Package ‘altcdfenvs’

February 2, 2024

Version 2.64.0
Title alternative CDF environments (aka probeset mappings)
Author Laurent Gautier <lgautier@gmail.com>
Maintainer Laurent Gautier <lgautier@gmail.com>
biocViews Microarray, OneChannel, QualityControl, Preprocessing, Annotation, ProprietaryPlatforms, Transcription
Depends R (>= 2.7), methods, BiocGenerics (>= 0.1.0), S4Vectors (>= 0.9.25), Biobase (>= 2.15.1), affy, makecdfenv, Biostrings, hypergraph
Suggests plasmodiumanophelescdf, hgu95acdf, hgu133aprobe, hgu133a.db, hgu133acdf, Rgraphviz, RColorBrewer
Description Convenience data structures and functions to handle cdfenvs
License GPL (>= 2)
LazyLoad yes
git_url https://git.bioconductor.org/packages/altcdfenvs
git_branch RELEASE_3_18
git_last_commit 86cec42
git_last_commit_date 2023-10-24
Repository Bioconductor 3.18
Date/Publication 2024-02-02

R topics documented:

AffyProbesMatch-class .................................................. 2
appendCdfEnvAffy ........................................................ 3
buildCdfEnv.biostrings ................................................. 4
AffyProbesMatch-class

Description

Store the results of a call to matchAffyProbes.

Objects from the Class

Objects can be created by calls of the form `new("AffyProbesMatch", ...)`. An object will store the result of matching probe sequences against target sequences.

Slots

- pm: Object of class "list": each element is vector of index values
- mm: Object of class "list": each element is vector of index values
- labels: Object of class "character"
- chip_type: Object of class "character" and of length 1.
- probes: Object of class "ANY": the probetable object used to perform the matches.

Methods

- `combine` signature(x = "AffyProbesMatch", y = "AffyProbesMatch"): combine two instances. This can be useful when splitting the list of target sequences to parallelized the job.
- `show` signature(x = "AffyProbesMatch"): Show the instance.
- `toHypergraph` signature(object = "AffyProbesMatch"): build an Hypergraph from the matches.
appendCdfEnvAffy

Examples

showClass("AffyProbesMatch")

appendCdfEnvAffy

append probe sets to a CdfEnvAffy

Description

append probe sets to a CdfEnvAffy

Usage

appendCdfEnvAffy(acdfenv, id, i, nocopy = TRUE)

Arguments

acdfenv instance of class CdfEnvAffy
id identifier for the probe set to add
i a matrix of indexes (see details)
nocopy whether to make a copy of the environment or not (see details)

Details

The matrix i must have one column per probe type. For typical Affymetrix chip types, there are two probe types: "pm" and "mm".

nocopy set to TRUE means that the environment is added the probe set 'in-situ' (this can boost execution speed if you add a lot of probe sets).

Value

An CdfEnvAffy is returned

Examples

data(cdfenvEx)

## pm and mm probe set
m <- matrix(1:10, ncol = 2)
colnames(m) <- c("pm", "mm")
appendCdfEnvAffy(cdfenvEx, "blabla", m)
indexProbes(cdfenvEx, c("pm", "mm"), "blabla")

## pm only probe set
m <- matrix(6:9, ncol = 1)
colnames(m) <- c("pm")
buildCdfEnv.biostrings

Build CDF environments

Description
Build CDF environment from Biostrings matchPDict results

Usage
buildCdfEnv.biostrings(apm, abatch = NULL, nrow.chip = NULL, ncol.chip = NULL, simplify = TRUE, x.colname = "x", y.colname = "y", verbose = FALSE)

Arguments
- `apm` AffyProbesMatch
- `abatch` AffyBatch
- `nrow.chip` number of rows for the chip type (see details)
- `ncol.chip` number of columns for the chip type (see details)
- `simplify` simplify the environment built (removing target names when there is no matching probe)
- `x.colname` column name
- `y.colname` column name
- `verbose` verbose TRUE/FALSE

Details
Whenever an abatch is specified, nrow.chip and ncol.chip are not needed. Specifying the an AffyBatch in abatch is the easiest way to specify information about the geometry of a chip type.

Value
An instance of class CdfEnvAffy.
Class "CdfEnvAffy"

Description

A class to hold the information necessary to handle the grouping of probes in set of probes, and to find XY coordinates of probes on a chip

Objects from the Class

Objects can be created by calls of the form `new("CdfEnvAffy", ...). Typically, there is an instance of the class for each type of chip (e.g. Hu6800, HG-U95A, etc…).

Slots

- `envir`: Object of class "environment". It has to be thought of as a hashtable: the keys are probe set identifiers, or gene names, and the values are indexes.
- `envName`: Object of class "character". A name for the environment.
- `index2xy`: Object of class "function". The function used to resolve index into xy coordinates. Unless you are an advanced user, you probably want to ignore this (and rely on the default provided with the package).
- `xy2index`: Object of class "function". The function used to resolve xy coordinates into index. Unless you are an advanced user, you probably want to ignore this (and rely on the default provided with the package).
- `nrow`: Object of class "integer". The number of rows of probes for the chip type.
- `ncol`: Object of class "integer". The number of columns of probes for the chip type.
- `probeTypes`: Object of class "character". The different types of probes stored for each probe set. In the case of Affymetrix chips, the probes are typically perfect match (pm) probes or mismatch probes (mm).
- `chipType`: Object of class "character". The name of the chip type the instance is associated with. This is useful when one starts to create alternative mappings of the probes on a chip (see associated vignette).

Methods

[ `signature(object = "CdfEnvAffy", i = "character", j = "missing", drop = "boolean")`: subset a cdf, that is return a new cdf containing only a subset of the probe sets. The subset of probe sets to take is identified as a vector of identifiers (mode "character").

- `coerce` `signature(object = "CdfEnvAffy", "environment")`: coerce an instance of the class to an environment.

- `coerce` `signature(object = "CdfEnvAffy", "Cdf")`: coerce an instance of the class to a Cdf.

- `geneNames` `signature(object="CdfEnvAffy")`: Return the names of the known probe sets (of course, it depends on the associated CDF).
index2xy signature(object = "CdfEnvAffy", i="integer"): convert index values into XY coordinates.

indexProbes signature(object = "CdfEnvAffy", which = "character", probeSetNames = NULL): obtain the indexes for the probes associated with the probe set name probeSetNames. When probeSetNames is set to NULL (default), the indexes are returned for the probe sets defined on the chip. See indexProbes.CdfEnvAffy

plot signature(x = "CdfEnvAffy", y = "missing"): Plot the chip. It mainly sets coordinates for further plotting (see examples). See plot.CdfEnvAffy

show signature(object = "CdfEnvAffy"): Print method.

xy2index signature(object = "CdfEnvAffy", x="integer", y="integer"): convert XY coordinates into index values.

toHypergraph signature(object = "CdfEnvAffy"): convert XY coordinates into index values.

Author(s)
Laurent Gautier

See Also
indexProbes.CdfEnvAffy, plot.CdfEnvAffy

Examples

```r
## build an instance
library(hgu95acdf)
cdfenv.hgu95a <- wrapCdfEnvAffy(hgu95acdf, 640, 640, "HG-U95A")
show(cdfenv.hgu95a)

## find the indexes for a probe set (pm only)
ip <- indexProbes(cdfenv.hgu95a, "pm", "1000_at")[[1]]
## get the XY coordinates for the probe set
xy <- index2xy(cdfenv.hgu95a, ip)

## plot the chip
plot(cdfenv.hgu95a)

## plot the coordinates
plotLocation(xy)

## subset the environment
cdfenv.hgu95a.mini <- cdfenv.hgu95a["1000_at"]
```
**cdfenvEx**  

**CdfEnvAffy**

---

**Description**

An example of `CdfEnvAffy`

**Usage**

```r
data(cdfenvEx)
```

**Format**

The format is: Formal class 'CdfEnvAffy' [package "altcdfenvs"] with 8 slots

- `@ index2xy`: function (object, i)
- `@ xy2index`: function (object, x, y)
- `@ envir`: length 2 <environment>
- `@ envName`: chr "ZG-DU33"
- `@ nrow`: int 100
- `@ ncol`: int 100
- `@ probeTypes`: chr [1:2] "pm" "mm"
- `@ chipType`: chr "ZG-DU33"

**Examples**

```r
data(cdfenvEx)
print(cdfenvEx)
```

---

**cdfenvs**  

**functions related to cdfenvs**

---

**Description**

A set of functions to handle cdfenvs

**Usage**

```r
wrapCdfEnvAffy(cdfenv, nrow.chip, ncol.chip, chiptype, check = TRUE, verbose = FALSE)
getCdfEnvAffy(abatch)
buildCdfEnv.matchprobes(matches, ids, probes.pack, abatch=NULL, nrow.chip=NULL, ncol.chip=NULL, chiptype=NULL, mm=NA, simplify = TRUE, x.colname = "x", y.colname = "y", verbose=FALSE)
```
Arguments

- **abatch**: an AffyBatch
- **cdfenv**: A cdfenv environment
- **check**: perform consistency check or not
- **chiptype**: A name for the chip type
- **ids**: a vector of probe set identifiers for the matches
- **matches**: a list as returned by the function `combineAffyBatch`
- **mm**: The value to store for MMs
- **ncol.chip**: The number of columns for the chip type
- **nrow.chip**: The number of rows for the chip type
- **probes.pack**: The name of the probe package
- **simplify**: Simplify the environment created by removing the ids without any matching probe
- **x.colname**, **y.colname**: see the `getxy.probeseq`
- **verbose**: verbosity (TRUE or FALSE)

Value

An instance of class `CdfEnvAffy`.

Examples

```r
## See the main vignette
```

Description

make a copy of a CdfEnvAffy

Usage

```r
copyCdfEnvAffy(acdfenv)
```

Arguments

- **acdfenv**: instance of class `CdfEnvAffy`

Details

Make a copy can be needed since a CdfEnvAffy contains an environment.
countduplicated

Value

A CdfEnvAffy

See Also

CdfEnvAffy-class, copyEnv

Description

This function counts the number of times the probes in a CdfEnvAffy are found in this object.

Usage

countduplicated(x, incomparables = FALSE, verbose = FALSE)

Arguments

x
An instance of CdfEnvAffy-class

incomparables (not implemented yet, keep away)

verbose verbose or not

Value

An environment is returned. Each element in this environment has the same identifier than its corresponding probe set in the CdfEnvAffy-class and contains the number of times a probe is in use in the environment (instead of an index number in the CdfEnvAffy-class).

Author(s)

Laurent

See Also

CdfEnvAffy-class
geneNames.CdfEnvAffy  get the names of the known probe sets

description
  get the names of the probe sets known to the CdfEnv

Usage
  geneNames.CdfEnvAffy(object)

Arguments
  object  CdfEnvAffy-class

Value
  a vector of mode character

getxy.probeseq  A function to get the XY coordinates from a probes sequences data frame

description
  A function to get the XY coordinates from a probes sequences data.frame

Usage
  getxy.probeseq(ppset.id = NULL, probeseq = NULL, i.row = NULL, xy.offset = NULL, x.colname = "x", y.colname = "y")

Arguments
  ppset.id  The probe sets of interest (a vector of mode character).
  probeseq The probe sequence data.frame (see details).
  i.row    Row indexes in the data.frame (see details).
  xy.offset Offset for the xy coordinates. if NULL, uses the default offset stored as an option for the affy package.
  x.colname, y.colname  The probe sequence packages have seen the names for the columns in their data.frame. This parameters exists to let us follow these changes.
**index2xy**

**Details**

The data.frame passed as argument `probeseq` is expected to have (at least) the following columns: `Probe.X`, `Probe.Y` and `Probe.Set.Name`. When the argument `ppset.id` is not null, the probe sets

**Value**

A matrix of two columns. The first column contains x coordinates, while the second column contains y coordinates.

**Warning**

The parameter `xy.offset.one` is here for historical reasons. This should not be touched, the option in the affy package should be modified if one wishes to modify this.

This function should not be confused with the methods `index2xy` and similar. Here the the XY coordinate come from a data.frame that stores information about an arbitrary number probes on the chip. (See the ‘probe sequence’ data packages on Bioconductor, and the package Biostrings).

The methods `index2xy` are meant to interact with instances of class `AffyBatch`.

**Author(s)**

Laurent

**Examples**

```r
##---- Should be DIRECTLY executable !! ----
index2xy Functions to shuttle from indexes to XY coordinates
```

**Description**

Functions to shuttle from indexes to XY coordinates.

**Usage**

```r
index2xy(object, ...) 
xy2index(object, ...) 
index2xy.CdfEnvAffy(object, i) 
xy2index.CdfEnvAffy(object, x, y)
```

**Arguments**

- `object`: An object of class `CdfEnvAffy`.
- `i`: A vector of indexes.
- `x, y`: Vectors of X and Y coordinates.
- `...`: Optional parameters (not used).
indexProbes.CdfEnvAffy

Value
A vector of integers (for xy2index methods), or a matrix of two columns (for index2xy methods).

See Also
CdfEnvAffy-class

Examples
## To be done...

indexProbes.CdfEnvAffy

Description
A function to get the index for probes

Usage
indexProbes.CdfEnvAffy(object, which, probeSetNames = NULL)

Arguments
object CdfEnvAffy
which which kind of probe are of interest (see details).
probeSetNames names of the probe sets of interest. If NULL, all the probe sets are considered.

Details
The parameter which let one specify which category of probes are of interest. In the case of Affymetrix chips, probes can be "pm" probes or "mm" probes. If the parameter is set to c("pm", "mm"), both are returned. Should other categories be defined, they can be handled as well.

Value
A list of indexes.

See Also
CdfEnvAffy-class, AffyBatch-class
matchAffyProbes

**Match the probes on an Affymetrix array**

**Description**

Match the individual probes on an Affymetrix array to arbitrary targets.

**Usage**

```r
mmProbes(probes)

matchAffyProbes(probes, targets, chip_type,
                 matchmm = TRUE,
                 selectMatches = function(x) which(elementNROWS(x) > 0),
                 ...)```

**Arguments**

- **probes**: a probetable object
- **targets**: a vector of references
- **chip_type**: a name for the chip type.
- **matchmm**: whether to match MM probes or not
- **selectMatches**: a function to select matches (see Details).
- **...**: further arguments to be passed to matchPDict.

**Details**

The matching is performed by the function `matchPDict`. The man page for that function will indicate what are the options it accepts.

In the case where a large number targets are given, like when each target represents a possible mRNA, it is expected to have a largely sparse incidence matrix, that is a low number of probes matching every target. For that reason, only the index of matching probes are associated with each given target, with the function `selectMatches` giving the definition of what are matching probes. The default function just count anything matching, but the user can specify a more stringent definition if wanted.

**Value**

- `mmProbes` returns a vector of MM probe sequences.
- `matchAffyProbes` returns an instance of `AffyProbesMatch-class`.

**Author(s)**

Laurent Gautier
See Also

matchPDict for details on how the matching is performed, AffyProbesMatch-class and buildCdfEnv.biostrings

Examples

library(hgu133aprobe)

filename <- system.file("exampleData", "sample.fasta",
    package="altcdfenvs")

fasta.seq <- readDNAStringSet(filename)

targets <- as.character(fasta.seq)

names(targets) <- sub("^>.+\|\(|\NM[^ \|]+|Hs[^ \|]+)\| ? .+$", "",
    names(targets))

m <- matchAffyProbes(hgu133aprobe,
    targets,
    "HG-U133A")

plot.CdfEnvAffy

A function to ‘plot’ a CdfEnvAffy

Description

A function to set the axis and plot the outline for a CdfEnvAffy

Usage

## S3 method for class 'CdfEnvAffy'
plot(x, xlab = "", ylab = "", main = x@chipType, ...)

Arguments

  x a CdfEnvAffy
  xlab label for the rows
  ylab label for the columns
  main label for the plot. The chip-type by default.
  ... optional parameters to be passed to the underlying function plot

Details

This function does not ‘plot’ much, but sets the coordinates for further plotting (see the examples).
**read.FASTA.entry**  
*Functions to work with FASTA files / connections*

**Description**

Set of functions to work with biological sequences stored in FASTA format.

**Usage**

```r
countskip.FASTA.entries(con, linebreaks = 3000)
grep.FASTA.entry(pattern, con, ...)
## S3 method for class 'FASTA'
print(x, ...)
read.FASTA.entry(con, linebreaks = 3000)
read.n.FASTA.entries(con, n, linebreaks = 3000)
read.n.FASTA.entries.split(con, n, linebreaks = 3000)
read.n.FASTA.headers(con, n, linebreaks = 3000)
read.n.FASTA.sequences(con, n, linebreaks = 3000)
skip.FASTA.entry(con, skip, linebreaks = 3000)
write.FASTA(x, file="data.fasta", append = FALSE)
```

**Arguments**

- `append` : append to the file (or not)
- `con` : a connection
- `file` : a file name
- `linebreaks` : (to optimize the parsing, probably safe to leave it as it is)
- `n` : number of entries to read
- `pattern` : a pattern (to be passed to the function grep)
- `skip` : number of entries to skip
- `x` : a FASTA sequence object
- `...` : optional arguments to be forwarded to the function `print` or to the function `grep`
Details

countskip.FASTA.entries skips the remaining FASTA entries currently remaining in the connection and return the count. grep.FASTA.entry returns the next FASTA entry in the connection that matches a given regular expression. print.FASTA prints a FASTA object. read.FASTA.entry reads the next FASTA entry in the connection. read.n.FASTA.entries reads the n next FASTA entries and returns a list of FASTA objects. read.n.FASTA.entries.split reads the n next FASTA entries and returns a list of two elements: headers and sequences. read.n.FASTA.headers reads the n next FASTA headers. read.n.FASTA.sequences reads the n next FASTA sequences. skip.FASTA.entry skips a given number of FASTA entries. write.FASTA write a FASTA object into a connection.

Value

The value returned depends on the function. See above.

Author(s)

Laurent Gautier

Examples

```r
filename <- system.file("exampleData", "sample.fasta", 
    package="altcdfenvs")
con <- file(filename, open="r")

fasta.seq <- grep.FASTA.entry("NM_001544\.2", con)
close(con)

print(fasta.seq)
```

```r
removeIndex A function to remove probes in an environment
```

Description

A function to remove probes in an environment, given their index.

Usage

```r
removeIndex(x, i, simplify = TRUE, verbose = FALSE)
```

Arguments

- `x` An instance of CdfEnvAffy-class
- `i` A vector of indexes (integers !).
- `simplify` Simply the resulting CdfEnvAffy (see details).
- `verbose` verbose output or not.
Details
The probes to be removed are set to NA in the CdfEnvAffy. When simplify is set to TRUE the probe sets are simplified whenever possible. For example, if both pm and mm for the same probe pair are set to NA, then the probe pair is removed from the probe set.

Value
An instance of CdfEnvAffy-class is returned.

Author(s)
Laurent Gautier

See Also
CdfEnvAffy-class

Examples
```r
## use plasmodiumanopheles chip as an example
if (require(plasmodiumanophelescdf)) {

  ## wrap in a (convenient) CdfEnvAffy object
  planocdf <- wrapCdfEnvAffy(plasmodiumanophelescdf, 712, 712, "plasmodiumanophelescdf")
  print(planocdf)

  ## ask for the probe indexed '10759' to be removed
  ## (note: if one wishes to remove from X/Y coordinates,
  ## the function xy2index can be of help).
  planocdfCustom <- removeIndex(planocdf, as.integer(10759))

  ## let see what happened (we made this example knowing in which
  ## probe set the probe indexed '10759' is found).
  indexProbes(planocdf, "pm", "200000_s_at")
  indexProbes(planocdfCustom, "pm", "200000_s_at")
  ## The 'second' pm probe (indexed '10579') in the probe set is now set
  ## to NA.
}
```
Arguments

object  Object derived from class AffyProbesMatch.
...  Unused.

Value

An Hypergraph-class object.

Description

Remove duplicated elements from a CdfEnvAffy

Usage

## S3 method for class 'CdfEnvAffy'
unique(x, incomparables = FALSE, simplify = TRUE, verbose = FALSE, ...)

Arguments

x  An instance of CdfEnvAffy-class
incomparables  (not yet implemented)
simplify  simplify the result
verbose  verbose or not
...  (here for compatibility with the generic unique)

Details

The parameter simplify has the same function as the one with the same name in countduplicated.

Value

An instance of CdfEnvAffy-class in which probes used several times are removed.

Warning

The function differs slightly from the generic unique. Here the elements found in several place a merely removed.

Author(s)

Laurent
validAffyBatch

See Also
countduplicated

Examples

## not yet here...

---

validAffyBatch  

*Check validity of a CdfEnvAffy.*

Description

Tries to see if a CdfEnvAffy, or a pair of AffyBatch / CdfEnvAffy is valid.

Usage

validAffyBatch(abatch, cdfenv)
validCdfEnvAffy(cdfenv, verbose=TRUE)
printValidCdfEnvAffy(x)

Arguments

abatch instance of AffyBatch-class
cdfenv instance of CdfEnvAffy-class
verbose verbose or not
x object returned by validCdfEnvAffy

Details

The function validAffyBatch calls in turn validCdfEnvAffy.

See Also

AffyBatch-class, CdfEnvAffy-class

Examples

## To be done...
Index

* IO
  * read.FASTA.entry, 15

* classes
  * AffyProbesMatch-class, 2
    * CdfEnvAffy-class, 5

* connection
  * read.FASTA.entry, 15

* datasets
  * cdfenvEx, 7

* hplot
  * plot.CdfEnvAffy, 14

* manipulate
  * appendCdfEnvAffy, 3
    * buildCdfEnv.biostrings, 4
    * cdfenvs, 7
  * copyCdfEnvAffy, 8
  * countduplicated, 9
  * geneNames.CdfEnvAffy, 10
  * getxy.probeseq, 10
  * index2xy, 11
  * indexProbes.CdfEnvAffy, 12
  * matchAffyProbes, 13
  * read.FASTA.entry, 15
  * removeIndex, 16
  * toHypergraph, 17
  * unique.CdfEnvAffy, 18
  * validAffyBatch, 19

[ , CdfEnvAffy, character, missing, missing-method
  (CdfEnvAffy-class), 5

AffyProbesMatch-class, 2
  * appendCdfEnvAffy, 3

buildCdfEnv.biostrings, 4, 14
  * buildCdfEnv.matchprobes (cdfenvs), 7

CdfEnv (cdfenvs), 7
  * CdfEnvAffy-class, 5
  * cdfenvEx, 7
  * cdfenvs, 7

coerce.CdfEnvAffy, Cdf-method
  (CdfEnvAffy-class), 5
  * coerce.CdfEnvAffy, environment-method
    (CdfEnvAffy-class), 5
  * combine, AffyProbesMatch, AffyProbesMatch-method
    (AffyProbesMatch-class), 2
  * connection, 15
  * copyCdfEnvAffy, 8
  * copyEnv, 9
  * countduplicated, 9, 19
  * countskip.FASTA.entries
    (read.FASTA.entry), 15

geneNames, CdfEnvAffy-method
  (CdfEnvAffy-class), 5
  * geneNames.CdfEnvAffy, 10
  * getCdfEnvAffy (cdfenvs), 7
  * getxy.probeseq, 10
  * grep.FASTA.entry (read.FASTA.entry), 15

Hypergraph, 2
  * index2xy, 11
  * index2xy, CdfEnvAffy-method
    (CdfEnvAffy-class), 5
  * indexProbes, CdfEnvAffy, character-method
    (CdfEnvAffy-class), 5
  * indexProbes.CdfEnvAffy, 6, 12
  * matchAffyProbes, 13
  * matchPDict, 13, 14
  * mmProbes (matchAffyProbes), 13

plot.CdfEnvAffy, missing-method
  (CdfEnvAffy-class), 5
  * plot.CdfEnvAffy, 6, 14
  * print.FASTA (read.FASTA.entry), 15

printValidCdfEnvAffy (validAffyBatch), 19

read.FASTA.entry, 15
INDEX

read.n.FASTA.entries
  (read.FASTA.entry), 15
read.n.FASTA.headers
  (read.FASTA.entry), 15
read.n.FASTA.sequences
  (read.FASTA.entry), 15
removeIndex, 16

show,AffyProbesMatch-method
  (AffyProbesMatch-class), 2
show,CdfEnvAffy-method
  (CdfEnvAffy-class), 5
skip.FASTA.entry (read.FASTA.entry), 15

toHypergraph, 17
toHypergraph,AffyProbesMatch-method
  (AffyProbesMatch-class), 2
toHypergraph,CdfEnvAffy-method
  (CdfEnvAffy-class), 5

unique.CdfEnvAffy, 18
validAffyBatch, 19
validCdfEnvAffy (validAffyBatch), 19

wrapCdfEnvAffy (cdfenvs), 7
write.FASTA (read.FASTA.entry), 15

xy2index (index2xy), 11
xy2index,CdfEnvAffy-method
  (CdfEnvAffy-class), 5