# Package 'Melissa'

November 30, 2024

**Type** Package

**Title** Bayesian clustering and imputationa of single cell methylomes

**Version** 1.23.0

**Description** Melissa is a Baysian probabilistic model for jointly clustering and imputing single cell methylomes. This is done by taking into account local correlations via a Generalised Linear Model approach and global similarities using a mixture modelling approach.

**Depends** R (>= 3.5.0), BPRMeth, GenomicRanges

**License** GPL-3 | file LICENSE

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.1.0

**Imports** data.table, parallel, ROCR, matrixcalc, mclust, ggplot2, doParallel, foreach, MCMCpack, cowplot, magrittr, mvtnorm, truncnorm, assertthat, BiocStyle, stats, utils

**Suggests** testthat, knitr, rmarkdown

**VignetteBuilder** knitr

**biocViews** ImmunoOncology, DNAMethylation, GeneExpression, GeneRegulation, Epigenetics, Genetics, Clustering, FeatureExtraction, Regression, RNASeq, Bayesian, KEGG, Sequencing, Coverage, SingleCell

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

**git_branch** devel

**git_last_commit** cffba8a

**git_last_commit_date** 2024-10-29

**Repository** Bioconductor 3.21

**Date/Publication** 2024-11-29

**Author** C. A. Kapourani [aut, cre]

**Maintainer** C. A. Kapourani <kapouranis.andreas@gmail.com>

# Contents



---

`binarise_files` *Binarise CpG sites*

---

## Description

Script for binarising CpG sites and formatting the coverage file so it can be directly used from the BPRMeth package. The format of each file is the following: <chr> <start> <met_level>, where met_level can be either 0 or 1. To read compressed files, e.g ending in .gz or .bz2, the R.utils package needs to be installed.

## Usage

```
binarise_files(indir, outdir = NULL, format = 1, no_cores = NULL)
```

## Arguments

| | |
|---|---|
| `indir` | Directory containing the coverage files, output from Bismark. |
| `outdir` | Directory to store the output files for each cell with exactly the same name. If NULL, then a directory called ‘binarised‘ inside ‘indir‘ will be create by default. |
| `format` | Integer, denoting the format of coverage file. When set to ‘1‘, the coverage file format is assumed to be: "<chr> <start> <end> <met_prcg> <met_reads> <unmet_reads>". When set to ‘2‘, then the format is assumed to be: "<chr> <start> <met_prcg> <met_reads> <unmet_reads>". |
| `no_cores` | Number of cores to use for parallel processing. If NULL, no parallel processing is used. |

## Value

No value is returned, the binarised data are stored in the outdir.

## Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

## See Also

create_melissa_data_obj, melissa, filter_regions

## Examples

```
## Not run:
# Met directory
met_dir <- "name_of_met_dir"

binarise_files(met_dir)

## End(Not run)
```

---

# cluster_ari *Compute clustering ARI*

---

## Description

cluster_ari computes the clustering performance in terms of the Adjusted Rand Index (ARI) metric.

## Usage

```
cluster_ari(C_true, C_post)
```

## Arguments

| | |
|---|---|
| C_true | True cluster assignemnts. |
| C_post | Posterior responsibilities of predicted cluster assignemnts. |

## Value

The clustering ARI.

## Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

`cluster_error` *Compute clustering assignment error* `cluster_error` *computes the clustering assignment error, i.e. the average number of incorrect cluster assignments:*

$$OE = \sum\_n = 1\^N(I(LT\_n \neq LP\_n))/N$$

## Description

Compute clustering assignment error

`cluster_error` computes the clustering assignment error, i.e. the average number of incorrect cluster assignments:

$$OE = \sum_{n=1}^{N}(I(LT_n \neq LP_n))/N$$

## Usage

```
cluster_error(C_true, C_post)
```

## Arguments

| | |
|---|---|
| `C_true` | True cluster assignemnts. |
| `C_post` | Posterior mean of predicted cluster assignemnts. |

## Value

The clustering assignment error

## Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

`create_melissa_data_obj` *Create methylation regions for all cells*

## Description

Wrapper function for creating methylation regions for all cells, which is the input object for Melissa prior to filtering.

## Usage

```
create_melissa_data_obj(
  met_dir,
  anno_file,
  chrom_size_file = NULL,
  chr_discarded = NULL,
  is_centre = FALSE,
  is_window = TRUE,
  upstream = -5000,
  downstream = 5000,
  cov = 5,
  sd_thresh = -1,
  no_cores = NULL
)
```

## Arguments

| | |
|---|---|
| met_dir | Directory of (binarised) methylation files, each file corresponds to a single cell. |
| anno_file | The annotation file with ‘tab‘ delimited format: "chromosome", "start", "end", "strand", "id", "name" (optional). Read the ‘BPRMeth‘ documentation for more details. |
| chrom_size_file | Optional file name to read genome chromosome sizes. |
| chr_discarded | Optional vector with chromosomes to be discarded. |
| is_centre | Logical, whether ’start’ and ’end’ locations are pre-centred. If TRUE, the mean of the locations will be chosen as centre. If FALSE, the ’start’ will be chosen as the center; e.g. for genes the ’start’ denotes the TSS and we use this as centre to obtain K-bp upstream and downstream of TSS. |
| is_window | Whether to consider a predefined window region around centre. If TRUE, then ’upstream’ and ’downstream’ parameters are used, otherwise we consider the whole region from start to end location. |
| upstream | Integer defining the length of bp upstream of ’centre’ for creating the genomic region. If is_window = FALSE, this parameter is ignored. |
| downstream | Integer defining the length of bp downstream of ’centre’ for creating the genomic region. If is_window = FALSE, this parameter is ignored. |
| cov | Integer defining the minimum coverage of CpGs that each region must contain. |
| sd_thresh | Optional numeric defining the minimum standard deviation of the methylation change in a region. This is used to filter regions with no methylation variability. |
| no_cores | Number of cores to be used for parallel processing of data. |

## Value

A `melissa_data_obj` object, with the following elements:

- `met`: A list of elements of length N, where N are the total number of cells. Each element in the list contains another list of length M, where M is the total number of genomic regions, e.g.

promoters. Each element in the inner list is an `I X 2` matrix, where I are the total number of observations. The first column contains the input observations x (i.e. CpG locations) and the 2nd column contains the corresponding methylation level.

- `anno_region`: The annotation object.
- `opts`: A list with the parameters that were used for creating the object.

## Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

## See Also

`binarise_files`, `melissa`, `filter_regions`

## Examples

```
## Not run:
# Met directory
met_dir <- "name_of_met_dir"
# Annotation file name
anno_file <- "name_of_anno_file"

obj <- create_melissa_data_obj(met_dir, anno_file)

# Extract annotation regions
met <- obj$met

# Extract annotation regions
anno <- obj$anno_region

## End(Not run)
```

---

# eval_cluster_performance

*Evaluate clustering performance*

---

## Description

`eval_cluster_performance` is a wrapper function for computing clustering performance in terms of ARI and clustering assignment error.

## Usage

```
eval_cluster_performance(obj, C_true)
```

### Arguments

| | |
|---|---|
| obj | Output of Melissa inference object. |
| C_true | True cluster assignemnts. |

### Value

The 'melissa' object, with an additional slot named 'clustering', containing the ARI and clustering assignment error performance.

### Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

### See Also

create_melissa_data_obj, melissa, filter_regions, eval_imputation_performance, eval_cluster_performance

### Examples

```
## Extract synthetic data
dt <- melissa_synth_dt

# Partition to train and test set
dt <- partition_dataset(dt)

# Create basis object from BPRMeth package
basis_obj <- BPRMeth::create_rbf_object(M = 3)

# Run Melissa
melissa_obj <- melissa(X = dt$met, K = 2, basis = basis_obj, vb_max_iter = 10,
  vb_init_nstart = 1, is_parallel = FALSE, is_verbose = FALSE)

# Compute cluster performance
melissa_obj <- eval_cluster_performance(melissa_obj, dt$opts$C_true)

cat("ARI: ", melissa_obj$clustering$ari)
```

---

## eval_imputation_performance

*Evaluate imputation performance*

---

### Description

eval_imputation_performance is a wrapper function for computing imputation/clustering performance in terms of different metrics, such as AUC and precision recall curves.

## Usage

```
eval_imputation_performance(obj, imputation_obj)
```

## Arguments

| | |
|---|---|
| obj | Output of Melissa inference object. |
| imputation_obj | List containing two vectors of equal length, corresponding to true methylation states and predicted/imputed methylation states. |

## Value

The 'melissa' object, with an additional slot named 'imputation', containing the AUC, F-measure, True Positive Rate (TPR) and False Positive Rate (FPR), and Precision Recall (PR) curves.

## Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

## See Also

create_melissa_data_obj, melissa, impute_test_met, filter_regions, eval_imputation_performance, eval_cluster_performance

## Examples

```
# First take a subset of cells to efficiency
# Extract synthetic data
dt <- melissa_synth_dt

# Partition to train and test set
dt <- partition_dataset(dt)

# Create basis object from BPRMeth package
basis_obj <- BPRMeth::create_rbf_object(M = 3)

# Run Melissa
melissa_obj <- melissa(X = dt$met, K = 2, basis = basis_obj, vb_max_iter = 10,
  vb_init_nstart = 1, is_parallel = FALSE, is_verbose = FALSE)

imputation_obj <- impute_test_met(obj = melissa_obj, test = dt$met_test)

melissa_obj <- eval_imputation_performance(obj = melissa_obj,
                                           imputation_obj = imputation_obj)

cat("AUC: ", melissa_obj$imputation$auc)
```

## extract_y *Extract responses y*

### Description

Given a list of observations, extract responses y

### Usage

```
extract_y(X, coverage_ind)
```

### Arguments

| | |
|---|---|
| X | Observations |
| coverage_ind | Which observations have coverage |

### Value

The design matrix H

## filter_regions *Filtering process prior to running Melissa*

### Description

Fuctions for filter genomic regions due to (1) low CpG coverage, (2) low coverage across cells, or (3) low mean methylation variability.

### Usage

```
filter_by_cpg_coverage(obj, min_cpgcov = 10)

filter_by_coverage_across_cells(obj, min_cell_cov_prcg = 0.5)

filter_by_variability(obj, min_var = 0.1)
```

### Arguments

| | |
|---|---|
| obj | Melissa data object. |
| min_cpgcov | Minimum CpG coverage for each genomic region. |
| min_cell_cov_prcg | Threshold on the proportion of cells that have coverage for each region. |
| min_var | Minimum variability of mean methylation across cells, measured in terms of standard deviation. |

## Details

The (1) ‘filter_by_cpg_coverage‘ function does not actually remove the region, it only sets NA to those regions. The (2) ‘filter_by_coverage_across_cells‘ function keeps regions from which we can share information across cells. The (3) ‘filter_by_variability‘ function keeps variable regions which are informative for cell subtype identification.

## Value

The filtered Melissa data object

## Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

## See Also

melissa, create_melissa_data_obj

## Examples

```
# Run on synthetic data from Melissa package
filt_obj <- filter_by_cpg_coverage(melissa_encode_dt, min_cpgcov = 20)

# Run on synthetic data from Melissa package
filt_obj <- filter_by_coverage_across_cells(melissa_encode_dt,
                                            min_cell_cov_prcg = 0.7)

# Run on synthetic data from Melissa package
filt_obj <- filter_by_variability(melissa_encode_dt, min_var = 0.1)
```

---

# impute_met_files *Impute/predict methylation files*

---

## Description

Make predictions of missing methylation states, i.e. perfrom imputation using Melissa. Each file in the directory will be used as input and a new file will be created in `outdir` with an additional column containing the predicted met state (value between 0 and 1). Note that predictions will be made only on `annotation regions` that were used for training Melissa. Check impute_test_met, if you want to make predictions only on test data.

## Usage

```
impute_met_files(
  met_dir,
  outdir = NULL,
  obj,
  anno_region,
  basis = NULL,
  is_predictive = TRUE,
  no_cores = NULL
)
```

## Arguments

| | |
|---|---|
| met_dir | Directory of methylation files, each file corresponds to a single cell. It should contain three columns <chr> <pos> <met_state> (similar to the input required by create_melissa_data_obj), where met_state can be any value that denotes missing CpG information, e.g. -1. Note that files can contain also CpGs for which we have coverage information, and we can check the predictions made by Melissa, hence the value can also be 0 (unmet) or (1) met. Predictions made by Melissa, will not change the <met_state> column. Melissa will just add an additional column named <predicted>. |
| outdir | Directory to store the output files for each cell with exactly the same name. If NULL, then a directory called 'imputed' inside 'met_dir' will be created by default. |
| obj | Output of Melissa inference object. |
| anno_region | Annotation region object. This will be the outpuf of create_melissa_data_obj function, e..g melissa_data$anno_region. This is required to select those regions that were used to train Melissa. |
| basis | Basis object, if NULL we perform imputation using Melissa, otherwise using BPRMeth (then obj should be BPRMeth output). |
| is_predictive | Logical, use predictive distribution for imputation, or choose the cluster label with the highest responsibility. |
| no_cores | Number of cores to be used for parallel processing of data. |

## Value

A new directory outdir containing files (cells) with predicted / imputed methylation states per CpG location.

## Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

## See Also

create_melissa_data_obj, melissa, filter_regions

## Examples

```
## Not run:
# Met directory
met_dir <- "name_of_met_dir"
# Annotation file name
anno_file <- "name_of_anno_file"
# Create data object
melissa_data <- create_melissa_data_obj(met_dir, anno_file)
# Run Melissa
melissa_obj <- melissa(X = melissa_data$met, K = 2)
# Annotation object
anno_region <- melissa_data$anno_region

# Peform imputation
impute_met_dir <- "name_of_met_dir_for_imputing_cells"
out <- impute_met_files(met_dir = impute_met_dir, obj = melissa_obj,
                        anno_region = anno_region)

## End(Not run)
```

---

`impute_test_met` *Impute/predict test methylation states*

---

## Description

Make predictions of missing methylation states, i.e. perfrom imputation using Melissa. This requires keepin a subset of data as a held out test set during Melissa inference. If you want to impute a whole directory containing cells (files) with missing methylation levels, see `impute_met_files`.

## Usage

```
impute_test_met(
  obj,
  test,
  basis = NULL,
  is_predictive = TRUE,
  return_test = FALSE
)
```

## Arguments

| | |
|---|---|
| `obj` | Output of Melissa inference object. |
| `test` | Test data to evaluate performance. |
| `basis` | Basis object, if NULL we perform imputation using Melissa, otherwise using BPRMeth. |

| | |
|---|---|
| is_predictive | Logical, use predictive distribution for imputation, or choose the cluster label with the highest responsibility. |
| return_test | Whether or not to return a list with the predictions. |

## Value

A list containing two vectors, the true methylation state and the predicted/imputed methylation states.

## Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

## See Also

create_melissa_data_obj, melissa, filter_regions, eval_imputation_performance, eval_cluster_performance

## Examples

```
# Extract synthetic data
dt <- melissa_synth_dt

# Partition to train and test set
dt <- partition_dataset(dt)

# Create basis object from BPRMeth package
basis_obj <- BPRMeth::create_rbf_object(M = 3)

# Run Melissa
melissa_obj <- melissa(X = dt$met, K = 2, basis = basis_obj, vb_max_iter=10,
   vb_init_nstart = 1, is_parallel = FALSE, is_verbose = FALSE)

imputation_obj <- impute_test_met(obj = melissa_obj,
                                  test = dt$met_test)
```

---

# init_design_matrix *Initialise design matrices*

---

## Description

Given a list of observations, initialise design matrices H for computational efficiency.

## Usage

```
init_design_matrix(basis, X, coverage_ind)
```

### Arguments

| | |
|---|---|
| basis | Basis object. |
| X | Observations |
| coverage_ind | Which observations have coverage |

### Value

The design matrix H

---

## log_sum_exp *Compute stable log-sum-exp*

---

### Description

log_sum_exp computes the log sum exp trick for avoiding numeric underflow and have numeric stability in computations of small numbers.

### Usage

```
log_sum_exp(x)
```

### Arguments

| | |
|---|---|
| x | A vector of observations |

### Value

The logs-sum-exp value

### Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

### References

https://hips.seas.harvard.edu/blog/2013/01/09/computing-log-sum-exp/

## Melissa — *Melissa: Bayesian clustering and imputation of single cell methylomes*

### Description

Bayesian clustering and imputation of single cell methylomes

### Usage

```
.datatable.aware
```

### Format

An object of class `logical` of length 1.

### Value

Melissa main package documentation.

### Author(s)

C.A.Kapourani <kapouranis.andreas@gmail.com>

### See Also

`melissa`, `create_melissa_data_obj`, `partition_dataset`, `plot_melissa_profiles`, `filter_regions`

## melissa — *Cluster and impute single cell methylomes using VB*

### Description

`melissa` clusters and imputes single cells based on their methylome landscape on specific genomic regions, e.g. promoters, using the Variational Bayes (VB) EM-like algorithm.

### Usage

```
melissa(
  X,
  K = 3,
  basis = NULL,
  delta_0 = NULL,
  w = NULL,
  alpha_0 = 0.5,
  beta_0 = NULL,
```

```
  vb_max_iter = 300,
  epsilon_conv = 1e-05,
  is_kmeans = TRUE,
  vb_init_nstart = 10,
  vb_init_max_iter = 20,
  is_parallel = FALSE,
  no_cores = 3,
  is_verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| X | The input data, which has to be a list of elements of length N, where N are the total number of cells. Each element in the list contains another list of length M, where M is the total number of genomic regions, e.g. promoters. Each element in the inner list is an I X 2 matrix, where I are the total number of observations. The first column contains the input observations x (i.e. CpG locations) and the 2nd columns contains the corresponding methylation level. |
| K | Integer denoting the total number of clusters K. |
| basis | A 'basis' object. E.g. see create_basis function from BPRMeth package. If NULL, will an RBF object with 3 basis functions will be created. |
| delta_0 | Parameter vector of the Dirichlet prior on the mixing proportions pi. |
| w | Optional, an Mx(D)xK array of the initial parameters, where first dimension are the genomic regions M, 2nd the number of covariates D (i.e. basis functions), and 3rd are the clusters K. If NULL, will be assigned with default values. |
| alpha_0 | Hyperparameter: shape parameter for Gamma distribution. A Gamma distribution is used as prior for the precision parameter tau. |
| beta_0 | Hyperparameter: rate parameter for Gamma distribution. A Gamma distribution is used as prior for the precision parameter tau. |
| vb_max_iter | Integer denoting the maximum number of VB iterations. |
| epsilon_conv | Numeric denoting the convergence threshold for VB. |
| is_kmeans | Logical, use Kmeans for initialization of model parameters. |
| vb_init_nstart | Number of VB random starts for finding better initialization. |
| vb_init_max_iter | Maximum number of mini-VB iterations. |
| is_parallel | Logical, indicating if code should be run in parallel. |
| no_cores | Number of cores to be used, default is max_no_cores - 1. |
| is_verbose | Logical, print results during VB iterations. |

## Value

An object of class melissa with the following elements:

- W: An (M+1) X K matrix with the optimized parameter values for each cluster, M are the number of basis functions. Each column of the matrix corresponds a different cluster k.

- W_Sigma: A list with the covariance matrices of the posterior parmateter W for each cluster k.
- r_nk: An (N X K) responsibility matrix of each observations being explained by a specific cluster.
- delta: Optimized Dirichlet paramter for the mixing proportions.
- alpha: Optimized shape parameter of Gamma distribution.
- beta: Optimized rate paramter of the Gamma distribution
- basis: The basis object.
- lb: The lower bound vector.
- labels: Cluster assignment labels.
- pi_k: Expected value of mixing proportions.

## Details

The modelling and mathematical details for clustering profiles using mean-field variational inference are explained here: http://rpubs.com/cakapourani/ . More specifically:

- For Binomial/Bernoulli observation model check: http://rpubs.com/cakapourani/vb-mixture-bpr
- For Gaussian observation model check: http://rpubs.com/cakapourani/vb-mixture-lr

## Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

## See Also

create_melissa_data_obj, partition_dataset, plot_melissa_profiles, impute_test_met, impute_met_files, filter_regions

## Examples

```
# Example of running Melissa on synthetic data

# Create RBF basis object with 4 RBFs
basis_obj <- BPRMeth::create_rbf_object(M = 4)

set.seed(15)
# Run Melissa
melissa_obj <- melissa(X = melissa_synth_dt$met, K = 2, basis = basis_obj,
   vb_max_iter = 10, vb_init_nstart = 1, vb_init_max_iter = 5,
   is_parallel = FALSE, is_verbose = FALSE)

# Extract mixing proportions
print(melissa_obj$pi_k)
```

## melissa_encode_dt *Synthetic ENCODE single cell methylation data*

### Description

Small synthetic ENCODE data generated by inferring methylation profiles from bulk ENCODE data, and subsequently generating single cells. It consists of N = 200 cells and M = 100 genomic regions. The data are in the required format for directly running Melissa and are used as a case study for the vignette.

### Usage

```
melissa_encode_dt
```

### Format

A list object containing methylation regions, annotation data and the options used for creating the data. This in general would be the output of the `create_melissa_data_obj` function. It has the following three objects:

- `met`: A list containing the methylation data, each element of the list is a different cell.
- `anno_region`: Corresponding annotation data for each genomic region.
- `opts`: Parameters/options used to generate the data.

### Value

Synthetic ENCODE methylation data

### See Also

`create_melissa_data_obj`

## melissa_gibbs *Gibbs sampling algorithm for Melissa model*

### Description

`melissa_gibbs` implements the Gibbs sampling algorithm for performing clustering of single cells based on their DNA methylation profiles, where the observation model is the Bernoulli distributed Probit Regression likelihood. NOTE: that Gibbs sampling is really slow and we recommend using the VB implementation: `melissa`.

## Usage

```
melissa_gibbs(
  X,
  K = 2,
  pi_k = rep(1/K, K),
  w = NULL,
  basis = NULL,
  w_0_mean = NULL,
  w_0_cov = NULL,
  dir_a = rep(1, K),
  lambda = 1/2,
  gibbs_nsim = 1000,
  gibbs_burn_in = 200,
  inner_gibbs = FALSE,
  gibbs_inner_nsim = 50,
  is_parallel = TRUE,
  no_cores = NULL,
  is_verbose = FALSE
)
```

## Arguments

| | |
|---|---|
| X | A list of length I, where I are the total number of cells. Each element of the list contains another list of length N, where N is the total number of genomic regions. Each element of the inner list is an L x 2 matrix of observations, where 1st column contains the locations and the 2nd column contains the methylation level of the corresponding CpGs. |
| K | Integer denoting the number of clusters K. |
| pi_k | Vector of length K, denoting the mixing proportions. |
| w | A N x M x K array, where each column contains the basis function coefficients for the corresponding cluster. |
| basis | A 'basis' object. E.g. see create_rbf_object from BPRMeth package |
| w_0_mean | The prior mean hyperparameter for w |
| w_0_cov | The prior covariance hyperparameter for w |
| dir_a | The Dirichlet concentration parameter, prior over pi_k |
| lambda | The complexity penalty coefficient for penalized regression. |
| gibbs_nsim | Argument giving the number of simulations of the Gibbs sampler. |
| gibbs_burn_in | Argument giving the burn in period of the Gibbs sampler. |
| inner_gibbs | Logical, indicating if we should perform Gibbs sampling to sample from the augmented BPR model. |
| gibbs_inner_nsim | Number of inner Gibbs simulations. |
| is_parallel | Logical, indicating if code should be run in parallel. |
| no_cores | Number of cores to be used, default is max_no_cores - 1. |
| is_verbose | Logical, print results during EM iterations |

## Value

An object of class `melissa_gibbs`.

## Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

## See Also

`melissa`, `create_melissa_data_obj`, `partition_dataset`, `filter_regions`

## Examples

```
# Example of running Melissa Gibbs on synthetic data

# Create RBF basis object with 4 RBFs
basis_obj <- BPRMeth::create_rbf_object(M = 4)

set.seed(15)
# Run Melissa Gibbs
melissa_obj <- melissa_gibbs(X = melissa_synth_dt$met, K = 2, basis = basis_obj,
   gibbs_nsim = 10, gibbs_burn_in = 5, is_parallel = FALSE, is_verbose = FALSE)

# Extract mixing proportions
print(melissa_obj$pi_k)
```

---

# melissa_synth_dt *Synthetic single cell methylation data*

---

## Description

Small synthetic data for quick analysis. It consists of N = 50 cells and M = 50 genomic regions.

## Usage

```
melissa_synth_dt
```

## Format

A list object containing methylation regions, annotation data and the options used for creating the data. This in general would be the output of the `create_melissa_data_obj` function. It has the following three objects:

- `met`: A list containing the methylation data, each element of the list is a different cell.
- `anno_region`: Corresponding annotation data for each genomic region.
- `opts`: Parameters/options used to generate the data.

## Value

Synthetic methylation data

## See Also

create_melissa_data_obj

---

`partition_dataset` *Partition synthetic dataset to training and test set*

---

## Description

Partition synthetic dataset to training and test set

## Usage

```
partition_dataset(
  dt_obj,
  data_train_prcg = 0.5,
  region_train_prcg = 0.95,
  cpg_train_prcg = 0.5,
  is_synth = FALSE
)
```

## Arguments

| | |
|---|---|
| `dt_obj` | Melissa data object |
| `data_train_prcg` | Percentage of genomic regions that will be fully used for training, i.e. across the whole region we will have no CpGs missing. |
| `region_train_prcg` | Fraction of genomic regions to keep for training set, i.e. some genomic regions will have no coverage at all during training. |
| `cpg_train_prcg` | Fraction of CpGs in each genomic region to keep for training set. |
| `is_synth` | Logical, whether we have synthetic data or not. |

## Value

The Melissa object with the following changes. The ‘met‘ element will now contain only the ‘training‘ data. An additional element called ‘met_test‘ will store the data that will be used during testing to evaluate the imputation performance. These data will not be seen from Melissa during inference.

## Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

## See Also

create_melissa_data_obj, melissa, filter_regions

## Examples

```
# Partition the synthetic data from Melissa package
dt <- partition_dataset(melissa_encode_dt)
```

---

# plot_melissa_profiles *Plot predictive methylaation profiles*

---

## Description

This function plots the predictive distribution of the methylation profiles inferred using the Melissa model. Each colour corresponds to a different cluster.

## Usage

```
plot_melissa_profiles(
  melissa_obj,
  region = 1,
  title = "Melissa profiles",
  x_axis = "genomic region",
  y_axis = "met level",
  x_labels = c("Upstream", "", "Centre", "", "Downstream"),
  ...
)
```

## Arguments

| | |
|---|---|
| melissa_obj | Clustered cell subtypes using Melissa inference functions. |
| region | Genomic region number. |
| title | Plot title |
| x_axis | x axis label |
| y_axis | x axis label |
| x_labels | x axis ticks labels |
| ... | Additional parameters |

## Value

A ggplot2 object.

## Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

## See Also

create_melissa_data_obj, melissa, filter_regions, eval_imputation_performance, eval_cluster_performance

## Examples

```
# Extract synthetic data
dt <- melissa_synth_dt

# Create basis object from BPRMeth package
basis_obj <- BPRMeth::create_rbf_object(M = 3)

# Run Melissa
melissa_obj <- melissa(X = dt$met, K = 2, basis = basis_obj, vb_max_iter = 10,
   vb_init_nstart = 1, is_parallel = FALSE, is_verbose = FALSE)

gg <- plot_melissa_profiles(melissa_obj, region = 10)
```

# Index