~~~
---
title: "Gating 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

# Gating in R

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

## Clear current workspace, close plots, load libraries

```{r}
rm(list=ls())
graphics.off()
library(flowCore)
library(flowDensity)
library(GEOmap)
```

## Tell R which directory we are working in and load the transformed flowSet object:

```{r}
setwd('~/Desktop/auto_flow_cyto_analysis/data/')
load('trans.fs.RData')
trans.fs
```

## Let's gate the CD3+ live cells!

```{r}
cd3 <- "R780-A"
dump <- "V450-A"
```

## First create a pooled flowFrame:

```{r}
source("./code/supportCode/support_functions.R")
pooled.frame <- getGlobalFrame(trans.fs)
{plotDens(pooled.frame, c(cd3, dump))
  abline(v = 1, lwd=2, col = "blue")
  abline(h = 1.5, lwd=2, col="blue")}
```

## By looking at this, it looks like CD3 = 1 and Dump Channel = 1.5 are good gates

```{r}
# abline(v = 1, lwd=2, col = "blue")
# abline(h = 1.5, lwd=2, col="blue")
```

## Let's verify by plotting whole flowSet:

```{r}
par(mfrow = c(5, 4), mar = c(3, 3, 2, 1), mgp = c(2, 1, 0))
for (i in 1:length(trans.fs)){
  plotDens(trans.fs[[i]], c(cd3, dump))
  abline(v = 1, lwd=2, col = "blue")
  abline(h = 1.5, lwd=2, col="blue")
}
```

## Looks pretty good. Now let's isolate the viable cells into a new flowSet

```{r}
viable.fs <- trans.fs
```

# Use fsApply

This time, instead of a for loop, let's use 'fsApply'. It applies a single function to each flowFrame object. So, we must first write our very own

```{r}
# Code for single flowFrame:
f <- trans.fs[[1]]
cd3pos.indices <- which(exprs(f)[, cd3] > 1)
dumpneg.indices <- which(exprs(f)[, dump] < 1.5)
combined <- intersect(cd3pos.indices, dumpneg.indices)
viable.f <- f[combined]
```

~~~
```

## Is this right? Let's check

```{r}
graphics.off()
plot(exprs(f)[, c(cd3, dump)], pch=".")
points(exprs(viable.f)[, c(cd3, dump)], pch=".", col = "green4")
```

## See ?chull for an idea of how to plot a gate using lines instead of just a different colour: run the example in the helpf file!

```{r}
plotDens(f, c(cd3, dump))
X <- exprs(viable.f)[, c(cd3, dump)]
gate.pts <- chull(X)
gate.pts <- c(gate.pts, gate.pts[1]) # Connect the first and last point to make gate closed.
lines(X[gate.pts, ], lwd=2, col="blue", lty="dashed")
```

## Functions

This is how you create your own function in R:

1) Give it a name of your choice, make sure it is uncommon enough that it won't accidentally interfere with existing R functions! I.e. don't create

2) inside the 'function(*)' put a variable name which will be only used within the function. I chose 'f', but you can call it 'x' and replace all t

```{r}
getViableFrame <- function(f){
  # Manipulate the input 'f':
  cd3pos.indices <- which(exprs(f)[, cd3] > 1)
  dumpneg.indices <- which(exprs(f)[, dump] < 1.5)
  combined <- intersect(cd3pos.indices, dumpneg.indices)
  viable.f <- f[combined]
  # Finally, return the desired altered version of the input:
  return (viable.f)
}
```

## Execute the function

Execute the function definition to enable it for use. Now try it:

```{r}
viable.f <- getViableFrame(f)
# Is this right? Let's check (again):
graphics.off()
plot(exprs(f)[, c(cd3, dump)], pch=".")
points(exprs(viable.f)[, c(cd3, dump)], pch=".", col = "green4")
```

## Now we can use fsApply:

```{r}
viable.fs <- fsApply(trans.fs, getViableFrame)
```

##3 Let's look at the proportions of live cells:

```{r}
live.counts <- fsApply(viable.fs, nrow)/fsApply(trans.fs, nrow)
plot(density(live.counts), main = "Proportion CD3+ viable cells")
```

## Twin Peaks?

It kind of looks like two peaks in the density. Can we do anything interesting just with this information?

Let's extract the clinical information we have first.

For this workshop, I have selected 20 FCS files for patients which have an event reported -- either death or progression to AIDS. We have the numbe

```{r}
survival <- fsApply(viable.fs, function(x) x@description$`CD Survival time from seroconversion`)
# Let's convert this to numbers so we can work with them:
survival <- as.numeric(survival)
survival
plot(survival, live.counts, pch = 19)
```

## Try kmeans:

```{r}
km <- kmeans(live.counts, 2)
plot(survival, live.counts, pch = 19, col = km$cluster, main = "K-means misclassifies 3 samples from each group.")
# Too few samples, but it looks like more of the low-survival patients have lower CD3+live proportions also.
# For a really good explanation of k-means and hierarchical clustering, see Andrew Moore's slides: http://www.autonlab.org/tutorials/kmeans11.pdf
```

## Automated Gating

So far we have preprocessed our data and have now isolated the live CD3+ cells.

Let's now consider automated gating for the remainder of the analysis.

Consider CD4 first:
~~~

```{r}
cd4 <- "V655-A"
pooled.frame <- getGlobalFrame(viable.fs)
par(mfrow = c(1, 2), mar = c(3, 3, 2, 1), mgp=c(2, 1, 0))
plotDens(pooled.frame, c(cd3, cd4))

#Try flowDensity -- the function 'deGate':
cd4.gate <- deGate(pooled.frame, cd4)
abline(h = cd4.gate, lwd=2, col="blue")
```

## Try flowDensity -- the function 'deGate'

```{r}
cd4.gate <- deGate(pooled.frame, cd4)
```

## Looks good, how did it do it?

```{r}
deGate(pooled.frame, cd4, graphs=TRUE)
```

## Try for every channel (except scatter, CD3 and the dump channel):

```{r}
ki67 <- "B515-A"
cd8 <- "V800-A"
cd127 <- "G560-A"
```

##  This time we will record the gates for all channels into a vector

```{r}
store.gates <- rep(-Inf, 4)
names(store.gates) <- c(cd4, cd8, cd127, ki67)
store.gates
par(mfrow = c(2, 2))
for (chan in c(cd4, cd8, cd127, ki67)){
  plotDens(pooled.frame, c(cd3, chan))
  store.gates[chan] <- deGate(pooled.frame, chan)
  abline(h = store.gates[chan])
}
store.gates
```

## All looks good except for CD127. Let's work on it:

```{r}
par(mfrow = c(5, 4), mar = c(3, 3, 1,1), mgp=c(2, 1, 0))
for (i in 1:length(viable.fs)){
  plotDens(viable.fs[[i]], c("SSC-A", cd127))
}
```

Still not obvious. This is where a control would be necessary! Let's ignore CD127 from now on.

Let's at least make sure CD4, CD8 and KI67 work for all samples:

```{r}
par(mfrow=c(5,4), mar = c(3, 2, 2, 1), mgp=c(2, 1, 0))

for (i in 1:20){
  plotDens(viable.fs[[i]], c(cd4, cd8))
  abline(v = store.gates[cd4], lwd=2, col="blue")
  abline(h = store.gates[cd8], lwd=2, col="blue")
}

par(mfrow=c(5,4), mar = c(3, 2, 2, 1), mgp=c(2, 1, 0))

for (i in 1:20){
  plotDens(viable.fs[[i]], c(cd8, ki67))
  abline(v = store.gates[cd8], lwd=2, col="blue")
  abline(h = store.gates[ki67], lwd=2, col="blue")
}
```