Package ‘flowFP’

March 22, 2017

Type Package

Title Fingerprinting for Flow Cytometry

Version 1.32.0

Author Herb Holyst <holyst@mail.med.upenn.edu>, Wade Rogers
<rogersw@mail.med.upenn.edu>

Date 2009

Maintainer Herb Holyst <holyst@mail.med.upenn.edu>

Depends R (>= 2.10), flowCore, flowViz

Imports Biobase, BiocGenerics (>= 0.1.6), graphics, grDevices,
methods, stats, stats4

Suggests RUnit

Description Fingerprint generation of flow cytometry data, used to
facilitate the application of machine learning and datamining
tools for flow cytometry.

Collate AllClasses.R flowFP-utils.R flowFPModel.R flowFP.R
AllGenerics.R

LazyLoad TRUE

License Artistic-2.0

biocViews FlowCytometry, CellBasedAssays, Clustering, Visualization

NeedsCompilation yes

R topics documented:

flowFP-package .................................................. 2
append-methods .................................................. 4
binBoundary-methods ......................................... 5
counts-methods ................................................ 6
flowFP ............................................................ 6
flowFP-class ...................................................... 8
flowFPModel ..................................................... 9
flowFPModel-class ............................................. 11
flowFPlex ........................................................ 13
flowFPlex-class ............................................... 14
Description

This package is used to explore flow cytometry data through the use of fingerprints. The broad aim of the package is to transform flow cytometric data into a form amenable to algorithmic analysis tools. Thus, it is useful to think of flowFP as an intermediate step between the acquisition of high-throughput flow cytometric data and empirical modeling, machine learning and knowledge discovery.

A fingerprint is a feature vector meant to efficiently represent the multivariate probability distribution function for a flow cytometry data set. It is produced by first creating a data-relevant model of a space, and then applying the model to a dataset, thereby producing fingerprints. Model creation is done through the flowFPModel constructor which can be customized via function arguments. After the model is built, it can be applied to arbitrary flowFrames or flowSets using the flowFP constructor. The resulting S4 object implements plotting and summary methods that allow the user to compare and contrast instances, using the flowFPModel as a sort of basis representation, akin for example to trigonometric functions in a Fourier Transform.

This package is closely integrated with flowCore. You will want to become familiar with it in order to effectively use flowFP.
Classes

`flowFPModel-class` is the fundamental class for the `flowFP` package. It represents the multivariate probability distribution function for a flow cytometry data set. Information is maintained in a number of slots, which should only be accessed through methods, described below, **not by direct use of the @ operator.** For a complete detailed list of slot names and descriptions look at the `flowFPModel-class` help page.

`flowFP-class` extends the `flowFPModel` and contains additional slots to record the assignment to and number of events in the bins of a `flowFPModel`. Methods are supplied to retrieve and visualize the the contents of a `flowFP`.

`flowFPplex-class` is a container for a set of congruent `flowFP` objects (by congruent, we mean that each `flowFP` is a description of the same set of instances). When constructing or appending `flowFPs` into a plex, simple error checking is done to ensure each instance in each `flowFP` is equivalent. Both the `sampleNames` and the `sampleClasses` slots are consulted for internal consistency. An error is generated if any of the `flowFPs` to be joined in a plex contain different `sampleNames` and/or `sampleClasses`.

Constructors

`flowFPModel` is the constructor for this class.

```r
flowFPModel(fcs, name="Default Model", parameters=NULL, nRecursions="auto", dequantize=TRUE, sampleSize=NULL)
```

- `fcs` Either an `flowFrame` or a `flowSet` used to create the model.
- `name` Name given the model.
- `parameters` Parameters to consider when constructing the model.(e.g. c(1,5)).
- `nRecursions` The number of level of recursive subdivision.
- `dequantize` Setting this value causes a small incremental value to be added to each event starting with 1e-8. This effectively reduces the number of duplicate values to break ties when binning.
- `sampleSize` The max number of events to use out of each fcs file in a `flowFrame` or `flowSet`.

`flowFP` is the constructor for this class.

```r
flowFP(fcs, model=NULL, sampleClasses=NULL, ...)
```

- `fcs` Either an `flowFrame` or a `flowSet` used to create the fingerprint.
- `model` A model created using `flowFPModel` or NULL in which case a model will be created from the fcs data supplied.
- `sampleClasses` List of sample class names to be assigned in order to the instances.
If the model is NULL the parameters are passed through to flowFPModel.

`flowFPPlex` is the constructor for this class.

`flowFPPlex(fingerprints=NULL)`

`fingerprints` Either an single `flowFP` or a list of `flowFPs` (e.g. `fingerprints = c(fp1, fp2)`). All of the fingerprints must share the same sample names, and class names (or no class names).

**Note**
For further information please see the vignette.

**Author(s)**
Herb Holyst <<holyst@mail.med.upenn.edu>>, Wade Rogers <<rogersw@mail.med.upenn.edu>>

**References**

**See Also**
`flowCore`, `flowViz`

**Examples**

```r
# load a flowSet to use as an example.
library(flowFP)
```

---

**Description**
This method joins one or more `flowFP` objects into a `flowFPPlex`. Conceptually, each instance’s fingerprint feature vector is extended. An error will occur if the sample names and the class names differ. If we imagine the fingerprints as a matrix where columns are features and the rows are instances, this method binds the columns onto the existing fingerprint matrix.

**Details**
These methods append one or more fingerprints into a `flowFPPlex`.  

---

`append-methods`  
*Methods to append flowFP fingerprints to a flowFPPlex*
Methods

append(x, values, after=length(x))

flowFPPlex methods:

x = "flowFPPlex", values = "flowFP", after = length(x@fingerprints)  Appends a fingerprint onto a flowFPPlex.

x = "flowFPPlex", values = "list", after = length(x@fingerprints)  Appends a list of fingerprints to a flowFPPlex.

x = "flowFPPlex", values = "flowFPPlex", after = length(x@fingerprints)  Combines two flowFPPlexes into one.

Author(s)

Herb Holyst <holyst@mail.med.upenn.edu>, Wade Rogers <rogersw@mail.med.upenn.edu>

See Also

flowFPPlex sampleNames sampleClasses

Examples

data(fs1)
fp1 = flowFP(fs1, name="fingerprint 1")
fp2 = flowFP(fs1, name="fingerprint 2")
plex = flowFPPlex(fp1)
plex = append(plex, fp2)
plex
counts-methods

Method for getting fingerprint data.

Description

These read-only methods retrieve data out of a fingerprint in three different transformations: "raw", "normalized", "log2norm".

Details

Transformations: There are three representations of the data that the user has access to, "raw" is the number of events in each bin, "normalized" is the ratio of the number of events divided by the expected number (the expected number of events is calculated by dividing the total number of events in the instance used to make the fingerprint by the number of features in the fingerprint). "log2norm" is the log2 of the normalized values.

Methods

counts(object, transformation=c("raw", "normalized", "log2norm"))

object = "flowFP" This method returns a feature vector for each instance in the flowFP, the user can change the way the features are represented, by using the transformation argument. see below.

object = "flowFPlex" For a flowFPlex the code assembles all of the counts matrices from all of the flowFP contained in the flowFPlex.

flowFP

Fingerprint constructor

Description

Constructor for flowFP-class.

Usage

flowFP(fcs, model = NULL, sampleClasses = NULL, ...)

Arguments

fcs A flowFrame or flowSet for which fingerprint(s) are desired.
model A model generated with the flowFPModel constructor, or NULL. if NULL, a default model will be silently generated from all instances in x.
sampleClasses An optional character vector describing modeling classes. If supplied, there must be exactly one element for each flowFrame in the flowSet in x (see Details).
... If model is NULL, additional arguments are passed on to the model constructor. see flowFPModel for details.
flowFP

Details

A flowFP object is a reduced representation of a flowFrame or flowSet with respect to a flowFP-Model. The model is derived by finding multivariate regions containing (nearly) equal numbers of events in a training set using one or more user-specified parameters (e.g. CD45 and Side Scatter). The resulting flowFP represents the probability (i.e. number of events) of the instance(s) in x in each of the bins in the underlying model.

This representation is carried in a slot (called counts) in the object, a matrix in which each row corresponds ordinally to the flowFrames in x. Note that if x is a flowFrame and not a flowSet, the counts matrix will have only 1 row. In any case, each row of the matrix is an individual fingerprint, where the fingerprint elements (columns) have a one-to-one correspondence with the bins in the underlying model.

Another slot in the flowFP object describes the bin index for each event, allowing the user to select events according to their membership in bins. Think of this as "micro-gating".

Value

An instance of an object of type flowFP.

Author(s)

Herb Holyst <holyst@mail.med.upenn.edu>, Wade Rogers <rogersw@mail.med.upenn.edu>

References


See Also

flowFP-class, flowFP, flowFPModel-class, flowFPModel

Examples

# load a flowSet to use as an example.
# fs <- read.flowSet(your fcs files...)
data(fs1)
model <- flowFPModel(fs1, parameters=c(4,5), nRecursions=6)

fp <- flowFP(fs1, model)

plot(fp)
**flowFP-class**

Fingerprint class description.

**Description**

This class represents fingerprints generated from a model derived from a `flowFrame` or `flowSet`.

**Objects from the Class**

Objects are created by calling the constructor `flowFP`.

**Slots**

- **counts**: This matrix is organized such that rows correspond to individual `flowFrames`, and columns to the bins in the underlying `flowFPModel`. A matrix element thus represents the number of events in a single bin from a single `flowFrame`. The sum of each row will equal the total number of events in the corresponding `flowFrame`.

- **tags**: This list specifies the bin number to which each event is assigned. There is a list element for each instance in the object. Each list element is a integer vector whose length is the number of events in the corresponding `flowFrame`.

- **sampleNames**: Names of instances, taken from the `flowSet` on construction.

- **sampleClasses**: Factor object describing the category of each instance. (e.g. "Normal" or "Diseased")

- **name**: Inherited from `flowFPModel`.

- **parameters**: Inherited from `flowFPModel`.

- **nRecursions**: Inherited from `flowFPModel`.

- **trainingSet**: Inherited from `flowFPModel`.

- **trainingSetParams**: Inherited from `flowFPModel`.

- **dequantize**: Inherited from `flowFPModel`.

- **split_val**: Inherited from `flowFPModel`.

- **split_axis**: Inherited from `flowFPModel`.

- **binBoundary**: Inherited from `flowFPModel`.

- **.cRecursions**: Inherited from `flowFPModel`.

- **.tmp_tags**: Inherited from `flowFPModel`.

**Extends**

Class `flowFPModel`, directly.

**Methods**

- `[ ] Used to select a subset of a fingerprint object.

- **counts** Returns the counts matrix of a fingerprint object. See **counts**.

- **length** Returns the number of instances in the fingerprint object.

- **nbins** Returns the number of bins (equal to \(2^nRecursions\)) in the fingerprint object.

- **sampleClasses** Returns the sampleClasses slot of the fingerprint object.
sampleClasses<- Set method for the sampleClasses slot of the fingerprint object. Note that the length of the provided object must be equal to the number of instances in the fingerprint object. If the provided object is not a factor it will be coerced to one.

tags Returns the tags slot. See tags.

show Shows the contents of the fingerprint object.

summary Gives a summary of the fingerprint object.

Author(s)
Herb Holyst <<holyst@mail.med.upenn.edu>>, Wade Rogers <<rogersw@mail.med.upenn.edu>>

References


See Also
flowFP, flowFPModel

Examples

# load a flowSet to use as an example.
library(flowFP)
data(fs1)
model <- flowFPModel(fs1, parameters=c(4,5), nRecursions=6)
fp <- flowFP(fs1, model)
plot(fp)
flowFPModel

Arguments

- `fcs`: Training data for model, either a `flowFrame` or `flowSet`.
- `parameters`: A vector of parameters to be considered during model construction.
- `nRecursions`: Number of times the FCS training data will be subdivided. Each level generated \(2^{n\text{Recursions}}\) bins. A warning will be generated if the number of expected events in each bin is < 1. (e.g. if your training set had 1000 events, and you specified `level=10`.)
- `dequantize`: If TRUE, all of the events in the training set will be made unique by adding a tiny value (proportional to the ordinal position of each event) to the data.
- `sampleSize`: Used to specify the per-`flowFrame` sample size of the data to use in model generation. If `NULL`, all of the data in `x` is used. Setting this to a smaller number will speed up processing, at the cost of accuracy.
- `name`: A descriptive name assigned to the model.
- `excludeTime`: If TRUE (default) and no parameters are specified, the model constructor will attempt to exclude the time parameter from consideration when building the model. The time parameter is identified as having the label 'time', matched ignoring case.

Details

This function is used to create a `flowFPModel`, which can then be used to create a set of fingerprints using `flowFP`. A model is a representation of the multivariate probability density function for the training set `x`. This representation is in the form of a set of `bins`, each of which contains (nearly) the same number of events in the training set. Thus, the model can be regarded as a multivariate histogram of `x` with fixed bin count and variable bin size. Bins will be smallest in regions of high density, and largest in regions that are sparsely populated.

Bins are constructed by recursively subdividing the multivariate space specified by `parameters`. At the first level, the entire space is divided in half in the direction of the parameter with the highest variance. At the next level, each of the halves from the first level is itself divided in half, again in the direction of the parameter whose variance (among the events in the parent bin) is the highest. Thus, for each level the number of resulting bins is doubled. The number of bins in the model is thus \(2^{n\text{Recursions}}\).

A model is of little interest by itself. Its only utility is to provide a sort of basis function for forming fingerprints. It may be interesting in some cases to use the entire set `x` to build the model, and then to generate fingerprints for each instance in `x`. For this reason, calling `flowFP` without a model causes `flowFPModel` to be invoked silently under the hood in order to create a model of the `flowSet` provided to `flowFP`.

Value

An object of type `flowFPModel` is returned.

Note

If a `flowSet` is provided as input, it is internally collapsed to a `flowFrame` for binning.

Author(s)

Herb Holyst <<holyst@mail.med.upenn.edu>>, Wade Rogers <<rogersw@mail.med.upenn.edu>>
References


See Also

`flowFP`, `flowFPModel`

Examples

```r
# load a flowSet to use as an example.
library(flowFP)
data(fs1)
fs1
model <- flowFPModel(fs1, parameters=c(4,5), nRecursions=6)

fp <- flowFP(fs1, model)
plot(fp)
```

---

**flowFPModel-class**

Fingerprint model class description.

**Description**

This class is the fundamental class for the package. It contains data and methods used to construct a model of the probability density function of a prototype dataset provided in the form of a `flowFrame` or `flowSet`.

**Objects from the Class**

Objects may only be created by calling the constructor function `flowFPModel`.

**Slots**

- **name**: A user-supplied descriptive name for the model.
- **parameters**: List of FCS parameters to use for model creation. Can be specified either by the names of the parameters or the indices of the parameters.
- **nRecursions**: Number of levels of recursive subdivision. The number of bins in the model will equal $2^n_{Recursions}$.
- **trainingSet**: Names of `flowFrames` from the `FlowSet` used to construct the model.
- **trainingSetParams**: Names of all of the parameters from the `flowFrames` from the `FlowSet` used to construct the model.
- **dequantize**: If TRUE, all of the event parameter values in the training set will be made unique by adding a tiny value (proportional to the ordinal position of each event) to the data.
split_val: A hairy array, aka list of vectors. Each list element is a vector representing the median values at which the data were split.

split_axis: A hairy array, aka list of vectors. Each list element is a vector representing the axis on which the data were split.

binBoundary: An object of class binBoundary, used to hold boundary information used primarily for visualization.

cRecursions: Private value to hold the number of levels of recursion used to construct this model. Using nRecursions the resolution of a fingerprint can be reduced, but it can never exceed this value.

tmp_tags: Scratch array, total number of events in the training set long, that keeps track of the event's bin number. (this exists only to provide the underlying C function with a persistent scratch space it needs for bookkeeping. Not useful to the user.)

Methods

cshow shows the contents of the model.

Note

When creating a model you must keep in mind that it doesn’t make sense to create more bins (which is $2^n_{Recursions}$) than the total number of events used to create the model. The constructor checks for this.

When creating a model, you should specify only parameters that are common to all of the instances (flowFrames) in training data. For example, it does not make sense to compare PE from one flowFrame with FITC from another.

Author(s)

Herb Holyst <<holyst@mail.med.upenn.edu>>, Wade Rogers <<rogersw@mail.med.upenn.edu>>

References


See Also

flowFPModel - Constructor.

Examples

# load a flowSet to use as an example.
library(flowFP)
data(fs1)
mod = flowFPModel(fs1)
flowFPPlex

Fingerprint collection constructor.

Description

This constructor creates a flowFPPlex, which is a collection of flowFPs.

Usage

flowFPPlex(fingerprints = NULL)

Arguments

fingerprints  List of flowFPs, or NULL. If NULL an empty flowFPPlex will be prepared, to which flowFPs may be added with append-methods.

Details

A flowFPPlex is a container object for a collection of flowFPs. It is useful in several ways. First, multiple-tube panels are often used to assay more reagents than can be done on a particular instrument. Second, it is sometimes useful to represent the same data with multiple flowFPModels.

Suppose that we have collected data from 1,000 patients, using an 8-tube panel. Imagine that Tubes 1 and 8 are isotype and viability tubes, respectively (we will ignore these tubes for now). The remaining Tubes 2-7 are of interest from a fingerprinting perspective. We wish to treat them as a unit. We might then create models that represent parameters in each of the tubes, across some (or all) of the patient samples (say, mod2, mod3, ..., mod7). We could then create corresponding flowFPs (say, fp2, fp3, ..., fp7). These can now be combined and treated as a single object of type flowFPPlex, as:

```r
> plex <- flowFPPlex(c(fp2, fp3, fp4, fp5, fp6, fp7))
```

or if you prefer,

```r
> plex <- flowFPPlex (fingerprints=NULL)
> plex <- append(plex, c(fp2, fp3, fp4, fp5, fp6, fp7))
```

The counts or density matrices can then be extracted simply using methods provided in flowFPPlex-class.

The second idea is to use multiple models to represent the same data. In this case we might create a model from, say, the "Normal" instances (call it mod_norm), and another model from the "Cancer" instances (mod_cancer). We might wish to do this to enhance the detection of regions of the distribution that are characteristically dominated by one type or the other. If our flowSet of all instances is called "fs1", then our two representations would be:

```r
> fp_norm <- flowFP (fs1, mod_norm)
> fp_cancer <- flowFP (fs2, mod_cancer)
```

and the plex is

```r
plex <- flowFPPlex (c(fp_norm, fp_cancer))
```

Value

Returns a flowFPPlex.
flowFPPlex-class

Author(s)
Herb Holyst <holyst@mail.med.upenn.edu>, Wade Rogers <rogersw@mail.med.upenn.edu>

See Also
flowFPPlex-class, flowFP

Examples
```r
data(fs1)
data(fs2)
mod1 <- flowFPModel(fs1, parameters=c(2,5))
mod2 <- flowFPModel(fs2, parameters=c(2,5))
fp1_1 <- flowFP(fs1, mod1)
fp1_2 <- flowFP(fs1, mod2)
plex <- flowFPPlex(c(fp1_1, fp1_2))
```

flowFPPlex-class  
Fingerprint collection class description.

Description
This class holds a collection of flowFPs.

Objects from the Class
Objects should be created by calling the constructor function flowFPPlex.

Slots
fngerprints: Holds a list of flowFP objects.

Methods
counts  Concatenates all of the counts for all of the fingerprints in the set.
length  returns the number of fingerprints in the set.
nInstances returns the number of flowFrames used to create this fingerprint.
nFeatures returns the total number of features from the collection of fingerprints.

Author(s)
Herb Holyst <holyst@mail.med.upenn.edu>, Wade Rogers <rogersw@mail.med.upenn.edu>

See Also
flowFPPlex - Constructor.

Examples
```r
showClass("flowFPPlex")
```
Example FCS flowSet

Description
An example FCS flowSet.

Usage
data(fs1)

Format
This is an object of class `flowSet` composed of 7 `flowFrames`.

Details
The `flowSet` corresponds to a Leukemia/Lymphoma panel for a sample. Each `flowFrame` corresponds to one tube. The first one is an isotype control, and the rest are stained with different antibody cocktails. Parameter 5 (FL3 Log) is notable in that it is CD45-ECD for all of the tubes. This makes it possible to gate the entire collection of tubes using CD45 vs SSC. This set has a very similar distribution of CD45 vs SSC across all 7 tubes.

Column names: FS Lin SS Log FL1 Log FL2 Log FL3 Log FL4 Log FL5 Log

Source
Data were generously provided by Clarient, Inc.

References

Examples
data(fs1)

Example FCS flowSet

Description
An example FCS flowSet.

Usage
data(fs2)

Format
This is an object of class `flowSet` composed of 7 `flowFrames`. 
Details

The flowSet corresponds to a Leukemia/Lymphoma panel for a sample. Each flowFrame corresponds to one tube. The first one is an isotype control, and the rest are stained with different antibody cocktails. Parameter 5 (FL3 Log) is notable in that it is CD45-ECD for all of the tubes. This makes it possible to gate the entire collection of tubes using CD45 vs SSC. In this set, the CD45 vs SSC distribution in Tube 5 differs significantly as compared with the other tubes.

column names: FS Lin SS Log FL1 Log FL2 Log FL3 Log FL4 Log FL5 Log

Source

Data were generously provided by Clarient, Inc.

References


Examples

data(fs2)

hasClasses-methods  

Returns TRUE if the flowFP has a class factor for its instances

Description

This read-only accessor returns TRUE if the data object flowFP or flowFPlex instances have been assigned a factor object representing the class membership of the instances.

Methods

hasClasses(object)

object = "flowFP" Returns TRUE if there are class assignments for each instance in this flowFP.

object = "flowFPlex" If a flowFPlex has classes associated with its instances, they must all be the same.

is.flowFP

Test to see if object is a flowFP

Description

This is a convenience function to check if the object is a flowFP.

Usage

is.flowFP(obj)
is.flowFPModel

Arguments

obj returns TRUE if the object is a flowFP

Value

returns TRUE if the object is a flowFP

See Also

flowFP

Examples

data(fs1)
fp = flowFP(fs1)
is.flowFP(fp)

is.flowFPModel  Test to see if object is a flowFPModel

Description

This is a convenience function to check if the object is a flowFPModel.

Usage

is.flowFPModel(obj)

Arguments

obj returns TRUE if the object is a flowFPModel.

Value

Returns TRUE is the object is a flowFPModel.

See Also

flowFPModel

Examples

data(fs1)
fp = flowFPModel(fs1)
is.flowFPModel(fp)
is.flowFPPlex  
Test to see if object is a flowFPPlex

Description
This is a convenience function to check if the object is a flowFPPlex.

Usage
is.flowFPPlex(obj)

Arguments
obj returns TRUE if the object is a flowFPPlex.

Value
returns TRUE if the object is a flowFPPlex.

See Also
flowFPPlex

Examples
data(fs1)
fp = flowFP(fs1)
plex = flowFPPlex(fp)
is.flowFPPlex(fp)

length-methods  
The number of fingerprints in a flowFPPlex

Description
This method sets or gets the number of fingerprints that have been concatenated together to form this flowFPPlex.

Methods
length(x)

x = "flowFPPlex" Sets or gets the number of fingerprints in a flowFPPlex

Notes
The get method returns the number of flowFP objects in the plex. It is not the number of instances or the number of features in the counts matrix!
The set method can only truncate the set of flowFP objects from the right.
**name-methods**

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>These methods return the model name(s) for an object.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>name(object)</code></td>
</tr>
</tbody>
</table>

- **object = "flowFP"** Returns the name of the model used to create these fingerprints.
- **object = "flowFPModel"** Returns the model name.
- **object = "flowFPlex"** Returns all of the model names found in the plex.


**nFeatures-methods**

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>These methods return the number of leaf bins that were generated or will be generated by a specific model of plex of models.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>nFeatures(object)</code></td>
</tr>
</tbody>
</table>

- **object = "flowFP"** The number of bins in a fingerprint.
- **object = "flowFPModel"** The number of bins that will be created by this model.
- **object = "flowFPlex"** The total number of bins in a plex fingerprint.


**nInstances-methods**

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>These methods return the number of <code>flowFrame</code>s that were used to create a fingerprint or plex.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>nInstances(object)</code></td>
</tr>
</tbody>
</table>

- **object = "flowFP"** The number of instances in a fingerprint.
- **object = "flowFPlex"** The total number of instances in a plex fingerprint.
nRecursions-methods  Methods to set or get nRecursions

Description

Gets or Sets the number of recursions for an object

Methods

nRecursions(object)

object = "flowFP"  Sets and Gets the number of recursions for viewing a flowFP object.

object = "flowFPModel"  Gets the number of recursions for viewing a flowFPModel object.

object = "flowFPlex"  Gets the number of recursions for viewing a flowFPlex object.

Notes

When a model is computed (either with the flowFPModel constructor, or implicitly with the flowFP constructor) the number of bins into which events are divided is determined. Because the binning process is recursive, given a high-resolution binning, any lower-resolution representation can easily be derived. The set methods do this derivation, and the get methods return the current resolution of the object.

Thus, the maximum value to which nRecursions may be set is determined by the value of nRecursions at construction time. If an attempt is made to set nRecursions higher than this value a warning will result.

parameters-methods  Return the parameters considered for fingerprinting

Description

These methods return the flow parameters that were taken into consideration at flowFPModel construction.

Methods

parameters(object)

object = "flowFP"  Returns the parameters considered when creating fingerprint.

object = "flowFPModel"  Returns the parameters that will be used to create a fingerprint.
Description

An example of 96 well FCS flow data.

Usage

data(plate)

Format

This is an object of class `flowSet` comprising 96 `flowFrames`. Each `flowFrame` corresponds to one sample. column names: FSC-H SSC-H FL1-H FL2-H FL3-H FL1-A FL4-H Time

Details

This data set is derived from the references below. Each well in a 96-well plate is described by a `flowFrame`. Data were drastically down-sampled to 1000 events per `flowFrame` so that they could be included in this package. Note that fluorescence parameters 5 (CD4 PerCP CY5.5) and 7 (CD3 APC) are common across the entire set.

Source

Original, non-sampled data are available at http://www.ficcs.org/software.html#Data_Files.

References


Examples

data(plate)
plot-methods

Methods for visualizing flowFP objects

Description
These methods allow the user to plot flowFP objects with a number of options.

Methods

plot (x, y, ...)

x = "flowFPModel", y = "missing" Visualize a flowFPModel.
  Optional Args: (parameters=NULL, alpha=1, border="gray", showbins=1:nFeatures(x))

x = "flowFPModel", y = "flowFrame" Visualize a flowFPModel along with a flowFrame.
  Optional Args: (parameters=NULL, alpha=1, border="gray", showbins=1:nFeatures(x))

x = "flowFPModel", y = "flowSet" Visualize a flowFPModel along with a flowSet.
  Optional Args: (parameters=NULL, alpha=1, border="gray", showbins=1:nFeatures(x))

x = "flowFP", y = "missing" Visualize Fingerprints.
  Optional Args: (type=c("tangle", "stack", "grid", "qc", "plate"), ...)
  See Notes for additional details.

x = "flowFP", y = "flowFrame" Visualize a single fingerprint with a flowFrame. See Notes.

x = "flowFP", y = "flowSet" Visualize Fingerprints with a flowSet.
  Optional Args: (x, y, showbins=NULL, showfp=TRUE, ...)

x = "flowFPPlex", y = "missing" Visualize Fingerprints in a flowFPPlex.
  Optional Args: (type=c("tangle", "stack", "grid", "qc", "plate"), ...)

Arguments

parameters Flow parameters involved in fingerprinting.

showbins Which bins to render. May be an integer, list of integers, or boolean. Default: 1:nFeatures(x).

showfp Boolean. Do we want to see the fingerprints (TRUE) or just the parameter plot (FALSE).

xlim Range of values for the x-axis. Default: range of the data.

ylim Range of values for the y-axis. Default: range of the data.

type Specify the type of plot. Current values supported are: "tangle", "stack", "grid", "qc", "plate".

transformation Specify transformation. Values may be: "raw", "normalized", "log2norm"

useClasses Boolean. Colors lines by sampleClass. Default=FALSE.

linecols Specify color(s) for fingerprint lines. Default = 'black'. When used in conjunction with
  useClasses, this can be a list of colors whose length matches the number of levels in the
  sampleClasses factor object.

alpha Opacity of rectangles representing bins.

border Color to draw the border of a bin. Default: 'gray'.

highlight Which fingerprints to highlight. Default=1:length(x).

main Title for the plot. Default: various.
xlab  Label for the x-axis.
vert_scale  Sets the vertical scale for some plots.
respect  Boolean. If TRUE, respect a square aspect ratio.
method  Determine QC metric for fingerprint deviation. Either sd (standard deviation) or max (maximum deviation).
red_limit  Value corresponding to a red color for the QC metric.
...  Other parameters passed through to generic plot.

Notes

In conjunction with showbins: generic plot args such as pch and cex can be used to visualize events in specific bins.

For type="tangle" methods: Optional Args: (transformation=c("raw", "normalized", "log2norm"),
For type="stack" methods: Optional Args: (transformation=c("raw", "normalized", "log2norm"),
For type="grid" methods: Optional Args: (vert_scale=3, main="Fingerprints", linecols="black",
For type="qc" methods: Optional Args: (main="Fingerprint Deviation Plot", transformation=c("log2norm", "raw", "normalized"), vert_scale=3, method=c("sd", "max"), red_limit=1.0, respect=FALSE)
For type="plate" methods: Optional Args: (main="Fingerprint Deviation Plot", transformation=c("log2norm", "raw", "normalized"), vert_scale=3, method=c("sd", "max"), red_limit=1.0)

sampleClasses-methods  Gets/sets the sample classes for a fingerprint.

Description

These methods get or set the sample classes for a fingerprint or a set of fingerprints.

Methods

sampleClasses(object)

object = "flowFP"  Returns the sampleClass factor object found in a flowFP
object = "flowFPPlex"  Returns the sampleClass factor object found in a flowFPPlex

sampleNames-methods  Methods to return sample names from a flowFP object.

Description

These methods get the sample names for a fingerprint or a set of fingerprints.

Methods

sampleNames(object)

object = "flowFP"  returns the sample names in a flowFP object.
object = "flowFPPlex"  returns the sample names in a flowFPPlex object.
### tags-methods

**Methods to view flowFP objects**

**Description**

These methods give basic information about the contents of a flowFP object.

**Methods**

```
show(object)
```

**object** = "flowFP" Show the sample names and model information for a flowFP.

**object** = "flowFPModel" Shows the samples used in creating the model.

**object** = "flowFPPlex" Shows all of the sample named and all of the models found in the plex.

### summary-methods

**Summaryizes a flowFP object.**

**Description**

These methods give a summary of the flowFP object.

**Methods**

```
summary(object)
```

**object** = "flowFP" Show the per-instance min/max/mean counts in a flowFP.

**object** = "flowFPModel" Shows length, class and mode info for a flowFPModel.

**object** = "flowFPPlex" Shows length, class and mode info for a flowFPPlex.

### tags-methods

**Returns tags from a flowFP object.**

**Description**

The tags method is a map into a flowFP object which defines the bin number to which each event in the object has been assigned in fingerprinting process.

**Methods**

```
tags(object)
```

**object** = "flowFP" returns a list of vectors.

**Notes**

This method returns a list of vectors. There is a one-to-one correspondence between vectors in the list and flowFrames in the flowSet in the flowFP.

For each vector in the list, there is a one-to-one correspondence between the vector elements and the events in the corresponding flowFrame. Each vector element is an integer index that indicates the bin number to which that event was assigned in the fingerprinting process.
[methods]

Indexing methods for flowFP and flowFPPlex

Description

These methods allow subset indexing for flowFP and flowFPPlex objects.

Methods

- \texttt{x = "flowFP", i = "numeric"} \quad \text{indexing returns a subset of the fingerprints in a flowFP.}

- \texttt{x = "flowFP", i = "character"} \quad \text{indexing returns a subset of the fingerprints in a flowFP. If \texttt{i} matches one or more \texttt{sampleClasses} in the flowFP, the subset operator returns all fingerprints of that class (or classes). If no matches with \texttt{sampleClass} are found, then the operator attempts to match \texttt{sampleNames}, and returns all matches found.}

- \texttt{x = "flowFPPlex", i = "numeric"} \quad \text{indexing returns a new flowFPPlex with the specified subset of fingerprints.}

- \texttt{x = "flowFPPlex", i = "numeric"} \quad \text{indexing returns a single flowFP object from a flowplex.}
Index

*Topic classes
  flowFP, 6
  flowFP-class, 8
  flowFPModel, 9
  flowFPModel-class, 11
  flowFPlex, 13
  flowFPlex-class, 14
*Topic datasets
  fs1, 15
  fs2, 15
  plate, 21
*Topic methods
  [-methods, 25
    append-methods, 4
    binBoundary-methods, 5
    counts-methods, 6
    hasClasses-methods, 16
    length-methods, 18
    name-methods, 19
    nFeatures-methods, 19
    nInstances-methods, 19
    nRecursions-methods, 20
    parameters-methods, 20
    plot-methods, 22
    sampleClasses-methods, 23
    sampleNames-methods, 23
    show-methods, 24
    summary-methods, 24
    tags-methods, 24
  
*Topic package
  flowFP-package, 2
    [,flowFP,character-method([-methods), 25
    [,flowFP,numeric-method([-methods), 25
    [,flowFPlex,ANY-method([-methods), 25
    [-methods([-methods), 25
    [[,flowFPlex,ANY-method([-methods), 25
    [[-methods([-methods), 25
    [<-.,flowFPlex,numeric,ANY,flowFP-method([-methods), 25
    append,flowFPlex,flowFP-method(append-methods), 4
    append,flowFPlex,flowFPlex-method(append-methods), 4
    append,flowFPlex,flowFPlex-method(append-methods), 4
    append-methods, 4, 13
  
binBoundary (binBoundary-methods), 5
binBoundary,flowFP-method
  (binBoundary-methods), 5
binBoundary,flowFPModel-method
  (binBoundary-methods), 5
binBoundary-methods, 5

counts, 8
counts (counts-methods), 6
counts,flowFP-method (counts-methods), 6
counts,flowFPlex-method (counts-methods), 6
counts-methods, 6

flowFP, 2–4, 6, 7–11, 13, 14, 16, 17, 20
flowFP-class, 6, 7, 8
flowFP-package, 2
flowFPModel, 2, 3, 6–9, 11–13, 17, 20
flowFPModel-class, 7, 9, 11
flowFPlex, 4, 5, 13, 14, 16, 18
flowFPlex-class, 13, 14, 14
flowFrame, 2, 6–8, 10, 11
flowSet, 2, 6–8, 10, 11
fs1, 15
fs2, 15

hasClasses (hasClasses-methods), 16
hasClasses,flowFP-method
  (hasClasses-methods), 16
hasClasses,flowFPlex-method
  (hasClasses-methods), 16
hasClasses-methods, 16

is.flowFP, 16
is.flowFPModel, 17
is.flowFPlex, 18

length,flowFPlex-method
  (length-methods), 18
INDEX

length-methods, 18
length<-,flowFP,numeric-method
(length-methods), 18
name (name-methods), 19
name,flowFP-method (name-methods), 19
name,flowFPModel-method (name-methods), 19
name,flowFPPlex-method (name-methods), 19
name-methods, 19
nFeatures (nFeatures-methods), 19
nFeatures,flowFP-method
(nFeatures-methods), 19
nFeatures,flowFPModel-method
(nFeatures-methods), 19
nFeatures,flowFPPlex-method
(nFeatures-methods), 19
nFeatures-methods, 19
nInstances (nInstances-methods), 19
nInstances,flowFP-method
(nInstances-methods), 19
nInstances,flowFPModel-method
(nInstances-methods), 19
nInstances,flowFPPlex-method
(nInstances-methods), 19
nInstances-methods, 19
nRecursions (nRecursions-methods), 20
nRecursions,flowFP-method
(nRecursions-methods), 20
nRecursions,flowFPModel-method
(nRecursions-methods), 20
nRecursions,flowFPPlex-method
(nRecursions-methods), 20
nRecursions-methods, 20
nRecursions<-(nRecursions-methods), 20
nRecursions<-,flowFP,numeric-method
(nRecursions-methods), 20
nRecursions<-,flowFPModel,numeric-method
(nRecursions-methods), 20
parameters,flowFP-method
(parameters-methods), 20
parameters,flowFPModel-method
(parameters-methods), 20
parameters-methods, 20
plot,flowFP,flowFrame-method
(plot-methods), 22
plot,flowFP,flowSet-method
(plot-methods), 22
plot,flowFP,missing-method
(plot-methods), 22
plot,flowFPModel,flowFrame-method
(plot-methods), 22
plot,flowFPModel,flowSet-method
(plot-methods), 22
plot,flowFPModel,missing-method
(plot-methods), 22
plot,flowFPPlex,missing-method
(plot-methods), 22
plot-methods, 22
sampleClasses, 5
sampleClasses (sampleClasses-methods), 23
sampleClasses,flowFP-method
(sampleClasses-methods), 23
sampleClasses,flowFPPlex-method
(sampleClasses-methods), 23
sampleClasses-methods, 23
sampleClasses<-(sampleClasses-methods), 23
sampleNames, 5
sampleNames (sampleNames-methods), 23
sampleNames,flowFP-method
(sampleNames-methods), 23
sampleNames,flowFPPlex-method
(sampleNames-methods), 23
sampleNames-methods, 23
show,flowFP-method (show-methods), 24
show,flowFPModel-method (show-methods), 24
show,flowFPPlex-method (show-methods), 24
show-methods, 24
summary,flowFP-method
(summary-methods), 24
summary-methods, 24
tags, 9
tags (tags-methods), 24
tags,flowFP-method (tags-methods), 24
tags-methods, 24