

Automated Gating of Flow Cytometry Data using the Bioconductor **openCyto** Framework

*Nichole Monhait, MPH Candidate
Colorado School of Public Health, Colorado State University*

2019-05-01

Contents

1	What's inside?	5
2	Getting Started	7
2.1	Function and R Object Definitions	7
2.2	Required Packages and Installation	8
3	Working with your Manual Gating Scheme	9
3.1	Read in flowJo file	9
3.2	Parse FCS files	10
3.3	Visualize and Verify	12
3.4	Clone	14
4	Create .csv	15
4.1	.csv Gating Template Structure	15
4.2	Creating the Template	16
4.3	Load .csv into R	20
5	Automate Gating	21
5.1	Apply Gating	21
5.2	Plot Automated Gating	21
5.3	Population Statistics	21
6	Customization	23
6.1	Hiding unwanted nodes	23
6.2	Renaming nodes	23
6.3	Adjusting plots	24

Chapter 1

What's inside?

The goal of this tutorial is to take manually gated data from a .wsp flowJo file and use the Bioconductor **openCyto** framework to create an automated gating procedure within R which can be replicated on any dataset. The basic functions used in this tutorial include:

```
openWorkspace()  
parseWorkspace()  
gatingTemplate()  
gating()  
plot()  
plotGate()
```

The example data used in this tutorial is from Colorado State University's Microbiology, Immunology, and Pathology Department. Alternatively, you can input your own flowJo Workspace and follow this tutorial, so long as the file is in .wsp format.

Chapter 2

Getting Started

Here is an overview of the process to automate flow cytometry data using R's **openCyto** and what you will need to successfully automate your own flow cytometry analysis. The general steps to accomplish this are as follows:

1. Read in a manually gated flowJo workspace in .wsp file format.
2. Parse raw FCS files from the read in workspace.
3. Visualize the manual gating template and resulting gates to verify gating scheme.
4. Create and read in a .csv gating template.
5. Automate gating.
6. Visualize automated gating template and gates to verify gating scheme.
7. Extract population statistics and relevant information.

This process is completed primarily with the **openCyto** package but calls upon other packages within the Bioconductor **openCyto** framework. Packages needed to complete this tutorial are listed at the end of this chapter. Descriptions of each function and R object used for this analysis are below.

2.1 Function and R Object Definitions

Function/Object Name	Definition
wsfile	flowJo .wsp file location
openWorkspace()	function used to read in **wsfile**
ws	read in data from flowJo
parseWorkspace()	function to extract FCS files from **ws**
gating_set	parsed FCS files to be gated
clone()	function used to create a clone of **gating_set**
gh	subset of gating_set
gt	.csv gating template
templateGen()	function used to generate a .csv template from existing manual gates
gatingTemplate()	function used to read in .csv template
gating()	function used to apply gates to a gating set
plot()	function to visualize gating tree
plotGate()	function to visualize gates

2.2 Required Packages and Installation

Before getting started, install and load the following libraries into a new R script. As you will see below, this tutorial uses the development version of `openCyto`. Use the following to ensure the correct packages are installed.

To install

```
devtools::install_github("RGLab/openCyto", ref = "trunk")
install.packages("data.table")
install.packages("flowWorkspace")
install.packages("flowCore")
install.packages("flowStats")
install.packages("flowClust")
install.packages("plyr")
```

To load

```
library(openCyto)
library(flowWorkspace)
library(data.table)
library(flowCore)
library(flowStats)
library(flowClust)
library(plyr)
```


Chapter 3

Working with your Manual Gating Scheme

Current methods for manually gating flow cytometry are both time consuming and costly, making automated gating an appealing option. The `openCyto` package allows users to take manually gated data from flowJo, reproduce those gates in R, and eventually automate the gating process.

The first step in this process is to bring a pre-existing flowJo file into R in order to recreate the gating environment. The remainder of this chapter will detail the following:

1. Read in flowJo .wsp file
2. Parse FCS files
3. Visualize and verify manual gates
4. Clone data for later automation

3.1 Read in flowJo file

Within flowJo, transformation, compensation, and gating can be saved as either .xml or .wsp filetypes. This tutorial will only detail steps from a .wsp filetype saved from flowJo. Saving analysis within flowJo is detailed [here](#).

The result of step 1 will be replication of the manual transformation, compensation, and gating from the flowJo workspace saved as an R object. Before you begin, be sure you have loaded the required packages outlined in the previous chapter.

Once all packages are loaded, save the .wsp file path as an R object called `wsfile`. Next, use `openWorkspace()` with your R object name to open the .wsp file in R, save this as `ws`. Here is an example of saving and opening `wsfile`. Following this step, `ws` will be saved as a flowWorkspace object containing groups of samples.

```
library(openCyto)
library(flowWorkspace)
library(data.table)
library(flowCore)
library(flowStats)
library(flowClust)
library(plyr)
```

```
wsfile <- "/Users/monhait/Desktop/flow_cyto/automated_gating/data/Young_v_Adult_D30_Tcell_Spleen.wsp"
ws <- openWorkspace(wsfile)

print(ws)

## FlowJo Workspace Version 20.0
## File location: /Users/monhait/Desktop/flow_cyto/automated_gating/data
## File name: Young_v_Adult_D30_Tcell_Spleen.wsp
## Workspace is open.
##
## Groups in Workspace
##      Name Num.Samples
## 1 All Samples      10
## 2      Samples      10
```

3.2 Parse FCS files

Once this file exists as an R object, the raw FCS files are then read using the `parseWorkspace` function. This function will read the FCS files and transform, compensate, and gate according to parameters defined from the .wsp flowJo workspace. The `parseWorkspace` call requires `ws` (the .wsp workspace that was just read in from flowJo) and the name of the samples to read in. Other options may be customized based on particular needs. A new R object named `gating_set` is then created and will be a `GatingSet` object. The `isNcdf = TRUE` call saves this output to disk rather than into memory because the files are large. Here is an example of parsing FCS files.

```
gating_set <- parseWorkspace(ws, name = "Samples", path = "/Users/monhait/Desktop/flow_cyto/automated_gating/data/Young_v_Adult_D30_Tcell_Spleen.wsp")

## Parsing 10 samples
## windows version of flowJo workspace recognized.
## version X
## Creating ncdfFlowSet...
## All FCS files have the same following channels:
## Time
## SSC-H
## SSC-A
## FSC-H
## FSC-A
## BV421-H
## BV480-H
## BV510-H
## BV570-H
## BV605-H
## BV650-H
## BV711-H
## BV785-H
## BB515-H
## Alexa Fluor 532-H
## PE-H
## PE-Dazzle594-H
## PE-Cy5-H
## PerCP-Cy5.5-H
```

```
## PerCP-eFluor 710-H
## PE-Cy7-H
## APC-H
## APC-R700-H
## APC-Fire 750-H
## Zombie Nir-H
## AF-H
## BV421-A
## BV480-A
## BV510-A
## BV570-A
## BV605-A
## BV650-A
## BV711-A
## BV785-A
## BB515-A
## Alexa Fluor 532-A
## PE-A
## PE-Dazzle594-A
## PE-Cy5-A
## PerCP-Cy5.5-A
## PerCP-eFluor 710-A
## PE-Cy7-A
## APC-A
## APC-R700-A
## APC-Fire 750-A
## Zombie Nir-A
## AF-A

## done!

## loading data: /Users/monhait/Desktop/flow_cyto/automated_gating/data/fcs/SPLEEN_ADULT 1_20180823_174
## Compensating
## gating ...
## write SPLEEN_ADULT 1_20180823_174115_Unmixed.fcs_142158 to empty cdf slot...
## loading data: /Users/monhait/Desktop/flow_cyto/automated_gating/data/fcs/SPLEEN_ADULT 2_20180823_174
## Compensating
## gating ...
## write SPLEEN_ADULT 2_20180823_174227_Unmixed.fcs_130445 to empty cdf slot...
## loading data: /Users/monhait/Desktop/flow_cyto/automated_gating/data/fcs/SPLEEN_ADULT 3_20180823_174
## Compensating
## gating ...
## write SPLEEN_ADULT 3_20180823_174339_Unmixed.fcs_140448 to empty cdf slot...
## loading data: /Users/monhait/Desktop/flow_cyto/automated_gating/data/fcs/SPLEEN_ADULT 4_20180823_174
## Compensating
## gating ...
## write SPLEEN_ADULT 4_20180823_174455_Unmixed.fcs_131634 to empty cdf slot...
```

```
## loading data: /Users/monhait/Desktop/flow_cyto/automated_gating/data/fcs/SPLEEN_ADULT 5_20180823_174
## Compensating
## gating ...
## write SPLEEN_ADULT 5_20180823_174556_Unmixed.fcs_127923 to empty cdf slot...
## loading data: /Users/monhait/Desktop/flow_cyto/automated_gating/data/fcs/SPLEEN_YOUNG 1_20180823_173
## Compensating
## gating ...
## write SPLEEN_YOUNG 1_20180823_173545_Unmixed.fcs_138610 to empty cdf slot...
## loading data: /Users/monhait/Desktop/flow_cyto/automated_gating/data/fcs/SPLEEN_YOUNG 2_20180823_173
## Compensating
## gating ...
## write SPLEEN_YOUNG 2_20180823_173649_Unmixed.fcs_129312 to empty cdf slot...
## loading data: /Users/monhait/Desktop/flow_cyto/automated_gating/data/fcs/SPLEEN_YOUNG 3_20180823_173
## Compensating
## gating ...
## write SPLEEN_YOUNG 3_20180823_173758_Unmixed.fcs_134302 to empty cdf slot...
## loading data: /Users/monhait/Desktop/flow_cyto/automated_gating/data/fcs/SPLEEN_YOUNG 4_20180823_173
## Compensating
## gating ...
## write SPLEEN_YOUNG 4_20180823_173907_Unmixed.fcs_131867 to empty cdf slot...
## loading data: /Users/monhait/Desktop/flow_cyto/automated_gating/data/fcs/SPLEEN_YOUNG 5_20180823_174
## Compensating
## gating ...
## write SPLEEN_YOUNG 5_20180823_174010_Unmixed.fcs_125124 to empty cdf slot...
## done!
```

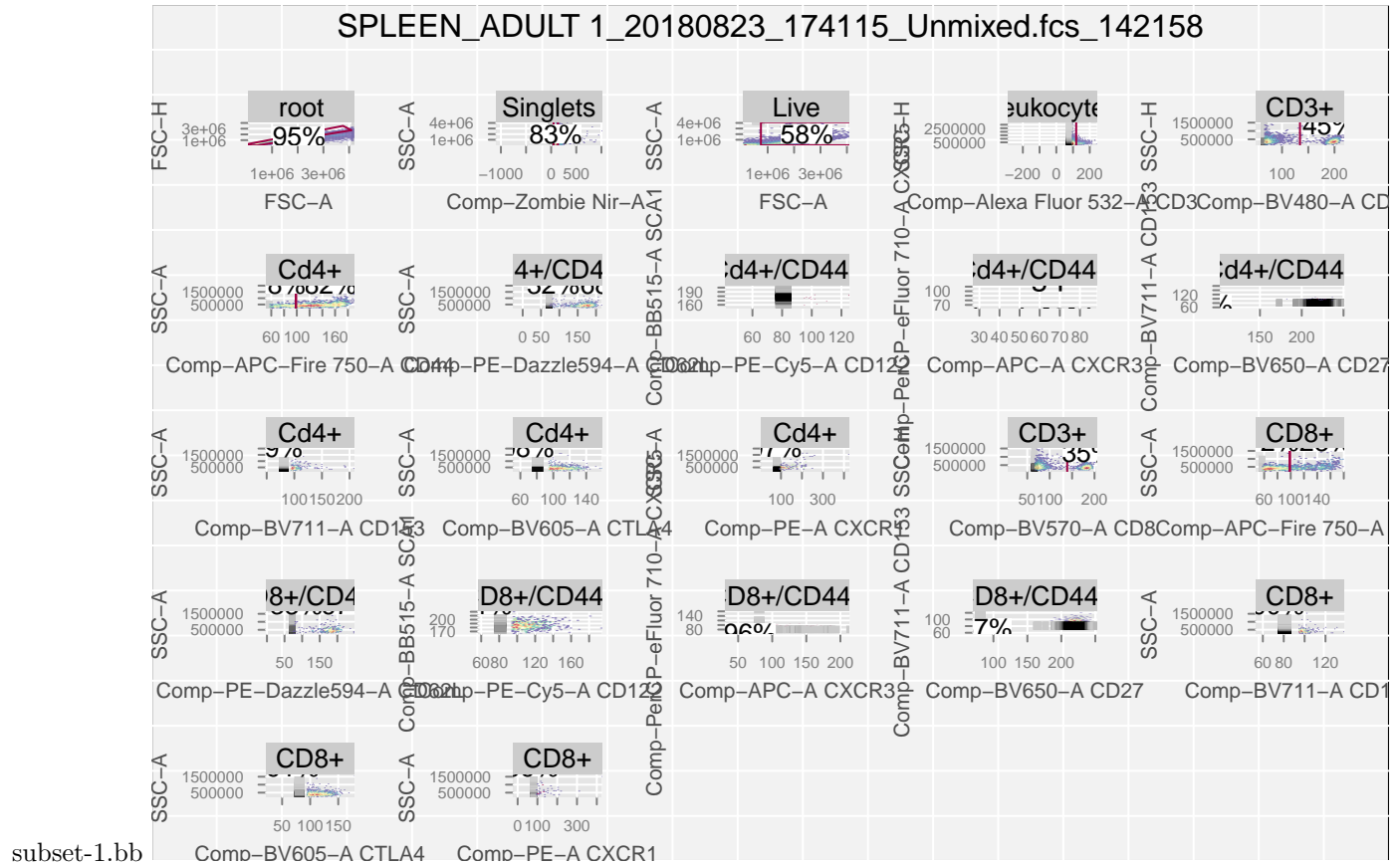
3.3 Visualize and Verify

It is helpful to now visualize both the gating template and actual gates on a subset of the data in order to verify the gating scheme. This will ensure consistency between the flowJo workspace and the manual gates recreated in R. First, save a subset of the gating_set as follows. The following saves the first FCS file of gating_set as gh.

```
gh <- gating_set[[1]]
print(gh)
```

```
## Sample: SPLEEN_ADULT 1_20180823_174115_Unmixed.fcs_142158
## GatingHierarchy with 51 gates
```

[illegible]



**Note the use of `flowWorkspace.par.set()` here. Chapter 5 of this tutorial will discuss customizations such as this one.

3.4 Clone

Now, the manual gating scheme has been replicated in R using `openCyto` and verified for consistency with the original `flowJo` workspace. The final step in this chapter will clone `gating_set` into a new R object named `auto_gating`.

```
auto_gating <- clone(gating_set)
print(auto_gating)
```

```
## A GatingSet with 10 samples
```

Chapter 4

Create .csv

The creation of a .csv gating template is arguably the most important step to automating flow cytometry analysis. The .csv template that you create will tell **openCyto** how to gate your data. Gating methods that are support currently by **openCyto** include:

quadrantGate
rangeGate
quantileGate
mindensity
tailgate
cytokine
flowClust
boundary
singletGate
transitional
polyfunctionalityGate
flowDensity

4.1 .csv Gating Template Structure

In the gating template, each row corresponds to a single cell population and the method used to gate that population. The .csv must contain 10 predefined columns that will be listed below. Most cell population names listed in columns must be uniquely and follow certain guidelines. These include:

*unique idenitifer for **alias** and **parent** columns*

no commas in **parent** columns (otherwise **opencyto** will assume the population has multiple parents)

*restrict **pop** column to quadrant-only strings (++,+)

The required 10 columns are below.

4.1.1 Template Columns

alias- unique name/identifier for cell population

pop- +/- pattern to determine which subset or quadrant will be gated

parent- unique identifier for the parent population

dims- channel or marker names for gating

gating_method- gating function (supported options listed above)

gating_args- arguments to be passed to gating function **collapseDataforGating**- data is collapsed and

replicated across all samples
groupBy- used to group samples into unique combinations
preprocessing_method- preprocessing function
preprocessing_args- arguments for preprocessing function

4.2 Creating the Template

The gating template can be created manually or assisted by the use of the `templateGen()` function. `TemplateGen()` will auto-fill the `alias`, `pop`, `parent`, and `dims` columns and the rest must be completed manually. To use `templateGen()`, you must input a `GatingHierarchy` object. In this example, that is `gh`, the subset created from `gating_set`.

```
gt <- templateGen(gh)
print(gt)
```

```
##           alias           pop
## 1         Singlets         Singlets
## 2           Live           Live
## 3       Leukocytes       Leukocytes
## 4           CD3+           CD3+
## 5           CD8+           CD8+
## 6       CXCR1-           CXCR1-
## 7       CXCR1+           CXCR1+
## 8       CTLA4-           CTLA4-
## 9       CTLA4+           CTLA4+
## 10      CD153-           CD153-
## 11      CD153+           CD153+
## 12      CD44-           CD44-
## 13 Q12: CD27- , CD153- Q12: CD27- , CD153-
## 14 Q11: CD27+ , CD153- Q11: CD27+ , CD153-
## 15 Q10: CD27+ , CD153+ Q10: CD27+ , CD153+
## 16 Q9: CD27- , CD153+ Q9: CD27- , CD153+
## 17 Q8: CXCR3- , CXCR5- Q8: CXCR3- , CXCR5-
## 18 Q7: CXCR3+ , CXCR5- Q7: CXCR3+ , CXCR5-
## 19 Q6: CXCR3+ , CXCR5+ Q6: CXCR3+ , CXCR5+
## 20 Q5: CXCR3- , CXCR5+ Q5: CXCR3- , CXCR5+
## 21 Q4: CD122- , SCA1-  Q4: CD122- , SCA1-
## 22 Q3: CD122+ , SCA1-  Q3: CD122+ , SCA1-
## 23 Q2: CD122+ , SCA1+  Q2: CD122+ , SCA1+
## 24 Q1: CD122- , SCA1+  Q1: CD122- , SCA1+
## 25           CD44+           CD44+
## 26      CD62L-           CD62L-
## 27      CD62L+           CD62L+
## 28           Cd4+           Cd4+
## 29      CXCR1-           CXCR1-
## 30      CXCR1+           CXCR1+
## 31      CTLA4-           CTLA4-
## 32      CTLA4+           CTLA4+
## 33      CD153-           CD153-
## 34      CD153+           CD153+
## 35      CD44-           CD44-
## 36 Q12: CD27- , CD153- Q12: CD27- , CD153-
## 37 Q11: CD27+ , CD153- Q11: CD27+ , CD153-
```



```

## 38 Q10: CD27+ , CD153+ Q10: CD27+ , CD153+
## 39 Q9: CD27- , CD153+ Q9: CD27- , CD153+
## 40 Q8: CXCR3- , CXCR5- Q8: CXCR3- , CXCR5-
## 41 Q7: CXCR3+ , CXCR5- Q7: CXCR3+ , CXCR5-
## 42 Q6: CXCR3+ , CXCR5+ Q6: CXCR3+ , CXCR5+
## 43 Q5: CXCR3- , CXCR5+ Q5: CXCR3- , CXCR5+
## 44 Q4: CD122- , SCA1- Q4: CD122- , SCA1-
## 45 Q3: CD122+ , SCA1- Q3: CD122+ , SCA1-
## 46 Q2: CD122+ , SCA1+ Q2: CD122+ , SCA1+
## 47 Q1: CD122- , SCA1+ Q1: CD122- , SCA1+
## 48          CD44+          CD44+
## 49          CD62L-          CD62L-
## 50          CD62L+          CD62L+
##
##          parent
## 1          root
## 2          /Singlets
## 3          /Singlets/Live
## 4          /Singlets/Live/Leukocytes
## 5          /Singlets/Live/Leukocytes/CD3+
## 6          /Singlets/Live/Leukocytes/CD3+/CD8+
## 7          /Singlets/Live/Leukocytes/CD3+/CD8+
## 8          /Singlets/Live/Leukocytes/CD3+/CD8+
## 9          /Singlets/Live/Leukocytes/CD3+/CD8+
## 10         /Singlets/Live/Leukocytes/CD3+/CD8+
## 11         /Singlets/Live/Leukocytes/CD3+/CD8+
## 12         /Singlets/Live/Leukocytes/CD3+/CD8+
## 13 /Singlets/Live/Leukocytes/CD3+/CD8+/CD44-
## 14 /Singlets/Live/Leukocytes/CD3+/CD8+/CD44-
## 15 /Singlets/Live/Leukocytes/CD3+/CD8+/CD44-
## 16 /Singlets/Live/Leukocytes/CD3+/CD8+/CD44-
## 17 /Singlets/Live/Leukocytes/CD3+/CD8+/CD44-
## 18 /Singlets/Live/Leukocytes/CD3+/CD8+/CD44-
## 19 /Singlets/Live/Leukocytes/CD3+/CD8+/CD44-
## 20 /Singlets/Live/Leukocytes/CD3+/CD8+/CD44-
## 21 /Singlets/Live/Leukocytes/CD3+/CD8+/CD44-
## 22 /Singlets/Live/Leukocytes/CD3+/CD8+/CD44-
## 23 /Singlets/Live/Leukocytes/CD3+/CD8+/CD44-
## 24 /Singlets/Live/Leukocytes/CD3+/CD8+/CD44-
## 25         /Singlets/Live/Leukocytes/CD3+/CD8+
## 26 /Singlets/Live/Leukocytes/CD3+/CD8+/CD44+
## 27 /Singlets/Live/Leukocytes/CD3+/CD8+/CD44+
## 28         /Singlets/Live/Leukocytes/CD3+
## 29         /Singlets/Live/Leukocytes/CD3+/Cd4+
## 30         /Singlets/Live/Leukocytes/CD3+/Cd4+
## 31         /Singlets/Live/Leukocytes/CD3+/Cd4+
## 32         /Singlets/Live/Leukocytes/CD3+/Cd4+
## 33         /Singlets/Live/Leukocytes/CD3+/Cd4+
## 34         /Singlets/Live/Leukocytes/CD3+/Cd4+
## 35         /Singlets/Live/Leukocytes/CD3+/Cd4+
## 36 /Singlets/Live/Leukocytes/CD3+/Cd4+/CD44-
## 37 /Singlets/Live/Leukocytes/CD3+/Cd4+/CD44-
## 38 /Singlets/Live/Leukocytes/CD3+/Cd4+/CD44-
## 39 /Singlets/Live/Leukocytes/CD3+/Cd4+/CD44-
## 40 /Singlets/Live/Leukocytes/CD3+/Cd4+/CD44-

```

```

## 41 /Singlets/Live/Leukocytes/CD3+/Cd4+/CD44-
## 42 /Singlets/Live/Leukocytes/CD3+/Cd4+/CD44-
## 43 /Singlets/Live/Leukocytes/CD3+/Cd4+/CD44-
## 44 /Singlets/Live/Leukocytes/CD3+/Cd4+/CD44-
## 45 /Singlets/Live/Leukocytes/CD3+/Cd4+/CD44-
## 46 /Singlets/Live/Leukocytes/CD3+/Cd4+/CD44-
## 47 /Singlets/Live/Leukocytes/CD3+/Cd4+/CD44-
## 48      /Singlets/Live/Leukocytes/CD3+/Cd4+
## 49 /Singlets/Live/Leukocytes/CD3+/Cd4+/CD44+
## 50 /Singlets/Live/Leukocytes/CD3+/Cd4+/CD44+
##
##          dims gating_method gating_args
## 1          FSC-A,FSC-H      <NA>      <NA>
## 2          Comp-Zombie Nir-A      <NA>      <NA>
## 3          FSC-A,SSC-A      <NA>      <NA>
## 4      Comp-Alexa Fluor 532-A,SSC-H      <NA>      <NA>
## 5          Comp-BV570-A,SSC-H      <NA>      <NA>
## 6          Comp-PE-A      <NA>      <NA>
## 7          Comp-PE-A      <NA>      <NA>
## 8          Comp-BV605-A      <NA>      <NA>
## 9          Comp-BV605-A      <NA>      <NA>
## 10         Comp-BV711-A      <NA>      <NA>
## 11         Comp-BV711-A      <NA>      <NA>
## 12         Comp-APC-Fire 750-A      <NA>      <NA>
## 13         Comp-BV650-A,Comp-BV711-A      <NA>      <NA>
## 14         Comp-BV650-A,Comp-BV711-A      <NA>      <NA>
## 15         Comp-BV650-A,Comp-BV711-A      <NA>      <NA>
## 16         Comp-BV650-A,Comp-BV711-A      <NA>      <NA>
## 17 Comp-APC-A,Comp-PerCP-eFluor 710-A      <NA>      <NA>
## 18 Comp-APC-A,Comp-PerCP-eFluor 710-A      <NA>      <NA>
## 19 Comp-APC-A,Comp-PerCP-eFluor 710-A      <NA>      <NA>
## 20 Comp-APC-A,Comp-PerCP-eFluor 710-A      <NA>      <NA>
## 21         Comp-PE-Cy5-A,Comp-BB515-A      <NA>      <NA>
## 22         Comp-PE-Cy5-A,Comp-BB515-A      <NA>      <NA>
## 23         Comp-PE-Cy5-A,Comp-BB515-A      <NA>      <NA>
## 24         Comp-PE-Cy5-A,Comp-BB515-A      <NA>      <NA>
## 25         Comp-APC-Fire 750-A      <NA>      <NA>
## 26         Comp-PE-Dazzle594-A      <NA>      <NA>
## 27         Comp-PE-Dazzle594-A      <NA>      <NA>
## 28         Comp-BV480-A,SSC-H      <NA>      <NA>
## 29         Comp-PE-A      <NA>      <NA>
## 30         Comp-PE-A      <NA>      <NA>
## 31         Comp-BV605-A      <NA>      <NA>
## 32         Comp-BV605-A      <NA>      <NA>
## 33         Comp-BV711-A      <NA>      <NA>
## 34         Comp-BV711-A      <NA>      <NA>
## 35         Comp-APC-Fire 750-A      <NA>      <NA>
## 36         Comp-BV650-A,Comp-BV711-A      <NA>      <NA>
## 37         Comp-BV650-A,Comp-BV711-A      <NA>      <NA>
## 38         Comp-BV650-A,Comp-BV711-A      <NA>      <NA>
## 39         Comp-BV650-A,Comp-BV711-A      <NA>      <NA>
## 40 Comp-APC-A,Comp-PerCP-eFluor 710-A      <NA>      <NA>
## 41 Comp-APC-A,Comp-PerCP-eFluor 710-A      <NA>      <NA>
## 42 Comp-APC-A,Comp-PerCP-eFluor 710-A      <NA>      <NA>
## 43 Comp-APC-A,Comp-PerCP-eFluor 710-A      <NA>      <NA>

```

```

## 44      Comp-PE-Cy5-A,Comp-BB515-A      <NA>      <NA>
## 45      Comp-PE-Cy5-A,Comp-BB515-A      <NA>      <NA>
## 46      Comp-PE-Cy5-A,Comp-BB515-A      <NA>      <NA>
## 47      Comp-PE-Cy5-A,Comp-BB515-A      <NA>      <NA>
## 48      Comp-APC-Fire 750-A      <NA>      <NA>
## 49      Comp-PE-Dazzle594-A      <NA>      <NA>
## 50      Comp-PE-Dazzle594-A      <NA>      <NA>
##      collapseDataForGating  groupBy preprocessing_method preprocessing_args
## 1      <NA>      <NA>      <NA>      <NA>
## 2      <NA>      <NA>      <NA>      <NA>
## 3      <NA>      <NA>      <NA>      <NA>
## 4      <NA>      <NA>      <NA>      <NA>
## 5      <NA>      <NA>      <NA>      <NA>
## 6      <NA>      <NA>      <NA>      <NA>
## 7      <NA>      <NA>      <NA>      <NA>
## 8      <NA>      <NA>      <NA>      <NA>
## 9      <NA>      <NA>      <NA>      <NA>
## 10     <NA>      <NA>      <NA>      <NA>
## 11     <NA>      <NA>      <NA>      <NA>
## 12     <NA>      <NA>      <NA>      <NA>
## 13     <NA>      <NA>      <NA>      <NA>
## 14     <NA>      <NA>      <NA>      <NA>
## 15     <NA>      <NA>      <NA>      <NA>
## 16     <NA>      <NA>      <NA>      <NA>
## 17     <NA>      <NA>      <NA>      <NA>
## 18     <NA>      <NA>      <NA>      <NA>
## 19     <NA>      <NA>      <NA>      <NA>
## 20     <NA>      <NA>      <NA>      <NA>
## 21     <NA>      <NA>      <NA>      <NA>
## 22     <NA>      <NA>      <NA>      <NA>
## 23     <NA>      <NA>      <NA>      <NA>
## 24     <NA>      <NA>      <NA>      <NA>
## 25     <NA>      <NA>      <NA>      <NA>
## 26     <NA>      <NA>      <NA>      <NA>
## 27     <NA>      <NA>      <NA>      <NA>
## 28     <NA>      <NA>      <NA>      <NA>
## 29     <NA>      <NA>      <NA>      <NA>
## 30     <NA>      <NA>      <NA>      <NA>
## 31     <NA>      <NA>      <NA>      <NA>
## 32     <NA>      <NA>      <NA>      <NA>
## 33     <NA>      <NA>      <NA>      <NA>
## 34     <NA>      <NA>      <NA>      <NA>
## 35     <NA>      <NA>      <NA>      <NA>
## 36     <NA>      <NA>      <NA>      <NA>
## 37     <NA>      <NA>      <NA>      <NA>
## 38     <NA>      <NA>      <NA>      <NA>
## 39     <NA>      <NA>      <NA>      <NA>
## 40     <NA>      <NA>      <NA>      <NA>
## 41     <NA>      <NA>      <NA>      <NA>
## 42     <NA>      <NA>      <NA>      <NA>
## 43     <NA>      <NA>      <NA>      <NA>
## 44     <NA>      <NA>      <NA>      <NA>
## 45     <NA>      <NA>      <NA>      <NA>
## 46     <NA>      <NA>      <NA>      <NA>

```

```
## 47          <NA>    <NA>          <NA>          <NA>
## 48          <NA>    <NA>          <NA>          <NA>
## 49          <NA>    <NA>          <NA>          <NA>
## 50          <NA>    <NA>          <NA>          <NA>
```

The auto-filled template will generate within the R Console and can then be saved locally with the following code.

```
write.csv(gt, "gt.csv")
```

If you choose to create the gating template manually, the same conventions must be followed. Start with a blank spreadsheet. Next, fill in the 10 required column names. From there, use the manual gating hierarchy to fill in each cell population *alias*. Fill in the remainder accordingly.

There will likely be troubleshooting involved in this process. This is a great place to start if you're seeking more information on the gating template.

4.3 Load .csv into R

When the .csv gating template is complete, it is then read into R and saved as **gt**. The gating template will be saved as a GatingTemplate object.

```
gt <- gatingTemplate("/Users/monhait/Desktop/flow_cyto/automated_gating/data/gating_template/auto_template.csv",
                    strict = FALSE, strip_extra_quotes = TRUE)
print(gt)
```

Chapter 5

Automate Gating

The flow cytometry equipment at CSU will compensate and transform the data automatically. Other tutorials may highlight the steps to compensate and transform data, but these are not relevant to CSU at this moment. In the event that equipment changes, it may be necessary to complete compensation and transformation steps to prepare data. More on the current equipment used as CSU here.

5.1 Apply Gating

At this point, you will either return to your cloned GatingSet object that contains raw FCS files or bring in new data to be gated using the same parameters. Apply **gt** to the GatingSet object, where **x = gt** and **y = data** to be gated.

```
gating(x = gt, y = auto_gating)
```

Just as before, plot both the gating hierarchy and the automated gates. You may notice extra nodes have been added to the hierarchy. Chapter 5 will highlight additional customization to remove unwanted nodes and improve upon visualization.

5.2 Plot Automated Gating

```
plotGate(auto_gating)
```

```
plot(auto_gating)
```

5.3 Population Statistics

Both counts and frequencies can be generated for analysis. This can be generate based on the analysis completed in R, or pulled directly from flowJo. To pull from flowJo, simply at **flowJo=TRUE** to either code chunk below.

Counts

```
head(getPopStats(auto_gating,satistic="count"))
```

```
##                               name Population      Parent
## 1: SPLEEN_ADULT 1_20180823_174115_Unmixed.fcs_142158  Singlets      root
## 2: SPLEEN_ADULT 1_20180823_174115_Unmixed.fcs_142158   Live    Singlets
## 3: SPLEEN_ADULT 1_20180823_174115_Unmixed.fcs_142158 Leukocytes      Live
## 4: SPLEEN_ADULT 1_20180823_174115_Unmixed.fcs_142158   CD3+ Leukocytes
## 5: SPLEEN_ADULT 1_20180823_174115_Unmixed.fcs_142158   Cd4+    CD3+
## 6: SPLEEN_ADULT 1_20180823_174115_Unmixed.fcs_142158 Cd4+/CD44+    Cd4+
##      Count ParentCount
## 1: 134378      142158
## 2: 111695      134378
## 3:  65104      111695
## 4:   4781      65104
## 5:   2151      4781
## 6:   1760      2151
```

Frequencies

```
head(getPopStats(auto_gating,satistic="freq"))
```

```
##                               name Population      Parent
## 1: SPLEEN_ADULT 1_20180823_174115_Unmixed.fcs_142158  Singlets      root
## 2: SPLEEN_ADULT 1_20180823_174115_Unmixed.fcs_142158   Live    Singlets
## 3: SPLEEN_ADULT 1_20180823_174115_Unmixed.fcs_142158 Leukocytes      Live
## 4: SPLEEN_ADULT 1_20180823_174115_Unmixed.fcs_142158   CD3+ Leukocytes
## 5: SPLEEN_ADULT 1_20180823_174115_Unmixed.fcs_142158   Cd4+    CD3+
## 6: SPLEEN_ADULT 1_20180823_174115_Unmixed.fcs_142158 Cd4+/CD44+    Cd4+
##      Count ParentCount
## 1: 134378      142158
## 2: 111695      134378
## 3:  65104      111695
## 4:   4781      65104
## 5:   2151      4781
## 6:   1760      2151
```

Chapter 6

Customization

It is possible that additional customization may be necessary when working with the `openCyto` framework. Below are three common customizations that will be outlined in this chapter.

1. Hiding unwanted nodes
2. Renaming nodes
3. Adjusting plots

6.1 Hiding unwanted nodes

When automating analysis, there may be nodes that were not predefined in the .csv gating template or nodes that may not be of interest in your particular analysis. Plotting the gating hierarchy using the `plot()` function will display this and then nodes can be hidden based on need with the following code. Below is an example of a “full” gating hierarchy and then the same hierarchy with the CD3+ node removed.

Full Hierarchy

```
plot(gh)
```

CD3+ Removed Hierarchy

To remove nodes, first save the unwanted nodes as an R object named `nodesToHide`. Next, use the code following the `lapply()` function, only replacing `gs` with your GatingSet object name.

```
nodesToHide <- "CD3+"  
lapply(nodesToHide, function(thisNode)setNode(gs, thisNode, FALSE))
```

6.2 Renaming nodes

Rename nodes based on your preferences with the following code. Within the `setNode` function, the first input is the current cell population name and the second is the desired change.

```
setNode("Live", "Viable")
```

6.3 Adjusting plots

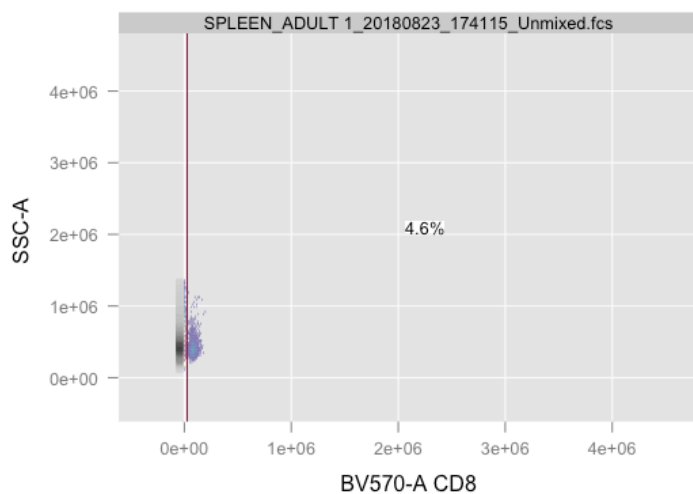
6.3.1 Adjust plot axes

As seen in chapter 2, it may be necessary to adjust the plot axes in order to best view the gates. This is done using the code below. Setting `xlim` and `ylim` to “data” adjusts plot based on the actual data range, rather than instrument specifications. Custom ranges can also be input numerically.

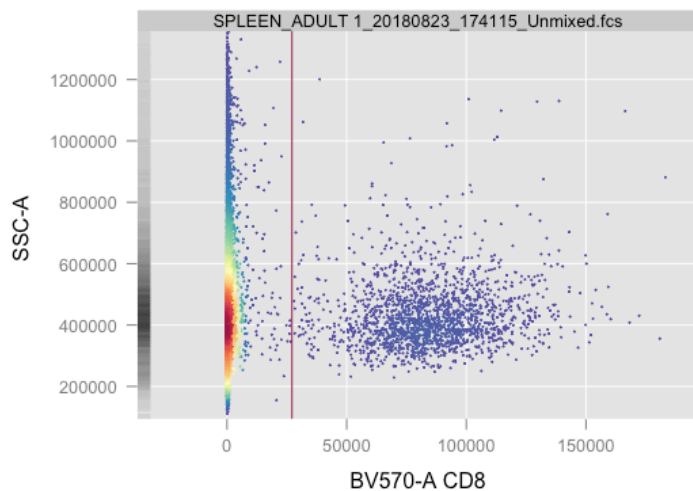
```
flowWorkspace.par.set("plotGate", list(xlim = "data",  
                                       ylim = "data"))
```

Here is a comparison of `xlim` and `ylim` set as “instrument” and then “data”.

Instrument



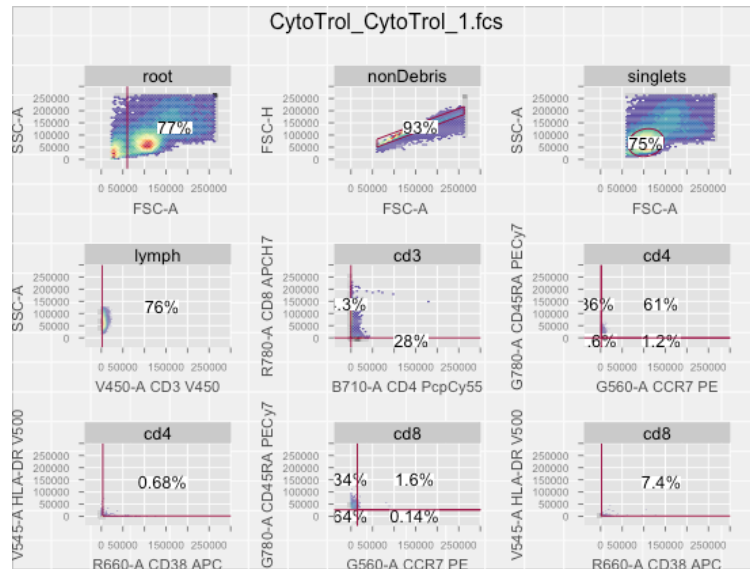
Data



6.3.2 Transform data for better visualization

Although data will not be altered in any way, transformation may allow for better visualization. The most common form of transformation for flow cytometry analysis is bioexponential. Below is a comparison of gates without transformation and gates that have been transformed.

Without Transformation



Transformed

