BEclear - Correct for batch effects in DNA methylation data

Markus Merl, David Rasp*
email: Davod.j.Rasp@gmail.com

Modified: September 28, 2018 Compiled: October 30, 2018

This example guides to the BEclear package to correct for batch effects in DNA methylation data. The package provides some functions to detect and correct such batch effects. The core function BEclear is based on Latent Factor Models [1] and can also be used to predict missing values in any other matrix containing real numbers.

Contents

1 Introduction

The individual chapters guide through the available methods of the BEclear package in a logical order following an example of correcting some batch affected DNA methylation data. This article should only give a small tutorial, more details about the individual methods can always be found in the help sections of the BEclear package, e.g. through typing "?calcMedians" on the R console. To work with the methods contained in the BEclear package, a matrix or data.frame with genes as rownames and samples as column names as well as a samples data.frame with the first column named "sample_id" and the second column named "batch_id" is needed as input. To run an example workflow, we first have to load the example data into our workspace. A matrix with 250 genes as rows and 40 samples as columns containing beta values, as well as a data.frame containing the samples and corresponding batch names emerges. To get an intuition of how the inuput data for the methods should look like, we print out the first 10 rows and five columns of the matrix on the screen and also the first 10 rows of the samples data.frame.

```
> library(BEclear)
> data(BEclearData)
> ex.data[1:10,1:5]
```

```
               s20        s21        s22        s23        s24
ACSM3    0.22978729 0.21628728 0.20719868 0.23292691 0.21205930
ADAM28   0.34350641 0.45796065 0.37496248 0.42052354 0.39337616
ADCK1    0.21761416 0.21203847 0.21308033 0.21713120 0.21438141
AFTPH    0.03149421 0.03067515 0.03035862 0.02930077 0.02363116
AKAP7    0.12652223 0.08984297 0.16380988 0.10872611 0.11501186
ANKRD24  0.05164166 0.04273068 0.03712615 0.04343010 0.04302312
ANKRD44  0.34317757 0.32560143 0.27817751 0.31322486 0.29840699
```

```
ANKS4B   0.57125501 0.54677390 0.52091911 0.60753275 0.54190976
APCDD1   0.48614910 0.42010326 0.44058872 0.52759982 0.44388208
APOBEC3G 0.36366491 0.33017162 0.37493338 0.35095431 0.44060871
```

```
> ex.samples[1:10,]
```

```
   sample_id batch_id
1         s20     b109
2         s21     b109
3         s22     b109
4         s23     b109
5         s24     b117
6         s25     b117
7         s26     b117
8         s27     b117
9         s28     b117
10        s29     b117
```

The beta values stored in the ex.data matrix are obtained from level 3 BRCA data from the TCGA portal [2]. Generally, beta values are claculated by dividing the methylated signal by the sum of the unmethylated and methylated signals from a DNA methylation microrarray. In the level 3 TCGA data, this calculation has already been done. The sample data used here contains averaged beta values of probes that belong to promoter regions of single genes. Another possibility would be to use beta values of single probes, whereby the probe names should then be used instead of the gene names as rownames of the matrix.

2 calcPvalues

Next, we want to know if our data is batch affected or not. We therefore calculate false discovery rate adjusted p-values by Kolmogorov-Smirnov (ks) test for every gene in every batch. We could also use any other adjustment method contained in the p.adjust function of the R stats package. If possible, we can run the method in parallel mode which is also the default of this paramter. If the method should be run on a machine with just one core, the parallel parameter should simply be set to FALSE. The resulting p-values tell us if a gene from a certain batch contains a batch effect or not. Thereby, every gene with a p-value below 0.01 is assumed as batch affected.

```
> library(data.table)
> samples <- data.table(ex.samples)
> data <- data.table(feature=rownames(ex.data), ex.data)
>     data <- melt(data = data, id.vars = "feature", variable.name = "sample",
+                  value.name = "beta.value")
>     setkey(data, "feature", "sample")
> pvals <- calcPvalues(data, samples, adjusted=TRUE, method="fdr")
```

```
INFO [2018-10-30 20:42:50] Calculating the p-values for 10 batches
INFO [2018-10-30 20:42:53] Adjusting p-values
```

Returned is a data.frame containing p-values for all 10 batches and for every gene. We print out the p-values for 10 genes and 4 batches:

```
> pvals[210:220,5:8]
```

```
                b136      b142      b155       b72
SLC17A2 2.124143e-01 0.7375169 0.8638664 0.3228734
SLC38A4 5.932453e-02 0.8830409 0.5966599 1.0000000
SMARCD2 3.222988e-04 0.5651447 0.5651447 0.6495726
SMOC1   5.932453e-02 0.8530364 0.8450292 0.4375000
SMOX    9.201325e-04 0.4531502 0.4592892 0.2500000
SNX21   7.385032e-06 0.9750337 0.9750337 1.0000000
SOD1    1.032961e-04 0.2955466 0.7605263 0.7605263
SPC24   3.222988e-04 0.1113360 0.1842105 0.7777778
SPINK2  3.222988e-04 0.8905788 0.8905788 0.8905788
SSRP1   3.222988e-04 0.9180162 0.9163293 0.9163293
SYCE2   3.222988e-04 0.9282471 0.9282471 1.0000000
```

We can see that most of the p-values are beyond 0.01, but some of the genes have p-values below our threshold value, e.g. the "SPINK2", "SNX21" or "SMOX" genes in batch b136, which all have a p-values < 0.0001 from which we conclude that the beta values of these genes corresponding to batch b136 does not follow the typical distribution for the beta values in the other batches which suggests a batch effect.

3 calcMedians

To see to which extend the found genes from the ks-test are affected by the batch effect, we calculate the median difference (mdif) values for every gene

and every batch in a similar way. We consider all genes with mdif values beyond or equal to 0.05 as batch affected since values beyond this threshold would already make a biological difference in the methylation level. Since beta values are bounded between 0 and 1, this threshold indicates more than 5% of the overall deviation of this [0;1] interval. Returned is again a data.frame containing mdif values for every gene in every batch. We print out again the mdif values for the same genes and batches as before.

```
> mdifs <- calcMedians(data, samples)
```

```
INFO [2018-10-30 20:42:53] Calculating the medians for 10 batches
```

```
> mdifs[210:220,5:8]
```

```
          b136         b142          b155           b72
210 0.01326067  0.011684347  0.0064376902 -0.0568709256
211 0.05427230 -0.024569384  0.0442606951 -0.0023979164
212 0.04377473  0.007261480  0.0015304622 -0.0049537648
213 0.02004716 -0.006187701  0.0088631859  0.0325320528
214 0.01276110  0.001521808  0.0016256588 -0.0139147353
215 0.03979558  0.009434266  0.0025460079  0.0001944529
216 0.04040058  0.010059310 -0.0018819630 -0.0107954042
217 0.04880502  0.005595816  0.0038687584 -0.0016017197
218 0.43006149 -0.028696780  0.0164482151 -0.0179958882
219 0.04399476  0.002045536  0.0015797555 -0.0038052277
220 0.03488399  0.002955042  0.0004563072 -0.0001072030
```

We can see that most of the mdif values are smaller than 0.05, but some of the genes have mdif values beyond this threshold value, especially the "SPINK2" gene in batch b136, which has a mdif value of ca. 0.43 which, together with the small pvalue, strongly indicates a batch effect for this gene.

4 calcSummary

Now, we summarize the results of the p-value and mdif calculations.

```
> summary <- calcSummary(medians=mdifs, pvalues=pvals)
```

```
INFO [2018-10-30 20:42:53] Generating a summary table
```

This method simply lists all genes that have p-values smaller than or equal to 0.01 and mdif values greater than 0.05, together with the corresponding batch number in a data.frame. Now we have a list of all genes supposed to be batch affected. Let us print the first 10 rows of this summary:

```
> summary[1:10,]
```

```
   gene batch     median       pvalue
1     2  b136 0.25390182 3.222988e-04
2     5  b136 0.22362551 2.975248e-05
3     7  b136 0.25784823 2.410328e-03
4     9  b136 0.20783921 1.557227e-06
5    13  b136 0.36590731 1.032961e-04
6    16  b136 0.23561895 5.885965e-03
7    19  b136 0.06133987 4.600662e-04
8    28  b136 0.34424122 3.222988e-04
9    41  b136 0.26502460 5.885965e-03
10   44  b136 0.23760637 5.885965e-03
```

We can see that all of the affected genes are from batch b136. Although we would define this batch as batch affected, some genes could also be found in other batches which are just slightly affected. We therefore would not talk about a batch effect, but the beta values of these found genes can also be corrected. When looking for such genes, we can find 1 in batch b117 and 2 in batch b61.

```
> summary[summary$batch != "b136",]
```

```
[1] gene   batch  median pvalue
<0 rows> (or 0-length row.names)
```

Overall, we found 54 genes with adjusted p-values below 0.01 and mdif values beyond or equal to 0.05. 51 of these genes were found in batch b136. Alltogether, this strongly indicates a batch effect.

5 calcScore

Outgoing from the summary, we can calculate a score that tells us for every batch if it contains a batch effect or not:

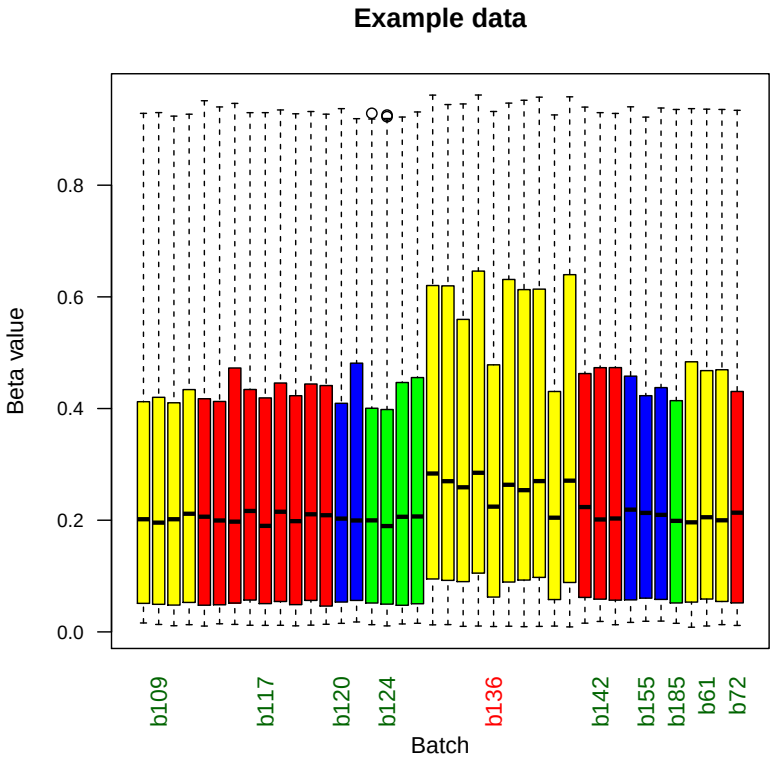
```
> score <- calcScore(ex.data, ex.samples, summary, dir=getwd())

INFO [2018-10-30 20:42:53] Calculating the scores for 10 batches

> score

   batch 0.05 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 BEscore
1   b109    0   0   0   0   0   0   0   0   0   0   0.000
2   b117    0   0   0   0   0   0   0   0   0   0   0.000
3   b120    0   0   0   0   0   0   0   0   0   0   0.000
4   b124    0   0   0   0   0   0   0   0   0   0   0.000
5   b136   10   2  31   7   1   0   0   0   0   0   0.752
6   b142    0   0   0   0   0   0   0   0   0   0   0.000
7   b155    0   0   0   0   0   0   0   0   0   0   0.000
8    b72    0   0   0   0   0   0   0   0   0   0   0.000
9   b185    0   0   0   0   0   0   0   0   0   0   0.000
10   b61    0   0   0   0   0   0   0   0   0   0   0.000
```

The method lists the number of affected genes in terms of mdif and p-values for every gene and delivers us a so called BEscore that tells us if we should correct the data for batch effects or not. The BEscore for batch b136 is 0.752 and tells us that we should correct the data. Additionally, the scoring table is stored as .RData file in the specified directory. As default value, the current working direcotry is used. Details about the scoring system can be found in the documentation of this method.

6 makeBoxplot

We can also visualize the result of the calculated score through a color-coded boxplot

```
> makeBoxplot(ex.data, ex.samples, score, bySamples=TRUE,
+         col="standard", main="Example data", xlab="Batch",
+         ylab="Beta value", scoreCol=TRUE)
```



The boxplot shows the example data separated by samples. The batch numbers are shown on the x-axis in a color that denotes if the batch is affected (red) or not (green) or if there is a slight batch affect assumed (yellow). We can easily recognize that the boxes of batch b136 behave different compared to the other batches.

7 clearBEgenes

Now we decided to correct the data for the found batch effect. Therefore we have to set all previously found affected beta values to NA:

```
> cleared.data <- clearBEgenes(ex.data, ex.samples, summary)
```

```
INFO [2018-10-30 20:42:53] Removing values with batch effect:
INFO [2018-10-30 20:42:53] 510 values ( 5.1 % of the data) set to NA
```

Returned is the input matrix with NA values as defined by the summary.

8 countValuesToPredict

If we already have a matrix with missing entries (not necessarily beta values), we can use this method to count the NA values within the matix:

```
> counted <- countValuesToPredict(cleared.data)

INFO [2018-10-30 20:42:53] 510 values ( 5.1 % of the data) set to NA
```

9 BEclear

This method performs the batch effect correction by predicting all the missing entries we formerly set to NA in our input matrix and returns a matrix that contains all beta values from the original input matrix, together with predicted beta values for the entries formerly set to NA. The correction is done by performing matrix completion using Latent Factor Models based on matrix factorization [1,3]:

```
> corrected.data <- BEclear(cleared.data, rowBlockSize=60, colBlockSize=60,
+                           epochs=50, outputFormat="RData", dir=getwd())
```

The result of the prediction can easily be seen when we again use the makeBoxplot method:

```
> makeBoxplot(corrected.data, ex.samples, score, bySamples=TRUE,
+         col="standard", main="Corrected example data",
+         xlab="Batch", ylab="Beta value", scoreCol=FALSE)
```

For more details about the prediction and the further parameters please read the documentation of this method. Note that the corrected data is also stored as .Rdata file in the specified directory.

10 findWrongValues & replaceWrongValues

Sometimes during the prediction, it can happen that values beyond the boundaries of beta values are returned, that means values smaller than zero or greater than one. findWrongValues simply returns a list of these values, together with the position in the output matrix, replaceWrongValues corrects these by simply setting the wrong values to zero or one, respectively. Since

we cannot guarantee that we get wrong values during the prediction with the example data, we forego these methods for now. Note that these methods are especially designed for the prediction of DNA methylation data.

11 correctBatchEffect

This method performs most of the previously introduced methods step by step in a logical order to simply correct the input data on the basis of the calculated score or not.

```
> result <- correctBatchEffect(ex.data, ex.samples,
+         parallel=TRUE, cores=4, adjusted=TRUE,
+         method="fdr", rowBlockSize=60, colBlockSize=60,
+         epochs=50, outputFormat="RData", dir=getwd())
```

Returned is a list containing the complete output of the previously introduced methods, e.g. result$medians returns the mdif values containing data.frame. For details see again the documentation of this method.

12 Conclusions

In this tutorial, we have followed the whole procedure of detecting and correcting a batch effect in DNA methylation data by using the methods from the BEclear package. This document is intended to give a first overview about the functionality of the package. More details and references can be found in the corresponding documentation.

13 References

[1] E. Candes, B. Recht, Exact matrix completion via convex optimization, Communications of the ACM, 55(6), p. 111-119, 2012.
[2] http://cancergenome.nih.gov/
[3] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, IEEE Computer, 42(8), p. 30-37, 2009.