Package 'GeneRegionScan'

April 22, 2026

Version 1.67.0

Date 2009-10-16

Title GeneRegionScan

Author Lasse Folkersen, Diego Diez

Maintainer Lasse Folkersen <lasfol@cbs.dtu.dk>

Depends methods, Biobase (>= 2.5.5), Biostrings

Imports S4Vectors (>= 0.9.25), Biobase (>= 2.5.5), affxparser, RColorBrewer, Biostrings

Suggests BSgenome, affy, AnnotationDbi

LazyLoad yes

Description

A package with focus on analysis of discrete regions of the genome. This package is useful for investigation of one or a few genes using Affymetrix data, since it will extract probe level data using the Affymetrix Power Tools application and wrap these data into a ProbeLevelSet. A ProbeLevelSet directly extends the expressionSet, but includes additional information about the sequence of each probe and the probe set it is derived from. The package includes a number of functions used for plotting these probe level data as a function of location along sequences of mRNA-strands. This can be used for analysis of variable splicing, and is especially well suited for use with exon-array data.

License GPL (>= 2)

biocViews Microarray, DataImport, SNP, OneChannel, Visualization

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

git_branch devel

git_last_commit e7590a3

git_last_commit_date 2025-10-29

Repository Bioconductor 3.24

Date/Publication 2026-04-21

Contents

`addSnpPdata` *Add SNP data from hapmap-style data file to pData of ExpressionSet*

Description

Function that will add genotypes to an ExpressionSet when given a data file with genotypes in the same format as outputted by http://www.hapmap.org.

Usage

```
addSnpPdata(object, listOfSnps, individualNames="sampleNames")
```

Arguments

`object`	A ProbeLevelSet object or a regular ExpressionSet object (in which case a probeData argument is required). See getLocalProbeIntensities and related functions on how to create a ProbeLevelSet.
`listOfSnps`	A character string giving the path of a file containing SNP data in the same format as used by the export function of www.hapmap.org.
`individualNames`	An optional character string giving the column name of a pData entry that holds the individual names as given in the listOfSnps. This defaults to using the sampleNames, and is useful in cases with replicate or triplicate samples

Details

A function that takes an ExpressionSet and the path for a SNP-file. The SNP-file should have the same format as files downloaded from http://www.hapmap.org. The function then checks if the sampleNames are present in the SNP-file. If they are, it decorates the ExpressionSet with the SNPs. If the snp name already exists as a column name in the pData, the function tries to merge the new information from listOfSnps with the new. Original NA values are overwritten by new values. Original values that are the same as new values are kept the same. Original values that are not the same as new values are preserved in their original state and a warning is given.

Value

The same ProbeLevelSet or ExpressionSet given as argument, but with the SNP type data added to the pData and with the extra information of the listOfSnps file added to the notes section.

Author(s)

Lasse Folkersen

See Also

`getLocalProbeIntensities`,`plotOnGene`

Examples

```
## Not run:
hapmapformatdata<-"~/somefile.txt"
probelevelsetwithsnps<-addSnpPdata(probelevelset,hapmapformatdata)

## End(Not run)
```

---

checkForFileInPath *Check For File In Path*

Description

Internal function used to check for files in path.

Usage

```
checkForFileInPath(filenames)
```

Arguments

`filenames`	A character vector of filenames to check for - c("xxxx.exe","xxxx") for example

Details

This function is not meant to be called by the user so it is not in the namespace.

Value

A character vector with the expanded paths and filenames to the found files.

Author(s)

Lasse Folkersen

See Also

plotOnGene

Examples

```
GeneRegionScan:::checkForFileInPath(c("python","python.exe"))
```

`doProbeLinear` *Do Probe Linear*

Description

Internal function used to calculate linear correlation between probe level intensity and factorial pData entries such as genotypes.

Usage

```
doProbeLinear(object, label, testType="linear model")
```

Arguments

`object`	A ProbeLevelSet object or a regular ExpressionSet object.
`label`	An optional character string specifying a column name in the pData of the object. If this argument is given, the gene plot will be colour coded based on the different groups (factors) in the pData entry. If a summaryType other than 'dots' is selected the summarisation is done stratified by the different groups in the pData.
`testType`	Character string, defining a statistic procedure to identify especially interesting probes.

Details

This function is not meant to be called by the user. It does the statistical calculations when called by plotStatistics.

One important note about the assumption that the alphabetical order is the same as value order. This usually works with SNP datatypes of the type AA, AB, BB. However the sorting is locale dependent and can produce odd results, which is why the value assignments are always printed. See Comparison for details on this. Further user-control over this aspect of the plotting function was omitted, giving more weight to the simplicity of the function interface. In case of problems it is recommended to rename the entries in the label so that they will be correctly sorted.

Value

A correlation list with p-value and fold-change in columns and probe IDs as rownames. Used by plotStatistics when decorating plots from plotOnGene.

Author(s)

Lasse Folkersen

See Also

plotStatistics, plotOnGene

Examples

```
data(exampleProbeLevelSet)
colnames(pData(exampleProbeLevelSet))
linearModelData<-GeneRegionScan:::doProbeLinear(exampleProbeLevelSet,"genotype3")
colnames(linearModelData)
#P value for the probe with ID 679083 in relation to "genotype3"
print(linearModelData["679083","p-value"])
```

---

doProbeTTest *Do Probe T-Test*

Description

Internal function used to calculate t-tests between probe level intensity and pData entries with two levels.

Usage

```
doProbeTTest(object, label, testType="students")
```

Arguments

object	A ProbeLevelSet object or a regular ExpressionSet object.
label	An optional character string specifying a column name in the pData of the object. If this argument is given, the gene plot will be colour coded based on the different groups (factors) in the pData entry. If a summaryType other than 'dots' is selected the summarisation is done stratified by the different groups in the pData.
testType	Character string, defining a statistic procedure to identify especially interesting probes.

Details

This function is not meant to be called by the user. It does the statistical calculations when called by plotStatistics

Value

A correlation list with p-value and fold-change in columns and probe IDs as rownames. Used by plotStatistics when decorating plots from plotOnGene.

Author(s)

Lasse Folkersen

See Also

plotStatistics, plotOnGene

Examples

```
data(exampleProbeLevelSet)
colnames(pData(exampleProbeLevelSet))
tTestData<-GeneRegionScan:::doProbeTTest(exampleProbeLevelSet,"gender")
colnames(tTestData)

#P value for the probe with ID 679083 in relation to "gender"
print(tTestData["679083","p-value"])
```

---

exampleProbeLevelSet *Example Dataset of class 'ProbeLevelSet'*

Description

This ProbeLevelSet is derived from Human exon ST 1.0 cel files downloaded from the http://www.ncbi.nlm.nih.gov/geo/. The pData in the ProbeLevelSet is entirely fictional, and the intensity have been changed for some probes to show a point about correlation to genotype.

Usage

```
data(exampleProbeLevelSet)
```

Format

A ProbeLevelSet object containing probe level intensity data for a region on chromosome 2.

Author(s)

Lasse Folkersen

Examples

```
data(exampleProbeLevelSet)
pData(exampleProbeLevelSet)
exprs(exampleProbeLevelSet)[1:10,]
```

`excludeDoubleMatchingProbes` *Exclude ProbeLevelSet probes that match more than once in genome*

Description

Function that will remove probes from ProbeLevelSet if they have more than one match in a given genome.

Usage

```
excludeDoubleMatchingProbes(object, genome="BSgenome.Hsapiens.UCSC.hg18", verbose=TRUE,
directions=c("matchForwardSense", "matchForwardAntisense", "matchReverseSense", "matchReverseA
 previousData = NULL)
```

Arguments

`object`	A ProbeLevelSet class object.
`genome`	character string with the name of the BSGenome in which sequences should be found. Defaults to the human genome.
`verbose`	TRUE or FALSE.
`directions`	character string with elements from c("matchForwardSense", "matchForwardAntisense", "matchReverseSense", "matchReverseAntisense"). Defines which directions (complementary and reverse mirrorings) that should be scanned. Defaults to all directions.
`previousData`	Optional: The output from a call to findSequenceInGenome. If given the scanning will be skipped, and the probes will be omitted directly. Useful in cases were datasets from the same region needs to be processed.

Details

This function will take quite a while to run, so if you have many sequences, overnight runs are recommended. BSgenome contains some alternative versions of chromosomes. They are marked with an underscore. This function automatically disregards chromosome names with an underscore, and this is known to work for the human genome. Nevertheless, check the output printed to terminal if all chromosomes are included. The function is a wrapper around findSequenceInGenome, which can be used for purposes that are more flexible (although that function really is just following the example in the BSgenome package)

At present, there is no functionality to check matches with known SNP or known splice forms taken into account.

Value

The ProbeLevelSet class object provided as argument, with all double matching probes removed. Double matching probes are probes whose sequence are found twice or more in the genome. In addition, the output of the matching investigation is saved in the notes of the ProbeLevelSet and can be further examined for information on the locations of the probe sequences in relation to the BSgenome sequences.

Author(s)

Lasse Folkersen

See Also

findSequenceInGenome, BSgenome, excludeDoubleMatchingProbes

Examples

```
## Not run:
#you can run this, but it takes a lot of time
probelevelsetwithnodoubles<-excludeDoubleMatchingProbes(probelevelset)

## End(Not run)
```

---

exonStructure *Add exon-structure to plots*

Description

Function that will paint the exon structure of a gene on the plots obtained by plotOnGene.

Usage

```
exonStructure(mrna, genome, maxMismatch=4, y=0)
```

Arguments

mrna	A gene sequence formatted as DNAstring, DNAstringSet or character-vectors with sequence.
genome	A number of gene sequences as DNAstring, vectors of DNAStrings, DNAstringSet or character-vectors with sequence.
maxMismatch	Integer. The maximum number of mismatches per exon that can be allowed before the exon is not allocated at the position in the template mrna. Defaults to 4.
y	Numeric. The vertical position of the exon structure (if ylim is changed)

Details

When given a sequence of the DNA divided by exons and a sequence of the corresponding mRNA string, this function will plot the layout of exons along the length of the x-axis on the current device. The sequences can be given as character vectors with sequence or as DNAString reads using the Biostrings package. Furthermore the genome must be divided with an entry for each exon. This is easily done by downloading the genome sequences of the gene-of-interest from http://genome.ucsc.edu and specifying "One FASTA record per region".

The function works by sequentially comparing each exon to the mrna. The location of the match start and end is taken as the exon boundaries and plotted. If more than one match is found a warning is given. If no match is found for an exon this is printed, but otherwise ignored. The mrna can in fact also be DNA sequence with introns. The important thing is that it serves as template for the exon matching.

Importantly, the exon-numbers technically refer to "number of exon in investigated transcript". For example if an DNA with exon structure for an isoform which does not include all exons in the gene is investigated, then there will be skips of exon numbers. To avoid this, the DNA for an isoform which do include all exons could be used. However, it is really more a biological issue: different sources can differ on where to start the counting for a given gene in any case.

Value

No value, but plots the layout of exons in a gene on the product of a call to plotOnGene.

Author(s)

Lasse Folkersen

See Also

geneRegionScan, plotOnGene

Examples

```
data(exampleProbeLevelSet)
plotOnGene(exampleProbeLevelSet, mrna, label="gender", testType="wilcoxon")
exonStructure(mrna, genomic, maxMismatch=2)
```

---

findProbePositions *Find positions of probes on a gene*

Description

Function that will the location of probes in a gene based on their sequence.

Usage

```
findProbePositions(object, gene, probeData=NULL,interval=NULL,directions="all",verbose=TRUE)
```

Arguments

object	A ProbeLevelSet object or a regular ExpressionSet object (in which case a probeData argument is required). See getLocalProbeIntensities and related functions on how to create a ProbeLevelSet.
gene	A number of gene sequences as DNAstring, DNAStringsSets, or character-vectors with sequence.
probeData	Optional if a ProbeLevelSet is submitted as object argument. Otherwise, it must be a data frame with rownames corresponding to the featureNames of the ExpressionSet and a column named "sequence" with the probe sequences as character strings
interval	Optional vector of two integers of bp positions. If given, the plot will only include the sequence from gene in the given interval. The x-axis annotation is preserved from original, so this is useful for zooming on specific regions.
verbose	TRUE or FALSE

directions — A character vector of the matching-directions that should be scanned (which combinations of complementary and reverse). Defaults to "all" which is shorthand for all possible directions, but can take anything from: c("matchForwardSense", "matchForwardAntisense", "matchReverseSense", "matchReverseAntisense")

Details

This function is principally used by the plotOnGene to assign positions of each probe relative to the gene sequence of interest. In a recent version of GeneRegionScan it was separated as a discrete function because of its use in alternative plotting

Value

A vector of positions of each probe in the object ProbeLevelSet, with names being probe ids

Author(s)

Lasse Folkersen

See Also

geneRegionScan, plotOnGene, plotCoexpression

Examples

```
data(exampleProbeLevelSet)

findProbePositions(exampleProbeLevelSet, mrna)
```

---

findSequenceInGenome *Find a sequence in genome*

Description

Wrapper around matchPDict that will accept a list of sequences and check if they are present in a given genome. Takes a long time to run.

Usage

```
findSequenceInGenome(sequences,
    genome="BSgenome.Hsapiens.UCSC.hg19", verbose=TRUE,
    directions=c("matchForwardSense", "matchForwardAntisense",
                 "matchReverseSense", "matchReverseAntisense"))
```

Arguments

sequences — vector of character strings to scan. Should only contain A, C, G and T. Will be converted to DNAString.

genome — character string with the name of the BSGenome in which sequences should be found. Defaults to the human genome.

verbose — TRUE or FALSE.

directions — character string with elements from c("matchForwardSense", "matchForwardAntisense", "matchReverseSense", "matchReverseAntisense"). Defines which directions (complementary and reverse mirrorings) that should be scanned. Defaults to all directions.

Details

This function will take quite a while to run, so if you have a many sequences, overnight runs are recommended. BSgenome contains some alternative versions of chromosomes. They are marked with an underscore. This function automatically disregards chromosome names with an underscore, and this is known to work for the human genome. Nevertheless, check the output printed to terminal if all chromosomes are included.

Value

A data frame with a row for each identified match. Columns are "entrynumber","hitposInChr","chr", and "sequence" describing, respectively: the index of the sequence match, the position in the chromosome at which it was found, which chromosome it was found on, the sequence itself

Author(s)

Lasse Folkersen

See Also

BSgenome, matchPDict, excludeDoubleMatchingProbes

Examples

```
## Not run:
#you can run this, but it takes quite a lot of time
example<-findSequenceInGenome("CTGGCGAGCAGCGAATAATGGTTT")

## End(Not run)
```

---

geneRegionScan *Gene Region Scan*

Description

The top-level wrapper function that outputs as much data as possible, concerning one or a few genes. Refer to the functions plotCoexpression and plotOnGene for further explanation of each part of the plot.

Usage

```
geneRegionScan(object, gene, genomicData=NULL, probeData=NULL, label=NULL, genename=NULL, summa
```

Arguments

object	A ProbeLevelSet object or a regular ExpressionSet object (in which case a probeData argument is required). See getLocalProbeIntensities and related functions on how to create a ProbeLevelSet.
gene	A number of gene sequences as DNAString, DNAstringSets, or character-vectors with sequence.
genomicData	Optional. If only one gene is specified this can be of the same form as given in exonStructure. If more than one gene is given, it must be a list, containing of one of these forms of the argument for each of the genes, in the same order.
probeData	Optional if a ProbeLevelSet is submitted as object argument. Otherwise, it must be a data frame with rownames corresponding to the featureNames of the ExpressionSet and a column named "sequence" with the probe sequences as character strings
label	An optional character string specifying a column name in the pData of the object. If this argument is given, the gene plot will be colour coded based on the different groups (factors) in the pData entry. If a summaryType other than 'dots' is selected the summarisation is done stratified by the different groups in the pData.
genename	Optional character string specifying a gene name to include in the plot. If not included and a FASTA sequence is given, it will default to the name in the FASTA sequence. Otherwise it will default to 'Unknown genename'.
summaryType	Character string specifying one of the following summary methods: 'median', 'mean', 'quartiles' or 'dots' (i.e. no summary). Specifies how all the sample values or all the samples values in a group if 'label' is given, should be summarised. Defaults to 'median'.
testType	Optional character string, defining a statistic procedure to identify especially interesting probes. Can be either 'linear model', 'students' or 'wilcoxons'. If given, a label must also be specified. In this case the plotStatistics function will be called and probes that are significantly changed between the groups in label at the P-value set in cutoff (see cutoff argument) will be circled.
forcePValue	Logical. Is used if the testType argument is used. If TRUE all significantly changed probes have P-value given on the plot. If FALSE, only plots with less than 10 significant probes write P-values. Plots can become very cluttered with data if set to TRUE
verbose	TRUE or FALSE
cutoff	Integer specifying at what p-value probes should be circled when using the 'testType' variable. Defaults to 0.2. For cutoffs higher than 0.05, all probes with P >0.05 will be circled in grey instead of black.
directions	A character vector of the matching-directions that should be scanned (which combinations of complementary and reverse). Defaults to "all" which is shorthand for all possible directions, but can take anything from: c("matchForwardSense", "matchForwardAntisense", "matchReverseSense", "matchReverseAntisense")
correlationCutoff	A number between 0 and 1. The limit at which Pearson correlation (in absolute values) should not be plotted below. Defaults to 0.5
probeLevelInfo	The information about each probe to include in the plot. Should be a vector of one or more of the following elements: probeid, probesetid, sequence. Default is only probeid.
ylim	Label of Y-axis as in default plots

Details

This function is a wrapper around plotOnGene and plotCoexpression. The output of plotOnGene is included in the top half of the plot and the output of plotCoexpression will be included in the bottom half of the plot. Refer to each of these functions for more detailed help.

In general, this function gives an overview of how intensity values of individual probes on a microarray are in relation to an actual gene or set of genes in a region of the genome.

The function has only been tested with up to four genes at the same time. A plot with more genes would probably also be too complicated to interpret with this method. In addition, the alignment of the top and bottom plots also becomes somewhat difficult with more genes. This alignment is also the reason why the function can not export to any device.

See getLocalProbeIntensities for more info on how to obtain ProbeLevelSets.

Value

No value, but plots the local expression levels of each probe found in the submitted sequence of the gene or genes as a function of its location along this sequence on the top half of a pdf-file. On the bottom half of the pdf-file it will plot the pairwise correlations between all probes found in the sequences. The pdf file will be named "report_" + title of ExpressionSet or ProbeLevelSet

Author(s)

Lasse Folkersen

See Also

plotCoexpression, plotOnGene, getLocalProbeIntensities

Examples

```
data(exampleProbeLevelSet)

#simple:
geneRegionScan(exampleProbeLevelSet,mrna)

#more complicated - note that we slice the mrna to simulate comparing two different isoforms
gene1<-mrna[[1]][1:1000]
gene2<-mrna[[1]][1500:3000]

geneRegionScan(exampleProbeLevelSet, list(gene1,gene2), genomicData=list(genomic,genomic), label="genotype3
  testType="linear model", forcePValue=TRUE, cutoff=0.1, directions="all", correlationCutoff=0.6,
   probeLevelInfo=c("probeid","sequence"))
```

---

genomic *genomic sequence of gene in ProbeLevelSet*

Description

This is the DNA sequence for fibronectin 1 isoform 2 preproprotein, which is used in the exampleProbeLevelSet. It has been split up so each exon in the gene is described in one FASTA entry.

Usage

```
data(exampleProbeLevelSet)
```

Format

It has been read from FASTA format using the readDNAStringSet function. The FASTA was downloaded from http://genome.ucsc.edu, specifying "One FASTA record per region" in the "Sequence Retrieval Region Options".

Author(s)

Lasse Folkersen

---

getLocalMetaprobeIntensities

Get Metaprobe Intensities locally

Description

Function that will create an expressionset from cel files.

Usage

```
getLocalMetaprobeIntensities(celfilePath, analysis="rma", metaProbeSetsFile=NULL, annotation=NUL
pgfPath=NULL, clfPath=NULL, cdfPath=NULL, verbose=TRUE)
```

Arguments

celfilePath	The path to a folder that contains the cel files of interest.
analysis	The analysis string passed to Affymetrix Power Tools. "RMA" and "RMA-sketch" are recommended starting points. See APT documentation for complete list of possibilities.
metaProbeSetsFile	Path to a file containing meta probe set information. These can be downloaded from www.affymetrix.com and has names like HuEx\-1_0\-st\-v2.r2.dt1.hg18.full.mps. This choice also decides which subset of the meta probe sets that will be used: Core, Extended or Full.
annotation	Character string specifying which type of arrays is investigated.
aptProbesetSummarizePath	The path to the apt-probeset-summarize file from the Affymetrix Power Tools package (including the filename itself). You can try to leave it as NULL and see what happens. If you have specified the argument before it will be remembered.
pgfPath	The path to a pgf file for the exon array of interest. This argument is mutually exclusive with the cdfPath argument. These files can be downloaded from www.affymetrix.com for the array of interest. Once given for a particular annotation, the location is saved for future use and can be given as NULL next time.

clfPath	The path to a clf file for the exon array of interest. This argument is mutually exclusive with the cdfPath argument. These files can be downloaded from www.affymetrix.com for the array of interest. Once given for a particular annotation, the location is saved for future use and can be given as NULL next time.
cdfPath	The path to a cdf file for the array of interest. This argument is mutually exclusive with the pgfPath and clfPath arguments. These files can be downloaded from www.affymetrix.com for the array of interest. Once given for a particular annotation, the location is saved for future use and can be given as NULL next time.
verbose	TRUE or FALSE.

Details

This function is a simple wrapper around Affymetrix Power Tools (APT). It is useful for importing meta probe set data from cel files to R environments. It will remember the location of annotation files, after the first use, which removes some of the typing otherwise included in using APT. This function has now been updated so that a windows edition of apt-probeset-summarize.exe from APT-1.12.0 is included in the installation package, so that pre-processing of cel files can be done with less setup. Linux users will still have to install APT for this function.

Value

An Expressionset with meta probe set values for all meta probes in the specified metaProbeSetFile.

Author(s)

Lasse Folkersen

See Also

getLocalProbeIntensities, plotOnGene

Examples

```
## Not run:
#must correct paths and give cel files before this example will work
listOfProbesets<-c("10321_at")
celfilePath<-path_to_some_cel_files
aptProbesetSummarizePath<-"~/apt/apt-cel-extract"
cdfPath<-"~/hgu133plus2.cdf"
getLocalMetaprobeIntensities(celfilePath, annotation="hgu133plus2", aptProbesetSummarizePath=aptProbesetSu

## End(Not run)
```

---

getLocalProbeIntensities

Get Probe Intensities locally

Description

Function that will create a ProbeLevelSet from cel files.

Usage

```
getLocalProbeIntensities(listOfProbesets, celfilePath, annotation=NULL, aptCelExtractPath=NULL
  cdfPath=NULL, verbose=TRUE)
```

Arguments

listOfProbesets	Either a character vector with the names of the probesets from which probes should be included in the ProbeLevelSet, or the path name of a file containing this
celfilePath	The path to a folder that contains the cel files of interest.
annotation	Character string specifying which type of arrays is investigated.
aptCelExtractPath	The path to the apt-cel-extract file from the Affymetrix Power Tools package (including the filename itself). You can try to leave it as NULL and see what happens. If the tool is in path, it will work anyway. If you have specified the argument before it will be remembered. If you are using an OS for which executables have been included in the package (win32 and linux64), it will use that.
pgfPath	The path to a pgf file for the exon array of interest. This argument is mutually exclusive with the cdfPath argument. These files can be downloaded from www.affymetrix.com for the array of interest. Once given for a particular annotation, the location is saved for future use and can be given as NULL next time.
clfPath	The path to a clf file for the exon array of interest. This argument is mutually exclusive with the cdfPath argument. These files can be downloaded from www.affymetrix.com for the array of interest. Once given for a particular annotation, the location is saved for future use and can be given as NULL next time.
cdfPath	The path to a cdf file for the array of interest. This argument is mutually exclusive with the pgfPath and clfPath arguments. These files can be downloaded from www.affymetrix.com for the array of interest. Once given for a particular annotation, the location is saved for future use and can be given as NULL next time.
verbose	TRUE or FALSE.

Details

This is the workhorse function for generating ProbeLevelSets. It has two main functions: 1) It extracts the probe level intensity, normalizes it to all probes in the data using quantiles normalization and transfers the results to the exprs part of a ProbeLevelSet. The extraction and normalization parts are done by a call to the Affymetrix Power Tools (APT) function apt-cel-extract. The analysis switch for this call is '-a quant-norm,pm-only' and the documentation for APT can be used for further reference. 2) It extracts the probe sequences of all probes of interest. This is done using the [arrayname]probe packages for the 3'IVT type arrays for which they are available. For the exon type arrays, for which this is not available it is done by direct parsing of the pgf files using the readPgf function of the affxparser package. The results of this is saved in the featureData of the ProbeLevelSet

Value

A ProbeLevelSet with probe intensity values for all probes in the specified probesets. The ProbeLevelSet inherits the ExpressionSet, but in addition it has the sequence of each probe stored in the featureData of the set.

Author(s)

Lasse Folkersen

See Also

getServerProbeIntensities, plotOnGene

Examples

```
## Not run:
#must correct paths and give cel files before this example will work
listOfProbesets<-c("10321_at")
celfilePath<-path_to_some_cel_files
aptCelExtractPath<-"~/apt/apt-cel-extract"
cdfPath<-"~/hgu133plus2.cdf"
getLocalProbeIntensities(listOfProbesets, celfilePath, annotation="hgu133plus2", aptCelExtractPath=aptCelEx

## End(Not run)
```

---

getMetaprobesetsFromRegionOfInterest

Get Meta Probe Set IDs From Region Of Interest

Description

Function that return the meta probe set ids located within a given region of the genome.

Usage

```
getMetaprobesetsFromRegionOfInterest(annotation, chromosome, start, end, pythonPath=NULL, transcr
```

Arguments

annotation	A characther string giving the type of array for which probesets are needed. This is used to save the location of the transcriptClustersFile.
chromosome	The chromosome of interest. Should be given as a character string of the type "Chr1", "Chr2", "ChrY".
start	A character string with the start location of interest in the chromosome.
end	A character string with the end location of interest in the chromosome.
pythonPath	Optional character string with the path for Python software. This is only needed for exon arrays. If Python is in path this will be recognised automatically. Python can be downloaded from http://www.python.org.
transcriptClustersFile	The location of a transcript cluster file such as HuEx-1_0-st-v2.na26.hg18.transcript.csv. These can be downloaded from http://www.affymetrix.com.

Details

Since meta probe sets are only found in exon arrays, this function does not work with 3' IVT array files. It parses the location files from the http://www.affymetrix.com website using a python script. Admittedly this is not optional from an R-only point-of-view, but it works and its fast. The function will be updated when more R-centric ways of parsing exon array annotation data is available.

Alternatively this data can just as well be retrieved from the web, but in some cases this function is faster an easier.

Value

A list of all probesets found in the given range on the given chromosome.

Author(s)

Lasse Folkersen

See Also

getLocalProbeIntensities, getProbesetsFromRegionOfInterest

Examples

```
 ## Not run:
 #must supply transcriptClustersFile for this to work
 metaprobesets<-getMetaprobesetsFromRegionOfInterest("notusedhere", chromosome=2, start="215889955", end="21

## End(Not run)
```

`getProbeLevelAnnotationForExonArrays`
Get ProbeLevel Annotation for Exon Arrays

Description

Internal function that will return exon probe sequence and probe id for all probes in a given list of probe sets.

Usage

```
getProbeLevelAnnotationForExonArrays(vectorOfProbesets, pgfPath)
```

Arguments

`vectorOfProbesets`	A character string with probeset IDs.
`pgfPath`	The path of a pgf file for the exon array type of interest. Can be downloaded from www.affymetrix.com.

Details

This function makes a call to readPgf in the affxparser package to get the sequence information. The readPgf will then load the entire pgf file and that might be quite memory intensive. However it is not possible to extract the indices of the probeset names without doing this. The function is primarily intended to be called by getLocalProbeIntensities.

The call to readPgf sometimes gives a is.na() warning. The reason for this is not known, but it does not seem to affect performance.

Value

A dataframe with a row for each probe in the submitted probeset. The columns with the names "probeset_name" and "sequence" contains this.

Author(s)

Lasse Folkersen

See Also

readPgf, getLocalProbeIntensities, plotOnGene

Examples

```
## Not run:
#must supply pgf file for this to work
getProbeLevelAnnotationForExonArrays("43254543", pgfPath="~/somewhere/some.pgf")

## End(Not run)
```

---

getProbesetsFromMetaprobeset

Get Probeset IDs From metaprobeset IDs

Description

Function that return the probesets mapping to a given set of metaprobesets.

Usage

```
getProbesetsFromMetaprobeset(annotation, metaprobesets, pythonPath=NULL, mpsToPsFile=NULL)
```

Arguments

annotation	A characther string giving the type of array for which probesets are needed. Will be used to load the .db file from bioconductor. Optional if mps to ps and transcript cluster file is given.
metaprobesets	A vector of characters giving the metaprobeset IDs for which to find the probe set IDs.
pythonPath	Optional character string with the path for Python software. This is only needed for exon arrays. If Python is in path this will be recognised automatically. Python can be downloaded from http://www.python.org.

`mpsToPsFile` The location of a transcript cluster file such as HuEx-1_0-st-v2.r2.dt1.hg18.full.mps. These can be downloaded from http://www.affymetrix.com. Best to use the "full" type file.

Details

This function will parse the relevant library files from the http://www.affymetrix.com website using a python script. Admittedly this is not optional from an R-only point-of-view, but it works and its fast. The function will be updated when more R-centric ways of parsing exon array annotation data is available.

Alternatively this data can just as well be retrieved from the web, but in some cases this function is faster an easier.

Value

A list of all probesets found in the given metaprobesets.

Author(s)

Lasse Folkersen

See Also

`getLocalProbeIntensities`, `getProbesetsFromRegionOfInterest`, `getMetaprobesetsFromRegionOfInterest`

Examples

```
## Not run:
#must supply mpsToPsFile and transcriptClustersFile for this to work
probesets<-getProbesetsFromMetaprobeset("notusedhere", c("3218528","2423669"), transcriptClustersFile=trans

## End(Not run)
```

---

getProbesetsFromRegionOfInterest

Get Probeset IDs From Region Of Interest

Description

Function that return the probesets located within a given region of the genome.

Usage

```
getProbesetsFromRegionOfInterest(annotation, chromosome, start, end, pythonPath=NULL, transcrip
```

Arguments

annotation	A characther string giving the type of array for which probesets are needed. Will be used to load the .db file from bioconductor. Optional if mps to ps and transcript cluster file is given.
chromosome	The chromosome of interest. Should be given as a character string of the type "Chr1", "Chr2", "ChrY".
start	A character string with the start location of interest in the chromosome.
end	A character string with the end location of interest in the chromosome.
pythonPath	Optional character string with the path for Python software. This is only needed for exon arrays. If Python is in path this will be recognised automatically. Python can be downloaded from http://www.python.org.
transcriptClustersFile	The location of a transcript cluster file such as HuEx-1_0-st-v2.na26.hg18.transcript.csv. These can be downloaded from http://www.affymetrix.com.
mpsToPsFile	The location of a transcript cluster file such as HuEx-1_0-st-v2.r2.dt1.hg18.full.mps. These can be downloaded from http://www.affymetrix.com. Best to use the "full" type file.

Details

While working with 3'IVT type affymetrix arrays, for which .db files exist within the bioconductor environment, this function works in a pretty simple way. However if exon arrays are used, it will change to parsing the relevant library files from the http://www.affymetrix.com website using a python script. Admittedly this is not optional from an R-only point-of-view, but it works and its fast. The function will be updated when more R-centric ways of parsing exon array annotation data is available.

Alternatively this data can just as well be retrieved from the web, but in some cases this function is faster an easier.

Value

A list of all probesets found in the given range on the given chromosome.

Author(s)

Lasse Folkersen

See Also

getLocalProbeIntensities, getMetaprobesetsFromRegionOfInterest

Examples

```
## Not run:
#must supply mpsToPsFile and transcriptClustersFile for this to work
probesets<-getProbesetsFromRegionOfInterest("notusedhere", chromosome=2, start="215889955", end="216106710"

## End(Not run)
```

`getSequence` *Get Sequence from a ProbeLevelSet*

Description

Function to retrieve the sequences of feature in a ProbeLevelSet.

Usage

```
getSequence(object, id)
```

Arguments

`object`	A ProbeLevelSet object
`id`	Optional character vector with the featureNames of the sequences of interest

Value

A character vector with the sequences of all probes in the ProbeLevelSet. The names of the vector are set to match the sequences.

Author(s)

Diego Diez

See Also

`ProbeLevelSet`

Examples

```
data(exampleProbeLevelSet)
getSequence(exampleProbeLevelSet)
```

`getServerProbeIntensities`
Get Probe Intensities from a server

Description

Wrapper around getLocalProbeIntensities that is designed to be run easily on remote computers.

Usage

```
getServerProbeIntensities(listOfProbesets, celfilePath, annotation=NULL, aptCelExtractPath=NUL
```

Arguments

listOfProbesets	Either a character vector with the names of the probesets from which probes should be included in the ProbeLevelSet, or the path name of a file containing this
celfilePath	The path to a folder that contains the cel files of interest. If serveradress is different from "localhost", this should be the path on the remote computer
annotation	Character string specifying which type of arrays is investigated. If a pgf file and clf file is specified this is optional.
aptCelExtractPath	The path to the apt-cel-extract file from the Affymetrix Power Tools package (including the filename itself). You can try to leave it as NULL and see what happens. If the tool is in path, it will work anyway. If you have specified the argument before it will be remembered. If you are using an OS for which executables have been included in the package (win32 and linux64), it will default to using that.
pgfPath	The path to a pgf file for the exon array of interest. This argument is mutually exclusive with the cdfPath argument. These files can be downloaded from www.affymetrix.com for the array of interest. Once given for a particular annotation, the location is saved for future use and can be given as NULL next time.
clfPath	The path to a clf file for the exon array of interest. This argument is mutually exclusive with the cdfPath argument. These files can be downloaded from www.affymetrix.com for the array of interest. Once given for a particular annotation, the location is saved for future use and can be given as NULL next time.
cdfPath	The path to a cdf file for the array of interest. This argument is mutually exclusive with the pgfPath and clfPath arguments. These files can be downloaded from www.affymetrix.com for the array of interest. Once given for a particular annotation, the location is saved for future use and can be given as NULL next time.
serveradress	Character string with the IP adress to a remote server. Defaults to "localhost", in which case the algorithm is run locally.
username	The username for the remote server. Optional if serveradress is "localhost".
password	The password for the remote server. Optional if serveradress is "localhost".
plinkPath	The path to the plink program. Can be found at http://www.chiark.greenend.org.uk/~sgtatham/putty/d Used to transfer commands to the remote server. Optional if serveradress is "localhost".
pscpPath	The path to the pscp program. Can be found at http://www.chiark.greenend.org.uk/~sgtatham/putty/d Used to transfer commands to the remote server. Optional if serveradress is "localhost".
verbose	TRUE or FALSE.

Details

This function is a wrapper around getLocalProbeIntensities. The main function of the getServerProbeIntensities function is to save all the data needed to start a getLocalProbeIntensities run, send it to a remote server, start the run, and return the data. This is useful if you have access to a fast server and want to work with large datasets of exon arrays.

Value

A ProbeLevelSet with probe quantiles normalized intensity values for all probes in the specified probesets. The ProbeLevelSet inherits the ExpressionSet, but in addition it has the sequence of each probe stored in the featureData of the set.

Author(s)

Lasse Folkersen

See Also

getLocalProbeIntensities, plotOnGene

Examples

```
## Not run:
#must correct paths, username and password before this example will work
listOfProbesets<-c("10321_at")
celfilePath<-path_to_some_cel_files #remote
aptCelExtractPath<-"~/apt/apt-cel-extract" #remote
cdfPath<-"~/hgu133plus2.cdf" #remote
username<-"user1"
password<-"ZaphodBeeblebrox"
plinkPath<-"~/plink" #local
pscpPath<-"~/pscp" #local
getServerProbeIntensities(listOfProbesets, celfilePath, annotation="hgu133plus2", aptCelExtractPath=aptCelE
username=username, password=password, plinkPath=plinkPath, pscpPath=pscpPath)


## End(Not run)
```

---

mrna *mRNA sequence of gene in ProbeLevelSet*

Description

This is the mRNA sequence for fibronectin 1 isoform 2 preproprotein, which is used in the exampleProbeLevelSet.

Usage

```
data(exampleProbeLevelSet)
```

Format

It has been read from FASTA format using the readDNAStringSet function.

Author(s)

Lasse Folkersen

Nondocumented-objects *Here goes every undocumented function.*

Description

Every function without help page is redirected here. Either because it is not intended to have a help page or because it is a bug. If you think it is the later please contact the package maintainer.

Author(s)

Lasse Folkersen and Diego Diez

plotCoexpression *Plot Coexpression of probes in a ProbeLevelSet*

Description

Function that will investigate all possible pairings in a set of probes, calculate the Pearson correlation coefficient and plot them in a meaningful way

Usage

```
plotCoexpression(object, gene, probeData=NULL, verbose=TRUE, directions="all", correlationCutof
  probeLevelInfo=c("probeid"))
```

Arguments

object	A ProbeLevelSet object or a regular ExpressionSet object (in which case a probeData argument is required). See getLocalProbeIntensities and related functions on how to create a ProbeLevelSet.
gene	A number of gene sequences as DNAstring, DNAStringSets or character vectors with sequence.
probeData	Optional if a ProbeLevelSet is submitted as object argument. Otherwise it must be a data frame with rownames corresponding to the featureNames of the ExpressionSet and a column named "sequence" with the probe sequences as character strings
verbose	TRUE or FALSE
directions	A character vector of the matching-directions that should be scanned (which combinations of complementary and reverse). Defaults to "all" which is shorthand for all possible directions, but can take anything from: c("matchForwardSense", "matchForwardAntisense", "matchReverseSense", "matchReverseAntisense")
correlationCutoff	A number between 0 and 1. The limit at which Pearson correlation (in absolute values) should not be plotted below. Defaults to 0.5
probeLevelInfo	The information about each probe to include in the plot. Should be a vector of one or more of the following elements: probeid, probesetid, sequence. Default is only probeid.

Details

This function takes a ProbeLevelSet or an ExpressionSet + probeData and the sequence of a gene. It then calculates pairwise Pearson correlation coefficients between all possible combinations of probes. Then it assigns each probe to a location along the length of the gene and plots a relational graph showing which probes has high correlation coefficients. The correlation coefficients are sorted by absolute values meaning that it will also include the negative correlations.

Value

No value, but plots a hapmap style plot of correlation values between all probes

Author(s)

Lasse Folkersen

See Also

geneRegionScan, plotOnGene

Examples

```
data(exampleProbeLevelSet)
  plotCoexpression(exampleProbeLevelSet, mrna, correlationCutoff=0.7, probeLevelInfo=c("probeid","sequence"
```

---

plotOnGene *Plot probe level data on a gene*

Description

Function that will investigate the probe level intensity of probes as a function of their location in a gene.

Usage

```
plotOnGene(object, gene, probeData=NULL, label=NULL, genename=NULL, summaryType="median",
interval=NULL, ylim=NULL, testType=NULL, forcePValue=FALSE, verbose=TRUE, cutoff=0.2, direction
```

Arguments

object	A ProbeLevelSet object or a regular ExpressionSet object (in which case a probeData argument is required). See getLocalProbeIntensities and related functions on how to create a ProbeLevelSet.
gene	A number of gene sequences as DNAstring, DNAStringSets, or character-vectors with sequence.
probeData	Optional if a ProbeLevelSet is submitted as object argument. Otherwise, it must be a data frame with rownames corresponding to the featureNames of the ExpressionSet and a column named "sequence" with the probe sequences as character strings

label	An optional character string specifying a column name in the pData of the object. If this argument is given, the gene plot will be colour coded based on the different groups (factors) in the pData entry. If a summaryType other than 'dots' is selected the summarisation is done stratified by the different groups in the pData. It can be a numeric or integer entry, but it will be coerced to factors.
genename	Optional character string specifying a gene name to include in the plot. If not included and a FASTA sequence is given, it will default to the name in the FASTA sequence. Otherwise, it will default to 'Unknown genename'.
summaryType	Character string specifying one of the following summary methods: 'median', 'mean', 'quartiles' or 'dots' (i.e. no summary). Specifies how all the sample values or all the samples values in a group if 'label' is given, should be summarised. Defaults to 'median'.
interval	Optional vector of two integers of bp positions. If given, the plot will only include the sequence from gene in the given interval. The x-axis annotation is preserved from original, so this is useful for zooming on specific regions.
ylim	Optional two integers. If given, this value will be the minimal and maximal value on the y-axis. This is useful if a few outlier probes have very high intensity values, as the default is to set the ylim from the maximal intensity value.
testType	Optional character string, defining a statistic procedure to identify especially interesting probes. Can be either 'linear model', 'students' or 'wilcoxons'. If given, a label must also be specified. In this case the plotStatistics function will be called and probes that are significantly changed between the groups in label at the P-value set in cutoff (see cutoff argument) will be circled.
forcePValue	Logical. Is used if the testType argument is used. If TRUE all significantly changed probes have P-value given on the plot. If FALSE, only plots with less than 10 significant probes write P-values. Plots can become very cluttered with data if set to TRUE
verbose	TRUE or FALSE
cutoff	Integer specifying at what p-value probes should be circled when using the 'testType' variable. Defaults to 0.2. For cutoffs higher than 0.05, all probes with P >0.05 will be circled in grey instead of black.
directions	A character vector of the matching-directions that should be scanned (which combinations of complementary and reverse). Defaults to "all" which is shorthand for all possible directions, but can take anything from: c("matchForwardSense", "matchForwardAntisense", "matchReverseSense", "matchReverseAntisense")
ylab	Label of Y-axis as in default plots

Details

At the very least, this function takes a ProbeLevelSet or an ExpressionSet + probeData and the sequence of a gene. It then compares the probe sequences given in the ProbeLevelSet or the probeData variable with the sequence of the gene given. Any probes with sequences found in the gene will be plotted, with their intensity level on the y-axis and their location in the gene on the x-axis. If no further arguments are given this gives a view of relative expression levels along the length of the gene, and can be used to investigate which exons are actively transcribed in the sample and which are not. An important argument that can be used for further investigation is the 'label' argument which specifies a column in the pData of the ExpressionSet. In this case the plots will be stratified by the factors specified in this column (so giving labels with numerical or Date class data will not work). This can be very useful when investigating how different sample conditions affect various

regions of a gene. A transcript isoform that is relatively upregulated in a diseased state will for example not be discovered if a probeset or metaprobeset covering the entire gene is used to summarize the data, since the average expression intensity for the gene will remain constant. Using the plotOnGene function, however, and specifying a case / control label will reveal tendencies for probes at certain exon locations to have relation to this label. The testType argument further supports this functionality by providing statistical testing and highlighting of probes that correlate significantly to the given label. In the case / control example, a student's t-test would highlight all probes that matched with the exons of the gene that was only found in the disease-specific transcript isoform. When interpreting the data it is suggested that specific attention is paid to the pattern of probes in the same exon. A single probe with a P-value < 0.05 might be a false positive caused by chance or by cross hybridization of the probe sequence to something else. A range of probes in the same exon that all show P-values below or close to 0.05, however, is much more likely to be an actual case of a transcript isoform having this particular exon or exons being regulated between the groups in the label. Exon structure can be easily plotted on the graph using the exonStructure function.

A special case is the search for SNPs which have effect on expression levels or variable splicing. The testType argument 'linear model' is designed for this. The linear model calls the internal function doProbeLinear which assign each of the levels in the 'label' column of the pdata a value between 1 and the number of levels, in the order in which they are sorted. For genotypes given as "AA", "AB", "BB" character strings this will give "AA" = 1, "AB" = 2 and "BB" = 3. The doProbeLinear then calculates a linear model between the intensity values and these numbers, and returns the P-value. In this case, low P-values can be interpreted as a case where the heterozygote samples have intermediary expression levels between the two homozygotes. This is the case that can be expected to be seen if the nucleotide type of SNP does in fact have any effect on the mRNA concentration levels in the sample.

Value

No value, but plots the local expression levels relations of each probe found in the submitted gene sequence as a function of its location along this sequence. Various statistics and summarizations on pdata can be employed, as specified in details.

Author(s)

Lasse Folkersen

See Also

geneRegionScan, plotCoexpression

Examples

```
data(exampleProbeLevelSet)

plotOnGene(exampleProbeLevelSet, mrna, summaryType="dots", interval=c(500,1000))

plotOnGene(exampleProbeLevelSet, mrna, label="genotype3", testType="linear model")
exonStructure(mrna, genomic)
```

plotStatistics *Plot Statistics*

Description

Internal function that will assist plotOnGene in identifying probes that are significantly changed in comparison with a given pData column.

Usage

```
plotStatistics(object, probeData, label, summaryType, testType, interval=NULL,
  forcePValue=FALSE, verbose=TRUE, positionVector, ylim, xlim, cutoff=0.2)
```

Arguments

object	A ProbeLevelSet object or a regular ExpressionSet object (in which case a probeData argument is required.
probeData	Optional if a ProbeLevelSet is submitted as object argument. Otherwise it must be a data frame with rownames corresponding to the featureNames of the ExpressionSet and a column named "sequence" with the probe sequences as character strings
label	An optional character string specifying a column name in the pData of the object. If this argument is given, the gene plot will be colour coded based on the different groups (factors) in the pData entry. If a summaryType other than 'dots' is selected the summarisation is done stratified by the different groups in the pData.
summaryType	Character string specifying one of the following summary methods: 'median', 'mean', 'quartiles' or 'dots' (i.e. no summary). Specifies how all the sample values or all the samples values in a group if 'label' is given, should be summarised. Defaults to 'median'.
testType	Character string, defining a statistic procedure to identify especially interesting probes.
interval	Optional vector of two integers of bp positions. If given, the plot will only include the sequence from gene in the given interval. The x-axis annotation is preserved from original, so this is useful for zooming on specific regions.
forcePValue	Logical. Is used if the testType argument is used. If TRUE all significantly changed probes have P-value given on the plot. If FALSE, only plots with less than 10 probes significant write P-values. Plots can become very cluttered with data if set to TRUE
verbose	TRUE or FALSE
positionVector	A vector of integers and names specifying the position along the x-axis of each probe.
ylim	A vector of two integers. Specifies the decided ylim value passed into the plotting function.
xlim	A vector of two integers. Specifies the decided xlim value passed into the plotting function.
cutoff	Integer specifying at what p-value probes should be circled when using the 'testType' variable. Defaults to 0.2. For cutoffs higher than 0.05, all probes with P >0.05 will be circled in grey instead of black.

Details

This function is not meant to be called by the user. It acts as an adapter between plotOnGene and either doProbeLinear or doProbeTTest depending on the testType variable.

Value

No value, but decorates an existing plotOnGene plot with circles and p-values for all probes that are significant when investigating label with testType

Author(s)

Lasse Folkersen

See Also

doProbeLinear, doProbeTTest, plotOnGene

ProbeLevelSet-class *Class ProbeLevelSet*

Description

This is the storage class for Probe Level data with attached probe annotation

Class definition

The class is derived from class ExpressionSet in the Biobase package.

Author(s)

Diego Diez

---

readGeneInput *Standardize reading of gene inputs*

Description

Internal function that will standardise the input of the many different form of sequences that can be used in Bioconductor.

Usage

```
readGeneInput(gene, genename=NULL, verbose=TRUE)
```

Arguments

gene	A number of gene sequences as DNAString, DNAStringSets or, character-vectors with sequence.
genename	Optional character string specifying a gene name to include in the plot. If not included and a FASTA sequence is given, it will default to the name in the FASTA sequence. Otherwise it will default to 'Unknown genename'.
verbose	TRUE or FALSE.

Details

This function is not meant to be run directly by the user. It will take a number of genes either as a vector of characters, a path to a FASTA format file, as a DNAstrings or DNAStringSet. It will then output them in FASTA format for use with other functions. Optional variable genename forces a new name.

The primary objective of this function is to make the sequence input simpler for other functions.

Value

A list of all sequences and their names, in exactly the same format as obtained with the deprecated function readFASTA.

Author(s)

Lasse Folkersen

See Also

geneRegionScan, plotOnGene

Examples

```
somegene<-"ATACCTTGTAGGACCTGATGATAGATGCATAGTAATATCGTA"
genename<-"My favourite gene"
GeneRegionScan:::readGeneInput(somegene,genename=genename)
```

---

translateSampleNames *Translate sample names using translation file*

Description

Function that will change sample names in an ExpressionSet using a translation file.

Usage

```
translateSampleNames(object, translationFile, from, to)
```

Arguments

object	An ExpressionSet
translationFile	Character string with the path to a tab-separated text file with translations of names. Alternatively a data frame containing translation information.
from	Character string with the translationFile column name containing the values to translate from.
to	Character string with the translationFile column name containing the values to translate to.

Details

Function that can translate the sampleNames of an ExpressionSet when given translation information. The translationFile argument gives the translation between sampleNames as they are and the sampleNames as they should be. It can either be a character string with the path to a tab-separated text-file with headers or a directly a data frame with the necessary information. The variables 'from' and 'to' specify the column names that contains the translation information. They must be present in the translationFile and the entries in 'from' must be present in the sampleNames of the current ExpressionSet. Extra entries in the translationFile are omitted but samples in the ExpressionSet without samples in the translationFile will raise an error.

This package is particularly useful after running cel-files through the Affymetrix Power Tools (with getLocalProbeIntensities for example), since this program has a tendency to change any odd character in the filename to something else.

Value

The same ExpressionSet given as argument, but with sampleNames changed as specified in translationFile

Author(s)

Lasse Folkersen

Examples

```
data(exampleProbeLevelSet)

fromThese<-sampleNames(exampleProbeLevelSet)

toThese<-sub(".cel","", sampleNames(exampleProbeLevelSet))
toThese<-sub("X","", toThese)


translationFile<-cbind(fromThese, toThese)
translationFile<-as.data.frame(translationFile)

translateSampleNames(exampleProbeLevelSet, translationFile, from="fromThese", to="toThese")
```

Index