Package 'BEclear'

May 13, 2024

Type Package

Title Correction of batch effects in DNA methylation data

Version 2.20.0

Date 2023-04-06

Description Provides functions to detect and correct for batch effects in DNA methylation data. The core function is based on latent factor models and can also be used to predict missing values in any other matrix containing real numbers.

License GPL-3

SystemRequirements C++11

Depends BiocParallel (>= 1.14.2)

Imports futile.logger, Rdpack, Matrix, data.table (>= 1.11.8), Rcpp, abind, stats, graphics, utils, methods, dixonTest, ids

Suggests testthat, BiocStyle, knitr, rmarkdown, pander, seewave

VignetteBuilder knitr

biocViews BatchEffect, DNAMethylation, Software, Preprocessing, StatisticalMethod

RoxygenNote 7.2.2

RdMacros Rdpack

Encoding UTF-8

URL https://github.com/uds-helms/BEclear

BugReports https://github.com/uds-helms/BEclear/issues

LinkingTo Rcpp

git_url https://git.bioconductor.org/packages/BEclear

git_branch RELEASE_3_19

git_last_commit 56d8c71

git_last_commit_date 2024-04-30

Repository Bioconductor 3.19

Date/Publication 2024-05-13

Author Livia Rasp [aut, cre] (<https://orcid.org/0000-0003-0164-2163>),
Markus Merl [aut],
Ruslan Akulenko [aut]

Maintainer Livia Rasp <livia.rasp@gmail.com>

Contents

`BEclear-package` *Correction of batch effects in DNA methylation data*

Description

Provides some functions to detect and correct for batch effects in DNA methylation data. The core function `correctBatchEffect` is based on Latent Factor Models and can also be used to predict missing values in any other matrix containing real numbers.

Details

BEclear-package

`correctBatchEffect`: The function combines most functions of the `BEclear-package` to one. This function performs the whole process of searching for batch effects and automatically correct them for a matrix of beta values stemming from DNA methylation data.
`correctBatchEffect`: This function predicts the missing entries of an input matrix (NA values) through the use of a Latent Factor Model.
`calcBatchEffects`: Compares the median value of all beta values belonging to one batch with the median value of all beta values belonging to all other batches. Returns a matrix containing this median difference value for every gene in every batch, columns define the batch numbers, rows the gene names.
And compares the distribution of all beta values corresponding to one batch with the distribution of all beta values corresponding to all other batches and returns a p-value which defines if the distributions are the same or not.
`calcSummary`: Summarizes the results of the `calcBatchEffects` function
`calcScore`: Returns a table with the number of found genes with found p-values less or equal to 0.01 and median values greater or equal to 0.05. A score is calculated depending on the number of found genes as well as the magnitude of the median difference values, this score is divided by the overall number of genes in the data and returned as "BEscore". See the methods details for further information and details about the score calculation.
`makeBoxplot`: A simple `boxplot` is done with boxes either separated by batches or by samples and describe the five number summary of all beta values corresponding to a batch or a sample, respectively. The batch_ids are shown on the x-axis with a coloring corresponding to the BEscore.
`clearBEgenes`: A function that simply sets all values to NA which were previously found by median value comparison and p-value calculation and are stored in a summary. The summary defines which values in the data matrix are set to NA.
`countValuesToPredict`: Simple function that counts all values in a matrix which are NA
`findOutsideValues`: A method which lists values below 0 or beyond 1 contained in the input matrix. These entries are stored in a data.frame together with the corresponding row and column position of the matrix.
`replaceOutsideValues`: A method which replaces values below 0 or beyond 1 contained in the input matrix. These entries outside the boundaries are replaced by 0 or 1, respectively.

Author(s)

Ruslan Akulenko, Markus Merl, Livia Rasp

References

Akulenko R, Merl M, Helms V (2016). "BEclear: Batch effect detection and adjustment in DNA methylation data." *PLoS ONE*, **11**(8), 1–17. ISSN 19326203, doi:10.1371/journal.pone.0159921, `http://www.ncbi.nlm.nih.gov/pubmed/27559732`.

See Also

Useful links:

- `https://github.com/uds-helms/BEclear`
- Report bugs at `https://github.com/uds-helms/BEclear/issues`

Examples

```
data(BEclearData)
## Calculate the batch effects
batchEffects <- calcBatchEffects(data = ex.data, samples = ex.samples,
adjusted = TRUE, method = "fdr")
med <- batchEffects$med
pvals <- batchEffects$pval

## Summarize p-values and median differences for batch affected genes
sum <- calcSummary(medians = med, pvalues = pvals)

## Calculates the score table
score.table <- calcScore(data = ex.data, samples = ex.samples, summary = sum)

## Simple boxplot for the example data separated by batch
makeBoxplot(
  data = ex.data, samples = ex.samples, score = score.table,
  bySamples = FALSE, main = "Some box plot"
)

## Simple boxplot for the example data separated by samples
makeBoxplot(
  data = ex.data, samples = ex.samples, score = score.table,
  bySamples = TRUE, main = "Some box plot"
)

## Sets assumed batch affected entries to NA
cleared <- clearBEgenes(data = ex.data, samples = ex.samples, summary = sum)
## Counts and stores number of entries to predict
numberOfEntries <- countValuesToPredict(data = cleared)
## Not run:
## Predicts the missing entries
predicted <- imputeMissingData(data = cleared)

## Find predicted entries outside the boundaries
outsideEntries <- findOutsideValues(data = predicted)

## Replace predicted entries outside the boundaries
corrected <- replaceOutsideValues(data = predicted)

## End(Not run)
```

BEclear example methylation data

Example data set for the BEclear-package

Description

Example data set for the BEclear-package

Usage

```
data(BEclearData)
```

Format

An example data matrix that is filled with beta values originally stemming from breast cancer data from the TCGA portal [1], colnames are sample ids, rownames are gene names. Generally, beta values are calculated by dividing the methylated signal by the sum of the unmethylated and methylated signals from a DNA methylation microrarray. 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.

References

Cancer Genome Atlas Research Network, Weinstein JN, Collisson EA, Mills GB, Shaw KRM, Ozenberger BA, Ellrott K, Shmulevich I, Sander C, Stuart JM (2013). "The Cancer Genome Atlas Pan-Cancer analysis project." *Nature genetics*, **45**(10), 1113–20. ISSN 1546-1718, doi:10.1038/ng.2764, http://www.ncbi.nlm.nih.gov/pubmed/24071849http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3919969.

`BEclear example sample data`

Example data set for the BEclear-package

Description

Example data set for the BEclear-package

Usage

```
data(BEclearData)
```

Format

An example data frame containing a column for the sample id and a column for the corresponding batch id, stemming from breast cancer data from the TCGA portal [1]

References

Cancer Genome Atlas Research Network, Weinstein JN, Collisson EA, Mills GB, Shaw KRM, Ozenberger BA, Ellrott K, Shmulevich I, Sander C, Stuart JM (2013). "The Cancer Genome Atlas Pan-Cancer analysis project." *Nature genetics*, **45**(10), 1113–20. ISSN 1546-1718, doi:10.1038/ng.2764, http://www.ncbi.nlm.nih.gov/pubmed/24071849http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3919969.

`calcBatchEffects` *Calculate the Batch Effects in a given data set*

Description

Calculates for each gene in every batch the median distance to the other batches and the p-value resulting from the Kolmogorov-Smirnov test.

Usage

```
calcBatchEffects(data, samples, adjusted=TRUE, method="fdr",
BPPARAM=SerialParam())
```

Arguments

`data`	a `data.table` with one column indicating the sample, one the features and a value column indicating the beta value or a matrix with rows as features and columns as samples
`samples`	data frame with two columns, the first column has to contain the sample numbers, the second column has to contain the corresponding batch number. Colnames have to be named as "sample_id" and "batch_id".
`adjusted`	should the p-values be adjusted or not, see "method" for available adjustment methods.
`method`	adjustment method for p-value adjustment (if TRUE), default method is "false discovery rate adjustment", for other available methods see the description of the used standard R package `p.adjust`.
`BPPARAM`	An instance of the `BiocParallelParam-class` that determines how to parallelisation of the functions will be evaluated.

Details

calcBatchEffects

1. medians Compares the median value of all beta values belonging to one batch with the median value of all beta values belonging to all other batches. Returns a matrix containing this median difference value for every gene in every batch, columns define the batch numbers, rows the gene names.
2. p-values Compares the distribution of all beta values corresponding to one batch with the distribution of all beta values corresponding to all other batches and returns a p-value which defines if the distributions are the same or not. Standard two sided Kolmogorov-Smirnov test is used to calculate the (adjusted) p-values.

Value

a matrix containing medians and p-values for all genes in all batches

See Also

ks.test

p.adjust

correctBatchEffect

Examples

```
## Shortly running example. For a more realistic example that takes
## some more time, run the same procedure with the full BEclearData
## dataset.

## Calculate fdr-adjusted p-values in non-parallel mode
data(BEclearData)
ex.data <- ex.data[31:90, 7:26]
ex.samples <- ex.samples[7:26, ]


res <- calcBatchEffects(data = ex.data, samples = ex.samples, method = "fdr")

## How to handle data-sets without defined batches
## https://github.com/Livia-Rasp/BEclear/issues/22
library(data.table)
data(BEclearData)

DT <- data.table(ex.samples)[, .(sample_id)]

## set the batch_id equal to the sample_id
## this way samples are treated as batches
DT[, batch_id := sample_id]

res <- calcBatchEffects(data = ex.data, samples = DT)
```

---

calcBatchEffectsForBatch

calcBatchEffectsForBatch

Description

function to calculate batch effects for every gene in a batch

Usage

```
calcBatchEffectsForBatch(batch, samples, data, BPPARAM = SerialParam())
```

Value

the medians p-values for genes in a batch

`calcBlockFrame` *calcBlockFrame*

Description

calculates every start and stop position for every block

Usage

```
calcBlockFrame(rowPositions, colPositions, rowBlockSize, colBlockSize)
```

Value

a data.frame containing the start and stop positions for each block

`calcPositions` *calcPositions*

Description

calcPositions

Usage

```
calcPositions(num, blockSize)
```

Value

returns vector with number of blocks, start - and stop position of last 2 blocks

`calcScore` *calculate batch effect score*

Description

Returns a table with the number of found genes with found p-values less or equal to 0.01 and median values greater or equal to 0.05. A score is calculated depending on the number of found genes as well as the magnitude of the median difference values, this score is divided by the overall number of genes in the data and returned as "BEscore". See details for further information and details about the score calculation. The returned data.frame is also stored in the specified directory as .RData file.

Usage

```
calcScore(data, samples, summary, saveAsFile=FALSE, dir=getwd())
```

Arguments

data	any matrix filled with beta values, column names have to be sample_ids corresponding to the ids listed in "samples", row names have to be gene names.
samples	data frame with two columns, the first column has to contain the sample numbers, the second column has to contain the corresponding batch number. Colnames have to be named as "sample_id" and "batch_id".
summary	a summary data.table containing the columns "gene", "batch", median" and "p-value" and consists of all genes which were found in the median and p-value calculations, see calcSummary function for more details.
saveAsFile	determining if the data.frame should also be saved as a file
dir	set the path to a directory the returned data.frame should be stored. The current working directory is defined as default parameter.

Details

calcScore

The returned data frame contains one column for the batch numbers, 11 columns containing the number of genes found in a certain range of the median difference value and a column with the calculated BEscore. These found genes are assumed to be batch affected due to their difference in median values and their different distribution of the beta values. The higher the found number of genes and the more extreme the median difference is, the more severe is the assumed batch effect supposed to be. We suggest that there is no need for a batch effect correction if the BEscore for a batch is less than 0.02. BEscores between 0.02 and 0.1 are lying in a "gray area" for which we assume a not severe batch effect, and values beyond 0.1 certainly describe a batch effect and should definitely be corrected.
The 11 columns containing the numbers of found genes count the median difference values which are ranging from >= 0.05 to < 0.1 ; >= 0.1 to < 0.2; >= 0.2 to < 0.3 and so on up to a limit of 1.
The BEscore is calculated by the sum of the weighted number of genes divided by the number of genes. Weightings are calculated by multiplication of the number of found genes between 0.05 and 0.1 by 1, between 0.1 and 0.2 by 2, between 0.2 and 0.3 by 4, between 0.3 and 0.4 by 6 and so on.

Value

A data.frame is returned containing the number of found genes assumed to be batch affected separated by batch and a BEscore for every batch. Furthermore there's a column dixonPval giving you a p-value regarding each BEscore according to a Dixon test. The data.frame is also stored in the specified directory as .RData file, if saveAsFile is TRUE.

References

Dixon WJ (1950). "Analysis of Extreme Values." *The Annals of Mathematical Statistics*, **21**(4), 488–506. ISSN 0003-4851, doi:10.1214/aoms/1177729747, http://projecteuclid.org/euclid.aoms/1177729747.

Dixon WJ (1951). "Ratios Involving Extreme Values." *The Annals of Mathematical Statistics*, **22**(1), 68–78. ISSN 0003-4851, doi:10.1214/aoms/1177729693, http://projecteuclid.org/euclid.aoms/1177729693.

Rorabacher DB (1991). "Statistical treatment for rejection of deviant values: critical values of Dixon's "Q" parameter and related subrange ratios at the 95% confidence level." *Analytical Chemistry*, **63**(2), 139–146. ISSN 0003-2700, doi:10.1021/ac00002a010, http://pubs.acs.org/doi/abs/10.1021/ac00002a010.

See Also

calcBatchEffects

calcSummary

correctBatchEffect

Examples

```
## Shortly running example. For a more realistic example that takes
## some more time, run the same procedure with the full BEclearData
## dataset.

## Whole procedure that has to be done to use this function.
data(BEclearData)
ex.data <- ex.data[31:90, 7:26]
ex.samples <- ex.samples[7:26, ]
## Calculate the batch effects
batchEffects <- calcBatchEffects(data = ex.data, samples = ex.samples,
adjusted = TRUE, method = "fdr")
med <- batchEffects$med
pvals <- batchEffects$pval

# Summarize p-values and median differences for batch affected genes
sum <- calcSummary(medians = med, pvalues = pvals)

# Calculates the score table
score.table <- calcScore(data = ex.data, samples = ex.samples, summary = sum)
```

---

calcSummary *Summarize median comparison and p-value calculation results*

Description

Summarizes the results of the calcBatchEffects function

Usage

```
calcSummary(medians, pvalues, mediansTreshold, pvaluesTreshold)
```

Arguments

medians	a matrix containing median difference values calculated by the calcBatchEffects function. For further details look at the documentation of this function.
pvalues	a matrix containing p-values calculated by the calcBatchEffects function. For further details look at the documentation of this function.
mediansTreshold	the threshold above or equal median values are regarded as batch effected, when the criteria for p-values is also met.
pvaluesTreshold	the threshold below or equal p-values are regarded as batch effected, when the criteria for medians is also met.

Details

calcSummary

All genes with a median comparison value >= 0.05 and a p-value of <= 0.01 are summarized into a data.frame. These genes are assumed to contain a batch effect

Value

Null if there are no batch effects detected, else a data.table with the columns "gene" containing the gene name, "batch" containing the batch number from which the gene was found, "median" and "p-value" containing the calculated median difference values and the p-values, respectively.

See Also

calcBatchEffects

correctBatchEffect

Examples

```
## Shortly running example. For a more realistic example that takes
## some more time, run the same procedure with the full BEclearData
## dataset.

## Whole procedure that has to be done to use this function.
data(BEclearData)
ex.data <- ex.data[31:90, 7:26]
ex.samples <- ex.samples[7:26, ]

## Calculate the batch effects
batchEffects <- calcBatchEffects(data = ex.data, samples = ex.samples,
adjusted = TRUE, method = "fdr")
med <- batchEffects$med
pvals <- batchEffects$pval

## Summarize p-values and median differences for batch affected genes
sum <- calcSummary(medians = med, pvalues = pvals)
```

`clearBEgenes` *Prepare a data matrix for the BEclear function*

Description

A function that simply sets all values to NA which were previously found by median value comparison and p-value calculation and are stored in a summary. The summary defines which values in the data matrix are set to NA.

Usage

```
clearBEgenes(data, samples, summary)
```

Arguments

`data`	any matrix filled with beta values, column names have to be sample_ids corresponding to the ids listed in "samples", row names have to be gene names.
`samples`	data frame with two columns, the first column has to contain the sample numbers, the second column has to contain the corresponding batch number. Colnames have to be named as "sample_id" and "batch_id".
`summary`	a summary data.frame containing the columns "gene", "batch", "median" and "p-value" and consists of all genes which were found in the median and p-value calculations, see `calcSummary` function for more details.

Details

clearBEgenes

All entries belonging to genes stated in the summary are set to NA for the corresponding batches in the data matrix. Please look at the descriptions of `calcBatchEffects` for more detailed information about the data which should be contained in the summary data.frame.

Value

A data matrix with the same dimensions as well as the same column and row names as the input data matrix is returned, all entries which are defined in the summary are now set to NA.

See Also

`calcBatchEffects`

`calcSummary`

`correctBatchEffect`

Examples

```
## Shortly running example. For a more realistic example that takes
## some more time, run the same procedure with the full BEclearData
## dataset.

## Whole procedure that has to be done to use this function.
data(BEclearData)
ex.data <- ex.data[31:90, 7:26]
ex.samples <- ex.samples[7:26, ]

## Calculate the batch effects
batchEffects <- calcBatchEffects(data = ex.data, samples = ex.samples,
adjusted = TRUE, method = "fdr")
meds <- batchEffects$med
pvals <- batchEffects$pval

## Summarize p-values and median differences for batch affected genes
sum <- calcSummary(medians = meds, pvalues = pvals)

## Set values for summarized BEgenes to NA
clearedMatrix <- clearBEgenes(data = ex.data, samples = ex.samples, summary = sum)
```

`combineBlocks` *combineBlocks*

Description

load and combine blocks

Usage

```
combineBlocks(blockFrame, rowPos, colPos, dir)
```

Value

the combined matrix consisting of the individual blocks

`correctBatchEffect` *Correct a batch effect in DNA methylation data*

Description

This method combines most functions of the `BEclear-package` to one. The method performs the whole process of searching for batch effects and automatically correct them for a matrix of beta values stemming from DNA methylation data.

Usage

```
correctBatchEffect(data, samples, adjusted=TRUE, method="fdr",
mediansTreshold = 0.05, pvaluesTreshold = 0.01, rowBlockSize=60,
colBlockSize=60, epochs=50, lambda = 1, gamma = 0.01, r = 10,
outputFormat="", dir=getwd(), BPPARAM=SerialParam())
```

Arguments

data	any matrix filled with beta values, column names have to be sample_ids corresponding to the ids listed in "samples", row names have to be gene names.
samples	data frame with two columns, the first column has to contain the sample numbers, the second column has to contain the corresponding batch number. Colnames have to be named as "sample_id" and "batch_id".
adjusted	should the p-values be adjusted or not, see "method" for available adjustment methods.
method	adjustment method for p-value adjustment, default method is "false discovery rate adjustment", for other available methods see the description of the used standard R package p.adjust. See calcBatchEffects for more information.
mediansTreshold	the threshold above or equal median values are regarded as batch effected, when the criteria for p-values is also met.
pvaluesTreshold	the threshold below or equal p-values are regarded as batch effected, when the criteria for medians is also met.
rowBlockSize	the number of rows that is used in a block if the function is run in parallel mode and/or not on the whole matrix. Set this, and the "colBlockSize" parameter to 0 if you want to run the function on the whole input matrix. See imputeMissingData and especially the details section of See imputeMissingData for more information about this feature.
colBlockSize	the number of columns that is used in a block if the function is run in parallel mode and/or not on the whole matrix. Set this, and the "rowBlockSize" parameter to 0 if you want to run the function on the whole input matrix. See imputeMissingData and especially the details section of See imputeMissingData for more information about this feature.
epochs	the number of iterations used in the gradient descent algorithm to predict the missing entries in the data matrix. See imputeMissingData for more information.
lambda	constant that controls the extent of regularization during the gradient descent
gamma	constant that controls the extent of the shift of parameters during the gradient descent
r	length of the second dimension of variable matrices R and L
outputFormat	you can choose if the finally returned data matrix should be saved as an .RData file or as a tab-delimited .txt file in the specified directory. Allowed values are "RData" and "txt". See imputeMissingData for more information.

dir	set the path to a directory the predicted matrix should be stored. The current working directory is defined as default parameter.
BPPARAM	An instance of the BiocParallelParam-class that determines how to parallelisation of the functions will be evaluated.

Details

correctBatchEffect

The function performs the whole process of searching for batch effects and automatically correct them for a matrix of beta values stemming from DNA methylation data. Thereby, the function is running most of the functions from the BEclear-package in a logical order.
First, median comparison values are calculated by the calcBatchEffects function, followed by the calculation of p-values also by the calcBatchEffects function. With the results from the median comparison and p-value calculation, a summary data frame is build using the calcSummary function, and a scoring table is established by the calcScore function. Now, found entries from the summary are set to NA in the input matrix using the clearBEgenes function, then the imputeMissingData function is used to predict the missing values and at the end, predicted entries outside the boundaries (values lower than 0 or greater than 1) are corrected using the replaceOutsideValues function.

Value

A list containing the following fields (for detailed information look at the documentations of the corresponding functions):

medians A data.frame containing all median comparison values corresponding to the input matrix.

pvalues A data.frame containing all p-values corresponding to the input matrix.

summary The summarized results of the median comparison and p-value calculation.

score.table A data.frame containing the number of found genes and a BEscore for every batch.

cleared.data the input matrix with all values defined in the summary set to NA.

predicted.data the input matrix after all previously NA values have been predicted.

corrected.predicted.data the predicted matrix after the correction for predicted values outside the boundaries.

References

Akulenko R, Merl M, Helms V (2016). "BEclear: Batch effect detection and adjustment in DNA methylation data." *PLoS ONE*, **11**(8), 1–17. ISSN 19326203, doi:10.1371/journal.pone.0159921, http://www.ncbi.nlm.nih.gov/pubmed/27559732.

Koren Y, Bell R, Volinsky C (2009). "Matrix Factorization Techniques for Recommender Systems." *Computer*, **42**(8), 30–37. ISSN 0018-9162, doi:10.1109/MC.2009.263, doi.ieeecomputersociety.org/10.1109/MC.2009.263http://ieeexplore.ieee.org/document/5197422/.

Candès EJ, Recht B (2009). "Exact Matrix Completion via Convex Optimization." *Foundations of Computational Mathematics*, **9**(6), 717–772. ISSN 1615-3375, doi:10.1007/s1020800990455, http://link.springer.com/10.1007/s10208-009-9045-5.

See Also

calcBatchEffects

calcSummary

calcScore

clearBEgenes

imputeMissingData

replaceOutsideValues

Examples

```
## Shortly running example. For a more realistic example that takes
## some more time, run the same procedure with the full BEclearData
## dataset.

## Whole procedure that has to be done to use this function.
## Correct the example data for a batch effect
data(BEclearData)
ex.data <- ex.data[31:90, 7:26]
ex.samples <- ex.samples[7:26, ]

# Note that row- and block sizes are just set to 10 to get a short runtime.
# To use these parameters, either use the default values or please note the
# description in the details section of \code{\link{imputeMissingData}}
result <- correctBatchEffect(
  data = ex.data, samples = ex.samples,
  adjusted = TRUE, method = "fdr", rowBlockSize = 10, colBlockSize = 10,
  epochs = 50, outputFormat = "RData", dir = getwd()
)

# Unlist variables
medians <- result$medians
pvals <- result$pvals
summary <- result$summary
score <- result$score.table
cleared <- result$clearedData
predicted <- result$predictedData
corrected <- result$correctedPredictedData
```

---

countValuesToPredict *Count NA entries in a matrix*

Description

Simple function that counts all values in a matrix which are NA

Usage

```
countValuesToPredict(data)
```

Arguments

data	any kind of matrix

Details

countValuesToPredict

Value

Returns a data frame with the number of NA entries per column. Since the function is mainly written for the usage in batch effect correction of DNA methylation data, the column names of the data frame are set to "sample" and "num_pred_values". Nevertheless, the function can be used with any other matrix containing anything but beta values.

See Also

clearBEgenes

correctBatchEffect

correctBatchEffect

Examples

```
## Shortly running example. For a more realistic example that takes
## some more time, run the same procedure with the full BEclearData
## dataset

## Whole procedure that has to be done to use this function.
data(BEclearData)
ex.data <- ex.data[31:90, 7:26]
ex.samples <- ex.samples[7:26, ]

## Calculate the batch effects
batchEffects <- calcBatchEffects(data = ex.data, samples = ex.samples,
adjusted = TRUE, method = "fdr")
meds <- batchEffects$med
pvals <- batchEffects$pval

## Summarize p-values and median differences for batch affected genes
sum <- calcSummary(medians = meds, pvalues = pvals)
clearedMatrix <- clearBEgenes(data = ex.data, samples = ex.samples, summary = sum)
numberOfEntries <- countValuesToPredict(data = clearedMatrix)
```

ex.corrected.data *Example matrix of corrected data for the BEclear-package*

Description

Example matrix containing a already batch effect corrected sample matrix of beta values from breast invasive carcinoma TCGA methylation data.[1] The matrix contains a small amount of predicted beta values outside of the boundaries to show the operating principles of some of the methods from the BEclear package.

Usage

```
data(BEclearCorrected)
```

Format

A matrix containing already corrected beta values of some samples from the breast invasive carcinoma TCGA methylation data. The colnames denote samples, rownames denote gene names.

References

Cancer Genome Atlas Research Network, Weinstein JN, Collisson EA, Mills GB, Shaw KRM, Ozenberger BA, Ellrott K, Shmulevich I, Sander C, Stuart JM (2013). "The Cancer Genome Atlas Pan-Cancer analysis project." *Nature genetics*, **45**(10), 1113–20. ISSN 1546-1718, doi:10.1038/ng.2764, http://www.ncbi.nlm.nih.gov/pubmed/24071849http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3919969.

findOutsideValues *Find DNA methylation values out of the boundaries*

Description

A method which lists values below 0 or beyond 1 contained in the input matrix. These entries are stored in a data.frame together with the corresponding row and column position of the matrix. Note that this method is especially designed for DNA methylation data.

Usage

```
findOutsideValues(data)
```

Arguments

data	any matrix filled with values that normally should be bounded between 0 and 1.

Details

findOutsideValues

Note that this method is especially designed to run after the batch effect correction of DNA methylation data, e.g. with the `correctBatchEffect` method. It can happen, that the predicted values are lying slightly below the lower bound of 0 or beyond the upper bound of 1. This method finds these inaccurately predicted entries. Another method called `replaceOutsideValues` replaces these values either by 0 or 1, respectively.

Value

A data frame containing the columns "level", "row", "col" and "value" defining if the value is below 0 or beyond 1 (level = 0 or level = 1), the row position and the column position in the input matrix and the value itself, respectively.

See Also

`replaceOutsideValues`

`correctBatchEffect`

Examples

```
data(BEclearCorrected)
# Find predicted values outside of the boundaries
outsideEntries <- findOutsideValues(data = ex.corrected.data)
```

---

gdepoch *gdepoch*

Description

Run a gradient descent epoch (L and R are starting points, gamma is stgammaize)

Usage

```
gdepoch(L, R, lambda, gamma, is, js, D, error_matrix = NULL)
```

Value

a list containing two matrices generated by the gradient descent for the current epoch

`imputeMissingData` *Matrix prediction using a Latent Factor Model*

Description

This function predicts the missing entries of an input matrix (NA values) through the use of a Latent Factor Model. You can run the function also in parallel mode and split up the matrix to a certain number of smaller matrices to speed up the prediction process. If you set the rowBlockSize and colBlockSize both to 0, the function is running on the whole matrix. Take a look at the details section for some deeper information about this. The default parameters are chosen with the intention to make an accurate prediction within an affordable time interval.

Usage

```
imputeMissingData(data, rowBlockSize=60,  colBlockSize=60, epochs=50,
lambda = 1, gamma = 0.01, r = 10, outputFormat="", dir = tempdir(),
BPPARAM=SerialParam())
```

Arguments

`data`	any matrix filled e.g. with beta values. The missing entries you want to predict have to be set to NA
`rowBlockSize`	the number of rows that is used in a block if the function is run in parallel mode and/or not on the whole matrix. Set this and the "colBlockSize" parameter to 0 if you want to run the function on the whole input matrix. We suggest to use a block size of 60 but you can also use any other block size, but the size has to be bigger than the number of samples in the biggest batch. Look at the details section for more information about this feature.
`colBlockSize`	the number of columns that is used in a block if the function is run in parallel mode and/or not on the whole matrix. Set this, and the "rowBlockSize" parameter to 0 if you want to run the function on the whole input matrix. We suggest to use a block size of 60 but you can also use any other block size, but the size has to be bigger than the number of samples in the biggest batch. Look at the details section for more information about this feature.
`epochs`	the number of iterations used in the gradient descent algorithm to predict the missing entries in the data matrix.
`lambda`	constant that controls the extent of regularization during the gradient descent
`gamma`	constant that controls the extent of the shift of parameters during the gradient descent
`r`	length of the second dimension of variable matrices R and L
`outputFormat`	you can choose if the finally returned data matrix should be saved as an .RData file or as a tab-delimited .txt file in the specified directory. Allowed values are "RData" and "txt".
`dir`	set the path to a directory the predicted matrix should be stored. The current working directory is defined as default parameter.

BPPARAM An instance of the BiocParallelParam-class that determines how to parallelisation of the functions will be evaluated.

Details

imputeMissingData

The method used to predict the missing entries in the matrix is called "latent factor model". In the following sections, the method itself is described as well as the correct usage of the parameters. The parameters are described in the same order as they appear in the usage section.
The method originally stems from recommender systems where the goal is to predict user ratings of products. It is based on matrix factorization and uses a discrete gradient descent (GDE) algorithm that stepwise predicts two matrices L and R with matching dimensions to the input matrix. These two matrices are initialized with random numbers and stepwise adjusted towards the values of the input matrix through the GDE algorithm. After every adjustment step, the global loss is calculated and the parameters used for the adjustment are possibly also adjusted so that the global loss is getting minimized and the prediction is getting accurate. After a predefined number of steps (called epochs) are executed by the GDE algorithm, the predicted matrix is calculated by matrix multiplication of L and R. Finally, all missing values in the input matrix are replaced with the values from the predicted matrix and the already known values from the input matrix are maintained. The completed input matrix is then returned at the end.
Description of the parameters:

- data: simply the input matrix with missing values set to NA
- rowBlockSize and colBlockSize: Here you can define the dimensions of the smaller matrices, the input matrix is divided into if the function is working in parallel mode. For details about these so called blocks, see the section "About the blocks" below.
- epochs: Defines the number of steps the gradient descent algorithm performs until the prediction ends. Note that the higher this number is, the more precisely is the prediction and the more time is needed to perform the prediction. If the step size is too small, the prediction would not be very good. We suggest to use a step size of 50 since we did not get better predictions if we took higher step sizes during our testing process.

About the blocks: You have the possibility to change the size of the blocks in which the input matrix can be divided. if you choose e.g. the rowBlockSize = 50 and the colBlockSize = 60 your matrix will be cut into smaller matrices of the size approximately 50x60. Note that this splitting algorithm works with every possible matrix size! If both size parameters do not fit to the dimensions of the input matrix, the remaining rows and columns of the input matrix are distributed over some blocks, so that the block sizes are roughly of the same size. All blocks are saved at the specified directory after the processing of a block has been done within an RData file. These RData files are continuously numbered and contain the row and column start and stop positions in their name. Next, these blocks are assembled into the returned matrix and this matrix is saved in the specified directory. Finally, single blocks are deleted. To see how this is done, simply run the example at the end of this documentation. We suggest to use the block size of 60 (default) but you can also use any other block size, as far as it is bigger than the number of samples in the biggest batch. This avoids having an entire row of NA values in a block which leads to a crash of the imputeMissingData method. In order to process the complete matrix without dividing into blocks, specify rowBlockSize = 0 and colBlockSize = 0. But if the input matrix is large (more than 200x200), it is not recommended due to exponential increase of computation time required.

Note that the size of the blocks affect the prediction accuracy. In case of very small blocks, the information obtained from neighbor entries is not sufficient. Thus, the larger the size of the block is, the more accurately those entries are predicted. Default size 60 is enough to have accurate prediction in a reasonable amount of time.

Value

Returns a data matrix with the same dimensions as well as same row and column names as the input matrix. According to the "outputFormat" parameter, either a .RData file containing only the returned matrix or a tab-delimited .txt file containing the content of the returned matrix is saved in the specified directory.

References

Akulenko R, Merl M, Helms V (2016). "BEclear: Batch effect detection and adjustment in DNA methylation data." *PLoS ONE*, **11**(8), 1–17. ISSN 19326203, doi:10.1371/journal.pone.0159921, http://www.ncbi.nlm.nih.gov/pubmed/27559732.

Koren Y, Bell R, Volinsky C (2009). "Matrix Factorization Techniques for Recommender Systems." *Computer*, **42**(8), 30–37. ISSN 0018-9162, doi:10.1109/MC.2009.263, doi.ieeecomputersociety.org/10.1109/MC.2009.263http://ieeexplore.ieee.org/document/5197422/.

Candès EJ, Recht B (2009). "Exact Matrix Completion via Convex Optimization." *Foundations of Computational Mathematics*, **9**(6), 717–772. ISSN 1615-3375, doi:10.1007/s1020800990455, http://link.springer.com/10.1007/s10208-009-9045-5.

Examples

```
## Shortly running example. For a more realistic example that takes
## some more time, run the same procedure with the full BEclearData
## dataset.

## Whole procedure that has to be done to use this function.
data(BEclearData)
ex.data <- ex.data[31:90, 7:26]
ex.samples <- ex.samples[7:26, ]

## Calculate the batch effects
batchEffects <- calcBatchEffects(data = ex.data, samples = ex.samples,
adjusted = TRUE, method = "fdr")
meds <- batchEffects$med
pvals <- batchEffects$pval

## Summarize p-values and median differences for batch affected genes
sum <- calcSummary(medians = meds, pvalues = pvals)

## Set entries defined by the summary to NA
clearedMatrix <- clearBEgenes(data = ex.data, samples = ex.samples, summary = sum)

# Predict the missing entries with standard values, row- and block sizes are
# just set to 10 to get a short runtime. To use these parameters, either use
# the default values or please note the description in the details section
# above
```

```
predicted <- imputeMissingData(
  data = clearedMatrix, rowBlockSize = 10,
  colBlockSize = 10
)
```

`imputeMissingDataForBlock` *imputeMissingDataForBlock*

Description

imputeMissingDataForBlock

Usage

```
imputeMissingDataForBlock(block, dir, epochs, lambda = 1, gamma = 0.01, r = 10)
```

Arguments

`fixedSeed`	determines if they seed should be fixed, which is important for testing

Value

number of the block processed

`localLoss` *Calculating the Local Loss*

Description

Calculates the gradient dR and dL for the current epoch.

Usage

```
localLoss(L, R, is, js, error_matrix)
```

Value

a list containing two matrices containing the local loss

loss *loss*

Description

computation of the loss of factorization LR

Usage

```
loss(L, R, lambda, D)
```

Arguments

L	a matrix describing the effects of the features
R	a matrix describing the effects of the samples
lambda	constant that controls the extent of regularization during the gradient descent
D	a matrix containing the measured values

Value

a list containing the loss calculated and the error matrix

makeBoxplot *produce simple predefined boxplot for methylation data*

Description

A simple boxplot is done with boxes either separated by batches or by samples and describe the five number summary of all beta values corresponding to a batch or a sample, respectively. The batch_ids are shown on the x-axis with a coloring corresponding to the BEscore.

Usage

```
makeBoxplot(data, samples, score, bySamples=FALSE, col="standard",
main="", xlab="Batch", ylab="Beta value", scoreCol=TRUE, log = FALSE)
```

Arguments

data	any matrix filled with beta values, column names have to be sample_ids corresponding to the ids listed in "samples", row names have to be gene names.
samples	data frame with two columns, the first column has to contain the sample numbers, the second column has to contain the corresponding batch number. Colnames have to be named as "sample_id" and "batch_id".

score	data frame produced by the calcScore function. Contains the number of presumably batch affected genes and a BEscore which is needed for the coloring of the batch_ids.
bySamples	should the boxes be separated by samples or not. If not, boxes are separated by the batch_ids.
col	colors for the boxes, refers to the standard boxplot R-function. If it is set to "standard", boxes are colored batch-wise (if separated by samples) or the standard color "yellow" is used (if separated by batches).
main	main title for the box plot. Default is an empty string.
xlab	label for the x-axis of the box plot. Default is "Batch".
ylab	label for the y-axis of the box plot. Default is "Beta value".
scoreCol	should the batch_ids on the a-axis be colored according to the BEscore or not? If not, black is used as color for all batch_ids.
log	TRUE, if the y-axis should be on a logarithmic scale.

Details

makeBoxplot

The color code for the batch_ids on the x-axis provides a simple "traffic light" the user can use to decide if he wants to correct for an assumed batch effect or not. Green means no batch effect, yellow a possibly existing not severe batch effect and red stands for an obviously existing batch effect that should be corrected. The traffic light colors are set according to the BEscore from the calcScore function, values from 0 to 0.02 are colored in green, from 0.02 to 0.1 in yellow and values over 0.1 are colored in red.

Value

Returns a boxplot on the graphic device with the features explained above.

See Also

calcScore

boxplot

correctBatchEffect

Examples

```
## Shortly running example. For a more realistic example that takes
## some more time, run the same procedure with the full BEclearData
## dataset.

## Whole procedure that has to be done to use this function.
data(BEclearData)
ex.data <- ex.data[31:90, 7:26]
ex.samples <- ex.samples[7:26, ]

## Prepare the data for the box plots
```

```
## Calculate the batch effects
batchEffects <- calcBatchEffects(data = ex.data, samples = ex.samples,
adjusted = TRUE, method = "fdr")
meds <- batchEffects$med
pvals <- batchEffects$pval

## Summarize p-values and median differences for batch affected genes
sum <- calcSummary(medians = meds, pvalues = pvals)

# Calculate the BEscore for the batch_id colorings of the x-axis
score <- calcScore(data = ex.data, samples = ex.samples, summary = sum)

## Simple boxplot for the example data separated by batch
makeBoxplot(
  data = ex.data, samples = ex.samples, score = score, bySamples = FALSE,
  main = "Some box plot"
)

## Simple boxplot for the example data separated by samples
makeBoxplot(
  data = ex.data, samples = ex.samples, score = score, bySamples = TRUE,
  main = "Some box plot"
)
```

`preprocessBEclear` *preprocessBEclear*

Description

this methods does some preprocessing steps for the later methods like removing rows containing only missing values

Usage

```
preprocessBEclear(data, samples)
```

Arguments

`data`	any matrix filled with beta values, column names have to be sample_ids corresponding to the ids listed in "samples", row names have to be gene names.
`samples`	data frame with two columns, the first column has to contain the sample numbers, the second column has to contain the corresponding batch number. Colnames have to be named as "sample_id" and "batch_id".

Details

Here we describe the preprocessing steps in the order they are executed:

- Values below 0 or above 1 are set to `NA`, as the other methods expect methylation beta values

- columns that only contain NAs are removed
- rows that only contain NAs are removed
- samples that are present in the data, but are not annoted in the samples are removed. If this is the case with your data-set, please check those samples.
- samples that are annoted but not in the data matrix are removed
- if there are duplicate sample names in the data matrix, all sample names get replaced through a new unique ID. In this case a `data.table` containing the mapping is returned as well

Value

a list containing the processed data and samples and a `data.table` containing mappings from the original sample names to the new ones. If sample names weren't changed this third object is NULL

Examples

```
data(BEclearData)
res <- preprocessBEclear(ex.data, ex.samples)
```

`replaceOutsideValues` *Replace DNA methylation values out of the boundaries*

Description

A method which replaces values below 0 or beyond 1 contained in the input matrix. These wrong entries are replaced by 0 or 1, respectively. Note that this method is especially designed for DNA methylation data.

Usage

```
replaceOutsideValues(data)
```

Arguments

`data`	any matrix filled with values that normally should be bounded between 0 and 1.

Details

replaceOutsideValues

Note that this method is especially designed to run after the batch effect correction of DNA methylation data, e.g. with the `imputeMissingData` method. It can happen, that the predicted values are lying slightly below the lower bound of 0 or beyond the upper bound of 1. This method finds these inaccurately predicted entries. Another method called `replaceOutsideValues` replaces these values either by 0 or 1, respectively. Another method called `findOutsideValues` returns a list of existing wrong values and can be run before the replacement.

Value

Returns the input matrix with every value previously below 0 changed to 0 and every value previously beyond 1 changed to 1.

See Also

findOutsideValues

correctBatchEffect

Examples

```
data(BEclearCorrected)
# Replace wrongly predicted values
corrected <- replaceOutsideValues(data = ex.corrected.data)
```

---

runGradientDescent *runGradientDescent*

Description

Runner for gradient descent (or stochastic gradient descent) for the specified number of epoch

Usage

```
runGradientDescent(L, R, lambda, epochs, gamma = 0.01, blockNr, is, js, D, r)
```

Value

a list containing two matrices generated by the gradient descent

Index