

# Package ‘methimpute’

December 17, 2025

**Type** Package

**Title** Imputation-guided re-construction of complete methylomes from WGBS data

**Version** 1.33.0

**Author** Aaron Taudt

**Maintainer** Aaron Taudt <aaron.taudt@gmail.com>

**Description**

This package implements functions for calling methylation for all cytosines in the genome.

**Depends** R (>= 3.4.0), GenomicRanges, ggplot2

**Imports** Rcpp (>= 0.12.4.5), methods, utils, grDevices, stats, GenomeInfoDb, IRanges, Biostrings, reshape2, minpack.lm, data.table

**Suggests** knitr, BiocStyle

**LinkingTo** Rcpp

**License** Artistic-2.0

**LazyLoad** yes

**VignetteBuilder** knitr

**RoxygenNote** 6.1.1

**biocViews** ImmunoOncology, Software, DNAMethylation, Epigenetics, HiddenMarkovModel, Sequencing, Coverage

**git\_url** <https://git.bioconductor.org/packages/methimpute>

**git\_branch** devel

**git\_last\_commit** ad2d2ed

**git\_last\_commit\_date** 2025-10-29

**Repository** Bioconductor 3.23

**Date/Publication** 2025-12-17

## Contents

|                                   |   |
|-----------------------------------|---|
| methimpute-package . . . . .      | 2 |
| arabidopsis_chromosomes . . . . . | 3 |
| arabidopsis_genes . . . . .       | 3 |
| arabidopsis_TEs . . . . .         | 3 |

|                                       |           |
|---------------------------------------|-----------|
| arabidopsis_toydata . . . . .         | 4         |
| binning . . . . .                     | 4         |
| binomialTestMethylation . . . . .     | 5         |
| callMethylation . . . . .             | 6         |
| callMethylationSeparate . . . . .     | 7         |
| collapseBins . . . . .                | 8         |
| distanceCorrelation . . . . .         | 10        |
| estimateTransDist . . . . .           | 10        |
| exportMethylome . . . . .             | 11        |
| extractCytosinesFromFASTA . . . . .   | 12        |
| getDistinctColors . . . . .           | 13        |
| getPosteriors . . . . .               | 14        |
| getStateColors . . . . .              | 14        |
| import . . . . .                      | 15        |
| importRene . . . . .                  | 16        |
| inflateMethylome . . . . .            | 17        |
| loadFromFiles . . . . .               | 17        |
| methimpute-objects . . . . .          | 18        |
| methimputeBinomialHMM . . . . .       | 18        |
| methimputeData . . . . .              | 19        |
| parameterScan . . . . .               | 19        |
| plotting . . . . .                    | 20        |
| print.methimputeBinomialHMM . . . . . | 21        |
| transCoord . . . . .                  | 22        |
| <b>Index</b>                          | <b>23</b> |

---

|                    |                                                                              |
|--------------------|------------------------------------------------------------------------------|
| methimpute-package | <i>methIMPUTE: Imputation-guided methylation status calling for WGBS-seq</i> |
|--------------------|------------------------------------------------------------------------------|

---

## Description

**methimpute** is an R-package for methylation status calling in Whole-Genome Bisulfite-sequencing (WGBS-seq) data. Its powerful Hidden Markov model implementation enables imputation of methylation status calls for cytosines without any coverage.

## Details

Please read the vignette for a tutorial on how to use this package. You can do this by typing `browseVignettes("methimpute")`. Here is an overview of all [plotting](#) functions.

## Author(s)

Aaron Taudt

---

`arabidopsis_chromosomes`*Chromosome lengths for Arabidopsis*

---

**Description**

A `data.frame` with chromosome lengths for Arabidopsis.

**Format**

A `data.frame`.

**Examples**

```
data(arabidopsis_chromosomes)
print(arabidopsis_chromosomes)
```

---

`arabidopsis_genes`*Gene coordinates for Arabidopsis (chr1)*

---

**Description**

A [GRanges-class](#) object for demonstration purposes in examples of package [methimpute](#). The object contains gene coordinates of chr1 from Arabidopsis.

**Format**

A [GRanges-class](#) object.

**Examples**

```
data(arabidopsis_genes)
print(arabidopsis_genes)
```

---

`arabidopsis_TEs`*Transposable element coordinates for Arabidopsis (chr1)*

---

**Description**

A [GRanges-class](#) object for demonstration purposes in examples of package [methimpute](#). The object contains transposable element coordinates of chr1 from Arabidopsis.

**Format**

A [GRanges-class](#) object.

**Examples**

```
data(arabidopsis_TEs)
print(arabidopsis_TEs)
```

---

|                     |                                                     |
|---------------------|-----------------------------------------------------|
| arabidopsis_toydata | <i>Toy data for Arabidopsis (200.000bp of chr1)</i> |
|---------------------|-----------------------------------------------------|

---

### Description

A `methimputeData` object for demonstration purposes in examples of package `methimpute`. The object contains the first 200.000 cytosines of chr1 from Arabidopsis.

### Format

A `methimputeData` object.

### Examples

```
data(arabidopsis_toydata)
print(arabidopsis_toydata)
```

---

|         |                                     |
|---------|-------------------------------------|
| binning | <i>Methimpute binning functions</i> |
|---------|-------------------------------------|

---

### Description

This page provides an overview of all `methimpute` binning functions.

### Usage

```
binCounts(data, binsize)

binPositions(data, binsize)

binMethylome(data, binsize, contexts = "total", columns.average = NULL)
```

### Arguments

|                 |                                                                                                                                             |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| data            | A <code>GRanges-class</code> object with metadata columns 'context' and 'counts' (which is a matrix with columns 'methylated' and 'total'). |
| binsize         | The window size used for binning.                                                                                                           |
| contexts        | A character vector with contexts for which the binning will be done.                                                                        |
| columns.average | A character vector with names of columns in data that should be averaged in bins.                                                           |

### Value

A `GRanges-class` object for `binCounts` and `binPostions`. A `list()` of `GRanges-class` objects for `binMethylome`.

## Functions

- `binCounts`: Get the aggregated number of counts in each bin (no context).
- `binPositions`: Get the number of cytosines in each bin (total and per context).
- `binMethylome`: Get number of cytosines and aggregated counts for the specified contexts.

## Examples

```
## Get some toy data
file <- system.file("data", "arabidopsis_toydata.RData",
                    package="methimpute")
data <- get(load(file))
print(data)
## Bin the data in various ways
binCounts(data, binsize=1000)
binPositions(data, binsize=1000)
binMethylome(data, binsize=1000, contexts=c("total", "CG"),
             columns.average=NULL)
```

---

`binomialTestMethylation`*Call methylation status*

---

## Description

Call methylation status of cytosines (or bins) with a binomial test.

## Usage

```
binomialTestMethylation(data, conversion.rate, min.coverage = 3,
                        p.threshold = 0.05)
```

## Arguments

|                              |                                                     |
|------------------------------|-----------------------------------------------------|
| <code>data</code>            | A <code>methimputeData</code> object.               |
| <code>conversion.rate</code> | A conversion rate between 0 and 1.                  |
| <code>min.coverage</code>    | Minimum coverage to consider for the binomial test. |
| <code>p.threshold</code>     | Significance threshold between 0 and 1.             |

## Details

The function uses a binomial test with the specified `conversion.rate`. P-values are then multiple testing corrected with the Benjamini & Yekutieli procedure. Methylated positions are selected with the `p.threshold`.

## Value

A vector with methylation statuses.

## Examples

```
## Get some toy data
file <- system.file("data", "arabidopsis_toydata.RData", package="methimpute")
data <- get(load(file))
data$binomial <- binomialTestMethylation(data, conversion.rate=0.998)
```

---

|                 |                                |
|-----------------|--------------------------------|
| callMethylation | <i>Call methylation status</i> |
|-----------------|--------------------------------|

---

## Description

Call methylation status of cytosines (or bins) with a Hidden Markov Model.

## Usage

```
callMethylation(data, fit.on.chrom = NULL, transDist = Inf, eps = 1,
  max.time = Inf, max.iter = Inf, count.cutoff = 500,
  verbosity = 1, num.threads = 2 + include.intermediate,
  initial.params = NULL, include.intermediate = FALSE,
  update = "independent", min.reads = 0)
```

## Arguments

|                      |                                                                                                                                                                                                                                                                                                               |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| data                 | A <a href="#">methimputeData</a> object.                                                                                                                                                                                                                                                                      |
| fit.on.chrom         | A character vector specifying the chromosomes on which the HMM will be fitted.                                                                                                                                                                                                                                |
| transDist            | The decaying constant for the distance-dependent transition matrix. Either a single numeric or a named numeric vector, where the vector names correspond to the transition contexts. Such a vector can be obtained from <a href="#">estimateTransDist</a> .                                                   |
| eps                  | Convergence threshold for the Baum-Welch algorithm.                                                                                                                                                                                                                                                           |
| max.time             | Maximum running time in seconds for the Baum-Welch algorithm.                                                                                                                                                                                                                                                 |
| max.iter             | Maximum number of iterations for the Baum-Welch algorithm.                                                                                                                                                                                                                                                    |
| count.cutoff         | A cutoff for the counts to remove artificially high counts from mapping artifacts. Set to Inf to disable this filtering (not recommended).                                                                                                                                                                    |
| verbosity            | An integer from 1 to 5 specifying the verbosity of the fitting procedure. Values > 1 are only for debugging.                                                                                                                                                                                                  |
| num.threads          | Number of CPU to use for the computation. Parallelization is implemented on the number of states, which is 2 or 3, so setting num.threads > 3 will not give additional performance increase.                                                                                                                  |
| initial.params       | A <a href="#">methimputeBinomialHMM</a> object. This parameter is useful to continue the fitting procedure for a <a href="#">methimputeBinomialHMM</a> object.                                                                                                                                                |
| include.intermediate | A logical specifying whether or not the intermediate component should be included in the HMM.                                                                                                                                                                                                                 |
| update               | One of c("independent", "constrained"). If update="independent" probability parameters for the binomial test will be updated independently. If update="constrained" the probability parameter of the intermediate component will be constrained to the mean of the unmethylated and the methylated component. |

`min.reads`      The minimum number of reads that a position must have to contribute in the Baum-Welch fitting procedure.

### Details

The Hidden Markov model uses a binomial test for the emission densities. Transition probabilities are modeled with a distance dependent decay, specified by the parameter `transDist`.

### Value

A `methimputeBinomialHMM` object.

### Examples

```
## Get some toy data
file <- system.file("data", "arabidopsis_toydata.RData", package="methimpute")
data <- get(load(file))
print(data)
model <- callMethylation(data)
print(model)
```

---

`callMethylationSeparate`

*Call methylation status*

---

### Description

Call methylation status of cytosines (or bins) with a separate Hidden Markov Model for each context.

### Usage

```
callMethylationSeparate(data, fit.on.chrom = NULL, transDist = Inf,
  eps = 1, max.time = Inf, max.iter = Inf, count.cutoff = 500,
  verbosity = 1, num.threads = 2 + include.intermediate,
  initial.params = NULL, include.intermediate = FALSE,
  update = "independent", min.reads = 0)
```

### Arguments

|                           |                                                                                                                                                                                                                                                          |
|---------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>data</code>         | A <code>methimputeData</code> object.                                                                                                                                                                                                                    |
| <code>fit.on.chrom</code> | A character vector specifying the chromosomes on which the HMM will be fitted.                                                                                                                                                                           |
| <code>transDist</code>    | The decaying constant for the distance-dependent transition matrix. Either a single numeric or a named numeric vector, where the vector names correspond to the transition contexts. Such a vector can be obtained from <code>estimateTransDist</code> . |
| <code>eps</code>          | Convergence threshold for the Baum-Welch algorithm.                                                                                                                                                                                                      |
| <code>max.time</code>     | Maximum running time in seconds for the Baum-Welch algorithm.                                                                                                                                                                                            |
| <code>max.iter</code>     | Maximum number of iterations for the Baum-Welch algorithm.                                                                                                                                                                                               |
| <code>count.cutoff</code> | A cutoff for the counts to remove artificially high counts from mapping artifacts. Set to <code>Inf</code> to disable this filtering (not recommended).                                                                                                  |

|                      |                                                                                                                                                                                                                                                                                                               |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| verbosity            | An integer from 1 to 5 specifying the verbosity of the fitting procedure. Values > 1 are only for debugging.                                                                                                                                                                                                  |
| num.threads          | Number of CPU to use for the computation. Parallelization is implemented on the number of states, which is 2 or 3, so setting num.threads > 3 will not give additional performance increase.                                                                                                                  |
| initial.params       | A <code>methimputeBinomialHMM</code> object. This parameter is useful to continue the fitting procedure for a <code>methimputeBinomialHMM</code> object.                                                                                                                                                      |
| include.intermediate | A logical specifying wheter or not the intermediate component should be included in the HMM.                                                                                                                                                                                                                  |
| update               | One of c("independent", "constrained"). If update="independent" probability parameters for the binomial test will be updated independently. If update="constrained" the probability parameter of the intermediate component will be constrained to the mean of the unmethylated and the methylated component. |
| min.reads            | The minimum number of reads that a position must have to contribute in the Baum-Welch fitting procedure.                                                                                                                                                                                                      |

## Details

The Hidden Markov model uses a binomial test for the emission densities. Transition probabilities are modeled with a distance dependent decay, specified by the parameter `transDist`.

## Value

A `methimputeBinomialHMM` object.

## Examples

```
## Get some toy data
file <- system.file("data", "arabidopsis_toydata.RData", package="methimpute")
data <- get(load(file))
print(data)
model <- callMethylationSeparate(data)
print(model)
```

---

|              |                                  |
|--------------|----------------------------------|
| collapseBins | <i>Collapse consecutive bins</i> |
|--------------|----------------------------------|

---

## Description

The function will collapse consecutive bins which have, for example, the same combinatorial state.

## Usage

```
collapseBins(data, column2collapseBy = NULL, columns2sumUp = NULL,
             columns2average = NULL, columns2getMax = NULL, columns2drop = NULL)
```



**Arguments**

|                                |                                                                                                                                                                                                                   |
|--------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>data</code>              | A data.frame containing the genomic coordinates in the first three columns.                                                                                                                                       |
| <code>column2collapseBy</code> | The number of the column which will be used to collapse all other inputs. If a set of consecutive bins has the same value in this column, they will be aggregated into one bin with adjusted genomic coordinates. |
| <code>columns2sumUp</code>     | Column numbers that will be summed during the aggregation process.                                                                                                                                                |
| <code>columns2average</code>   | Column numbers that will be averaged during the aggregation process.                                                                                                                                              |
| <code>columns2getMax</code>    | Column numbers where the maximum will be chosen during the aggregation process.                                                                                                                                   |
| <code>columns2drop</code>      | Column numbers that will be dropped after the aggregation process.                                                                                                                                                |

**Details**

The following tables illustrate the principle of the collapsing:

Input data:

| seqnames | start | end | column2collapseBy | moreColumns | columns2sumUp |
|----------|-------|-----|-------------------|-------------|---------------|
| chr1     | 0     | 199 | 2                 | 1 10        | 1 3           |
| chr1     | 200   | 399 | 2                 | 2 11        | 0 3           |
| chr1     | 400   | 599 | 2                 | 3 12        | 1 3           |
| chr1     | 600   | 799 | 1                 | 4 13        | 0 3           |
| chr1     | 800   | 999 | 1                 | 5 14        | 1 3           |

Output data:

| seqnames | start | end | column2collapseBy | moreColumns | columns2sumUp |
|----------|-------|-----|-------------------|-------------|---------------|
| chr1     | 0     | 599 | 2                 | 1 10        | 2 9           |
| chr1     | 600   | 999 | 1                 | 4 13        | 1 6           |

**Value**

A data.frame.

**Author(s)**

Aaron Taudt

**Examples**

```
## Load example data
## Get an example multiHMM
data(arabidopsis_toydata)
df <- as.data.frame(arabidopsis_toydata)
shortdf <- collapseBins(df, column2collapseBy='context', columns2sumUp='width', columns2average=7:8)
```

---

|                     |                             |
|---------------------|-----------------------------|
| distanceCorrelation | <i>Distance correlation</i> |
|---------------------|-----------------------------|

---

### Description

Compute the distance correlation from a [methimputeData](#) object.

### Usage

```
distanceCorrelation(data, distances = 0:50, separate.contexts = FALSE)
```

### Arguments

|                                |                                                                                                                                                           |
|--------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>data</code>              | A <a href="#">methimputeData</a> object.                                                                                                                  |
| <code>distances</code>         | An integer vector specifying the distances for which the correlation will be calculated.                                                                  |
| <code>separate.contexts</code> | A logical indicating whether contexts are treated separately. If set to TRUE, correlations will only be calculated between cytosines of the same context. |

### Value

A list() with an array containing the correlation values and the corresponding [ggplot](#).

### Examples

```
## Get some toy data
file <- system.file("data", "arabidopsis_toydata.RData",
                    package="methimpute")
data <- get(load(file))
distcor <- distanceCorrelation(data)
print(distcor$plot)
```

---

|                   |                            |
|-------------------|----------------------------|
| estimateTransDist | <i>transDist parameter</i> |
|-------------------|----------------------------|

---

### Description

Obtain an estimate for the `transDist` parameter (used in function [callMethylation](#)) by fitting an exponential function to the supplied correlations (from [distanceCorrelation](#)).

### Usage

```
estimateTransDist(distcor, skip = 2, plot.parameters = TRUE)
```

**Arguments**

- `distcor`            The output produced by [distanceCorrelation](#).
- `skip`                Skip the first n cytosines for the fitting. This can be necessary to avoid periodicity artifacts due to the context definition.
- `plot.parameters`    Whether to plot fitted parameters on to the plot or not.

**Value**

A list() with fitted transDist parameters and the corresponding [ggplot](#).

**Examples**

```
## Get some toy data
file <- system.file("data", "arabidopsis_toydata.RData",
  package="methimpute")
data <- get(load(file))
distcor <- distanceCorrelation(data)
fit <- estimateTransDist(distcor)
print(fit)
```

---

|                 |                           |
|-----------------|---------------------------|
| exportMethylome | <i>Export a methylome</i> |
|-----------------|---------------------------|

---

**Description**

Export a methylome as a TSV file.

**Usage**

```
exportMethylome(model, filename)
```

**Arguments**

- `model`                A [methimputeBinomialHMM](#) object.
- `filename`            The name of the file to be exported.

**Value**

NULL

**Examples**

```
## Not run:
## Get some toy data
file <- system.file("data", "arabidopsis_toydata.RData", package="methimpute")
data <- get(load(file))
print(data)
model <- callMethylation(data, max.iter=10)
exportMethylome(model, filename = tempfile())

## End(Not run)
```

---

```
extractCytosinesFromFASTA
```

*Extract cytosine coordinates*

---

## Description

Extract cytosine coordinates and context information from a FASTA file. Cytosines in ambiguous reference contexts are not reported.

## Usage

```
extractCytosinesFromFASTA(file, contexts = c("CG", "CHG", "CHH"),
  anchor.C = NULL)
```

## Arguments

|                       |                                                                                                                                                                                                                                                                                                            |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>file</code>     | A character with the file name.                                                                                                                                                                                                                                                                            |
| <code>contexts</code> | The contexts that should be extracted. If the contexts are named, the returned object will use those names for the contexts.                                                                                                                                                                               |
| <code>anchor.C</code> | A named vector with positions of the anchoring C in the contexts. This is necessary to distinguish contexts such as C*C*CG ( <code>anchor.C = 2</code> ) and *C*CCG ( <code>anchor.C = 1</code> ). Names must match the contexts. If unspecified, the first C within each context will be taken as anchor. |

## Value

A [GRanges-class](#) object with coordinates of extracted cytosines and meta-data column 'context'.

## Examples

```
## Read a non-compressed FASTA files:
filepath <- system.file("extdata", "arabidopsis_sequence.fa.gz", package="methimpute")

## Only CG context
cytosines <- extractCytosinesFromFASTA(filepath, contexts = 'CG')
table(cytosines$context)

## Split CG context into subcontexts
cytosines <- extractCytosinesFromFASTA(filepath,
  contexts = c('DCG', 'CCG'),
  anchor.C = c(DCG=2, CCG=2))
table(cytosines$context)

## With contexts that differ only by anchor
cytosines <- extractCytosinesFromFASTA(filepath,
  contexts = c('DCG', 'CCG', 'CCG', 'CWG', 'CHH'),
  anchor.C = c(DCG=2, CCG=2, CCG=1, CWG=1, CHH=1))
table(cytosines$context)

## With named contexts
contexts <- c(CG='DCG', CG='CCG', CHG='CCG', CHG='CWG', CHH='CHH')
cytosines <- extractCytosinesFromFASTA(filepath,
```

```

      contexts = contexts,
      anchor.C = c(DCG=2, CCG=2, CCG=1, CWG=1, CHH=1))
table(cytosines$context)

```

---

|                   |                            |
|-------------------|----------------------------|
| getDistinctColors | <i>Get distinct colors</i> |
|-------------------|----------------------------|

---

## Description

Get a set of distinct colors selected from [colors](#).

## Usage

```

getDistinctColors(n, start.color = "blue4", exclude.colors = c("white",
  "black", "gray", "grey", "\\<yellow\\>", "yellow1", "lemonchiffon"),
  exclude.brightness.above = 1, exclude.rgb.above = 210)

```

## Arguments

|                                       |                                                                                                                                                                                    |
|---------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>n</code>                        | Number of colors to select. If <code>n</code> is a character vector, <code>length(n)</code> will be taken as the number of colors and the colors will be named by <code>n</code> . |
| <code>start.color</code>              | Color to start the selection process from.                                                                                                                                         |
| <code>exclude.colors</code>           | Character vector with colors that should not be used.                                                                                                                              |
| <code>exclude.brightness.above</code> | Exclude colors where the 'brightness' value in HSV space is above. This is useful to obtain a matt palette.                                                                        |
| <code>exclude.rgb.above</code>        | Exclude colors where all RGB values are above. This is useful to exclude whitish colors.                                                                                           |

## Details

The function computes the euclidian distance between all [colors](#) and iteratively selects those that have the furthest closes distance to the set of already selected colors.

## Value

A character vector with colors.

## Author(s)

Aaron Taudt

## Examples

```

cols <- getDistinctColors(5)
pie(rep(1,5), labels=cols, col=cols)

```

---

|               |                                |
|---------------|--------------------------------|
| getPosteriors | <i>Get original posteriors</i> |
|---------------|--------------------------------|

---

**Description**

Transform the 'posteriorMeth', 'posteriorMax', and 'status' columns into original posteriors from the HMM.

**Usage**

```
getPosteriors(data)
```

**Arguments**

data                    The \$data entry from a [methimputeBinomialHMM](#) object.

**Value**

A matrix with posteriors.

---

|                |                         |
|----------------|-------------------------|
| getStateColors | <i>Get state colors</i> |
|----------------|-------------------------|

---

**Description**

Get the colors that are used for plotting.

**Usage**

```
getStateColors(states = NULL)
```

**Arguments**

states                    A character vector.

**Value**

A character vector with colors.

**See Also**

[plotting](#)

**Examples**

```
cols <- getStateColors()
pie(1:length(cols), col=cols, labels=names(cols))
```

---

|        |                               |
|--------|-------------------------------|
| import | <i>Methimpute data import</i> |
|--------|-------------------------------|

---

## Description

This page provides an overview of all [methimpute](#) data import functions.

## Usage

```
importBSMAP(file, chrom.lengths = NULL, skip = 1, contexts = c(CG =
  "NCGN", CHG = "NNCHG", CHH = "NNCHH"))

importMethylpy(file, chrom.lengths = NULL, skip = 1, contexts = c(CG
  = "CGN", CHG = "CHG", CHH = "CHH"))

importBSSeeker(file, chrom.lengths = NULL, skip = 0)

importBismark(file, chrom.lengths = NULL, skip = 0)
```

## Arguments

|               |                                                                                                                                                                                                                                                                               |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| file          | The file to import.                                                                                                                                                                                                                                                           |
| chrom.lengths | A data.frame with chromosome names in the first, and chromosome lengths in the second column. Only chromosomes named in here will be returned. Alternatively a tab-separated file with such a data.frame (with headers).                                                      |
| skip          | The number of lines to skip. Usually 1 if the file contains a header and 0 otherwise.                                                                                                                                                                                         |
| contexts      | A character vector of the contexts that are to be assigned. Since some programs report 5-letter contexts, this parameter can be used to obtain a reduced number of contexts. Will yield contexts CG, CHG, CHH by default. Set contexts=NULL to obtain all available contexts. |

## Value

A [methimputeData](#) object.

## Functions

- importBSMAP: Import a BSMAP methylation extractor file.
- importMethylpy: Import a Methylpy methylation extractor file.
- importBSSeeker: Import a BSSeeker methylation extractor file.
- importBismark: Import a Bismark methylation extractor file.

## Examples

```
## Get an example file in BSSeeker format
file <- system.file("extdata", "arabidopsis_bsseeker.txt.gz", package="methimpute")
data(arabidopsis_chromosomes)
bsseeker.data <- importBSSeeker(file, chrom.lengths=arabidopsis_chromosomes)
```

```
## Get an example file in Bismark format
file <- system.file("extdata", "arabidopsis_bismark.txt", package="methimpute")
data(arabidopsis_chromosomes)
arabidopsis_chromosomes$chromosome <- sub('chr', '', arabidopsis_chromosomes$chromosome)
bismark.data <- importBismark(file, chrom.lengths=arabidopsis_chromosomes)

## Get an example file in BSMAP format
file <- system.file("extdata", "arabidopsis_BSMAP.txt", package="methimpute")
data(arabidopsis_chromosomes)
bsmap.data <- importBSMAP(file, chrom.lengths=arabidopsis_chromosomes)

## Get an example file in Methylypy format
file <- system.file("extdata", "arabidopsis_methylypy.txt", package="methimpute")
data(arabidopsis_chromosomes)
arabidopsis_chromosomes$chromosome <- sub('chr', '', arabidopsis_chromosomes$chromosome)
methylypy.data <- importMethylypy(file, chrom.lengths=arabidopsis_chromosomes)
```

---

importRene

---

*Import a Rene methylation extractor file*


---

## Description

Import a Rene methylation extractor file into a [GRanges-class](#) object.

## Usage

```
importRene(file, chrom.lengths = NULL, skip = 1)
```

## Arguments

|               |                                                                                                                                                                                                                          |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| file          | The file to import.                                                                                                                                                                                                      |
| chrom.lengths | A data.frame with chromosome names in the first, and chromosome lengths in the second column. Only chromosomes named in here will be returned. Alternatively a tab-separated file with such a data.frame (with headers). |
| skip          | The number of lines to skip. Usually 1 if the file contains a header and 0 otherwise.                                                                                                                                    |

## Value

A [methimputeData](#) object.

## Examples

```
## Get an example file in Rene format
file <- system.file("extdata", "arabidopsis_rene.txt", package="methimpute")
data(arabidopsis_chromosomes)
rene.data <- methimpute::importRene(file, chrom.lengths=arabidopsis_chromosomes)
```



---

|                  |                                                       |
|------------------|-------------------------------------------------------|
| inflateMethylome | <i>Inflate an imported methylation extractor file</i> |
|------------------|-------------------------------------------------------|

---

### Description

Inflate an imported methylation extractor file to contain all cytosine positions. This is useful to obtain a full methylome, including non-covered cytosines, because most methylation extractor programs only report covered cytosines.

### Usage

```
inflateMethylome(methylome, methylome.full)
```

### Arguments

`methylome` A [GRanges-class](#) with methylation counts.  
`methylome.full` A [GRanges-class](#) with positions for all cytosines or a file with such an object.

### Value

The `methylome.full` object with added metadata column 'counts'.

### Examples

```
## Get an example file in BSseeker format
file <- system.file("extdata", "arabidopsis_bsseeker.txt.gz", package="methimpute")
bsseeker.data <- importBSseeker(file)
bsseeker.data

## Inflate to full methylome (including non-covered sites)
data(arabidopsis_toydata)
full.methylome <- inflateMethylome(bsseeker.data, arabidopsis_toydata)
full.methylome
```

---

|               |                                                 |
|---------------|-------------------------------------------------|
| loadFromFiles | <i>Load <b>methimpute</b> objects from file</i> |
|---------------|-------------------------------------------------|

---

### Description

Wrapper to load [methimpute](#) objects from file and check the class of the loaded objects.

### Usage

```
loadFromFiles(files, check.class = c("GRanges", "methimputeBinomialHMM"))
```

### Arguments

`files` A list of [GRanges-class](#) or [methimputeBinomialHMM](#) objects or a character vector with files that contain such objects.  
`check.class` Any combination of `c('GRanges', 'methimputeBinomialHMM')`. If any of the loaded objects does not belong to the specified class, an error is thrown.

**Value**

A list of [GRanges-class](#) or [methimputeBinomialHMM](#) objects.

**Examples**

```
## Get some files that you want to load
file <- system.file("data", "arabidopsis_toydata.RData",
                    package="methimpute")

## Load and print
data <- loadFromFiles(file)
print(data)
```

---

|                    |                           |
|--------------------|---------------------------|
| methimpute-objects | <i>methimpute objects</i> |
|--------------------|---------------------------|

---

**Description**

[methimpute](#) defines several objects.

- [methimputeData](#): Returned by [importBSSeeker](#), [importBismark](#) and [inflateMethylome](#).
- [methimputeBinomialHMM](#): Returned by [callMethylation](#).

---

|                       |                              |
|-----------------------|------------------------------|
| methimputeBinomialHMM | <i>methimputeBinomialHMM</i> |
|-----------------------|------------------------------|

---

**Description**

The [methimputeBinomialHMM](#) is a `list()` which contains various entries (see Value section). The main entry of this object is `$data`, which contains the methylation status calls and posterior values. See Details for a description of all columns.

**Details**

The `$data` entry in this object contains the following columns:

- `context` The sequence context of the cytosine.
- `counts` Counts for methylated and total number of reads at each position.
- `distance` The distance in base-pairs from the previous to the current cytosine.
- `transitionContext` Transition context in the form "previous-current".
- `posteriorMax` Maximum posterior value of the methylation status call, can be interpreted as the confidence in the call.
- `posteriorMeth` Posterior value of the "methylated" component.
- `posteriorUnmeth` Posterior value of the "unmethylated" component.
- `status` Methylation status.
- `rc.meth.lvl` Recalibrated methylation level, calculated as  $r\$data\$rc.meth.lvl = r\$data\$params\$emissionParams * r\$data\$posteriorUnmeth + r\$params\$emissionParams\$Methylated[data\$context,] * r\$data\$posteriorMeth$ , where `r` is the [methimputeBinomialHMM](#) object.

**Value**

A list() with the following entries:

|                 |                                                                                                              |
|-----------------|--------------------------------------------------------------------------------------------------------------|
| convergenceInfo | A list() with information about the convergence of the model fitting procedure.                              |
| params          | A list() with fitted and non-fitted model parameters.                                                        |
| params.initial  | A list() with initial values for the model parameters.                                                       |
| data            | A <a href="#">GRanges-class</a> with cytosine positions and methylation status calls.                        |
| segments        | The data entry where coordinates of consecutive cytosines with the same methylation status have been merged. |

**See Also**

[methimpute-objects](#)

---

|                |                       |
|----------------|-----------------------|
| methimputeData | <i>methimputeData</i> |
|----------------|-----------------------|

---

**Description**

A [GRanges-class](#) object containing cytosine coordinates with meta-data columns 'context' and 'counts'.

**See Also**

[methimpute-objects](#)

---

|               |                                 |
|---------------|---------------------------------|
| parameterScan | <i>Perform a parameter scan</i> |
|---------------|---------------------------------|

---

**Description**

Perform a parameter scan for an arbitrary parameter.

**Usage**

```
parameterScan(f, param, values, ...)
```

**Arguments**

|        |                                                               |
|--------|---------------------------------------------------------------|
| f      | A function for which to perform the scan.                     |
| param  | A character with the parameter for which to perform the scan. |
| values | A vector with parameter values for which to perform the scan. |
| ...    | Other parameters passed through to f.                         |

**Value**

A data.frame with loglikelihood values.

## Description

This page provides an overview of all [methimpute](#) plotting functions.

## Usage

```
plotHistogram(model, total.counts, binwidth = 1)

plotScatter(model, datapoints = 1000)

plotTransitionProbs(model)

plotConvergence(model)

plotEnrichment(model, annotation, windowsize = 100, insidewindows = 20,
  range = 1000, category.column = NULL, plot = TRUE,
  df.list = NULL)

plotPosteriorDistance(model, datapoints = 1e+06, binwidth = 5,
  max.coverage.y = 0, min.coverage.x = 3, xmax = 200,
  xbreaks.interval = xmax/10, cutoffs = NULL)
```

## Arguments

|                  |                                                                                                                                                  |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| model            | A <a href="#">methimputeBinomialHMM</a> object.                                                                                                  |
| total.counts     | The number of total counts for which the histogram is to be plotted.                                                                             |
| binwidth         | The bin width for the histogram/boxplot.                                                                                                         |
| datapoints       | The number of randomly selected datapoints for the plot.                                                                                         |
| annotation       | A <a href="#">GRanges-class</a> object with coordinates for the annotation.                                                                      |
| windowsize       | Resolution in base-pairs for the curve upstream and downstream of the annotation.                                                                |
| insidewindows    | Number of data points for the curve inside the annotation.                                                                                       |
| range            | Distance upstream and downstream for which the enrichment profile is calculated.                                                                 |
| category.column  | The name of a column in data that will be used for facetting of the plot.                                                                        |
| plot             | Logical indicating whether a plot or the underlying data.frame is to be returned.                                                                |
| df.list          | A list() of data.frames, output from <code>plotEnrichment(..., plot=FALSE)</code> . If specified, option data will be ignored.                   |
| max.coverage.y   | Maximum coverage for positions on the y-axis.                                                                                                    |
| min.coverage.x   | Minimum coverage for positions on the x-axis.                                                                                                    |
| xmax             | Upper limit for the x-axis.                                                                                                                      |
| xbreaks.interval | Interval for breaks on the x-axis.                                                                                                               |
| cutoffs          | A vector with values that are plotted as horizontal lines. The names of the vector must match the context levels in <code>data\$context</code> . |

**Value**

A `ggplot` object.

**Functions**

- `plotHistogram`: Plot a histogram of count values and fitted distributions.
- `plotScatter`: Plot a scatter plot of read counts colored by methylation status.
- `plotTransitionProbs`: Plot a heatmap of transition probabilities.
- `plotConvergence`: Plot the convergence of the probability parameters.
- `plotEnrichment`: Plot an enrichment profile around an annotation.
- `plotPosteriorDistance`: Maximum posterior vs. distance to nearest covered cytosine.

**Examples**

```
## Get some toy data
file <- system.file("data", "arabidopsis_toydata.RData",
                    package="methimpute")
data <- get(load(file))
print(data)
model <- callMethylation(data)
## Make nice plots
plotHistogram(model, total.counts=5)
plotScatter(model)
plotTransitionProbs(model)
plotConvergence(model)
plotPosteriorDistance(model$data)

## Get annotation data and make an enrichment profile
# Note that this looks a bit ugly because our toy data
# has only 200000 datapoints.
data(arabidopsis_genes)
plotEnrichment(model, annotation=arabidopsis_genes)
```

---

```
print.methimputeBinomialHMM
```

*Print model object*

---

**Description**

Print model object

**Usage**

```
## S3 method for class 'methimputeBinomialHMM'
print(x, ...)
```

**Arguments**

|                  |                                              |
|------------------|----------------------------------------------|
| <code>x</code>   | A <code>methimputeBinomialHMM</code> object. |
| <code>...</code> | Ignored.                                     |

**Value**

An invisible NULL.

---

|            |                                      |
|------------|--------------------------------------|
| transCoord | <i>Transform genomic coordinates</i> |
|------------|--------------------------------------|

---

**Description**

Add two columns with transformed genomic coordinates to the [GRanges-class](#) object. This is useful for making genomewide plots.

**Usage**

```
transCoord(gr)
```

**Arguments**

`gr`                    A [GRanges-class](#) object.

**Value**

The input [GRanges-class](#) with two additional metadata columns 'start.genome' and 'end.genome'.

# Index

arabidopsis\_chromosomes, [3](#)  
arabidopsis\_genes, [3](#)  
arabidopsis\_TEs, [3](#)  
arabidopsis\_toydata, [4](#)  
  
binCounts (binning), [4](#)  
binMethylome (binning), [4](#)  
binning, [4](#)  
binomialTestMethylation, [5](#)  
binPositions (binning), [4](#)  
  
callMethylation, [6](#), [10](#), [18](#)  
callMethylationSeparate, [7](#)  
collapseBins, [8](#)  
colors, [13](#)  
  
distanceCorrelation, [10](#), [10](#), [11](#)  
  
estimateTransDist, [6](#), [7](#), [10](#)  
exportMethylome, [11](#)  
extractCytosinesFromFASTA, [12](#)  
  
getDistinctColors, [13](#)  
getPosteriors, [14](#)  
getStateColors, [14](#)  
ggplot, [10](#), [11](#), [21](#)  
  
import, [15](#)  
importBismark, [18](#)  
importBismark (import), [15](#)  
importBSMAP (import), [15](#)  
importBSSeeker, [18](#)  
importBSSeeker (import), [15](#)  
importMethylpy (import), [15](#)  
importRene, [16](#)  
inflateMethylome, [17](#), [18](#)  
  
loadFromFiles, [17](#)  
  
methimpute, [3](#), [4](#), [15](#), [17](#), [18](#), [20](#)  
methimpute (methimpute-package), [2](#)  
methimpute-objects, [18](#)  
methimpute-package, [2](#)  
methimputeBinomialHMM, [6–8](#), [11](#), [14](#), [17](#), [18](#),  
[18](#), [20](#), [21](#)  
  
methimputeData, [4–7](#), [10](#), [15](#), [16](#), [18](#), [19](#)  
  
parameterScan, [19](#)  
plotConvergence (plotting), [20](#)  
plotEnrichment (plotting), [20](#)  
plotHistogram (plotting), [20](#)  
plotPosteriorDistance (plotting), [20](#)  
plotScatter (plotting), [20](#)  
plotting, [2](#), [14](#), [20](#)  
plotTransitionProbs (plotting), [20](#)  
print.methimputeBinomialHMM, [21](#)  
  
transCoord, [22](#)