Modeling Gene Accessibility in ATAC-seq Time Series Data

Olivia Flynn

1. Introduction

Gene regulatory networks (GRNs) are graphs of genes that interact with each other through the proteins they produce. They are helpful for solving biological and biomedical problems such as drug design, the study of diseases, and diagnosis/prognosis determination [1].

Each cell in an organism contains an identical set of deoxyribonucleic acid (DNA), which encodes instructions on how to synthesize various proteins in sequences of DNA base pairs. The DNA is segmented into coding (gene) and intergenic regions, and each gene produces proteins through the process of gene expression. A single gene may produce a set of proteins called isoforms through varying steps in the expression process. The isoforms of a gene vary in structure, which leads to a potential difference in function as well.

A gene's expression can be initiated, excited, or inhibited when specific proteins called transcription factors (TFs) bind to DNA sequences associated with the gene, called regulatory elements (REs). The type of RE that initiates gene expression is called a promoter, and a single gene may have one or more promoters (resulting in isoform variations). For a TF to bind, its structure must be compatible with the RE sequence and the RE must be accessible. Chromosomes are extremely compact structures of chromatin (DNA and other proteins), and the DNA can only be accessed for binding when the chromatin is uncoiled. Once a gene is expressed, the resulting protein may be transported elsewhere or remain in the cell nucleus to bind to REs, further regulating gene expression in the cell. Genes both create and are controlled by TFs, forming GRNs that outline the interactions underpinning numerous biological processes.

To analyze the dynamics of gene activity, the cell environment must be captured and represented as digital data. DNA is sequenced to extract various properties such as the levels of gene expression (RNA-seq) and the chromatin accessibility across the genome (ATAC-seq). Aside from the work detailed in this report, a data collection experiment is being performed in parallel with the goal of generating a time series of RNA- and ATAC-seq datasets for a fruit fly genome after mating. While the current modeling work uses data from the fruit fly species Drosophila melanogaster (D. melanogaster), the data generated in the parallel experiment will be from a different fruit fly species. The objective of this work is to create a model that predicts the chromatin accessibility of the genome in the next time step, given only information in the current time step. The predictions of such a model could reveal previously uncovered patterns in accessibility that are useful for inferring GRNs.

The rest of the report is organized as follows. In Section 2, we survey some existing methods for inferring GRNs from combined gene expression and chromatin accessibility data, and describe the preliminary data used for our analysis. Sections 3 and 4 describe the analysis and results completed so far, and Section 5 discusses general observations and future directions. All scripts mentioned in the report are open source and available on the internet[1].

2. Related work

There are several existing methods for inferring GRNs using a time series of gene expression and chromatin accessibility data. Surveying such models can inform the development of our model, and allow the newly generated data to be analyzed for GRNs during the development of our model. Clustering is a common theme among the approaches, and one salient issue among approaches is their species-specific operation.

2.1 Time Course Regulatory Analysis (TimeReg)

Duren et al. propose a method of inferring and analyzing GRNs using a time course of gene expression and chromatin accessibility data called time course regulatory analysis(TimeReg) [2]. TimeReg consists of three major steps. First, paired expression and chromatin accessibility data is used to infer GRNs at each time point by generating matrices that represent the association strength between each transcription factor and target gene (TF-TG matrix), and between each regulatory element and target gene (RE-TG matrix). Next, non-negative matrix factorization is used to identify core regulatory modules within the TF-TG matrix. Lastly, the regulatory groups are compared from one time point to another using Jaccard similarity in order to observe the expression behavior of the modules over time and identify elements driving regulation. Additionally, gene ontology (GO) enrichment analysis was performed on various groups of genes throughout the process to examine the salient associated biological functions.

[1] https://github.com/oaflynn/atac-seq-models

Two limitations of the TimeReg software prevent immediate use with our fruit fly data. First is the issue of access to a suitable environment to run TimeReg. The software must be run on a Linux system with a Matlab license. Second, the TimeReg software currently only supports the analysis of mouse and human data. The software depends on several sources of prior data about the genomes, such as TF-TG correlations across external public data, motif binding strengths, and information about cis-regulatory elements.

2.2 SOMatic

Jansen et al. propose a method of analyzing paired RNA- and ATAC-seq data sets in single cell (sc) format using self organizing maps (SOMs) [3]. In single cell data, information about several individual cells are included, while bulk data describes aggregate information over a group of cells. A SOM is a type of neural network that clusters and classifies data by fitting the nodes of the network to the distribution of data [4]. In [3], a single data point represents a gene in the RNA-seq or a peak in the ATAC-seq, and the features were the expression level/peak signal in each of the cells sequenced. A software package called SOMatic[2] creates separate SOMs for the scRNa- and scATAC-seq data sets, partitions them into metaclusters, then links the metaclusters from each data set. This results in groups of genes that have similar chromatin accessibility profiles and are located near similarly expressed genes, which are analyzed for motif enrichment. Starting with a single TF/motif with a known functionality, a GRN is inferred by seeing which linked metaclusters are enriched with the motif, and what TFs they produce. A single-step time series was used to verify the regulatory relationship predicted by SOMatic.

The SOMatic package is intended for use with single cell data, which presents an issue because the generated data will be bulk data. It may be possible to modify SOMatic to process bulk data, or to simulate single cell data by treating each replication as a cell. The Github page for SOMatic includes tutorials for creating SOMs from RNA- and ATAC-seq data, creating metaclusters, and linking the metaclusters.

2.3 Big Data Regression for predicting DNase-I hypersensitivity (BIRD)

Zhou et al. propose a big data regression model[3] for predicting chromatin accessibility using scRNA-seq data [5]. Though chromatin accessibility is measured using DNase-seq data in [5], ATAC-seq conveys similar information. The advantage of predicting chromatin accessibility using RNA-seq is that RNA-seq data is more readily available than other types of sequencing data, including ATAC- and DNase-seq.

BIRD is a collection of many regression models-- two for each gene locus in the genome. Each locus has a locus-level model and a pathway-level model. The locus-level model clusters co-expressed genes, selects a subset of the most informative clusters for prediction, and uses those as parameters. This reduces the complexity of the model and mitigates noise in the data. The pathway-level model clusters the genes by both co-expression and correlations between accessibility in training data. After feature selection, the model is fit to predict the mean DNase-seq level in a cluster. The final prediction is a weighted average of the predictions from

[2] https://github.com/csjansen/SOMatic
[3] https://github.com/WeiqiangZhou/BIRD

the locus-level and pathway-level models. In [5], BIRD was successfully used to predict chromatin accessibility in time series data. Zhou et al. also examined the selected parameters of the regression models and found that possible regulatory TFs were enriched in pathway-level models, while few models included target genes or genes that were close to the model's locus.

While BIRD can be used on either single cell or bulk data, the only pre-trained models are for the human genome. The human model was created using RNA- and DNase-seq data from 40 cell types, then evaluated using a set of 17 other cell types. An analogous model could be created for fruit flies in order to predict chromatin accessibility.

2.4 Data FIles

Several data files were used in the subsequent analysis, summarized in Table 1.

Table 1: Summary of the data used for openness analysis and prediction

File name	Description
Gene Transfer Format (dm6.ensGene.gtf)	Specifies the genome structure of D. melanogaster. The relevant information is the chromosome, start and end base pair locations, strand direction, and FBgn ID for each gene.
Narrowpeak ([sample name].narrowPeak)	Information about each significant peak called from ATAC-seq data. The relevant information is the chromosome, start and end locations, signal value (peak height) and p-value.
Promoter file (promoter_single_gene.csv)	Information about the promoter for each gene in D. melanogaster. The relevant information includes the chromosome, start and end locations, gene symbol ID, and strand direction. Currently, all promoters are listed at a fixed length of 60 base pairs.
FBgn-Gene Symbol conversion (fbgn_annotation_ID_fb_2020_04.tsv)	ID mappings between the FBgn and gene symbol formats. Each row is a unique gene symbol and has columns for the primary and secondary FBgn IDs associated with it.

3. Measuring Gene Openness

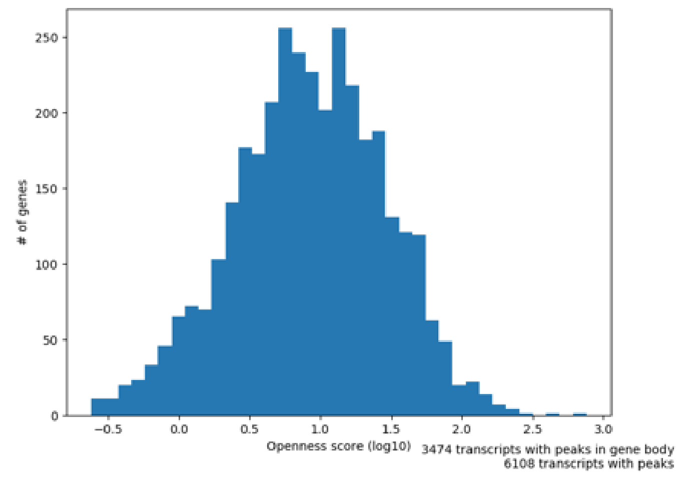
We first combined data from several sources into a more concise representation of chromatin accessibility. Duren et al. [2] proposes one measure of chromatin accessibility within a region of base pairs, called the openness score (Equation 1).

$$O = \frac{X/L}{(Y+\delta)/L_0} \quad (1)$$

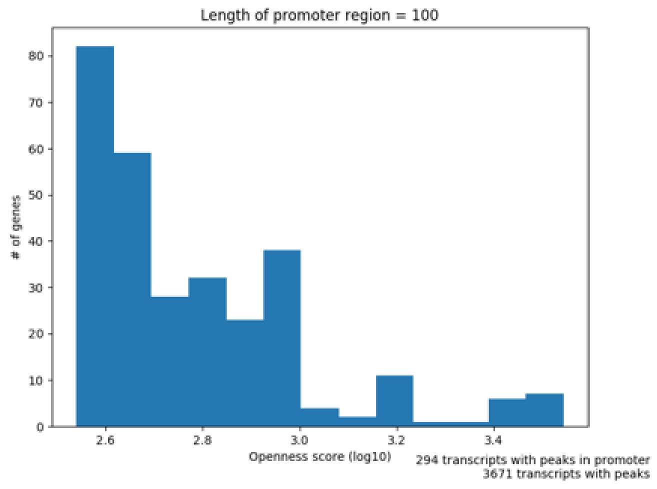
Where X is the sum of all peak heights within the region, Y is the sum of all peak heights within the genome, L is the length of the region, L_0 is the length of the genome, and δ is a pseudocount to avoid dividing by 0 (We use δ = 5). In [2], the X and Y values are computed from transcript reads in RNA-seq data but we have modified the input of the equation to accommodate ATAC-seq data in NarrowPeak format. A script[4] was written to obtain the openness score for a list of genes at a given time point and output a CSV file with this information to be used for further analysis.

4. Analysis of Openness Data

After generating openness score data for each NarrowPeak file, we visualized it using histograms and volcano plots. We first plotted histograms to visualize the distribution of various features. Figure 1 shows these distributions for the 48hr time point of replication 1, with the length of the promoter region set to 100 base pairs.

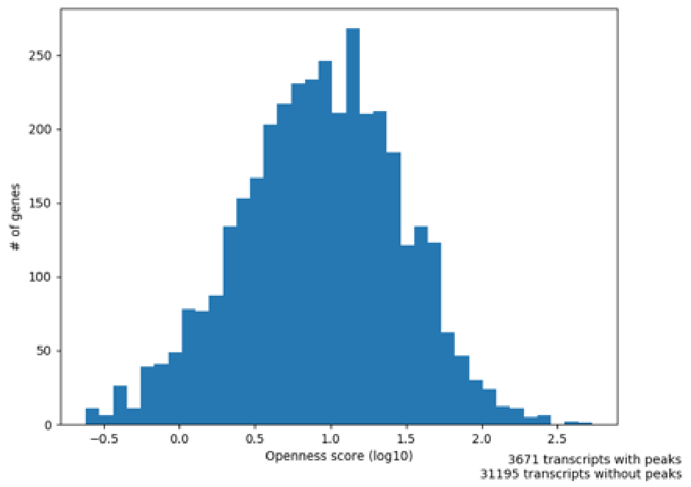


(a)

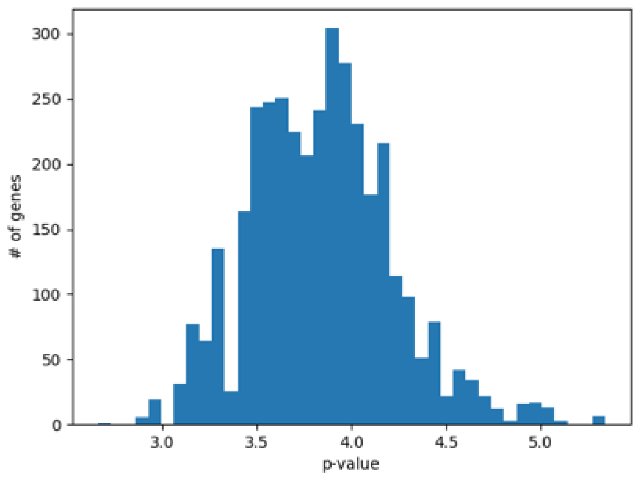


(b)

[4] openness_score.py in the project repository



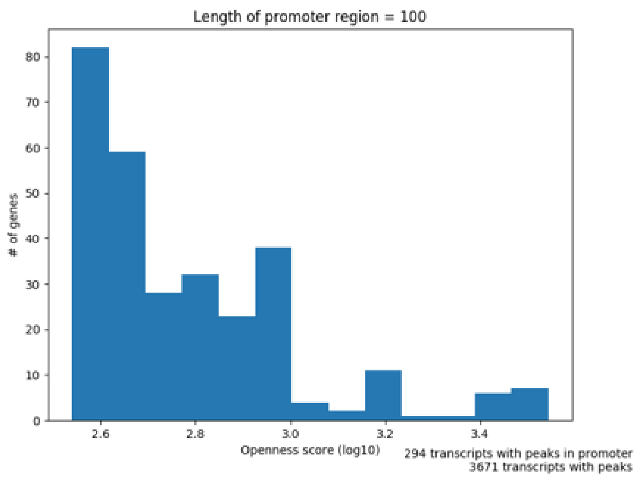
(c)



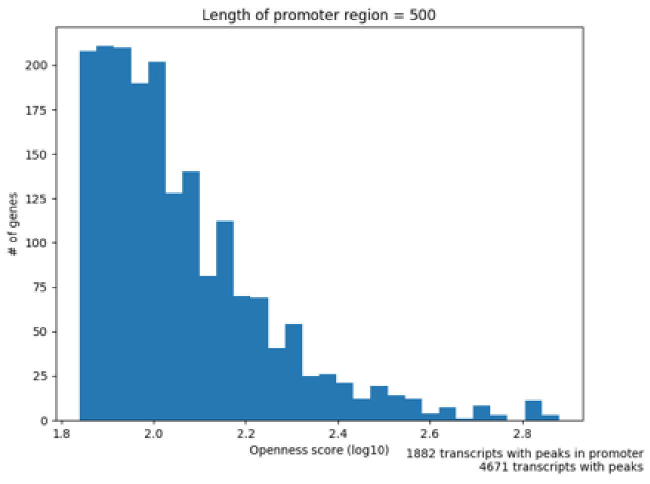
(d)

Figure 1: (a) Distribution of openness scores from the region within the gene body (b) Distribution of openness scores from within the promoter region of each gene where the promoter region is 100 base pairs long (c) Distribution of openness scores combining the gene body and promoter regions of each gene (d) Distribution of the average p-value for peaks included in each gene's openness score

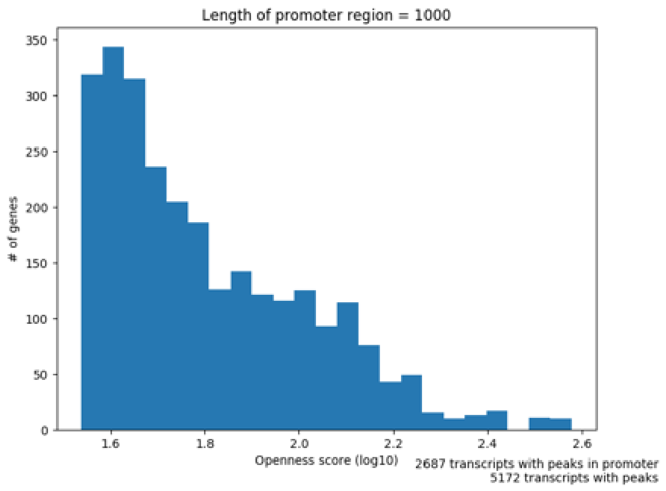
We observe that the distribution of the combined openness scores resembles the distribution of openness in the gene body alone, and that the distribution of the promoter region looks quite different. We plotted the openness score distribution within the promoter region at different promoter lengths to understand why (Figure 2). The reason that the promoter openness distribution has a sharp cliff on the left side is that the L term has a relatively small value. As the value of L decreases, the minimum possible openness score increases.



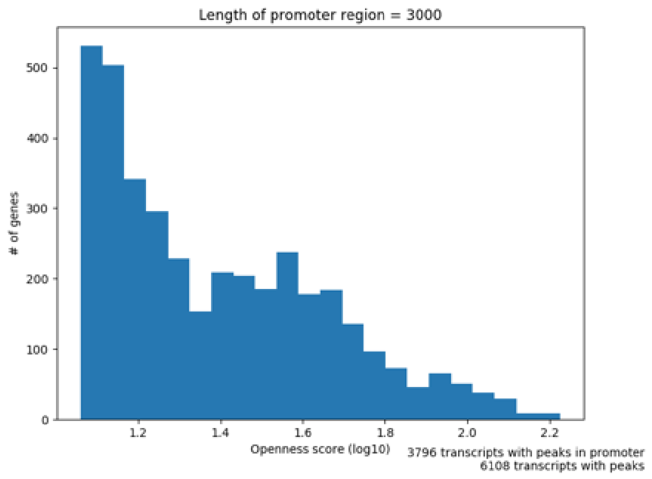
(a)



(b)



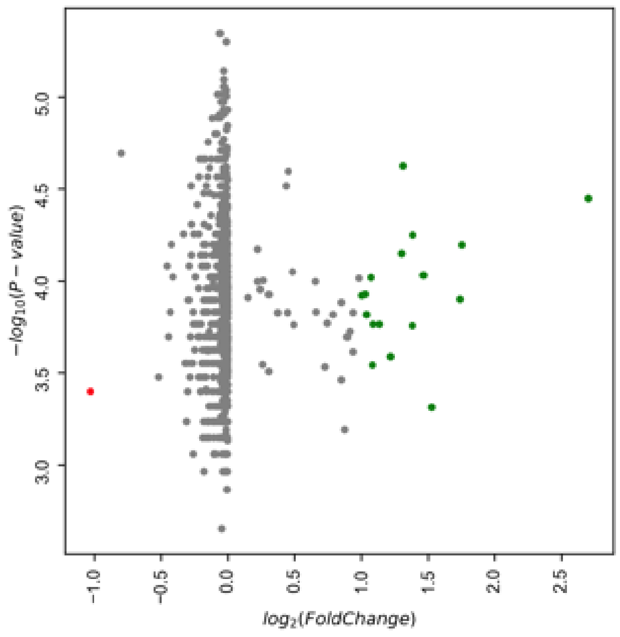
(c)



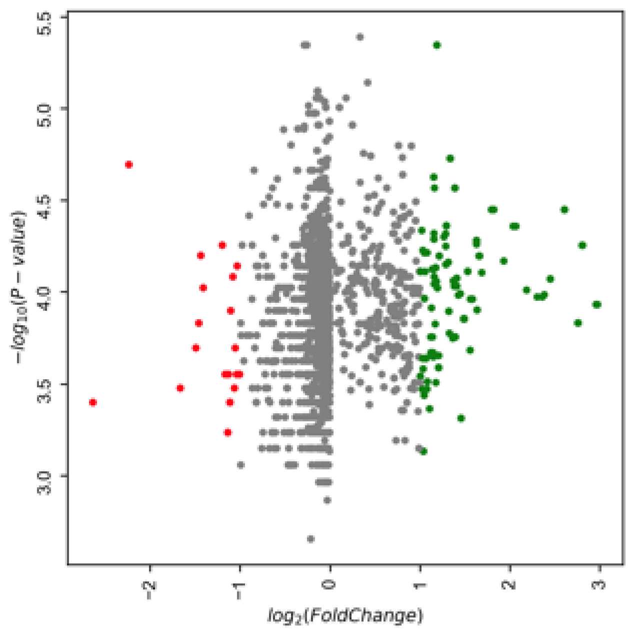
(d)

Figure 2: Distribution of openness scores from the promoter region, where the length of the promoter region is (a) 100 base pairs (b) 500 base pairs (c) 1000 base pairs and (d) 3000 base pairs

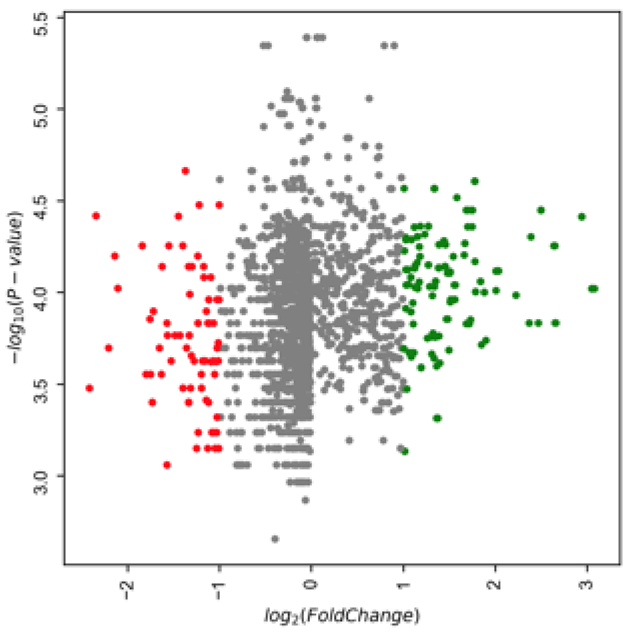
We also plotted the fold change in openness score when considering the gene body region alone vs. the gene body combined with the promoter region, as the size of the promoter region increases (Figure 3). As length of promoter region goes up, the effect on openness score is higher. At first in the positive direction, but longer promoter lengths also increase the L term, causing some openness scores to change in the negative direction due to the X term staying (nearly) constant.



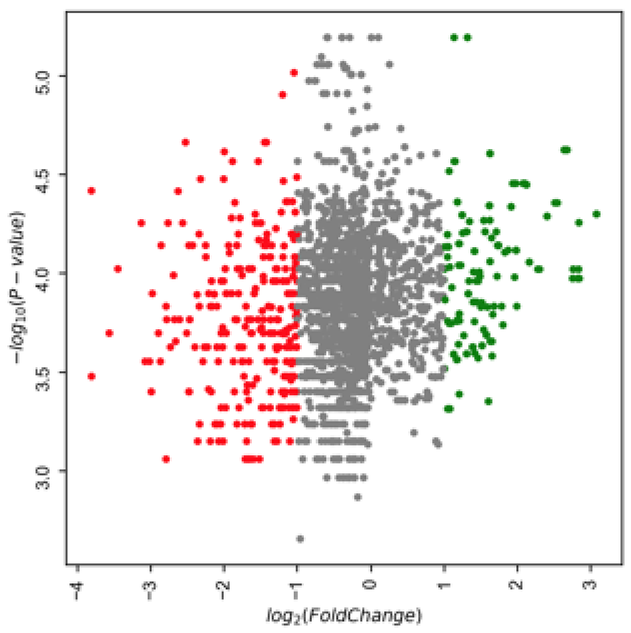
(a)



(b)



(c)

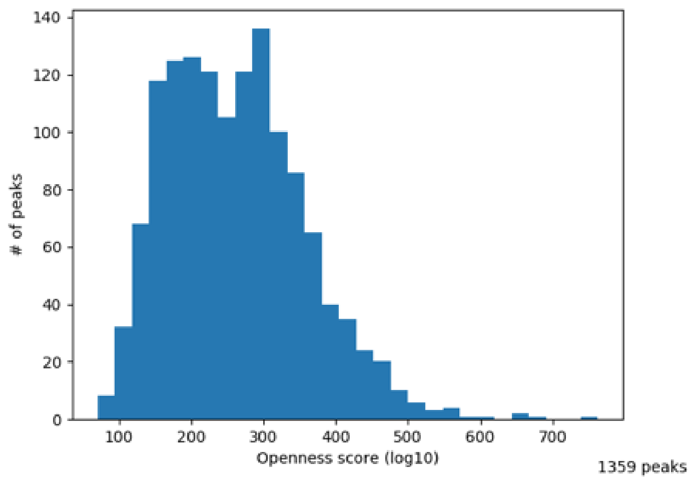


(d)

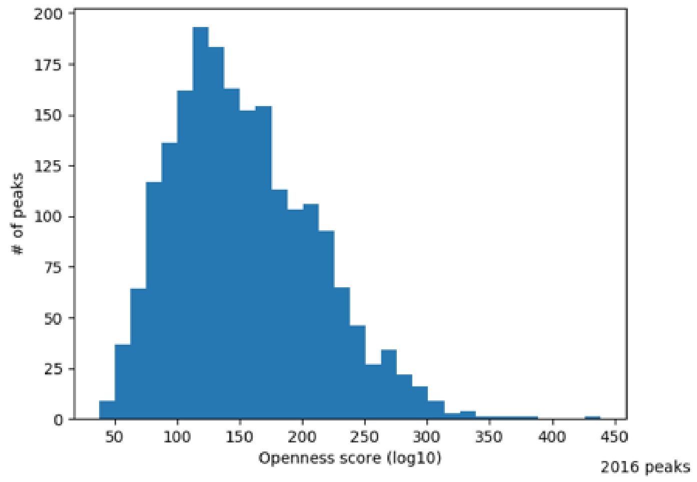
Figure 3: Fold change vs. average p-value for each gene after incorporating the promoter region into the openness score, for promoter region lengths of (a) 100 base pairs, (b) 500 base pairs, (c) 1000 base pairs, (d) 3000 base pairs

Similar plots were created to visualize openness among peaks from the NarrowPeak file at a finer granularity, as opposed to grouping them by gene (Figure 4). However, instead of investigating how openness score changes with the size of the promoter region, we compare how openness changes across a single timestep[5]. We observe that there are more peaks at the 72hr point compared to 48hrs, but the peaks in the 72hr data have a smaller range of openness scores which are generally lower than those at 48hrs. Only peaks conserved over the time step are included in the volcano plot (i.e. a peak exists in the NarrowPeak data for both time steps within 50 base pairs of each other). From this, we observe that only about a third of peaks are conserved, and the openness of these peaks decreases slightly from 48hrs to 72hrs.

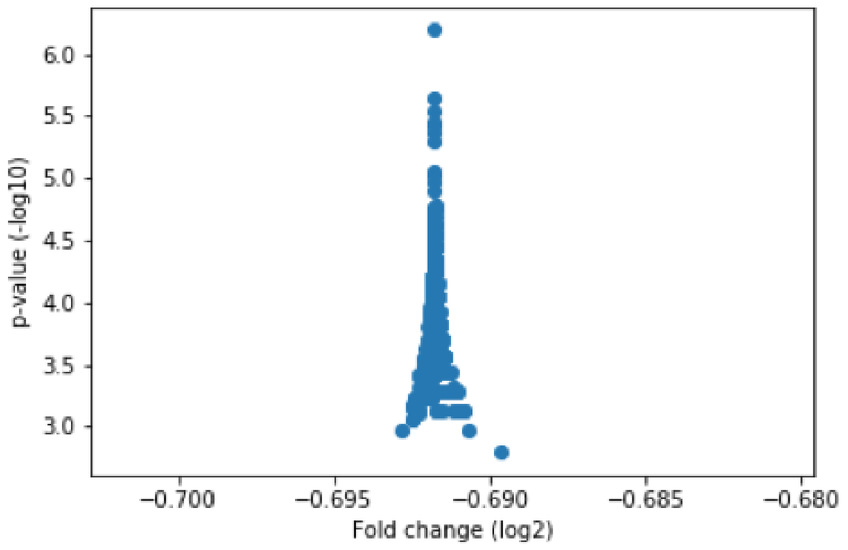
[5] openness_peaks.py in the project repository



(a)



(b)



(c)

Figure 4: Distribution of openness score among peaks in replication 1 at (a) 48hrs (b) 72hrs. (c) Volcano plot of openness score from 48 to 72hrs

After this exploration of the openness score data, we decided to focus on changes in chromatin accessibility around the promoter region. In subsequent analyses, the openness score is generated with a promoter region of 500 base pairs, and only considering the first 500 base pairs within the gene body. Figure 5 shows the distribution of various values within the generated data.

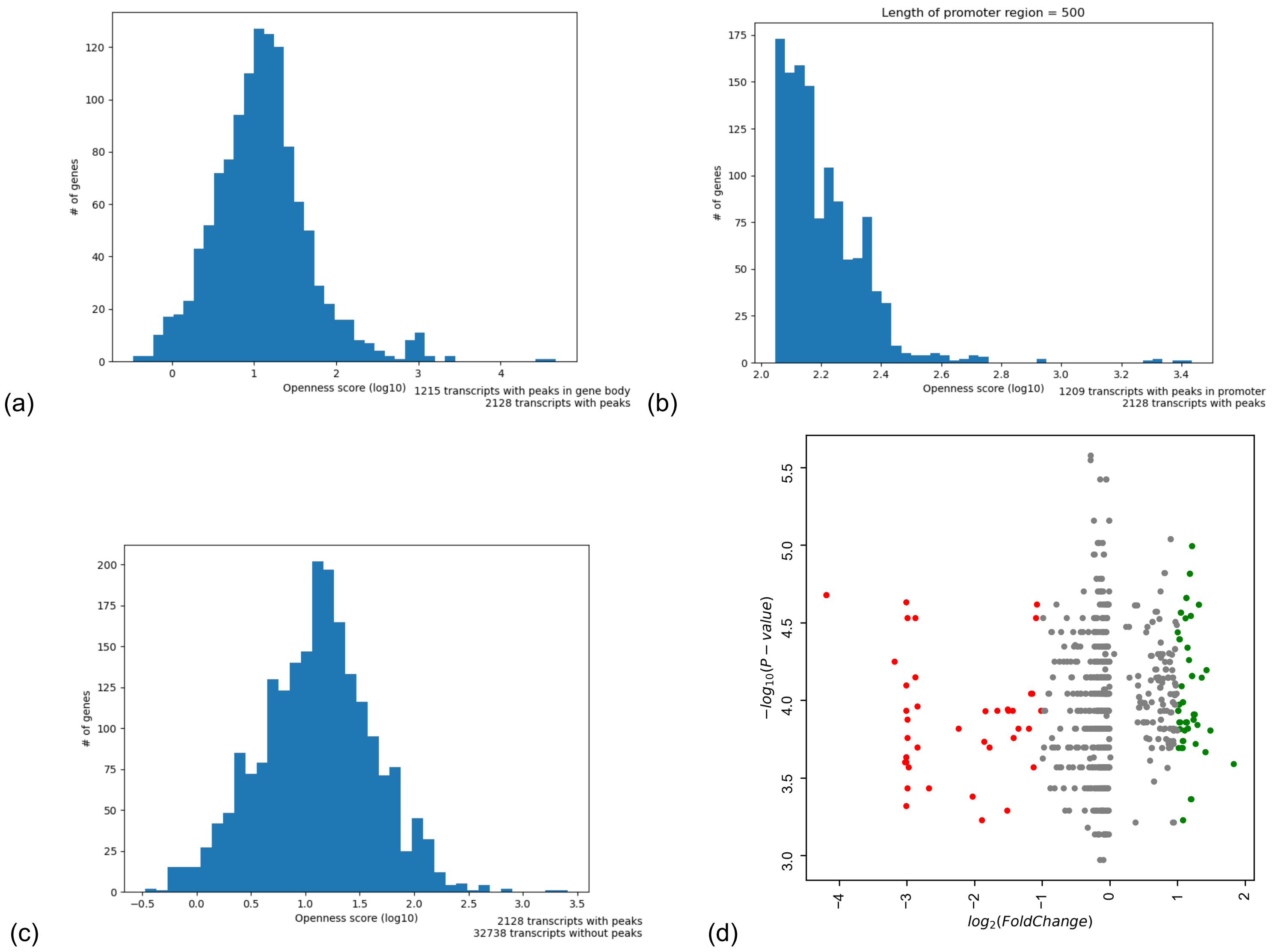
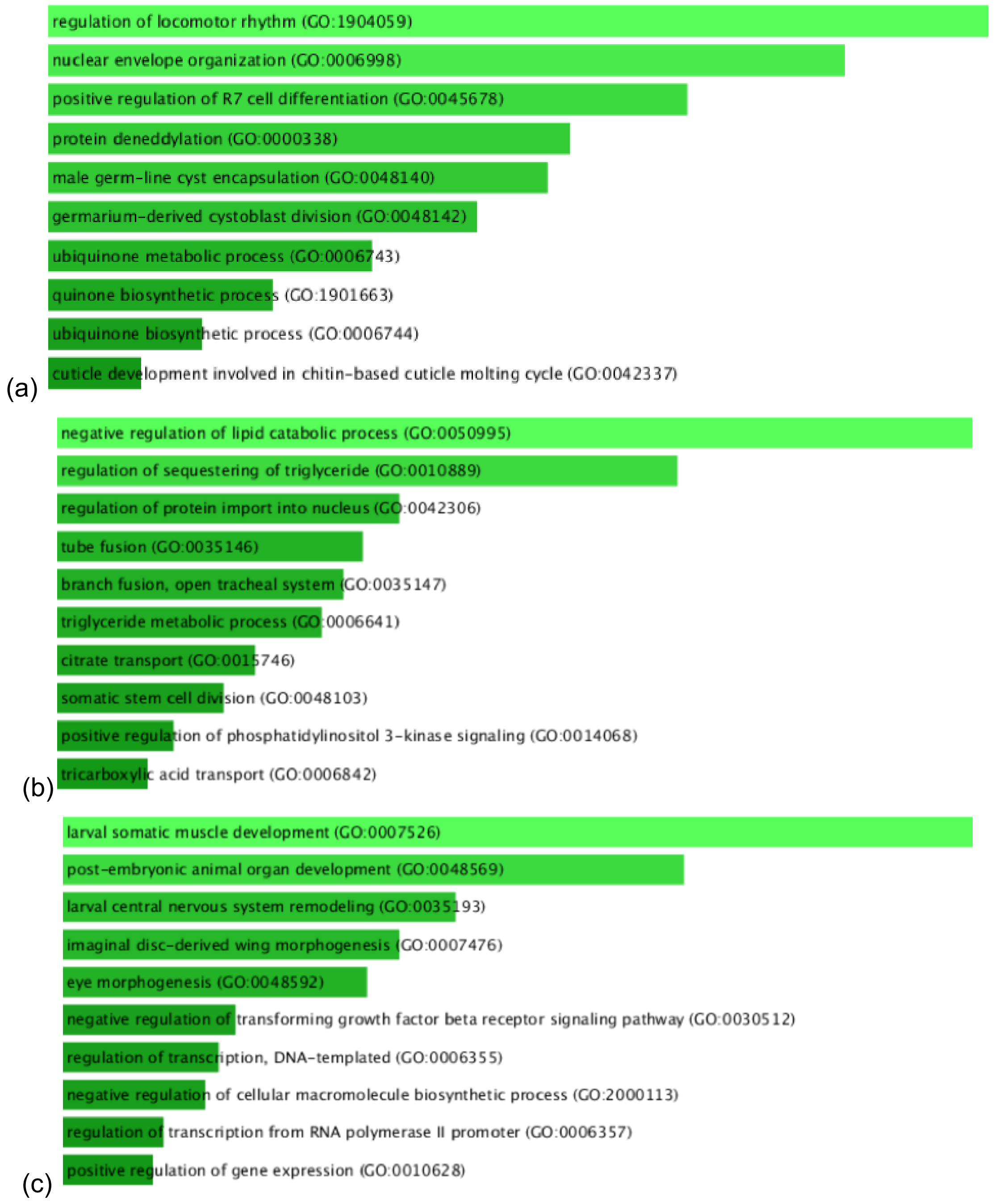


Figure 5: (a) Distribution of openness scores from the region within the gene body (b) Distribution of openness scores from within the promoter region of each gene (c) Distribution of openness scores combining the gene body and promoter regions of each gene (d) Volcano chart comparing the openness score of the gene body vs. when the promoter is included

4.1. GO Analysis

Gene ontology (GO) databases consist of GO terms that describe the functionality of a given gene. Each gene can have zero or more GO terms associated with it, and GO terms can also represent relationships to other genes or functionalities. We analyzed the time series data in the context of several gene ontology (GO) databases to learn about the functionality of the accessible genes and to see whether GO databases would be helpful in inferring accessibility. Overall, the results were not very informative and GOs are not necessarily complete for all species, so we did not consider GO term enrichment or relationships in our inference model.

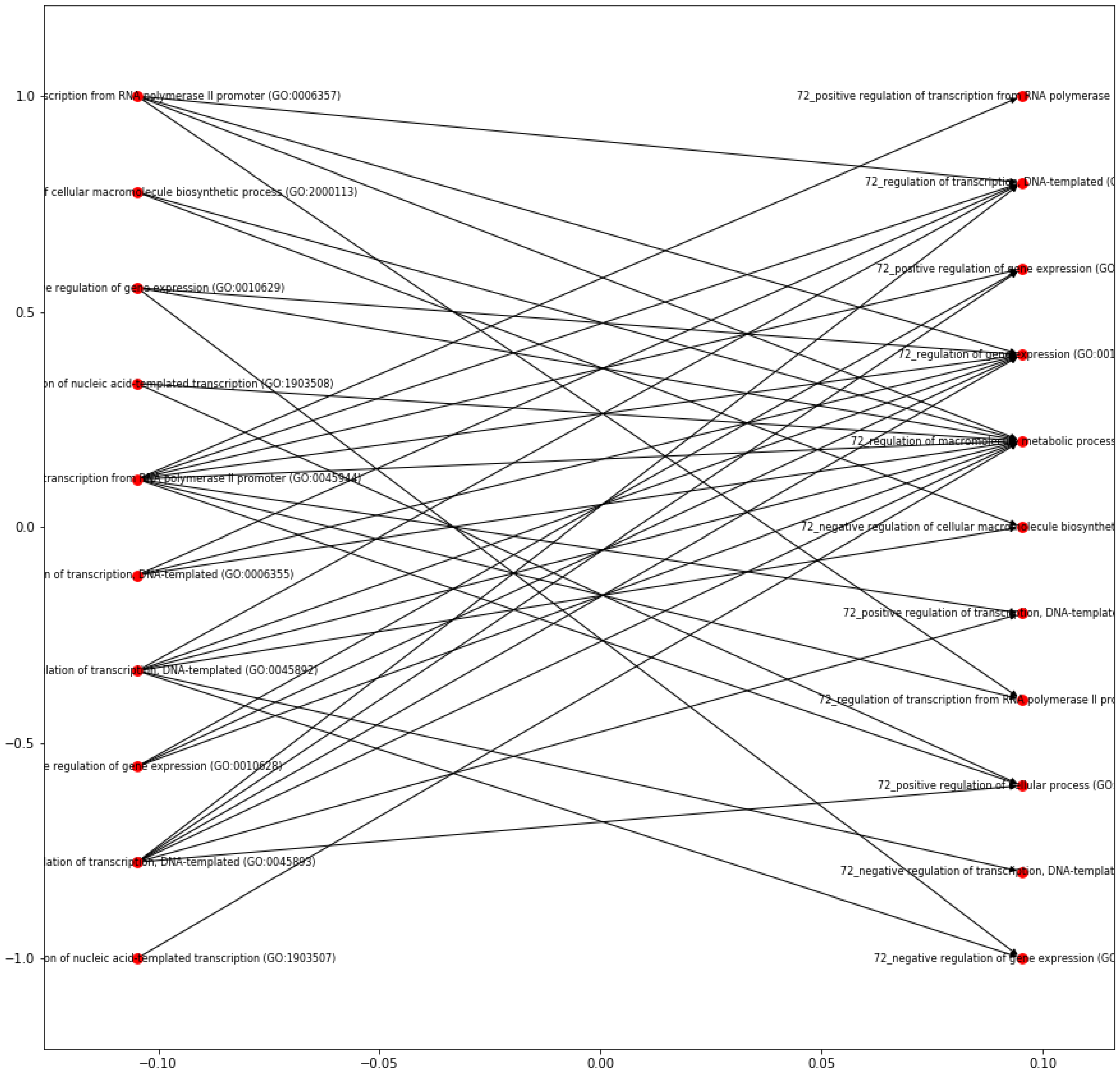
GO term enrichment is a measure of how many accessible genes are associated with a given GO term, compared to the total number of genes associated with the term. We used the online tool FlyEnrichr to measure the GO term enrichment of our data [6, 7]. Before analyzing the enrichment, we summarized the data from all three replications by excluding any gene where the median openness score was zero. We also partitioned the genes into three groups: only present at 48hrs, only present at 72, and present in both time steps[6]. The most enriched terms are shown in Figure 6.



[6] "GO NLP analysis for openness.ipynb" in the project repository

Figure 6: The most enriched GO terms among genes in (a) the 48hr time point only (b) the 72hr time point only (c) both time points. The length of the bar represents a composite score of the p-value and z-score of the deviation from the expected rank. The color of the bar represents the significance (lighter is more significant).

One type of relationship between GO terms is the “regulates” relationship. We investigated regulation relationships from GO terms present in the 48hr time point to those in the 72hr time point using the ‘regulates_closure’ attribute on terms retrieved from the AmiGO API [8-10]. We visualized the result as a bipartite graph (Figure 7), and learned that the terms about regulation are quite general, limiting the potential benefit of incorporating them in the inference model.



scription from RNA polymerase II promoter (GO:0006357)
f cellular macromolecule biosynthetic process (GO:2000113)
e regulation of gene expression (GO:0010629)
on of nucleic acid-templated transcription (GO:1903508)
transcription from RNA polymerase II promoter (GO:0045944)
n of transcription, DNA-templated (GO:0006355)
lation of transcription, DNA-templated (GO:0045892)
e regulation of gene expression (GO:0010628)
lation of transcription, DNA-templated (GO:0045893)
on of nucleic acid-templated transcription (GO:1903507)
72_positive regulation of transcription from RNA polymerase
72_regulation of transcription, DNA-templated (C
72_positive regulation of gene expression (GO
72_regulation of gene expression (GO:001
72_regulation of macromolecule metabolic process
72_negative regulation of cellular macromolecule biosynthet
72_positive regulation of transcription, DNA-templat
72_regulation of transcription from RNA polymerase II pro
72_positive regulation of cellular process (GO:
72_negative regulation of transcription, DNA-templat
72_negative regulation of gene expression (GC
1.0
0.5
0.0
−0.5
−1.0
−0.10
−0.05
0.00
0.05
0.10

Figure 7: Regulation relationships between GO terms present at the 48hr time point and the 72hr time point.

4.2. Promoter Similarity

In order to be expressed, a gene's promoter must be accessible and a transcription factor (TF) must bind to it. The ability of a TF and promoter to bind is also determined by the sequence of base pairs that make up the promoter. We investigated the similarity between promoter sequences in our data using two approaches: Ratcliff-Obershelp similarity and k-means clustering[7]. Both methods produce groupings of similar promoters, and we later used these as input to linear regression models, discussed in Section 4.2. Before applying the similarity measures, promoter sequences were pulled from the Drosophila melanogaster genome using the Ensembl API [11] along with a script to access promoter sequences in bulk[8].

Ratcliff-Obershelp similarity measures the amount of overlap between two sequences by repeatedly finding the longest common subsequence, noting its length, and eliminating it from both strings until one there are no more common subsequences. The total length of the common subsequences is then divided by the total length of the two strings to produce a score in [0, 1]. In our experiment, we computed the Ratcliff-Obershelp similarity between the promoters of each pair of genes and considered the genes to be in the same cluster if the similarity score was greater than a fixed threshold. Table 2 shows the distribution of genes among clusters for a threshold of 0.5. We tried several thresholds and found that too high a threshold caused each gene to be in its own cluster, and too low a threshold caused all genes to be in a single cluster.

Table 2: Distribution of genes among clusters generated by Ratcliff-Obershelp similarity

Number of clusters	**Nodes per cluster**
1	522
1	4
4	2
73	1

To use k-means clustering, we converted the string promoter sequence into a one-hot representation. Each letter in the original sequence was converted to a four-element vector, where the position of the 1 value indicated the base being represented. K-means clustering distributed the promoters among groups much more evenly compared to Ratcliff-Obershelp similarity, which is to be expected. One benefit of using the k-means clustering method is that it produces a vector representing the center of each cluster. By converting the cluster center back to a string sequence, salient patterns within the cluster can be observed.

[7] "Promoter similarity.ipynb" in the project repository
[8] extract_seq.py in the project repository

4.3. Linear modeling experiments

To begin inferring the accessibility of genes in future time steps, we tested various configurations of simple linear regression models. Several non-linear models were tested as well.

The first model[9] simply took each gene's openness at 48hr and 72hrs as an (input, output) pair, and a visualization of the result is shown in Figure 8 below. We can see that the relationship between openness at the two time points is not linear, but it is positive. While the model has a small mean squared error (0.002), the maximum squared error is relatively high (0.512), indicating a poor fit.

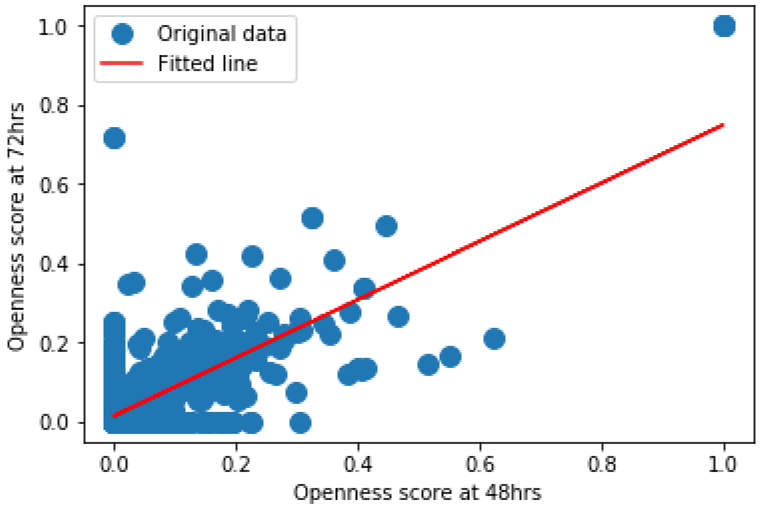


Figure 8: (48hr, 72hr) pairs of openness scores plotted as blue dots, and the fitted line produced by the first model

The second model focused on learning a function for a single gene. The input was a vector containing the openness scores for all of the genes in the replication at the 48hr time point, and the output value was the openness score for the particular gene and replication at the 72hr time point. This model was able to fit the data well, with near-zero sum, mean, and max squared error values. However, the model is limited because each replication only yields one data sample per time step and there is only one time step worth of data.

Two similar models were created using the clustering algorithms discussed in Section 4.2 with the aim of reducing the number of parameters learned by the model. The openness scores of the genes in each cluster were aggregated by averaging, and a vector of these was used as the input to the model. The output was still the openness score of the particular gene at the 72hr time point. Though there were still only three data samples total, the number of parameters to learn was reduced from the number of genes present to the number of clusters. Using the 79 clusters from Ratcliff-Obershelp similarity, the mean squared error was 0.001 and the maximum

[9] "Linear regression.ipynb" in the project repository

squared error was 0.002. Since the number of clusters can be specified when using k-means clustering, we plotted the performance in terms of log10(mean squared error) at each value of k from 1 to 607 (the number of promoters). Figure 9 shows that the mean squared error becomes very small as the number of clusters increases.

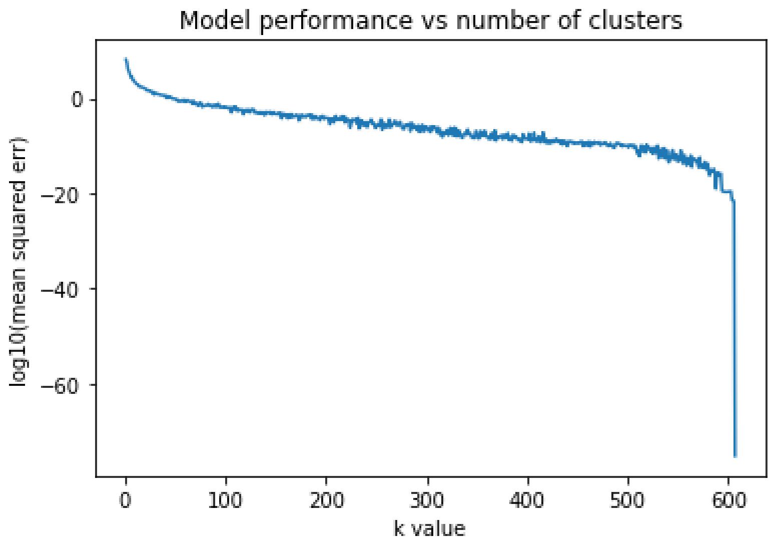


Figure 9: Performance of the linear model using k clusters

The next iterations[10] of our model (the linear promoter model) also reduce the number of parameters by condensing the openness values across the replication using the promoter sequence of each gene. The promoters (in one-hot vector representation) are formed into a 607x240 matrix P, and multiplied with the vector of openness scores for the replication, a(t) (length = 607) to produce a vector $P_{rep} = a(t)^T P$. Illustrated in Figure 10, P_{rep} is a 240 element vector, where each element is a measure of how many genes in the replication share a base in the same position of their promoter and how open they are. The one-hot vector for the promoter of the gene being predicted is appended to complete the model input, x.

[10] “Linear promoter model.ipynb” in the project repository

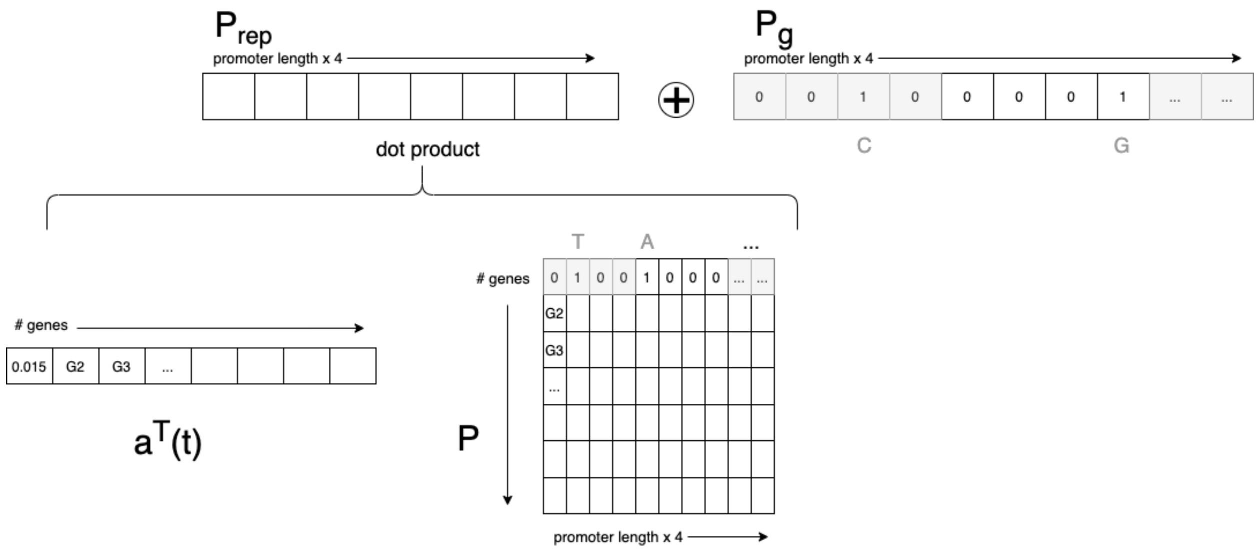


Figure 10: The input for the linear promoter model is constructed using the promoters and openness scores of genes present in the replication. P is a matrix of promoter vectors, a(t) is the openness score for each gene at time point t, P_g is the promoter vector for the specific gene, and ⊕ represents array concatenation. Examples of the one-hot vector representation of bases are also included.

Again, the goal of the model is to predict the openness score of the gene at 72hrs. In addition to reducing the number of parameters, this model also increases the number of data samples available. Because the gene's promoter sequence is appended to the input, there will be a unique input for every gene in each replication, creating a total of 607x3 = 1821 samples. Fitting the model to ⅔ of these, a low mean squared error was achieved (8.154e-8) but plotting the actual and predicted output values shows that the model is still quite inaccurate (Figure 11). Additionally, there is a large mean relative error of 11.444 and the model only predicts the correct direction of change from the openness of the gene at 48hrs 47.776% of the time.

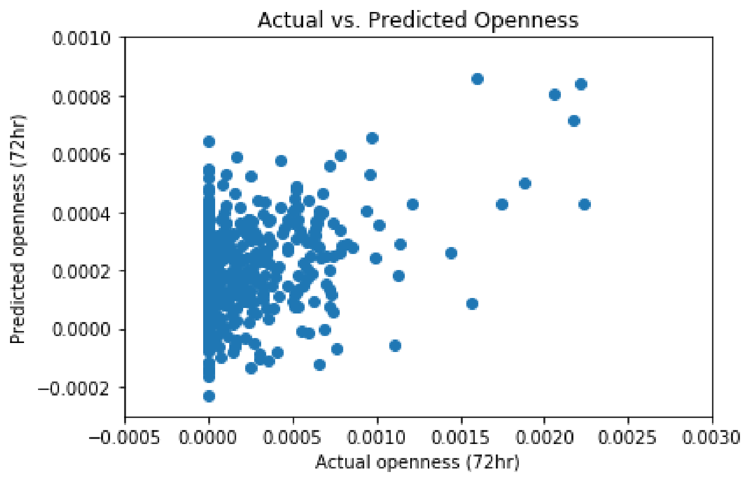


Figure 11: Actual vs. predicted openness using the linear promoter model

Several variations of the linear promoter model were implemented, but showed similar performance. These included adding a parameter to act as an intercept bias, only providing the promoter vector representation of the particular gene being predicted, weighting the promoter representation by the particular gene's openness, and using a spread function on the vector representation to allow for variation in the provided promoter sequence.

In one variation of the linear promoter model, the parameters (θ) were fit to output (y) values that were transformed using the inverse softplus function. The softplus function, plotted in Figure 12, is defined as $sp(x) = \ln(1 + e^x)$. It's useful in this application because openness scores cannot be negative, and all output values of softplus are positive. Additionally, the y value of the function grows quite slowly compared to x when y is less than 1. This allows a wider range of x values to produce a similar set of y values, which is useful because many of the openness scores are very close to zero, even though the corresponding inputs may differ a lot.

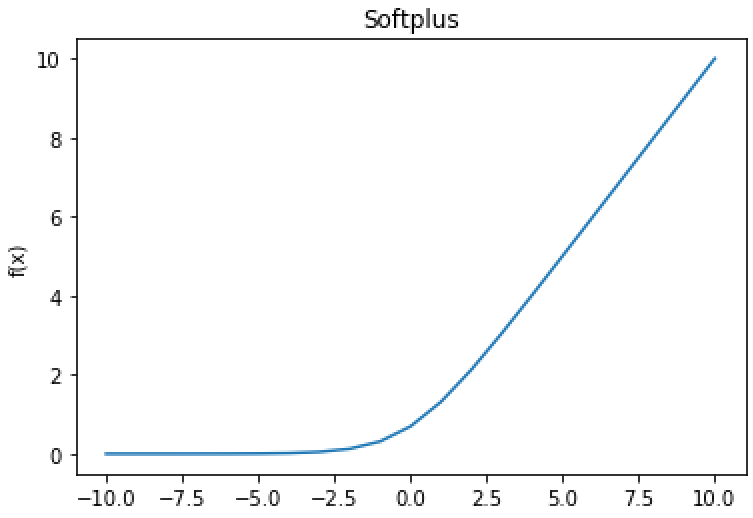


Figure 12: A plot of the softplus function

The target y value was further transformed by scaling (multiplying by 100) and adding a constant 4.5418e-5. P_{rep} was also scaled up by 10. The expected y value was scaled because the actual y values are extremely small and scaling them increases the distance between them, making it easier to achieve the desired output by taking the dot product with θ. The constant value was added so that the minimum $x^T\theta$ value needed is -10 (sp^{-1}(4.5418e-5) = -10), since the inverse softplus value of 0 is negative infinity, which is not feasible to obtain from $x^T\theta$. P_{rep} was scaled up because these values end up being quite small as well, but not as small as the y values because P_{rep} was computed by summing many openness scores. θ was fit to the transformed y values, so in order to recover an estimated y value, opposite transformations were applied after computing $x^T\theta$, as summarized in Equations 2 and 3.

$$y_{expected} = sp^{-1}\left(\left(y_{actual} * 100\right) + 4.5416e^{-}5\right) \Leftarrow fit\, x^{T}\theta (2)$$

$$y_{estimated} = \frac{sp(x^T\theta) - 4.541e^{-}5}{100} \quad (3)$$

Equations 2 and 3 summarize the softplus linear promoter model. θ is fit to the set of $y_{expected}$, and then used to produce $y_{estimated}$ by undoing the transformations on y_{actual} to compute $y_{expected}$.

This model resulted in a slightly better accuracy in terms of the direction of change in openness between the two time steps, of 52%. It also avoided the problem of producing negative outputs. However, there was still a high maximum relative error of 24.143, and the mean relative error was 1.65. Figure 13 shows the performance of the model in terms of the predicted and actual value of each test sample.

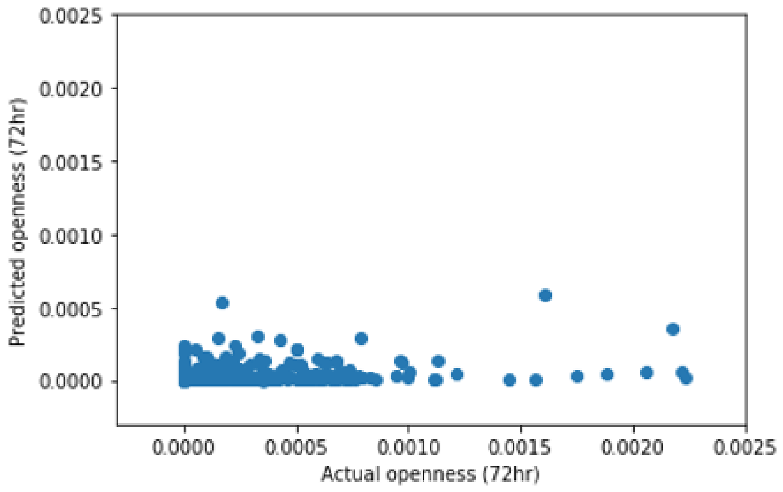


Figure 13: Performance of the softplus linear promoter model

4.4. Non-linear modeling experiments

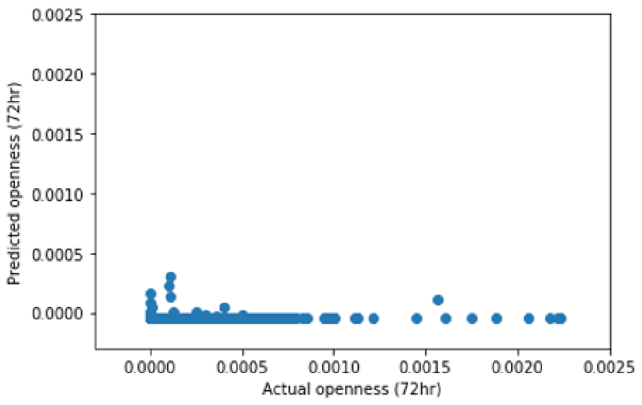
A support vector classifier (SVC) and a multilayer perceptron (MLP) model were implemented to predict the direction of change in openness over the time step. The SVC attained 72.488% accuracy by predicting a decrease in openness every time. The MLP was configured with varying numbers and sizes of hidden layers, and achieved accuracies around 50-65%.

Gradient descent was implemented as an extension of the softplus linear promoter model, because softplus is a nonlinear function. The cost function was the sum of squared errors relative to the actual y value. To replicate the effects of the transformations on the target output of the softplus linear promoter model, transformations were applied while extracting predictions from the model. Since the transformations are now applied to the estimated output and not the target output, they are the opposite of the transformations done to create the expected output for the linear model (i.e. the output of the softplus function is scaled down by dividing by 100 and we subtract 4.5418e-5). This differs slightly from the linear promoter model because the softplus output is scaled before subtraction, but I found that this way yields better results. The final cost function is shown in Equation 4.

$$\sum \frac{\left(y_{actual} - \left(\frac{sp(x^T\theta)}{100} - 4.5418e^{-}5\right)\right)^2}{max(y_{actual}, 1e^{-}5)} \quad (4)$$

Several cost functions were tried, varying in the transformations used and their placement in the cost function. When $x^T\theta$ was scaled by 10, the gradient descent did not function because it caused softplus to output infinity, and when $sp(x^T\theta)$ was scaled by 10, the gradient became zero and theta never changed. This could be because the numerator became very large compared to the denominator, causing the cost to mostly be determined by the denominator, reducing the impact of theta and thus turning the gradient to 0.

The gradient descent was tested using two different initialization strategies for θ. In one initialization strategy, theta was initialized using random numbers. After running 1000 iterations of gradient descent, the end cost was 25.350. The direction accuracy was better than any of the linear models at 72.817%. This model also had lower mean and maximum relative errors (3.132 and 16.627, respectively), but had a higher minimum relative error than the linear models (0.221, compared to <0.1). Figure 14 shows the performance on the test set and the training cost over the training period.

(a)


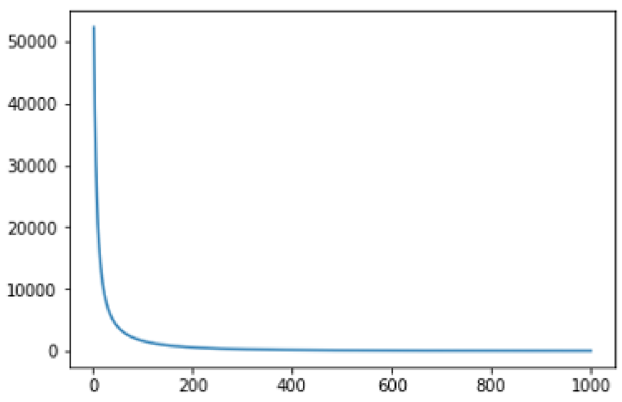
(b)


Figure 14: (a) Performance of the randomly initialized gradient descent model (b) Cost curve of this model

Based on the elbow point in the plot of the cost function, I also tried training for only 200 iterations in order to prevent overfitting. It generally did not perform better, but did create more varied outputs. The direction accuracy was 71.499%, the mean relative error was 14.428, and the maximum relative error was 2622.579. Figure 15 shows the performance on the test set.

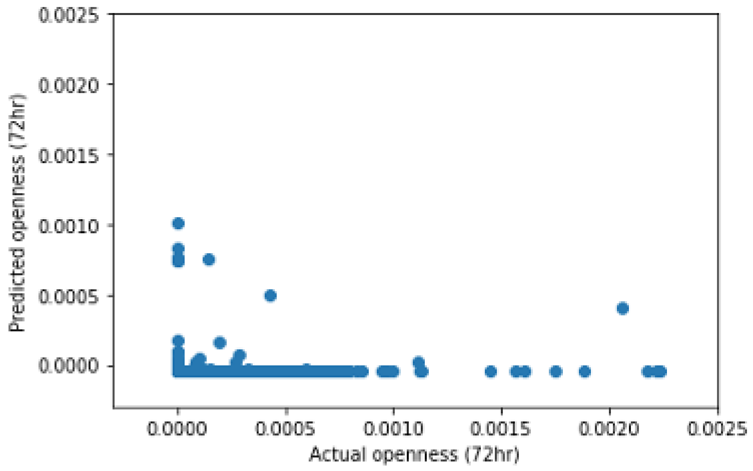


Figure 15: Performance of the randomly initialized gradient descent model after 200 training iterations.

In the second θ initialization strategy, I used the θ produced from the softplus linear promoter model, hoping that gradient descent would improve upon it. However, θ did not change very much which may mean that it already represented a local minima on the cost function, or that the training was not effective because the training set had already been used when fitting θ with the linear model. Interestingly, the performance graph for this model and the linear model look very similar, but the performance of this one is much better. The direction accuracy was 70.675%, mean relative error was 2.247, and maximum relative error was 19.519. The performance and cost function over training are shown in Figure 16.

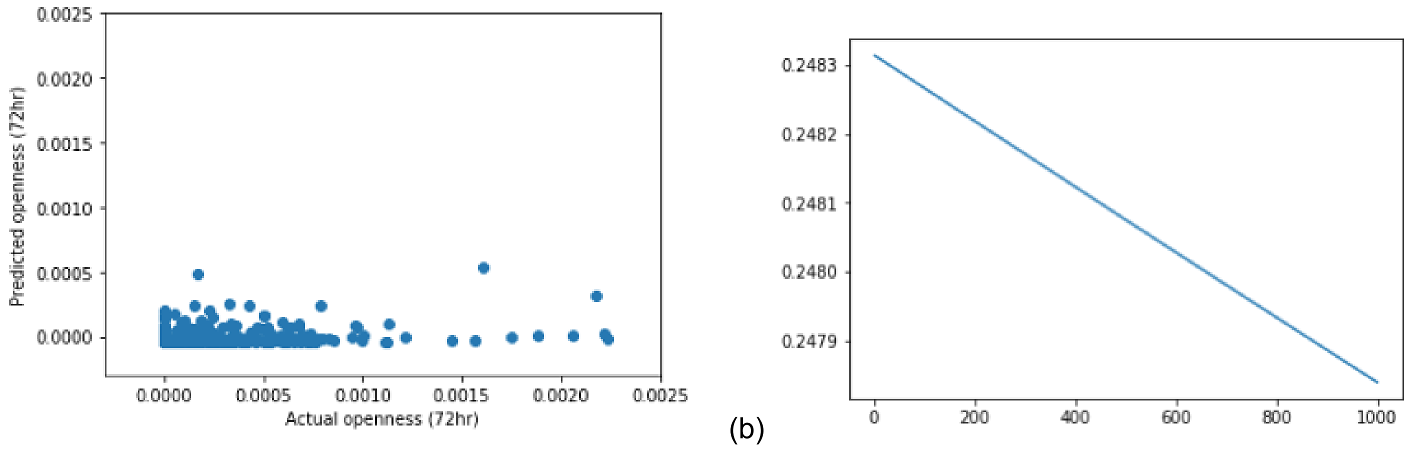


Figure 16: (a) Performance of the pre-trained gradient descent model (b) Cost curve of this model

5. Discussion and Future Work

Though the performance of the proposed models show that improvements are needed, the work done so far uncovers new avenues of investigation. First, related work has shown that clustering genes by their biological activities and features can help to infer chromatin accessibility and GRNs. We briefly explored clustering the genes by promoter similarity, but it was not incorporated into later models. A more careful consideration of cluster creation methods could be useful, since genes that are regulated by the same proteins require similar binding structures/sequences in their promoters and may behave similarly. Such recurring sequences are called motifs, and are documented in publicly available databases.

Incorporating more biological information into the model can lead to detecting patterns that underlie the behavior of the genome in terms of chromatin accessibility. For instance, gene expression data can be used to validate which genes were actually open in the previous time step, since a gene's promoter must be accessible for it to be expressed. This is particularly useful, since ATAC-seq data is noisy. The RNA-seq data can be used to reduce noise in the ATAC-seq data and to give more context to the model.

Addressing the noisiness and sparseness of ATAC-seq data could also lead to model improvements. In the non-linear models, it was common for the output to be zero when the actual openness score was non-zero. This makes sense since the openness score is zero much of the time, but it may prevent the model from learning how to predict non-zero openness scores. Strategies for modeling sparse data should be investigated, which might include additional preprocessing steps or using different types of models. It may also be helpful to first create a model that predicts whether the openness in the next time step will be zero or not, and have a separate model that predicts a non-zero openness score.

Since gradient descent produced the best results so far, this approach should be a focus of future work. The items that were previously discussed can be used to create a new model that is trained using gradient descent. Testing more variations of the cost function could improve the performance further, and any constant numbers in the cost function could be optimized, either by trial and error or algorithmically.

6. Conclusion

Over the course of this work, we have reviewed existing GRN and gene accessibility inference models and used the openness score metric proposed in [2] to process ATAC-seq data into per-gene openness scores at each time step for D. melanogaster data. We analyzed the processed data to learn about the distribution and dynamics of the openness score over a single time step, and proposed several models to predict openness from one time step to another. The results show that while the sparsity of the data and the complexity of genomic activity present challenges in modeling gene accessibility, avenues for further investigation and development are abundant.

References

1. Emmert-Streib, F., Dehmer, M., & Haibe-Kains, B. (2014). Gene regulatory networks and their applications: understanding biological and medical problems in terms of networks. Frontiers in cell and developmental biology, 2, 38.
2. Duren, Z., Chen, X., Xin, J., Wang, Y., & Wong, W. H. (2020). Time course regulatory analysis based on paired expression and chromatin accessibility data. Genome research, 30(4), 622-634.

3. Jansen, C., Ramirez, R. N., El-Ali, N. C., Gomez-Cabrero, D., Tegner, J., Merkenschlager, M., Conesa, A., & Mortazavi, A. (2019). Building gene regulatory networks from scATAC-seq and scRNA-seq using Linked Self Organizing Maps. PLoS computational biology, 15(11), e1006555.
4. Kohonen, T. & Honkela, T. (2007). Kohonen Network. Scholarpedia. 2 (1): 1568. doi: 10.4249/scholarpedia.1568
5. Zhou, W., Ji, Z. *et al*. (2017). Genome-wide prediction of DNase I hypersensitivity using gene expression. Nature communications, 8(1), 1-17. doi:10.1038/s41467-017-01188-x
6. Chen, E. Y., Tan, C. M., Kou, Y., Duan, Q., Wang, Z., Meirelles, G. V., Clark, N. R. & Ma'ayan, A. (2013). Enrichr: interactive and collaborative HTML5 gene list enrichment analysis tool. BMC bioinformatics, 14(1), 128. doi:10.1186/1471-2105-14-128
7. Kuleshov, M. V., Jones, M. R., Rouillard, A. D., Fernandez, N. F., Duan, Q., Wang, Z., Koplev S., Jenkins S. L., Jagodnik K. M., Lachmann A., McDermott, M. G., Monteiro C. D., Gundersen G. W. & Ma'ayan A. (2016). Enrichr: a comprehensive gene set enrichment analysis web server 2016 update. Nucleic acids research, 44(W1), W90-W97. doi:10.1093/nar/gkw377
8. Ashburner, M. *et al.* (2000). Gene ontology: tool for the unification of biology. Nature genetics, 25(1), 25-29. doi:10.1038/75556
9. Gene Ontology Consortium. (2019). The gene ontology resource: 20 years and still GOing strong. Nucleic acids research, 47(D1), D330-D338. doi:10.1093/nar/gky1055
10. Carbon, S., Ireland, A., Mungall, C. J., Shu, S., Marshall, B., Lewis, S., AmiGO Hub & Web Presence Working Group. (2009). AmiGO: online access to ontology and annotation data. Bioinformatics, 25(2), 288-289. doi:10.1093/bioinformatics/btn615
11. Yates, A. D. *et al*. (2020). Ensembl 2020. Nucleic acids research, 48(D1), D682-D688. doi:10.1093/nar/gkz966