# Package ‘MetaCyto’

October 15, 2023

**Type** Package

**Title** MetaCyto: A package for meta-analysis of cytometry data

**Version** 1.22.0

**Author** Zicheng Hu, Chethan Jujjavarapu, Sanchita Bhattacharya, Atul J. Butte

**Maintainer** Zicheng Hu <zicheng.hu@ucsf.edu>

**Description** This package provides functions for preprocessing, automated gating and meta-analysis of cytometry data. It also provides functions that facilitate the collection of cytometry data from the ImmPort database.

**License** GPL (>= 2)

**RoxygenNote** 6.0.1

**Imports** flowCore (>= 1.4),tidyr (>= 0.7),fastcluster,ggplot2,metafor,cluster,FlowSOM, grDevices, graphics, stats, utils

**Depends** R (>= 3.4)

**Suggests** knitr, dplyr, rmarkdown

**VignetteBuilder** knitr, rmarkdown

**biocViews** ImmunoOncology, CellBiology, FlowCytometry, Clustering, StatisticalMethod, Software, CellBasedAssays, Preprocessing

**git_url** https://git.bioconductor.org/packages/MetaCyto

**git_branch** RELEASE_3_17

**git_last_commit** 59687dc

**git_last_commit_date** 2023-04-25

**Date/Publication** 2023-10-15

## R topics documented:


autoCluster.batch . . . . . 2
clusterStats . . . . . 4
collectData . . . . . 5
densityPlot . . . . . 5

fcsInfoParser . . . . . 6
filterLabels . . . . . 7
findCutoff . . . . . 8
flowHC . . . . . 9
flowSOM.MC . . . . . 9
glmAnalysis . . . . . 10
labelCluster . . . . . 11
labelSummary . . . . . 12
markerFinder . . . . . 13
metaAnalysis . . . . . 14
nameUpdator . . . . . 15
panelSummary . . . . . 16
plotGA . . . . . 17
preprocessing . . . . . 18
preprocessing.batch . . . . . 19
sampleInfoParser . . . . . 20
searchCluster . . . . . 21
searchCluster.batch . . . . . 22
set2Frame . . . . . 23

**Index** **25**

---

`autoCluster.batch` *Cluster the preprocessed fcs files from different studies in batch*

---

## Description

A function that clusters the pre-processed fcs files from different studies in batch.

## Usage

```
autoCluster.batch(preprocessOutputFolder,
  excludeClusterParameters = c("TIME"), labelQuantile = 0.95,
  clusterFunction = flowSOM.MC, minPercent = 0.05, ...)
```

## Arguments

`preprocessOutputFolder`
: Directory where the preprocessed results are stored. Should be the same with the outpath argument in preprocessing.batch function.

`excludeClusterParameters`
: A vector specifying the name of markers not to be used for clustering and labeling. Typical example includes: Time, cell_length.

`labelQuantile`
: A number between 0.5 and 1. Used to specify the minimum percent of cells in a cluster required to express higher or lower level of a marker than the cutoff value for labeling.

`clusterFunction`
: The name of unsupervised clustering function the user wish to use for clustering the cells. The default is "flowSOM.MC". The first argument of the function must take a flow frame, the second argument of the function must take a vector of excludeClusterParameters. The function must return a list of clusters containing cell IDs. flowSOM.MC and flowHC are implemented in the package. For other methods, please make your own wrapper functions.

`minPercent`
: A number between 0 and 0.5. Used to specify the minimum percent of cells in the positive and negative region after bisection. Keep it small to avoid bisecting uni-mode distributions.

`...`
: Pass arguments to clusterFunction

## Value

A vector of labels identified in the cytometry data.

## Examples

```
#get meta-data
fn=system.file("extdata","fcs_info.csv",package="MetaCyto")
fcs_info=read.csv(fn,stringsAsFactors=FALSE,check.names=FALSE)
fcs_info$fcs_files=system.file("extdata",fcs_info$fcs_files,
                               package="MetaCyto")
# Make sure the transformation parameter "b" and the "assay" argument
# are correct of FCM and CyTOF files
b=assay=rep(NA,nrow(fcs_info))
b[grepl("CyTOF",fcs_info$study_id)]=1/8
b[grepl("FCM",fcs_info$study_id)]=1/150
assay[grepl("CyTOF",fcs_info$study_id)]="CyTOF"
assay[grepl("FCM",fcs_info$study_id)]="FCM"
# preprocessing
preprocessing.batch(inputMeta=fcs_info,
                    assay=assay,
                    b=b,
                    outpath="Example_Result/preprocess_output",
                    excludeTransformParameters=c("FSC-A","FSC-W","FSC-H",
                    "Time","Cell_length"))
# Make sure marker names are consistant in different studies
files=list.files("Example_Result",pattern="processed_sample",
                 recursive=TRUE,full.names=TRUE)
nameUpdator("CD8B","CD8",files)
# find the clusters
excludeClusterParameters=c("FSC-A","FSC-W","FSC-H","SSC-A",
                           "SSC-W","SSC-H","Time",
                          "CELL_LENGTH","DEAD","DNA1","DNA2")
cluster_label=autoCluster.batch(
              preprocessOutputFolder="Example_Result/preprocess_output",
              excludeClusterParameters=excludeClusterParameters,
              labelQuantile=0.95,
              clusterFunction=flowHC)
```

clusterStats *Derive summary statistics for clusters*

## Description

A function that derives summary statistics for clusters.

## Usage

```
clusterStats(fcsFrame, clusterList, fcsNames)
```

## Arguments

| | |
|---|---|
| fcsFrame | A flow Frame object returned from preprocessing function. Must contain a parameter called "sample_id" that specify the origin of each cell using integer IDs. |
| clusterList | A list, each element should be a vector containing the IDs of all cells that belong to a cluster. |
| fcsNames | A vector of fcs file names. Each element corresponds to an integer ID in the "sample_id" parameter in fcsFrame. |

## Value

Returns a data frame, rows correspond to each fcs file, columns correspond to MFI of markers or fractions.

## Examples

```
# Find fcs files
files=system.file("extdata","SDY420/ResultFiles/CyTOF_result",
                  package="MetaCyto")
files=list.files(files,pattern="fcs$",full.names=TRUE)
# Preprocess
fcs = preprocessing(fcsFiles=files,assay ="CyTOF",b=1/8)
# Search clusters
cluster_list=searchCluster(fcsFrame=fcs,
                           clusterLabel=c("CD3+|CD8+","CD3-|CD19+"))
# derive summary statistics
cluster_stats=clusterStats(fcsFrame=fcs,
                           clusterList=cluster_list$clusterList,
                           fcsNames=files)
```

## collectData — *Collect and combine data from multiple csv files of the same format*

### Description

A function that collect and combine data from multiple csv files of the same format.

### Usage

```
collectData(files, longform = TRUE)
```

### Arguments

| | |
|---|---|
| `files` | A vector containing the paths of csv files to be combined. |
| `longform` | True or False. Used to specify if the table in each csv file should be converted into long form before combining. |

### Value

A dataframe containing combined information from multiple csv files.

### Examples

```
# find all the files we want to combine
fn=system.file("extdata","",package="MetaCyto")
fn=list.files(fn,pattern="cluster_stats_in_each_sample",full.names=TRUE)
# Comine the data
all_data = collectData(fn,longform=TRUE)
```

## densityPlot — *Draw density plot for each cell cluster.*

### Description

A function that draws density plot for each cell cluster.

### Usage

```
densityPlot(fcsFrame, clusterList, cutoff, markerToPlot = NULL)
```

### Arguments

| | |
|---|---|
| `fcsFrame` | A flow Frame object returned from preprocessing function. |
| `clusterList` | A list, each element should be a vector containing the IDs of all cells that belong to a cluster |
| `cutoff` | A vector of cutoff values to bisect the distribution of each marker. The names of the vector should be the same as the marker names. |
| `markerToPlot` | A vector specifying markers included in the plot. If NULL, all markers will be plotted. |

### Details

The plot can be very large, we suggest plotting it into a pdf jpeg file directly.

### Value

NULL. The plot will show up automatically.

### Examples

```
# Find fcs files
files=system.file("extdata","SDY420/ResultFiles/CyTOF_result",
                  package="MetaCyto")
files=list.files(files,pattern="fcs$",full.names=TRUE)
# Preprocess
fcs = preprocessing(fcsFiles=files,assay ="CyTOF",b=1/8)
# Search clusters
cluster_list=searchCluster(fcsFrame=fcs,
                           clusterLabel=c("CD3+|CD8+","CD3-|CD19+"))
# plot density plot for clusters
densityPlot(fcs,
            clusterList=cluster_list$clusterList,
            cutoff=cluster_list$cutoff,
            markerToPlot=c("CD3","CD8","CD19"))
```

---

## `fcsInfoParser` *Organize fcs files in a study from ImmPort into panels*

---

### Description

A function that organizes fcs files in a study from ImmPort into panels.

### Usage

```
fcsInfoParser(metaData, studyFolder, fcsCol = "ZBXFN", assay = c("FCM",
  "CyTOF"))
```

### Arguments

| | |
|---|---|
| metaData | A data frame. Must contain a column listing the names of fcs files included in the study. |
| studyFolder | Path of the directory containing all the files of a study from ImmPort. |
| fcsCol | A string specifying the name of the column in metaData that lists fcs files included in the study. |
| assay | Either "FCM" or "CyTOF" to indicate the type of cytometry data. |

### Value

A dataframe containing 2 columns: a column called "fcs_files" that contains the location (relative to the working directory) of each fcs file on the hard drive and a column called "study_id" that specify what study each fcs file belongs to.

### Examples

```
fn=system.file("extdata",
              "SDY736/SDY736-DR19_Subject_2_Flow_cytometry_result.txt",
              package="MetaCyto")
meta_data=read.table(fn,sep='\t',header=TRUE, check.names= FALSE)
# Organize fcs file into panels
fn=system.file("extdata","SDY736",package="MetaCyto")
fcs_info_SDY736=fcsInfoParser(metaData=meta_data,
                              studyFolder=fn,
                              fcsCol="File Name",
                              assay="FCM")
```

---

## filterLabels *Filter cluster labels*

---

### Description

A function that filter cluster labels.

### Usage

```
filterLabels(labels, minPlus, minMarker, maxMarker)
```

### Arguments

| | |
|---|---|
| labels | A vector containing labels for cell clusters |
| minPlus | An integer, used to specify the minimum number of "+" a label should contain. |
| minMarker | An integer, used to specify the minimum number of markers a label should contain. |
| maxMarker | An integer, used to specify the max number of markers a label should contain. |

### Value

Returns a vector of labels that pass through the filter.

### Examples

```
labels= c("CD3-|CD4-|CD8-","CD3+|CD4+|CD8-",
         "CD3+|CD4-|CD8+","CD3+|CD4-|CD8+|CCR7+|CD45RA-|CCR6-")
labels=filterLabels(labels=labels,minPlus=1,minMarker=2,maxMarker=5)
```

---

## findCutoff *Find cutoff for a 1D distribution*

---

### Description

A function that finds cutoff for a 1D distribution.

### Usage

```
findCutoff(x, returnSil = FALSE, useBL = TRUE, minX = 0)
```

### Arguments

| | |
|---|---|
| x | A vector of values. |
| returnSil | Logic, used to specify if the max average silhouette is returned |
| useBL | Logic, used to specify if outliers should be ignored |
| minX | A numerical value, used to specify the min value allowed for the cutoff. |

### Value

If returnSil=F, returns a single cutoff value. Otherwise, returns a list containing the cutoff value and the max average silhouette

### Examples

```
x=c(rnorm(1000),rnorm(1000,5))
findCutoff(x)
```

## flowHC — *Cluster cytometry data using hierarchical clustering*

### Description

A function that cluster cytometry data using hierarchical clustering.

### Usage

```
flowHC(fcsFrame, excludeClusterParameters, minimumClusterSizePercent = 0.05)
```

### Arguments

`fcsFrame` A flow frame.

`excludeClusterParameters` A vector specifying the name of markers not to be used for clustering.

`minimumClusterSizePercent` A number between 0 and 1, used to specify the minimum size of a cluster relative to all events.

### Value

A list of clusters. Each cluster contains the ID of all cells that belong to the cluster.

### Examples

```
# Find fcs files
files=system.file("extdata","SDY420/ResultFiles/CyTOF_result",
                 package="MetaCyto")
files=list.files(files,pattern="fcs$",full.names=TRUE)
# Preprocess
fcs = preprocessing(fcsFiles=files,assay ="CyTOF",b=1/8)
# cluster using flowHC
cluster_list=flowHC(fcsFrame=fcs,
                    excludeClusterParameters=c("Time","Cell_length"))
```

## flowSOM.MC — *Cluster cytometry data using FlowSOM*

### Description

A function that cluster cytometry data using FlowSOM.

### Usage

```
flowSOM.MC(fcsFrame, excludeClusterParameters, xdim = 10, ydim = 10,
  k = 40, seed = NULL)
```

### Arguments

| | |
|---|---|
| fcsFrame | A flow frame. |
| excludeClusterParameters | A vector specifying the name of markers not to be used for clustering. |
| xdim | An integer, defines the width of SOM |
| ydim | An integer, defines the height of SOM |
| k | An integer, defines the number of clusters to be identified by flowSOM. |
| seed | Set seed for reproducible results. |

### Value

A list of clusters. Each cluster contains the ID of all cells that belong to the cluster.

### Examples

```
# Find fcs files
files=system.file("extdata",
                  "SDY420/ResultFiles/CyTOF_result",package="MetaCyto")
files=list.files(files,pattern="fcs$",full.names=TRUE)
# Preprocess
fcs = preprocessing(fcsFiles=files,assay ="CyTOF",b=1/8)
# cluster using flowSOM.MC
cluster_list=flowSOM.MC(fcsFrame=fcs,
                        excludeClusterParameters=c("Time","Cell_length"))
```

---

## glmAnalysis *Perform generalized linear model analysis to estimate effect size.*

---

### Description

A function that performs generalized linear model analysis to estimate effect size.

### Usage

```
glmAnalysis(value = "value", variableOfInterst = "Subject Age", parameter,
  otherVariables = c("Gender"), studyID = "study", label = "label", data,
  CILevel = 0.95, ifScale = c(TRUE, FALSE))
```

### Arguments

| | |
|---|---|
| value | A string to specify the column name of the dependent variable (y) |
| variableOfInterst | A string to specify the column name of the independent variable of interest (x1) |
| parameter | A string to specify what summary statistics is the dependent variable. |
| otherVariables | A string vector to specify the column names of independent variables included in the regression model other than the variableOfInterst. |

| | |
|---|---|
| studyID | A string to specify the column name of study ID. |
| label | A string to specify the name the column that contains the cluster label or name. |
| data | A data frame containing the data. Usually a long form data frame returned by collectData. |
| CILevel | A number between 0 to 1, used to specify the confidence levels. e.g. 0.95 represent 95 percent confidence interval. |
| ifScale | A vector of two logic values, specifying if the dependent variable and the variableOfInterst should be scaled when calculating the effect size. |

## Details

The function use the model value ~ variableOfInterst + otherVariables + studyID to estimate the effect size. Use it as a screening tool. Use metaAnalysis function to analyze an effect size in more detail.

## Value

Returns data frame describing the overall effect size of variableOfInterst on value. May be slightly different from the value reported from the function metaAnalysis.

## Examples

```
library(dplyr)
#collect all summary statistics
fn=system.file("extdata","",package="MetaCyto")
files=list.files(fn,pattern="cluster_stats_in_each_sample",
                recursive=TRUE,full.names=TRUE)
fcs_stats=collectData(files,longform=TRUE)
# Collect sample information
files=list.files(fn,pattern="sample_info",recursive=TRUE,full.names=TRUE)
sample_info=collectData(files,longform=FALSE)
# join the cluster summary statistics with sample information
all_data=inner_join(fcs_stats,sample_info,by="fcs_files")
# See the fraction of what clusters are affected by
# age (while controlling for Gender)
GA=glmAnalysis(value="value",variableOfInterst="Subject Age",
              parameter="fraction",
              otherVariables=c("Gender"),studyID="study_id",label="label",
              data=all_data,CILevel=0.95,ifScale=c(TRUE,FALSE))
```

---

# labelCluster *Label each cluster as "+" or "-" or neutral for each marker*

---

## Description

A function that labels each cluster as "+" or "-" or neutral for each marker

### Usage

```
labelCluster(fcsFrame, clusterList, excludeClusterParameters = c("TIME"),
  minPercent = 0.05, labelQuantile = 0.95, cutoff = NULL)
```

### Arguments

| | |
|---|---|
| `fcsFrame` | A flowFrame object. |
| `clusterList` | A list, each element should be a vector containing the IDs of all cells that belongs to a cluster |
| `excludeClusterParameters` | A vector specifying the name of markers not to be used for labeling. |
| `minPercent` | A number between 0 and 0.5. Used to specify the minimum percent of cells in the positive and negative region after bisection. Keep it small to avoid bisecting uni-mode distributions. |
| `labelQuantile` | A number between 0.5 and 1. Used to specify the minimum percent of a cluster required to be larger or smaller than the cutoff value for labeling. |
| `cutoff` | A vector of cutoff values to bisect the distribution of each marker. The names of the vector should be the same as the marker names. If NULL, the cutoff value will be determined automatically. |

### Value

Returns a list with two components: 1) clusterLabel, contains a vector of labels, each corresponds to a cluster in clusterList. 2) cutoff, contains a vector of cutoff values used to bisect each marker.

### Examples

```
# Find fcs files
files=system.file("extdata","SDY420/ResultFiles/CyTOF_result",
                  package="MetaCyto")
files=list.files(files,pattern="fcs$",full.names=TRUE)
# Preprocess
fcs = preprocessing(fcsFiles=files,assay ="CyTOF",b=1/8)
# cluster using flowSOM.MC
cluster_list=flowSOM.MC(fcsFrame=fcs,
                        excludeClusterParameters=c("Time","Cell_length"))
# label each clusters
cluster_label=labelCluster(fcsFrame=fcs,clusterList=cluster_list,
                           excludeClusterParameters=c("Time","Cell_length"))
```

---

## `labelSummary` *Make a summary for the labels (cell populations) identified in different cytometry panels.*

---

### Description

A function that summarizes the labels (cell populations) identified in different cytometry panels.

### Usage

```
labelSummary(allData, minStudy = 2)
```

### Arguments

| | |
|---|---|
| allData | A table containing the summary statistics of cell populations. Often is the output of the function "collectData". |
| minStudy | Minimum number of cytometry panels a label should appear in. Set it >1 to find cell populations identified in more than 1 cytometry panel for meta-analysis. |

### Value

A data frame summarizing the labels identified in different cytometry panels.

### Examples

```
fn=system.file("extdata","",package="MetaCyto")
files=list.files(fn,pattern="cluster_stats_in_each_sample",
                 recursive=TRUE,full.names=TRUE)
fcs_stats=collectData(files,longform=TRUE)
label_summary = labelSummary(allData=fcs_stats,minStudy=2)
```

---

## markerFinder *Find markers in a flow frame object*

---

### Description

A function that finds markers in a flow frame object.

### Usage

```
markerFinder(fcsFrame)
```

### Arguments

| | |
|---|---|
| fcsFrame | A flow frame object. |

### Details

If the antibody name is available, the antibody name will be returned, otherwise, the channel name will be returned.

### Value

Returns a vector of markers.

## Examples

```
library(flowCore)
files=system.file("extdata","SDY420/ResultFiles/CyTOF_result",
                  package="MetaCyto")
files=list.files(files,pattern="fcs$",full.names=TRUE)[1]
fcs = read.FCS(files)
markers = markerFinder(fcs)
```

---

`metaAnalysis` *Performs meta-analysis*

---

## Description

A function that performs meta-analysis

## Usage

```
metaAnalysis(value, variableOfInterst, otherVariables, studyID, data, CILevel,
  main, ifScale = c(TRUE, FALSE), cex = 1)
```

## Arguments

| | |
|---|---|
| `value` | A string to specify the column name of the dependent variable (y) |
| `variableOfInterst` | A string to specify the column name of the independent variable of interest (x1) |
| `otherVariables` | A string vector to specify the column names of independent variables included in the regression model other than the variableOfInterst. |
| `studyID` | A string to specify the column name of study ID. |
| `data` | A data frame containing the data |
| `CILevel` | A number between 0 to 1, used to specify the confidence interval to be plotted in the forest plot. |
| `main` | A string to specify the title of the forest plot |
| `ifScale` | A vector of two logic values, specifying if the dependent variable and the variableOfInterst should be scaled when calculating the effect size. |
| `cex` | A number specifying the amount by which plotting text and symbols should be scaled relative to the default in the forest plot. |

## Value

Returns data frame describing the effect size of variableOfInterst on value in each individual studies, as well as the over all effect size. In addition, it returns the random effect model and the leave one out analysis result.

## Examples

```
library(dplyr)
#collect all summary statistics
fn=system.file("extdata","",package="MetaCyto")
files=list.files(fn,pattern="cluster_stats_in_each_sample",recursive=TRUE,
                 full.names=TRUE)
fcs_stats=collectData(files,longform=TRUE)
# Collect sample information
files=list.files(fn,pattern="sample_info",recursive=TRUE,full.names=TRUE)
sample_info=collectData(files,longform=FALSE)
# join the cluster summary statistics with sample information
all_data=inner_join(fcs_stats,sample_info,by="fcs_files")

# plot forrest plot to see if the proportion of CCR7+ CD8 T cell
# is affected by age (while controlling for Gender)
L="CD3+|CD4-|CD8+|CCR7+"
dat=subset(all_data,all_data$parameter_name=="fraction"&
            all_data$label==L)
MA=metaAnalysis(value="value",variableOfInterst="Subject Age",main=L,
                otherVariables=c("Gender"),studyID="study_id",
                data=dat,CILevel=0.95,ifScale=c(TRUE,FALSE))
```

---

`nameUpdator` *Used to update marker names*

---

## Description

A function that updates marker names in the files output by the preprocessing.batch function.

## Usage

```
nameUpdator(oldNames, newNames, files)
```

## Arguments

| | |
|---|---|
| `oldNames` | A vector of marker names you wish to change |
| `newNames` | A vector of marker names you wish each oldNames to be changed to. |
| `files` | A list of "processed_sample_summary.csv" files output by the preprocessing.batch function, in which the name change will occur. |

## Value

Null

## Examples

```
#get meta-data
fn=system.file("extdata","fcs_info.csv",package="MetaCyto")
fcs_info=read.csv(fn,stringsAsFactors=FALSE,check.names=FALSE)
fcs_info$fcs_files=system.file("extdata",fcs_info$fcs_files,
                               package="MetaCyto")
# make sure the transformation parameter "b" and the "assay" argument
# are correct of FCM and CyTOF files
b=assay=rep(NA,nrow(fcs_info))
b[grepl("CyTOF",fcs_info$study_id)]=1/8
b[grepl("FCM",fcs_info$study_id)]=1/150
assay[grepl("CyTOF",fcs_info$study_id)]="CyTOF"
assay[grepl("FCM",fcs_info$study_id)]="FCM"
# preprocessing
preprocessing.batch(inputMeta=fcs_info,
                    assay=assay,
                    b=b,
                    outpath="Example_Result/preprocess_output",
                    excludeTransformParameters=c("FSC-A","FSC-W","FSC-H",
                    "Time","Cell_length"))
# Make sure marker names are consistant in different studies
files=list.files("Example_Result",pattern="processed_sample",recursive=TRUE,
                 full.names=TRUE)
nameUpdator("CD8B","CD8",files)
```

---

panelSummary *Summarize markers in panels.*

---

## Description

A function that summarizes markers in cytometry panels.

## Usage

```
panelSummary(panelInfo, folder, cluster = TRUE, plotImage = TRUE,
  width = 20, height = 20)
```

## Arguments

| | |
|---|---|
| panelInfo | A data frame returned by the collectData function. It should contain all the information outputted by the preprocessing.batch function. |
| folder | The directory where the output should be written. |
| cluster | True or False. Used to indicate if the markers and panels should be clustered in the plot. |
| plotImage | True or False. Used to indicate if a plot summarizing markers in panels should be produced. |
| width | Used to specify the width of the plot |
| height | Used to specify the height of the plot |

### Value

A dataframe describing what markers are in each panel.

### Examples

```
fn=system.file("extdata","",package="MetaCyto")
fn=list.files(fn,pattern="processed_sample",full.names=TRUE)
panel_info=collectData(fn,longform=FALSE)
dir.create("Example_Result")
PS=panelSummary(panel_info,"Example_Result",cluster=FALSE,width=30,height=20)
```

---

## plotGA — *Plot the result from the glmAnalysis function*

---

### Description

A function that plots the result from the glmAnalysis function.

### Usage

```
plotGA(GA, size = 16)
```

### Arguments

| | |
|---|---|
| GA | A data frame returned from the function glmAnalysis. |
| size | The font size of texts in the plot |

### Value

The plot will show up automatically.

### Examples

```
library(dplyr)
#collect all summary statistics
fn=system.file("extdata","",package="MetaCyto")
files=list.files(fn,pattern="cluster_stats_in_each_sample",recursive=TRUE,
                full.names=TRUE)
fcs_stats=collectData(files,longform=TRUE)
# Collect sample information
files=list.files(fn,pattern="sample_info",recursive=TRUE,full.names=TRUE)
sample_info=collectData(files,longform=FALSE)
# join the cluster summary statistics with sample information
all_data=inner_join(fcs_stats,sample_info,by="fcs_files")
# See the fraction of what clusters are affected
# by age (while controlling for Gender)
GA=glmAnalysis(value="value",variableOfInterst="Subject Age",
             parameter="fraction",
              otherVariables=c("Gender"),studyID="study_id",label="label",
```

```
              data=all_data,CILevel=0.95,ifScale=c(TRUE,FALSE))
GA=GA[order(GA$Effect_size),]
# plot the effect sizes
plotGA(GA)
```

---

preprocessing *Preprocess fcs files from a single experiment*

---

## Description

A function that preprocesses fcs files from a single experiment.

## Usage

```
preprocessing(fcsFiles, assay = c("FCM", "CyTOF"), b = 1/200,
  fileSampleSize = 5000, compFiles = NULL,
  excludeTransformParameters = c("FSC-A", "FSC-W", "FSC-H", "Time",
  "Cell_length"))
```

## Arguments

| | |
|---|---|
| fcsFiles | A vector specifying the location of all fcs files. |
| assay | Either "FCM" or "CyTOF" to indicate the type of cytometry data. |
| b | A positive number used to specify the arcsinh transformation. f(x) = asinh (b*x) where x is the original value and f(x) is the value after transformation. The suggested value is 1/150 for flow cytometry (FCM) data and 1/8 for CyTOF data. If b = 0, the transformation is skipped. |
| fileSampleSize | An integer specifying the number of events sampled from each fcs file. If NULL, all the events will be pre-processed and wrote out to the new fcs files. |
| compFiles | A vector specifying the paths of user supplied compensation matrix for each fcs file. The matrix must be stored in csv files. |
| excludeTransformParameters | A vector specifying the name of parameters not to be transformed (left at linear scale). |

## Value

Returns a flowFrame object containing the preprocessed cytometry data. Cells from different fcs files are combined into one flow frame. A new parameter, sample_id, is introduced to indicate the origin of each cell.

## Examples

```
# Find fcs files
files=system.file("extdata","SDY420/ResultFiles/CyTOF_result",
                  package="MetaCyto")
files=list.files(files,pattern="fcs$",full.names=TRUE)
# Preprocess
fcs = preprocessing(fcsFiles=files,assay ="CyTOF",b=1/8)
```

`preprocessing.batch` *Preprocessing fcs files from different studies in batch.*

## Description

It transform and compensate for the raw fcs files and write out the processed data to a new set of fcs files.

## Usage

```
preprocessing.batch(inputMeta, assay = c("FCM", "CyTOF"), outpath,
  b = 1/150, fileSampleSize = 5000,
  excludeTransformParameters = c("FSC-A", "FSC-W", "FSC-H", "Time",
  "Cell_length"))
```

## Arguments

`inputMeta` A data frame containing 2 columns: a column called "fcs_files" that contains the location (relative to the working directory) of each fcs file on the hard drive and a column called "study_id" that specify what study each fcs file belongs to.

`assay` A vector, each element is either "FCM" or "CyTOF" to indicate the type of cytometry data.

`outpath` A string indicating the directory the pre-processed fcs files will be written to.

`b` A positive number used to specify the arcsinh transformation. f(x) = asinh (b*x) where x is the original value and f(x) is the value after transformation. The suggested value is 1/150 for flow cytometry (FCM) data and 1/8 for CyTOF data. If b = 0, the transformation is skipped.

`fileSampleSize` An integer specifying the number of events sampled from each fcs file. If NULL, all the events will be pre-processed and wrote out to the new fcs files.

`excludeTransformParameters` A vector specifying the name of parameters not to be transformed (left at linear scale).

## Details

The function takes a data frame which specifies the location of the fcs files and the panels the fcs files belong to. It transform the cytometry data using the arcsinh transformation. For flow cytometry data, it compensate the data using the compensation matrix supplied in the fcs file. the preprocessed fcs files and a table called "processed_sample_summary.csv" will be wrote out to outpath as well.

## Value

Does not return anything. The output is written to the directory specified by the "outpath".

## Examples

```
#get meta-data
fn=system.file("extdata","fcs_info.csv",package="MetaCyto")
fcs_info=read.csv(fn,stringsAsFactors=FALSE,check.names=FALSE)
fcs_info$fcs_files=system.file("extdata",fcs_info$fcs_files,package="MetaCyto")
# make sure the transformation parameter "b" and the "assay" argument are
# correct for FCM and CyTOF files
b=assay=rep(NA,nrow(fcs_info))
b[grepl("CyTOF",fcs_info$study_id)]=1/8
b[grepl("FCM",fcs_info$study_id)]=1/150
assay[grepl("CyTOF",fcs_info$study_id)]="CyTOF"
assay[grepl("FCM",fcs_info$study_id)]="FCM"
# preprocessing
preprocessing.batch(inputMeta=fcs_info,
                    assay=assay,
                    b=b,
                    outpath="Example_Result/preprocess_output",
                    excludeTransformParameters=c("FSC-A","FSC-W","FSC-H",
                    "Time","Cell_length"))
```

---

`sampleInfoParser` *Collect sample information for fcs files in a study from ImmPort.*

---

## Description

A function that collects sample information for fcs files in a study from ImmPort.

## Usage

```
sampleInfoParser(metaData, studyFolder, fcsCol = "ZBXFN", assay = "FCM",
  attrCol)
```

## Arguments

| | |
|---|---|
| `metaData` | A data frame. Must contain a column listing the names of fcs files included in the study. |
| `studyFolder` | Path of the directory containing all the files of a study from ImmPort. |
| `fcsCol` | A string specifying the name of the column in metaData that lists fcs files included in the study. |
| `assay` | Either "FCM" or "CyTOF" to indicate the type of cytometry data. |
| `attrCol` | A vector of column names. Used to specify the information about each cytometry the user wish to include in the analysis. |

## Value

A dataframe containing sample information.

## Examples

```
fn=system.file("extdata",
               "SDY736/SDY736-DR19_Subject_2_Flow_cytometry_result.txt",
               package="MetaCyto")
meta_data=read.table(fn,sep='\t',header=TRUE,check.names=FALSE)
# Find the AGE, GENDER info from selected_data
fn=system.file("extdata","SDY736",package="MetaCyto")
sample_info_SDY736=sampleInfoParser(metaData=meta_data,
                                    studyFolder=fn,
                                    assay="FCM",
                                    fcsCol="File Name",
                                    attrCol=c("Subject Age","Gender"))
```

---

# searchCluster *Search for clusters using pre-defined labels*

---

## Description

A function that searches for clusters using pre-defined labels (cell definitions).

## Usage

```
searchCluster(fcsFrame, clusterLabel, cutoff = NULL, rmNull = TRUE,
  preGate = NULL)
```

## Arguments

| | |
|---|---|
| `fcsFrame` | A flowFrame object. |
| `clusterLabel` | A vector of labels, such as "CD3+\|CD4+\|CD8-". Each marker is followed by "+" or "-" and are separated by "\|". |
| `cutoff` | A vector of cutoff values to bisect the distribution of each marker. The names of the vector should be the same as the marker names. If NULL, the cutoff value will be determined automatically. |
| `rmNull` | True or False. Used to specify if a cluster with 0 cells should be returned or not. |
| `preGate` | A character string specifying the gated used to clean up the data. For example, use "PI-" to only analyze live cell. Or use "Cell_Length+" to only analyze non-debris. |

## Value

Returns a list with two components: 1) clusterList is a list in which each element of the list is a vector containing the ID of all cells in a cluster. The names correspond to the labels specified in clusterLabel. 2) cutoff, contains a vector of cutoff values used to bisect each marker.

## Examples

```
# Find fcs files
files=system.file("extdata","SDY420/ResultFiles/CyTOF_result",
                  package="MetaCyto")
files=list.files(files,pattern="fcs$",full.names=TRUE)
# Preprocess
fcs = preprocessing(fcsFiles=files,assay ="CyTOF",b=1/8)
# Search clusters
cluster_list=searchCluster(fcsFrame=fcs,
                           clusterLabel=c("CD3+|CD8+","CD3-|CD19+"))
```

---

# searchCluster.batch *Search for clusters using pre-defined labels in cytometry data from different studies in batch*

---

## Description

A function that searches for clusters using pre-defined labels in cytometry data from different studies in batch.

## Usage

```
searchCluster.batch(preprocessOutputFolder, outpath = "search_output",
  clusterLabel, ifPlot = TRUE)
```

## Arguments

| | |
|---|---|
| `preprocessOutputFolder` | Directory where the pre-processed results are stored. |
| `outpath` | A string indicating the directory the results should be written to. |
| `clusterLabel` | A vector containing labels, such as c("CD3+\|CD4+\|CD8-") |
| `ifPlot` | True or False. Used to specify if a the density plot for each cluster should be plotted |

## Details

The function writes out the summary statistics for each cluster. A separate directory will be created for each study.

## Value

Results will be written to the outpath folder

## Examples

```
#get meta-data
fn=system.file("extdata","fcs_info.csv",package="MetaCyto")
fcs_info=read.csv(fn,stringsAsFactors=FALSE,check.names=FALSE)
fcs_info$fcs_files=system.file("extdata",fcs_info$fcs_files,
                               package="MetaCyto")
# Make sure the transformation parameter "b" and the "assay" argument
# are correct of FCM and CyTOF files
b=assay=rep(NA,nrow(fcs_info))
b[grepl("CyTOF",fcs_info$study_id)]=1/8
b[grepl("FCM",fcs_info$study_id)]=1/150
assay[grepl("CyTOF",fcs_info$study_id)]="CyTOF"
assay[grepl("FCM",fcs_info$study_id)]="FCM"
# preprocessing
preprocessing.batch(inputMeta=fcs_info,
                    assay=assay,
                    b=b,
                    outpath="Example_Result/preprocess_output",
                    excludeTransformParameters=c("FSC-A","FSC-W","FSC-H",
                    "Time","Cell_length"))
# Make sure marker names are consistant in different studies
files=list.files("Example_Result",pattern="processed_sample",recursive=TRUE,
                 full.names=TRUE)
nameUpdator("CD8B","CD8",files)
# find the clusters
cluster_label=c("CD3-|CD19+","CD3+|CD4-|CD8+")
searchCluster.batch(preprocessOutputFolder="Example_Result/preprocess_output",
                    outpath="Example_Result/search_output",
                    clusterLabel=cluster_label)
```

---

`set2Frame` *Combine cells in a flow set into a flow frame.*

---

## Description

A function that combines cells in a flow set into a flow frame.

## Usage

```
set2Frame(flowSet)
```

## Arguments

| | |
|---|---|
| `flowSet` | A flow set object |

## Value

Returns a flowFrame object. All cells from flow set are combined into one flow frame. A new parameter, sample_id, is introduced to indicate the origin of each cell.

## Examples

```
library(flowCore)
files=system.file("extdata","SDY420/ResultFiles/CyTOF_result",
                  package="MetaCyto")
files=list.files(files,pattern="fcs$",full.names=TRUE)
flow_set = read.flowSet(files)
flow_frame = set2Frame(flow_set)
```

# Index

autoCluster.batch, 2

clusterStats, 4
collectData, 5

densityPlot, 5

fcsInfoParser, 6
filterLabels, 7
findCutoff, 8
flowHC, 9
flowSOM.MC, 9

glmAnalysis, 10

labelCluster, 11
labelSummary, 12

markerFinder, 13
metaAnalysis, 14

nameUpdator, 15

panelSummary, 16
plotGA, 17
preprocessing, 18
preprocessing.batch, 19

sampleInfoParser, 20
searchCluster, 21
searchCluster.batch, 22
set2Frame, 23