BEclearCL Documentation

David Rasp[*1]

[1]Center for Bioinformatics, Saarland University, Saarbruecken, Germany

[*]David.J.Rasp@gmail.com

2019-06-06

Abstract

This command-line tool provides functions to detect and correct for batch effects in DNA methylation data. The core function for the data imputation is based on latent factor models (Candès and Recht 2009) and can also be used to predict missing values in any other matrix containing real numbers. In this documentation we guide you through the installation and usage of it. For the corresponding R-package visit *BEclear* (Akulenko, Merl, and Helms 2016).

1 Getting Started

You have the choice to either download the latest tarball from https://github.com/David-J-R/BEclear-CL/archive/master.zip and expand it or clone the repository with the following command:

```
git clone git@github.com:David-J-R/BEclear-CL.git
```

For using the command-line version of BEclear, you need to have R with a version of at least 3.5 installed on your system. Furthermore you need the following R packages:

- *BEclear* (>= 2.0)
- *optparse*

To install them you can either run our provided `install_requirements.R` script or install them by typing the following in your R environment:

```
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")

BiocManager::install("BEclear")

install.packages("optparse")
```

To make the command-line tool executable, you need to call:

```
chmod +x BEclearCL.R
```

2 Usage

2.1 Example

The data-set used in this example is from The Cancer Genome Atlas (Cancer Genome Atlas Research Network et al. 2013).

```
./BEclearCL.R -f testDataSet.txt -o testImputed.txt -s testSamples.txt
```

2.2 Parameters

In the following we will document the command-line parameters of the tool and the format of input and output. The parameters will be explained consecutively:

- -f *filename*, –file=*filename*

  Where *filename* is the path to the input file containing a matrix with real numbers. The matrix can already contain missing values, named `NA`. The features must be represented by the rows and the samples by columns. For the columns the file has to contain a header. For the rows it can, but does not need to contain an identifier in the first column. Columns have to be tab-seperated.

  If you do not set this parameter, BEclearCL expects this input from STDIN, so that it can be integrated into pipelines.

- -o *filename*, –out=*filename*

  Where *filename* is the path where the output should be written to. The format is the same as for the input.

  If you do not set this parameter, BEclearCL prints the output to STDOUT.

- -d, –detection

  If this parameter is set batch effects are detected. Otherwise the tool only does the data imputation and expects the input to already contain missing values, named `NA`.

- -s *filename* –samples=*filename*

  Where *filename* is the path to the input file containing assignment of samples to batches. It must contain a column named sample_id with the sample ID (same as in the input data) and one named batch_id with the coresponding IDs for the batches.

  If this parameter is not provided each sample is tested against all other samples to find and correct for sample specific effects.

- -c *number*, –cores=*number*

  Where *number* represents the *number* of cores you want the tool to use. It uses a snow backend for doing so.

- -b *filename*, –BEscore=*filename*

  Where *filename* is the path where the table containing the calculated batch scores should be written to.

  If not set, the file is not written.

- -v, –verbose

  If set, the tool will write output about its progress to the STDOUT. By default it does not.

- -r, –replace

  If values outside of the interval between 0 and 1 should be cropped to 0 or 1, which can be necessary depending on your type of data. For example for DNA methylation data.

  By default this is not done.

3 Theoretical Background

In this section we explain the theoretical background behind the method.

3.1 Detection of batch effects

For the detection of batch effects we calculate the median difference between the beta values of a gene in a batch and the values of this gene in all other batches. Furthermore we use a non-parametric Kolmogorov-Smirnov test (`ks.test`) to compare the distribution of the beta value for this gene in the batch and the other batches.

If one gene in a batch has a p-value determined by the `ks.test` of less or equal 0.01 and a median difference of greater or equal 0.05 it is considered batch effected. By default the p-values are adjusted by the false discovery rate developed by Benjamini and Hochberg (1995).

3.2 Imputation of missing values

For the imputation of missing values we use a slightly modified version of the stochastic gradient descent method described by Koren, Bell, and Volinsky (2009). In this section we will describe our implementation of this method and how to use it.

We assume that our complete data matrix D_{ij} can be described by the effects of a matrix L_i, which represents the effect of the features (genes in our case) and a matrix R_j describing the effect of the samples in the following way:

$$D_{ij} = L_i^T \times R_j. \tag{1}$$

The method can either be run on the complete data set or the data set can be divided into blocks on which the method is applied. This division into blocks allows for parallelisation of the method, which can be useful to speed up the process. We have found that a block-size of 60x60 works well(Akulenko, Merl, and Helms 2016).

The error for each block is calculated in the following way:

$$errorMatrix_{ij} = Block_{ij} - L_i^T \times R_j. \tag{2}$$

We try to minimize the following loss function through a gradient descent:

$$min_{L,R} \sum_{ij \in K} (errorMatrix_{ij}^2) + \lambda \times (\|L_i\|_F^2 + \|R_j\|_F^2). \tag{3}$$

Where K is the set of tuples (i, j) for which the value is present. λ is the penalty coefficient, which controls how restrictive the selection of variables should be. The default of λ is 1.

Another coefficient γ controls the size of the step by which the two matrices L_i and R_j are modified. It is initialized by default with 0.01 and its value changes during the iterations (epochs).

For the first iteration the matrices L_i and R_j are filled with random values generated by the `rnorm` function from the *stats* package and the initial loss and error matrix are calculated.

Then for each iteration the following is done:

- L_i and R_j are modified proportional by γ through the following calculation:
    - $$L_i = L_i + 2 \times \gamma \times (errorMatrix_{ij} \times R_j - \lambda \times L_i). \tag{4}$$
    - $$R_j = R_j + 2 \times \gamma \times (errorMatrix_{ij} \times L_i - \lambda \times R_j). \tag{5}$$
- Then the new error matrix and loss are calculated.
- If the old loss is smaller than the new one:
    - $\gamma = \gamma \div 2.$
- Else:
    - $\gamma = \gamma \times 1.05.$

The L_i and R_j matrices at the end of the last iteration are then used to impute the missing data. The default number of iterations is 50.

References

Akulenko, Ruslan, Markus Merl, and Volkhard Helms. 2016. "BEclear: Batch effect detection and adjustment in DNA methylation data." *PLoS ONE* 11 (8): 1–17. https://doi.org/10.1371/journal.pone.0159921.

Benjamini, Yoav, and Yosef Hochberg. 1995. "Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing." *Journal of the Royal Statistical Society. Series B (Methodological)* 57 (1). [Royal Statistical Society, Wiley]: 289–300. http://www.jstor.org/stable/2346101.

Cancer Genome Atlas Research Network, John N Weinstein, Eric A Collisson, Gordon B Mills, Kenna R Mills Shaw, Brad A Ozenberger, Kyle Ellrott, Ilya Shmulevich, Chris Sander, and Joshua M Stuart. 2013. "The Cancer Genome Atlas Pan-Cancer analysis project." *Nature Genetics* 45 (10): 1113–20. https://doi.org/10.1038/ng.2764.

Candès, Emmanuel J., and Benjamin Recht. 2009. "Exact Matrix Completion via Convex Optimization." *Foundations of Computational Mathematics* 9 (6): 717–72. https://doi.org/10.1007/s10208-009-9045-5.

Koren, Yehuda, Robert Bell, and Chris Volinsky. 2009. "Matrix Factorization Techniques for Recommender Systems." *Computer* 42 (8): 30–37. https://doi.org/10.1109/MC.2009.263.