Bioinformatics, 34(7), 2018, 1141–1147
doi: 10.1093/bioinformatics/btx635
Advance Access Publication Date: 9 October 2017
Original Paper

Gene expression

Detecting hidden batch factors through data-adaptive adjustment for biological effects

Haidong Yi[1,†], Ayush T. Raman[2,3,†], Han Zhang[1,4,*], Genevera I. Allen[5] and Zhandong Liu[2,3,*]

[1]College of Computer and Control Engineering, Nankai University, Tianjin 300350, China, [2]Graduate Program in Structural and Computational Biology and Molecular Biophysics, Baylor College of Medicine, Houston, TX 77030, USA, [3]Department of Pediatrics, Neurological Research Institute, Baylor College of Medicine, Houston, TX 77030, USA, [4]Tianjin Key Laboratory of Intelligent Robotics, Nankai University, Tianjin 300350, China and [5]Department of Statistics, Rice University, Houston, TX, 77030 USA

*To whom correspondence should be addressed.
†The authors wish it to be known that the first two authors contributed equally.

Associate Editor: Ziv Bar-Joseph

Received on March 5, 2017; revised on September 5, 2017; editorial decision on September 29, 2017; accepted on October 6, 2017

Abstract

Motivation: Batch effects are one of the major source of technical variations that affect the measurements in high-throughput studies such as RNA sequencing. It has been well established that batch effects can be caused by different experimental platforms, laboratory conditions, different sources of samples and personnel differences. These differences can confound the outcomes of interest and lead to spurious results. A critical input for batch correction algorithms is the knowledge of batch factors, which in many cases are unknown or inaccurate. Hence, the primary motivation of our paper is to detect hidden batch factors that can be used in standard techniques to accurately capture the relationship between gene expression and other modeled variables of interest.

Results: We introduce a new algorithm based on data-adaptive shrinkage and semi-Non-negative Matrix Factorization for the detection of unknown batch effects. We test our algorithm on three different datasets: (i) Sequencing Quality Control, (ii) Topotecan RNA-Seq and (iii) Single-cell RNA sequencing (scRNA-Seq) on Glioblastoma Multiforme. We have demonstrated a superior performance in identifying hidden batch effects as compared to existing algorithms for batch detection in all three datasets. In the Topotecan study, we were able to identify a new batch factor that has been missed by the original study, leading to under-representation of differentially expressed genes. For scRNA-Seq, we demonstrated the power of our method in detecting subtle batch effects.

Availability and implementation: DASC R package is available via Bioconductor or at https://github.com/zhanglabNKU/DASC.

Contact: zhanghan@nankai.edu.cn or zhandonl@bcm.edu

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

1.1 Batch effect

Batch effects are defined as the systematic non-biological variations added to the 'omics' dataset during data acquisition. Batch effects can arise from variations in experimental platforms, laboratory conditions, sources of samples and personnel changes (Leek *et al.*, 2010; Luo *et al.*, 2010; Scherer, 2009). If batch effects are not removed properly, they will confound and diminish the biological signals leading to incorrect conclusions (Akey *et al.*, 2007; Gilad and Mizrahi-Man, 2015). It is therefore mandatory to correct for batch effect before any downstream Bioinformatics analysis (Belorkar and Wong, 2016; Cusanovich *et al.*, 2014; Hornung *et al.*, 2016; Leek *et al.*, 2010).

Algorithms for batch detection and correction can be classified into two main categories: (i) location–scale (L/S) adjustment method and (ii) matrix factorization method (Lazar *et al.*, 2012). The L/S method uses the central idea of transforming the mean (Location) and variance (Scale) to remove batches from the dataset. One of the most widely used L/S adjustment algorithm is ComBat (Evan *et al.*, 2007) and modified Combat (Stein *et al.*, 2015). It uses an *Emperical Bayes* framework for the estimation of the L/S parameters that represent batch effects. These parameters are then used to remove the non-biological variations from the dataset. Distance Weight Discrimination (DWD) (Benito *et al.*, 2004) is another algorithm based on the L/S adjustment method. It uses *Support Vector Machine* algorithm along with batch information to find the optimal hyperplane or DWD hyperplane. The dataset is then adjusted by projecting the batches on the DWD plane and then subtracting out the DWD plane from the dataset.

One of the major limiting factors for the use of the L/S based algorithms in practice is the knowledge of batches, which is not trivial especially in big data projects like ENCODE (Feingold, 2004) and TCGA (Weinstein *et al.*, 2013) because of mislabeling, sample swapping and personnel changes. Matrix factorization based methods eliminate this limiting factor by estimating batch effects from the data. Some of the most commonly used algorithms that employ this method are Surrogate Variable Analysis (sva) and Removal of Unwanted Variation (RUV). sva (Leek and Storey, 2007) computes a residual matrix after estimating the effect of covariates of interest such as genotype and condition on each gene. It then uses Singular Value Decomposition (SVD) on the residual matrix to compute unmodeled latent factors. RUV (Gagnon-Bartsch and Speed, 2012; Risso *et al.*, 2014) assumes a set of control genes, i.e. housekeeping genes, or spike-in controls that are not affected by the known covariates, and it then uses factor analysis (and SVD transformation) on a set of control genes to remove these technical variations from the expression matrix. Some of the other commonly known algorithms that use SVD or factor analysis methods are Independent Surrogate Variable Analysis (Teschendorff *et al.*, 2011), PEER (Stegle *et al.*, 2010) and gPCA (Reese *et al.*, 2013).

The major limitation of SVD based approaches is the orthogonality assumption between batch factors, which is almost never true in real dataset (Mostafavi *et al.*, 2013; Teschendorff *et al.*, 2011). Another limitation among all the batch detection methods (Leek and Storey, 2007; Risso *et al.*, 2014; Tung *et al.*, 2016) is the use of linear regression models, which give the same amount of adjustment to all the genes whose expressions are measured in the experiment. The use of negative controls is also a strong assumption when the data is highly confounded (Mostafavi *et al.*, 2013) or when there is variability in spike-in controls (Risso *et al.*, 2014). Overall, it has been observed that SVD based algorithms may not guarantee the removal of all the unknown covariates. Like SVD, non-negative matrix factorization (NMF) is another dimension reduction algorithm proposed by Lee and Seung (1999) and Lee (2001). NMF has been widely used to extract useful feature information in the image analysis (Lee and Seung, 1999) and cancer subtype studies (Brunet *et al.*, 2004). In contrast to PCA methods, NMF allows only positive addition of non-negative basis components and does not require the orthogonality assumption on the basis components as well. Furthermore, there is a close connection between NMF and clustering. Ding *et al.* (2005) proved that NMF is equivalent to spectral clustering in certain settings.

We propose a data-adaptive, non-parametric and non-regression approach to remove the biological signal to prepare the data for batch detection and then apply a semi-non-negative matrix factorization (semi-NMF) method (Ding *et al.*, 2010) to obtain the estimation of the hidden batch factors associated with the samples. To isolate the batch signal, we use fusion penalties that shrink each individual expression profile towards the means of its corresponding biological group in a non-parametric and data-adaptive manner. To ensure the stability of the estimated batch factors, we derive a consensus matrix by applying semi-NMF multiple times. There are three major advantages of our approach compared to existing approaches: (i) it estimates batch effects from the data, (ii) it makes no assumptions on data probability distributions and (iii) it makes no assumptions on all genes affected at the same level by batch effects. We evaluate the performance of our method in three genome-wide expression studies involving large and small sample sizes. Our approach accurately identifies the hidden batch factors from high-throughput genomic datasets and outperforms the current algorithms used for batch detection.

1.2 Notation

In this section, we introduce the notation that will be used in the rest of the paper. We use uppercase boldface letters, such as $\mathbf{U}$, to denote matrices, and boldface lowercase letters, such as $\mathbf{x}$, to denote vectors. U_{ij} denotes the element at the ith row and jth column of $\mathbf{U}$. x_i denotes the ith element of $\mathbf{x}$. $\mathbf{u}_i$ denotes the ith column of the matrix $\mathbf{U}$. $\mathbf{U}^T$ is the transpose of $\mathbf{U}$ and $\mathbf{U}^{-1}$ is the inverse of $\mathbf{U}$. $tr(\cdot)$ denotes the trace of a matrix. $\mathbf{U} \succcurlyeq 0$ means that $\mathbf{U}$ is positive semi-definite (psd) and $\mathbf{U} \succ 0$ means that $\mathbf{U}$ is positive definite (pd). We use calligraphy letter such as $\mathcal{K}$ to denote set. $\|\cdot\|_1$ and $\|\cdot\|_2$ denote L_1 norm and L_2 norm.

2 Materials and methods

2.1 Batch detection through a data-adaptive shrinkage and clustering

We assume dataset $\mathbf{X}_{p\times n} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ be the normalized (For microarray datasets, it can be normalized using rma, gcrma or MAS5. For RNA-Seq dataset, it can be normalized using either abundance estimation or non-abundance normalization methods like DESeq or TMM (edgeR). We used DESeq normalization for our analysis on SEQC and Topotecan datasets.) gene expression matrix with n samples and p genes, where $\mathbf{x}_i = (x_{1i}, \ldots, x_{pi})^T$ is the gene expression vector for sample i. Let $\mathbf{G}$ be the group label matrix, where genotype and/or conditions of each samples are defined. Mathematically, $\mathbf{G}$ is an $n \times n$ indicator matrix, where n is the total number of samples. It can be defined as

$$G_{ij} = \begin{cases} 1, & \text{if sample } i \text{ and } j \text{ under same condition} \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

Our algorithm consists of two steps. First, we apply a data-adaptive shrinkage on $\mathbf{X}$ based on the group label $\mathbf{G}$. The batch free matrix $\mathbf{U}$

Algorithm 1 Data-adaptive shrinkage and clustering

```
Input:
    X: normalized data matrix, G: group label matrix, $\lambda$: regularization parameter, $k$: number of batch factors.[set $\boldsymbol{\lambda}$ as $\lambda$ in the below equation]
1: $\mathbf{U}^* \leftarrow \arg_{\mathbf{U}} J(\mathbf{U}, \mathbf{X}, \mathbf{G}, \boldsymbol{\lambda})$.    ▷ data-adaptive shrinkage
2: $\mathbf{B} \leftarrow \mathbf{X} - \mathbf{U}^*$.
3: $\mathbf{W}_k, \mathbf{H}_k = \arg_{\mathbf{W},\mathbf{H}} ||\mathbf{B} - \mathbf{W}\mathbf{H}||_F$.    ▷ matrix factorization
4: Repeat step 3 for M times with different initializations.
5: Calculate consensus matrix $\bar{\mathbf{C}}_k$.    ▷ average clusters
6: Get batch factors $\mathbf{b}(\lambda, k)$ from $\bar{\mathbf{C}}_k$.
Output:
    Batch factors $\mathbf{b}(\lambda, k)$
```

is the data-adaptive means estimated using fusion penalties. This step allows us to obtain batch matrix **B**, which is the residual matrix of **X** and **U**. In the second step, we use semi-NMF to extract the batch factors from matrix **B**. It can be factorized into **W** and **H**, where **W** is a $p \times k$ matrix and **H** is $k \times n$ non-negative matrix. We do the matrix decomposition for different rank $k \in \mathcal{K}$ and choose the best k^*, which maximizes the dispersion coefficient. Finally, the estimated batch factors are saved in the consensus matrix $\bar{\mathbf{C}}_{k^*}$. The entire process of the data-adaptive shrinkage and clustering (DASC) is described in Algorithm 1.

2.2 Data-adaptive shrinkage

2.2.1 Non-parametric model to remove biological signal

By definition, **G** can be treated as a graph adjacency matrix. Let $\mathbf{L} \triangleq \mathbf{D} - \mathbf{G}$, where **D** is a diagonal matrix whose elements $D_{ii} = \sum_j G_{ij}$. **L** is the Laplacian matrix of **G** (Chung, 1997). There are two constraints on **U** based on empirical observations of transcriptome data: (i) vector $\mathbf{u}_i$ should be close to $\mathbf{x}_i$ and (ii) $\mathbf{u}_i$ and $\mathbf{u}_j$ should be similar to each other, if $G_{ij} = 1$. These constraints can be formulated within the optimization framework with the following cost function:

$$\min_{\mathbf{U}} J(\mathbf{U}) = \sum_{i=1}^{n} ||\mathbf{x}_i - \mathbf{u}_i||_2^2 + \lambda f(\mathbf{U}), \tag{2}$$

where $\mathbf{x}_i, \mathbf{u}_i \in R^p, \mathbf{U} \in R^{p \times n}$, $f(\mathbf{U})$ is the regularization item and λ is the regularization parameter. There are three forms of the penalty term $f(\mathbf{U})$, which are as follows:

1. $f(\mathbf{U}) = tr(\mathbf{U}\mathbf{L}\mathbf{U}^T) = \sum_{i=1}^{n}\sum_{j=1}^{n} G_{ij} \parallel \mathbf{u}_i - \mathbf{u}_j \|_2^2$,
2. $f(\mathbf{U}) = \sum_{i=1}^{n}\sum_{j=1}^{n} G_{ij} \parallel \mathbf{u}_i - \mathbf{u}_j \|_1$,
3. $f(\mathbf{U}) = \sum_{i=1}^{n}\sum_{j=1}^{n} G_{ij} \parallel \mathbf{u}_i - \mathbf{u}_j \|_2$.

The first penalty function is equivalent to a ridge regression with the ridge penalty defined by experimental conditions (Hastie *et al.*, 2003; Hoerl and Kennard, 1970). It shrinks the data points with similar labels together. Second and third penalty functions are equivalent to data-adaptive shrinkage (Chi and Lange, 2013). The third penalty function has an extra property of providing a smoother clustering path as compared to the L_1 norm penalty function (Hastie *et al.*, 2003).

2.2.2 Solution for data-adaptive shrinkage

Proposition 2.1 Optimization problems (2) with L_1, L_2, L_2^2 penalty functions are convex.

According to Proposition 2.1, the objective function defined in Equation (2) has a *unique global solution* $\mathbf{U}^*$. Moreover, the closed solution is available with L_2^2 penalty. Taking the derivative of the function (2) with L_2^2 penalty, the closed solution for **U** is

$$\mathbf{U}^* = \mathbf{X}(\mathbf{I} + \lambda\mathbf{L})^{-1}. \tag{3}$$

There are no closed solutions when L_1 and L_2 norms are used. Hence, we use numerical algorithms such as ADMM and AMA (Chi and Lange, 2013) to find the solution. We implemented a continuous data-adaptive shrinkage by solving Equation (2) using ADMM and AMA in the R package. However, in some special cases of λ, the closed solution is available. For example, if $\lambda = 0$, $\mathbf{U}^* = \mathbf{X}$.

Proposition 2.2 When λ is sufficiently large, the samples in $\mathbf{u}_i^*$ will give the mean of samples that $G_{ij} = 1$.

2.3 Batch factor estimation

We hypothesize that the hidden batch information can be estimated from the residuals between **X** and $\mathbf{U}^*$. Let us define the batch factor matrix **B** as

$$\mathbf{B} \triangleq \mathbf{X} - \mathbf{U}^*. \tag{4}$$

To estimate the hidden batch factor, we applied semi-NMF (Ding *et al.*, 2010) to **B**. The choice of using semi-NMF over SVD is based on our empirical observation that many batch factors are not orthogonal to each other, hence violating the strict orthogonality assumption of SVD. Semi-NMF is more flexible for the extraction of latent variables and input data with negative values.

We decompose the batch matrix **B** using semi-NMF as

$$\mathbf{B}_{\pm} \approx \mathbf{W}_{\pm}\mathbf{H}_{+}, \tag{5}$$

where $\mathbf{B} \in R^{p \times n}$, $\mathbf{W} \in R_{\pm}^{p \times k}$, $\mathbf{H} \in R_{+}^{k \times n}$. The matrix **W** is the basis matrix and the positive matrix **H** is the batch factor. The batch factors can be divided into k clusters and batch assignments depend on the relative values in each column of **H**. In our implementation, a max operator is applied to each column of **H** to obtain the batch assignments $\mathbf{b}(\lambda, k)$.

For each random initialization, a batch factor will be estimated and saved into a connectivity matrix **C** of size $n \times n$ with entry $C_{ij} = 1$, if samples i and j belong to the same cluster and $C_{ij} = 0$, if they belong to different clusters. To reach a stable clustering assignment independent of the initialization point, consensus matrix $\bar{\mathbf{C}}$ is estimated by averaging multiple runs of semi-NMF (Brunet *et al.*, 2004). The hidden batch factor can be estimated by running a hierarchical clustering on 1 − **C** and cutting the dendrogram at a fixed height.

2.4 Metrics for method evaluation and tuning parameter selection

In cases where the batch label is known, we propose to use two different metrics: (i) purity and (ii) entropy coefficient (Kim and Park, 2007) to evaluate the performance of DASC and other existing methods in the literature. Suppose DASC predicts c clusters (predicted batch labels), while the data has q categories (real batch labels). Then, purity is defined as

$$P = \frac{1}{n}\sum_{k=1}^{c} \max_{1 \le j \le q} \left(n_k^j\right), \tag{6}$$

where n is the number of samples and n_k^j is the number of samples in the cluster k that belong to category $j (1 \le j \le q)$. The value of P ranges from 0 to 1, where one represents the highest accuracy in clustering the samples.

A second metric used to measure the performance of our algorithm is Entropy (Kim and Park, 2007), which is defined as

$$E = -\frac{1}{n \log_2 q} \sum_{k=1}^{c} \sum_{j=1}^{q} n_k^j log_2 \frac{n_k^j}{n_k}, \tag{7}$$

where q is the number of true batch labels and n_k is the size of cluster k. Smaller values of entropy represent better clustering results.

In practice, both the number of hidden factors and its value are unknown. To determine the right tuning parameters, we propose to use the dispersion coefficient summarized using the consensus matrix $\bar{\mathbf{C}}$ for the estimated hidden batch factors. The entries of $\bar{\mathbf{C}}$ reflect the probability that the samples i and j belong to the same cluster. The dispersion coefficient ρ is defined as

$$\rho = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} 4 * \left(\bar{C}_{ij} - \frac{1}{2}\right)^2. \tag{8}$$

The values of ρ ranges between 0 and 1, where larger values represent stable clustering results.

There are two tuning parameters λ and k in DASC. The optimum tuning parameters are identified by searching over a grid of possible combination of λ and k that reaches the maximum dispersion level.

2.5 Computational complexity

The computational complexity of Algorithm 1 comes from two parts: (i) getting the residual matrix $\mathbf{B}$ and (ii) estimating batch factors from $\mathbf{H}$.

For L_2^2 norm penalty, the closed form solution exists. Hence, the total computational complexity is small, $\mathcal{O}(n^2p)$. For L_1 and L_2 norm penalty, the computational complexity of numerical algorithms ADMM and AMA are $\mathcal{O}(n^2p)$ and $\mathcal{O}(np)$, respectively per iteration, when the minimum spanning tree of adjacent matrix $\mathbf{G}$ is used as input. The use of ADMM and AMA induces a much higher computational cost since it could take many iterations to reach convergence for both algorithms.

To find a solution for semi-NMF, $\mathbf{W}$ and $\mathbf{H}$ are alternatively updated until convergence. The time complexity for updating $\mathbf{W}$ and $\mathbf{H}$ are in the order of $m(pnk + nk^2)$ and $m(npk + kp^2 + n^2k)$, respectively, where m is the number of iterations. Therefore, the total time complexity of semi-NMF is $\mathcal{O}(mkp^2)$. To reach convergence, m is typically below 100 in practice.

3 Results

3.1 Datasets description

In our first case study, we used a benchmark dataset from the RNA Sequencing Quality Control (SEQC) project (Su *et al.*, 2014) to test the accuracy of our algorithm. The SEQC consists of four different types of RNA: A (Universal Human Reference RNA), B (Human Brain Reference RNA), C and D, where C and D are mixture of A and B at a defined ratio of 3:1 and 1:3, respectively. RNA-Seq datasets generated using Illumina (ILM) and Life Technology (LIF) were combined across six different sequencing centers. The ILM datasets were generated at the Australian Genome Research (AGR), Beijing Genome Institute (BGI) and Cornell (CNL), whereas the Life Technology datasets were generated at University of Liverpool (LIV), Northwestern University (NWU) and Penn State University (PSU). The entire gene expression dataset consisted of 1396 samples with 43 827 genes. ERCC probes and spike-in mix samples (E and F) were not used in our analysis. Due to its deep coverage, detailed annotation, well-controlled sample preparation and clear batch labels, this large dataset provides a prefect benchmark for batch detection algorithms.

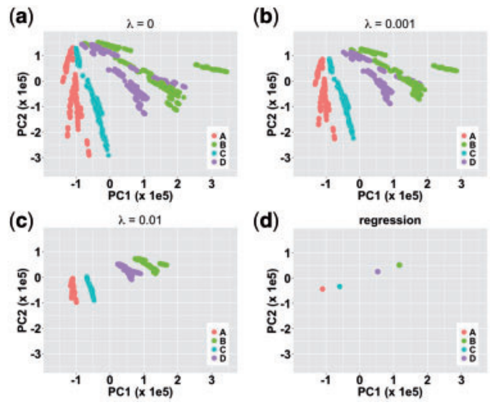

Fig. 1. The regularization path of DASC. Data-adaptive shrinkage was applied to SEQC dataset with 43 827 genes and the results of the shrinkage at multiple levels of λ is plotted using the first two principal components. In contrast, the linear regression results used by sva are also plotted using the same PCA mapping

3.2 Detecting batch effects in SEQC dataset

To demonstrate the importance of the data-adaptive shrinkage step, we analyzed the performance of DASC on SEQC dataset and compared our results to sva and other algorithms.

3.2.1 Regularization path of DASC

To illustrate the difference between our data-adaptive shrinkage method and cell mean regression in sva, we plot the shrinkage track with different parameters of λ in Figure 1. We observed that the samples of same group label (A, B, C, D) clustered together with the increasing amount of λ (Fig. 1). When λ is sufficiently large, the solution of DASC could converge to the mean point of each sample type. This could transform our convex optimization model into a cell mean linear regression model. Cell mean regression model over shrinks the center of each group and may lead to the loss of inter-sample variation in the dataset. In contrast, our model maintains the inter-sample variation, which is important for batch detection.

3.2.2 Both center and platform factors can be estimated by DASC

DASC was able to detect both the major source of batch effects, which was due to sequencing platform and the secondary batch effects due to different sequencing centers (Fig. 2b). For samples from AGR, BGI and CNL, DASC was able to cluster them in the correct group with high accuracy. For LIV, NWU and PSU, DASC was also able to cluster most of the samples into correct groups.

To evaluate the impact of penalty term, DASC was applied to SEQC data without the shrinkage step, it was still able to identify sequencing platforms as the two batches between the samples, however, it failed to identify the difference between sequencing centers (Fig. 2a and Table 1).

Next, we applied DASC with multiple values of λ (Table 1). Our results indicated the performance of DASC is relatively stable with respect to several evaluation metrics including purity, entropy and dispersion. The optimum recovery on batch factors is achieved when the regularization parameter λ is small, ranging from 10^{-4} to 10^{-3}.

When our data-adaptive shrinkage was replaced by linear regression, we observed sub-optimal performance (Table 1) as expected by our theoretic analysis in Section 2.2.2.

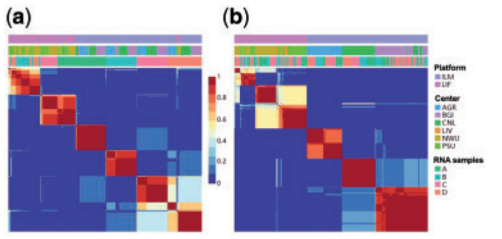

Fig. 2. (**a**) Consensus matrix on the original matrix **X** with $k = 6$ for SEQC dataset using the semi-NMF algorithm. (**b**) Consensus matrix on the batch matrix B with $\lambda = 0.001$ and $k = 6$ for SEQC dataset using the DASC algorithm

Table 1. Performance dependency of DASC ($k = 6$) on various λ values for the matrix $\mathbf{X}^*$ of size 43 827 × 1396

SEQC dataset	DASC				
λ	–	10^{-4}	10^{-3}	10^{-2}	Regression
Purity (platform)	0.966	1.000	1.000	1.000	1.000
Purity (center)	0.414	0.881	0.881	0.770	0.770
Entropy (platform)	0.200	0.000	0.000	0.000	0.000
Entropy (center)	0.623	0.168	0.167	0.248	0.248
Dispersion	0.833	0.883	0.863	0.852	0.862
R (Variance shrinkage ratio)	0.000	0.034	0.259	0.777	1.000

Note: Purity, entropy, dispersion and variance shrinkage ratio were averaged over 30 runs. '–' represents semi-NMF results on **X** without data-adaptive shrinkage step.

3.2.3 Comparison with other approaches

Next, we applied sva (Leek, 2014) on the same SEQC dataset and compared its performances with DASC. We used svaseq() function from the sva package in R (Leek *et al.*, 2012) to detect the optimum number of surrogate variables. Only 1 factor was estimated by sva and it was able to roughly estimate the difference between sequencing platforms, but was not able to detect the variation due to sequencing centers (Fig. 3a).

To further investigate the results of sva, we initialized the argument n.sv to 2, which represents the number of surrogate variables in the svaseq() function. We observed a similar result, but with larger separation on platform differences. Some of the centers were completely overlapped on each other. Our results suggest that sva was able to classify the differences between the platform, but failed to identify the variations due to different sequencing centers (Fig. 3b).

We also compared DASC results against clustering algorithms such as k-means and hierarchical clustering. We chose $k = 6$, which is the same as DASC in k-means clustering. The purity and entropy for $k = 6$ on an ensemble model of sva and k-means clustering were 0.64 and 0.33, respectively, whereas the purity and entropy for $k = 6$ on ensemble model of sva and hierarchical clustering were 0.53 and 0.46, respectively (Table 2). Overall, our analyses demonstrate that DASC is superior to methods compared in this study on SEQC dataset.

3.2.4 DASC is robust to data distribution assumption

RNA-Seq data are often modeled using Poisson or negative binomial distribution. A log transform is used in svaseq (Leek, 2014) to reduce the large variance in RNA-Seq counts. To illustrate the impact of data distribution, we applied sva to a RNA-Seq dataset with and without log transform. Without log transform, the performance of sva deteriorates (Supplementary Fig. S3) as compared to the sva results on log-transformed data (Fig. 3b).

DASC uses a non-parametric data-adaptive model and solves the problem in a unified optimization framework. Depending on the type of data transform, the estimated data center will converge to the generalized mean when $\lambda \to \infty$. For example, DASC will converge to the arithmetic mean or geometric mean when counts or log-transformed counts are used. Empirically, DASC gives similar results with or without log transform (Fig. 2b and Supplementary Fig. S2). Hence, DASC is robust to data distribution assumption.

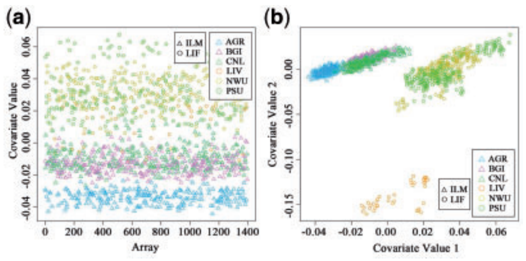

Fig. 3. Comparison of surrogate variables estimated by sva using SEQC dataset from ILM and LIF across six different sequencing centers: AGR, BGI, CNL, LIV, NWU and PSU. (**a**) Surrogate variables from SEQC dataset for n.sv = 1. (**b**) Surrogate variables from SEQC dataset with n.sv = 2

Table 2. Comparison of DASC to other methods: DASC was compared to sva + k-means/hierarchical clustering and PCA + k-means/hierarchical clustering on SEQC dataset

SEQC dataset	Purity (center)	Entropy (center)	Purity (platform)	Entropy (platform)
DASC	**0.77 ± 0.05**	**0.26 ± 0.04**	1.00 ± 0.00	0.00 ± 0.00
sva + k-means	0.64 ± 0.03	0.33 ± 0.01	1.00 ± 0.00	0.00 ± 0.00
sva + HC	0.53 ± 0.00	0.46 ± 0.00	1.00 ± 0.00	0.00 ± 0.00
PCA + k-means	0.43 ± 0.03	0.62 ± 0.03	0.84 ± 0.13	0.53 ± 0.26
PCA + HC	0.50 ± 0.00	0.60 ± 0.00	0.88 ± 0.00	0.51 ± 0.00
P-value	$< 10^{-15}$	$< 10^{-11}$	–	–

Note: HC, hierarchical clustering.

3.3 Detecting unknown batch effects in Topotecan RNA-Seq dataset

Next, we applied DASC to another RNA-Seq dataset (King *et al.*, 2013), where neurons treated with Topotecan, a topoisomerase inhibitor, were compared to those treated with vehicle (Supplementary Fig. S4). In this study, the authors reported that topoisomerase inhibitors reduce the expression of long genes (length >100 kb) that are associated with Autism in cortical neurons.

We identified three different batches using DASC (Fig. 4a). To further evaluate the validity of our estimated batch factor, we investigated the header of the raw FASTQ files. The header encodes for the following information such as sequencing platform id, the run number from the instrument and the flow cell lane. Our initial analysis on the headers showed that RNA-Seq samples were sequenced by two different ILM machines. Additionally, one batch of samples had paired-end reads ($n = 3$ samples each of Vehicle and Topotecan treatments) and other had single-end reads ($n = 2$ samples each of Vehicle and Topotecan treatments). These findings were consistent with the batch factors identified by DASC (Fig. 4a and Supplementary Fig. S5). The third batch represented the samples with low sequencing depth or total number of mapped reads less than 9 million (Fig. 4a and Supplementary Fig. S7). Whereas sva was only able to detect two batches from the dataset that were based on sequencing depth of the samples (Fig. 4b). It failed to detect batches due to platform differences or sequencing type (single-end or paired-end reads).

Moreover, the list of differentially expressed genes by Topotecan is drastically expanded from 182 genes to 1771 genes when we only use the single-end sequencing data (Fig. 5). A small number of differently expressed genes were identified by paired-end sequencing data. This is primarily due to the low sequencing depth on the paired-end sequencing data. In our reanalysis, all reads were aligned using STAR (Dobin *et al.*, 2013) and the raw counts were computed using *quantMode* function in STAR. The raw counts were then normalized and differently expressed genes were generated using DESeq2 (Love *et al.*, 2014) at FDR <0.05, consistent with the original study of King *et al.* (2013).

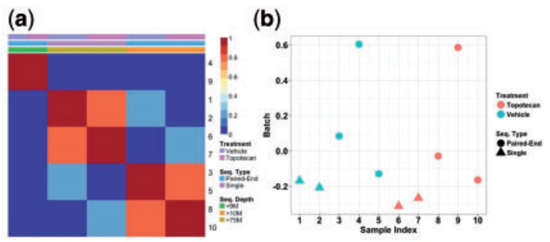

Fig. 4. Batch detection by DASC and sva on Topotecan normalized counts RNA-Seq dataset (GSE43900). (**a**) The batches estimated by DASC algorithm with $\lambda = 1$ and $k = 3$. (**b**) Scatter plot of the batches estimated by sva. The row names used in heatmap are identical to the sample index used in the scatter plot

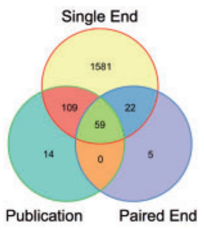

Fig. 5. Overlap between the differentially expressed gene lists from single-end, paired-end samples and the original list from GSE43900 dataset

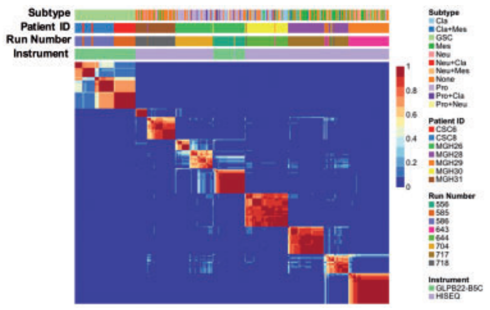

Fig. 6. Detecting run numbers in scRNA-Seq. Consensus matrix on the batch matrix B with $\lambda = 0.001$ and $k = 13$ for GBM dataset (GSE57872) using the DASC algorithm

3.4 Detecting batch effects in single-cell sequencing dataset

To further determine if DASC can detect more subtle batch effects, we applied our method to a single-cell RNA sequencing (scRNA-Seq) dataset (Patel *et al.*, 2014). The dataset consisted of samples from five glioblastoma patients and two gliomasphere cell lines (GSE57872; Supplementary Fig. S8). We used raw counts dataset from recount2 (Collado-Torres *et al.*, 2017). Low-quality samples and genes with average counts less than 30 were removed. After filtering, a total number of 36 295 genes across 531 samples were used for further analysis. The raw-count matrix was then normalized using *computeSumFactors* function (Lun *et al.*, 2016) in the R package scran (McCarthy *et al.*, 2017). We extracted the batch information from the headers of the fastq files for all the 531 samples. Next, DASC and sva were used to detect batch effects in this dataset.

DASC detected 13 different batches (Fig. 6; Supplementary Fig. S9 and Supplementary Table S4) in the dataset, out of which 4 were present among gliomasphere cell line samples. It was able to accurately detect the samples that were generated due to different runs on the sequencer (Fig. 6 and Table 3). Moreover, it detected samples from MGH26 that were sequenced in two different runs (i.e. samples that ran on GLPB22-B5C with a run ID 556 and on HISEQ with a run ID 704). Similar results have been reported by Hicks *et al.* (2017). In contrast, sva estimated 74 surrogate variables and did not detect batches due to platform differences or run number (Supplementary Fig. S10). Furthermore, DASC demonstrated high accuracy in detecting sequencing platform and run number as compared to any of the other methods (Table 3).

Overall, our results demonstrated that DASC can extract and detect more subtle batch effects even in scRNA-Seq data.

4 Discussion

DASC is an effective method to identify hidden batch effects in large consortium datasets. Our method uses data-adaptive shrinkage to get the appropriate estimate of ‘batch-free’ data. The output of DASC is more stable and robust due to the use of consensus matrix and data-adaptive shrinkage method.

From the case study of SEQC dataset, DASC outperforms all the other algorithms compared in this study based on purity and entropy measurement. From the second case study, DASC identified a strong batch effect missed by the original study, which verifies the effectiveness of our method and importance of batch correction. In a scRNA-Seq study, DASC outperformed existing methods in detecting day-to-day sequencing variations.

Table 3. Comparison of DASC to other methods: DASC was compared to sva + *k*-means/hierarchical clustering and PCA + *k*-means/hierarchical clustering on GBM dataset (GSE57872)

GBM dataset	Purity (run no.)	Entropy (run no.)	Purity (platform)	Entropy (platform)
DASC	**0.956 ± 0.005**	**0.068 ± 0.002**	**0.986 ± 0.0005**	**0.055 ± 0.007**
sva + *k*-means	0.525 ± 0.025	0.514 ± 0.035	0.709 ± 0.025	0.856 ± 0.005
sva + HC	0.207 ± 0.025	0.956 ± 0.018	0.709 ± 0.035	0.865 ± 0.015
PCA + *k*-means	0.648 ± 0.037	0.380 ± 0.024	0.775 ± 0.018	0.702 ± 0.03
PCA + HC	0.499 ± 0.021	0.527 ± 0.023	0.709 ± 0.027	0.817 ± 0.018

Note: HC, hierarchical clustering.

Moreover, we showed that the results of DASC is independent of data distribution assumption compared to PCA and sva. Altogether, DASC is a general and flexible algorithm for detecting unknown batch effects. It can be generalized to other omics datasets as well.

5 Conclusions

We presented an unsupervised batch effects detection algorithm via data-adaptive shrinkage and semi-NMF. DASC can be used for detecting batch effects from large and heterogeneous datasets. We have shown the efficiency of DASC in three case studies. In all cases, DASC was able to accurately identify the hidden batch labels and to improve the downstream Bioinformatics analysis. DASC can also be applied to detect batch effects in other types of genomic data where quantitative values are measured. Finally, the batch information can be further used as a covariate in conjunction with other variables of interest for downstream Bioinformatics analysis.

Acknowledgements

We thank R. Gaujoux and C. Seoighe for their help on NMF package (Gaujoux and Seoighe, 2010).

Funding

This work was supported by the Natural Science Foundation of Tianjin, China [15JCYBJC18900] and the National Natural Science Foundation of China [31728013]. This work was also supported by the National Science Foundation [DMS-1263932], the Cancer Prevention and Research Institute of Texas [RP-170387], the National Institute of Health [R01-GM120033], the T.T. and W.F. Chao Foundation, the Robert A. and Renee E. Belfer Family Foundation and the Houston Endowment.

Conflict of Interest: none declared.

References

Akey,J.M. *et al.* (2007) On the design and analysis of gene expression studies in human populations. *Nat. Genet.*, **39**, 17–22.

Belorkar,A. and Wong,L. (2016) GFS: fuzzy preprocessing for effective gene expression analysis. *BMC Bioinformatics*, **17**, 169.

Benito,M. *et al.* (2004) Adjustment of systematic microarray data bases. *Bioinformatics*, **20**, 105–114.

Brunet,J.-P. *et al.* (2004) Metagenes and molecular pattern discovery using matrix factorization. *Proc. Natl. Acad. Sci. USA*, **101**, 4164–4169.

Chi,E.C. and Lange,K. (2013) Splitting methods for convex clustering. *J. Comput. Graph. Statist.*, **212**, 21–44.

Chung,F.R. (1997) *Spectral Graph Theory.* Vol. 92, American Mathematical Society.

Collado-Torres,L. *et al.* (2017) Reproducible RNA-seq analysis using recount2. *Nat. Biotechnol.*, **35**, 319–321.

Cusanovich,D.A. *et al.* (2014) The functional consequences of variation in transcription factor binding. *PLoS Genet.*, **10**, e1004226.

Ding,C. *et al.* (2005) On the equivalence of nonnegative matrix factorization and k-means- spectral clustering. *Factorization.*

Ding,C.H. *et al.* (2010) Convex and semi-nonnegative matrix factorizations. *IEEE Trans. Pattern Anal. Mach. Intell.*, **32**, 45–55.

Dobin,A. *et al.* (2013) STAR: ultrafast universal RNA-seq aligner. *Bioinformatics*, **29**, 15–21.

Feingold,E.A. (2004) The ENCODE (Encyclopedia of DNA Elements) project. *Science*, **306**, 636–640.

Gagnon-Bartsch,J.A. and Speed,T.P. (2012) Using control genes to correct for unwanted variation in microarray data. *Biostatistics*, **13**, 539–552.

Gaujoux,R. and Seoighe,C. (2010) A flexible R package for nonnegative matrix factorization. *BMC Bioinformatics*, **11**, 1.

Gilad,Y. and Mizrahi-Man,O. (2015) A reanalysis of mouse encode comparative gene expression data. *F1000Res.*, **4**, 121.

Hastie,T. *et al.* (2003) *The Elements of Statistical Learning*. Springer.

Hicks,S.C. *et al.* (2017) Missing data and technical variability in single-cell RNA-sequencing experiments. *bioRxiv*, 025528.

Hoerl,A.E. and Kennard,R.W. (1970) Ridge regression: biased estimation for nonorthogonal problems. *Technometrics*, **12**, 55–67.

Hornung,R. *et al.* (2016) Improving cross-study prediction through addon batch effect adjustment or addon normalization. *Bioinformatics*.

Johnson,W.E. *et al.* (2007) Adjusting batch effects in microarray expression data using empirical Bayes methods. *Biostatistics*, **8**, 118–127.

Kim,H. and Park,H. (2007) Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis. *Bioinformatics*, **23**, 1495–1502.

King,I.F. *et al.* (2013) Topoisomerases facilitate transcription of long genes linked to autism. *Nature*, **501**, 58–62.

Lazar,C. *et al.* (2012) Batch effect removal methods for microarray gene expression data integration: a survey. *Brief. Bioinform.*, bbs037.

Lee,D.D. (2001) Algorithms for non-negative matrix factorization. *Adv. Neural Inform. Process. Syst.*, **13**, 556–562.

Lee,D.D. and Seung,H.S. (1999) Learning the parts of objects by non-negative matrix factorization. *Nature*.

Leek,J.T. (2014) svaseq: removing batch effects and other unwanted noise from sequencing data. *Nucleic Acids Res.*, **42**, gku864.

Leek,J.T. and Storey,J.D. (2007) Capturing heterogeneity in gene expression studies by surrogate variable analysis. *PLoS Genet.*, **3**, 1724–1735.

Leek,J.T. *et al.* (2010) Tackling the widespread and critical impact of batch effects in high-throughput data. *Nat. Rev. Genet.*, **11**, 733–739.

Leek,J.T. *et al.* (2012) The sva package for removing batch effects and other unwanted variation in high-throughput experiments. *Bioinformatics*, **28**, 882–883.

Love,M.I. *et al.* (2014) Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biol.*, **15**, 1.

Lun,A.T. *et al.* (2016) Pooling across cells to normalize single-cell RNA sequencing data with many zero counts. *Genome Biol.*, **17**, 75.

Luo,J. *et al.* (2010) A comparison of batch effect removal methods for enhancement of prediction performance using MAQC-II microarray gene expression data. *Pharmacogenomics J.*, **10**, 278–291.

McCarthy,D.J. *et al.* (2017) Scater: pre-processing, quality control, normalization and visualization of single-cell RNA-seq data in R. *Bioinformatics*, **33**, 1179–1186.

Mostafavi,S. *et al.* (2013) Normalizing RNA-sequencing data by modeling hidden covariates with prior knowledge. *PLoS One*, **8**, e68141.

Patel,A.P. *et al.* (2014) Single-cell RNA-seq highlights intratumoral heterogeneity in primary glioblastoma. *Science*, **344**, 1396–1401.

Reese,S.E. *et al.* (2013) A new statistic for identifying batch effects in high-throughput genomic data that uses guided principal component analysis. *Bioinformatics*, **29**, 2877–2883.

Risso,D. *et al.* (2014) Normalization of RNA-seq data using factor analysis of control genes or samples. *Nat. Biotechnol.*, **32**, 896–902.

Scherer,A. (2009) *Batch Effects and Noise in Microarray Experiments: Sources and Solutions*, Vol. **868**, John Wiley & Sons.

Stegle,O. *et al.* (2010) A Bayesian framework to account for complex non-genetic factors in gene expression levels greatly increases power in eQTL studies. *PLoS Comput. Biol.*, **6**, e1000770.

Stein,C.K. *et al.* (2015) Removing batch effects from purified plasma cell gene expression microarrays with modified combat. *BMC Bioinformatics*, **16**, 1.

Su,Z. *et al.* (2014) A comprehensive assessment of RNA-seq accuracy, reproducibility and information content by the sequencing quality control consortium. *Nat. Biotechnol.*, **32**, 903–914.

Teschendorff,A.E. *et al.* (2011) Independent surrogate variable analysis to deconvolve confounding factors in large-scale microarray profiling studies. *Bioinformatics*, **27**, 1496–1505.

Tung,P.-Y. *et al.* (2016) Batch effects and the effective design of single-cell gene expression studies. *bioRxiv*, 062919.

Weinstein,J.N. *et al.* (2013) The Cancer Genome Atlas Pan-Cancer analysis project. *Nat. Genet.*, **45**, 1113–1120.