CytoSOM
User Guide

V1 02 april 2020

Here is the guide introduction for the easy implementation and use of the CytoSOM package in RStudio. This manual is addressed to both bioinformaticians and novice users that want to extract information from flow cytometry experiments.

Table des matières

Before beginning

The CytoSOM analysis requires the installation on your computer of the following softwares:

- R
- RStudio
- FowJo

Then, the full analyse of your data requires three steps:

- Step 1: Load your raw data
- Step 2: Build your tree
- Step 3: Export your results

Each step contains the description of the functions required to move forward, that need to be executed in a strict consecutive fashion. Each command in this user guide will be written like this:

```
Example of command you need to write in RStudio
```

Pre-step

1. Gate your population of interest on FlowJo

The very first step of your analysis is to select your population of interest in FlowJo. After opening FlowJo and loading your FCS files, you have to gate the cells that will be taken by CytoSOM for a deeper analysis. For instance, you can gate by removing debris and doublets, and then selecting living cells. You can also be more stringent and already select a subclass like leukocytes (CD45+) or lymphocytes (CD3+).

2. Organize your data for CytoSOM processing

Before launching the analysis, you need to prepare your working directory folder that should contain:

- Your FCS files contained in a subsequent folder (corresponds to “dirFCS”)
- Your FlowJo workspace where your gated your population of interest
- Your table giving the corresponding treatment of each FCS file (see Step 1-4)

Step 1: Prepare and load your raw data

Now that your data are organised in your working directory, you can open RStudio and follow the procedure. The first step of the CytoSOM analysis is to implement all the data required for their following process.

1. Load the CytoSOM package through the *library* function

First, you need to install the CytoSOM package by writing the following command:

```
install.packages("devtools")

devtools::install_github(“gautierstoll/CytoSOM”)
```

This step is needed only once, as all the functions required for the CytoSOM analysis will be loaded on your computer. For further analysis you would need to bring functions inside RStudio by the command:

```
Library(CytoSOM)
```

2. Choose your working directory through *setwd* function

Once the functions contained in your package are loaded in RStudio, you need to give the working directory where you prepared your data in the pre-step. This is done by the common function:

```
setwd("path to your folder")
```

To find the path of your working directory, you can open it and click on the title bar. The complete path to your working directory will be written here.

Example:

```
setwd("/Users/Gautier/Experiments/Experiment1/Experiment1_CytoSOM/")
```

3. Load your FCS files through the *DownLoadCytoData* function

Next, you need to give the FCS files that will be read through the *DownLoadCytoData* function that will then extract your population of interest:

```
DownLoadCytoData(dirFCS="", gatingName="", fcsPattern="", compensate="")
```

You need to replace the arguments by yours informations:

-**dirFCS** = "The exact name of the folder that encloses your FCS files"

-**gatingName** = "The exact name of your population of interest gated in FlowJo"

Your population of interest is the population that will be taken for the subsequent analysis of your data and the build of the CytoSOM tree. Its name must be the exact name given in the FlowJo workspace.

-**fcsPattern** = "a specific annotation of the fcs files"

A pattern can be necessary if you want to keep all your FCS files enclosed in the same folder but you want just a part of them to be included in the CytoSOM analysis.

-**compensate** = FALSE or TRUE

Example:

```
CytoData=DownLoadCytoData(dirFCS="FCS files", gatingName="CD45+",
fcsPattern="tube", compendate=FALSE)
```

4. Load the corresponding treatment of each FCS files by the *read.csv* function

Finally, in order to compare the impact of experimental conditions on the population of the metaclusters further generated, you need to tell CytoSOM which fcs files corresponds to which of your treatment. This is done by loading in RStudio a csv file through the read.csv function:

```
read.csv("Name of your table",sep=";")
```

The table should contain 3 columns:

- One column entitled "files" where the exact name of your FCS files are written
- One column entitled "Treatment" containing the corresponding experimental condition for each fcs file
- One optional column entitled "NormalizationFactor" that gives a factor of normalization to be applied for each fcs file for the statistical analysis (*Step 3.2*)

```
tableTreatmentFCS=read.csv("Experiment1 treatments",sep=";")
```

Step 2: Build your tree

1. Build your Tree through BuildFSOMTree function

This phase is the core of the analysis and is based on the already published flowSOM algorithm. It consists of the generation of a tree that results from a clusterization of each cell extracted from population of interest gated in the flowJo workspace loaded.

```
buildFSOMTree(fSOMDloaded,prettyNames,clustDim,metaClNb, fSOMSeed)
```

-**fSOMDloaded** = the object generated by the DownLoadCytoData function

-**prettyNames** = the name of the markers you want to take into account for the flowSOM clusterization.

Example: c("Marker 1","Marker2","Marker3")

-**clustDim** = dimension of the grif for the clusterization

-**metaClNb** = number of metaclusters choosen

-**fSOMSeed** = number that will fix the way the data are analysed. This ensures that with the same raw data and the same parameters that you choose, CytoSOM will give you the exact results.

Example:

```
CytoTree=buildFSOMTree(CytoData, c("CD8","CD4","FoxP3"),4,7,1)
```

2. Remove the smallest clusters through PlotStarsMSTRm function

Once your tree is built, you may need to remove small clusters that prevent an appropriate visualization of the resulting clustering and metaclustering. Thanks to the PlotStarsMSTRm function, you can choose to remove a define number of the smallest clusters, thanks to the command:

```
PlotStarsMSTRm(fSOMObject,metaClustFactors,mainTitle,nbRm)
```

-**fSOMObject** = the object generated by the function buildFSOMTree

-metaClustFactors =

-**mainTitle** = “The title you want to give to your generated tree”

-**nbRm** = number of the smallest clusters to remove

Example:

```
PlotStarsMSTRm(fSOMObject=CytoTree$fSOMTree,metaClustFactors=CytoTree$
metaCl,mainTitle=”Experiment1”,nbRm=0)
```

3. Rename your resulting metaclusters through *TreeMetaRenaming* function

As the number of metaclusters generated can prevent their efficient visualisation in the legend of the trees plots, a naming through their expression in specific markers can be helpful. This is realized through the function:

```
TreeMetaRenaming (TreeMetaCl,Markers,NameType)
```

-**TreeMetacl** = object generated by the buildFSOMTree function

-**Markers** = vector of the markers to include for the renaming of the metaclusters

-**NameType** = “robustName”, “nonRobustName”, “shortRobustName” or “number”

Example:

```
CytoTree=TreeMetaRenaming(TreeMetaCl=CytoTree,
Markers=c(“CD11b”,”F4/80”,”CD4”,”CD8”), NameType=”shortRobustName”
```

Step 3: Export your results

After your tree is generated, the final step is to export in PDF format all your generated results. They include:

-the tree build in step 2-1, and its subsequent visualization according to the conditions given in the table in step 1-4

-the boxplots comparing the percentage or the absolute number of each experimental conditions in each metacluster

-the boxplots comparing, for a choosen marker, its MFI for each experimental condition in each metacluster

1. Export PDF files containing CytoSOM trees through plotTreeSet function

In order to get the tree generated through the buildFSOMtree function, plotTreeSet command automatically save a PDF file in your working directory (set in step 1-2) that encompasses:

- the general tree generated through the buildFSOMTree function
- one tree representation for each experimental condition, where the size of the clusters are proportional to the percentage size of the population

The function also creates another PDF file that plot the median fluorescence intensity (MFI) of a specified marker on the general tree. There will be as many PDF files as markers chosen. One file contains:

- the MFI of the specified marker on the general tree
- the MFI of the specified marker for each experimental condition

```
plotTreeSet(TreeMetacl,markers,Title,rmClNb,treatmentTable,globalScale=TRUE)
```

-**TreeMetacl** = object generated by the buildFSOMTree function

-**markers** = vector of markers for which MFI in each metacluster will be illustrated on the tree thanks to a color scale

-**Title** = “title for the PDF file generated”

-**rmClNb** = number of the smallest clusters to remove

-**treatmentTable** = table in csv format that contains the treatment related to each FCS file (already loaded in step 1-4)

-**globalScale** = fix a common color scale for all the experimental conditions specific trees if TRUE.

Example:

```
plotTreeSet(TreeMetacl=CytoTree, markers=c(“PD-1”,”CTLA-4”),
rmClNb=1,treatmentTable=TreatmentsExp1,globalScale=TRUE)
```

2. Export PDF files containing the comparison of your treatments for each metaclusters through the *BoxPlotMetaClust* function

The BoxPlotMetaClust function automatically saves a PDF file in your working directory (set in step 1-2) that compares the impact of your(s) experimental condition(s) to a chosen control for each metacluster. The PDF file includes those comparisons through several representations:

- The boxplots comparing the population of your(s) experimental condition(s) inside each metacluster, followed by their representation including statistical analysis
- One heatmap illustrating the impact of your experimental condition in comparison to a chosen control, with the optional addition of its clustered representation
- One heatmap that represents the p-value of the pair-wise comparison, also with the optional inclusion of its clusterization illustration

```
BoxPlotMetaClust(TreeMetaCl, Title, treatmentTable, ControlTreatment,
BottomMargin, yLab, Norm, Robust,ClustHeat)
```

-**TreeMetaCl** = the object generated by the buildFSOMTree function

-**Title** = “title of the PDF file generated”

-**treatmentTable** = table containing the experimental conditions

-**ControlTreatment** = “experimental condition choosen as the control”

-**BottomMargin** = value of the graphical space let for the name of the experimental condition in the bottom of the boxplots

-**yLab** = “Labels of the ordinate”

-**Robust** = dunn test if TRUE or Tukey test if FALSE

-**ClustHeat** = Hierarchical clustering (TRUE or FALSE)

Example:

```
BoxPlotMetaClust(CytoTree, “Experiment 1”, TreatmentsExp1,
ControlTreatment=”Water”, BottomMargin=3, yLab = “CD3+”,
Norm=FALSE,Robust=, ClustHeat=TRUE)
```

3. Export PDF files containing the statistics relative to the MFI of choosen markers for each metacluster through the *BoxPlotMarkerMetaclust* function

In order to compare the expression of specified markers by your(s) experimental condition(s) inside each metacluster, you can display the BoxPlotMarkerMetaclust function. For one selected marker, it saves a PDF file in your working directory (set in step 1-2) containing:

- Boxplots for each metacluster comparing the marker’s MFI by experimental condition, and their representation with the supplementary graphical statistical analyses
- Heatmaps comparing, for each experimental condition, the selected marker’s MFI inside each metacluster, with its clusterization optional representation
- Heatmaps with the pair-wise comparison of the selected marker’s MFI for each metacluster, with the optional illustration of its clusterization

```
BoxPlotMarkerMetaClust(TreeMetaCl,Title,treatmentTable,ControlTreatment,BottomMargin,Marker,Robust,ClustHeat)
```

-**TreeMetaCl** = object generated by BuildFSOMTree function

-**Title** = title of your PDF file

-**treatmentTable** = table containing the condition for each corresponding FCS file

-**ControlTreatment** = Condition you consider as your control for statistics purpose

-**BottomMargin** = value of the graphical space let for the name of the experimental condition in the bottom of the boxplots

-**Marker** = Markers for which this function will be applied

-**Robust** = dunn test if TRUE or Tukey test if FALSE

-**ClustHeat** = Hierarchical clustering (TRUE or FALSE)

Example:

```
BoxPlotMarkerMetaClust(TreeMetaCl=CytoTree, Title="Experiment 1", ControlTreatment=Control, BottomMargin=3, Markers=c("PD-1", "CTLA-4","PD-L1"), Robust=TRUE, ClustHeat=TRUE)
```