CytoSOM
User Guide

Version 0.1.4

24th april 2020

Here is the complete guide 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. The compact description of the workflow and its associated functions are available in the "README" section at the following address "https://github.com/gautierstoll/CytoSOM".

In case of issue with the execution of the script or for any question about CytoSOM, you can send an email to: **cytosom.package@gmail.com**.

Before beginning

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

- R
- RStudio
- FowJo

In order to avoid troubles with CytoSOM execution, we recommend to always work with the more recent version of R.

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 function will be described as this:

Function description

And each function will be followed by an example of a command that will be written like this:

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

Pre-step: install the required packages and organize your data for CytoSOM processing

I. Install the packages necessary for the CytoSOM analysis

The CytoSOM process requires functions that have been developed by several others bioinformaticians. Those attached functions are contained in packages that need to be installed before CytoSOM installation.

1. First packages to install

Before CytoSOM installation, you have to install packages that are available on Bioconductor. For this, you have to open RStudio and write:

```
if (!requireNamespace("BiocManager", quietly = TRUE))

install.packages("BiocManager")

BiocManager::install("Name of the package")
```

This step has to be repeated for each of the following package: CytoML, flowCore, flowSOM and flowWorkspace.

2. CytoSOM installation on Github

Then, you need to install the CytoSOM package that is available on Github, by writing the following commands:

```
install.packages("devtools")

devtools::install_github("gautierstoll/CytoSOM")
```

Those installation steps are needed only once, as all the functions required for the CytoSOM analysis will be installed on your computer.

II. Organize your data

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”)
- One FlowJo workspace where your gated your population of interest (see Step 1-III-A-1). A manual gating on RStudio is also available and allow you to perform the analysis without FlowJo software (see step 1-III-B).
- Your table giving the corresponding treatment of each FCS file (see Step 1-IV)

Step 1: Load your raw data

Now that your data are organised in your working directory and that all the packages are installed on your computer, 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.

I. Load the CytoSOM package

Once all the packages are installed, you only need to bring functions inside RStudio by loading the CytoSOM package through the command:

```
Library(CytoSOM)
```

Of note, the CytoSOM package may be updated since your first installation, so we recommend you to check if you get its last version and to install again both devtools and CytoSOM if not.

II. 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/")
```

Another option to set the working directory, is to click on "Session" on the top of RStudio interface then "Set Working Directory", "Choose directory" and to select the appropriate repository where all your data are implemented.

III. Load and read your FCS files

When your data are well organized, CytoSOM library loaded and the working directory chosen on RStudio, the first step is to load your FCS files that will be analysed. Because your files may contain events that you don't want to be included in the CytoSOM processing (e.g dead or non-leukocytes cells), you can clean your data and select your population of interest through 2 options:

- Option A: execute a pre-gating step on FlowJo (*Step 1.3.A*) and read your files on RStudio
- Option B: perform both a manual gating of your raw data and their consequent reading directly on RStudio (*Step 1.3.B*)

A. FlowJo option

3. Gate your population of interest on 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+). Then save your workspace in your working directory already set in step 1-II.

4. Load and read your data through the *DownLoadCytoData* function

Next, you need to give the FCS files that you included in your FlowJo workspace and that will be read through the *DownLoadCytoData* function:

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"

It is important that the folder contains the same FCS files that were used for the FlowJo gating step.

-**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)
```

B. Manual gating on RStudio

Although FowJo is a well-known and common software employed by biologists to analyse cytometry data, some scientists may not have access to it (probably because a licence must be payed for its service). In order to permit the CytoSOM analysis in any case, we have developed the possibility to perform the pre-gating step directly on RStudio.

1. Load your FCS files through the *ReadInput* FlowSOM function

To load FCS files on RStudio and read the raw data they contained, you first have to apply the following function that is included in the FlowSOM package:

FlowSOM::ReadInput(input, pattern, compensate)

-**input** = "The exact name of the folder that encloses your FCS files" (same argument as "dirFCS" in *DownLoadCytoData* function)

-**pattern** = "a specific annotation of the fcs files" (same argument as "fcsPattern" in *DownLoadCytoData* function)

-compensate = FALSE or TRUE

Example:

```
RawDataExp1=FlowSOM::ReadInput(input="FCS files", pattern=
"tube",compensate=FALSE)
```

2. Gate your population of interest through *InteractivePolyGate* function

Now that your FCS files are loaded, you can select a first population by the InteractivePolyGate function, which will display a 2D gate of chosen markers where you will be able to manually gate your raw data:

InteractivePolyGate (RawData, marker1, marker2, fcsFiles, xlim=NULL, ylim=NULL, log="")

-**RawData** = object generated by the *ReadInput* function (Step1-III-B-1)

-**marker1** = "marker that will be plot in the x axis"

-**marker2** = "marker that will be plot in the y axis"

-**fcsFiles** = vector of FCS files that will be displayed in the gate.

Example: c(1-4) if you want only your fourth first files to be displayed.

-**xlim** = vector that indicates the limits of the x axis.

Example: c(0,40000)

The determination of the adequate axis limits can necessitate to repeat this step until the cells are all contained in the gate.

-**ylim** = vector that indicates the limits of the y axis.

-**log** = "name of the axis which will be displayed in logarithmic scale"

Example:

```
CellsPolygon=InteractivePolyGate(RawData=RawDataExp1, marker1 ="FSC-A",
marker2="SSC-A",fcsFiles=c(1:20), xlim=c(0,50000), ylim=c(0,50000))
```

Once this function is performed, a 2D RStudio window of your raw data appears. You can now left-click with your mouse to manually draw a polygon that will enclose the cells you want to extract. Sides of the polygon will be drawn until you right-click and stop its design by clicking to “stop”. Once you ended your polygon, the object generated by *InteractivePolyGate* function will contain the informations you gated.

If you close the 2D window before right-clicking to finish your polygon, no data will be saved.

3. Select the data contained in your first gating through *PolygonGatingRawData* function

To extract the cells included in the polygon that you designed on your raw data, you have to launch the *PolygonGatingRawData* function:

PolygonGatingRawData (RawData,Polygons,gatingName)

-**RawData** = object generated by the *ReadInput* function (Step1-III-B-1)

-**Polygons** = list of the polygons that you previously drawn.

-**gatingName** = "Name you want to give to the cells you gated"

Example:

```
Cells=PolygonGatingRawData(RawData=RawDataExp1,
Polygons=list(CellsPolygon), gatingName="Cells")
```

4. Extract the data included in a second gating through *PolygonGatingGatedData* function

The object generated by *PolygonGatingRawData* can be directly employed for the further CytoSOM analysis. But if you need to perform more than one 2D gating, you can perform again the *InteractivePolyGate* function to draw a new gate. In this case, you have to use the *PolygonGatingGatedData* function to extract the data:

PolygonGatingGatedData(GatedData,Polygons,gatingName)

-**GatedData** = object generated by the *PolygonGatingRawData* or *PolygonGatingGatedData* functions

-**Polygons** = list of the polygons drawn through the function *InteractivePolyGate* and that will be employed to extract the data

-**gatingName** = "Name you want to give to the cells you gated"

Example:

```
LivingCells=PolygonGatingGatedData(GatedData=Cells, Polygons=list(Alive),
gatingName="Alive")
```

It is to note that you can execute as many 2D gates as you desire, until you reach the population of interest that will be used for the CytoSOM tree building.

IV. 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*)

```
TreatmentsExp1=read.csv("TreatmentsExp1.csv",sep=";")
```

Step 2: Build your tree

I. 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:

- *DownLoadCytoData* function when FlowJo has been employed to gate the population of interest
- *PolyGatingRawData* or *PolyGatingGatedData* functions when you extracted manually on RStudio your population of interest

-**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 chosen

-**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)
```

II. 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 fSOMTree contained in the object generated by the function buildFSOMTree

-**metaClustFactors** = description of the metaclusters contained in the object generated by the function buildFSOMTree

-**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)
```

III. 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 chosen marker, its MFI for each experimental condition in each metacluster

I. 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 of 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"),
Title="Experiment1", rmClNb=1, treatmentTable=TreatmentsExp1,
globalScale=TRUE)
```

II. 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 chosen 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"

-**Norm** = TRUE if the "NormalizationFactor" given in the "treatmentTable" will be applied to display statistics. FALSE if experimental conditions will be compared as percentage of total metaclusters populations.

-**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)
```

III. Export PDF files containing the statistics relative to the MFI of chosen 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 the PDF file generated"

-**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** = Marker 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, Marker=c("PD-1"), Robust=TRUE,
ClustHeat=TRUE)
```

Going further: metacluster sub-clustering

After a first CytoSOM complete analysis, you may want to perform a second one in order to go deeper in the description of your data. Thus, it is possible to extract the data contained in a chosen selection of metaclusters and to directly execute the script from step 2. For this, you will need to provide the exact name of the metaclusters of interest (e.g their number or their renaming if the function *TreeMetaRenaming* has been applied) in the *DataFromMetaClust* function.

I. Find the exact name of a metacluster through the *FindMetaClustNames* function

After the execution of the *buildFSOMTree* function, the metaclusters created are given a number that is displayed in the legend. If you perform the *TreeMetaRenaming* function, that allows the renaming of all your metaclusters according to specific markers expression, the metaclusters get an extra name that is attached to their number. In this case, you can access to the exact name of your metacluster through the *FindMetaClustNames* function:

FindMetaClustNames(subName,TreeMetaCl)

subName = part of the name contained in the metacluster of interest (e.g number or marker)

TreeMetaCl = the object generated by the buildFSOMTree function

Example:

```
FindMetaClustNames("CD8", CytoTree)
```

II. Extract the data from specific metaclusters through *DataFromMetaClust* function

Depending on your data and the number of markers included for clustering, the CytoSOM analysis can render a lot of metaclusters, for which it could be useful to perform a second clustering and meta-clustering. As you may have first selected your population of interest through a 2D scatter plot gating either on FlowJo or directly on RStudio, you can extract the data contained in specific metaclusters through the *DataFromMetaClust* function:

DataFromMetaClust (FSOMData,TreeMetaCl,MetaClusters)

-**FSOMData** = the data contained in the object generated by:

- *DownLoadCytoData* function when FlowJo has been employed to gate the population of interest
- *PolyGatingRawData* or *PolyGatingGatedData* functions when you extracted manually on RStudio your population of interest

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

-**Metaclusters** = vector of the metaclusters to extract. You have to fill the numbers of the metaclusters, e.g c(4,9,13) or their exact associated renaming if the function TreeMetaRenaming has been applied (step2-III), e.g c("4_CD8+","9_CD4+","13_CD4+FoxP3-").

Example:

```
Metacluster9=DataFromMetaClust(FSOMData=CytoData$fSOMData,
TreeMetaCl=CytoTree, Metaclusters=c("9_CD4+"))
```

You can then employ the object generated by *DataFromMetaClust* function as an argument to launch the *buildFSOMTree* function, and perform a new round of CytoSOM analysis.