Package 'virtualArray'

September 24, 2012

Type Package

Title Build virtual array from different microarray platforms

Version 1.0.0

Date 2012-03-08

Author Andreas Heider

Maintainer Andreas Heider <aheider@trm.uni-leipzig.de>

Description This package permits the user to combine raw data of different microarray platforms into one virtual array. It consists of several functions that act subsequently in a semi-automatic way. Doing as much of the data combination and letting the user concentrate on analysing the resulting virtual array.

Depends R (>= 2.3.0), methods, plyr, preprocessCore

Imports affy, affyPLM, AnnotationDbi, Biobase, gcrma, GEOquery,graphics, methods, reshape2, stats, utils

Suggests affydata, plier, limma, lumi, org.Hs.eg.db

Enhances multicore

License GPL-3

biocViews
Microarray, OneChannel, DataImport, Preprocessing,Bioinformatics, MultipleComparisons

LazyLoad yes

R topics documented:

virtualArray-package 2
normalize.ExpressionSet.gq 3
normalize.ExpressionSet.mc 4
normalize.ExpressionSet.mrs 5
normalize.ExpressionSet.nordi 6
normalize.ExpressionSet.qd 7
virtualArrayBuildExprs 8
virtualArrayBuildfData 9

virtualArrayBuildSampleInfo 10
virtualArrayComBat 11
virtualArrayCompile 13
virtualArrayDirs 14
virtualArrayExpressionSets 15
virtualArrayHclust 16
virtualArrayLoadRaw 17
virtualArrayMergeRecurse 18

Index **20**

`virtualArray-package` *Combine raw data of several ExpressionSets into a "virtual array".*

Description

Using this package one can create a new ExpressionSet object, that holds both phenoData and expression values of different microarray platforms.

Details

Package:	virtualArray
Type:	Package
Version:	0.99.9
Date:	2012-03-08
License:	GPL V3
LazyLoad:	yes

Current tools for the analysis of microarray data only allow the comparison of datasets generated on the same platform and chip generation, and restrict meta-analysis studies to the evaluation of study results from different groups, rather than the direct comparison of the available raw datasets. With the help of the virtualArray package it becomes easy to combine raw expression data of different microarray platforms into one "virtual array". Thus, the user may compare his own data to other datasets including public sources, regardless of the platform and chip generation used, or perform meta-analysis directly based on available raw datasets. The package generates a combined virtual array as a "ExpressionSet" object from different datasets by matching raw data entries based on probe, transcript, gene or protein identifiers. Redundancies, gaps, and batch effects are removed before proceeding with data analysis.

virtualArray consists of several subsequent functions, requiring minimal user input. Briefly, (1) raw data are loaded into R, (2) probe sets for each platform are annotated, (3) genes, proteins or transcripts common to all platforms are matched, including checks for redundancy or missing values, data are (4) compiled into a new "virtual array", (5) normalized and (6) subjected to batch effect removal using empirical Bayes methods [1], or other batch effect removal methods. Six of which have been implemented into this package. The generated "virtual array" can than be directly analyzed in R/Bioconducor or exported for use in other suitable software (e.g. MeV).

There are essentially four modes of operation:

Firstly, the "virtualArrayCompile" function can integrate the major (but not all) human microarray platforms in a default mode requiring only minimal user input.

The second way is built around the "virtualArray.ExpressionSet" function. This approach allows to integrate any kind of raw expression data that can be loaded into an ExpressionSet object in R/BioC. The downside of this mode is that the user will have to deal with details such as log2-transformations, 16 bit - 20 bit transformations, assignment of correct annotations, etc.

Finally, each of these two approaches can be used in a supervised or non-supervised mode. The non-supervised mode uses empirical Bayes methods (implemented through "ComBat.R", [1]) to adjust for batch effects between the single datasets. In the supervised mode the user assigns additional covariates next to the batch assignment. This means e.g. "treated" and "untreated", or "non-differentiated" and "differentiated". Note that this information has to be valid, as it impacts the results you will get.

Last but not least it is possible to use the package to integrate data without batch effect removal. This way you can use other methods of batch effect removal than the empirical Bayes methods approach.

The combined data is presented as a regular Bioconductor "ExpressionSet" object, which permits using all of R/Bioconductor's power on the dataset as a whole.

To load the package type:

> library(virtualArray)

Author(s)

Andreas Heider

Maintainer: Andreas Heider <aheider@trm.uni-leipzig.de>

See Also

virtualArray.ExpressionSet, virtualArrayCompile

Examples

```
## Please see the vignette for a comprehensive example
```

---

normalize.ExpressionSet.gq

gene quantiles normalization

Description

Performs gene quantile normalization for cross-platform adjustments of microarray data. It is a modification of the median rank scores normalization.

Usage

```
normalize.ExpressionSet.gq(ExpressionSet = NULL, Batch = NULL, ...)
```

Arguments

ExpressionSet	An ExpressionSet object; if it contains a "Batch" column in its pData slot, this information can be used instead of the "Batch" parameter.
Batch	A character or numeric vector containing batch contribution of the samples. The order of the items should be the same as the samples (column) in the supplied ExpressionSet. Defining this parameter overrides the information of the pData slot.
...	Can be used to pass on parameters to underlying functions.

Details

Please see the references for details.

Value

An ExpressionSet object with adjusted expression matrix.

Author(s)

Andreas Heider

References

Xia XQ, McClelland M, Porwollik S, Song W, Cong X, Wang Y: WebArrayDB: cross-platform microarray data analysis and public data repository. Bioinformatics 2009, 25:2425-2429.

See Also

virtualArrayExpressionSets

Examples

```
# library(affydata)
# data(Dilution)
# Dilution <- rma(Dilution)
# normalize.ExpressionSet.gq(Dilution,Batch=c("A","B","A","B"))
```

---

normalize.ExpressionSet.mc

mean centering normalization

Description

Performs mean centering on the supplied ExpressionSet object.

Usage

```
normalize.ExpressionSet.mc(ExpressionSet = NULL, Batch = NULL, ...)
```

Arguments

ExpressionSet	An ExpressionSet object; if it contains a "Batch" column in its pData slot, this information can be used instead of the "Batch" parameter.
Batch	A character or numeric vector containing batch contribution of the samples. The order of the items should be the same as the samples (column) in the supplied ExpressionSet. Defining this parameter overrides the information of the pData slot.
...	Can be used to pass on parameters to underlying functions.

Details

Please see the references for details.

Value

An ExpressionSet object with adjusted expression matrix.

Author(s)

Andreas Heider

References

Tibshirani R, Hastie T, Narasimhan B, Chu G: Diagnosis of multiple cancer types by shrunken centroids of gene expression. Proceedings of the National Academy of Sciences of the United States of America 2002, 99:6567-6572.

See Also

virtualArrayExpressionSets

Examples

```
# library(affydata)
# data(Dilution)
# Dilution <- rma(Dilution)
# normalize.ExpressionSet.mc(Dilution,Batch=c("A","B","A","B"))
```

---

normalize.ExpressionSet.mrs

median rank scores normalization

Description

The median rank scores algorithm is a modification of the quantile normalization for multi-platform microarray data.

Usage

```
normalize.ExpressionSet.mrs(ExpressionSet = NULL, Batch = NULL, ...)
```

Arguments

ExpressionSet	An ExpressionSet object; if it contains a "Batch" column in its pData slot, this information can be used instead of the "Batch" parameter.
Batch	A character or numeric vector containing batch contribution of the samples. The order of the items should be the same as the samples (column) in the supplied ExpressionSet. Defining this parameter overrides the information of the pData slot.
...	Can be used to pass on parameters to underlying functions.

Details

Please see the references for details.

Value

An ExpressionSet object with adjusted expression matrix.

Author(s)

Andreas Heider

References

Warnat P, Eils R, Brors B: Cross-platform analysis of cancer microarray data improves gene expression based classification of phenotypes. BMC bioinformatics 2005, 6:265.

See Also

virtualArrayExpressionSets

Examples

```
# library(affydata)
# data(Dilution)
# Dilution <- rma(Dilution)
# normalize.ExpressionSet.mrs(Dilution,Batch=c("A","B","A","B"))
```

---

normalize.ExpressionSet.nordi

normal discretization normalization

Description

Performs a discretization of continous expression levels to discrete steps.

Usage

```
normalize.ExpressionSet.nordi(ExpressionSet = NULL, pvalue = 0.01, alpha = 0.05)
```

Arguments

ExpressionSet	An ExpressionSet object
pvalue	The pvalue determines the sensitivity for detecting outliers in each column of the gene expression matrix.
alpha	The alpha value determines the size of the tails of the normal distributions considered to be over or under expressed.

Details

Please see the references for details.

Value

An ExpressionSet object with adjusted expression matrix.

Author(s)

Andreas Heider

References

"GenMiner: Mining informative association rules from genomic data." Ricardo Martinez, Claude Pasquier and Nicolas Pasquier, Proceedings of the IEEE BIBM international conference on Bioinformatics and Biomedecine, pages 15-22, IEEE Computer Society, 2007.

Martinez R, Pasquier N, Pasquier C: GenMiner: mining non-redundant association rules from integrated gene expression data and annotations. Bioinformatics 2008, 24:2643-2644.

See Also

virtualArrayExpressionSets

Examples

```
# library(affydata)
# data(Dilution)
# Dilution <- rma(Dilution)
# normalize.ExpressionSet.nordi(Dilution)
```

---

normalize.ExpressionSet.qd

quantile discretization normalization

Description

Performs a discretization of continous expression levels to discrete steps.

Usage

```
normalize.ExpressionSet.qd(ExpressionSet = NULL, nbin = 8, ...)
```

Arguments

ExpressionSet	An ExpressionSet object.
nbin	Numeric vector of length 1; indicates the number of binnings for the discretization.
...	Can be used to pass on parameters to underlying functions.

Details

Please see the references for details.

Value

An ExpressionSet object with adjusted expression matrix.

Author(s)

Andreas Heider

References

Warnat P, Eils R, Brors B: Cross-platform analysis of cancer microarray data improves gene expression based classification of phenotypes. BMC bioinformatics 2005, 6:265.

See Also

virtualArrayExpressionSets

Examples

```
# library(affydata)
# data(Dilution)
# Dilution <- rma(Dilution)
# normalize.ExpressionSet.qd(Dilution)
```

---

virtualArrayBuildExprs

Format ExpressionSets as data.frames with identifiers in first column

---

Description

This function takes ExpressionSets as input and formats the expression matrix into a data.frame whose identifiers are put into the first column

Usage

```
virtualArrayBuildExprs(x)
```

Arguments

x	An ExpressionSet

Details

This function is normally only called by "virtualArray.ExpressionSet". It can be used, however, to extract the "exprs" slot of an ExpressionSet as a data.frame. At the same time the identifiers (rownames) are saved in column "1" of the data.frame.

Value

The value returned is a data.frame which is based on the expression matrix of the input ExpressionSet.

Author(s)

Andreas Heider

See Also

virtualArray-package, virtualArray.ExpressionSet, virtualArrayCompile

Examples

```
## first we need to load dummy data
library(affydata)
data(Dilution)
## we apply RMA to get an ExpressionSet
Dilution <- rma(Dilution,normalize=FALSE)
## now we derive a data.frame from the expression values
new_exprs_data.frame <- virtualArrayBuildExprs(Dilution)
head(new_exprs_data.frame,n=10)
```

---

virtualArrayBuildfData

Collapses expression values according to given identifiers

Description

Expression values are collapsed according to given identifiers, e.g. gene symbols. In the same run, a data.frame with 2 columns to fill the "fData" slot of the ExpressionSet is build. The data is pulled from the Bioconductor annotation package defined in the "annotation" slot of the ExpressionSet.

Usage

```
virtualArrayBuildfData(x, identifier = "SYMBOL", collapse_fun = median)
```

Arguments

x	Name of an ExpressionSet as a character vector
identifier	Wich identifier to pull from the annotation (default="SYMBOL"). Theoretically it can be anything that is supported by the annotation package. Practically it is limited to identifiers giving a 1 to 1 mapping, such as ENTREZID, SYMBOL, GENENAME, UNIPROT, UNIGENE, etc. Identifiers giving 1 to 1+ mappings will be supported in future versions of the package.
collapse_fun	Which function to use to treat multiple lines targetting the same gene (default=median)

Details

This function is normally only called from "virtualArray.ExpressionSet". You can use it, however, to collapse expression values referring to the same identifier together with their corresponding annotation in the "exprs" and "fData" slots by means of a user specified function (default is "median"). Note, that it is critical to define the correct Bioconductor annotation package in the "annotation" slot of the ExpressionSet and that the name of the ExpressionSet must be supplied as a character vector, due to the implementation in "virtualArray.ExpressionSet".

Value

The value returned is an ExpressionSet whose "fData" slot has been filled with the selected identifiers. The rows in the expression matrix ("exprs" slot) have been collapsed to the selected identifiers. Therefore the size of the expression matrix has decreased during the process.

Author(s)

Andreas Heider

See Also

virtualArray-package, virtualArray.ExpressionSet, virtualArrayCompile

Examples

```
## first we need to load dummy data
library(affydata)
data(Dilution)
## we apply RMA to get an ExpressionSet
Dilution <- rma(Dilution,normalize=FALSE)
## now we collapse the expression values as stated in "Details"
Dilution_genesymbols <- virtualArrayBuildfData(x="Dilution")
## as you can see, we now have only one row per gene symbol
Dilution_genesymbols
```

---

virtualArrayBuildSampleInfo

Build data.frame suitable as "pData" for several ExpressionSets

---

Description

A data.frame is created that holds the sample names and names of the ExpressionSets of the supplied list of ExpressionSets.

Usage

```
virtualArrayBuildSampleInfo(x)
```

Arguments

x	A list of ExpressionSets. Each entry must be represented by a character vector.

Details

This function is normally only called by "virtualArray.ExpressionSet".

Value

A data.frame is returned. It consists of the columns "Array.name", "Sample.name", "Batch" and "Covariate.1". The "Batch" column represents the name of the originating ExpressionSet. The column "Covariate.1" is only a dummy holding running numbers, one for each sample. It has to be edited and filled with meaningful information when "virtualArray" is run in supervised mode. Finally the "sample_info" data.frame can be used to fill the "pData" slot for a merged ExpressionSet derived from several ones.

Author(s)

Andreas Heider (2011)

See Also

virtualArray-package, virtualArray.ExpressionSet, virtualArrayCompile

Examples

```
## first we need to load dummy data
library(affydata)
data(Dilution)
data(sample.ExpressionSet)
## we apply RMA to get an ExpressionSet
Dilution <- rma(Dilution,normalize=FALSE)
## we store the names of the ExpressionSets in a list
all_ExpressionSets <- list("Dilution","sample.ExpressionSet")
## now we generate a new data.frame suitable for "pData"
virtualArrayBuildSampleInfo(all_ExpressionSets)
```

---

virtualArrayComBat *Removes batch effects from microarray derived expression matrices. Modified version.*

Description

This is a modified version of the R script "ComBat.R" (see references). It is used to adjust for batch effects in microarray data. The modification is restricted to make the script accept expression matrices and data.frames instead of plain text files.

Usage

```
virtualArrayComBat(expression_xls, sample_info_file, type = "txt", write = FALSE, covariates = "
```

Arguments

expression_xls	The expression matrix to adjust.
sample_info_file	The sample information data.frame regarding batch contribution and possibly covariates.
type	The type of input; Defaults to "txt".
write	Write output to external file or provide new expression matrix.
covariates	Describe which Covariates to use in the process and which to dismiss. The default is to use "all".
par.prior	Logical; set prior parameters or not; Use prespecified values for the variables ("TRUE") or start a priori ("FALSE").
filter	Filter for genes not present in a given percentage of the samples. Requires present/absent calls in the data. Can be either "FALSE" or a numeric between "0" and "1". Recommended is "0.8" or "FALSE".
skip	Columns to skip in the input "expression_xls" matrix.
prior.plots	Create quantile-quantile and kernel density plots including prior estimates to assess the quality of the estimation.

Value

Returns a matrix holding adjusted expression values.

Note

Original code by Johnson, WE, Rabinovic, A, and Li, C, made available in this package by Andreas Heider

Author(s)

Original author: Johnson, WE, Rabinovic, A, and Li, C (2007)

Modified by: Andreas Heider (2011)

References

Johnson, WE, Rabinovic, A, and Li, C (2007). Adjusting batch effects in microarray expression data using Empirical Bayes methods. Biostatistics 8(1):118-127.

See Also

virtualArray-package, virtualArray.ExpressionSet, virtualArrayCompile

Examples

```
## EMPTY
```

`virtualArrayCompile` *Compiles a single virtual array holding user provided microarray data from multiple platforms.*

Description

The function takes ExpressionSets of rawdata provided by the "virtualArrayLoadRaw" function together with annotation data.frames provided by the function "virtualArray.load.annot" and matches the rawdata on the basis of the given identifier. Currently only "Gene.Symbol" is supported by "virtualArrayCompile" but this will be extendend in future versions. Meanwhile you can use "virtualArray.ExpressionSet" to use other identifiers.

Usage

```
virtualArrayCompile(root_dir = getwd(), identifier = "Gene.Symbol", covars= 3, virtualArray = vi
```

Arguments

`root_dir`	The top level directory holding the rawdata and the "sample_info.txt" file.
`identifier`	The annotation identifier based on which the comparison between the different platforms is made.
`covars`	Numer of columns to read from "sample_info" data.frame or ".txt". Defaults to 3, which denotes only to use the sample/array naming and the "Batch" column. A 4 expects the "Covariate.1" column to be filled with meaningful information regarding grouping of samples. When the value is the character vector "all" even more columns can be used.
`virtualArray`	A list which holds the intermediate output of the "virtArray" package. It contains both raw data as well as annotation data.frames. It defaults to "virtualArray" but can be any other name.

Value

An ExpressionSet is created, holding data of all supplied microarray platforms.

Author(s)

Andreas Heider (2011)

See Also

virtualArray-package, virtualArray.ExpressionSet, virtualArrayCompile

Examples

```
# This is example is commented out, because it would slow down building the package!
# virtualArrayDirs()
# getGEOSuppFiles(baseDir="rawdata/Affymetrix/U133Plus2/",makeDirectory=F,GEO="GSM589512")
# gunzip(filename="rawdata/Affymetrix/U133Plus2/GSM589512.CEL.gz",remove=T)
# getGEOSuppFiles(baseDir="rawdata/Affymetrix/U133A/",makeDirectory=F,GEO="GSM589506")
# getGEOSuppFiles(baseDir="rawdata/Affymetrix/U133A/",makeDirectory=F,GEO="GSM589509")
# gunzip(filename="rawdata/Affymetrix/U133A/GSM589506.CEL.gz",remove=T)
# virtualArray <- virtualArrayLoadRaw()
# my_ExpressionSet <- virtualArrayCompile(virtualArray=virtualArray)
```

`virtualArrayDirs` *Build virtualArray directory structure to hold raw data of multiple microarray chips*

Description

This function simply creates a directory tree in the current working directory or a user specified one. This directory tree helps organizing raw data files and at the same time gives R/Bioconductor the clue who to handle the provided data.

Usage

```
virtualArrayDirs(root_dir = getwd())
```

Arguments

`root_dir`	If declared, this variable sets the top level directory where the whole tree will be placed.

Details

The directory tree consists of a top level folder called "rawdata". Inside this folder there are directories for "Affymetrix", "Agilent" and "Illumina" respectively. Each folder holds sub-folders for the specific chips of the respective manufacturer. The user has to supply copies of the raw data into the appropriate directories for the package to work as expected.

Value

This function returns no value. It just creates a directory tree.

Author(s)

Andreas Heider (2011)

See Also

virtualArray-package, virtualArray.ExpressionSet, virtualArrayCompile

Examples

```
## we use the function in the current working directory
## please note that this will write to your hard drive
virtualArrayDirs()
## now we can see what happened
dir(recursive=TRUE)
```

`virtualArrayExpressionSets` *Combine different ExpressionSets into one*

Description

This function selects all ExpressionSets in the current environment and builds a new single ExpressionSet of the raw data included in the input. This is done by annotating the expression values with the selected identifiers, that are pulled from Bioconductor annotation packages. Then lines targetting the same gene are collapsed by the specified function. In the next step compatible rows of the expression matrices are merged. As a final step batch effects resulting from different platforms or labs can be removed in a supervised or non-supervised mode.

Usage

```
virtualArrayExpressionSets(all_expression_sets=FALSE, identifier = "SYMBOL", covars = 3, collaps
```

Arguments

`all_expression_sets`	Logical or a character vector. If "FALSE", "virtualArray" tries to catch all ExpressionSets in the current environment. If set to a character vector holding names of ExpressionSets, these are used instead of all available ones.
`identifier`	annotation identifier by which the expression values are combined
`covars`	numerical or character vector of length "1". See details for more info.
`collapse_fun`	The function to be used to collapse expression values targetting the same gene/identifier. Defaults to "median".
`removeBatcheffect`	Logical or character vector. "FALSE" will lead to just a combined ExpressionSet, you will then have to use other functions to remove the batch effects. You can set it to "EB", "GQ", "MRS", QD" ,"NORDI" or "MC" to use empirical Bayes methods, gene quantiles, median rank scores, quantile discretization ,normal discretization or mean centering to remove batch effects, respectively.
`sampleinfo`	If you run in non-interactive mode, you can specify a data.frame to be used as the input "sample_info". Note that normally this data.frame is created on the fly, so you will need to set it up manually in this case.
`supervised`	Logical; select if you want to run in supervised or non-supervised mode. In non-supervised mode only contribution of samples to batches are relevant for batch effect removal, whereas in supervised mode the 4th column of "sample_info" or even more columns are used to group samples based on the users decision. Please see the vignette for more information to this option.
`...`	Can be used to pass on parameters to underlying functions.

Details

The "covars" argument determines the mode of batch removal. It refers to the columns in the sample_info data.frame which contains information about all ExpressionSets, their samples and relations thereof. The default value of "3" will use only the different ExpressionSets for batch effect removal, this is referred to as the non-supervised mode. The supervised mode is to be accessed by

using a higher numerical value than "3" or the character vector "all". In this case the sample_info data.frame has to be modified manually to contain more information on the batches in additional columns. Please note, that during computation you will be notified that "sample_info.txt" has been written to your current working directory for you to modify and save it. If you do so, please select "y" to use the additional columns. Also note that you can not provide a covariate that is distributed only in one batch, this way the procedure will fail.

Value

A new ExpressionSet is returned that combines all ExpressionSets from the current environment.

Author(s)

Andreas Heider (2011)

See Also

virtualArray-package, virtualArray.ExpressionSet, virtualArrayCompile, normalize.ExpressionSet.nordi, normalize.ExpressionSet.mrs, normalize.ExpressionSet.qd, normalize.ExpressionSet.gq

Examples

```
# Due to the flexibility of this function and the time
# it takes to get meaningful results, please see the
# vignette for a comprehensive example, governing
# several modes of usage. Thanks.
```

---

virtualArrayHclust *Plot a hclust object with colored labels*

Description

This function takes a hclust object and a vector of colors as its input and plots the hclust object with colored labels.

Usage

```
virtualArrayHclust(hclust, lab = hclust$labels, lab.col = rep(1, length(hclust$labels)), hang =
```

Arguments

hclust	a hclust object
lab	labels to use
lab.col	colors to use for each label
hang	see "hclust"
...	see "hclust"

Details

This function is normally only called by "virtualArray.ExpressionSet".

Value

A modified hclust object is returned.

Note

The function was written by Eva Chan and incorporated into this package by Andreas Heider

Author(s)

Original author: Eva Chan

Modified by: Andreas Heider (2011)

See Also

virtualArray-package, virtualArray.ExpressionSet, virtualArrayCompile

Examples

```
# we use the USArrests dataset
new.arrests <- USArrests
# now we need to specify a function to create colors
color_me <- function(x){
if(x >= 100){"red"}
else{"blue"}
}
# lets build the colors in relation to the "Assaults" statistic
new.arrests[,5] <- sapply(new.arrests[,2],color_me)
# we need a distence matrix to build our tree
dist1 <- dist(new.arrests[,1:4],method="euclidian")
# and we need a conventional hclust object
hc_nocolor <- hclust(dist1,method="average")
# now we can hook up to that object and change the label colors
# note that this call will automatically plot the tree
virtualArrayHclust(hc_nocolor,lab=rownames(new.arrests),lab.col=new.arrests[,5])
```

---

virtualArrayLoadRaw *Load user provided raw data files in the virtualArray directory structure*

Description

This function loads all raw data files provided by the user in the virtualArray directory structure, that was created by virtualArrayDirs in the first place.

Usage

```
virtualArrayLoadRaw(root_dir = getwd(), affy_sum = "rma")
```

Arguments

root_dir	The top level directory holding the rawdata and the "sample_info.txt" file.
affy_sum	Select the summerization method to use for Affymetrix chips; defaults to "RMA".

Details

The function first reads in the virtualArray directory including raw data files. Based on this a "sample_info" data.frame is created holding information about chip type and the corresponding file name. Then "sample_info" is checked for the presence of certain microarray platforms. If present, the function loads raw data of all platforms. The "sample_info" data.frame is exported to "sample_info.txt". It is up to the user to change the values of the "Covariate 1" column to something meaningful, thus providing the additional information for the batch effect removal in the final step of the process.

Value

This function does not return a value, but creates objects in the current environment holding raw data of every provided chip type. The objects are named "rawdata_*". Where "*" is the chip type.

Author(s)

Andreas heider (2011)

See Also

virtualArray-package, virtualArray.ExpressionSet, virtualArrayCompile

Examples

```
## EMPTY
```

---

virtualArrayMergeRecurse

Merge a list of data.frames by common rows

Description

A list of data.frames is merged together by rows common to all data.frames. The column by which the merging is performed can be selected with "by".

Usage

```
virtualArrayMergeRecurse(dfs, by, ...)
```

Arguments

dfs	A list of data.frames
by	Select the columns to be used for the comparison
...	Parameters passed on to the "merge" function.

Details

The function was adopted from the "reshape" package and modified to be suitable to handle problems arising when combining data from different microarray platforms in the "virtualArray" package. This function is normally only called by "virtualArray.ExpressionSet". It can be used, however, to merge several data.frames by common rows, whereas rows that do not match between the data.frames are discarded.

Value

A data.frame consisting of the matching parts of the input data.frames.

Author(s)

Andreas heider (2011)

See Also

virtualArray-package, virtualArray.ExpressionSet, virtualArrayCompile

Examples

```
## we set up 2 example data.frames
authors <- data.frame(
name = I(c("Tukey", "Venables", "Tierney", "Ripley", "McNeil")),
nationality = c("US", "Australia", "US", "UK", "Australia"),
deceased = c("yes", rep("no", 4)))
books <- data.frame(
name = I(c("Tukey", "Venables", "Tierney",
"Ripley", "Ripley", "McNeil", "R Core")),
title = c("Exploratory Data Analysis",
"Modern Applied Statistics ...",
"LISP-STAT",
"Spatial Statistics", "Stochastic Simulation",
"Interactive Data Analysis",
"An Introduction to R"),
other.author = c(NA, "Ripley", NA, NA, NA, NA,
  "Venables & Smith"))
## let's have a look at the data
authors
books
## we store the data.frames in a list
my_dfs <- list(authors,books)
## now we can combine the 2 data.frames
virtualArrayMergeRecurse(my_dfs,by="name")
```

Index

∗Topic **ExpressionSet**
virtualArrayBuildfData, 9
∗Topic **array**
virtualArrayDirs, 14
∗Topic **batch effect removal**
normalize.ExpressionSet.gq, 3
normalize.ExpressionSet.mc, 4
normalize.ExpressionSet.mrs, 5
normalize.ExpressionSet.nordi, 6
normalize.ExpressionSet.qd, 7
∗Topic **batch effects**
virtualArrayComBat, 11
∗Topic **batch**
virtualArrayComBat, 11
∗Topic **combine**
virtualArrayBuildExprs, 8
virtualArrayBuildfData, 9
virtualArrayBuildSampleInfo, 10
virtualArrayDirs, 14
virtualArrayExpressionSets, 15
virtualArrayHclust, 16
virtualArrayLoadRaw, 17
virtualArrayMergeRecurse, 18
∗Topic **compare**
virtualArrayCompile, 13
∗Topic **cross-platform**
normalize.ExpressionSet.gq, 3
normalize.ExpressionSet.mc, 4
normalize.ExpressionSet.mrs, 5
normalize.ExpressionSet.nordi, 6
normalize.ExpressionSet.qd, 7
∗Topic **merge**
virtualArrayDirs, 14
∗Topic **normalization**
normalize.ExpressionSet.gq, 3
normalize.ExpressionSet.mc, 4
normalize.ExpressionSet.mrs, 5
normalize.ExpressionSet.nordi, 6
normalize.ExpressionSet.qd, 7
∗Topic **package**
virtualArray-package, 2
∗Topic **virtualArray**
normalize.ExpressionSet.gq, 3
normalize.ExpressionSet.mc, 4
normalize.ExpressionSet.mrs, 5
normalize.ExpressionSet.nordi, 6
normalize.ExpressionSet.qd, 7
virtualArrayBuildExprs, 8
virtualArrayBuildfData, 9
virtualArrayBuildSampleInfo, 10
virtualArrayExpressionSets, 15
virtualArrayHclust, 16
virtualArrayMergeRecurse, 18
∗Topic **virtual**
virtualArrayBuildExprs, 8
virtualArrayBuildfData, 9
virtualArrayBuildSampleInfo, 10
virtualArrayCompile, 13
virtualArrayDirs, 14
virtualArrayExpressionSets, 15
virtualArrayHclust, 16
virtualArrayLoadRaw, 17
virtualArrayMergeRecurse, 18

normalize.ExpressionSet.gq, 3
normalize.ExpressionSet.mc, 4
normalize.ExpressionSet.mrs, 5
normalize.ExpressionSet.nordi, 6
normalize.ExpressionSet.qd, 7

virtualArray *(*virtualArray-package*)*, 2
virtualArray-package, 2
virtualArrayBuildExprs, 8
virtualArrayBuildfData, 9
virtualArrayBuildSampleInfo, 10
virtualArrayComBat, 11
virtualArrayComBat,character-method *(*virtualArrayComBat*)*, 11
virtualArrayComBat,data.frame-method *(*virtualArrayComBat*)*, 11
virtualArrayComBat,ExpressionSet-method *(*virtualArrayComBat*)*, 11
virtualArrayCompile, 13
virtualArrayDirs, 14
virtualArrayExpressionSets, 15
virtualArrayHclust, 16
virtualArrayLoadRaw, 17

virtualArrayMergeRecurse, 18