Compare Pho4 chimera mode of incorporation

Bin He, Lindsey Snyder

2020-12-23 (updated 12/10/23)

Contents


```
require(tidyverse)
require(flowCore)
require(flowClust)
require(openCyto)
require(ggcyto)
require(ggridges)
require(cowplot)
```

```
old <- theme_set(theme_minimal())
```

Data

Plate design

See `20201221-sample-list.txt`

Analysis

Import data and edit the meta data

import the data from the RDSS (just once) and then write it to the local disk

```
data.path = "./FCS_3.1/"
fs <- read.flowSet(path = data.path, pattern = ".fcs", # specify the file pattern
                   transformation = FALSE,  # the original values are already linearized.
                   emptyValue = FALSE,  alter.names = TRUE,   # change parameter names to R format
                   column.pattern = ".H|FSC|SSC") # only load the height variables for the fluorescent
```

Remove a duplicate sample, yH279 (1B), which wasn't used in the following analysis

```
fs0 = fs # make a copy
fs = fs0[which(!grepl("1B", sampleNames(fs)))]
```

Simplify the sample names

```
oriNames <- sampleNames(fs)
shortNames <- str_split(oriNames, pattern = " ", simplify = TRUE)[,1] %>%
  gsub("20201223-retest-Pho4-GFP_Group_", "", .)
sampleNames(fs) <- shortNames
```

Metadata

```
sample <- read.csv("./20201223-sample-list.csv") %>%
  column_to_rownames(var = "Sample")
pData(fs) <- sample
```

EDA

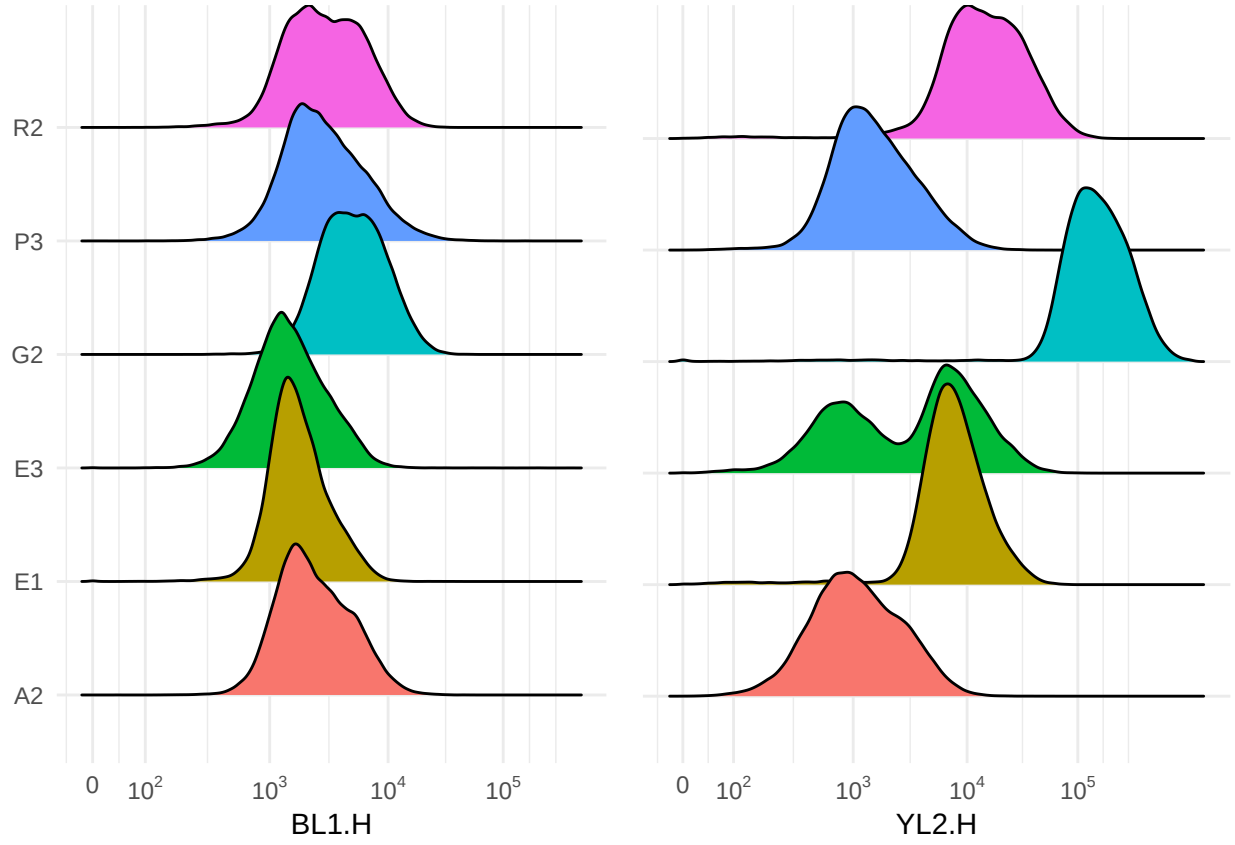
The code below demonstrates how to subset a flowSet, how to apply logicle (or other) transformations in ggcyto() (not on the original dataset)

```
test <- shortNames[c(2, 15, 17, 23, 36, 41)]
p1 <- ggplot(fs[test], aes(x = `BL1.H`)) + geom_density_ridges(aes(y = name, fill = name)) + scale_x_log
p2 <- ggplot(fs[test], aes(x = `YL2.H`)) + geom_density_ridges(aes(y = name, fill = name)) + scale_x_log
theme.elements <- list(
  #theme_cowplot(),
  theme(legend.position = "none", axis.title.y = element_blank(), axis.text.y = element_blank())
)
p <- plot_grid(p1 + theme.elements + theme(axis.text.y = element_text()), p2 + theme.elements, ncol = 2
```

```
## Picking joint bandwidth of 0.0317
```

```
## Picking joint bandwidth of 0.04
```

```
p
```



Gating

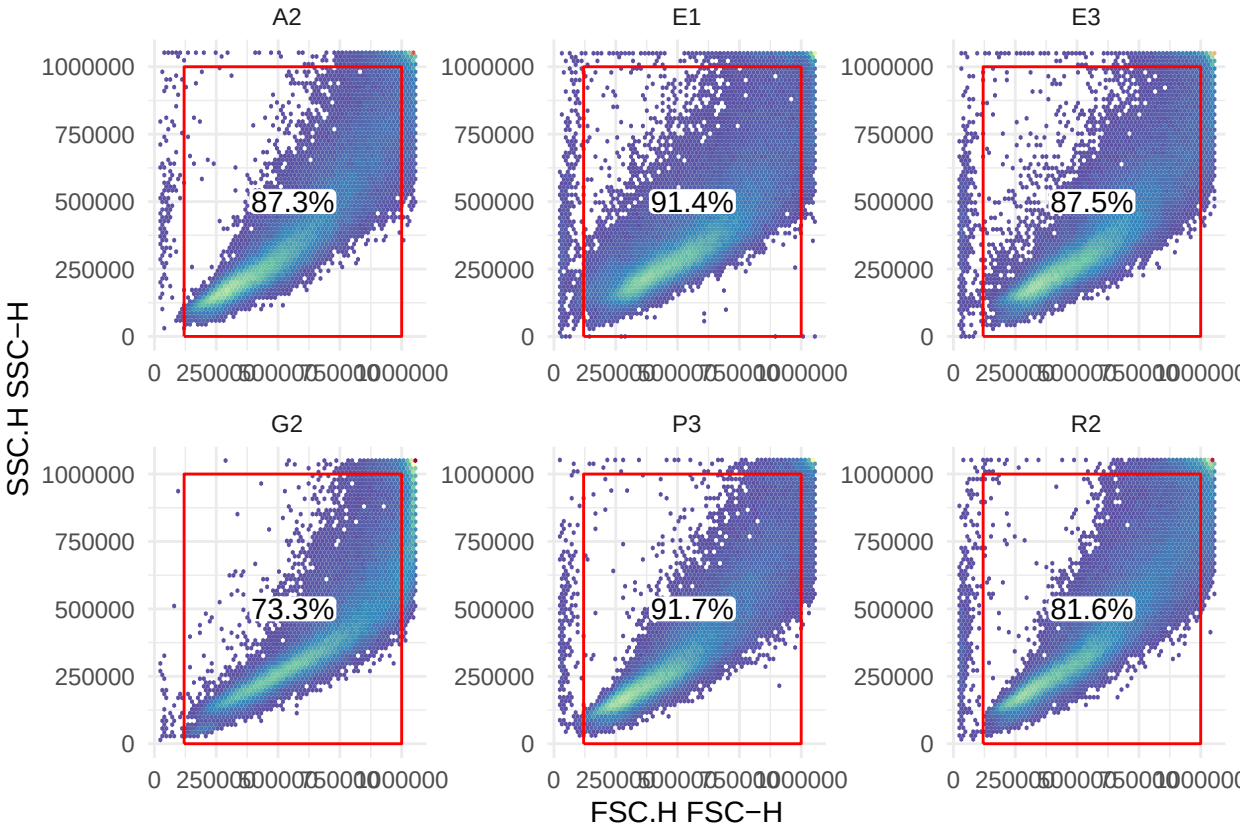
Next we use a series of plots to guide our gating strategy for identifying the population we want to work with.

Remove outliers

We first gate on FSC.H and SSC.H to remove outliers (events that are too big or too small). The Attune instrument we use can record six decades ($10^{0\text{-}10}6$), with the first two decades mostly occupied by electronic noise.

Let's first define a gate and visualize it in a plot before adding it to a GatingSet.

```
#test <- shortNames[c(3, 12*2+3, 12*5+3, 12*7+3)]
outlier.gate <- rectangleGate(filterId = "-outlier", "FSC.H" = c(1.2e5, 1e6), "SSC.H" = c(1e2, 1e6))
ggcyto(fs[test], aes(x = FSC.H, y = SSC.H), subset = "root") +
  geom_hex(bins = 64) + geom_gate(outlier.gate) + geom_stats()
```



Add this gate to the GatingSet

```
gs <- GatingSet(fs) # create a GatingSet
gs_pop_add(gs, outlier.gate, parent = "root")
```

```
## [1] 2
```

```
recompute(gs)
```

```
## done!
```

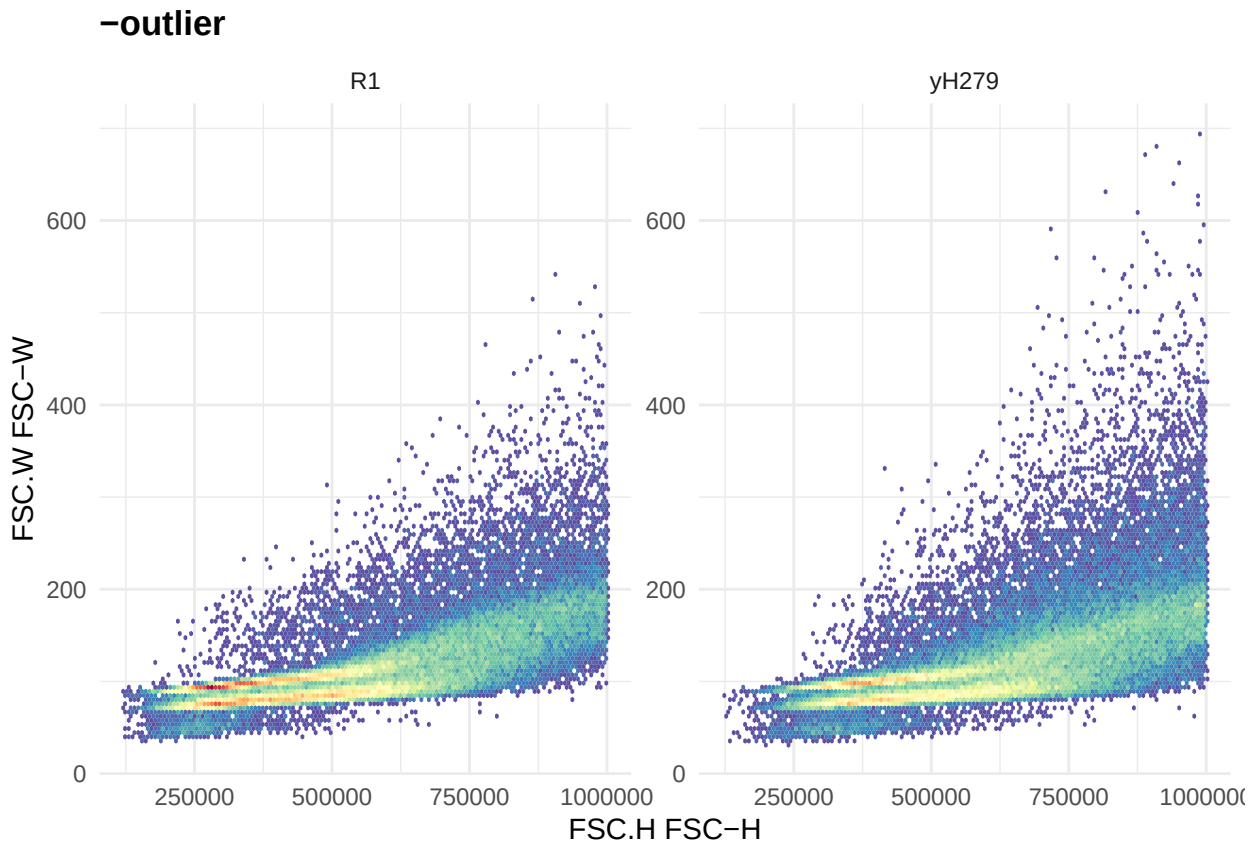
Singlet

Next let's remove multiplets on FSC.H vs FSC.W. To do this, we could either manually set up a polygon gate, or use the automatic clustering function provided by the `flowClust` package. Note that in the original implementation, the `flowClust()` function or the `tmixFilter()` version that was supposed to allow for integration with the `flowCore` package, both were designed with different downstream actions in mind than what I want to do here (visualize with `ggcyto() + geom_gate()`). The `openCyto` package written by the same group of authors who created `flowClust` and `ggcyto` has a helper function to make this possible. See this post for a discussion on alternative ways to achieve this.

Update switch to a polygon gate as the clustering is not working well. ***Update 2023/10/11*** visualize how the two populations differ in fluorescence

FSC.H vs FSC.W plot

```
ggcyto(gs[c(1, 40)], aes(x = FSC.H, y = FSC.W), subset = "-outlier") + geom_hex(bins = 128)
```



> There is clearly a second population with the same FSC.H but higher FSC.W, > indicative of doublets presence. The diffused population further to the > upper right likely represent multiplets, since they have a steeper slope, > indicating an increase in width without significant change in peak height

Is it necessary to gate out the doublets and multiplets? We can separate the singlets from the rest and compare their fluorescence readings

```
scPars <- ggcyto_par_set(limits = list(x = c(0,NA), y = c(30,300)))
#singlet.gate <- gate_flowclust_2d(ex, "FSC.H", "FSC.W", filterId = "singlet", K = 2, quantile = 0.8)
# switch to a polygongate
polygon <- matrix(c(1e5, 1e5, 1e6, 1e6, 60, 75, 135,60), ncol = 2)
colnames(polygon) <- c("FSC.H", "FSC.W")
singlet.gate <- polygonGate(filterId = "singlet", .gate = polygon)
```

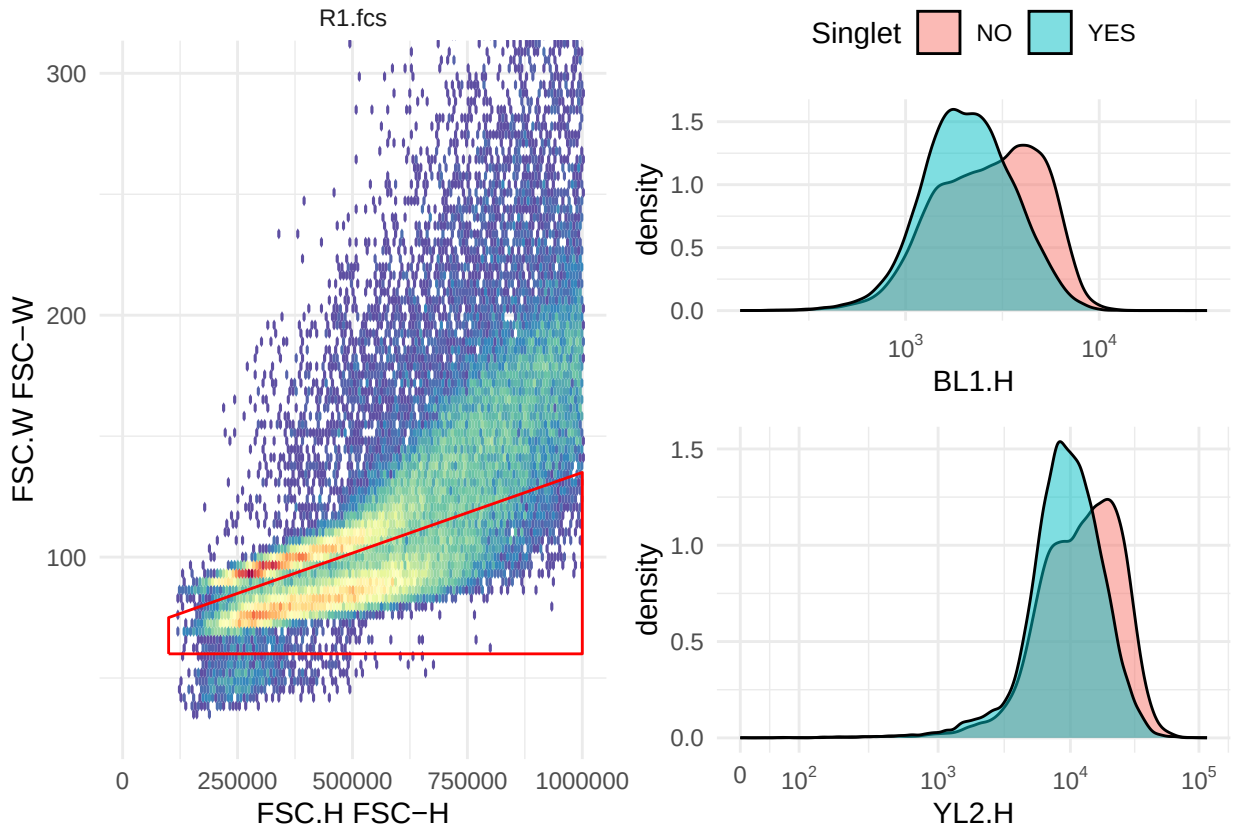
```
# use one sample for test. start by generating a filtered result
#tmp <- fs[[40]]
tmp <- gs_pop_get_data(gs[[40]], "-outlier")[[1]] %>%
  cytoframe_to_flowFrame()
tmp.ff <- filter(tmp, singlet.gate)
p.list <- list(
  scale_x_logicle(),
  scale_fill_discrete("Singlet", labels = c("NO", "YES")),
  theme(legend.position = "top")
)
```

```
p1 <- ggcyto(tmp, aes(x = FSC.H, y = FSC.W)) + geom_hex(bins = 128) + scPars +
  geom_gate(singlet.gate) +  geom_stats(type = c("percent", "count"), adjust = c(0.8, 0.2))
```

```
## Coordinate system already present. Adding new coordinate system, which will
## replace the existing one.
```

```
p2 <- ggplot(tmp, aes(x = `BL1.H`)) + geom_density(aes(fill = tmp %in% tmp.ff), alpha = 0.5) + p.list
p3 <- ggplot(tmp, aes(x = `YL2.H`)) + geom_density(aes(fill = tmp %in% tmp.ff), alpha = 0.5) + p.list
p.col <- plot_grid(p2, p3 + theme(legend.position = "none"), nrow = 2)
plot_grid(as.ggplot(p1), p.col)
```

```
## Warning: Removed 1 rows containing missing values (‘geom_label()‘).
```



Add this gate to the gatingSet

```
gs_pop_add(gs, singlet.gate, parent = "-outlier", name = "singlet")
```

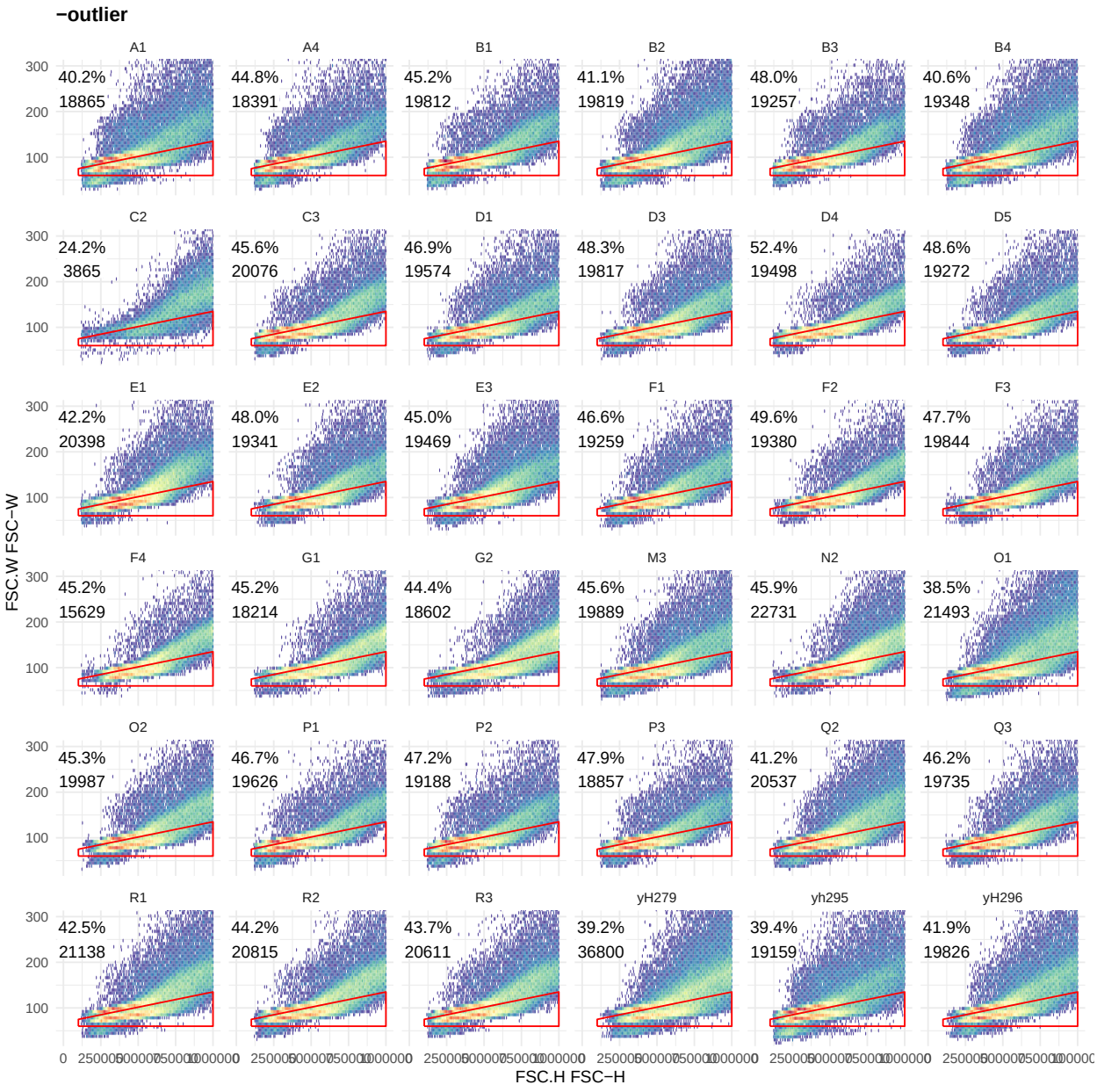
```
## [1] 3
```

```
recompute(gs)
```

```
## done!
```

```
ggcyto(gs[sample(1:length(fs), 36)], aes(x = FSC.H, y = FSC.W), subset = "-outlier") + geom_hex(bins =
  facet_wrap(~name, ncol = 6)+ scPars +
  geom_stats(type = c("percent", "count"), location = "fixed", adjust = c(125e3, 250))
```

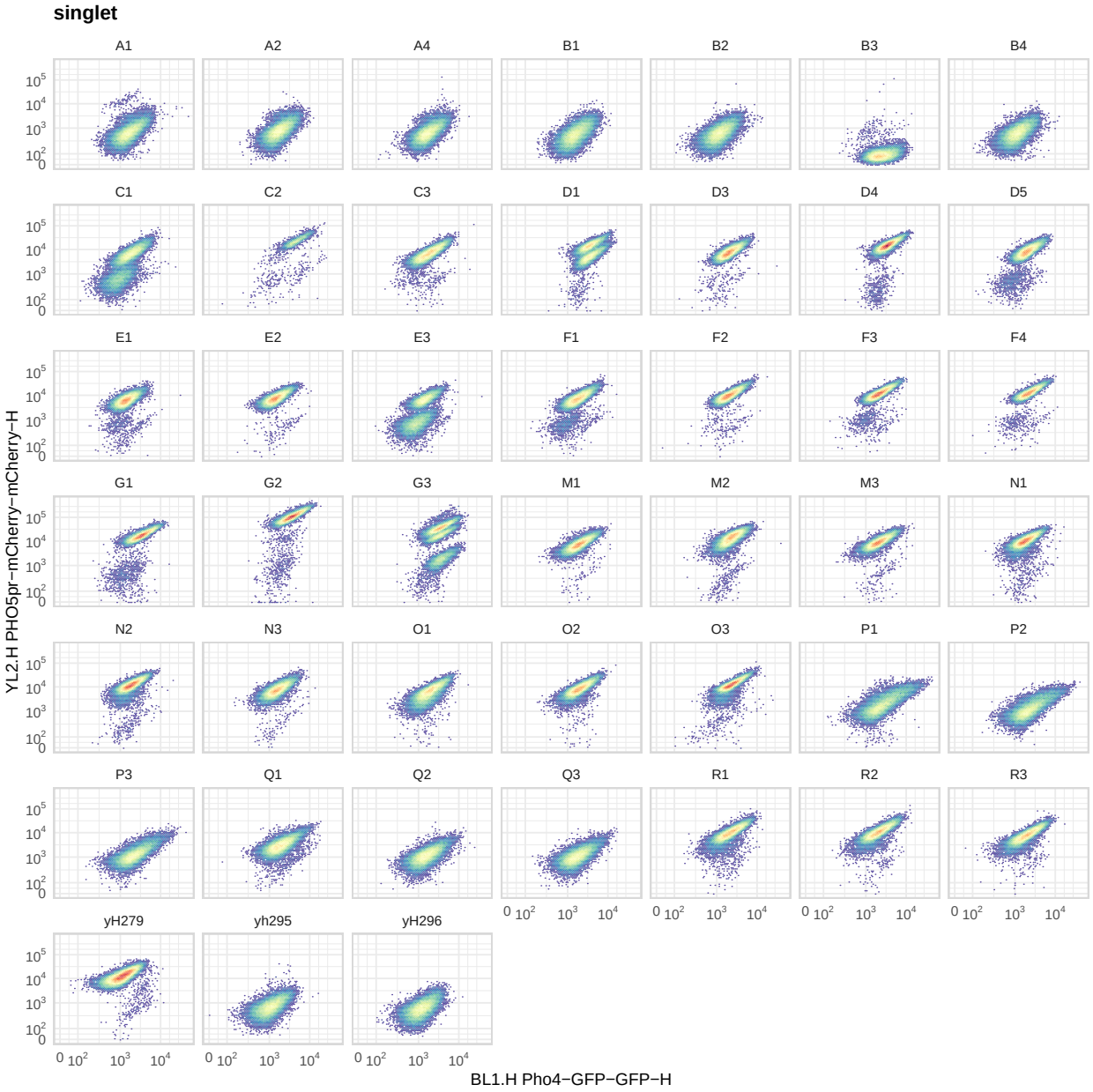
```
## Coordinate system already present. Adding new coordinate system, which will
## replace the existing one.
```



PHO5-mCherry induction

Plot the singlet events on GFP-RFP 2d space and check for the presence of multiple populations. If present, we will use flowClust to identify the main population and move forward.

```
ggcyto(gs, aes(x = BL1.H, y = YL2.H), subset = "singlet") + geom_hex(bins = 80) +
  facet_wrap(~name, ncol = 7) + scale_x_logicle() + scale_y_logicle() + panel_border()
```



Be careful when working with the GatingSet and GatingHierarchy objects – these are strictly reference classes, meaning that most of the operations work by pointers and the operations will change the underlying data. For example, the first line of the code below (commented out) obtains a pointer to the underlying data rather than making a copy of that data. any operations on it will change the original data as a result.

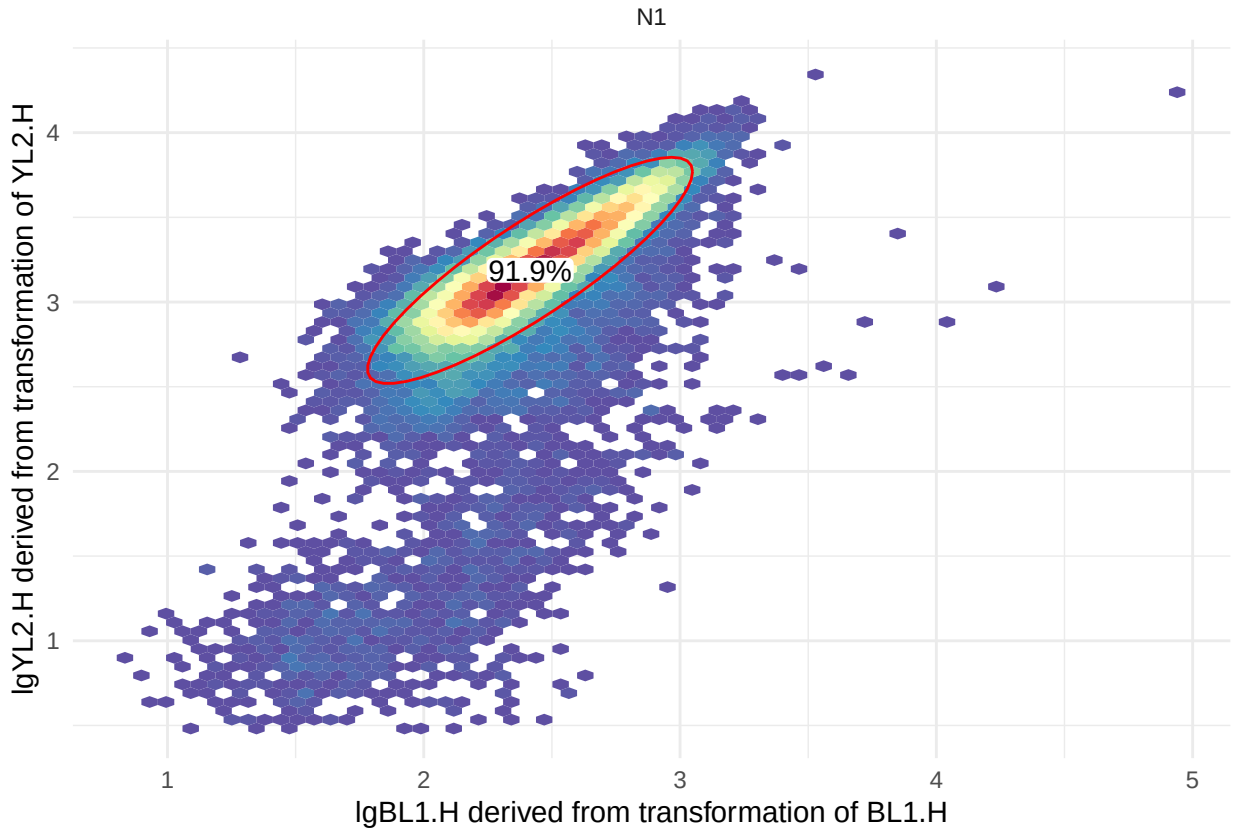
```
#ex <- gs_pop_get_data(gs, "singlet")[[1]]
ex <- fs[["N1"]]
# set the parameters for the cluster gate
```

```
k = 1; q = 0.9
# end setting
# transform the two fluorescent parameters for clustering
lgcl <- logicleTransform("induction")
ex <- transform(ex, lgBL1.H = lgcl(`BL1.H`), lgYL2.H = lgcl(`YL2.H`))
fluo.gate <- gate_flowclust_2d(ex, "lgBL1.H", "lgYL2.H", K = k, quantile = q)
```

```
## The prior specification has no effect when usePrior=no
```

```
## Using the serial version of flowClust
```

```
ggcyto(ex, aes(x = lgBL1.H, y = lgYL2.H)) + geom_hex(bins = 64) + geom_gate(fluo.gate) + geom_stats()#
```



Even though flowClust is supposed to perform its own transformation (modified Box-Cox), empirically I found the clustering seem to work better on logicle transformed data for the two fluorescent channels. Therefore I'm transforming the underlying data of the GatingSet. Note that it seems to be difficult to "create new parameters" to store the transformed data, while keeping the original data intact. Instead, the transformation functions constructed using the constructor `logicle_trans()` stores the inverse transformation functions, which can be used to perform the inverse transformation when needed. Followed the manual for GatingSet here

```
lgcl <- logicle_trans()
transList <- transformerList(c(lgBL1.H = "BL1.H", lgclYL2.H = "YL2.H"), lgcl)
transform(gs, transList)
```

```
## A GatingSet with 45 samples
```

to obtain the original data, use `gs_pop_get_data(gs[[1]], inverse.transform = TRUE)`

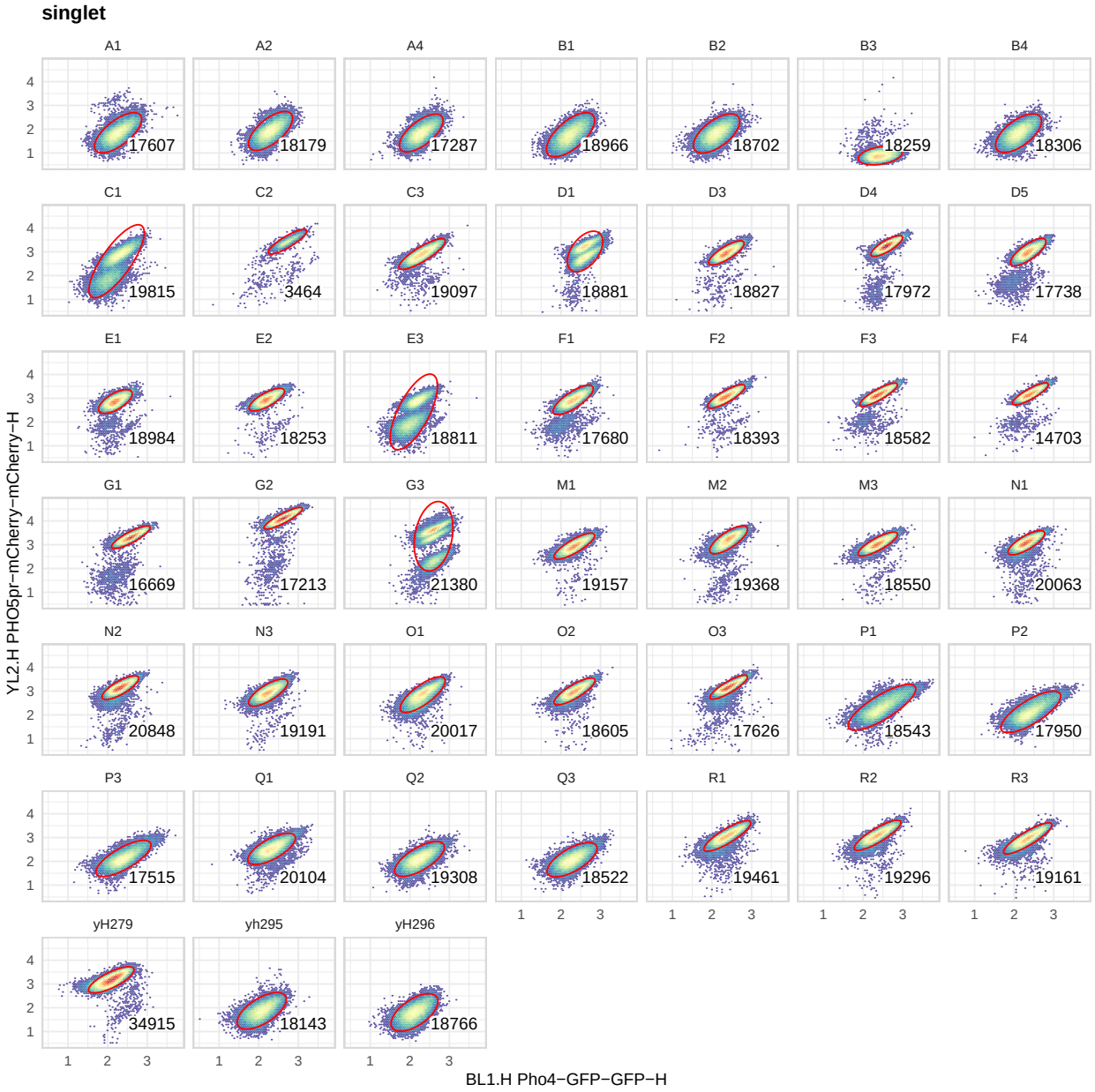
Now we can do the flowClust gating

```
dat <- gs_pop_get_data(gs, "singlet") # get parent data
inductionGate <- fsApply(dat, function(fr)
  openCyto::gate_flowclust_2d(fr, "BL1.H", "YL2.H", K = k, quantile = q)
)
gs_pop_add(gs, inductionGate, parent = "singlet", name = "induction")
```

```
## [1] 4
```

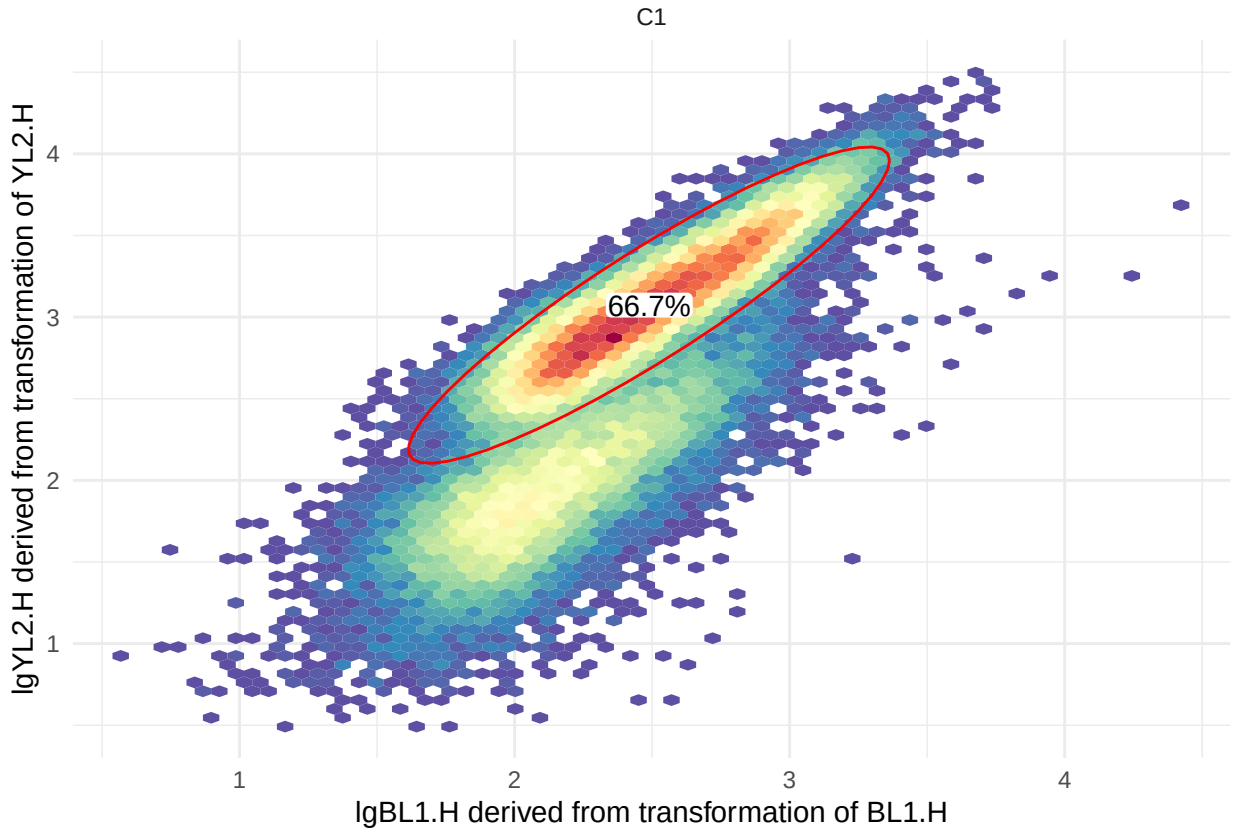
```
recompute(gs)
```

```
ggcyto(gs, aes(x = BL1.H, y = YL2.H), subset = "singlet") + geom_hex(bins = 64) +
  geom_gate("induction") + geom_stats(location = "data", adjust = c(0.8, 0.2), type = "count") +
  facet_wrap(~name, ncol = 7) + panel_border()
```



Notice that the clustering results for 239-555-1/3 showed two populations. Here we will use a 2 cluster gating strategy to select the RFP-expressing population.

```
tmp <- fs[["C1"]]
lgcl <- logicleTransform("induction")
tmp <- transform(tmp, lgBL1.H = lgcl(`BL1.H`), lgYL2.H = lgcl(`YL2.H`))
fluo.gate <- gate_flowclust_2d(tmp, "lgBL1.H", "lgYL2.H", K = 2, quantile = q, target = c(3,4))
ggcyto(tmp, aes(x = lgBL1.H, y = lgYL2.H)) + geom_hex(bins = 64) + geom_gate(fluo.gate) + geom_stats()#
```

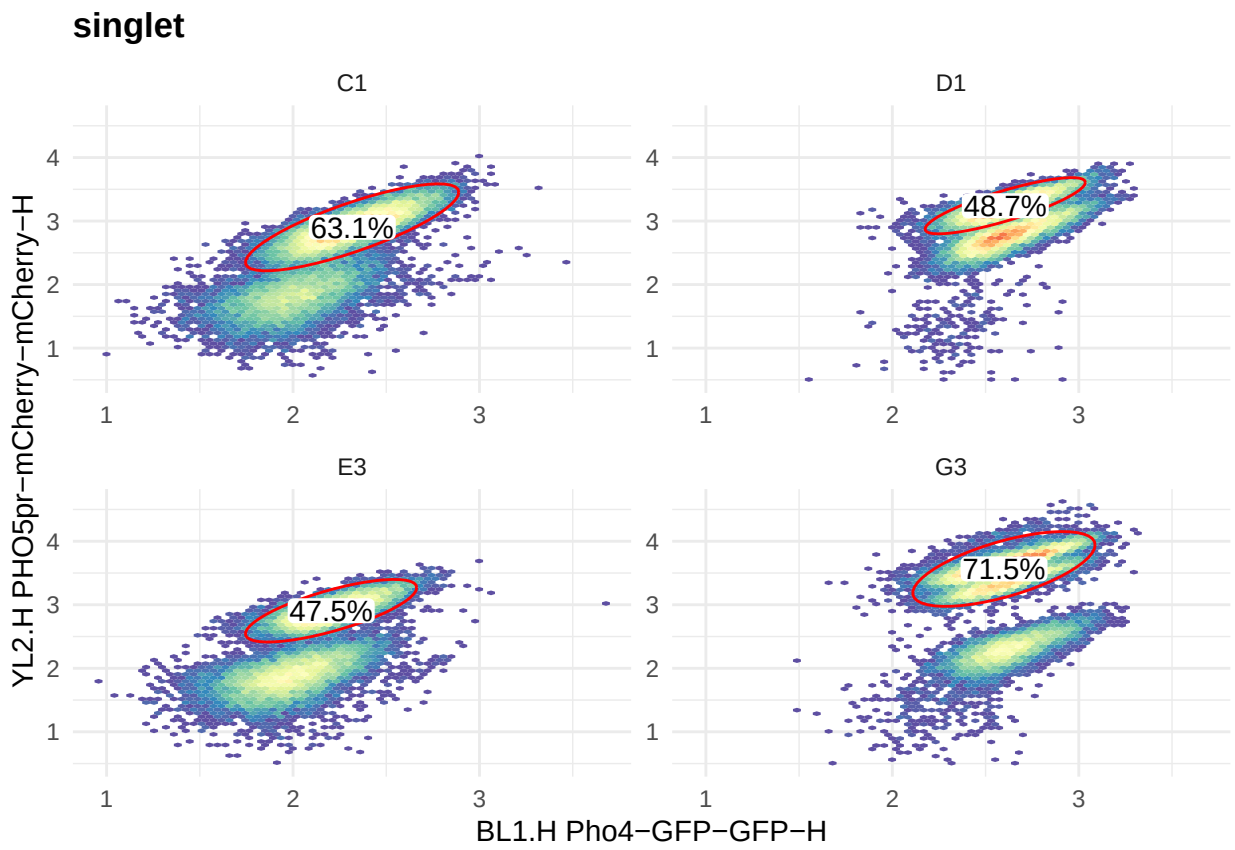


Implement the 2 cluster gates

```
list.redo <- c("C1", "D1", "E3", "G3")
newGate <- lapply(list.redo, function(x){
  gate_flowclust_2d(dat[[x]], "BL1.H", "YL2.H", K = 2, quantile = 0.9, target = c(3,4))
})
```

```
## The prior specification has no effect when usePrior=no

## Using the serial version of flowClust

## The prior specification has no effect when usePrior=no

## Using the serial version of flowClust

## The prior specification has no effect when usePrior=no

## Using the serial version of flowClust

## The prior specification has no effect when usePrior=no

## Using the serial version of flowClust
```

```
names(newGate) <- list.redo
#newGate["218-555-1"] <- gate_flowclust_2d(dat[["218-555-1"]], "BL1.H", "YL2.H", K = 3, quantile = 0.9,
ggcyto(gs[list.redo], aes(x = BL1.H, y = YL2.H), subset = "singlet") + geom_hex(bins = 64) +
  geom_gate(newGate) + geom_stats()
```



> Sample G3 still includes two pops.

update the inductionGate object

```
gs_pop_set_gate(gs[list.redo], "induction", newGate)
```

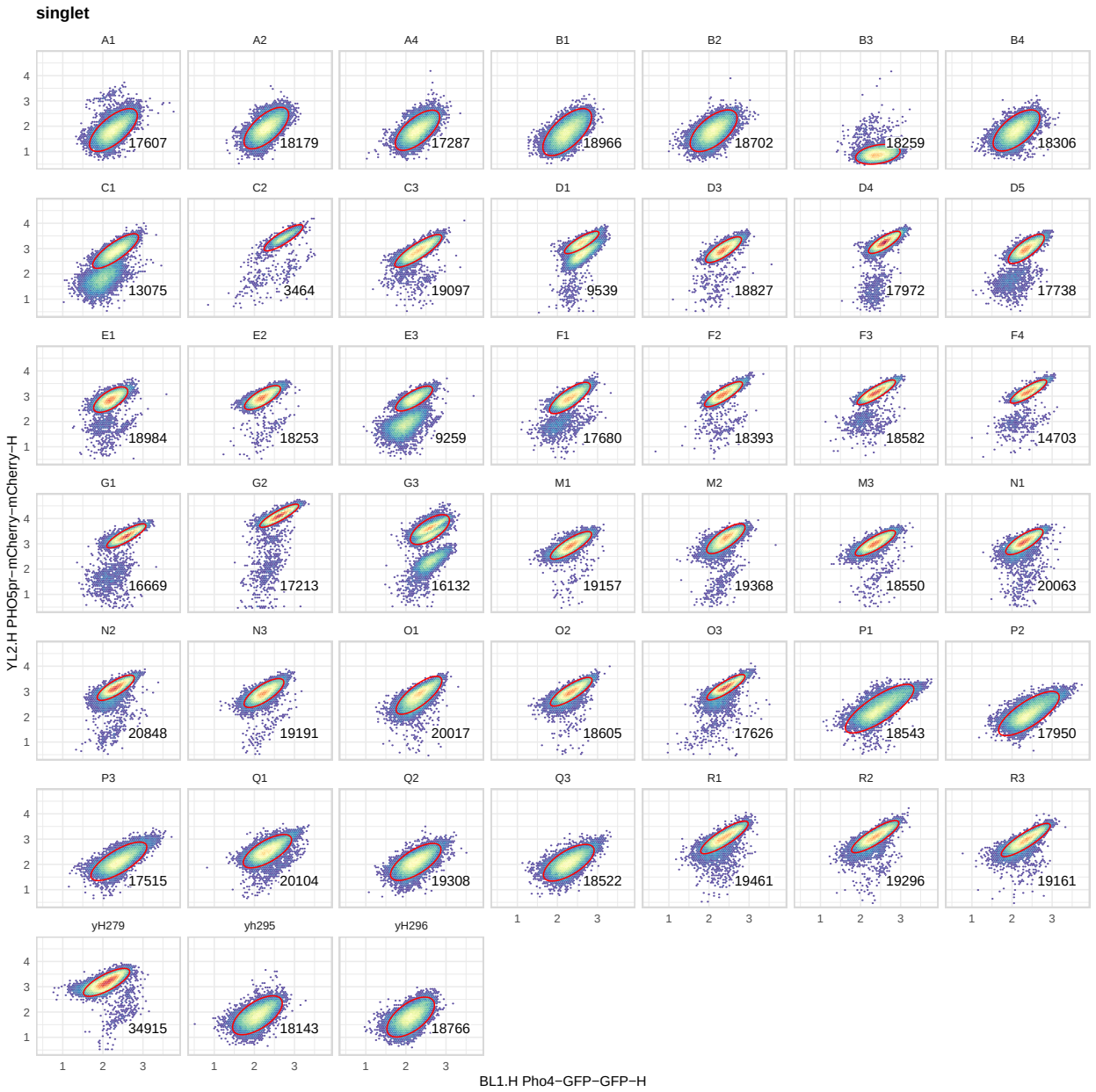
```
## [[1]]
## NULL
##
## [[2]]
## NULL
##
## [[3]]
## NULL
##
## [[4]]
## NULL
```

```
recompute(gs[list.redo], "induction")
```

```
## done!
```

Check the results

```
ggcyto(gs, aes(x = BL1.H, y = YL2.H), subset = "singlet") +
  geom_hex(bins = 64) + geom_gate("induction") +
  geom_stats(type = "count", location = "data", adjust = c(0.8, 0.2)) +
  facet_wrap(~name, ncol = 7) + panel_border()
```



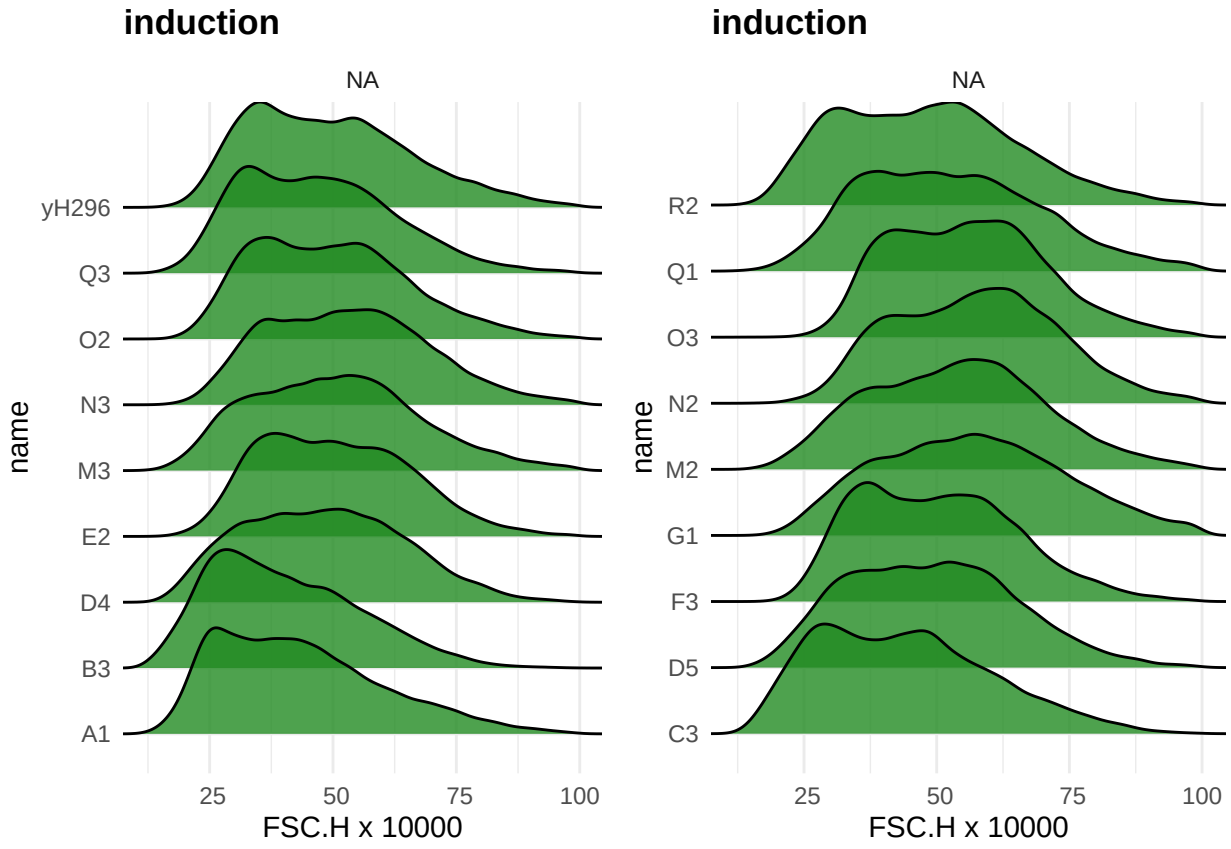
Normalization

The amount of fluorescence per cell should scale with cell size. As a result, if the cell size distribution is significantly different across samples, there may be a problem. Is that the case?

```
mult_format <- function() {
     function(x) format(x/10000,digits = 2)
}
tmp <- gs[sample(1:length(fs), 18)]
```

```
p1 <- ggcyto(tmp[1:9], aes(x = FSC.H), subset = "induction") +
  geom_density_ridges(aes(y = name), fill = "forestgreen", alpha = 0.8) +
  scale_x_continuous(labels = mult_format(), name = "FSC.H x 10000") +
  facet_wrap(~NA)
p2 <- ggcyto(tmp[10:18], aes(x = FSC.H), subset = "induction") +
  geom_density_ridges(aes(y = name), fill = "forestgreen", alpha = 0.8) +
  scale_x_continuous(labels = mult_format(), name = "FSC.H x 10000") +
  facet_wrap(~NA)
plot_grid(as.ggplot(p1), as.ggplot(p2))
```

```
## Picking joint bandwidth of 19900
## Picking joint bandwidth of 19900
```



Test normalization formula

```
# get population data
fs.out <- gs_pop_get_data(gs, y = "induction", inverse.transform = TRUE) # get the inverse transformed
# come up with an approximate FSC.H value for an average event to be used a scalar for the next step
```

```
mfsc <- 5e5 # based on the mode of the median of FSC.H from all samples
# fs.out is of the cytoframe class, which is a reference class. need to convert to flowframe for transf
# https://www.bioconductor.org/packages/devel/bioc/vignettes/flowWorkspace/inst/doc/flowWorkspace-Intro
exponent <- 1.2
```

```
# calculate ratio
norm.data <- fsApply(fs.out, function(cf) {
  cf <- cbind(cf,
              nRFP = cf[,"YL2.H"] / (cf[, "FSC.H"]/mfsc)^(exponent),
              nGFP = cf[,"BL1.H"] / (cf[, "FSC.H"]/mfsc)^(exponent))
  apply(cf[, c("FSC.H", "BL1.H", "YL2.H", "nGFP", "nRFP")], 2, median)
  }, use.exprs = TRUE) %>%
  as_tibble(rownames = "name")  %>%
  mutate(across(BL1.H:nRFP, ~ round(.x, 1)))
```

Output

The goal is to export the gated events and calculate the RFP/GFP and take the median, which will be used in downstream analyses.

Get the population stats

```
stats <- gs_pop_get_stats(gs) %>%
  as_tibble() %>%
  mutate(pop = gsub(".*/", "", pop), pop = gsub("-outlier", "cells", pop)) %>%
  pivot_wider(names_from = pop, names_prefix = "n_", values_from = count)
```

Export the data

```
# pull all info together in a single tibble
final <- left_join(as_tibble(pData(fs)), stats, by = c("name" = "sample")) %>%
  left_join(norm.data, by = "name")
#final <- final[gtools::mixedorder(final$well),] # sort by well number, thanks to
# https://stackoverflow.com/questions/58531533/r-sorting-all-columns-in-data-frame-by-an-alphanumeric-c
write_tsv(final, "20201223-gated-median-out.txt")
```