SeqNet: an R package for simulating RNA-sequence datasets from regulatory networks

Tyler Grimes

***Abstract*—With the advent of next-generation sequencing, researchers are now tackling more complex questions in systems biology. A differential network analysis between two groups looks for differences in gene-gene interactions; these interactions are a consequence of an underlying regulatory network. Various methods have been developed to learn these networks from gene expression data. However, without a gold standard dataset, comparisons of different methods can be subjective. SeqNet is an R package that provides researchers a way to simulate RNA-seq data from a predefined network. This allows for a more objective comparison between different methods. The package is available on GitHub and can be downloaded with the R command: 'devtools::install_github("tgrimes/SeqNet")'.**

I. Introduction

WHEN analyzing gene expression data, a common objective is to look for differentially expressed genes between groups. This analysis identifies genes that may be responsible for the difference in phenotype between the groups. With advances in sequencing technologies, in particular RNA-Sequencing, researchers can now explore more complex questions. An extension of looking at individual gene expression is to look at how expression among all genes vary together. This leads to the development of co-expression networks, which are used to infer the underlying regulatory network of the genes. Now when comparing groups, the goal is shifted to a differential network analysis; we look for differences in the co-expression networks, such as differentially connected genes or different modular structures.

When developing new methodology for differential network analysis, it is desirable to compare the performance of different methods. This can be done, to some extent, using real datasets. For instance, if the differentially connected genes are collected into a gene set, which is then used for classification, prediction of survival time, or some other estimation of a phenotype of interest, then various methods can be compared based on the predictive performance of the gene set they each produce. Alternatively, if there is no response variable to use, methods can be justified by arguing that the gene set they produce is somehow consistent with published results, but comparisons among methods are difficult in this case. The main downside to using real datasets is that we can't validate whether the gene sets actually contain differentially connected genes because the truth of the underlying networks is unknown.

Another way to compare methods is through a simulation study. In simulations, the underlying networks are chosen from the outset, which means the performance of different methods can be evaluated based on the underlying truth. Simulations also enable greater control of the settings in which comparisons are made - the performance of a method will vary in different situations, and simulation studies can help identify the particular settings in which a method is likely to perform well. The downside to simulations is that we don't know exactly how to simulate the biological process of gene expression, because the process itself is not fully understood. Hence simulated data may not be characteristic of real data. This means there is no guarantee that good performance in the simulation will correspond to good performance in the real world.

A variety of approaches to generate gene expression data based have been developed. Generators are typically either based on interaction kinetics, which use differential equation models [1]–[4], or on probability models, which can be parametric or nonparametric [5], [6]. In methods papers, probabilistic-based simulations are commonly done using a Gaussian model [7]–[12]. However, to simulate raw RNA-seq count data, a discrete probability model should be used instead. By simulating raw counts, a more thorough comparison of various methods can be made, including the effect different transformations of the RNA-seq data have on each method's performance.

In this paper, we propose a method based on the negative-binomial distribution. This distribution is a generalization of the Poisson distribution that allows for over-dispersion, which is characteristic of RNA-seq data. The method allows the user to specify an underlying association network and generates raw RNA-seq expression data from this network.

II. Methods

The RNA-seq counts are assumed to have a negative-binomial distribution. The mean expression level for each gene will be a function of the assumed regulatory network. The SeqNet package contains tools for creating the underlying network and generating samples based on this network.

Negative-binomial distribution

Consider $X \sim NB(r, p)$, with $r > 0$ and $p \in (0, 1)$, having the probability mass function

$$P(X = x) = \binom{x + r - 1}{r} (1 - p)^r p^x,$$

for $x \in \{0, 1, 2, ...\}$. Samples can be drawn from this distribution using the gamma-poisson mixture model. That is, if $\theta \sim Gamma(r, \beta)$ and $X \sim Poisson(\theta)$, then the marginal distribution of X will be negative-binomial with

$p = \beta/(1+\beta)$. The mean, variance, and excess kurtosis of X expressed in terms of r and β are,

$$E(X) = \frac{pr}{1-p} = r\beta$$
$$Var(X) = \frac{pr}{(1-p)^2} = r\beta(1+\beta)$$
$$Kurt(X) = \frac{6}{r} + \frac{(1-p)^2}{pr} = \frac{6}{r} + \frac{1}{r\beta(1+\beta)}$$

Note, as $\beta \to 0$, the variance and mean become equivalent. That is, the distribution is asymptotically Poisson as β goes to zero.

The simulation will make use of three parameters:

μ : The desired mean of X

γ : The over-dispersion parameter

k : The tail-behavior parameter

The mean $\mu > 0$ is the desired mean expression level for the gene, the over-dispersion parameter $\gamma > 0$ determines the amount of variation in the expression, and the tail-behavior parameter $k \in [1, 2]$ controls the amount of kurtosis in the distribution of expression counts. In the next section, we show how the gamma-poisson mixture is parameterized using these values.

Gamma-Poisson mixture: Consider $\theta \sim Gamma(r, \beta)$ with the parameterization,

$$r = \mu^{2-k}/\gamma$$
$$\beta = \mu^{k-1}\gamma$$

where $k \in [1, 2]$, $\mu > 0$, and $\gamma > 0$.

In this setup, the variance and excess kurtosis of the marginal distribution of $X \sim Poisson(\theta)$ for fixed μ, k, and γ are

$$Var(X) = \mu + \mu^k\gamma$$
$$Kurt(X) = \frac{6\gamma}{\mu^{2-k}} + \frac{1}{\mu + \mu^k\gamma}$$

Note that for fixed μ and k, as $\gamma \to 0$, the variance approaches μ and the kurtosis approaches $1/\mu$, i.e. the distribution becomes a Poisson.

As γ increases, the variance increases by $\mu^k\gamma$, and for large γ, the kurtosis is approximately $6\gamma/\mu^{2-k}$. The main property of k is that it determines the amount of kurtosis; when $k = 1$, the kurtosis will tend to be small due to the factor of $1/\mu$. When $k = 2$, the kurtosis will depend only on the value of γ chosen. In general, for fixed γ, larger values of k will lead to larger kurtosis. We find that $\gamma = 1$ and $k = 1.5$ provides a good balance in general, and these are the default values in the R package.

Regulatory network

The SeqNet package provides a framework for representing gene regulatory networks. When developing this framework, the goal was to provide a flexible way to build the network while also imposing a meaningful structure. It is known that genes can be involved in multiple pathways, so a simple adjacency matrix is not sufficient for fully specifying a network, particularly if each pathway is to be treated independently. Therefore, three different components (also referred to as 'structures') are considered that act as building blocks for the network. These include cliques, hubs, and modules. This framework is described in the following sections.

Cliques, hubs, and modules: The clique, hub, and module structures are motivated by biological mechanism of gene expression. Let $\mathcal{G} = \{1, ..., p\}$ denote the set of genes in the network.

Let $\mathcal{C} \subset \mathcal{G}$ be a clique in the network. Cliques are completely connected subgraphs. The expression of genes in $\mathcal{C}$ are considered to be regulated by some unobserved latent factor. We assume that the strength of the association within a clique are all equal; that is, the latent factor has an equal effect (in magnitude) on all genes in the clique. However, a subset of genes, $\mathcal{C}^- \subset \mathcal{C}$, can have a negative association with the latent factor. The remaining genes, $\mathcal{C}^+ = \mathcal{C} \backslash \mathcal{C}^-$, are positively associated with the latent factor. At the gene level, this means that genes within the same partition ($\mathcal{C}^+$ or $\mathcal{C}^-$) are positively associated, while genes in different partitions are negatively associated.

Let $\mathcal{H} = (h_0, h_1, ..., h_d) \in \mathcal{G}^{d+1}$, with each h_i distinct, be a hub of degree $d < p$ in the network. By convention, the first entry, h_0, specifies the central gene in the hub, while the remaining entries, h_1, ..., h_d, are the genes connected to h_0. For this structure, we think of h_0 as a transcription factor that controls the expression of the other d genes. Here, each of the d connections are weighted independently, meaning h_0 will not necessarily have the same affect on all of its connected genes. This reflects the fact that a single transcription factor can affect different genes in different ways. This also means that the genes $h_1, ..., h_d$ will not inherit any associations with one another through h_0.

Let $\mathcal{M} \subset \mathcal{G}$ denote a module in the network. Modules are more general structures than cliques or hubs. Each module has a corresponding adjacency matrix, $A^{\mathcal{M}} \in \{0, 1\}^{|\mathcal{M}| \times |\mathcal{M}|}$, that specify the connections between genes within the module. Modules serve two main purposes. First, it is useful for representing known pathways, where each connection in the pathway is represented by an edge in the module. Second, modules are useful for quickly generating realistic networks. By default, modules are randomly connected using the Watts-Srogatz small-world model [13], which generates a scale-free graph that is characteristic of biological networks [14]. Each connection within a module will have independent weights, similar to the hub components.

Adjacency matrix: The overall network can be summarized by a matrix $A \in \{0, 1\}^{p \times p}$. The diagonal of A will contain all zeros. Consider two genes indexed by $i, j \in \mathcal{G}$, $i \neq j$. If the two genes are not connected in any of the cliques, hubs, or modules, then $A_{ij} = A_{ji} = 0$. Otherwise, the genes are connected in some structure, and we set $A_{ij} = A_{ji} = 1$.

As mentioned previously, the adjacency matrix does not fully specify the network; overlapping connections from different structures are summarized as a single connection.

R: creating a network: A network is constructed by specifying the cliques, hubs, and modules through the 'create_network()' function. Each individual clique, hub, or module in the network is referred to as an instance of that component. All components are optional (if there are no instances of any component, then the network will have no connections). Components can be specified in the following three ways. Examples of R code are shown. Each example constructs a network containing 5 cliques and 2 modules.

(i) Declare the number of instances of each component through 'n_cliques', 'n_hubs', and 'n_modules'.

```
> create_network(n_cliques = 5, n_modules = 2)
```

An appropriate size of each instance is randomly generated if none are provided by...

(ii) declaring the size(s) of each component through 'clique_size', 'hub_size', and 'module_size'. These arguments can either be single integers or vectors. In the latter case, suppose (for instance) 'module_size' is a vector, then 'n_module' can be NULL, and the length of the vector 'module_size' will determine the number of module instances to generate.

```
> create_network(clique_size = c(4, 4, 5, 5, 10),
                 module_size = c(40, 100))
```

Genes will be randomly selected for inclusion into each instance.

(iii) A list of vectors through 'cliques', 'hubs', and 'modules', where the length of the list determines the number of instances of the component, and each vector in the list specifies a set of genes for each instance.

```
> create_network(cliques = list(1:4, 5:9, 10:14,
                                15:19, 20:29),
                 module = list(30:79, 80:179))
```

If a list is provided, then values from (i) and (ii) will be ignored.

Generator algorithm

The following algorithm describes how a sample is generated from the underlying network. This is split into two procedures. First, weights are generated for each component in the network. Then, gene expressions are generated using these weights. These two steps are repeated n times to generate a sample of size n.

Algorithm for generating connection weights: Let $\mathcal{G} = \{1, ..., p\}$ denote the set of genes in the network.

1. Let $\mathcal{C}_k \subset \mathcal{G}$, $k \in \{1, ..., n\_cliques\}$, denote the kth clique in the network. For each clique, generate a weight $W^{\mathcal{C}_k} \sim N(0, 1)$.
2. Let $\mathcal{H}_k = (h_0^k, ..., h_{d_k}^k)$, $k \in \{1, ..., n\_hubs\}$, denote the ith hub in the network. For each hub, generate a set of weights $w_j^{\mathcal{H}_k} \sim N(0, 1)$ for j from $1, ..., d_k$.
3. Let $\mathcal{M}_i \subset \mathcal{G}$, $i \in \{1, ..., n\_modules\}$, denote the ith module with corresponding adjacency matrix $A^{\mathcal{M}_i} \subset \{0, 1\}^{|\mathcal{M}_i| \times |\mathcal{M}_i|}$. For each module, generate weights $W^{\mathcal{M}_i} \in \mathbb{R}^{|\mathcal{M}_i| \times |\mathcal{M}_i|}$ with $W_{jk}^{\mathcal{M}_i} \sim N(0, 1)$ if $A_{jk}^{\mathcal{M}_i} = 1$, and $W_{jk}^{\mathcal{M}_i} = 0$ otherwise.

Algorithm for generating samples: With weights generated for each network component, a sample of expression counts can now be generated. Let μ_i be the baseline mean for gene i, γ be the over-dispersion parameter, and k be the tail-behavior parameter.

1. Let $i = 1$.
2. Add the weights of each connection (if any) to gene i in each clique, hub, and module instance; call this sum w_i. If there are no connections to gene i, then $w_i = 0$.
3. Set $\tilde{\mu}_i = \mu_i \exp(w_i)$. This adjusts the mean expression of gene i based on the weights of the network connections for this sample.
4. Generate $\theta \sim Gamma(\tilde{\mu}_i^{2-k}/\gamma, \tilde{\mu}^{k-1}\gamma)$.
5. Generate $x_i \sim Poisson(\theta)$.
6. Repeat steps 2 - 5 for $i = 1, ..., p$.

R: generating RNA-seq dataset: After a network is created using the 'create_network()' function, samples are generated from this network using the 'gen_gamma_poisson()' function.

```
> n ← 100 # Desired sample size.
> p ← 1000 # Number of genes.
> network ← create_network(p, modules = list(1:500))
> x ← gen_gamm_poisson(n, network)$x
> plot(network, as_subgraph = TRUE, main = "Network")
```

This code generates a sample of size $n = 100$. The underlying network consists of a random, scale-free network of 500 genes. The remaining 500 genes have no connections in the network. A visual plot of the network is created using the 'plot()' function; setting 'as_subgraph = TRUE' omits the 500 genes without connections from the plot.

III. Results and Discussion

Preliminary results of the generator suggest that the samples generated have the associations defined by the underlying network. This can be shown by simulating large samples ($n \approx 1000$), feeding the data into any gene co-expression algorithm, and showing that the inferred network closely matches the true network. If the true network can be recovered from the data, this is evidence that the data does, in fact, reflect the underlying network. So far, the generator has been verified using correlations, WGCNA [7], and cPLS [5] - each method was able to successfully recover the network with large samples.

Further validation still needs to be performed. The distribution of gene expression counts will be compared to a

reference distribution of RNA-seq counts. This process will also help determine the best default values for γ and k, the over-dispersion and tail-behavior parameters, respectively.

Currently, the object generated by 'create_network()' is of class 'network'. The 'plot()' function has been implemented for this class, but other functions still need to be added such as 'print()' and 'show()'.

Functions also need to be included to add, modify, or delete components in the network. It will be useful to provide a function that creates a network (or module component) based on a given adjacency matrix; this will allow users to easily represent known regulatory networks as a module in the network.

Finally, a vignette will be added to the package once some of the other implementation details are worked out.

IV. Availability

The SeqNet R package is currently available on GitHub at: "https://github.com/tgrimes/SeqNet".

V. Bibliography

References

[1] P. Mendes, W. Sha, and K. Ye, "Artificial gene networks for objective comparison of analysis algorithms," *Bioinformatics*, vol. 19, no. suppl_2, pp. ii122–ii129, 2003.

[2] H. Hache, C. Wierling, H. Lehrach, and R. Herwig, "Genge: systematic generation of gene regulatory networks," *Bioinformatics*, vol. 25, no. 9, pp. 1205–1207, 2009.

[3] B. Di Camillo, G. Toffolo, and C. Cobelli, "A gene network simulator to assess reverse engineering algorithms," *Annals of the New York Academy of Sciences*, vol. 1158, no. 1, pp. 125–142, 2009.

[4] T. Van den Bulcke, K. Van Leemput, B. Naudts, P. van Remortel, H. Ma, A. Verschoren, B. De Moor, and K. Marchal, "Syntren: a generator of synthetic gene expression data for design and analysis of structure learning algorithms," *BMC bioinformatics*, vol. 7, no. 1, p. 43, 2006.

[5] M. Pesonen, J. Nevalainen, S. Potter, S. Datta, and S. Datta, "A combined pls and negative binomial regression model for inferring association networks from next-generation sequencing count data," 2016, under Review.

[6] S. Benidt and D. Nettleton, "Simseq: a nonparametric approach to simulation of rna-sequence datasets," *Bioinformatics*, vol. 31, no. 13, pp. 2131–2140, 2015.

[7] P. Langfelder and S. Horvath, "Wgcna: an r package for weighted correlation network analysis," *BMC bioinformatics*, vol. 9, no. 1, p. 559, 2008.

[8] R. Gill, S. Datta, and S. Datta, "A statistical framework for differential network analysis from microarray data," *BMC Bioinformatics*, vol. 11, p. 95, 2010.

[9] Y. Rahmatallah, F. Emmert-Streib, and G. Glazko, "Gene sets net correlations analysis (gsnca): a multivariate differential coexpression test for gene sets," *Bioinformatics*, vol. 30, no. 3, pp. 360–368, 2013.

[10] M. J. Ha, V. Baladandayuthapani, and K.-A. Do, "Dingo: differential network analysis in genomics," *Bioinformatics*, vol. 31, no. 21, pp. 3413–3420, 2015.

[11] C. Siska and K. Kechris, "Differential correlation for sequencing data," *BMC research notes*, vol. 10, no. 1, p. 54, 2017.

[12] Z. Wang, H. Fang, N. L.-S. Tang, and M. Deng, "Vcnet: vector-based gene co-expression network construction and its application to rna-seq data," *Bioinformatics*, p. btx131, 2017.

[13] D. J. Watts and S. H. Strogatz, "Collective dynamics of small-worldnetworks," *nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[14] A.-L. Barabasi and Z. N. Oltvai, "Network biology: understanding the cell's functional organization," *Nature reviews genetics*, vol. 5, no. 2, pp. 101–113, 2004.