MCTA First Steps

Carine P. Beatrici, Fabricio A. B. Silva

May 15, 2020

1 Introduction

MCTA (Multiparametric Color Tendency Analysis) is a new method for flow cytometry data analysis that shows the phenotypic profile of cellular subpopulations considering multiple labels at the same time, extending and complementing conventional analysis. In this document, the main functionalities of this program and associated scripts are discussed.

The dependencies to compile and execute the MCTA application are:

1. A Linux/MacOS system; The current version of MCTA is nor compatible with Windows
2. The **make** utility;
3. The **gawk** utility (https://www.gnu.org/software/gawk/);
4. The **gfortran** compiler (https://gcc.gnu.org/fortran/);
5. The **gnuplot** application (http://www.gnuplot.info);
6. The **R** compiler (https://www.r-project.org);
7. The Bioconductor package **flowcore**;
8. The **git** application (https://git-scm.com/downloads).

For more details on the flowcore package, please access

`https://www.bioconductor.org/packages/release/bioc/html/flowCore.htm`

Please consult the documentation of your operating systems to install any of the above applications/utilities if needed. Please note that the MCTA application was tested on Linux and MacOS systems only.

2 Files in the Repository

A short description of the files available in the https://github.com/flowcytometry/MCTA repository is below:

1. FCS2CSV.R - R script that converts the FCS file in text format, used internally by the MCTA application.
2. Makefile - Compiles the code into an executable file.
3. Sample_37.fcs - Flow cytometry experimental data used as an example. The data in this file were generated by an experiment corresponding to mechanically macerated livers obtained from control C57BL/10 mice.
4. Sample_39.fcs - Flow cytometry experimental data used as an example. The data in this file were generated by an experiment corresponding to mechanically macerated livers obtained from *Trypanosoma cruzi* infected mice.
5. in.dat - This configuration file uses the Sample_39.fcs file as data source. The MCTA application will generate the color dot-plot according to this configuration file.

6. in_Sample_37.dat - Another example of the configuration file, this time using Sample_37.fcs as input file.
7. MCTA.f90 - Fortran program that generates the colored dot-plot.
8. MCTA.pdf - This Tutorial.
9. script-gnu-color - Gnuplot script, this script uses the information in the temporary files to generate the colored dot-plot.
10. script_hue_filter- Interactive dot plot script. This script filters resultant colors based on hue values (0.0 - 360). It requires the fluo.dat file generated by MCTA.
11. mm.r - R script required by script_hue_filter.

3 Compiling

The first step is to make sure the computer has all the dependencies installed and working correctly. Pleases refer to the web sites listed in the Introduction section. The second step is to download all files from the github repository
(git clone https://github.com/flowcytometry/MCTA). To do so, run the git command in the destination directory:

```
git clone https://github.com/flowcytometry/MCTA
```

To compile the code, open a Linux/MacOS terminal in the same folder where the files were saved. Then compile the code using "make":

```
> make
```

Note that all files described in section 2 should be in the same directory. If the compilation goes well and there are no compilation errors, you should see an executable file named "MCTA". To execute the program just type the following command in the linux terminal:

```
> ./MCTA
```

To get the graphics with the colored dot-plots, we need to provide an input configuration file ("in.dat"). In the github repository, we provide examples of this configuration file, as described in section 2.

4 The Configuration File

To build the color dot plots, it is necessary to provide a configuration file. In MCTA this configuration file is named **in,dat**. Here we show an example configuration file and explain how to edit it in order to perform different analysis on the data. Since **in.dat** is a text file, you can use any text editor available in you operating system (e.g. vim, nano, etc.) to create or edit the **in.dat** file.

The input file must be named as **in.dat**. One available example is shown below. This example corresponds to the **in.dat** file available in the github repository:

```
1   Sample_39.fcs
2   9
3   5
4   1 5 6 7 9 10 12 14 15
5   525 575 675 660 774
6   10.09  17.35  46.76  44.37  13.47
7   100.00     1.40      0.00    0.00     0.00
8    18.22   100.00      0.00    0.00     0.0
9     2.80    15.42    100.00    0.00     0.00
10    0.00     0.41      6.86  100.00    14.49
11    0.00     0.00      0.00    6.54   100.00
12  0 0 0 0 0
```

4.1 Detailed Description of the in.dat file

The example shown in the previous subsection is composed of 12 lines. The number of lines may vary as a function of the number of channels, for instance. It is important to keep the correct order of the input lines. We can group the input lines in eight items, organized by their function in the process, as described below:

```
1) name of the FCS file.........................Sample_39.fcs
2) total number of columns in the fcs file.......9
3) number of fluorescence channels considered in the analysis.........5
4) columns to be copied into memory..................1 5 6 7 9 10 12 14 15
first the fsc, ssc and optional data
after the fluorescence channels
5) wavelength values for the channel colors......525 575 675 660 774
6) Background values.............................10.09  17.35  46.76  44.37  13.47
7) Compensation values.....................................
...100.00    1.40    0.00    0.00     0.00
....18.22  100.00    0.00    0.00     0.00
.....2.80   15.42  100.00    0.00     0.00
.....0.00    0.41    6.86  100.00    14.49
.....0.00    0.00    0.00    6.54   100.00
8) Fluorescence channels used in the color resultant calculation..0 0 0 0 0
```

The in.dat must provide the above information in order, without any other text between lines. A more detailed description of the eight items follows:

1) The first line is the complete name of the Flow Cytometry Standard (FCS) file. The program will read this FCS file and process it.

2) The second line is the number of columns that the program will copy from the FCS file.

3) Number of fluorescence signals. For example, if you want to analyze seven fluorescence signals and there are five other parameters in the FCS file (such as time, etc.), the second line must have 12 and the third line must have 7.

4) The columns to be copied. Here you need to know which columns of the FCS file have the information you want to display in the color dotplot. Please list first general data columns (like event ID, time, morphology, etc.) and then the fluorescence signal columns.

If you do not know which columns to choose, you can open the FCS file and read the first line or execute the MCTA application. The program will show you the first line of the file with all the names of the columns for you to choose. Column counting starts at 1.

5) The wavelength values can be related or not to real filter values. The MCTA application automatically distributes colors evenly along the hue circle, ordered according to the wavelength values informed.

6) The background labeling corresponds to the maximum fluorescence value of negative events. Any positive event for a given channel must have superior fluorescence intensity, when compared with the background reference value. This value must be in agreement with the data.

7) The conventional compensation table. The values must be between [0,100] and the value 100 is required when the same channel is considered for both dimensions of the matrix.

8) To calculate the color tendency for all the channels, set these values as zeros (one zero for each color channel being considered in the analysis) or complete with the sequence according to the number of channels (e.g. 1 2 3 4 5). By doing this, all the channels are taken into account to define the resultant color.

To calculate the resultant color only for positive events for a subset of channels, type the corresponding channel numbers followed by zeros, e.g. 2 4 0 0 0.

In this case, only the positive events for both the second and fourth channels will be considered and will receive a color, all others are going to stay black. In this example, for resultant color calculation, the fluorescence of all channels but the second and fourth channels will be considered.

To calculate the resultant color for negative events in a set of channels, just put the number of channels as negative numbers, e.g. 2 -4 0 0 0. Using this configuration, the events are going to receive a color if and only if they are positive for the second channel and negative for the fourth

channel (single positive events). In this example, for resultant color calculation, the fluorescence of the second and fourth channels will not be considered.

5 Execution Example

When you run the MCTA application, you obtain one output file, **result_file.eps**, with the color dot plot.

The color dot plot corresponding to the **Sample_39.fcs** and **in.dat** files discussed before is shown in figure 1. The color bar in the right shows the distribution of colors in the HSL representation, along with the respective angles.

Please make sure you rename output files before running new tests, otherwise old output files will be overwritten.

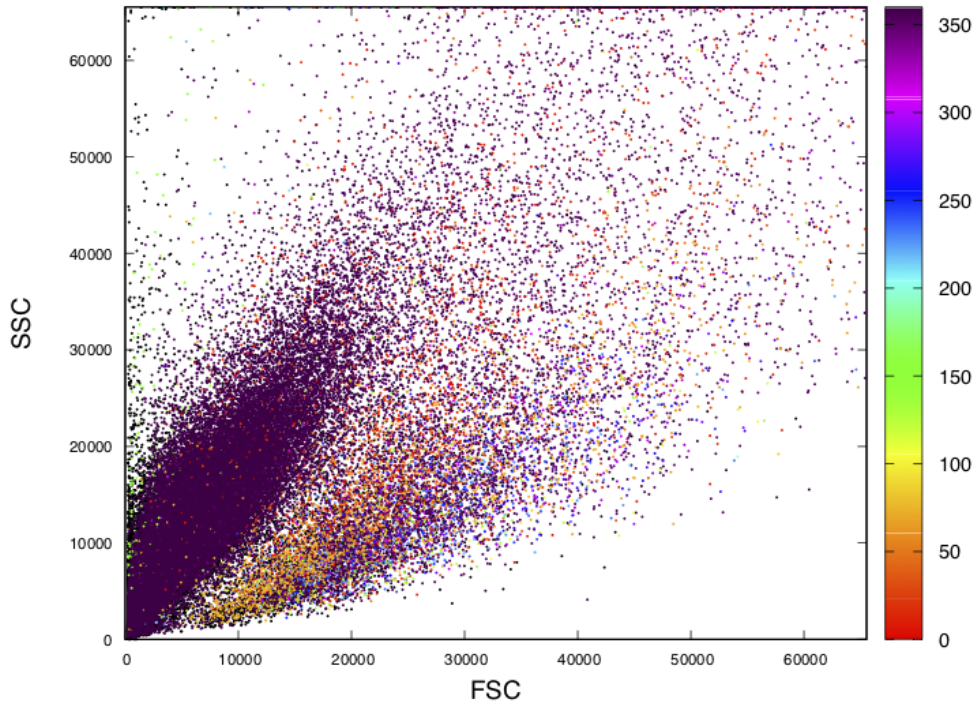

Figure 1: The color dot plot for 5 colors.

5.1 Selection of Positive Filters

For the flow cytometry analysis, it may be interesting to see not all the colors in the tendency plot at the same time. In some situations, when showing all the colors, the results may hide populations due to the high number of other cell types in the same region. To select only events with a positive signal for the channel 1 (FITC in this example) edit the file **in.dat**. Change the line number 12 from "0 0 0 0 0" or "1 2 3 4 5" to "1 0 0 0 0". The results for this selection are presented in figure 2.

5.2 Selection of Negative Filters

For the flow cytometry analysis, it may be interesting to see not all the colors in the tendency, because the results may hide populations due to the high number of other cell types in the same region. To select only events with negative signal for the channel 1 (FITC in this example) edit the file **in.dat**. Change the line number 12 from "0 0 0 0 0" or "1 2 3 4 5" to "-1 0 0 0 0". The results for this selection are presented in figure 3.

Comparing figures 2 and 3 we can see that the colored regions are different. In fact, all the dots colored in the figure 2 are now black. The only black dots in both figures are the negative events for all fluorescence channels.

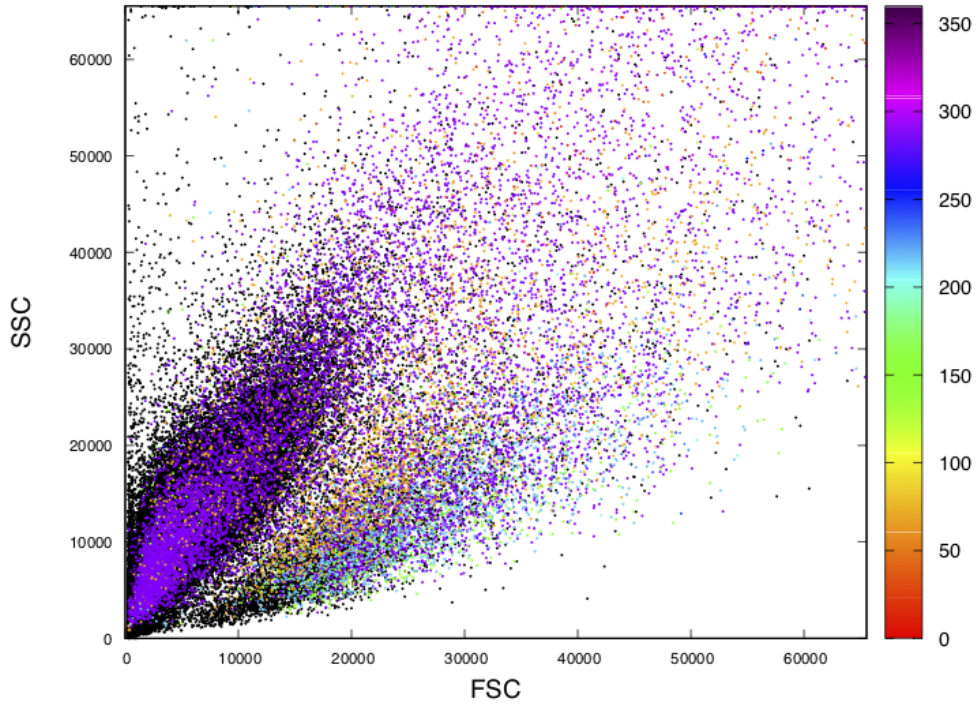

Figure 2: In this case we selected only the positive dots for the channel number 1, which corresponds to the FITC marker.

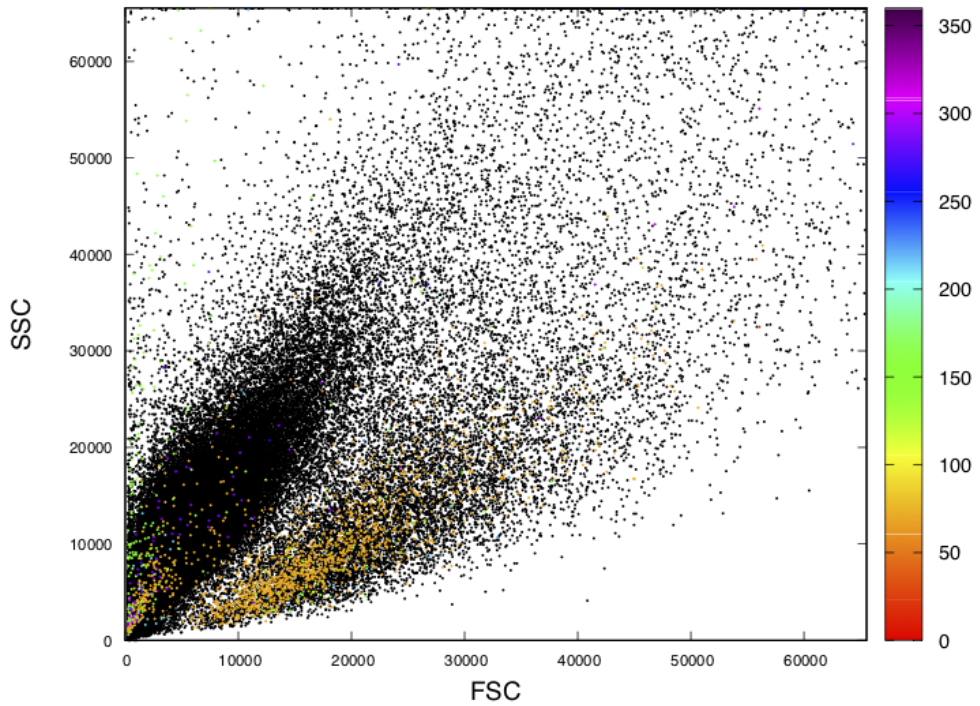

Figure 3: The color dot plot, as described in section 5.2. In this case we selected only the negative dots for the channel number 1, which correspond to the FITC marker.

5.3 Selection of Filter Combinations

It is possible to select any combination of channels for positive or negative events. To select a combination of positive and negative values the user should edit the input file **in.dat** in line 12. For example, if you want to see the color tendency for all events with the second filter (fluorescence)

positive and the fourth filter (fluorescence) negative, the line 12 must be changed to "2 -4 0 0 0".

5.4 Examples of Filter Selection

Additional examples of the use of fluorescence combinations are provided in this subsection:

- **0 1 0 0 0** - Process events with positive signal for channel 2.
- **2 -4 3 0 0** - Process events with positive signals for channels 1 and 3 and negative signal for channel 2
- **1 2 3 -4 0** - Process events with positive signals for channels 1, 2 and 3 and negative signal for channel 4.

6 Gnuplot Scripts

The MCTA application creates two temporary files (**fluo.dat** and **nofl.dat**) and images are generated via gnuplot by the script provided (*script-gnu-color*). The **fluo.dat** file generated by MCTA corresponds to events associated with a color.

The MCTA application calls gnuplot automatically. The gnuplot script provided in the github repository were parametrized specifically for the example configuration file provided (**in.dat**). However, it is up to the user to edit the gnuplot script for new data.

To create the color dot plot, the script below reads the temporary files and plot them into the final eps images (see the *script_ gnu_ color* file).

```
1....set termoption enhanced
2....set title font ", 20"
3....set xlabel 'FSC'
4....set xlabel font ", 18"
5....set xtics font ", 11"
6....set ylabel font ", 18"
7....set ytics font ", 11"
8....set ylabel 'SSC' enhanced
9....set nokey
10...set cbrange [0:360]
11...set palette defined ( 0 '#ff0000',\
.......................1 '#ff7d00',\
.......................2 '#ffff00',\
.......................3 '#00ff00',\
.......................4 '#00ffff',\
.......................5 '#0000ff',\
.......................6 '#ff00ff',\
.......................7 '#440044')
12...plot [0:65533][0:65533] "nofl.dat" using 2:4:5 with points
     pt 7 ps 0.3 lc rgb "black"
13...set term postscript eps enhanced color
14...set output "result_file.eps"
15...replot "fluo.dat" using 2:4:5 with points palette pt 7 ps 0.3
```

1... Enable the color dot mode in gnuplot. Do not change this line.
2... Set the title font size.
3... Set the x axis label, the default is FSC.
4... Set the x axis label font size.
5... Set the tics label font size of the x axis.
6... Set the y axis label font size.
7... Set the tics label font size of the y axis.
8... Set the y axis label, the default is SSC.
11.. Define the palette, i.e. the colors of the graphic bar. The default values can be a visual approximation of the wavelength values.
12...The values inside the braces are the region plotted. For example, [0:1024][0:1024] will plot only

the values between 0 and 1024 in both axes. **Please make sure these values are adjusted to include all data.** The **nofl.dat** temporary file corresponds to events not associated with any color and are plotted in black.

After the word using there are three numbers (2:4:5 in this example). These parameters are very important, since they define the data sources for the dot-plot. The first number is the column where the FSC information is located. The second number is where the SSC information is located. The third number is where the color is located.

To know where the data you want to plot are located in temporary files you have to check the **in.dat** configuration file. In this file, when you set the number of total columns and the number of fluorescence channels, temporary files store data accordingly. Therefore, in this example, in line 2, the total number of data columns equals 10 and, on line 3, we have 6 fluorescence channels, so there are 4 columns of other parameters (such as event ID, FSC, SSC). By doing this, resultant color results will be stored in the fifth column of both temporary files.

When you choose the columns in the configuration file (line number 4 of the **in.dat** file), you choose the order of the temporary file columns. If you keep the second column as the FSC and the fourth column as the SSC, there is no need to edit this script. However, it will depend on how you selected columns in the **in.dat** configuration file.

13 and 14 ..The lines 13 and 14 set the output format and file.
15... the same edition made in line 12 is required here. The **fluo.dat** temporary file corresponds to events associated with a color.

7 Resultant Filter Script - Hue Values

This script is used to filter resultant colors on the color dot plot generated by the MCTA application based on hue values (0.0-360.0). In order to filter resultant colors, the user should run the *script_hue_filter* script in the same directory where the **fluo.dat** file, generated by the MCTA application, is located. The user should inform the upper and lower limits of the filter, for instance:

Lower limit (0.0-360.0): 300
Upper limit (0.0-360.0): 340

The script generates an encapsulated postscript file (**result_filter.eps**) containing the new color plot generated when the filter is applied. See Figure 4 for an example of the application of the resultant filter in the same data used to generate Figure 1. The corresponding command line execution is shown in Figure 5.

Since the *script_hue_filter* script is based on the script described in section 6, most remarks of that section are also valid for *script_hue_filter*.

The *mm.R* script is executed by *script_hue_filter* to generate statistics related to the resultant filtering operation. Please note that both *mm.R* and *script_hue_filter* must have execution permission (use the *chmod* Linux command).

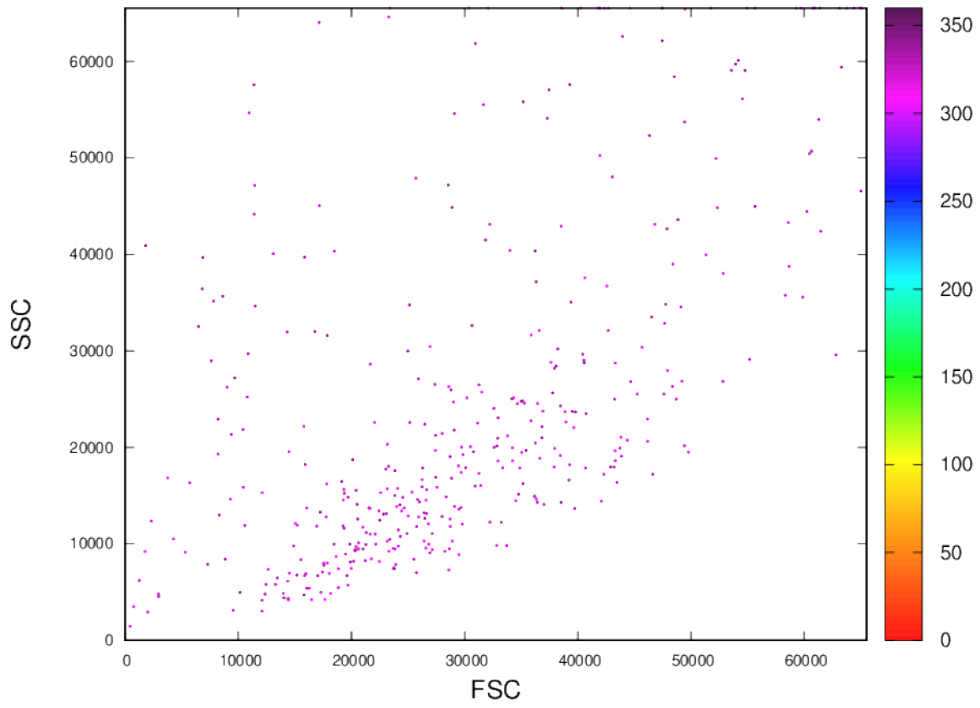

Figure 4: Color dot plot when the 300-340 resultant color filter is applied.

```
Lower limit (0.0-360.0): 300
Upper limit (0.0-360.0): 340
Number Events:  559
Number Columns: 14
CHANNEL: 1
Median
71.1492
Geometric Mean
70.40674
CHANNEL: 2
Median
15.5471
Geometric Mean
13.80425
CHANNEL: 3
Median
0
Geometric Mean
1.874025
CHANNEL: 4
Median
24.92087
Geometric Mean
12.13583
CHANNEL: 5
Median
0
Geometric Mean
2.004024
```

Figure 5: Execution of the 300-340 resultant color filter.

The first part of the script is below:

```
1...#!/bin/bash
2...read -p 'Lower limit (0.0-360.0): ' lowerlim
3...read -p 'Upper limit (0.0-360.0): ' upperlim
4...gawk -v up="$upperlim" -v lo="$lowerlim" '{if (($5>=lo) && ($5<=up)) print $0}'
    fluo.dat > tmp.dat
5...NUMEVENTOS=$(< "tmp.dat" wc -l)
6...echo 'Number Events: '$NUMEVENTOS
7...if (($NUMEVENTOS == 0))
8...then
9.....echo "There are zero events associated with the defined limits"
10....exit
11..fi
12..NUMWORDS=$(< "tmp.dat" wc -w)
13..NUMCOL=$[NUMWORDS/NUMEVENTOS]
14..echo 'Number Columns: '$NUMCOL
15..let NUMCOL+=1
16..COUNTER=10
17..while [ $COUNTER -lt $NUMCOL ]; do
18....gawk -v up="$upperlim" -v lo="$lowerlim" '{if (($5>=lo) && ($5<=up)) print $'$COUNTER'}'
      fluo.dat > testeMM.dat
19....echo 'CHANNEL: '$[COUNTER-9]
20...../mm.r < testeMM.dat
21....let COUNTER+=1
22..done
```

Lines 2 and 3 read filter limits defined by the user. Line 4 filters the fluo.dat file, creating a temporary file (tmp.dat) with only the events with resultant filter values (column #5 of the fluo.dat file) inside the interval defined by the user. Lines 5-11 counts the number of filtered events and checks if the total number of events is greater than zero. Lines 12-16 and the loop correspondent to lines 17-21 generates statistics on filtered events (using the R script mm.r in line 20).

The second part of the script is equivalent to the Gnuplot script presented in section 6, and it generates an image plotting the filtered events.

8 Channel Statistics

Several channel statistics are computed by the MCTA application by the end of each execution. In this version, the MCTA provides the geometric mean, the geometric standard deviation and the median for each fluorescence channel. For the statistic measures computation only non-zero fluorescence values are considered. Statistic measures for the **Sample_39.fcs** file are shown below:

```
Channel: 1  Geometric Mean: 11.5453758  Geo Std dev: 5.15835667  Median: 12.7025604
Channel: 2  Geometric Mean: 9.64021587  Geo Std dev: 5.11912155  Median: 10.4752026
Channel: 3  Geometric Mean: 35.7162018  Geo Std dev: 5.14500427  Median: 37.1795692
Channel: 4  Geometric Mean: 43.4695969  Geo Std dev: 3.62519193  Median: 53.8346291
Channel: 5  Geometric Mean: 9.50257301  Geo Std dev: 4.24476528  Median: 11.5507116
```