

# Package ‘flowBeads’

October 17, 2024

**Type** Package

**Version** 1.42.0

**Title** flowBeads: Analysis of flow bead data

**Author** Nikolas Pontikos

**Date** 2013-03-01

**Description** This package extends flowCore to provide functionality specific to bead data. One of the goals of this package is to automate analysis of bead data for the purpose of normalisation.

**Maintainer** Nikolas Pontikos <n.pontikos@gmail.com>

**Depends** R (>= 2.15.0), methods, Biobase, rrcov, flowCore

**Imports** flowCore, rrcov, knitr, xtable

**Suggests** flowViz

**License** Artistic-2.0

**InstallableEverywhere** yes

**Collate** 'AllGenerics.R' 'AllClasses.R' 'get-methods.R'  
'show-methods.R' 'plot-methods.R' 'beads.R' 'beads1-data.R'  
'beads2-data.R' 'dakomef-data.R' 'cytoalmef-data.R'  
'flowBeads-package.R'

**biocViews** ImmunoOncology, Infrastructure, FlowCytometry,  
CellBasedAssays

**git\_url** <https://git.bioconductor.org/packages/flowBeads>

**git\_branch** RELEASE\_3\_19

**git\_last\_commit** e04615c

**git\_last\_commit\_date** 2024-04-30

**Repository** Bioconductor 3.19

**Date/Publication** 2024-10-16

## Contents

flowBeads-package . . . . .	2
absoluteNormalise . . . . .	3
BeadFlowFrame-class . . . . .	3
beads1 . . . . .	4
beads2 . . . . .	4
cytoalmef . . . . .	4
dakomef . . . . .	4
gateBeads . . . . .	5
GatedBeadFlowFrame-class . . . . .	5
generateReport . . . . .	6
getClusteringStats . . . . .	6
getDate . . . . .	6
getMEFparams . . . . .	6
getMEFtransform . . . . .	7
getParams . . . . .	7
getTransformFunction . . . . .	7
hasMEF . . . . .	7
length . . . . .	8
mefTransform . . . . .	8
plot . . . . .	8
relativeNormalise . . . . .	9
show . . . . .	9
toMEF . . . . .	9

<b>Index</b>	<b>10</b>
--------------	-----------

---

flowBeads-package      *flowBeads*

---

### Description

Bioconductor package for working with calibration beads in flow cytometry. Based on flowCore package.

### Author(s)

Nikolas Pontikos <n.pontikos@gmail.com>

---

absoluteNormalise	<i>absoluteNormalise</i>
-------------------	--------------------------

---

### Description

Absolute normalise to align peaks of bead.data to MEF.

### Arguments

bead.data	<a href="#">GatedBeadFlowFrame</a>
mef.data	<a href="#">data.frame</a>

### Value

A list of affine functions from transformed MFI relative coordinates to transformed MEF absolute coordinates.

---

BeadFlowFrame-class	<i>BeadFlowFrame</i>
---------------------	----------------------

---

### Description

Extension of [flowFrame](#) specific for bead data.

The constructor take as arguments the FCS file and the file containing the MEF values of the beads on the different detector channels

### Usage

```
BeadFlowFrame(fcs.filename, bead.filename)
```

### Arguments

fcs.filename	The file name of the FCS to load. File is loaded with the <a href="#">read.FCS</a> function.
bead.filename	The file name of the MEF configuration files indicating the type of beads in the FCS file. The bead.file is read with <a href="#">read.csv</a> .

### Slots

fcs.filename: The file name of the FCS file from which to read.

bead.filename: The file name of the bead config file.

beads.mef: The [data.frame](#) containing the MEF of the bead populations on different channels.

trans: The transform  $f$  to linearise the fluorescence.

inv.trans: The inverse transform of  $f^{-1}$ .

---

beads1	<i>Dako beads on day 1</i>
--------	----------------------------

---

**Description**

Dako beads on day 1

---

beads2	<i>Dako beads on day 2</i>
--------	----------------------------

---

**Description**

Dako beads on day 2

---

cytocalmef	<i>Cytocal config file</i>
------------	----------------------------

---

**Description**

Cytocal config file

---

dakomef	<i>Dako config file</i>
---------	-------------------------

---

**Description**

Dako config file

---

`gateBeads`*gateBeads*

---

**Description**

`gateBeads` gates on all channels, apply scatter gate first. Find parameters in MEF data.frame which are also present in `BeadFlowFrame` The number of expected bead populations is by default six and it is assumed that there is the same number of beads in each population.

**Arguments**

<code>bead.data</code>	The <code>BeadFlowFrame</code> object to gate.
<code>K</code>	The number of bead populations expected.
<code>verbose</code>	Whether to print debug information.

**Value**

[GatedBeadFlowFrame](#)

**Examples**

```
data(beads1)
gateBeads(beads1)
```

---

`GatedBeadFlowFrame-class`*GatedBeadFlowFrame*

---

**Description**

`GatedBeadFlowFrame`

**Arguments**

<code>labels</code>	The resulting labels of the clustering assigning each event to a different bead population.
<code>clustering.stats</code>	Three dimensional array summarising the stats per channel and population.
<code>mef.transform</code>	The list of MEF transforms

generateReport      *generateReport*

---

**Description**

Generate an HTML report from a Markdown template using [knitr](#).

**Arguments**

bead.data      [GatedBeadFlowFrame](#)  
output.file      name of the file to which to output the HTML report.

**See Also**

[knitr](#)

---

getClusteringStats      *getClusteringStats*

---

**Description**

Returns clustering stats as a 3-dimensional array.

---

getDate      *getDate*

---

**Description**

getDate

**Arguments**

flow.frame      [flowFrame](#) object on which to get the date field

---

getMEFparams      *getMEFparams*

---

**Description**

Returns all the MEF parameter names.

---

`getMEFtransform`      *getMEFtransform*

---

**Description**

Returns MEF transform function.

---

`getParams`      *getParams*

---

**Description**

Returns all the parameter names except the scatter channels.

---

`getTransformFunction`      *getTransformFunction*

---

**Description**

Returns transform function. The default is the logicle transform for FCS 3 and the log10 transform for FCS 2.

---

`hasMEF`      *hasMEF*

---

**Description**

Checks whether we have the MEF for a channel name.

**Arguments**

`bead.data`      [BeadFlowFrame](#)  
`parameter`      [character](#)

---

length	<i>length</i>
--------	---------------

---

**Description**

Returns the number of events in a `flowFrame` object.

**Arguments**

flow.frame      `flowFrame` object on which to get number of beads

---

mefTransform	<i>Logicle transformation constructor</i>
--------------	---

---

**Description**

Input parameters are to be provided in decades

**Usage**

```
mefTransform(transformationId = "mefTransform", alpha,
             beta)
```

**Arguments**

transformationId	The name of the transformation.
alpha	The intercept of the MEF transform.
beta	The slope of the MEF transform.

---

plot	<i>Plot the results of the clustering. Plot only the requested channel which should have a corresponding entry in the MEF files</i>
------	---

---

**Description**

Plot the results of the clustering. Plot only the requested channel which should have a corresponding entry in the MEF files

Ungated bead data, simply draw all channels individually (no colours).

If no argument specified then plot all parameters



---

relativeNormalise	<i>relativeNormalise</i>
-------------------	--------------------------

---

**Description**

Relative normalise to align peaks of bead.data1 to those of bead.data2 Returns a list of affine functions from transformed MFI day one coordinates to transformed MFI day two coordinates. This permits comparison of channels across two days, provided the detector is stable, even in the absence of absolute MEF values.

**Arguments**

bead.data1: [GatedBeadFlowFrame](#) object with MFIs from day one  
 bead.data2: [GatedBeadFlowFrame](#) object with MFIs from day two

**Value**

A list of affine functions from MFI day one coordinates to MFI day two coordinates.

---

show	<i>BeadFlowFrame</i>
------	----------------------

---

**Description**

BeadFlowFrame  
 GatedBeadFlowFrame

---

toMEF	<i>toMEF</i>
-------	--------------

---

**Description**

Given bead.data and a flow.data apply the MEF transform to matching channels in flow.data.

**Arguments**

bead.data The GatedBeadFlowFrame object containing the MEF transform.  
 flow.data The flowFrame object on which to apply the transform.

# Index

- \* **datasets**
  - beads1, 4
  - beads2, 4
  - cytocalmef, 4
  - dakomef, 4
- \* **package**
  - flowBeads-package, 2
- absoluteNormalise, 3
- absoluteNormalise, GatedBeadFlowFrame, data.frame-method (absoluteNormalise), 3
- BeadFlowFrame, 7
- BeadFlowFrame (BeadFlowFrame-class), 3
- BeadFlowFrame-class, 3
- beads1, 4
- beads2, 4
- character, 7
- cytocalmef, 4
- dakomef, 4
- data.frame, 3
- dBeadFlowFrame (BeadFlowFrame-class), 3
- flowBeads (flowBeads-package), 2
- flowBeads-package, 2
- flowFrame, 3, 6, 8
- gateBeads, 5
- gateBeads, BeadFlowFrame-method (gateBeads), 5
- GatedBeadFlowFrame, 3, 5, 6, 9
- GatedBeadFlowFrame (GatedBeadFlowFrame-class), 5
- GatedBeadFlowFrame-class, 5
- generateReport, 6
- generateReport, GatedBeadFlowFrame, character-method (generateReport), 6
- getClusteringStats, 6
- getClusteringStats, GatedBeadFlowFrame-method (getClusteringStats), 6
- getDate, 6
- getDate, flowFrame-method (getDate), 6
- getMEFparams, 6
- getMEFparams, BeadFlowFrame-method (getMEFparams), 6
- getMEFtransform, 7
- getMEFtransform, GatedBeadFlowFrame-method (getMEFtransform), 7
- getParams, 7
- getParams, flowFrame-method (getParams), 7
- getTransformFunction, 7
- getTransformFunction, BeadFlowFrame-method (getTransformFunction), 7
- hasMEF, 7
- hasMEF, BeadFlowFrame, character-method (hasMEF), 7
- knitr, 6
- length, 8
- length, flowFrame-method (length), 8
- mefTransform, 8
- plot, 8
- plot, BeadFlowFrame, character-method (plot), 8
- plot, BeadFlowFrame, missing-method (plot), 8
- plot, GatedBeadFlowFrame, character-method (plot), 8
- read.csv, 3
- read.FCS, 3
- relativeNormalise, 9
- relativeNormalise, GatedBeadFlowFrame, GatedBeadFlowFrame-method (relativeNormalise), 9

show, [9](#)

toMEF, [9](#)

toMEF,GatedBeadFlowFrame,flowFrame-method  
    (toMEF), [9](#)