# Module 3 code: Preprocessing FCM data and Quality Assurance

# Author: Radina Droumeva

# Start fresh: remove all loaded variables:
\texttt{rm(list=ls())}

\texttt{graphics.off()} # close all graphics, for efficiency

# Ensure flowCore library is loaded:
\texttt{library(flowCore)}

\texttt{setwd('/home/rguru/Documents/Workshop/data')} # To see a list of files in the 'fullFCS' folder, use \texttt{dir()}, it returns a vector of file names
\texttt{files <- dir('fullFCS/')} files

# To select the first file in the directory, subset the vector on its first index:
\texttt{firstFile <- paste('fullFCS/', files[1], sep = '')}

\texttt{firstFile}

\texttt{f <- read.FCS(firstFile)}

# Compensation:
\texttt{?compensate}

# A spill-over matrix is often available within the meta data:
\texttt{M <- f@description$'SPILL'}

\texttt{M}

\texttt{f.comp <- compensate(f, M)}

\texttt{summary(f)}
\texttt{summary(f.comp)}

# 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 to facilitate this workshop. They can be loaded:
\texttt{load('fs.RData')}

\texttt{length(fs)}

\texttt{fs[[1]]}

# Removing margin events.

# Let's look at the scatter channels first. There are margin events at
library(flowViz)

plot(fs[[1]], c("FSC-A", "SSC-A"), ylim = c(0, 1000), smooth=FALSE)

abline(v = 250000, col = "blue", lwd = 3, lty = "dashed")

margin.cells <- which(exprs(fs[[1]])[, "FSC-A"] >= 250000)

length(margin.cells)

nrow(fs[[1]])

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

margin.percent

# Let us only plot those cells:
# First, construct a matrix of the expression data:
A <- exprs(f)[, c("FSC-A", "SSC-A")]

dim(A)

A[1, ]

plot(A, pch=".", ylim = c(0, 1000))

points(A[margin.cells, ], pch=".", col = "red", cex=2)

legend('top', legend = paste("Margin Events:" , margin.percent, "%"), col = "red", pch = 19)

# Now remove the cells on the margin:
f.clean.margin <- f[-margin.cells]

nrow(f.clean.margin)

# Transformations
# Try a simple transformation on some made up numbers:
a <- c(1, 10, 100, 500, 1000)
log10(a)

asinh(a)

lgcl <- logicleTransform() # see ?transform for the flowCore package
print(lgcl(a))

# Let's try the CD3 values now
vals <- exprs(f.clean.margin)[, "R780-A"]

vals[1:4]
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")

# Notice not much difference between log, arcsinh and logicle transform for the SSC-A channel
# Transformation: log
f <- f.clean.margin
par (mfrow = c(2, 2), mar = c(3,3, 3, 1), mgp=c(2, 1, 0))
plot(f, c("FSC-A", "SSC-A"), smooth=FALSE, ylim = c(0, 5000), main = "No transformation")

# Simply replace the "SSC-A" values with the log10 transformed values
exprs(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")

####### PRACTICE ####################################################################

# Try the same for CD3 (R780-A):

f <- f.clean.margin
plot(f, c("FSC-A", "R780-A"), smooth=F, main = "No transformation")
exprs(f)[, 'R780-A'] <- log10(exprs(f)[, 'R780-A'])
plot(f, c("FSC-A", "R780-A"), ylim = log10(c(1, 5000)), smooth=FALSE, main = "Log Transformation")

legend('bottom',
... legend=paste(round(100*length(which(exprs(f.clean.margin)[, "R780-A"])), na.rm=TRUE))
<1))/nrow(f), 2), "% cells on axis")

# Asinh
f2 <- f.clean.margin
exprs(f2)[, "R780-A"] <- asinh(exprs(f2)[, "R780-A"])
plot(f2, c("FSC-A", "R780-A"), ylim = asinh(c(-500, 5000)), smooth=FALSE, 
... main = "Arcsinh")

# Logicle transform
f3 <- f.clean.margin
exprs(f3)[, "R780-A"] <- lgcl(exprs(f3)[, "R780-A"])
plot(f3, c("FSC-A", "R780-A"), ylim = lgcl(c(-500, 5000)), smooth=FALSE, 
... main = "Logicle")

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

# Fancier plotting functionality from flowDensity
library(GEOmap)
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))

# Now we must apply these steps to all samples!
# Instead of just removing margins, let's also remove high/low-scatter cells
# (http://jid.oxfordjournals.org/content/201/2/272.full.pdf for gating strategy on this data set)
geometrics.off()
load('fs.RData')
# We can use a for loop to loop through all samples and apply the same steps. Here is a simple for loop:
for (i in 1:3){
  print (i^2)
}
for (chan in colnames(fs)[4:ncol(fs[[1]]))]{
  print (chan)
}
# Generate a pooled frame to define debris gate:
source("../code/supportCode/support_functions.R") # support functions provided by Radina, examine later to see what you can use when you fly home!
global.frame <- getGlobalFrame(fs)
plot(global.frame, c("FSC-A", "SSC-A"), ylim = c(0, 1000), smooth=F)
abline(v = c(35000, 125000), col = "blue", lwd = 2)
abline(h = 600, col = "blue", lwd = 2)

# Instead of dot plots, let's look at this 1 dimension at a time:
plot(density(exprs(global.frame)[, "FSC-A"]))
abline(v=125000)
abline(v=35000)
plot(density(exprs(global.frame)[, "SSC-A"])), xlim = c(0, 1000))
abline(v=600)

# 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)
}
# Apply debris gate to whole flow set:
# First, start with a copy of the original flowSet 'fs' (normally you will work directly with 'fs' itself)
clean.fs <- fs
for (i in 1:20){ # Loop over the length of the flowSet
  f <- fs[[i]]
  # First restrict the FSC-A values:
  fsc.indices <- intersect(which(exprs(f)[, "FSC-A"] < 125000),
                           which(exprs(f)[, "FSC-A"] > 35000))
# Then restrict SSC-A values and intersect with FSC-A restriction above:

```
ssc.indices <- intersect(which(exprs(f)[, "SSC-A"] > 0),
                         which(exprs(f)[, "SSC-A"] < 600))
non.debris.indices <- intersect(fsc.indices, ssc.indices)
```

# Now only select the non debris cells and place the cleaned up flowFrame into the flowSet:

```
f.clean <- f[non.debris.indices]
clean.fs[[i]] <- f.clean
```

# 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 = "")
}
```

# Create a new pooled sample to estimate parameters for logicle transform:

```
global.frame <- getGlobalFrame(clean.fs)
lgcl <- estimateLogicle(global.frame, colnames(fs)[3:9])
```

# Apply the transformation to each sample in a for loop similarly to above.

```
trans.fs <- clean.fs
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){
  plotDens(trans.fs[[i]], c(5, 8))
}
```
## Quality Assurance

# Load helper package and one support function from Radina

```r
library(flowQ)
source("../code/supportCode/qaProcess.GenericNumber.R")
```

# Define directory where QA results will be saved:

```r
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:

```r
# (You may get an error the first time, as there is no "QA" folder yet. If later you set this up on your own computer and don't use Ubuntu OS, you can just delete the folder yourself in the file browser)
system(paste('rm -r ', save.dir, "*", collapse="", sep=""))
```

# First, check the raw cell counts (use 'fs')

```r
load('fs.RData')
qa.raw.count <- qaProcess.cellnumber(fs, outdir=save.dir, cFactor=Inf)
```

# Next, examine non-debris cell counts from 'clean.fs'

```r
qa.nonDebris.count <- qaProcess.cellnumber(set = clean.fs,
... outdir=save.dir)
```

# Use Radina's "pretty" non-debris count and set threshold at 10000 cells:

```r
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:

```r
url <- writeQAReport(fs, list(qa.raw.count, qa.nonDebris.count,
... qa.nonDebris.pretty), outdir=save.dir)
browseURL(url)
```
# You can do quality checks without relying on flowQ.
# Can you say anything about the sample quality based on the following plot?

```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)
}
```

# Save the transformed flowSet object for tomorrow!
```
save(trans.fs, file = '/home/rguru/Documents/Workshop/data/trans.fs.RData')
```

# Radina has provided a set of functions to process typical data sets:
```r
# graphics.off()
# source('../code/supportCode/flowPrep.R')
# source('../code/supportCode/support_functions.R')
# fs.prep <- flowPrep(fs, apply.comp=FALSE, plot.preproc=TRUE,
# ... plot.for.lympho=TRUE)
```