Canadian Bioinformatics Workshops

www.bioinformatics.ca
Module 3 -

Preprocessing and Quality Assurance of FCM Data

Radina Droumeva

Bioinformatics Specialist, BC Cancer Agency
Vancouver, British Columbia, Canada

*CBW: Flow Cytometry Data Analysis using R*

June 17, 2013
Module Overview:

- Preprocessing of FCM data: compensation, transformation, margin events removal
- Quality assurance: example QA tests – viable cell count, margin event count
Preprocessing Steps

- Compensate your data well, otherwise further analysis will be meaningless!
- Construct (i.e. code) a procedure to objectively remove debris, doublets, margin events.
- Choose a transformation strategy carefully.
- Remove dead cells and proceed with gating.
Transformations

- **Log**: cannot handle negative values (in FlowJo this is how cells end up bunched up on the axes), but creates a nice spread in the upper decade values of data.

- **Biexponential**: these include arcsinh and logicle transforms.
  - Arcsinh is good for data with few or no negative expression values after compensation – typically well designed panels for 'easy' surface markers which do not have significant spillover issues.
  - Logicle transform is similar to arcsinh but with more parameters to refine the transformation.

Each transformation is defined as the inverse of the following:

- **Log**: \( y = e^x \) (the inverse is \( x = \ln(y) \))
- **Arcsinh**: \( y = \frac{e^x - e^{-x}}{2} \) (\( x = \ln(x + \sqrt{x^2 + 1}) \))
- **Biexponential**: \( y = a \times e^{b \times (x-w)} - c \times e^{-d \times (x-w)} + f \)
Example: transformation choice irrelevant
Example: transformation choice matters!

No transformation

Log Transformation

Arcsinh

Logicle
The following brings up a help menu

# compensate

# A spill-over matrix is often available within the meta data as a keyword SPILL:

\[
M \leftarrow f@description$`\text{SPILL}'
\]

\[
M \# [\text{output truncated}]
\]

<table>
<thead>
<tr>
<th></th>
<th>B515-A</th>
<th>R780-A</th>
<th>R710-A</th>
<th>R660-A</th>
<th>V800-A</th>
<th>V655-A</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1,]</td>
<td>1.000000000</td>
<td>0.000000000</td>
<td>0.000000000</td>
<td>8.841571e-05</td>
<td>0.000249456</td>
<td>0.0006451592</td>
</tr>
<tr>
<td>[2,]</td>
<td>0.000000000</td>
<td>1.000000000</td>
<td>0.071187589</td>
<td>1.484480e-01</td>
<td>0.338903191</td>
<td>0.0097166031</td>
</tr>
<tr>
<td>[3,]</td>
<td>0.000000000</td>
<td>0.331404885</td>
<td>1.000000000</td>
<td>6.196476e-02</td>
<td>0.120978670</td>
<td>0.0040525548</td>
</tr>
<tr>
<td>[4,]</td>
<td>0.000000000</td>
<td>0.088621087</td>
<td>0.389424140</td>
<td>1.000000e+00</td>
<td>0.029758767</td>
<td>0.0655528159</td>
</tr>
<tr>
<td>[5,]</td>
<td>0.000000000</td>
<td>0.136617914</td>
<td>0.010757316</td>
<td>0.000000e+00</td>
<td>1.000000000</td>
<td>0.0001564711</td>
</tr>
<tr>
<td>[6,]</td>
<td>0.000000000</td>
<td>0.000123651</td>
<td>0.019462610</td>
<td>2.182063e-01</td>
<td>0.004953222</td>
<td>1.0000000000</td>
</tr>
<tr>
<td>[7,]</td>
<td>0.000000000</td>
<td>0.000000000</td>
<td>0.000000000</td>
<td>0.000000e+00</td>
<td>0.001055595</td>
<td>0.0022871224</td>
</tr>
<tr>
<td>[8,]</td>
<td>0.000000000</td>
<td>0.000000000</td>
<td>0.000000000</td>
<td>0.000000e+00</td>
<td>0.000000000</td>
<td>0.0081178700</td>
</tr>
<tr>
<td>[9,]</td>
<td>0.003122391</td>
<td>0.008525686</td>
<td>0.001024237</td>
<td>1.162641e-03</td>
<td>0.125401051</td>
<td>0.0181422023</td>
</tr>
<tr>
<td>[10,]</td>
<td>0.002015498</td>
<td>0.069645294</td>
<td>0.194715488</td>
<td>1.007772e-03</td>
<td>0.151611172</td>
<td>0.0012703512</td>
</tr>
<tr>
<td>[11,]</td>
<td>0.001685226</td>
<td>0.054339938</td>
<td>0.277852246</td>
<td>3.430083e-01</td>
<td>0.061752810</td>
<td>0.0775228397</td>
</tr>
<tr>
<td>[12,]</td>
<td>0.000000000</td>
<td>0.00713148</td>
<td>0.048212751</td>
<td>7.319044e-02</td>
<td>0.150563193</td>
<td>0.3862934411</td>
</tr>
<tr>
<td>[13,]</td>
<td>0.001684238</td>
<td>0.000000000</td>
<td>0.000000000</td>
<td>9.533732e-05</td>
<td>0.003463008</td>
<td>0.0157118359</td>
</tr>
</tbody>
</table>
Compensating (continued)

1 # To compensate, simply supply the flowFrame object and the compensation matrix M
2 f.comp <- compensate(f, M)
3 # To evaluate the effect, notice the summary statistics (output truncated to fit screen)
4 summary(f)

<table>
<thead>
<tr>
<th></th>
<th>FSC-A</th>
<th>FSC-H</th>
<th>SSC-A</th>
<th>B515-A</th>
<th>R780-A</th>
<th>R710-A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min.</td>
<td>23410</td>
<td>27010</td>
<td>-8.015</td>
<td>-67.28</td>
<td>-67.12</td>
<td>-44.56</td>
</tr>
<tr>
<td>1st Qu.</td>
<td>34160</td>
<td>33990</td>
<td>168.300</td>
<td>1989.00</td>
<td>528.50</td>
<td>1127.00</td>
</tr>
<tr>
<td>Median</td>
<td>41640</td>
<td>40840</td>
<td>219.900</td>
<td>2852.00</td>
<td>898.30</td>
<td>1733.00</td>
</tr>
<tr>
<td>Mean</td>
<td>44670</td>
<td>43190</td>
<td>331.900</td>
<td>3203.00</td>
<td>1253.00</td>
<td>2342.00</td>
</tr>
<tr>
<td>3rd Qu.</td>
<td>50880</td>
<td>49320</td>
<td>279.000</td>
<td>3345.00</td>
<td>1516.00</td>
<td>2638.00</td>
</tr>
<tr>
<td>Max.</td>
<td>262100</td>
<td>256500</td>
<td>46250.00</td>
<td>261600.00</td>
<td>261600.00</td>
<td>261500.00</td>
</tr>
</tbody>
</table>

1 summary(f.comp)

<table>
<thead>
<tr>
<th></th>
<th>FSC-A</th>
<th>FSC-H</th>
<th>SSC-A</th>
<th>B515-A</th>
<th>R780-A</th>
<th>R710-A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min.</td>
<td>23410</td>
<td>27010</td>
<td>-8.015</td>
<td>-67.34</td>
<td>-26310.000</td>
<td>-14180.0</td>
</tr>
<tr>
<td>1st Qu.</td>
<td>34160</td>
<td>33990</td>
<td>168.300</td>
<td>1982.00</td>
<td>-2.611</td>
<td>675.6</td>
</tr>
<tr>
<td>Median</td>
<td>41640</td>
<td>40840</td>
<td>219.900</td>
<td>2843.00</td>
<td>153.500</td>
<td>1155.0</td>
</tr>
<tr>
<td>Mean</td>
<td>44670</td>
<td>43190</td>
<td>331.900</td>
<td>3203.00</td>
<td>293.800</td>
<td>1620.0</td>
</tr>
<tr>
<td>3rd Qu.</td>
<td>50880</td>
<td>49320</td>
<td>279.000</td>
<td>3336.00</td>
<td>446.700</td>
<td>1877.0</td>
</tr>
<tr>
<td>Max.</td>
<td>262100</td>
<td>256500</td>
<td>46250.00</td>
<td>261600.00</td>
<td>199400.000</td>
<td>149800.0</td>
</tr>
</tbody>
</table>
Truncating data for workshop timeliness...

Now the file is compensated. Due to the large size of the files, they have been compensated and truncated to only a few of the colours and down to 20,000 cells to facilitate this workshop. They can be loaded:

```r
load('fs.RData')
length(fs)
[1] 20
fs[[1]]
```

flowFrame object '185809.fcs' with 20000 cells and 9 observables:

<table>
<thead>
<tr>
<th>name</th>
<th>desc</th>
<th>range</th>
<th>minRange</th>
<th>maxRange</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P1 FSC-A</td>
<td>262207</td>
<td>21245.75000</td>
<td>262206</td>
<td></td>
</tr>
<tr>
<td>$P2 FSC-H</td>
<td>262207</td>
<td>24960.25000</td>
<td>262206</td>
<td></td>
</tr>
<tr>
<td>$P3 SSC-A</td>
<td>261588</td>
<td>-16.42619</td>
<td>261587</td>
<td></td>
</tr>
<tr>
<td>$P4 B515-A</td>
<td>KI67</td>
<td>261588</td>
<td>-99.52692</td>
<td>261587</td>
</tr>
<tr>
<td>$P5 R780-A</td>
<td>CD3</td>
<td>261588</td>
<td>-73.71679</td>
<td>261587</td>
</tr>
<tr>
<td>$P8 V800-A</td>
<td>CD8</td>
<td>261588</td>
<td>-110.58464</td>
<td>261587</td>
</tr>
<tr>
<td>$P9 V655-A</td>
<td>CD4</td>
<td>261588</td>
<td>-65.14906</td>
<td>261587</td>
</tr>
<tr>
<td>$P10 V450-A VIVID / CD14</td>
<td>261588</td>
<td>-38.39849</td>
<td>261587</td>
<td></td>
</tr>
<tr>
<td>$P16 G560-A</td>
<td>CD127</td>
<td>261588</td>
<td>-64.14511</td>
<td>261587</td>
</tr>
</tbody>
</table>

211 keywords are stored in the 'description' slot.
# Let’s look at the scatter channels first. There are margin events at the upper-end of the FSC-A channel:

```r
plot(fs[[1]], c("FSC-A", "SSC-A"), ylim = c(0, 5000), smooth = FALSE)
```

# We can draw a straight ’v’ertical line as roughly the cut-off point for margin events

```r
abline(v = 250000, col = "blue", lwd = 3, lty = "dashed")
```
Removing margin events

# First, select the margin events using 'which':
margin.cells ← which(exprs(fs[[1]])[, "FSC-A"] ≥ 250000)

# Look at how many there are, and how many cells the sample has:
length(margin.cells)
[1] 601

nrow(fs[[1]])
[1] 20000

# Calculate the percentage of cells on the FSC-A margin:
margin.percent ← 100 * length(margin.cells)/nrow(fs[[1]])
margin.percent
[1] 3.005
Visualize margin events

1  # Overtop of the current plot we will isolate and plot the
   margin cells.
2  # First, construct a matrix of the relevant dimensions. Feel
   free to explore this matrix by printing some rows and
   columns from it.
3  f ← fs[[1]]  # for ease of coding
4  A ← exprs(f)[, c("FSC-A", "SSC-A")]
5
6  # (plot not shown, run code to see what happens)
7  # 'points' layers the new plot on top of the current plot
8  points(A[margin.cells, ], pch=".", col = "red", cex=2)
9  # See ?legend and ?paste
10 legend('top', legend = paste("Margin Events:",
                               margin.percent, "\%"), col = "red", pch = 19)
11
12 # Actually remove the margin cells:
13 f.clean.margin ← f[-margin.cells]
14 nrow(f.clean.margin)
15 [1] 19399
# Try a simple transformation:

```r
a <- c(1, 10, 100, 500, 1000)
log10(a)
```

```r
[1] 0.00000 1.00000 2.00000 2.69897 3.00000
```

```r
asinh(a)
```

```r
```

# Define a logicle transformation, see ?transform for help

```r
lgcl <- logicleTransform()
print(lgcl(a))
```

```r
[1] 0.5050419 0.5503692 0.9593149 1.7145863 2.0445191
```
# Let's try the CD3 values now in place of 'a' above
vals ← exprs(f.clean.margin)[, "R780-A"]
vals[1:4] # first 4 values
[1] 825.3061 606.6377 1298.6326 578.4181

# Let's set up a 2 by 2 plot region and try different transformations.
# See ?par for a glimpse of the extensive graphing capabilities of R
par(mfrow = c(2, 2), mar = c(3, 3, 3, 1), mgp=c(2, 1, 0))
plot(density(vals), xlim = c(0, 20000), main="Untransformed CD3 values")
plot(density(log10(vals), na.rm=TRUE), main="Log Transform")
plot(density(asinh(vals)), main = "Asinh")
plot(density(lgcl(vals)), main = "Logicle")
Example: transformation choice matters!

Untransformed CD3 values

Log Transform

Asinh

Logicle
# Notice not much difference between log, arcsinh and logicle transform for the SSC-A channel

# Transformation: log
f ← f.clean.margin

plot(f, c("FSC-A", "SSC-A"), smooth=FALSE, ylim = c(0, 5000),
     main = "No transformation")

# You can replace the whole "SSC-A" column in the expression matrix of the data with its transformed version:
extprs(f)[, 'SSC-A'] ← log10(exprs(f)[, 'SSC-A'])
plot(f, c("FSC-A", "SSC-A"), ylim = log10(c(1, 5000)),
     smooth=FALSE, main = "Log Transformation")

# Arc sinh: almost identical for SSC-A
f2 ← f.clean.margin

exprs(f2)[, "SSC-A"] ← asinh(exprs(f2)[, "SSC-A"])
plot(f2, c("FSC-A", "SSC-A"), ylim = asinh(c(0, 5000)),
     smooth=FALSE, main = "Arcsinh")

# Logicle transform
f3 ← f.clean.margin

exprs(f3)[, "SSC-A"] ← lgcl(exprs(f3)[, "SSC-A"])
plot(f3, c("FSC-A", "SSC-A"), ylim = lgcl(c(0, 5000)),
     smooth=FALSE, main = "Logicle")
Create the same plots for CD3 (R780-A) on your own!
For now we have used the default parameter values for the logicle transform. The parameters are described in `?logicleTransform`. There is a function in `flowCore` which estimates the parameters based on the data.

```r
# Now use the estimateLogicle transform to apply to some other channels
lgcl <- estimateLogicle(f.cclean.margin, colnames(f)[3:9])
f.trans <- transform(f.cclean.margin, lgcl)

# Load the flowDensity package for fancier plots:
library(flowDensity)
par(mfrow = c(2,2),mar = c(3, 3, 3, 1), mgp=c(2, 1, 0))
plotDens(f.trans, c("FSC-A", "SSC-A"))
plotDens(f.trans, c(5, 8))
plotDens(f.trans, c(4, 7))
plotDens(f.trans, c(6, 9))
```
All channels transformed

Module 3: Preprocessing
COFFEE BREAK 15min
Preprocessing the whole flow set

So far we have successfully preprocessed a single FCS file. Next, we will use `for` loops to apply the same steps to the whole flow set.

```r
# Here is a simple for loop which loops over numbers:
for (i in 1:3) {
  print (i^2)
}
[1] 1
[1] 4
[1] 9

# A for loop which loops over the channel names starting with the 4th one:
for (chan in colnames(fs)[4:ncol(fs[[1]])]) {
  print (chan)
}
[1] "B515-A"
[1] "R780-A"
[1] "V800-A"
[1] "V655-A"
[1] "V450-A"
[1] "G560-A"
```
Planning the algorithm

In programming, it is very important to first have a plan before writing any code. Here is the full preprocessing strategy:

- Remove margin events and debris by:
  - pooling all data into one representative sample;
  - defining threshold values on FSC-A and SSC-A which remove most debris and margin events;
  - looping through the flow set and subsetting each flowFrame object to cells falling within the FSC-A and SSC-A values defined above.

- Pool the clean (debris-free) data and use to estimate suitable logicle transformation parameters.

- Transform the data by looping through each flowFrame and applying the transformation.
Defining the debris gates

1 # Load a provided list of support functions. Go through these in your own time and consider changing them to suit your needs.

2 source("../code/supportCode/support_functions.R")

3 # Generate a pooled sample by randomly selecting cells from the whole flow set. The function 'getGlobalFrame' can be found in the above script.

4 global.frame ← getGlobalFrame(fs)

5 plot(global.frame, c("FSC-A", "SSC-A"), ylim = c(0, 1000), smooth=F)

6 abline(v = c(35000, 125000))

7 abline(h = 600)
Visually validating debris gates

```r
# Plot these gates over all frames to ensure they are appropriate
par(mfrow = c(5,4), mar = c(2,2,0,0))
for (i in 1:20){
  plot(fs[[i]], c("FSC-A", "SSC-A"), ylim = c(0, 1000),
       smooth=F)
  abline(v = 35000, col = "blue", lwd=2)
  abline(v=125000, col = "blue", lwd = 2)
  abline(h=600, col = "blue", lwd = 2)
}
```
Removing cells classified as debris

1 # Apply debris gate to whole flow set:
2 # First, start with a copy of the original flowSet 'fs' (normally you will work directly with 'fs' itself)
3 clean.fs ← fs
4 for (i in 1:20){ # Loop over the length of the flowSet
5   f ← fs[[i]]
6 # First restrict the FSC-A values:
7   fsc.indices ← intersect(which(exprs(f)[, "FSC-A"] < 125000), which(exprs(f)[, "FSC-A"] > 35000))
8 # Then restrict SSC-A values and intersect with FSC-A restriction above:
9   ssc.indices ← intersect(which(exprs(f)[, "SSC-A"] > 0), which(exprs(f)[, "SSC-A"] < 600))
10  non.debris.indices ← intersect(fsc.indices, ssc.indices)
11 # Now only select the non debris cells and place the cleaned up flowFrame into the flowSet:
12  f.clean ← f[non.debris.indices]
13  clean.fs[[i]] ← f.clean
14 }
Visually validating debris gates

# See the results:
par(mfrow = c(5, 4), mar = c(2, 2, 1, 1), mgp = c(2, 1, 0))
for (i in 1:20) {
  plotDens(clean.fs[[i]], c("FSC-A", "SSC-A"), main = "")
}

Module 3: Preprocessing
Defining and applying a logicle transform

```r
# Create a new pooled sample to estimate parameters for logicle transform:
global.frame ← getGlobalFrame(clean.fs)

# Define and apply the logicle transform
lgcl ← estimateLogicle(global.frame, colnames(fs)[3:9])
trans.fs ← clean.fs # Like before, for debugging purposes
    leave clean.fs alone
for (i in 1:20){
    trans.fs[[i]] ← transform(clean.fs[[i]], lgcl)
}

# See the results for CD3 and the viability channel:
par(mfrow = c(5, 4), mar = c(2, 2, 2, 1))
for (i in 1:20){
    # CD3 is the 5th channel, viability is the 8th
    plotDens(trans.fs[[i]], c(5, 8))
}
```

Module 3: Preprocessing
Preprocessing done.
Quality Assurance

Before proceeding with the analysis, it is important to perform some quality checks and exclude/flag suspicious samples. Here are some examples of potential quality checks:

- **High margin cell count**: if too many cells lie on the margins, this could indicate a technical issue with acquisition.

- **Viable cell count**: in HIV data set there are actually 466 samples, we must exclude ones which have very low viable cell counts (e.g. < 5000).

- **Density check**: In multi-tube/plate experiments, sometimes the same donor’s sample is divided up and analyzed multiple times using different staining. In such a case we expect the FSC and SSC densities to look almost identical across tubes/wells – if one looks very different, this could be due to a technical issue with acquisition and that tube/well should be flagged for inspection or removed from further analysis.
In this data set the minimum viable cell count was set at 5000 cells – the samples with lower counts are in red, and the gray line is the median viable cell count.
Example QA: FSC and Time density

Module 3: Preprocessing
QA analysis in R using flowQ

# Load helper package and one support function from Radina library(flowQ) source("../code/supportCode/qaProcess.GenericNumber.R")

# Defined directory where QA results will be saved: save.dir ← "/home/rguru/Documents/Workshop/QA/

# This removes the current contents of the QA folder. Do this for space efficiency and to ensure correct results: system(paste(’rm -r ’, save.dir, "*", collapse="", sep=""))

# First, check the raw cell counts (use ’fs’) # See ?qaProcess.cellnumber for explanation qa.raw.count ← qaProcess.cellnumber(fs, outdir=save.dir, cFactor=Inf)

Now in the file browser, navigate to the save.dir folder and check what’s in there.
QA analysis in R using flowQ (continued)

```r
# Next, examine non-debris cell counts from 'clean.fs'
qa.nonDebris.count <- qaProcess.cellnumber(set = clean.fs, outdir=save.dir)

# Use Radina's "pretty" non-debris count and set threshold at 10000 cells:
numbers <- as.vector(fsApply(clean.fs, nrow))
frameIDs <- as.vector(sampleNames(fs))
qa.nonDebris.pretty <- qaProcess.GenericNumber(numbers = numbers, frameIDs=frameIDs, outdir=save.dir, cutoff =10000, name="Cell count")

# flowQ provides an easy html report function:
url <- writeQAResultsReport(fs, list(qa.raw.count, qa.nonDebris.count, qa.nonDebris.pretty), outdir=save.dir)
browseURL(url)
```

For more information on flowQ check out bioconductor and refer to the documentation of the two helper functions provided in the Workshop/code/supportCode folder (flowQhelperFunctions.pdf).
# You can do quality checks without relying on flowQ:

```r
graphics.off() # closes all current plots, good for computer speed

plot(density(exprs(trans.fs[[1]])[, "FSC-A"]), xlim=c(35000, 125000), ylim = c(0,0.00005), lwd=2, main = "FSC Density", sub="", xlab="FSC-A")

for (i in 2:10){
  lines(density(exprs(trans.fs[[i]])[, "FSC-A"]), col=i, lwd =2)
}
```
DINNER 1hr
Open Lab Time!

1 # Open lab assignment:
2 # Use either your own FCS file or the one below provided by
   the flowQ package:
3 library(flowQ)
4 data(qData) # a data set containing 8 flowSets with 4
   samples each
5 f ← qData[[1]][[1]] # first flowSet, first flowFrame
6 f
Questions 1 and 2

# Question 1: Using the keyword information embedded in the FCS file, can you answer any of the following (note if using your own file, your annotation may be missing some of this information)

# a) Date of file acquisition? This can be used if you want to use a control sample and must make sure it was acquired on the same day as your stained sample!

# b) What is the tube name? This can be used as a quality check.

# c) What is the unique patient identifier? This can be used as a quality check.

# Question 2: create a 2D dot plot using the scatter channels.
# For the fcs file defined above, first take a look through the keywords:

f@description

# The answers then follow:

answer.to.a ← f@description$'DATE'
answer.to.b ← f@description$'TUBE NAME'
answer.to.c ← f@description$'PATIENT ID'
Question 2: SOLUTION

# Many ways to plot, here are 3:
# Using R's basic plotting functionality, use 'exprs' -- this contains the expression values we want to plot
plot(exprs(f)[, c("FSC-A", "SSC-A")], pch=".", main = "Plot 1")

# Using flowViz plotting, slightly simplified
library(flowViz)
plot(f, c("FSC-A", "SSC-A"), main = "Plot 2")
contour(f, c("FSC-A", "SSC-A"), nlevels=20, add=TRUE) #
# flowViz actually has a tonne of cool things you can plot, check out the documentation:

# Using plotDens in flowDensity
plotDens(f, c("FSC-A", "SSC-A"))
# Question 3: based on this plot, design and execute an algorithm to remove the debris/margin events and only retain the lymphocytes. Then, plot the original frame, and overtop of that plot (hint: see ?points, hint2: remember to use pch="." ) the new "clean" frame containing only the lymphocytes in a different colour (to see names of colours you can use see ?colours). Note that if you are using your own FCS file and the lymphocytes are not the relevant population in terms of FSC-SSC 2D dot plot, then feel free to come up with something useful to do using those channels -- e.g. remove the debris by setting a threshold on low-scatter cells, removing doublets by plotting the FSC-W channel instead of side scatter, etc..

# Question 4: Note that in Question 1 you looked through the keywords and there was no compensation matrix provided. Assume the data is compensated. If using your own FCS file -- do you need to apply the compensation? If so, see ?compensate and apply it.
Question 3: SOLUTION

# Roughly, the FSC-A boundaries for the lymphocytes look like 250 and 600, and for SSC-A: 100 and 400.

abline(v = c(250, 600), lwd = 2, col = "blue")
abline(h = c(100, 400), lwd = 2, col = "blue")

# Now to actually extract the lymphocytes, first we find the indices in terms of the FSC-A restriction:

fsc.indices ← intersect(which(exprs(f)[, "FSC-A"] > 250), which(exprs(f)[, "FSC-A"] < 600))

# Now restrict on SSC-A and find the cells in the intersection:

ssc.indices ← intersect(which(exprs(f)[, "SSC-A"] > 100), which(exprs(f)[, "SSC-A"] < 400))
lymphocytes ← intersect(fsc.indices, ssc.indices)

# Create a new flowFrame object which only contains the lymphocyte cells of 'f':

f.clean ← f[lymphocytes]

# Now let's first plot f and overtop plot f.clean:

plot(f, c("FSC-A", "SSC-A"), smooth=FALSE)
points(exprs(f.clean)[, c("FSC-A", "SSC-A")], pch=".", col = "red")
# Typically, the compensation matrix will be in one of the keywords of the FCS file:

```
# compensation.matrix ← f@description$‘SPILL‘ or f@
description$‘SPILLOVER‘ or something similar.
```

# Then all you have to do is:

```
# f.compensated ← compensate(f, compensation.matrix)
```
Question 5

1. Which channels need to be transformed? Pick one of the channels and create a 2 by 2 plot of:
2. a) SSC-A vs Untransformed Channel of choice. Remember -- you should now be using your 'clean' flowFrame with the lymphocytes only, not the original one.
3. b) SSC-A vs Log10 of values in Channel of choice
4. c) SSC-A vs Asinh of the channel values
5. d) Create a logicle transformation object, call it 'lgcl', and create a plot of SSC-A vs logicle transform of channel values

6. NOTE: If possible, please pick a 'nice' channel for this excercise, e.g. a major stain such as CD3, CD4, CD8, etc. -- one for which you expect to see two nicely separated positive and negative fractions!
7. Hint 1: if your points seem squished, remember you can add 'ylim = c(0, some suitable value)' inside your 'plot' statement!
8. Hint 2: (if using qData) Does your logicle transform not look nearly as good as you hoped? See ?logicleTransform and see what parameter you should change and try again. If it still does not look right -- do not worry, this is actually a fault with the data. Can you think of a way to demonstrate that to convince yourself?
# Let’s pick CD4. First, we will create all transformations and then plot them.
# I do not want to type "PE-Cy7-A" over and over, so I will create a variable:
mychan ← "PE-Cy7-A"

# Log10:
f.log ← f.clean
exprs(f.log)[, mychan] ← log10(exprs(f.clean)[, mychan])

# Asinh:
f.asinh ← f.clean
exprs(f.asinh)[, mychan] ← asinh(exprs(f.clean)[, mychan])

# Logicle:
lgcl ← logicleTransform(t = 1024) # See ?logicleTransform to find out why I set t = 1024!
f.lgcl ← f.clean
exprs(f.lgcl)[, mychan] ← lgcl(exprs(f.clean)[, mychan])

# Estimate Logicle, just to try:
estimate.lgcl ← estimateLogicle(f.clean, mychan)
f.est ← transform(f.clean, estimate.lgcl)
# Now open a new plot with 2 by 2 slots:
par(mfrow = c(3, 2))  # Actually, I did 2 by 3 to add a couple of plots extra!

# Plot the untransformed data
plot(f.clean, c("SSC-A", mychan), main = "Untransformed", smooth=FALSE, ylim = c(0, 400))

# Plot the log 10 transformed data
plot(f.log, c("SSC-A", mychan), main = "Log 10", smooth=FALSE, ylim = c(0, 3))

# Plot the asinh transformed data
plotDens(f.asinh, c("SSC-A", mychan), main = "Asinh")

# Plot the Logicle transformed data
plotDens(f.lgcl, c("SSC-A", mychan), main = "Logicle")

# Plot the Estimate Logicle transformed data
plotDens(f.est, c("SSC-A", mychan), main = "Estimate Logicle")

# Display the histogram of the clean expression data
hist(exprs(f.clean)[, mychan], 30)

# Add a legend indicating the percentage of cells with values equal to 1
legend('top', paste(round(length(which(exprs(f.clean)[, mychan] == 1))/nrow(f.clean)*100,1), "% of cells have
values equal to 1 for ", mychan, sep = ""))
# Note there appear to be a bunch of bunched up values on the axis... The truth is, those values were on the axis to begin with, so transforming them cannot turn them into anything other than a bunch of points on the axis! This is likely an ancient computer system attached to the flow cytometer which does this to the data when the operator saves the files -- we cannot reverse-engineer this. How would we catch this better? Use a Quality Assurance check! Here is one idea:

# For the purposes of flowQ, we cannot work with a single frame, so consider the flowSet it came from, qData[[1]]:

```
save.dir ← '/home/rguru/Documents/Workshop/assignmentQA/'
```

# See the help for why I added "absolute.value = 0.1". Also try running it without that and see what happens.

# Note that when we omit the 'channels' parameter, it checks the margin event count for every channel.

```q
qa.check ← qAProcess.marginevents(set = qData[[1]], outdir = save.dir, absolute.value = 0.1)
```

```
url ← writeQAResult(qData[[1]], list(qa.check), outdir = save.dir)
```

```
browseURL(url)
```
Question 6

# Question 6: Transform all channels which you believe should be transformed and come up with one plot (with multiple small plots if necessary) which allows you to see what the transformed data looks like.
# Cycle through each channel and transform (except scatter channels and Time!)

```r
f.trans <- f.clean
for (chan in colnames(f.clean)[3:7]){
  exprs(f.trans)[, chan] <- lgcl(exprs(f.clean)[, chan])
}
```

# Note the last plot below will be against the 'Time' parameter -- this is useful in quality assurance!

```r
par(mfrow = c(3, 2))
for (chan in colnames(f.trans)[3:8]){
  plotDens(f.trans, c("SSC-A", chan))
}
```
END OF DAY 1!