

# Package ‘FBN’

January 2, 2012

**Type** Package

**Title** FISH Based Normalization and Copy Number inference of SNP microarray data

**Version** 1.5

**Date** 2010-09-02

**Author** Adrian Andronache <adi.andronache@gmail.com>, Luca Agnelli  
<luca.agnelli@gmail.com>

**Maintainer** Adrian Andronache <adi.andronache@gmail.com>, Luca Agnelli  
<luca.agnelli@gmail.com>

**Description** Normalizes the data from a file containing the raw values  
of the SNP probes of microarray data by using the FISH probes and their corresponding CNs.

**License** GPL (>= 2)

**LazyLoad** yes

**Repository** CRAN

**Date/Publication** 2010-09-02 11:28:08

## R topics documented:

FBN-package . . . . .	2
FBN.histogramMaxima . . . . .	2
FBN.kmeans . . . . .	4
FBN.valueCenter . . . . .	5
FBNormalizatiion . . . . .	6
FISHprobes . . . . .	8
hmcls . . . . .	9
meanFilter . . . . .	9
medianFilter . . . . .	10

<b>Index</b>	<b>11</b>
--------------	-----------

---

FBN-package

*FISH Based Normalization and Copy Number inference of SNP microarray data*

---

### Description

Normalizes the data from a file containing the raw values of the SNP probes of microarray data by using the FISH probes and their corresponding CNs.

### Details

Package: FBN  
Type: Package  
Version: 1.5  
Date: 2010-09-02  
License: GPL (>=2)  
LazyLoad: yes

To start using the FBN package, call `library(FBN)`.

### Author(s)

Adrian Andronache <adi.andronache@gmail.com>  
Luca Agnelli <luca.agnelli@gmail.com>

### References

Agnelli L et al. (2009), "A SNP Microarray and FISH-Based Procedure to Detect Allelic Imbalances in Multiple Myeloma: an Integrated Genomics Approach Reveals a Wide Gene Dosage Effect", *Genes Chrom Cancer*

---

FBN.histogramMaxima

*Local maxima of the histogram of SNP microarray data*

---

### Description

Finds at most 6 of the local maxima of the histogram of the `inputData` representing the SNP microarray data. Firstly, the function finds all local maxima onto the histogram, and finally removes those that are closer than `minSpan`. The histogram is estimated with equi-spaced breaks (also the default), defined by `breaksData` (see the documentation of `hist` for more details).

### Usage

```
FBN.histogramMaxima(inputData, minSpan, breaksData)
FBN.histogramMaxima(inputData, minSpan = 0.2, breaksData = NULL)
```

## Arguments

inputData	A vector of SNP microarray values for which the local maxima of the histogram are desired
minSpan	The minimum distance separating consecutive local maxima. If minSpan is negative, then a smoothing filter is applied on the histogram to reduce its noise, and therefore, estimates the most significant maxima.
breaksData	One of: <ul style="list-style-type: none"><li>• a vector giving the breakpoints between histogram cells,</li><li>• a single number giving the number of cells for the histogram,</li><li>• a character string naming an algorithm to compute the number of cells (see the Details section of <a href="#">hist</a>),</li><li>• a function to compute the number of cells.</li></ul>

## Details

This function has been designed based on SNP microarray data and FISH resolution. As in FISH analyses high numbers of signals do not allow a correct identification of discrete CNs, an empirical number of no more than 6 local maxima is therefore imposed. These maxima are used to initialize the k-means algorithm to determine the CN clusters.

## Value

Returns a vector containing at most 6 values of the inputData in which the histogram shows local maxima.

## Author(s)

Adrian Andronache <adi.andronache@gmail.com>  
Luca Agnelli <luca.agnelli@gmail.com>

## See Also

[hist](#), [FBNormalization](#)

## Examples

```
require(stats)
require(graphics)

x = c(rnorm(1000, 1, .2), rnorm(1000, 2, .2))
y = FBN.histogramMaxima(x, minSpan = .1)
h = hist(x)
par(new = TRUE)
plot(y, vector(mode=mode(y), length = length(y)), xlim = c(min(h$breaks),
  max(h$breaks)), ylim = c(0,max(h$counts)), xlab = NA, ylab = NA,
  col = 'red' )

x = c(1, 2, 2, 3, 4, 5, 5, 5, 6, 7, 8, 9, 10, 10, 10, 11)
```

```

y = FBN.histogramMaxima(x, minSpan = 3, breaksData = 100)
h = hist(x, 100)
par(new = TRUE)
plot(y,vector(mode=mode(y), length = length(y)), xlim = c(min(h$breaks),
  max(h$breaks)), ylim = c(0,max(h$counts)), xlab = NA, ylab = NA,
  col = 'red' )

```

---

FBN.kmeans

*K-Means clustering of SNP microarray data*


---

### Description

Performs a k-means clustering of SNP microarray data. Returns clusters of values as being putatively characterized by different CN.

### Usage

```

FBN.kmeans(inputData, minSpan, breaksData)
FBN.kmeans(inputData = NULL, minSpan = 0.2, breaksData = NULL)

```

### Arguments

inputData	A vector of values containig the SNP microarray data
minSpan	The minimum distance separating consecutive local maxima that are to be detected on the histogram of the inputData. These maxima are used to initialize the k-means clustering process. For more details concerning the local maxima detection, check the documentation of <a href="#">FBN.histogramMaxima</a>
breaksData	One of: <ul style="list-style-type: none"> <li>• a vector giving the breakpoints between histogram cells,</li> <li>• a single number giving the number of cells for the histogram,</li> <li>• a character string naming an algorithm to compute the number of cells (see Details section of <a href="#">hist</a>),</li> <li>• a function to compute the number of cells.</li> </ul>

### Details

This fuction takes as input the vector of raw SNP microarray values, and perform a k-means clustering trying to identify the groups of raw values characterized by different CNs. The clustering process is initialized with the local maxima detected on the histogram of the input data (see the documentation of [FBN.histogramMaxima](#)). To increase the robustness of the clustering process and to remove possible small or noisy clusters, a double filtering is done: firstly, removing those clusters populated by less than 1% values from the entire inputData, and then, due to putative noisy histograms, merging those clusters whose centers are closer than 0.2 in nominal values.

### Value

An object of class [kmeans](#)

**Author(s)**

Adrian Andronache <adi.andronache@gmail.com>  
 Luca Agnelli <luca.agnelli@gmail.com>

**Examples**

```
require(stats)
require(graphics)
x = c(rnorm(1000, 1, .2), rnorm(1000, 2, .2))
y = FBN.kmeans(x, minSpan = .001)
h = hist(x)
par(new = TRUE)
plot(y$centers, vector(mode=mode(y$centers), length = length(y$centers)),
      xlim = c(min(h$breaks), max(h$breaks)), ylim = c(0,max(h$counts)),
      xlab = NA, ylab = NA, col = 'red' )
```

---

 FBN.valueCenter

*The SNP normalization function*


---

**Description**

Normalization of the raw SNP microarray values, by multiplication (on linear scale) or addition (in log scale) of all the raw SNP values with the normalization factor. The normalization factor is estimated such that it brings the normalizingValue of the raw SNP values onto the nominalValueCN.

**Usage**

```
FBN.valueCenter(inputData, normalizingValue, nominalValueCN,
                logScale)
FBN.valueCenter(inputData = NULL, normalizingValue = NULL,
                nominalValueCN = 2, logScale = FALSE)
```

**Arguments**

inputData	The vector of raw SNP values, as they come out from, e.g. Circular Binary Segmentation in DNACopy package from Bioconductor
normalizingValue	The value representing the center of the cluster identified as having a certain CN
nominalValueCN	The nominal value representing a certain CN on which the normalizingValue has to be brought.
logScale	A logical value, specifying whether the data is on linear (FALSE) or logarithmic scale (TRUE).

**Details**

The `nominalValueCN` is a real value representing the CN, e.g. CN= 2 has a `nominalValueCN` of 2, but all other CN=  $n$  ( $n \neq 2$ ) will have a `nominalValueCN` different from  $n$ . Such `nominalValueCN` is identified by the `FBN.kmeans` function.

**Value**

Returns a vector containing the normalized values of the `inputData`

**Author(s)**

Adrian Andronache <adi.andronache@gmail.com>  
Luca Agnelli <luca.agnelli@gmail.com>

**See Also**

[FBN.kmeans](#), [FBNormalization](#)

**Examples**

```
require(stats)
require(graphics)
x = c(rnorm(1000, 1, .1), rnorm(1000, 1.5, .1))
y = FBN.valueCenter(x, normalizingValue = 1, nominalValueCN = 2,
logScale = FALSE)
par(mfrow = c(2, 1), new = FALSE)
h = hist(x)
par(new = TRUE)
plot(1, 0, col = 'red', xlim = c(min(h$breaks), max(h$breaks)),
ylim = c(0,max(h$counts)), xlab = NA, ylab = NA)
par(new = FALSE)
h = hist(y)
par(new = TRUE)
plot(2, 0, col = 'red', xlim = c(min(h$breaks), max(h$breaks)),
ylim = c(0,max(h$counts)), xlab = NA, ylab = NA)
```

---

FBNormalization

*FISH Based Normalization and Copy Number inference of SNP microarray data*

---

**Description**

Normalizes the data from a file containing the raw values of the SNP probes of microarray data by using the FISH probes and their corresponding CNs.

**Usage**

```
FBNormalization(rawDataFileName, fishProbesFileName,
normDataFileName, debugFlag, plotFlag, plotAndSaveFlag)
FBNormalization(rawDataFileName = NULL, fishProbesFileName = NULL,
normDataFileName = NULL, debugFlag = FALSE, plotFlag = FALSE, plotAndSaveFlag = FALSE)
```

**Arguments**

rawDataFileName

The file containing the raw values of the SNP probes. It should be .txt format, tab delimited, containing on each column the SNP probes data of the different samples. The structure of the input file should be as follows:

- a column containing the identification name of the SNP probes, entitled 'SNP',
- a column containing the chromosome number of the SNP probes, entitled 'chrom',
- a column containing the physical position of the SNP probes, entitled 'physical.position',
- a column containing the cytoband of the SNP probe, entitled 'cytoband',
- 'n' columns containing the values of the SNP probes, entitled with the custom identification name of the samples.

fishProbesFileName

The file containing the FISH probes information. It should be .txt format, tab delimited, containing a separate row for each available FISH probe and on each column the various CN data revealed by FISH of the different samples. The structure of the input file should be as follows:

- each row should represent one FISH probe
- one column containing the name of the BAC clone, entitled 'BACclone';
- one column containing the chromosome number where the FISH probe is located, entitled 'chromosome';
- one column containing the cytoband location of the Fish probe, entitled 'cytoband';
- one column containing the start position of the physical localization of the FISH probe, entitled 'start.loc';
- one column containing the end position of the physical localization of the FISH probe, entitled 'end.loc';
- 'n' columns containing the values of the CN corresponding to the FISH probe, entitled with the custom identification name of the samples.

normDataFileName

The name of the output file where the normalized data will be written. Also, the function writes a file starting with the same name as normDataFileName and ending with '\_thresholds.txt', containing the thresholds values that can be used to infer CNs to the normalized data.

debugFlag

Logical value, specifying whether the function should run in debug mode. If TRUE, it plots the various histograms and the detected centers of the clusters in the different samples.

`plotFlag` Logical value, specifying whether the function should plot the histograms of the individual data. If TRUE, it plots the various histograms and the detected centers of the clusters in the different samples.

`plotAndSaveFlag` Logical value, specifying whether the function should plot and save the various plots of the histograms.

### Details

For further details, see Supporting Information in Agnelli L et al. (2009), "A SNP Microarray and FISH-Based Procedure to Detect Allelic Imbalances in Multiple Myeloma: an Integrated Genomics Approach Reveals a Wide Gene Dosage Effect", *Genes Chrom Cancer*

### Value

No value is returned. The function writes on the disk the output files as previously described.

### Author(s)

Adrian Andronache <adi.andronache@gmail.com>  
Luca Agnelli <luca.agnelli@gmail.com>

### Examples

```
## set path to FBN package data directory
rawDataFileName = '../data/hmcls.txt'
fishProbesFileName = '../data/FISHprobes.txt'
normDataFileName = 'hmcls_NORM.txt'
FBNormalization(rawDataFileName, fishProbesFileName, normDataFileName,
debugFlag = FALSE)
```

---

FISHprobes	<i>Sample data for FBN package</i>
------------	------------------------------------

---

### Description

Four FISH probes information for the two sample Human Myeloma Cell Lines (AMO1 and NCI-H929).

### Format

tab delimited .txt file

### References

Agnelli L et al. (2009), "A SNP Microarray and FISH-Based Procedure to Detect Allelic Imbalances in Multiple Myeloma: an Integrated Genomics Approach Reveals a Wide Gene Dosage Effect", *Genes Chrom Cancer*

---

hmcls	<i>Sample data for FBN package</i>
-------	------------------------------------

---

**Description**

Two Human Myeloma Cell Lines (AMO1 and NCI-H929) profiled on GeneChip Human Mapping 250K NspI arrays.

**Format**

image data .rda

**References**

Agnelli L et al. (2009), "A SNP Microarray and FISH-Based Procedure to Detect Allelic Imbalances in Multiple Myeloma: an Integrated Genomics Approach Reveals a Wide Gene Dosage Effect", *Genes Chrom Cancer*

---

meanFilter	<i>1D Mean Filter</i>
------------	-----------------------

---

**Description**

1-dimensional mean filter with a specified windowSize of the inputData

**Usage**

```
meanFilter(inputData, windowSize)
meanFilter(inputData = NULL, windowSize = 3)
```

**Arguments**

inputData	The vector of values to be filtered
windowSize	The half-size of the filtering window (default windowSize = 3)

**Details**

Classical implementation of a mean filter, using a sliding window. By default, the half-size of the sliding window is set to 3 unless otherwise specified.

**Value**

The output data has the same size of the input data. If the window half-size is smaller or equal to 1, then the input data is passed directly to the output data.

**Author(s)**

Adrian Andronache <adi.andronache@gmail.com>  
Luca Agnelli <luca.agnelli@gmail.com>

**Examples**

```
x <- meanFilter(c(0, 0, 0, 1, 1, 1, 0, 0, 1, 0))  
x <- meanFilter(c(0, 0, 0, 1, 0, 0, 0, 0, 1, 0), windowSize = 5)
```

---

medianFilter

*1D Median Filter*

---

**Description**

1-dimensinal median filter with a specified windowSize of the inputData

**Usage**

```
medianFilter(inputData, windowSize)  
medianFilter(inputData = NULL, windowSize = 3)
```

**Arguments**

inputData	The vector of values to be filtered
windowSize	The half-size of the filtering window (default windowSize = 3)

**Details**

Classical implementation of a median filter, using a sliding window. By default, the half-size of the sliding window is set to 3 unless otherwise specified.

**Value**

The output data has the same size of the input data. If the window half-size is smaller or equal to 1, then the input data is passed directly to the output data.

**Author(s)**

Adrian Andronache <adi.andronache@gmail.com>  
Luca Agnelli <luca.agnelli@gmail.com>

**Examples**

```
x <- medianFilter(c(0, 0, 0, 1, 1, 1, 0, 0, 1, 0))  
x <- medianFilter(c(0, 0, 0, 1, 0, 0, 0, 0, 1, 0), windowSize = 5)
```

# Index

## \*Topic **datasets**

FISHprobes, 8

hmcls, 9

## \*Topic **package**

FBN-package, 2

FBN (FBN-package), 2

FBN-package, 2

FBN.histogramMaxima, 2, 4

FBN.kmeans, 4, 6

FBN.valueCenter, 5

FBNormalization, 3, 6, 6

FISHprobes, 8

hist, 2–4

hmcls, 9

kmeans, 4

meanFilter, 9

medianFilter, 10