Data Normalization and Batch Effects

Agus Salim, Ph.D

Big Data Analytics: Application to Modern Genetics
Universitas Jember, 23 Dec 2015
Issues with Modern Genetics Data

Lab experiments usually follow strict steps and protocols, but there can still be problems. Some examples:

- Unequal amount of titration and different RNA/DNA concentration
- Array quality varies from manufacturers: unequal hybridization quality between arrays
- Non-uniform performance of the sequencer
- Lab temperature varies between experiments (temp. affects chemical processes)
- Samples are processed by different lab person

Problems 1-2 may be minimized via normalization

Problems 3-5 can be handled by identifying and removing effects due to the unwanted factors.
Some Examples

Proteomics Data: Three Biological Replicates with Two different Labeling Procedure

Replicates with different procedure are almost uncorrelated!
Some Examples

Sequencing Data: Reads quality deteriorates for latter reads
Some Examples

Two-color microarray: spots with higher intensity tends to be 'down-regulated'
Normalization Methods

- For two-color microarray, we can perform within-array (within-sample) normalization followed by between-array normalization.
- For other types of data in general, only between-array normalization is possible.
- **Note:** all normalization methods implicitly assume that the vast majority of genes are unchanged (not differentially-expressed).
- This assumption is quite plausible under most situations, e.g., most genes need to work properly for Human to be alive.
The most common method is *lowess* normalization. It is based on fitting *lowess* curve to the *MA plot*.

MA plot has two axes; \( M \) as vertical and \( A \) as horizontal axes.

\[
M = \log_2(R/G) = \log_2(R) - \log_2(G)
\]

\[
A = \frac{1}{2}[\log_2(R) + \log_2(G)]
\]

The idea is that a good experiment should yield log intensity ratio (\( M \)) that has an average approximately 0 (not differentially expressed), regardless of the magnitude of log intensity (\( A \)).
Lowess (Loess) Normalization

- Lowess = locally weighted scatterplot smoothing
- It estimates $y$-value at $x_0$, $\hat{y}(x_0)$ as a weighted sum of $y$-values in the neighborhood, with the weight decreasing the further the corresponding $x$-values from $x_0$.
- In mathematical notation,

$$\hat{y}(x_0) = \frac{\sum_{i=1}^{n} w\left(\frac{x_i - x_0}{h}\right) y(x_i)}{\sum_{i=1}^{n} w\left(\frac{x_i - x_0}{h}\right)}$$

- Various weight function $w(.)$ can be used with $h$ is the bandwidth parameter; tricube weight $w(u) = (1 - |u|^3)^3 I_{[|u|<1]}$ is popular.
Lowess: illustration

The Lowess plot illustrates the local linear regression result. The red points represent the data points, with the yellow area highlighting the local neighborhood around the point of interest $X_0$. The green line is the predicted value $\hat{Y}(X_0)$ for the local neighborhood, while $Y(X)$ represents the observed values. The local linear regression result is shown as a smooth curve through these points.
Lowess (Loess) Normalization

- Fit a lowess curve with $M$ as $y$ and $A$ as $x$ axis
- Denote $\hat{M}(a_i)$ as the estimate of $M$ when $A = a_i$
- The corrected log intensity ratio is calculated as the residual,

$$M^c(a_i) = M(a_i) - \hat{M}(a_i)$$
Lowess: illustration

Raw

Corrected
Between-array Normalization

- Some samples may have a slightly higher concentration (due to titration error)
- Performing between-array normalization ensures that data distribution across arrays is the same.
- The most popular between-array normalization method is Quantile Normalization
- Quantile Normalization ensures that data from different arrays have the same distribution.
Quantile Normalization

- Pick a reference array; when there is no reference array, we can use the average of all arrays as reference.
- Sort the values in the reference array, so that $y(k)$ is the $k$th smallest value with rank $= k$.
- For array $i$, perform the normalization as follows:
  1. Rank the expression values so that $r_{ij}$ is the rank of expression value for gene $j$
  2. Replace the expression values for gene $j$, $x_{ij}$ with $y(r_{ij})$, that is we replace the original value with the value in the reference array that has the same rank.
Batch Effects

- Normalization is great at removing between-array mean difference
- But often, the between-array differences are highly non-linear and also affects the correlation
- Example: SILAC Proteomics Data
Batch Effects

- We call this remaining difference between-array 'batch effect'
- This is because, often, omics data are processed on multiple days (months) in batches
- There are several approaches for removing the 'batch effect', but we will discuss RUV approaches of Terry Speed
Batch Effects: Removing Unwanted Variations (RUV) approach

- Suppose we have $m$ arrays under different experimental conditions. The information about experimental condition is contained in $(m \times p)$ matrix $X$.
- Let $y_i$ be the row vector representing normalized data for gene $i$.
- Assume (realistically) that some genes will be affected by the experimental condition and others will not.
- Without interference from unwanted factors $W$, we can reasonably model the data as
Removing Unwanted Variations (RUV) approach

- For genes affected by experimental condition
  \[ y_i = \alpha_i + X\beta_i + \epsilon_i \]

- For genes unaffected by experimental condition
  \[ y_i = \alpha_i + \epsilon_i \]

- But what if there is interference from unwanted factor?
Now, for genes affected by experimental condition

\[ y_i = \alpha_i + X\beta_i + W\gamma_i + \epsilon_i \]

And, for genes unaffected by experimental condition

\[ y_i = \alpha_i + W\gamma_i + \epsilon_i \]

But what if there is interference from unwanted factor?
It will be straightforward to adjust for $W$ if the unwanted factors are known.

and..sometimes the unwanted factors are known (e.g., lab persons, order of run etc)

But..more often the unwanted factors are unknown and needs to be inferred from the data!!?

Let’s look at the model for genes unaffected by the experimental condition

$$y_i = \alpha_i + W\gamma_i + \epsilon_i$$
Removing Unwanted Variations (RUV) approach

- Apart from random variations, the only variables causing variations in the unaffected genes are $\mathbf{W}$.
- Let’s denote the between-array covariance matrix of the unwanted genes as, $S_c = \frac{1}{n}(Y - \mu)(Y - \mu)'$
- We can estimate $\mathbf{W}$ using eigenvectors of $S_c$.
- This is because, using eigen-decomposition $S_c = \mathbf{U}\mathbf{\Lambda}\mathbf{U}'$
- So that $y_i = \mu_i + \mathbf{U}\sqrt{\lambda^*}$, $\lambda^*$ is a vector with element $\lambda^*_i = n\lambda_i$
- Hence, $\mathbf{U}$ is an excellent estimate for $\mathbf{W}$.
Removing Unwanted Variations (RUV) approach

RUV algorithm (2-step)

1. Pick a set of 'control' (unaffected) genes and estimate the unwanted factors as the eigenvectors of the covariance matrix $S_c$

2. Use the eigenvectors $U$ to fit regression model,

$$y_i = \alpha_i + X\beta_i + U\gamma_i + \epsilon_i$$

Calculate the adjusted data as $y_i^{corr} = y_i - U\hat{\gamma}_i$
SILAC Proteomics Data: 23 experiments (3 replicates each) with two labelling procedure over 5 different batches

Normalized Data: Almost no difference in mean, but inter-replicate correlation often low
SILAC Proteomics Data: 23 experiments, 3 replicates each

Using RUV (4-step), we identified 8 unwanted factors. The inter-replicate correlation is better after adjustment.

Agus Salim, Ph.D
Data Normalization and Batch Effects
Using RUV (4-step), we identified 8 unwanted factors. The inter-replicate correlation is better after adjustment.
RUV: the estimated unwanted factors

(a)-(b) The first two pairs of unwanted factors, by labeling procedure
(c)-(d) The first two pairs of unwanted factors, by batch
Some Useful References