Quality Assessment of Ungated High Throughput Flow Cytometry Data-using the flowQ package

N. Gopalakrishnan, F. Hahne

April 22, 2010

1 Introduction

The advent of high throughput methods has resulted in a large increase in the amount of data available from cytometry (FCM) experiments. This large amount of information needs to be summarized and presented in a visually appealing manner, so that the researcher can draw appropriate inferences from the data.

Quality assessment (QA) is an important step in the data analysis pipeline, often helping researchers identify differences in samples originating from changes in conditions that are probably not biologically motivated. The general aim is to establish a quality control criterion to give special consideration to these samples or even exclude them from further analysis.

The flowQ package provides a range of highly extensible tools to perform QA on one or several *flowSets*, as well as a unified infrastucture to present the results of such analyses through diagnostic plots, numeric output, and qualitative indicators, all linked together using interactive HTML. Since QA criteria differ considerably depending on the experimental setup, we have tried to keep our framework as generic as possible. See the sister vignette "Extending flowQ: how to implement QA processes" of the *flowQ* package for details about extensibility. The focus of this vignette is on the application of existing methods as well as the general ideas behind the established QA criteria.

2 QA components and the HTML reports

The design of *flowQ* is modular, reflecting the typical need for multiple QA criteria to comprehensively check several aspects of FCM data. The outcome for each criterion is represented by a single module, which is implemented in objects of class *qaProcess*. Unless the user wants to extend the funcionality of the package, these objects are typically created from *flowSets* using one of the existing constructor functions. Roughly, the internal structure of a *qaProcess* object can be thought of as a list of QA results for each sample in a *flowSet*, including visualizations by means of diagnostic plots and sample-specific quantitative or qualitative values (Figure 1). This design allows the software to bundle the results of multiple modules using a module-independent front end. For the convenient interactive navigation on a computer, the front end is a clickable HTML report, but in the setting of an automated pipeline, this might as well be direct storage in a data base.

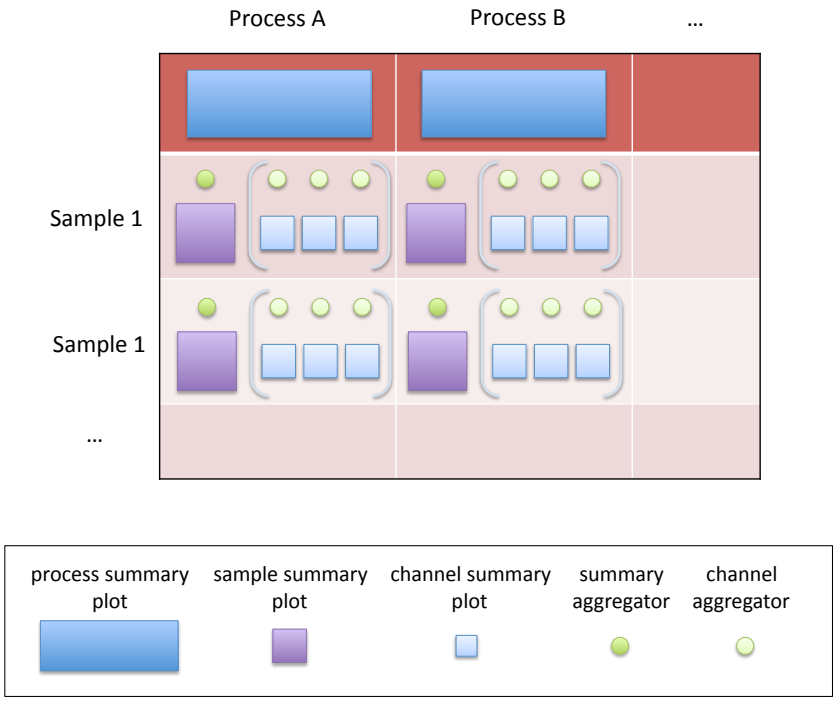


Figure 1: Schematics of an flowQ QA report. Columns in the table represent different QA criteria, rows represent samples. For each criterion there may be a single overall summary plot. For each sample in the respective QA criterion there may be a single detailed plot as well as several channel-specific plots if necessary. Accordingly, for each sample, the overall QA result is sumamrized by a single aggregator, and optional channel-specific results can be summarized by additional channel aggregators.

The most simple FCM experiments can be handled by a single *flowSet*, however more complicated designs such as longitudinal sample studies or multi-panel experiments are prevalent, and we have taken some steps to accomodate such designs in our framework. We begin by taking a look at the simple designs first. More complicted multi-panel designs are treated in section 4.

3 Quality assessment of single panel experiments

There are four methods in flowQ that are implemented for the analysis of single panel experiments. Neither of them assumes any relationship between the samples — with the exception of basic sample groupings — and they should be universally applicable to almost any FCM data.

Cell number

The most simple QA criterion is available through the `qaProcess.cellnumber` constructor. It will check whether the total number of events for a given sample is above a certain threshold. This treshold can either be absolute (by using the `absolute.value` argument), or determinded as an outlier from the average number of events per sample. Assuming that there are groups of samples within the *flowSet*, we can perform the outlier detection with respect to the particular group of a sample rather than the whole set. The inputs for `qaProcess.cellnumber` used here are the `GvHD` *flowSet*, which we can be loaded from the flowCore package, the output directory for the summary plot that is generated (see Figure 2), and a tuning parameter that controls the sensitivity of the outlier detection:

```
> library(flowQ)
> data(GvHD)
> GvHD <- GvHD[1:10]
> dest <- file.path(tempdir(), "flowQ")
> qp1 <- qaProcess.cellnumber(GvHD, outdir = dest, cFactor = 0.75)
```

Boundary events

A typical FCM instrument has a certain dynamic range over which it acquires a signal. All fluorescence intensities that fall out of this range will be accumulated as margin events, i.e., they are artificially given the minimum or the maximum value of the dynamic range. A high number of these events usually indicates potential problems with compensation, instrument settings or drifts. We can use the `qaProcess.marginevents` constructor to create a *qaProcess* object that checks for abnormal accumulation of boundary events. The inputs are very similar to the previous example, but we need to additionally specify the channels we want to include in the analysis. The default is to take all, but for now we will only use FSC and SSC. We also don't create pdf versions of the graphics in order to save some time and disk space.

```
> qp2 <- qaProcess.marginevents(GvHD, channels = c("FSC-H", "SSC-H"),
+     outdir = dest, pdf = FALSE)
```

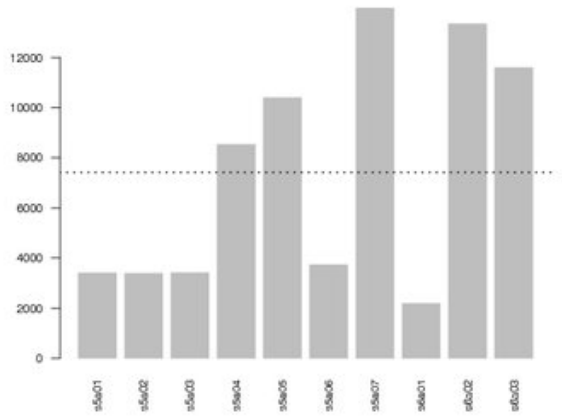


Figure 2: Summary graphics for the cell number QA criterion produced by the `qaProcess.cellnumber` function.

An example for the summary plot created for the FSC channel is shown in Figure 3. Figure 4 shows a sample-specific density plots, which are accessible through the HTML drilldown.

Time anomalies

Most FCM instruments record a time tick for each event they measure. This information can be used to detect drifts on the instrument over time, abnormal flow rates or sudden jumps in measurement intensities. Two QA criteria have been implemented in flowQ to address time-dependent anomalies in the `qaProcess.timeline` and `qaProcess.timeflow` constructors. The former tries to find unexpected non-random acquisition of fluorescence intensities for one or several FCM channels, while the latter checks for steady, uniterupted flow rates.

We can produce *qaProcess* objects for both criteria using the code in the following chunk:

```
> GvHD <- transform(GvHD, `FL1-H` = asinh(`FL1-H`), `FL2-H` = asinh(`FL2-H`))
> qp3 <- qaProcess.timeline(GvHD, channel = "FL1-H", outdir = dest,
+     cutoff = 1)
> qp4 <- qaProcess.timeflow(GvHD, outdir = dest, cutoff = 2)
```

Figure 5 shows a sample overview plot produced by `qaProcess.timeline`. These are smoothed regression lines of fluorescence intensity vs. time, and we expect a straight horizontal line for well-behaved samples. Wiggly lines, sudden jumps or trends all indicate potential problems. The `timeFilter` class in flowCore can be used to remove problematic events from a *flowFrame*.

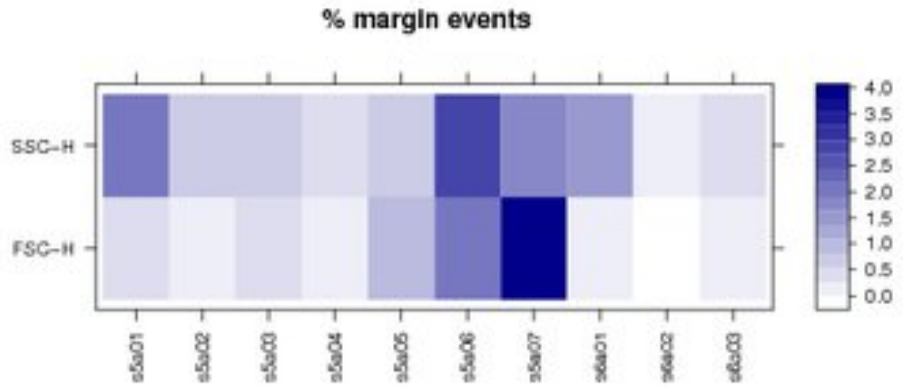


Figure 3: Summary graphics of the FSC-H channel for the boundary event QA criterion produced by the `qaProcess.marginevent` function.

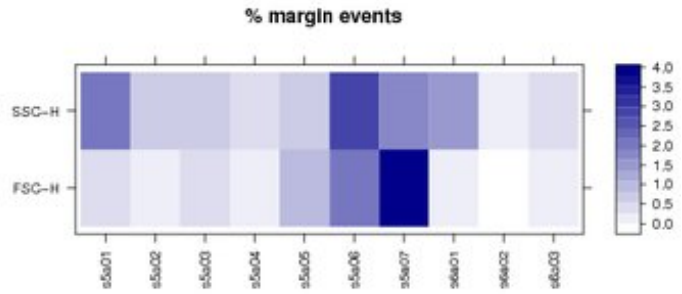


Figure 4: Detailed sample-specific graphics of the FSC-H channel for the boundary event QA criterion produced by the `qaProcess.margineventfunction`.

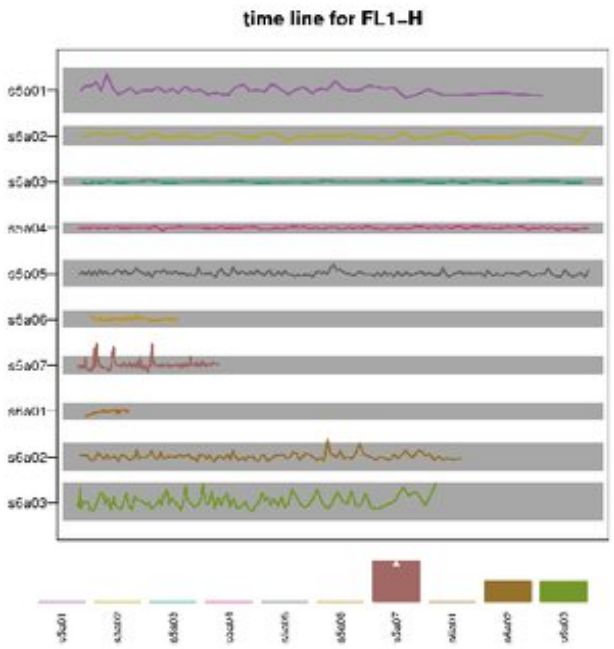


Figure 5: Summary graphics of the FL1-H channel for the time line QA criterion produced by the `qaProcess.timeline` function.

HTML report

As mentioned in the introduction, the *qaProcess* objects are independent entities containing the QA results. Presentation of these results is the job of a frontend. In flowQ we have developed a frontend to create interactive HTML output for one or several *qaProcess* objects in the `writeQAReport` function. The inputs to the function are a *flowSet*, a list of *qaProcess* objects generated for the same data, and an output directory to generate the report in. It makes sense to use the same output directory as before when creating the different *qaProcess* objects since the images do not have to be copied any more. The following code chunk produces the HTML report for all four QA criteria introduced before:

```
> url <- writeQAReport(GvHD, processes = list(qp1, qp2, qp3, qp4),
+     outdir = dest)
```

We can take a look at the final report by pointing a browser to the url returned by the function.

```
> browseURL(url)
```

4 Quality assessment of multi-panel experiments

In many flow cytometry experiments, samples from patients are often divided into several aliquots. The aliquots are then stained using antibody-dye combinations that are specific for certain antigens presented on the cell surface or for particular intracellular markers. The basis for many quality control procedures is that morphological parameters like Forward and Side Scatter, which are dependent on the cell size and the granularity of the cell, should be similar across aliquots. Additionally, certain fluorescent dyes utilized in the staining procedure may be replicated in some of the aliquots. Flourescence intensities recorded from such dyes are also expected to be similar across aliquots.

Several one and two dimensional methods have been developed in the flowQ package that helps users perform quality assessment. The package also provides infrastructure to generate interactive quality reports based on a unified HTML output.

Our sample data set involves samples collected from 4 individuals which were then split into 8 aliquots.Each aliquot was then stained with a different combination of stains as shown in the figure below.

4.1 Data preprocessing and transformation

The first step in the data analysis pipeline involves reading in the cytometry data which can be achieved using the `read.FCS` function for FCS files or using the `load` function for flow cytometry data that was saved from a previous R workspace. In our example, the data file qData.rda resides in the data folder of the flowQ package and can be loaded with the `data` command.

```
> data(qData)
> qData[[1]][[1]]
```

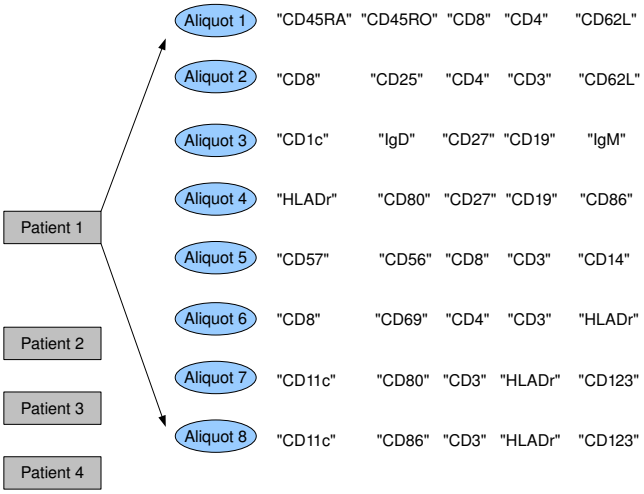


Figure 6: Fluorescence dyes used in the experiment

```
flowFrame object 'pid02050'
with 4500 cells and 8 observables:
        name   desc range minRange maxRange
$P1    FSC-A  FSC-A  1024        0     1023
$P2    SSC-A  SSC-A  1024        0     1023
$P3   FITC-A CD45RA  1024        1    10000
$P4     PE-A CD45RO  1024        1    10000
$P5    FL3-A    CD8  1024        1    10000
$P6 PE-Cy7-A    CD4  1024        1    10000
$P7    APC-A  CD62L  1024        1    10000
$P8     Time   <NA>  1024        0     1023
82 keywords are stored in the 'description' slot
```

The data read into an R session is a *list* containing 8 *flowSets*, each *flowSet* corresponding to a set of aliquots. Each *flowSets* contains data from four patients. During the experiment, blood was drawn from these four patients and split into the 8 aliquots after initial processing but prior to any staining. It is fair to assume that these aliquots should be similar in all aspects that are not staining related.

The description column in the parameters slot of each *flowFrame* contains information regarding the stains used for each sample. In data sets that do not contain the stain information, the description field of each *flowFrame* needs to be updated with the corresponding stain information. Description fields for parameters like forward/side scatter and Time are to be filled with `NA`, since they are not associated to a specific staining marker. The methods `pData` and `parameters` can be used to update the description field.

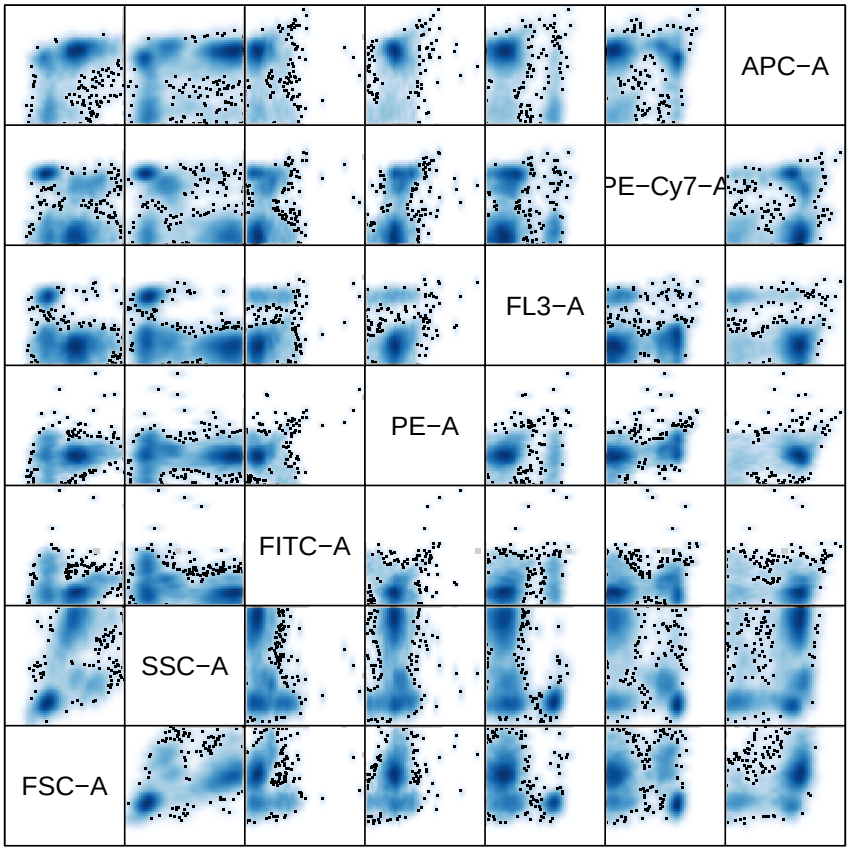
The fluorescence parameters first need to be transformed for better visualization of the data. For our sample dataset, all flow parameters except forware and side scatter were transformed

using the `asinh` transformation.

```
> tData <- lapply(qData, function(x) transformList(colnames(x)[3:7],
+     asinh) %on% x)
```

A pairwise plot of all parameters for the first *flowSet* is shown below

```
> library(flowViz)
> plot(tData[[1]][[1]])
```



4.2 QA process for boundary values

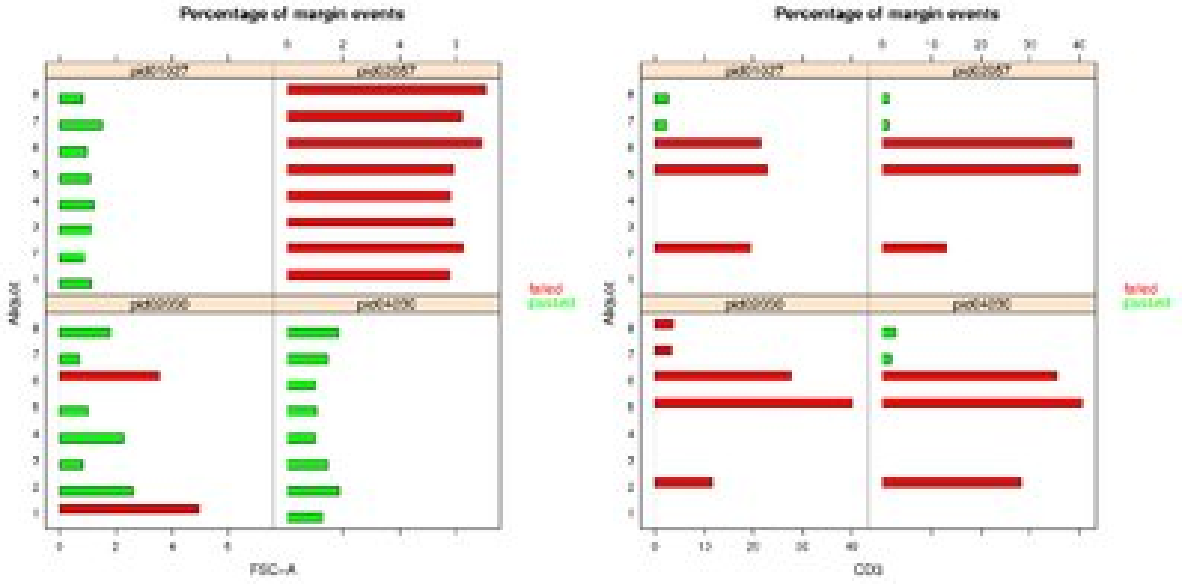
A large number of boundary values could possibly cause problems during the data analysis stage. Since the QA procedure compares the flow parameters for each patient across the eight aliquots, we have to ensure that the percentage of boundary events is not unproportionally high. This could cause issues with other QA checks, especially for the 1D QA processes introduced later, which involve estimating the density and KL distances or when 2D summary statistics such as the means are computed. Moreover, high numbers of boundary events often indicate compensatation problems or instrumental drifts during the data acquisition.

The `qaProcess.BoundaryPlot` allows the user to quickly visualize the percentage of boundary events for each patient across aliquots in the data set. The output generated from the function can be written to an html report using the `writeQAReport`. A related function, `qaProcess.marginevents` can be used to detect boundary artifacts for single *flowSets* in settings without aliquots.

The percentage of boundary events for "FSC-A" and "CD3" that exceed a set cutoff of 3% can be visualized using the `qaProcess.BoundaryPlot` function as shown below.

```
> resBoundary <- qaProcess.BoundaryPlot(tData, dyes = c("FSC-A",
+     "CD3"), outdir = dest, cutoff = 3, pdf = TRUE)
> imagePath <- resBoundary@summaryGraph@fileNames[2]
```

The percentage of boundary events for "FSC-A" and "CD3" are shown in the figure below. Each panel in the figure represents data from a patient with each horizontal bar indicating the percentage of boundary events for an aliquot. The horizontal bars for aliquots whose boundary events exceed our setcutoff value of 3% are colored red.



Clearly the data could benefit some filtering of the boundary events. We proceed to create a boundary filter to remove the boundary events from our dataset.

A boundary filter can be used to remove events at the boundaries for each channel.The data after excluding the boundary events is stored as a *list* of *flowSet*s

```
> createBoundaryFilterList <- function(flowSet) {
+     len <- length(colnames(flowSet))
+     tmp <- fsApply(flowSet, range)
+     tmp <- lapply(tmp, function(x) {
+         x[[colnames(x)[len]]] <- NULL
+         x
+     })
+     res <- lapply(tmp, function(y) {
```

```
+         apply(y, 2, function(x) {
+             c((x[1] + 2 * .Machine$double.eps), (x[2] - 2 * .Machine$double.eps))
+         })
+     })
+     filtList <- lapply(res, function(x) {
+         rectangleGate(filterId = "boundary", .gate = x)
+     })
+     return(filtList)
+ }
> boundData <- list()
> for (i in seq_len(length(tData))) {
+     wfNew <- workFlow(tData[[i]], name = "panel")
+     filtList <- createBoundaryFilterList(Data(wfNew[["base view"]]))
+     flt <- filterList(x = filtList, filterId = "boundary")
+     add(wfNew, flt)
+     boundData[[i]] <- Data(wfNew[["boundary+"]])
+     rm(wfNew)
+     cat(i)
+     cat(".")
+ }
```

4.3 Data normalization

When data from different aliquots are compared, shifts in flourescence intensities can be observed for the same fluorescence dye. Biologically there should have been no differences as the cell types came from a single sample from the patient. These shifts need to be corrected for before proceeding with any gating. Additionally, the QA processes for density,ECDF plots etc relies on the distance between the plots from the same patient to identify patient panels that are potential outliers. Proper alignment of data from different aliquots is necessary and can be done using the `warpSet` function. The function normalizes data based on landmarks estimated from high density regions in the data.

The parameters that are duplicated across the aliquots for each patient can be identified using the `locateDuplicatedParameters`. For each patient, the data obtained from flourscence dyes CD8, CD27 and CD4 for each of the eight aliquots are normalized so that the peaks in flow parameters for each patient aligns up between aliquots.

```
> library(flowStats)
> patientID = sampleNames(boundData[[1]])
> ls <- length(patientID)
> nData <- normalizeSets(flowList = boundData, dupes = c("CD8",
+     "CD27", "CD4"))
```

4.4 ECDF plots

Emperical cumulative distribution function gives the probability that a randomly picked sample is less than or equal to its value. ECDF plots can easily reveal differences in the distribution

of the flow cytometry parameters although they do not reveal much information regarding the underlying shape of the distribution.

The `qaProcess.ECDFPlot` produces ECDF plots of the flow cytometry parameters. The ECDF plots are grouped by patient so that data from the eight aliquots for each patient appear together in one plot, thereby allowing direct comparison of any differences in their distribution.The cytometer channels from which the data was obtained are also displayed in the legend.

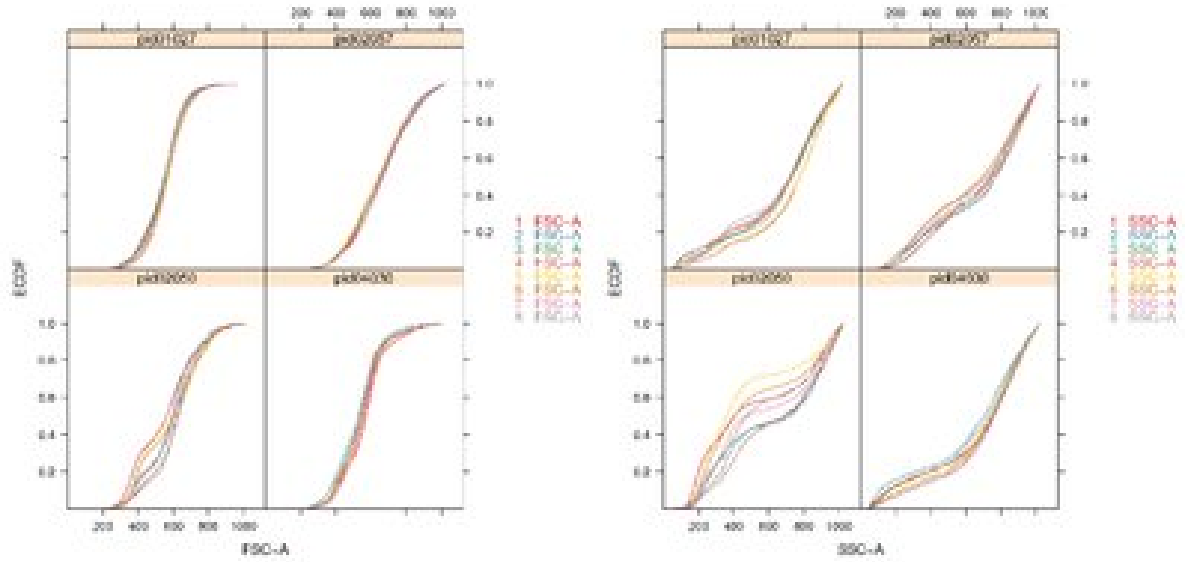
To identify patient panels that have highest variation amongst the ECDF plots for a particular parameter, pairwise KL distances between values from each aliquot were estimated. For each patient, the flourescence parameter distances were then summed up. This yields a measure that could be useful for better comparison of the magnitude of differences. Univariate outlier detection was performed on the normalized distance values to identify patient panels where ecdf plots are different from each other.

```
> dyes <- c("FSC-A", "SSC-A")
> resFSCECDF <- qaProcess.ECDFPlot(nData, dyes = dyes, outdir = dest,
+     alpha = 0.4, pdf = TRUE)

creating summary plots.............
creating frame plots...............

   Patient Parameter Passed Distance
1 pid02050     FSC-A  FALSE 0.068040
2 pid04030     FSC-A   TRUE 0.030310
3 pid01027     FSC-A   TRUE 0.025130
4 pid02057     FSC-A   TRUE 0.006143
5 pid02050     SSC-A  FALSE 0.092800
6 pid04030     SSC-A   TRUE 0.023330
7 pid01027     SSC-A   TRUE 0.035460
8 pid02057     SSC-A   TRUE 0.021560
```

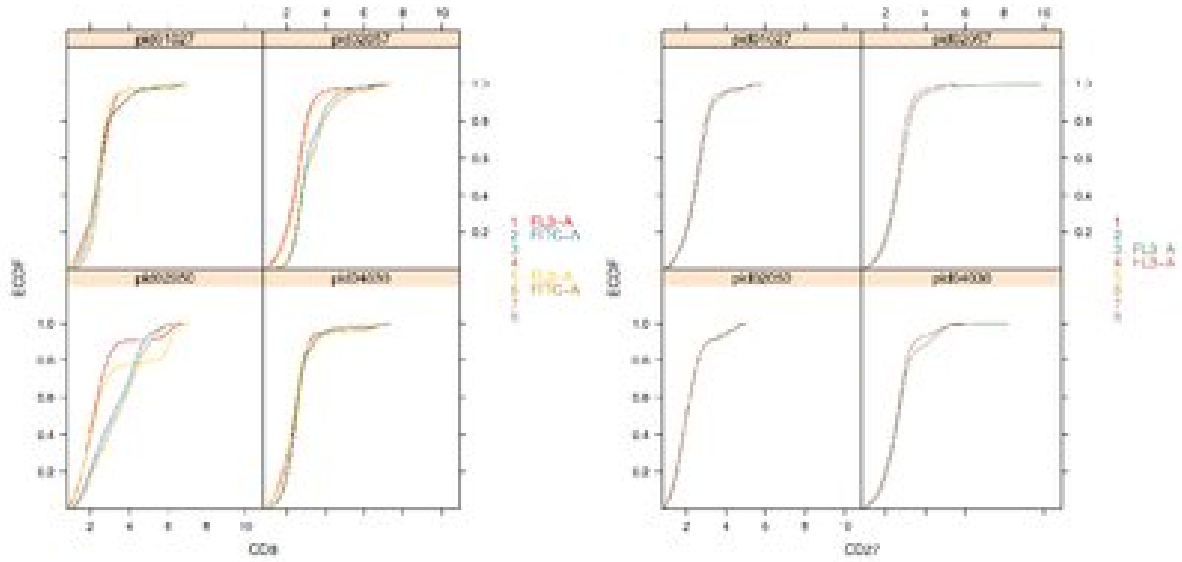
ECDF plots for forward and side scatter is shown below.




```
> resCD8ECDF <- qaProcess.ECDFPlot(nData, dyes = c("CD8", "CD27"),
+     outdir = dest, alpha = 0.4, pdf = TRUE)
```

```
   Patient Parameter Passed Distance
1 pid02050       CD8  FALSE  0.61010
2 pid04030       CD8   TRUE  0.12820
3 pid01027       CD8   TRUE  0.18300
4 pid02057       CD8   TRUE  0.28810
5 pid02050      CD27   TRUE  0.01182
6 pid04030      CD27   TRUE  0.05562
7 pid01027      CD27   TRUE  0.03593
8 pid02057      CD27  FALSE  0.22380
```

ECDF plots for dyes CD8 and CD27 are shown below



Each line in the ECDF plot for a patient is plotted in a different color corresponding to the aliquot from which the sample was selected. The distance between the ECDF lines for each patient indicates how different the data from each aliquots are. If the flourescence parameters are similar across the aliquots the ECDF lines would be close together(example FSC-A for "pid01027").If the aliquot intensities were different, then the ECDF plots for the aliquots would diverge out.(example SSC-A for "pid02050")

The legend on the right corner of each plot also contains information regarding the channel from which the fluorescence intensities were recorded. If a particular dye is absent from an aliquot, the corresponding legend name entry for channel information is left empty.

From the ECDF plots it appears that the forward/side scatter as well as the CD8 intensities for patient "pid02050" appears to be different from the rest of the group indicating some problem with the experimental or data collection procedure.However, the intensity for dye CD27 for patient "pid02050" appears similar to the rest of the group. These results are also confirmed by higher distance measures calculated for patient "pid02050" for parameter SSC-A, FSC-A and CD8. A larger number for the distance measure indicates a greater difference between the flourescence intensities from the aliquots.

4.5 Density plots

Density plots reveal useful information regarding the underlying shape of the distribution. They describe the probability of a variable being at a specific value.

Density plots can be produced by the `qaProcess.DensityPlot` function. Each panel in the plot produced represents data from a patient. The density data for parameters from each aliquot are represented in a different color. Additionally, the channels from which they were acquired are shown in the legend.

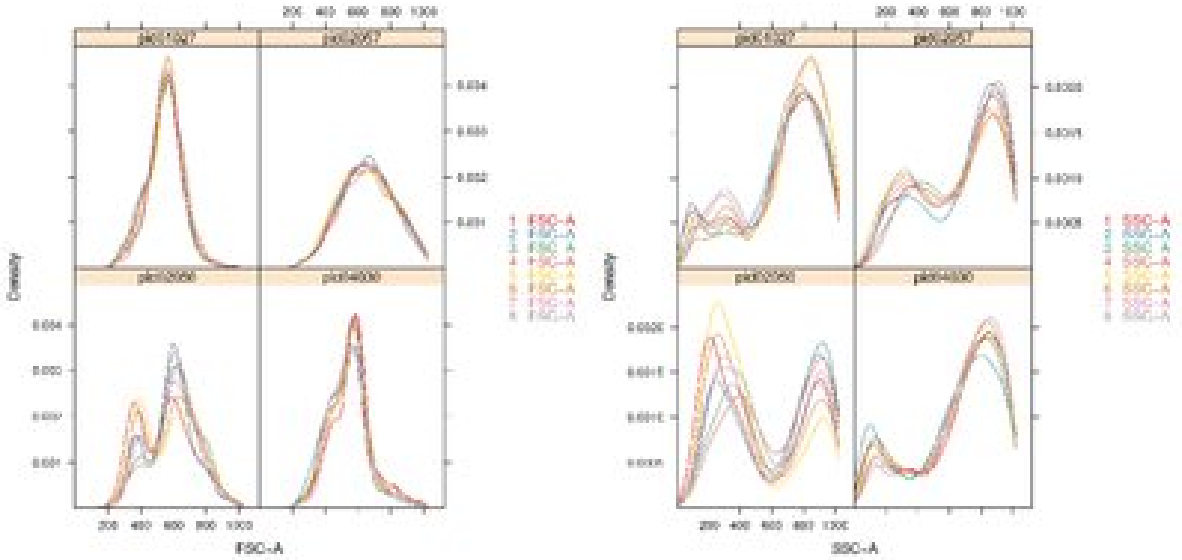
For each flourescence parameter from a patient, pairwise KL distances between the data from the aliquots were computed by binning the parameter values for each aliquot. Univariate outlier detection was performed on the sum of the pairwise distances to identify patient panels where density plots are different from the rest of the group.

Density plot for forward/side scatter can be generated by passing these arguments to the dyes input of the `qaProcess.densityPlot`

```
> resDensityFSC <- qaProcess.DensityPlot(nData, dyes = c("FSC-A",
+     "SSC-A"), outdir = dest, alpha = 0.2, pdf = TRUE)
```

```
   Patient Parameter Passed Distance
1 pid02050     FSC-A   TRUE 0.068040
2 pid04030     FSC-A   TRUE 0.030310
3 pid01027     FSC-A   TRUE 0.025130
4 pid02057     FSC-A   TRUE 0.006143
5 pid02050     SSC-A  FALSE 0.092800
6 pid04030     SSC-A   TRUE 0.023330
7 pid01027     SSC-A   TRUE 0.035460
8 pid02057     SSC-A   TRUE 0.021560
```

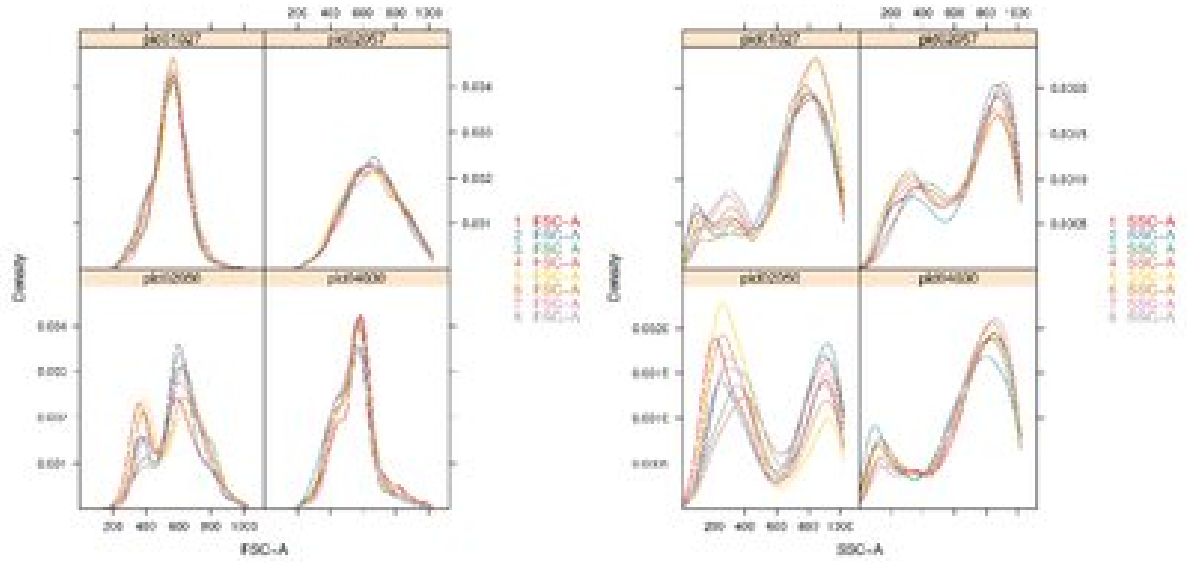
The density plots for forward/side scatter are show below.




```
> resDensityCD8 <- qaProcess.DensityPlot(nData, dyes = c("CD8",
+     "CD27"), outdir = dest, alpha = 0.2, pdf = TRUE)
```

■genTbl2Dens,echo=false■= imagePath<-resDensityCD8@summaryGraph@fileNames[2] getDistance(resDensityCD8,c("CD8","CD27"))

The density plots for dyes CD8 and CD27 are show below.



As with the ECDF plots, the closer the density plot lines are for a particular patient, the smaller the difference is between the aliquots. From the FSC/SSC density plots, the density lines appears to be most further apart for patient "pid02050". For patient "pid02050" clearly distinct bimodal distribution is observed from the FSC-A density plot while the other patients have a unimodal distribution The calculated distance measure was also found to be highest in patient "pid02050" for parameter FSC-A.

A difference in the shape of the distribution for patient "pid02050" can also be observed from the CD8 density plot. This result can also be observed in the calculated distance measures for parameter CD8

While the FSC-A, SSC-A and CD8 plots for patient "pid02050 appear to be significantly different from the rest of the group, the density plots for CD27 look very similar for all the patients as indicated by the close overlap of the density lines. Another observation from stain CD8 is that the two density plot lines for channel FITC-A lines up together and appears to be different from that for FL3-A channel even though they represent intensities from the same dye. Perhaps this could be caused by lack of proper compensation for spectral overlap.

The legend for the plot indicates the channel from which the intensities were recorded as well as the aliquot from which the sample was obtained. For stains that are absent from an aliquot, the corresponding channel is left empty in the legend.

4.6 2D Summary statistics

Two dimensional summaries often provide additional information not available in single dimensional approaches such as the density plot as they display information regarding the consolidated distribution of parameters.Parameters pairs such as forward and sideward scatter occur in measurements from all the aliquots.

Statistical measures such as the mean, median etc can be calculated for each parameter and can be represented in the form of a scatter plot for each patient. The summary statistic of the flourescence parameter pair from each aliquot represents a point in the scatter plot panel of each patient.

The `2DStatsPlot` generates scatter plots of summary statistics of fluorescence parameter pairs occurring in each aliquot. Any parameter pairs in an aliquot that occur in more than once in the *list* of *flowSets* can be passed as input to this function.

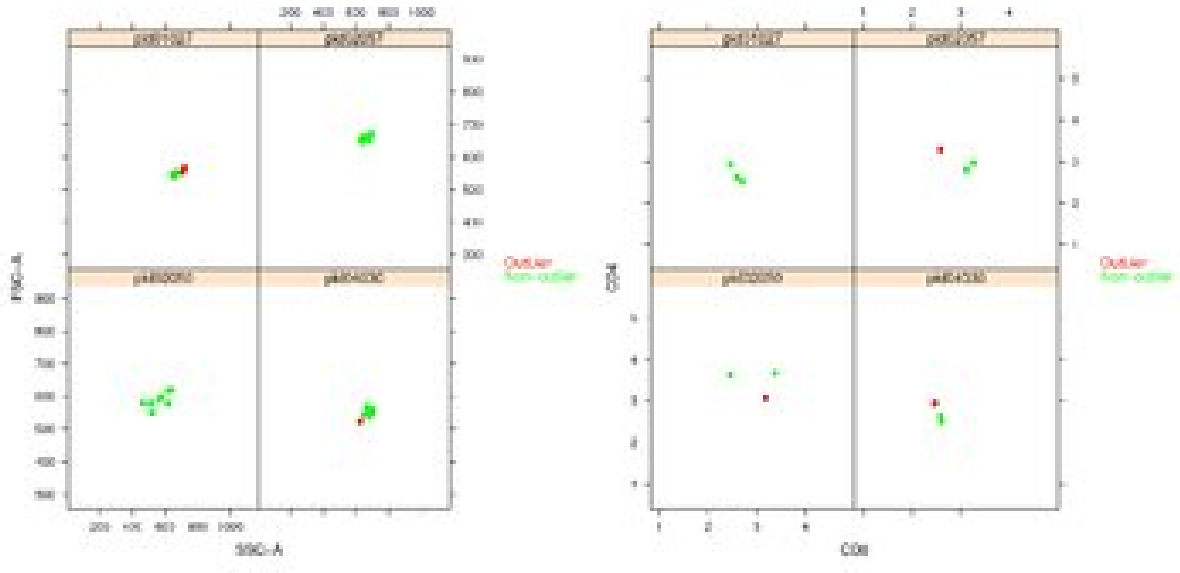
Two dimensional outlier detection is performed on the scatter plot panels for each patient to identify aliquots with the summary statistic measure different from the rest. The summary plot on the top of the HTML report shows the an overview of the selected summary statistic measure for each patient. The outliers are displayed in red.

The scatter plot for each parameter pair can also be observed in horizontal panels for each patient. For each detailed patient panel,the fluorescence channel from which the parameter was recorded as well as the aliquot number are displayed in the legend. The outlier aliquots are marked with a red dot.

The mean flourescence intensities for a pair wise combination of parameter values of "FSC-A","SSC-A","CD4" and "CD8" can be generated by passing them as input to the `qaProcess.2DStatsPlot`

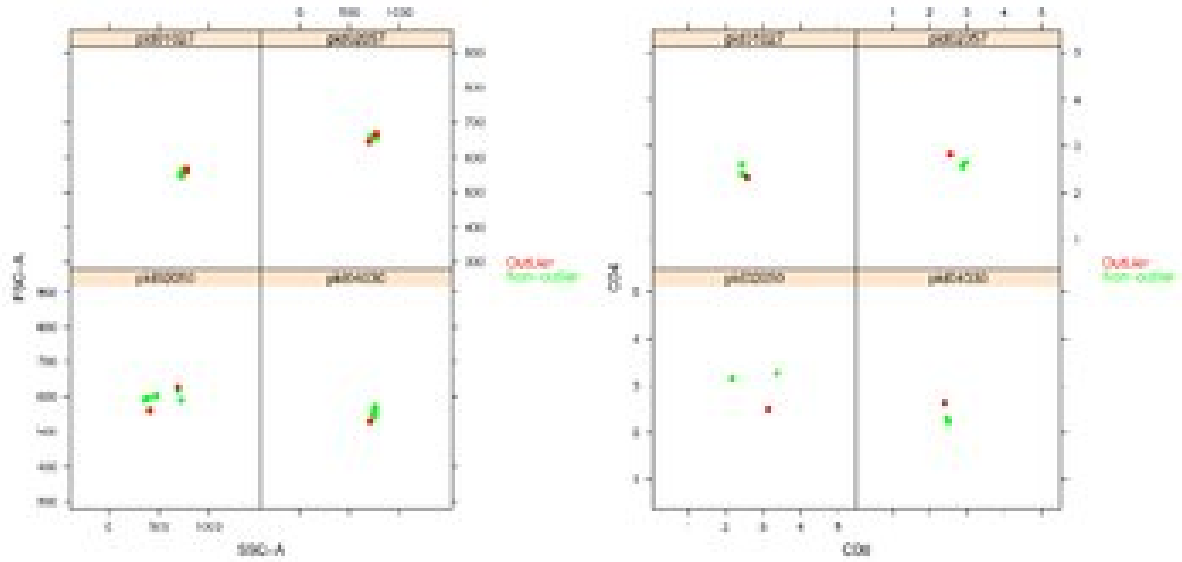
```
> par <- c("FSC-A", "SSC-A", "CD4", "CD8")
> resMean <- qaProcess.2DStatsPlot(nData, dyes = par, outdir = dest,
+     func = mean, outBound = 0.28, pdf = TRUE)
> imagePath <- resMean@summaryGraph@fileNames[2]
```

The scatter plot of mean FSC/SSC and CD4/CD8 value pairs are shown in the figure below.



```
> par <- c("FSC-A", "SSC-A", "CD4", "CD8")
> resMedian <- qaProcess.2DStatsPlot(nData, dyes = par, outdir = dest,
+     func = median, outBound = 0.28, pdf = TRUE)
> imagePath <- resMedian@summaryGraph@fileNames[2]
```

The scatter plot of median FSC/SSC and CD4/CD8 value pairs are shown in the figure below.



For the mean and median summary plots from each patient, if the data from the eight aliquots were similar, we would expect the dots to cluster together. We expect an aliquot with an experimental artifact to be spread out from the rest of the group. These outliers can then be identified by two dimensional outlier detection method. Outliers in the plot are marked in red color.

From the mean and median summary plots for FSC-A/SSC-A, patient "pid02050" seems to have data most scattered, indicating differences between the aliquots from which the parameters were recorded. Data from outlier aliquots most distant from the rest of the points for each patient is shown in red. The outlier aliquot was well as the channel from which it was acquired can be identified by going into the detailed patient panel information presented in the HTML report generated by the `writeQAReport`

4.7 Kullback-Leibler distance plots

The Kullback-Leibler Information between densities f1 and f2 is defined as

$$KLI = \int \log \frac{f_1(x)}{f_2(x)} f_1(x) dx \tag{1}$$

Density estimation followed by numerical integration can be used to calculate the pairwise KL distance between the parameters values for each aliquot to identify patient panels that are potentially different from the rest. However such an approach may be computationally intensive, especially for flow cytometry data.

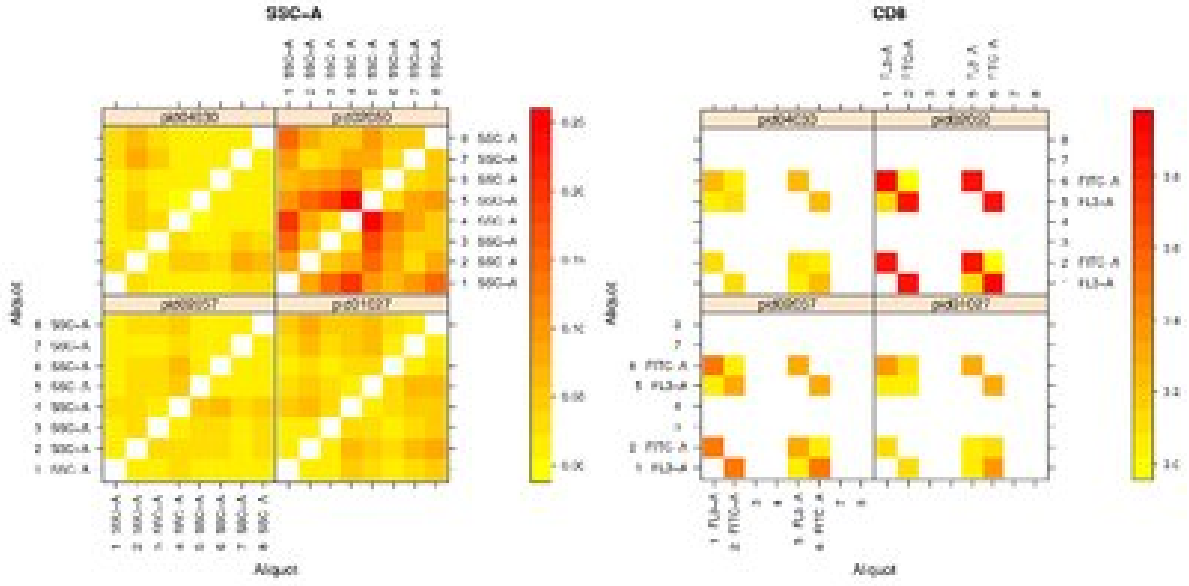
An alternative approach is to bin the flourscence data to obtain an estimate of density. The estimated density values can then be used to compute pair wise KL distances for the aliquots from which data was obtained.The aliquots which were not stained for a particular parameter are labelled as missing.

For each patient , the pair wise KL distances can then be visualized by a plot with a color scale representing the value for the distance matrix. Aliquots that are very similar to each other will have a lower color intensity on the color scale.

The KLDistance plot can be generated using the function `qaProcess.KLDistPlot`

```
> resKLdist <- qaProcess.KLDistPlot(nData, dyes = c("SSC-A", "CD8"),
+     outdir = dest, alpha = 0.05, pdf = TRUE)
> writeQAReport(nData[[1]], list(resKLdist), outdir = dest)
> imagePath <- resKLdist@summaryGraph@fileNames[2]
```

The KL Distance for the aliquots for parameters "SSC-"A and "CD8" are shown below.



When KL distance plots from several patients are plotted side by side with the same intensity axis for the color, we can draw conclusions regarding which which patient panel has aliquots most similar to each other as well as which patient exhibited the largest difference in aliquots. For the SSC-A plot above it is clear that patient "pid02057" has the least color intensity and "pid02050" has the highest intensity. From the CD8 plot, it is very clear that patient "pid02050" has higher color intensities and is different from the rest of the group.These result are consistent with what has been observed in the ECDF plots for these parameters discussed earlier.

KL distance plots are more useful in identifying differences between aliquots within a patient. From the CD8 dye plot for patient "pid02050", the highest intensities occur for comparisons between aliquot pairs (1,6) , (1,2) and (5,6) and (5,2). It is intersting to note that dyes 1 and 5 were obtained from the FL3-A channel while aliquots 2,6 were obtained from the FITC-A channel. These difference in patient "pid02050" are more evident when the KL distance plots are viewed simultaneously with the ECDF plots for this particular patient.

4.8 Conclusions

The visualization techniques presented in this package provide different views of the underlying statistical properties of the data. A combination of the visualization techniques could help the user identify deviant samples. The visualization tools provided by the flowQ package could potentially help identify samples differences that are probably not biologically motivated and

hence could be further investigated before spending time and resources for gating and detailed analysis.