Package 'bigmelon'

October 17, 2017

Type Package

Title Illumina methylation array analysis for large experiments

Version 1.2.0

Date 2017-04-06

Description Methods for working with Illumina arrays using gdsfmt.

License GPL-3

Depends R (>= 3.3), wateRmelon (>= 1.19.1), gdsfmt (>= 1.0.4), methods, methylumi, minfi (>= 1.21.0), Biobase

Imports stats, utils, GEOquery, graphics

Suggests BiocGenerics, BiocStyle, minfiData, parallel

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

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

NeedsCompilation no

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

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

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:	1.1.13
Date:	2017-04-06
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

Details

Currently, the function only works with MethylumiSet objects and will produce unexpected results if the number of rows of the new objects does not match the existing .gds object. To prevent any errors from occuring it is recommended that raw .idat files are read in with `readEPIC` or appended `iadd` to ensure that all rows are the same length and have the same annotation.

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. 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)

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)
## 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 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.

fot returns a vector corresponding to probe design (either 'I' or 'II') which is used for normalising differences between probe designs.

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 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 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

–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.

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 and adding combined pData post-function

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")
```

---

dasenrank *Dasen Quantile Normalization by storing ranks*

Description

Performs 'dasen' normalization for .gds format objects while storing the ranked methylated/unmethylated intensities until they are needed down stream.

This will eliminate the secondary re-sorting required by quantile normalisation as it is perfomed down-stream - either by using `computebeta.gds` or manually with '['

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 J. Gorrie-Stone - tgorri at essex.ac.uk

Examples

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

`estimateCellCounts` *Cell Proportion Estimation using bigmelon*

Description

Estimates 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 = 0.25,
    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

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")
```

GEOtoGDS *Download GEO data and pipe into a gdsfmt object*

Description

Uses GEOquery to download a GSE Accession into the current working directory, only works for GSE's with raw idat files.

Usage

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

Arguments

geo	A GEO Accession number. "GSE########" or a tarball 'GSE######.tar.gz'
gds	A character string specifying the name of the .gds file to write to.
method	String to indicate what method to download data from GEO off. Default is 'wget'
keepidat	Logical, indicates whether or not to keep the raw idat files in the working directory after downloading - if FALSE, removed
keeptar	Logical, indicates whether or not to keep the downloaded tarball in the working directory - if FALSE, removed.
...	Additional Arguments to pass to other functions (if any)

Value

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

Author(s)

Tyler Gorrie-Stone Who to contact: <tgorri@essex.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 Human-Methylation micro-array design. 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 J. Gorrie-Stone Who to contact: <tgorri@essex.ac.uk>

Examples

```
data(melon)
e <- es2gds(melon, "melon.gds")
output <- getquantilesandranks(gds = e, 'methylated', onetwo = fot(e), perc = 1, rank.node = NULL)
str(output)
# 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 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. If using pfilter.gds, function will return a list of two elements. Both logical vectors with length(nrow) and length(ncol), which can be used for subsetting. Otherwise will be used to subset data directly.

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.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.	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
parallel	Can supply either a cluster object (made from makeCluster) or a integer depicting number of cores to be used. This is only used if method="sorted".
method	Indicates what 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*

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