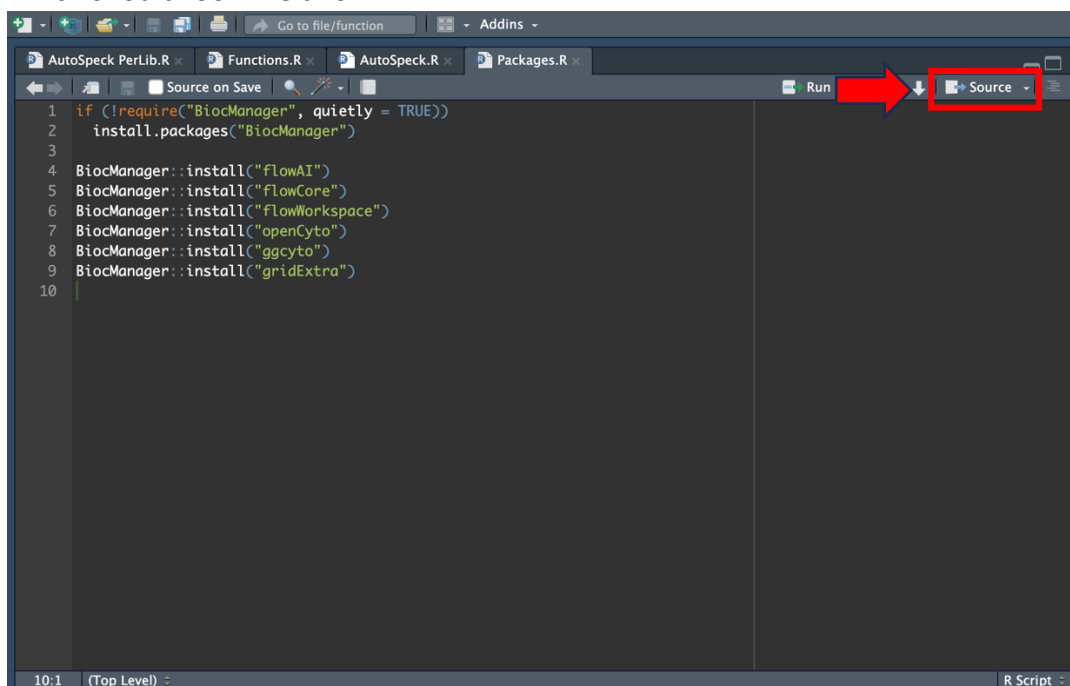


Automated Speck Assay Manual

Welcome! This manual will provide you with a guide on how to use our script for analysing your own speck assays, along with how to customise the settings to best suit your data and assay setup, even if you don't know much (or anything) about R.

Setting up the program on your own machine

1. First off, you'll need to have R and RStudio installed
2. You'll notice there are multiple files included in our repository, most of the time you won't need to worry about these, but for the first time you'll need to open "Packages.R" which should look like this:



The screenshot shows the RStudio interface with the 'Packages.R' script open in the editor. The script contains the following code:

```
1 if (!require("BiocManager", quietly = TRUE))
2   install.packages("BiocManager")
3
4 BiocManager::install("flowAI")
5 BiocManager::install("flowCore")
6 BiocManager::install("flowWorkspace")
7 BiocManager::install("openCyto")
8 BiocManager::install("ggcyto")
9 BiocManager::install("gridExtra")
10
```

In the top right corner of the RStudio window, the 'Source' button is highlighted with a red box, and a red arrow points to it from the right.

3. Click 'Source' (highlighted in the top right) to run the code, which will install the packages required to run the program
4. Once this is done, you can close this file

Setting your parameters

1. Once you have all the required packages, we can then move on to opening the main program and setting your parameters
2. First, you should load your FCS files so that they may be accessible by the program. Take your files and move them to the “Data” folder
3. Open “AutoSpeck.R”
4. The first parameter you will need to set is “Path”, which is the location of your data. Here’s an example below:

```
1 rm(list=ls())
2
3
4 source('Functions.R')
5
6 path <- 'Data/20230622_NLrp3 Library 12/P2'
7
8 fs <- fcsImportLogicle(path, T, T)
9
10 # Create an empty gating set
11 gs <- GatingSet(fs)
12
13 # Pick sample to be used as control for gating
14 gatingControl <- strtoi(readline("Please enter sample number to be used as control for gating: "))
15
16 # QA Step
17 ggcyto(gs, subset = 'root', aes(x = 'FSC.A', y = 'SSC.A')) + geom_hex(bins = 200)
18
19 ggsave(filename = paste0('total', '.png'), device = 'png',
20         path = resultDir,
21         limitsize = F,
22         width = 1920,
23         height = 1080,
24         units = 'px',
25         scale = 4,
26         plot = ggcyto(gs, subset = 'root', aes(x = 'FSC.A', y = 'SSC.A')) + geom_hex(bins = 200))
27
28 # Debris Gate
29 gate1dc(gatingSet = gs, parentPop = 'root', xchannel = 'FSC.A', name = 'debris',
30         plot = F, positive = T, range = c(0.1e5, 1e5), smoothing = 1.5, peaks = NULL,
31         save = T, controlSample = 1)
```

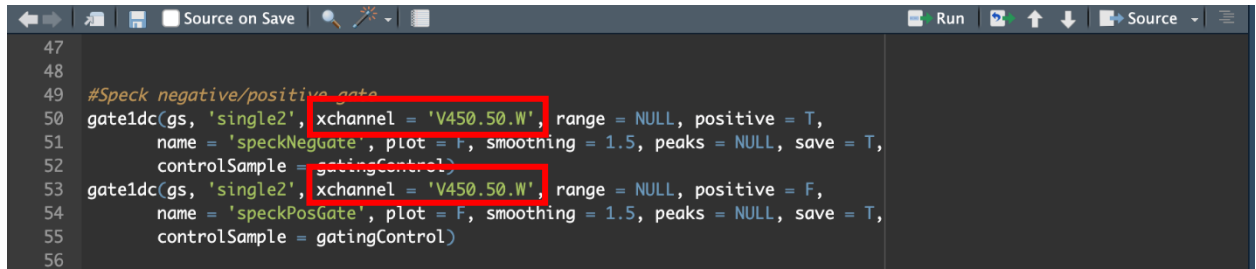
5. Next, we can move on to setting the required gates. You can see the channels available by typing “source("Functions.R")”, hitting enter and then typing the command “channelInfo(“Path to your data”)” (Channel names may be slightly altered during import of the data. If you’re running this for the first time it is good to check)
6. Our first gates are for filtering out debris and isolating single cells:

```
# Debris Gate
gate1dc(gatingSet = gs, parentPop = 'root', xchannel = 'FSC.A', name = 'debris',
        plot = F, positive = T, range = c(0.1e5, 1e5), smoothing = 1.5, peaks = NULL,
        save = T, controlSample = 1)

#Single cell gates
gate2dc(gs, 'debris', xchannel = 'SSC.W', ychannel = 'SSC.A', quantile = 0.95,
        name = 'single1', plot = F, kpop = 2, save = T, target = c(4, 3.5), controlSample = gatingControl)

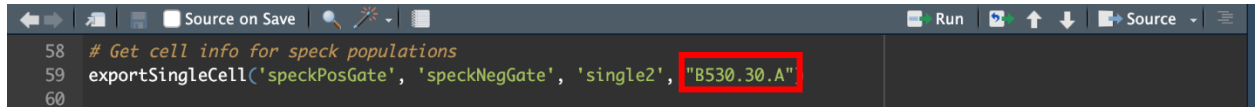
gate2dc(gs, 'single1', xchannel = 'FSC.A', ychannel = 'FSC.H', quantile = 0.95,
        name = 'single2', plot = F, kpop = 1, save = T, controlSample = gatingControl)
```

7. The Debris gate is one-dimensional, meaning it gates based on a histogram of a single channel, in this case 'FSC.A'. If you wish to use a different channel, simply change the name under the **xchannel** parameter (A detailed breakdown of each function will be included at the end of this manual if you wish to further tailor your output)
8. The next gates are for the purpose of sorting single cells. In much the same way, you can change these channels as required, however this will likely not be the case as FSC and SSC channels should always be present in your files
9. Next up are the speck gates. This is done in the following step:



```
47  
48  
49 #Speck negative/positive gate  
50 gate1dc(gs, 'single2', xchannel = 'V450.50.W', range = NULL, positive = T,  
51      name = 'speckNegGate', plot = F, smoothing = 1.5, peaks = NULL, save = T,  
52      controlSample = gatingControl)  
53 gate1dc(gs, 'single2', xchannel = 'V450.50.W', range = NULL, positive = F,  
54      name = 'speckPosGate', plot = F, smoothing = 1.5, peaks = NULL, save = T,  
55      controlSample = gatingControl)  
56
```

10. Like before, here you can input the channel you wish to perform the gating on. We have found that speck positive cells can be gated using a 1d gate on the width parameter of your speck marker. In our setup above, the cells are ASC-BFP, so we are using the 'V450.50.W' channel. It is important to make sure that the settings are the same for both the functions above, as they should generate the same gate. The only difference between them is that one gates on the positive population and the other on the negatives.
11. Finally, we can move on to how the data is analysed. Here we have a function for exporting the single cell fluorescence values for NLRP3:



```
58 # Get cell info for speck populations  
59 exportSingleCell('speckPosGate', 'speckNegGate', 'single2', "B530.30.A")  
60
```

12. The only value we need to worry about here is the final one, in the above example set to "B530.30.A". This is the channel we wish to export single cell values for, which should be a readout of NLRP3 expression. As previously, change this to whichever channel you are using depending on your fluorophore. In our example, B530.30.A is used as our NLRP3 expression is measured via GFP.

Running the Program

1. You should now be good to go! Click “Source” in the top right to run the program. You will be prompted on which channels should be transformed from linear to logistic:

```
name
$P1    FSC.A
$P2    FSC.H
$P3    FSC.W
$P4    SSC.A
$P5    SSC.H
$P6    SSC.W
$P7    V450.50.A
$P8    V450.50.W
$P9    B530.30.A
$P10   B530.30.W
$P11   YG610.20.A
$P12   YG610.20.W
$P13   Time
Please input the channels to be converted to log scale, (eg. 5:10): |
```

2. This currently accepts an input as a range of values (eg. 5:7 would be values 5, 6, 7), which represent the various fluorescence channels present in the file. Select a range that includes everything other than time and FSC. In the example above this would be 4:12.
3. You will then be prompted to input the sample number corresponding to your control sample. This will be used as the control both for setting gates and generating an ASC50 value. If following our experimental setup, WT NLRP3 is used. Keep in mind that R will sort samples by name. You can check the position of your samples by sorting your FCS files by name in file explorer (windows) or finder (mac).
4. Once the program has completed, you should be able to see a new folder inside the ‘Results’ folder, which will contain graphical representations of all the gating applied, a summary graph of relative ec50 values, the fitted curves used to generate ec50 values, and an excel sheet of all the output data.

Function Guide

Here is a list of the custom functions used in this program, along with a guide on how to use them.

`fcsImportLogicle` Import FCS data and transform channels to logicle values

Usage

```
fcsImportLogicle(  
    path,  
    clean,  
    logTrans  
)
```

Arguments

<code>path</code>	A string value containing the directory of the FCS files to be imported
<code>clean</code>	A TRUE or FALSE value indicating if the user wants data cleaned using
<code>logTrans</code>	A TRUE or FALSE value indicating if the data should be transformed to a logicle or not

`gate2dc` Create a two-dimensional gate

Usage

```
gate2dc(  
    gatingSet,  
    parentPop,  
    xchannel,  
    ychannel,  
    quantile,  
    name,  
    plot,  
    kpop,  
    save,  
    target,  
    controlSample  
)
```

Arguments

gatingSet	Name of the “GatingSet” flowWorkspace object (gs by default)
parentPop	Name of the parent gate
xchannel	X axis fluorescence channel
ychannel	Y axis fluorescence channel
quantile	How restrictive the gate is to the population; higher value indicates more restricted gate.
name	Name of the gate
plot	TRUE or FALSE value indicating if the plots should be displayed in R
kpop	The expected number of populations present
save	TRUE or FALSE value indicating if the gating plots should be saved to the output folder
target	A list containing two values, the x and y values indicating the expected location of the population.
controlSample	The sample number to be used to set gating. Already assigned by the gatingControl variable

gate1dc Create a one-dimensional gate

Usage

```
gate1dc(  
  gatingSet,  
  parentPop,  
  xchannel,  
  range,  
  name,  
  plot,  
  positive,  
  smoothing,  
  peaks,  
  save,  
  controlSample  
)
```

Arguments

gatingSet	Name of the “GatingSet” flowWorkspace object (gs by default)
parentPop	Name of the parent gate
xchannel	X axis fluorescence channel
range	A list containing the range of data

name	Name of the gate
plot	TRUE or FALSE value indicating if the plots should be displayed in R
positive	TRUE or FALSE value indicating if the positive or negative population should be gated
smoothing	Degree of smoothing applied to the histogram
peaks	Numeric vector of the locations of peaks, usually left as NULL
save	TRUE or FALSE value indicating if the gating plots should be saved to the output folder
controlSample	The sample number to be used to set gating. Already assigned by the gatingControl variable

exportSingleCell	Export single cell fluorescence values for a particular channel
------------------	-----------------------------------------------------------------

Usage

```
exportSingleCell(
  speckPosGate,
  speckNegGate,
  ascGate,
  facsChannel
)
```

Exports the global variables:

speckName: List of well IDs for each sample

speckPosRaw/speckNegRaw: List of all single cell fluorescence values for speck positive/negative population

speckAll: Combination of speckPosRaw and speckNegRaw

Arguments

speckPosGate	String containing name of gate on speck positive population
speckNegGate	String containing name of gate on speck negative population
ascGate	String containing name of gate on total population, pre speck gating
facsChannel	String containing name of FACS channel to be exported

stepBin	Bin single cell values using a single wide bin that steps through data
---------	------------------------------------------------------------------------

Usage

```
stepBin(  
    index,  
    stepLen,  
    speckAll,  
    speckPosRaw,  
    speckNegRaw  
)
```

Arguments

index	Numeric index of sample to be analysed
stepLen	Length of bin step
speckAll	Exported data from exportSingleCell
speckPosRaw	
speckNegRaw	