lumi

April 19, 2010

`addControlData2lumi` *Add the control probe data into the controlData slot of LumiBatch object*

Description

Add the control probe profile data, outputted by BeadStudio, into the controlData slot of LumiBatch object.

Usage

```
addControlData2lumi(controlData, x.lumi)
```

Arguments

`controlData`	the control data can be a data.frame or the control probe filename outputted by BeadStudio
`x.lumi`	a LumiBatch object, to which controlData will be added.

Details

The controlData slot in LumiBatch object is a data.frame with first two columns as "controlType" and "ProbeID". The rest columns are the expression amplitudes for individual samples.

Value

Return the LumiBatch object with controlData slot filled.

Author(s)

Pan Du

See Also

`getControlData`, `plotControlData`

Examples

```
## Not runnable
# controlFile <- 'Control_Probe_Profile.txt'
# x.lumi <- addControlData2lumi(controlFile, x.lumi)
```

`addNuID2lumi` *Add the nuID information to the LumiBatch object*

Description

Replace the Illumina Id (Target ID or Probe Id) as nuID (nucleotide universal identifier) for indexing genes in the LumiBatch object

Usage

```
addNuID2lumi(x.lumi, annotationFile=NULL, sep = NULL, lib.mapping = NULL,  annot
```

Arguments

`x.lumi`	a LumiBatch object
`annotationFile`	a annotation file, which includes the Illumina ID (target or probe ids) and probe sequence information
`sep`	the separation used in the annotation file. Automatically detect the separator if it is "," or "\t".
`lib.mapping`	a Illumina ID mapping package, e.g, lumiHumanIDMapping
`annotationColName`	the annotation column name in the annotation file used for the probe sequence and TargetID and ProbeID
`verbose`	a boolean to decide whether to print out some messages

Details

Since the default Illumina IDs (TargetID (ILMN_Gene ID) and ProbeId (Probe_Id)) are not consistent between different arrays and batches, we invented a nuID, which is one-to-one matching with the probe sequence. This function is to replace the Illumina ID with the nuID. If the annotation library (the unzipped manifest file (.bgx)) is provided, the function will automatically check whether the Illumina ID is provided for the microarray data. We recommend output the data using ProbeID when using Illumina BeadStudio software, because the TargetID (ILMN_Gene ID) are not unique.

Value

a LumiBatch object with Illumina ID replaced by nuID.

Author(s)

Pan Du

References

Du, P., Kibbe, W.A., Lin, S.M., "nuID: A universal naming schema of oligonucleotides for Illumina, Affymetrix, and other microarrays", submitted.

See Also

IlluminaID2nuID, lumiR

Examples

```
## load example data
# data(example.lumi)

## specify the annotation file for the Illumina chip
# annotationFile <- 'Human_RefSeq-8.csv'
## Replace the Target ID with nuID
# lumi.nuID <- addNuID2lumi(example.lumi, annotationFile)

## An alternative way is to load the Annotation library and match the targetID (or Probe
# lumi.nuId <- addNuID2lumi(example.lumi, lib.mapping='lumiHumanIDMapping')
```

---

affyExpresso *Preprocess Affymetrix data by integrating VST with expresso method*

Description

Preprocess Affymetrix data by integrating VST with expresso method

Usage

```
affyExpresso(afbatch, bg.correct = TRUE, bgcorrect.method = NULL, bgcorrect.para
```

Arguments

afbatch	a vector of CEL file names or an AffyBatch object, see AffyBatch-class
bg.correct	a boolean to express whether background correction is wanted or not
bgcorrect.method	the name of the background adjustment method
bgcorrect.param	a list of parameters for bgcorrect.method (if needed/wanted)
variance.stabilize	a boolean to express whether variance stabilization is wanted or not
varianceStabilize.method	the name of the variance stabilizing transform, same as lumiT function
varianceStabilize.param	a list of parameters for transformation method
normalize	normalization step wished or not
normalize.method	the normalization method to use

`normalize.param`
: a list of parameters to be passed to the normalization method (if wanted)

`pmcorrect.method`
: the name of the PM adjustement method

`pmcorrect.param`
: a list of parameters for pmcorrect.method (if needed/wanted)

`summary.method`
: the method used for the computation of expression values

`summary.param`
: a list of parameters to be passed to the summary.method (if wanted)

`summary.subset`
: a list of 'affyids'. If NULL, a expression summary value is computed for everything on the chip

`verbose`
: logical value. If TRUE it writes out some messages

Details

This function basically integrates the VST (variance stabilizing transformation) transformation into the expresso function in the affy package. The variance stabilization is based on the mean and variance relations of pixel intensities of each probe.

Value

Return an object of class ExpressionSet.

Note

The performance of this function is still under evaluation.

Author(s)

Pan Du

See Also

`rma` and `vst`

`affyVstRma` *Preprocess Affymetrix data by integrating VST with RMA method*

Description

Preprocess Affymetrix data by integrating VST with RMA method

Usage

```
affyVstRma(afbatch, bgcorrect.method = "none", bgcorrect.param = list(), VST.par
```

Arguments

afbatch	a vector of CEL file names or an AffyBatch object, see AffyBatch-class
bgcorrect.method	the name of the background adjustment method
bgcorrect.param	a list of parameters for bgcorrect.method (if needed/wanted)
VST.param	a list of parameters for vst method
verbose	logical value. If TRUE it writes out some messages.
...	other parameters used by rma function

Details

This function basically integrates the VST (variance stabilizing transformation) transformation into the rma function in the affy package. The variance stabilization is based on the mean and variance relations of pixel intensities of each probe.

Value

Return an object of class ExpressionSet.

Note

The performance of this function is still under evaluation.

Author(s)

Pan Du

See Also

expresso and vst

---

bgAdjust *Background adjustment for Illumina data*

Description

The method adjusts the data by subtracting an offset, which is estimated based on the quantile of the control probes

Usage

```
bgAdjust(lumiBatch, probs = 0.5, ...)
```

Arguments

lumiBatch	A LumiBatch object with controlData slot include control probe information
probs	The quantile used to estimate the background
...	other parameters used by quantile method

Details

The method adjusts the data by subtracting an offset, which is estimated based on the quantile of the control probes. The control probe information is kept in the controlData slot of the LumiBatch object. If no control data information, the method will do nothing.

Value

It returns a LumiBatch object with background adjusted.

Author(s)

Pan Du

See Also

lumiB

Examples

```
data(example.lumi)
## Here will assume the minimum of the control probe as the background,
##    because there is no negative control (blank beads) information for the Barnes data.
example.lumi.b <- bgAdjust(example.lumi, probs=0)
```

---

boxplot-methods *boxplot of a ExpressionSet object*

Description

Creating boxplot of sample intensities in a ExpressionSet object

Usage

```
## S4 method for signature 'ExpressionSet':
boxplot(x, range = 0, main, logMode = TRUE, subset = 5000, ...)
```

Arguments

x	a ExpressionSet object
range	parameter of boxplot
main	title of the boxplot
logMode	whether plot the data in log2 scale or not
subset	subset of rows used to plot. It can be an index vector, or the length of a random subset
...	optional arguments to boxplot.

Details

The boxplot function has a "subset" parameter. By default, it is set as 5000, i.e., randomly selected 5000 probes to plot the boxplot. The purpose of this is to plot the picture faster, but it will also make the boxplot has slightly different each time. If the user wants to make sure the boxplot is the same each time, you can set the "subset" parameter as NULL.

See Also

LumiBatch-class, boxplot

Examples

```
## load example data
data(example.lumi)

boxplot(example.lumi)
```

---

density-methods *Density plot of a ExpressionSet object*

Description

Creating density plot of sample intensities in a ExpressionSet object. It is equivalent to hist-methods.

Usage

```
## S4 method for signature 'ExpressionSet':
density(x, logMode=TRUE, xlab = NULL, ylab = "density", type = "l",
col=1:dim(x)[2], lty=1:dim(x)[2], lwd=1, xlim = NULL, index.highlight = NULL, co
symmetry = NULL, addLegend = TRUE, subset = 5000, main="", ...)
```

Arguments

x	a ExpressionSet object
logMode	determine whether the density plot is based on a log2 scale
xlab	xlab of the density plot
ylab	ylab of the density plot
type	parameter of plot function
col	line colors of the density plot
lty	line types of the density plot
lwd	line width of plot function
xlim	parameter of the plot function
index.highlight	the column index of the highlighted density curve
color.highlight	color of highlighted density curve
symmetry	the boundary position suppose to be symmetric distributed
addLegend	whether add legend to the plot or not
subset	subset of rows used to plot. It can be an index vector, or the length of a random subset
main	title for the plot
...	additional parameters for density function

See Also

LumiBatch-class, hist-methods, density

Examples

```
## load example data
data(example.lumi)

density(example.lumi)
```

`detectionCall` *Estimate the detectable probe ratio*

Description

Estimate the detectable probe ratio of each probe, sample or just return an AP matrix

Usage

```
detectionCall(x.lumi, Th = 0.01, type = c('probe', 'sample', 'matrix'))
```

Arguments

`x.lumi`	a LumiBatch object
`Th`	the threshold. By default, when the detection p-value is less than 0.01, we suppose it is detectable. For the old version of BeadStudio output (version 2 or earlier), the threshold will automatically transferred as 1 - Th, because in the old format, value close to 1 is suppose to be detectable.
`type`	determine to calculate the detection count by probe or by sample

Value

If the type is 'probe', then returns the presentCount of each probe. If the type is 'sample', then return the detectable probe ratio of each sample. If the type is 'matrix', then return the AP matrix, in which 'A' represents absent (the detect p-value less than threshold) and 'P' represents present.

Author(s)

Pan Du

See Also

lumiQ

Examples

```
## load example data
data(example.lumi)
## load example data
data(example.lumi)

## estimate the detect call (percentage of expressed genes) of each sample
temp <- detectionCall(example.lumi, type='sample')
print(temp)

## estimate the present count of each gene (probe)
temp <- detectionCall(example.lumi, type='probe')
hist(temp)
```

---

detectOutlier *Detect the outlier sample (or gene)*

Description

Detect the outlier sample (or gene) based on distance to the cluster center

Usage

```
detectOutlier(x, metric = "euclidean", standardize = TRUE, Th = 2, ifPlot = FALS
```

Arguments

x	a LumiBatch object, ExpressionSet object or a matrix with each column corresponding to a sample or other profile
metric	the distance matric
standardize	standardize the profile or not
Th	the threshold of outlier,
ifPlot	to plot the result (as a hierarchical tree) or not

Details

The current outlier detection is based on the distance from the sample to the center (average of all samples after removing 10 percent samples farthest away from the center). The assumption of the outlier detection is that there is only one single cluster and the distance from the sample to the center is Gaussian distributed.

The outlier is detected when its distance to the center is larger than a certain threshold. The threshold is calculated as Th * median distances to the center.

The profile relations can be visualized as a hierarchical tree.

Value

Plot the results or return the outlier (a logic vector) with the distance matrix and threshold as attributes.

Author(s)

Pan Du

See Also

lumiQ

Examples

```
## load example data
data(example.lumi)

## detect the outlier (Further improvement needed.)
temp <- detectOutlier(example.lumi, ifPlot=TRUE)
```

---

estimateLumiCV *Estimate the coefficient of variance matrix of LumiBatch object*

Description

Estimate the coefficient of variance matrix of LumiBatch object for each measurement or probe.

Usage

```
estimateLumiCV(x.lumi, type = c("measurement", "probe"), ifPlot = FALSE, ...)
```

Arguments

x.lumi	a LumiBatch object
type	estimate the coefficient of variance of each measurement or each probe
ifPlot	determince whether to plot the density plot or not
...	optional arguments to plot.

Details

By default, the coefficient of variance is the ratio of the mean and variance of the bead expression values. Basically, it is the ration of exprs and se.exprs element of LumiBatch object. If the type is "probe", it is the ratio of the mean and variance of probe expression profile.

Value

A matrix of coefficient of variance

Author(s)

Pan Du

See Also

lumiQ

Examples

```
## load example data
data(example.lumi)

## estimate the coefficient of variance and plot the density plot of it
cv <- estimateLumiCV(example.lumi, ifPlot = TRUE)
```

`example.lumi` *Example LumiBatch object includes example data*

Description

Example data as a LumiBatch object which is a subset of Barnes data (Barnes, 2005)

Usage

```
data(example.lumi)
```

Format

A 'LumiBatch' object

Details

The data is from (Barnes, 2005). It used Sentrix HumanRef-8 Expression BeadChip. Two samples "100US" and "95US:5P" (each has two technique replicates) were selected. In order to save space, 8000 genes were randomly selected. As a result, the example data includes 8000 genes, each has 4 measurements. The full data set was included in the Bioconductor Experiment data package lumiBarnes.

The entire data set has been built as a lumiBarnes data object and can be downloaded from Bioconductor Experiment Data.

References

Barnes, M., Freudenberg, J., Thompson, S., Aronow, B. and Pavlidis, P. (2005) Ex-perimental comparison and cross-validation of the Affymetrix and Illumina gene expression analysis platforms, Nucleic Acids Res, 33, 5914-5923.

The detailed data information can be found at: http://www.bioinformatics.ubc.ca/pavlidis/lab/platformCompare/

Examples

```
## load the data
data(example.lumi)

## summary of the data
example.lumi
```

`getChipInfo` *Get Illumina Chip Information based on probe identifiers*

Description

Retrieve the matched Illumina chip information by searching the provided probe identifiers through the Illumina identifiers in all manifest files.

Usage

```
getChipInfo(x, lib.mapping = NULL, species = c("Human", "Mouse", "Rat", "Unknown
```

Arguments

`x`	a vector of probe identifiers, ExpressionSet object or a matrix with probe identifiers as row names
`lib.mapping`	the ID mapping library. If it is provided, the parameter "species" will be ignored.
`species`	species of the chip designed for. If users do not know it, it can be set as "Unknown".
`chipVersion`	chipVersion information returned by function `getChipInfo`
`idMapping`	determine whether return the idMapping information (between Illumina ID and nuID)
`returnAllMatches`	determine whether return all matches or just the best match
`verbose`	determine whether print some warning information

Details

The function searches the provided probe Identifiers (Illumina IDs or nuIDs) through all the manifest file ID information kept in the IDMapping libraries (lumiHumanIDMapping, lumiMouseIDMapping, lumiRatIDMapping). The Illumina IDs kept in the library include "Search_key" ("Search_Key"), "Target" ("ILMN_Gene"), "Accession", "Symbol", "ProbeId" ("Probe_Id"). To determine the best match, the function calculate the number of matched probes. The higher "matchedProbeNumber" is claimed as better. When the "matchedProbeNumber" is the same, the manifest file with fewer probes is claimed as better. If x is NULL and chipVersion is provided, it will return the entire mapping table of the chip.

Value

The function returns a list with following items:

`chipVersion`	the file name of the manifest file for the corresponding version and release
`species`	the species of the chip designed for
`IDType`	the type of probe identifier
`chipProbeNumber`	the number of probes in the manifest file
`matchedProbeNumber`	the number of input probes matching the manifest file
`idMapping`	id mapping information between Illumina ID and nuID

When parameter "returnAllMatches" is TRUE, the items of "chipVersion", "IDType", "chipProbeNumber", "inputProbeNumber", "matchedProbeNumber" will be a vector corresponding to the matched manifest files, whose "matchedProbeNumber" is larger than zero, and the "idMapping" will be a matrix with each column corresponding to one matched manifest file. All of the items are sorted from the best match to worst (The higher "matchedProbeNumber" is claimed as better. When the "matchedProbeNumber" is the same, the manifest file with fewer probes is claimed as better.).

Author(s)

Pan Du

See Also

nuID2IlluminaID, IlluminaID2nuID

Examples

```
## load example data
data(example.lumi)
if (require(lumiHumanIDMapping)) {
chipInfo <- getChipInfo(example.lumi, species='Human')
chipInfo
}
```

---

getControlData *Get control probe information*

Description

Get control probe information from Bead Studio output or a LumiBatch object.

Usage

```
getControlData(x, type = c('data.frame', 'LumiBatch'), ...)
```

Arguments

x	the control data can be a LumiBatch object or the Control Probe Profile file outputted by BeadStudio
type	determine the return data type
...	other parameters used by lumiR function

Value

By default, it returns a data.frame with first two columns as "controlType" and "ProbeID". The rest columns are the expression amplitudes for individual samples. When type is 'LumiBatch', it returns a LumiBatch object, which basically is the return of lumiR without combining duplicated TargetIDs. As the return is a LumiBatch object, it includes more information, like probe number, detection p-value and standard error of the measurement.

Author(s)

Pan Du

See Also

addControlData2lumi

Examples

```
controlFile <- system.file('doc', 'Control_Probe_Profile.txt', package='lumi')
## return a data.frame
controlData <- getControlData(controlFile)
class(controlData)
names(controlData)

## return a LumiBatch object
controlData <- getControlData(controlFile, type='LumiBatch')
summary(controlData)
```

---

getControlProbe *Get the control probe Ids*

Description

Get the control probe Ids corresponding to the control probe type provided. The control probe ids are kept in the second column of controlData data.frame.

Usage

```
getControlProbe(controlData, type = NULL)
```

Arguments

controlData	a LumiBatch object including control data or a control data data.frame
type	the type of control probe (case insensitive), which can be get by using getControlType function

Value

returns the corresponding probe Ids for the control type.

Author(s)

Pan Du

See Also

addControlData2lumi

Examples

```
controlFile <- system.file('doc', 'Control_Probe_Profile.txt', package='lumi')
## return a data.frame
controlData <- getControlData(controlFile)
getControlType(controlData)
getControlProbe(controlData, type='housekeeping')
```

getControlType *Get the types of the control probes*

Description

Get the types of the control probes, which is in the first column of the controlData data.frame.

Usage

```
getControlType(controlData)
```

Arguments

controlData	a LumiBatch object including control data or a control data data.frame

Value

return the unique type of control probe type.

Author(s)

Pan Du

See Also

addControlData2lumi

Examples

```
controlFile <- system.file('doc', 'Control_Probe_Profile.txt', package='lumi')
## return a data.frame
controlData <- getControlData(controlFile)
getControlType(controlData)
```

getNuIDMappingInfo *get the mapping information from nuID to RefSeq ID*

Description

Get the mapping information (including mapping quality information) of nuIDs to the most recent RefSeq release. These information was kept in the IDMapping libraries.

Usage

```
getNuIDMappingInfo(nuID = NULL, lib.mapping)
```

Arguments

nuID	a vector of nuIDs. If it is NULL, all mappings will be returned.
lib.mapping	the ID mapping library

Details

The function basically return the nuID mapping information kept in the "nuID_MappingInfo" table of IDMapping libraries (lumiHumanIDMapping, lumiMouseIDMapping, lumiRatIDMapping). For more details of nuID mapping, please refer to the help of corresponding IDMapping library.

Value

It returns a data.frame with each row corresponding to an input nuID.

Author(s)

Warren Kibbe, Pan Du, Simon Lin

Examples

```
## load example data
data(example.lumi)
if (require(lumiHumanIDMapping)) {
nuIDs <- featureNames(example.lumi)
mappingInfo <- getNuIDMappingInfo(nuIDs, lib.mapping='lumiHumanIDMapping')
head(mappingInfo)
}
```

---

hist-methods *Density plot of a ExpressionSet object*

Description

Creating density plot of sample intensities in a ExpressionSet object. It is equivalent to density-methods.

Usage

```
## S4 method for signature 'ExpressionSet':
hist(x, ...)
```

Arguments

x	a ExpressionSet object
...	other parameters for density-methods function

See Also

LumiBatch-class, density-methods, hist

Examples

```
## load example data
data(example.lumi)

hist(example.lumi)
```

`id2seq` *Transfer a nuID as a nucleotide sequence*

Description

The nuID (nucleotide universal identifier) is uniquely corresponding to probe sequence. The nuID is also self-identification and error checking

Usage

```
id2seq(id)
```

Arguments

`id`	a nuID (nucleotide universal identifier)

Details

A reverse of `seq2id`. Please refer to reference for more details.

Value

a string of nucleotide sequence

Author(s)

Pan Du

References

Du, P., Kibbe, W.A. and Lin, S.M., "nuID: A universal naming schema of oligonucleotides for Illumina, Affymetrix, and other microarrays", Biology Direct 2007, 2:16 (31May2007).

See Also

`seq2id`

Examples

```
seq <- 'ACGTAAATTTCAGTTTAAAACCCCCCG'
id <- seq2id(seq)
id
id2seq(id)
```

`IlluminaID2nuID` *Matching Illumina IDs to nuID based on Illumina ID mapping library*

Description

Matching Illumina IDs to nuID based on Illumina ID mapping libraries.

Usage

```
IlluminaID2nuID(IlluminaID, lib.mapping=NULL, species = c("Human", "Mouse", "Rat
```

Arguments

`IlluminaID`	a vector of Illumina IDs
`lib.mapping`	the ID mapping library. If it is provided, the parameter "species" will be ignored.
`species`	the species of the chip designed for. If users do not know it, it can be set as "Unknown".
`chipVersion`	chipVersion information returned by function `getChipInfo`
`...`	other parameters of `getChipInfo`

Details

When the parameter "chipVersion" is not provided, this function basically returned the "idMapping" item returned by function `getChipInfo`.

Value

The mapping information from Illumina ID to nuID. It will be a matrix with each column corresponding to one matched manifest file when parameter "returnAllMatches" is TRUE. In this case, the columns are sorted from the best match to worst. If IlluminaID is NULL and chipVersion is provided, it will return all mapping information of the chip.

Author(s)

Pan Du

See Also

`getChipInfo`, `nuID2IlluminaID`

`inverseVST` *Inverse VST transform*

Description

Inverse transform of VST (variance stabilizing transform), see `vst`.

Usage

```
inverseVST(x, fun = c('asinh', 'log'), parameter)
```

Arguments

`x`	a VST transformed LumiBatch object or a numeric matrix or vector
`fun`	function used in VST transform
`parameter`	parameter of VST function

Details

Recover the raw data from VST transformed data returned by `vst`. This function can be directly applied to the VST transformed or VST + RSN normalized LumiBatch object to reverse transform the data to the original scale.

Value

Return the raw data before VST transform

Author(s)

Pan Du

References

Lin, S.M., Du, P., Kibbe, W.A., "Model-based Variance-stabilizing Transformation for Illumina Mi-croarray Data", submitted

See Also

`vst`

Examples

```
## load example data
data(example.lumi)

## get the gene expression mean for one chip
u <- exprs(example.lumi)[,1]
## get the gene standard deviation for one chip
std <- se.exprs(example.lumi)[,1]

## do variance stabilizing transform
```

```
transformedU <- vst(u, std)

## do inverse transform and recover the raw data
parameter <- attr(transformedU, 'parameter')
transformFun <- attr(transformedU, 'transformFun')
recoveredU <- inverseVST(transformedU, fun=transformFun, parameter=parameter)

## compare with the raw data
print(u[1:5])
print(recoveredU[1:5])

## do inverse transform of the VST + RSN processed data
lumi.N <- lumiExpresso(example.lumi[,1:2])
## Inverse transform.
## Note: as the normalization is involved, the processed data will be different from the
lumi.N.raw <- inverseVST(lumi.N)
```

`is.nuID` *nuID self-identification*

Description

Self-identify nuID (nucleotide universal identifier) by verify the check code value and the checksum value

Usage

```
is.nuID(id)
```

Arguments

`id`	nuId or other string

Value

Return TRUE if id is a nuID, or else return FALSE.

Author(s)

Pan Du

References

Du, P., Kibbe, W.A. and Lin, S.M., "nuID: A universal naming schema of oligonucleotides for Illumina, Affymetrix, and other microarrays", Biology Direct 2007, 2:16 (31May2007).

See Also

`seq2id`, `id2seq`

Examples

```
## check the function using a random sequence
id <- 'adfasdfafd'
is.nuID(id) # FALSE

## check the function using a read nuID
seq <- 'ACGTAAATTTCAGTTTAAAACCCCCCG'
id <- seq2id(seq)
is.nuID(id) # TRUE
```

LumiBatch-class *Class LumiBatch: contain and describe Illumina microarray data*

Description

This is a class representation for Illumina microarray data. It extends ExpressionSet.

Extends

Directly extends class ExpressionSet.

Creating Objects

```
new('LumiBatch', exprs = [matrix], se.exprs = [matrix], beadNum = [matrix],
detection = [matrix], phenoData = [AnnotatedDataFrame], history = [data.frame],
...)
```

LumiBatch instances are usually created through new("LumiBatch", ...). The arguments to new should include exprs and se.exprs, others can be missing, in which case they are assigned default values.

Objects can be created using the function lumiR.

Slots

Slot specific to LumiBatch:

history: a data.frame recording the operation history of the LumiBatch object.

controlData: a data.frame with first two columns as "controlType" and "ProbeID". The rest columns are the control probe expression amplitudes for individual samples.

QC: a the quality control information of the LumiBatch object, returned by lumiQ function.

Slots inherited from ExpressionSet:

assayData contains equal dimensional matrices: exprs (contains gene expression level, which is the mean of its bead replicates.), se.exprs (contains gene expression standard error, which is the standard error of its bead replicates.), beadNum (records the number of beads for the probe.), detection (records the detection p-value of the probe. The number is from [0,1]. By default, < 0.01 indicates good detection.). For more details of assayData, please see ExpressionSet

phenoData: See eSet

experimentData: See eSet

annotation: See eSet

Methods

Class-specific methods:

`se.exprs(LumiBatch)`, `se.exprs(LumiBatch,matrix)<-`: Access and set elements named `se.exprs` in the `AssayData-class` slot.

`beadNum(LumiBatch)`, `beadNum(LumiBatch)<-`: Access and set elements named `beadNum` in the `AssayData-class` slot. Use "beadNum(LumiBatch) <- NULL" to remove the beadNum element.

`detection(LumiBatch)`, `detection(LumiBatch)<-`: Access and set elements named `detection` in the `AssayData-class` slot. Use "detection(LumiBatch) <- NULL" to remove the detection element.

`getHistory(LumiBatch)`: Access the operation history of `LumiBatch` object.

Derived from `ExpressionSet` (For the directly inherited methods, please see `ExpressionSet` and `eSet`):

`combine(LumiBatch,missing)`: Combine two LumiBatch objects, including `history` slot. See `eSet`

`exprs(LumiBatch)`, `exprs(LumiBatch,matrix)<-`: Access and set elements named `exprs` in the `AssayData-class` slot.

`object[(i,j)`: Conduct subsetting of the data in a LumiBatch object

Standard generic methods (For the directly inherited methods, please see `ExpressionSet` and `eSet`):

`initialize(LumiBatch)`: Object instantiation, used by `new`; not to be called directly by the user.

`validObject(LumiBatch)`: Validity-checking method, ensuring that `exprs` and `se.exprs` is a member of `assayData`. Other validity check is the same as `checkValidity(ExpressionSet)`.

`show(LumiBatch)` A summary of the `LumiBatch` object.

Author(s)

Pan Du, Simon Lin

See Also

`lumiR`, `lumiT`, `lumiN`, `boxplot-methods`, `pairs-methods`, `MAplot-methods`

Examples

```
## load example data
data(example.lumi)

## show the summary of the data
# summary(example.lumi)
example.lumi

## get express matrix
temp <- exprs(example.lumi)

## get a subset
temp <- example.lumi[,1] ## retrieve the first sample
```

```
## get the probe id
featureNames(example.lumi)[1:3]

## combine LumiBatch objects
temp <- combine(example.lumi[,1], example.lumi[,3])
temp
```

---

lumiB *Background correction of Illumina data*

Description

Background correction of Illumina data

Usage

```
lumiB(x.lumi, method = c('none', 'bgAdjust', 'forcePositive', 'bgAdjust.affy'),
```

Arguments

x.lumi	an ExpressionSet inherited object or a data matrix with columns as samples and rows as genes. For 'bgAdjust' method, it should be a LumiBatch Object
method	the background correction method, it can be any function with a LumiBatch Object as the first argument and return a LumiBatch Object
verbose	a boolean to decide whether to print out some messages
...	other parameters used by the user provided background correction method

Details

We assume the BeadStudio output data is background corrected. So by default, it will do nothing. The 'bgAdjust' method will estimate the background based on the control probe information, which is kept in the controlData slot of LumiBatch object. The 'forcePositive' method will force all expression values to be positive by adding an offset (minus minimum value plus one), it does nothing if all expression values are positive. The purpose of this is to avoid NA when do logarithm transformation. 'none' does not but return the LumiBatch object. 'bgAdjust.affy' will call the bg.adjust function in affy package. User can also provide their own function with a LumiBatch Object as the first argument and return a LumiBatch Object with background corrected.

Thanks Kevin Coombes (M.D. Anderson Cancer Center) suggested adding this function.

Value

Return an object with background corrected. The class of the return object is the same as the input object x.lumi.

Author(s)

Pan Du, Kevin Coombes

See Also

bgAdjust, lumiExpresso

Examples

```
## load example data
data(example.lumi)

## Do the default background correction method
lumi.B <- lumiB(example.lumi, method='bgAdjust', probs=0)
```

---

lumiExpresso *From raw Illumina probe intensities to expression values*

Description

Goes from raw Illumina probe intensities to expression values

Usage

```
lumiExpresso(lumiBatch, bg.correct = TRUE, bgcorrect.param = list(method='bgAdju
varianceStabilize.param = list(), normalize = TRUE, normalize.param = list(), QC
QC.param = list(), verbose = TRUE)
```

Arguments

lumiBatch	a LumiBatch object, which can be the return of lumiR
bg.correct	a boolean to decide whether to do background correction or not
bgcorrect.param	a list of parameters of lumiB
variance.stabilize	a boolean to decide whether to do variance stabilization or not
varianceStabilize.param	a list of parameters of lumiT
normalize	a boolean to decide whether to do normalization or not
normalize.param	a list of parameters of lumiN
QC.evaluation	a boolean to decide whether to do quality control estimation before and after preprocessing
QC.param	a list of parameters of lumiQ
verbose	a boolean to decide whether to print out some messages

Details

The function is to encapsulate the major functions of Illumina preprocessing. It is organized in a similar way as the expresso function in affy package.

Value

return a processed LumiBatch object. The operation history can be track in the history slot of the object.

Author(s)

Pan Du

See Also

lumiB, lumiT, lumiN

Examples

```
## load example data
data(example.lumi)

## Do all the default preprocessing in one step
lumi.N <- lumiExpresso(example.lumi)

## Do customized preprocessing. No variance stabilizing or log transform, use Quantile no
lumi.N <- lumiExpresso(example.lumi, variance.stabilize=FALSE, normalize.param = list(met
```

---

lumiN *Between chip normalization of a LumiBatch object*

Description

A main function of between chip normalization of a LumiBatch object. Currently, four methods ("rsn", "ssn", "quantile", "loess", "vsn") are supported.

Usage

```
lumiN(x.lumi, method = c("quantile", "rsn", "ssn", "loess", "vsn", "rankinvarian
```

Arguments

x.lumi	an ExpressionSet inherited object or a data matrix with columns as samples and rows as genes
method	five different between chips normalization methods ("quantile", "rsn", "ssn", "loess", "vsn", "rankinvariant") are supported
verbose	a boolean to decide whether to print out some messages
...	other parameters used by corresponding method

Details

lumiN is an interface for different normalization methods. Currently it supports "RSN" (See rsn), "SSN" (See ssn), "loess" (See normalize.loess), "quantile" (See normalize.quantiles), "VSN" (See vsn) and "rankinvariant" (See rankinvariant) . See details in individual functions. Note: the "VSN" normalization should be directly applied to the raw data instead of the lumiT processed data.

Value

Return an object with expression values normalized. The class of the return object is the same as the input object x.lumi. If it is a LumiBatch object, it also includes the VST transform function and its parameters as attributes: "transformFun", "parameter". See `inverseVST` for details.

Author(s)

Pan Du, Simon Lin

See Also

`rsn`, `ssn`, `rankinvariant`

Examples

```
## load example data
data(example.lumi)

## Do lumi transform
lumi.T <- lumiT(example.lumi)

## Do lumi between chip normaliazation
lumi.N <- lumiN(lumi.T, method='rsn', ifPlot=TRUE)
```

---

lumi-package *A package for preprocessing Illumina microarray data*

Description

lumi R package is designed to preprocess the Illumina microarray (BeadArray) data. It includes functions of Illumina data input, quality control, variance stabilization, normalization and gene annotation.

Details

Package:	lumi
Type:	Package
Version:	1.1.0
Date:	2007-03-23
License:	LGPL version 2 or newer

Author(s)

Pan Du, Simon Lin Maintainer: Pan Du <dupan@northwestern.edu>

References

1. Du, P., Kibbe, W.A. and Lin, S.M., (2008) 'lumi: a pipeline for processing Illumina microarray', Bioinformatics 24(13):1547-1548

2. Lin, S.M., Du, P., Kibbe, W.A., (2008) 'Model-based Variance-stabilizing Transformation for Illumina Microarray Data', Nucleic Acids Res. 36, e11

3. Du, P., Kibbe, W.A. and Lin, S.M., (2007) 'nuID: A universal naming schema of oligonucleotides for Illumina, Affymetrix, and other microarrays', Biology Direct, 2, 16

`lumiQ` *Quality control evaluation of the LumiBatch object*

Description

Quality control evaluation of the LumiBatch object and returns a summary of the data

Usage

```
lumiQ(x.lumi, logMode = TRUE, detectionTh = 0.01, verbose = TRUE)
```

Arguments

`x.lumi`	a LumiBatch object
`logMode`	transform as log2 or not (the function can check whether it is already log transformed.)
`detectionTh`	the detection threshold used by `detectionCall`
`verbose`	a boolean to decide whether to print out some messages

Details

Quality control of a LumiBatch object includes estimating the mean and standard deviation of the chips, detectable probe ratio of each chip, sample (chip) relations, detecting outliers of samples (chips). The produced QC information is kept in the QC slot of LumiBatch class. The summary function will provide a summary of the QC information (See example).

Value

a LumiBatch object with QC slot keeping the QC information

Author(s)

Pan Du

See Also

`LumiBatch`, `plot,LumiBatch-method`

Examples

```
## load example data
data(example.lumi)

## Do quality control estimation
lumi.Q <- lumiQ(example.lumi)

## A summary of the QC
summary(lumi.Q, 'QC')

## Plot the results
## plot the pairwise sample correlation
plot(lumi.Q, what='pair')

## see more examples in "plot,LumiBatch-method" help documents
```

`lumiR.batch` *Read BeadStudio output files in batch*

Description

Read BeadStudio output files in batch and combine them as a single LumiBatch object

Usage

```
lumiR.batch(fileList, convertNuID = TRUE, lib.mapping = NULL, detectionTh = 0.01
```

Arguments

`fileList`	a vector of file names or a directory keeping the data files in the format of .csv
`convertNuID`	determine whether convert the probe identifier as nuID
`lib.mapping`	same as `lumiR` parameter lib.mapping (optional)
`detectionTh`	the p-value threshold of determining detectability of the expression. See more details in `lumiQ`
`QC`	determine whether to do quality control assessment after read in the data.
`transform`	determine whether to do transform after input each file
`sampleInfoFile`	a Tab-separated text file or a data.frame keeping the sample information (optional)
`verbose`	a boolean to decide whether to print out some messages
`...`	other parameters used by `lumiR`

Details

The function basically call lumiR for individual files and then combine the returns. The sampleInfoFile parameter is optional. It provides the sample information (for phenoData slot in LumiBatch object), it is a Tab-separated text file. ID column is required. It represents sample ID, which is defined based on the column names of BeadStudio output file. For example, sample ID of column "1881436070_A_STA.AVG_Signal" is "1881436070_A_STA". The sample ID column

can also be found in the "Samples Table.txt" file output by BeadStudio. Another "Label" column (if provided) will be used as the sampleNames of LumiBatch object. All information of sampleInfoFile will be directly added in the phenoData slot in LumiBatch object.

To save memory space in the case of reading large data set, we can do transformation using lumiT function right after input the data, and the information like se.exprs, beadNum will be removed from the LumiBatch object after transformation.

Value

A LumiBatch object which combines the individual LumiBatch object corresponding to each file

Author(s)

Pan Du

See Also

lumiR

Examples

```
## fileList <- c('file1.csv', 'file2.cvs')
## x.lumi <- lumiR.batch(fileList, sampleInfoFile='sampleInfo.txt')
```

---

lumiR *Read in Illumina expression data*

Description

Read in Illumina expression data. We assume the data was saved in a comma or tab separated text file.

Usage

```
lumiR(fileName, sep = NULL, detectionTh = 0.01, na.rm = TRUE, convertNuID = TRUE
QC = TRUE, columnNameGrepPattern = list(exprs='AVG_SIGNAL', se.exprs='BEAD_STD',
inputAnnotation=TRUE, annotationColumn=c('ACCESSION', 'SYMBOL', 'PROBE_SEQUENCE'
```

Arguments

fileName	fileName of the data file
sep	the separation character used in the text file.
detectionTh	the p-value threshold of determining detectability of the expression. See more details in lumiQ
na.rm	determine whether to remove NA
convertNuID	determine whether convert the probe identifier as nuID
lib.mapping	a Illumina ID mapping package, e.g, lumiHumanIDMapping, used by addNuID2lumi
dec	the character used in the file for decimal points.

`parseColumnName`
: determine whether to parse the column names and retrieve the sample information (Assume the sample information is separated by "_".)

`checkDupId`
: determine whether to check duplicated TargetIDs or ProbeIds. The duplicated ones will be averaged.

`QC`
: determine whether to do quality control assessment after read in the data.

`columnNameGrepPattern`
: the string grep patterns used to determine the slot corresponding columns.

`inputAnnotation`
: determine whether input the annotation information outputted by BeadStudio if exists.

`annotationColumn`
: the column names of the annotation information outputted by BeadStudio

`verbose`
: a boolean to decide whether to print out some messages

`...`
: other parameters used by `read.table` function

Details

The function can automatically determine the separation character if it is Tab or comma. Otherwise, the user should specify the separator manually. If the annotation library is provided, the Illumina Id will be replaced with nuID, which is used as the index Id for the lumi annotation packages. If the annotation library is not provided, it will try to directly convert the probe sequence (if provided in the BeadStudio output file) as nuIDs.

The parameter "columnNameGrepPattern" is designed for some advanced users. It defines the string grep patterns used to determine the slot corresponding columns. For example, for the "exprs" slot in LumiBatch object, it is composed of the columns whose name includes "AVG_SIGNAL". In some cases, the user may not want to read the "detection" and "beadNum" related columns to save memory. The user can set the "detection" and "beadNum" as NA in "columnNameGrepPattern". If the 'se.exprs' is set as NA or the corresponding columns are not available, then lumiR will create a ExpressionSet object instead of LumiBatch object.

The parameter "parseColumnName" is designed to parse the column names and retrieve the sample information. We assume the sample information is separated by "_" and the last element after "_" is the sample label (sample names of the LumiBatch object). If the parsed sample labels are not unique, then the entire string will be used as the sample label. For example: "1881436055_A_STA 27aR" is included in one of the column names of BeadStudio output file. Here, the program will first treat "STA 27aR" as the sample label. If it is not unique across the samples, "1881436055_A_STA 27aR" will be the sample label. If it is still not unique, the program will report warning messages. All the parsed information is kept in the phenoData slot. By default, "parseColumnName" is FALSE. We suggest the users use it only when they know what they are doing.

Current version of lumiR can adaptively read the output of BeadStudio Verson 1 and 3. The format Version 3 made quite a few changes comparing with previous versions. One change is the detection value. It was called detectable when the detection value is close to one for Version 1 format. However, the detection value became a p-value in the Version 3. As a result, the detectionTh is automatically changed based on the version. The detectionTh 0.01 for the Version 3 will be changed as the detectionTh 0.99 for Version 1. Another big change is that Version 3 separately output the control probe (gene) information and a "Samples Table". As a result, the controlData slot in LumiBatch class was added to keep the control probe (gene) information, and a QC slot to keep the quality control information, including the "Sample Table" output by BeadStudio version 3.

The recent version of BeadStudio can also output the annotation information together with the expression data. In the users also want to input the annotation information, they can set the parameter

"inputAnnotation" as TRUE. At the same time, they can also specify which columns to be inputted by setting parameter "annotationColumn". The BeadStudio annotation columns include: SPECIES, TRANSCRIPT, ILMN_GENE, UNIGENE_ID, GI, ACCESSION, SYMBOL, PROBE_ID, ARRAY_ADDRESS_ID, PROBE_TYPE, PROBE_START, PROBE_SEQUENCE, CHROMOSOME, PROBE_CHR_ORIENTATION, PROBE_COORDINATES, DEFINITION, ONTOLOGY_COMPONENT, ONTOLOGY_PROCESS, ONTOLOGY_FUNCTION, SYNONYMS, OBSOLETE_PROBE_ID. As the annotation data is huge, by default, we only input: ACCESSION, SYMBOL, PROBE_START, CHROMOSOME, PROBE_CHR_ORIENTATION, PROBE_COORDINATES, DEFINITION. As some annotation information may be outdated. We recommend using Bioconductor annotation packages to retrieve the annotation information.

Value

return a LumiBatch object

Author(s)

Simon Lin, Pan Du

See Also

LumiBatch, addNuID2lumi

Examples

```
## specify the file name
# fileName <- 'Barnes_gene_profile.txt' # Not Run
## load the data
# x.lumi <- lumiR(fileName)

## load the data with empty detection and beadNum slots
# x.lumi <- lumiR(fileName, columnNameGrepPattern=list(detection=NA, beadNum=NA))
```

---

lumiT *Transfer the Illumina data to stabilize the variance*

Description

Transfer the Illumina data to stabilize the variance.

Usage

```
lumiT(x.lumi, method = c("vst", 'log2', 'cubicRoot'), ifPlot = FALSE, stdCorrect
```

Arguments

x.lumi	LumiBatch object
method	four methods are supported: "vst", "log2", "cubicRoot"
ifPlot	determine whether to plot the intermediate results
stdCorrection	determine transfer the standard error of the mean as the standard deviation, used for 'vst' method.

simpleOutput	determine whether to simplify the output LumiBatch object, which will set the se.exprs, detection and beadNum slots as NULL.
verbose	a boolean to decide whether to print out some messages
...	other parameters used by vst

Details

lumiT is an interface of difference variance stabilizing transformation. See vst for details of VST (Variance Stabilizing Transform) of Illumina data.

The adding of the parameter "stdCorrection" is for the value correction of the STDEV (or STDERR) columns when 'vst' method is selected. The STDEV (or STDERR) columns of the BeadStudio output file is the standard error of the mean of the bead intensities corresponding to the same probe. (Thanks Gordon Smyth kindly provided this information.). As the variance stabilization (see vst function) requires the information of the standard deviation instead of the standard error of the mean, the value correction is required. The corrected value will be x * sqrt(N), where x is the old value (standard error of the mean), N is the number of beads corresponding to the probe.

Value

Return a LumiBatch object with transformed expression values. It also includes the VST transform function and its parameters as attributes: "transformFun", "parameter". See inverseVST for details.

Author(s)

Pan Du, Simon Lin

References

Lin, S.M., Du, P., Kibbe, W.A., (2008) 'Model-based Variance-stabilizing Transformation for Illumina Microarray Data', Nucleic Acids Res. 36, e11

See Also

vst

Examples

```
## load example data
data(example.lumi)

## Do default VST variance stabilizing transform
lumi.T <- lumiT(example.lumi, ifPlot=TRUE)
```

MAplot-methods *MAplot of a ExpressionSet object*

Description

Creating pairwise MAplot of sample intensities in a ExpressionSet object

Usage

```
## S4 method for signature 'ExpressionSet':
MAplot(object, ..., smoothScatter = FALSE, logMode = TRUE, subset = 5000, main =
```

Arguments

object	an ExpressionSet object
...	optional arguments to MAplot.
smoothScatter	whether use smoothScatter function to plot points
logMode	whether plot the data in log2 scale or not
subset	subset of rows used to plot. It can be an index vector, or the length of a random subset
main	title of the plot

Details

To increase the plot efficiency, by default, we only plot RANDOMLY selected subset of points (based on parameter "subset"). If users want to plot all the points, they can set the parameter "subset = NULL". When smoothScatter is set as TRUE, the subsetting will be suppressed because smoothScatter function has good plot efficiency for large number of points.

See Also

LumiBatch-class, MAplot

Examples

```
## load example data
data(example.lumi)

MAplot(example.lumi)

MAplot(example.lumi, smoothScatter=TRUE)
```

monoSmu *Monotonic smooth method*

Description

Fit the monotonic-constraint spline curve

Usage

```
monoSmu(x, y, newX = NULL, nSupport = min(200, length(x)), nKnots = 6, rotate =
```

Arguments

x	a vector represents x values
y	a vector represents y values
newX	the new values to be transformed. If not provided, "x" will be used.
nSupport	downsampled data points
nKnots	parameter used by monoSpline
rotate	determine whether to rotate the axis with 45 degrees in clockwise, i.e., fit the curve in the MA-plot.
ifPlot	determine whether to plot intermediate results
xlab	the xlab of the plot
ylab	the ylab of the plot
...	parameters used by supsmu and plot

Details

function called by lumiN.rsn. The function first fits a monotonic spline between vector x and y, then transforms the vector newX based on the fitted spline. (After transformation the fitted spline is supposed to be a diagonal line, i.e., x=y)

Value

Return the transformed "newX" based on the smoothed curve

Author(s)

Simon Lin, Pan Du

References

Lin, S.M., Du, P., Kibbe, W.A., (2008) 'Model-based Variance-stabilizing Transformation for Illumina Microarray Data', Nucleic Acids Res. 36, e11

See Also

monoSpline

`monoSpline` *Fitting a curve with monotonic spline*

Description

Fitting a curve with monotonic spline

Usage

```
monoSpline(x, y, newX=NULL, nKnots = 6, ifPlot = FALSE)
```

Arguments

`x`	a vector represents x values
`y`	a vector represents y values
`newX`	the new values to be transformed. If not provided, "x" will be used.
`nKnots`	parameter used by function smoothCon in package mgcv
`ifPlot`	determine whether to plot intermediate results

Details

Function internally called by monoSmu

Value

return the transformed "newX" based on the smoothed curve

Author(s)

Simon Lin, Pan Du

See Also

`monoSmu`

`nuID2EntrezID` *Map nuID to Entrez ID*

Description

Map nuID to EntrezID through RefSeq ID based on IDMapping libraries.

Usage

```
nuID2EntrezID(nuID = NULL, lib.mapping, filterTh = c(Strength1 = 95, Uniqueness
```

Arguments

nuID	a vector of nuIDs. If it is NULL, all mappings will be returned.
lib.mapping	the ID mapping library
filterTh	the mapping quality filtering threshold used to filter the ID mapping.
returnAllInfo	determine to return the detailed mapping information or just the matched RefSeq IDs

Details

This function is based on the return of getNuIDMappingInfo function. The mapping from nuID to EntrezID was based on the mapping from nuID to RefSeqID and RefSeqID to EntrezID. It uses mapping quality information to filter out the bad mappings from nuID to RefSeqID. The parameter "filterTh" is obsolete for lumi ID mapping package > version 1.3, which only keeps the perfect mapping. For the old version of ID mapping package (< 1.3), the names of "filterTh" are basically the field names of "nuID_MappingInfo" table, which include 'Strength1', 'Strength2', 'Uniqueness' and 'Total hits'. For the definition of these metrics, please refer to the IDMapping library or see the reference website.

Value

returns the matched Entrez IDs or a data.frame with each row corresponding to an input nuID (when "returnAllInfo" is TRUE).

Author(s)

Warren Kibbe, Pan Du, Simon Lin

References

https://prod.bioinformatics.northwestern.edu/nuID/

See Also

See Also getNuIDMappingInfo

Examples

```
## load example data
data(example.lumi)
if (require(lumiHumanIDMapping)) {
nuIDs <- featureNames(example.lumi)
mappingInfo <- nuID2EntrezID(nuIDs, lib.mapping='lumiHumanIDMapping')
head(mappingInfo)
}
```

`nuID2IlluminaID` *Matching nuIDs to Illumina IDs based on Illumina ID mapping library*

Description

Matching nuIDs to Illumina IDs based on Illumina ID mapping library

Usage

```
nuID2IlluminaID(nuID, lib.mapping=NULL, species = c("Human", "Mouse", "Rat", "Un
```

Arguments

`nuID`	a vector of nuIDs
`lib.mapping`	the ID mapping library. If it is provided, the parameter "species" will be ignored.
`species`	the species of the chip designed for. If users do not know it, it can be set as "Unknown".
`idType`	the Illumina ID type
`chipVersion`	chipVersion information returned by function `getChipInfo`
`...`	other parameters of `getChipInfo`

Details

The parameter "idType" represents different types of Illumina IDs. It returns the entire table when idType = "All". When idType = 'Probe', it returns "ProbeId" or "Probe_Id". When idType = 'Gene', it returns "Target" or "ILMN_Gene" IDs.

This function basically returned the "idMapping" item returned by function `getChipInfo`. If nuID is NULL and chipVersion is provided, it will return all mapping information of the chip.

Value

The mapping information from nuID to Illumina ID. It will be a matrix with each column corresponding to one matched manifest file when parameter "returnAllMatches" is TRUE. In this case, the columns are sorted from the best match to worst.

Author(s)

Pan Du

See Also

`getChipInfo`, `IlluminaID2nuID`

Examples

```
## load example data
data(example.lumi)
nuIDs <- featureNames(example.lumi)
if (require(lumiHumanIDMapping)) {
illuminaID <- nuID2IlluminaID(nuIDs[1:5], lib='lumiHumanIDMapping')
illuminaID
}
```

`nuID2probeID` *Mapping nuID into Illumina ProbeID*

Description

Mapping nuID into Illumina ProbeID.

Usage

```
nuID2probeID(nuID, lib.mapping = "lumiHumanIDMapping", ...)
```

Arguments

`nuID`	a vector of nuID
`lib.mapping`	an Illumina ID mapping library
`...`	other parameters of `nuID2IlluminaID`

Details

The function will call `nuID2IlluminaID` when ID mapping library were provided.

Value

see function `nuID2IlluminaID`

Author(s)

Pan Du

References

Du, P., Kibbe, W.A. and Lin, S.M., "nuID: A universal naming schema of oligonucleotides for Illumina, Affymetrix, and other microarrays", Biology Direct 2007, 2:16 (31May2007).

See Also

`probeID2nuID`, `nuID2IlluminaID`

Examples

```
if (require(lumiHumanIDMapping)) {
    nuID2probeID("B2J6WGhV.RevOJYff4", lib.mapping = "lumiHumanIDMapping")
}
```

`nuID2RefSeqID` *Map nuID to RefSeq ID*

Description

Map nuID to RefSeq ID based on IDMapping libraries.

Usage

```
nuID2RefSeqID(nuID = NULL, lib.mapping, filterTh = c(Strength1 = 95, Uniqueness
```

Arguments

`nuID`	a vector of nuIDs. If it is NULL, all mappings will be returned.
`lib.mapping`	the ID mapping library
`filterTh`	the mapping quality filtering threshold used to filter the ID mapping. Obsolete for lumi ID mapping package > version 1.3!
`returnAllInfo`	determine to return the detailed mapping information or just the matched RefSeq IDs

Details

This function is based on the return of `getNuIDMappingInfo` function. It uses mapping quality information to filter out the bad mappings. The parameter "filterTh" is obsolete for lumi ID mapping package > version 1.3, which only keeps the perfect mapping. For the old version of ID mapping package (< 1.3), the names of "filterTh" are basically the field names of "nuID_MappingInfo" table, which include 'Strength1', 'Strength2', 'Uniqueness' and 'Total hits'. For the definition of these metrics, please refer to the IDMapping library or see the reference website.

Value

returns the matched RefSeq IDs or a data.frame with each row corresponding to an input nuID (when "returnAllInfo" is TRUE).

Author(s)

Warren Kibbe, Pan Du, Simon Lin

References

https://prod.bioinformatics.northwestern.edu/nuID/

See Also

See Also `getNuIDMappingInfo`

Examples

```
## load example data
data(example.lumi)
if (require(lumiHumanIDMapping)) {
nuIDs <- featureNames(example.lumi)
mappingInfo <- nuID2RefSeqID(nuIDs, lib.mapping='lumiHumanIDMapping')
head(mappingInfo)
}
```

---

nuID2targetID *Mapping nuID into Illumina TargetID*

Description

Mapping nuID into Illumina TargetID or GeneID.

Usage

```
nuID2targetID(nuID, lib.mapping = "lumiHumanIDMapping", ...)
```

Arguments

nuID	a vector of nuID
lib.mapping	an Illumina ID mapping library
...	other parameters of nuID2IlluminaID

Details

The function will call nuID2IlluminaID when ID mapping library were provided.

Value

see function nuID2IlluminaID

Author(s)

Pan Du

References

Du, P., Kibbe, W.A. and Lin, S.M., "nuID: A universal naming schema of oligonucleotides for Illumina, Affymetrix, and other microarrays", Biology Direct 2007, 2:16 (31May2007).

See Also

targetID2nuID, nuID2IlluminaID

Examples

```
if (require(lumiHumanIDMapping)) {
    nuID2targetID("B2J6WGhV.RevOJYff4", lib.mapping = "lumiHumanIDMapping")
}
```

pairs-methods *Pair plot of an ExpressionSet object*

Description

Creating pairs plot of sample intensities in an ExpressionSet object

Usage

```
## S4 method for signature 'ExpressionSet':
pairs(x, ..., smoothScatter = FALSE, logMode = TRUE, subset = 5000, fold=2, main
```

Arguments

x	a ExpressionSet object
...	optional arguments to pairs.
smoothScatter	whether use smoothScatter function to plot points
logMode	whether plot the data in log2 scale
subset	subset of rows used to plot. It can be an index vector, or the length of a random subset
fold	The fold-change threshold used to estimate the number of probes having high fold-changes
main	title of the plot

Details

To increase the plot efficiency, by default, we only plot RANDOMLY selected subset of points (based on parameter "subset"). If users want to plot all the points, they can set the parameter "subset = NULL". When smoothScatter is set as TRUE, the subsetting will be suppressed because smoothScatter function has good plot efficiency for large number of points.

See Also

LumiBatch-class, pairs

Examples

```
## load example data
data(example.lumi)

pairs(example.lumi)

pairs(example.lumi, smoothScatter=TRUE)
```

`plotControlData` *Plot the mean expression (with standard deviation bar) of different type of control probes*

Description

Plot the mean expression (with standard deviation bar) of different type of control probes. Multiple control types can be plotted in a single plot. The available control types can be get by running getControlType(controlData).

Usage

```
plotControlData(controlData, type = NULL, slideIndex = NULL, logMode = FALSE, ne
```

Arguments

`controlData`	a LumiBatch object including control data or a control data data.frame
`type`	the control probe type (case insensitive), which can be get by running getControlType(controlData)
`slideIndex`	the slide index or ID corresponding to each sample
`logMode`	whether show the data in log2 scale
`new`	whether refresh the new plot or add it on the old one
`...`	other parameters used by default plot function

Details

When multiple control types are selected, they will be plotted in a two-column plot.

Value

plot the picture and return TRUE if everything is OK

Author(s)

Pan Du

See Also

`addControlData2lumi`

Examples

```
controlFile <- system.file('doc', 'Control_Probe_Profile.txt', package='lumi')
controlData <- getControlData(controlFile)
getControlType(controlData)
plotControlData(controlData, type='NEGATIVE')
```

plotHousekeepingGene *Plot the housekeeping gene expression profile*

Description

Plot the housekeeping gene expression profile

Usage

```
plotHousekeepingGene(controlData, lib = NULL, slideIndex = NULL, addLegend = TRU
```

Arguments

controlData	a LumiBatch object including control data or a control data data.frame
lib	the annotation library (for retrieving the gene name)
slideIndex	the slide index or ID corresponding to each sample
addLegend	whether add legend or not
logMode	whether show the data in log2 scale
...	other parameters used by default matplot function

Value

plot the picture and return TRUE if everything is OK

Author(s)

Pan Du

See Also

addControlData2lumi, plotControlData

Examples

```
controlFile <- system.file('doc', 'Control_Probe_Profile.txt', package='lumi')
controlData <- getControlData(controlFile)
plotHousekeepingGene(controlData)
```

plot-methods *Plot of a LumiBatch object*

Description

Creating quality control plots of a LumiBatch object

Usage

```
## S4 method for signature 'LumiBatch,missing':
plot(x, what = c("density", "boxplot", "pair", "MAplot", "sampleRelation", "outl
```

Arguments

x	a LumiBatch object returned by lumiQ
what	one of the six kinds of QC plots
main	the title of the QC plot
...	additional parameters for the corresponding QC plots

Details

The parameter "what" of plot function controls the type of QC plots, which includes:

- **density**: the density plot of the chips, see hist-methods
- **boxplot**: box plot of the chip intensities, see boxplot-methods
- **pair**: the correlation among chips, plot as a hierarchical tree, see pairs-methods
- **MAplot**: the MAplot between chips, see MAplot-methods
- **sampleRelation**: plot the sample relations. See plotSampleRelation
- **outlier**: detect the outliers based on the sample distance to the center. See detectOutlier
- **cv**: the density plot of the coefficients of variance of the chips. See estimateLumiCV

See Also

LumiBatch-class, hist-methods, boxplot-methods, MAplot-methods, pairs-methods, plotSampleRelation, estimateLumiCV, detectOutlier

Examples

```
## load example data
data(example.lumi)

## Quality control estimation
lumi.Q <- lumiQ(example.lumi)

## summary
summary(lumi.Q)

## plot the density
plot(lumi.Q, what='density')
```

```
## plot the pairwise sample correlation
plot(lumi.Q, what='pair')

## plot the pairwise MAplot
plot(lumi.Q, what='MAplot')

## sample relations
plot(lumi.Q, what='sampleRelation', method='mds', color=c('100US', '95US:5P', '100US', '9

## detect outlier based on the distance to the mean profile
plot(lumi.Q, what='outlier')

## Density plot of coefficient of variance
plot(lumi.Q, what='cv')
```

`plotSampleRelation` *visualize the sample relations*

Description

plot the sample relations based on MDS or hierarchical clustering

Usage

```
plotSampleRelation(x, selProbe = NULL, cv.Th = 0.1, standardize = TRUE, method =
```

Arguments

`x`	a LumiBatch object, ExpressionSet object or a matrix with each column corresponding to a sample
`selProbe`	the selected probes used to determine the sample relations. If not provide, all the probes will be used.
`cv.Th`	the threshold of the coefficient of variance of probes used to select probes to estimate sample relations
`standardize`	standardize the expression profiles or not
`method`	"MDS" or "hierarchical clustering"
`dimension`	the principle components to visualize the MDS plot
`color`	the color for each sample during plot. Only support the "mds" method
`...`	Other parameters used by plot function.

Details

Estimate the sample relations based on selected probes (based on large coefficient of variance (mean / standard variance)). Two methods can be used: MDS (Multi-Dimensional Scaling) or hierarchical clustering methods.

Value

Plot the results or return the distance matrix.

Author(s)

Pan Du

See Also

lumiQ, LumiBatch, , plot.LumiBatch

Examples

```
## load example data
data(example.lumi)

## plot the sample relations with MDS
## the color of sample is automatically set based on the sample type
plotSampleRelation(example.lumi, col=c('100US', '95US:5P', '100US', '95US:5P'))

## plot the sample relations with hierarchical clustering
plotSampleRelation(example.lumi, method='cluster')
```

---

plotStringencyGene *plot the Stringency related control probe profiles*

Description

Plot the Stringency related control probe (Low-Stringency, Medium-Stringency and High-Stringency) profiles. Using getControlType function to view available stringency types.

Usage

```
plotStringencyGene(controlData, lib = NULL, slideIndex = NULL, addLegend = TRUE,
```

Arguments

controlData	a LumiBatch object including control data or a control data data.frame
lib	the annotation library (for retrieving the gene name)
slideIndex	the slide index or ID corresponding to each sample
addLegend	whether add legend or not
logMode	whether show the data in log2 scale
...	other parameters used by default matplot function

Value

plot the picture and return TRUE if everything is OK

Author(s)

Pan Du

See Also

addControlData2lumi, plotControlData

Examples

```
controlFile <- system.file('doc', 'Control_Probe_Profile.txt', package='lumi')
controlData <- getControlData(controlFile)
plotStringencyGene(controlData)
```

---

plotVST *plot the VST (Variance Stabilizing Transform) function*

Description

plot the VST (Variance Stabilizing Transform) function of VST transformed LumiBatch object or parameters of VST function.

Usage

```
plotVST(x, transFun = NULL, plotRange = NULL, addLegend = TRUE, ...)
```

Arguments

x	a LumiBatch object after lumiT transform, or a matrix or data.frame with VST parameter
transFun	a character vector of transformation function (asinh or log2)
plotRange	the plot range of untransformed data
addLegend	add legend or not
...	other parameter used by plot function

Value

invisibly return the untransformed and transformed values.

Author(s)

Pan Du

See Also

vst

Examples

```
## load example data
data(example.lumi)

## Do default VST variance stabilizing transform
lumi.T <- lumiT(example.lumi, ifPlot=TRUE)

## plot the transform function
plotVST(lumi.T)
```

`probeID2nuID` *Mapping Illumina ProbeID as nuID*

Description

Mapping Illumina ProbeID as nuID.

Usage

```
probeID2nuID(probeID, lib.mapping = "lumiHumanIDMapping", ...)
```

Arguments

`probeID`	a vector of Illumina ProbeID
`lib.mapping`	an Illumina ID mapping library
`...`	other parameters of `IlluminaID2nuID`

Details

The function will call `IlluminaID2nuID` when ID mapping library were provided.

Value

see function `IlluminaID2nuID`

Author(s)

Pan Du

References

Du, P., Kibbe, W.A. and Lin, S.M., "nuID: A universal naming schema of oligonucleotides for Illumina, Affymetrix, and other microarrays", Biology Direct 2007, 2:16 (31May2007).

See Also

`nuID2probeID`, `IlluminaID2nuID`

Examples

```
if (require(lumiHumanIDMapping)) {
    probeID2nuID('0001240020', lib='lumiHumanIDMapping')
}
```

`produceGEOPlatformFile` *Produce GEO Platform Submission File in SOFT format*

Description

Produce GEO Sample Submission File in SOFT format based on the provided LumiBatch object and Illumina ID Mapping library

Usage

```
produceGEOPlatformFile(x.lumi, lib.mapping = NULL, nuIDMode = TRUE, includeAllCh
```

Arguments

`x.lumi`	The LumiBatch object keeping all probes
`lib.mapping`	The Illumina ID Mapping library, e.g., "lumiHumanIDMapping"
`nuIDMode`	Determine whether producing the platform indexed by nuID
`includeAllChipProbe`	Determine whether including all probes in the Manifest file or just the probes used in the x.lumi object
`fileName`	Filename of the GEO Platform File name

Details

The function produces the GEO platform submission file based on the chip information kept in the Illumina ID Mapping library (specified by lib.mapping parameter). The determination of chip type will be automatically done by selecting the best matching of the probe IDs with individual chips.

Value

Save the result as a text file in SOFT platform submission format.

Author(s)

Pan Du

References

http://www.ncbi.nlm.nih.gov/projects/geo/info/soft2.html

See Also

`produceGEOSubmissionFile`

Examples

```
# data(example.lumi)
# produceGEOPlatformFile(example.lumi, lib.mapping='lumiHumanIDMapping')
```

`produceGEOSampleInfoTemplate`
Produce the template of GEO sample information

Description

Produce the template of GEO sample information, which is used for function `produceGEOSubmissionFile`.

Usage

```
produceGEOSampleInfoTemplate(lumiNormalized, lib.mapping = NULL, fileName = "GEO
```

Arguments

`lumiNormalized`	The normalized data (LumiBatch object)
`lib.mapping`	The Illumina ID Mapping library, e.g., "lumiHumanIDMapping"
`fileName`	The file name of Tab separated sample information file

Details

This function just produces a template of sample information with some default fillings. Users need to fill in the detailed sample descriptions, especially the Sample_title, Sample_description and some protocols. No blank fields are allowed. Function `produceGEOSubmissionFile` will produce the file GEO submission file based on this sample information. The users should not use "\#" in the description as it is a reserved character.

Value

Save the result as a Tab separated text file or return a data.frame if the fileName is NULL.

Author(s)

Pan Du

References

http://www.ncbi.nlm.nih.gov/projects/geo/info/soft2.html

See Also

`produceGEOSubmissionFile`

produceGEOSubmissionFile *Produce GEO Sample Submission File in SOFT format*

Description

Produce GEO Sample Submission File in the SOFT format based on the provided LumiBatch object and sample information

Usage

```
produceGEOSubmissionFile(lumiNormalized, lumiRaw, lib.mapping = NULL, idType = '
```

Arguments

lumiNormalized	The normalized data (LumiBatch object)
lumiRaw	The raw data (LumiBatch object), e.g., returned by lumiR
lib.mapping	The Illumina ID Mapping library, e.g., "lumiHumanIDMapping"
idType	the idType parameter of function nuID2IlluminaID
sampleInfo	The sample information filename or data.frame, which is returned by produceGEOSampleInfoTe
fileName	The file name of GEO Submission file
supplementaryRdata	determine whether produce the Rdata supplement data, which include both lumiNormalized and lumiRaw R objects.
...	other parameters used by function nuID2IlluminaID

Details

The function produces the GEO sample submission file including both normalized and raw data information in the SOFT format. The sample information should be provided by the user as a data.frame or Tab separated text file following the format of the template, which can be produced by function produceGEOSampleInfoTemplate. Users need to fill in the detailed sample descriptions in the template, especially the Sample_title, Sample_description and some protocols. Users are also suggested to fill in the "Sample_platform_id" by checking information of the GEO Illumina platform.

When the parameter "supplementaryRdata" is TRUE, the R objects, lumiNormalized, lumiRaw and sampleInfo, will be saved in a file named 'supplementaryData.Rdata'.

Value

Save the result as a text file in SOFT sample submission format. The supplementary Rdata will be saved in a file 'supplementaryData.Rdata'.

Author(s)

Pan Du

References

http://www.ncbi.nlm.nih.gov/projects/geo/info/soft2.html

See Also

produceGEOSampleInfoTemplate, produceGEOPlatformFile

Examples

```
## Not run
## Produce the sample information template
# produceGEOSampleInfoTemplate(lumiNormalized, lib.mapping = NULL, fileName = "GEOsampleI
## After editing the 'GEOsampleInfo.txt' by filling in sample information
# produceGEOSubmissionFile(lumiNormalized, lumiRaw, lib='lumiHumanIDMapping', sampleInfo=
```

---

rankinvariant *Rank Invariant Normalization*

Description

This function basically adjusts the samples to the same background level and then optionally scales to the same foreground level.

Usage

```
rankinvariant(x.lumi, targetArray = NULL, rrc = .05, lowRank = seq(.5, .25, -.05
```

Arguments

x.lumi	an ExpressionSet inherited object or a data matrix with columns as samples and rows as genes
targetArray	A target chip is the model for other chips to normalize. It can be a column index, a vector or a LumiBatch object with one sample.
rrc	The relative rank change allowed for a gene to be selected as rank invariant
lowRank	A vector with, in decreasing order, the minimum ranks where candidate genes can be selected as rank invariant
highRank	The maximum rank where candidate genes can be selected as rank invariant
minSize	Fraction of genes required to be selected as rank invariant
maxit	Maximum number of iterations for rlm to reach convergence

Details

Rank invariant normalization uses a set of genes that are rank invariant between a given sample and a target sample. The target sample can be predefined by setting the targetArray argument. If targetArray is NULL the average expression of all samples will be the target. Rank invariant genes are found for each sample seperately by calculation the relative rank change for each gene. Furthermore, only genes with ranks between the lowRank and highRank are considered. If the number of probes is less than minSize multiplies by the number of genes the next lowRank value tried. If no rank invariant set can be found an error is thrown.

The default settings of this function are the same as used Genomstudio (Illumina). The results produced by this method are similar, but not identical to Genomestudio.

Value

Return an object with expression values normalized. The class of the return object is the same as the input object x.lumi.

Author(s)

Arno Velds (contact: a.velds (at) nki.nl)

See Also

lumiN

`rsn` *Robust Spline Normalization between chips*

Description

Robust spline normalization (monotonic curves) between chips

Usage

```
rsn(x.lumi, targetArray = NULL, excludeFold = 2, span = 0.03, ifPlot = FALSE, ..
```

Arguments

`x.lumi`	an ExpressionSet inherited object or a data matrix with columns as samples and rows as genes
`targetArray`	A target chip is the model for other chips to normalize. It can be a column index, a vector or a LumiBatch object with one sample.
`excludeFold`	exclude the genes with fold change larger than "excludeFold" during fitting the curve in normalization
`span`	the span parameter used by monoSmu
`ifPlot`	determine whether to plot intermediate results
`...`	other parameters used by monoSmu

Details

The robust spline normalization (RSN) algorithm combines the features of quantile and loess normalization. It is designed to normalize the variance-stabilized data. The function will check whether the data is variance stabilized (vst or log2 transform), if not, it will automatically run lumiT before run rsn. For details of the algorithm, please see the reference.

The targetArray can be a column index, a vector or a LumiBatch object with one sample, which corresponds to an external sample to be normalized with. This is very useful for handling large data set or normalizing the data set with a common reference (targetArray).

Value

Return an object with expression values normalized. The class of the return object is the same as the input object x.lumi. If it is a LumiBatch object, it also includes the VST transform function and its parameters as attributes: "transformFun", "parameter". See inverseVST for details.

Author(s)

Pan Du, Simon Lin

See Also

lumiN, monoSmu

`seq2id` *Transfer a nucleotide sequence as a nuID*

Description

The nuID (nucleotide universal identifier) is uniquely corresponding to probe sequence. The nuID is also self-identification and error checking

Usage

```
seq2id(seq)
```

Arguments

`seq`	a nucleotide sequence composed of A, C, G, T (U).

Details

The nuID is a exact mapping of nucleotide sequence based on Base64 encoding scheme. A character set A-Z, a-z, 0-9, "_" and "." is used to represent to the base-64 numbers of 0-63. The first character of nuID is a checking code, which provide information of both the number of padded "A"s at the nucleotide sequence and error checking. Please refer to reference for more details.

Value

A string represents nuID

Author(s)

Pan Du

References

Du, P., Kibbe, W.A. and Lin, S.M., "nuID: A universal naming schema of oligonucleotides for Illumina, Affymetrix, and other microarrays", Biology Direct 2007, 2:16 (31May2007).

See Also

id2seq

Examples

```
seq <- 'ACGTAAATTTCAGTTTAAAACCCCCCG'
id <- seq2id(seq)
id
id2seq(id)
```

ssn *Simple Scaling Normalization*

Description

This function basically adjusts the samples to the same background level and then optionally scales to the same foreground level.

Usage

```
ssn(x.lumi, targetArray = NULL, scaling = TRUE, bgMethod=c('density', 'mean', 'm
```

Arguments

x.lumi	an ExpressionSet inherited object or a data matrix with columns as samples and rows as genes
targetArray	A target chip is the model for other chips to normalize. It can be a column index, a vector or a LumiBatch object with one sample.
scaling	determine whether do scaling or just background shift
bgMethod	optional methods of determining the background level
fgMethod	optional methods of determining the foreground level
...	other parameters used by density function

Details

This function basically adjusts the samples to the same background level and then optionally scales to the same foreground level. The adjustment is based on the raw scale data (For the transformed data, it still estimates the parameters in the raw scale by inverse transformation.).

Comparing with other normalization methods, like quantile and curve-fitting methods, SSN is a more conservative method. The only assumption is that each sample has the same background levels and the same scale (if do scaling). There are three methods ('density', 'mean' and 'median') for background estimation. If bgMethod is 'none', then the background level will be set as 0, i.e., no background adjustment. For the 'density' bgMethod, it estimates the background based on the mode of probe intensities based on the assumption that the background level intensity is the most frequent value across all the probes in the chip. For the foreground level estimation, it also provides three methods ('mean', 'density', 'median'). For the 'density' fgMethod, it assumes the background probe levels are symmetrically distributed. Then we estimate the foreground levels by taking the intensity mean of all other probes except from the background probes. For the 'mean' and 'median' methods (for both bgMethod and fgMethod), it basically estimates the level based on the mean or median of all probes of the sample. If the fgMethod is the same as bgMethod (except 'density' method), no scaling will be performed.

Value

Return an object with expression values normalized. The class of the return object is the same as the input object x.lumi.

Author(s)

Pan Du, Simon Lin

See Also

lumiN

`targetID2nuID` *Mapping Illumina TargetID (GeneID) into nuID*

Description

Mapping Illumina TargetID (GeneID) into nuID.

Usage

```
targetID2nuID(targetID, lib.mapping = "lumiHumanIDMapping", ...)
```

Arguments

`targetID`	a vector of Illumina TargetID (GeneID)
`lib.mapping`	an Illumina ID mapping library
`...`	other parameters of `IlluminaID2nuID`

Details

The function will call `IlluminaID2nuID` when ID mapping library were provided.

Value

see function `IlluminaID2nuID`

Author(s)

Pan Du

References

Du, P., Kibbe, W.A. and Lin, S.M., "nuID: A universal naming schema of oligonucleotides for Illumina, Affymetrix, and other microarrays", Biology Direct 2007, 2:16 (31May2007).

See Also

`nuID2targetID`, `IlluminaID2nuID`

Examples

```
if (require(lumiHumanIDMapping)) {
    targetID2nuID('GI_21389350-S', lib='lumiHumanIDMapping')
}
```

vst *Variance Stabilizing Transformation*

Description

Stabilizing the expression variance based on the bead level expression variance and mean relations

Usage

```
vst(u, std, nSupport = min(length(u), 500), backgroundStd=NULL, fitMethod = c('l
```

Arguments

u	mean expression of the beads with same sequence
std	expression standard deviation of the beads with same sequence
nSupport	the number of down-sampling to speed processing
backgroundStd	pre-estimated background standard deviation level
fitMethod	methods of fitting the relations between expression variance and mean relations
lowCutoff	cutoff ratio to determine the low expression range. Do not change this until you now what you are doing.
ifPlot	plot intermediate results or not

Details

The variance-stabilizing transformation (VST) takes the advantage of larger number of technical replicates available on the Illumina microarray. It models the mean-variance relationship of the within-array technical replicates at the bead level of Illumina microarray. An arcsinh transform is then applied to stabilize the variance. See reference for more details.

For the methods of fitting the relations between expression variance and mean relations, the 'linear' method is more robust and provides detailed parameters for inverseVST.

Value

Return the transformed (variance stabilized) expression values.

Author(s)

Pan Du, Simon Lin

References

Lin, S.M., Du, P., Kibbe, W.A., "Model-based Variance-stabilizing Transformation for Illumina Mi-croarray Data", submitted

See Also

lumiT, inverseVST

Examples

```
## load example data
data(example.lumi)

## get the gene expression mean for one chip
u <- exprs(example.lumi)[,1]
## get the gene standard deviation for one chip
std <- se.exprs(example.lumi)[,1]

## do variance stabilizing transform
transformedU <- vst(u, std)

## do variance stabilizing transform with plotting intermediate result
transformedU <- vst(u, std, ifPlot=TRUE)
```

Index