Cytofast - vignette for the Spitzer study

K.A. Stam & G. Beyrend

2018-10-01

Introduction

Cytofast is a R-package aimed at quickly visualizing and analyzing the output of Cytosplore, which is software that characterizes immune subsets from flow or mass cytometry (CyTOF) data. Cytosplore is mainly used for its fine implementation of t-sne (and HSNE) algorithm to identify and cluster distinct populations of cells. However, further downstream analysis is unsupported, such as relating these population of cells to a phenotype and/or ascribing any experimental effects. Therefore we introduce the *cytofast* package, which provides a quick visualization and quantification of clusters (or combination of clusters) playing a key role in your study. Note that is also possible to use any other clustering algorithm.

In this vignette will guide the users through the basic workflow designed around the package. As an example we will use the mass cytometry dataset generated by Spitzer (2017). Here we work with a small example to explain the functionality of the package, for the complete workfow and a more thorough analysis we refer to our peer reviewed paper (Beyrend et al. 2018).

The workflow is designed around several steps, which are addressed per section. In each section we showcase some functions, which may or may not be of your interest. A summary of the most used functions is as follows:

1. `readCytosploreFCS`: reading *.fcs* files from Cytosplore into R
2. `cytoHeatmaps`: generating two aligned heatmaps

- the upper panel represents the phenotype (based on the markers) of each cluster
- the lower panel represents the relative abundance of the clusters in respect to the samples

3. `msiPlot`: generating histograms of the signal intensity for each marker

4. `cytoBoxPlots`: creates boxplots (percentage of total given cells) and representing each cluster as a percentage of the total analyzed immune cells

5. Eventually, we propose further possible analysis by presenting results obtained with a global test comparison.

1. Creating clusters

Before running your samples through *cytofast*, the data should be clustered. We recommend using Cytosplore for this task, however it is certainly possible to use any other clustering algorithm. As an example, we will also show how data clustered by FlowSOM can be parsed for usage with cytofast.

Creating clusters in *Cytosplore*

After downloading *Cytosplore*, hosted at www.cytosplore.org, upload the FCS files by clicking on *File* then *Open FCS File(s)*. Be sure to add unique sample tag as channel when prompted and select a cofactor for hyperbolic arcsinh transformation (default is 5).

Click on *Run H-SNE* and wait for the map being generated. This step may require some time depending on the number of cells analyzed.

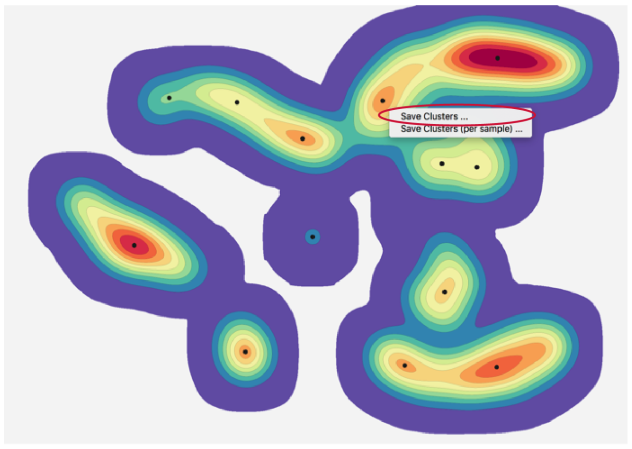


Figure 1: Saving clusters in Cytosplore

Exporting the data

Once the tSNE map is generated, we can save the clusters defined by *Cytosplore* by right-clicking on the tSNE map and save the clusters as **"Save Clusters . . . "**" (Figure 1). We can choose the directory of the output files as prompted by *Cytosplore*; make sure to note this location, because we will use this directory to load the .fcs files into R.

We advise using a simple name (characters only) when renaming the output files, this makes identification and further handling easier. After saving, we will have a folder with the created .fcs files, with each file corresponding to your identified clusters in *Cytosplore*. The next step will be loading the files into R with the help of `cytofast`.

2. Reading the data (clusters)

Let's start with loading the package.

```
library(cytofast)
```

To follow this vignette we added some pre-clustered (by Cytosplore) .fcs files to the package. These are ten downsampled clusters from the Spitzer study and should be treated as a toy-example to discover the package. A complete analysis of the Spitzer data is reported and published in *Beyrend et al., CSBJ, 2018.*

To import the *.fcs* files created by Cytosplore, we use the function `readCytosploreFCS`. Simply give the directory where you saved your .fcs files and the function will store the data in a `cfList` (cytofast list). It is a S3 object type, so easy to use and can be manipulated like any list in *R*. When not using Cytosplore, check the function `cfList` to create such a list yourself. The main components of a `cfList` are a data frame `expr` containing the expression of the markers for all measured cells, `counts` a data frame with cell counts per cluster per sample and `samples` a data frame with all the meta information.

```
## List of 3
##  $ samples:'data.frame': 0 obs. of  0 variables
##  $ expr   :'data.frame': 10 obs. of  2 variables:
##   ..$ sampleID : Factor w/ 10 levels "1","2","3","4",..: 1 2 3 4 5 6 7 8 9 10
##   ..$ clusterID: Factor w/ 10 levels "a","b","c","d",..: 1 2 3 4 5 6 7 8 9 10
##  $ counts :'data.frame': 0 obs. of  0 variables
##  - attr(*, "class")= chr "cfList"
```

Note that the `expr` slot needs both a column named “sampleID” and “clusterID”, which are used to identify the origin of an individual cell. When using `readCytosploreFCS` this is all done automatically.

From here you can choose to use your own data, or it’s possible to follow along with the added data from Spitzer. So, let’s load the data into R:

```
dirFCS <- system.file("extdata", package="cytofast")
cfData <- readCytosploreFCS(dir = dirFCS, colNames = "description")
```

```
## Reading .FCS cluster: 1
## Reading .FCS cluster: 2
## Reading .FCS cluster: 3
## Reading .FCS cluster: 4
## Reading .FCS cluster: 5
## Reading .FCS cluster: 6
## Reading .FCS cluster: 7
## Reading .FCS cluster: 8
## Reading .FCS cluster: 9
## Reading .FCS cluster: 10
```

In total there were 10 .fcs files (also meaning 10 clusters) and are now stored into a `cfList`. We can acces the list and have a quick look at the marker expression of the cells.

```
cfData$expr[1:5, 1:5]
```

```
##           clusterID sampleID      Time Event_length        BC1
## 1 cytofastData1.fcs       16 6912180.0           36 581.915466
## 2 cytofastData1.fcs       20  778687.5           31   9.465774
## 3 cytofastData1.fcs        0 2145997.8           41   1.146241
## 4 cytofastData1.fcs        9  893060.8           45  14.375663
## 5 cytofastData1.fcs       15 7234341.0           34 347.519806
```

As shown, both clusterID and sampleID are added to the expression data. We also see that there are some ‘markers’ that we don’t want in our heatmaps. Therefore, in the next section we will clean-up the data and add some meta information to the `cfList`.

Cleaning up the data

Some channels are not needed (e.g. DNA1, DNA2, barcoding, etc…). We can remove those and only keep the markers that we are actually intrested in.

```
cfData$expr <- cfData$expr[,-c(3:10, 13:16, 55:59, 61:63)]
```

Next we will add some meta data to the `cfList`. Make sure that the sampleID corresponds with the sampleID of the meta data! When using Cytosplore, the sampleID is based on the sample tag (CSPLR_ST).

```
data(spitzer)
meta <- spitzer[match(row.names(cfData$samples), spitzer$CSPLR_ST),] # match sampleID
cfData$samples <- cbind(cfData$samples, meta)
```

Lastly, we will make some small changes to the labeling of the data. This helps with the visualization of the heatmaps and discovering relations of clusters.

```
levels(cfData$expr$clusterID) <- gsub("[^0-9]", "", levels(cfData$expr$clusterID))
levels(cfData$expr$sampleID) <- cfData$samples$sample_name
```

3. Visualization

Heatmaps

However, first we will address the last slot `counts`, which is still empty. To fill this slot we can call the function `cellCounts`. The default function will simply add the raw cell counts per cluster per sample into the cfList.

```
cfData <- cellCounts(cfData)
head(cfData$counts)
```

```
##           1 10  2  3  4  5  6  7  8  9
## 1_PD1_D3 32 68 19 20 19 27 51 14 24 46
## 1_PD1_D8 33 43 34 17 15 31 39 17 26 50
## 1_B6_D3   1  3 16  8 19 14 47  7  9 37
## 1_B6_D8   9  9 16 22 36 22 37 12  4 67
## 1_CD1_D3  0 10 10 10 18 17 41  9  9 37
## 4_CD1_D3  0 10 17 16 15 12 28 10  6 38
```

In many cases the abundance of the cells could be biased in respect to the sample size of the donors, therefore it could make more sense to look at the frequency of the clusters in respect to the total amount of cells per sample. We will also standardize the data (mean zero, unit variance) to make a fairer comparison between clusters.

```
cfData <- cellCounts(cfData, frequency = T, scale = T)
head(cfData$counts)
```

```
##                   1         10          2          3           4
## 1_PD1_D3  0.2123060  2.1764833 -1.2802619 -1.1578036 -0.79710186
## 1_PD1_D8  0.3233091  0.8532170  0.2047896 -1.3588553 -0.93108986
## 1_B6_D3  -1.0578183 -1.4106237 -0.1400055 -1.5386779 -0.02643354
## 1_B6_D8  -0.6210715 -1.0437467 -1.0236987 -0.2207599  0.44452716
## 1_CD1_D3 -1.1419325 -0.6061463 -1.2022531 -1.1693452 -0.10806673
## 4_CD1_D3 -1.1419325 -0.5380984  0.2152418  0.1136000 -0.28046313
##                     5         6          7          8         9
## 1_PD1_D3 -0.508726298 0.8491796 -1.1515787 -0.5534802 0.4825108
## 1_PD1_D8 -0.009465912 0.4231830 -0.9064829 -0.3232428 0.6899801
## 1_B6_D3  -0.434072346 2.6414025 -1.1571346 -0.9826655 1.3671319
## 1_B6_D8  -0.229890594 0.8322060 -0.9975814 -1.8546953 1.9479917
## 1_CD1_D3  0.104783253 2.1375120 -0.9031520 -0.9826655 1.3671319
## 4_CD1_D3 -0.665685717 1.1849822 -0.7009684 -1.3517980 1.5746208
```

Now everything is ready to visualize the data. One of the main functions of this package is `cytoHeatmaps`, which is used to visualize both the phenotype of the created clusters and their heterogeneity in respect to the samples.

```
cytoHeatmaps(cfData, group="group", legend=TRUE)
```

The function will plot two heatmaps: + on the upper panel the phenotype (based on the markers) of each cluster + on the lower panel the relative abundance of the clusters in respect to the samples

A hierarchical clustering was performed on subset frequencies using the Spearmans rank correlation, resulting in a dendogram displayed on the side of the sample abundancy heatmap. This will group sample sharing similarities.
Groups can pre-defined as an input to 'cytoHeatmaps'.

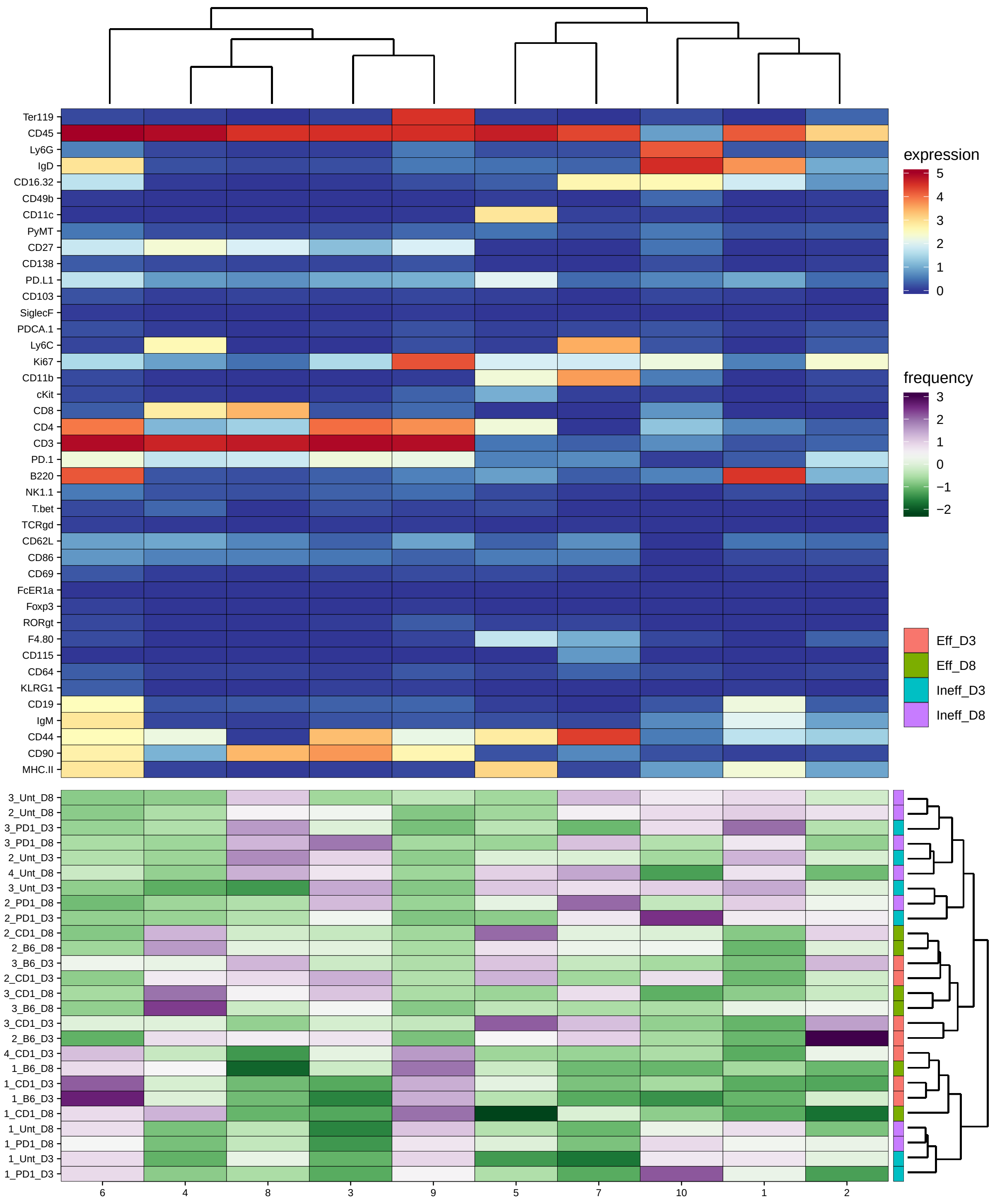
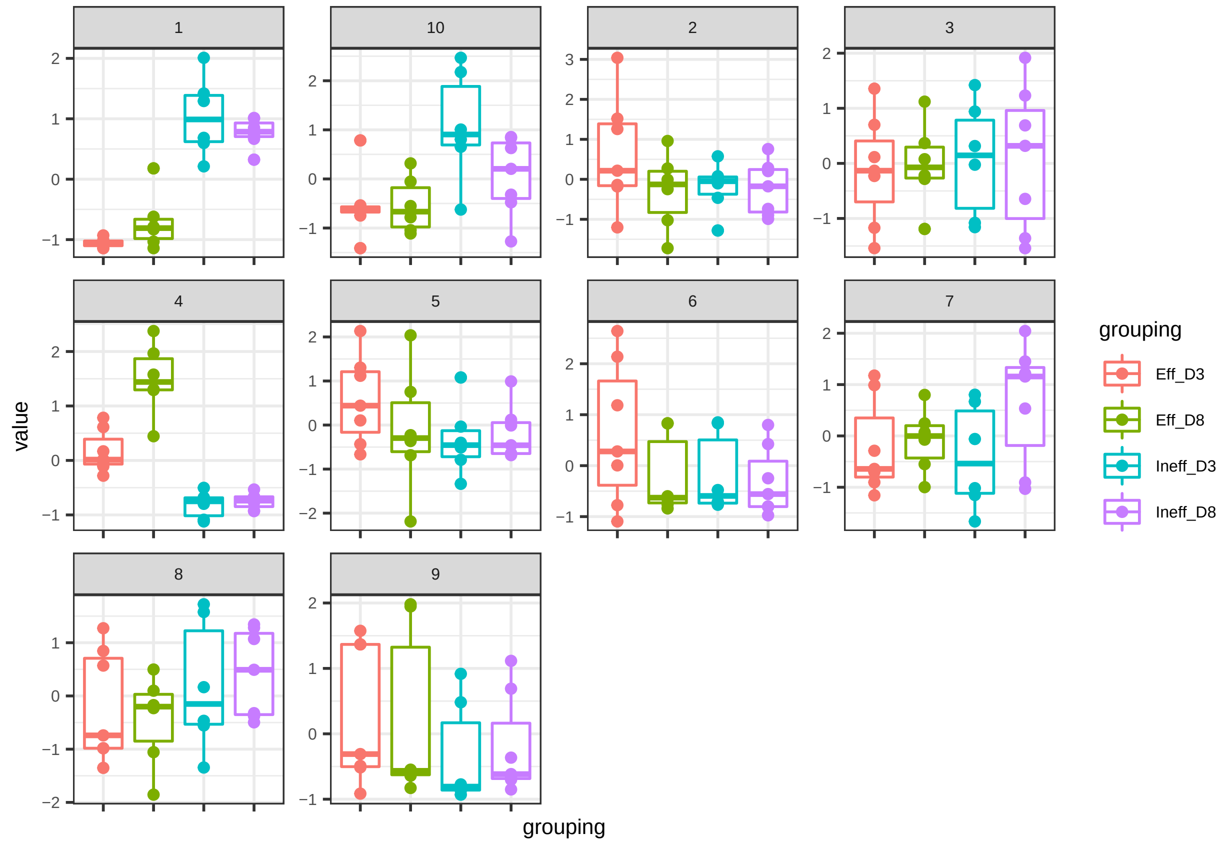


Figure 2: Cytofast heatmap

boxplots

We can also represents the data obtained in a quantitative way by calling the function `cytoBoxplots`. The output of this function represents the proportion of each sample for every cluster.

```
cytoBoxplots(cfData, group = "group")
```



density plots (median signal intensity)

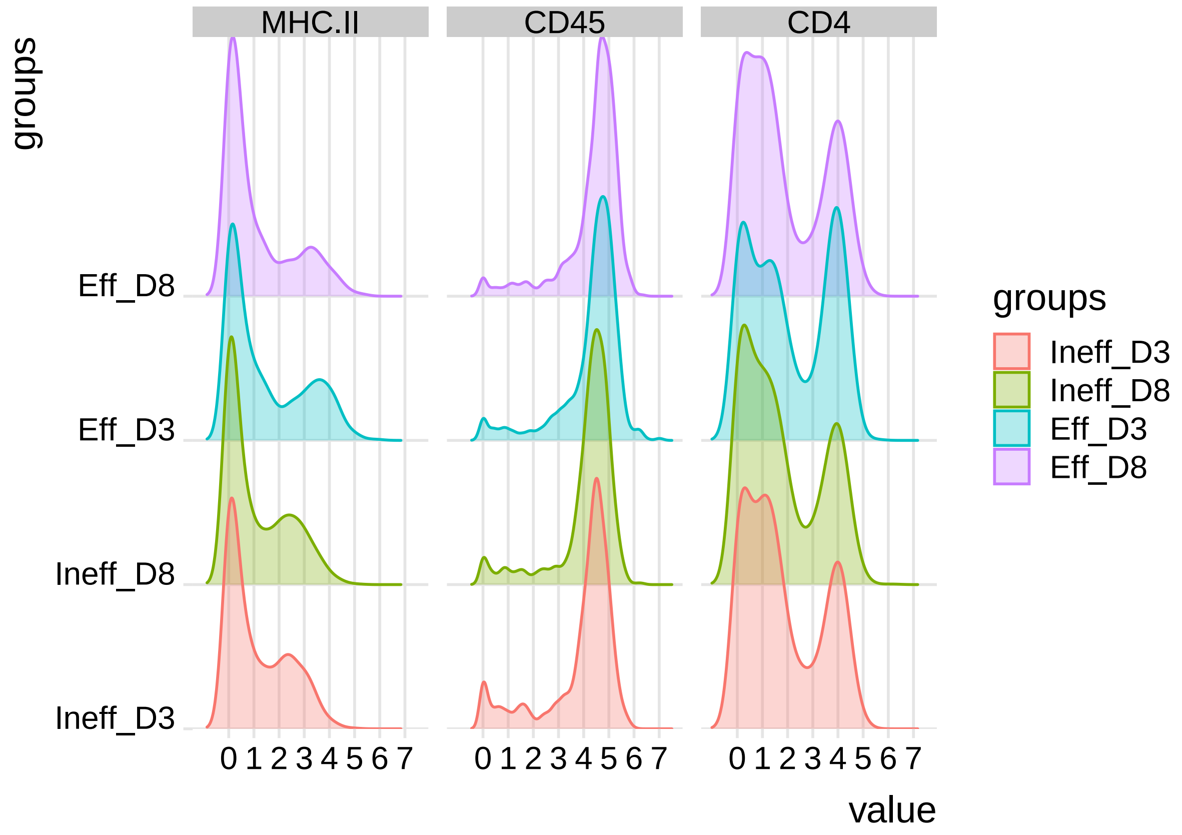
According to the *Spitzer et al., Cell, 2017* study, an effective therapy lead to an upregulation of MHC-II. This actually can be seen by drawing the median intensity plots for this marker, as shown below together with CD45 and CD4.

```
msiPlot(cfData, markers = c("MHC.II", "CD45", "CD4"), group = 'group')
```

```
## Picking joint bandwidth of 0.288
```

```
## Picking joint bandwidth of 0.151
```

```
## Picking joint bandwidth of 0.336
```



On this MSI plot, it indeed appears that the MHC-II signal is higher in the Effective therapy day 3 and day 8 compared to the Ineffective therapy day 3 and day 8.

4. Statistical testing

simple t.test

For further analysis, we will use the function 'globaltest'. This will guide us to find a cluster or a combination of clusters which are significantly different between groups.

global test

References

Beyrend, Guillaume, Koen Stam, Thomas Höllt, Ferry Ossendrop, and Ramon Arens. 2018. "Cytofast: An R-based roadmap for quantitative analysis of flow and mass cytometry data to discover immune cell subsets and correlations between groups." *Manuscript Submitted for Publication*.

Spitzer, Matthew H, Yaron Carmi, Nathan E Reticker-Flynn, Serena S Kwek, Deepthi Madhireddy, Maria M Martins, Pier Federico Gherardini, et al. 2017. "Systemic Immunity is Required for Effective Cancer Immunotherapy HHS Public Access." *Cell* 26 (168(3)): 487–502. doi:10.1016/j.cell.2016.12.022.