~~~
---
title: "Preprocessing FCM data in R"
author: "R. Burke Squires (adapted from Radina Droumeva)"
date: "7/11/2017"
output:
  html_document:
    toc: yes
  html_notebook:
    toc: yes
  pdf_document:
    highlights: tango
    keep_tex: yes
    number_sections: yes
    toc: yes
---

LICENSE


```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = FALSE)
```



## Make sure R knows which directory our data will be read from

```{r}
getwd()
setwd('./data')
dir()
dir('fullFCS/')
```

# Start fresh: remove all loaded variables:


```{r}
rm(list=ls())
graphics.off() # close all graphics, for efficiency
```

# Ensure flowCore library is loaded:

```{r}
library(flowCore)
setwd('./data')
```

# To see a list of files in the 'fullFCS' folder, use dir(), it returns a vector of file names

```{r}
files <- dir('./data/fullFCS/')
files
```

# To select the first file in the directory, subset the vector on its first index:

```{r}
firstFile <- paste('./data/fullFCS/', files[1], sep = "")
firstFile
f <- read.FCS(firstFile)
f
```

###########################################################

# Compensation:

```{r}
?compensate
```

# A spill-over matrix is often available within the meta data:
```{r}
M <- f@description$`SPILL`
M
f.comp <- compensate(f, M)
summary(f)
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 facilit
load('./data/fs.RData')
length(fs)
fs[[1]]
```

#############################################################
# Removing margin events.

# Let's look at the scatter channels first. There are margin events at the upper-end of the FSC-A channel:
```{r}
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")
~~~

~~~
```

```{r}
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
```

```{r}
f <- fs[[1]]
# 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:
```{r}
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
```{r}
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
```{r}
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
```{r}
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
```{r}
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):
```{r}
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"] <1))/nrow(f), 2), "% cells on axis"))
```

# Asinh
```{r}
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
```{r}
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
```{r}
lgcl <- estimateLogicle(f.clean.margin, colnames(f)[3:9])
f.trans <- transform(f.clean.margin, lgcl)
```

# Fancier plotting functionality from flowDensity
```{r}
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)
```{r}
graphics.off()
load('./data/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:
```{r}
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:
```{r}
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
```{r}
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)

```{r}
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:

```{r}
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:

```{r}
~~~

~~~~
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.

# Apply the transformation to each sample in a for loop similarly to above.
# Note that if your transformation looks bad, you will have to change it:
# sometimes using a pooled sample does not work well -- consider using one
# representative sample instead, or using the generic logicle transform instead
# of the estimateLogicle. You may even try 'asinh'.
```{r}
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))
}
```
~~~~