Adjustment of spurious correlations in co-expression measurements from RNA-Seq: Supplementary Material 1

Ping-Han Hsieh, Camila Lopes-Ramos, Geir Kjetil Sandve, Kimberly Glass, and Marieke Lydia Kuijjer

March 25, 2021

1 Quantile Normalization

Given a read count data $\mathbf{X}$ and corresponding ranks for each value in the read count matrix $\mathbf{R}$ (with average ranking):

$$\mathbf{X} = [x_{ij}] \in \mathbb{N}_0^{p \times n},\ \mathbf{R} = [r_{ij}] \in \mathbb{N}_0^{p \times n}$$

, where p is the number of genes and n is the number of samples. Quantile normalization starts from sorting the read counts in $\mathbf{X}$ for each sample (column), with the sorted read count denoted as $\mathbf{S}$ (Figure 1, step 1). The procedure continues by computing the mean for each row of $\mathbf{S}$ (Figure 1, step 2 and 3):

$$RankMean(i) = \frac{1}{n}\sum_{j=1}^{n} s_{i,j}$$

Finally, the procedure substitutes the read count with the mean value of the corresponding quantile (Figure 1, step 6):

$$x_{i,j}^{(norm)} = f(R_{i,j})$$

, where f is a transformation that map rank to its mean value across group of samples:

$$f(r) = \begin{cases} RankMean(r), & r \text{ is integer} \\ 0.5 \times (RankMean(r-0.5) + RankMean(r+0.5)), & \text{otherwise} \end{cases}$$

One very important detail is the substitution of the normalized expression for genes with the same read count in each sample. Suppose that the i^{th} to $(i+k)^{th}$ entries in the j^{th} column vector of $\mathbf{S}$ have the same read count. Then, the ranks recorded in $\mathbf{R}$ corresponding to these entries will be $\mathbf{R}_{i:i+1,j} = (2i+k)/2$ (Figure 1 step 5). This is often referred to as the *average* ranking method when dealing with tied values. If the rank is not an integer, the normalized expression value will be $0.5 \times (RankMean(r-0.5) + RankMean(r+0.5))$.

For low expression values in the count matrix, it is common to have the same number of counts with many other entries due to the zero-inflated natural of RNA-Seq data. The ranks correspond to these values will be highly dependent on the number of entries with the same values in that particular sample. Take non-expressed genes for an example, when computing the mean value corresponds each rank, as long as there exists a non-zero entry across all the samples, the mean value will be a non-zero value. If we compute the number of non-expressed genes for each sample, denoted as a vector:

$$\mathbf{z} = \begin{bmatrix} z_1 & z_2 & \cdots & z_n \end{bmatrix}^T$$

The rank for these non-expressed genes in each sample will then be

$$\mathbf{r}^{(ne)} = \begin{bmatrix} (z_1+1)/2 & (z_2+1)/2 & (z_n+1)/2 \end{bmatrix}^T$$

When $z_j > 2 \times \min(\mathbf{z}) - 1$, the normalized expression value for all the non-expressed genes of $sample_j$, will be a non-zero value, depends on the number of non-expressed genes in the sample. The above mentioned problem is therefore more likely to happen in large-scale heterogeneous datasets where the library size is not uniformly distributed. For instance, the GTEx and ENCODE datasets that we analyzed in the our study (Figure 2). Eventually, this leads to false positive associations between genes that exclusively express in a small proportion of samples. Note that this can also happen for other lowly-expressed genes depends on how many genes share the same read count in one sample.

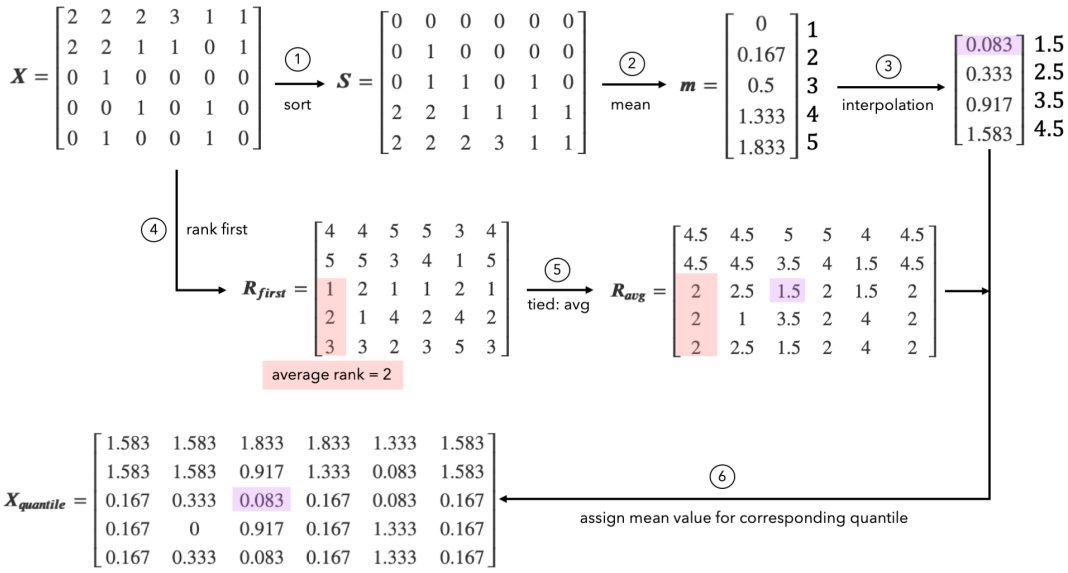

Figure 1: Quantile normalization.

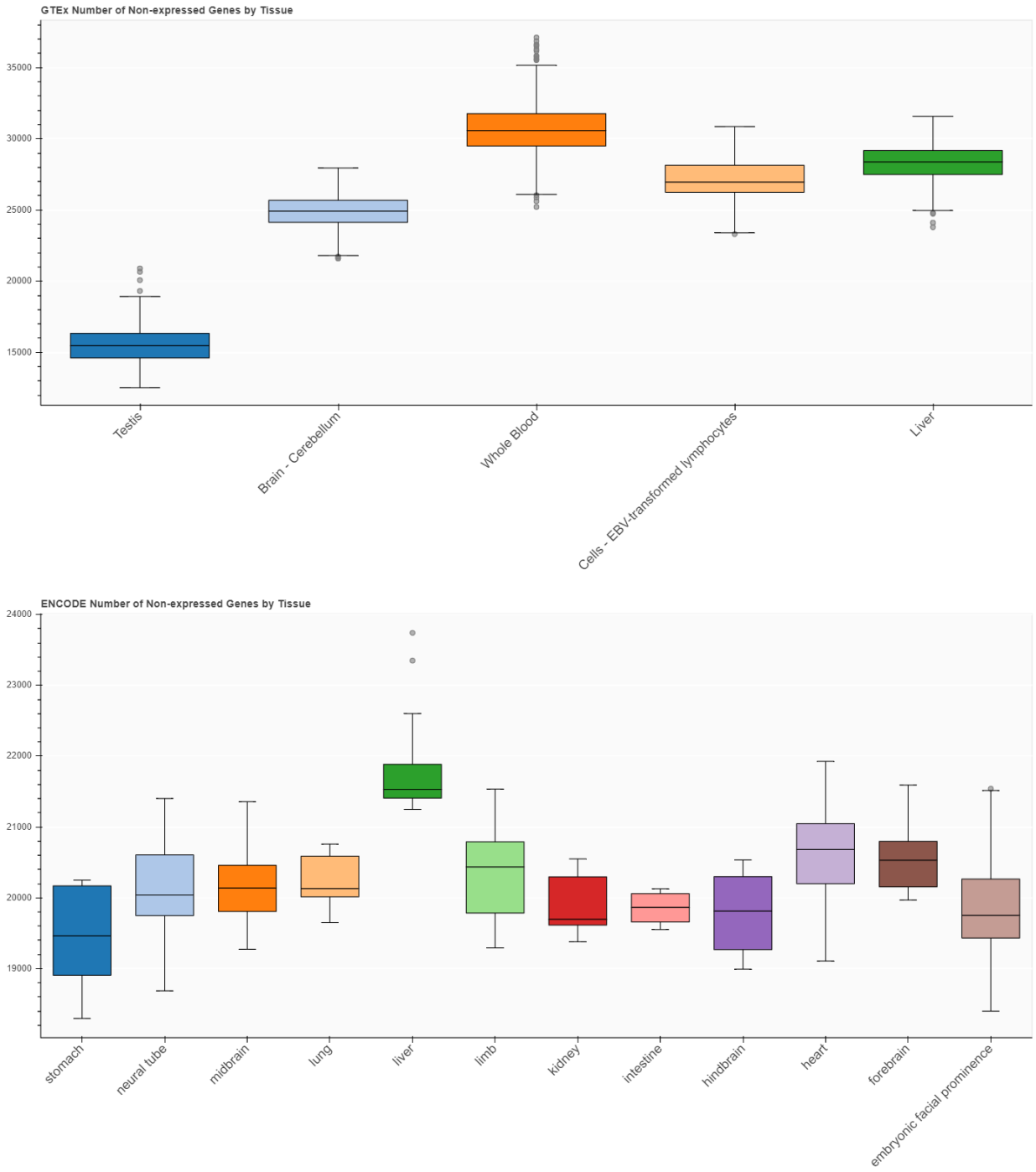

Figure 2: Number of non-expressed genes in each sample.

2 Smooth Quantile Normalization

Smooth quantile normalization, implemented in Bioconductor package *qsmooth*, compares the biological variability within each given group and the background and use this information to compute the weighted

average for means of each quantiles in each group (Figure 3 step 5). This is particularly useful when we want to preserve the global shift in the distribution of expression across different groups of samples.

Given expression data from samples of K different groups. Instead of computing only the mean value for each quantile consider all the samples, smooth quantile normalization also computes the mean value of each quantile considering only the samples belonging to the given group, as well as the mean value of the quantile considering all samples:

$$GroupRankMean(i,k) = (\frac{1}{n_k}\sum_{j\in N^{(k)}} s_{i,j})$$

, where $N^{(k)}$ denotes the set of sample indices in group k ($N^{(0)}$ denotes indices of all samples), and n_k denotes the number of samples in group k (Figure 3, step 2). The procedure continues by computing the explained sum of squares SSB and the total sum of squares SST:

$$\mathrm{SSB}_i = \sum_{k=1}^{K}(GroupRankMean(i,0) - GroupRankMean(i,k))^2$$
$$\mathrm{SST}_i^{(g)} = \sum_{j=1}^{n}(GroupRankMean(i,g) - GroupRankMean(i,0))^2$$

, where g denotes the group index for sample. The weight can then be calculated using SSB and SST (Figure 3, step 6). Finally, the normalized expression value assigned back is the weighted average for the means derived from the sample group and from the background of the corresponding rank (Figure 3. step 7):

$$w_i = \mathrm{smooth}(1 - \frac{SSB_i}{SST_i})$$
$$m'_i = w_i GroupRankMean(i,0) + (1-w_i)GroupRankMean(i,j)$$

, where j is the group index of the sample. Another difference between the implementation of smooth quantile normalization and quantile normalization is the way they process genes with the same read count. Given the i^{th} to $(i+l)^{th}$ entries of the sorted expression matrix from one particular sample $\mathbf{S}_j$ have the same read count, then the ranks recorded in $\mathbf{R}$ of the genes corresponding to these entries will be randomly selected from $\{i, i+1, ..., i+l\}$ without replacement. (Figure 3, step 3). This is often referred to as the *random* ranking method when dealing with tied values. The normalized expression for these genes will be (Figure 3, step 8):

$$\mathbf{X}^{(norm)}_{i:i+l,j} = \frac{1}{l}\sum_{a=i}^{i+l} m'_a$$

Similar to quantile normalization, smooth quantile normalization also suffers from the tied values presented in the data.

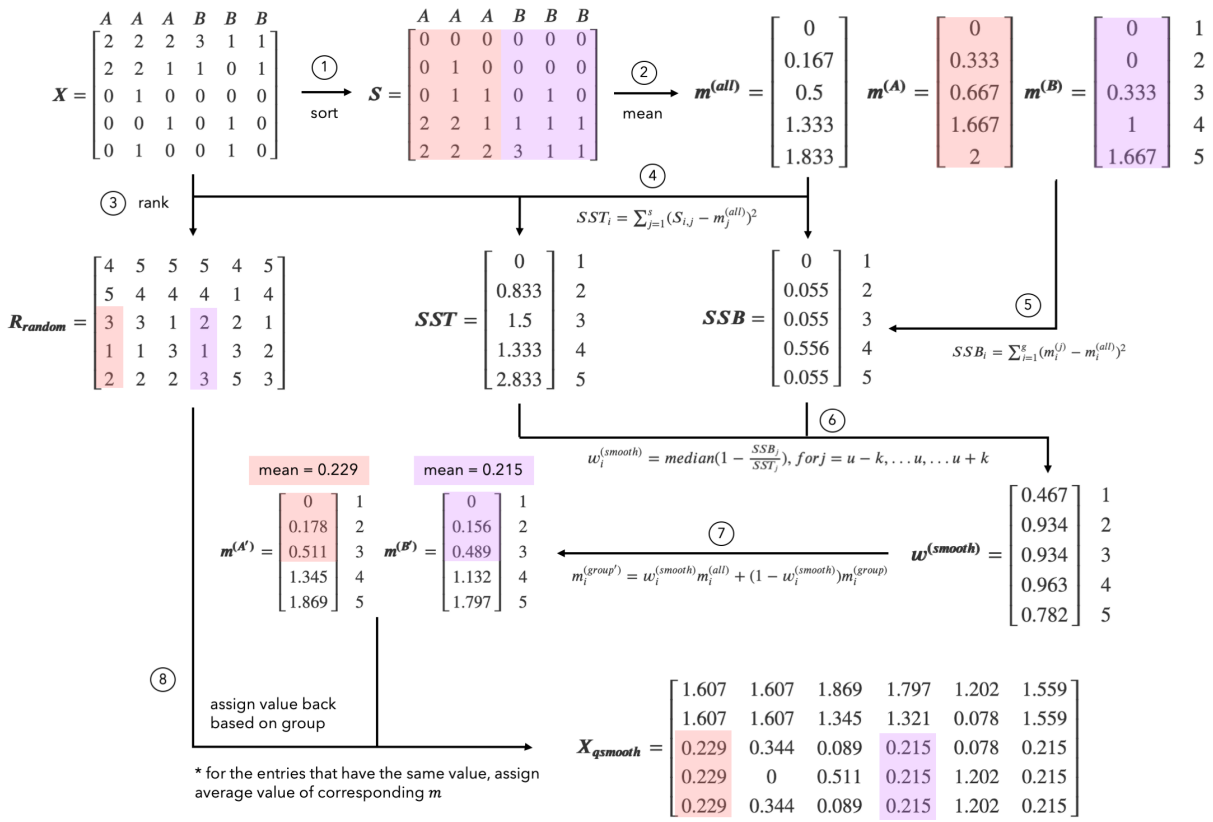

Figure 3: Smooth Quantile normalization.

3 Supplementary Information of CAIMAN algorithm

3.1 Construct group specific expression profile

Given the quantile-normalized expression, and the group information, CAIMAN first performs log2-transformation with a pseudo count of 1 and concatenates the resulting dataset into a vector of expression profile for each group. Since smooth quantile normalization ensures the distribution of expression is the same for samples in the same group, CAIMAN randomly selects at most 100 samples to construct this group specific expression profile to improve efficiency in terms of computational time and memory usage. This sampling procedure makes CAIMAN scalable to datasets consist of large amount of samples. The expression profile for each group is denoted as $\mathbf{x}$. Next, CAIMAN augments this group specific expression profile by concatenating with its negative transformation:

$$\mathbf{x}^{(aug)} = \left[\mathbf{x} \parallel -1 \odot \mathbf{x}\right]^T$$

3.2 Initialization of parameters

CAIMAN uses a Gaussian mixture model to approximate the distribution of augmented group specific expression profile. CAIMAN uses the 2 components with mean equals to 0 to model the distribution of genes with low signal. These components are refer to as the *center components*. Note that CAIMAN uses two center components to make sure the posterior probability is comparable with the flanking components. However, the parameter is the same for the two center components. Therefore, we use the term "center component" in singular form throughout the main manuscript. CAIMAN initializes the centered components with the following rules:

$$\mu_0 = \mu_1 = 0$$
$$\sigma_0 = \sigma_1 = 1$$

For genes with higher signal, CAIMAN uses k-number of flanking components initialized with means symmetrical to 0:

$$\mu_{2z} = p_z$$
$$\mu_{2z+1} = -p_z$$
$$\sigma_{2z} = \sigma_{2z+1} = 3$$

, where $z = 1, ...k/2$ and $\mathbf{p}$ is the series of evenly spaced numbers over 99^{th} to 70^{th} percentile over k interval. Finally, we initialize the probability of the model to be the same across all components:

$$G_z \sim \text{Normal}(\mu_z, \sigma_z)$$
$$\mathbf{x}^{(aug)} \sim \sum_{z=0}^{k} \pi_z G_z$$

3.3 Fitting and correction process

CAIMAN fits the above mentioned Gaussian mixture model using a modified expectation maximization algorithm with the means of centered component remain constant to 0:

$$\gamma_{ij} = \frac{\pi_j \phi(x_i^{(aug)}, \mu_i, \sigma_j)}{\sum_{z=0}^{2k} \pi_z \phi_z \phi(x_i^{(aug)}, \mu_i, \sigma_j)}$$
$$\mu_j = \frac{\gamma_{ij}}{\sum_{i=1}^{n} \gamma_{ij}} \quad \text{for } j \notin \{0, 1\}$$
$$\sigma_j = \frac{\sum_{i=1}^{n} \gamma_{ij} (x^{aug)_i - \mu_j})}{\sum_{i=1}^{n} \gamma_{ij}}$$
$$\pi_j = \frac{1}{n} \sum_{i=1}^{n} \gamma_{ij}$$

, where i and n is the index for each entry and number of entries for $\mathbf{x}^{(\mathbf{aug})}$. To ensure the values of parameters are identical for the positive components and their negative counterpart, CAIMAN then takes the average of standard deviation and the absolute mean values after each iteration:

$$\mu_{2z} \leftarrow \frac{1}{2}(\mu_{2z} - \mu_{2z+1})$$
$$\mu_{2z+1} \leftarrow -\frac{1}{2}(\mu_{2z} - \mu_{2z+1})$$
$$\sigma_{2z} \leftarrow \frac{1}{2}(\sigma_{2z} + \sigma_{2z+1})$$
$$\sigma_{2z+1} \leftarrow \frac{1}{2}(\sigma_{2z} + \sigma_{2z+1})$$

Lastly, we computed the posterior probability for each entry belonging to the center and flanking components:

$$P_c(x) = \sum_{z=0}^{1} P(G_z|x)$$
$$P_f(x) = 1 - P_c(x)$$

If $P_c(x) > P_f(x)$, CAIMAN considers the input value as a low signal, and replace the original value with 0.

3.4 Number of Flanking Components

The number of flanking components can be optimized by setting the parameter *--adaptive*. By setting this parameter, the CAIMAN algorithm starts fitting the model by 2 flanking components on each side, CAIMAN increases the number of flanking components if the likelihood ratio test for model with more flanking components yield a significant difference in the negative log-likelihood compares to the models with fewer flanking components ($p\_value < 1e-5$)

4 Validation dataset

Using the read counts of 96 spike-ins genes $\mathbf{X}^{(spikein)}$, we established the validation dataset with the following equation:

$$\mathbf{x}_j^{(val)} = \frac{\mathbf{x}_j^{(count)}}{s_j}, \quad s_j = \frac{n \sum_{i=1}^{s} x_{ij}^{(spikein)}}{\sum_{k=1}^{n} \sum_{i=1}^{s} x_{ik}^{(spikein)}}$$

where $n = 126$ is the number of samples, $s = 96$ is the number of spike-in genes, and $\mathbf{x}_j$ denotes the column j vector in the matrix $\mathbf{X}$.

5 Number of tissue exclusive genes

ENCODE		GTEx	
Kidney	27	Testis*	3992
Intestine	13	Brain - Cerebellumn	122
Lung	12	Whole Blood	81
Heart	11	Cells - EBV-transformed lymphocytes	54
Liver	11	Liver	75
Stomach	9		
Embryonic Facial Prominence*	3		
Limb*	1		
Neural Tube*	1		
Forebrain*	1		

* Excluded from analysis for visualization purpose.

6 Comparison with MIXnorm

We applied *MIXnorm* to the ENCODE dataset without the tissue information using the command *func_MIXnorm* with default parameters. It took 1502.6 seconds to complete the normalization and inference of the posterior probability for non-expressed genes. In contrast, it took 5.89 seconds for CAIMAN to complete the inference and correction. All of the above mentioned analysis is conducted using a laptop with Intel Core i7-8750H processor (in single thread) with 32Gib of memory. Note that MIXnorm is not developed for the purpose of correcting false-positive associations. However, MIXnorm implemented an efficient nested expectation–maximization, which is closely related to the expectation-maximization algorithm implemented in CAIMAN.