Katrien Quintelier, Artuur Couckuyt, Annelies Emmaneel, Sofie Van Gassen, Yvan Saeys

Ghent University

September, 2014

Abstract

This vignette describes a protocol for analyzing high-dimensional cytometry data using *FlowSOM*, a clustering and visualization algorithm based on a self-organizing map (SOM). FlowSOM is used to distinguish cell populations from cytometry data in an unsupervised way and can help to gain deeper insights in fields such as immunology and oncology.

1 Installation

```
> # install.packages("BiocManager")
> # BiocManager::install("FlowSOM")
```

2 Preprocessing

FlowSOM handles different inputs, such as a flowFrame, a flowSet or an array of filepaths. For this purpose we will make use of a flowFrame. This allows easier preprocessing. We start with compensating the data and then we transform the data with the logicle function. For CyTOF data an arcsinh transformation is preferred which is also found in the *flowCore* package. Besides compensation and transformation, we also recommend cleaning the data by removing margin events and by using cleaning algorithms.

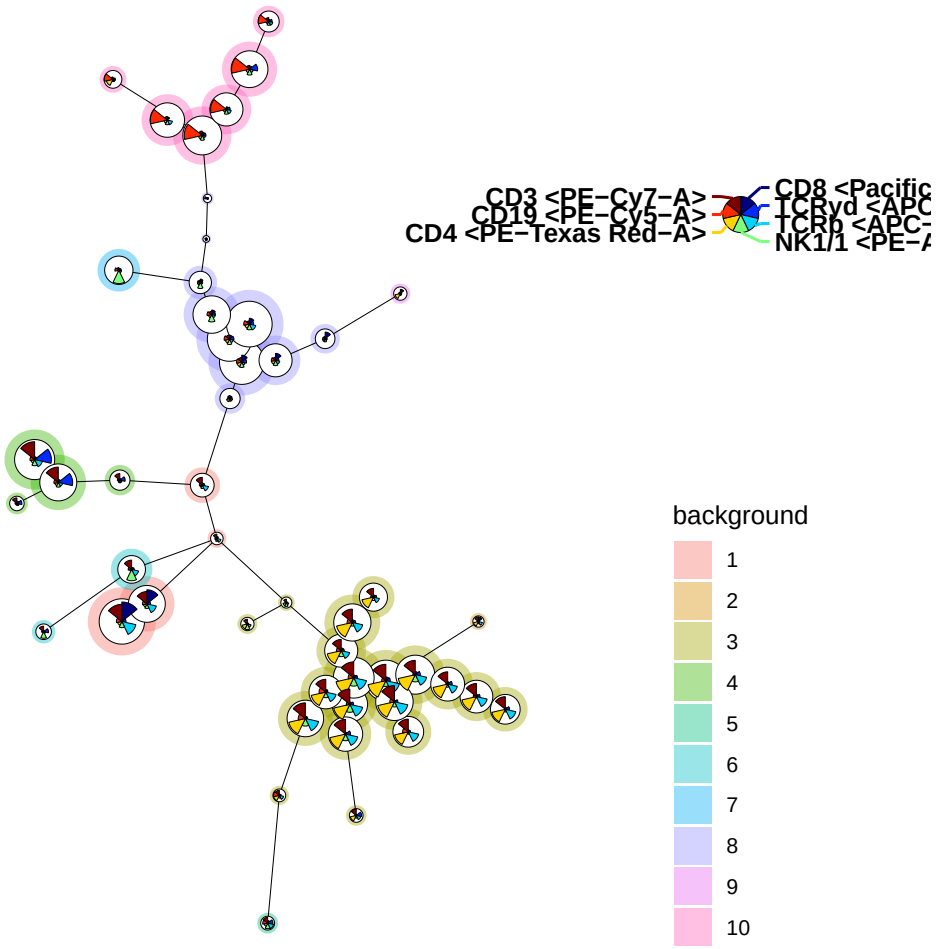
```
> fileName <- system.file("extdata", "68983.fcs", package="FlowSOM")
> ff <- flowCore::read.FCS(fileName)
> # Compensation
> comp <- flowCore::keyword(ff)[["SPILL"]]
> ff <- flowWorkspace::compensate(ff, comp)
> # Transformation
> transformList <- flowCore::estimateLogicle(ff, channels = colnames(comp))
> ff <- flowWorkspace::transform(ff, transformList)
```

3 The easy way

The easiest way to use this package is using the wrapper function `FlowSOM`. It has less options than using the separate functions, but in general it has enough power. It returns a list, of which the first item is the FlowSOM object (as required as input by many functions in this package) and the second item is the result of the metaclustering.

```
> set.seed(42)
> library(FlowSOM)
> fSOM <- FlowSOM(ff,
+                 # Input options:
+                 compensate = FALSE,
+                 transform = FALSE,
+                 scale = FALSE,
+                 # SOM options:
+                 colsToUse = c(9, 12, 14:18), xdim = 7, ydim = 7,
+                 # Metaclustering options:
+                 nClus = 10)
```

```
> p <- PlotStars(fSOM, backgroundValues = fSOM$metaclustering)
> print(p, newpage = FALSE)
```



A pdf file with most relevant figures can be generated using the FlowSOMmary function.

```
> FlowSOMmary(fsom = fSOM,
+             plotFile = "FlowSOMmary.pdf")

pdf
  2
```

From the resulting object you can get the cluster and metacluster label for every individual cell.

```
> head(GetClusters(fSOM))

[1] 14 29 45 31 42  1

> head(GetMetaclusters(fSOM))

[1] 3  4  10 7  8  1
Levels: 1 2 3 4 5 6 7 8 9 10
```

4 Reading the data

The FlowSOM package has several input options.

The first possibility is to use an array of character strings, specifying paths to files or directories. When given a path to a directory, all files in the directory will be considered. This process does not happen recursively. You can specify a pattern to use only a selection of the files. The default pattern is `".fcs"`, making sure that only fcs-files are selected. When you are already working with your data in *R*, it might be easier to use a *flowFrame* or *flowSet* from the *flowCore* package as input. This is also supported. If multiple paths or a *flowSet* are provided, all data will be concatenated. You should check and apply normalization if needed using other packages.

When reading the data, several preprocessing options are available. The data can be automatically compensated using a specified matrix, or using the `$SPILL` variable from the fcs-file (when `compensate` is `TRUE` but no value is given for `spillover`). The data can be transformed for specified columns. If no columns are provided, all columns from the spillover matrix will be transformed. Finally, the data can be scaled. By default, it will scale to a mean of zero and standard deviation of one. However, specific scaling parameters can be set (see the base *R* `scale` function for more detail).

```
> set.seed(42)
> library(flowCore)
> library(FlowSOM)
> fileName <- system.file("extdata", "68983.fcs", package = "FlowSOM")
> fSOM <- ReadInput(fileName, compensate = TRUE, transform = TRUE,
+                   toTransform = c(8:18), scale = TRUE)
> ff <- suppressWarnings(flowCore::read.FCS(fileName))
> fSOM <- ReadInput(ff, compensate = TRUE, transform = TRUE,
+                   toTransform = c(8:18), scale = TRUE)
```

This function returns a FlowSOM object, which is actually a *list* containing several parameters. The data is stored as a matrix in `$data`, and all parameter settings to read the data are also stored. The begin and end indices of the subsets from the different files can be found in `$metadata`.

```
> str(fSOM, max.level = 2)

List of 13
 $ pattern          : chr ".fcs"
 $ compensate       : logi TRUE
 $ spillover        : NULL
 $ transform        : logi TRUE
 $ toTransform      : chr [1:11] "FITC-A" "Pacific Blue-A" "AmCyan-A" "Qdot 605-A" ...
 $ transformFunction:Formal class 'transform' [package "flowCore"] with 2 slots
 $ transformList    : NULL
 $ scale            : logi TRUE
 $ prettyColnames   : Named chr [1:18] "Time <Time>" "FSC-A <FSC-A>" "FSC-H <FSC-H>" "FSC-W <FSC-W>" ...
  ..- attr(*, "names")= chr [1:18] "Time" "FSC-A" "FSC-H" "FSC-W" ...
 $ data             : num [1:19225, 1:18] -1.65 -1.65 -1.65 -1.65 -1.65 ...
  ..- attr(*, "dimnames")=List of 2
 $ metaData         :List of 1
  ..$ 68983.fcs: num [1:2] 1 19225
 $ scaled.center    : Named num [1:18] 3356 88594 68698 84405 36886 ...
  ..- attr(*, "names")= chr [1:18] "Time" "FSC-A" "FSC-H" "FSC-W" ...
 $ scaled.scale     : Named num [1:18] 2038 15064 3236 12997 13923 ...
  ..- attr(*, "names")= chr [1:18] "Time" "FSC-A" "FSC-H" "FSC-W" ...
 - attr(*, "class")= chr "FlowSOM"
```

5 Building the self-organizing map

The next step in the algorithm is to build a self-organizing map. Several parameters for the self-organizing map algorithm can be provided, such as the dimensions of the grid, the learning rate, the number of times the dataset has to be presented. However, the most important parameter to decide is on which columns the self-organizing map should be trained. This should contain all the parameters that are useful to identify cell types, and exclude parameters of which you want to study the behavior on all cell types such as activation markers.

The `BuildSOM` function expects a FlowSOM object as input, and will return a FlowSOM object with all information about the self organizing map added in the `$map` parameter of the FlowSOM object.

```
> fSOM <- BuildSOM(fSOM, colsToUse = c(9, 12, 14:18))
> str(fSOM$map, max.level = 2)

List of 17
 $ xdim        : num 10
 $ ydim        : num 10
 $ rlen        : num 10
 $ mst         : num 1
 $ alpha       :List of 1
  ..$ : num [1:2] 0.05 0.01
 $ radius      :List of 1
```

```
  ..$ : num [1:2] 6 0
 $ init        : logi FALSE
 $ distf       : num 2
 $ grid        :'data.frame':        100 obs. of  2 variables:
  ..$ Var1: int [1:100] 1 2 3 4 5 6 7 8 9 10 ...
  ..$ Var2: int [1:100] 1 1 1 1 1 1 1 1 1 1 ...
  ..- attr(*, "out.attrs")=List of 2
 $ codes       : num [1:100, 1:7] -0.49455 -0.48778 -0.55062 -0.00147 -0.60334 ...
  ..- attr(*, "dimnames")=List of 2
 $ mapping     : num [1:19225, 1:2] 1 92 9 95 49 90 100 48 90 24 ...
 $ nNodes      : int 100
 $ colsUsed    : Named chr [1:7] "Pacific Blue-A" "APC-A" "APC-Cy7-A" "PE-A" ...
  ..- attr(*, "names")= chr [1:7] "CD8" "TCRyd" "TCRb" "NK1/1" ...
 $ medianValues: num [1:100, 1:18] -0.573 -0.342 -0.553 -0.262 0.118 ...
  ..- attr(*, "dimnames")=List of 2
 $ cvValues    : num [1:100, 1:18] -3.1 -4.03 -3.45 -12.26 9.77 ...
  ..- attr(*, "dimnames")=List of 2
 $ sdValues    : num [1:100, 1:18] 0.991 0.914 1.013 0.998 0.988 ...
  ..- attr(*, "dimnames")=List of 2
 $ pctgs       : Named num [1:100] 0.012 0.0108 0.0151 0.0113 0.0144 ...
  ..- attr(*, "names")= chr [1:100] "1" "2" "3" "4" ...
```

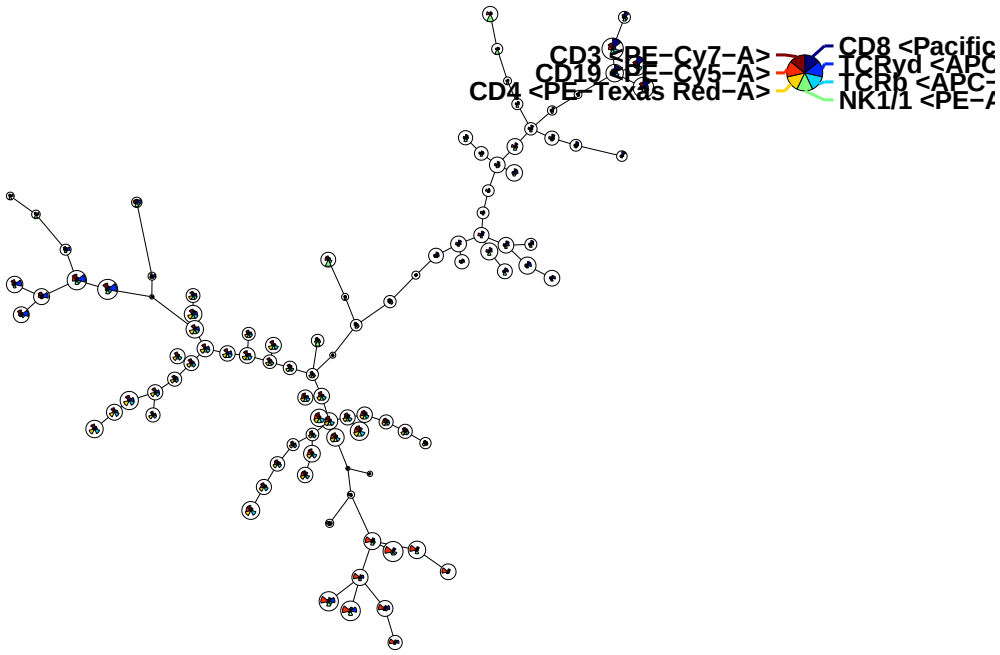
6 Building the minimal spanning tree

The third step of FlowSOM is to build the minimal spanning tree. This will again return a FlowSOM object, with extra information contained in the `$MST` parameter.

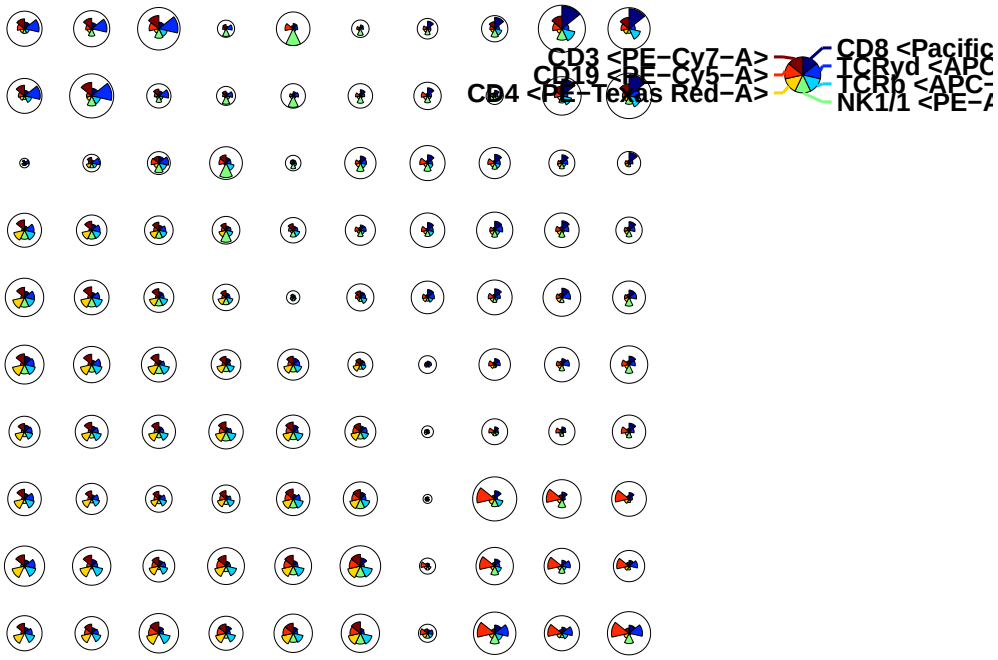
```
> fSOM <- BuildMST(fSOM)
> str(fSOM$MST, max.level =  1)

List of 2
 $ graph:List of 10
  ..- attr(*, "class")= chr "igraph"
 $ l    : num [1:100, 1:2] 2.05 4.1 3.92 4.67 5.08 ...
```

Once this step is finished, the FlowSOM object can be used for visualization. You can plot the nodes in several layouts ("MST": Minimal spanning tree (default),"grid": SOM grid, "matrix": alternative layout. You can make use of `PlotFlowSOM` or you can use the `PlotStars` function.

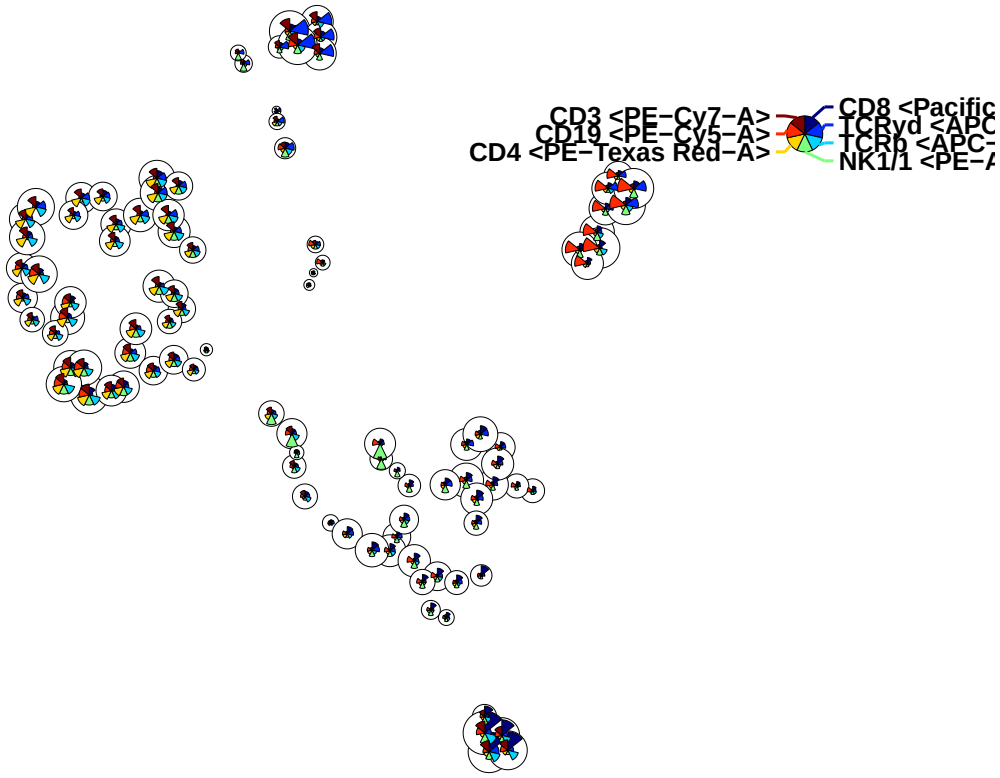
```
> PlotStars(fSOM)
```




```
> PlotStars(fSOM, view = "grid")
```

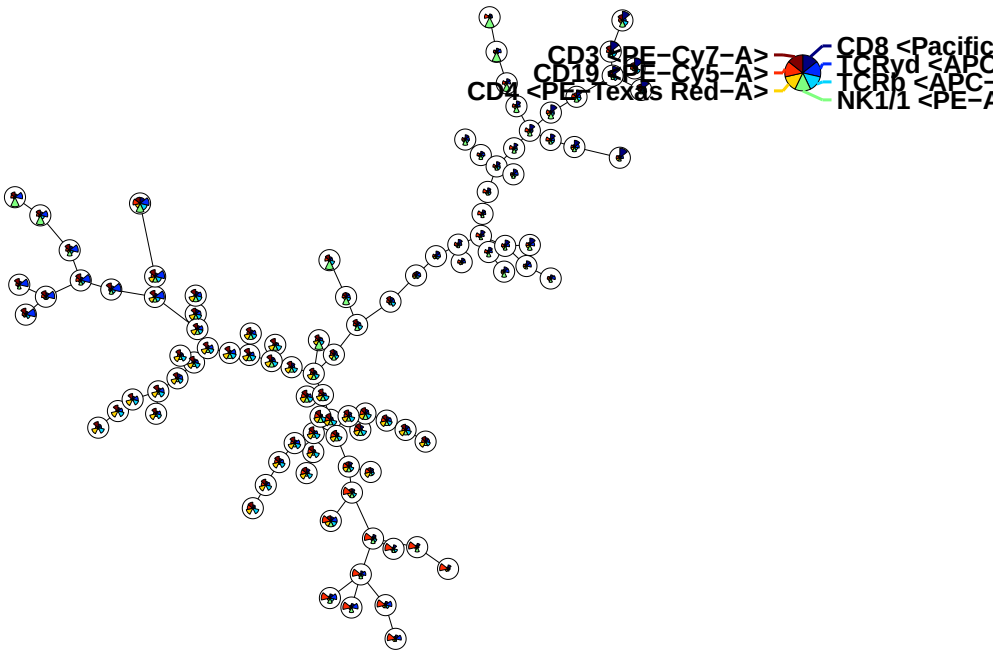



```
> set.seed(1)
> tsne <- Rtsne::Rtsne(fSOM$map$codes, perplexity = 6)
> PlotStars(fSOM, view = tsne$Y, maxNodeSize = 2)
```



If you do not want the size to depend on the number of cells assigned to a node, you can use the parameter `equalNodeSize = T` in every plotting function.

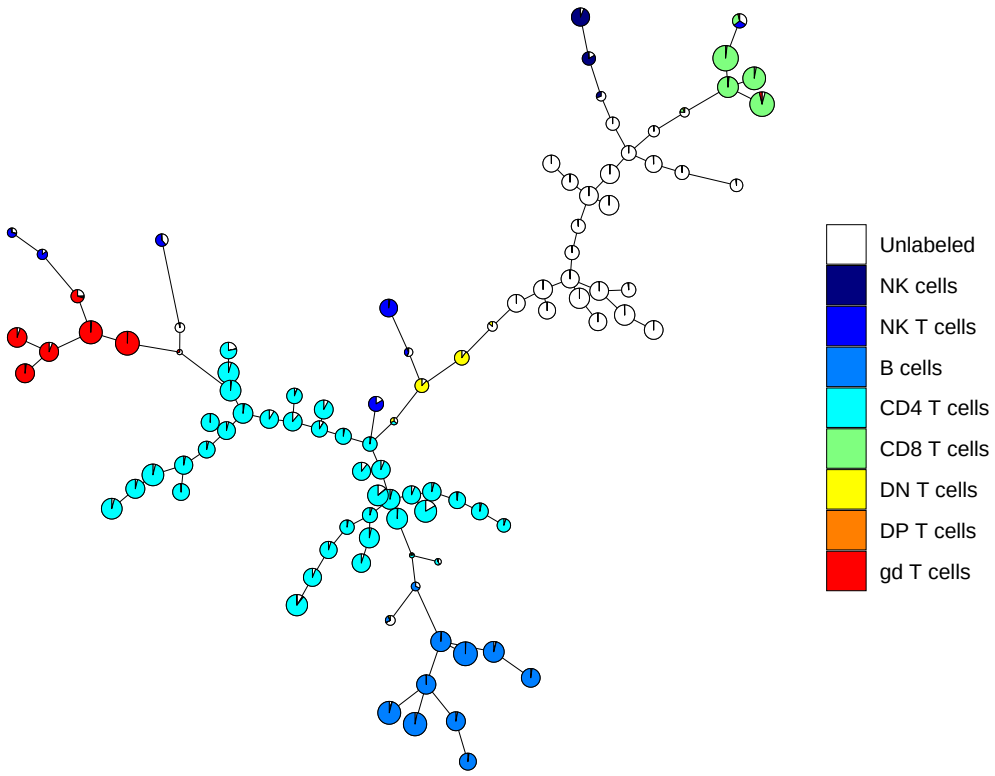
```
> p <- PlotStars(fSOM, equalNodeSize = TRUE)
> print(p, newpage = FALSE)
```



It might also be interesting to compare with a manual gating. The `cellTypes` can be any factor which has a value for each individual cell, so you can also map other clustering results.

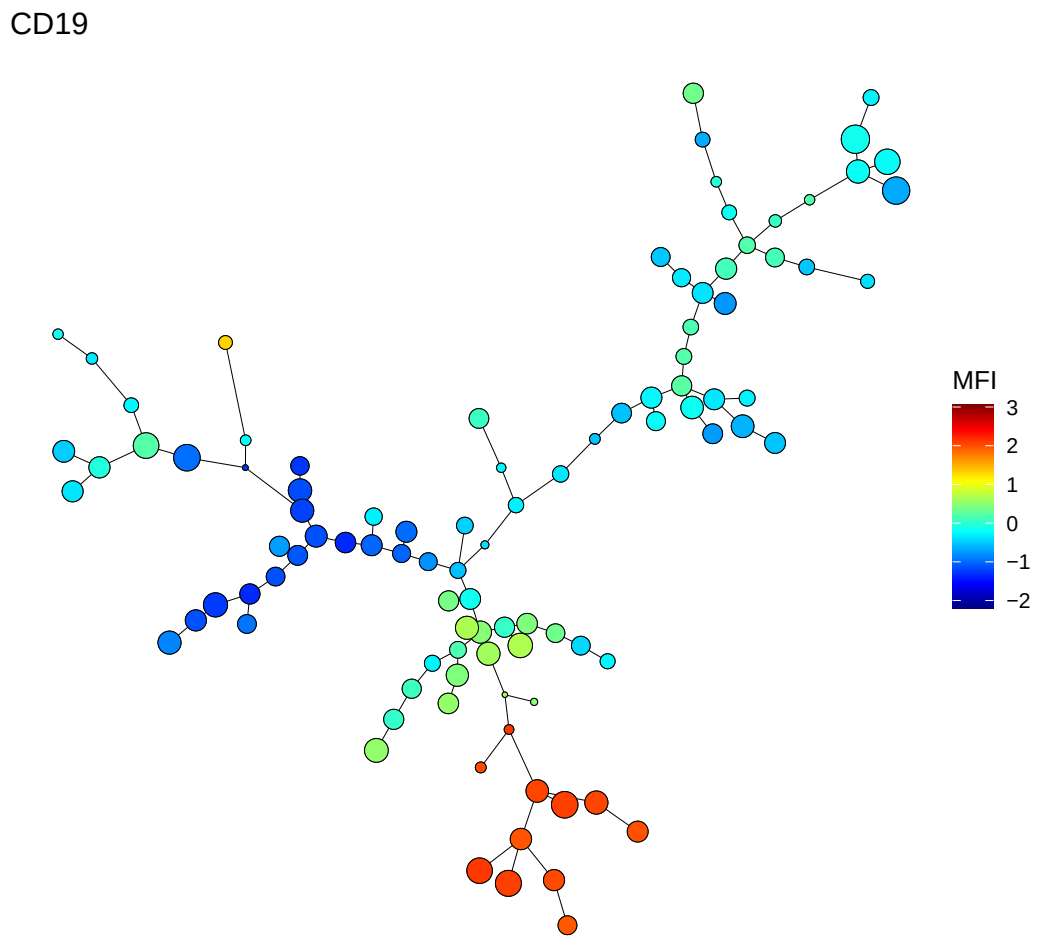
```
> wsp_file <- system.file("extdata", "gating.wsp", package = "FlowSOM")
> # Specify the cell types of interest for assigning one label per cell
> cell_types <- c("B cells",
+                 "gd T cells", "CD4 T cells", "CD8 T cells",
+                 "NK cells","NK T cells")
> # Parse the FlowJo workspace
> gatingResult <- GetFlowJoLabels(fileName,
+                                 wsp_file,
+                                 cell_types = cell_types)

[1] "Processing /tmp/RtmpMfSOOG/Rinst31b1a512f66889/FlowSOM/extdata/68983.fcs"
```

```
> PlotPies(fSOM, cellTypes = gatingResult$manual)
```



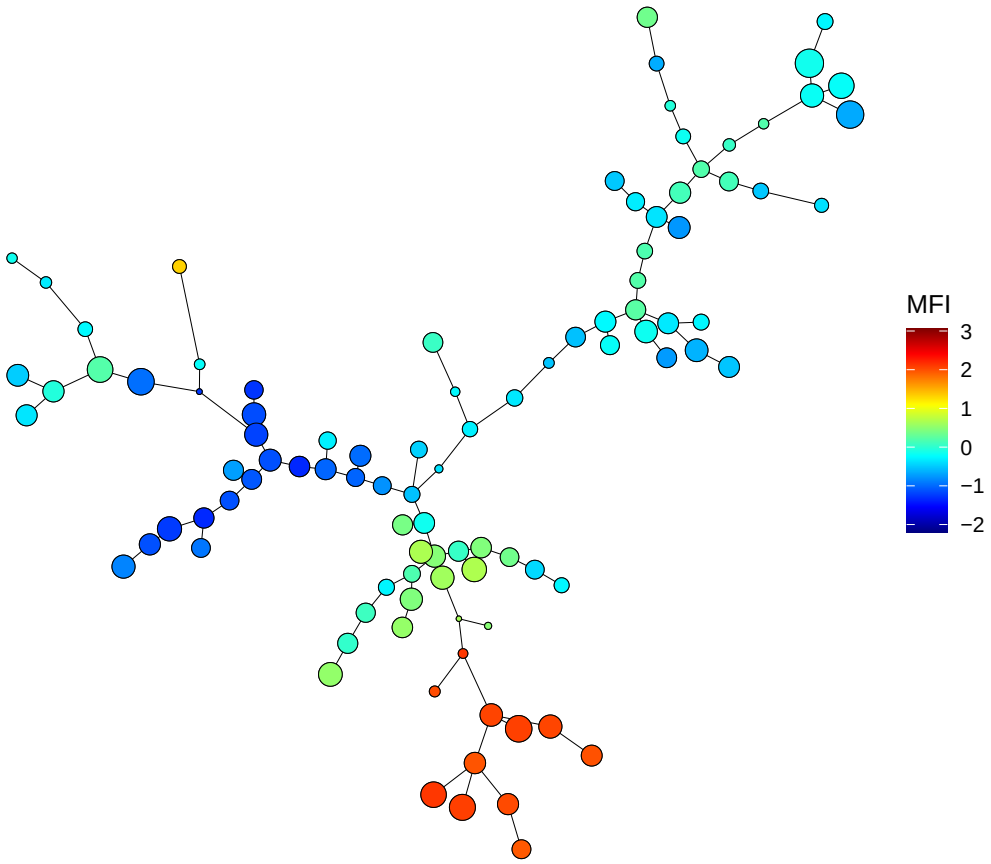
If you are interested in one specific marker, you can use the `PlotMarker` function.

```
> p <- PlotMarker(fSOM, "PE-Cy5-A")
> print(p, newpage = FALSE)
```



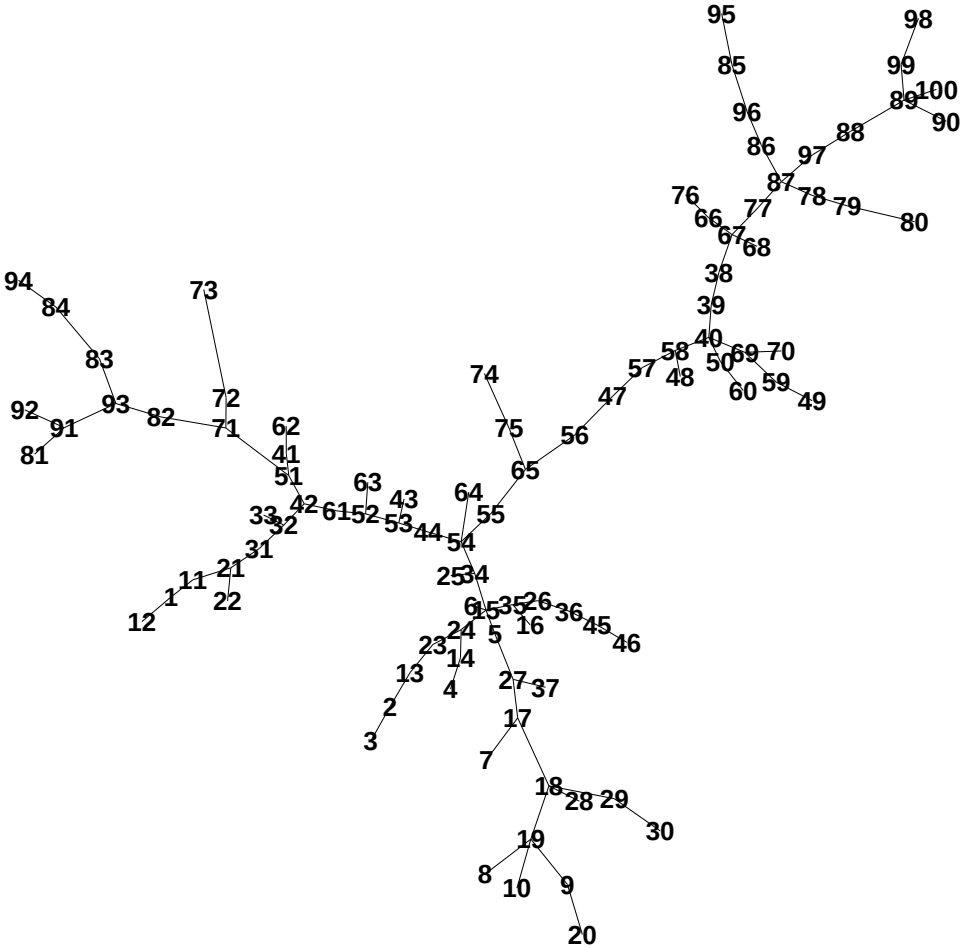

```
> p <- PlotMarker(fSOM, "CD19")
> print(p, newpage = FALSE)
```

CD19



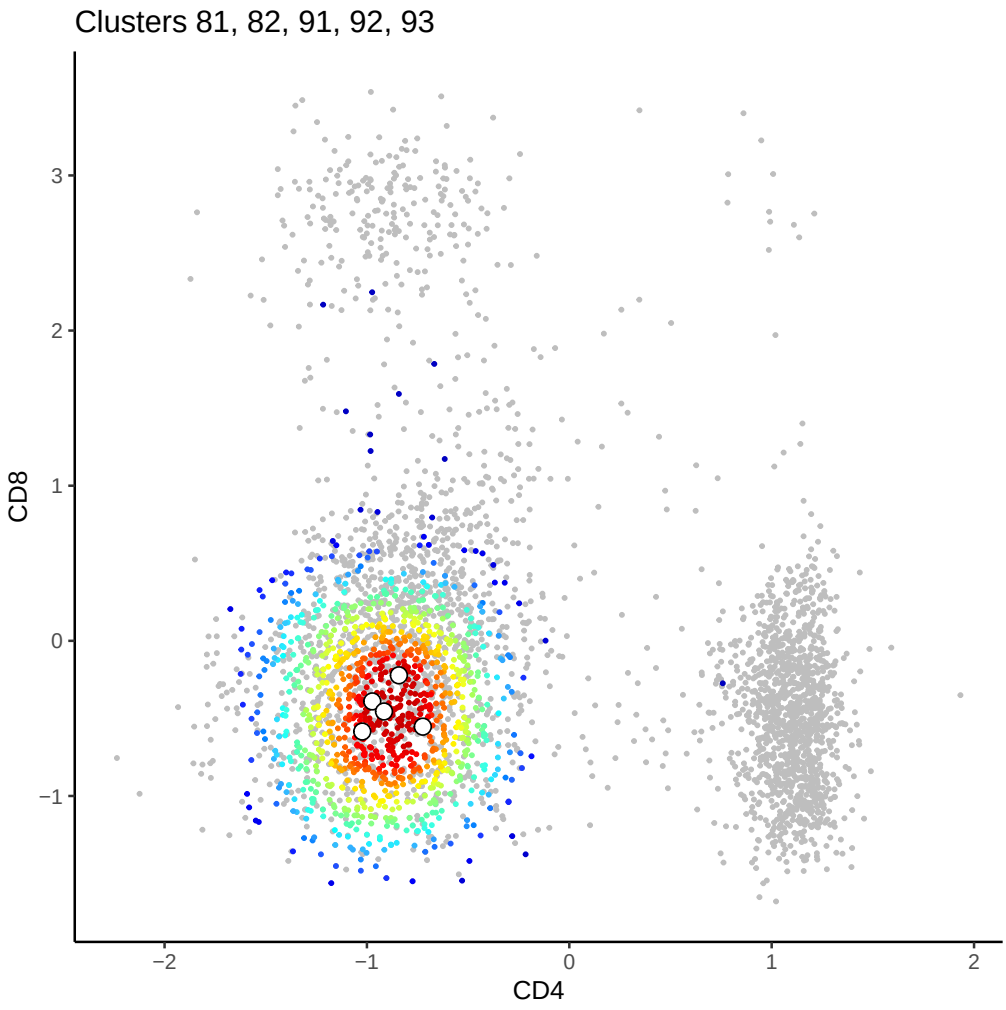
If you need to refer to the nodes, it might be useful to number them.

```
> PlotNumbers(fSOM)
```



You can use this number for a 2D scatter plot indicating the node values.

```
> plot <- Plot2DScatters(fSOM,
+                        channelpairs = list(c("PE-Texas Red-A", "Pacific Blue-A")),
+                        clusters = list(c(81, 82, 91, 92, 93)),
+                        plotFile = NULL)
> print(plot[[1]])
```



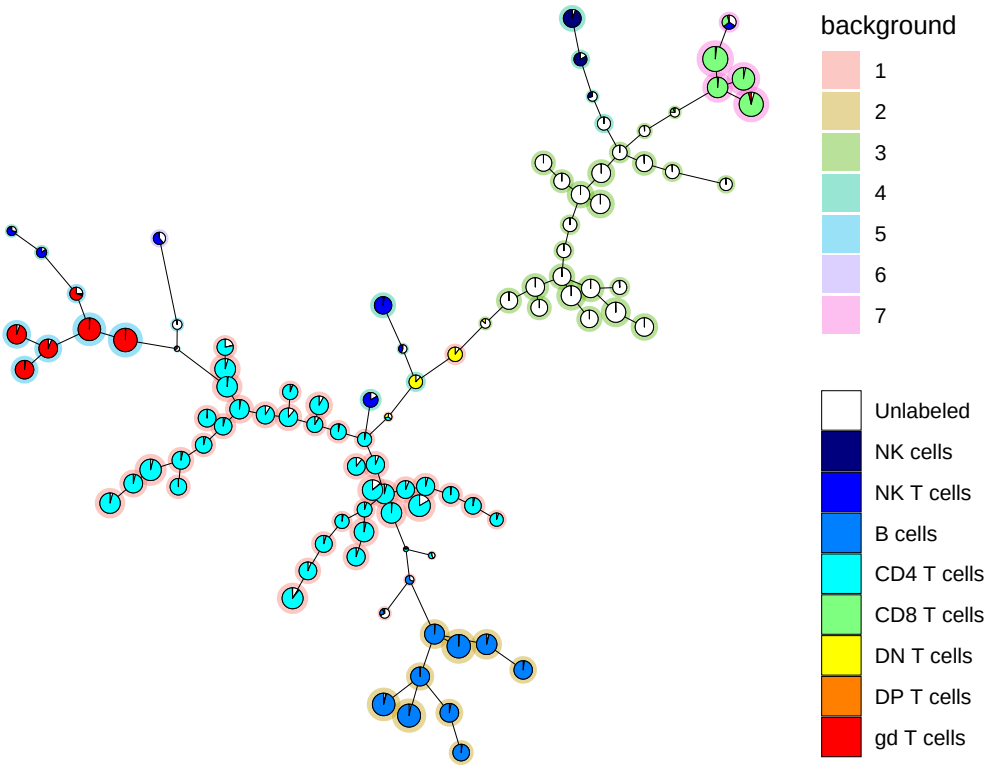
7 Meta-clustering the data

The fourth step of the FlowSOM algorithm is to perform a meta-clustering of the data. This can be the first step in further analysis of the data, and often gives a good approximation of manual gating results.

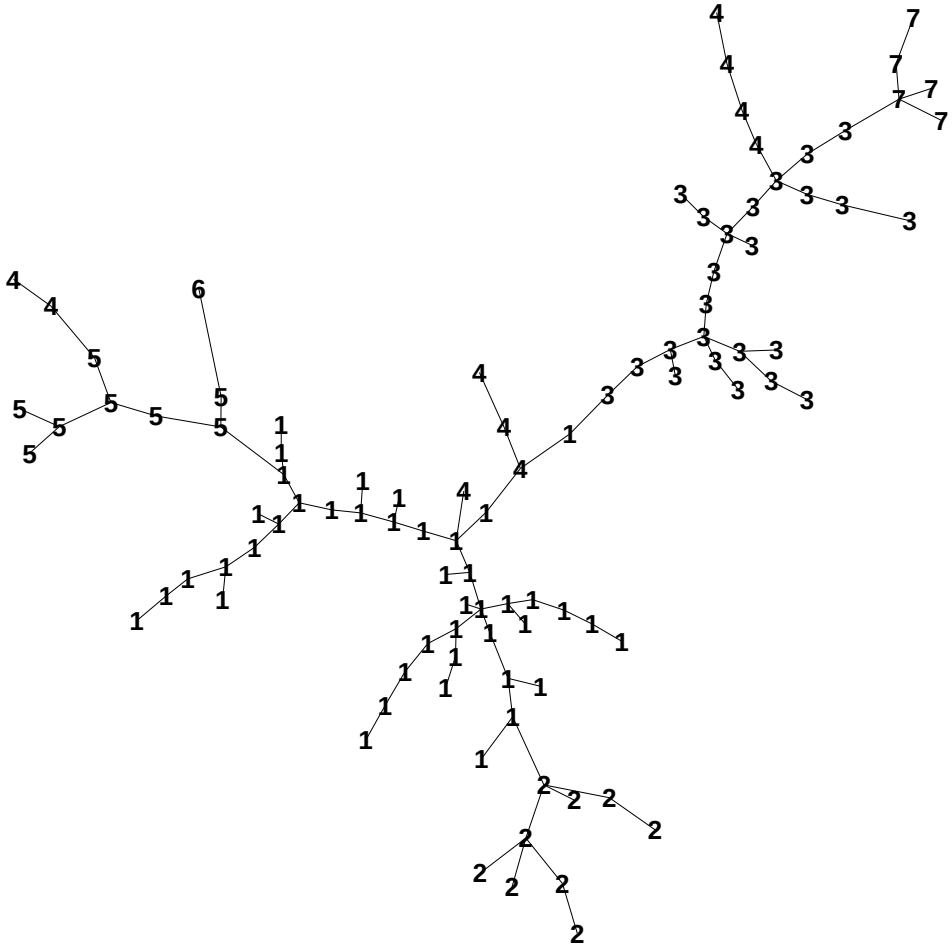
If you have background knowledge about the number of cell types you are looking for, it might be optimal to provide this number to the algorithm.

```
> metaClustering <- as.character(metaClustering_consensus(fSOM$map$codes,k = 7))
```

```
> PlotPies(fSOM,
+          cellTypes=gatingResult$manual,
+          backgroundValues = metaClustering)
```




```
> PlotLabels(fSOM, labels = metaClustering)
```



You can also extract the meta-clustering for each cell individually

```
> metaClustering_perCell <- GetMetaclusters(fSOM, metaClustering)
> table(metaClustering_perCell)

metaClustering_perCell
   1    2    3    4    5    6    7
8248 2460 4170 1123 1589   98 1537
```

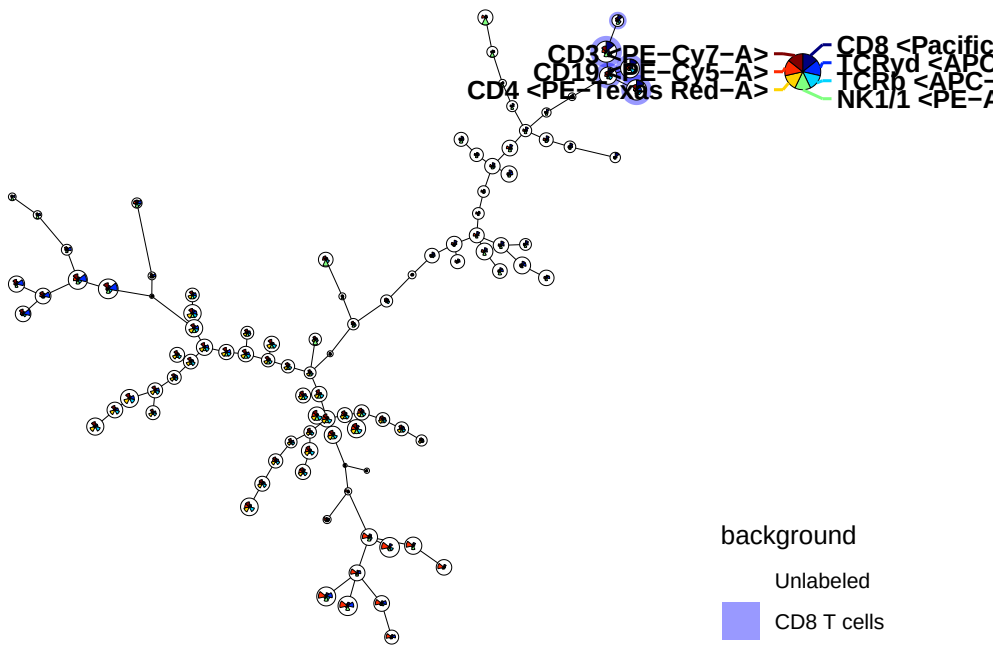
8 Detecting nodes with a specific pattern

If you do not have a manual gating to map on the tree, it might be time- consuming to interpret all the different nodes. Therefore, you can also query the tree to indicate nodes similar to a specified pattern. This function is still being optimized, so make sure to check the marker values to see if it corresponds to your expectations.

```
> # Look for CD8+ ab T cells
> query <- c("PE-Cy7-A" = "high", #CD3
+            "APC-Cy7-A" = "high", #TCRb
```

```
+             "Pacific Blue-A" = "high") #CD8
> query_res <- QueryStarPlot(fSOM, query, equalNodeSize = TRUE, plot = FALSE)
> cellTypes <- factor(rep("Unlabeled", fSOM$map$nNodes),
+                     levels=c("Unlabeled", "CD8 T cells"))
> cellTypes[query_res$selected] <- "CD8 T cells"
```

```
> p <- PlotStars(fSOM, backgroundValues=cellTypes,
+                backgroundColor=c("#FFFFFF00","#0000FF"))
> print(p, newpage = FALSE)
```



9 Comparing different groups

It is possible to compare between groups with the FlowSOM package as well. The tree should be build on either a concatenation of all files, or a representative subset of all cell types. Then a list identifying which files belong to specific groups should be defined, and the differences will be computed. For a smaller number of samples, you can look at the fold change between the groups. For coloring, a treshold is used. A treshold of 0.50 means the difference should

be at least 50% of the max of both groups, which corresponds with a 2-fold change. The higher the threshold, the stricter, a threshold of 0 will colour each node. For a larger number of samples you can also use a wilcox test. This will be selected when a value is provided for the p_thresh parameter.

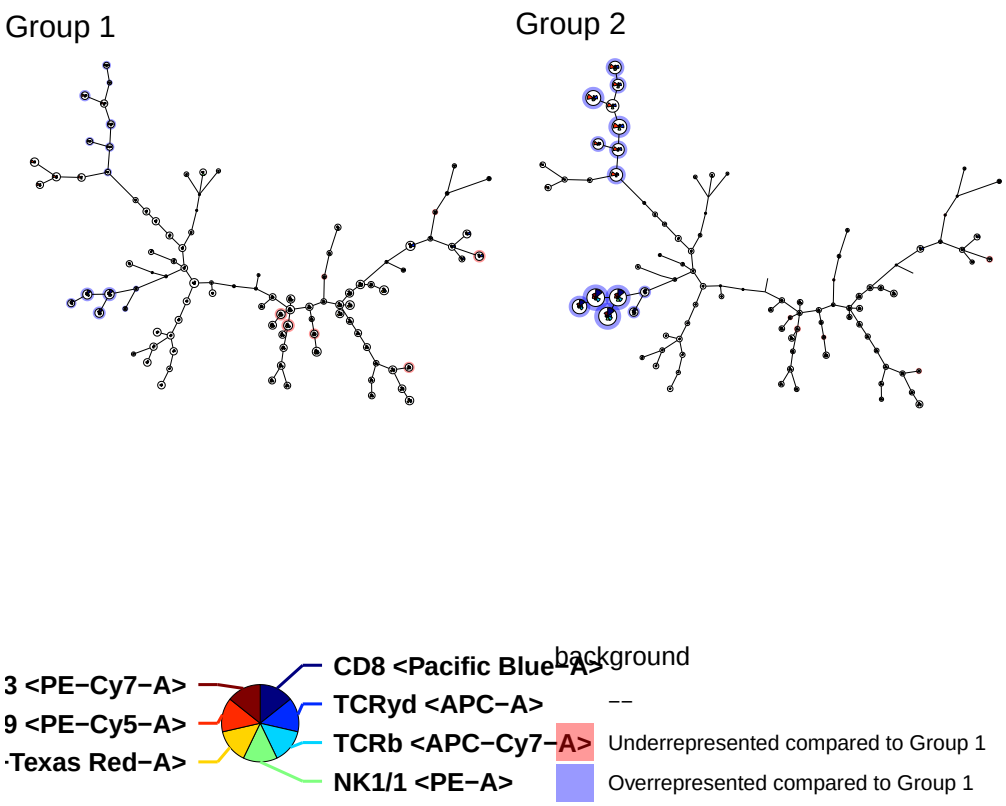
```
> library(FlowSOM)
> set.seed(1)
> # Build FlowSom result
>  fileName <- system.file("extdata", "68983.fcs", package = "FlowSOM")
>  ff <- flowCore::read.FCS(fileName)
>  ff <- flowCore::compensate(ff, ff@description$SPILL)
>  ff <- flowCore::transform(ff,
+           flowCore::transformList(colnames(ff@description$SPILL),
+                                   flowCore::logicleTransform()))
>  flowSOM.res <- FlowSOM(ff, scale = TRUE, colsToUse = c(9, 12, 14:18),
+                          nClus = 10)
> # 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
>  for(i in 1:5){
+    flowCore::write.FCS(ff[sample(nrow(ff), 1000), ],
+                        file = paste0("ff_tmp", i, ".fcs"))
+  }
>  for(i in 6:10){
+     flowCore::write.FCS(ff[c(sample(nrow(ff),500),
+                          sample(which(GetMetaclusters(flowSOM.res) == 1), 250),
+                          sample(which(GetMetaclusters(flowSOM.res) == 9), 250)), ],
+                      file = paste0("ff_tmp", i, ".fcs"))
+  }
> # Get the count matrix
>  percentages <- GetFeatures(fsom = flowSOM.res,
+                             files = paste0("ff_tmp",1:10,".fcs"),
+                             type = "percentages")
```

```
Mapping file 1 of 10.
Mapping file 2 of 10.
Mapping file 3 of 10.
Mapping file 4 of 10.
Mapping file 5 of 10.
Mapping file 6 of 10.
Mapping file 7 of 10.
Mapping file 8 of 10.
Mapping file 9 of 10.
Mapping file 10 of 10.
```

```
> # Perform the statistics
> groups <- list("Group 1" = paste0("ff_tmp", 1:5, ".fcs"),
+                "Group 2" = paste0("ff_tmp", 6:10, ".fcs"))
> MC_stats <- GroupStats(percentages[["metacluster_percentages"]], groups)
> C_stats <- GroupStats(percentages[["cluster_percentages"]], groups)
> # Process the fold changes vector
```

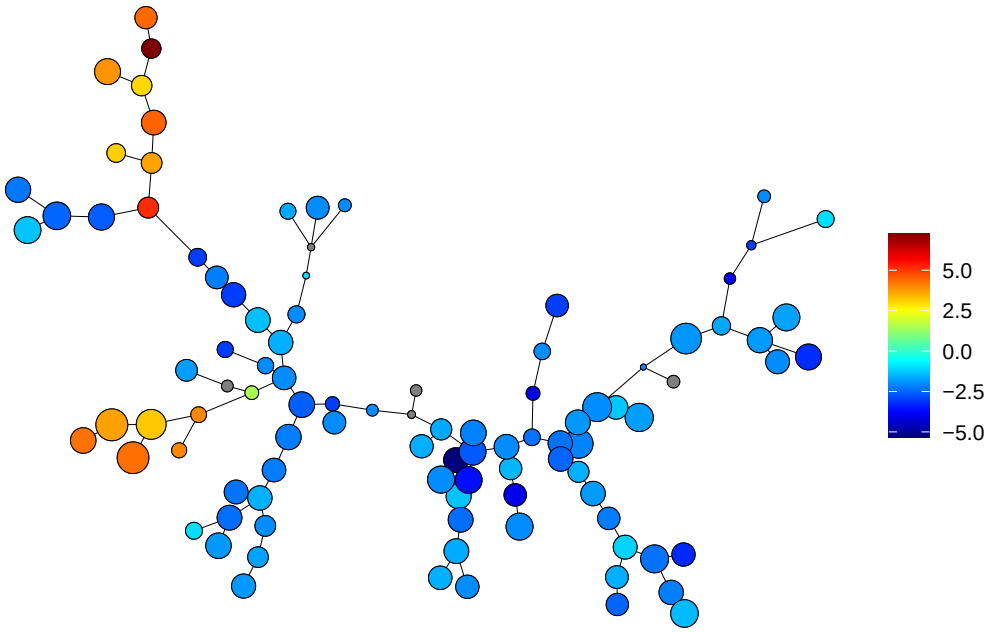
```
> fold_changes <- C_stats["fold changes", ]
> fold_changes <- factor(ifelse(fold_changes < -3, "Underrepresented compared to Group 1",
+                        ifelse(fold_changes > 3, "Overrepresented compared to Group 1",
+                        "--")), levels = c("--", "Underrepresented compared 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", ],
+                   refNodeSize = max(C_stats[c("medians Group 1", "medians Group 2"),]),
+                   backgroundValues = fold_changes,
+                   backgroundColors = c("white", "red", "blue"),
+                   list_insteadof_ggarrange = TRUE)
> gr_2 <- PlotStars(flowSOM.res, title = "Group 2",
+                   nodeSizes = C_stats["medians Group 2", ],
+                   refNodeSize = max(C_stats[c("medians Group 1", "medians Group 2"),]),
+                   backgroundValues = fold_changes,
+                   backgroundColors = c("white", "red", "blue"),
+                   list_insteadof_ggarrange = TRUE)
> p <- ggpubr::ggarrange(plotlist = list(gr_1$tree, gr_2$tree,
+                                        gr_2$starLegend, gr_2$backgroundLegend),
+                        ncol = 2, nrow = 2,
+                        heights = c(4, 1))
> print(p, newpage = FALSE)
```



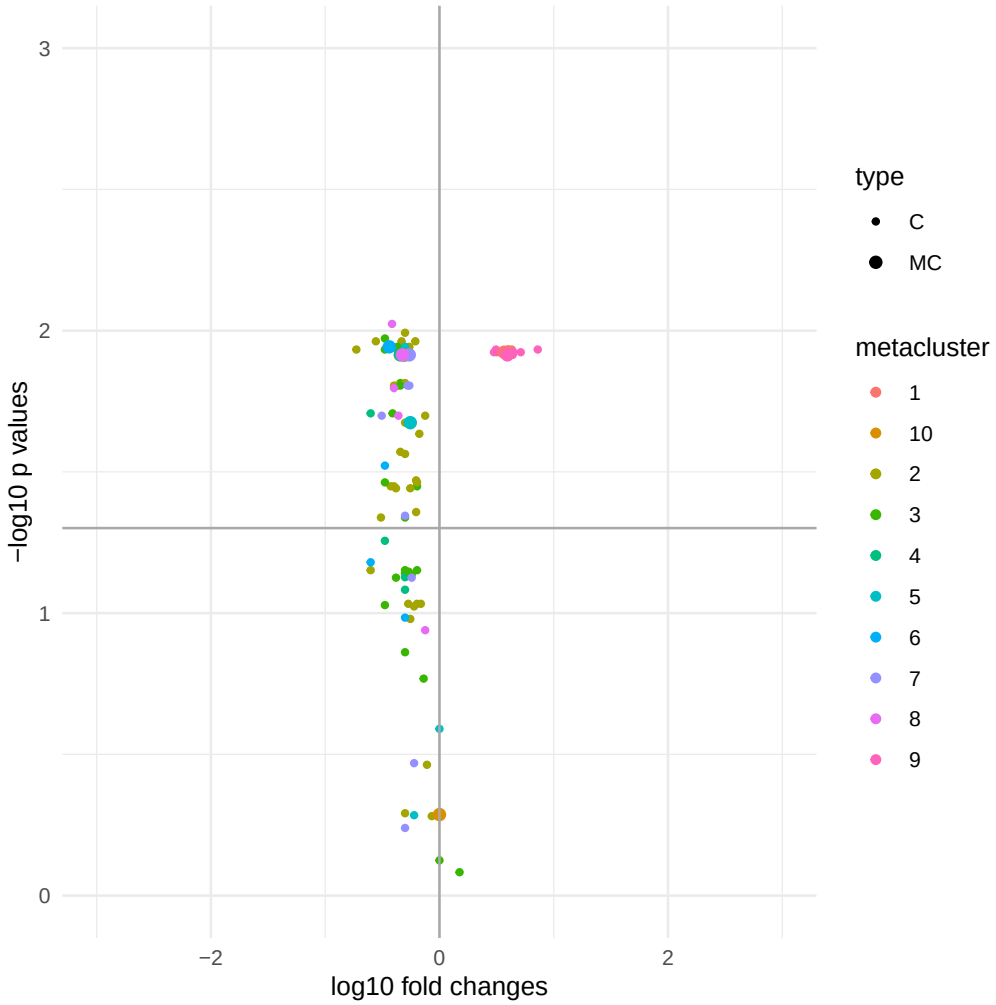

```
> ## p values
> p <- PlotVariable(flowSOM.res,
+                   title = "Fold change group 1 vs. group 2",
+                   variable = C_stats["fold changes", ])
> print(p, newpage = FALSE)
```

Fold change group 1 vs. group 2




```
> ## 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, ]),
+                                 "feature" = c(colnames(C_stats), colnames(MC_stats)),
+                                 "metacluster" = c(as.character(flowSOM.res$metaclustering),
+                                                   levels(flowSOM.res$metaclustering)),
+                                 "type" = c(rep("C", ncol(C_stats)),
+                                            rep("MC", ncol(MC_stats))),
+                                 check.names = FALSE),
+                      ggplot2::aes(x = `log10 fold changes`,
+                                   y = `-log10 p values`,
+                                   size = type,
+                                   col = metacluster)) +
+   ggplot2::xlim(-3, 3) +
+   ggplot2::ylim(0, 3) +
+   ggplot2::geom_point() +
+   ggplot2::theme_minimal() +
```

```
+   ggplot2::geom_vline(xintercept = 0, col = "darkgrey") +
+   ggplot2::geom_hline(yintercept = -log10(0.05), col = "darkgrey") +
+   ggplot2::scale_size_manual(values =c("C" = 1, "MC" = 2))
> print(p, newpage = FALSE)
```



10 Summary

In summary, the FlowSOM package provides some new ways to look at cytometry data. It can help to keep an overview of how all markers are behaving on different cell types, and to reduce the probability of overlooking interesting things that are present in the data.