Appendix A

RNA-Seq normalization methods

The goal of normalization in RNA-Seq data is to remove sample specific differences arising due to library size (i.e. sequencing depth), sequencing platform or other technical variations as well as within-sample transcript/gene-specific effects related to gene length and GC-content [1]. Presently, there are many methods available for normalization of RNA-Seq data [1,2]. These methods try to scale raw read counts in each sample/lane/run by a single sample specific or lane-specific factor reecting library size [1].
To understand the present normalization methods, consider N RNA-seq experiments, here N represents a single RNA-Seq run or sample which produces sequencing data for a particular RNA-Seq sample. From these RNA-Seq experiments, we have N FASTQ files which contains short reads produced by the different RNA-Seq samples. We map these short reads to transcriptome (a collection of T transcript sequences) of interest using an aligner software such as Kallisto or Salmon [3,4]. After mapping the short reads from N FASTQ files to the transcriptome we observe the following:
Let R_{ij} be the number of reads that mapped to the i^{th} transcript from j^{th} sample. Let $X_j = \sum_i R_{ij}$ be the total number of reads mapped to the transcriptome from j^{th} sample. X_j is also known as the library size of the j^{th} sample[1]. Let L_i be the length of i^{th} transcript.
RPKM, ERPKM, FPKM, CPM, and TPM are common metrics that quantify abundance of transcripts while normalizing for sequencing depth and gene length within a sample [5]. These metrics are comparable among different samples only if we assume that the total expression is same under different samples/conditions [6].

– RPKM: RPKM (Reads Per Kilobase Million) rescales gene counts to correct for differences in both library sizes and transcript length. RPMK for i^{th} transcript in j^{th} sample is computed as:

$$RPKM_{ij} = 10^9 \frac{R_{ij}}{X_j * L_i}$$

– ERPKM: ERPKM (Effective Reads Per Kilobase Million) is same as RPKM but replaces transcript length by effective length.

$$ERPKM = 10^9 \frac{R_{ij}}{X_j * L'_i}$$

Where, $L'_i = L_i - l_r + 1$ is the effective length and l_r is the read length.

– FPKM: FPKM (Fragments Per Kilobase Million) is the expected number of fragments per kilboase of transcript per million reads. This is almost same as RPKM but if the data is paired then only one of the mates is counted, i.e. fragments are counted rather than reads.

$$FPKM = 10^9 \frac{R_{ij}}{X_j * L_i}$$

– CPM: CPM (counts per million) scales the raw counts by the number of fragments sequenced (X) times one million

$$CPM = 10^6 \frac{R_{ij}}{X_j}$$

– TPM: TPM (transcripts per million) divides the raw counts by transcript length and the library size and scales this ratio by one million.

$$TPM = 10^6 \frac{R_{ij}}{L_i} \left(\frac{1}{\sum_k \frac{R_k j}{L_k}} \right)$$

[1] Here we consider the library size to be the total number of reads mapped which is different that the total number of reads produced by RNA-Seq. However, these two numbers should be very close to each other.

Methods like Total count (TC), upper quartile (UQ), median (Med) and quantile normalization are based on simple descriptive statistics of the data. These methods assume the DE and non-DE genes equally affected by technical effects. One main drawback of above normalization methods is that they forcefully change the distributions of the samples assuming them to be the same.

– TC: Divides gene counts by the total number of mapped reads (or library size) for each sample/lane and multiplied by the mean total count across all the samples of the dataset.

$$TC = \frac{R_{ij}}{X_j}.\left(\frac{\sum_j R_{ij}}{N}\right)$$

– UQ: Upper quartile, instead of total counts, upper quartile of counts more than 0 is computed for each sample and is used as a scale factor.

$$UQ = \frac{R_{ij}}{Q_{75,j}}.\left(\frac{\sum_j R_{ij}}{N}\right)$$

– Med: Median counts, instead of total counts, of counts more than 0 is computed for each sample and is used as a scale factor.

$$Med = \frac{R_{ij}}{median_j}.\left(\frac{\sum_j R_{ij}}{N}\right)$$

– Quantile: Quantile normalization forces the distribution of normalized data to be the same for each sample by replacing each quantile with the mean or median of that quantile across all samples.

Algorithm 1 Quantile Normalization

```
1: Sort each column of the expression matrix to make each row contain the same quantiles of each sample.
2: Replace each entry in the sorted read count matrix with the mean or median of that row.
3: Change order of each column back to the original order.
```

Other methods like TMM [7] and DESeq [8] are proposed for differential expression analysis of RNA-seq data. These methods assume that majority of genes are **not** differentially expressed across the different samples.

– TMM: Calculates scaling factors for each sample with one sample being considered as a reference sample. The scaling factor is computed as the weighted mean of log ratios for each sample with respect to the reference, after exclusion of the most expressed genes and the genes with the largest log ratios.
Let r^{th} sample to be the reference sample. The log-fold change for i^{th} feature in j^{th} sample, with respect to the r^{th} sample, is defined as:

$$M_{ij}^r = log_2 \frac{R_{ij}/X_j}{R_{ir}/X_r}$$

The absolute expression level is defined as:

$$A_{ij}^r = \frac{1}{2} log_2 \left(\frac{R_{ij}}{X_j}\frac{R_{ir}}{X_r}\right)$$

From the j^{th} sample the genes are trimmed based on M^r_{ij} and A^r_{ij} values. The remaining set of genes, G, is used to compute the scaling factor as:

$$log_2\left(TMM_j^{(r)}\right) = \frac{\sum_{i\in G}\omega^r_{ij}M^r_{ij}}{\sum_{i\in G}\omega^r_{ij}}$$

– DESeq: Calculates DESeq scaling factor for a given sample as the median of the ratio, for each gene, of its read count over its geometric mean across all the samples.

$$\hat{s} = median_i\left\{\frac{k_{ij}}{\left(\prod_{v=1}^{m} k_{iv}\right)^{1/m}}\right\}$$

Above mentioned normalization methods (TMM and DESeq) assumes that only a small subset of genes are differentially expressed across conditions [9]. Such assumption can only be true across fairly consistent samples. In large heterogeneous datasets from multiple tissues or conditions such assumptions are easily violated. Thus applicability of these methods are limited to datasets where only a small proportion of genes are upregulated.
Certain methods, such as RUV [10], use control sequences for normalization. These methods assume that certain control sequences exist in data which behaves as expected under different conditions. For example negative controls don't change expression under different conditions. Under the assumption that technical effects which affect the genes also effect the controls in the same way, the controls could be used for normalization.
Two recently proposed methods assumes that RNA-Seq data from multiple different studies are heterogeneous and have other batch effects other than due to the library size.

– Smooth quantile: Smooth quantile (qsmooth) normalization [11] is a generalization of quantile normalization. It assumes that the statistical distributions of each sample should be similar within a biological group. It computes a weight at every quantile that compares the variability between groups relative to within groups.
– Graph based algorithm: A graph-based normalization algorithm was proposed by [12]. This algorithm assumes that a set of reference transcripts are present in the RNA-Seq samples. These set of reference transcripts are expressed at equivalent levels across conditions. This algorithm first identify these reference transcripts from the data using a graph-based algorithm. Then, the identified references are used to normalize the data.

Appendix B

Batch correction methods

This section describes two commonly used batch correction methods used with RNA-Seq data: Combat and SVA.

Combat

Combat was designed to correct batch effects in microarray data with small batch sizes [13]. Combat requires knowledge of batch covariates to correct the batch effects. This method assumes that the batch effects often affect many genes in similar ways. Combat uses empirical Bayes estimation by pooling information across all genes in each batch. The steps involved in ComBat are described below:

1. **Normalize data:** Normalize data and filter out genes which are missing in more than 80% samples to eliminate noise. Assume there are G genes observed over m batches with n_i samples within batch i for $i = 1, \ldots, m$. The model assumed is:

$$Y_{ijg} = \alpha_g + X\beta_g + \gamma_{ig} + \delta_{ig}\epsilon_{ijg} \tag{1}$$

   Where the errors, $\epsilon \sim N(0, \sigma_g^2)$

2. **Standardize the data:** The data is standardize so that genes have similar overall mean and variance. The model parameters α_g,β_g, and γ_{ig} are estimated for $i = 1, \dots, m$ and $g = 1, \dots, G$ using gene-wise ordinary least square approach. The standardize data Z_{ijg} is calculated as

$$Z_{ijg} = \frac{Y_{ijg} - \widehat{\alpha_g} - X\widehat{\beta_g}}{\widehat{\sigma_g}} \tag{2}$$

Where $\widehat{\sigma_g^2} = \frac{1}{N}\sum(Y_{ijg} - \widehat{\alpha_g} - X\widehat{\beta_g} - \widehat{\gamma_{ig}})^2$
3. **EB batch effect parameter estimates:** it is assumed that $Z_{ijg} \sim N(\gamma_{ig}, \delta_{ig}^2)$ where γ and δ are different parameters from the ones in Equation 1. The priors are assumed to be normal and inverse gamma for γ and δ respectively i.e. $\gamma_{ig} \sim N(Y_i, \tau_i^2)$ and $\delta_{ig}^2 \sim InverseGamma(\lambda_i, \theta_i)$
The hyperparameters, $\gamma_i, \tau_i^2, \lambda_i, \theta_i$, are estimated empirically from the standardized data. Based on these distributional assumptions estimates for batch effect parameters, γ_{ig} and δ_{ig}^2 can be obtained as γ_{ig}^* and δ_{ig}^{2*}
4. **Adjust data for batch effects:** The adjusted data, y_{ijg}^* can be calculated as

$$y_{ijg}^* = \frac{\widehat{\sigma_g}}{\widehat{\delta_{ig}^*}}(Z_{ijg} - \widehat{\gamma_{ig}^*}) + \widehat{\alpha_g} + X\widehat{\beta_g} \tag{3}$$

SVA

SVA was originally proposed for microarray data [14]; a variant Svaseq was later described for RNA-Seq data [15]. A short description of the savseq model is below.
Expression of i^{th} gene in j^{th} sample is modeled as

$$g_{ij} = b_{i0} + b_{i1y_j} + c_i a_j + d_i u_j + e_{ij} \tag{4}$$

Where the first term on the right is the baseline expression level. The second term models the phenotype effect. The third term is effect of known batch variable (a_j). The fourth term is effect of unknown atrifact (u_j) and last term are the measurement errors.
The input to SVA is normalized expression values. The general framework of SVA is:

1. Identify genes that are only affected by unknown artifacts.
2. Perform a decomposition of the data for just these genes to identify the estimates of the artifacts.
3. Include the artifact estimates in subsequent analyses as if they were known.

SVA algorithm assumes that the effect of L factors, g_l, could be modeled using an additive effect model. SVA algorithm also assumes that effect of L factors could be estimated using K orthogonal vectors which spans the same linear space as g_l such that $K < L$ [14]. Then, to estimate the effect due to the L unknown factors, SVA estimates K surrogate variables from the data.
In the first step, SVA identifies set of genes that are only affected by unknown artifacts. In presence of negative controls in RNA-Seq data, this step could be skipped and the negative controls are used. Otherwise, a model without unknown effects if fit to the expression data and the residual matrix $\boldsymbol{R}$ is computed. In second step, SVD is performed on $\boldsymbol{R}$ and significant eigengenes are identified. Then, surrogate variables are constructed for each K eigengenes identified. In third step, the estimates of surrogate variables are passed on to any subsequent analysis.

Appendix C

General Factor Analysis Model

Factor analysis (FA) aims to model observed data into an unobserved systematic part and an unobserved error part. FA models the observed variables as linear functions of the unobserved *factors*. The components of the error vector are assumed to be independent.

Model definition Let $\boldsymbol{X}$ be the observed data matrix. $\boldsymbol{X}$ contains n column vectors containing observations of t random variables. In this case, n columns corresponds to n RNA-Seq samples which observes the counts of t transcripts. We want to model each observation of $\boldsymbol{X}$ with a linear combination of k *factors* where $k << t$. These factors are latent or hidden. It is assumed that the *factors* are linearly related to the variables.
Each observation of $\boldsymbol{X}$ could be written as:

$$x_{ij} = \mu_{ij} + l_{i1}f_{1j} + l_{i2}f_{2j} + l_{i3}f_{3j} + \cdots + l_{ik}f_{kj} + \epsilon_{ij} \tag{5}$$

Where, μ_{ij} is the mean expression of i^{th} transcript in the j^{th} sample and ϵ_{ij} is the random error term in the measurment of i^{th} transcript in the j^{th} sample. l_{ij} is the loading or weight of i^{th} transcript on the factor f_{ji}. l_{ij} can also be thought as regression coefficients in the multiple regression model (Equation 5).
In matrix notation the model can be written as:

$$\boldsymbol{X} = \boldsymbol{\mu} + \boldsymbol{L}\boldsymbol{F} + \boldsymbol{\epsilon} \tag{6}$$

In Equation 6, $\boldsymbol{L}$ is the **loading matrix** and $\boldsymbol{F}$ is defined as **factors**. i^{th} row in $\boldsymbol{L}$ contains loadings or weights for i^{th} observation in $\boldsymbol{X}$. The j^{th} column in $\boldsymbol{F}$ contains the factors for the j^{th} column in $\boldsymbol{X}$.
Following assumptions are imposed on the model:

1. $\boldsymbol{F}$ and $\boldsymbol{\epsilon}$ are independent.
2. $\boldsymbol{E}(\boldsymbol{F}) = 0$
3. Factors are uncorrelated i.e. $Cov(\boldsymbol{F}) = I$

In simple terms, we are trying to model the observed counts, x_{ij} in terms of k latent factors. The interpretation of these hidden factors may not always be easy and straightforward. Each transcript has a vector of k factor loadings. These loadings can be thought of as the weights each factor contributes towards the count x_{ij}. Whereas, each sample has associated k factors. The value of these factors is different for different samples.

Optimal number of factors Application of FA could be potentially limited when the number of latent factors are unknown or hard to guess. The interpretation of the model can strongly depend on the number of latent factors modelled. For this problem, there are few methods to objectively find number of factors in a FA model:

– Parallel analysis is a Monte-Carlo based statistical method used to determine the number of factors [16]. This method compares the observed eigenvalues to the eigenvalues generated from a Monte-Carlo simulated matrix created from random data of the same size. A factor or component is retained if the associated eigenvalue is bigger than the 95th percentile of the distribution of eigenvalues derived from the random data.
– Velicer's MAP test [17] involves using PCA followed by partial correlations to estimate the number of factors.
– Bayesian Information Criterion (BIC) is a widely used model selection method which has been also applied to find the optimal number of factors in FA models [18].
– Akaike information criterion (AIC) is an estimator of the relative quality of statistical models for a given set of data which has also been applied to FA [19].
– Corrected AIC (CAIC) a modified version of AIC, which corrects the bias in AIC, was proposed for FA [20]

The f-scLVM model

f-scLVM is a factor analysis based method for modelling single-cell Seq data in terms of known cell covariates, annotated and unannotated factors [21]. f-scLVM models the observed data using the known cell covariates, annotated and unannotated factors. The known cell covariates are known beforehand e.g., tissue type, disease, and biological treatment. The annotated factors corresponds to a set on biologically relevant factors which could affect expression of certain genes. For example known pathways are used as annotated factors. The unannotated factors are unknown effects on the data. These could be technical effects such as batch effects or other undocumented biological effects. The number of unannotated factors is hyper-parameter which is input by the user and is not inferred by the sc-LVM algorithm.

The basic model for a $N\ X\ G$ log-count matrix $\boldsymbol{Y}$ is defined as

$$\boldsymbol{Y} = \sum_{c=1}^{C} \boldsymbol{u_c V_c^T} + \sum_{a=1}^{A} \boldsymbol{p_a R_a^T} + \sum_{h=1}^{H} \boldsymbol{s_h Q_h^T} + \boldsymbol{\psi} \tag{7}$$

Here, $\boldsymbol{u_c}$ are the known cell covariates; $\boldsymbol{V_c^T}$ are the associated weights and the first term on the right models the effect of cell covariates. $\boldsymbol{p_a}$ are the unannotated factors; $\boldsymbol{R_a^T}$ are the associated weights and the second term on the right models the effect of the annotated factors. $\boldsymbol{s_h}$ are the unannotated factors; $\boldsymbol{Q_h^T}$ are the associated weights and the third term on the right models the effect of the unannotated factors. $\boldsymbol{\psi}$ is the residual matrix.
f-scLVM model uses deterministic variational Bayesian approximations for estimation and puts many assumptions on the prior distributions [21]. For full details of the model please see the original paper by Buettner et. al. [21].

Appendix D

Methods to evaluate RNA-Seq normalization methods

To compare quality of RNA-Seq normalization one can use the following methods:

1. **Using simulated datasets:** Using simulated datasets with known technical effects is an effective way to test if the described model fits the data as expected.
2. **Evaluating coexpression networks and GO enrichment:** After technical effects are removed from the data, we expect to see only the biological signal in the data. It is well known that genes working together in common biological pathways will show strong coexpression patterns. If such patterns exist in data then it is likely that the technical effects were removed from the data. GO term analysis can reveal if coexpressed genes share common biological pathways or not.
3. **Clustering samples:** After technical effects are removed, we expect samples from similar biological conditions to show similar expression patterns. Thus after correct normalization we can expect samples from same biological condition to cluster together. One can also use PCA to easily visualize the multidimensional data in reduced dimensions.
4. **Evaluating DEG:** Examining the differentially expressed genes between conditions can be helpful in evaluating the quality of biological signal in the data. If there is evidence that the differentially expressed genes between conditions are biologically associated with the conditions then it may imply that the data contains correct biological signals.
5. **Data entropy** Data entropy method was proposed for quantifying the effects of normalization methods on data [22]. For a given RNA-Seq dataset X with G genes and N samples The data entropy, π is define as

$$\pi = log_2 \sum_{j=1}^{N} s_j log_2 \left(1 + s_j\right)$$

   where s_j is the j^{th} singular value of X.
   Assuming that the normalization method reduces randomness in the data then it is expected that $\pi_{raw} \geq \pi_{normalized}$. However, a drawback of this approach is that it doesn't quantify that if the normalization preserves biologically meaningful signals.

References

1. Dillies, M.-A., Rau, A., Aubert, J., Hennequet-Antier, C., Jeanmougin, M., Servant, N., Keime, C., Marot, G., Castel, D., Estelle, J., et al. (2013) A comprehensive evaluation of normalization methods for Illumina high-throughput RNA sequencing data analysis. *Briefings in bioinformatics,* **14**(6), 671–683.
2. Li, P., Piao, Y., Shon, H. S., and Ryu, K. H. (2015) Comparing the normalization methods for the differential analysis of Illumina high-throughput RNA-Seq data. *BMC bioinformatics,* **16**(1), 347.
3. Patro, R., Duggal, G., Love, M. I., Irizarry, R. A., and Kingsford, C. (2017) Salmon provides fast and bias-aware quantification of transcript expression. *Nature methods,* **14**(4), 417.
4. Bray, N. L., Pimentel, H., Melsted, P., and Pachter, L. (2016) Near-optimal probabilistic RNA-seq quantification. *Nature biotechnology,* **34**(5), 525.

5. Conesa, A., Madrigal, P., Tarazona, S., Gomez-Cabrero, D., Cervera, A., McPherson, A., Szcześniak, M. W., Gaffney, D. J., Elo, L. L., Zhang, X., et al. (2016) A survey of best practices for RNA-seq data analysis. *Genome biology,* **17**(1), 13.
6. Evans, C., Hardin, J., and Stoebel, D. M. (2017) Selecting between-sample RNA-Seq normalization methods from the perspective of their assumptions. *Briefings in bioinformatics,* **19**(5), 776–792.
7. Robinson, M. D. and Oshlack, A. (2010) A scaling normalization method for differential expression analysis of RNA-seq data. *Genome biology,* **11**(3), R25.
8. Anders, S. and Huber, W. (2010) Differential expression analysis for sequence count data. *Genome biology,* **11**(10), R106.
9. Paulson, J. N., Chen, C.-Y., Lopes-Ramos, C. M., Kuijjer, M. L., Platig, J., Sonawane, A. R., Fagny, M., Glass, K., and Quackenbush, J. (2017) Tissue-aware RNA-Seq processing and normalization for heterogeneous and sparse data. *BMC bioinformatics,* **18**(1), 437.
10. Risso, D., Ngai, J., Speed, T. P., and Dudoit, S. (2014) Normalization of RNA-seq data using factor analysis of control genes or samples. *Nature biotechnology,* **32**(9), 896.
11. Hicks, S. C., Okrah, K., Paulson, J. N., Quackenbush, J., Irizarry, R. A., and Bravo, H. C. (2017) Smooth quantile normalization. *Biostatistics,* **19**(2), 185–198.
12. Tran, D.-T., Bhaskara, A., Might, M., and Balagurunathan, K. (2018) A graph-based algorithm for RNA-seq data normalization. *BioRxiv,* p. 321471.
13. Johnson, W. E., Li, C., and Rabinovic, A. (2007) Adjusting batch effects in microarray expression data using empirical Bayes methods. *Biostatistics,* **8**(1), 118–127.
14. Leek, J. T. and Storey, J. D. (2007) Capturing heterogeneity in gene expression studies by surrogate variable analysis. *PLoS genetics,* **3**(9), e161.
15. Leek, J. T. (2014) Svaseq: removing batch effects and other unwanted noise from sequencing data. *Nucleic acids research,* **42**(21), e161–e161.
16. Horn, J. L. (1965) A rationale and test for the number of factors in factor analysis. *Psychometrika,* **30**(2), 179–185.
17. Velicer, W. F. (1976) Determining the number of components from the matrix of partial correlations. *Psychometrika,* **41**(3), 321–327.
18. Hirose, K., Kawano, S., Konishi, S., and Ichikawa, M. (2011) Bayesian information criterion and selection of the number of factors in factor analysis models. *Journal of Data Science,* **9**(2), 243–259.
19. Akaike, H. (1987) Factor analysis and AIC. In *Selected Papers of Hirotugu Akaike* pp. 371–386 Springer.
20. Ogasawara, H. (2016) Bias correction of the Akaike information criterion in factor analysis. *Journal of Multivariate Analysis,* **149**, 144–159.
21. Buettner, F., Pratanwanich, N., McCarthy, D. J., Marioni, J. C., and Stegle, O. (2017) f-scLVM: scalable and versatile factor analysis for single-cell RNA-seq. *Genome biology,* **18**(1), 212.
22. Han, H. and Men, K. (2018) How does normalization impact RNA-seq disease diagnosis?. *Journal of biomedical informatics,* **85**, 80–92.