# Package 'BERT'

October 24, 2025

**Title** High Performance Data Integration for Large-Scale Analyses of Incomplete Omic Profiles Using Batch-Effect Reduction Trees (BERT)

**Version** 1.5.0

**Description** Provides efficient batch-effect adjustment of data with missing values. BERT orders all batch effect correction to a tree of pairwise computations. BERT allows parallelization over sub-trees.

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.2

**biocViews** BatchEffect, Preprocessing, ExperimentalDesign, QualityControl

**URL** https://github.com/HSU-HPC/BERT/

**BugReports** https://github.com/HSU-HPC/BERT/issues

**License** GPL-3

**Depends** R (>= 4.3.0)

**Imports** cluster, comprehenr, foreach (>= 1.5.2), invgamma, iterators (>= 1.0.14), janitor (>= 2.2.0), limma (>= 3.46.0), logging (>= 0.10-108), sva (>= 3.38.0), SummarizedExperiment, methods, BiocParallel

**Suggests** testthat (>= 3.0.0), knitr, rmarkdown, BiocStyle

**Config/testthat/edition** 3

**VignetteBuilder** knitr

**git_url** https://git.bioconductor.org/packages/BERT

**git_branch** devel

**git_last_commit** 3fb455b

**git_last_commit_date** 2025-06-06

**Repository** Bioconductor 3.22

**Date/Publication** 2025-10-24

**Author** Yannis Schumann [aut, cre] (ORCID: <https://orcid.org/0000-0002-2379-200X>), Simon Schlumbohm [aut] (ORCID: <https://orcid.org/0000-0002-0083-5142>)

**Maintainer** Yannis Schumann <yannis.schumann@desy.de>

# Contents



---

`adjustment_step` *Adjust a hierarchy level sequentially.*

---

## Description

This function uses ComBat or limma to adjust an entire hierarchy level.

## Usage

```
adjustment_step(data, mod, combatmode, method)
```

## Arguments

| | |
|---|---|
| `data` | Matrix or dataframe in the format (samples, features). Additional column names are \"Batch\", \"Cov_X\" (were X may be any number), \"Label\" and \"Sample\". |
| `mod` | Dataframe with potential covariables to use. May be emty. |
| `combatmode` | Integer, encoding the parameters to use for ComBat. 1 (default) par.prior = TRUE, mean.only = FALSE 2 par.prior = TRUE, mean.only = TRUE 3 par.prior = FALSE, mean.only = FALSE 4 par.prior = FALSE, mean.only = TRUE Will be ignored, if method=="limma". |
| `method` | Adjustment method to use. Should either be \"ComBat\" or \"limma\". |

## Value

A matrix/dataframe mirroring the shape of the input. The data will be batch-effect adjusted by BERT.

adjust_node *Adjust two batches to each other.*

## Description

This function is called by the BERT algorithm and should not be called by the user directly.

## Usage

```
adjust_node(data, b1, b2, mod, combatmode, method)
```

## Arguments

| | |
|---|---|
| data | Matrix or dataframe in the format (samples, features). Additional column names are "Batch", "Cov_X" (were X may be any number), "Label" and "Sample". |
| b1 | The first batch to adjust. |
| b2 | The second batch to adjust. |
| mod | Dataframe with potential covariables to use. May be emty. |
| combatmode | Integer, encoding the parameters to use for ComBat. 1 (default) par.prior = TRUE, mean.only = FALSE 2 par.prior = TRUE, mean.only = TRUE 3 par.prior = FALSE, mean.only = FALSE 4 par.prior = FALSE, mean.only = TRUE Will be ignored, if method=="limma". |
| method | Adjustment method to use. Should either be "ComBat" or "limma". "None" is also allowed for testing purposes and will yield no batch effect correction. |

## Value

A matrix/dataframe mirroring the shape of the input. The data will be batch-effect adjusted by the specified method.

BERT *Adjust data using the BERT algorithm.*

## Description

This function uses the hierarchical BERT algorithm to adjust data with batch effects. It assumes that the data is in the format (samples, features) and that missing values are indicated by NA. An additional column labelled "Batch" should indicate the batch. Furthermore all columns named "Cov_1", "Cov_2", ... will be considered as covariate for adjustment. Columns labelled "Label" and "Sample" will be ignored, all other columns are assumed to contain data.

## Usage

```
BERT(
  data,
  cores = NULL,
  combatmode = 1,
  corereduction = 4,
  stopParBatches = 2,
  backend = "default",
  method = "ComBat",
  qualitycontrol = TRUE,
  verify = TRUE,
  labelname = "Label",
  batchname = "Batch",
  referencename = "Reference",
  samplename = "Sample",
  covariatename = NULL,
  BPPARAM = NULL,
  assayname = NULL
)
```

## Arguments

| | |
|---|---|
| data | Matrix dataframe/SummarizedExperiment in the format (samples, features). Additional column names are "Batch", "Cov_X" (were X may be any number), "Label", "Sample" and "Reference". Must contain at least two features. |
| cores | The number of cores to use for parallel adjustment. Increasing this number leads to faster adjustment, especially on Linux machines. The default is NULL, in which case the BiocParallel::bpparam() backend will be used. If an integer is given, a backend with the corresponding number of workers will be created and registered as default for usage. |
| combatmode | Integer, encoding the parameters to use for ComBat. 1 (default) par.prior = TRUE, mean.only = FALSE 2 par.prior = TRUE, mean.only = TRUE 3 par.prior = FALSE, mean.only = FALSE 4 par.prior = FALSE, mean.only = TRUE Will be ignored, if method!="ComBat". |
| corereduction | Reducing the number of workers by at least this number. Only used if cores is an integer. |
| stopParBatches | The minimum number of batches required at a hierarchy level to proceed with parallelized adjustment. If the number of batches is smaller, adjustment will be performed sequentially to avoid overheads. |
| backend | The backend to choose for communicating the data. Valid choices are "default" and "file". The latter will use temp files for communicating data chunks between the processes. after adjusting all sub-trees as far as possible with the previous number of cores. |
| method | Adjustment method to use. Should either be "ComBat", "limma" or "ref". Also allows "None" for testing purposes, which will perform no BE adjustment |
| qualitycontrol | Boolean indicating, whether ASWs should be computed before and after batch effect adjustment. If TRUE, will compute ASW with respect to the "Batch" and "Label" column (if existent). |
| verify | Whether the input matrix/dataframe needs to be verified before adjustment (faster if FALSE) |

| | |
|---|---|
| labelname | A string containing the name of the column to use as class labels. The default is "Label". |
| batchname | A string containing the name of the column to use as batch labels. The default is "Batch". |
| referencename | A string containing the name of the column to use as ref. labels. The default is "Reference". |
| samplename | A string containing the name of the column to use as sample name. The default is "Sample". |
| covariatename | A vector containing the names of columns with categorical covariables. The default is NULL, for which all columns with the pattern "Cov" will be selected. |
| BPPARAM | An instance of BiocParallelParam that will be used for parallelization. The default is null, in which case the value of cores determines the behaviour of BERT. |
| assayname | User-defined string that specifies, which assay to select, if the input data is a SummarizedExperiment. The default is NULL. |

## Value

A matrix/dataframe/SummarizedExperiment mirroring the shape of the input. The data will be batch-effect adjusted by BERT.

## Examples

```
# generate dataset with 1000 features, 5 batches, 10 samples per batch and
# two genotypes
data = generate_dataset(1000,5,10,0.1, 2)
corrected = BERT(data, cores=2)
```

---

# chunk_data *Chunks data into n segments with (close-to) equivalent number of batches and stores them in temporary RDS files*

---

## Description

Chunks data into n segments with (close-to) equivalent number of batches and stores them in temporary RDS files

## Usage

```
chunk_data(data, n, backend = "default")
```

## Arguments

| | |
|---|---|
| data | Dataframe with the data to adjust |
| n | The number of chunks to create |
| backend | The backend to choose for communicating the data, Valid choices are "default" and "file". The latter will use temp files for communicating data chunks between the processes. |

## Value

Vector with the absolute paths to the temporary files, where the data is stored

## compute_asw *Compute the average silhouette width (ASW) for the dataset with respect to both label and batch.*

### Description

Columns labelled Batch, Sample, Label, Reference and Cov_1 will be ignored.

### Usage

```
compute_asw(dataset)
```

### Arguments

| | |
|---|---|
| dataset | Dataframe in the shape (samples, features) with additional columns Batch and Label. |

### Value

List with fields "Label" and "Batch" for the ASW with regards to Label and Batch respectively.

### Examples

```
# generate dataset with 1000 features, 5 batches, 10 samples per batch and
# two genotypes
data = generate_dataset(1000,5,10,0.1, 2)
asw = compute_asw(data)
asw
```

## count_existing *Count the number of numeric features in this dataset. Columns labeled "Batch", "Sample" or "Label" will be ignored.*

### Description

Count the number of numeric features in this dataset. Columns labeled "Batch", "Sample" or "Label" will be ignored.

### Usage

```
count_existing(dataset)
```

### Arguments

| | |
|---|---|
| dataset | Dataframe in the shape (samples, features) with optional columns "Batch", "Sample" or "Label". |

### Value

Integer indicating the number of numeric values

## Examples

```
# generate dataset with 1000 features, 5 batches, 10 samples per batch and
# two genotypes
data = generate_dataset(1000,5,10, 0.1, 2)
count_existing(data)
```

---

`format_DF` *Format the data as expected by BERT.*

---

## Description

This function is called automatically by BERT. It removes empty columns and removes a (usually very small) number of numeric values, if features are unadjustable for lack of data.

## Usage

```
format_DF(
  data,
  labelname = "Label",
  batchname = "Batch",
  referencename = "Reference",
  samplename = "Sample",
  covariatename = NULL,
  assayname = NULL
)
```

## Arguments

| | |
|---|---|
| `data` | Matrix or dataframe in the format (samples, features). |
| `labelname` | A string containing the name of the column to use as class labels. The default is "Label". |
| `batchname` | A string containing the name of the column to use as batch labels. The default is "Batch". |
| `referencename` | A string containing the name of the column to use as ref. labels. The default is "Reference". |
| `samplename` | A string containing the name of the column to use as sample name. The default is "Sample". |
| `covariatename` | A vector containing the names of columns with categorical covariables. The default is NULL, for which all columns with the pattern "Cov" will be selected. Additional column names are "Batch", "Cov_X" (were X may be any number), "Label" and "Sample". |
| `assayname` | User-defined string that specifies, which assay to select, if the input data is a SummarizedExperiment. The default is NULL. |

## Value

The formatted matrix.

`generate_dataset` *Generate dataset with batch-effects and biological labels using a simple LS model*

## Description

The data will be already in the correct format for BERT.

## Usage

```
generate_dataset(
  features,
  batches,
  samplesperbatch,
  mvstmt,
  classes,
  housekeeping = NULL,
  deterministic = FALSE
)
```

## Arguments

| | |
|---|---|
| `features` | Integer indicating the number of features (e.g. genes/proteins) in the dataset. |
| `batches` | Integer indicating the number of batches in the dataset. |
| `samplesperbatch` | Integer indicating the number of of samples per batch. |
| `mvstmt` | Float (in [0,1)) indicating the fraction of missing values per batch. |
| `classes` | Integer indicating the number of classes in the dataset. |
| `housekeeping` | If NULL, no huosekeeping features will be simulatd. Else, housepeeping indicates the fraction of of housekeeping features. |
| `deterministic` | Whether to assigns the classes deterministically, instead of random sampling |

## Value

A dataframe containing the simulated data.

## Examples

```
# generate dataset with 1000 features, 5 batches, 10 samples per batch and
# two genotypes
data = generate_dataset(1000,5,10, 0.1, 2)
```

generate_data_covariables *Generate dataset with batch-effects and 2 classes with a specified imbalance.*

## Description

The data will be already in the correct format for BERT.

## Usage

```
generate_data_covariables(
  features,
  batches,
  samplesperbatch,
  mvstmt,
  imbalcov,
  housekeeping = NULL
)
```

## Arguments

| | |
|---|---|
| features | Integer indicating the number of features (e.g. genes/proteins) in the dataset. |
| batches | Integer indicating the number of batches in the dataset. |
| samplesperbatch | Integer indicating the number of of samples per batch. |
| mvstmt | Float (in [0,1)) indicating the fraction of missing values per batch. |
| imbalcov | Float indicating the probability for one of the classes to be drawn as class label for each sample. The second class will have probability of 1-imbalcov |
| housekeeping | If NULL, no huosekeeping features will be simulatd. Else, housepeeping indicates the fraction of of housekeeping features. |

## Value

A dataframe containing the simulated data. Column Cov_1 will contain the simulated, imbalanced labels.

## Examples

```
# generate dataset with 1000 features, 5 batches, 10 samples per batch and
# two genotypes. The class ratio will either be 7:3 or 3:7 per batch.
data = generate_data_covariables(1000,5,10, 0.1, 0.3)
```

`get_adjustable_features` *Check, which features contain enough numeric data to be adjusted (at least 2 numeric values)*

## Description

This function will be called automatically be BERT on data from each batch independently.

## Usage

```
get_adjustable_features(data_batch)
```

## Arguments

| | |
|---|---|
| `data_batch` | Matrix or dataframe in the format (samples, features). Additional column names are "Batch", "Cov_X" (were X may be any number), "Label", "Reference" and "Sample". |

## Value

A logical with TRUE for adjustable features and FALSE for features with too many missing values.

`get_adjustable_features_with_mod` *Check, which features contain enough numeric data to be adjusted (at least 2 numeric values per batch and covariate level)*

## Description

This function will be called automatically be BERT n data from each batch independently.

## Usage

```
get_adjustable_features_with_mod(data_batch, mod_batch)
```

## Arguments

| | |
|---|---|
| `data_batch` | Matrix or dataframe in the format (samples, features). Additional column names are "Batch", "Cov_X" (were X may be any number), "Label" and "Sample". |
| `mod_batch` | Matrix or dataframe in the format (samples, covariates). Contains only the covariates as covariates. |

## Value

A logical with TRUE for adjustable features and FALSE for features with too many missing values.

## identify_adjustableFeatures_refs

*Identifies the adjustable features using only the references. Similar to the function in adjust_features.R but with different arguments*

### Description

Identifies the adjustable features using only the references. Similar to the function in adjust_features.R but with different arguments

### Usage

```
identify_adjustableFeatures_refs(x, batch, idx)
```

### Arguments

| | |
|---|---|
| x | the data matrix |
| batch | the list with the batches |
| idx | the vector indicating whether the respective sample is to be used as references |

### Value

vector indicating whether each feature can be adjusted

## identify_references

*Identifies the references to use for this specific batch effect adjustment*

### Description

Identifies the references to use for this specific batch effect adjustment

### Usage

```
identify_references(batch, references)
```

### Arguments

| | |
|---|---|
| batch | vector of batch numbers. Must contain 2 unique elements |
| references | vector that contains 0, if the sample is to be c-adjusted and a class otherwise |

### Value

the indices of the reference samples

`ordinal_encode` *Ordinal encoding of a vector.*

## Description

This function is usually called by BERT during formatting of the input. The idea is, that Label, Batch and Covariables should only be integers

## Usage

```
ordinal_encode(column)
```

## Arguments

| | |
|---|---|
| `column` | The categorical vector |

## Value

The encoded vector

`parallel_bert` *Adjusts all chunks of data (in parallel) as far as possible.*

## Description

Adjusts all chunks of data (in parallel) as far as possible.

## Usage

```
parallel_bert(
  chunks,
  BPPARAM = BiocParallel::bpparam(),
  method = "ComBat",
  combatmode = 1,
  backend = "default"
)
```

## Arguments

| | |
|---|---|
| `chunks` | vector with the filenames to the temp files where the sub-matrices are stored |
| `BPPARAM` | The BiocParallel backend to use. The default is the currently registered backend. |
| `method` | the BE-correction method to use. Possible choices are ComBat and limma |
| `combatmode` | The mode to use for combat (ignored if limma). Encoded options 'are the same as for HarmonizR |
| `backend` | The backend to choose for communicating the data, Valid choices are "default" and "file". The latter will use temp files for communicating data chunks between the processes. |

## Value

dataframe with the adjusted matrix

## removeBatchEffectRefs *A method to remove batch effects estimated from a subset (references) per batch only. Source code is heavily based on limma::removeBatchEffects by Gordon Smyth and Carolyn de Graaf*

### Description

A method to remove batch effects estimated from a subset (references) per batch only. Source code is heavily based on limma::removeBatchEffects by Gordon Smyth and Carolyn de Graaf

### Usage

```
removeBatchEffectRefs(x, batch, references)
```

### Arguments

| | |
|---|---|
| x | the data matrix with samples in columns and features in rows |
| batch | the batch list as vector. |
| references | a vector of integers, indicating whether the corresponding sample is to be co-adjusted (0) or may be used as a reference (>0) |

### Value

the corrected data matrix

## replace_missing *Replaces missing values (NaN) by NA, this appears to be faster*

### Description

Replaces missing values (NaN) by NA, this appears to be faster

### Usage

```
replace_missing(data)
```

### Arguments

| | |
|---|---|
| data | The data as dataframe |

### Value

The data with the replaced MVs

## strip_Covariable *Strip column labelled Cov_1 from dataframe.*

### Description

Strip column labelled Cov_1 from dataframe.

### Usage

```
strip_Covariable(dataset)
```

### Arguments

| | |
|---|---|
| dataset | Dataframe in the shape (samples, features) with additional column Cov_1 |

### Value

Dataset without column Cov_1.

## validate_bert_input *Verifies that the input to BERT is valid.*

### Description

Verifies that the input to BERT is valid.

### Usage

```
validate_bert_input(
  data,
  cores,
  combatmode,
  corereduction,
  stopParBatches,
  backend,
  method,
  qualitycontrol,
  verify,
  labelname,
  batchname,
  referencename,
  samplename,
  covariatename,
  assayname
)
```

## Arguments

| | |
|---|---|
| data | Matrix dataframe/SummarizedExperiment in the format (samples, features). Additional column names are "Batch", "Cov_X" (were X may be any number), "Label", "Sample" and "Reference". Must contain at least two features. |
| cores | The number of cores to use for parallel adjustment. Increasing this number leads to faster adjustment, especially on Linux machines. The default is 1. |
| combatmode | Integer, encoding the parameters to use for ComBat. 1 (default) par.prior = TRUE, mean.only = FALSE 2 par.prior = TRUE, mean.only = TRUE 3 par.prior = FALSE, mean.only = FALSE 4 par.prior = FALSE, mean.only = TRUE Will be ignored, if method!="ComBat". |
| corereduction | Reducing the number of workers by at least this number |
| stopParBatches | The minimum number of batches required at a hierarchy level to proceed with parallelized adjustment. If the number of batches is smaller, adjustment will be performed sequentially to avoid overheads. |
| backend | The backend to choose for communicating the data. Valid choices are "default" and "file". The latter will use temp files for communicating data chunks between the processes. after adjusting all sub-trees as far as possible with the previous number of cores. |
| method | Adjustment method to use. Should either be "ComBat", "limma" or "ref". Also allows "None" for testing purposes, which will perform no BE adjustment |
| qualitycontrol | Boolean indicating, whether ASWs should be computed before and after batch effect adjustment. If TRUE, will compute ASW with respect to the "Batch" and "Label" column (if existent). |
| verify | Whether the input matrix/dataframe needs to be verified before adjustment (faster if FALSE) |
| labelname | A string containing the name of the column to use as class labels. The default is "Label". |
| batchname | A string containing the name of the column to use as batch labels. The default is "Batch". |
| referencename | A string containing the name of the column to use as ref. labels. The default is "Reference". |
| samplename | A string containing the name of the column to use as sample name. The default is "Sample". |
| covariatename | A vector containing the names of columns with categorical covariables. The default is NULL, for which all columns with the pattern "Cov" will be selected. |
| assayname | User-defined string that specifies, which assay to select, if the input data is a SummarizedExperiment. The default is NULL. |

## Value

None. Will instead throw an error, if input is not as intended.

validate_input_generate_dataset *Validate the user input to the function generate_dataset. Raises an error if and only if the input is malformatted.*

## Description

Validate the user input to the function generate_dataset. Raises an error if and only if the input is malformatted.

## Usage

```
validate_input_generate_dataset(
  features,
  batches,
  samplesperbatch,
  mvstmt,
  classes,
  housekeeping,
  deterministic
)
```

## Arguments

| | |
|---|---|
| features | Integer indicating the number of features (e.g. genes/proteins) in the dataset. |
| batches | Integer indicating the number of batches in the dataset. |
| samplesperbatch | Integer indicating the number of of samples per batch. |
| mvstmt | Float (in [0,1)) indicating the fraction of missing values per batch. |
| classes | Integer indicating the number of classes in the dataset. |
| housekeeping | If NULL, no huosekeeping features will be simulatd. Else, housepeeping indicates the fraction of of housekeeping features. |
| deterministic | Whether to assigns the classes deterministically, instead of random sampling |

## Value

None

verify_references *Verify that the Reference column of the data contains only zeros and ones (if it is present at all)*

## Description

Verify that the Reference column of the data contains only zeros and ones (if it is present at all)

## Usage

```
verify_references(batch)
```

## Arguments

| | |
|---|---|
| batch | the dataframe for this batch (samples in rows, samples in columns) |

## Value

either TRUE (everything correct) or FALSE (something is not correct)

# Index