Package 'bigmelon'

April 14, 2017

Type Package

Title Illumina methylation array analysis for large experiments

Version 1.0.0

Date 2016-09-28

Author Tyler Gorrie-Stone, Ayden Saffari, Karim Malki, Leonard C Schalkwyk

Maintainer Tyler Gorrie-Stone <tgorri@essex.ac.uk>

Description Methods for working with Illumina arrays using gdsfmt.

License GPL-3

Depends R (>= 3.3), wateRmelon (>= 1.17.1), gdsfmt (>= 1.0.4), methods, methylumi, minfi, Biobase

Imports stats, utils

Suggests BiocGenerics, BiocStyle, minfiData

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

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

NeedsCompilation no

R topics documented:

`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:	0.1.2
Date:	2014-09-04
License:	GPL3

Author(s)

Tyler Gorrie-Stone Leonard C Schalkwyk, Ayden Saffari, Karim Malki. Who to contact: <tgorri@essex.ac.uk>, <leonard.schalkwyk@kcl.ac.uk>

References

[1]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.`, `methylumi`.

`app2gds` *Append a MethyLumiSet object to a CoreArray Genomic Data Structure (GDS) data file*

Description

This function will append a MethyLumiSet data object to a CoreArray Genomic Data Structure(GDS) data file, and return as a gds.class object.

Usage

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

Arguments

m	The MethyLumiSet object to be appended to the CoreArray Genomic Data Structure (GDS) data file
bmln	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

Value

A gds.class object, which points to the appended .gds file.

Author(s)

Leonard C Schalkwyk, Ayden Saffari, Tyler Gorrie-Stone Who to contact: <tgorri@essex.ac.uk>

See Also

es2gds, iadd.

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)

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

---

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

Description

Quick function that will copy designated gdsn.class node within a gds object to a 'backup'folder. If 'backup' folder does not exist, this is created.

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)

Value

gds object is modified to have a new folder 'backup' with supplied node copied inside

Author(s)

Tyler Gorrie-Stone <tgorri@essex.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)
```

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 returns the data stored in the corresponding node as either a gdsn.class object or a 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 indexed using matrix-like '[' operations. With an optional name argument which optionally allows for row and col names to be automatically appended to returned matrix.

The QC functions (returns QCdata split into separate matrices for methylated values, unmethylated values, and probe names)

exprs returns a data.frame of beta values for all probes across all samples.

Value

Returns specified node representing the called accessor

Author(s)

Leonard C Schalkwyk, Ayden Saffari, Tyler Gorrie-Stone Who to contact: <tgorri@essex.ac.uk>

See Also

bigmelon, methylumi

Examples

```
data(melon)
e <- es2gds(melon,'wat_melon.gds')
betas(e)
betas(e)[,]
```

```
methylated(e)[1:5, ]
unmethylated(e)[ ,1:5]
pvals(e)[1:5, 1:5]
fData(e)
pData(e)
colnames(e)
rownames(e)
exprs(e)

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

---

bigmelon-normalization

Bigmelon Quantile Normalization methods.

Description

Functions used to perform quantile normalization 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)
design.qn.gdsn(gds, target, newnode, onetwo)
db.gdsn(gds, mns, uns)
dfsfit.gdsn(gds, targetnode, newnode, roco, onetwo)
```

Arguments

gds	A gds.class object
node	"name" of desired output gdsn.class node
mns	gdsn.class node that corresponds to "methylated" intensities.
uns	gdsn.class node that corresponds to "unmethylated" intensities.
onetwo	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	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 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 ("temp.gds") in the current working directory and is remove it when normalization is complete. Current methods supplied by default arguments will replace the raw intensities with normalized intensities.

Value

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

Author(s)

Tyler J Gorrie-Stone <tgorri@essex.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
```

---

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

`combo.gds` *Combine two different gds objects.*

Description

Combines the shared gdsn.class nodes between two gds objects depending on primary gds.object dimensions.

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

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.

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

Value

a new gds object that has both files within it

Note

Will lose information relating to "pData". Therefore we recommend compiling separate pData object manually.

Author(s)

Tyler Gorrie-Stone <tgorri@essex.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")
```

---

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: <tgorri@essex.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')
```

`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

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

Examples

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

---

iadd *Add data from IDAT files for a single barcode to a gds file*

Description

Add data from IDAT files for a single barcode to a gds file. or Add data from many IDAT files from a single directory to a gds file.

Usage

```
iadd(bar, gds,...)
iadd2(path, gds, chunksize = NULL, ...)
```

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
...	Additional Arguments passed to 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: <tgorri@essex.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	remove sites having this 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	remove sites having this percentage of sample with a detection p-value greater than pnthresh, default = 1

Value

See pfilter

Author(s)

Tyler Gorrie-Stone, Original (wateRmelon) Function by Ruth Pidsley

See Also

pfilter

Examples

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

---

prcomp.gdsn.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 'gdsn.class'
prcomp(x, center = FALSE, scale. = FALSE, tol = NULL,
rank. = NULL, retx = FALSE, perc = 0.01, npcs = NULL, ...)
```

Arguments

x	GDS Node
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.	Old. (Still functional) Number of principal components to be returned
retx	a logical value indicating whether the rotated variables should be returned.
perc	Percentage of data to be used.
npcs	Number of principal components to be returned
...	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(betas(e))
closefn.gds(e)
unlink("melon.gds")
```

`pwod.gdsn` *Probe-Wise Outlier Detection*

Description

'P'robe-'W'ise 'O'utlier 'D'etection via interquartile ranges

Usage

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

Arguments

`node`	gdsn.class object containing array to be filtered
`mul`	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. Outliers are probable low MAF/SNP heterozygotes.

Value

Nothing returned. Supplied gds object will have new node with outlier probes coerced to NAs

Author(s)

Tyler Gorrie-Stone

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 file 'paths' for row and column names.*

Description

Quickly change contents of gdsn.class node "paths".

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

Function is important within many functions and can lead to errors if row and column names are not correctly specified. By default, es2gds can recognize whether a methylumiset object was read in through readEPIC or methylumiR and will set the row and col names paths correctly. Will fail noisily if given a pathway that does not exist.

Value

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

Author(s)

Tyler J. Gorrie-Stone Who to contact: <tgorri@essex.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 ported from wateRmelon*

Description

methods for extraenous functions from wateRmelon see respective manual pages.

Usage

```
## S4 method for signature 'gdsn.class,gdsn.class'
qual(norm, raw)
```

Arguments

norm	normalized node
raw	raw node

Details

Filler Text

Value

Returns expected output of functions from wateRmelon

See Also

wateRmelon

Index