life are listed as regulators. Namely D3 which is a regulator 7 times as well as C4 which is a regulator 2 times for the highly correlated genes.

3 Methods

To determine model performance, I used a hard partition between a test and a train dataset. I chose three different model types, Lasso Regression, Random Forests, and Feed Forward Neural Networks to examine the relationship between gene expression and cell age. Of note, lasso regression had to be used in place of a standard multiple regression because there are more predictors than observations. This leads to a singular matrix when calculating the beta coefficients unless a regularizing term is added.

Gene Name	**Regulated By**	**Corr.**
Target39	B2, D1	0.370
Target716	B8	0.354
Target714	B8	0.336
Target375	C1, D3	0.323
Target715	B8	0.296
Target360	B2, D3	0.291
Target467	B6, Target23	0.289
Target79	B6, Target1	0.287
Target892	B3	0.284
Target741	B12, Target15	0.281
Target107	C4, Target1, Target2	0.278
Target102	D3, Target1	0.277
Target92	A3, D3, Target1	0.277
Target351	D3	0.275
Target34	B3	0.275
Target202	B2, D3	0.275
Target75	C4, Target1	0.273
Target126	B11, Target2	0.271
Target746	B9, Target16	0.271
Target115	B6, Target1	0.268
Target840	B3	0.264
Target642	B1	0.261
Target137	B6, Target2	0.260
Target310	D3	0.260
Target672	B1, B3	-0.211

Table 1: Top 25 genes with the highest absolute correlation with cell age.

3.1 Model Comparison

Because we are interested in not only model predictive capacity but also the ability to decipher which variables drive age, we need a method to compare model interpretability. Although we do not have the ground truth to determine exactly which variables drive age the most, we do know that the genes listed in Figure 1 are the genes that drive the cell differenti-

ation and as such should be a good proxy for cell age as well. I refer to these genes as the driving genes.

To quantitatively compare models, we will allow each model to select 25 important genes. These genes will then be used in Algorithm 1 to compute the model interpretability score. There are some obvious issues with the algorithm such as assuming that all driving genes are equally important, which as seen from Table 1 we can see that doesn't appear to be true. However, given the difficulty of the problem, I propose weighting them equally as a proxy and reason that it can still give a rough sense of the model interpretatbility if you accept the assumption that the driving genes are important genes.

This model interpretability score will be between 0 and 1. A score of zero indicates that none of the important genes are driver genes or are regulated by driver genes. A score of 1 indicates that all of the important genes selected by the model are driver genes.

Algorithm 1 Computing Model Interpretability

$n = 25$
$vi =$ Variable Importances from Model
$gene =$ List of genes deemed important
$w_i = \frac{vi_i}{\sum_{i=1}^{n} vi_i}$
$score = 0$
for $i \in 1 : n$ **do**
  while $gene_i$ is not a driver gene **do**
    $gene_i =$ regulator of $gene_i$
    $w_i = w_i * .5$
  end while
  $score = w_i + score$
end for

4 Experiments

4.1 Lasso

To tune the lambda parameter, we compared the MSE from the test set with lambda value used in the model as seen in Figure 2. The optimal value for lambda selected is 6.136. After selecting lambda, the MSE for the final model was 10723.23. The 25 genes deemed the most important by the model are those

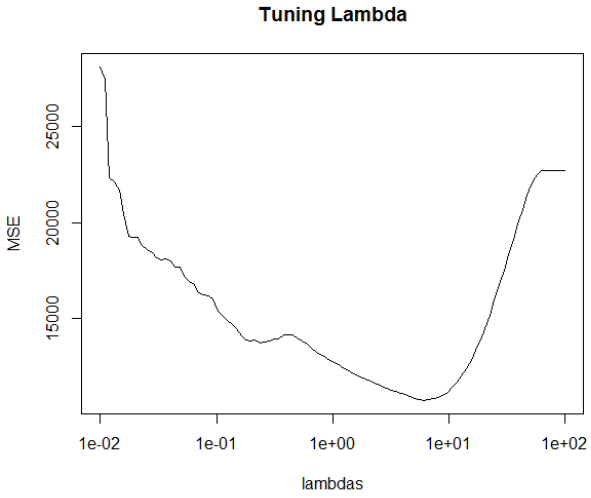

Figure 2: Comparing MSE's generated from the lasso model when varying the lambda parameter.

listed in Table 2. Of note, 6 of them are listed in Table 1 and one, B13, is one of the driver genes.

When we compute the model interpretability score we get 0.4956.

4.2 Random Forest

The random forest was trained with the default parameters. The MSE for the model was 984.50 which is far superior to the lasso model. When we look at the variable importance plot as seen in Figure 3, we see that 18 of the 25 variables deemed important are also listed on Table 1 for having high correlation with cell age. When we look at the score for correctly getting driving variables, we get 0.7032.

4.3 Neural Network

The neural network struggled to learn in this framework. Initially until I was able to properly tune the learning rate, the neural network kept on learning the null model. This is likely because neural networks are data hungry models and we have significantly less observations than we have predictors. Another diffi-

Gene Name	Beta Coefficients
HK2477	97.35434
Target18	47.04507
Target473	45.25426
Target892	33.84109
Target325	32.17078
Target115	31.46609
Target918	30.71390
Target184	30.40129
Target440	30.07492
Target860	29.73931
Target241	28.88838
B13_TF1	27.57796
HK1464	26.82258
Target468	26.50961
Target923	25.57203
Target875	25.32590
Target842	24.98265
HK3883	24.26942
Target442	22.60229
Target450	22.28934
Target574	22.18656
Target672	-21.01602
Target102	20.89633
Target39	19.93176
Target715	19.67148

Table 2: Top 25 genes with the largest beta coefficients in Lasso model.

culty is that the covariates are correlated with one another due to the nature of the data. These coupled together make it difficult to learn a generalizable model using a neural network.

The best results were found by using a standard feed forward network architecture with two hidden layers, the first with 750 nodes and the second with 50 nodes. L1 regularization was added to the weights with a lambda value of .0001. Finally, the MSE was used as a loss, adam was used as the optimizer, and relu was used as the activation function. Tuning was done using a grid search based approach. The final MSE for the network was 25066.65, far worse than the previous two models. As mentioned earlier, training the neural network proved to be difficult with this set

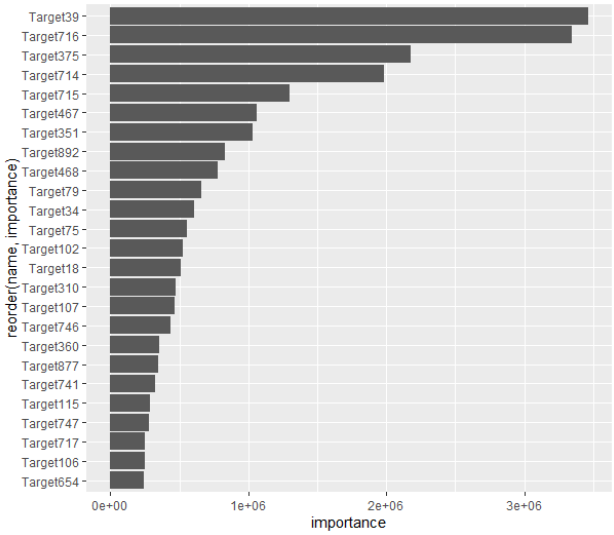

Figure 3: Random forest variable importance plot showing the top 25 variables.

of features. It is possible that using an auto encoder to find a different representation for the inputs, especially a lower dimensional one, would boost model performance.

To generate variable importances from the neural network, a very similar approach was taken as that used to calculate variable importances for random forests. First, a model was tuned and trained. Once the neural network is trained, to determine a given variables importance, we permute its values across observations. We then use the increase in MSE as a measure of variable importance. If the MSE goes up dramatically, it indicates that the variable is important. If it doesn't change much at all, it indicates that the variable is unimportant.

The top 25 features from the neural network can be seen in Table 3. Of the 25 genes, only 1 of them is also in Table 1. Not surprisingly when we look at the score for model interpretability we get 0.2702. This isn't surprising because 16 of the 25 genes listed as important are housekeeping genes which are by nature independent from cell age.

Gene Name	Variable Importance
Target193	-14.061
HK1998	-13.193
Target672	-12.953
HK2076	-11.355
Target709	-9.744
Target860	-9.502
Target629	-9.498
Target742	-9.203
HK2104	-8.887
Target823	-8.791
HK3018	-8.414
HK3160	-8.340
HK836	-8.281
HK552	-8.268
HK2261	-8.133
Target654	-7.928
HK2375	-7.674
HK939	-7.623
HK1443	-7.619
HK2752	-7.592
HK3025	-7.414
HK1711	-7.326
HK2638	-7.266
HK214	-7.238
Target759	-7.170

Table 3: Top 25 variables with the highest importance score in the neural network.

5 Conclusion

As can be seen from the analysis, this is a difficult problem and more work needs to be done to have conclusive results regarding which variables are most important for predicting cell age. Looking at Table 1, we can see that none of the top 25 genes correlated with time are any of the the 31 driving genes. This, coupled with the fact that only 1 variable selected from the three models as important was a driving gene, raises the question of whether these driving genes are important in the first place to predicting cell age. Another limitation of this analysis is that only a training and a testing dataset were used. It is possible that the tuning of model parameters allowed the model to overfit to the test dataset. Therefore, a validation dataset would be needed to accurately measure the model's predictive performance on unseen data.

Of the three models, the random forest model far outperformed the lasso regression and neural network models. A comparison of the model's results can be seen in Table 4. This should not be surprising given the nature of the problem. These variables are highly correlated and involve higher order interactions between one another. A lasso regression is unable to "learn" such interactions and the neural network without sufficient data is also unable to "learn" them. The random forest however, is able to due to its inherent structure.

Model	MSE	Model Score
Lasso	10723.23	0.4956
Random Forest	984.50	0.7032
Neural Network	25066.65	0.2702

Table 4: Comparing models.

In the future, it would be interesting to see how sample size and the number of housekeeping and target genes effect these model's ability to find the driving variables. Future work could also include feature engineering, feature extraction (through ICA, PCA, etc), or using an autoencoder to create new features to plug into the models.

References

[1] Cannoodt et al. "Spearheading future omics analyses using dyngen, a multi-modal simulator of single cells". In: *Nat Commun* 3942.12 (2021). DOI: https://doi.org/10.1038/s41467-021-24152-2.

[2] Saelens et al. "A comparison of single-cell trajectory inference methods". In: *Nat Biotechnol* 37 (2019). DOI: https://doi.org/10.1038/s41587-019-0071-9.

Code available at: https://github.com/tjkerby/Advanced_Regression_Final_Project