Package 'recountmethylation'

May 25, 2024

Version 1.14.0

Title Access and analyze public DNA methylation array data compilations

Description
Resources for cross-study analyses of public DNAm array data from NCBI GEO repo, produced using Illumina's Infinium HumanMethylation450K (HM450K) and MethylationEPIC (EPIC) platforms.
Provided functions enable download, summary, and filtering of large compilation files. Vignettes detail background about file formats, example analyses, and more. Note the disclaimer on package load and consult the main manuscripts for further info.

License Artistic-2.0

Encoding UTF-8

URL https://github.com/metamaden/recountmethylation

BugReports https://github.com/metamaden/recountmethylation/issues

LazyData FALSE

Depends R (>= 4.1)

Imports minfi, HDF5Array, rhdf5, S4Vectors, utils, methods, RCurl, R.utils, BiocFileCache, basilisk, reticulate, DelayedMatrixStats

Suggests minfiData, minfiDataEPIC, knitr, testthat, ggplot2, gridExtra, rmarkdown, BiocStyle, GenomicRanges, limma, ExperimentHub, AnnotationHub

VignetteBuilder knitr

biocViews DNAMethylation, Epigenetics, Microarray, MethylationArray, ExperimentHub

RoxygenNote 7.2.0

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

git_branch RELEASE_3_19

git_last_commit 5bf2240

git_last_commit_date 2024-04-30

Repository Bioconductor 3.19

Date/Publication 2024-05-24

Author Sean K Maden [cre, aut] (<https://orcid.org/0000-0002-2212-4894>),
Brian Walsh [aut] (<https://orcid.org/0000-0003-0913-1528>),
Kyle Ellrott [aut] (<https://orcid.org/0000-0002-6573-5900>),
Kasper D Hansen [aut] (<https://orcid.org/0000-0003-0086-0687>),
Reid F Thompson [aut] (<https://orcid.org/0000-0003-3661-5296>),
Abhinav Nellore [aut] (<https://orcid.org/0000-0001-8145-1484>)

Maintainer Sean K Maden <maden@ohsu.edu>

Contents

`ba.background` *ba.background*

Description

Get background signal for BeadArray metric calculations.

Get the Biotin staining Red BeadArray metric.

Get the Biotin staining Green BeadArray metric.

Get the Non-polymorphic Red BeadArray metric.

Get the Non-polymorphic Green BeadArray metric.

Get the Bisulfite Conversion I Red BeadArray metric.

Get the Restoration BeadArray metric.

Get the Specificity I Red BeadArray metric.

Get the Specificity I Green BeadArray metric.

Get the Specificity II Green BeadArray metric.

Get the Extension Red BeadArray metric.

Get the Extension Green BeadArray metric.

Get the Hybridication (high vs. medium) BeadArray metric.

Get the Hybridication (medium vs. low) BeadArray metric.

Get the Target Removal 1 BeadArray metric.

Get the Target Removal 2 BeadArray metric.

Get the Bisulfite Conversion I Green BeadArray metric.

Get the Bisulfite Conversion II BeadArray metric.

Usage

```
ba.background(cdf, ct = "Extension")

ba.biotinstaining.red(
  rs,
  biotin.baseline,
  rm,
  cdf,
  mnum = 1,
  mtot = 17,
  cnamei = "biotin.stain.red",
  ct = "Biotin|DNP"
)

ba.biotinstaining.grn(
  gs,
  biotin.baseline,
  rm,
  cdf,
  mnum = 2,
  mtot = 17,
  cnamei = "biotin.stain.grn",
  ct = "Biotin|DNP"
)

ba.nonpolymorphic.red(
  rs,
  rm,
  cdf,
```

```
  mnum = 3,
  mtot = 17,
  cnamei = "nonpolymorphic.red",
  ct = "NP"
)

ba.nonpolymorphic.grn(
  gs,
  rm,
  cdf,
  mnum = 4,
  mtot = 17,
  cnamei = "nonpolymorphic.grn",
  ct = "NP"
)

ba.bisulfiteconv1.red(
  rs,
  rm,
  cdf,
  mnum = 5,
  mtot = 17,
  cnamei = "bisulfite.conv1.red",
  ct = "Conversion I-"
)

ba.restoration(
  gs,
  rm,
  cdf,
  addr.bkg,
  baseline = 3000,
  mnum = 6,
  mtot = 17,
  cnamei = "restoration.grn",
  ct = "RESTORATION"
)

ba.specificity1.red(
  rs,
  rm,
  cdf,
  mnum = 7,
  mtot = 17,
  cnamei = "specificityI.red"
)

ba.specificity1.grn(
```

```
  gs,
  rm,
  cdf,
  mnum = 8,
  mtot = 17,
  cnamei = "specificityI.grn"
)

ba.specificity2(
  rs,
  gs,
  rm,
  cdf,
  mnum = 9,
  mtot = 17,
  cnamei = "specificityII",
  ct = "Specificity 2"
)

ba.extension.red(
  rs,
  rm,
  cdf,
  mnum = 10,
  mtot = 17,
  cnamei = "extension.red",
  ct = "Extension.*"
)

ba.extension.grn(
  gs,
  rm,
  cdf,
  mnum = 11,
  mtot = 17,
  cnamei = "extension.grn",
  ct = "Extension.*"
)

ba.hybridization.hi.vs.med(
  gs,
  rm,
  cdf,
  mnum = 12,
  mtot = 17,
  cnamei = "hyb.hi.med",
  ct = "High|Medium"
)
```

```
ba.hybridization.med.vs.low(
  gs,
  rm,
  cdf,
  mnum = 13,
  mtot = 17,
  cnamei = "hyb.med.low",
  ct = "Low|Medium"
)

ba.targetremoval1(
  gs,
  rm,
  cdf,
  baseline = 3000,
  mnum = 14,
  mtot = 17,
  cnamei = "target.removal1",
  ct = "Extension|Target Removal 1"
)

ba.targetremoval2(
  gs,
  rm,
  cdf,
  baseline = 3000,
  mnum = 15,
  mtot = 17,
  cnamei = "target.removal2",
  ct = "Extension|Target Removal 2"
)

ba.bisulfiteconv1.grn(
  gs,
  rm,
  cdf,
  mnum = 16,
  mtot = 17,
  cnamei = "bisulfite.conv1.grn",
  ct = "Conversion I-"
)

ba.bisulfiteconv2(
  rs,
  gs,
  rm,
  cdf,
```

```
  mnum = 17,
  mtot = 17,
  cnamei = "bisulfite.conv2",
  ct = "Conversion II-"
)
```

Arguments

cdf	Control probe data frame (required).
ct	Column name/metric title (character, required).
rs	Red signal matrix (required).
biotin.baseline	Baseline signal for the biotin stain assay (required).
rm	Results matrix containing control metrics (required).
mnum	Metric number out of total (numeric).
mtot	Total metrics to be calculated (numeric, 17).
cnamei	Column name (character, required).
gs	Green signal matrix (required).
addr.bkg	Background signal probe address (required).
baseline	Baseline measure for signals (integer, 3000).

---

bactrl *bactrl*

Description

Get the BeadArray control metrics from HM450K platform red/grn signals.

Usage

```
bactrl(
  cdf = NULL,
  baset = "reduced",
  rg = NULL,
  rs = NULL,
  gs = NULL,
  baseline = 3000,
  biotin.baseline = 1
)
```

Arguments

cdf	Control probe annotations (optional, NULL, data.frame, cols = properties, rows = probes).
baset	Either the most informative BeadArray metrics (a.k.a. "reduced" set, 5 metrics, recommended), or the full set of 17 metrics ("all").
rg	An RGChannelSet object (optional, NULL).
rs	Red signal data (optional, NULL, data.frame, columns are probes, rows are samples, column names are addresses, rownames are samples/GSM IDs).
gs	Green signal data (optional, NULL, data.frame, columns are probes, rows are samples, column names are addresses, rownames are samples/GSM IDs).
baseline	Baseline measure for signals (integer, 3000).
biotin.baseline	Baseline to use for biotin controls (integer, 1).

Details

This function calculates the BeadArray quality metrics based on Illumina's documentation and previous work (see references). Based on previous work, this function can calculate the full set of 17 metrics (e.g. baset = 'all'), or the reduced set of 5 metrics (e.g. baset = 'reduced'), where the latter is recommended for most purposes. For additional details, consult the BeadArray metrics vignette in this package.

Value

Matrix of BeadArray signal values

References

1. S. K. Maden, et. al., "Human methylome variation across Infinium 450K data on the Gene Expression Omnibus," NAR Genomics and Bioinformatics, Volume 3, Issue 2, June 2021

2. Illumina, "Illumina Genome Studio Methylation Module v1.8," Nov. 2010.

3. Illumina, "BeadArray Controls Reporter Software Guide," Oct. 2015.

4. J. A. Heiss and A. C. Just, "Identifying mislabeled and contaminated DNA methylation microarray data: an extended quality control toolset with examples from GEO," Clinical Epigenetics, vol. 10, June 2018. '

See Also

bathresh

Examples

```
dir <- system.file("extdata", "bactrl", package = "recountmethylation")
rgf <- get(load(file.path(dir, "rgf-cgctrl-test_hm450k-minfidata.rda")))
mba <- bactrl(rg = rgf)
```

bathresh *Get BeadArray control outcomes from a matrix of metric signals*

Description

Get Illumina's prescribed minimum quality thresholds for BeadArray metrics.

Usage

```
bathresh()
```

Value

Data frame of minimum BeadArray quality thresholds.

References

1. Illumina, "Illumina Genome Studio Methylation Module v1.8," Nov. 2010.
2. Illumina, "BeadArray Controls Reporter Software Guide," Oct. 2015.

See Also

bactrl

Examples

```
dfthresh <- bathresh()
```

data_mdpost *Retrieve all available sample metadata from an HDF5 database.*

Description

Retrieve all available sample metadata in a dataset from an HDF5 database. Returns data in metadata dataset "dsn" contained in an h5 file located at path "dbn."

Usage

```
data_mdpost(dbn = "remethdb2.h5", dsn = "mdpost")
```

Arguments

dbn	Path to h5 HDF5 database file.
dsn	Name or group path to HDF5 dataset containing the sample metadata and learned annotations.

Value

data.frame of available sample metadata.

See Also

hread()

Examples

```
path <- system.file("extdata", "h5test", package = "recountmethylation")
fn <- list.files(path)
dbpath <- file.path(path, fn)
mdp <- data_mdpost(dbn = dbpath, dsn = "mdpost")
dim(mdp) # [1]  2 19
```

---

gds_idat2rg *Get IDATs as an RGChannelSet from GEO/GDS*

Description

Queries and downloads GSM IDAT files in GEO Data Sets db, then returns the assay data as an "RGChannelSet", calling gds_idatquery() then minfi::read.metharray().

Usage

```
gds_idat2rg(
  gsmvi,
  rmdl = TRUE,
  ext = "gz",
  dfp = "./idats/",
  burl = paste0("ftp://ftp.ncbi.nlm.nih.gov/", "geo/samples/"),
  silent = TRUE
)
```

Arguments

gsmvi	A vector of GSM IDs (alphanumeric character strings).
rmdl	Whether to remove downloaded IDAT files when finished (default TRUE).
ext	Extension for downloaded files (default "gz").
dfp	Destination for IDAT downloads.
burl	Base URL string for the IDAT query (default "ftp://ftp.ncbi.nlm.nih.gov/geo/samples/").
silent	Whether to suppress warnings on download removal (default TRUE).

Value

An RGChannelSet object

See Also

gds_idatquery(), read.metharray()

Examples

```
gsmvi <- c("GSM2465267", "GSM2814572")
fpath <- file.path(tempdir(), "gds_idat2rg_example")
rg <- try(gds_idat2rg(gsmvi, dfp = fpath))
```

---

gds_idatquery *Query and download IDATs from GEO Data Sets*

Description

Queries GEO Data Sets for IDATs, and downloads available IDATs. This uses anticipated string pattern to construct the URL path for the query. IDATs are detected from the supplement for a GSE record.

Usage

```
gds_idatquery(
  gsmvi,
  ext = "gz",
  expand = TRUE,
  verbose = FALSE,
  dfp = "idats",
  burl = paste0("ftp://ftp.ncbi.nlm.nih.gov/geo/samples/")
)
```

Arguments

gsmvi	Vector of valid GSM IDs.
ext	Filename extension.
expand	Whether to expand compressed files.
verbose	Whether to show verbose messages (default FALSE).
dfp	Destination directory for downloads.
burl	Base URL string for RCurl query.

Value

Lists the basename paths and filenames of IDATs downloaded.

Examples

```
query <- try(gds_idatquery(gsmvi = c("GSM2465267", "GSM2814572")))
```

getdb *Access database files.*

Description

Combines download and load functions for databases. If the "namematch" argument isn't provided, the latest available file is downloaded. All files include metadata for the available samples.

There are 6 functions. Functions with "h5se" access HDF5-SummarizedExperiment files, and "h5" functions access HDF5 databases. The 4 h5se functions are "rg" (RGChannelSet), "gm" (MethylSet), "gr" (GenomicRatioSet), and "test" (data for 2 samples from "gr"). The 2 h5 functions are "rg" (red and green signal datasets), and "test" (data for 2 samples from "rg"). See vignette for details about file types and classes.

Usage

```
getdb_h5se_test(
  platform = NULL,
  dfp = NULL,
  namematch = "remethdb-h5se_gr-test.*",
  verbose = FALSE
)

getdb_h5_test(
  platform = NULL,
  namematch = "remethdb-h5_rg-test_.*",
  dfp = NULL,
  verbose = FALSE
)

getdb_h5se_gr(
  platform = c("hm450k", "epic"),
  dfp = NULL,
  namematch = "remethdb_h5se-gr_.*",
  verbose = FALSE
)

getdb_h5se_gm(
  platform = c("hm450k", "epic"),
  dfp = NULL,
  namematch = "remethdb_h5se-gm_.*",
  verbose = FALSE
)

getdb_h5se_rg(
  platform = c("hm450k", "epic"),
  dfp = NULL,
```

```
  namematch = "remethdb-h5se_rg_.*",
  verbose = FALSE
)

getdb_h5_rg(
  platform = c("hm450k", "epic"),
  dfp = NULL,
  namematch = "remethdb-h5_rg_.*",
  verbose = FALSE
)
```

Arguments

platform	Valid supported DNAm array platform type. Currently either "epic" for EPIC/HM850K, or "hm450k" for HM450K.
dfp	Folder to search for database file (optional, if NULL then searches cache dir specified by BiocFileCache).
namematch	Filename pattern to match when searching for database (see defaults).
verbose	Whether to return verbose messages (default FALSE).

Value

Either a SummarizedExperiment object for h5se functions, or a file path for h5 functions.

See Also

get_rmdl()

Examples

```
h5 <- getdb_h5_test(dfp = tempdir())
```

---

getrg *Query and store data from h5 file signal tables*

Description

Queries signal datasets in an h5 HDF5 database file. Handles identity queries to rows (GSM IDs) or columns (bead addresses). Returns query matches either as a list of datasets or a single RGChannelSet, with option to include sample metadata.

Usage

```
getrg(
  dbn,
  gsmv = NULL,
  cgv = NULL,
  data.type = c("se"),
  dsv = c("redsignal", "greensignal"),
  all.gsm = FALSE,
  all.cg = TRUE,
  metadata = TRUE,
  md.dsn = "mdpost",
  verbose = FALSE
)
```

Arguments

dbn	Name of the HDF5 database file.
gsmv	Vector valid GSM IDs (rows) to query, either NULL or vector of length > 2 valid GSM IDs, or "all.gsm" should be TRUE.
cgv	Vector of valid bead addresses (columns) to query in the signal datasets (default NULL).
data.type	Format for returned query matches, either as datasets "df" or RGChannelSet "se" object.
dsv	Vector of raw signal datasets or group paths to query, including both the red channel 'redsignal' and green channel 'greensignal' datasets.
all.gsm	Whether to query all available GSM IDs.
all.cg	Whether to query all available CpG probe addresses.
metadata	Whether to access available postprocessed metadata for queried samples.
md.dsn	Name of metadata dataset in h5 file.
verbose	Whether to post status messages.

Value

Returns either an RGChannelSet or list of data.frame objects from dataset query matches.

See Also

rgse()

Examples

```
path <- system.file("extdata", "h5test", package = "recountmethylation")
fn <- list.files(path)
dbpath <- file.path(path, fn)
rg <- getrg(dbn = dbpath, all.gsm = TRUE, metadata = FALSE)
dim(rg) # [1] 11162     2
```

```
class(rg)
# [1] "RGChannelSet"
# attr(,"package")
# [1] "minfi"
```

`get_crossreactive_cpgs` *get_crossreactive_cpgs*

Description

Get cross-reactive CpG probe IDs for Illumina BeadArray platforms.

Usage

```
get_crossreactive_cpgs(probeset = "all")
```

Arguments

`probeset`	Specify the set of probes to filter ("all", "hm450k", "epic", "chen", "pidlsey", "illumina").

Details

Prior work showed significant cross-reactivity at subsets of CpG probes on Illumina's BeadArray platforms, including HM450K and EPIC. This was primarily due to the probe sequence, as the targeted 50-bp sequence can be either too short or too degenerate to bind a particular DNA region with high specificity. This can cause cross-reaction with off-target DNA locations, including at entirely different chromosomes than the target sequence. Consult the individual publication sources for details about the identification and consequences of cross-reactive CpG probes.

You can retrieve a cross-reactive probe set in a variety of ways. For instance, declare the publication source with either "chen" (for Chen et al 2013), "pidsley" (for Pidsley et al 2016), or "illumina" (for official Illumina documentation), or declare the platform category as either "all" (both HM450K and EPIC), "hm450k", or "epic."

Value

Vector of cross-reactive CpG probe IDs.

References

1. Yi-an Chen, Mathieu Lemire, Sanaa Choufani, Darci T. Butcher, Daria Grafodatskaya, Brent W. Zanke, Steven Gallinger, Thomas J. Hudson & Rosanna Weksberg (2013) Discovery of cross-reactive probes and polymorphic CpGs in the Illumina Infinium HumanMethylation450 microarray, Epigenetics, 8:2, 203-209, DOI: 10.4161/epi.23470

2. Pidsley, R., Zotenko, E., Peters, T.J. et al. Critical evaluation of the Illumina MethylationEPIC BeadChip microarray for whole-genome DNA methylation profiling. Genome Biol 17, 208 (2016). https://doi.org/10.1186/s13059-016-1066-1

See Also

bactrl, get_qcsignal

Examples

```
length(get_crossreactive_cpgs("all"))      # 46324
length(get_crossreactive_cpgs("hm450k"))   # 30540
length(get_crossreactive_cpgs("epic"))     # 43410
length(get_crossreactive_cpgs("chen"))     # 29233
length(get_crossreactive_cpgs("pidsley"))  # 43254
length(get_crossreactive_cpgs("illumina")) # 1031
```

`get_fh` *get_fh*

Description

Get the hashed features for a data table. Uses reticulate package to call the Python script to do feature hashing on a table of data. It is assumed the input table has sample data in rows, with probe data in columns. The input data table should have row names but not column names.

Usage

```
get_fh(csv_savepath, csv_openpath, ndim = 1000, lstart = 1)
```

Arguments

`csv_savepath`	Name/path of hashed features table to write (required, string, writes new csv where rows = samples, cols = hashed features).
`csv_openpath`	Name/path of table to hash (required, string, assumes a csv where rows = samples, cols = probes).
`ndim`	Number of hashed features (integer, 1000).
`lstart`	Line index to start on (0-based for Python, required, int, 0).

Value

Path to new hashed featuers table.

Examples

```
# get example bval csv
# of_fpath <- system.file("extdata", "fhtest",
#        package = "recountmethylation")
# of_fpath <- file.path(of_fpath, "tbval_test.csv")
# write new hashed features results
# get_fh(csv_savepath = "bval_fn.csv", csv_openpath = of_fpath, ndim = 100)
```

get_qcsignal *get_qcsignal*

Description

Get the medians of the log2-transformed M and U signals. This function uses the DelayedMatrixStats implementations of colMedians for rapid calculations on DelayedArray-formatted matrices.

Usage

```
get_qcsignal(se = NULL, mm = NULL, mu = NULL, sample_idv = NULL)
```

Arguments

se	Valid SummarizedExperiment object, such as a MethylSet or similar object for which getMeth() and getUnmeth() methods are defined (optional).
mm	Matrix of methylated/M signals (optional, not required if se provided).
mu	Matrix of unmethylated/U signals (optional, not required if se provided).
sample_idv	Vector of sample IDs to label rows in the returned data frame (optional, uses mm colnames instead if not provided).

Details

Calculates the log2 of median signal for methylated/M and unmethylated/U signals separately.

Value

Data frame of signal summaries.

See Also

bactrl

Examples

```
library(minfiData)
data(MsetEx)
se <- MsetEx
class(se)
# [1] "MethylSet"
# attr(,"package")
# [1] "minfi"
ms <- get_qcsignal(se)
```

get_rmdl *Get DNAm assay data.*

Description

Uses RCurl to download the latest HDF5-SummarizedExperiment or HDF5 database compilation files objects from the server. Calls servermatrix and performs various quality checks to validate files and downloads. This function is wrapped in the getdb() set of functions (type '?getdb' for details).

Usage

```
get_rmdl(
  which.class = c("rg", "gm", "gr", "test"),
  which.type = c("h5se", "h5"),
  which.platform = c("hm450k", "epic"),
  fn = NULL,
  dfp = "downloads",
  url = "https://methylation.recount.bio/",
  show.files = FALSE,
  download = TRUE,
  sslver = FALSE,
  verbose = TRUE
)
```

Arguments

which.class	Either "rg", "gm", "gr", or "test" for RGChannelSet, MethylSet, GenomicRatioSet, or 2-sample subset.
which.type	Either "h5se" for an HDF5-SummarizedExperiment or "h5" for an HDF5 database.
which.platform	Supported DNAm array platform type. Currently supports either "epic" for EPIC/HM850K, or "hm450k" for HM450K.
fn	Name of file on server to download (optional, default NULL).
dfp	Download destination directory (default "downloads").
url	The server URL to locate files for download.
show.files	Whether to print server file data to console (default FALSE).
download	Whether to download (TRUE) or return queried filename (FALSE).
sslver	Whether to use server certificate check (default FALSE).
verbose	Whether to return verbose messages (default TRUE).

Value

New filepath to dir containing the downloaded data.

See Also

servermatrix(), getURL(), loadHDF5SummarizedExperiment(), h5ls()

Examples

```
# prints file info from server:
path <- try(get_rmdl(which.class = "test", which.type = "h5se",
show.files = TRUE, download = FALSE))
```

`get_servermatrix` *get_servermatrix*

Description

Get a matrix of server files. If the RCurl call fails, a matrix is loaded from the stored package files at 'sm_path'.

Usage

```
get_servermatrix(
  dn = NULL,
  sslver = FALSE,
  printmatrix = TRUE,
  url = "https://methylation.recount.bio/",
  verbose = FALSE,
 sm_path = system.file("extdata", "servermatrix_rda", package = "recountmethylation")
)
```

Arguments

`dn`	Server data returned from RCurl (default NULL).
`sslver`	Whether to use SSL certificate authentication for server connection (default FALSE).
`printmatrix`	Whether to print the data matrix to console (default TRUE).
`url`	Server website url (default "https://methylation.recount.bio/").
`verbose`	Whether to show verbose messages (default FALSE).
`sm_path`	Path to the servermatrix_rda dir containing the stored servermatrix files (default: system.file...).

Value

Matrix of server files and file metadata

See Also

servermatrix, get_rmdl, smfilt

Examples

```
sm <- get_servermatrix(url = "")
```

---

hread *Query and store an HDF5 dataset on row and column indices.*

Description

Connect to an HDF5 database h5 file with rhdf5::h5read(). Returns the subsetted data.

Usage

```
hread(ri, ci, dsn = "redsignal", dbn = "remethdb2.h5")
```

Arguments

ri	Row indices in dataset.
ci	Column indices in dataset.
dsn	Name of dataset or group of dataset to connect with.
dbn	Path to h5 database file.

Value

HDF5 database connection object.

Examples

```
# Get tests data pointer
path <- system.file("extdata", "h5test", package = "recountmethylation")
fn <- list.files(path)
dbpath <- file.path(path, fn)
# red signal, first 2 assay addr, 3 samples
reds <- hread(1:2, 1:3, d = "redsignal", dbn = dbpath)
dim(reds) # [1] 2 3
```

make_si *make_si*

Description

Make search index from table of hashed features. Additional details about the hnswlib search index parameters (e..g ‘space_val‘, ‘efc_val‘, ‘m_val‘, and ‘ef_val‘) can be found in the Python package docstrings and ReadMe.

Usage

```
make_si(
  fh_csv_fpath,
  si_fname = "new_search_index.pickle",
  si_dict_fname = "new_index_dict.pickle",
  threads = 4,
  space_val = "l2",
  efc_val = 2000,
  m_val = 1000,
  ef_val = 2000
)
```

Arguments

fh_csv_fpath	Name/path of csv (e.g. a table of hashed features) containing data for the index (required, string, "bvaltest.csv", where rows = samples, cols = features).
si_fname	Name of new search index file to save (required, string, "new_search_index.pickle")
si_dict_fname	Name of new index dictionary, with string labels, to save (required, string, "new_index_dict.pickle").
threads	Number of threads for processing new index (required, int, 4).
space_val	Space value for new search index (required, valid string, l2’).
efc_val	EFC value for the index (required, int, 2000).
m_val	M value for the index (required, int, 1000).
ef_val	EF value for the index (required, int, 2000).

Value

Boolean, TRUE if new search index and dictionary created, FALSE if creating the new search index and dictionary files failed, otherwise NULL.

Examples

```
# fh_csv_fpath <- system.file("extdata", "fhtest",
# package = "recountmethylation")
# fh_csv_fpath <- file.path(fh_csv_fpath, "bval_fn.csv")
# make_si(fh_csv_fpath)
```

`matchds_1to2` *Match two datasets on rows and columns*

Description

Match 2 datasets using the character vectors of row or column names. This is used to assemble an "RGChannelSet" from a query to an h5 dataset.

Usage

```
matchds_1to2(
  ds1,
  ds2,
  mi1 = c("rows", "columns"),
  mi2 = c("rows", "columns"),
  subset.match = FALSE
)
```

Arguments

`ds1`	First dataset to match
`ds2`	Second dataset to match
`mi1`	Match index of ds1 (either "rows" or "columns")
`mi2`	Match index of ds2 (either "rows" or "columns")
`subset.match`	If index lengths don't match, match on the common subset instead

Value

A list of the matched datasets.

Examples

```
# get 2 data matrices
ds1 <- matrix(seq(1, 10, 1), nrow = 5)
ds2 <- matrix(seq(11, 20, 1), nrow = 5)
rownames(ds1) <- rownames(ds2) <- paste0("row", seq(1, 5, 1))
colnames(ds1) <- colnames(ds2) <- paste0("col", c(1, 2))
ds2 <- ds2[rev(seq(1, 5, 1)), c(2, 1)]
# match row and column names
lmatched <- matchds_1to2(ds1, ds2, mi1 = "rows", mi2 = "rows")
lmatched <- matchds_1to2(lmatched[[1]], lmatched[[2]], mi1 = "columns",
 mi2 <- "columns")
# check matches
ds1m <- lmatched[[1]]
ds2m <- lmatched[[2]]
identical(rownames(ds1m), rownames(ds2m))
identical(colnames(ds1m), colnames(ds2m))
```

query_si *query_si*

Description

Query an HNSW search index. Does K Nearest Neighbors lookup on a previously saved search index object, returning the K nearest neighbors of the queried sample(s). The ‘query_si()‘ function returns verbose output, which can be silenced with suppressMessages()‘.

Usage

```
query_si(
  sample_idv,
  fh_csv_fpath,
  si_fname = "new_search_index",
  si_fpath = ".",
  lkval = c(1, 2)
)
```

Arguments

sample_idv	Vector of valid sample IDs, or GSM IDs, which are included in the rownames of the hashed features table at fh_csv_fpath (requried, vector of char strings).
fh_csv_fpath	Path to the hashed features table, which includes rownames corresponding to sample ID strings in the sample_idv vector (required, char).
si_fname	Base filename of the search index object, used to find the search index and index dict files, which are expected to be located at si_fapth (required, char).
si_fpath	Path to the directory containing the search index and index dict files (required, char).
lkval	Vector of K nearest neighbors to return per query (optional, int, c(1,2)).

Value

Nearest neighbors results of search index query.

Examples

```
# file paths
# fh table
# fh_csv_fname <- system.file("extdata", "fhtest",
# package = "recountmethylation")
# fh_csv_fname <- file.path(fh_csv_fname, "bval_fh10.csv")
# si dict
# index_dict_fname <- system.file("extdata", "sitest",
# package = "recountmethylation")
# index_dict_fname <- file.path(index_dict_fname, "new_index_dict.pickle")
```

```
# set sample ids to query
# sample_idv <- c("GSM1038308.1548799666.hlink.GSM1038308_5958154021_R01C01",
#               "GSM1038309.1548799666.hlink.GSM1038309_5958154021_R02C01")
# set a list of k nearest neighbors to query
# lkval <- c(1,2,3)

# get query results as a data frame (with verbose results messaging)
# dfk <- query_si(sample_idv = sample_idv, lkval = lkval,
#               fh_csv_fname = "bval_fn.csv",
#               index_dict_fname = "new_index_dict.pickle")
# returns:
# Starting basilisk process...
# Defining the virtual env dependencies...
# Running virtual environment setup...
# Sourcing Python functions...
# Querying the search index...
# Getting hashed features data for samples...
# Getting index data for sample:
# GSM1038308.1548799666.hlink.GSM1038308_5958154021_R01C01'
# Getting index data for sample:
# GSM1038309.1548799666.hlink.GSM1038309_5958154021_R02C01'
# Beginning queries of k neighbors from lk...
# ii =  0 , ki =  1
# Loading search index...
# Querying 2 elements in data with k = 1 nearest neighbors...
# Query completed, time: 0.0007359981536865234
# Applying labels to query results...
# Returning data (sample id, k index, and distance)...
# ii =  1 , ki =  2
# Loading search index...
# Querying 2 elements in data with k = 2 nearest neighbors...
# Query completed, time: 0.0006208419799804688
# Applying labels to query results...
# Returning data (sample id, k index, and distance)...
# ii =  2 , ki =  3
# Provided k '3' > n si samples, skipping...
# Returning query results...
```

`rgse` *Form an RGChannelSet from a list containing signal data matrices*

Description

Forms an RGChannelSet from signal data list. This is called by certain queries to h5 files.

Usage

```
rgse(ldat, verbose = FALSE)
```

Arguments

ldat	List of raw signal data query results. Must include 2 data.frame objects named "redsignal" and "greensignal."
verbose	Whether to post status messages.

Value

Returns a RGChannelSet object from raw signal dataset queries.

See Also

getrg(), RGChannelSet()

Examples

```
path <- system.file("extdata", "h5test", package = "recountmethylation")
fn <- list.files(path)
dbpath <- file.path(path, fn)
rg <- getrg(dbn = dbpath, all.gsm = TRUE, metadata = FALSE)
dim(rg) # [1] 11162     2
class(rg)
# [1] "RGChannelSet"
# attr(,"package")
# [1] "minfi"
```

---

servermatrix *servermatrix*

Description

Called by get_rmdl() to get a matrix of database files and file info from the server. Verifies valid versions and timestamps in filenames, and that h5se directories contain both an assays and an se.rds file.

Usage

```
servermatrix(
  dn = NULL,
  sslver = FALSE,
  printmatrix = TRUE,
  url = "https://methylation.recount.bio/",
  verbose = FALSE
)
```

Arguments

dn	Server data returned from RCurl (default NULL).
sslver	Whether to use SSL certificate authentication for server connection (default FALSE).
printmatrix	Whether to print the data matrix to console (default TRUE).
url	Server website url (default "https://methylation.recount.bio/").
verbose	Whether to show verbose messages (default FALSE).

Value

Matrix of server files and file metadata

See Also

get_rmdl, smfilt

Examples

```
dn <- "remethdb-h5se_gr-test_0-0-1_1590090412 29-May-2020 07:28 -"
sm <- get_servermatrix(dn = dn)
```

---

setup_sienv *setup_sienv*

Description

Set up a new virtual environment for search index construction using the basilisk package.

Usage

```
setup_sienv(
  env.name = "dnam_si_hnswlib",
  pkgv = c("python==3.7.1", "hnswlib==0.5.1", "pandas==1.2.2", "numpy==1.20.1",
    "mmh3==3.0.0", "h5py==3.2.1")
)
```

Arguments

env.name	Name of the new virtual environment (required, "dnam_si-hnswlib")
pkgv	Vector of the dependencies and their versions for the new virtual environment (required, format: "packagename==versionnum").

Value

New basilisk environment object.

smfilt *smfilt*

Description

Filters the data matrix returned from servermatrix().

Usage

```
smfilt(sm, typesdf = NULL)
```

Arguments

sm	Data matrix returned from servermatrix().
typesdf	Data.frame containing database file info for dm filters.

Value

Filtered data matrix of server file info.

See Also

get_rmdl, servermatrix

Examples

```
dm <- matrix(c("remethdb_h5-rg_epic_0-0-2_1589820348.h5","08-Jan-2021",
"09:46","66751358297"), nrow = 1)
smfilt(dm)
```

Index