affypdnn

April 19, 2010

`hgu133a.pdnn.params`
Chip-type specific data

Description

Chip-type specific data structure.

Usage

```
data(hgu95av2.pdnn.params)
```

Format

The format is a list:

Eg environment (one entry per dinucleotide)

Wg numerical vector.

En environment (one entry per dinucleotide)

Wn numerical vector.

gene.Sn list (one entry per affyID)

gene.Sg list (one entry per affyID)

gene.xy list (one entry per affyID)

params.gene environment

Details

These chip-specific data structures are generated from the data files made available by the author of the PDNN paper (see the section source). They are stored as `data` to save some computation time. The data structures were made using the function `pdnn.params.chiptype`. The data files are included in the the directory 'data' of the package.

Note

To lower the size of the package, the only chip-specific data structures included in the package is the one for HG-U95Av2.

Source

Li Zhang, Michael F. Miles and Kenneth D. Aldape - A model of molecular interactions on short oligonucleotide arrays, 2003, Nature Biotech., vol. 21, n.7

Examples

```
## give the path the original energy parameter files included in the package
list.files(system.file("exampleData", package="affypdnn"),
           "^pdnn-energy-parameter_", full.names=TRUE)
```

`expressopdnn` *Position Dependant Nearest Neighbors model for affy*

Description

A wrapper to perform the PDNN method.

Usage

```
pdnn.scalevalue.exprSet(eset, scale.to=500)
expressopdnn(abatch,
      # background correction
             bg.correct = FALSE,
             bgcorrect.method = NULL,
             bgcorrect.param = list(),
      # normalize
             normalize = FALSE,
             normalize.method = NULL,
             normalize.param = list(),

              pmcorrect.method = c("pdnn", "pdnnpredict"),

      # pdnn
             findparams.param = list(),
      # expression values
             summary.subset = NULL,
      # PDNN expression values scaling
             eset.normalize = TRUE,
             scale.to = 500,
      # misc.
             verbose = TRUE)
```

Arguments

`abatch`	object of `AffyBatch-class`.
`bg.correct`	a boolean to express whether background correction is wanted or not.
`bgcorrect.method`	the name of the background adjustment method.
`bgcorrect.param`	a list of parameters for bgcorrect.method (if needed/wanted).

eset — an object of ExpressionSet-class.

normalize — normalization step wished or not.

normalize.method — the normalization method to use.

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

pmcorrect.method — the name of the PM adjustement method (only two choices here, default to 'pdnn').

findparams.param — a list of parameters to be passed to find.params.pdnn.

eset.normalize — is any normalization step on expression values to be performed.

scale.to — a value to scale against.

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

verbose — logical value. If TRUE it writes out some messages.

Details

expressopdnn is very similar to expresso. It is mainly a wrapper around the pre-processing steps 'background correction', 'normalization', 'perfect match correction' and the PDNN method to compute expression values (see the first reference for more details about the preprocessing steps and and the second reference for further details about the PDNN method).

The wrapper expresso has no way to handle easily the computation of chip-wide results that have to be used during the computeExprSet step. An easy way to overcome this was to write this simple wrapper.

pdnn.scalevalue is performed after the expression values have computed to somehow 'normalize' the values between different chips. When setting normalize to TRUE this step might be considered unnecessary (and the eset.normalize set to FALSE).

Value

An object of ExpressionSet-class, with an attribute pps.warnings as returned by the method computeExprSet.

See Also

expresso and generateExprVal.method.pdnn

Examples

```
## load pre-computed parameters
data(hgu95av2.pdnn.params)

library(affydata)
data(Dilution)

## one CEL to go faster
afbatch <- Dilution[, 1]
```

```
## Take only few IDs (the 10 first)
ids <- ls(getCdfInfo(afbatch))[1:10]
eset <- expressopdnn(afbatch, bg.correct=FALSE,
                     normalize=FALSE,
                     findparams.param=list(params.chiptype=hgu95av2.pdnn.params,
                                           give.warnings=FALSE),
                     summary.subset=ids)
```

`find.params.pdnn` *A function to find the experiment specific PDNN parameters*

Description

A function to find the paramaters specific to the chips in an AffyBatch object.

Usage

```
find.params.pdnn(abatch, params.chiptype, optim.method = "BFGS", verbose = TRUE,
```

Arguments

`abatch`	an instance of `AffyBatch-class`.
`params.chiptype`	chip-type specific parameters (see details)
`optim.method`	method for the optimization function `optim`. When `FALSE`, a steepest-descent method of our own is used.
`verbose`	verbosity (`TRUE` or `FALSE`)
`give.warnings`	report probeset IDs in the `abatch` that could not be found in the `params.chiptype`

Details

This function fits PDNN parameters that are specific to experimental values. The parameters common to all the chips of a certain type are returned by the function `pdnn.params.chiptype`. If `NULL`, the parameter files included in the package will be used whenever possible...

Value

A list of

`lambda`	The lambda's
`Bs`	The B's
`Ns`	The N's
`Fs`	The F's

References

Li Zhang, Michael F. Miles and Kenneth D. Aldape - A model of molecular interactions on short oligonucleotide arrays, 2003, Nature Biotech., vol. 21, n.7

See Also

pdnn.params.chiptype, generateExprVal.method.pdnn

Examples

```
## load a chip-specific parameter file
## (as returned by the function pdnn.params.chiptype)
data(hgu95av2.pdnn.params)

## load experimental data
library(affydata)
data(Dilution)

## one CEL to go faster
afbatch <- Dilution[, 1]
params <- find.params.pdnn(afbatch, hgu95av2.pdnn.params, optim.method =
FALSE, give.warnings=FALSE)
```

---

generateExprVal.method.pdnn

Compute PM correction and summary expression value

Description

Computes PM correction and summary expression value with PDNN method.

Usage

```
pmcorrect.pdnn(object, params, gene=NULL, gene.i=NULL,
               params.chiptype=NULL, outlierlim=3, callingFromExpresso=FALSE)
pmcorrect.pdnnpredict(object, params, gene=NULL, gene.i=NULL,
               params.chiptype=NULL, outlierlim=3, callingFromExpresso=FALSE)
generateExprVal.method.pdnn(probes, params)
```

Arguments

object	object of ProbeSet.
probes	matrix of PM-corrected signals (should be coming out of pmcorrect.pdnn).
params	experiments specific parameters.
gene	gene (probe set) ID (from wich the gene.i would be derived).
gene.i	gene index (see details).
params.chiptype	chip-specific parameters.
outlierlim	threshold for tagging a probe as an outlier.
callingFromExpresso	is the function called through expresso. DO NOT play with that.

Details

Only one of `gene`, `gene.i` should be specified. For most the users, this is `gene`. `pmcorrect.pdnn` and `pmcorrect.pdnnpredict` return what is called GSB and GSB + NSB + B in the paper by Zhang Li and collaborators.

Value

`pmcorrect.pdnn` and `pmcorrect.pdnnpredict` return a matrix (one row per probe, one column per chip) with attributes attached. `generateExprVal` returns a `list`:

`exprs`	expression values
`se.exprs`	se expr. val.

See Also

`pdnn.params.chiptype`

Examples

```
data(hgu95av2.pdnn.params)
library(affydata)
data(Dilution)

## only one CEL to go faster
abatch <- Dilution[, 1]

## get the chip specific parameters
params <- find.params.pdnn(abatch, hgu95av2.pdnn.params)

## The thrill part: do we get like in the Figure 1-a of the reference ?
par(mfrow=c(2,2))
##ppset.name <- sample(featureNames(abatch), 2)
ppset.name <- c("41206_r_at", "31620_at")
ppset <- probeset(abatch, ppset.name)
for (i in 1:2) {
  ##ppset[[i]] <- transform(ppset[[i]], fun=log) # take the log as they do
  probes.pdnn <- pmcorrect.pdnnpredict(ppset[[i]], params,
                                       params.chiptype=hgu95av2.pdnn.params)
  ##probes.pdnn <- log(probes.pdnn)
  plot(ppset[[i]], main=paste(ppset.name[i], "\n(raw intensities)"))
  matplotProbesPDNN(probes.pdnn, main=paste(ppset.name[i], "\n(predicted intensities)"))
}

## pick the 50 first probeset IDs
## (to go faster)
ids <- featureNames(abatch)[1:100]

## compute the expression set (object of class 'ExpressionSet')
eset <- computeExprSet(abatch, pmcorrect.method="pdnn",
                       summary.method="pdnn", ids=ids,
                       summary.param = list(params, params.chiptype=hgu95av2.pdnn.params)
```

matplotProbesPDNN *Plot the PDNN computed probe intensities*

Description

Plot the probe intensities as computed by 'pmcorrect.pdnn' or 'pmcorrect.pdnnpredict'

Usage

```
matplotProbesPDNN(x, type="l", ...)
```

Arguments

x	a matrix (and attributes) as returned by pmcorrect.pdnn or pmcorrect.pdnnpredict.
type	type of plot (same as in matplot)
...	optional arguments to be passed to matplot

Details

The crosses are the probe intensities which are considered 'ok' by the outlier detection part of the algorithm, while the circles are the ones considered 'outliers'

Value

Only used for its side-effect.

See Also

pmcorrect.pdnn and pmcorrect.pdnnpredict

Examples

```
# see 'pmcorrect.pdnn'
```

params.dilution *Parameters for the Dilution dataset*

Description

PDNN parameters for the Dilution dataset

Usage

```
data(params.dilution)
```

Format

The format is: List of 6 $ lambda :List of 12625 (probesets) ... $ Bs : num [1:4] 111.9 57.3 120.5 50.1 $ Ns : num [1:4] 2967 2998 2992 2999 $ Fs : num [1:4] 0.607 0.662 0.600 0.656 $ names.abatch: chr [1:12625] "1000_at" "1001_at" "1002_f_at" ... $ names.i : int [1:12625] 1 2 3 4 5 6 7 8 9 10 ...

Details

These data are provided to reduce the time needed to run the vignette.

Examples

```
data(params.dilution)
```

`pdnn.params.chiptype` *A function to fit PDNN parameters*

Description

A function to fit PDNN parameters that are chip-type specific

Usage

```
pdnn.params.chiptype(energy.param.file, probes.file = NULL, probes.pack= NULL,
                     probes.data.frame = NULL,
                     seq.name, x.name, y.name, affyid.name, verbose = TRUE)
```

Arguments

`energy.param.file`	Path to the energy data file (see details)
`probes.file`	Path to the probe files (see details)
`probes.pack`	Name of the probe pack (see details)
`probes.data.frame`	A `data.frame`
`seq.name, x.name, y.name, affyid.name`	The names of the columns in the data.frame from `probes.pack` or `probes.file` for the probe sequences, the X positions, the Y positions and the probe set ID respectively
`verbose`	verbosity (`TRUE` or `FALSE`)

Details

The parameters `probes.file`, `probes.pack` and `probes.data.frame` are mutually exclusive. The function fits PDNN parameters that are specific to chip-types (hence specific to the probe sequences). It requires data files like the one found on Li Zhang's web page: (http://odin.mdacc.tmc.edu/~zhangli/P
This should be computed once for all for a given chip type. Computed values for the chips are included in the package. This allows 'automagic' use of them when these chips types are used (as done in the function `expressopdnn`).

Value

A list of:

Eg	environment. One entry per dinucleotide.
Wg	numerical vector
En	environment. One entry per dinucleotide.
Wn	numerical vector
params.gene	environment. One entry per gene, each entry is is a list of elements Sg, Sn, xy and gene.i

warning

The X and Y positions in the data.frame are expected to be original ones in the Affymetrix files (starting at zero. They are offset by one within this function.

See Also

find.params.pdnn

Examples

```
if (interactive()) {
  energy.file <- system.file("exampleData", "pdnn-energy-parameter_hg-u95av2.txt", packag

  params.chiptype <- pdnn.params.chiptype(energy.file, probes.pack="hgu95av2probe")
}
```

---

transform.ProbeSet *A function to transform a ProbeSet*

Description

A function to transform the PMs and MMs in a ProbeSet.

Usage

```
## S3 method for class 'ProbeSet':
transform(`_data`, fun = I, ...)
```

Arguments

_data	object of ProbeSet-class.
fun	a function. The identity function by default.
...	optional arguments for fun.

Details

The function fun is applied to the slots pm and mm. The function vsnh in the package vsn is a recommended argument for fun.

Value

An object of class `ProbeSet`.

Note

This function should make to the package `affy` for the version 1.4.x.

Author(s)

Laurent

Examples

```
library(affydata)

data(Dilution)

ppset.name <- sample(featureNames(Dilution), 1)
ppset <- probeset(Dilution, ppset.name)[[1]]
ppset.log <- transform(ppset, fun=log)

par(mfrow=c(1,2))
plot(ppset)
plot(ppset.log)
```

Index