Flow cytometry data analysis using Bioconductor - A typical work flow.

Nishant Gopalakrishnan

July 23, 2009

1 Introduction

The advent of high throughput techniques has resulted in the generation of large flow cytometry (FCM) data sets. Along with the task of acquiring the data comes the task of storing, managing, quality control, data analysis and data summarization to a condensed form that can be interpreted by the researcher.

Open source Bioconductor packages for analysis of flow cytometry data provides a unified framework for bioinformaticians to develop methods to analyze and interpret flow cytometry data. The packages relevant to flow cytometry include flowCore, flowViz, flowStats, flowQ, flowUtils, and flowClust.

- flowCore - handles importing, storing, preprocessing and assessment of data from flow cytometry experiments.
- flowViz - provides graphical methods for visualization of flow cytometry data.
- flowQ - provides quality control and quality assessment tools for flow cytometry data.
- flowStats - provides tools and methods to analyze flow cytometry data that is beyond the basic infrastructure provided in the flowCore package.
- flowUtils - provides utilities, mainly to integrate foreign FCM analysis tools.
- flowClust - implements mixture model based clustering algorithms for FCM data.

This document outlines a typical work flow for flow cytometry data analysis using the infrastructure provided by the aforementioned Bioconductor packages. Our sample data set consists of 14 patient samples from 2 different groups treated with either drug A and B. Each sample has been stained for CD3,CD4,CD8, CD69 and HLADR.

We begin by reading in flow cytometry data from raw FCS files, storing them in appropriate data structures and performing quality control checks on the data. We proceed to identify sub populations of helper and cytotoxic T lymphocytes using sequential gating strategies based on the staining information provided. Our goal is to calculate the proportion of helper and cytotoxic T cells that exhibit the HLADr activation marker amongst the two groups of patients in our data set.

2 Importing flow cytometry data

Flow cytometry data is typically saved in files with FCS or LMD (List mode data) extensions, with each file corresponding to a sample/tube from a single patient. Depending on the complexity of the experimental design, a flow cytometry data set could potentially consist of hundreds of FCS files.

FCS or LMD files can be read into an R environment using the `read.FCS` (for individual FCS files) or `read.flowSet` function (for a set of FCS files) provided in the flowCore package. We first proceed to load up the required libraries that we will be using for our data analysis procedure.

```
> library(flowCore)
> library(flowQ)
> library(flowViz)
> library(flowStats)
```

3 Data structures for flow cytometry data

The package flowCore has several data structures implemented for storing and manipulating flow cytometry data. The basic container for storing flow data in the flowCore package is a *flowFrame*.

The *flowFrame* data structure has three slots

- exprs slot stores a matrix containing fluorescence intensity information. The column names of the matrix correspond to the parameter names and each row of the matrix corresponds to a single recorded event. The exprs slot can be manipulated using the `exprs` function.
- parameters slot stores an annotated data frame containing information regarding the parameters (i.e., the stains) that were recorded by the flow cytometer. The parameters slot can be manipulated using the `parameters` function.
- description slot stores a list containing all the information that was produced by the instrument during the measurement, e.g. the FCS file version, fluorescence parameters, system the sample was acquired on, etc.

Flow cytometry experiments typically involve data from several patients. It is useful to have data from an experiment organized together along with the metadata information. The flowCore package provides the `flowSet` container that organizes several *flowFrame*s together.

In more complex multi-panel experiments, samples from patients are often divided into several aliquots. The aliquots are then stained using antibody-dye combinations that are specific to certain antigens presented on the cell surface or for specific intracellular markers. In such cases it is convenient to represent the data as a list of *flowSet*s, each element in the list corresponding to data recorded from a single aliquot.

4 Importing our example data set into a flowSet

For our example, we proceed by reading in FCS files from 14 patients and creating a flowSet. We assume that the files provided are available in the working directory.

To read in the flowSet we make use of the `read.flowSet` function. The phenoData information is provided as a tab delimited file. The file annotation.txt file contains a matrix with row names as samples and variables as columns.

```
> flowData <- read.flowSet(path = ".", phenoData = "annotation.txt",
+     transformation = FALSE)
> sampleNames(flowData) <- as.character(pData(flowData)[, "PatientID"])
```

The name column in the parameters slot of each *flowFrame* contains the name of the fluorescence channels of the cytometer from which the data were obtained. The description column in the parameters slot of each *flowFrame* contains information regarding the dyes used for each sample. In addition, some information about the data range of the respective channels and the bitwidth is provided.

```
> pData(parameters(flowData[[1]]))
```

```
        name          desc range minRange maxRange
$P1    FSC-A          <NA>  1024        0     1023
$P2    SSC-A          <NA>  1024        0     1023
$P3   FITC-A   CD8 FITC-A  1024        1    10000
$P4     PE-A    CD69 PE-A  1024        1    10000
$P5    FL3-A           CD4  1024        1    10000
$P6 PE-Cy7-A CD3 PE-Cy7-A  1024        1    10000
$P7    APC-A  HLADr APC-A  1024        1    10000
$P8     Time          <NA>  1024        0     1023
```

In a clinical data collection process, these information fields may not always be updated with the appropriate fields especially when the samples have been run by different lab personnel or when an error was made in the data entry process. In such cases, the corresponding fields can be updated by using the `pData` and `parameters` methods. We proceed to modify the description slot to remove the duplicated fluorescence channel information. The description fields for parameters like forward/side scatter and Time have been updated with NA, since they are not associated with a specific staining marker.

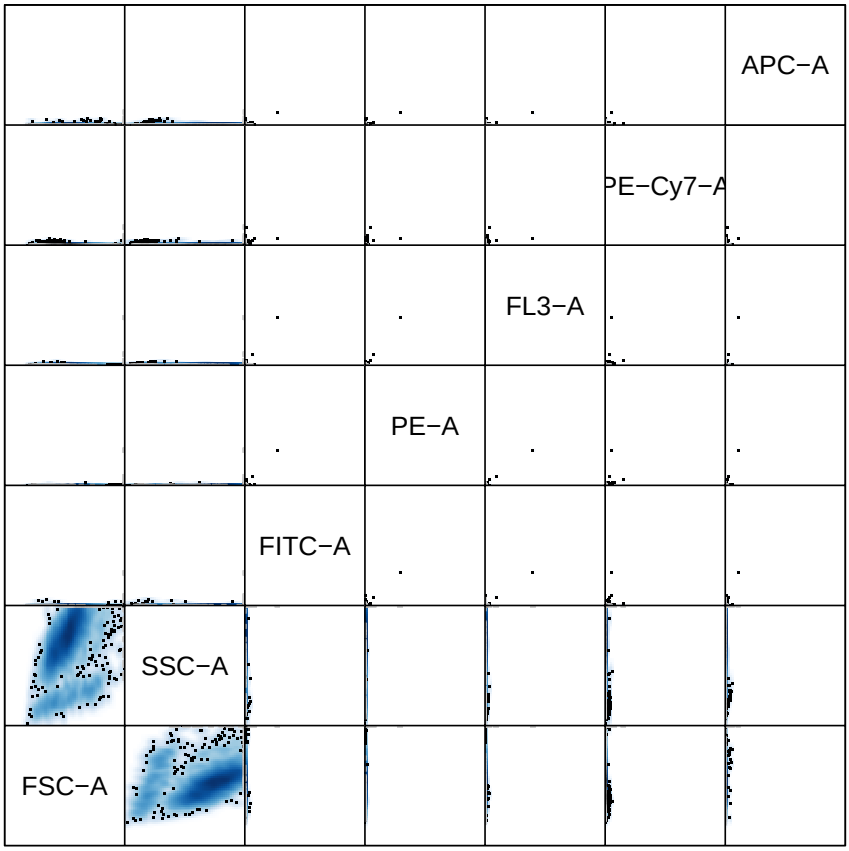
```
> for (i in seq_len(length(flowData))) {
+     pData(parameters(flowData[[i]]))[, "desc"] <- c("NA", "NA",
+         "CD8", "CD69", "CD4", "CD3", "HLADr", "NA")
+ }
> flowData[[1]]

flowFrame object 'pid349'
with 4000 cells and 8 observables:
```

```
        name  desc range minRange maxRange
$P1     FSC-A    NA  1024        0     1023
$P2     SSC-A    NA  1024        0     1023
$P3    FITC-A   CD8  1024        1    10000
$P4      PE-A  CD69  1024        1    10000
$P5     FL3-A   CD4  1024        1    10000
$P6  PE-Cy7-A   CD3  1024        1    10000
$P7     APC-A HLADr  1024        1    10000
$P8      Time    NA  1024        0     1023
98 keywords are stored in the 'descripton' slot
```

5 TransformData

The package flowViz has several functions available for visualizing flow cytometry data. A scatter plot matrix of all the parameters of a *flowFrame* can be obtained by making use of the `splom` function. The scatter plot matrix for a *flowFrame* is displayed below.

```
> print(splom(flowData[[1]]))
```



Scatter Plot Matrix

From the figure it is clear that the channels FITC-A, PE-A, FL3-A, PE-CY7-A and APC-A need some form of transformation for better visualization of the data. The transformations implemented in the flowCore package are listed below.

truncateTransform $y = \begin{cases} a & x < a \\ x & x \geq a \end{cases}$

scaleTransform $f(x) = \frac{x-a}{b-a}$

linearTransform $f(x) = a + bx$

quadraticTransform $f(x) = ax^2 + bx + c$

lnTransform $f(x) = \log(x)\, \frac{r}{d}$

logTransform $f(x) = \log_b(x)\, \frac{r}{d}$

biexponentialTransform $f^{-1}(x) = ae^{bx} - ce^{dx} + f$

logicleTransform A special form of the biexponential transform with parameters selected by the data.

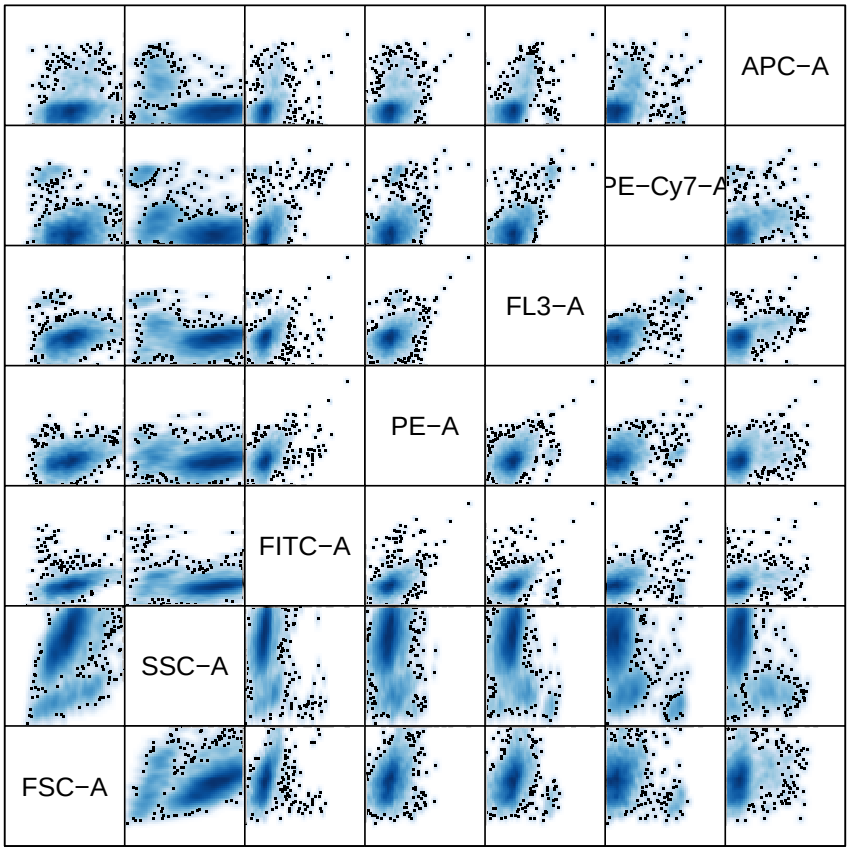
arcsinhTransform $f(x) = asinh(a + bx) + c$

We proceed to transform the channels FITC-A, PE-A, FL3-A, PE-CY7-A and APC-A using the `asinh` transformation.

```
> tData <- transform(flowData, transformList(colnames(flowData)[3:7],
+     asinh))
```

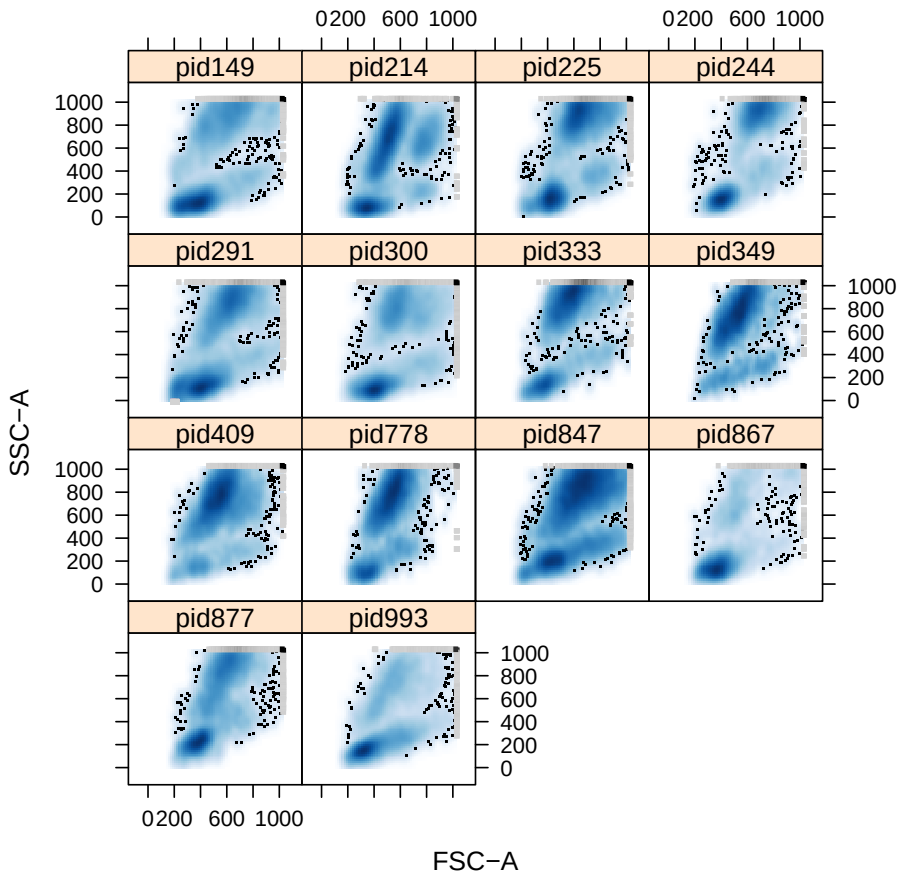
The transformed scatter plot matrix for the first *flowFrame* and the forward/side scatter plot for the flowSets forward/side scatter channels is displayed below.

```
> print(splom(tData[[1]]))
```



Scatter Plot Matrix

```
> print(xyplot(`SSC-A` ~ `FSC-A`, data = tData))
```



6 Data Quality assessment

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 several functions aimed towards quality control of flow cytometry data both at the level of single panel experiments as well as complex multi-panel experiment designs. In addition, the flowQ package provides infrastructure for generation summary html reports of the results generated.

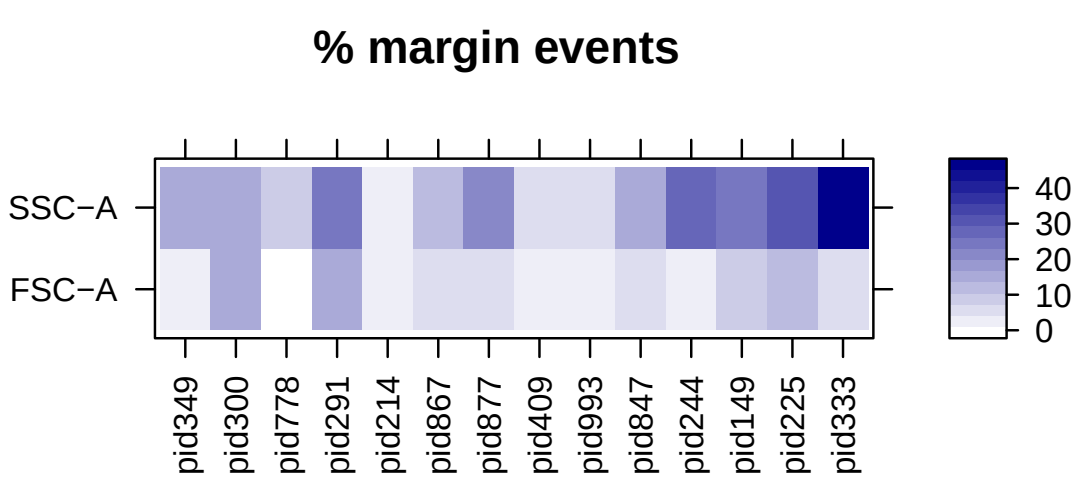
Since our data set contains only data from a single panel, we proceed to run some of the quality control functions for a single panel.

6.1 Boundary events

Fluorescence events that fall out of the dynamic range over which the flow cytometer acquires signal are recorded as margin events by recording them with minimum or maximum values

of the dynamic range. Events that accumulate at the boundaries should be removed before performing operations such as clustering or density estimations. The presence of boundary events in a flowSet can be detected by use of the `qaProcess.marginevents`. We proceed to detect boundary events in the forward and side scatter channels.

```
> dest <- file.path(tempdir(), "flowQ")
> qp1 <- qaProcess.marginevents(flowData, channels = c("FSC-A",
+     "SSC-A"), outdir = dest, pdf = TRUE, sum.dimensions = c(5,
+     3))
```



It is clear from the figure that the side scatter channel for patient pid333 has a large number of boundary events.The boundary events often need to be removed before proceeding with operations such as clustering. This can be done by use of the `boundaryFilter` function.

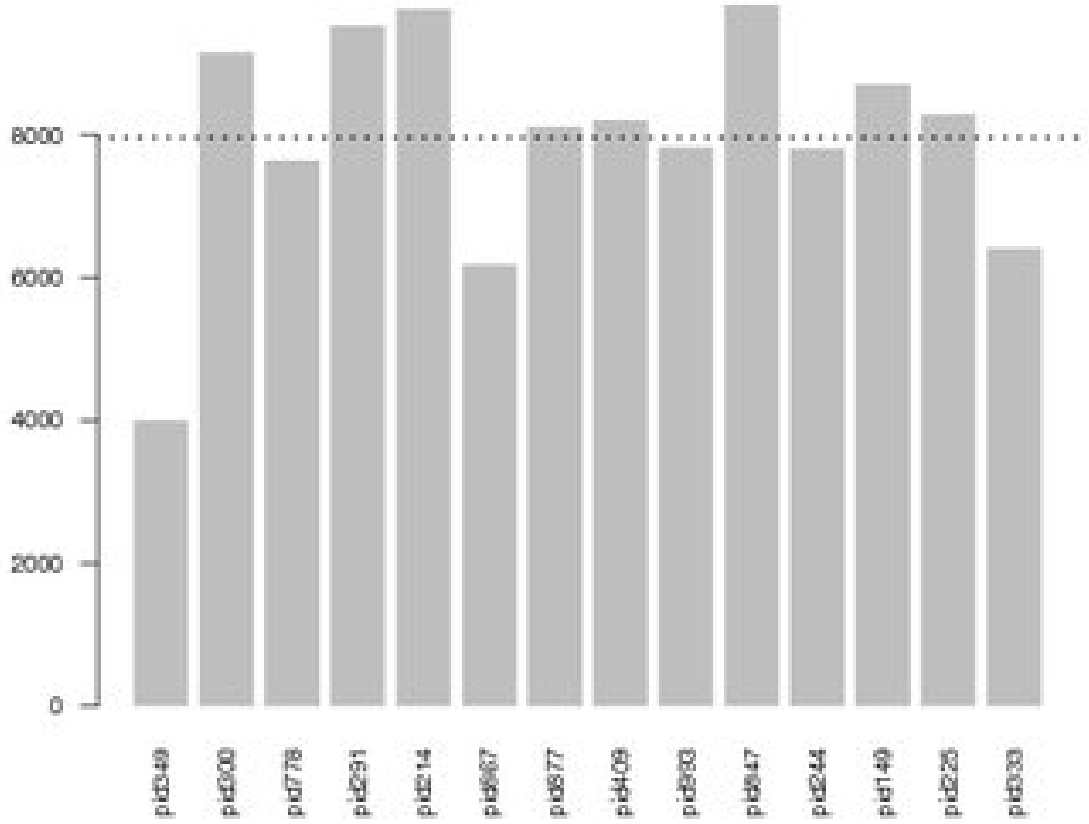
6.2 Cell number

The number of events per sample is expected to be comparable across *flowFrame*s in a *flowSet*. The `qaProcess.cellnumber` function can be used to identify outliers in a *flowSet* based on the distribution of event counts for the entire *flowSet*.Very high or low event counts when compared to the rest of the group can identify potential problems with data collection.

Results from our data set indicates that each sample has approximately the same number of events, except for sample pid349 which has less number of events when compared to the rest of the group.

```
> qp2 <- qaProcess.cellnumber(tData, outdir = dest, cFactor = 2,
+     pdf = TRUE)
```

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



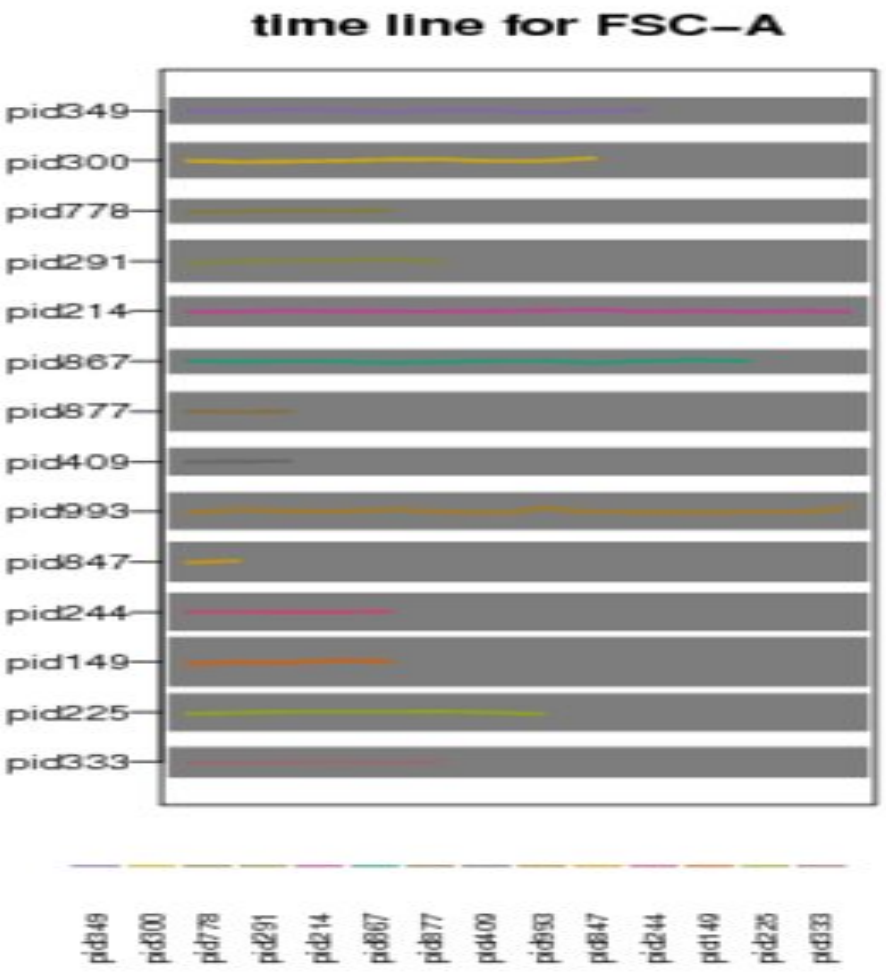
6.3 Time anomalies

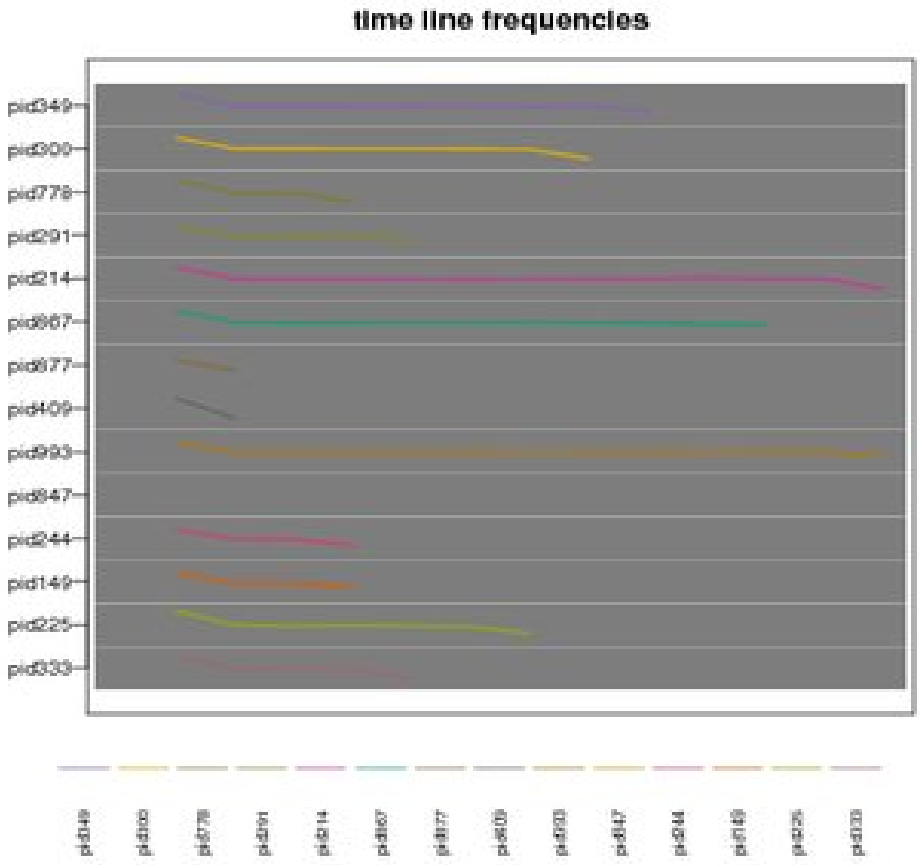
Events are recorded by the cytometer in a random manner and hence the average signal intensity is not expected to show variations over time. However problems with the data acquisition system that records fluorescence intensities could lead to a trend in florescence intensity over time. Additionally issues such as clogging of the flow tube or problems with the cell suspension could lead to trends in the number of events collected over time.

The `qaProcess.timeflow` and `qaProcesss.timeline` function can be used to detect disturbances in flow over time or drifts in the instrument over time. We proceed to identify such

issues by calling these functions on the forward scatter channel.

```
> qp3 <- qaProcess.timeline(tData, channels = "FSC-A", outdir = dest,
+     cutoff = 1, pdf = TRUE)
> qp4 <- qaProcess.timeflow(tData, channels = "FSC-A", outdir = dest,
+     cutoff = 2, pdf = TRUE)
```





For the timeline plots , we expect a horizontal lines for well behaved samples Sudden jumps, vertical lines or a trend over time could indicate problems during data acquisition.. The `timeFilter` class in flowCore can be used to remove problematic events from a *flowFrame*.

The timeflow plots visualize the acquisition rate over time, i.e., the number of events that were recorded in a given time tick.

7 Summary html reports

The `writeQAReport` can be used to combine the graphical outputs and information from the results of `qaProcess` functions into a single html report.

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

From the QA reports generated our data does not seem to have serious quality issues and hence we proceed with the detailed analysis of our *flowSet* making use of the *workFlow* infrastructure provided by the flowCore package to organize our analysis operations.

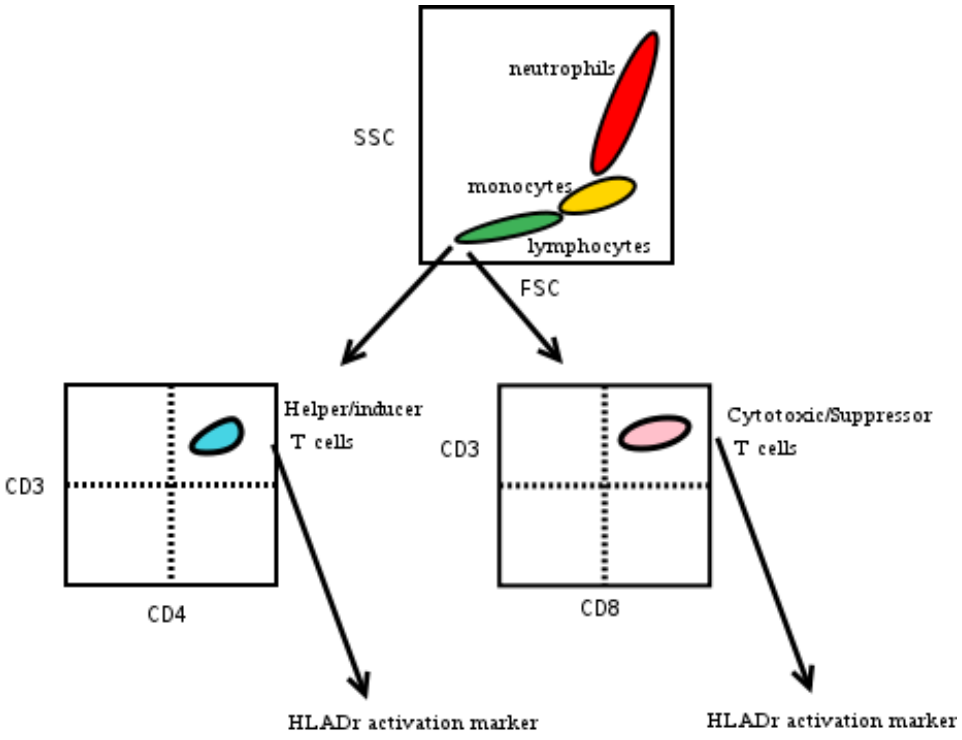


Figure 1: Sequential gating strategy for our sample data set

8 Detailed analysis of our data set

Our data set consists of 14 flowFrames consisting of patients from two groups that underwent treatments using two drugs A and B. Each sample has been stained for CD3,CD4,CD8, CD69 and HLADR. After the quality control process of identifying samples that could potentially have data differences that are not biologically motivated, we proceed with detailed analysis of our data set to identify sub population that are of interest.

We make use of workFlows implemented in the flowCore package to keep track of operations performed on our data set, organize the intermediate results generated as well as handle the naming schemes of intermediate variables that are created during the analysis.

Our goal in this data analysis process is to separate out the activated T helper/inducer cells and cytotoxic/suppressor T cells from the rest of the population and compare the results amongst the two groups of patients(Drugs A and B). The figure above summarizes our gating strategy for identifying the cells of interest.

```
> colnames(flowData) <- c("FSC", "SSC", "CD8", "CD69", "CD4", "CD3",
+     "HLADr", "Time")
> wf <- workFlow(flowData, name = "biocExample")
```

The data is first transformed for better visualization of data using the `asinh` transformation. We make use of the `transformList` to transform the fluorescence channels stained for CD8,

CD69, CD4, CD3 and HLADr. The transformList object created can then be added to the workflow to transform the corresponding channels for the flowSet included in the workflow.

```
> tf <- transformList(colnames(Data(wf[["base view"]]))[3:7], asinh,
+     transformationId = "asinh")
> add(wf, tf)
> wf

A flow cytometry workflow called 'biocExample'
The following data views are provided:

 Basic view 'base view'
 on a flowSet
 not associated to a particular action item

        View 'asinh'
         on a flowSet linked to
         transform action item 'action_asinh'
```

Before we proceed with the gating operations, we first remove the boundary events that we identified from our QA analysis for the FSC and SSC channels. This is done by use of the `boundaryFilter` function. We proceed to create a boundary filter object and add it to our workflow

```
> boundFilt <- boundaryFilter(filterId = "boundFilt", x = c("FSC",
+     "SSC"))
> add(wf, boundFilt, parent = "asinh")
```

8.1 T lymphocyte population

The first step in our gating strategy is to identify the lymphocyte subpopulation using the `lymphGate` fucntion. It selects elliptical cell subpopulations from two dimensional projections by fitting a bivariate normal distribution to a preselected rectangular area.

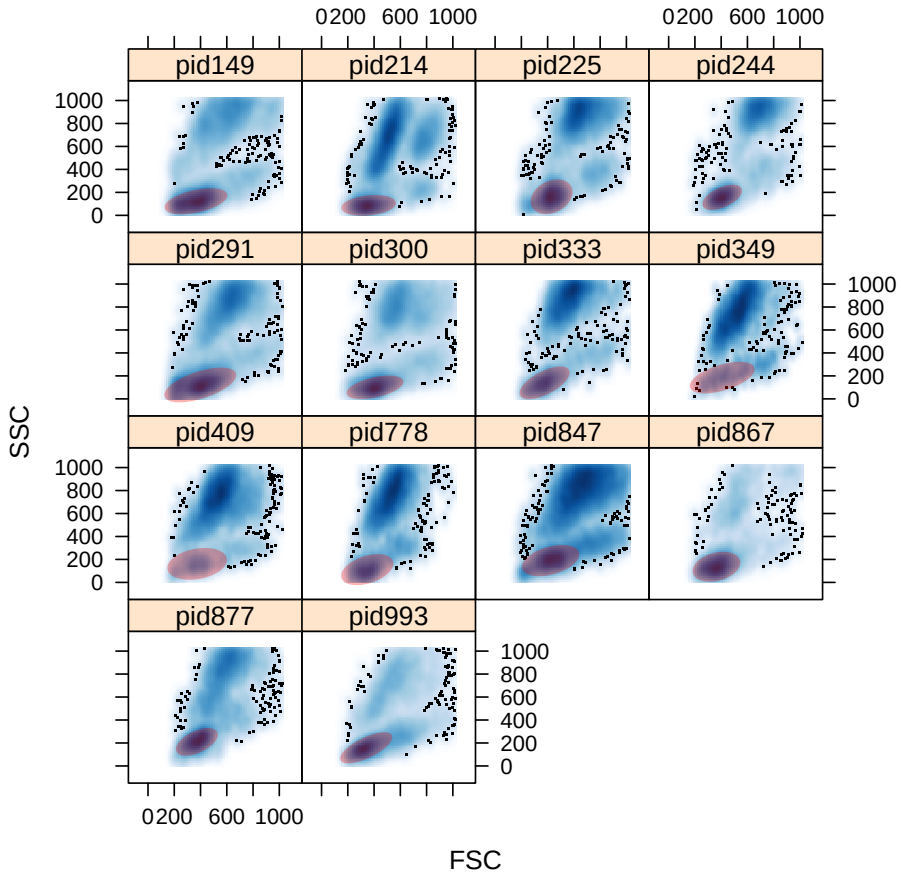
We make make use of the fact that the CD3 reagent binds specifically to the T lymphocytes for the preselection process. The `lymphGate` identifies a rectangular area containing the lymphocytes by gating areas that are CD3 positive and then fits a bivariate normal distribution to this selected area using the `norm2Filter` function.

After the lymph gate gets added to the workflow, our initial population gets split into two groups - one containing T lymphocytes (Tcells+) and the other containing all the other cell groups(Tcells-)

The T cell population selected by the lymph Gate is shown in the figure below

```
> lg <- lymphGate(Data(wf[["boundFilt+"]]), channels = c("FSC",
+     "SSC"), preselection = "CD3", filterId = "TCells", eval = FALSE,
+     scale = 2.5)
> add(wf, lg$n2gate, parent = "boundFilt+")
```

```
> print(xyplot(SSC ~ FSC | PatientID, wf[["TCells+"]], par.settings = list(gate = list(col =
+     fill = "red", alpha = 0.3))))
```



8.2 T helper inducer cells and Cytotoxic suppressor populations

The dyes used in flow cytometry are chosen so that they can distinguish between sub populations when used in combination. For our example, CD4 binds to helper/inducer T lymphocytes and monocytes. A combination of CD3/CD4 reagents can be used to separate the helper/inducer T cells from the rest of the T cells and monocytes (if any are included by use of a more liberal lymph gate).

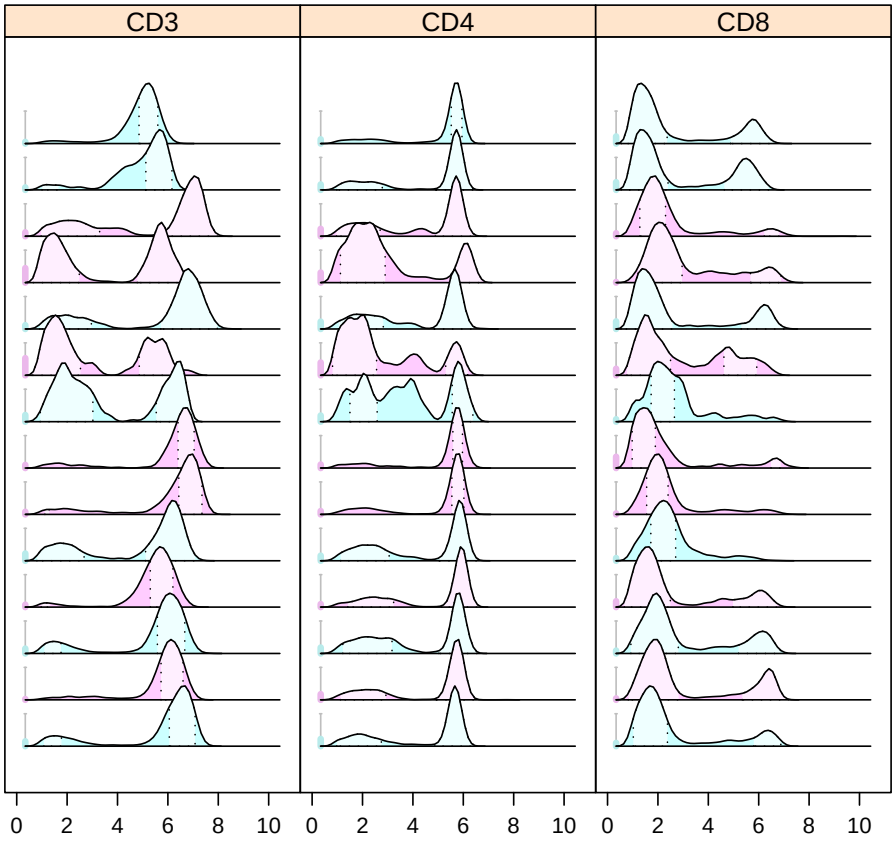
Similarly, CD8 binds to cytotoxic/suppressor cells and NK cells. A combination of CD3/CD8 can be used to separate the cytotoxic/suppressor T cells from the remaining T cells and NK cells.

8.3 Need for data normalization

The density plots for the CD3, CD4 and CD8 channels are shown below. Our goal is to identify the cell population groups that are CD3+/CD4+ and CD3+/CD8+. While this can

be achieved by making use of the `quadGate` function on each flowFrame, this process can be tedious as the gate dimensions required for each sample could vary considerably especially in case of CD3 from our example because the peaks in the density plot are not aligned.

```
> print(densityplot(PatientID ~ ., Data(wf[["TCells+"]]), channels = c("CD3",
+     "CD4", "CD8"), groups = GroupID, scales = list(y = list(draw = F)),
+     filter = lapply(c("CD3", "CD4", "CD8"), curv1Filter), layout = c(3,
+         1)))
```



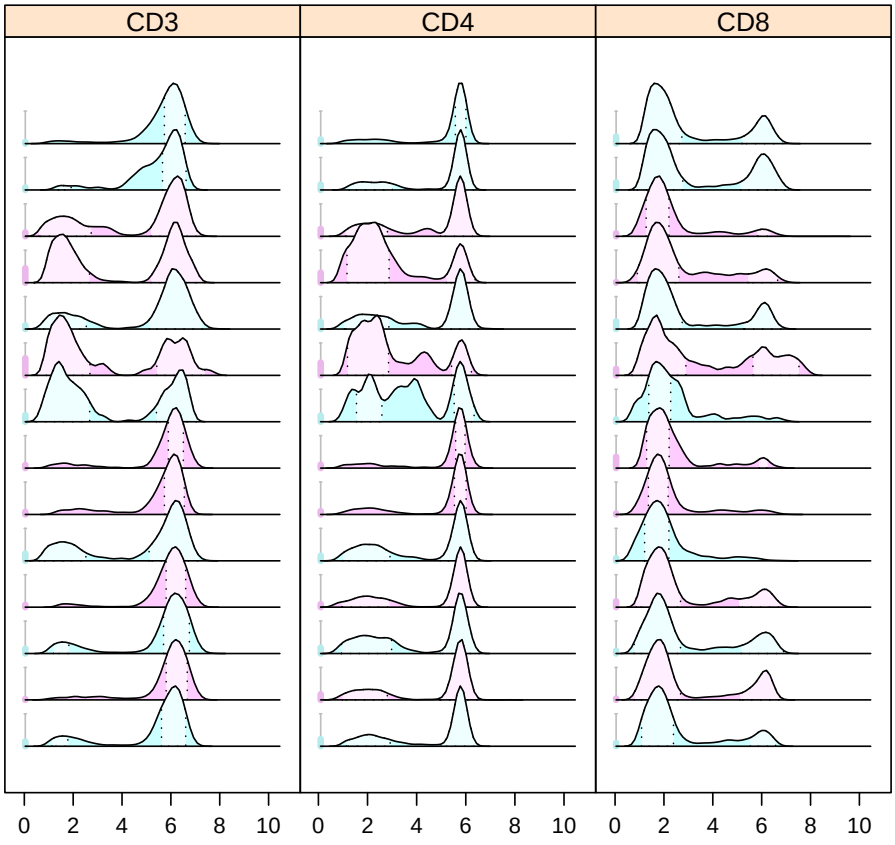
A single quad gate could be used if the data could be normalized so that the respective peaks align.

8.4 Data normalization

The `warpSet` function could be used to normalize the data based on the identification of high density areas (termed "landmarks") and the subsequent computation of appropriate transformation functions for each flowFrame so that the identified landmarks are aligned.

The density plots for the CD3,CD4 and CD8 fluorescence parameters after normalization is shown below.

```
> pars <- colnames(Data(wf[["base view"]]))[c(3, 4, 5, 6)]
> norm <- normalization(normFun = function(x, parameters, ...) warpSet(x,
+     parameters, ...), parameters = pars, normalizationId = "Warping")
> add(wf, norm, parent = "TCells+")

Estimating landmarks for channel CD8 ...
Estimating landmarks for channel CD69 ...
Estimating landmarks for channel CD4 ...
Estimating landmarks for channel CD3 ...
Registering curves for parameter CD8 ...
Registering curves for parameter CD69 ...
Registering curves for parameter CD4 ...
Registering curves for parameter CD3 ...

> print(densityplot(PatientID ~ ., Data(wf[["Warping"]]), channels = c("CD3",
+     "CD4", "CD8"), groups = GroupID, scales = list(y = list(draw = F)),
+     filter = lapply(c("CD3", "CD4", "CD8"), curv1Filter), layout = c(3,
+         1)))
```

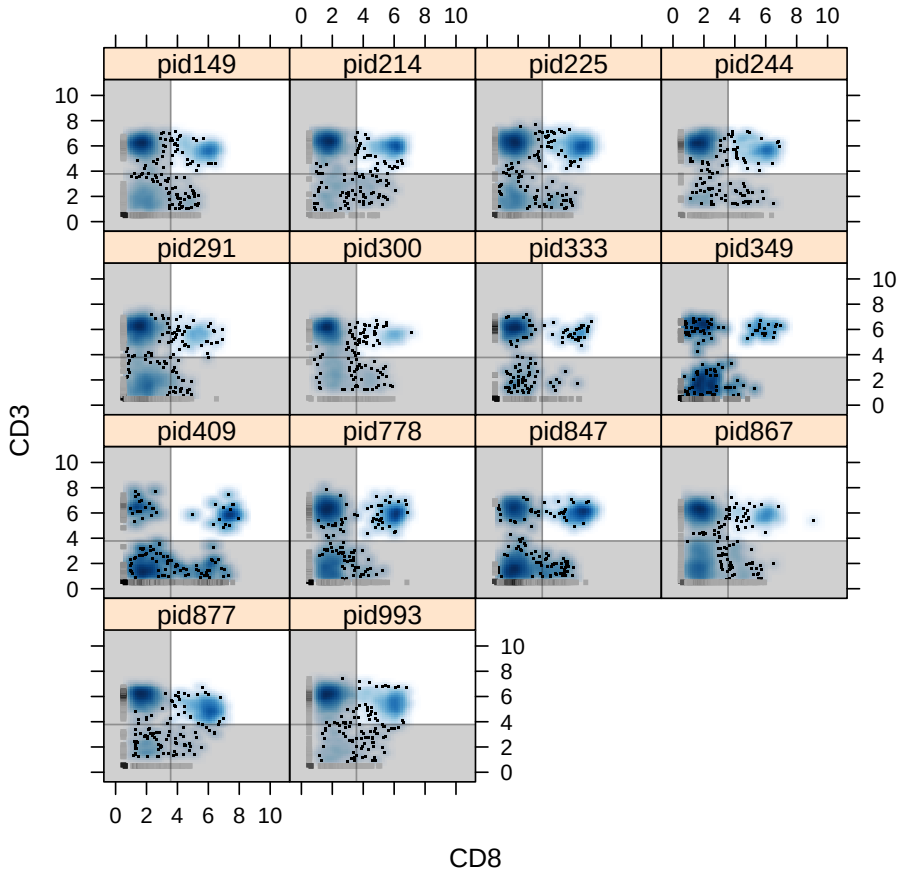


8.5 Quadrant gates to identify T helper/inducer cell and cytotoxic/suppressor T cell populations

The `quadrantGate` function can be used to separate two dimensional data into positive and negative quadrants based on the density estimates of the parameters under consideration. Essentially, this creates a single quadrant gate based on the joint data of all flowFrames.

We apply the quadrantGate function to the normalized data to identify the CD3+CD4+ sub population.

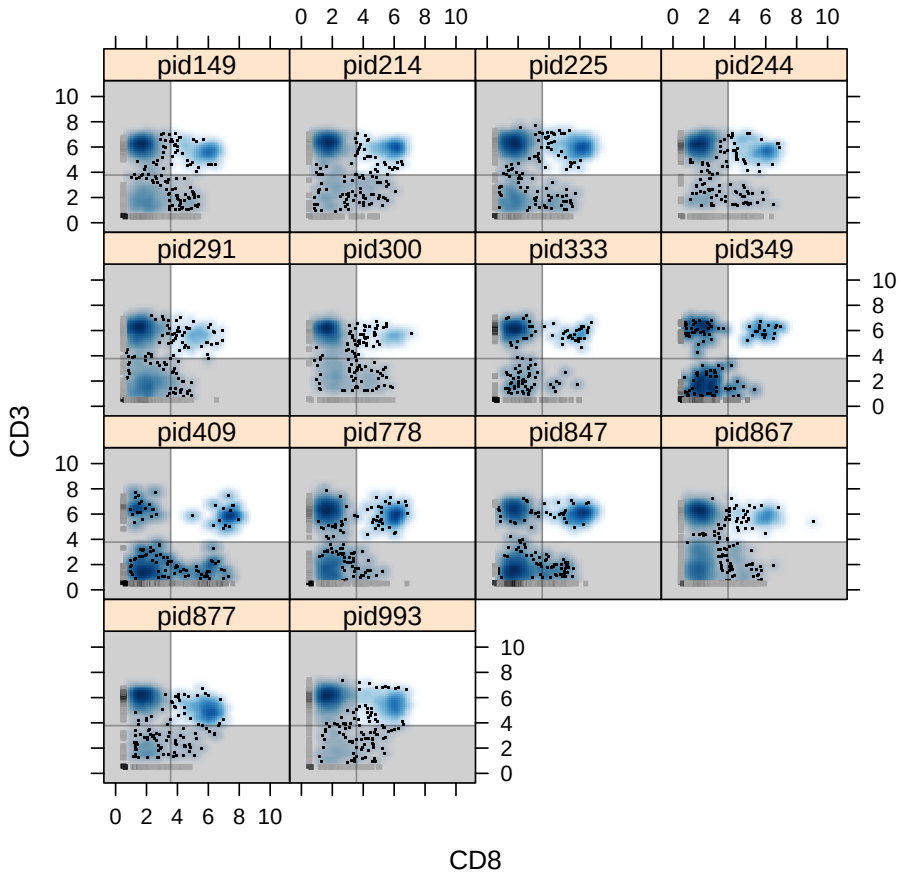
```
> qgate <- quadrantGate(Data(wf[["Warping"]]), stains = c("CD3",
+     "CD4"), plot = FALSE, filterId = "CD3CD4")
> add(wf, qgate, parent = "Warping")

> print(xyplot(CD3 ~ CD4 | PatientID, wf[["CD3+CD4+"]]))
```



Similarly, we apply the quadrantGate function to the CD3 and CD8 channels to identify the CD3+CD8+ population corresponding to the cytotoxic/suppressor T cells.

```
> qgate <- quadrantGate(Data(wf[["Warping"]]), stains = c("CD3",
+     "CD8"), plot = FALSE, filterId = "CD3CD8")
> add(wf, qgate, parent = "Warping")
```

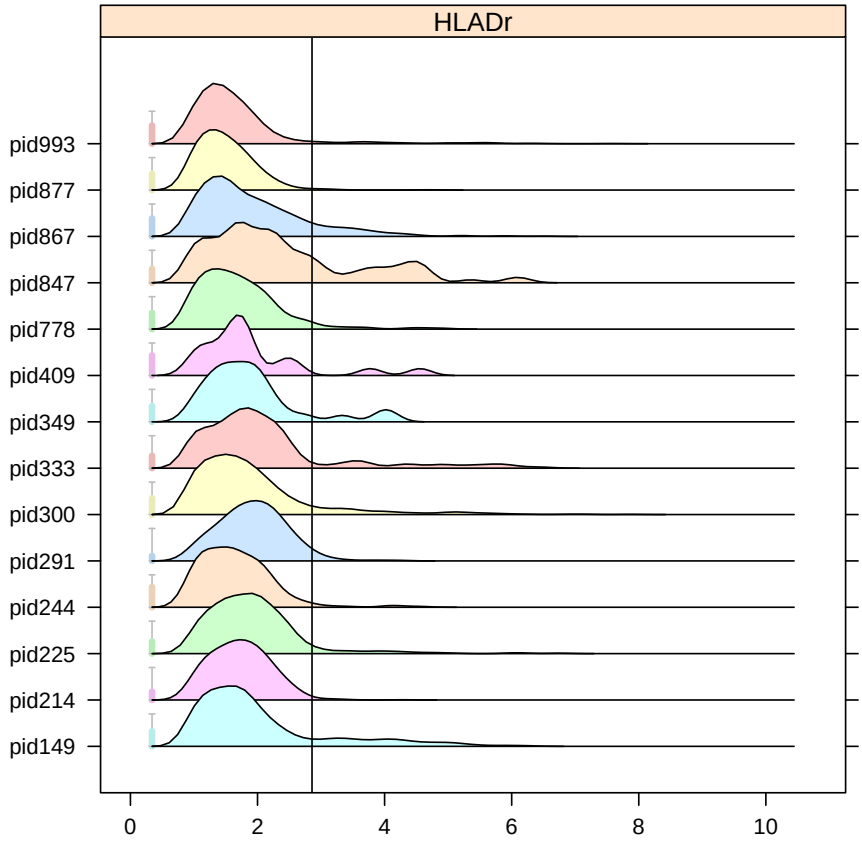
```
> print(xyplot(CD3 ~ CD8 | PatientID, wf[["CD3+CD8+"]]))
```



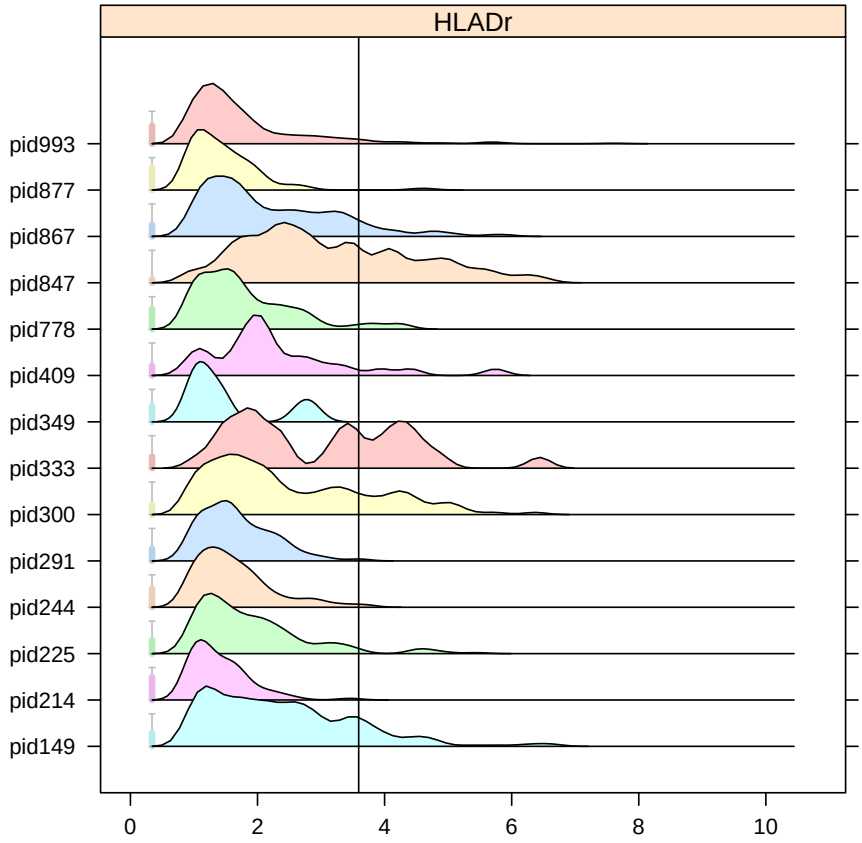
8.6 T Cell activation marker HLADr

We are interested in the proportion of T cells that exhibit the activation marker HLADr for the helper and cytotoxic T cells. We make use of the `rangeGate` to identify T cells that express the HLADr marker from a one dimensional density estimate of the data.

```
> HLADr1 <- rangeGate(Data(wf[["CD3+CD4+"]]), stain = "HLADr",
+     plot = FALSE, alpha = 0.75, filterId = "CD3+CD4+HLAct")
> add(wf, HLADr1, parent = "CD3+CD4+")

> print(densityplot(PatientID ~ HLADr, Data(wf[["CD3+CD4+"]]),
+     refline = HLADr1@min))
```

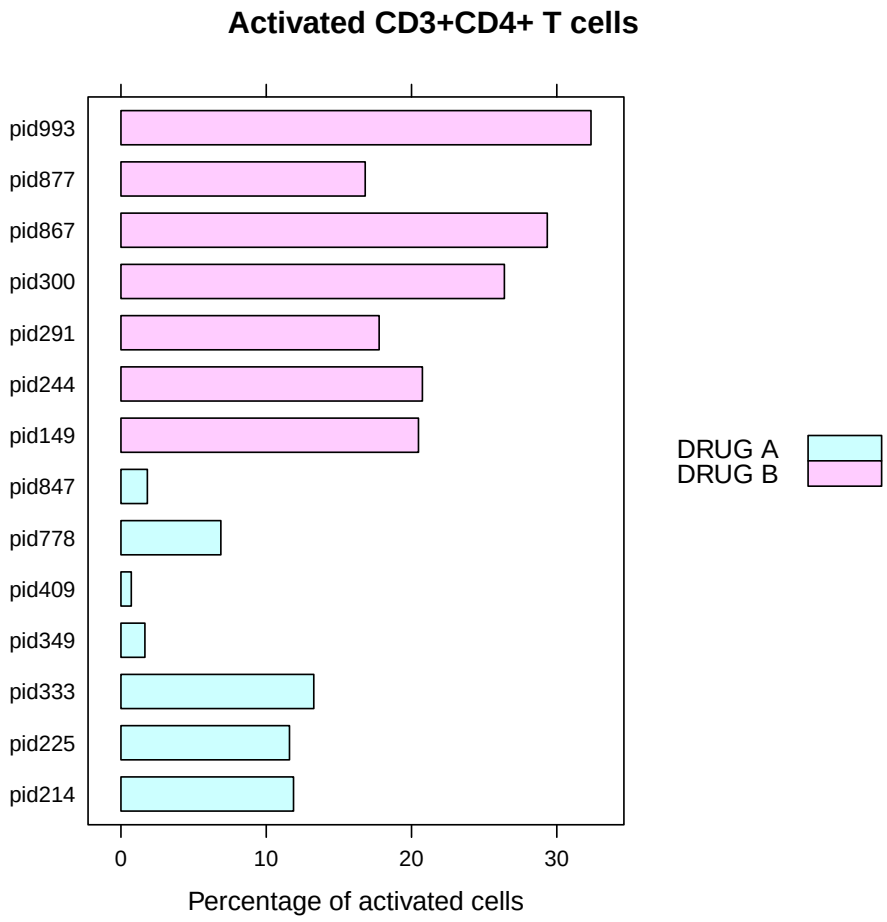



```
> HLADr2 <- rangeGate(Data(wf[["CD3+CD8+"]]), stain = "HLADr",
+     plot = FALSE, alpha = 0.75, filterId = "CD3+CD8+HLAct")
> add(wf, HLADr2, parent = "CD3+CD8+")

> print(densityplot(PatientID ~ HLADr, Data(wf[["CD3+CD8+"]]),
+     refline = HLADr2@min))
```



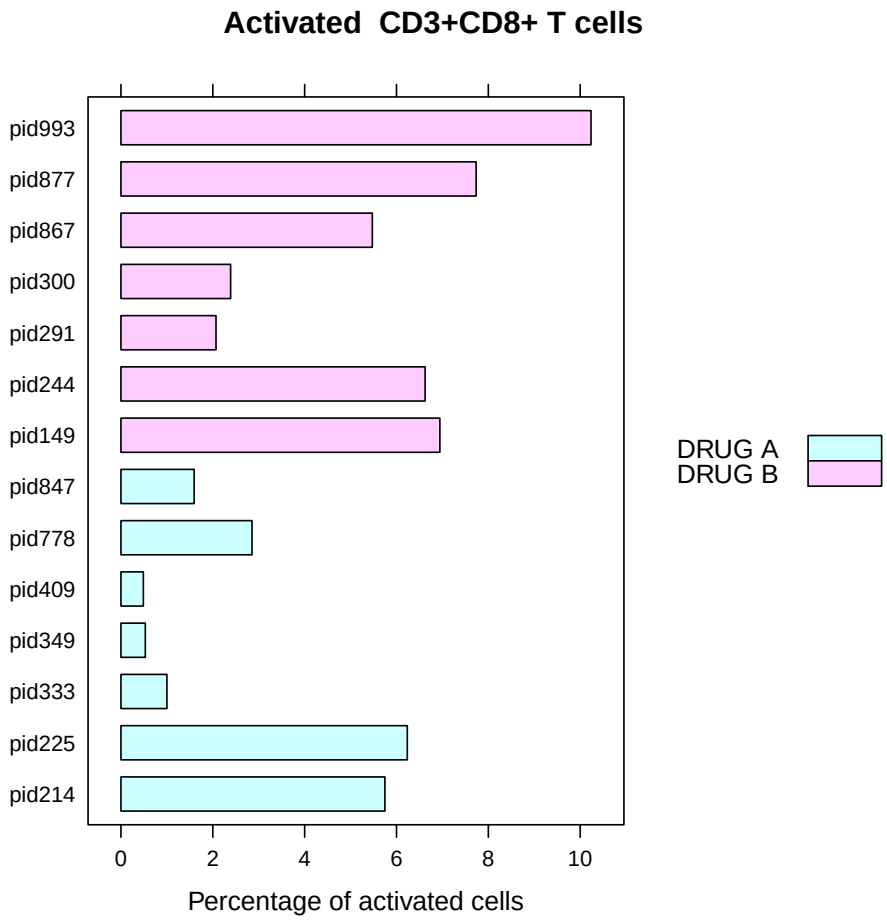
The proportion of helper T cells that exhibit the HLADr activation marker to the total count of events in the original sample is calculated and plotted below.

```
> pr <- fsApply(Data(wf[["CD3+CD4+HLAct-"]]), nrow) * 100/fsApply(Data(wf[["boundFilt+"]]),
+     nrow)
> res <- data.frame(pr, pData(Data(wf[["CD3+CD4+"]]))[c("GroupID",
+     "PatientID")])

> print(barchart(reorder(PatientID, as.numeric(factor(GroupID))) ~
+     pr, data = res, groups = GroupID, stack = TRUE, auto.key = list(points = FALSE,
+     rectangles = TRUE, space = "right"), main = "Activated CD3+CD4+ T cells",
+     xlab = "Percentage of activated cells"))
```



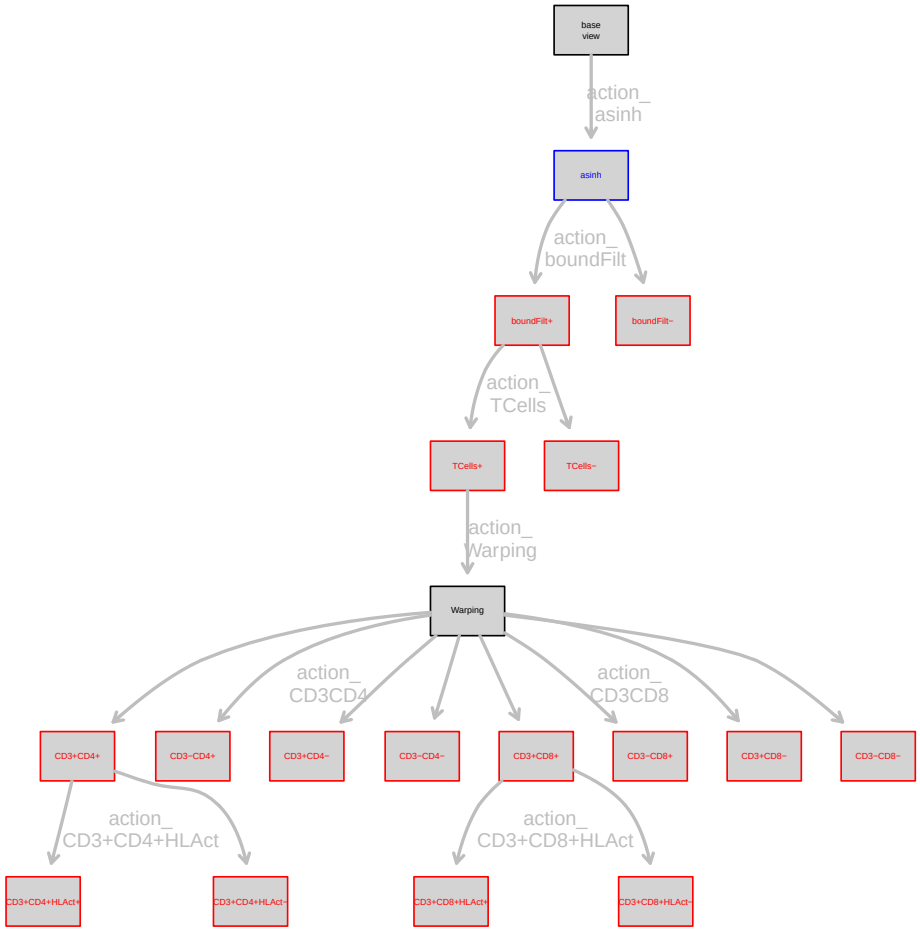
Similarly, the proportion of cytotoxic T Cells that express the activation marker HLADr to the total count of events in the original sample is calculated and plotted below.

```
> pr <- fsApply(Data(wf[["CD3+CD8+HLAct-"]]), nrow) * 100/fsApply(Data(wf[["boundFilt+"]]),
+     nrow)
> res <- data.frame(pr, pData(Data(wf[["CD3+CD8+"]]))[c("GroupID",
+     "PatientID")])

> print(barchart(reorder(PatientID, as.numeric(factor(GroupID))) ~
+     pr, data = res, groups = GroupID, stack = TRUE, auto.key = list(points = FALSE,
+     rectangles = TRUE, space = "right"), main = "Activated  CD3+CD8+ T cells",
+     xlab = " Percentage of activated cells "))
```



The data analysis operations that we have performed can be visualized by plotting the workFlow object

```
> plot(wf)
```


base view
action_ asinh
asinh
action_ boundFilt
boundFilt+
boundFilt−
action_ TCells
TCells+
TCells−
action_ Warping
Warping
action_ CD3CD4
action_ CD3CD8
CD3+CD4+
CD3−CD4+
CD3+CD4−
CD3−CD4−
CD3+CD8+
CD3−CD8+
CD3+CD8−
CD3−CD8−
action_ CD3+CD4+HLAct
action_ CD3+CD8+HLAct
CD3+CD4+HLAct+
CD3+CD4+HLAct−
CD3+CD8+HLAct+
CD3+CD8+HLAct−