

# Package ‘FlowSOM’

May 10, 2024

**Version** 2.12.0

**Date** 2023-04-21

**Title** Using self-organizing maps for visualization and interpretation of cytometry data

**Depends** R (>= 4.0), igraph

**Imports** stats, utils, colorRamps, ConsensusClusterPlus, dplyr, flowCore, ggforce, ggnewscale, ggplot2, ggpubr, grDevices, magrittr, methods, rlang, Rtsne, tidyr, BiocGenerics, XML

**Suggests** BiocStyle, testthat, CytoML, flowWorkspace, ggrepel, scattermore, pheatmap, ggpointdensity

**Description** FlowSOM offers visualization options for cytometry data, by using Self-Organizing Map clustering and Minimal Spanning Trees.

**License** GPL (>= 2)

**LazyData** true

**URL** <http://www.r-project.org>, <http://dambi.ugent.be>

**biocViews** CellBiology, FlowCytometry, Clustering, Visualization, Software, CellBasedAssays

**RoxygenNote** 7.2.3

**Encoding** UTF-8

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

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

**git\_last\_commit** 128641a

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

**Repository** Bioconductor 3.19

**Date/Publication** 2024-05-10

**Author** Sofie Van Gassen [aut, cre],  
Artuur Couckuyt [aut],  
Katrien Quintelier [aut],  
Annelies Emmaneel [aut],  
Britt Callebaut [aut],  
Yvan Saeys [aut]

**Maintainer** Sofie Van Gassen <sofie.vangassen@ugent.be>

## Contents

AddAnnotation . . . . .	4
AddBackground . . . . .	5
AddFlowFrame . . . . .	6
AddLabels . . . . .	6
AddMST . . . . .	7
AddNodes . . . . .	8
AddPies . . . . .	9
AddScale . . . . .	9
AddStars . . . . .	10
AddStarsPies . . . . .	11
AggregateFlowFrames . . . . .	11
AutoMaxNodeSize . . . . .	13
BuildMST . . . . .	13
BuildSOM . . . . .	14
CountGroups . . . . .	15
Dist.MST . . . . .	16
FlowSOM . . . . .	17
FlowSOMmary . . . . .	19
FlowSOMsubset . . . . .	20
FlowSOM_colors . . . . .	21
FMeasure . . . . .	21
GetChannels . . . . .	22
GetClusterCVs . . . . .	23
GetClusterMFIs . . . . .	23
GetClusterPercentagesPositive . . . . .	24
GetClusters . . . . .	25
GetCounts . . . . .	25
GetCVs . . . . .	26
GetFeatures . . . . .	27
GetFlowJoLabels . . . . .	28
GetMarkers . . . . .	30
GetMetaclusterCVs . . . . .	31
GetMetaclusterMFIs . . . . .	31
GetMetaclusterPercentagesPositive . . . . .	32
GetMetaclusters . . . . .	33
GetMFIs . . . . .	34
GetPercentages . . . . .	35
get_channels . . . . .	35
get_markers . . . . .	36
gg_color_hue . . . . .	37
GroupStats . . . . .	37
Initialize_KWSP . . . . .	39
Initialize_PCA . . . . .	40
ManualVector . . . . .	41

MapDataToCodes . . . . .	41
MetaclusterCVs . . . . .	42
MetaClustering . . . . .	42
metaClustering_consensus . . . . .	43
MetaclusterMFIs . . . . .	44
NClusters . . . . .	45
NewData . . . . .	45
NMetaclusters . . . . .	47
ParseArcs . . . . .	48
ParseEdges . . . . .	48
ParseLayout . . . . .	49
ParseNodeSize . . . . .	49
ParseQuery . . . . .	50
ParseSD . . . . .	51
Plot2DScatters . . . . .	51
PlotCenters . . . . .	53
PlotClusters2D . . . . .	54
PlotDimRed . . . . .	56
PlotFileScatters . . . . .	57
PlotFlowSOM . . . . .	59
PlotGroups . . . . .	61
PlotLabels . . . . .	63
PlotManualBars . . . . .	64
PlotMarker . . . . .	65
PlotNode . . . . .	67
PlotNumbers . . . . .	68
PlotOutliers . . . . .	69
PlotOverview2D . . . . .	70
PlotPies . . . . .	71
PlotSD . . . . .	73
PlotStarLegend . . . . .	74
PlotStars . . . . .	75
PlotVariable . . . . .	76
print.FlowSOM . . . . .	77
Purity . . . . .	78
QueryMultiple . . . . .	78
QueryStarPlot . . . . .	79
query_multiple . . . . .	81
ReadInput . . . . .	82
SaveClustersToFCS . . . . .	83
ScaleStarHeights . . . . .	84
SOM . . . . .	85
TestOutliers . . . . .	86
UpdateFlowSOM . . . . .	88
UpdateMetaclusters . . . . .	88
UpdateNodeSize . . . . .	90
%>% . . . . .	91

---

AddAnnotation

*AddAnnotation*


---

## Description

Add annotation to a FlowSOM plot

## Usage

```
AddAnnotation(
  p,
  fsom,
  toAnnotate = NULL,
  prefix = list(metaclusters = "MCL ", clusters = "CL "),
  ...
)
```

## Arguments

<code>p</code>	Plot to add annotation to. When using <a href="#">PlotStars</a> , please use <code>list_insteadof_ggarrange = TRUE</code> .
<code>fsom</code>	FlowSOM object that goes with the plot.
<code>toAnnotate</code>	A named list with "metaclusters" and/or "clusters" as names and a vector with the (meta)clusters that need to be annotated. Names can be abbreviated. Use a named vector with the old names as values and new labels as names for custom labeling.
<code>prefix</code>	Prefix to be added to labels. Default is "MCL " and "CL " for metaclusters and clusters respectively.
<code>...</code>	Arguments passed to <code>geom_text_repel</code> .

## Value

The updated plot

## Examples

```
# Identify the files
fcs <- flowCore::read.FCS(system.file("extdata", "68983.fcs",
                                     package = "FlowSOM"))

# Build a FlowSOM object
flowSOM.res <- FlowSOM(fcs,
  scale = TRUE,
  compensate = TRUE,
  transform = TRUE,
  toTransform = 8:18,
  colsToUse = c(9, 12, 14:18),
  nClus = 10,
```

```

seed = 1)

p <- PlotStars(flowSOM.res, backgroundValues = flowSOM.res$metaclustering,
  list_insteadof_ggarrange = TRUE)
annotationList <- list("metaclusters" = c("CD8 T cells" = "1", "B cells" = "8"),
  "clusters" = c(97))
AddAnnotation(p, flowSOM.res, toAnnotate = annotationList,
  prefix = list("metaclusters" = "", clusters = "CL "))

```

---

AddBackground

*AddBackground*


---

## Description

Function plots the background

## Usage

```

AddBackground(
  p,
  backgroundValues,
  backgroundColors = NULL,
  backgroundLim = NULL
)

```

## Arguments

**p** ggplot object

**backgroundValues** Vector of values to be plotted as background for the nodes

**backgroundColors** Color palette to be used for the background coloring. Can be either a function or an array specifying colors.

**backgroundLim** Background limits (can be used to ensure consistent Color palette between plots). If NULL (default), will be automatically adapted to the data.

## Value

Returns nothing, but plots the background

## See Also

[PlotFlowSOM](#), [AddLabels](#), [AddNodes](#), [AddPies](#), [AddStars](#)

AddFlowFrame

*Add a flowFrame to the data variable of the FlowSOM object*

---

**Description**

Add a flowFrame to the data variable of the FlowSOM object

**Usage**

```
AddFlowFrame(fsom, flowFrame)
```

**Arguments**

fsom	FlowSOM object, as constructed by the ReadInput function
flowFrame	flowFrame to add to the FlowSOM object

**Value**

FlowSOM object with data added

**See Also**

[ReadInput](#)

---

AddLabels*AddLabels*

---

**Description**

AddLabels

**Usage**

```
AddLabels(  
  p,  
  labels,  
  hjust = 0.5,  
  layout = NULL,  
  textSize = 3.88,  
  textColor = "black",  
  ...  
)
```

**Arguments**

p	ggplot object
labels	Labels to be added to each node
hjust	Horizontal adjust for labels. Default is centered.
layout	Dataframe with x and y columns. If null, the dataframe from the ggplot object will be reused.
textSize	Size for geom_text. Default (=3.88) is from geom_text.
textColor	Color for geom_text. Default = black.
...	Additional parameters to pass to geom_text

**Value**

Returns the ggplot object with labels added

**See Also**

[PlotLabels](#), [PlotNumbers](#)

---

AddMST

*AddMST*

---

**Description**

Function plots the MST

**Usage**

```
AddMST(p, fsom)
```

**Arguments**

p	ggplot object
fsom	FlowSOM object, as generated by <a href="#">FlowSOM</a>

**Value**

Returns nothing, but plots the MST for FlowSOM MST view

**See Also**

[PlotFlowSOM](#), [ParseEdges](#), [AddStarsPies](#), [AddLabels](#), [AddNodes](#), [AddBackground](#), [AddPies](#), [AddStars](#)

---

 AddNodes

*AddNodes*


---

### Description

Function plots the nodes

### Usage

```
AddNodes(
  p,
  nodeInfo = NULL,
  values = NULL,
  lim = NULL,
  colorPalette = NULL,
  fillColor = "white",
  showLegend = TRUE,
  label = "",
  ...
)
```

### Arguments

<code>p</code>	ggplot object
<code>nodeInfo</code>	Dataframe with for every node an x, y and size value, if null the dataframe from the ggplot object will be reused.
<code>values</code>	Values used for coloring the nodes. Default = NULL, in which case all nodes are filled in <code>fillColor</code> .
<code>lim</code>	The limits of the color scale, not used if <code>values = NULL</code> .
<code>colorPalette</code>	Color palette for color in nodes, not used if <code>values = NULL</code> . A vector of colors or a color function.
<code>fillColor</code>	Fixed fill for node colors, default = white.
<code>showLegend</code>	Boolean, default = TRUE.
<code>label</code>	Title for the legend.
<code>...</code>	Additional arguments to pass to <code>geom_circle</code>

### Value

Returns nothing, but plots the nodes

### See Also

[PlotFlowSOM](#), [PlotMarker](#), [PlotVariable](#), [AddLabels](#), [AddBackground](#), [AddPies](#), [AddStars](#), [AddStarsPies](#)



---

AddPies	<i>AddPies</i>
---------	----------------

---

**Description**

Function plots the pies

**Usage**

```
AddPies(p, fsom, cellLabels, layout = NULL, colorPalette = NULL)
```

**Arguments**

p	ggplot object
fsom	FlowSOM object, as generated by <a href="#">BuildMST</a>
cellLabels	Array of factors indicating the cell labels
layout	Coordinates of nodes. Uses dataframe of the ggplot object if NULL.
colorPalette	Color palette to be used for colors. Can be either a function or an array specifying colors.

**Value**

ggplot object with the pies added

**See Also**

[PlotFlowSOM](#), [AddLabels](#), [AddNodes](#), [AddBackground](#), [PlotPies](#), [AddStars](#), [ParseArcs](#)

---

AddScale	<i>AddScale</i>
----------	-----------------

---

**Description**

AddScale

**Usage**

```
AddScale(  
  p,  
  values = NULL,  
  colors = NULL,  
  limits = NULL,  
  showLegend = TRUE,  
  labelLegend = "",  
  type = "fill"  
)
```

**Arguments**

p	ggplot object
values	Values used for the fill
colors	Colors to use (can be a vector or a function)
limits	Limits to use in the scale
showLegend	Boolean on whether to show the legend
labelLegend	Label to show as title of the legend
type	fill (default) or color

**Value**

ggplot object with scale added

---

AddStars

*AddStars*

---

**Description**

Function plots the stars

**Usage**

```
AddStars(p, fsom, markers = fsom$map$colsUsed, colorPalette = NULL)
```

**Arguments**

p	ggplot object
fsom	FlowSOM object, as generated by <a href="#">BuildMST</a>
markers	Determines which markers to plot. Default = "fsom\$map\$colsUsed"
colorPalette	Color palette to be used for colors. Can be either a function or an array specifying colors.

**Value**

ggplot object with the stars added

**See Also**

[PlotFlowSOM](#), [AddLabels](#), [AddNodes](#), [AddBackground](#), [PlotStars](#), [AddPies](#), [ParseArcs](#)

---

AddStarsPies	<i>AddStarsPies</i>
--------------	---------------------

---

**Description**

Function plots stars or pies

**Usage**

```
AddStarsPies(p, arcs, colorPalette, showLegend = TRUE)
```

**Arguments**

p	ggplot object
arcs	Dataframe that contains all the data for the plotting the pies or stars
colorPalette	A vector of colors or a color function
showLegend	Boolean on whether to show the legend

**Value**

Returns nothing, but plots the stars or pies

**See Also**

[PlotFlowSOM](#), [AddLabels](#), [AddNodes](#), [AddBackground](#), [AddPies](#), [AddStars](#), [ParseArcs](#), [PlotStars](#), [PlotPies](#)

---

AggregateFlowFrames	<i>Aggregate multiple FCS files together</i>
---------------------	--

---

**Description**

Aggregate multiple FCS files to analyze them simultaneously. A new FCS file is written, which contains about `cTotal` cells, with `ceiling(cTotal/nFiles)` cells from each file. Two new columns are added: a column indicating the original file by index, and a noisy version of this for better plotting opportunities (index plus or minus a value between 0 and 0.1).

**Usage**

```
AggregateFlowFrames(
  fileNames,
  cTotal,
  channels = NULL,
  writeOutput = FALSE,
  outputFile = "aggregate.fcs",
  keepOrder = FALSE,
  silent = FALSE,
  sampleWithReplacement = FALSE,
  ...
)
```

**Arguments**

<code>fileNames</code>	Character vector containing full paths to the FCS files or a flowSet to aggregate
<code>cTotal</code>	Total number of cells to write to the output file
<code>channels</code>	Channels/markers to keep in the aggregate. Default NULL takes all channels of the first file.
<code>writeOutput</code>	Whether to write the resulting flowFrame to a file. Default FALSE
<code>outputFile</code>	Full path to output file. Default "aggregate.fcs"
<code>keepOrder</code>	If TRUE, the random subsample will be ordered in the same way as they were originally ordered in the file. Default = FALSE.
<code>silent</code>	If FALSE, prints an update every time it starts processing a new file. Default = FALSE.
<code>sampleWithReplacement</code>	If TRUE and more cells per file are requested than actually present, all cells will be included plus additional resampling. Otherwise, at most all cells will be included once. Default = FALSE.
<code>...</code>	Additional arguments to pass to read.FCS

**Value**

This function does not return anything, but will write a file with about `cTotal` cells to `outputFile`

**See Also**

[ceiling](#)

**Examples**

```
# Define filename
fileName <- system.file("extdata", "68983.fcs", package = "FlowSOM")
# This example will sample 2 times 500 cells.
ff_new <- AggregateFlowFrames(c(fileName, fileName), 1000)
```

---

AutoMaxNodeSize	<i>AutoMaxNodeSize</i>
-----------------	------------------------

---

**Description**

Calculate node size

**Usage**

```
AutoMaxNodeSize(layout, overlap)
```

**Arguments**

layout	Coordinates of nodes
overlap	Parameter that determines how much overlap there will be. If negative the nodes will be smaller

**Details**

Function that calculates the minimum distance between the nodes to use this to adapt the maxNodeSize for better plotting

**Value**

Returns the maxNodeSize with some overlap

**See Also**

[PlotFlowSOM](#), [ScaleStarHeights](#), [ParseNodeSize](#)

---

BuildMST	<i>BuildMST</i>
----------	-----------------

---

**Description**

Build Minimal Spanning Tree

**Usage**

```
BuildMST(fsom, silent = FALSE, tSNE = FALSE)
```

**Arguments**

fsom	FlowSOM object, as generated by <a href="#">BuildSOM</a>
silent	If TRUE, no progress updates will be printed
tSNE	If TRUE, an alternative t-SNE layout is computed as well

**Details**

Add minimal spanning tree description to the FlowSOM object

**Value**

FlowSOM object containing MST description

**See Also**

[BuildSOM](#), [PlotStars](#)

**Examples**

```
# Read from file, build self-organizing map
fileName <- system.file("extdata", "68983.fcs", package="FlowSOM")
flowSOM.res <- ReadInput(fileName, compensate=TRUE, transform = TRUE,
                        scale = TRUE)
flowSOM.res <- BuildSOM(flowSOM.res, colsToUse = c(9, 12, 14:18))

# Build the Minimal Spanning Tree
flowSOM.res <- BuildMST(flowSOM.res)
```

---

BuildSOM

*Build a self-organizing map*

---

**Description**

Build a SOM based on the data contained in the FlowSOM object

**Usage**

```
BuildSOM(fsom, colsToUse = NULL, silent = FALSE, outlierMAD = 4, ...)
```

**Arguments**

<code>fsom</code>	FlowSOM object containing the data, as constructed by the <a href="#">ReadInput</a> function
<code>colsToUse</code>	Markers, channels or indices to use for building the SOM
<code>silent</code>	if TRUE, no progress updates will be printed
<code>outlierMAD</code>	Number of MAD when a cell is considered an outlier. See also <a href="#">TestOutliers</a>
<code>...</code>	options to pass on to the SOM function ( <code>xdim</code> , <code>ydim</code> , <code>rlen</code> , <code>mst</code> , <code>alpha</code> , <code>radius</code> , <code>init</code> , <code>distf</code> , <code>importance</code> )

**Value**

FlowSOM object containing the SOM result, which can be used as input for the [BuildMST](#) function

**References**

This code is strongly based on the kohonen package. R. Wehrens and L.M.C. Buydens, Self- and Super-organising Maps in R: the kohonen package J. Stat. Softw., 21(5), 2007

**See Also**

[ReadInput](#), [BuildMST](#)

**Examples**

```
# Read from file
fileName <- system.file("extdata", "68983.fcs", package = "FlowSOM")
flowSOM.res <- ReadInput(fileName, compensate = TRUE, transform = TRUE,
                        scale = TRUE)

# Build the Self-Organizing Map
# E.g. with gridsize 5x5, presenting the dataset 20 times,
# no use of MST in neighborhood calculations in between
flowSOM.res <- BuildSOM(flowSOM.res, colsToUse = c(9, 12, 14:18),
                      xdim = 5, ydim = 5, rlen = 20)

# Build the minimal spanning tree and apply metaclustering
flowSOM.res <- BuildMST(flowSOM.res)
metacl <- MetaClustering(flowSOM.res$map$codes,
                        "metaClustering_consensus", max = 10)
```

---

CountGroups

*Calculate differences in cell counts between groups*

---

**Description**

Calculate differences in cell counts between groups

**Usage**

```
CountGroups(fsom, groups, plot = TRUE, silent = FALSE)
```

**Arguments**

fsom	FlowSOM object as generated by BuildSOM
groups	List containing an array with file names for each group
plot	Logical. If TRUE, make a starplot of each individual file
silent	Logical. If TRUE, print progress messages

**Value**

Distance matrix

**See Also**

GroupStats

**Examples**

```

set.seed(1)
fileName <- system.file("extdata", "68983.fcs", package="FlowSOM")
flowSOM.res <- FlowSOM(fileName, compensate = TRUE, transform = TRUE,
                      scale = TRUE, colsToUse = c(9,12,14:18), nClus = 10)

ff <- flowCore::read.FCS(fileName)
# Make an additional file without cluster 7 and double amount of cluster 5
selection <- c(which(GetClusters(flowSOM.res) %in%
                    which(flowSOM.res$metaclustering != 7)),
              which(GetClusters(flowSOM.res) %in%
                    which(flowSOM.res$metaclustering == 5)))

ff_tmp <- ff[selection,]
flowCore::write.FCS(ff_tmp, file="ff_tmp.fcs")

# Compare only the file with the double amount of cluster 10
features <- GetFeatures(flowSOM.res,
                      c(fileName, "ff_tmp.fcs"),
                      level = "clusters",
                      type = "percentages")
stats <- GroupStats(features$cluster_percentages,
                    groups = list("AllCells" = c(fileName),
                                  "Without_ydTcells" = c("ff_tmp.fcs")))

```

Dist.MST

---

*Calculate distance matrix using a minimal spanning tree neighborhood*

---

**Description**

Calculate distance matrix using a minimal spanning tree neighborhood

**Usage**

```
Dist.MST(X)
```

**Arguments**

X matrix in which each row represents a point

**Value**

Distance matrix



FlowSOM

*Run the FlowSOM algorithm***Description**

Method to run general FlowSOM workflow. Will scale the data and uses consensus meta-clustering by default.

**Usage**

```
FlowSOM(
  input,
  pattern = ".fcs",
  compensate = FALSE,
  spillover = NULL,
  transform = FALSE,
  toTransform = NULL,
  transformFunction = flowCore::logicleTransform(),
  transformList = NULL,
  scale = FALSE,
  scaled.center = TRUE,
  scaled.scale = TRUE,
  silent = TRUE,
  colsToUse = NULL,
  nClus = 10,
  maxMeta = NULL,
  importance = NULL,
  seed = NULL,
  ...
)
```

**Arguments**

<code>input</code>	a <code>flowFrame</code> , a <code>flowSet</code> , a matrix with column names or an array of paths to files or directories
<code>pattern</code>	if <code>input</code> is an array of file- or directorynames, select only files containing <code>pattern</code>
<code>compensate</code>	logical, does the data need to be compensated
<code>spillover</code>	spillover matrix to compensate with If <code>NULL</code> and <code>compensate = TRUE</code> , we will look for <code>\$SPILL</code> description in FCS file.
<code>transform</code>	logical, does the data need to be transformed with the transformation given in <code>transformFunction</code> .
<code>toTransform</code>	column names or indices that need to be transformed. Will be ignored if <code>transformList</code> is given. If <code>NULL</code> and <code>transform = TRUE</code> , column names of <code>\$SPILL</code> description in FCS file will be used.
<code>transformFunction</code>	Defaults to <code>logicleTransform()</code>

<code>transformList</code>	transformList to apply on the samples.
<code>scale</code>	logical, does the data needs to be rescaled. Default = FALSE
<code>scaled.center</code>	see <a href="#">scale</a>
<code>scaled.scale</code>	see <a href="#">scale</a>
<code>silent</code>	if TRUE, no progress updates will be printed
<code>colsToUse</code>	Markers, channels or indices to use for building the SOM. Default (NULL) is all the columns used to build the FlowSOM object.
<code>nClus</code>	Exact number of clusters for meta-clustering. Ignored if maxMeta is specified. Default = 10.
<code>maxMeta</code>	Maximum number of clusters to try out for meta-clustering. If NULL (default), only one option will be computed (nClus).
<code>importance</code>	array with numeric values. Parameters will be scaled according to importance
<code>seed</code>	Set a seed for reproducible results
<code>...</code>	options to pass on to the SOM function (xdim, ydim, rlen, mst, alpha, radius, init, distf)

### Value

A list with two items: the first is the flowSOM object containing all information (see the vignette for more detailed information about this object), the second is the metaclustering of the nodes of the grid. This is a wrapper function for [ReadInput](#), [BuildSOM](#), [BuildMST](#) and [MetaClustering](#). Executing them separately may provide more options.

### See Also

[scale](#), [ReadInput](#), [BuildSOM](#), [BuildMST](#), [MetaClustering](#)

### Examples

```
# Read from file
fileName <- system.file("extdata", "68983.fcs", package = "FlowSOM")
flowSOM.res <- FlowSOM(fileName, compensate = TRUE, transform = TRUE,
                      scale = TRUE, colsToUse = c(9, 12, 14:18), nClus = 10)
# Or read from flowFrame object
ff <- flowCore::read.FCS(fileName)
ff <- flowCore::compensate(ff, flowCore::keyword(ff)[["SPILL"]])
ff <- flowCore::transform(ff,
                          flowCore::transformList(colnames(flowCore::keyword(ff)[["SPILL"]]),
                                                    flowCore::logicIcTransform()))
flowSOM.res <- FlowSOM(ff,
                      scale = TRUE,
                      colsToUse = c(9, 12, 14:18),
                      nClus = 10)

# Plot results
PlotStars(flowSOM.res,
          backgroundValues = flowSOM.res$metaclustering)
```



---

FlowSOMSubset	<i>FlowSOMSubset</i>
---------------	----------------------

---

**Description**

FlowSOM subset

**Usage**

```
FlowSOMSubset(fsom, ids)
```

**Arguments**

<code>fsom</code>	FlowSOM object, as generated by <a href="#">BuildMST</a>
<code>ids</code>	Array containing the ids to keep

**Details**

Take a subset from a FlowSOM object

**Value**

FlowSOM object containing updated data and median values, but with the same grid

**See Also**

[BuildMST](#)

**Examples**

```
# Read two files (Artificially, as we just split 1 file in 2 subsets)
fileName <- system.file("extdata", "68983.fcs", package = "FlowSOM")
ff1 <- flowCore::read.FCS(fileName)[1:1000, ]
flowCore::keyword(ff1)[["FIL"]] <- "File1"
ff2 <- flowCore::read.FCS(fileName)[1001:2000, ]
flowCore::keyword(ff2)[["FIL"]] <- "File2"

flowSOM.res <- FlowSOM(flowCore::flowSet(c(ff1, ff2)), compensate = TRUE,
                      transform = TRUE, scale = TRUE,
                      colsToUse = c(9, 12, 14:18), maxMeta = 10)

# see $metadata for subsets:
flowSOM.res$metadata

# Use only the second file, without changing the map
fSOM2 <- FlowSOMSubset(flowSOM.res,
                      (flowSOM.res$metadata[[2]][1]):
                      (flowSOM.res$metadata[[2]][2]))
```

---

FlowSOM_colors	<i>FlowSOM default colors</i>
----------------	-------------------------------

---

**Description**

FlowSOM default colors

**Usage**

```
FlowSOM_colors(n)
```

**Arguments**

n                      Number of colors to generate

**Value**

array of n colors

---

FMeasure	<i>F measure</i>
----------	------------------

---

**Description**

Compute the F measure between two clustering results

**Usage**

```
FMeasure(realClusters, predictedClusters, silent = FALSE)
```

**Arguments**

realClusters      Array containing real cluster labels for each sample

predictedClusters

Array containing predicted cluster labels for each sample

silent             Logical, if FALSE (default), print some information about precision and recall

**Value**

F measure score

**Examples**

```
# Generate some random data as an example
realClusters <- sample(1:5,100,replace = TRUE)
predictedClusters <- sample(1:6, 100, replace = TRUE)

# Calculate the FMeasure
FMeasure(realClusters,predictedClusters)
```

---

 GetChannels

*GetChannels*


---

**Description**

Get channel names for an array of markers, given a flowFrame or a FlowSOM object. As available in "name". [grep](#) is used to look for the markers. Other regex can be added.

**Usage**

```
GetChannels(object, markers, exact = TRUE)
```

**Arguments**

object	The flowFrame or the FlowSOM object of interest
markers	Vector with markers or channels of interest. Also accepts the index of the marker found in the object.
exact	If TRUE (default), the grep pattern will be extended to start with <code>^\Q</code> and end with <code>\E\$</code> , so only exact matches are possible.

**Value**

Corresponding channel names

**See Also**

[GetMarkers](#)

**Examples**

```
# Read the flowFrame
fileName <- system.file("extdata", "68983.fcs", package = "FlowSOM")
ff <- flowCore::read.FCS(fileName)
GetChannels(ff, c("FSC-A", "CD3", "FITC-A"))
GetMarkers(ff, c("FSC-A", "CD3", "FITC-A"))
```

---

GetClusterCVs                      *Get CV values for all clusters*

---

**Description**

Get CV values for all clusters

**Usage**

```
GetClusterCVs(fsom)
```

**Arguments**

fsom                      FlowSOM object as generated by the FlowSOM function or the BuildSOM function

**Value**

Matrix with coefficient of variation values for each marker

```
fileName <- system.file("extdata", "68983.fcs", package = "FlowSOM")
flowSOM.res <- FlowSOM(fileName, compensate = TRUE, transform = TRUE, scale = TRUE, colsToUse = c(9, 12, 14:18), nClus = 10)
cvs <- GetClusterCVs(flowSOM.res)
```

---

GetClusterMFIs                      *Get MFI values for all clusters*

---

**Description**

Get MFI values for all clusters

**Usage**

```
GetClusterMFIs(fsom, colsUsed = FALSE, prettyColnames = FALSE)
```

**Arguments**

fsom                      FlowSOM object as generated by the FlowSOM function or the BuildSOM function

colsUsed                  logical. Should report only the columns used to build the SOM. Default = FALSE.

prettyColnames          logical. Should report pretty column names instead of standard column names. Default = FALSE.

**Value**

Matrix with median values for each marker

**Examples**

```

fileName <- system.file("extdata", "68983.fcs", package = "FlowSOM")
flowSOM.res <- FlowSOM(fileName, compensate = TRUE, transform = TRUE,
                      scale = TRUE, colsToUse = c(9, 12, 14:18), nClus = 10)
mfis <- GetClusterMFIs(flowSOM.res)

```

---

GetClusterPercentagesPositive

*Get percentage-positive values for all clusters*

---

**Description**

Get percentage-positive values for all clusters

**Usage**

```

GetClusterPercentagesPositive(
  fsom,
  cutoffs,
  colsUsed = FALSE,
  prettyColnames = FALSE
)

```

**Arguments**

fsom	FlowSOM object as generated by the FlowSOM function or the BuildSOM function
cutoffs	named numeric vector. Upper bounds of negative population fluorescence-intensity values for each marker / channel.
colsUsed	logical. Should report only the columns used to build the SOM. Default = FALSE.
prettyColnames	logical. Should report pretty column names instead of standard column names. Default = FALSE.

**Value**

Matrix with percentages of cells that are positive in selected markers per each cluster

**Examples**

```

fileName <- system.file("extdata", "68983.fcs", package = "FlowSOM")
flowSOM.res <- FlowSOM(fileName, compensate = TRUE, transform = TRUE,
                      scale = TRUE, colsToUse = c(9, 12, 14:18), nClus = 10)
perc_pos <- GetClusterPercentagesPositive(flowSOM.res, cutoffs = c('CD4' = 5000))

```



---

GetClusters	<i>Get cluster label for all individual cells</i>
-------------	---

---

**Description**

Get cluster label for all individual cells

**Usage**

```
GetClusters(fsom)
```

**Arguments**

fsom	FlowSOM object as generated by the FlowSOM function or the BuildSOM function
------	--

**Value**

vector label for every cell

**Examples**

```
fileName <- system.file("extdata", "68983.fcs", package = "FlowSOM")
flowSOM.res <- FlowSOM(fileName, compensate = TRUE, transform = TRUE,
                      scale = TRUE, colsToUse = c(9, 12, 14:18), nClus = 10)
cluster_labels <- GetClusters(flowSOM.res)
```

---

GetCounts	<i>GetCounts</i>
-----------	------------------

---

**Description**

Get counts of number of cells in clusters or metaclusters

**Usage**

```
GetCounts(fsom, level = "metaclusters")
```

**Arguments**

fsom	FlowSOM object
level	Character string, should be either "clusters" or "metaclusters" (default) or abbreviations.

**Value**

A named vector with the counts

**Examples**

```
# Read from file, build self-organizing map and minimal spanning tree
fileName <- system.file("extdata", "68983.fcs", package = "FlowSOM")
ff <- flowCore::read.FCS(fileName)
ff <- flowCore::compensate(ff, flowCore::keyword(ff)[["SPILL"]])
ff <- flowCore::transform(ff, flowCore::estimateLogicle(ff,
                                                    flowCore::colnames(ff)[8:18]))

flowSOM.res <- FlowSOM(ff,
                      scale = TRUE,
                      colsToUse = c(9, 12, 14:18),
                      nClus = 10,
                      seed = 1)

GetCounts(flowSOM.res)
GetCounts(flowSOM.res, level = "clusters")
```

---

GetCVs

*Get CV values for all clusters*

---

**Description**

Get CV values for all clusters

**Usage**

```
GetCVs(fsom)
```

**Arguments**

fsom	FlowSOM object as generated by the FlowSOM function or the BuildSOM function
------	--

**Value**

Matrix with coefficient of variation values for each marker

```
fileName <- system.file("extdata", "68983.fcs", package="FlowSOM")
flowSOM.res <- FlowSOM(fileName,
                      compensate=TRUE,transform=TRUE, scale=TRUE,colsToUse=c(9,12,14:18),nClus=10)
cvs <- GetClusterCVs(flowSOM.res)
```

---

 GetFeatures

*GetFeatures*


---

### Description

Map FCS files on an existing FlowSOM object

### Usage

```
GetFeatures(
  fsom,
  files,
  level = c("clusters", "metaclusters"),
  type = "counts",
  MFI = NULL,
  positive_cutoffs = NULL,
  filenames = NULL,
  silent = FALSE
)
```

### Arguments

fsom	FlowSOM object as generated by the FlowSOM function or the BuildSOM function
files	Either a vector of FCS files or paths to FCS files
level	Level(s) of interest. Default is c("clusters", "metaclusters"), but can also be only one of them. Can be abbreviated.
type	Type of features to extract. Default is "counts", can be a vector of "counts", "percentages", "MFIs" and/or "percentages_positive" or abbreviations.
MFI	Vector with channels / markers for which the MFI values must be returned when "MFIs" is in type
positive_cutoffs	Named vector with fluorescence-intensity values per channel / marker that are the upper bounds for a negative population when "percentages_positive" is in type
filenames	An optional vector with filenames that will be used as rownames in the count matrices. If NULL (default) either the paths will be used or a numerical vector.
silent	Logical. If TRUE, print progress messages. Default = FALSE.

### Value

matrix with features per population - type combination

**Examples**

```

# Build FlowSom result
fileName <- system.file("extdata", "68983.fcs", package = "FlowSOM")
ff <- flowCore::read.FCS(fileName)
ff <- flowCore::compensate(ff, flowCore::keyword(ff)[["SPILL"]])
ff <- flowCore::transform(ff,
  flowCore::transformList(colnames(flowCore::keyword(ff)[["SPILL"]]),
    flowCore::logicTransform()))
flowSOM.res <- FlowSOM(ff[1:1000, ],
  scale = TRUE,
  colsToUse = c(9, 12, 14:18),
  nClus = 10)

# Map new data
counts <- GetFeatures(fsom = flowSOM.res,
  level = "clusters",
  files = c(ff[1001:2000, ], ff[2001:3000, ]))
features <- GetFeatures(fsom = flowSOM.res,
  files = c(ff[1001:2000, ], ff[2001:3000, ]),
  type = c("counts", "percentages", "MFIs"),
  MFI = "APC-A",
  filenames = c("ff_1001-2000", "ff_2001-3000"))

# Get percentages of positive cells
positive_cutoffs <- c('CD8' = 1.5,
  'CD4' = 0.3,
  'CD19' = 1.3,
  'CD3' = -0.3)

perc_pos <- GetFeatures(fsom = flowSOM.res,
  files = c(ff[1001:2000, ], ff[2001:3000, ]),
  type = c("percentages_positive"),
  positive_cutoffs = positive_cutoffs,
  filenames = c("ff_1001-2000", "ff_2001-3000"))

```

---

GetFlowJoLabels

*Process a FlowJo workspace file*


---

**Description**

Reads a FlowJo workspace file using the flowWorkspace library and returns a list with a matrix containing gating results and a vector with a label for each cell from a set of specified gates

**Usage**

```

GetFlowJoLabels(
  files,
  wspFile,

```

```

    group = "All Samples",
    cellTypes = NULL,
    getData = FALSE,
    ...
)

```

### Arguments

files	The FCS files of interest
wspFile	The FlowJo wsp file to read
group	The FlowJo group to parse. Default "All Samples".
cellTypes	Cell types to use for final labeling the cells. Should correspond with a subset of the gate names in FlowJo.
getData	If true, flowFrames are returned as well.
...	Extra arguments to pass to CytoML::flowjo_to_gatingset

### Value

This function returns a list, which for every file contains a list in which the first element ("matrix") is a matrix containing filtering results for each specified gate and the second element ("manual") is a vector which assigns one label to each cell. If only one file is given, only one list is returned instead of a list of lists.

### See Also

[PlotPies](#)

### Examples

```

# Identify the files
fcs_file <- system.file("extdata", "68983.fcs", package = "FlowSOM")
wspFile <- system.file("extdata", "gating.wsp", package = "FlowSOM")

# Specify the cell types of interest for assigning one label per cell
cellTypes <- c("B cells",
              "gd T cells", "CD4 T cells", "CD8 T cells",
              "NK cells", "NK T cells")

# Parse the FlowJo workspace
gatingResult <- GetFlowJoLabels(fcs_file, wspFile,
                              cellTypes = cellTypes,
                              getData = TRUE)

# Check the number of cells assigned to each gate
colSums(gatingResult$matrix)

# Build a FlowSOM tree
flowSOM.res <- FlowSOM(gatingResult$flowFrame,
                      colsToUse = c(9, 12, 14:18),
                      nClus = 10,

```

```

seed = 1)

# Plot pies indicating the percentage of cell types present in the nodes
PlotPies(flowSOM.res,
         gatingResult$manual,
         backgroundValues = flowSOM.res$metaclustering)

```

---

GetMarkers

*GetMarkers*


---

### Description

Get marker names for an array of channels, given a flowFrame or a FlowSOM object. As available in "desc". If this is NA, defaults to channel name. [grep](#) is used to look for the markers. Other regex can be added.

### Usage

```
GetMarkers(object, channels, exact = TRUE)
```

### Arguments

object	The flowFrame or the FlowSOM object of interest
channels	Vector with markers or channels of interest. Also accepts the index of the channel in the object.
exact	If TRUE (default), the grep pattern will be extended to start with <code>^\Q</code> and end with <code>\E\$</code> , so only exact matches are possible.

### Value

Corresponding marker names

### See Also

[GetChannels](#)

### Examples

```

# Read the flowFrame
fileName <- system.file("extdata", "68983.fcs", package = "FlowSOM")
ff <- flowCore::read.FCS(fileName)
GetChannels(ff, c("FSC-A", "CD3", "FITC-A"))
GetMarkers(ff, c("FSC-A", "CD3", "FITC-A"))

```

---

GetMetaclusterCVs      *GetMetaclusterCVs*

---

### Description

Compute the coefficient of variation for the metaclusters

### Usage

```
GetMetaclusterCVs(fsom, colsUsed = FALSE, prettyColnames = FALSE)
```

### Arguments

fsom	Result of calling the FlowSOM function
colsUsed	Logical. Should report only the columns used to build the SOM. Default = FALSE.
prettyColnames	Logical. Should report pretty column names instead of standard column names. Default = FALSE.

### Value

Metacluster CVs

### Examples

```
fileName <- system.file("extdata", "68983.fcs", package = "FlowSOM")
ff <- flowCore::read.FCS(fileName)
ff <- flowCore::compensate(ff, flowCore::keyword(ff)[["SPILL"]])
ff <- flowCore::transform(ff,
  flowCore::transformList(colnames(flowCore::keyword(ff)[["SPILL"]]),
    flowCore::logicTransform()))
flowSOM.res <- FlowSOM(ff,
  scale = TRUE,
  colsToUse = c(9, 12, 14:18),
  nClus = 10)
cvs <- GetMetaclusterCVs(flowSOM.res)
```

---

GetMetaclusterMFIs      *GetMetaclusterMFIs*

---

### Description

Compute the median fluorescence intensities for the metaclusters

### Usage

```
GetMetaclusterMFIs(fsom, colsUsed = FALSE, prettyColnames = FALSE)
```

**Arguments**

<code>fsom</code>	Result of calling the FlowSOM function
<code>colsUsed</code>	Logical. Should report only the columns used to build the SOM. Default = FALSE.
<code>prettyColnames</code>	Logical. Should report pretty column names instead of standard column names. Default = FALSE.

**Value**

Metacluster MFIs

**Examples**

```

fileName <- system.file("extdata", "68983.fcs", package = "FlowSOM")
ff <- flowCore::read.FCS(fileName)
ff <- flowCore::compensate(ff, flowCore::keyword(ff)[["SPILL"]])
ff <- flowCore::transform(ff,
  flowCore::transformList(colnames(flowCore::keyword(ff)[["SPILL"]]),
    flowCore::logicTransform()))
flowSOM.res <- FlowSOM(ff,
  scale = TRUE,
  colsToUse = c(9, 12, 14:18),
  nClus = 10)
mfis <- GetMetaclusterMFIs(flowSOM.res)

```

---

GetMetaclusterPercentagesPositive

*Get percentage-positive values for all metaclusters*

---

**Description**

Get percentage-positive values for all metaclusters

**Usage**

```

GetMetaclusterPercentagesPositive(
  fsom,
  cutoffs,
  colsUsed = FALSE,
  prettyColnames = FALSE
)

```



**Arguments**

fsom	FlowSOM object as generated by the FlowSOM function or the BuildSOM function
cutoffs	named numeric vector. Upper bounds of negative population fluorescence-intensity values for each marker / channel.
colsUsed	logical. Should report only the columns used to build the SOM. Default = FALSE.
prettyColnames	logical. Should report pretty column names instead of standard column names. Default = FALSE.

**Value**

Matrix with percentages of cells that are positive in selected markers per each metacluster

**Examples**

```
fileName <- system.file("extdata", "68983.fcs", package = "FlowSOM")
flowSOM.res <- FlowSOM(fileName, compensate = TRUE, transform = TRUE,
                      scale = TRUE, colsToUse = c(9, 12, 14:18), nClus = 10)
perc_pos <- GetMetaclusterPercentagesPositive(flowSOM.res, cutoffs = c('CD4' = 5000))
```

---

GetMetaclusters	<i>Get metacluster label for all individual cells</i>
-----------------	---

---

**Description**

Get metacluster label for all individual cells

**Usage**

```
GetMetaclusters(fsom, meta = NULL)
```

**Arguments**

fsom	FlowSOM object as generated by the FlowSOM function or the BuildSOM function
meta	Metacluster label for each FlowSOM cluster. If this is NULL, the fsom argument should be as generated by the FlowSOM function, and fsom\$metaclustering will be used.

**Value**

vector label for every cell

**Examples**

```

fileName <- system.file("extdata", "68983.fcs", package = "FlowSOM")
flowSOM.res <- FlowSOM(fileName, compensate = TRUE, transform = TRUE,
                      scale = TRUE, colsToUse = c(9, 12, 14:18), nClus = 10)
metacluster_labels <- GetMetaclusters(flowSOM.res)
metacluster_labels <- GetMetaclusters(flowSOM.res,
                                     meta = flowSOM.res$metaclustering)

```

---

GetMFIs

*Get MFI values for all clusters*


---

**Description**

Get MFI values for all clusters

**Usage**

```
GetMFIs(fsom, colsUsed = FALSE, prettyColnames = FALSE)
```

**Arguments**

fsom	FlowSOM object as generated by the FlowSOM function or the BuildSOM function
colsUsed	logical. Should report only the columns used to build the SOM. Default = FALSE.
prettyColnames	logical. Should report pretty column names instead of standard column names. Default = FALSE.

**Value**

Matrix with median values for each marker

**Examples**

```

fileName <- system.file("extdata", "68983.fcs", package="FlowSOM")
flowSOM.res <- FlowSOM(fileName, compensate=TRUE,transform=TRUE,
                      scale=TRUE,colsToUse=c(9,12,14:18),nClus=10)
mfis <- GetClusterMFIs(flowSOM.res)

```

---

GetPercentages	<i>GetPercentages</i>
----------------	-----------------------

---

**Description**

Get percentages of number of cells in clusters or metaclusters

**Usage**

```
GetPercentages(fsom, level = "metaclusters")
```

**Arguments**

<code>fsom</code>	FlowSOM object
<code>level</code>	Character string, should be either "clusters" or "metaclusters" (default) or abbreviations.

**Value**

A named vector with the percentages

**Examples**

```
# Read from file, build self-organizing map and minimal spanning tree
fileName <- system.file("extdata", "68983.fcs", package = "FlowSOM")
ff <- flowCore::read.FCS(fileName)
ff <- flowCore::compensate(ff, flowCore::keyword(ff)[["SPILL"]])
ff <- flowCore::transform(ff, flowCore::estimateLogicle(ff,
                                                         flowCore::colnames(ff)[8:18]))
flowSOM.res <- FlowSOM(ff,
                      scale = TRUE,
                      colsToUse = c(9, 12, 14:18),
                      nClus = 10,
                      seed = 1)
GetPercentages(flowSOM.res)
GetPercentages(flowSOM.res, level = "clusters")
```

---

<code>get_channels</code>	<i>get_channels</i>
---------------------------	---------------------

---

**Description**

Get channel names for an array of markers, given a flowFrame

**Usage**

```
get_channels(ff, markers)
```

**Arguments**

ff                    The flowFrame of interest  
markers              Vector with markers or channels of interest

**Value**

Corresponding channel names

**See Also**

[get\\_markers](#)

**Examples**

```
# Read the flowFrame
fileName <- system.file("extdata", "68983.fcs", package="FlowSOM")
ff <- flowCore::read.FCS(fileName)
GetChannels(ff, c("FSC-A", "CD3", "FITC-A"))
GetMarkers(ff, c("FSC-A", "CD3", "FITC-A"))
```

---

get\_markers

*get\_markers*

---

**Description**

Get marker names, given a flowFrame. As available in "desc". If this is NA, defaults to channel name.

**Usage**

```
get_markers(ff, markers)
```

**Arguments**

ff                    The flowFrame of interest  
markers              Vector with markers or channels of interest

**Value**

Corresponding marker names

**See Also**

[get\\_channels](#)

**Examples**

```
# Read the flowFrame
fileName <- system.file("extdata", "68983.fcs", package="FlowSOM")
ff <- flowCore::read.FCS(fileName)
GetChannels(ff, c("FSC-A", "CD3", "FITC-A"))
GetMarkers(ff, c("FSC-A", "CD3", "FITC-A"))
```

---

gg_color_hue	<i>gg_color_hue</i>
--------------	---------------------

---

**Description**

Helper function to get the ggplot colors

**Usage**

```
gg_color_hue(n)
```

**Arguments**

n	Number of colors
---	------------------

**Value**

array with hexadecimal color values

---

GroupStats	<i>GroupStats</i>
------------	-------------------

---

**Description**

Calculate statistics between 2 groups based on the [GetFeatures](#) output

**Usage**

```
GroupStats(features, groups)
```

**Arguments**

features	Feature matrix as generated by <a href="#">GetFeatures</a> , e.g. a percentages matrix
groups	Named list with file or patient IDs per group (should match with the rownames of the matrix).

**Value**

Matrix with the medians per group, the p-values (the raw, Benjamini Hochberg corrected one and the  $-\log_{10}$ ) that resulted from a Wilcox test and the fold and  $\log_{10}$  fold changes between the medians of the 2 groups

**Examples**

```
# Build FlowSom result
fileName <- system.file("extdata", "68983.fcs", package = "FlowSOM")
ff <- flowCore::read.FCS(fileName)
ff <- flowCore::compensate(ff, flowCore::keyword(ff)[["SPILL"]])
ff <- flowCore::transform(ff,
  flowCore::transformList(colnames(flowCore::keyword(ff)[["SPILL"]]),
    flowCore::logicIcTransform()))
flowSOM.res <- FlowSOM(ff, scale = TRUE, colsToUse = c(9, 12, 14:18),
  nClus = 10)

# Create new data
# To illustrate the output, we here generate new FCS files (with more
# cells in metaclusters 1 and 9).
# In practice you would not generate any new file but use your different
# files from your different groups
flowCore::write.FCS(ff[sample(1:nrow(ff), 1000), ], file = "ff_tmp1.fcs")
flowCore::write.FCS(ff[sample(1:nrow(ff), 1000), ], file = "ff_tmp2.fcs")
flowCore::write.FCS(ff[sample(1:nrow(ff), 1000), ], file = "ff_tmp3.fcs")
ff_tmp <- ff[c(1:1000,
  which(flowSOM.res$map$mapping[, 1] %in%
    which(flowSOM.res$metaclustering == 9)),
  which(flowSOM.res$map$mapping[, 1] %in%
    which(flowSOM.res$metaclustering == 1))), ]
flowCore::write.FCS(ff_tmp[sample(1:nrow(ff_tmp), 1000), ],
  file = "ff_tmp4.fcs")
flowCore::write.FCS(ff_tmp[sample(1:nrow(ff_tmp), 1000), ],
  file = "ff_tmp5.fcs")

# Get the count matrix
percentages <- GetFeatures(fsom = flowSOM.res,
  files = c("ff_tmp1.fcs",
    "ff_tmp2.fcs",
    "ff_tmp3.fcs",
    "ff_tmp4.fcs",
    "ff_tmp5.fcs"),
  type = "percentages")

# Perform the statistics
groups <- list("Group 1" = c("ff_tmp1.fcs", "ff_tmp2.fcs", "ff_tmp3.fcs"),
  "Group 2" = c("ff_tmp4.fcs", "ff_tmp5.fcs"))
MC_stats <- GroupStats(percentages[["metacluster_percentages"]], groups)
C_stats <- GroupStats(percentages[["cluster_percentages"]], groups)

# Process the fold changes vector
```

```

fold_changes <- C_stats["fold changes", ]
fold_changes <- factor(ifelse(fold_changes < -3,
                             "Underrepresented compared to Group 1",
                             ifelse(fold_changes > 3,
                                     "Overrepresented compared to Group 1",
                                     "--")),
                      levels = c("--",
                                  "Underrepresented compared to Group 1",
                                  "Overrepresented compared to Group 1"))
fold_changes[is.na(fold_changes)] <- "--"

# Show in figure
## Fold change
gr_1 <- PlotStars(flowSOM.res,
                  title = "Group 1",
                  nodeSizes = C_stats["medians Group 1", ],
                  list_insteadof_ggarrange = TRUE)
gr_2 <- PlotStars(flowSOM.res, title = "Group 2",
                  nodeSizes = C_stats["medians Group 2", ],
                  backgroundValues = fold_changes,
                  backgroundColors = c("white", "red", "blue"),
                  list_insteadof_ggarrange = TRUE)
p <- ggpubr::ggarrange(plotlist = c(list(gr_1$tree), gr_2),
                       heights = c(3, 1))
ggplot2::ggsave("Groups_foldchanges.pdf", p, width = 10)

## p values
p <- PlotVariable(flowSOM.res, title = "Wilcox test group 1 vs. group 2",
                  variable = C_stats["p values", ])
ggplot2::ggsave("Groups_pvalues.pdf", p)

## volcano plot
p <- ggplot2::ggplot(data.frame("-log10 p values" = c(C_stats[4, ],
                                                    MC_stats[4, ]),
                           "log10 fold changes" = c(C_stats[7, ],
                                                    MC_stats[7, ]),
                           check.names = FALSE), ggplot2::aes(x = `log10 fold changes`,
                                                              y = `-log10 p values`)) +
  ggplot2::xlim(-3, 3) +
  ggplot2::ylim(0, 3) +
  ggplot2::geom_point()

```

---

Initialize\_KWSP

*Select k well spread points from X*


---

## Description

Select k well spread points from X

**Usage**

```
Initialize_KWSP(X, xdim, ydim)
```

**Arguments**

X	matrix in which each row represents a point
xdim	x dimension of the grid
ydim	y dimension of the grid

**Value**

array containing the selected selected rows

**Examples**

```
points <- matrix(1:1000, ncol = 10)
selection <- Initialize_KWSP(points, 3, 3)
```

---

Initialize_PCA	<i>Create a grid from first 2 PCA components</i>
----------------	--

---

**Description**

Create a grid from first 2 PCA components

**Usage**

```
Initialize_PCA(data, xdim, ydim)
```

**Arguments**

data	matrix in which each row represents a point
xdim	x dimension of the grid
ydim	y dimension of the grid

**Value**

array containing the selected selected rows

**Examples**

```
points <- matrix(1:1000, ncol = 10)
selection <- Initialize_PCA(points, 3, 3)
```



---

ManualVector	<i>Summarize the gating matrix into one vector, only including the cell types of interest</i>
--------------	---

---

**Description**

Extract the compensated and transformed data and all gate labels.

**Usage**

```
ManualVector(manualMatrix, cellTypes)
```

**Arguments**

manualMatrix	Matrix containing boolean values, indicating for every gate (column) whether the cell (row) is part of it or not.
cellTypes	Cell types to use in the summary vector. All others will be ignored and cells which do not fall in one of these gates will get the label "Unknown". Order is important!

**Value**

A factor with one label for every cell

---

MapDataToCodes	<i>Assign nearest node to each datapoint</i>
----------------	--

---

**Description**

Assign nearest node to each datapoint

**Usage**

```
MapDataToCodes(codes, newdata, distf = 2)
```

**Arguments**

codes	matrix with nodes of the SOM
newdata	datapoints to assign
distf	Distance function (1 = manhattan, 2 = euclidean, 3 = chebyshev, 4 = cosine)

**Value**

Array with nearest node id for each datapoint

---

MetaclusterCVs	<i>MetaclusterCVs</i>
----------------	-----------------------

---

**Description**

Compute the coefficient of variation for the metaclusters

**Usage**

```
MetaclusterCVs(fsom)
```

**Arguments**

fso	Result of calling the FlowSOM function
-----	--

**Value**

Metacluster CVs

**Examples**

```
fileName <- system.file("extdata", "68983.fcs", package="FlowSOM")
ff <- flowCore::read.FCS(fileName)
ff <- flowCore::compensate(ff, ff@description$SPILL)
ff <- flowCore::transform(ff,
  flowCore::transformList(colnames(ff@description$SPILL),
    flowCore::logicleTransform()))
flowSOM.res <- FlowSOM(ff, scale=TRUE, colsToUse=c(9,12,14:18), nClus=10)
cvs <- GetMetaclusterCVs(flowSOM.res)
```

---

MetaClustering	<i>MetaClustering</i>
----------------	-----------------------

---

**Description**

Cluster data with automatic number of cluster determination for several algorithms

**Usage**

```
MetaClustering(data, method, max = 20, seed = NULL, ...)
```

**Arguments**

data	Matrix containing the data to cluster
method	Clustering method to use
max	Maximum number of clusters to try out
seed	Seed to pass on to given clustering method
...	Extra parameters to pass along

**Value**

Numeric array indicating cluster for each datapoint

**See Also**

[metaClustering\\_consensus](#)

**Examples**

```
# Read from file, build self-organizing map and minimal spanning tree
fileName <- system.file("extdata", "68983.fcs", package = "FlowSOM")
flowSOM.res <- ReadInput(fileName, compensate = TRUE, transform = TRUE,
                        scale = TRUE)
flowSOM.res <- BuildSOM(flowSOM.res, colsToUse = c(9, 12, 14:18))
flowSOM.res <- BuildMST(flowSOM.res)

# Apply metaclustering
metacl <- MetaClustering(flowSOM.res$map$codes,
                        "metaClustering_consensus",
                        max = 10)

# Get metaclustering per cell
flowSOM.clustering <- metacl[flowSOM.res$map$mapping[, 1]]
```

---

metaClustering\_consensus

*MetaClustering*

---

**Description**

Cluster data using hierarchical consensus clustering with k clusters

**Usage**

```
metaClustering_consensus(data, k = 7, seed = NULL)
```

**Arguments**

data	Matrix containing the data to cluster
k	Number of clusters
seed	Seed to pass to consensusClusterPlus

**Value**

Numeric array indicating cluster for each datapoint

**See Also**[MetaClustering](#)**Examples**

```
# Read from file, build self-organizing map and minimal spanning tree
fileName <- system.file("extdata", "68983.fcs", package = "FlowSOM")
flowSOM.res <- ReadInput(fileName, compensate = TRUE, transform = TRUE,
                        scale = TRUE)
flowSOM.res <- BuildSOM(flowSOM.res, colsToUse = c(9, 12, 14:18))
flowSOM.res <- BuildMST(flowSOM.res)

# Apply consensus metaclustering
metacl <- metaClustering_consensus(flowSOM.res$map$codes, k = 10)
```

---

 MetaclusterMFIs

*MetaclusterMFIs*


---

**Description**

Compute the median fluorescence intensities for the metaclusters

**Usage**

```
MetaclusterMFIs(fsom)
```

**Arguments**

fsom                      Result of calling the FlowSOM function

**Value**

Metacluster MFIs

**Examples**

```
fileName <- system.file("extdata", "68983.fcs", package="FlowSOM")
ff <- flowCore::read.FCS(fileName)
ff <- flowCore::compensate(ff, ff@description$SPILL)
ff <- flowCore::transform(ff,
                        flowCore::transformList(colnames(ff@description$SPILL),
                                                flowCore::logicleTransform()))
flowSOM.res <- FlowSOM(ff, scale=TRUE, colsToUse=c(9,12,14:18), maxMeta=10)
mfis <- GetMetaclusterMFIs(flowSOM.res)
```

---

NClusters

*NClusters*

---

### Description

Extracts the number of clusters from a FlowSOM object

### Usage

```
NClusters(fsom)
```

### Arguments

fsom                      FlowSOM object

### Value

The number of clusters

### Examples

```
# Build FlowSom result
fileName <- system.file("extdata", "68983.fcs", package = "FlowSOM")
ff <- flowCore::read.FCS(fileName)
flowSOM.res <- FlowSOM(ff,
                       compensate = TRUE, transform = TRUE, scale = TRUE,
                       colsToUse = c(9, 12, 14:18),
                       maxMeta = 10)
NClusters(flowSOM.res)
```

---

NewData

*NewData*

---

### Description

Map new data to a FlowSOM grid

### Usage

```
NewData(
  fsom,
  input,
  madAllowed = 4,
  compensate = NULL,
  spillover = NULL,
  transform = NULL,
```

```

    toTransform = NULL,
    transformFunction = NULL,
    transformList = NULL,
    scale = NULL,
    scaled.center = NULL,
    scaled.scale = NULL,
    silent = FALSE
)

```

### Arguments

<code>fson</code>	FlowSOM object
<code>input</code>	A <code>flowFrame</code> , a <code>flowSet</code> or an array of paths to files or directories
<code>madAllowed</code>	A warning is generated if the distance of the new data points to their closest cluster center is too big. This is computed based on the typical distance of the points from the original dataset assigned to that cluster, the threshold being set to $\text{median} + \text{madAllowed} * \text{MAD}$ . Default is 4.
<code>compensate</code>	logical, does the data need to be compensated. If <code>NULL</code> , the same value as in the original FlowSOM call will be used.
<code>spillover</code>	spillover matrix to compensate with. If <code>NULL</code> , the same value as in the original FlowSOM call will be used.
<code>transform</code>	logical, does the data need to be transformed. If <code>NULL</code> , the same value as in the original FlowSOM call will be used.
<code>toTransform</code>	column names or indices that need to be transformed. If <code>NULL</code> , the same value as in the original FlowSOM call will be used.
<code>transformFunction</code>	If <code>NULL</code> , the same value as in the original FlowSOM call will be used.
<code>transformList</code>	If <code>NULL</code> , the same value as in the original FlowSOM call will be used.
<code>scale</code>	Logical, does the data needs to be rescaled. If <code>NULL</code> , the same value as in the original FlowSOM call will be used.
<code>scaled.center</code>	See <a href="#">scale</a> . If <code>NULL</code> , the same value as in the original FlowSOM call will be used.
<code>scaled.scale</code>	See <a href="#">scale</a> . If <code>NULL</code> , the same value as in the original FlowSOM call will be used.
<code>silent</code>	Logical. If <code>TRUE</code> , print progress messages. Default = <code>FALSE</code> .

### Details

New data is mapped to an existing FlowSOM object. The input is similar to the [ReadInput](#) function. A new FlowSOM object is created, with the same grid, but a new mapping, node sizes and mean values. The same preprocessing steps (compensation, transformation and scaling) will happen to this file as was specified in the original FlowSOM call. The scaling parameters from the original grid will be used.

### Value

A new FlowSOM object

**See Also**

[FlowSOMSubset](#) if you want to get a subset of the current data instead of a new dataset

**Examples**

```
# Build FlowSom result
fileName <- system.file("extdata", "68983.fcs", package = "FlowSOM")
ff <- flowCore::read.FCS(fileName)
ff <- flowCore::compensate(ff, flowCore::keyword(ff)[["SPILL"]])
ff <- flowCore::transform(ff,
  flowCore::transformList(colnames(flowCore::keyword(ff)[["SPILL"]]),
    flowCore::logicIcTransform()))
flowSOM.res <- FlowSOM(ff[1:1000, ],
  scale = TRUE,
  colsToUse = c(9, 12, 14:18),
  nClus = 10)

# Map new data
fSOM2 <- NewData(flowSOM.res, ff[1001:2000, ])
```

---

NMetaclusters

*NMetaclusters*


---

**Description**

Extracts the number of metaclusters from a FlowSOM object

**Usage**

```
NMetaclusters(fsom)
```

**Arguments**

```
fsom          FlowSOM object
```

**Value**

The number of metaclusters

**Examples**

```
# Build FlowSom result
fileName <- system.file("extdata", "68983.fcs", package = "FlowSOM")
ff <- flowCore::read.FCS(fileName)
flowSOM.res <- FlowSOM(ff,
  compensate = TRUE, transform = TRUE, scale = TRUE,
  colsToUse = c(9, 12, 14:18),
  maxMeta = 10)
NMetaclusters(flowSOM.res)
```

---

ParseArcs

*ParseArcs*

---

### Description

Parses stars

### Usage

```
ParseArcs(x, y, arcValues, arcHeights)
```

### Arguments

x	x coordinate of node
y	y coordinate of node
arcValues	A named vector with the frequency of how the node should be divided
arcHeights	The heights of the arcs

### Details

Function that parses the FlowSOM object into a dataframe for the star values for ggplot

### Value

A dataframe ready to use with ggplot, consisting of the coordinates of centers, the radius and angles of the star values

### See Also

[PlotFlowSOM](#), [ParseEdges](#), [ParseNodeSize](#), [ParseQuery](#), [ParseSD](#)

---

ParseEdges

*ParseEdges*

---

### Description

Parses edges

### Usage

```
ParseEdges(fsom)
```

### Arguments

fsom	FlowSOM object, as generated by <a href="#">FlowSOM</a>
------	---



**Details**

Function that parses the graph edges of the FlowSOM object into a dataframe

**Value**

A dataframe consisting of start and end coordinates of edges

**See Also**

[PlotFlowSOM](#), [ParseNodeSize](#), [ParseArcs](#), [ParseQuery](#), [ParseSD](#), [AddMST](#)

---

ParseLayout	<i>ParseLayout</i>
-------------	--------------------

---

**Description**

ParseLayout

**Usage**

```
ParseLayout(fsom, layout)
```

**Arguments**

fsom	FlowSOM object
layout	"MST", "grid" or a matrix/dataframe with 2 columns and 1 row per cluster

**Value**

dataframe with 2 columns and 1 row per cluster

---

ParseNodeSize	<i>ParseNodeSize</i>
---------------	----------------------

---

**Description**

Parses node size

**Usage**

```
ParseNodeSize(nodeSizes, maxNodeSize, refNodeSize)
```

**Arguments**

nodeSizes	A vector with node sizes
maxNodeSize	Determines the maximum node size.
refNodeSize	Reference for node size against which the nodeSizes will be scaled. Default = max(nodeSizes)

**Details**

Function that parses the mapping of the FlowSOM object into node sizes relative to the abundances of cells per cluster

Scales node size relative to the abundances of cells per cluster

**Value**

A vector is returned consisting of node sizes

**See Also**

[PlotFlowSOM](#), [ParseEdges](#), [AutoMaxNodeSize](#), [ParseArcs](#), [ParseQuery](#), [ParseSD](#)

---

ParseQuery

*ParseQuery*

---

**Description**

Parses query

**Usage**

ParseQuery(fsom, query)

**Arguments**

fsom	FlowSOM object, as generated by <a href="#">FlowSOM</a>
query	Array containing "high" or "low" for the specified column names of the FlowSOM data

**Details**

Identify nodes in the tree which resemble a certain profile of "high" or "low" marker expressions.

**Value**

A list, containing the ids of the selected nodes, the individual scores for all nodes and the scores for each marker for each node

**See Also**

[PlotFlowSOM](#), [ParseEdges](#), [ParseNodeSize](#), [ParseArcs](#), [QueryStarPlot](#), [ParseSD](#)

---

ParseSD

*ParseSD Parses SD in FlowSOM object*

---

**Description**

Calculates the standard deviation of a FlowSOM object

**Usage**

```
ParseSD(fsom, marker = NULL)
```

**Arguments**

fsom	FlowSOM object, as generated by <a href="#">FlowSOM</a>
marker	If a marker is given, the standard deviation for this marker is shown. Otherwise, the maximum ratio is used.

**Value**

A vector containing the SDs

**See Also**

[PlotFlowSOM](#), [ParseEdges](#), [ParseNodeSize](#), [ParseArcs](#), [ParseQuery](#), [PlotSD](#)

---

Plot2DScatters

*Plot2DScatters*

---

**Description**

Function to draw 2D scatter plots of FlowSOM (meta)clusters

**Usage**

```
Plot2DScatters(  
  fsom,  
  channelpairs,  
  clusters = NULL,  
  metaclusters = NULL,  
  maxBgPoints = 3000,  
  sizeBgPoints = 0.5,  
  maxPoints = 1000,
```

```

    sizePoints = 0.5,
    xLim = NULL,
    yLim = NULL,
    xyLabels = c("marker"),
    density = TRUE,
    centers = TRUE,
    colors = NULL,
    plotFile = "2DScatterPlots.png"
  )

```

### Arguments

<code>fsom</code>	FlowSOM object, as created by <a href="#">FlowSOM</a>
<code>channelpairs</code>	List in which each element is a pair of channel or marker names
<code>clusters</code>	Vector or list (to combine multiple clusters in one plot) with indices of clusters of interest
<code>metaclusters</code>	Vector or list (to combine multiple metaclusters in one plot) with indices of metaclusters of interest
<code>maxBgPoints</code>	Maximum number of background cells to plot
<code>sizeBgPoints</code>	Size of the background cells
<code>maxPoints</code>	Maximum number of (meta)cluster cells to plot
<code>sizePoints</code>	Size of the (meta)cluster cells
<code>xLim</code>	Optional vector of a lower and upper limit of the x-axis
<code>yLim</code>	Optional vector of a lower and upper limit of the y-axis
<code>xyLabels</code>	Determines the label of the x- and y-axis. Can be "marker" and/or "channel" or abbreviations. Default = "marker".
<code>density</code>	Default is TRUE to color the (meta)cluster points according to density. Set to FALSE to use a plain color
<code>centers</code>	Default is TRUE to show the cluster centers
<code>colors</code>	Colors for all the cells in the selected nodes (ordered list). First the clusters are colored, then the metaclusters. If NULL, the default ggplot colors, indexed by metacluster number, are used.
<code>plotFile</code>	If a filepath for a png is given (default = 2DScatterPlots.png), the plots will be plotted in the corresponding png file. If NULL, a list of ggplot objects will be returned

### Details

Plot multiple 2D scatter plots in a png file. A subset of `fsom$data` is plotted in gray, and those of the selected clusters and metaclusters are plotted in color.

### Value

If `plot` is TRUE, nothing is returned and a plot is drawn in which background cells are plotted in gray and the cells of the selected nodes in color. If `plot` is FALSE, a ggplot objects list is returned.

**Examples**

```

# Identify the files
fcs <- flowCore::read.FCS(system.file("extdata", "68983.fcs",
                                     package = "FlowSOM"))

# Build a FlowSOM object
flowSOM.res <- FlowSOM(fcs,
                      scale = TRUE,
                      compensate = TRUE,
                      transform = TRUE,
                      toTransform = 8:18,
                      colsToUse = c(9, 12, 14:18),
                      nClus = 10,
                      seed = 1)

# Make the 2D scatter plots of the clusters and metaclusters of interest
Plot2DScatters(fsom = flowSOM.res,
               channelpairs = list(c("PE-Cy7-A", "PE-Cy5-A"),
                                   c("PE-Texas Red-A", "Pacific Blue-A")),
               clusters = c(1, 48, 49, 82, 95),
               metaclusters = list(c(1, 4), 9),
               density = FALSE)

Plot2DScatters(fsom = flowSOM.res,
               channelpairs = list(c("PE-Texas Red-A", "Pacific Blue-A")),
               metaclusters = list(c(1, 4)),
               density = FALSE,
               colors = list(c("red", "green")))

```

---

PlotCenters

*PlotCenters*


---

**Description**

Plot cluster centers on a 2D plot

**Usage**

```
PlotCenters(fsom, marker1, marker2, MST = TRUE)
```

**Arguments**

fsom	FlowSOM object, as generated by <a href="#">BuildMST</a>
marker1	Marker to show on the x-axis
marker2	Marker to show on the y-axis
MST	Type of visualization, if 1 plot tree, else plot grid

**Details**

Plot FlowSOM nodes on a 2D scatter plot of the data

**Value**

Nothing is returned. A 2D scatter plot is drawn on which the nodes of the grid are indicated

**See Also**

[PlotStars](#), [PlotPies](#), [PlotMarker](#), [BuildMST](#)

**Examples**

```
# Read from file, build self-organizing map and minimal spanning tree
fileName <- system.file("extdata", "68983.fcs", package="FlowSOM")
flowSOM.res <- ReadInput(fileName, compensate=TRUE, transform=TRUE,
                        scale=TRUE)
flowSOM.res <- BuildSOM(flowSOM.res, colsToUse=c(9,12,14:18))
flowSOM.res <- BuildMST(flowSOM.res)

# Plot centers
plot <- Plot2DScatters(flowSOM.res,
                      channelpairs = list(c("FSC-A", "SSC-A")),
                      clusters = list(seq_len(NClusters(flowSOM.res))),
                      maxPoints = 0,
                      plotFile = NULL)
```

---

PlotClusters2D

*PlotClusters2D*

---

**Description**

Plot nodes on scatter plot

**Usage**

```
PlotClusters2D(
  fsom,
  marker1,
  marker2,
  nodes,
  col = "#FF0000",
  maxBgPoints = 10000,
  pchBackground = ".",
  pchCluster = ".",
  main = "",
  xlab = fsom$prettyColnames[marker1],
```

```

ylab = fsom$prettyColnames[marker2],
xlim = c(min(fsom$data[, marker1]), max(fsom$data[, marker1])),
ylim = c(min(fsom$data[, marker2]), max(fsom$data[, marker2])),
...
)

```

### Arguments

fsom	FlowSOM object, as generated by <a href="#">BuildMST</a>
marker1	Marker to plot on the x-axis
marker2	Marker to plot on the y-axis
nodes	Nodes of which the cells should be plotted in red
col	Colors for all the cells in the selected nodes (ordered array)
maxBgPoints	Maximum number of background points to plot
pchBackground	Character to use for background cells
pchCluster	Character to use for cells in cluster
main	Title of the plot
xlab	Label for the x axis
ylab	Label for the y axis
xlim	Limits for the x axis
ylim	Limits for the y axis
...	Other parameters to pass on to plot

### Details

Plot a 2D scatter plot. All cells of `fsom$data` are plotted in black, and those of the selected nodes are plotted in red. The nodes in the grid are indexed starting from the left bottom, first going right, then up. E.g. In a 10x10 grid, the node at top left will have index 91.

### Value

Nothing is returned. A plot is drawn in which all cells are plotted in black and the cells of the selected nodes in red.

### See Also

[PlotNumbers](#), [PlotCenters](#), [BuildMST](#)

### Examples

```

## Deprecated - use Plot2DScatters instead ##

# Read from file, build self-organizing map and minimal spanning tree
fileName <- system.file("extdata", "68983.fcs", package = "FlowSOM")
flowSOM.res <- ReadInput(fileName, compensate = TRUE, transform = TRUE,
                          scale = TRUE)

```

```

flowSOM.res <- BuildSOM(flowSOM.res, colsToUse = c(9, 12, 14:18))
flowSOM.res <- BuildMST(flowSOM.res)

# Plot cells
## Not run:
Plot2DScatters(flowSOM.res, c(1, 2), clusters = 91)

## End(Not run)

```

---

PlotDimRed

*PlotDimRed*


---

## Description

Plot a dimensionality reduction

## Usage

```

PlotDimRed(
  fsom,
  colsToUse = fsom$map$colsUsed,
  colorBy = "metaclusters",
  colors = NULL,
  lim = NULL,
  cTotal = NULL,
  dimred = Rtsne::Rtsne,
  extractLayout = function(dimred) {
    dimred$Y
  },
  label = TRUE,
  returnLayout = FALSE,
  seed = NULL,
  title = NULL,
  ...
)

```

## Arguments

<code>fsom</code>	FlowSOM object, as generated by <a href="#">BuildMST</a>
<code>colsToUse</code>	The columns used for the dimensionality reduction. Default = <code>fsom\$map\$colsUsed</code> .
<code>colorBy</code>	Defines how the dimensionality reduction will be colored. Can be "metaclusters" (default), "clusters" (or abbreviations) or a marker/channel/index.
<code>colors</code>	A vector of custom colors. Default returns ggplot colors for categorical variables and the FlowSOM colors for continuous variables. When using a categorical variable, the vector must be as long as the levels of the categorical variable.
<code>lim</code>	Limits for the colorscale



<code>cTotal</code>	The total amount of cells to be used in the dimensionality reduction. Default is all the cells.
<code>dimred</code>	A dimensionality reduction function. Default = <code>Rtsne::Rtsne</code> . Alternatively, a <code>data.frame</code> or matrix with either equal number of rows to the <code>fsum</code> or an <code>OriginalID</code> column. Recommended to put <code>cTotal</code> to <code>NULL</code> when providing a matrix (or ensuring that the <code>dimred</code> corresponds to subsampling the <code>flowSOM</code> data for <code>cTotal</code> cells with the same seed).
<code>extractLayout</code>	A function to extract the coordinates from the results of the <code>dimred</code> default = <code>function(dimred)dimred\$Y</code> .
<code>label</code>	If <code>label = TRUE</code> (default), labels are added to plot.
<code>returnLayout</code>	If <code>TRUE</code> , this function returns a <code>dataframe</code> with the layout of <code>dimred</code> and the original IDs and the plot. Default = <code>FALSE</code> .
<code>seed</code>	A seed for reproducibility.
<code>title</code>	A title for the plot.
<code>...</code>	Additional arguments to pass to <code>dimred</code> .

### Details

Plot a dimensionality reduction of `fsum$data`

### Value

A dimensionality reduction plot made in `ggplot2`

### Examples

```
file <- system.file("extdata", "68983.fcs", package="FlowSOM")
flowSOM.res <- FlowSOM(file, compensate = TRUE, transform = TRUE,
  scale = TRUE,
  colsToUse = c(9, 12, 14:18), nClus = 10, silent = FALSE,
  xdim = 7, ydim = 7)
PlotDimRed(flowSOM.res, cTotal = 5000, seed = 1, title = "t-SNE")
PlotDimRed(flowSOM.res, cTotal = 5000, colorBy = "CD3", seed = 1,
  title = "t-SNE")
```

---

PlotFileScatters

*PlotFileScatters*

---

### Description

Make a scatter plot per channel for all provided files

**Usage**

```

PlotFileScatters(
  input,
  fileID = "File",
  channels = NULL,
  yLim = NULL,
  yLabel = "marker",
  quantiles = NULL,
  names = NULL,
  groups = NULL,
  color = NULL,
  legend = FALSE,
  maxPoints = 50000,
  ncol = NULL,
  nrow = NULL,
  width = NULL,
  height = NULL,
  silent = FALSE,
  plotFile = "FileScatters.png"
)

```

**Arguments**

input	Either a flowSet, a flowFrame with a file ID column (e.g. output from the <a href="#">AggregateFlowFrames</a> includes a "File" column) or a vector of paths pointing to FCS files
fileID	Name of the file ID column when the input is a flowFrame, default to "File" (File ID column in the <a href="#">AggregateFlowFrames</a> flowFrame output).
channels	Vector of channels or markers that need to be plotted, if NULL (default), all channels from the input will be plotted
yLim	Optional vector of a lower and upper limit of the y-axis
yLabel	Determines the label of the y-axis. Can be "marker" and/or "channel" or abbreviations. Default = "marker".
quantiles	If provided (default NULL), a numeric vector with values between 0 and 1. These quantiles are indicated on the plot
names	Optional parameter to provide filenames. If NULL (default), the filenames will be numbers. Duplicated filenames will be made unique.
groups	Optional parameter to specify groups of files, should have the same length as the input. If NULL (default), all files will be plotted in the same color
color	Optional parameter to provide colors. Should have the same lengths as the number of groups (or 1 if groups is NULL)
legend	Logical parameter to specify whether the group levels should be displayed. Default is FALSE
maxPoints	Total number of data points that will be plotted per channel, default is 50000
ncol	Number of columns in the final plot, optional

nrow	Number of rows in the final plot, optional
width	Width of png file. By default NULL the width parameter is estimated based on the input.
height	Height of png file. By default NULL the width parameter is estimated based on the input.
silent	If FALSE, prints an update every time it starts processing a new file. Default = FALSE.
plotFile	Path to png file, default is "FileScatters.png". If NULL, the output will be a list of ggplots

### Value

List of ggplot objects if plot is FALSE, otherwise filePlot with plot is created.

### Examples

```
# Preprocessing
fileName <- system.file("extdata", "68983.fcs", package = "FlowSOM")
ff <- flowCore::read.FCS(fileName)
ff <- flowCore::compensate(ff, flowCore::keyword(ff)[["SPILL"]])
ff <- flowCore::transform(ff,
  flowCore::transformList(colnames(flowCore::keyword(ff)[["SPILL"]]),
    flowCore::logicTransform()))

flowCore::write.FCS(ff[1:1000, ], file = "ff_tmp1.fcs")
flowCore::write.FCS(ff[1001:2000, ], file = "ff_tmp2.fcs")
flowCore::write.FCS(ff[2001:3000, ], file = "ff_tmp3.fcs")

# Make plot
PlotFileScatters(input = c("ff_tmp1.fcs", "ff_tmp2.fcs", "ff_tmp3.fcs"),
  channels = c("Pacific Blue-A",
    "Alexa Fluor 700-A",
    "PE-Cy7-A"),
  maxPoints = 1000)
```

---

 PlotFlowSOM

*PlotFlowSOM*


---

### Description

Base layer to plot a FlowSOM result

**Usage**

```
PlotFlowSOM(
  fsom,
  view = "MST",
  nodeSizes = fsom$map$pctgs,
  maxNodeSize = 1,
  refNodeSize = max(nodeSizes),
  equalNodeSize = FALSE,
  backgroundValues = NULL,
  backgroundColors = NULL,
  backgroundLim = NULL,
  title = NULL
)
```

**Arguments**

<code>fsom</code>	FlowSOM object, as created by <a href="#">FlowSOM</a>
<code>view</code>	Preferred view, options: "MST", "grid" or "matrix" with a matrix/dataframe consisting of coordinates. Default = "MST"
<code>nodeSizes</code>	A vector containing node sizes. These will automatically be scaled between 0 and <code>maxNodeSize</code> and transformed with a sqrt. Default = <code>fsom\$MST\$sizes</code>
<code>maxNodeSize</code>	Determines the maximum node size. Default is 1.
<code>refNodeSize</code>	Reference for node size against which the <code>nodeSizes</code> will be scaled. Default = <code>max(nodeSizes)</code>
<code>equalNodeSize</code>	If TRUE, the nodes will be equal to <code>maxNodeSize</code> . If FALSE (default), the nodes will be scaled to the number of cells in each cluster
<code>backgroundValues</code>	Values to be used for background coloring, either numerical values or something that can be made into a factor (e.g. a clustering)
<code>backgroundColors</code>	Color palette to be used for the background coloring. Can be either a function or an array specifying colors.
<code>backgroundLim</code>	Only used when <code>backgroundValues</code> are numerical. Defaults to min and max of the <code>backgroundValues</code> .
<code>title</code>	Title of the plot

**Details**

Base layer of the FlowSOM plot, where you can choose layout (MST, grid or coordinates of your own choosing), background colors and node size. Can then be extended by e.g. [AddStars](#), [AddLabels](#), [AddPies](#), ...

**Value**

A ggplot object with the base layer of a FlowSOM plot

**See Also**

[PlotStars](#), [PlotVariable](#), [PlotMarker](#), [PlotLabels](#), [PlotNumbers](#), [PlotPies](#), [QueryStarPlot](#), [PlotSD](#)

**Examples**

```
# Locate file on file system
fcs_file <- system.file("extdata", "68983.fcs", package = "FlowSOM")

# Build FlowSOM model
flowSOM.res <- FlowSOM(fcs_file,
  scale = TRUE,
  compensate = TRUE,
  transform = TRUE,
  toTransform = 8:18,
  colsToUse = c(9, 12, 14:18),
  nClus = 10,
  seed = 1)

# Plot with background coloring
PlotFlowSOM(flowSOM.res,
  backgroundValues = flowSOM.res$metaclustering) %>%
  AddLabels(seq(100))
```

---

 PlotGroups

*PlotGroups*


---

**Description**

Plot differences between groups

**Usage**

```
PlotGroups(fsom, groups, threshold = NULL, pThreshold = 0.05, ...)
```

**Arguments**

<code>fsom</code>	FlowSOM object, as generated by <a href="#">BuildMST</a>
<code>groups</code>	Groups result as generated by <a href="#">CountGroups</a>
<code>threshold</code>	Relative difference in groups before the node is colored
<code>pThreshold</code>	Threshold on p-value from wilcox-test before the node is colored. If this is not NULL, threshold will be ignored.
<code>...</code>	Additional arguments to pass to <a href="#">PlotFlowSOM</a>

**Details**

Plot FlowSOM trees, where each node is represented by a star chart indicating mean marker values, the size of the node is relative to the mean percentage of cells present in each

**Value**

A vector containing the labels assigned to the nodes for all groups except the first

**See Also**

[PlotStars](#), [PlotVariable](#), [PlotFlowSOM](#), [PlotLabels](#), [PlotNumbers](#), [PlotMarker](#), [PlotPies](#), [QueryStarPlot](#), [PlotSD](#)

**Examples**

```
#Run FlowSOM
fileName <- system.file("extdata", "68983.fcs", package="FlowSOM")
fsom <- FlowSOM(fileName, compensate = TRUE, transform = TRUE,
               scale = TRUE, colsToUse = c(9,12,14:18), nClus = 10)

ff <- flowCore::read.FCS(fileName)
# Make an additional file without cluster 7 and double amount of cluster 5
selection <- c(which(GetClusters(fsom) %in% which(fsom$metaclustering != 7)),
              which(GetClusters(fsom) %in% which(fsom$metaclustering == 5)))
ff_tmp <- ff[selection,]
flowCore::write.FCS(ff_tmp, file="ff_tmp.fcs")

# Compare only the file with the double amount of cluster 10
features <- GetFeatures(fsom,
                      c(fileName, "ff_tmp.fcs"),
                      level = "clusters",
                      type = "percentages")
stats <- GroupStats(features$cluster_percentages,
                   groups = list("AllCells" = c(fileName),
                                "Without_ydTcells" = c("ff_tmp.fcs")))

fold_changes <- stats["fold changes", ]
fold_changes_label <- factor(iffelse(fold_changes < -1.5,
                                   "Underrepresented compared to Group 1",
                                   iffelse(fold_changes > 1.5,
                                         "Overrepresented compared to Group 1",
                                         "--")),
                           levels = c("--",
                                       "Underrepresented compared to Group 1",
                                       "Overrepresented compared to Group 1"))
fold_changes_label[is.na(fold_changes_label)] <- "--"
gr_1 <- PlotStars(fsom,
                 title = "All Cells",
                 nodeSizes = stats["medians AllCells", ],
                 list_insteadof_ggarrange = TRUE)
gr_2 <- PlotStars(fsom, title = "Group 2",
                 nodeSizes = stats["medians Without_ydTcells", ],
```

```

        backgroundValues = fold_changes_label,
        backgroundColors = c("white", "red", "blue"),
        list_insteadof_ggarrange = TRUE)
p <- ggpubr::ggarrange(plotlist = c(list(gr_1$tree), gr_2),
                      heights = c(3, 1))
p

```

---

PlotLabels

*PlotLabels*


---

### Description

Plot labels for each cluster

### Usage

```

PlotLabels(
  fsom,
  labels,
  maxNodeSize = 0,
  textSize = 3.88,
  textColor = "black",
  ...
)

```

### Arguments

<code>fsom</code>	FlowSOM object, as generated by <a href="#">FlowSOM</a>
<code>labels</code>	A vector of labels for every node.
<code>maxNodeSize</code>	Determines the maximum node size. Default is 0.
<code>textSize</code>	Size for <code>geom_text</code> . Default (=3.88) is from <code>geom_text</code> .
<code>textColor</code>	Color for <code>geom_text</code> . Default = black.
<code>...</code>	Additional arguments to pass to <a href="#">PlotFlowSOM</a>

### Details

Plot FlowSOM grid or tree, with in each node a label. Especially useful to show metacluster numbers

### Value

Nothing is returned. A plot is drawn in which each node is represented by a label.

### See Also

[PlotStars](#), [PlotVariable](#), [PlotFlowSOM](#), [PlotMarker](#), [PlotNumbers](#), [PlotPies](#), [QueryStarPlot](#), [PlotSD](#)

**Examples**

```
# Read from file, build self-organizing map and minimal spanning tree
fileName <- system.file("extdata", "68983.fcs", package="FlowSOM")
ff <- flowCore::read.FCS(fileName)
ff <- flowCore::compensate(ff, flowCore::keyword(ff)[["SPILL"]])
ff <- flowCore::transform(ff,
  flowCore::transformList(colnames(flowCore::keyword(ff)[["SPILL"]]),
    flowCore::logicIcTransform()))

flowSOM.res <- FlowSOM(ff,
  scale = TRUE,
  colsToUse = c(9, 12, 14:18),
  nClus = 10,
  seed = 1)

# Plot the node IDs
PlotLabels( flowSOM.res,
  flowSOM.res$metaclustering)
```

---

PlotManualBars

*PlotManualBars*


---

**Description**

Function to plot the manual labels per FlowSOM (meta)cluster in a barplot

**Usage**

```
PlotManualBars(
  fsom,
  fcs = NULL,
  manualVector,
  manualOrder = NULL,
  colors = NULL,
  list_insteadof_plots = FALSE
)
```

**Arguments**

fsom	FlowSOM object, as generated by <a href="#">FlowSOM</a> or by <a href="#">NewData</a> . The clusters and metaclusters will be plotted in the order of the factor levels.
fcs	FCS file that should be mapped on the FlowSOM object. Default is NULL.
manualVector	Vector with cell labels, e.g. obtained by manual gating
manualOrder	Optional vector with unique cell labels to fix in which order the cell labels should be shown
colors	Optional color vector, should have the same length as the number of unique cell labels



```
list_insteadof_plots
      If FALSE (default), it returns multiple plots. If TRUE, it returns a list of ggplot
      objects
```

### Value

Either a plot or a ggplot objects list is returned.

### Examples

```
# Identify the files
fcs_file <- system.file("extdata", "68983.fcs", package = "FlowSOM")
gating_file <- system.file("extdata", "gatingResult.csv", package = "FlowSOM")

# Specify the cell types of interest for assigning one label per cell
cellTypes <- c("B cells",
              "gd T cells", "CD4 T cells", "CD8 T cells",
              "NK cells", "NK T cells")

# Load manual labels (e.g. GetFlowJoLabels can be used to extract labels from
# an fcs file)

gatingResult <- as.factor(read.csv(gating_file, header = FALSE)[, 1])

# Build a FlowSOM object
flowSOM.res <- FlowSOM(fcs_file,
                      scale = TRUE,
                      compensate = TRUE,
                      transform = TRUE,
                      toTransform = 8:18,
                      colsToUse = c(9, 12, 14:18),
                      nClus = 10,
                      seed = 1)

# Make the barplot of the manual labels
pdf("PlotManualBars.pdf")
PlotManualBars(fsom = flowSOM.res,
              fcs = fcs_file,
              manualVector = gatingResult,
              manualOrder = c(cellTypes, "Unlabeled"),
              colors = c("#F8766D", "#B79F00", "#00BA38", "#00BFC4",
                        "#619CFF", "#F564E3", "#D3D3D3"))

dev.off()
```

### Description

Plot comparison with other clustering

**Usage**

```
PlotMarker(
  fsom,
  marker,
  refMarkers = fsom$map$colsUsed,
  title = GetMarkers(fsom, marker),
  colorPalette = FlowSOM_colors,
  lim = NULL,
  ...
)
```

**Arguments**

fsom	FlowSOM object
marker	A vector of markers/channels to plot.
refMarkers	Is used to determine relative scale of the marker that will be plotted. Default are all markers used in the clustering.
title	A vector with custom titles for the plot. Default is the marker name.
colorPalette	Color palette to use. Can be a function or a vector.
lim	Limits for the scale
...	Additional arguments to pass to <a href="#">PlotFlowSOM</a> , e.g. view, backgroundValues, equalNodeSize ...

**Details**

Plot FlowSOM grid or tree, colored by node values for a specific marker

**Value**

A ggplot figure is returned in which every cluster is colored according to the MFI value for the specified marker

**See Also**

[PlotStars](#), [PlotVariable](#)

**Examples**

```
# Build FlowSOM model
fileName <- system.file("extdata", "68983.fcs", package = "FlowSOM")
flowSOM.res <- FlowSOM(fileName,
  compensate = TRUE, transform = TRUE, scale = FALSE,
  colsToUse = c(9, 12, 14:18),
  nClus = 10,
  seed = 1)

# Plot one marker
PlotMarker(flowSOM.res,
  "CD19")
```

```

PlotMarker(flowSOM.res,
           "CD19",
           colorPalette = c("gray", "red"))

# Plot all markers
PlotMarker(flowSOM.res,
           c(9, 12, 14:18))

# Use specific limits if the ones from the columns used for clustering
# are not relevant for your marker of choice
PlotMarker(flowSOM.res,
           "FSC-A",
           lim = c(55000, 130000))

# Example with additional FlowSOM plotting options
PlotMarker(flowSOM.res,
           "CD19",
           view = "grid",
           equalNodeSize = TRUE,
           backgroundValues = 1:100 == 27,
           backgroundColors = c("white", "red"))

```

---

PlotNode

*PlotNode Plot star chart*


---

## Description

Plot a star chart indicating median marker values of a single node

## Usage

```

PlotNode(
  fsom,
  id,
  markers = fsom$map$colsUsed,
  colorPalette = grDevices::colorRampPalette(c("#00007F", "blue", "#007FFF", "cyan",
        "#7FFF7F", "yellow", "#FF7F00", "red", "#7F0000")),
  main = paste0("Cluster ", id)
)

```

## Arguments

fsom	FlowSOM object, as generated by <a href="#">BuildMST</a> or the first element of the list returned by <a href="#">FlowSOM</a>
id	Id of the node to plot (check <a href="#">PlotNumbers</a> to get the ids)
markers	Array of markers to use. Default: the markers used to build the tree

colorPalette    Color palette to be used for the markers  
 main            Title of the plot

**Value**

Nothing is returned. A plot is drawn in which the node is represented by a star chart indicating the median fluorescence intensities.

**See Also**

[PlotStars](#), [PlotNumbers](#), [FlowSOM](#)

**Examples**

```
# Read from file, build self-organizing map and minimal spanning tree
fileName <- system.file("extdata", "68983.fcs", package="FlowSOM")
flowSOM.res <- FlowSOM(fileName, compensate=TRUE, transform=TRUE,
  scale=TRUE, colsToUse=c(9,12,14:18), nClus=10)

# Deprecated, it is currently not possible anymore to plot an individual
# node alone. If necessary, zooming in on a node can be approximated by
# exaggerating the size of the node.
PlotStars(flowSOM.res, nodeSizes = c(100, rep(0,99)), maxNodeSize = 10)
```

---

PlotNumbers

*PlotNumbers*

---

**Description**

Plot cluster ids for each cluster

**Usage**

```
PlotNumbers(fsom, level = "clusters", maxNodeSize = 0, ...)
```

**Arguments**

fsom            FlowSOM object  
 level          Character string, should be either "clusters" or "metaclusters". Can be abbreviated.  
 maxNodeSize   Determines the maximum node size. Default is 0. See [PlotFlowSOM](#) for more options.  
 ...            Additional arguments to pass to [PlotLabels](#) and to [PlotFlowSOM](#)

**Details**

Plot FlowSOM grid or tree, with in each node the cluster id.

**Value**

Nothing is returned. A plot is drawn in which each node is labeled by its cluster id.

**See Also**

[PlotStars](#), [PlotVariable](#), [PlotFlowSOM](#), [PlotLabels](#), [PlotMarker](#), [PlotPies](#), [QueryStarPlot](#), [PlotSD](#)

**Examples**

```
# Read from file, build self-organizing map and minimal spanning tree
fileName <- system.file("extdata", "68983.fcs", package = "FlowSOM")
ff <- flowCore::read.FCS(fileName)
ff <- flowCore::compensate(ff, flowCore::keyword(ff)[["SPILL"]])
ff <- flowCore::transform(ff, flowCore::estimateLogicle(ff,
                                                    flowCore::colnames(ff)[8:18]))

flowSOM.res <- FlowSOM(ff,
                      scale = TRUE,
                      colsToUse = c(9, 12, 14:18),
                      nClus = 10,
                      seed = 1)

# Plot the node IDs
PlotNumbers(flowSOM.res)
PlotNumbers(flowSOM.res, "metaclusters")

PlotNumbers(flowSOM.res,
            view = "grid")

PlotNumbers(flowSOM.res,
            maxNodeSize = 1,
            equalNodeSize = TRUE)
```

---

PlotOutliers

*PlotOutliers*


---

**Description**

Visual overview of outliers

**Usage**

```
PlotOutliers(fsom, outlierReport)
```

**Arguments**

`fsom`                    FlowSOM object.  
`outlierReport`        Outlier overview as generated by TestOutliers()

**Value**

Plot

**Examples**

```
# Identify the files
fcs <- flowCore::read.FCS(system.file("extdata", "68983.fcs",
                                     package = "FlowSOM"))

# Build a FlowSOM object
flowSOM.res <- FlowSOM(fcs,
                      scale = TRUE,
                      compensate = TRUE,
                      transform = TRUE,
                      toTransform = 8:18,
                      colsToUse = c(9, 12, 14:18),
                      nClus = 10,
                      seed = 1)

outlierReport <- TestOutliers(flowSOM.res)
p <- PlotOutliers(flowSOM.res, outlierReport)
```

---

PlotOverview2D

*PlotOverview2D*

---

**Description**

Plot metaclusters on scatter plots

**Usage**

```
PlotOverview2D(fsom, markerlist, metaclusters, colors = NULL, ff, ...)
```

**Arguments**

fsom	FlowSOM object, as generated by <a href="#">FlowSOM</a> . If using a FlowSOM object as generated by <a href="#">BuildMST</a> , it needs to be wrapped in a list, list(FlowSOM = fsom, metaclustering = metaclustering).
markerlist	List in which each element is a pair of marker names
metaclusters	Metaclusters of interest
colors	Named vector with color value for each metacluster. If NULL (default) color-brewer "paired" is interpolated
ff	flowFrame to use as reference for the marker names
...	Other parameters to pass on to PlotClusters2D

**Details**

Write multiple 2D scatter plots to a png file. All cells of fsom\$data are plotted in black, and those of the selected metaclusters are plotted in color.

**Value**

Nothing is returned, but a plot is drawn for every markerpair and every metacluster. The individual cells are colored, and the center of each FlowSOM cluster is indicated with a blue cross.

**See Also**

[PlotClusters2D](#)

**Examples**

```
## Deprecated - use Plot2DScatters instead ##

# Read from file, build self-organizing map and minimal spanning tree
fileName <- system.file("extdata", "68983.fcs", package = "FlowSOM")
flowSOM.res <- FlowSOM(fileName,
                       compensate = TRUE, transform = TRUE, scale = TRUE,
                       colsToUse = c(9, 12, 14:18),
                       nClus = 10,
                       seed = 1)

# Plot cells
markers_of_interest = list(c("FSC-A", "SSC-A"),
                           c("CD3", "CD19"),
                           c("TCRb", "TCRyd"),
                           c("CD4", "CD8"))
metaclusters_of_interest = 1:10

# Recommended to write to png

## Not run:
png("Markeroverview.png",
     width = 500 * length(markers_of_interest),
     height = 500 * length(metaclusters_of_interest))
Plot2DScatters(flowSOM.res,
               channelpairs = markers_of_interest,
               metaclusters = metaclusters_of_interest)
dev.off()

## End(Not run)
```

---

PlotPies

*PlotPies*

---

**Description**

Plot comparison with other clustering

**Usage**

```
PlotPies(
  fsom,
  cellTypes,
  colorPalette = grDevices::colorRampPalette(c("white", "#00007F", "blue", "#007FFF",
    "cyan", "#7FFF7F", "yellow", "#FF7F00", "red", "#7F0000")),
  ...
)
```

**Arguments**

<code>fsom</code>	FlowSOM object, as generated by <a href="#">FlowSOM</a>
<code>cellTypes</code>	Array of factors indicating the celltypes
<code>colorPalette</code>	Color palette to use.
<code>...</code>	Additional arguments to pass to <a href="#">PlotFlowSOM</a>

**Details**

Plot FlowSOM grid or tree, with pies indicating another clustering or manual gating result

**Value**

ggplot plot

**See Also**

[PlotStars](#), [PlotVariable](#), [PlotFlowSOM](#), [PlotLabels](#), [PlotNumbers](#), [PlotMarker](#), [QueryStarPlot](#), [PlotSD](#)

**Examples**

```
# Identify the files
fcs_file <- system.file("extdata", "68983.fcs", package = "FlowSOM")
gating_file <- system.file("extdata", "gatingResult.csv", package = "FlowSOM")

# Specify the cell types of interest for assigning one label per cell
cellTypes <- c("B cells",
  "gd T cells", "CD4 T cells", "CD8 T cells",
  "NK cells", "NK T cells")

# Load manual labels (e.g. GetFlowJoLabels can be used to extract labels from
# an fcs file)

gatingResult <- as.factor(read.csv(gating_file, header = FALSE)[, 1])

# Build a FlowSOM tree
flowSOM.res <- FlowSOM(fcs_file,
  scale = TRUE,
  compensate = TRUE,
```



```

        transform = TRUE,
        toTransform = 8:18,
        colsToUse = c(9, 12, 14:18),
        nClus = 10,
        seed = 1)

# Plot pies indicating the percentage of cell types present in the nodes
PlotPies(flowSOM.res,
        gatingResult,
        backgroundValues = flowSOM.res$metaclustering)

```

---

PlotSD

*PlotSD*


---

### Description

Plot FlowSOM grid or tree, colored by standard deviation.

### Usage

```
PlotSD(fsom, marker = NULL, ...)
```

### Arguments

fsom	FlowSOM object, as generated by <a href="#">FlowSOM</a>
marker	If a marker/channel is given, the sd for this marker is shown. Otherwise, the maximum ratio is used.
...	Additional arguments to pass to <a href="#">PlotFlowSOM</a>

### Value

Nothing is returned. A plot is drawn in which each node is colored depending on its standard deviation

### See Also

[PlotStars](#), [PlotVariable](#), [PlotFlowSOM](#), [PlotLabels](#), [PlotNumbers](#), [PlotMarker](#), [PlotPies](#), [QueryStarPlot](#)

### Examples

```

# Read from file, build self-organizing map and minimal spanning tree
fileName <- system.file("extdata", "68983.fcs", package = "FlowSOM")
flowSOM.res <- ReadInput(fileName, compensate = TRUE, transform = TRUE,
        scale = TRUE)
flowSOM.res <- BuildSOM(flowSOM.res, colsToUse = c(9, 12, 14:18))
flowSOM.res <- BuildMST(flowSOM.res)

```

```
PlotSD(flowSOM.res)
```

---

PlotStarLegend	<i>PlotStarLegend</i>
----------------	-----------------------

---

### Description

Plots star legend

### Usage

```
PlotStarLegend(markers, colors, starHeight = 1)
```

### Arguments

markers	Vector of markers used in legend
colors	Color palette for the legend. Can be a vector or a function.
starHeight	Star height. Default = 1.

### Details

Function makes the legend of the FlowSOM star plot

### Value

Returns nothing, but plots a legend for FlowSOM star plot

### See Also

[PlotFlowSOM](#)

### Examples

```
PlotStarLegend(c("CD3", "CD4", "CD8"),  
              FlowSOM_colors(3))
```

---

 PlotStars

*PlotStars*


---

## Description

Plot star charts

## Usage

```
PlotStars(
  fsom,
  markers = fsom$map$colsUsed,
  colorPalette = FlowSOM_colors,
  list_insteadof_ggarrange = FALSE,
  ...
)
```

## Arguments

<code>fsom</code>	FlowSOM object, as generated by <a href="#">BuildMST</a>
<code>markers</code>	Markers to plot (will be parsed by <a href="#">GetChannels</a> )
<code>colorPalette</code>	Color palette to use
<code>list_insteadof_ggarrange</code>	If FALSE (default), the plot and the legend are combined by <a href="#">ggarrange</a> . If TRUE, the separate elements are returned in a list, to allow further customization.
<code>...</code>	Additional arguments to pass to <a href="#">PlotFlowSOM</a>

## Details

Plot FlowSOM grid or tree, where each node is represented by a star chart indicating median marker values

## Value

Nothing is returned. A plot is drawn in which each node is represented by a star chart indicating the median fluorescence intensities. Resets the layout back to 1 plot at the end.

## See Also

[PlotMarker](#), [PlotVariable](#), [PlotFlowSOM](#), [PlotLabels](#), [PlotNumbers](#), [PlotPies](#), [QueryStarPlot](#), [PlotSD](#)

**Examples**

```
# Read from file, build self-organizing map and minimal spanning tree
fileName <- system.file("extdata", "68983.fcs", package = "FlowSOM")
flowSOM.res <- FlowSOM(fileName, compensate = TRUE, transform = TRUE,
                      scale = TRUE, colsToUse = c(9, 12, 14:18))

# Plot stars indicating the MFI of the cells present in the nodes
PlotStars(flowSOM.res, backgroundValues = flowSOM.res$metaclustering)

newLayout <- igraph::layout_with_fr(flowSOM.res[["MST"]][["graph"]])
PlotStars(flowSOM.res, backgroundValues = flowSOM.res$metaclustering,
          view = newLayout)

PlotStars(flowSOM.res, backgroundValues = flowSOM.res$metaclustering,
          view = "grid")
```

---

PlotVariable

*PlotVariable*


---

**Description**

Plot a variable for all nodes

**Usage**

```
PlotVariable(
  fsom,
  variable,
  variableName = "",
  colorPalette = FlowSOM_colors,
  lim = NULL,
  ...
)
```

**Arguments**

<code>fsom</code>	FlowSOM object
<code>variable</code>	A vector containing a value for every cluster
<code>variableName</code>	Label to show on the legend
<code>colorPalette</code>	Color palette to use. Can be a function or a vector.
<code>lim</code>	Limits for the scale
<code>...</code>	Additional arguments to pass to <code>PlotFlowSOM</code> , e.g. <code>view</code> , <code>backgroundValues</code> , <code>equalNodeSize</code> ...

**Details**

Plot FlowSOM grid or tree, colored by node values given in variable

**See Also**

[PlotStars](#), [QueryStarPlot](#), [PlotFlowSOM](#), [PlotLabels](#), [PlotNumbers](#), [PlotMarker](#), [PlotPies](#), [PlotSD](#)

**Examples**

```
# Build FlowSOM model
fileName <- system.file("extdata", "68983.fcs", package = "FlowSOM")
flowSOM.res <- FlowSOM(fileName,
                      compensate = TRUE, transform = TRUE, scale = FALSE,
                      colsToUse = c(9, 12, 14:18),
                      nClus = 10,
                      seed = 1)

# Plot some random values
rand <- runif(flowSOM.res$map$nNodes)
PlotVariable(flowSOM.res,
             variable = rand,
             variableName = "Random")

PlotVariable(flowSOM.res,
             variable = flowSOM.res$metaclustering,
             variableName = "Metaclustering") %>%
  AddLabels(labels = flowSOM.res$metaclustering)
```

---

print.FlowSOM	<i>Print FlowSOM object</i>
---------------	-----------------------------

---

**Description**

Print FlowSOM object

**Usage**

```
## S3 method for class 'FlowSOM'
print(x, ...)
```

**Arguments**

x	FlowSOM object to print information about
...	Further arguments, not used

**Examples**

```
fileName <- system.file("extdata", "68983.fcs", package = "FlowSOM")
flowSOM.res <- FlowSOM(fileName, compensate = TRUE, transform = TRUE,
                      scale = TRUE, colsToUse = c(9, 12, 14:18), nClus = 10)
print(flowSOM.res)
```

---

Purity	<i>Calculate mean weighted cluster purity</i>
--------	---

---

**Description**

Calculate mean weighted cluster purity

**Usage**

```
Purity(realClusters, predictedClusters, weighted = TRUE)
```

**Arguments**

realClusters	array with real cluster values
predictedClusters	array with predicted cluster values
weighted	logical. Should the mean be weighted depending on the number of points in the predicted clusters

**Value**

Mean purity score, worst score, number of clusters with score < 0.75

**Examples**

```
# Generate some random data as an example
realClusters <- sample(1:5, 100, replace = TRUE)
predictedClusters <- sample(1:6, 100, replace = TRUE)

# Calculate the FMeasure
Purity(realClusters, predictedClusters)
```

---

QueryMultiple	<i>QueryMultiple</i>
---------------	----------------------

---

**Description**

Function which takes a named list of multiple cell types, where every item is a named vector with values "high"/"low" and the names correspond to the markers or channels (e.g. as generated by parse\_markertable).

**Usage**

```
QueryMultiple(fsom, cellTypes, plotFile = "queryMultiple.pdf", ...)
```

**Arguments**

<code>fsom</code>	FlowSOM object
<code>cellTypes</code>	Description of the cell types. Named list, with one named vector per cell type containing "high"/"low" values
<code>plotFile</code>	Path to a pdf file to save the plots (default is <code>queryMultiple.pdf</code> ). If NULL, no plots will be generated
<code>...</code>	Additional arguments to pass to <a href="#">QueryStarPlot</a>

**Value**

A label for every FlowSOM cluster (Unknown or one of the celltype names of the list, if selected by `QueryStarPlot`)

**Examples**

```
file <- system.file("extdata", "68983.fcs", package = "FlowSOM")
ff <- flowCore::read.FCS(file)
# Use the wrapper function to build a flowSOM object (saved in flowSOM.res)
# and a metaclustering (saved in flowSOM.res[["metaclustering"]])
flowSOM.res <- FlowSOM(ff, compensate = TRUE, transform = TRUE, scale = TRUE,
  colsToUse = c(9, 12, 14:18), nClus = 10, silent = FALSE,
  xdim = 7, ydim = 7)
cellTypes <- list("CD8 T cells" = c("PE-Cy7-A" = "high",
  "APC-Cy7-A" = "high",
  "Pacific Blue-A" = "high"),
  "B cells" = c("PE-Cy5-A" = "high"),
  "NK cells" = c("PE-A" = "high",
  "PE-Cy7-A" = "low",
  "APC-Cy7-A" = "low"))
query_res <- QueryMultiple(flowSOM.res, cellTypes, "query_multiple.pdf")
```

---

QueryStarPlot

*QueryStarPlot*

---

**Description**

Query a certain cell type

**Usage**

```
QueryStarPlot(
  fsom,
  query,
  plot = TRUE,
  colorPalette = FlowSOM_colors,
  backgroundColors = "#CA0020",
  ...
)
```

**Arguments**

<code>fsom</code>	FlowSOM object, as generated by <a href="#">BuildMST</a>
<code>query</code>	Array containing "high" or "low" (or abbreviations) for the specified column names of the FlowSOM data.
<code>plot</code>	If true, a plot with a gradient of scores for the nodes is shown.
<code>colorPalette</code>	Color palette to be used for colors for "stars", "pies" or "marker". Can be either a function or an array specifying colors.
<code>backgroundColors</code>	Color to use for nodes with a high score in the plot. Default is red.
<code>...</code>	Additional arguments to pass to <a href="#">PlotFlowSOM</a>

**Details**

Identify nodes in the tree which resemble a certain profile of "high" or "low" marker expressions.

**Value**

A list, containing the ids of the selected nodes, the individual scores for all nodes and the scores for each marker for each node

**See Also**

[PlotStars](#), [PlotVariable](#), [PlotFlowSOM](#), [PlotLabels](#), [PlotNumbers](#), [PlotMarker](#), [PlotPies](#), [PlotSD](#)

**Examples**

```
file <- system.file("extdata", "68983.fcs", package="FlowSOM")
flowSOM.res <- FlowSOM(file, compensate = TRUE, transform = TRUE,
  scale = TRUE, colsToUse = c(9, 12, 14:18), nClus = 10,
  silent = FALSE, xdim = 7, ydim = 7)
query <- c("CD3" = "high", #CD3
  "CD4" = "low", #TCRb
  "CD8" = "high") #CD8
query_res <- QueryStarPlot(flowSOM.res, query, equalNodeSize = TRUE)

cellTypes <- factor(rep("Unlabeled", 49),
  levels = c("Unlabeled", "CD8 T cells"))
cellTypes[query_res$selected] <- "CD8 T cells"
PlotStars(flowSOM.res,
  backgroundValues = cellTypes,
  backgroundColors = c("#FFFFFF00", "#ca0020aa"))
```



---

query_multiple	<i>query_multiple</i>
----------------	-----------------------

---

### Description

Function which takes a named list of multiple cell types, where every item is a named vector with values "high"/"low" and the names correspond to the markers or channels (e.g. as generated by `parse_markertable`).

### Usage

```
query_multiple(fsom, cell_types, pdf_name = "query_multiple.pdf", ...)
```

### Arguments

<code>fsom</code>	FlowSOM object
<code>cell_types</code>	Description of the cell types. Named list, with one named vector per cell type containing "high"/"low" values
<code>pdf_name</code>	Path to a pdf file to save figures
<code>...</code>	Additional arguments to pass to <a href="#">QueryStarPlot</a>

### Value

A label for every FlowSOM cluster (Unknown or one of the celltype names of the list, if selected by `QueryStarPlot`)

### See Also

[QueryStarPlot](#)

### Examples

```
file <- system.file("extdata", "68983.fcs", package="FlowSOM")
ff <- flowCore::read.FCS(file)
# Use the wrapper function to build a flowSOM object (saved in flowSOM.res)
# and a metaclustering (saved in flowSOM.res[["metaclustering"]])
flowSOM.res <- FlowSOM(ff,compensate = TRUE, transform = TRUE,scale = TRUE,
  colsToUse = c(9,12,14:18), nClus = 10, silent = FALSE,
  xdim=7, ydim=7)
cell_types <- list("CD8 T cells" = c("PE-Cy7-A" = "high",
  "APC-Cy7-A" = "high",
  "Pacific Blue-A" = "high"),
  "B cells" = c("PE-Cy5-A" = "high"),
  "NK cells" = c("PE-A" = "high",
  "PE-Cy7-A" = "low",
  "APC-Cy7-A" = "low"))
query_res <- QueryMultiple(flowSOM.res, cell_types, "query_multiple.pdf")
```

ReadInput

*Read FCS-files or flowFrames***Description**

Take some input and return FlowSOM object containing a matrix with the preprocessed data (compensated, transformed, scaled)

**Usage**

```
ReadInput(
  input,
  pattern = ".fcs",
  compensate = FALSE,
  spillover = NULL,
  transform = FALSE,
  toTransform = NULL,
  transformFunction = flowCore::logicleTransform(),
  transformList = NULL,
  scale = FALSE,
  scaled.center = TRUE,
  scaled.scale = TRUE,
  silent = FALSE
)
```

**Arguments**

input	a flowFrame, a flowSet, a matrix with column names or an array of paths to files or directories
pattern	if input is an array of file- or directorynames, select only files containing pattern
compensate	logical, does the data need to be compensated
spillover	spillover matrix to compensate with If NULL and compensate = TRUE, we will look for \$SPILL description in FCS file.
transform	logical, does the data need to be transformed
toTransform	column names or indices that need to be transformed. Will be ignored if transformList is given. If NULL and transform = TRUE, column names of \$SPILL description in FCS file will be used.
transformFunction	Defaults to logicleTransform()
transformList	transformList to apply on the samples.
scale	logical, does the data needs to be rescaled
scaled.center	see <a href="#">scale</a>
scaled.scale	see <a href="#">scale</a>
silent	if TRUE, no progress updates will be printed. Default = FALSE

**Value**

FlowSOM object containing the data, which can be used as input for the BuildSOM function

**See Also**

[scale](#), [BuildSOM](#)

**Examples**

```
# Read from file
fileName <- system.file("extdata", "68983.fcs", package = "FlowSOM")
flowSOM.res <- ReadInput(fileName, compensate = TRUE, transform = TRUE,
                        scale = TRUE)

# Or read from flowFrame object
ff <- flowCore::read.FCS(fileName)
ff <- flowCore::compensate(ff, flowCore::keyword(ff)[["SPILL"]])
ff <- flowCore::transform(ff,
                        flowCore::transformList(colnames(flowCore::keyword(ff)[["SPILL"]]),
                        flowCore::logicleTransform()))
flowSOM.res <- ReadInput(ff, scale = TRUE)

# Build the self-organizing map and the minimal spanning tree
flowSOM.res <- BuildSOM(flowSOM.res, colsToUse = c(9, 12, 14:18))
flowSOM.res <- BuildMST(flowSOM.res)

# Apply metaclustering
metacl <- MetaClustering(flowSOM.res$map$codes,
                        "metaClustering_consensus", max = 10)

# Get metaclustering per cell
flowSOM.clustering <- metacl[flowSOM.res$map$mapping[, 1]]
```

---

SaveClustersToFCS

*Write FlowSOM clustering results to the original FCS files*

---

**Description**

Write FlowSOM clustering results to the original FCS files

**Usage**

```
SaveClustersToFCS(
  fsom,
  originalFiles,
  preprocessedFiles = NULL,
  selectionColumn = NULL,
  silent = FALSE,
```

```

    outputDir = ".",
    suffix = "_FlowSOM.fcs",
    ...
)

```

### Arguments

<code>fsom</code>	FlowSOM object as generated by BuildSOM
<code>originalFiles</code>	FCS files that should be extended
<code>preprocessedFiles</code>	FCS files that correspond to the input of FlowSOM, If NULL (default), the originalFiles are used.
<code>selectionColumn</code>	Column of the FCS file indicating the original cell ids. If NULL (default), no selection is made.
<code>silent</code>	If FALSE (default), print some extra output
<code>outputDir</code>	Directory to save the FCS files. Default to the current working directory (".")
<code>suffix</code>	Suffix added to the filename. Default <code>_FlowSOM.fcs</code>
<code>...</code>	Options to pass on to the <code>read.FCS</code> function (e.g. <code>truncate_max_range</code> )

### Value

Saves the extended FCS file as `[originalName]_FlowSOM.fcs`

### Examples

```

fileName <- system.file("extdata", "68983.fcs", package = "FlowSOM")
flowSOM.res <- FlowSOM(fileName, compensate = TRUE, transform = TRUE,
                      scale = TRUE, colsToUse = c(9, 12, 14:18), nClus = 10)
SaveClustersToFCS(flowSOM.res, fileName)

```

---

ScaleStarHeights      *ScaleStarHeights*

---

### Description

Scales starheights

### Usage

```
ScaleStarHeights(data, nodeSizes)
```

### Arguments

<code>data</code>	Median values of relevant markers extracted from FlowSOM object
<code>nodeSizes</code>	A vector that is returned from <code>ParseNodeSize</code>

**Details**

Function that scales the star values between 0 and the node size

**Value**

A dataframe consisting of the scaled values of the stars. The stars are scaled between 0 and the maximum of all stars

**See Also**

[PlotFlowSOM](#), [ParseNodeSize](#), [AutoMaxNodeSize](#)

---

SOM

*Build a self-organizing map*

---

**Description**

Build a self-organizing map

**Usage**

```
SOM(
  data,
  xdim = 10,
  ydim = 10,
  rlen = 10,
  mst = 1,
  alpha = c(0.05, 0.01),
  radius = stats::quantile(nhbrdist, 0.67) * c(1, 0),
  init = FALSE,
  initf = Initialize_KWSP,
  distf = 2,
  silent = FALSE,
  map = TRUE,
  codes = NULL,
  importance = NULL
)
```

**Arguments**

<code>data</code>	Matrix containing the training data
<code>xdim</code>	Width of the grid
<code>ydim</code>	Hight of the grid
<code>rlen</code>	Number of times to loop over the training data for each MST
<code>mst</code>	Number of times to build an MST

alpha	Start and end learning rate
radius	Start and end radius
init	Initialize cluster centers in a non-random way
initf	Use the given initialization function if init == T (default: Initialize_KWSP)
distf	Distance function (1 = manhattan, 2 = euclidean, 3 = chebyshev, 4 = cosine)
silent	If FALSE, print status updates
map	If FALSE, data is not mapped to the SOM. Default TRUE.
codes	Cluster centers to start with
importance	array with numeric values. Parameters will be scaled according to importance

**Value**

A list containing all parameter settings and results

**References**

This code is strongly based on the kohonen package. R. Wehrens and L.M.C. Buydens, Self- and Super-organising Maps in R: the kohonen package J. Stat. Softw., 21(5), 2007

**See Also**

[BuildSOM](#)

---

TestOutliers

*TestOutliers*

---

**Description**

Test if any cells are too far from their cluster centers

**Usage**

```
TestOutliers(
  fsm,
  madAllowed = 4,
  fsmReference = NULL,
  plotFile = NULL,
  channels = NULL
)
```

**Arguments**

<code>fSom</code>	FlowSOM object
<code>madAllowed</code>	Number of median absolute deviations allowed. Default = 4.
<code>fSomReference</code>	FlowSOM object to use as reference. If NULL (default), the original <code>fSom</code> object is used.
<code>plotFile</code>	If NULL (default), no plot will be created. If a filepath is given for a pdf, the plot will be written in the corresponding file
<code>channels</code>	If channels are given, the number of outliers in the original space for those channels will be calculated and added to the final results table.

**Details**

For every cluster, the distance from the cells to the cluster centers is used to label cells which deviate too far as outliers. The threshold is chosen as the median distance + `madAllowed` times the median absolute deviation of the distances.

**Value**

An outlier report

**See Also**

[FlowSOMSubset](#) if you want to get a subset of the current data instead of a new dataset

**Examples**

```
# Build FlowSom result
fileName <- system.file("extdata", "68983.fcs", package = "FlowSOM")
ff <- flowCore::read.FCS(fileName)
flowSOM.res <- FlowSOM(ff,
  compensate = TRUE, transform = TRUE, scale = TRUE,
  colsToUse = c(9, 12, 14:18),
  nClus = 10)

# Map new data
outlier_report <- TestOutliers(flowSOM.res,
  madAllowed = 5,
  channels = flowSOM.res$map$colsUsed)

# Number of cells which is an outlier for x channels
outlier_on_multiple_markers <- table(rowSums(outlier_report$channel_specific != 0))
outlier_type <- paste(GetClusters(flowSOM.res),
  apply(outlier_report$channel_specific, 1, paste0, collapse = ""))
outlier_counts <- table(grep(".*1.*", outlier_type, value = TRUE))
outliers_of_interest <- names(which(outlier_counts > 10))
outlier_boolean <- outlier_type %in% outliers_of_interest
```

UpdateFlowSOM

*UpdateFlowSOM*

---

**Description**

Update old FlowSOM object to a new one and checks if it is a flowSOM object

**Usage**

```
UpdateFlowSOM(fsom)
```

**Arguments**

fsom                      FlowSOM object, as generated by [BuildMST](#) or [FlowSOM](#)

**Details**

Determines whether or not the fsom input is of class "FlowSOM" and returns the FlowSOM object and metaclustering object inside fsom

**Value**

A FlowSOM object

**See Also**

[PlotFlowSOM](#)

---

UpdateMetaclusters

*UpdateMetaclusters*

---

**Description**

Adapt the metacluster levels. Can be used to rename the metaclusters, split or merge existing metaclusters, add a metaclustering and/or reorder the levels of the metaclustering.

**Usage**

```
UpdateMetaclusters(  
  fsom,  
  newLabels = NULL,  
  clusterAssignment = NULL,  
  levelOrder = NULL  
)
```



**Arguments**

<code>fsom</code>	Result of calling the FlowSOM function.
<code>newLabels</code>	Optional. Named vector, with the names the original metacluster names and the values the replacement. Can be used to rename or merge metaclusters.
<code>clusterAssignment</code>	Optional. Either a named vector, with the names the cluster numbers (characters) or a vector of length <code>NClusters(fsom)</code> . Can be used to assign clusters to existing or new metaclusters.
<code>levelOrder</code>	Optional. Vector showing the preferred order of the <code>fsom</code> metacluster levels.

**Value**

Updated FlowSOM object

**Examples**

```

fileName <- system.file("extdata", "68983.fcs", package = "FlowSOM")
ff <- flowCore::read.FCS(fileName)
ff <- flowCore::compensate(ff, flowCore::keyword(ff)[["SPILL"]])
ff <- flowCore::transform(ff,
  flowCore::transformList(colnames(flowCore::keyword(ff)[["SPILL"]]),
    flowCore::logicTransform()))
flowSOM.res <- FlowSOM(ff,
  scale = TRUE,
  colsToUse = c(9, 12, 14:18),
  nClus = 10,
  seed = 1)

PlotStars(flowSOM.res, backgroundValues = flowSOM.res$metaclustering)
GetCounts(flowSOM.res)

# Merge MC8 and MC9
flowSOM.res <- UpdateMetaclusters(flowSOM.res, newLabels = c("8" = "8+9",
  "9" = "8+9"))

PlotStars(flowSOM.res, backgroundValues = flowSOM.res$metaclustering)
GetCounts(flowSOM.res)

# Split cluster 24 from metacluster 2 and order the metacluster levels
flowSOM.res <- UpdateMetaclusters(flowSOM.res,
  clusterAssignment = c("24" = "debris?"),
  levelOrder = c("debris?", as.character(c(1:7)),
    "8+9", "10"))

PlotStars(flowSOM.res, backgroundValues = flowSOM.res$metaclustering)
PlotNumbers(flowSOM.res, level = "metaclusters")

GetCounts(flowSOM.res)

```

---

UpdateNodeSize	<i>UpdateNodeSize</i>
----------------	-----------------------

---

**Description**

Update nodesize of FlowSOM object

**Usage**

```
UpdateNodeSize(
  fsom,
  count = NULL,
  reset = FALSE,
  transform = sqrt,
  maxNodeSize = 15,
  shift = 0,
  scale = NULL
)
```

**Arguments**

<code>fsom</code>	FlowSOM object, as generated by <a href="#">BuildMST</a>
<code>count</code>	Absolute cell count of the sample
<code>reset</code>	Logical. If TRUE, all nodes get the same size
<code>transform</code>	Transformation function. Use e.g. square root to let counts correspond with area of node instead of radius
<code>maxNodeSize</code>	Maximum node size after rescaling. Default: 15
<code>shift</code>	Shift of the counts, defaults to 0
<code>scale</code>	Scaling of the counts, defaults to the maximum of the value minus the shift. With shift and scale set as default, the largest node will be <code>maxNodeSize</code> and an empty node will have size 0

**Details**

Add size property to the graph based on cellcount for each node

**Value**

Updated FlowSOM object

**See Also**

[BuildMST](#)

**Examples**

```
# Read from file, build self-organizing map and minimal spanning tree
fileName <- system.file("extdata", "68983.fcs", package="FlowSOM")
flowSOM.res <- ReadInput(fileName, compensate=TRUE, transform=TRUE,
                        scale=TRUE)
flowSOM.res <- BuildSOM(flowSOM.res, colsToUse=c(9,12,14:18))
flowSOM.res <- BuildMST(flowSOM.res)

# Give all nodes same size
PlotStars(flowSOM.res, equalNodeSize = TRUE)

# Node sizes relative to amount of cells assigned to the node
PlotStars(flowSOM.res)
```

---

%>%

*Pipe operator*

---

**Description**

See `magrittr::%>%` for details.

**Usage**

```
lhs %>% rhs
```

# Index

## \* internal

- [%>%, 91](#)
  - [%>%, 91, 91](#)
- [AddAnnotation, 4](#)
- [AddBackground, 5, 7–11](#)
- [AddFlowFrame, 6](#)
- [AddLabels, 5, 6, 7–11, 60](#)
- [AddMST, 7, 49](#)
- [AddNodes, 5, 7, 8, 9–11](#)
- [AddPies, 5, 7, 8, 9, 10, 11, 60](#)
- [AddScale, 9](#)
- [AddStars, 5, 7–9, 10, 11, 60](#)
- [AddStarsPies, 7, 8, 11](#)
- [AggregateFlowFrames, 11, 58](#)
- [AutoMaxNodeSize, 13, 50, 85](#)
  
- [BuildMST, 9, 10, 13, 14, 15, 18, 20, 53–56, 61, 67, 70, 75, 80, 88, 90](#)
- [BuildSOM, 13, 14, 14, 18, 83, 86](#)
  
- [ceiling, 12](#)
- [CountGroups, 15, 61](#)
  
- [Dist.MST, 16](#)
  
- [FlowSOM, 7, 17, 19, 48, 50–52, 60, 63, 64, 67, 68, 70, 72, 73, 88](#)
- [FlowSOM\\_colors, 21](#)
- [FlowSOMmary, 19](#)
- [FlowSOMSubset, 20, 47, 87](#)
- [FMeasure, 21](#)
  
- [get\\_channels, 35, 36](#)
- [get\\_markers, 36, 36](#)
- [GetChannels, 22, 30](#)
- [GetClusterCVs, 23](#)
- [GetClusterMFIs, 23](#)
- [GetClusterPercentagesPositive, 24](#)
- [GetClusters, 25](#)
- [GetCounts, 25](#)
  
- [GetCVs, 26](#)
- [GetFeatures, 27, 37](#)
- [GetFlowJoLabels, 28](#)
- [GetMarkers, 22, 30](#)
- [GetMetaclusterCVs, 31](#)
- [GetMetaclusterMFIs, 31](#)
- [GetMetaclusterPercentagesPositive, 32](#)
- [GetMetaclusters, 33](#)
- [GetMFIs, 34](#)
- [GetPercentages, 35](#)
- [gg\\_color\\_hue, 37](#)
- [grep, 22, 30](#)
- [GroupStats, 37](#)
  
- [Initialize\\_KWSP, 39](#)
- [Initialize\\_PCA, 40](#)
  
- [ManualVector, 41](#)
- [MapDataToCodes, 41](#)
- [MetaclusterCVs, 42](#)
- [MetaClustering, 18, 42, 44](#)
- [metaClustering\\_consensus, 43, 43](#)
- [MetaclusterMFIs, 44](#)
  
- [NClusters, 45](#)
- [NewData, 45, 64](#)
- [NMetaclusters, 47](#)
  
- [ParseArcs, 9–11, 48, 49–51](#)
- [ParseEdges, 7, 48, 48, 50, 51](#)
- [ParseLayout, 49](#)
- [ParseNodeSize, 13, 48, 49, 49, 51, 85](#)
- [ParseQuery, 48–50, 50, 51](#)
- [ParseSD, 48–51, 51](#)
- [Plot2DScatters, 51](#)
- [PlotCenters, 53, 55](#)
- [PlotClusters2D, 54, 71](#)
- [PlotDimRed, 56](#)
- [PlotFileScatters, 57](#)
- [PlotFlowSOM, 5, 7–11, 13, 48–51, 59, 61–63, 66, 68, 69, 72–77, 80, 85, 88](#)

PlotGroups, 61  
PlotLabels, 7, 61, 62, 63, 68, 69, 72, 73, 75,  
77, 80  
PlotManualBars, 64  
PlotMarker, 8, 54, 61–63, 65, 69, 72, 73, 75,  
77, 80  
PlotNode, 67  
PlotNumbers, 7, 55, 61–63, 68, 68, 72, 73, 75,  
77, 80  
PlotOutliers, 69  
PlotOverview2D, 70  
PlotPies, 9, 11, 29, 54, 61–63, 69, 71, 73, 75,  
77, 80  
PlotSD, 51, 61–63, 69, 72, 73, 75, 77, 80  
PlotStarLegend, 74  
PlotStars, 4, 10, 11, 14, 54, 61–63, 66, 68,  
69, 72, 73, 75, 77, 80  
PlotVariable, 8, 61–63, 66, 69, 72, 73, 75,  
76, 80  
print.FlowSOM, 77  
Purity, 78  
  
query\_multiple, 81  
QueryMultiple, 78  
QueryStarPlot, 51, 61–63, 69, 72, 73, 75, 77,  
79, 79, 81  
  
ReadInput, 6, 14, 15, 18, 46, 82  
  
SaveClustersToFCS, 83  
scale, 18, 46, 82, 83  
ScaleStarHeights, 13, 84  
SOM, 85  
  
TestOutliers, 14, 86  
  
UpdateFlowSOM, 88  
UpdateMetaclusters, 88  
UpdateNodeSize, 90