Package ‘bigmelon’

January 19, 2026

Type Package

Title Illumina methylation array analysis for large experiments

Version 1.36.0

Description Methods for working with Illumina arrays using gdsfmt.

License GPL-3

Depends R (>= 3.3), wateRmelon (>= 1.25.0), gdsfmt (>= 1.0.4), methods, minfi (>= 1.21.0), Biobase, methylumi

Imports stats, utils, GEOquery, graphics, BiocGenerics, illuminaio

Suggests BiocGenerics, RUnit, BiocStyle, minfiData, parallel, IlluminaHumanMethylation450kanno.ilmn12.hg19, IlluminaHumanMethylationEPICanno.ilm10b2.hg19, bumphunter

LazyLoad yes

Collate zzz.R es2gds.R dasenGds.R dbGdsn.R dfsfitGdsn.R gds2mlumi.R gdsnclass_methods.R inout.R pfilterGds.R prcompGdsn.R pwodGdsn.R qnGdsn.R comboGds.R ranknorm.R GEOtoGDS.R ecc.gdsn.R bumphunterGdsn.R chainsaw.R

biocViews DNAMethylation, Microarray, TwoChannel, Preprocessing, QualityControl, MethylationArray, DataImport, CpGIsland

NeedsCompilation no

RoxygenNote 7.3.1

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

git_branch RELEASE_3_22

git_last_commit 8527321

git_last_commit_date 2025-10-29

Repository Bioconductor 3.22

Date/Publication 2026-01-19

Author Tyler J. Gorrie-Stone [aut],
Ayden Saffari [aut],
Karim Malki [aut],
Leonard C. Schalkwyk [cre, aut]

Maintainer Leonard C. Schalkwyk <lschal@essex.ac.uk>

Contents

`bigmelon-package` *Write large-scale Illumina array datasets to CoreArray Genomic Data Structure (GDS) data files for efficient storage, access, and modification.*

Description

Functions for storing Illumina array data as CoreArray Genomic Data Structure (GDS) data files (via the gdsfmt package), appending these files , and applying array normalization methods from the wateRmelon package.

Details

Package:	bigmelon
Type:	Package
Version:	1.13.8
Date:	2020-02-24
License:	GPL3

Author(s)

Tyler Gorrie-Stone Leonard C Schalkwyk, Ayden Saffari, Karim Malki. Who to contact: <t.gorrie-stone@qmul.ac.uk>, <lschal@essex.ac.uk>

References

[1]Tyler J Gorrie-Stone, Melissa C Smart, Ayden Saffari, Karim Malki, Eilis Hannon, Joe Burrage, Jonathan Mill, Meena Kumari, Leonard C Schalkwyk: Bigmelon: tools for analysing large DNA methylation datasets, Bioinformatics, Volume 35, Issue 6, 15 March 2019, Pages 981-986. https://doi.org/10.1093/bioinformatics/bty713

[2]Pidsley R, Wong CCY, Volta M, Lunnon K, Mill J, Schalkwyk LC: A data-driven approach to preprocessing Illumina 450K methylation array data. BMC genomics, 14(1), 293.

See Also

`es2gds`, `dasen`, `wateRmelon`, `gdsfmt`.

`app2gds` *Append a MethyLumiSet object to a preexisting gds file*

Description

This function will append a MethyLumiSet object to a gds file and return a gds.class object.

Usage

```
app2gds(m, bmln)
```

Arguments

`m`	The MethyLumiSet object to be appended to the gds file, with the same number of rows as the gds file.
`bmln`	Either: A gds.class object Or: A character string specifying the filepath of an existing .gds file to write to. Or: A character string specifying the file path of a new .gds file to write to

Details

Currently this function only takes a MethyLumiSet object as the only type of eligible input. This function will also produce unexpected results if the number of rows of the new object does not match the existing .gds file. Hopefully the function will noisily fail if this is the case however to prevent any errors from occuring it is recommended that raw idat files are read in using `readEPIC` or appended with `iadd` or `iadd2` to ensure that all rows are of the same length and have the same annotation.

Value

A gds.class object pointed towards the newly appended .gds file.

Author(s)

Leonard C Schalkwyk, Ayden Saffari, Tyler Gorrie-Stone Who to contact: <t.gorrie-stone@qmul.ac.uk>

See Also

es2gds, iadd, iadd2.

Examples

```
#load example dataset
data(melon)

#split data into halves
melon_1 <- melon[,1:6]
melon_2 <- melon[,7:12]

#convert first half to gds
e <- es2gds(melon_1,'1_half_melon.gds')

#append second half to existing gds file
f <- app2gds(melon_2,e)

unlink("1_half_melon.gds")
```

---

backup.gdsn *Copy gds node to a backup folder within gds object*

Description

This function will copy a designated gdsn.class object stored inside a gds object to a backup folder (aptly named backup). If the backup folder does not exist, this will be created. This is a wrapper to `copyto.gdsn` which should be used if one wishes to copy a gds node to a seperate gds file.

Usage

```
backup.gdsn(gds = NULL, node)
```

Arguments

gds	If NULL, function will call `getfolder.gdsn` to find the root node. Otherwise, user can specify a separate gds.class object to copy the specified node to.
node	gdsn.class object (a gds node) which can be specified using `index.gdsn`

Value

A gdsobject with an additional folder called backup with supplied node within.

Author(s)

Tyler Gorrie-Stone <t.gorrie-stone@qmul.ac.uk>

See Also

copyto.gdsn

Examples

```
data(melon)
e <- es2gds(melon, "melon.gds")
nod <- index.gdsn(e, "betas")
backup.gdsn(gds = NULL, node = nod)
closefn.gds(e)
unlink("melon.gds")
```

---

bigmelon-accessors *Bigmelon accessors*

Description

Functions to access data nodes in gds.class objects.

Usage

```
## S4 method for signature 'gds.class'
betas(object)
## S4 method for signature 'gds.class'
methylated(object)
## S4 method for signature 'gds.class'
unmethylated(object)
## S4 method for signature 'gds.class'
pvals(object)
## S4 method for signature 'gds.class'
fData(object)
## S4 method for signature 'gds.class'
pData(object)
## S4 method for signature 'gds.class'
QCmethylated(object)
## S4 method for signature 'gds.class'
QCunmethylated(object)
## S4 method for signature 'gds.class'
QCrownames(object)
## S4 method for signature 'gds.class'
getHistory(object)
## S4 method for signature 'gds.class'
colnames(x, do.NULL=TRUE, prefix=NULL)
## S4 method for signature 'gds.class'
rownames(x, do.NULL=TRUE, prefix=NULL)
## S4 method for signature 'gds.class'
exprs(object)
## S4 method for signature 'gds.class'
fot(x)
```

Arguments

`object`	A gds.class object.

for colnames and rownames:

`x`	A gds.class object.
`do.NULL`	logical. If 'FALSE' and names are 'NULL', names are created.
`prefix`	prefix: for created names.

Details

Each function will return the data stored in the corresponding node as either a gdsn.class object, matrix, or data.frame. These are names following the conventions of the methylumi package and perform similar functions.

Each function which returns a gdsn.class object can be further index using the '[' operators. This includes an additional name argument which optionally returns the named attributes to the data as these are not stored inside the gdsn.node.

The QC functions will return the QC data as a matrix, these are seperated for methylation and unmethylated values and the rownames.

exprs will return a data.frame from beta values for all probes and all samples.

Value

Returns specified node representing the called accessor

Author(s)

Leonard C Schalkwyk, Ayden Saffari, Tyler Gorrie-Stone Who to contact: <t.gorrie-stone@qmul.ac.uk>

Examples

```
data(melon)
e <- es2gds(melon,'wat_melon.gds')
betas(e)
head(betas(e)[,])
methylated(e)[1:5, 1:3]
unmethylated(e)[1:3 ,1:5]
pvals(e)[1:5, 1:5]
head(fData(e))
head(pData(e))
head(colnames(e))
head(rownames(e))

closefn.gds(e)
unlink("wat_melon.gds")
```

bigmelon-internal *Internal bigmelon functions*

Description

Internal bigmelon functions, not intended for end user. What are you doing here!?

Usage

```
newgds(file)
handle(gds)
findgdsobj(gds)
subSet(x, i, j, ..., drop = FALSE)
```

Arguments

`file`	A character string specifying the name of the .gds file to write to.
`gds`	Either a gds.class object, or a character string specifying an existing .gds file.
`x`	A gds.class object
`i`	rows (probes) to select for subsetting
`j`	columns (samples) to select for subsetting
`drop`	passed on to '[' indexing operator.

Details

newgds creates a new .gds file stub with the given name. handle is used by app2gds to return a file handle for the given object. findgdsobj is used by handle to search the workspace for a gds.class object linked to the file name specified. subSet is used by internal functions to select a subset of rows (probes) and columns (samples) from a .gds file (overwriting the existing).

Value

Nothing.

Author(s)

Leonard C Schalkwyk, Ayden Saffari, Tyler Gorrie-Stone Who to contact: <t.gorrie-stone@qmul.ac.uk>

Examples

```
data(melon)
gfile <- es2gds(melon, "melon.gds")
betas(gfile)[1:5,1:3]
closefn.gds(gfile)
unlink("melon.gds")
```

bigmelon-normalization *Bigmelon Quantile Normalization methods.*

Description

Set of functions that are used to perform quantile normalization methods on gds.class objects.

Usage

```
## S4 method for signature 'gds.class'
dasen(mns, fudge = 100, ret2 = FALSE, node="betas",...)
dasen.gds(gds, node, mns, uns, onetwo, roco, fudge, ret2)
qn.gdsn(gds, target, newnode)
db.gdsn(gds, mns, uns)
dfsfit.gdsn(gds, targetnode, newnode, roco, onetwo)
```

Arguments

`gds`	A gds.class object
`node`	The "name" of desired output `gdsn.class` node
`mns`	The `gdsn.class` node that corresponds to "methylated" intensities.
`uns`	The `gdsn.class` node that corresponds to "unmethylated" intensities.
`onetwo`	The `gdsn.class` node that corresponds to probe designs (in reference to 450k and EPIC arrays) OR character string pointing to location of `gdsn.class` node. e.g. "fData/DESIGN" OR vector containing probe design types of length > 1.
`roco`	This allows a background gradient model to be fit. This is split from data column names by default. roco=NULL disables model fitting (and speeds up processing), otherwise roco can be supplied as a character vector of strings like 'R01C01' (only 3rd and 6th characters used).
`fudge`	The value added to total intensity to prevent denominactors close to zero when calculation betas. default = 100
`ret2`	if TRUE, appends the newly calculated methylated and unmethylated intensities to original gds (as specified in gds arguement). Will overwrite the raw intensities.
`target`	Target `gdsn.class` node to perform normalization on. If using "*****.gds" method you do not need to specify this.
`targetnode`	Target `gdsn.class` node to perform normalization on. If using "*****.gds" method you do not need to specify this.
`newnode`	"name" of desired output `gdsn.class` node. If using "*****.gds" method you do not need to specify this.
`...`	Additional args such as roco or onetwo.

Details

Each function performs a normalization method described within the wateRmelon package. Functions: `qn.gdsn`, `design.qn.gdsn`, `db.gdsn` and `dfsfit.gdsn` are described to allow users to create their own custom normalization methods. Otherwise calling `dasen` or `dasen.gds` e.t.c will perform the necessary operations for quantile normalization.

Each 'named' normalization method will write a temporary gds object called "temp.gds" into the current working directory and it is removed when normalization is complete. Current methods supplied by default arguments will replace the raw betas with normalized betas, but leave the methylated and unmethylated intensities unprocessed.

Value

Normalization methods return nothing but will affect the gds file and replace/add nodes given to the function.

Author(s)

Tyler J Gorrie-Stone <t.gorrie-stone@qmul.ac.uk>

See Also

`wateRmelon`, `dasen`

Examples

```
data(melon)
e <- es2gds(melon,'wat_melon.gds')
dasen(e)
closefn.gds(e) # Close gds object
unlink('wat_melon.gds') # Delete Temp file
```

`bigPepo` *tea.gds - description and validation for gds objects*

Description

Checks the validity, file location and read/write status of a gds object

Usage

```
bigPepo(path, gds, manifest, chunksize = NULL, force = TRUE, ...)
```

Details

For DNAm use with the bigmelon package. For interactive use the default verbose option also prints further details. Returns a list.

`bumphunterEngine.gdsn` *Bumphunter using bigmelon*

Description

Estimate regions for which a genomic profile deviates from its baseline value. Originally implemented to detect differentially methylated genomic regions between two populations. Functions identically to `bumphunter`.

Usage

```
bumphunterEngine.gdsn(mat, design, chr = NULL, pos, cluster = NULL, coef = 2, cutoff = NULL, pickCutc
```

Arguments

`mat`	A gdsn.class object (e.g betas(gfile)
`design`	Design matrix with rows representing samples and columns representing covariates. Regression is applied to each row of mat.
`chr`	A character vector with the chromosomes of each location.
`pos`	A numeric vector representing the chromosomal position.
`cluster`	The clusters of locations that are to be analyzed together. In the case of microarrays, the clusters are many times supplied by the manufacturer. If not available the function `clusterMaker` can be used to cluster nearby locations.
`coef`	An integer denoting the column of the design matrix containing the covariate of interest. The hunt for bumps will be only be done for the estimate of this coefficient.
`cutoff`	A numeric value. Values of the estimate of the genomic profile above the cutoff or below the negative of the cutoff will be used as candidate regions. It is possible to give two separate values (upper and lower bounds). If one value is given, the lower bound is minus the value.
`pickCutoff`	Should bumphunter attempt to pick a cutoff using the permutation distribution?
`pickCutoffQ`	The quantile used for picking the cutoff using the permutation distribution.
`maxGap`	If cluster is not provided this maximum location gap will be used to define cluster via the `clusterMaker` function.
`nullMethod`	Method used to generate null candidate regions, must be one of 'bootstrap' or 'permutation' (defaults to 'permutation'). However, if covariates in addition to the outcome of interest are included in the design matrix (ncol(design)>2), the 'permutation' approach is not recommended. See vignette and original paper for more information.
`smooth`	A logical value. If TRUE the estimated profile will be smoothed with the smoother defined by `smoothFunction`
`smoothFunction`	A function to be used for smoothing the estimate of the genomic profile. Two functions are provided by the package: `loessByCluster` and `runmedByCluster`.
`useWeights`	A logical value. If `TRUE` then the standard errors of the point-wise estimates of the profile function will be used as weights in the loess smoother `loessByCluster`. If the `runmedByCluster` smoother is used this argument is ignored.

`B`	An integer denoting the number of resamples to use when computing null distributions. This defaults to 0. If `permutations` is supplied that defines the number of permutations/bootstraps and `B` is ignored.
`permutations`	is a matrix with columns providing indexes to be used to scramble the data and create a null distribution when `nullMethod` is set to permutations. If the bootstrap approach is used this argument is ignored. If this matrix is not supplied and `B`>0 then these indexes are created using the function `sample`.
`verbose`	logical value. If `TRUE`, it writes out some messages indicating progress. If `FALSE` nothing should be printed.
`...`	further arguments to be passed to the smoother functions.

Details

This function is a direct replication of the `bumphunter` function by Rafael A. Irizarry, Martin J. Aryee, Kasper D. Hansen, and Shan Andrews.

Author(s)

Original Function by Rafael A. Irizarry, Martin J. Aryee, Kasper D. Hansen, and Shan Andrews. Bigmelon implementation by Tyler Gorrie-Stone Who to contact if this all goes horribly wrong: <t.gorrie-stone@qmul.ac.uk>

`cantaloupe` *Small MethyLumi 450k data sets for testing*

Description

Small MethyLumi 450k data sets intended for testing purposes only.

Usage

```
data(cantaloupe)
data(honeydew)
```

Format

cantaloupe: MethyLumiSet with assayData containing 841 features, 3 samples. honeydew: MethyLumiSet with assayData containing 841 features, 4 samples.

Value

Loads data into R

chainsaw *Chainsaw – modify gds file by subsetting all nodes*

Description

Currently the '[' function for the gds.class objects used by bigmelon only subsets a single node. This function does more like what you would normally expect a subsetting function to do, it returns a subset of the entire object. It may in future be a replacement for '[.gds.class'.

Usage

```
chainsaw(gfile, i = "", j = "", v = FALSE, cleanup = TRUE)
```

Arguments

`gfile`	A gds.class object.
`i`	Specifies rows (ie probes) in the desired subset, similar to behaviour of '['
`j`	Specifies columns (ie sampless) in the desired subset, similar to behaviour of '['
`v`	If true, spew many messages.
`cleanup`	If true, run a cleanup function that can substantially reduce the file size.

Details

This function is intended for use in the preprocessing and QC phase of a DNA methylation workflow. For efficiency, bigmelon stores data in a file, and the gds.class object is a file handle. True to its name, chainsaw chops the underlying file, this is a side affect of the function and is not affected by assignment of the return value.

Value

a gds.class object. This is a handle to the same file that the gfile argument points to. It's not generally useful to have two handles to the same file, but it may make code more readable. In interactive use, if not assigned, the returned object is usefully pretty-printed.

combo.gds *Combine two different gds objects together*

Description

Function will attempt to combine together the shared gdsn.class nodes between two gds object depending on the dimensions of the primary gds.class object.

Usage

```
combo.gds(file, primary, secondary)
```

Arguments

`file`	Name of the new gds file to be created.
`primary`	A gds.class object.
`secondary`	A gds.class object.

Details

–EXPERIMENTAL– Will crudely combine shared nodes between primary and secondary based on the dimensions / rownames of the primary node. NAs will be coerced where probes are missing from secondary gds.

Currently will only look for nodes with the names "betas", "methylated", "unmethylated", "pvals" and "NBeads".

Value

Returns (and creates) as new gds file in the specified location with the combination of two gds objects together.

Note

Will lose information relating to "pData". Therefore we recommend compiling separate pData object manually and adding combined pData post-function

Author(s)

Tyler Gorrie-Stone <t.gorrie-stone@qmul.ac.uk>

Examples

```
data(melon)
a <- es2gds(melon[,1:6], "primary.gds")
b <- es2gds(melon[,7:12], "secondary.gds")

ab <- combo.gds("combo.gds", primary = a, secondary = b)

closefn.gds(a)
unlink("primary.gds")
closefn.gds(b)
unlink("secondary.gds")
closefn.gds(ab)
unlink("combo.gds")
```

`dasenrank` *Dasen Quantile Normalization by storing ranks*

Description

This performs an experimental variant of dasen normalisation for .gds format objects which stores the ranks of the methylated and unmethylated intensities inside of the normalised values and interpolates the quantiles when they are needed.

Notably this eliminates a secondary re-sorting pass which is required by quantile normalisatoin as it will be performed downstream using `computebeta.gds` which will produce normalise betas or manually with '[' which will access the ranks and interpolate the specific quantiles as needed.

Usage

```
dasenrank(gds, mns, uns, onetwo, roco, calcbeta = NULL, perc = 1)
computebeta.gds(gds, new.node, mns, uns, fudge)
```

Arguments

`gds`	gds.class object which contains methylated and unmethylated intensities. The function will write two (four) nodes to this object called 'mnsrank' and 'unsrank' which contain the ranks of the given nodes.
`mns`	gdsn.class object OR character string that refers to location in gds that relates to the (raw) methylated intensities.
`uns`	gdsn.class object OR character string that refers to location in gds that relates to the (raw) unmethylated intensities.
`onetwo`	gdsn.class object OR character string that refers to location in gds that contains information relating to probe design OR vector of length equal to the number of rows in the array that contains 'I' and 'II' in accordance to Illumina HumanMethylation micro-array design.
`roco`	Sentrix (R0#C0#) position of all samples.
`calcbeta`	Default = NULL, if supplied with a string, a new gdsn.node will be made with supplied string, which will contain the calculated betas.
`perc`	A number between 0 and 1 that relates to the given proportion of columns that are used to normalise the data. Default is set to 1, but incase there are lots of samples to normalise this number can be reduce to increase speed of code.
`new.node`	Character string depicting name of new betas node in given gds object.
`fudge`	Arbitrary value to offset low intensities

Details

calcbeta is a known bottle-neck for this code! Also function is highly experimental.

Value

Nothing is returned to the R environment, however the supplied gds will have 4 or 5 gdsn.nodes added. These are: 'mnsrank', 'unsrank', 'isnamnsrank' (hidden), 'isnaunsrank'(hidden) and calcbeta if supplied. 'mnsrank' and 'unsrank' have been given some attributes - which contain the calculated quantiles from `getquantilesandranks`.

Author(s)

Tyler Gorrie-Stone Who to contact: <t.gorrie-stone@qmul.ac.uk>

Examples

```
data(melon)
e <- es2gds(melon, "melon.gds")
#dasenrank(gds = e)
closefn.gds(e)
unlink("melon.gds")
```

`dim.gds.class` *dim.gds.class S3 method returning dimensions of data represented by a gds file handle.*

Description

dim.gds.class S3 method returning dimensions of data represented by a gds file handle.

Usage

```
## S3 method for class 'gds.class'
dim(gfile, v = FALSE)
```

`es2gds` *Coersion method for MethyLumiSet, RGChannelSet or MethylSet to CoreArray Genomic Data Structure (GDS) data file*

Description

The es2gds function takes a MethyLumiSet, RGChannelSet or MethylSet data object and converts it into a CoreArray Genomic Data Structure (GDS) data file (via the gdsfmt package), returning this as a gds.class object for use with bigmelon.

Usage

```
es2gds(m, file, qc = TRUE)
```

Arguments

`m`	A MethyLumiSet, RGChannelSet or MethylSet object
`file`	A character string specifying the name of the .gds file to write to.
`qc`	When set to true (default), data from control probes included.

Value

A gds.class object, which points to the newly created .gds file.

Author(s)

Leonard C Schalkwyk, Ayden Saffari, Tyler Gorrie-Stone Who to contact: <t.gorrie-stone@qmul.ac.uk>

See Also

`app2gds`, `iadd`.

Examples

```
#load example dataset
data(melon)
#convert to gds
e <- es2gds(melon,'melon.gds')
closefn.gds(e)
unlink('melon.gds')
```

---

estimateCellCounts *Cell Proportion Estimation using bigmelon*

Description

Estimates the relative proprotion of pure cell types within a sample, identical to estimateCellCounts. Currently, only a reference data-set exists for 450k arrays. As a result, if performed on EPIC data, function will convert gds to 450k array dimensions (this will not be memory efficient).

Usage

```
    estimateCellCounts.gds(
        gds,
        gdPlatform = c("450k", "EPIC", "27k"),
        mn = NULL,
        un = NULL,
        bn = NULL,
        perc = 1,
        compositeCellType = "Blood",
        probeSelect = "auto",
        cellTypes = c("CD8T","CD4T","NK","Bcell","Mono","Gran"),
        referencePlatform = c("IlluminaHumanMethylation450k",
            "IlluminaHumanMethylationEPIC",
            "IlluminaHumanMethylation27k"),
        returnAll = FALSE,
        meanPlot = FALSE,
        verbose=TRUE,
        ...)
```

Arguments

gds	An object of class gds.class, which contains (un)normalised methylated and unmethylated intensities
gdPlatform	Which micro-array platform was used to analysed samples
mn	'Name' of gdsn node within gds that contains methylated intensities, if NULL it will default to 'methylated' or 'mnsrank' if dasenrank was used prior
un	'Name' of gdsn node within gds that contains unmethylated intensities, if NULL it will default to 'unmethylated' or 'unsrank' if dasenrank was used prior
bn	'Name' of gdsn node within gds that contains un(normalised) beta intensities. If NULL - function will default to 'betas'.

`perc`	Percentage of query-samples to use to normalise reference dataset. This should be 1 unless using a very large data-set which will allow for an increase in performance
`compositeCellType`	Which composite cell type is being deconvoluted. Should be either "Blood", "CordBlood", or "DLPFC"
`probeSelect`	How should probes be selected to distinguish cell types? Options include "both", which selects an equal number (50) of probes (with F-stat p-value < 1E-8) with the greatest magnitude of effect from the hyper- and hypo-methylated sides, and "any", which selects the 100 probes (with F-stat p-value < 1E-8) with the greatest magnitude of difference regardless of direction of effect. Default input "auto" will use "any" for cord blood and "both" otherwise, in line with previous versions of this function and/or our recommendations. Please see the references for more details.
`cellTypes`	Which cell types, from the reference object, should be we use for the deconvolution? See details.
`referencePlatform`	The platform for the reference dataset; if the input `rgSet` belongs to another platform, it will be converted using `convertArray`.
`returnAll`	Should the composition table and the normalized user supplied data be return?
`verbose`	Should the function be verbose?
`meanPlot`	Whether to plots the average DNA methylation across the cell-type discrimating probes within the mixed and sorted samples.
`...`	Other arguments, i.e arguments passed to plots

Details

See `estimateCellCounts` for more information regarding the exact details. estimateCellCounts.gds differs slightly, as it will impose the quantiles of type I and II probes onto the reference Dataset rather than normalising the two together. This is 1) More memory efficient and 2) Faster - due to not having to normalise out a very small effect the other 60 samples from the reference set will have on the remaining quantiles.

Optionally, a proportion of samples can be used to derive quantiles when there are more than 1000 samples in a dataset, this will further increase performance of the code at a cost of precision.

`finalreport2gds` *Read finalreport files and convert to genomic data structure files*

Description

Function to easily load Illumina methylation data into a genomic data structure (GDS) file.

Usage

```
finalreport2gds(finalreport, gds, ...)
```

Arguments

`finalreport`	A filename of the text file exported from GenomeStudio
`gds`	The filename for the gds file to be created
`...`	Additional arguments passed to `methylumiR`

Details

Creates a .gds file.

Value

A gds.class object

Author(s)

Tyler Gorrie-Stone Who to contact: <t.gorrie-stone@qmul.ac.uk>

Examples

```
finalreport <- "finalreport.txt"
## Not run: finalreport2gds(finalreport, gds="finalreport.gds")
```

---

gds2mlumi *Convert Genomic Data Structure file to Methylumiset or Methylset object.*

Description

Convert a Genomic Data Structure object back into a methylumi object, with subsetting features.

Usage

```
gds2mlumi(gds, i, j)
gds2mset(gds, i, j, anno)
```

Arguments

gds	a gds object
i	Index of rows
j	Index of Columns
anno	If NULL, function will attempt to guess the annotation to be used. Otherwise can be specified with either "27k", "450k", "epic" or "unknown".

Value

A methylumi object

Author(s)

Tyler Gorrie-Stone Who to contact: <t.gorrie-stone@qmul.ac.uk>

Examples

```
data(melon)
e <- es2gds(melon, "melon.gds")
gds2mlumi(e)
closefn.gds(e)
unlink("melon.gds")
```

GEOtoGDS *Download data from GEO and convert it into a gdsfmt object*

Description

Uses the GEOquery R package to download a GSE Accession into the current working directory. This will only work for GSE's that have raw idat files associated with them.

Usage

```
geotogds(geo, gds, method = "wget", keepidat = F, keeptar = F, ...)
```

Arguments

`geo`	Either a GEO accession number ('GSE########') or a previously downloaded tarball 'GSE######.tar.gz'
`gds`	A character string that specifies the path and name of the .gds file you want to write to.
`method`	Character value to indicate which method should be used to download data from GEO. Default is 'wget'
`keepidat`	Logical, indicate whether or not raw idat files in the working directon should be removed after parsing, if FALSE: idat files will be removed.
`keeptar`	Logical, indicate whether or not the downloaded tarball should be removed after parsing, if FALSE: the tarball will be removed.
`...`	Additional Arguments to pass to other functions (if any)

Value

geotogds will return a gds.class object that will point towards a the newly created .gds file with majority of downloaded contents inside.

Author(s)

Tyler Gorrie-Stone Who to contact: <t.gorrie-stone@qmul.ac.uk>

Examples

```
#load example dataset
# gfile <- geotogds("GSE*******", "Nameoffile.gds")
# Will not work if gds has no idats submitted. May also fail if idats
# are not deposited in a way readily readable by readEPIC().
# closefn.gds(gfile)
```

getquantilesandranks *Compute the quantiles and ranks for a given gdsn.node*

Description

Used inside dasenrank to generate the quantiles for both type 'I' and type 'II' probes to normalise DNA methylation data using bigmelon.

Usage

```
getquantilesandranks(gds, node, onetwo, rank.node = NULL, perc = 1)
```

Arguments

gds	A gds.class object
node	A gdsn.class object, or a character string that refers to a node within supplied gds.
onetwo	gdsn.class object OR character string that refers to location in gds that contains information relating to probe design OR vector of length equal to the number of rows in the array that contains 'I' and 'II' in accordance to Illumina HumanMethylation micro-array design. This can be obtained with fot(gds)
rank.node	Default = NULL. If supplied with character string, function will calculate the ranks of given node and store them in gds. Additionally, the computed quantiles will now instead be attributed to rank.node which can be accessed with get.attr.gdsn
perc	A number between 0 and 1 that relates to the given proportion of columns that are used to normalise the data. Default is set to 1, but in cases where there many of samples to normalise this number can be reduced to increase speed of code.

Details

Used in dasenrank, can be used externally for testing purposes.

Value

If rank.node is NULL. A list containing quantiles, intervals and supplied probe design will be returned. If rank.node was supplied, nothing will be returned. Instead a new node will be created in given gds that has the otherwise returned list attached as an attribute. Which can be accessed with get.attr.gdsn

Author(s)

Tyler Gorrie-Stone Who to contact: <t.gorrie-stone@qmul.ac.uk>

Examples

```
data(melon)
e <- es2gds(melon, "melon.gds")
output <- getquantilesandranks(gds = e, 'methylated', onetwo = fot(e), perc = 1, rank.node = NULL)
# with-out put.
#getquantilesandranks(gds = e, 'methylated', onetwo = fot(e), perc = 1, rank.node = 'mnsrank')
```

```
closefn.gds(e)
unlink("melon.gds")
```

`iadd` *Add data from Illumina IDAT files to a gds file.*

Description

iadd will add data from multiple, specified, idat files providing '/path/to/barcode' is valid path to a specified gds file. Barcode here implies the first part of the idat file name i.e without '_(Red|Grn).idat'

iadd2 will add data from all idat files that are stored within a single directory to a gds file.

idats2gds will add data from a set of barcodes into the same gds file, one by one, optionally will handle idat files from different maps providing force=TRUE. However, will not combine from arrays of differing types e.g. 450k vs EPIC.

Usage

```
iadd(bar, gds, n = TRUE, force = TRUE, target_cpgs = NULL, ...)
iadd2(path, gds, chunksize = NULL, force=TRUE, ...)
idats2gds(barcodes, gds, n=TRUE, force = FALSE, ...)
```

Arguments

`bar`	The barcode for an IDAT file OR the file path of the file containing red or green channel intensities for that barcode (this will automatically locate and import both files regardless of which one you provide)
`path`	The file path where (multiple) IDAT files exist. iadd2 will process every idat within the specified directory.
`gds`	Either: A gds.class object Or: A character string specifying the name of an existing .gds file to write to. Or: A character string specifying the name of a new .gds file to write to
`chunksize`	If NULL, iadd2 will read in all barcodes in one go. Or if supplied with a numeric value, iadd2 will read in that number of idat files in batches
`n`	Logical, whether or not bead-counts are extracted from idat files.
`force`	Logical, whether or not rownames from the first idat file are applied to all idat files. Useful in combining together idat files of differing lengths.
`target_cpgs`	A vector of CpGs to specifically read in and set the dimensions of array to.
`barcodes`	A vector of barcodes to load into an existing gds file.
`...`	Additional Arguments passed to wateRmelons methylumIDATepic.

Value

returns a gds.class object, which points to the appended .gds file.

Author(s)

Tyler Gorrie-Stone, Leonard C Schalkwyk, Ayden Saffari. Who to contact: <tyler.gorrie-stone@diamond.ac.uk>

See Also

es2gds, app2gds.

Examples

```
if(require('minfiData')){
bd <- system.file('extdata', package='minfiData')
gfile <- iadd2(file.path(bd, '5723646052'), gds = 'melon.gds')
closefn.gds(gfile)
unlink('melon.gds')
}
```

`pfilter.gds` *Basic data filtering for Illumina methylation data in gds objects*

Description

The pfilter function filters data sets based on bead count and detection p-values. The user can set their own thresholds or use the default pfilter settings. This specific function will take a Genomic Data Structure (GDS) file as input and perform pfilter similar to how `pfilter` in wateRmelon is performed.

Usage

```
## S4 method for signature 'gds.class'
pfilter(mn, perCount = NULL, pnthresh = NULL,
perc = NULL, pthresh = NULL)
```

Arguments

`mn`	a gds object OR node corresponding to methylated intensities
`perCount`	Threshold specifying which sites should be removed if they have a given percentage of samples with a beadcount <3, default = 5
`pnthresh`	cut off for detection p-value, default= 0.05
`perc`	remove sample having this percentage of sites with a detection p-value greater than pnthresh, default = 1
`pthresh`	Threshold specifying which sites should be removed if they have a given percentage of samples with a detection p-value greater than pnthresh, default = 1

Value

See `pfilter`. If using pfilter.gds, function If using pfilter.gds function will return a list of containing two locical vectors of length(nrow) and lneght(ncol) which can be used to subset data. Otherwise if called using pfilter data will be subsetted automatically.

Author(s)

Tyler Gorrie-Stone, Original (wateRmelon) Function by Ruth Pidsley Who to Contact: <t.gorrie-stone@qmul.ac.uk

See Also

pfilter

Examples

```
data(melon)
e <- es2gds(melon, "melon.gds")
pfilter(e)
closefn.gds(e)
unlink("melon.gds")
```

---

prcomp.gds.class — *Principal Component Analysis for high-dimensional data*

Description

Performs principal components analysis on the given gds object and returns the results as an object of class "prcomp".

Usage

```
## S3 method for class 'gds.class'
prcomp(x, node.name, center = FALSE, scale. = FALSE,
rank. = NULL, retx = FALSE, tol = NULL, perc = 0.01,
npcs = NULL, parallel = NULL, method = c('quick', 'sorted'), verbose = FALSE, ...)
```

Arguments

`x`	A gds.class object.
`node.name`	Name of the gdsn.class node to learn principal components from
`center`	Logical value indicating whether variables should be shifted to be zero centered.
`scale.`	Logical value indicating whether the variables should be scaled to have unit variance
`tol`	a value indicating the magnitude below which components should be omitted.
`rank.`	(Still functional) Number of principal components to be returned
`retx`	a logical value indicating whether the rotated variables should be returned.
`perc`	The percentage of the number of rows that should be used to calculate principle components. Ranging from 0 to 1, a value of 1 would indicicate all rows will be used.
`npcs`	Number of principal components to be returned
`parallel`	Can supply either a cluster object (made from makeCluster) or a integer describing the number of cores to be used. This is only used if method="sorted".
`method`	Indicates whhich method to use out of "quick" and "sorted". "quick" stochastically selects number of rows according to perc. And the supplies them to svd. "sorted" determines the interquartile range for each row then selects the top percentage (according to perc) of probes with the largest interquartile range and supplies selected rows to svd.
`verbose`	A logical value indicating whether message outputs are displayed.
`...`	arguments passed to or from other methods. If "x" is a formula one might specify "scale." or "tol".

Details

The calculation is done by a singular value decomposition of the (centered and possibly scaled) data matrix, not by using "eigen" on the covariance matrix. This is generally the preferred method for numerical accuracy. The "print" method for these objects prints the results in a nice format and the "plot" method produces a scree plot.

Value

An object of prcomp class

Examples

```
data(melon)
e <- es2gds(melon, "melon.gds")
prcomp(e, node.name="betas", perc=0.01, method='quick')
closefn.gds(e)
unlink("melon.gds")
```

---

pwod.gdsn *Probe-Wise Outlier Detection for DNA methylation data.*

Description

Function performed outlier detection for each probe (row) using Tukey's Interquartile Range method.

Usage

```
pwod.gdsn(node, mul = 4)
```

Arguments

node	gdsn.class node that contains the data matrix to be filtered
mul	The number of interquartile ranges used to determine outlying probes. Default is 4 to ensure only very obvious outliers are removed.

Details

Detects outlying probes across arrays in methylumi and minfi objects.

Value

Nothing is returned. However the supplied gds object (of-which the node is a child of) will have a new node with NAs interdispersed where outliers are found.

Author(s)

Tyler Gorrie-Stone Who to contact: <t.gorrie-stone@qmul.ac.uk>

See Also

pwod

Examples

```
data(melon)
e <- es2gds(melon, "melon.gds")
pwod(e)
closefn.gds(e)
unlink("melon.gds")
```

`redirect.gds` *Change the location of the paths for row and column names in a gds file.*

Description

Change how a gds object ascribes row and column names by changing the paths node of a gds file. Although a bit laborious, the row and column information is not preserved inside a node which contains data. Thusly during accession the row and column names are attributed after data has been accessed.

Usage

```
redirect.gds(gds, rownames, colnames)
```

Arguments

`gds`	`gds.class` object containing node named "paths".
`rownames`	Character string that points to named part of supplied gds that corresponds to rownames. e.g. "fData/Target_ID". Default = "fData/Probe_ID"
`colnames`	Character string that points to names part of supplied gds that corresponds to colnames. e.g. "pData/Sample_ID". Default = "pData/barcode"

Details

This function is particularly important within many functions inside bigmelon and may lead to downstream errors if the row and column names are not correctly specified. If data is read in through es2gds the path nodex should be correctly set up and all downstream analysis will carry out as normal. Will fail noisily if given a pathway that does not exist.

Value

Changes the gdsn.class node named "paths" to supplied rownames and colnames within supplied gds.class object.

Author(s)

Tyler J. Gorrie-Stone Who to contact: <t.gorrie-stone@qmul.ac.uk>

See Also

`add.gdsn`, `app2gds`, `es2gds`

Examples

```
data(melon)
e <- es2gds(melon, "melon.gds") # Create gds object
redirect.gds(e, rownames = "fData/TargetID", colnames = "pData/sampleID")
# Deleting Temp files
closefn.gds(e)
unlink("melon.gds")
```

---

wm-port *Functions imported from wateRmelon*

Description

Manual page for methods for the extraenous functions from wateRmelon. For more details for specific functions see the respective manual pages in wateRmelon.

Usage

```
## S4 method for signature 'gdsn.class,gdsn.class'
qual(norm, raw)
## S4 method for signature 'gds.class'
predictSex(x, x.probes=NULL, pc=2, plot=TRUE, irlba=TRUE, center=FALSE, scale.=FALSE)
## S4 method for signature 'gdsn.class'
predictSex(x, x.probes=NULL, pc=2, plot=TRUE, irlba=TRUE, center=FALSE, scale.=FALSE)
```

Arguments

`norm`	normalized node (gdsn.class)
`raw`	raw node (gdsn.class)
`x`	gdsclass object or node corresponding to betas
`x.probes`	Default is NULL, is required to be supplied in bigmelon. logical or numeric vector containing indicies of X chromosome probes
`pc`	The principal component to guess sex across (usually the 2nd one in most cases)
`plot`	Logical, indicated whether or not to plot the prediction
`irlba`	Logical, indicate whether or not to use the faster method to generate principal components
`center`	Logical, indicate whether or not to center data around 0
`scale.`	Logical, indicate whether or not to scale data prior to prcomp

Details

For the full usage and description of any functions that link to this manual page, please visit the respective manual pages from wateRmelon.

Value

Returns expected output of functions from wateRmelon

See Also

`wateRmelon`

Index