Characterization of 786-O and 786-O/VHL Cell-Derived Extracellular Vesicle Subpopulations

PART I - CD24 MEASUREMENT USING FLOW CYTOMETRY

Supplementary Information:

Data analysis and annotated source code

Guillaume Pelletier

07 November 2020

Abstract

In this analysis, we suggest a "best practices" method using open-source software (R/bioconductor) for analyzing small-particle flow cytometry data in respect to the potential biases attributed to the "swarm effect". A special experimental design is required for analysis of the swarm effect in samples containing small particles, an example of which is described here.

A deconvolution technique is applied for the first time to the analysis of swarm effect on population counts using maximum-likelihood (ML) expectation-maximization (EM) estimates for mixed distribution modeling. Constraints derived from experimental design allows for simplification of distribution parameters. We implement a noise-reduction technique to minimize noise-related artefacts across sample dilutions. We suggest a simple statistical method for interpreting population count differences across samples and across sample dilutions.

The results of significance testing for our data appear to indicate a statistically significant difference between 786-O and VHL cell-derived extracellular vesicles for the absolute count of CD24mid events, as well as for the overall proportion of CD24+ events and the overall proportion of CD24mid events. While multiple comparisons statistical testing reveals no significant differences in CD24hi population counts, fingerprinting reveals that most of the overall between-sample variation occurs in the CD24hi region. Lack of significance is perhaps owing to the difficulty of detecting rare events in very dilute samples. These data also provide with a better understanding of below-threshold particle size differences between samples: in particular, 786-O cells are found to generate significantly more CD24hi EVs smaller than the size detection threshold of the instrument compared to 786-O/VHL EVs.

Special thanks to Pr Luc H Boudreau, Pr Sandra Turcotte, the Atlantic Cancer Research Institute, NBIF and Mr David G Sebolsky for supporting this research.

Contents

1 Production environment

1.1 Required libraries

In order to install all of the required libraries listed below, you will first need to source from **bioconductor**. For more information about **bioconductor**, please visit https://www.bioconductor.com/.

When troubleshooting any of the following code, please note that the order of library imports matters, as certain packages may interfere with each other in unpredictable ways.

```
library("Biobase")
library("gridExtra")
library("reshape2")
library("HH")
library("mixtools")
library("diptest")
library("ggplot2")
library("dplyr")
library("tidyr")
library("purrr")
library("car")

# Flow cytometry specific packages
library("flowCore")
library("flowWorkspace")
library("flowUtils")
library("flowStats")
library("flowFP")
library("MetaCyto")
library("ggcyto")

# Used to generate this annotated PDF report
# https://haozhu233.github.io/kableExtra/awesome_table_in_pdf.pdf
library("kableExtra")
library("pander")
library("knitr")
```

1.2 SessionInfo()

This may be useful for troubleshooting and is included for the sake of reproducibility.

```
pander(sessionInfo(), locale=TRUE, compact=TRUE)
```

R version 4.0.2 (2020-06-22)

Platform: x86_64-apple-darwin17.0 (64-bit)

locale: en_US.UTF-8||en_US.UTF-8||en_US.UTF-8||C||en_US.UTF-8||en_US.UTF-8

attached base packages: *grid*, *parallel*, *stats*, *graphics*, *grDevices*, *utils*, *datasets*, *methods* and *base*

other attached packages: *MetaCyto(v.1.11.0)*, *car(v.3.0-9)*, *carData(v.3.0-4)*, *purrr(v.0.3.4)*, *pander(v.0.6.3)*, *kableExtra(v.1.2.1)*, *HH(v.3.1-42)*, *multcomp(v.1.4-13)*, *TH.data(v.1.0-10)*, *MASS(v.7.3-53)*, *survival(v.3.2-3)*, *mvtnorm(v.1.1-1)*, *latticeExtra(v.0.6-29)*, *diptest(v.0.75-7)*, *mixtools(v.1.2.0)*, *knitr(v.1.29)*, *reshape2(v.1.4.4)*, *tidyr(v.1.1.2)*, *dplyr(v.1.0.2)*, *flowFP(v.1.47.1)*, *flowViz(v.1.53.0)*, *lattice(v.0.20-41)*, *flowStats(v.4.1.0)*, *ggcyto(v.1.17.0)*, *ncdfFlow(v.2.35.1)*, *BH(v.1.72.0-3)*, *RcppArmadillo(v.0.9.900.3.0)*, *ggplot2(v.3.3.2)*, *flowUtils(v.1.53.0)*, *flowWorkspace(v.4.1.9)*, *flowCore(v.2.1.2)*, *Biobase(v.2.49.1)*, *BiocGenerics(v.0.35.4)* and *gridExtra(v.2.3)*

loaded via a namespace (and not attached): *readxl(v.1.3.1)*, *backports(v.1.1.10)*, *Hmisc(v.4.4-1)*, *igraph(v.1.2.5)*, *plyr(v.1.8.6)*, *ConsensusClusterPlus(v.1.53.0)*, *splines(v.4.0.2)*, *gmp(v.0.6-0)*, *fda(v.5.1.5.1)*, *digest(v.0.6.25)*, *htmltools(v.0.5.0)*, *magrittr(v.1.5)*, *checkmate(v.2.0.0)*, *CytoML(v.2.1.12)*, *cluster(v.2.1.0)*, *ks(v.1.11.7)*, *aws.signature(v.0.6.0)*, *openxlsx(v.4.1.5)*, *fastcluster(v.1.1.25)*, *RcppParallel(v.5.0.2)*, *matrixStats(v.0.56.0)*, *sandwich(v.2.5-1)*, *cytolib(v.2.1.18)*, *jpeg(v.0.1-8.1)*, *colorspace(v.1.4-1)*, *rvest(v.0.3.6)*, *rrcov(v.1.5-5)*, *haven(v.2.3.1)*, *xfun(v.0.17)*, *jsonlite(v.1.7.1)*, *crayon(v.1.3.4)*, *hexbin(v.1.28.1)*, *graph(v.1.67.1)*, *zoo(v.1.8-8)*, *glue(v.1.4.2)*, *gtable(v.0.3.0)*, *zlibbioc(v.1.35.0)*, *webshot(v.0.5.2)*, *kernlab(v.0.9-29)*, *IDPmisc(v.1.1.20)*, *Rgraphviz(v.2.33.0)*, *Rmpfr(v.0.8-1)*, *DEoptimR(v.1.0-8)*, *abind(v.1.4-5)*, *scales(v.1.1.1)*, *Rcpp(v.1.0.5)*, *viridisLite(v.0.3.0)*, *xtable(v.1.8-4)*, *htmlTable(v.2.0.1)*, *foreign(v.0.8-80)*, *mclust(v.5.4.6)*, *FlowSOM(v.1.21.0)*, *Formula(v.1.2-3)*, *tsne(v.0.1-3)*, *stats4(v.4.0.2)*, *vcd(v.1.4-7)*, *htmlwidgets(v.1.5.1)*, *httr(v.1.4.2)*, *RColorBrewer(v.1.1-2)*, *ellipsis(v.0.3.1)*, *pkgconfig(v.2.0.3)*, *XML(v.3.99-0.5)*, *nnet(v.7.3-14)*, *tidyselect(v.1.1.0)*, *rlang(v.0.4.7)*, *later(v.1.1.0.1)*, *munsell(v.0.5.0)*, *cellranger(v.1.1.0)*, *tools(v.4.0.2)*, *generics(v.0.0.2)*, *aws.s3(v.0.3.21)*, *evaluate(v.0.14)*, *stringr(v.1.4.0)*, *fastmap(v.1.0.1)*, *yaml(v.2.2.1)*, *zip(v.2.1.1)*, *robustbase(v.0.93-6)*, *nlme(v.3.1-149)*, *RBGL(v.1.65.0)*, *mime(v.0.9)*, *leaps(v.3.1)*, *xml2(v.1.3.2)*, *compiler(v.4.0.2)*, *rstudioapi(v.0.11)*, *curl(v.4.3)*, *png(v.0.1-7)*, *tibble(v.3.0.3)*, *pcaPP(v.1.9-73)*, *stringi(v.1.5.3)*, *forcats(v.0.5.0)*, *Matrix(v.1.2-18)*, *vctrs(v.0.3.4)*, *pillar(v.1.4.6)*, *lifecycle(v.0.2.0)*, *RUnit(v.0.4.32)*, *lmtest(v.0.9-38)*, *data.table(v.1.13.0)*, *corpcor(v.1.6.9)*, *httpuv(v.1.5.4)*, *R6(v.2.4.1)*, *promises(v.1.1.1)*, *KernSmooth(v.2.23-17)*, *RProtoBufLib(v.2.1.0)*, *rio(v.0.5.16)*, *codetools(v.0.2-16)*, *withr(v.2.2.0)*, *metafor(v.2.4-0)*, *S4Vectors(v.0.27.12)*, *hms(v.0.5.3)*, *rpart(v.4.1-15)*, *rmarkdown(v.2.3)*, *segmented(v.1.2-0)*, *shiny(v.1.5.0)* and *base64enc(v.0.1-3)*

1.3 Global variables and convenience functions

```
# This is where all my data files reside on my local machine
setwd("/Users/gp/Documents/Maitrise/Results/N3L2/CD24 quantification/")

# The following colours are used consistently enough in this document
colorPalette <- list(
  "red"   = "#E87D72",
  "blue"  = "#54BCC2",
  "green" = "#00AA00"
)


# Output dataframe in PDF-friendly format
printDataFrame <- function(dataframe, caption, fontsize=7) {
  knitr::kable(
    dataframe,
    format="latex",
    booktabs=TRUE,
    longtable=TRUE,
    caption=caption
  ) %>%
  kableExtra::kable_styling(
    font_size=fontsize,
    latex_options=c(
      "hold position",
      "repeat_header",
      "striped"
    )
  )
}

# kmeans() returns an error when one of the clusters turns out empty. Since in
# context we expect this to happen, we can define more useful behaviour.
safeClustering <- function(
  data,
  centers=matrix(data=c(0, 0, 0, 0), ncol=2),
  principal=1
) {
  # Linter bindings
  cluster <- NULL

  attemptClustering <- try(
    expr={
      km <- stats::kmeans(data, centers=centers)$cluster
    },
    silent=TRUE
  )

  if (class(attemptClustering) == "try-error") {
    km <- principal
  }

  km
```

```
}

# This function takes an anova object, the return value of aov(), and returns
# the p-Value of row 1 (ie.: [[1]])
getANOVApValue <- function(aov) {
  summary(aov)[["Error: Within"]][[1]][["Pr(>F)"]][1]
}
```

2 Data pre-processing

2.1 Experiment meta-data

The **Biobase** package provides a standard class called `AnnotatedDataFrame` to store and manipulate the experiment metadata.

References:

- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4509590/

Technical documentation:

- https://bioconductor.org/packages/devel/bioc/vignettes/Biobase/inst/doc/ExpressionSetIntroduction.pdf
- https://bioconductor.org/packages/devel/bioc/manuals/Biobase/man/Biobase.pdf

Experiment metadata is stored in a tab-delimited `.txt` file, which can be read using `read.table()`. In this case, as the FC500 instrument does not support integration with any of the **bioconductor** tools, the `phenoData.txt` file was created by hand in **Microsoft Excel**. Alternatively, any other plain text editor may be used. Let's take a look at the contents of this file:

```
pData <- utils::read.table(
  "phenoData.txt",
  header=TRUE,
  row.names=1,
  sep="\t",
  as.is=TRUE
)

printDataFrame(
  pData,
  "phenoData.txt",
  fontsize=7
)
```

Table 1: phenoData.txt

	ID	CellLine	N	A23187	Dilution	FacetRow	FacetCol	Temps
data/786 n1/786n1_Ca1uM4h_samp_20x 00016119 613.LMD	1	786-O	1	1	20	r1	c1	4
data/786 n1/786n1_Ca1uM4h_samp_60x 00016122 616.LMD	2	786-O	1	1	60	r1	c2	4
data/786 n1/786n1_Ca1uM4h_samp_100x 00016125 619.LMD	3	786-O	1	1	100	r1	c3	4
data/786 n1/786n1_Ca1uM4h_samp_140x 00016128 622.LMD	4	786-O	1	1	140	r1	c4	4
data/786 n1/786n1_Ca1uM4h_samp_180x 00016131 625.LMD	5	786-O	1	1	180	r1	c5	4
data/786 n1/786n1_Ca1uM4h_samp_220x 00016134 628.LMD	6	786-O	1	1	220	r1	c6	4
data/786 n2/786n2_Ca1uM4h_samp_20x 00016158 632.LMD	7	786-O	2	1	20	r2	c1	4
data/786 n2/786n2_Ca1uM4h_samp_60x 00016161 635.LMD	8	786-O	2	1	60	r2	c2	4
data/786 n2/786n2_Ca1uM4h_samp_100x 00016164 638.LMD	9	786-O	2	1	100	r2	c3	4
data/786 n2/786n2_Ca1uM4h_samp_140x 00016167 641.LMD	10	786-O	2	1	140	r2	c4	4
data/786 n2/786n2_Ca1uM4h_samp_180x 00016170 644.LMD	11	786-O	2	1	180	r2	c5	4
data/786 n2/786n2_Ca1uM4h_samp_220x 00016173 647.LMD	12	786-O	2	1	220	r2	c6	4
data/786 n3/786n3_Ca1uM4h_samp_20x 00016176 650.LMD	13	786-O	3	1	20	r3	c1	4
data/786 n3/786n3_Ca1uM4h_samp_60x 00016179 653.LMD	14	786-O	3	1	60	r3	c2	4
data/786 n3/786n3_Ca1uM4h_samp_100x 00016182 656.LMD	15	786-O	3	1	100	r3	c3	4
data/786 n3/786n3_Ca1uM4h_samp_140x 00016185 659.LMD	16	786-O	3	1	140	r3	c4	4
data/786 n3/786n3_Ca1uM4h_samp_180x 00016188 662.LMD	17	786-O	3	1	180	r3	c5	4
data/786 n3/786n3_Ca1uM4h_samp_220x 00016191 665.LMD	18	786-O	3	1	220	r3	c6	4
data/VHL n1/VHLn1_Ca1uM4h_samp_20x 00016196 670.LMD	19	VHL	1	1	20	r4	c1	4
data/VHL n1/VHLn1_Ca1uM4h_samp_60x 00016199 673.LMD	20	VHL	1	1	60	r4	c2	4
data/VHL n1/VHLn1_Ca1uM4h_samp_100x 00016202 676.LMD	21	VHL	1	1	100	r4	c3	4
data/VHL n1/VHLn1_Ca1uM4h_samp_140x 00016205 679.LMD	22	VHL	1	1	140	r4	c4	4
data/VHL n1/VHLn1_Ca1uM4h_samp_180x 00016208 682.LMD	23	VHL	1	1	180	r4	c5	4
data/VHL n1/VHLn1_Ca1uM4h_samp_220x 00016211 685.LMD	24	VHL	1	1	220	r4	c6	4

Table 1: phenoData.txt *(continued)*

	ID	CellLine	N	A23187	Dilution	FacetRow	FacetCol	Temps
data/VHL n2/VHLn2_Ca1uM4h_samp_20x 00016215 689.LMD	25	VHL	2	1	20	r5	c1	4
data/VHL n2/VHLn2_Ca1uM4h_samp_60x 00016218 692.LMD	26	VHL	2	1	60	r5	c2	4
data/VHL n2/VHLn2_Ca1uM4h_samp_100x 00016221 695.LMD	27	VHL	2	1	100	r5	c3	4
data/VHL n2/VHLn2_Ca1uM4h_samp_140x 00016224 698.LMD	28	VHL	2	1	140	r5	c4	4
data/VHL n2/VHLn2_Ca1uM4h_samp_180x 00016227 701.LMD	29	VHL	2	1	180	r5	c5	4
data/VHL n2/VHLn2_Ca1uM4h_samp_220x 00016230 704.LMD	30	VHL	2	1	220	r5	c6	4
data/VHL n3/VHLn3_Ca1uM4h_samp_20x 00016234 708.LMD	31	VHL	3	1	20	r6	c1	4
data/VHL n3/VHLn3_Ca1uM4h_samp_60x 00016237 711.LMD	32	VHL	3	1	60	r6	c2	4
data/VHL n3/VHLn3_Ca1uM4h_samp_100x 00016240 714.LMD	33	VHL	3	1	100	r6	c3	4
data/VHL n3/VHLn3_Ca1uM4h_samp_140x 00016243 717.LMD	34	VHL	3	1	140	r6	c4	4
data/VHL n3/VHLn3_Ca1uM4h_samp_180x 00016246 720.LMD	35	VHL	3	1	180	r6	c5	4
data/VHL n3/VHLn3_Ca1uM4h_samp_220x 00016249 723.LMD	36	VHL	3	1	220	r6	c6	4

The row labels correspond to the path of all the sample data files (relative to `getwd()`, as defined previously with `setwd()`). There are two cell lines in this experiment: the ATCC 786-O renal cell carcinoma cell line, as well as a 786-O/VHL line with functional VHL gene. We can see that this experiment features biological triplicates, and that each sample is measured in serial dilutions of 1:20, 1:60, 1:100, 1:140, 1:180 and 1:220. The experimental conditions corresponding to each sample file are otherwise identical (adherent cells are incubated with 1 uM A23187 in serum-starved medium for 4 hours before conditioned medium is collected, processed and analyzed).

The variables `FacetRow` and `FacetCol` have been added for convenience, as these data will be organized in an ordered grid, much like a plate. Let's label our experiment columns and rows while we're at it: these labels will be used for plotting later on.

```
# To make the faceted graph easier to interpret we can label columns and rows
facetLabels <- c(
  `c1` = "1:20",
  `c2` = "1:60",
  `c3` = "1:100",
  `c4` = "1:140",
  `c5` = "1:180",
  `c6` = "1:220",
  `r1` = "786 n1",
  `r2` = "786 n2",
  `r3` = "786 n3",
  `r4` = "VHL n1",
  `r5` = "VHL n2",
  `r6` = "VHL n3"
)
```

The `AnnotatedDataFrame` is then created directly from `pData`, an object of type `DataFrame`.

```
# Otherwise flowCore::read.flowSet() doesn't work
pData$filename <- row.names(pData)

phenoData <- new(
  "AnnotatedDataFrame",
  data=pData,
  varMetadata=data.frame(
    labelDescription=BiocGenerics::colnames(pData),
    row.names=BiocGenerics::colnames(pData)
  )
)
```

```
phenoData
```

```
## An object of class 'AnnotatedDataFrame'
##   rowNames: data/786 n1/786n1_Ca1uM4h_samp_20x 00016119 613.LMD
##     data/786 n1/786n1_Ca1uM4h_samp_60x 00016122 616.LMD ... data/VHL
##     n3/VHLn3_Ca1uM4h_samp_220x 00016249 723.LMD (36 total)
##   varLabels: ID CellLine ... filename (9 total)
##   varMetadata: labelDescription
```

2.2 Basic data structure

2.2.1 Creating a FlowSet

At this stage, we use our `AnnotatedDataFrame` to create a `FlowSet` for our experiment. A `FlowSet` is simply a collection of related `FlowFrame` objects which store individual measurements. Each row in `pData` contains the file path of each individual measurement that comprises a `FlowFrame` within the `FlowSet`. As we will see later, a `FlowFrame` object can be plotted or manipulated on its own, or a `FlowSet` object can be used to plot or manipulate a collection of `FlowFrames` in a consistent fashion. In other words, in **flowCore** parlance a `FlowFrame` is a flow cytometry measurement (which corresponds to a single row in the `pData`) whereas a `FlowSet` could be an entire experiment made up of many samples (which corresponds to the collection of all rows in the `pData`). See the **flowCore** package for more in-depth explanations.

```
# Load all samples into memory as a flowSet object
flowset <- flowCore::read.flowSet(
  path=getwd(),
  dataset=1,
  transformation=FALSE,
  alter.names=TRUE,
  phenoData=phenoData
)

flowset
```

```
## A flowSet with 36 experiments.
##
## column names(15): FS.Lin SS.Lin ... FL5.Lin TIME
```

There are other ways to access flow data in R, however I have found this method to be the most elegant. Loading `FlowSet` data using an `AnnotatedDataFrame` has the following advantages:

- **Clearer experimental design:** the experiment metadata explicitly defined in the `AnnotatedDataFrame` should clarify the experimental design.
- **Better code separation:** the file paths are all located in an external tab-delimited `.txt` file rather than being defined at any given location within the source code itself. File handling can otherwise take up a significant number of lines of code.
- **Cleaner code:** with this method, an entire experiment is set up within a `FlowSet` in one command, without the need for repetitive code for file handling or cumbersome operations on individual `FlowFrame` objects.

2.2.2 Constructing a minimal GatingSet

While the `FlowSet` contains the actual data for the experiment, we need a means to construct and manipulate a gating tree. This is where the `GatingSet` object provided by the **flowWorkspace** package comes in. A `GatingSet` object contains a reference to a `FlowSet` experiment, as well as additional objects such as transformations (eg.: linear, log, biexponential, logicle, etc), compensation matrices, gates, as well as population statistics in the gating tree. The **flowWorkspace** tools allow for non-destructive manipulation of flow cytometry data and integrate very well with **flowJo** (which is of little help to me, but is still noteworthy).

The ideal workflow as of writing this appears to be creating an XML workspace with the **flowJo** commercial software in order to import directly into a fully constructed `GatingSet` object. Unfortunately, **flowWorkspace** does not yet appear to support importing of standard GatingML-compliant files, although as we will see, some limited support for this is offered by **flowUtils**.

Technical documentation:

- https://www.bioconductor.org/packages/release/bioc/html/flowWorkspace.html

We start by constructing a minimal `GatingSet` for the experiment.

```
gatingset <- flowWorkspace::GatingSet(flowset)
gatingset
```

```
## A GatingSet with 36 samples
```

This `GatingSet` is currently quite useless, but we will be making good use of it shortly.

2.2.3 Accessing instrument settings

While this section may at first glance appear unimportant or to be full of gibberish and utter nonsense, there are at least two good reasons for diving into the instrument settings included in `.FCS` or `.LMD` files:

- **Low-level data access: TODO: Include explanation of all the things you can find in there**
- **MIFlowCyt:** https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2773297/

As per MIFlowCyt, the following list of instrument and acquisition parameters is included for informational purposes.

```
# Extract experiment's instrument metadata
keywords <- flowCore::keyword(gatingset[[1]])

# Format
keywordsDataFrame <- data.frame(BiocGenerics::t(data.frame(keywords)))
keywordsDataFrame$key <- BiocGenerics::rownames(keywordsDataFrame)
keywordsDataFrame <- keywordsDataFrame[c(2, 1)] # Reorder columns

BiocGenerics::colnames(keywordsDataFrame) <- c("key", "value")
BiocGenerics::rownames(keywordsDataFrame) <- c()

# Print
printDataFrame(
  keywordsDataFrame,
  "Instrument settings",
  fontsize=6
)
```

Table 2: Instrument settings

key	value
FCSversion	3
X.ABSCALFACTOR	NOT SET
X.Acquisition.Protocol.Offset	0000565308
X.AUX_SIGNAL	N/A
X.BARCODE	NoRead
X.BASELINEOFFSET	OFF
X.BUILDNUMBER	3707
X.CAROUSEL	1
X.COMPENSATIONMODE	Advanced
X.CYTOMETERID	AG10041
X.DISCRIMINATOR	FS,1
X.FILEGUID	9DF5EB9B1AA39E49A66D67F4D2E0DDC7
X.HENELASER	ENABLED
X.LOCATION	
X.P10ADDRESS	14
X.P10C	ARITHMETIC
X.P10GAIN	1.000000
X.P10Q	FL1 Lin
X.P10U	
X.P10X	0.0, 0.0
X.P10Z	ON
X.P11ADDRESS	18
X.P11C	ARITHMETIC
X.P11GAIN	1.000000
X.P11Q	FL2 Lin

Table 2: Instrument settings *(continued)*

key	value
X.P11U	
X.P11X	0.0, 0.0
X.P11Z	ON
X.P12ADDRESS	22
X.P12C	ARITHMETIC
X.P12GAIN	1.000000
X.P12Q	FL3 Lin
X.P12U	
X.P12X	0.0, 0.0
X.P12Z	ON
X.P13ADDRESS	26
X.P13C	ARITHMETIC
X.P13GAIN	1.000000
X.P13Q	FL4 Lin
X.P13U	
X.P13X	0.0, 0.0
X.P13Z	ON
X.P14ADDRESS	32
X.P14C	ARITHMETIC
X.P14GAIN	1.000000
X.P14Q	FL5 Lin
X.P14U	
X.P14X	0.0, 0.0
X.P14Z	ON
X.P15ADDRESS	4
X.P15C	ARITHMETIC
X.P15Q	TIME
X.P15U	
X.P15X	1023.0,500.0
X.P15Z	ON
X.P1ADDRESS	6
X.P1C	ARITHMETIC
X.P1GAIN	2.000000
X.P1Q	FS Lin
X.P1U	
X.P1X	0.0, 0.0
X.P1Z	ON
X.P2ADDRESS	8
X.P2C	ARITHMETIC
X.P2GAIN	5.000000
X.P2Q	SS Lin
X.P2U	
X.P2X	0.0, 0.0
X.P2Z	ON
X.P3ADDRESS	15
X.P3C	GEOMETRIC
X.P3GAIN	1.000000
X.P3Q	FL1 Log
X.P3U	LOG Channels
X.P3Z	ON
X.P4ADDRESS	19
X.P4C	GEOMETRIC
X.P4GAIN	1.000000
X.P4Q	FL2 Log
X.P4U	LOGChannels
X.P4Z	ON
X.P5ADDRESS	23
X.P5C	GEOMETRIC
X.P5GAIN	1.000000
X.P5Q	FL3 Log
X.P5U	LOG Channels
X.P5Z	ON
X.P6ADDRESS	27
X.P6C	GEOMETRIC
X.P6GAIN	1.000000
X.P6Q	FL4 Log
X.P6U	
X.P6Z	ON

Table 2: Instrument settings *(continued)*

key	value
X.P7ADDRESS	33
X.P7C	GEOMETRIC
X.P7GAIN	1.000000
X.P7Q	FL5 Log
X.P7U	
X.P7Z	ON
X.P8ADDRESS	7
X.P8C	GEOMETRIC
X.P8GAIN	2.000000
X.P8Q	FS Log
X.P8U	
X.P8Z	ON
X.P9ADDRESS	9
X.P9C	GEOMETRIC
X.P9GAIN	5.000000
X.P9Q	SS Log
X.P9U	
X.P9Z	ON
X.PANEL	MitoSafe with CD44 (FL1 FITC)
X.RATIO_DENOMINATOR	N/A
X.RATIO_NUMERATOR	N/A
X.RATIODENOMINATORMUX	8
X.RATIONUMERATORMUX	6
X.RESAVEDFILE	RUNTIME PROTOCOL
X.SAMPLEID1	786n1_Ca1uM4h_samp_20x
X.SAMPLEID2	
X.SAMPLEID3	
X.SAMPLEID4	
X.SETTINGSFILE	MPs - MTDRFM et CD44 (FITC).PRO
X.SETTINGSFILEDATETIME	16-May-2018, 15:24:45
X.STOPREASON	PROTOCOL
X.TARGETLASERPOWER	20 mW
X.TUBENO	1
X.Y2KDATE	20180516
X.BEGINANALYSIS	0
X.BEGINDATA	4325
X.BEGINSTEXT	0
X.BTIM	16:53:04
X.BYTEORD	4,3,2,1
X.CELLS	
X.CYT	Cytomics FC 500
X.DATATYPE	F
X.DATE	16-May-18
X.DFC1TO2	3.00
X.DFC1TO3	5.20
X.DFC1TO4	0.00
X.DFC1TO5	0.00
X.DFC2TO1	0.00
X.DFC2TO3	0.00
X.DFC2TO4	0.00
X.DFC2TO5	0.00
X.DFC3TO1	0.00
X.DFC3TO2	0.00
X.DFC3TO4	0.00
X.DFC3TO5	0.00
X.DFC4TO1	1.60
X.DFC4TO2	0.00
X.DFC4TO3	2.60
X.DFC4TO5	25.00
X.DFC5TO1	0.00
X.DFC5TO2	0.00
X.DFC5TO3	0.00
X.DFC5TO4	0.00
X.ENDANALYSIS	0
X.ENDDATA	1118044
X.ENDSTEXT	0
X.ETIM	17:01:24
X.EXP	

Table 2: Instrument settings *(continued)*

key	value
X.FIL	786n1_Ca1uM4h_samp_20x 00016119 613.LMD
X.INST	Universite de Moncton
X.INSTADDRESS	
X.MODE	L
X.NEXTDATA	0
X.OP	El Guigui
X.P10B	32
X.P10E	0,0
X.P10G	1.000000
X.P10N	FL1.Lin
X.P10R	1024
X.P10S	FL1 Lin
X.P10V	440
X.P11B	32
X.P11E	0,0
X.P11G	1.000000
X.P11N	FL2.Lin
X.P11R	1024
X.P11S	FL2 Lin
X.P11V	555
X.P12B	32
X.P12E	0,0
X.P12G	1.000000
X.P12N	FL3.Lin
X.P12R	1024
X.P12S	FL3 Lin
X.P12V	250
X.P13B	32
X.P13E	0,0
X.P13G	1.000000
X.P13N	FL4.Lin
X.P13R	1024
X.P13S	FL4 Lin
X.P13V	306
X.P14B	32
X.P14E	0,0
X.P14G	1.000000
X.P14N	FL5.Lin
X.P14R	1024
X.P14S	FL5 Lin
X.P14V	250
X.P15B	32
X.P15E	0,0
X.P15N	TIME
X.P15R	1024
X.P15S	TIME
X.P15V	0
X.P1B	32
X.P1E	0,0
X.P1G	2.000000
X.P1N	FS.Lin
X.P1R	1024
X.P1S	FS Lin
X.P1V	503
X.P2B	32
X.P2E	0,0
X.P2G	5.000000
X.P2N	SS.Lin
X.P2R	1024
X.P2S	SS Lin
X.P2V	508
X.P3B	32
X.P3E	0,0
X.P3G	1.000000
X.P3N	FL1.Log
X.P3R	1024
X.P3S	FL1 Log
X.P3V	440
X.P4B	32

Table 2: Instrument settings *(continued)*

key	value
X.P4E	0,0
X.P4G	1.000000
X.P4N	FL2.Log
X.P4R	1024
X.P4S	FL2 Log
X.P4V	555
X.P5B	32
X.P5E	0,0
X.P5G	1.000000
X.P5N	FL3.Log
X.P5R	1024
X.P5S	FL3 Log
X.P5V	250
X.P6B	32
X.P6E	0,0
X.P6G	1.000000
X.P6N	FL4.Log
X.P6R	1024
X.P6S	FL4 Log
X.P6V	306
X.P7B	32
X.P7E	0,0
X.P7G	1.000000
X.P7N	FL5.Log
X.P7R	1024
X.P7S	FL5 Log
X.P7V	250
X.P8B	32
X.P8E	0,0
X.P8G	2.000000
X.P8N	FS.Log
X.P8R	1024
X.P8S	FS Log
X.P8V	503
X.P9B	32
X.P9E	0,0
X.P9G	5.000000
X.P9N	SS.Log
X.P9R	1024
X.P9S	SS Log
X.P9V	508
X.PAR	15
X.PROJ	
X.RUNNUMBER	00016119
X.SMNO	00016119
X.SRC	
X.SYS	CXP
X.TOT	18562
ACQTIME	500.0
FILENAME	/private/var/folders/23/z5mf0z_x2sj5xw7m95mdhgyh0000gn/T/Rtmp32Qboy/file1277e5ca78ba6/data_786 n1_786n1_Ca1uM4h_samp_20x 00016119 613.LMD
GUID	data_786 n1_786n1_Ca1uM4h_samp_20x 00016119 613.LMD
GUID.original	data/786 n1/786n1_Ca1uM4h_samp_20x 00016119 613.LMD
ORIGINALGUID	data_786 n1_786n1_Ca1uM4h_samp_20x 00016119 613.LMD
TESTFILE	MPs - MTDRFM et CD44 (FITC)
TESTNAME	MPs - MTDRFM et CD44 (FITC)
X.CYTOLIB_VERSION	2.1.18
transformation	applied
flowCore_.P1Rmax	1024
flowCore_.P1Rmin	0
flowCore_.P2Rmax	1024
flowCore_.P2Rmin	0
flowCore_.P3Rmax	1024
flowCore_.P3Rmin	0
flowCore_.P4Rmax	1024
flowCore_.P4Rmin	0
flowCore_.P5Rmax	1024
flowCore_.P5Rmin	0
flowCore_.P6Rmax	1024

Table 2: Instrument settings *(continued)*

key	value
flowCore_.P6Rmin	0
flowCore_.P7Rmax	1024
flowCore_.P7Rmin	0
flowCore_.P8Rmax	1024
flowCore_.P8Rmin	0
flowCore_.P9Rmax	1024
flowCore_.P9Rmin	0
flowCore_.P10Rmax	1024
flowCore_.P10Rmin	0
flowCore_.P11Rmax	1024
flowCore_.P11Rmin	0
flowCore_.P12Rmax	1024
flowCore_.P12Rmin	0
flowCore_.P13Rmax	1024
flowCore_.P13Rmin	0
flowCore_.P14Rmax	1024
flowCore_.P14Rmin	0
flowCore_.P15Rmax	1024
flowCore_.P15Rmin	0

2.3 GatingML

```
flowEnv <- new.env()
flowUtils::read.gatingML("GatingML.xml", flowEnv)
```

2.3.1 Applying the gating tree onto the GatingSet

In theory, with an appropriate GatingML-compliant file this should be incredibly straightforward. However, support for standard GatingML-compliant files in **flowWorkspace** is nonexistent. From what I have been able to figure out, support for this in **flowUtils** is either broken, incomplete or improperly documented. As a result, the following code is not as elegant a demonstration as it could be, but this should not detract from the fact that the R/bioconductor open sourcee flow cytometry tools are extremely powerful.

References:

- https://onlinelibrary.wiley.com/doi/epdf/10.1002/cyto.a.20637

Technical documentation:

- http://flowcyt.sourceforge.net/gating/latest.pdf
- http://bioconductor.riken.jp/packages/3.7/bioc/manuals/flowUtils/man/flowUtils.pdf

Let's begin this mess by defining two convenience functions (which I hope to get rid of in the near future). We can then construct a gating tree from the gates defined in the GatingML file.

```
getGate <- function(gateName) {
  flowEnv[[gateName]]@filters[[1]]
}

getGateParent <- function(gateName) {
  flowEnv[[gateName]]@filters[[2]]@name
}

flowWorkspace::gs_pop_add(
  gs=gatingset,
  gate=getGate("Boundary"),
  parent=getGateParent("Boundary")
)

flowWorkspace::gs_pop_add(
  gs=gatingset,
  gate=getGate("NonNoise"),
  parent=getGateParent("NonNoise")
)

flowWorkspace::gs_pop_add(
  gs=gatingset,
  gate=getGate("BeadsA"),
  parent=getGateParent("BeadsA")
)

flowWorkspace::gs_pop_add(
  gs=gatingset,
  gate=getGate("BeadsB"),
  parent=getGateParent("BeadsB")
)
```

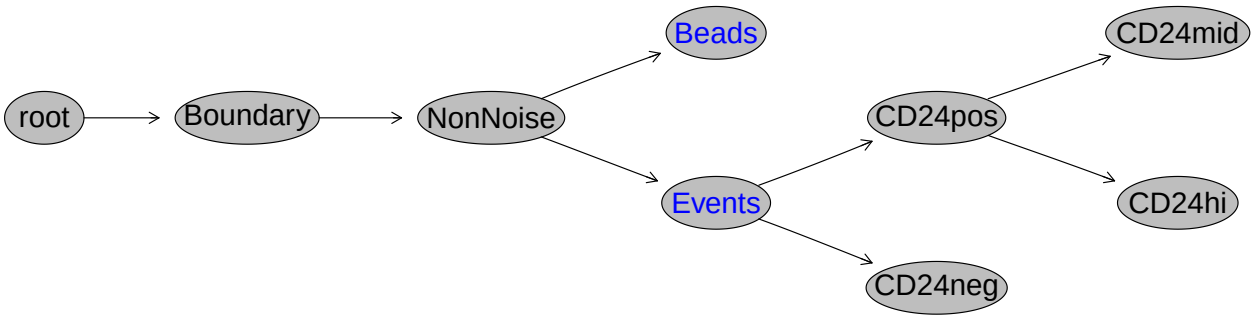
```
flowWorkspace::gs_pop_add(
  gs=gatingset,
  gate=flowWorkspace::booleanFilter(`BeadsA & BeadsB`),
  name="Beads",
  parent="NonNoise"
)

flowWorkspace::gs_pop_add(
  gs=gatingset,
  gate=flowWorkspace::booleanFilter(`NonNoise & !Beads`),
  name="Events",
  parent="NonNoise"
)

flowWorkspace::gs_pop_add(
  gs=gatingset,
  gate=getGate("CD24pos"),
  parent=getGateParent("CD24pos")
)

flowWorkspace::gs_pop_add(
  gs=gatingset,
  gate=getGate("CD24neg"),
  parent=getGateParent("CD24neg")
)

flowWorkspace::gs_pop_add(
  gs=gatingset,
  gate=getGate("CD24mid"),
  parent=getGateParent("CD24mid")
)

flowWorkspace::gs_pop_add(
  gs=gatingset,
  gate=getGate("CD24hi"),
  parent=getGateParent("CD24hi")
)
```

Because individually they are completely irrelevant to our analysis, we can hide the "helper" bead count nodes from our gating tree, instead focusing on the semantically more useful `Beads` boolean gate. We can then apply the gating tree to the `GatingSet`: this will also compute population statistics down the tree.

```
invisible(flowWorkspace::gs_pop_set_visibility(gatingset, "BeadsA", FALSE))
invisible(flowWorkspace::gs_pop_set_visibility(gatingset, "BeadsB", FALSE))

# Apply the gating tree to the underlying data
flowWorkspace::recompute(gatingset)
```

We can then plot the gating tree in order to visualize the general gating strategy for this experiment.

```
plot(gatingset, bool=TRUE)
```



2.4 Data normalization

There is some purely instrumental variability: beads sometimes appear at slightly different locations, so we normalize for this here.

References:

- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3648208/

Technical documentation:

- https://bioconductor.org/packages/release/bioc/manuals/flowStats/man/flowStats.pdf

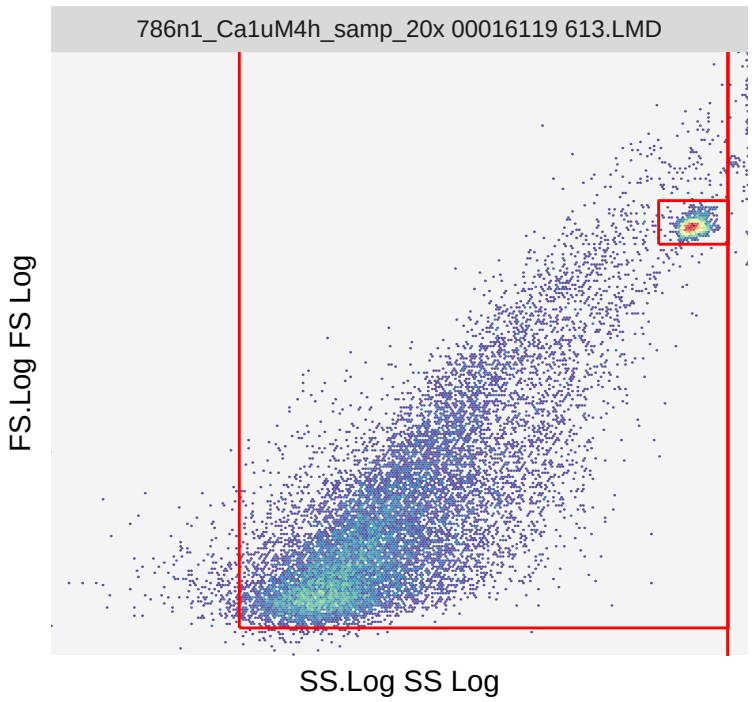
```
## Estimating landmarks for channel FS.Log ...Estimating landmarks for channel SS.Log ...
## Registering curves for parameter FS.Log ...
## Registering curves for parameter SS.Log ...
## Estimating landmarks for channel FS.Log ...Estimating landmarks for channel SS.Log ...
## Registering curves for parameter FS.Log ...
## Registering curves for parameter SS.Log ...
```

3 Data visualization

3.1 Single sample morphology plot

Gated cytometry data can be plotted with ggCyto.

```
ggcyto::ggcyto(
  data=gatingset[[1]],
  mapping=ggplot2::aes(
    x=SS.Log,
    y=FS.Log),
  subset="root"
) +
  ggplot2::geom_hex(bins=256) +
  ggplot2::scale_x_continuous(expand=c(0, 0)) +
  ggplot2::scale_y_continuous(expand=c(0, 0)) +
  ggcyto::geom_gate("Boundary") +
  ggcyto::geom_gate("NonNoise") +
  ggcyto::geom_gate("BeadsA") +
  ggplot2::theme(
    legend.position="none",
    panel.background=ggplot2::element_rect(fill="#F4F4F4"),
    panel.grid.major.x=ggplot2::element_blank(),
    panel.grid.minor.x=ggplot2::element_blank(),
    panel.grid.major.y=ggplot2::element_blank(),
    panel.grid.minor.y=ggplot2::element_blank(),
    panel.spacing=grid::unit(0.1, units="lines"),
    axis.text.x=ggplot2::element_blank(),
    axis.text.y=ggplot2::element_blank(),
    axis.ticks=ggplot2::element_blank())
```



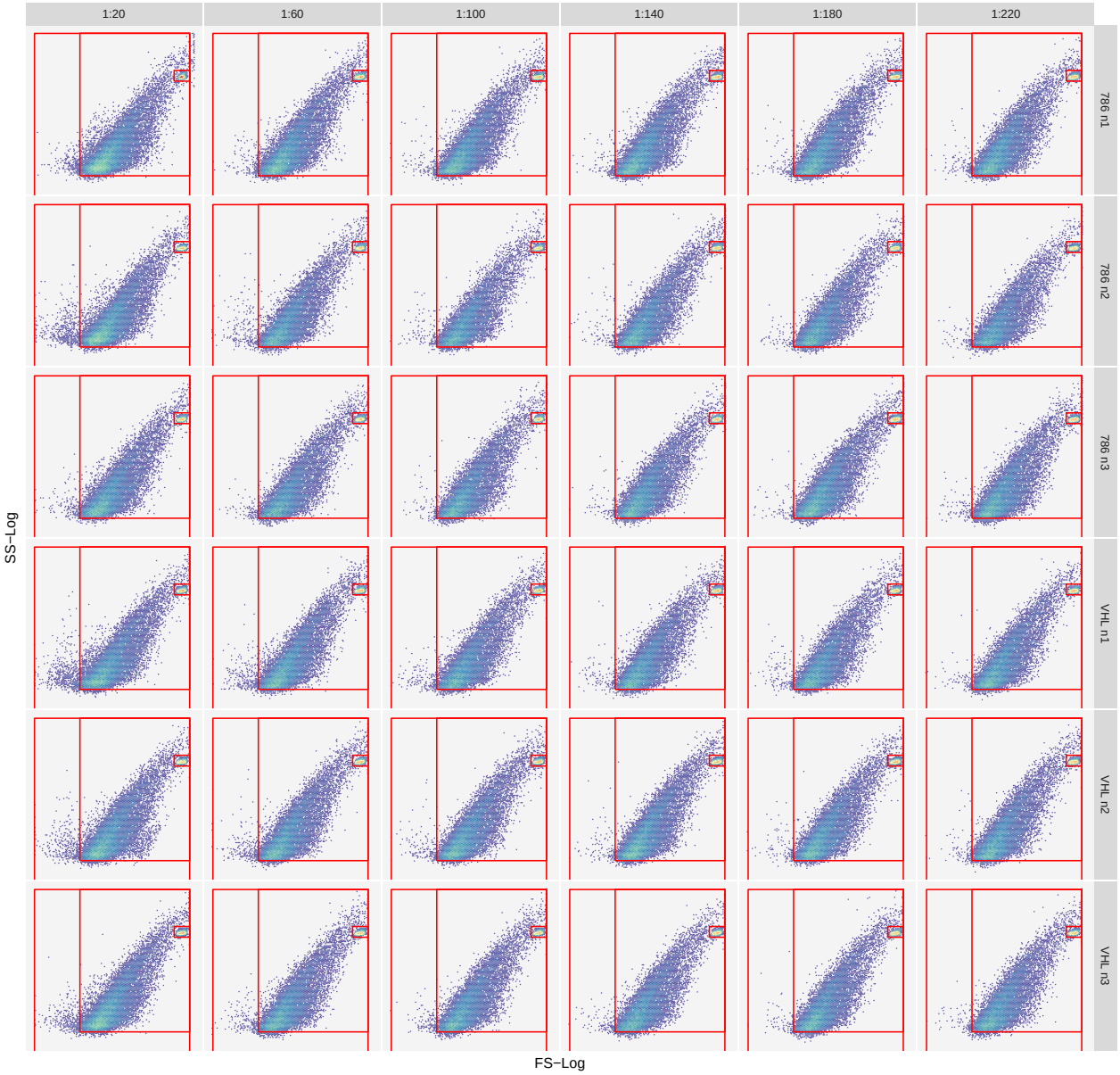
3.2 Whole experiment morphology plot

Let's plot the entire Gatingset so we can visualize the morphology plane on all samples simultaneously!

```
ggcyto::ggcyto(
  data=gatingset,
  mapping=ggplot2::aes(
    x=SS.Log,
    y=FS.Log),
  subset="root"
) +
  ggplot2::labs(
    title="Stratégie de gating sur les paramètres de morphologie",
    subtitle=paste0(
      "Élimination des évènements en bordure, élimination du bruit et",
      "identification des billes de comptage"),
    x="FS-Log",
    y="SS-Log") +
  ggplot2::geom_hex(bins=128) +
  ggcyto::geom_gate("Boundary") +
  ggcyto::geom_gate("NonNoise") +
  ggcyto::geom_gate("BeadsA") +
  ggplot2::facet_grid(
    rows=dplyr::vars(FacetRow),
    cols=dplyr::vars(FacetCol),
    labeller=ggplot2::as_labeller(facetLabels)) +
  ggplot2::theme(
    legend.position="none",
    panel.background=ggplot2::element_rect(fill="#F4F4F4"),
    panel.grid.major.x=ggplot2::element_blank(),
    panel.grid.minor.x=ggplot2::element_blank(),
    panel.grid.major.y=ggplot2::element_blank(),
    panel.grid.minor.y=ggplot2::element_blank(),
    panel.spacing=grid::unit(0.1, units="lines"),
    axis.text.x=ggplot2::element_blank(),
    axis.text.y=ggplot2::element_blank(),
    axis.ticks=ggplot2::element_blank())
```

Stratégie de gating sur les paramètres de morphologie

Élimination des évènements en bordure, élimination du bruit etidentification des billes de comptage



3.3 Auxiliary bead gate visualization

Visualizing the auxiliary bead gates:

```
gatingset <- flowStats::normalize(
  data=gatingset,
  populations=c("BeadsB"),
  dims=c("SS.Lin"),
  minCountThreshold=100
)
```

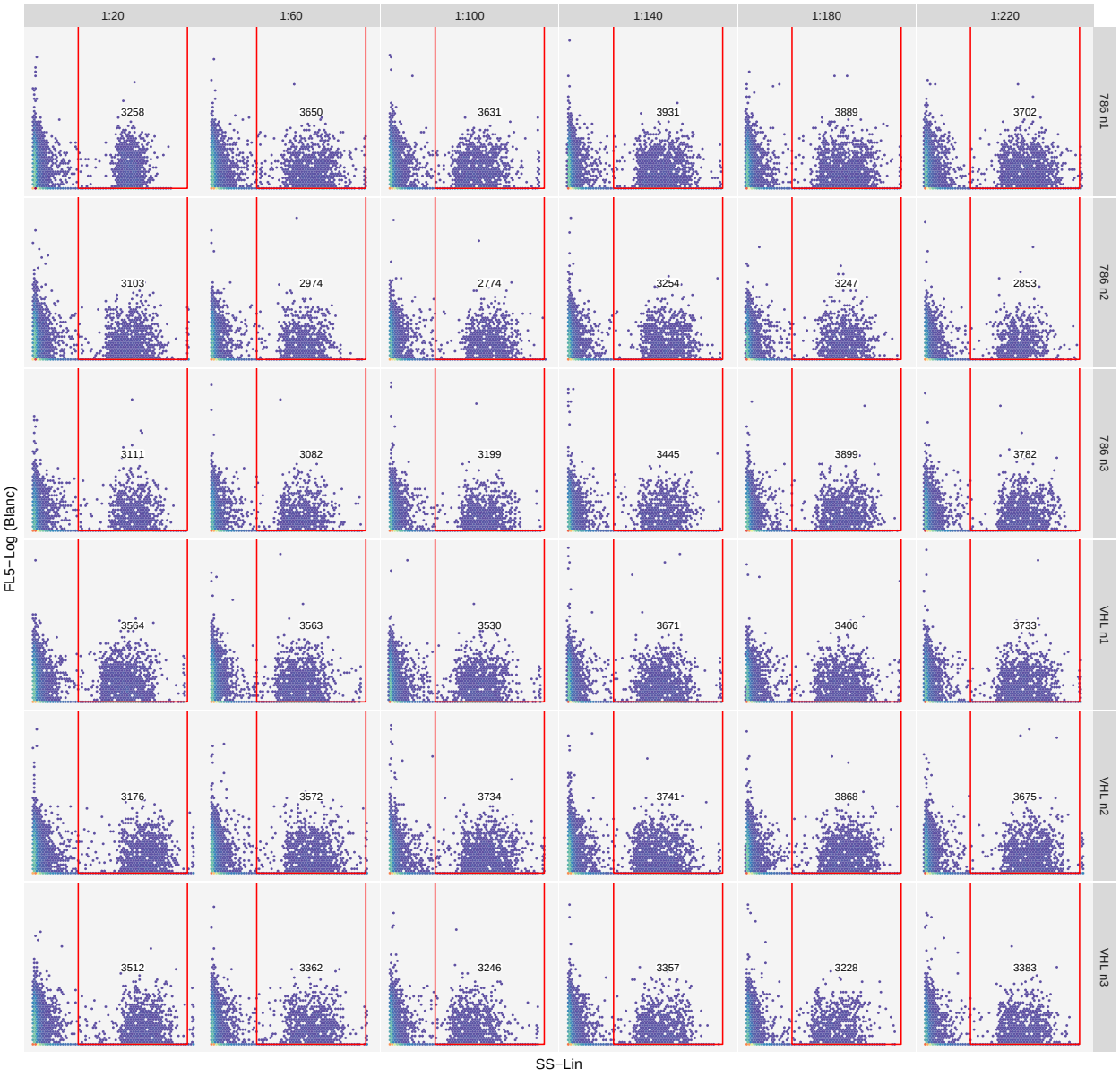
```
## cloning the gatingSet...
## Normalize BeadsB
## Estimating landmarks for channel SS.Lin ...
## Registering curves for parameter SS.Lin ...
## normalizing sample 1
## normalizing sample 2
## normalizing sample 3
## normalizing sample 4
## normalizing sample 5
## normalizing sample 6
## normalizing sample 7
## normalizing sample 8
## normalizing sample 9
## normalizing sample 11
## normalizing sample 12
## normalizing sample 13
## normalizing sample 14
## normalizing sample 15
## normalizing sample 16
## normalizing sample 17
## normalizing sample 18
## normalizing sample 19
## normalizing sample 20
## normalizing sample 21
## normalizing sample 22
## normalizing sample 23
## normalizing sample 24
## normalizing sample 25
## normalizing sample 26
## normalizing sample 27
```

```
## normalizing sample 28
## normalizing sample 29
## normalizing sample 30
## normalizing sample 31
## normalizing sample 32
## normalizing sample 33
## normalizing sample 34
## normalizing sample 35
## normalizing sample 36
## normalizing sample 37
## done!
```

```
ggcyto::ggcyto(
  data=gatingset,
  mapping=ggplot2::aes(
    x=SS.Lin,
    y=FL5.Log),
  subset="NonNoise"
) +
  ggplot2::labs(
    title="Stratégie de gating pour l'élimination des beads",
    subtitle="Gating auxiliaire des billes de comptage",
    x="SS-Lin",
    y="FL5-Log (Blanc)") +
  ggplot2::geom_hex(bins=64) +
  ggcyto::geom_gate("BeadsB") +
  ggcyto::geom_stats(
    size=2.7,
    type="count") +
  ggplot2::facet_grid(
    rows=dplyr::vars(FacetRow),
    cols=dplyr::vars(FacetCol),
    labeller=ggplot2::as_labeller(facetLabels)) +
  ggplot2::theme(
    legend.position="none",
    panel.background=ggplot2::element_rect(fill="#F4F4F4"),
    panel.grid.major.x=ggplot2::element_blank(),
    panel.grid.minor.x=ggplot2::element_blank(),
    panel.grid.major.y=ggplot2::element_blank(),
    panel.grid.minor.y=ggplot2::element_blank(),
    panel.spacing=grid::unit(0.1, units="lines"),
    axis.text.x=ggplot2::element_blank(),
    axis.text.y=ggplot2::element_blank(),
    axis.ticks=ggplot2::element_blank())
```

Stratégie de gating pour l'élimination des beads

Gating auxiliaire des billes de comptage



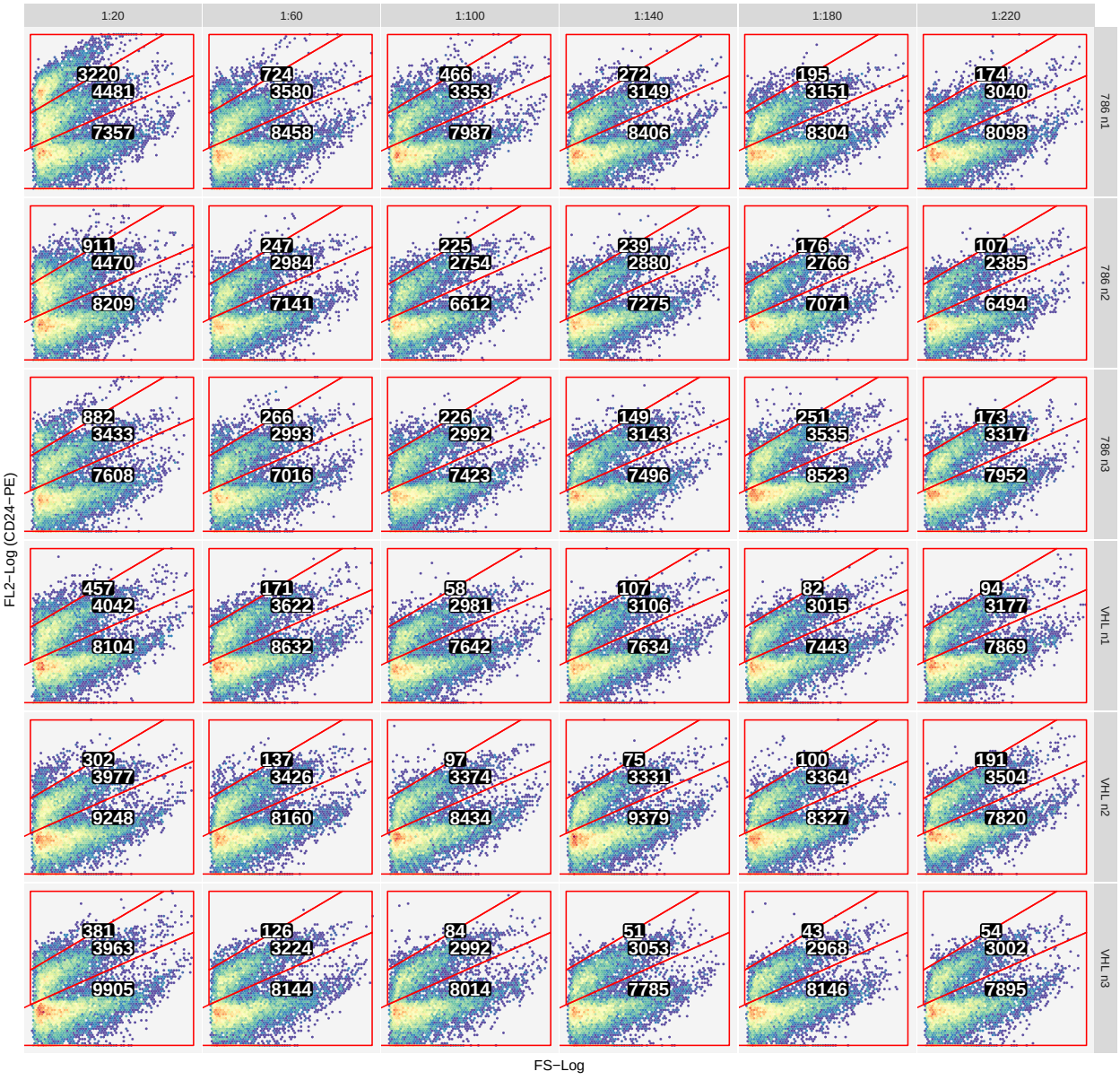
3.4 Whole experiment fluorescence with event counts

And finally, let's visualize the non-bead events on the fluorescence channel (anti-CD24 antibodies)

```
ggcyto::ggcyto(
  data=gatingset,
  mapping=ggplot2::aes(
    x=FS.Log,
    y=FL2.Log),
  subset="Events"
) +
  ggplot2::labs(
    title="Quantification des microparticules marquées au CD24-PE",
    subtitle=paste0(
      "Comparaison des 786-O (n=3) et des 786-O/VHL (n=3). ",
      "[Gated: Events]"
    ),
    x="FS-Log",
    y="FL2-Log (CD24-PE)") +
  ggplot2::geom_hex(bins=64) +
  ggcyto::geom_gate(c(
    "CD24hi",
    "CD24mid",
    "CD24neg")) +
  ggcyto::geom_stats(
    color="#FFFFFF",
    fill="#000000",
    size=5,
    fontface="bold",
    type="count") +
  ggplot2::facet_grid(
    rows=dplyr::vars(FacetRow),
    cols=dplyr::vars(FacetCol),
    labeller=ggplot2::as_labeller(facetLabels)) +
  ggplot2::theme(
    legend.position="none",
    panel.background=ggplot2::element_rect(fill="#F4F4F4"),
    panel.grid.major.x=ggplot2::element_blank(),
    panel.grid.minor.x=ggplot2::element_blank(),
    panel.grid.major.y=ggplot2::element_blank(),
    panel.grid.minor.y=ggplot2::element_blank(),
    panel.spacing=grid::unit(0.1, units="lines"),
    axis.text.x=ggplot2::element_blank(),
    axis.text.y=ggplot2::element_blank(),
    axis.ticks=ggplot2::element_blank())
```

Quantification des microparticules marquées au CD24−PE

Comparaison des 786−O (n=3) et des 786−O/VHL (n=3). [Gated: Events]



4 Analysis

4.1 Data-driven methods

4.1.1 Fingerprinting

Fingerprinting is a quick and easy way to observe population-agnostic, data-driven differences between a large number of samples.

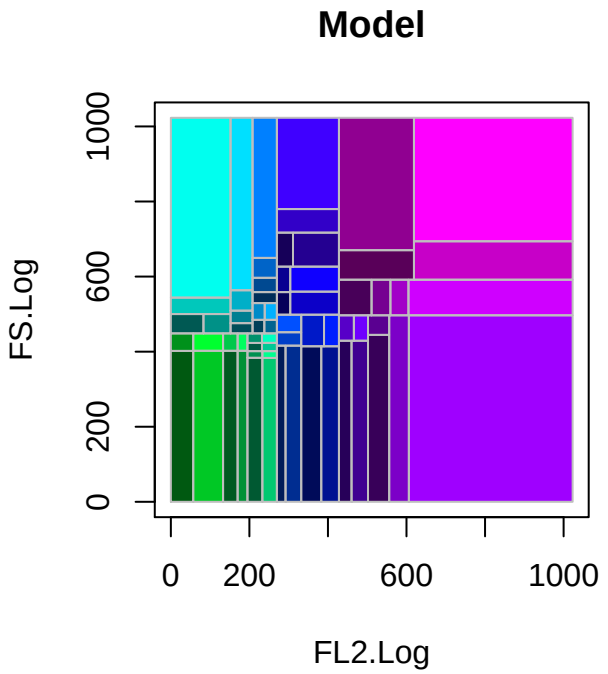
References:

- https://www.ncbi.nlm.nih.gov/pubmed/19956416

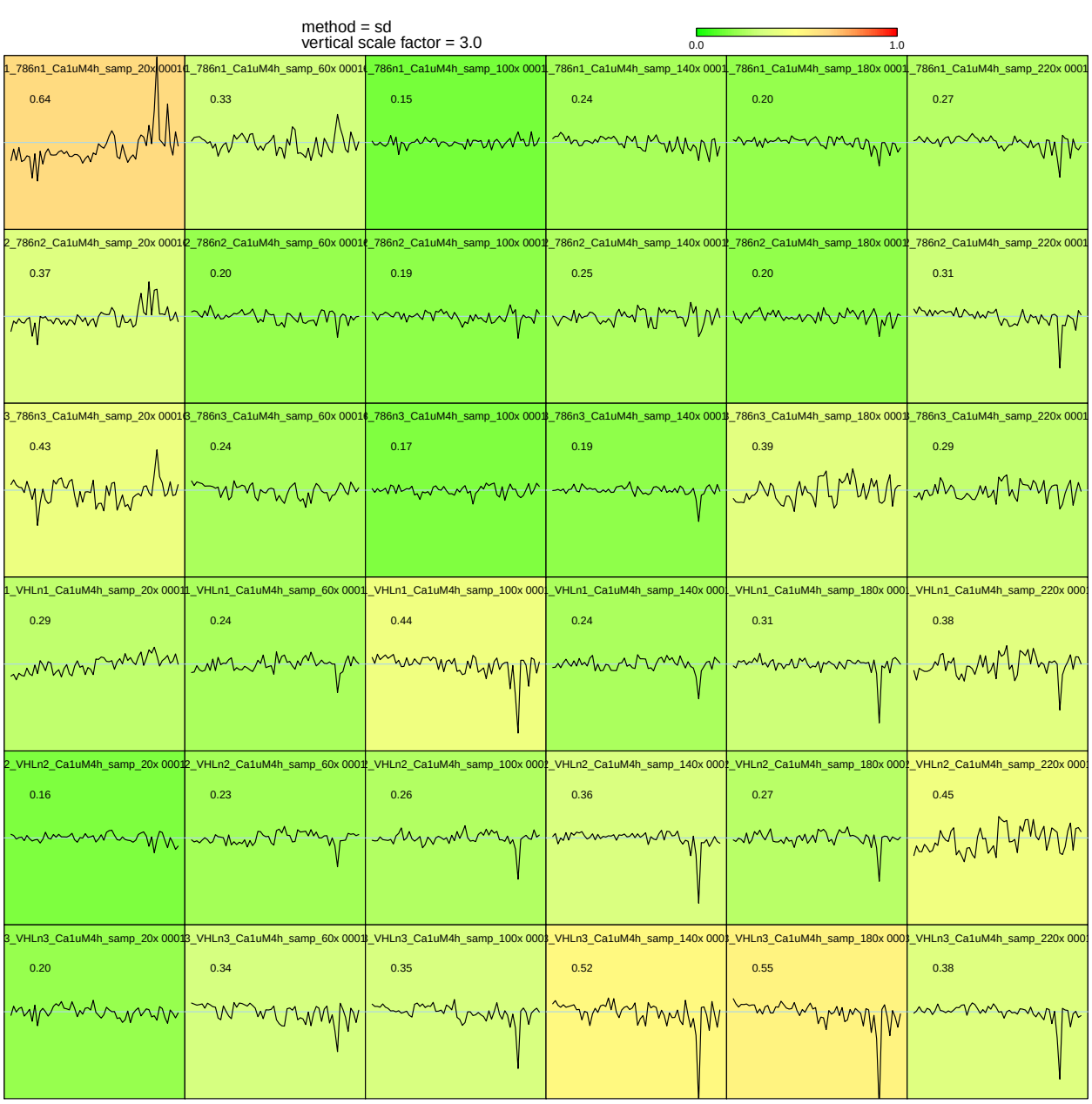
Technical documentation:

*https://bioconductor.org/packages/release/bioc/vignettes/flowFP/inst/doc/flowFP_HowTo.pdf

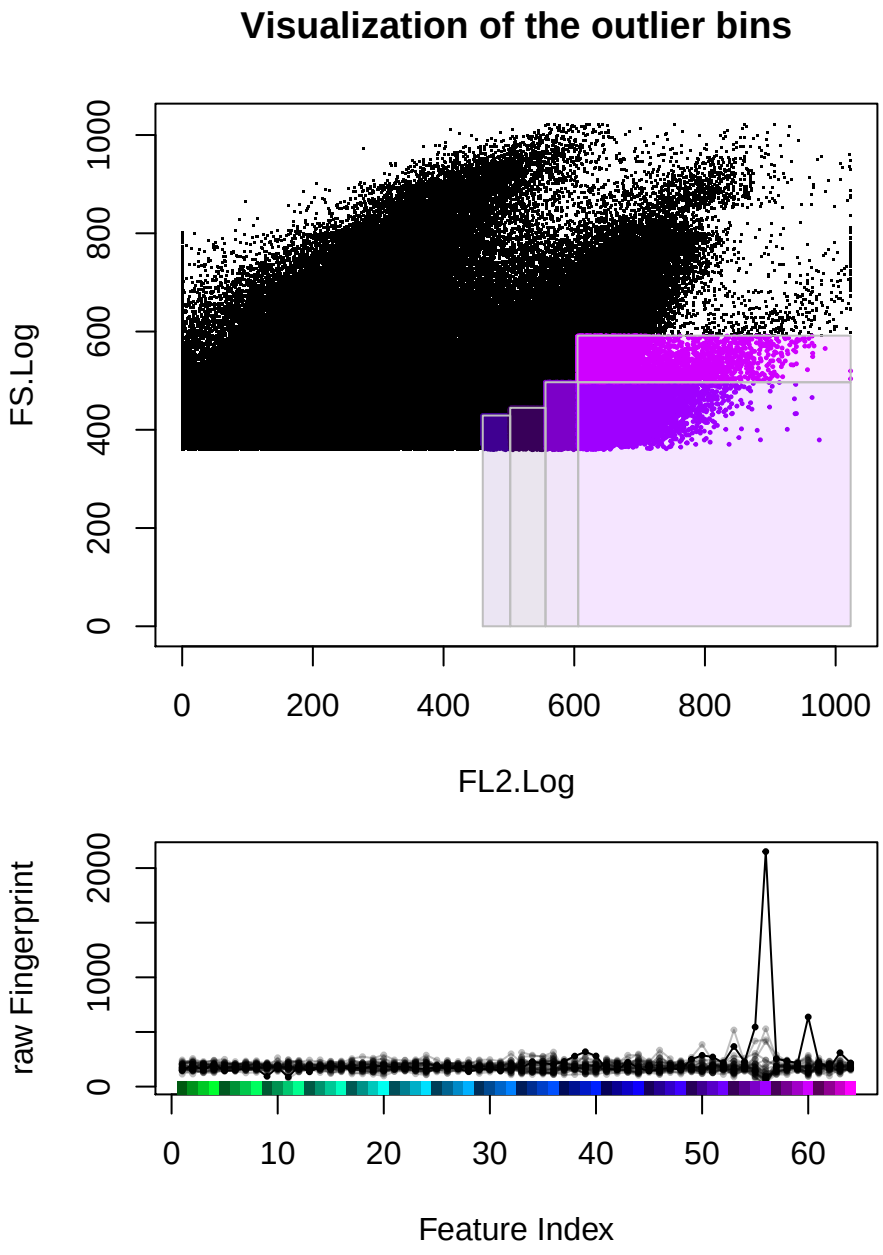
```
# Remove all irrelevant populations, such as beads and noise
fingerprintingData <- flowWorkspace::gs_pop_get_data(gatingset, "Events")
fingerprintingData <- flowWorkspace::cytoset_to_flowSet(fingerprintingData)

# Generate a fingerprinting model with the data in the "Events"-gated FlowSet
fingerprintingModel <- flowFP::flowFPModel(
  fcs=fingerprintingData,
  name="CD24/FS.Log Model",
  parameters=c("FS.Log", "FL2.Log"),
  nRecursions=6
)

# Generate fingerprints for all samples
fingerprints <- flowFP::flowFP(
  fcs=fingerprintingData,
  model=fingerprintingModel
)

# Visualize model
plot(fingerprintingModel)
```


Model
FS.Log
0
200
600
1000
0
200
600
1000
FL2.Log

```r
# Detect outliers
plot(fingerprints, type="qc", main="Outliers")
```



```r
# Visualize a specific sample and its outlier bins
plot(
  fingerprints,
  fingerprintingData,
  hi=1,
  showbins=c(50, 53, 55, 56, 60),
  pch=20,
  cex=0.3,
  main="Visualization of the outlier bins"
)
```



4.2 Swarm deconvolution

This is a novel application of a deconvolution strategy that has been validated in entirely different contexts.

Reference:

- http://tinyheero.github.io/2016/01/03/gmm-em.html

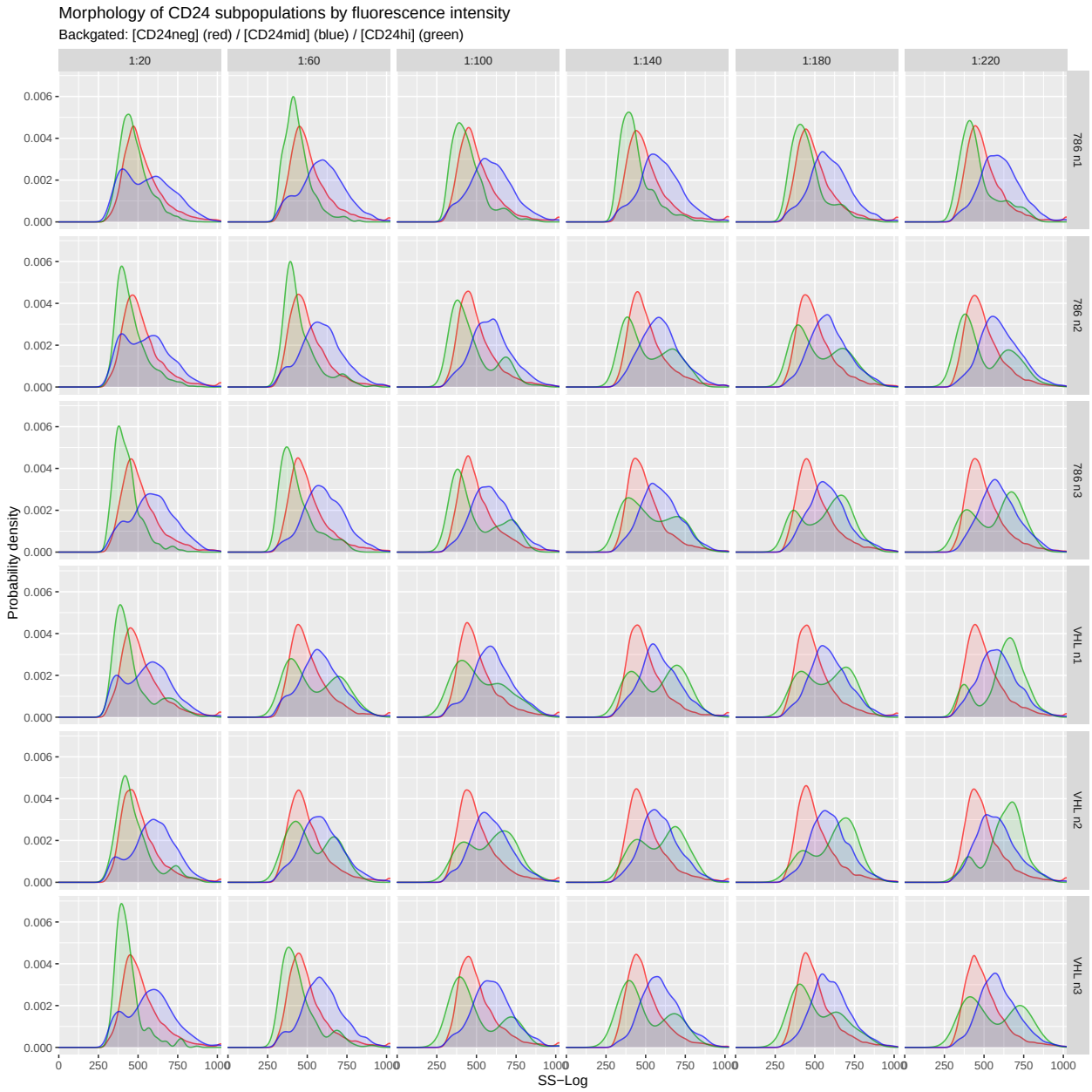
DECONVOLUTION TOOLS:

- http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0067620
- https://bioconductor.org/packages/release/bioc/vignettes/flowFit/inst/doc/HowTo-flowFit.pdf

4.2.1 Visualizing the swarm effect

We can back-gate any of our fluorescent populations into a morphology axis:

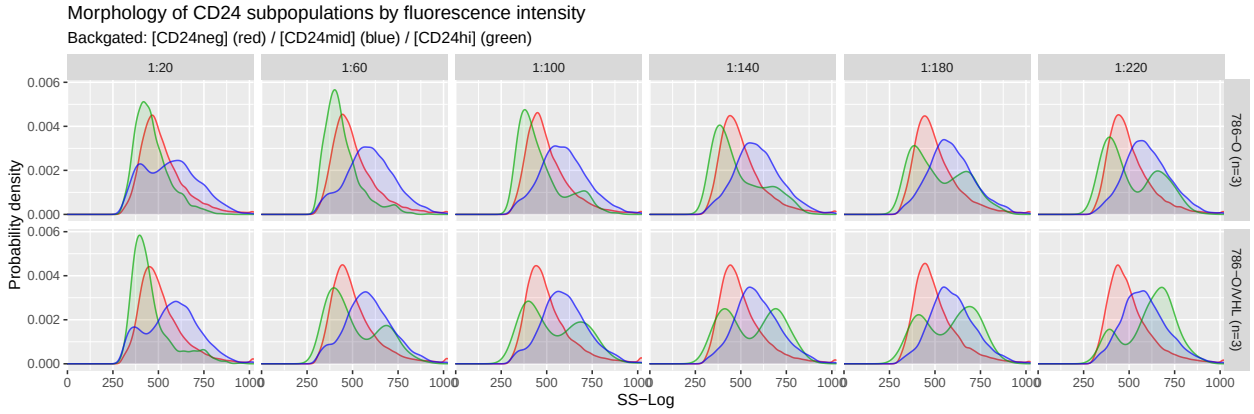
```
# Morphology characteristics of fluorescence-gated data can be visualized to
# gain quantitative insight into the magnitude of per-subpopulation swarm effect
ggplot2::ggplot(
  data=flowWorkspace::gs_pop_get_data(gatingset, "Events"),
  mapping=ggplot2::aes(x=SS.Log)
) +
  ggplot2::labs(
    title="Morphology of CD24 subpopulations by fluorescence intensity",
    subtitle="Backgated: [CD24neg] (red) / [CD24mid] (blue) / [CD24hi] (green)",
    x="SS-Log",
    y="Probability density") +
  ggplot2::scale_x_continuous(
    limits=c(0, 1023),
    expand=c(0, 0)) +
  ggplot2::geom_density(
    data=flowWorkspace::gs_pop_get_data(gatingset, "CD24neg"),
    color="#FF0000AA",
    fill="red",
    alpha=0.1) +
  ggplot2::geom_density(
    data=flowWorkspace::gs_pop_get_data(gatingset, "CD24hi"),
    color="#00AA00AA",
    fill="#00AA00",
    alpha=0.1) +
  ggplot2::geom_density(
    data=flowWorkspace::gs_pop_get_data(gatingset, "CD24mid"),
    color="#0000FFAA",
    fill="blue",
    alpha=0.1) +
  ggplot2::facet_grid(
    rows=dplyr::vars(FacetRow),
    cols=dplyr::vars(FacetCol),
    labeller=ggplot2::as_labeller(facetLabels))
```




```
cellLineOrdering <- c("786-O", "VHL")
dilutionOrdering <- c("20", "60", "100", "140", "180", "220")

eventsData <- flowWorkspace::gs_pop_get_data(gatingset, "Events")
eventsData <- flowWorkspace::cytoset_to_flowSet(eventsData)
cd24negData <- flowWorkspace::gs_pop_get_data(gatingset, "CD24neg")
cd24negData <- flowWorkspace::cytoset_to_flowSet(cd24negData)
cd24midData <- flowWorkspace::gs_pop_get_data(gatingset, "CD24mid")
cd24midData <- flowWorkspace::cytoset_to_flowSet(cd24midData)
cd24hiData <- flowWorkspace::gs_pop_get_data(gatingset, "CD24hi")
cd24hiData <- flowWorkspace::cytoset_to_flowSet(cd24hiData)

eventsData@phenoData@data$Dilution <- factor(
  eventsData@phenoData@data$Dilution,
```

```
  levels=dilutionOrdering)
cd24negData@phenoData@data$Dilution <- factor(
  cd24negData@phenoData@data$Dilution,
  levels=dilutionOrdering)
cd24midData@phenoData@data$Dilution <- factor(
  cd24midData@phenoData@data$Dilution,
  levels=dilutionOrdering)
cd24hiData@phenoData@data$Dilution <- factor(
  cd24hiData@phenoData@data$Dilution,
  levels=dilutionOrdering)

eventsData@phenoData@data$CellLine <- factor(
  eventsData@phenoData@data$CellLine,
  levels=cellLineOrdering)
cd24negData@phenoData@data$CellLine <- factor(
  cd24negData@phenoData@data$CellLine,
  levels=cellLineOrdering)
cd24midData@phenoData@data$CellLine <- factor(
  cd24midData@phenoData@data$CellLine,
  levels=cellLineOrdering)
cd24hiData@phenoData@data$CellLine <- factor(
  cd24hiData@phenoData@data$CellLine,
  levels=cellLineOrdering)

ggplot2::ggplot(
  data=eventsData,
  mapping=ggplot2::aes(x=SS.Log)
) +
  ggplot2::labs(
    title="Morphology of CD24 subpopulations by fluorescence intensity",
    subtitle="Backgated: [CD24neg] (red) / [CD24mid] (blue) / [CD24hi] (green)",
    x="SS-Log",
    y="Probability density") +
  ggplot2::scale_x_continuous(
    limits=c(0, 1023),
    expand=c(0, 0)) +
  ggplot2::geom_density(
    data=cd24negData,
    color="#FF0000AA",
    fill="red",
    alpha=0.1) +
  ggplot2::geom_density(
    data=cd24hiData,
    color="#00AA00AA",
    fill="#00AA00", # Dark green
    alpha=0.1) +
  ggplot2::geom_density(
    data=cd24midData,
    color="#0000FFAA",
    fill="blue",
    alpha=0.1) +
  ggplot2::facet_grid(
    rows=dplyr::vars(CellLine),
```

```
    cols=dplyr::vars(Dilution),
    labeller=ggplot2::as_labeller(c(
      "20"    = "1:20",
      "60"    = "1:60",
      "100"   = "1:100",
      "140"   = "1:140",
      "180"   = "1:180",
      "220"   = "1:220",
      "786-O" = "786-O (n=3)",
      "VHL"   = "786-O/VHL (n=3)"
    )))
```



4.2.2 A step-by-step example: CD24mid swarm deconvolution

4.2.2.1 Mixed distribution modeling Let's compute the ML estimate of the mixed distribution using an expectation-maximization algorithm provided by mixtools. This will serve as an initial estimation of subpopulation characteristics before we proceed with model optimization.

Technical documentation:

- https://cran.r-project.org/web/packages/mixtools/mixtools.pdf
- https://cran.r-project.org/web/packages/mixtools/vignettes/mixtools.pdf

```
estimate <- list(
  "mu"    = c(373, 650),
  "sigma" = c(35.6, 120)
)

generateSwarmModel <- function(
  gs,
  axis="SS.Log",
  gate="CD24pos",
  mu=c(0, 0),
  sigma=c(1, 1),
  maxIterations=10000
) {
  # Linter bindings
  pNeg <-
  pPos <-
  FacetRow <-
  FacetCol <- NULL

  numberOfFlowframes <- length(flowWorkspace::sampleNames(gs))
  swarmModel <- list()

  for (i in seq_len(numberOfFlowframes)) {
    y <- flowWorkspace::gs_pop_get_data(gs, gate)
    y <- flowWorkspace::cytoset_to_flowSet(y)
    y <- y[[i]]@exprs[, axis]

    mixEM <- mixtools::normalmixEM(y, mu=mu, sigma=sigma, maxit=maxIterations)

    # Extract important results obtained from normalmixEM()
    distribution1 <- c(
      "mu"         = mixEM$mu[1],
      "sigma"      = mixEM$sigma[1],
      "proportion" = mixEM$lambda[1]
    )

    distribution2 <- c(
      "mu"         = mixEM$mu[2],
      "sigma"      = mixEM$sigma[2],
      "proportion" = mixEM$lambda[2]
    )

    # Ensure we correctly identify the true positive distribution
    if (distribution1[["mu"]] < distribution2[["mu"]]) {
      negDistribution <- distribution1
```

```
      posDistribution <- distribution2
    } else {
      negDistribution <- distribution2
      posDistribution <- distribution1
    }

    hartigansDipTestStatistic <-
      diptest::dip.test(y, B=2000, simulate.p.value=FALSE)$p.value

    # Assign values to the appropriate columns
    swarmModel[[i]] <- c(
      "id"        = i,
      "muNeg"     = negDistribution[["mu"]],
      "sigmaNeg"  = negDistribution[["sigma"]],
      "pNeg"      = negDistribution[["proportion"]],
      "muPos"     = posDistribution[["mu"]],
      "sigmaPos"  = posDistribution[["sigma"]],
      "pPos"      = posDistribution[["proportion"]],
      "hartigans" = hartigansDipTestStatistic
    )
  }

  # Create the dataframe from all generated values
  swarmModel <- BiocGenerics::as.data.frame(
    BiocGenerics::do.call("rbind", swarmModel)
  )

  # Compute number of events in each distribution
  swarmModel$nNeg <- round(length(y) * swarmModel$pNeg, digits=0)
  swarmModel$nPos <- round(length(y) * swarmModel$pPos, digits=0)

  # Population statistics
  swarmModel$welchTTest <-
    abs(swarmModel$muNeg - swarmModel$muPos) /
      sqrt(
        ((swarmModel$sigmaNeg^4)/swarmModel$nNeg) +
        ((swarmModel$sigmaPos^4)/swarmModel$nPos))

  # Assign faceting values for plotting convenience
  swarmModel$facetRow <- pData$FacetRow
  swarmModel$facetCol <- pData$FacetCol

  swarmModel
}

# TODO: mu and sigma may need to be tweaked recursively on a per-experiment
# basis
cd24midSwarmModel <- generateSwarmModel(
  gs=gatingset,
  axis="SS.Log",
  gate="CD24mid",
  mu=estimate$mu,
  sigma=estimate$sigma,
```

```
  maxIterations=10000
)
```

```
## number of iterations= 60
## number of iterations= 53
## number of iterations= 125
## number of iterations= 78
## number of iterations= 147
## number of iterations= 1916
## number of iterations= 67
## number of iterations= 51
## number of iterations= 2227
## number of iterations= 1907
## number of iterations= 1418
## number of iterations= 1641
## number of iterations= 57
## number of iterations= 85
## number of iterations= 1324
## number of iterations= 1107
## number of iterations= 1587
## number of iterations= 2906
## number of iterations= 52
## number of iterations= 68
## number of iterations= 54
## number of iterations= 1490
## number of iterations= 1090
## number of iterations= 1152
## number of iterations= 52
## number of iterations= 1736
## number of iterations= 2006
## number of iterations= 1821
## number of iterations= 3721
## number of iterations= 778
## number of iterations= 44
## number of iterations= 55
## number of iterations= 86
## number of iterations= 2283
## number of iterations= 3879
## number of iterations= 1423
```

Let's check out how our model data looks so far.

```
printDataFrame(
  dataframe=cd24midSwarmModel,
  caption=
    "Swarm deconvolution model: initial ML estimate on a per-sample basis",
  fontsize=6
)
```

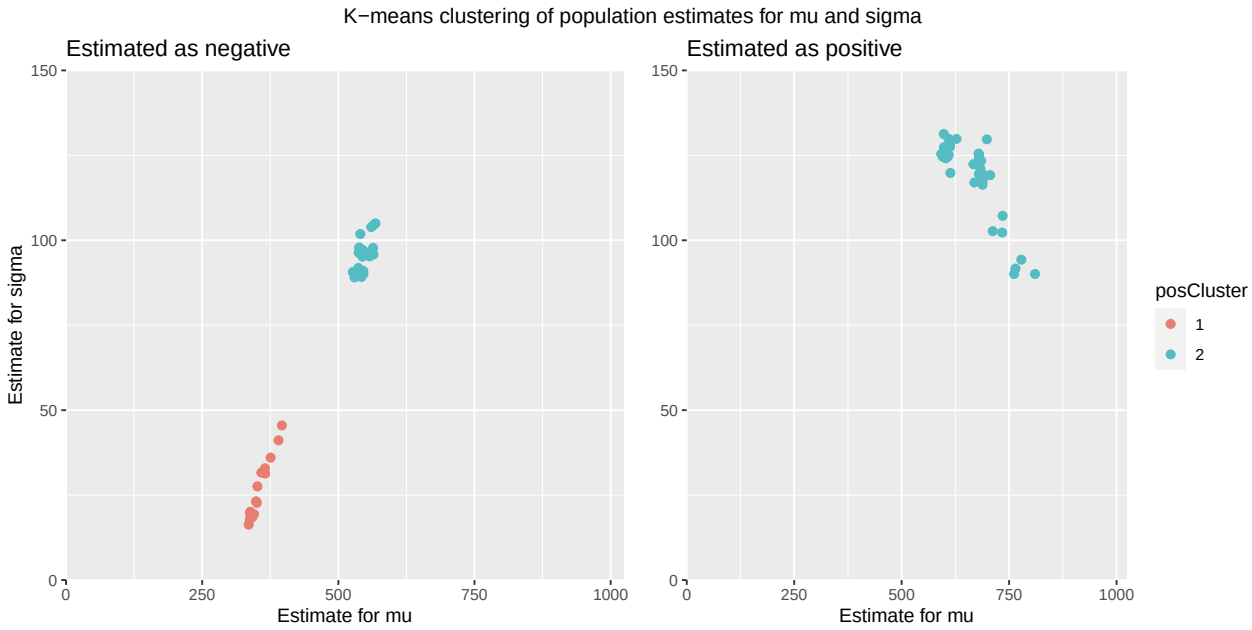
Table 3: Swarm deconvolution model: initial ML estimate on a per-sample basis

id	muNeg	sigmaNeg	pNeg	muPos	sigmaPos	pPos	hartigans	nNeg	nPos	welchTTest	facetRow	facetCol
1	396.2477	45.51179	0.2801629	627.5732	129.83076	0.7198371	0.0015918	841	2161	0.6259356	r1	c1
2	365.6779	31.34523	0.0687591	612.1960	127.49640	0.9312409	0.8445203	206	2796	0.7827308	r1	c2
3	342.1468	18.49875	0.0232760	597.9968	131.28649	0.9767240	0.9904514	70	2932	0.7972076	r1	c3
4	338.2334	20.10951	0.0193678	597.3776	125.52543	0.9806322	0.8242218	58	2944	0.8778202	r1	c4
5	335.3437	16.31792	0.0112025	595.6916	124.57192	0.9887975	0.9456182	34	2968	0.9024751	r1	c5
6	568.2024	104.97603	0.7500644	698.1721	129.71595	0.2499356	0.9158683	2252	750	0.1978752	r1	c6
7	390.0944	41.10509	0.2334694	600.1357	124.72709	0.7665306	0.0000535	701	2301	0.6354659	r2	c1
8	350.2875	22.73414	0.0343862	598.2290	126.10599	0.9656138	0.9643520	103	2899	0.8272571	r2	c2
9	565.0091	104.57181	0.8674236	778.5772	94.28697	0.1325764	0.8426692	2604	398	0.4319169	r2	c3
10	549.8177	96.44425	0.7158907	705.8512	119.17178	0.2841093	0.7700217	2149	853	0.2966222	r2	c4
11	557.3301	95.26453	0.8215360	761.5212	90.06698	0.1784640	0.9806157	2466	536	0.5166990	r2	c5
12	542.4338	89.21970	0.5940336	688.6061	117.54744	0.4059664	0.9795325	1783	1219	0.3334535	r2	c6
13	375.7032	36.03064	0.0919899	612.7808	127.85776	0.9080101	0.8964370	276	2726	0.7346465	r3	c1
14	337.9095	19.61390	0.0118429	609.5814	125.18705	0.9881571	0.8381467	36	2966	0.9214887	r3	c2
15	536.5526	91.84112	0.5725889	679.5750	119.51510	0.4274111	0.9261474	1719	1283	0.3194784	r3	c3
16	529.8057	89.06979	0.5405686	669.2863	117.00383	0.4594314	0.9138822	1623	1379	0.3337231	r3	c4
17	560.4540	103.86702	0.9054335	810.3193	90.08118	0.0945665	0.8852133	2718	284	0.4767543	r3	c5
18	546.1494	90.07903	0.6949469	711.8546	102.68031	0.3050531	0.9078441	2086	916	0.4237487	r3	c6
19	358.6729	31.62526	0.1609952	598.3625	127.39874	0.8390048	0.0018273	483	2519	0.7339639	r4	c1
20	348.7769	23.16879	0.0364806	592.1534	125.38916	0.9635194	0.8697411	110	2892	0.8199796	r4	c2
21	337.4722	17.93602	0.0182036	607.5958	124.60818	0.9817964	0.9922402	55	2947	0.9337305	r4	c3
22	538.3543	97.87543	0.6483710	679.7746	124.45695	0.3516290	0.9746790	1946	1056	0.2699923	r4	c4
23	544.3212	95.18792	0.6443354	679.9557	125.18040	0.3556646	0.9901663	1934	1068	0.2598915	r4	c5
24	537.2170	96.36119	0.6111640	667.0201	122.39056	0.3888360	0.6375427	1835	1167	0.2653685	r4	c6
25	351.4054	27.56467	0.0835581	613.4682	119.82474	0.9164419	0.6567281	251	2751	0.9429594	r5	c1
26	540.4398	101.83783	0.6484748	679.0153	125.51385	0.3515252	0.9657327	1947	1055	0.2571137	r5	c2
27	544.9635	97.14286	0.6599103	683.0622	121.15666	0.3400897	0.9899165	1981	1021	0.2729455	r5	c3
28	545.7691	90.98038	0.6740031	688.1655	116.35255	0.3259969	0.9173520	2023	979	0.3028510	r5	c4
29	561.3446	96.23725	0.7975746	733.7756	102.28001	0.2024254	0.9397857	2394	608	0.3711630	r5	c5
30	526.6744	90.65752	0.6652867	684.9448	123.38835	0.3347133	0.7712460	1997	1005	0.3077638	r5	c6
31	365.2537	32.90396	0.1302943	608.9051	129.88875	0.8697057	0.0497850	391	2611	0.7280116	r6	c1
32	341.3810	19.05063	0.0242346	602.5765	124.13978	0.9757654	0.9569939	73	2929	0.9072445	r6	c2
33	344.9374	19.41129	0.0135502	603.9171	124.30391	0.9864498	0.9764470	41	2961	0.8930675	r6	c3
34	563.7529	97.75941	0.8380877	764.7883	91.68066	0.1619123	0.7718399	2516	486	0.4716603	r6	c4
35	564.4278	95.80230	0.7905234	734.9316	107.18617	0.2094766	0.7572704	2373	629	0.3442267	r6	c5
36	544.5915	90.05085	0.6605705	688.8162	119.42659	0.3394295	0.9710040	1983	1019	0.2989199	r6	c6

Let's visualize this.

```
clusterModel <- function(
  model,
  centers=matrix(c(0, 0, 0, 0), ncol=2)
) {
  numberOfClusters <- BiocGenerics::nrow(centers)

  model$negCluster <- factor(
    safeClustering(
      data=model[, c("muNeg", "sigmaNeg")],
      centers=centers,
      principal=1
    ),
    levels=seq_len(numberOfClusters)
  )

  model$posCluster <- factor(
    safeClustering(
      data=model[, c("muPos", "sigmaPos")],
      centers=centers,
      principal=2
    ),
    levels=seq_len(numberOfClusters)
  )

  model
}

plotModelClusters <- function(model, ylim=c(0, 1023), xlim=c(0, 1023)) {
  plot1 <- ggplot2::ggplot(
    data=model,
    mapping=ggplot2::aes(
      x=muNeg,
      y=sigmaNeg,
      color=negCluster
    )
  ) +
    ggplot2::labs(
      title="Estimated as negative",
      x="Estimate for mu",
      y="Estimate for sigma") +
    ggplot2::coord_cartesian(
      ylim=ylim,
      xlim=xlim) +
    ggplot2::scale_x_continuous(expand=c(0, 0)) +
    ggplot2::scale_y_continuous(expand=c(0, 0)) +
    ggplot2::theme(legend.position="none") +
    ggplot2::geom_point(size=2) +
    ggplot2::scale_color_manual(
      values=c(
        "1" = colorPalette[["red"]],
        "2" = colorPalette[["blue"]]
      ),
```

```
      drop=FALSE)

  plot2 <- ggplot2::ggplot(
    data=model,
    mapping=ggplot2::aes(
      x=muPos,
      y=sigmaPos,
      color=posCluster
    )
  ) +
    ggplot2::labs(
      title="Estimated as positive",
      x="Estimate for mu",
      y=ggplot2::element_blank()) +
    ggplot2::coord_cartesian(
      ylim=ylim,
      xlim=xlim) +
    ggplot2::scale_x_continuous(expand=c(0, 0)) +
    ggplot2::scale_y_continuous(expand=c(0, 0)) +
    ggplot2::geom_point(size=2) +
    ggplot2::scale_color_manual(
      values=c(
        "1" = colorPalette[["red"]],
        "2" = colorPalette[["blue"]]),
      drop=FALSE)

  gridExtra::grid.arrange(
    plot1, plot2,
    layout_matrix=cbind(c(1), c(2)),
    top="K-means clustering of population estimates for mu and sigma"
  )
}

cd24midSwarmModel <- clusterModel(
  model=cd24midSwarmModel,
  centers=matrix(c(estimate$mu, estimate$sigma), ncol=2)
)

plotModelClusters(
  model=cd24midSwarmModel,
  ylim=c(0, 150),
  xlim=c(0, 1023)
)
```




```
# Get an idea of the distribution of the calculated averages
# Good results would be unimodal for each single parameter

# Bimodal: looks like ~375 would be a better starting parameter for negative pop
# Bimodal: based on values of mu, ~30 would be a good estimator of the negative
#          pop stddev
# Unimodal: looks like 650 is a good starting parameter for positive pop
# Unimodal: looks like 120 is a good starting parameter for positive pop stddev
```

4.2.2.2 Parameter optimization

Now that we've generated a mixed model for every sample, we can compute best-fit parameters for the whole experiment globally.

```
estimateGlobalParameters <- function(model) {
  # Calculate means weighted in favour of multimodality:
  #   result hovers around 373
  # Calculate stddev weighted in favour of multimodality:
  #   result hovers around 36
  negStats <- model %>%
    dplyr::filter(negCluster == 1, posCluster == 2) %>%
      dplyr::summarise(
        mu    = stats::weighted.mean(muNeg,    w=1-hartigans),
        sigma = stats::weighted.mean(sigmaNeg, w=1-hartigans)
      )

  posStats <- model %>%
    dplyr::filter(negCluster == 1, posCluster == 2) %>%
      dplyr::summarise(
        mu    = stats::weighted.mean(muPos,    w=1-hartigans),
        sigma = stats::weighted.mean(sigmaPos, w=1-hartigans)
      )

  params <- rbind("neg" = negStats, "pos" = posStats)
  params
}

estimatedParameters <- estimateGlobalParameters(model=cd24midSwarmModel)

printDataFrame(
  dataframe=estimatedParameters,
  caption="Estimated parameters"
)
```

Table 4: Estimated parameters

	mu	sigma
neg	371.0690	34.76509
pos	608.1583	127.05950

```
applyGlobalParameters <- function(model, params) {
  # Update our model with our best global parameter estimates
  model$muNeg    <- params["neg", "mu"]
  model$sigmaNeg <- params["neg", "sigma"]
  model$muPos    <- params["pos", "mu"]
  model$sigmaPos <- params["pos", "sigma"]
  model
}

cd24midSwarmModel <- applyGlobalParameters(
  model=cd24midSwarmModel,
  params=estimatedParameters
)

printDataFrame(
```

```
  dataframe=cd24midSwarmModel,
  caption="Swarm deconvolution model: best-fit mu and sigma global parameters",
  fontsize=5
)
```

Table 5: Swarm deconvolution model: best-fit mu and sigma global parameters

id	muNeg	sigmaNeg	pNeg	muPos	sigmaPos	pPos	hartigans	nNeg	nPos	welchTTest	facetRow	facetCol	negCluster	posCluster
1	371.069	34.76509	0.2801629	608.1583	127.0595	0.7198371	0.0015918	841	2161	0.6259356	r1	c1	1	2
2	371.069	34.76509	0.0687591	608.1583	127.0595	0.9312409	0.8445203	206	2796	0.7827308	r1	c2	1	2
3	371.069	34.76509	0.0232760	608.1583	127.0595	0.9767240	0.9904514	70	2932	0.7972076	r1	c3	1	2
4	371.069	34.76509	0.0193678	608.1583	127.0595	0.9806322	0.8242218	58	2944	0.8778202	r1	c4	1	2
5	371.069	34.76509	0.0112025	608.1583	127.0595	0.9887975	0.9456182	34	2968	0.9024751	r1	c5	1	2
6	371.069	34.76509	0.7500644	608.1583	127.0595	0.2499356	0.9158683	2252	750	0.1978752	r1	c6	2	2
7	371.069	34.76509	0.2334694	608.1583	127.0595	0.7665306	0.0000535	701	2301	0.6354659	r2	c1	1	2
8	371.069	34.76509	0.0343862	608.1583	127.0595	0.9656138	0.9643520	103	2899	0.8272571	r2	c2	1	2
9	371.069	34.76509	0.8674236	608.1583	127.0595	0.1325764	0.8426692	2604	398	0.4319169	r2	c3	2	2
10	371.069	34.76509	0.7158907	608.1583	127.0595	0.2841093	0.7700217	2149	853	0.2966222	r2	c4	2	2
11	371.069	34.76509	0.8215360	608.1583	127.0595	0.1784640	0.9806157	2466	536	0.5166990	r2	c5	2	2
12	371.069	34.76509	0.5940336	608.1583	127.0595	0.4059664	0.9795325	1783	1219	0.3334535	r2	c6	2	2
13	371.069	34.76509	0.0919899	608.1583	127.0595	0.9080101	0.8964370	276	2726	0.7346465	r3	c1	1	2
14	371.069	34.76509	0.0118429	608.1583	127.0595	0.9881571	0.8381467	36	2966	0.9214887	r3	c2	1	2
15	371.069	34.76509	0.5725889	608.1583	127.0595	0.4274111	0.9261474	1719	1283	0.3194784	r3	c3	2	2
16	371.069	34.76509	0.5405686	608.1583	127.0595	0.4594314	0.9138822	1623	1379	0.3337231	r3	c4	2	2
17	371.069	34.76509	0.9054335	608.1583	127.0595	0.0945665	0.8852133	2718	284	0.4767543	r3	c5	2	2
18	371.069	34.76509	0.6949469	608.1583	127.0595	0.3050531	0.9078441	2086	916	0.4237487	r3	c6	2	2
19	371.069	34.76509	0.1609952	608.1583	127.0595	0.8390048	0.0018273	483	2519	0.7339639	r4	c1	1	2
20	371.069	34.76509	0.0364806	608.1583	127.0595	0.9635194	0.8697411	110	2892	0.8199796	r4	c2	1	2
21	371.069	34.76509	0.0182036	608.1583	127.0595	0.9817964	0.9922402	55	2947	0.9337305	r4	c3	1	2
22	371.069	34.76509	0.6483710	608.1583	127.0595	0.3516290	0.9746790	1946	1056	0.2699923	r4	c4	2	2
23	371.069	34.76509	0.6443354	608.1583	127.0595	0.3556646	0.9901663	1934	1068	0.2598915	r4	c5	2	2
24	371.069	34.76509	0.6111640	608.1583	127.0595	0.3888360	0.6375427	1835	1167	0.2653685	r4	c6	2	2
25	371.069	34.76509	0.0835581	608.1583	127.0595	0.9164419	0.6567281	251	2751	0.9429594	r5	c1	1	2
26	371.069	34.76509	0.6484748	608.1583	127.0595	0.3515252	0.9657327	1947	1055	0.2571137	r5	c2	2	2
27	371.069	34.76509	0.6599103	608.1583	127.0595	0.3400897	0.9899165	1981	1021	0.2729455	r5	c3	2	2
28	371.069	34.76509	0.6740031	608.1583	127.0595	0.3259969	0.9173520	2023	979	0.3028510	r5	c4	2	2
29	371.069	34.76509	0.7975746	608.1583	127.0595	0.2024254	0.9397857	2394	608	0.3711630	r5	c5	2	2
30	371.069	34.76509	0.6652867	608.1583	127.0595	0.3347133	0.7712460	1997	1005	0.3077638	r5	c6	2	2
31	371.069	34.76509	0.1302943	608.1583	127.0595	0.8697057	0.0497850	391	2611	0.7280116	r6	c1	1	2
32	371.069	34.76509	0.0242346	608.1583	127.0595	0.9757654	0.9569939	73	2929	0.9072445	r6	c2	1	2
33	371.069	34.76509	0.0135502	608.1583	127.0595	0.9864498	0.9764470	41	2961	0.8930675	r6	c3	1	2
34	371.069	34.76509	0.8380877	608.1583	127.0595	0.1619123	0.7718399	2516	486	0.4716603	r6	c4	2	2
35	371.069	34.76509	0.7905234	608.1583	127.0595	0.2094766	0.7572704	2373	629	0.3442267	r6	c5	2	2
36	371.069	34.76509	0.6605705	608.1583	127.0595	0.3394295	0.9710040	1983	1019	0.2989199	r6	c6	2	2

4.2.2.3 Resolving the distributions using an EM algorithm REFERENCE:

- http://tinyheero.github.io/2016/01/03/gmm-em.html

This first function is the expectation step of the EM algorithm:

```
# Expectation Step of the EM Algorithm
eStep <- function(x, muVector, sdVector, alphaVector) {
  negDistributionProbabilities <-
    stats::dnorm(x, mean=muVector[1], sd=sdVector[1]) * alphaVector[1]
  posDistributionProbabilities <-
    stats::dnorm(x, mean=muVector[2], sd=sdVector[2]) * alphaVector[2]

  sumOfProbabilities <-
    negDistributionProbabilities + posDistributionProbabilities

  negPosteriorDistribution <-
    negDistributionProbabilities / sumOfProbabilities
  posPosteriorDistribution <-
    posDistributionProbabilities / sumOfProbabilities

  sumOfProbabilitiesLn <- log(sumOfProbabilities, base=exp(1))
  logLikelihood <- sum(sumOfProbabilitiesLn)

  list(
    "logLikelihood" = logLikelihood,
    "posteriorDf" = BiocGenerics::cbind(
      negPosteriorDistribution,
      posPosteriorDistribution
    )
  )
}
```

And this is for the maximization step of the EM algorithm:

```
# Maximization Step of the EM Algorithm
mStep <- function(x, posteriorDf) {
  negProportion <- sum(posteriorDf[, 1])
  posProportion <- sum(posteriorDf[, 2])

  negMu <- (1/negProportion) * sum(posteriorDf[, 1] * x)
  posMu <- (1/posProportion) * sum(posteriorDf[, 2] * x)

  negVar <- sum(posteriorDf[, 1] * (x - negMu)^2) * (1/negProportion)
  posVar <- sum(posteriorDf[, 2] * (x - posMu)^2) * (1/posProportion)

  negAlpha <- negProportion / length(x)
  posAlpha <- posProportion / length(x)

  list(
    "mu"    = c(negMu, posMu),
    "var"   = c(negVar, posVar),
    "alpha" = c(negAlpha, posAlpha)
  )
}
```

And we can now iterate over both steps up to 50 times for each sample:

```
solveSwarmProportions <- function(gs, gate, axis, model) {
  # Linter bindings
  muNeg <-
  muPos <-
  sigmaNeg <-
  sigmaPos <-
  pNeg <-
  pPos <-
  nNeg <-
  nPos <-
  nTot <-
  alpha <- NULL

  populationsMu <- c(model$muNeg[1], model$muPos[1])
  populationsSd <- c(model$sigmaNeg[1], model$sigmaPos[1])
  populationsProportions <- c(model$pNeg[1], model$pPos[1])

  numberOfSamples <- length(flowWorkspace::sampleNames(gs))

  for (i in seq_len(numberOfSamples)) {
    y <- flowWorkspace::gs_pop_get_data(gs, gate)
    y <- flowWorkspace::cytoset_to_flowSet(y)
    y <- y[[i]]@exprs[, axis]

    eStepResultForAlpha <- function(alpha) {
      eStep(
        y,
        muVector=populationsMu,
        sdVector=populationsSd,
        alphaVector=alpha
      )
    }

    mStepResultForPosterior <- function(posterior) {
      mStep(y, posteriorDf=posterior)
    }

    for (j in 1:50) {
      if (j == 1) {
        eStepResult <- eStepResultForAlpha(populationsProportions)
        mStepResult <- mStepResultForPosterior(eStepResult[["posteriorDf"]])
        currentLogLikelihood <- eStepResult[["logLikelihood"]]
        logLikelihoodVector  <- eStepResult[["logLikelihood"]]
      } else {
        eStepResult <- eStepResultForAlpha(mStepResult[["alpha"]])
        mStepResult <- mStepResultForPosterior(eStepResult[["posteriorDf"]])
        logLikelihoodVector <-
          c(logLikelihoodVector, eStepResult[["logLikelihood"]])
        logLikelihoodDiff <-
          abs((currentLogLikelihood - eStepResult[["logLikelihood"]]))

        if (logLikelihoodDiff < 1e-6) {
```

```
          break
        } else {
          currentLogLikelihood <- eStepResult[["logLikelihood"]]
        }
      }

      model$pNeg[i] <- mStepResult$alpha[1]
      model$pPos[i] <- mStepResult$alpha[2]
    }
  }

  model$nTot <- model$nNeg + model$nPos
  model$nNeg <- round(model$nTot * model$pNeg, digits=0)
  model$nPos <- round(model$nTot * model$pPos, digits=0)
  model
}

cd24midSwarmModel <- solveSwarmProportions(
  gs=gatingset,
  gate="CD24mid",
  axis="SS.Log",
  model=cd24midSwarmModel
)
```

Let's check out how our model data looks so far.

```
printDataFrame(
  dataframe=cd24midSwarmModel,
  caption="Swarm deconvolution final model for [CD24mid]",
  fontsize=5
)
```

Table 6: Swarm deconvolution final model for [CD24mid]

id	muNeg	sigmaNeg	pNeg	muPos	sigmaPos	pPos	hartigans	nNeg	nPos	welchTTest	facetRow	facetCol	negCluster	posCluster	nTot
1	371.069	34.76509	0.2047369	608.1583	127.0595	0.7952631	0.0015918	615	2387	0.6259356	r1	c1	1	2	3002
2	371.069	34.76509	0.0694246	608.1583	127.0595	0.9305754	0.8445203	208	2794	0.7827308	r1	c2	1	2	3002
3	371.069	34.76509	0.0473863	608.1583	127.0595	0.9526137	0.9904514	142	2860	0.7972076	r1	c3	1	2	3002
4	371.069	34.76509	0.0308621	608.1583	127.0595	0.9691379	0.8242218	93	2909	0.8778202	r1	c4	1	2	3002
5	371.069	34.76509	0.0215781	608.1583	127.0595	0.9784219	0.9456182	65	2937	0.9024751	r1	c5	1	2	3002
6	371.069	34.76509	0.0066618	608.1583	127.0595	0.9933382	0.9158683	20	2982	0.1978752	r1	c6	2	2	3002
7	371.069	34.76509	0.2022688	608.1583	127.0595	0.7977312	0.0000535	607	2395	0.6354659	r2	c1	1	2	3002
8	371.069	34.76509	0.0481041	608.1583	127.0595	0.9518959	0.9643520	144	2858	0.8272571	r2	c2	1	2	3002
9	371.069	34.76509	0.0195274	608.1583	127.0595	0.9804726	0.8426692	59	2943	0.4319169	r2	c3	2	2	3002
10	371.069	34.76509	0.0093124	608.1583	127.0595	0.9906876	0.7700217	28	2974	0.2966222	r2	c4	2	2	3002
11	371.069	34.76509	0.0018668	608.1583	127.0595	0.9981332	0.9806157	6	2996	0.5166990	r2	c5	2	2	3002
12	371.069	34.76509	0.0003041	608.1583	127.0595	0.9996959	0.9795325	1	3001	0.3334535	r2	c6	2	2	3002
13	371.069	34.76509	0.0846687	608.1583	127.0595	0.9153313	0.8964370	254	2748	0.7346465	r3	c1	1	2	3002
14	371.069	34.76509	0.0084719	608.1583	127.0595	0.9915281	0.8381467	25	2977	0.9214887	r3	c2	1	2	3002
15	371.069	34.76509	0.0043574	608.1583	127.0595	0.9956426	0.9261474	13	2989	0.3194784	r3	c3	2	2	3002
16	371.069	34.76509	0.0098683	608.1583	127.0595	0.9901317	0.9138822	30	2972	0.3337231	r3	c4	2	2	3002
17	371.069	34.76509	0.0263780	608.1583	127.0595	0.9736220	0.8852133	79	2923	0.4767543	r3	c5	2	2	3002
18	371.069	34.76509	0.0008702	608.1583	127.0595	0.9991298	0.9078441	3	2999	0.4237487	r3	c6	2	2	3002
19	371.069	34.76509	0.1841532	608.1583	127.0595	0.8158468	0.0018273	553	2449	0.7339639	r4	c1	1	2	3002
20	371.069	34.76509	0.0586220	608.1583	127.0595	0.9413780	0.8697411	176	2826	0.8199796	r4	c2	1	2	3002
21	371.069	34.76509	0.0167500	608.1583	127.0595	0.9832500	0.9922402	50	2952	0.9337305	r4	c3	1	2	3002
22	371.069	34.76509	0.0280624	608.1583	127.0595	0.9719376	0.9746790	84	2918	0.2699923	r4	c4	2	2	3002
23	371.069	34.76509	0.0110263	608.1583	127.0595	0.9889737	0.9901663	33	2969	0.2598915	r4	c5	2	2	3002
24	371.069	34.76509	0.0250814	608.1583	127.0595	0.9749186	0.6375427	75	2927	0.2653685	r4	c6	2	2	3002
25	371.069	34.76509	0.0815032	608.1583	127.0595	0.9184968	0.6567281	245	2757	0.9429594	r5	c1	1	2	3002
26	371.069	34.76509	0.0313841	608.1583	127.0595	0.9686159	0.9657327	94	2908	0.2571137	r5	c2	2	2	3002
27	371.069	34.76509	0.0119020	608.1583	127.0595	0.9880980	0.9899165	36	2966	0.2729455	r5	c3	2	2	3002
28	371.069	34.76509	0.0021683	608.1583	127.0595	0.9978317	0.9173520	7	2995	0.3028510	r5	c4	2	2	3002
29	371.069	34.76509	0.0001875	608.1583	127.0595	0.9998125	0.9397857	1	3001	0.3711630	r5	c5	2	2	3002
30	371.069	34.76509	0.0277244	608.1583	127.0595	0.9722756	0.7712460	83	2919	0.3077638	r5	c6	2	2	3002
31	371.069	34.76509	0.1385664	608.1583	127.0595	0.8614336	0.0497850	416	2586	0.7280116	r6	c1	1	2	3002

Table 6: Swarm deconvolution final model for [CD24mid] *(continued)*

id	muNeg	sigmaNeg	pNeg	muPos	sigmaPos	pPos	hartigans	nNeg	nPos	welchTTest	facetRow	facetCol	negCluster	posCluster	nTot
32	371.069	34.76509	0.0269819	608.1583	127.0595	0.9730181	0.9569939	81	2921	0.9072445	r6	c2	1	2	3002
33	371.069	34.76509	0.0152019	608.1583	127.0595	0.9847981	0.9764470	46	2956	0.8930675	r6	c3	1	2	3002
34	371.069	34.76509	0.0000852	608.1583	127.0595	0.9999148	0.7718399	0	3002	0.4716603	r6	c4	2	2	3002
35	371.069	34.76509	0.0000012	608.1583	127.0595	0.9999988	0.7572704	0	3002	0.3442267	r6	c5	2	2	3002
36	371.069	34.76509	0.0004289	608.1583	127.0595	0.9995711	0.9710040	1	3001	0.2989199	r6	c6	2	2	3002

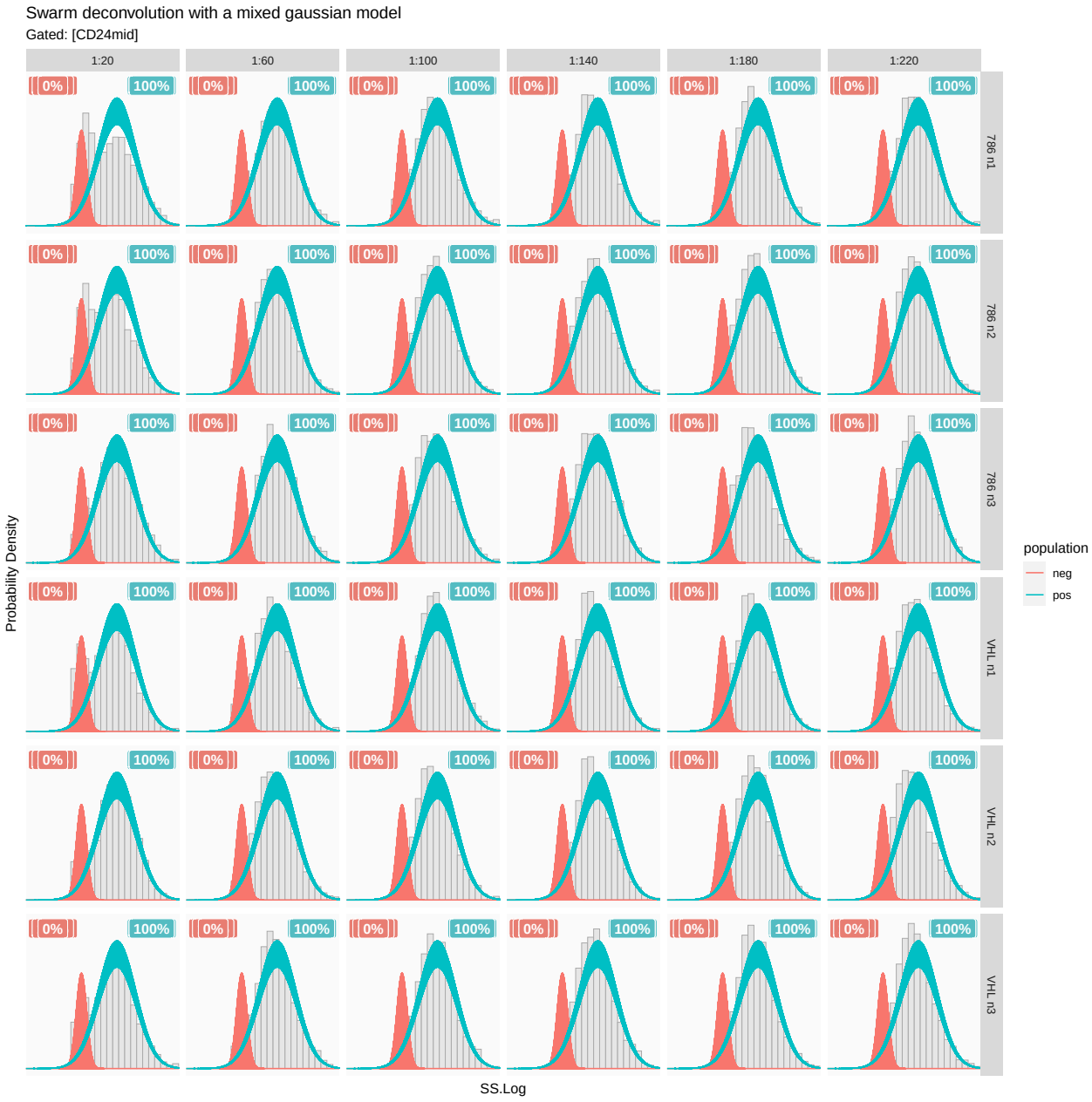
4.2.2.4 Plotting the solutions

Now that we have obtained the optimal solution for our data set, we can create model distributions for use in plotting:

```
genDistributionsForPlotting <- function(model) {
  numberOfFlowframes <- length(row.names(model))
  mixedModelDistributions <- list()

  for (i in seq_len(numberOfFlowframes)) {
    x <- seq(0, 1023)

    negDistribution <-
      stats::dnorm(x, mean=model$muNeg[i], sd=model$sigmaNeg[i]) * model$pNeg[i]

    posDistribution <-
      stats::dnorm(x, mean=model$muPos[i], sd=model$sigmaPos[i]) * model$pPos[i]

    facetRow <- paste0("r", ((i-1) %/% 6)+1)
    facetCol <- paste0("c", ((i-1) %% 6)+1)

    tempData <- data.frame(
      "distributionXAxis" = x,
      "distributionYAxis" = c(negDistribution, posDistribution),
      "population"        = rep(c("neg", "pos"), each=1024),
      "facetRow"          = rep(facetRow, times=2048),
      "facetCol"          = rep(facetCol, times=2048)
    )

    mixedModelDistributions <-
      BiocGenerics::rbind(mixedModelDistributions, tempData)
  }

  mixedModelDistributions
}

plotDeconvolutedExperiment <- function(gs, model, gate, axis) {
  mixedModelDistributions <- genDistributionsForPlotting(model)

  ggplot2::ggplot(
    data=flowWorkspace::gs_pop_get_data(gs, gate),
    mapping=ggplot2::aes_string(x=axis)
  ) +
    ggplot2::scale_x_continuous(
      limits=c(0, 1024),
      expand=c(0, 0)) +
    ggplot2::geom_histogram(
      mapping=ggplot2::aes_string(
        x=axis,
        y="..density.."),
      binwidth=40,
      colour="#AAAAAA",
      fill="#E6E6E6",
      size=0.1) +
    ggplot2::geom_line(
      data=mixedModelDistributions,
```

```
      mapping=ggplot2::aes(
        x=distributionXAxis,
        y=distributionYAxis,
        colour=population)) +
    ggplot2::labs(
      title="Swarm deconvolution with a mixed gaussian model",
      subtitle=paste0("Gated: [", gate, "]"),
      x=axis,
      y="Probability Density") +
    ggplot2::theme(
      panel.grid.major=ggplot2::element_blank(),
      panel.grid.minor=ggplot2::element_blank(),
      panel.background=ggplot2::element_rect(fill="#FAFAFA"),
      axis.text.x=ggplot2::element_blank(),
      axis.text.y=ggplot2::element_blank(),
      axis.ticks.x=ggplot2::element_blank(),
      axis.ticks.y=ggplot2::element_blank()) +
    ggplot2::geom_label(
      data=model,
      x=180,
      y=0.00345,
      mapping=ggplot2::aes(label=paste0(round(pNeg*100, 1), "%")),
      fill="#E87D72",
      color="#FFFFFF",
      fontface="bold",
      size=4) +
    ggplot2::geom_label(
      data=model,
      x=840,
      y=0.00345,
      mapping=ggplot2::aes(label=paste0(round(pPos*100, 1), "%")),
      fill="#54BCC2",
      color="#FFFFFF",
      fontface="bold",
      size=4) +
    ggplot2::facet_grid(
      rows=dplyr::vars(FacetRow),
      cols=dplyr::vars(FacetCol),
      labeller=ggplot2::as_labeller(facetLabels))
}

plotDeconvolutedExperiment(
  gs=gatingset,
  model=cd24midSwarmModel,
  gate="CD24mid",
  axis="SS.Log"
)
```

```
## Warning: Removed 72 rows containing missing values (geom_bar).
```


Swarm deconvolution with a mixed gaussian model
Gated: [CD24mid]
1:20
1:60
1:100
1:140
1:180
1:220
786 n1
786 n2
786 n3
VHL n1
VHL n2
VHL n3
0%
100%
Probability Density
SS.Log
population
neg
pos

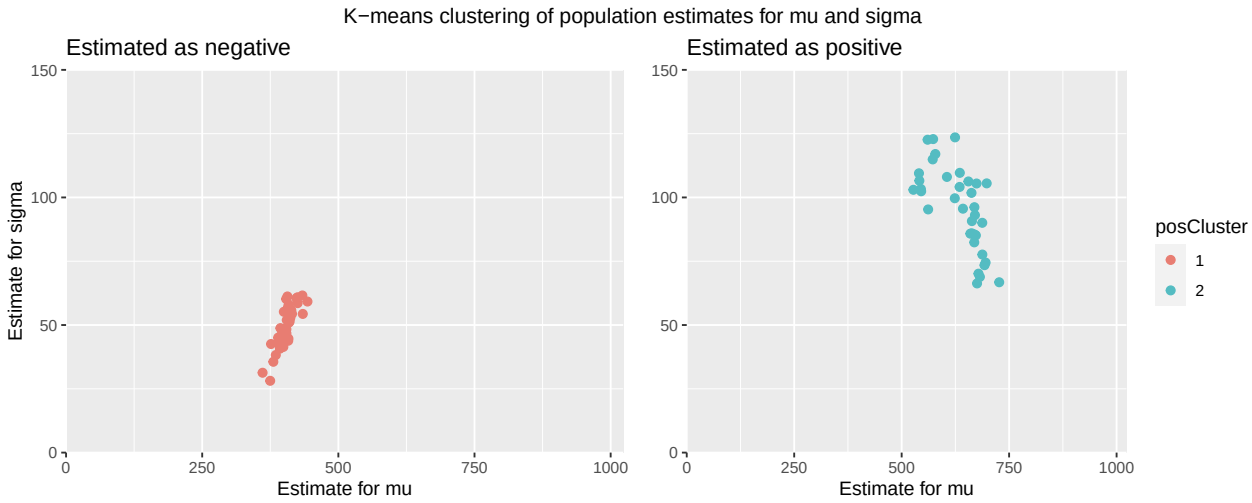
4.2.3 Putting it all together: CD24hi swarm deconvolution

```
estimate <- list(
  "mu"    = c(373, 650),
  "sigma" = c(80, 80)
)

cd24hiSwarmModel <- generateSwarmModel(
  gs=gatingset,
  axis="SS.Log",
  gate="CD24hi",
  mu=estimate[["mu"]],
  sigma=estimate[["sigma"]],
  maxIterations=10000
)
```

```
## number of iterations= 178
## number of iterations= 98
## number of iterations= 101
## number of iterations= 52
## number of iterations= 81
## number of iterations= 56
## number of iterations= 93
## number of iterations= 84
## number of iterations= 56
## number of iterations= 49
## number of iterations= 62
## number of iterations= 18
## number of iterations= 80
## number of iterations= 73
## number of iterations= 141
## number of iterations= 88
## number of iterations= 34
## number of iterations= 23
## number of iterations= 46
## number of iterations= 31
## number of iterations= 46
## number of iterations= 18
## number of iterations= 41
## number of iterations= 12
## number of iterations= 277
## number of iterations= 34
## number of iterations= 33
## number of iterations= 39
## number of iterations= 37
## number of iterations= 16
## number of iterations= 38
## number of iterations= 57
## number of iterations= 79
## number of iterations= 7
## number of iterations= 28
## number of iterations= 32
```

```
cd24hiSwarmModel <- clusterModel(
  model=cd24hiSwarmModel,
```

```
  centers=matrix(c(estimate[["mu"]], estimate[["sigma"]]), ncol=2)
)

plotModelClusters(
  model=cd24hiSwarmModel,
  ylim=c(0, 150),
  xlim=c(0, 1023)
)
```




```
cd24hiSwarmModel <- applyGlobalParameters(
  model=cd24hiSwarmModel,
  params=estimateGlobalParameters(cd24hiSwarmModel)
)

cd24hiSwarmModel <- solveSwarmProportions(
  gs=gatingset,
  gate="CD24hi",
  axis="SS.Log",
  model=cd24hiSwarmModel
)

printDataFrame(
  dataframe=cd24hiSwarmModel,
  caption="Swarm deconvolution final model for [CD24hi]",
  fontsize=5
)
```

Table 7: Swarm deconvolution final model for [CD24hi]

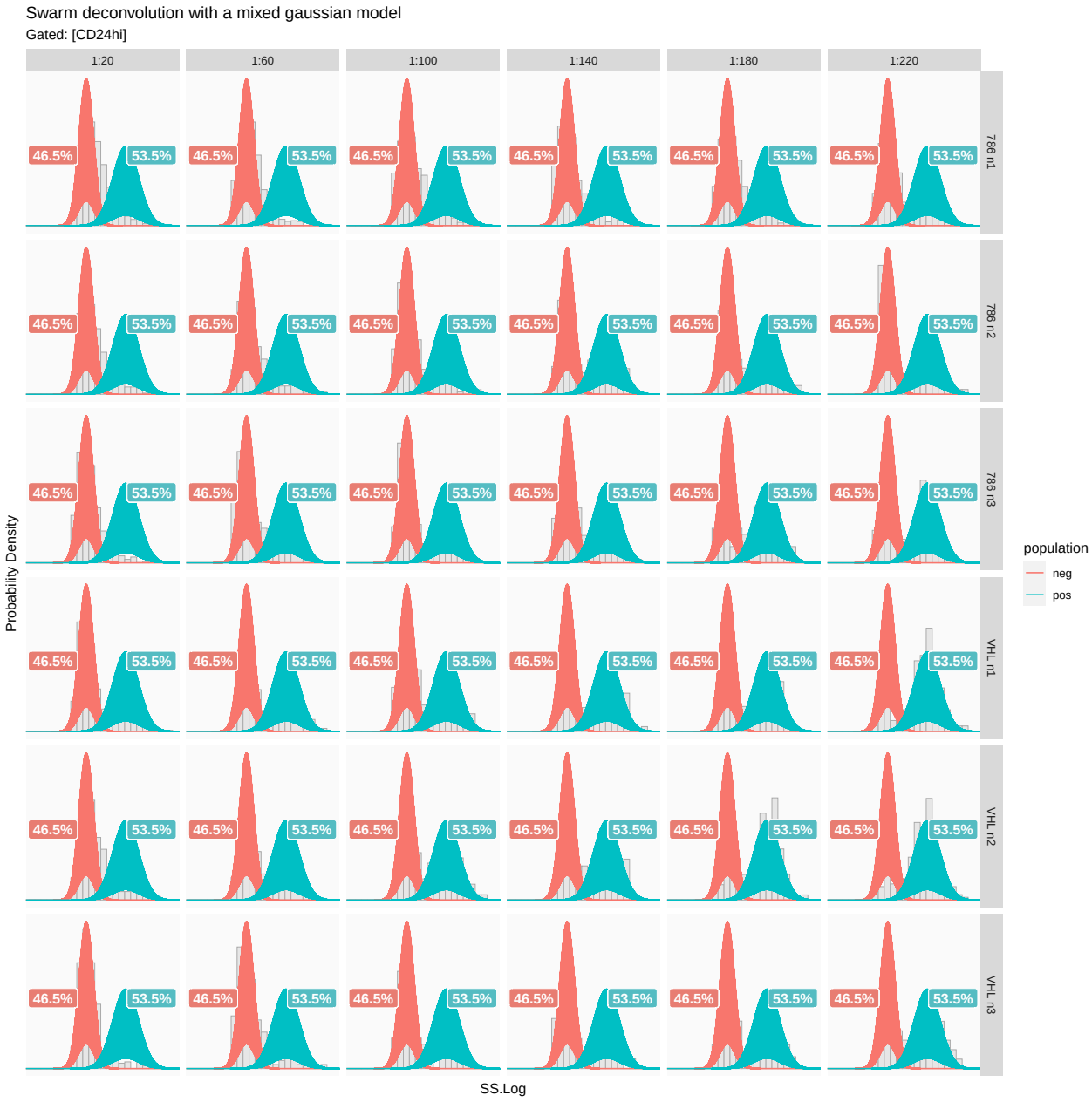
id	muNeg	sigmaNeg	pNeg	muPos	sigmaPos	pPos	hartigans	nNeg	nPos	welchTTest	facetRow	facetCol	negCluster	posCluster	nTot
1	403.0371	49.24023	0.7273042	666.7125	87.27655	0.2726958	0.4342748	39	15	0.0577825	r1	c1	1	2	54
2	403.0371	49.24023	0.8693641	666.7125	87.27655	0.1306359	0.9191252	47	7	0.0418459	r1	c2	1	2	54
3	403.0371	49.24023	0.8123519	666.7125	87.27655	0.1876481	0.9924966	44	10	0.0441529	r1	c3	1	2	54
4	403.0371	49.24023	0.8267913	666.7125	87.27655	0.1732087	0.9821544	45	9	0.0546781	r1	c4	1	2	54
5	403.0371	49.24023	0.7940721	666.7125	87.27655	0.2059279	0.9923969	43	11	0.0585480	r1	c5	1	2	54
6	403.0371	49.24023	0.7457578	666.7125	87.27655	0.2542422	0.9932503	40	14	0.0871633	r1	c6	1	2	54
7	403.0371	49.24023	0.8000095	666.7125	87.27655	0.1999905	0.9105907	43	11	0.0551437	r2	c1	1	2	54
8	403.0371	49.24023	0.8026227	666.7125	87.27655	0.1973773	0.9945522	43	11	0.0466224	r2	c2	1	2	54
9	403.0371	49.24023	0.7170213	666.7125	87.27655	0.2829787	0.0899201	39	15	0.1324204	r2	c3	1	2	54
10	403.0371	49.24023	0.5486570	666.7125	87.27655	0.4513430	0.1515512	30	24	0.1416749	r2	c4	1	2	54
11	403.0371	49.24023	0.5088566	666.7125	87.27655	0.4911434	0.1195755	27	27	0.1218257	r2	c5	1	2	54
12	403.0371	49.24023	0.5918487	666.7125	87.27655	0.4081513	0.0023661	32	22	0.1873885	r2	c6	1	2	54
13	403.0371	49.24023	0.8878484	666.7125	87.27655	0.1121516	0.8378481	48	6	0.0433730	r3	c1	1	2	54
14	403.0371	49.24023	0.8144606	666.7125	87.27655	0.1855394	0.9901901	44	10	0.0607365	r3	c2	1	2	54

Table 7: Swarm deconvolution final model for [CD24hi] *(continued)*

id	muNeg	sigmaNeg	pNeg	muPos	sigmaPos	pPos	hartigans	nNeg	nPos	welchTTest	facetRow	facetCol	negCluster	posCluster	nTot
15	403.0371	49.24023	0.6344645	666.7125	87.27655	0.3655355	0.0236009	34	20	0.1033086	r3	c3	1	2	54
16	403.0371	49.24023	0.5095365	666.7125	87.27655	0.4904635	0.5332946	28	26	0.1414072	r3	c4	1	2	54
17	403.0371	49.24023	0.3407183	666.7125	87.27655	0.6592817	0.0006891	18	36	0.1579793	r3	c5	1	2	54
18	403.0371	49.24023	0.3621821	666.7125	87.27655	0.6378179	0.0051107	20	34	0.2355597	r3	c6	1	2	54
19	403.0371	49.24023	0.7580413	666.7125	87.27655	0.2419587	0.9549641	41	13	0.0609299	r4	c1	1	2	54
20	403.0371	49.24023	0.4967615	666.7125	87.27655	0.5032385	0.0243285	27	27	0.1689622	r4	c2	1	2	54
21	403.0371	49.24023	0.5537867	666.7125	87.27655	0.4462133	0.6175997	30	24	0.1158113	r4	c3	1	2	54
22	403.0371	49.24023	0.3902403	666.7125	87.27655	0.6097597	0.0080804	21	33	0.2393066	r4	c4	1	2	54
23	403.0371	49.24023	0.3867308	666.7125	87.27655	0.6132692	0.0056401	21	33	0.1965352	r4	c5	1	2	54
24	403.0371	49.24023	0.1914883	666.7125	87.27655	0.8085117	0.3538832	10	44	0.2603656	r4	c6	1	2	54
25	403.0371	49.24023	0.7394516	666.7125	87.27655	0.2605484	0.9925139	40	14	0.0674255	r5	c1	1	2	54
26	403.0371	49.24023	0.5137019	666.7125	87.27655	0.4862981	0.0061349	28	26	0.2209996	r5	c2	1	2	54
27	403.0371	49.24023	0.3321749	666.7125	87.27655	0.6678251	0.2398218	18	36	0.1556182	r5	c3	1	2	54
28	403.0371	49.24023	0.3119173	666.7125	87.27655	0.6880827	0.2860918	17	37	0.2059403	r5	c4	1	2	54
29	403.0371	49.24023	0.2362630	666.7125	87.27655	0.7637370	0.2530825	13	41	0.1959022	r5	c5	1	2	54
30	403.0371	49.24023	0.1497181	666.7125	87.27655	0.8502819	0.7357624	8	46	0.2186312	r5	c6	1	2	54
31	403.0371	49.24023	0.8810596	666.7125	87.27655	0.1189404	0.9939827	48	6	0.0392644	r6	c1	1	2	54
32	403.0371	49.24023	0.8033041	666.7125	87.27655	0.1966959	0.9546455	43	11	0.0487752	r6	c2	1	2	54
33	403.0371	49.24023	0.6448605	666.7125	87.27655	0.3551395	0.0632619	35	19	0.1160994	r6	c3	1	2	54
34	403.0371	49.24023	0.6206474	666.7125	87.27655	0.3793526	0.1258550	34	20	0.2565231	r6	c4	1	2	54
35	403.0371	49.24023	0.5794298	666.7125	87.27655	0.4205702	0.3785653	31	23	0.1469781	r6	c5	1	2	54
36	403.0371	49.24023	0.4649803	666.7125	87.27655	0.5350197	0.0095742	25	29	0.2754266	r6	c6	1	2	54

```
plotDeconvolutedExperiment(
  gs=gatingset,
  model=cd24hiSwarmModel,
  gate="CD24hi",
  axis="SS.Log"
)
```

```
## Warning: Removed 72 rows containing missing values (geom_bar).
```




```
distForAggregatePlotting <- function(model) {
  numberOfFlowframes <- length(row.names(model))
  mixedModelDistributions <- list()

  for (i in seq_len(numberOfFlowframes)) {
    x <- seq(0, 1023)

    negDistribution <- stats::dnorm(
      x,
      mean=model$muNeg[i],
      sd=model$sigmaNeg[i]
    ) * model$pNeg[i]

    posDistribution <- stats::dnorm(
```

```
      x,
      mean=model$muPos[i],
      sd=model$sigmaPos[i]
    ) * model$pPos[i]

    cellLine <- model$cellLine[i]
    dilution <- model$dilution[i]

    tempData <- data.frame(
      "distributionXAxis" = x,
      "distributionYAxis" = c(negDistribution, posDistribution),
      "population"        = rep(c("neg", "pos"), each=1024),
      "cellLine"          = rep(cellLine, times=2048),
      "dilution"          = rep(dilution, times=2048)
    )

    mixedModelDistributions <-
      BiocGenerics::rbind(mixedModelDistributions, tempData)
  }

  mixedModelDistributions
}

plotDeconvAggregateExperiment <- function(gs, model, gate, axis) {
  # Linter bindings
  cellLine <-
  dilution <- NULL

  mapFacetRowToCellLine <- wrapr::qc(
    "r1" = "786-O",
    "r2" = "786-O",
    "r3" = "786-O",
    "r4" = "VHL",
    "r5" = "VHL",
    "r6" = "VHL"
  )

  mapFacetColToDilution <- wrapr::qc(
    "c1" = "20",
    "c2" = "60",
    "c3" = "100",
    "c4" = "140",
    "c5" = "180",
    "c6" = "220"
  )

  model <- model %>%
    dplyr::mutate(., cellLine=mapFacetRowToCellLine[facetRow])

  model <- model %>%
    dplyr::mutate(., dilution=mapFacetColToDilution[facetCol])

  model <- model %>%
    group_by(cellLine, dilution) %>%
```

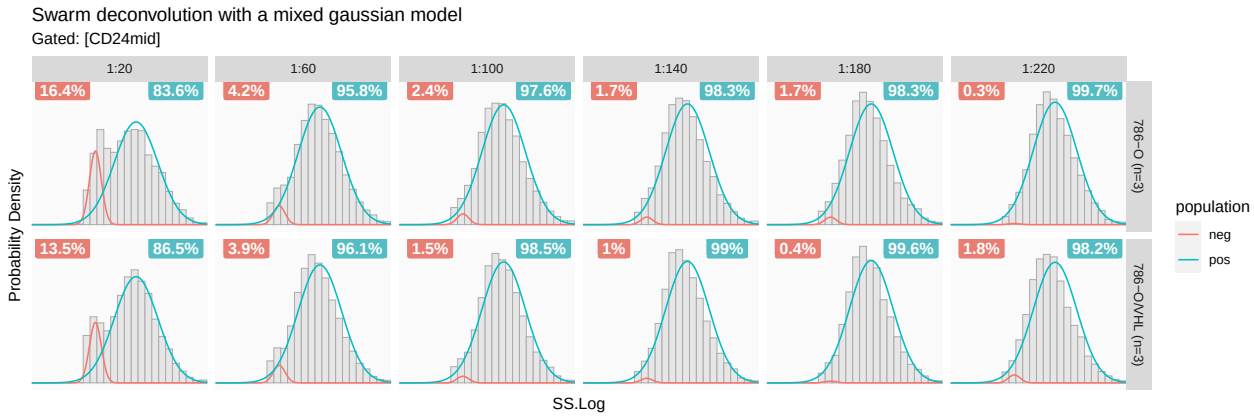
```
    summarize(
      muNeg    = mean(muNeg),
      sigmaNeg = mean(sigmaNeg),
      muPos    = mean(muPos),
      sigmaPos = mean(sigmaPos),
      pNeg     = sum(nNeg) / sum(nTot),
      pPos     = sum(nPos) / sum(nTot),
      nNeg     = sum(nNeg),
      nPos     = sum(nPos),
      nTot     = sum(nTot)
    )

mixedModelDistributions <- distForAggregatePlotting(model)

cellLineOrdering <- c("786-O", "VHL")
dilutionOrdering <- c("20", "60", "100", "140", "180", "220")

model$CellLine <- factor(model$cellLine, levels=cellLineOrdering)
model$Dilution <- factor(model$dilution, levels=dilutionOrdering)

mixedModelDistributions$CellLine <- factor(
  mixedModelDistributions$cellLine,
  levels=cellLineOrdering
)

mixedModelDistributions$Dilution <- factor(
  mixedModelDistributions$dilution,
  levels=c(dilutionOrdering)
)

data <- flowWorkspace::gs_pop_get_data(gs, gate)
data <- flowWorkspace::cytoset_to_flowSet(data)

data@phenoData@data$Dilution <- factor(
  data@phenoData@data$Dilution,
  levels=dilutionOrdering
)

data@phenoData@data$CellLine <- factor(
  data@phenoData@data$CellLine,
  levels=cellLineOrdering
)

ggplot2::ggplot(
  data=data,
  mapping=aes_string(x=axis)
) +
  ggplot2::scale_x_continuous(
    limits=c(0, 1024),
    expand=c(0, 0)) +
  ggplot2::geom_histogram(
    mapping=ggplot2::aes_string(
      x=axis,
```

```
      y="..density.."),
    binwidth=40,
    colour="#AAAAAA",
    fill="#E6E6E6",
    size=0.1) +
  ggplot2::geom_line(
    data=mixedModelDistributions,
    mapping=ggplot2::aes(
      x=distributionXAxis,
      y=distributionYAxis,
      colour=population)) +
  ggplot2::labs(
    title="Swarm deconvolution with a mixed gaussian model",
    subtitle=paste0("Gated: [", gate, "]"),
    x=axis,
    y="Probability Density") +
  ggplot2::theme(
    panel.grid.major=ggplot2::element_blank(),
    panel.grid.minor=ggplot2::element_blank(),
    panel.background=ggplot2::element_rect(fill="#FAFAFA"),
    axis.text.x=ggplot2::element_blank(),
    axis.text.y=ggplot2::element_blank(),
    axis.ticks.x=ggplot2::element_blank(),
    axis.ticks.y=ggplot2::element_blank()) +
  ggplot2::geom_label(
    data=model,
    x=180,
    y=0.00345,
    mapping=ggplot2::aes(label=paste0(round(pNeg*100, 1), "%")),
    fill="#E87D72",
    color="#FFFFFF",
    fontface="bold",
    size=4) +
  ggplot2::geom_label(
    data=model,
    x=840,
    y=0.00345,
    mapping=ggplot2::aes(label=paste0(round(pPos*100, 1), "%")),
    fill="#54BCC2",
    color="#FFFFFF",
    fontface="bold",
    size=4) +
  ggplot2::facet_grid(
    rows=dplyr::vars(CellLine),
    cols=dplyr::vars(Dilution),
    labeller=ggplot2::as_labeller(c(
      "20"    = "1:20",
      "60"    = "1:60",
      "100"   = "1:100",
      "140"   = "1:140",
      "180"   = "1:180",
      "220"   = "1:220",
      "786-O" = "786-O (n=3)",
```

```
        "VHL"    = "786-O/VHL (n=3)"
      )))
}

plotDeconvAggregateExperiment(
  gs=gatingset,
  model=cd24midSwarmModel,
  gate="CD24mid",
  axis="SS.Log"
)
```

```
## `summarise()` regrouping output by 'cellLine' (override with `.groups` argument)
```

```
## Warning: Removed 24 rows containing missing values (geom_bar).
```




```
plotDeconvAggregateExperiment(
  gs=gatingset,
  model=cd24hiSwarmModel,
  gate="CD24hi",
  axis="SS.Log"
)
```

```
## `summarise()` regrouping output by 'cellLine' (override with `.groups` argument)
```

```
## Warning: Removed 24 rows containing missing values (geom_bar).
```

4.3 Small particle quantification

4.3.1 Building our datasets

```
# Create a dataframe to hold all the experimental data obtained
experiment <- pData

# Get a list of all relevant populations
populations <- flowWorkspace::gs_get_pop_paths(gatingset, path="auto")
populations <- populations[!(
  populations %in% c("root", "Boundary", "NonNoise")
)]

populations
```

```
## [1] "Beads"   "Events"  "CD24pos" "CD24neg" "CD24mid" "CD24hi"
```

```
# Get a list of all samples
samples <- row.names(experiment)

# Get all population statistics
populationStatistics <- flowWorkspace::gs_pop_get_count_fast(gatingset)

# Populate the experimental data with all the relevant gate population counts,
# in "tidy" format
#
# References:
# https://www.jstatsoft.org/article/view/v059i10

for (sample in samples) {
  # https://github.com/RGLab/flowWorkspace/issues/341#issuecomment-691742173
  escapedSampleName <- gsub("[/:\\\\]", "_", sample)
  sampleStatistic <- populationStatistics %>%
    dplyr::filter(populationStatistics$name == escapedSampleName)

  sampleStatistic$Population <- sampleStatistic$Population %>%
    purrr::map(function(x) {
      tail(strsplit(x, "/")[[1]], n=1)
    })

  for (population in populations) {
    samplePopulationStatistic <- sampleStatistic %>%
      dplyr::filter(sampleStatistic$Population == population)

    experiment[sample, population] <- samplePopulationStatistic$Count
  }
}

orderOfCellLines <- c("786-O", "VHL")

experiment$Temps    <- as.factor(experiment$Temps)
experiment$A23187   <- as.factor(experiment$A23187)
experiment$N        <- as.factor(experiment$N)
experiment$CellLine <- factor(experiment$CellLine, levels=orderOfCellLines)
```

```r
# Integrate swarm deconvolution data. The true-positive statistic for the CD24+
# population can be derived from CD24mid and CD24hi
experiment$cd24midTp <- cd24midSwarmModel$pPos
experiment$cd24hiTp  <- cd24hiSwarmModel$pPos
experiment$cd24posTp <- (
  (experiment$CD24mid * experiment$cd24midTp) +
  (experiment$CD24hi * experiment$cd24hiTp)) / experiment$CD24pos

# Let's not go too crazy with significant digits
experiment$cd24midTp <- round(experiment$cd24midTp, 3)
experiment$cd24hiTp  <- round(experiment$cd24hiTp, 3)
experiment$cd24posTp <- round(experiment$cd24posTp, 3)
```

```
interestingColumns <- c(
  "Beads",
  "Events",
  "CD24pos",
  "CD24neg",
  "CD24mid",
  "CD24hi",
  "cd24posTp",
  "cd24midTp",
  "cd24hiTp"
)

printDataFrame(
  dataframe=experiment[, interestingColumns],
  caption="Gated event counts",
  fontsize=5
)
```

Table 8: Gated event counts

	Beads	Events	CD24pos	CD24neg	CD24mid	CD24hi	cd24posTp	cd24midTp	cd24hiTp
data/786 n1/786n1_Ca1uM4h_samp_20x 00016119 613.LMD	3069	15086	7701	7357	4481	3220	0.577	0.795	0.273
data/786 n1/786n1_Ca1uM4h_samp_60x 00016122 616.LMD	3390	12765	4304	8458	3580	724	0.796	0.931	0.131
data/786 n1/786n1_Ca1uM4h_samp_100x 00016125 619.LMD	3417	11809	3819	7987	3353	466	0.859	0.953	0.188
data/786 n1/786n1_Ca1uM4h_samp_140x 00016128 622.LMD	3688	11828	3421	8406	3149	272	0.906	0.969	0.173
data/786 n1/786n1_Ca1uM4h_samp_180x 00016131 625.LMD	3670	11650	3346	8304	3151	195	0.933	0.978	0.206
data/786 n1/786n1_Ca1uM4h_samp_220x 00016134 628.LMD	3561	11313	3214	8098	3040	174	0.953	0.993	0.254
data/786 n2/786n2_Ca1uM4h_samp_20x 00016158 632.LMD	2897	13595	5381	8209	4470	911	0.697	0.798	0.200
data/786 n2/786n2_Ca1uM4h_samp_60x 00016161 635.LMD	2843	10372	3231	7141	2984	247	0.894	0.952	0.197
data/786 n2/786n2_Ca1uM4h_samp_100x 00016164 638.LMD	2643	9591	2979	6612	2754	225	0.928	0.980	0.283
data/786 n2/786n2_Ca1uM4h_samp_140x 00016167 641.LMD	3102	10394	3119	7275	2880	239	0.949	0.991	0.451
data/786 n2/786n2_Ca1uM4h_samp_180x 00016170 644.LMD	3110	10013	2942	7071	2766	176	0.968	0.998	0.491
data/786 n2/786n2_Ca1uM4h_samp_220x 00016173 647.LMD	2748	8986	2492	6494	2385	107	0.974	1.000	0.408
data/786 n3/786n3_Ca1uM4h_samp_20x 00016176 650.LMD	2936	11928	4315	7608	3433	882	0.751	0.915	0.112
data/786 n3/786n3_Ca1uM4h_samp_60x 00016179 653.LMD	2917	10277	3259	7016	2993	266	0.926	0.992	0.186
data/786 n3/786n3_Ca1uM4h_samp_100x 00016182 656.LMD	3047	10641	3218	7423	2992	226	0.951	0.996	0.366
data/786 n3/786n3_Ca1uM4h_samp_140x 00016185 659.LMD	3314	10788	3292	7496	3143	149	0.968	0.990	0.490
data/786 n3/786n3_Ca1uM4h_samp_180x 00016188 662.LMD	3719	12309	3786	8523	3535	251	0.953	0.974	0.659
data/786 n3/786n3_Ca1uM4h_samp_220x 00016191 665.LMD	3650	11442	3490	7952	3317	173	0.981	0.999	0.638
data/VHL n1/VHLn1_Ca1uM4h_samp_20x 00016196 670.LMD	3348	12604	4499	8104	4042	457	0.758	0.816	0.242
data/VHL n1/VHLn1_Ca1uM4h_samp_60x 00016199 673.LMD	3350	12426	3793	8632	3622	171	0.922	0.941	0.503
data/VHL n1/VHLn1_Ca1uM4h_samp_100x 00016202 676.LMD	3322	10681	3039	7642	2981	58	0.973	0.983	0.446
data/VHL n1/VHLn1_Ca1uM4h_samp_140x 00016205 679.LMD	3476	10848	3213	7634	3106	107	0.960	0.972	0.610
data/VHL n1/VHLn1_Ca1uM4h_samp_180x 00016208 682.LMD	3223	10540	3097	7443	3015	82	0.979	0.989	0.613
data/VHL n1/VHLn1_Ca1uM4h_samp_220x 00016211 685.LMD	3597	11140	3271	7869	3177	94	0.970	0.975	0.809
data/VHL n2/VHLn2_Ca1uM4h_samp_20x 00016215 689.LMD	2992	13528	4279	9248	3977	302	0.872	0.918	0.261
data/VHL n2/VHLn2_Ca1uM4h_samp_60x 00016218 692.LMD	3397	11723	3563	8160	3426	137	0.950	0.969	0.486
data/VHL n2/VHLn2_Ca1uM4h_samp_100x 00016221 695.LMD	3499	11905	3471	8434	3374	97	0.979	0.988	0.668
data/VHL n2/VHLn2_Ca1uM4h_samp_140x 00016224 698.LMD	3535	12786	3406	9379	3331	75	0.991	0.998	0.688
data/VHL n2/VHLn2_Ca1uM4h_samp_180x 00016227 701.LMD	3701	11792	3464	8327	3364	100	0.993	1.000	0.764
data/VHL n2/VHLn2_Ca1uM4h_samp_220x 00016230 704.LMD	3499	11515	3695	7820	3504	191	0.966	0.972	0.850
data/VHL n3/VHLn3_Ca1uM4h_samp_20x 00016234 708.LMD	3278	14249	4344	9905	3963	381	0.796	0.861	0.119
data/VHL n3/VHLn3_Ca1uM4h_samp_60x 00016237 711.LMD	3200	11494	3350	8144	3224	126	0.944	0.973	0.197
data/VHL n3/VHLn3_Ca1uM4h_samp_100x 00016240 714.LMD	3054	11090	3076	8014	2992	84	0.968	0.985	0.355
data/VHL n3/VHLn3_Ca1uM4h_samp_140x 00016243 717.LMD	3234	10889	3104	7785	3053	51	0.990	1.000	0.379
data/VHL n3/VHLn3_Ca1uM4h_samp_180x 00016246 720.LMD	3092	11157	3011	8146	2968	43	0.992	1.000	0.421
data/VHL n3/VHLn3_Ca1uM4h_samp_220x 00016249 723.LMD	3223	10951	3056	7895	3002	54	0.991	1.000	0.535

```r
# Constants for bead quantification procedure
normalizedToXBeads <- 5000

normalizedExperiment <- experiment

normalizedExperiment$Events <-
  (experiment$Events / experiment$Beads) *
    normalizedToXBeads * experiment$Dilution * 10

normalizedExperiment$CD24pos <-
  (experiment$CD24pos / experiment$Beads) *
    normalizedToXBeads * experiment$Dilution * 10

normalizedExperiment$CD24neg <-
  (experiment$CD24neg / experiment$Beads) *
    normalizedToXBeads * experiment$Dilution * 10

normalizedExperiment$CD24mid <-
  (experiment$CD24mid / experiment$Beads) *
    normalizedToXBeads * experiment$Dilution * 10

normalizedExperiment$CD24hi <-
  (experiment$CD24hi / experiment$Beads) *
    normalizedToXBeads * experiment$Dilution * 10

# To be or not to be, that is the question.
normalizedExperiment$Events  <- round(normalizedExperiment$Events,  0)
normalizedExperiment$CD24pos <- round(normalizedExperiment$CD24pos, 0)
normalizedExperiment$CD24neg <- round(normalizedExperiment$CD24neg, 0)
normalizedExperiment$CD24mid <- round(normalizedExperiment$CD24mid, 0)
normalizedExperiment$CD24hi  <- round(normalizedExperiment$CD24hi,  0)

# For normalized data, bead counts lose their significance...
# but we may be interested in the events' corresponding concentration
# in undiluted conditioned-medium (CM)
normalizedExperiment$Beads    <- NULL
normalizedExperiment$cmVolume <- 40 # TEMPORARY, FIX THIS
```

```
interestingColumns <- c(
  "cmVolume",
  "Events",
  "CD24pos",
  "CD24neg",
  "CD24mid",
  "CD24hi",
  "cd24posTp",
  "cd24midTp",
  "cd24hiTp"
)

printDataFrame(
  dataframe=normalizedExperiment[, interestingColumns],
  caption="Gated event counts",
  fontsize=5
)
```

Table 9: Gated event counts

	cmVolume	Events	CD24pos	CD24neg	CD24mid	CD24hi	cd24posTp	cd24midTp	cd24hiTp
data/786 n1/786n1_Ca1uM4h_samp_20x 00016119 613.LMD	40	4915608	2509286	2397198	1460085	1049202	0.577	0.795	0.273
data/786 n1/786n1_Ca1uM4h_samp_60x 00016122 616.LMD	40	11296460	3808850	7484956	3168142	640708	0.796	0.931	0.131
data/786 n1/786n1_Ca1uM4h_samp_100x 00016125 619.LMD	40	17279778	5588235	11687152	4906351	681885	0.859	0.953	0.188
data/786 n1/786n1_Ca1uM4h_samp_140x 00016128 622.LMD	40	22450108	6493221	15954989	5976952	516269	0.906	0.969	0.173
data/786 n1/786n1_Ca1uM4h_samp_180x 00016131 625.LMD	40	28569482	8205450	20364033	7727248	478202	0.933	0.978	0.206
data/786 n1/786n1_Ca1uM4h_samp_220x 00016134 628.LMD	40	34946083	9928110	25014883	9390621	537489	0.953	0.993	0.254
data/786 n2/786n2_Ca1uM4h_samp_20x 00016158 632.LMD	40	4692786	1857439	2833621	1542975	314463	0.697	0.798	0.200
data/786 n2/786n2_Ca1uM4h_samp_60x 00016161 635.LMD	40	10944777	3409427	7535350	3148786	260640	0.894	0.952	0.197
data/786 n2/786n2_Ca1uM4h_samp_100x 00016164 638.LMD	40	18144154	5635641	12508513	5209989	425653	0.928	0.980	0.283
data/786 n2/786n2_Ca1uM4h_samp_140x 00016167 641.LMD	40	23455190	7038362	16416828	6499033	539329	0.949	0.991	0.451
data/786 n2/786n2_Ca1uM4h_samp_180x 00016170 644.LMD	40	28976527	8513826	20462701	8004502	509325	0.968	0.998	0.491
data/786 n2/786n2_Ca1uM4h_samp_220x 00016173 647.LMD	40	35970160	9975255	25994905	9546943	428311	0.974	1.000	0.408
data/786 n3/786n3_Ca1uM4h_samp_20x 00016176 650.LMD	40	4062670	1469687	2591281	1169278	300409	0.751	0.915	0.112
data/786 n3/786n3_Ca1uM4h_samp_60x 00016179 653.LMD	40	10569421	3351731	7215632	3078162	273569	0.926	0.992	0.186
data/786 n3/786n3_Ca1uM4h_samp_100x 00016182 656.LMD	40	17461437	5280604	12180834	4909747	370857	0.951	0.996	0.366
data/786 n3/786n3_Ca1uM4h_samp_140x 00016185 659.LMD	40	22786964	6953530	15833434	6638805	314725	0.968	0.990	0.490
data/786 n3/786n3_Ca1uM4h_samp_180x 00016188 662.LMD	40	29787846	9162140	20625706	8554719	607421	0.953	0.974	0.659
data/786 n3/786n3_Ca1uM4h_samp_220x 00016191 665.LMD	40	34482740	10517808	23964932	9996438	521370	0.981	0.999	0.638
data/VHL n1/VHLn1_Ca1uM4h_samp_20x 00016196 670.LMD	40	3764636	1343787	2420550	1207288	136499	0.758	0.816	0.242
data/VHL n1/VHLn1_Ca1uM4h_samp_60x 00016199 673.LMD	40	11127761	3396716	7730149	3243582	153134	0.922	0.941	0.503
data/VHL n1/VHLn1_Ca1uM4h_samp_100x 00016202 676.LMD	40	16076159	4574052	11502107	4486755	87297	0.973	0.983	0.446
data/VHL n1/VHLn1_Ca1uM4h_samp_140x 00016205 679.LMD	40	21845800	6470368	15373418	6254891	215478	0.960	0.972	0.610
data/VHL n1/VHLn1_Ca1uM4h_samp_180x 00016208 682.LMD	40	29432206	8648154	20784052	8419175	228979	0.979	0.989	0.613
data/VHL n1/VHLn1_Ca1uM4h_samp_220x 00016211 685.LMD	40	34067278	10003058	24064220	9715596	287462	0.970	0.975	0.809
data/VHL n2/VHLn2_Ca1uM4h_samp_20x 00016215 689.LMD	40	4521390	1430147	3090909	1329211	100936	0.872	0.918	0.261
data/VHL n2/VHLn2_Ca1uM4h_samp_60x 00016218 692.LMD	40	10352958	3146600	7206359	3025611	120989	0.950	0.969	0.486
data/VHL n2/VHLn2_Ca1uM4h_samp_100x 00016221 695.LMD	40	17012003	4959989	12052015	4821378	138611	0.979	0.988	0.668
data/VHL n2/VHLn2_Ca1uM4h_samp_140x 00016224 698.LMD	40	25318812	6744554	18572277	6596040	148515	0.991	0.998	0.688
data/VHL n2/VHLn2_Ca1uM4h_samp_180x 00016227 701.LMD	40	28675493	8423669	20249392	8180492	243178	0.993	1.000	0.764
data/VHL n2/VHLn2_Ca1uM4h_samp_220x 00016230 704.LMD	40	36200343	11616176	24584167	11015719	600457	0.966	0.972	0.850
data/VHL n3/VHLn3_Ca1uM4h_samp_20x 00016234 708.LMD	40	4346858	1325198	3021660	1208969	116229	0.796	0.861	0.119
data/VHL n3/VHLn3_Ca1uM4h_samp_60x 00016237 711.LMD	40	10775625	3140625	7635000	3022500	118125	0.944	0.973	0.197
data/VHL n3/VHLn3_Ca1uM4h_samp_100x 00016240 714.LMD	40	18156516	5036018	13120498	4898494	137525	0.968	0.985	0.355
data/VHL n3/VHLn3_Ca1uM4h_samp_140x 00016243 717.LMD	40	23569264	6718615	16850649	6608225	110390	0.990	1.000	0.379
data/VHL n3/VHLn3_Ca1uM4h_samp_180x 00016246 720.LMD	40	32475097	8764230	23710867	8639069	125162	0.992	1.000	0.421
data/VHL n3/VHLn3_Ca1uM4h_samp_220x 00016249 723.LMD	40	37375427	10430034	26945392	10245734	184300	0.991	1.000	0.535

```
dataset1 <- normalizedExperiment

dataset1$cd24posTp <- NULL
dataset1$cd24midTp <- NULL
dataset1$cd24hiTp <- NULL

dataset1$cd24posProp <- dataset1$CD24pos / dataset1$Events
dataset1$cd24midProp <- dataset1$CD24mid / dataset1$Events
dataset1$cd24hiProp <- dataset1$CD24hi / dataset1$Events

interestingColumns <- c(
  "cmVolume",
  "Events",
  "CD24pos",
  "CD24neg",
  "CD24mid",
  "CD24hi",
  "cd24posProp",
  "cd24midProp",
  "cd24hiProp"
)

printDataFrame(
  dataframe=dataset1[, interestingColumns],
  caption="Raw gate event counts, exhibiting swarm effect",
  fontsize=5
)
```

Table 10: Raw gate event counts, exhibiting swarm effect

	cmVolume	Events	CD24pos	CD24neg	CD24mid	CD24hi	cd24posProp	cd24midProp	cd24hiProp
data/786 n1/786n1_Ca1uM4h_samp_20x 00016119 613.LMD	40	4915608	2509286	2397198	1460085	1049202	0.5104732	0.2970304	0.2134430
data/786 n1/786n1_Ca1uM4h_samp_60x 00016122 616.LMD	40	11296460	3808850	7484956	3168142	640708	0.3371720	0.2804544	0.0567176
data/786 n1/786n1_Ca1uM4h_samp_100x 00016125 619.LMD	40	17279778	5588235	11687152	4906351	681885	0.3233974	0.2839360	0.0394614
data/786 n1/786n1_Ca1uM4h_samp_140x 00016128 622.LMD	40	22450108	6493221	15954989	5976952	516269	0.2892289	0.2662327	0.0229963
data/786 n1/786n1_Ca1uM4h_samp_180x 00016131 625.LMD	40	28569482	8205450	20364033	7727248	478202	0.2872103	0.2704721	0.0167382
data/786 n1/786n1_Ca1uM4h_samp_220x 00016134 628.LMD	40	34946083	9928110	25014883	9390621	537489	0.2840979	0.2687174	0.0153805
data/786 n2/786n2_Ca1uM4h_samp_20x 00016158 632.LMD	40	4692786	1857439	2833621	1542975	314463	0.3958073	0.3287972	0.0670099
data/786 n2/786n2_Ca1uM4h_samp_60x 00016161 635.LMD	40	10944777	3409427	7535350	3148786	260640	0.3115118	0.2876976	0.0238141
data/786 n2/786n2_Ca1uM4h_samp_100x 00016164 638.LMD	40	18144154	5635641	12508513	5209989	425653	0.3106037	0.2871442	0.0234595
data/786 n2/786n2_Ca1uM4h_samp_140x 00016167 641.LMD	40	23455190	7038362	16416828	6499033	539329	0.3000770	0.2770829	0.0229940
data/786 n2/786n2_Ca1uM4h_samp_180x 00016170 644.LMD	40	28976527	8513826	20462701	8004502	509325	0.2938180	0.2762409	0.0175772
data/786 n2/786n2_Ca1uM4h_samp_220x 00016173 647.LMD	40	35970160	9975255	25994905	9546943	428311	0.2773203	0.2654129	0.0119074
data/786 n3/786n3_Ca1uM4h_samp_20x 00016176 650.LMD	40	4062670	1469687	2591281	1169278	300409	0.3617540	0.2878102	0.0739437
data/786 n3/786n3_Ca1uM4h_samp_60x 00016179 653.LMD	40	10569421	3351731	7215632	3078162	273569	0.3171159	0.2912328	0.0258831
data/786 n3/786n3_Ca1uM4h_samp_100x 00016182 656.LMD	40	17461437	5280604	12180834	4909747	370857	0.3024152	0.2811766	0.0212386
data/786 n3/786n3_Ca1uM4h_samp_140x 00016185 659.LMD	40	22786964	6953530	15833434	6638805	314725	0.3051539	0.2913422	0.0138116
data/786 n3/786n3_Ca1uM4h_samp_180x 00016188 662.LMD	40	29787846	9162140	20625706	8554719	607421	0.3075798	0.2871882	0.0203916
data/786 n3/786n3_Ca1uM4h_samp_220x 00016191 665.LMD	40	34482740	10517808	23964932	9996438	521370	0.3050166	0.2898969	0.0151197
data/VHL n1/VHLn1_Ca1uM4h_samp_20x 00016196 670.LMD	40	3764636	1343787	2420550	1207288	136499	0.3569500	0.3206918	0.0362582
data/VHL n1/VHLn1_Ca1uM4h_samp_60x 00016199 673.LMD	40	11127761	3396716	7730149	3243582	153134	0.3052470	0.2914856	0.0137614
data/VHL n1/VHLn1_Ca1uM4h_samp_100x 00016202 676.LMD	40	16076159	4574052	11502107	4486755	87297	0.2845239	0.2790937	0.0054302
data/VHL n1/VHLn1_Ca1uM4h_samp_140x 00016205 679.LMD	40	21845800	6470368	15373418	6254891	215478	0.2961836	0.2863201	0.0098636
data/VHL n1/VHLn1_Ca1uM4h_samp_180x 00016208 682.LMD	40	29432206	8648154	20784052	8419175	228979	0.2938330	0.2860531	0.0077799
data/VHL n1/VHLn1_Ca1uM4h_samp_220x 00016211 685.LMD	40	34067278	10003058	24064220	9715596	287462	0.2936266	0.2851885	0.0084381
data/VHL n2/VHLn2_Ca1uM4h_samp_20x 00016215 689.LMD	40	4521390	1430147	3090909	1329211	100936	0.3163069	0.2939828	0.0223241
data/VHL n2/VHLn2_Ca1uM4h_samp_60x 00016218 692.LMD	40	10352958	3146600	7206359	3025611	120989	0.3039325	0.2922460	0.0116864
data/VHL n2/VHLn2_Ca1uM4h_samp_100x 00016221 695.LMD	40	17012003	4959989	12052015	4821378	138611	0.2915582	0.2834104	0.0081478
data/VHL n2/VHLn2_Ca1uM4h_samp_140x 00016224 698.LMD	40	25318812	6744554	18572277	6596040	148515	0.2663851	0.2605193	0.0058658
data/VHL n2/VHLn2_Ca1uM4h_samp_180x 00016227 701.LMD	40	28675493	8423669	20249392	8180492	243178	0.2937585	0.2852782	0.0084803
data/VHL n2/VHLn2_Ca1uM4h_samp_220x 00016230 704.LMD	40	36200343	11616176	24584167	11015719	600457	0.3208858	0.3042987	0.0165871
data/VHL n3/VHLn3_Ca1uM4h_samp_20x 00016234 708.LMD	40	4346858	1325198	3021660	1208969	116229	0.3048634	0.2781248	0.0267386
data/VHL n3/VHLn3_Ca1uM4h_samp_60x 00016237 711.LMD	40	10775625	3140625	7635000	3022500	118125	0.2914564	0.2804942	0.0109622
data/VHL n3/VHLn3_Ca1uM4h_samp_100x 00016240 714.LMD	40	18156516	5036018	13120498	4898494	137525	0.2773670	0.2697926	0.0075744
data/VHL n3/VHLn3_Ca1uM4h_samp_140x 00016243 717.LMD	40	23569264	6718615	16850649	6608225	110390	0.2850583	0.2803747	0.0046836

Table 10: Raw gate event counts, exhibiting swarm effect *(continued)*

	cmVolume	Events	CD24pos	CD24neg	CD24mid	CD24hi	cd24posProp	cd24midProp	cd24hiProp
data/VHL n3/VHLn3_Ca1uM4h_samp_180x 00016246 720.LMD	40	32475097	8764230	23710867	8639069	125162	0.2698754	0.2660213	0.0038541
data/VHL n3/VHLn3_Ca1uM4h_samp_220x 00016249 723.LMD	40	37375427	10430034	26945392	10245734	184300	0.2790613	0.2741302	0.0049310

4.3.1.1 Dataset 1 - Raw gate counts

```
dataset2 <- normalizedExperiment

dataset2$CD24mid <- dataset2$CD24mid * dataset2$cd24midTp
dataset2$CD24hi  <- dataset2$CD24hi  * dataset2$cd24hiTp
dataset2$CD24pos <- dataset2$CD24mid + dataset2$CD24hi

dataset2$cd24posProp <- dataset2$CD24pos / dataset2$Events
dataset2$cd24midProp <- dataset2$CD24mid / dataset2$Events
dataset2$cd24hiProp  <- dataset2$CD24hi  / dataset2$Events

dataset2$cd24posTp <- NULL
dataset2$cd24midTp <- NULL
dataset2$cd24hiTp  <- NULL

printDataFrame(
  dataframe=dataset2[, interestingColumns],
  caption="Swarm-deconvoluted event counts",
  fontsize=5
)
```

Table 11: Swarm-deconvoluted event counts

	cmVolume	Events	CD24pos	CD24neg	CD24mid	CD24hi	cd24posProp	cd24midProp	cd24hiProp
data/786 n1/786n1_Ca1uM4h_samp_20x 00016119 613.LMD	40	4915608	1447200	2397198	1160768	286432.15	0.2944091	0.2361392	0.0582699
data/786 n1/786n1_Ca1uM4h_samp_60x 00016122 616.LMD	40	11296460	3033473	7484956	2949540	83932.75	0.2685331	0.2611031	0.0074300
data/786 n1/786n1_Ca1uM4h_samp_100x 00016125 619.LMD	40	17279778	4803947	11687152	4675753	128194.38	0.2780098	0.2705910	0.0074188
data/786 n1/786n1_Ca1uM4h_samp_140x 00016128 622.LMD	40	22450108	5880981	15954989	5791666	89314.54	0.2619578	0.2579794	0.0039784
data/786 n1/786n1_Ca1uM4h_samp_180x 00016131 625.LMD	40	28569482	7655758	20364033	7557249	98509.61	0.2679698	0.2645217	0.0034481
data/786 n1/786n1_Ca1uM4h_samp_220x 00016134 628.LMD	40	34946083	9461409	25014883	9324887	136522.21	0.2707430	0.2668364	0.0039067
data/786 n2/786n2_Ca1uM4h_samp_20x 00016158 632.LMD	40	4692786	1294187	2833621	1231294	62892.60	0.2757822	0.2623802	0.0134020
data/786 n2/786n2_Ca1uM4h_samp_60x 00016161 635.LMD	40	10944777	3048990	7535350	2997644	51346.08	0.2785795	0.2738881	0.0046914
data/786 n2/786n2_Ca1uM4h_samp_100x 00016164 638.LMD	40	18144154	5226249	12508513	5105789	120459.80	0.2880404	0.2814013	0.0066390
data/786 n2/786n2_Ca1uM4h_samp_140x 00016167 641.LMD	40	23455190	6683779	16416828	6440542	243237.38	0.2849595	0.2745892	0.0103703
data/786 n2/786n2_Ca1uM4h_samp_180x 00016170 644.LMD	40	28976527	8238572	20462701	7988493	250078.58	0.2843188	0.2756884	0.0086304
data/786 n2/786n2_Ca1uM4h_samp_220x 00016173 647.LMD	40	35970160	9721694	25994905	9546943	174750.89	0.2702711	0.2654129	0.0048582
data/786 n3/786n3_Ca1uM4h_samp_20x 00016176 650.LMD	40	4062670	1103535	2591281	1069889	33645.81	0.2716281	0.2633464	0.0082817
data/786 n3/786n3_Ca1uM4h_samp_60x 00016179 653.LMD	40	10569421	3104421	7215632	3053537	50883.83	0.2937172	0.2889029	0.0048142
data/786 n3/786n3_Ca1uM4h_samp_100x 00016182 656.LMD	40	17461437	5025842	12180834	4890108	135733.66	0.2878252	0.2800519	0.0077733
data/786 n3/786n3_Ca1uM4h_samp_140x 00016185 659.LMD	40	22786964	6726632	15833434	6572417	154215.25	0.2951965	0.2884288	0.0067677
data/786 n3/786n3_Ca1uM4h_samp_180x 00016188 662.LMD	40	29787846	8732587	20625706	8332296	400290.44	0.2931594	0.2797213	0.0134380
data/786 n3/786n3_Ca1uM4h_samp_220x 00016191 665.LMD	40	34482740	10319076	23964932	9986442	332634.06	0.2992534	0.2896070	0.0096464
data/VHL n1/VHLn1_Ca1uM4h_samp_20x 00016196 670.LMD	40	3764636	1018180	2420550	985147	33032.76	0.2704590	0.2616845	0.0087745
data/VHL n1/VHLn1_Ca1uM4h_samp_60x 00016199 673.LMD	40	11127761	3129237	7730149	3052211	77026.40	0.2812099	0.2742879	0.0069220
data/VHL n1/VHLn1_Ca1uM4h_samp_100x 00016202 676.LMD	40	16076159	4449415	11502107	4410480	38934.46	0.2767710	0.2743491	0.0024219
data/VHL n1/VHLn1_Ca1uM4h_samp_140x 00016205 679.LMD	40	21845800	6211196	15373418	6079754	131441.58	0.2843199	0.2783031	0.0060168
data/VHL n1/VHLn1_Ca1uM4h_samp_180x 00016208 682.LMD	40	29432206	8466928	20784052	8326564	140364.13	0.2876756	0.2829066	0.0047691
data/VHL n1/VHLn1_Ca1uM4h_samp_220x 00016211 685.LMD	40	34067278	9705263	24064220	9472706	232556.76	0.2848852	0.2780588	0.0068264
data/VHL n2/VHLn2_Ca1uM4h_samp_20x 00016215 689.LMD	40	4521390	1246560	3090909	1220216	26344.30	0.2757028	0.2698762	0.0058266
data/VHL n2/VHLn2_Ca1uM4h_samp_60x 00016218 692.LMD	40	10352958	2990618	7206359	2931817	58800.65	0.2888660	0.2831864	0.0056796
data/VHL n2/VHLn2_Ca1uM4h_samp_100x 00016221 695.LMD	40	17012003	4856114	12052015	4763521	92592.15	0.2854522	0.2800094	0.0054428
data/VHL n2/VHLn2_Ca1uM4h_samp_140x 00016224 698.LMD	40	25318812	6685026	18572277	6582848	102178.32	0.2640340	0.2599983	0.0040357
data/VHL n2/VHLn2_Ca1uM4h_samp_180x 00016227 701.LMD	40	28675493	8366280	20249392	8180492	185787.99	0.2917571	0.2852782	0.0064790
data/VHL n2/VHLn2_Ca1uM4h_samp_220x 00016230 704.LMD	40	36200343	11217667	24584167	10707279	510388.45	0.3098774	0.2957784	0.0140990
data/VHL n3/VHLn3_Ca1uM4h_samp_20x 00016234 708.LMD	40	4346858	1054754	3021660	1040922	13831.25	0.2426473	0.2394655	0.0031819
data/VHL n3/VHLn3_Ca1uM4h_samp_60x 00016237 711.LMD	40	10775625	2964163	7635000	2940892	23270.62	0.2750804	0.2729208	0.0021596
data/VHL n3/VHLn3_Ca1uM4h_samp_100x 00016240 714.LMD	40	18156516	4873838	13120498	4825017	48821.38	0.2684346	0.2657457	0.0026889
data/VHL n3/VHLn3_Ca1uM4h_samp_140x 00016243 717.LMD	40	23569264	6650063	16850649	6608225	41837.81	0.2821498	0.2803747	0.0017751
data/VHL n3/VHLn3_Ca1uM4h_samp_180x 00016246 720.LMD	40	32475097	8691762	23710867	8639069	52693.20	0.2676439	0.2660213	0.0016226
data/VHL n3/VHLn3_Ca1uM4h_samp_220x 00016249 723.LMD	40	37375427	10344334	26945392	10245734	98600.50	0.2767683	0.2741302	0.0026381

4.3.1.2 Dataset 2 - Swarm-deconvoluted

```r
# This optimization function will figure out how much noise is in the population
# (ie.: for blank subtraction)
subtractNoise <- function(noiseCeiling, events, tpRate, beads, dilution) {
  normalizedToXBeads <- 5000
  multiplier <- (normalizedToXBeads * dilution * 10) / beads
  BiocGenerics::var(((events * tpRate) - noiseCeiling) * multiplier)
}

cd24hiNoiseCeiling <- optimize(
  f=subtractNoise,
  interval=c(1, 5000),
  events=experiment$CD24hi,
  tpRate=experiment$cd24hiTp,
  beads=experiment$Beads,
  dilution=experiment$Dilution
)$minimum

cd24midNoiseCeiling <- optimize(
  f=subtractNoise,
  interval=c(100, 10000),
  events=experiment$CD24mid,
  tpRate=experiment$cd24midTp,
  beads=experiment$Beads,
  dilution=experiment$Dilution
)$minimum

cd24negNoiseCeiling <- optimize(
  f=subtractNoise,
  interval=c(100, 9000),
  events=experiment$CD24neg,
  tpRate=1,
  beads=experiment$Beads,
  dilution=experiment$Dilution
)$minimum

# Make sure the noiseCeiling is not set above the measured signal
if (any(experiment$CD24hi < cd24hiNoiseCeiling)) {
  cd24hiNoiseCeiling <- min(experiment$CD24hi)
}
if (any(experiment$CD24mid < cd24midNoiseCeiling)) {
  cd24midNoiseCeiling <- min(experiment$CD24mid)
}
if (any(experiment$CD24neg < cd24negNoiseCeiling)) {
  cd24negNoiseCeiling <- min(experiment$CD24neg)
}

list(
  "cd24hiNoiseCeiling: "  = cd24hiNoiseCeiling,
  "cd24midNoiseCeiling: " = cd24midNoiseCeiling,
  "cd24negNoiseCeiling: " = cd24negNoiseCeiling
)
```

4.3.1.3 Dataset 3 - Swarm-deconvoluted and normalized

```
## $`cd24hiNoiseCeiling: `
## [1] 43
##
## $`cd24midNoiseCeiling: `
## [1] 2385
##
## $`cd24negNoiseCeiling: `
## [1] 6494
```

```r
# Let's make a separate dataframe for these data
dataset3 <- experiment

# Normalize with noise reduction
factor <- (experiment$Dilution * normalizedToXBeads * 10) / experiment$Beads

dataset3$CD24mid <- (
  (experiment$CD24mid * experiment$cd24midTp) - cd24midNoiseCeiling) * factor

dataset3$CD24hi <- (
  (experiment$CD24hi * experiment$cd24hiTp) - cd24hiNoiseCeiling) * factor

dataset3$CD24neg <- (experiment$CD24neg - cd24negNoiseCeiling) * factor

# Replace negative values by zero
dataset3$CD24neg[dataset3$CD24neg < 0] <- 0
dataset3$CD24mid[dataset3$CD24mid < 0] <- 0
dataset3$CD24hi[dataset3$CD24hi < 0]   <- 0

# To be or not to be, that is the question
dataset3$CD24neg <- round(dataset3$CD24neg, digits=0)
dataset3$CD24mid <- round(dataset3$CD24mid, digits=0)
dataset3$CD24hi  <- round(dataset3$CD24hi, digits=0)

# Compute remaining columns
dataset3$CD24pos <- dataset3$CD24mid + dataset3$CD24hi
dataset3$Events  <- dataset3$CD24neg + dataset3$CD24pos

dataset3$cd24posProp <- dataset3$CD24pos / dataset3$Events
dataset3$cd24midProp <- dataset3$CD24mid / dataset3$Events
dataset3$cd24hiProp  <- dataset3$CD24hi  / dataset3$Events

# We don't need these columns any longer
dataset3$Beads     <- NULL
dataset3$cd24posTp <- NULL
dataset3$cd24midTp <- NULL
dataset3$cd24hiTp  <- NULL
dataset3$cmVolume  <- 40

printDataFrame(
  dataframe=dataset3[, interestingColumns],
  caption="Swarm-deconvoluted and denoised event counts",
  fontsize=5
)
```

Table 12: Swarm-deconvoluted and denoised event counts

	cmVolume	Events	CD24pos	CD24neg	CD24mid	CD24hi	cd24posProp	cd24midProp	cd24hiProp
data/786 n1/786n1_Ca1uM4h_samp_20x 00016119 613.LMD	40	937261	656062	281199	383641	272421	0.6999779	0.4093214	0.2906565
data/786 n1/786n1_Ca1uM4h_samp_60x 00016122 616.LMD	40	2622853	884800	1738053	838920	45880	0.3373426	0.3198502	0.0174924
data/786 n1/786n1_Ca1uM4h_samp_100x 00016125 619.LMD	40	3435788	1251123	2184665	1185849	65274	0.3641444	0.3451461	0.0189983
data/786 n1/786n1_Ca1uM4h_samp_140x 00016128 622.LMD	40	4901588	1272521	3629067	1264823	7698	0.2596140	0.2580435	0.0015705
data/786 n1/786n1_Ca1uM4h_samp_180x 00016131 625.LMD	40	6147167	1708475	4438692	1708475	0	0.2779288	0.2779288	0.0000000
data/786 n1/786n1_Ca1uM4h_samp_220x 00016134 628.LMD	40	6916056	1961268	4954788	1957574	3694	0.2835819	0.2830477	0.0005341
data/786 n2/786n2_Ca1uM4h_samp_20x 00016158 632.LMD	40	1048071	456079	591992	408029	48050	0.4351604	0.3893143	0.0458461
data/786 n2/786n2_Ca1uM4h_samp_60x 00016161 635.LMD	40	1169639	486909	682730	480937	5972	0.4162900	0.4111841	0.0051058
data/786 n2/786n2_Ca1uM4h_samp_100x 00016164 638.LMD	40	856215	632984	223231	593871	39113	0.7392816	0.6936003	0.0456813
data/786 n2/786n2_Ca1uM4h_samp_140x 00016167 641.LMD	40	2967144	1204733	1762411	1058530	146203	0.4060244	0.3567505	0.0492740

Table 12: Swarm-deconvoluted and denoised event counts *(continued)*

	cmVolume	Events	CD24pos	CD24neg	CD24mid	CD24hi	cd24posProp	cd24midProp	cd24hiProp
data/786 n2/786n2_Ca1uM4h_samp_180x 00016170 644.LMD	40	2881979	1212204	1669775	1086563	125641	0.4206151	0.3770197	0.0435954
data/786 n2/786n2_Ca1uM4h_samp_220x 00016173 647.LMD	40	2626	2626	0	0	2626	1.0000000	0.0000000	1.0000000
data/786 n3/786n3_Ca1uM4h_samp_20x 00016176 650.LMD	40	655988	276560	379428	257560	19000	0.4215931	0.3926291	0.0289639
data/786 n3/786n3_Ca1uM4h_samp_60x 00016179 653.LMD	40	1144188	607335	536853	600675	6660	0.5308000	0.5249793	0.0058207
data/786 n3/786n3_Ca1uM4h_samp_100x 00016182 656.LMD	40	2566045	1041595	1524450	976423	65172	0.4059145	0.3805167	0.0253978
data/786 n3/786n3_Ca1uM4h_samp_140x 00016185 659.LMD	40	3714563	1598087	2116476	1534698	63389	0.4302221	0.4131571	0.0170650
data/786 n3/786n3_Ca1uM4h_samp_180x 00016188 662.LMD	40	7767004	2856813	4910191	2560583	296230	0.3678140	0.3296745	0.0381395
data/786 n3/786n3_Ca1uM4h_samp_220x 00016191 665.LMD	40	7395789	3001816	4393973	2798771	203045	0.4058818	0.3784276	0.0274541
data/VHL n1/VHLn1_Ca1uM4h_samp_20x 00016196 670.LMD	40	773854	292970	480884	272781	20189	0.3785856	0.3524967	0.0260889
data/VHL n1/VHLn1_Ca1uM4h_samp_60x 00016199 673.LMD	40	2869536	954909	1914627	916390	38519	0.3327747	0.3193513	0.0134234
data/VHL n1/VHLn1_Ca1uM4h_samp_100x 00016202 676.LMD	40	2548650	820775	1727875	820775	0	0.3220430	0.3220430	0.0000000
data/VHL n1/VHLn1_Ca1uM4h_samp_140x 00016205 679.LMD	40	3617409	1321667	2295742	1276819	44848	0.3653629	0.3529651	0.0123978
data/VHL n1/VHLn1_Ca1uM4h_samp_180x 00016208 682.LMD	40	4336926	1686910	2650016	1666620	20290	0.3889644	0.3842860	0.0046784
data/VHL n1/VHLn1_Ca1uM4h_samp_220x 00016211 685.LMD	40	6485079	2280186	4204893	2179128	101058	0.3516050	0.3360218	0.0155832
data/VHL n2/VHLn2_Ca1uM4h_samp_20x 00016215 689.LMD	40	1355518	435063	920455	423090	11973	0.3209570	0.3121242	0.0088328
data/VHL n2/VHLn2_Ca1uM4h_samp_60x 00016218 692.LMD	40	2317671	846373	1471298	825547	20826	0.3651825	0.3561968	0.0089857
data/VHL n2/VHLn2_Ca1uM4h_samp_100x 00016221 695.LMD	40	4158771	1386550	2772221	1355404	31146	0.3334038	0.3259146	0.0074892
data/VHL n2/VHLn2_Ca1uM4h_samp_140x 00016224 698.LMD	40	7589976	1877105	5712871	1860075	17030	0.2473137	0.2450699	0.0022437
data/VHL n2/VHLn2_Ca1uM4h_samp_180x 00016227 701.LMD	40	6919373	2461929	4457444	2380708	81221	0.3558023	0.3440641	0.0117382
data/VHL n2/VHLn2_Ca1uM4h_samp_220x 00016230 704.LMD	40	7753249	3584629	4168620	3209422	375207	0.4623389	0.4139454	0.0483935
data/VHL n3/VHLn3_Ca1uM4h_samp_20x 00016234 708.LMD	40	1354632	314058	1040574	313344	714	0.2318401	0.2313130	0.0005271
data/VHL n3/VHLn3_Ca1uM4h_samp_60x 00016237 711.LMD	40	2251830	704955	1546875	704955	0	0.3130587	0.3130587	0.0000000
data/VHL n3/VHLn3_Ca1uM4h_samp_100x 00016240 714.LMD	40	3408841	920301	2488540	920301	0	0.2699748	0.2699748	0.0000000
data/VHL n3/VHLn3_Ca1uM4h_samp_140x 00016243 717.LMD	40	4240259	1445887	2794372	1445887	0	0.3409903	0.3409903	0.0000000
data/VHL n3/VHLn3_Ca1uM4h_samp_180x 00016246 720.LMD	40	6505498	1696960	4808538	1696960	0	0.2608501	0.2608501	0.0000000
data/VHL n3/VHLn3_Ca1uM4h_samp_220x 00016249 723.LMD	40	6887372	2105802	4781570	2105802	0	0.3057483	0.3057483	0.0000000

4.3.2 Computing summary data

```
# Compute summary statistics for each biological triplicate
dataset1Summary <- dataset1 %>%
  dplyr::group_by(CellLine, Dilution) %>%
    dplyr::summarise(
      n          = dplyr::n(),
      eventsAvg  = BiocGenerics::mean(Events),
      eventsSem  = BiocGenerics::sd(Events) / dplyr::n(),
      cd24posAvg = BiocGenerics::mean(CD24pos),
      cd24posSem = BiocGenerics::sd(CD24pos) / dplyr::n(),
      cd24negAvg = BiocGenerics::mean(CD24neg),
      cd24negSem = BiocGenerics::sd(CD24neg) / dplyr::n(),
      cd24hiAvg  = BiocGenerics::mean(CD24hi),
      cd24hiSem  = BiocGenerics::sd(CD24hi) / dplyr::n(),
      cd24midAvg = BiocGenerics::mean(CD24mid),
      cd24midSem = BiocGenerics::sd(CD24mid) / dplyr::n(),
      cd24posPropAvg = BiocGenerics::mean(CD24pos/Events),
      cd24posPropSem = BiocGenerics::sd(CD24pos/Events) / dplyr::n(),
      cd24negPropAvg = BiocGenerics::mean(CD24neg/Events),
      cd24negPropSem = BiocGenerics::sd(CD24neg/Events) / dplyr::n(),
      cd24midPropAvg = BiocGenerics::mean(CD24mid/Events),
      cd24midPropSem = BiocGenerics::sd(CD24mid/Events) / dplyr::n(),
      cd24hiPropAvg  = BiocGenerics::mean(CD24hi/Events),
      cd24hiPropSem  = BiocGenerics::sd(CD24hi/Events) / dplyr::n()
    )
```

4.3.2.1 Dataset 1 - Raw gate counts

```
## `summarise()` regrouping output by 'CellLine' (override with `.groups` argument)
```

```
interestingColumns <- function(metric) {
  c(
    "CellLine",
    "Dilution",
    "n",
    paste0("events", metric),
    paste0("cd24pos", metric),
    paste0("cd24neg", metric),
    paste0("cd24mid", metric),
    paste0("cd24hi", metric),
    paste0("cd24posProp", metric),
    paste0("cd24negProp", metric),
    paste0("cd24midProp", metric),
    paste0("cd24hiProp", metric)
  )
}

printDataFrame(
  dataframe=dataset1Summary[, interestingColumns("Avg")],
  caption="Normalized experiment summary: calculated means",
  fontsize=5
)
```

Table 13: Normalized experiment summary: calculated means

CellLine	Dilution	n	eventsAvg	cd24posAvg	cd24negAvg	cd24midAvg	cd24hiAvg	cd24posPropAvg	cd24negPropAvg	cd24midPropAvg	cd24hiPropAvg
786-O	20	3	4557021	1945471	2607367	1390779	554691.3	0.4226781	0.5764409	0.3045460	0.1181322
786-O	60	3	10936886	3523336	7411979	3131697	391639.0	0.3219332	0.6779236	0.2864616	0.0354716
786-O	100	3	17628456	5501493	12125500	5008696	492798.3	0.3121388	0.6877766	0.2840856	0.0280532
786-O	140	3	22897421	6828371	16068417	6371597	456774.3	0.2981533	0.7018186	0.2782193	0.0199340
786-O	180	3	29111285	8627139	20484147	8095490	531649.3	0.2962027	0.7037973	0.2779671	0.0182356
786-O	220	3	35132994	10140391	24991573	9644667	495723.3	0.2888116	0.7111589	0.2746757	0.0141359
VHL	20	3	4210961	1366377	2844373	1248489	117888.0	0.3260401	0.6739088	0.2975998	0.0284403
VHL	60	3	10752115	3227980	7523836	3097231	130749.3	0.3002120	0.6997612	0.2880753	0.0121367
VHL	100	3	17081559	4856686	12224873	4735542	121144.3	0.2844830	0.7155170	0.2774322	0.0070508
VHL	140	3	23577959	6644512	16932115	6486385	158127.7	0.2825423	0.7174008	0.2757380	0.0068043
VHL	180	3	30194265	8612018	21581437	8412912	199106.3	0.2858223	0.7141494	0.2791176	0.0067048
VHL	220	3	35881016	10683089	25197926	10325683	357406.3	0.2978579	0.7021421	0.2878725	0.0099854

```
printDataFrame(
  dataframe=dataset1Summary[, interestingColumns("Sem")],
  caption="Normalized experiment summary: calculated SEMs",
  fontsize=5
)
```

Table 14: Normalized experiment summary: calculated SEMs

CellLine	Dilution	n	eventsSem	cd24posSem	cd24negSem	cd24midSem	cd24hiSem	cd24posPropSem	cd24negPropSem	cd24midPropSem	cd24hiPropSem
786-O	20	3	147459.9	175120.18	72885.24	65417.31	142772.149	0.0259720	0.0262449	0.0071674	0.0275381
786-O	60	3	121194.6	82979.84	57299.50	15787.25	71932.310	0.0044971	0.0045298	0.0018315	0.0061429
786-O	100	3	151916.7	64252.89	137822.30	58111.14	55343.349	0.0035250	0.0035701	0.0009955	0.0033140
786-O	140	3	170521.4	97777.10	102597.69	116281.44	41185.826	0.0027116	0.0026962	0.0041978	0.0017674
786-O	180	3	206753.3	162769.01	44049.38	140390.78	22480.039	0.0034640	0.0034640	0.0028302	0.0006379
786-O	220	3	253707.3	109233.88	338395.74	104836.43	19644.826	0.0048124	0.0048076	0.0044283	0.0006448
VHL	20	3	132085.9	18667.61	122890.33	23304.02	5946.483	0.0091245	0.0091351	0.0071709	0.0023737
VHL	60	3	129312.1	48719.97	93009.60	42251.08	6479.503	0.0025370	0.0025459	0.0021921	0.0004843
VHL	100	3	347307.1	82567.63	274308.70	72959.71	9772.560	0.0023652	0.0023652	0.0023197	0.0004775
VHL	140	3	578840.8	50456.69	533661.59	66857.51	17733.128	0.0050192	0.0050192	0.0045036	0.0009048
VHL	180	3	670389.5	57710.80	621138.69	76450.88	21476.670	0.0046035	0.0046116	0.0037828	0.0008312
VHL	220	3	559011.0	278600.70	511839.10	217912.78	72238.663	0.0070770	0.0070770	0.0050874	0.0019934

```r
# Do the same with swarm-deconvoluted data
dataset2Summary <- dataset2 %>%
  dplyr::group_by(CellLine, Dilution) %>%
    dplyr::summarise(
      n          = dplyr::n(),
      eventsAvg  = BiocGenerics::mean(Events),
      eventsSem  = BiocGenerics::sd(Events) / dplyr::n(),
      cd24posAvg = BiocGenerics::mean(CD24pos),
      cd24posSem = BiocGenerics::sd(CD24pos) / dplyr::n(),
      cd24negAvg = BiocGenerics::mean(CD24neg),
      cd24negSem = BiocGenerics::sd(CD24neg) / dplyr::n(),
      cd24hiAvg  = BiocGenerics::mean(CD24hi),
      cd24hiSem  = BiocGenerics::sd(CD24hi) / dplyr::n(),
      cd24midAvg = BiocGenerics::mean(CD24mid),
      cd24midSem = BiocGenerics::sd(CD24mid) / dplyr::n(),
      cd24posPropAvg = BiocGenerics::mean(CD24pos/Events),
      cd24posPropSem = BiocGenerics::sd(CD24pos/Events) / dplyr::n(),
      cd24negPropAvg = BiocGenerics::mean(CD24neg/Events),
      cd24negPropSem = BiocGenerics::sd(CD24neg/Events) / dplyr::n(),
      cd24midPropAvg = BiocGenerics::mean(CD24mid/Events),
      cd24midPropSem = BiocGenerics::sd(CD24mid/Events) / dplyr::n(),
      cd24hiPropAvg  = BiocGenerics::mean(CD24hi/Events),
      cd24hiPropSem  = BiocGenerics::sd(CD24hi/Events) / dplyr::n()
    )
```

4.3.2.2 Dataset 2 - Swarm-deconvoluted

```
## `summarise()` regrouping output by 'CellLine' (override with `.groups` argument)
```

```
printDataFrame(
  dataframe=dataset2Summary[, interestingColumns("Avg")],
  caption="Normalized experiment summary (deconvoluted): calculated means",
  fontsize=5
)
```

Table 15: Normalized experiment summary (deconvoluted): calculated means

CellLine	Dilution	n	eventsAvg	cd24posAvg	cd24negAvg	cd24midAvg	cd24hiAvg	cd24posPropAvg	cd24negPropAvg	cd24midPropAvg	cd24hiPropAvg
786-O	20	3	4557021	1281641	2607367	1153984	127656.85	0.2806064	0.5764409	0.2539552	0.0266512
786-O	60	3	10936886	3062295	7411979	3000240	62054.22	0.2802766	0.6779236	0.2746314	0.0056452
786-O	100	3	17628456	5018679	12125500	4890550	128129.28	0.2846251	0.6877766	0.2773481	0.0072770
786-O	140	3	22897421	6430464	16068417	6268208	162255.72	0.2807046	0.7018186	0.2736658	0.0070388
786-O	180	3	29111285	8208972	20484147	7959346	249626.21	0.2818160	0.7037973	0.2733105	0.0085055
786-O	220	3	35132994	9834059	24991573	9619424	214635.72	0.2800892	0.7111589	0.2739521	0.0061371
VHL	20	3	4210961	1106498	2844373	1082095	24402.77	0.2629364	0.6739088	0.2570087	0.0059277
VHL	60	3	10752115	3028006	7523836	2974973	53032.56	0.2817188	0.6997612	0.2767984	0.0049204
VHL	100	3	17081559	4726455	12224873	4666339	60116.00	0.2768859	0.7155170	0.2733681	0.0035178
VHL	140	3	23577959	6515428	16932115	6423609	91819.24	0.2768345	0.7174008	0.2728920	0.0039425
VHL	180	3	30194265	8508323	21581437	8382042	126281.77	0.2823589	0.7141494	0.2780687	0.0042902
VHL	220	3	35881016	10422422	25197926	10141906	280515.24	0.2905103	0.7021421	0.2826558	0.0078545

```
printDataFrame(
  dataframe=dataset2Summary[, interestingColumns("Sem")],
  caption="Normalized experiment summary (deconvoluted): calculated SEMs",
  fontsize=5
)
```

Table 16: Normalized experiment summary (deconvoluted): calculated SEMs

CellLine	Dilution	n	eventsSem	cd24posSem	cd24negSem	cd24midSem	cd24hiSem	cd24posPropSem	cd24negPropSem	cd24midPropSem	cd24hiPropSem
786-O	20	3	147459.9	57391.81	72885.24	26971.97	46092.949	0.0040442	0.0262449	0.0051456	0.0091673
786-O	60	3	121194.6	12432.67	57299.50	17348.94	6316.257	0.0042258	0.0045298	0.0046383	0.0005156
786-O	100	3	151916.7	70414.05	137822.30	71672.90	2545.713	0.0019100	0.0035701	0.0019635	0.0001934
786-O	140	3	170521.4	158782.81	102597.69	139310.56	25758.596	0.0056743	0.0026962	0.0050819	0.0010682
786-O	180	3	206753.3	179674.72	44049.38	129448.37	50296.974	0.0042600	0.0034640	0.0026246	0.0016654
786-O	220	3	253707.3	146578.60	338395.74	112226.86	34653.947	0.0055328	0.0048076	0.0045254	0.0010254
VHL	20	3	132085.9	40889.39	122890.33	40941.31	3248.959	0.0059218	0.0091351	0.0052451	0.0009326
VHL	60	3	129312.1	29553.65	93009.60	22347.72	9112.714	0.0023023	0.0025459	0.0018581	0.0008234
VHL	100	3	347307.1	80029.32	274308.70	74567.92	9518.772	0.0028365	0.0023652	0.0023941	0.0005575
VHL	140	3	578840.8	88017.49	533661.59	99352.44	15230.422	0.0037129	0.0050192	0.0037381	0.0007075
VHL	180	3	670389.5	55547.62	621138.69	78089.37	22551.891	0.0043020	0.0046116	0.0035002	0.0008211
VHL	220	3	559011.0	253073.33	511839.10	207933.65	70013.761	0.0057521	0.0070770	0.0038443	0.0019331

```
# Compute summary statistics for each biological triplicate
dataset3Summary <- dataset3 %>%
  dplyr::group_by(CellLine, Dilution) %>%
    dplyr::summarise(
      n          = dplyr::n(),
      eventsAvg  = BiocGenerics::mean(Events),
      eventsSem  = BiocGenerics::sd(Events) / dplyr::n(),
      cd24posAvg = BiocGenerics::mean(CD24pos),
      cd24posSem = BiocGenerics::sd(CD24pos) / dplyr::n(),
      cd24negAvg = BiocGenerics::mean(CD24neg),
      cd24negSem = BiocGenerics::sd(CD24neg) / dplyr::n(),
      cd24hiAvg  = BiocGenerics::mean(CD24hi),
      cd24hiSem  = BiocGenerics::sd(CD24hi) / dplyr::n(),
      cd24midAvg = BiocGenerics::mean(CD24mid),
      cd24midSem = BiocGenerics::sd(CD24mid) / dplyr::n(),
      cd24posPropAvg = BiocGenerics::mean(CD24pos/Events),
      cd24posPropSem = BiocGenerics::sd(CD24pos/Events) / dplyr::n(),
      cd24negPropAvg = BiocGenerics::mean(CD24neg/Events),
      cd24negPropSem = BiocGenerics::sd(CD24neg/Events) / dplyr::n(),
      cd24midPropAvg = BiocGenerics::mean(CD24mid/Events),
      cd24midPropSem = BiocGenerics::sd(CD24mid/Events) / dplyr::n(),
      cd24hiPropAvg  = BiocGenerics::mean(CD24hi/Events),
      cd24hiPropSem  = BiocGenerics::sd(CD24hi/Events) / dplyr::n()
    )
```

4.3.2.3 Dataset 3 - Swarm-deconvoluted and normalized

```
## `summarise()` regrouping output by 'CellLine' (override with `.groups` argument)
```

```
printDataFrame(
  dataframe=dataset3Summary[, interestingColumns("Avg")],
  caption="Denoised and normalized experiment summary: calculated means",
  fontsize=5
)
```

Table 17: Denoised and normalized experiment summary: calculated means

CellLine	Dilution	n	eventsAvg	cd24posAvg	cd24negAvg	cd24midAvg	cd24hiAvg	cd24posPropAvg	cd24negPropAvg	cd24midPropAvg	cd24hiPropAvg
786-O	20	3	880440	462900.3	417539.7	349743.3	113157.00	0.5189105	0.4810895	0.3970883	0.1218222
786-O	60	3	1645560	659681.3	985878.7	640177.3	19504.00	0.4281442	0.5718558	0.4186712	0.0094730
786-O	100	3	2286016	975234.0	1310782.0	918714.3	56519.67	0.5031135	0.4968865	0.4730877	0.0300258
786-O	140	3	3861098	1358447.0	2502651.3	1286017.0	72430.00	0.3652868	0.6347132	0.3426504	0.0226365
786-O	180	3	5598717	1925830.7	3672886.0	1785207.0	140623.67	0.3554527	0.6445473	0.3282077	0.0272450
786-O	220	3	4771490	1655236.7	3116253.7	1585448.3	69788.33	0.5631545	0.4368455	0.2204918	0.3426628
VHL	20	3	1161335	347363.7	813971.0	336405.0	10958.67	0.3104609	0.6895391	0.2986447	0.0118163
VHL	60	3	2479679	835412.3	1644266.7	815630.7	19781.67	0.3370053	0.6629947	0.3295356	0.0074697
VHL	100	3	3372087	1042542.0	2329545.3	1032160.0	10382.00	0.3084739	0.6915261	0.3059774	0.0024964
VHL	140	3	5149215	1548219.7	3600995.0	1527593.7	20626.00	0.3178889	0.6821111	0.3130084	0.0048805
VHL	180	3	5920599	1948599.7	3971999.3	1914762.7	33837.00	0.3352056	0.6647944	0.3297334	0.0054722
VHL	220	3	7041900	2656872.3	4385027.7	2498117.3	158755.00	0.3732307	0.6267693	0.3519052	0.0213256

```
printDataFrame(
  dataframe=dataset3Summary[, interestingColumns("Sem")],
  caption="Denoised and normalized experiment summary: calculated SEMs",
  fontsize=5
)
```

Table 18: Denoised and normalized experiment summary: calculated SEMs

CellLine	Dilution	n	eventsSem	cd24posSem	cd24negSem	cd24midSem	cd24hiSem	cd24posPropSem	cd24negPropSem	cd24midPropSem	cd24hiPropSem
786-O	20	3	67374.36	63280.98	52954.33	26919.67	46229.791	0.0523186	0.0523186	0.0035744	0.0488194
786-O	60	3	282152.08	68015.04	218490.96	60743.80	7614.959	0.0324240	0.0324240	0.0342564	0.0023181
786-O	100	3	437462.56	104789.10	332673.74	100059.54	5024.901	0.0685304	0.0685304	0.0639289	0.0046435
786-O	140	3	325170.53	70094.62	330479.40	79596.82	23231.237	0.0307705	0.0307705	0.0261707	0.0081116
786-O	180	3	829421.86	281191.38	583563.58	246666.57	49560.752	0.0240473	0.0240473	0.0165206	0.0079173
786-O	220	3	1378972.49	507611.68	904427.75	478671.53	38468.298	0.1277431	0.1277431	0.0656056	0.1898100
VHL	20	3	111856.13	25559.42	98215.95	25920.94	3259.014	0.0246446	0.0246446	0.0205687	0.0043465
VHL	60	3	113075.75	41778.99	79056.24	35355.24	6426.908	0.0087727	0.0087727	0.0077676	0.0022796
VHL	100	3	268563.16	100682.39	180007.44	94775.39	5994.050	0.0112739	0.0112739	0.0104131	0.0014413
VHL	140	3	712193.20	97172.14	615284.32	100029.93	7546.406	0.0207743	0.0207743	0.0197134	0.0022020
VHL	180	3	462341.66	148194.88	386083.86	134601.85	14090.397	0.0221648	0.0221648	0.0209845	0.0019697
VHL	220	3	216017.74	269392.69	114631.44	205699.33	64714.571	0.0268347	0.0268347	0.0186067	0.0082342

4.3.3 Building sample graphs

```
# Let's generate plots to show the swarm effect

abcTemplate <- function(
  data,
  title,
  subtitle,
  measurement,
  scaleYLimits,
  scaleYLabels,
  legendPosition
) {
  measurement <- rlang::sym(measurement)

  ggplot2::ggplot(
    data=data,
    mapping=ggplot2::aes(
      x=Dilution,
      y=!!measurement,
      colour=CellLine,
      group=interaction(CellLine, FacetRow)
    )
  ) +
    ggplot2::labs(
      title=title,
      subtitle=subtitle,
      x=ggplot2::element_blank(),
      y=ggplot2::element_blank()) +
    ggplot2::geom_point() +
    ggplot2::geom_line() +
    ggplot2::scale_x_continuous(
      limits=c(20, 220),
      breaks=c(0, 20, 60, 100, 140, 180, 220),
      minor_breaks=c(40, 80, 120, 160, 200)) +
    ggplot2::scale_y_continuous(
      limits=scaleYLimits,
      labels=scaleYLabels) +
    ggplot2::theme(
      legend.position=legendPosition,
      legend.justification=c(1, 1),
      legend.title=ggplot2::element_blank())
}

plotA1 <- abcTemplate(
  data=dataset1,
  title="Proportion of CD24+ events",
  subtitle="Swarm effect",
  measurement="cd24posProp",
  scaleYLimits=c(0, 1),
  scaleYLabels=scales::number,
  legendPosition="none"
)
```

```
plotA2 <- abcTemplate(
  data=dataset1,
  title="Proportion of CD24mid events",
  subtitle="Swarm effect",
  measurement="cd24midProp",
  scaleYLimits=c(0, 1),
  scaleYLabels=scales::number,
  legendPosition="none"
)

plotA3 <- abcTemplate(
  data=dataset1,
  title="Proportion of CD24hi events",
  subtitle="Swarm effect",
  measurement="cd24hiProp",
  scaleYLimits=c(0, 1),
  scaleYLabels=scales::number,
  legendPosition=c(1, 1)
)

plotA4 <- abcTemplate(
  data=dataset1,
  title="Absolute count of CD24+ events",
  subtitle="Swarm effect",
  measurement="CD24pos",
  scaleYLimits=c(0, 1.2e7),
  scaleYLabels=scales::scientific,
  legendPosition="none"
)

plotA5 <- abcTemplate(
  data=dataset1,
  title="Absolute count of CD24mid events",
  subtitle="Swarm effect",
  measurement="CD24mid",
  scaleYLimits=c(0, 1.2e7),
  scaleYLabels=scales::scientific,
  legendPosition="none"
)

plotA6 <- abcTemplate(
  data=dataset1,
  title="Absolute count of CD24hi events",
  subtitle="Swarm effect",
  measurement="CD24hi",
  scaleYLimits=c(0, 1.2e6),
  scaleYLabels=scales::scientific,
  legendPosition=c(1, 1)
)
```

4.3.3.1 Dataset 1 - Raw gate counts

```r
# Let's generate plots for the swarm-deconvoluted data

plotB1 <- abcTemplate(
  data=dataset2,
  title="",
  subtitle="Swarm-deconvoluted",
  measurement="cd24posProp",
  scaleYLimits=c(0, 1),
  scaleYLabels=scales::number,
  legendPosition="none"
)

plotB2 <- abcTemplate(
  data=dataset2,
  title="",
  subtitle="Swarm-deconvoluted",
  measurement="cd24midProp",
  scaleYLimits=c(0, 1),
  scaleYLabels=scales::number,
  legendPosition="none"
)

plotB3 <- abcTemplate(
  data=dataset2,
  title="",
  subtitle="Swarm-deconvoluted",
  measurement="cd24hiProp",
  scaleYLimits=c(0, 1),
  scaleYLabels=scales::number,
  legendPosition="none"
)

plotB4 <- abcTemplate(
  data=dataset2,
  title="",
  subtitle="Swarm-deconvoluted",
  measurement="CD24pos",
  scaleYLimits=c(0, 1.2e7),
  scaleYLabels=scales::scientific,
  legendPosition="none"
)

plotB5 <- abcTemplate(
  data=dataset2,
  title="",
  subtitle="Swarm-deconvoluted",
  measurement="CD24mid",
  scaleYLimits=c(0, 1.2e7),
  scaleYLabels=scales::scientific,
  legendPosition="none"
)
```

```
plotB6 <- abcTemplate(
  data=dataset2,
  title="",
  subtitle="Swarm-deconvoluted",
  measurement="CD24hi",
  scaleYLimits=c(0, 1.2e6),
  scaleYLabels=scales::scientific,
  legendPosition="none"
)
```

4.3.3.2 Dataset 2 - Swarm-deconvoluted

```
# Let's generate plots to show the final result

plotC1 <- abcTemplate(
  data=dataset3,
  title="",
  subtitle="Swarm deconvoluted + Denoised",
  measurement="cd24posProp",
  scaleYLimits=c(0, 1),
  scaleYLabels=scales::number,
  legendPosition="none"
)

plotC2 <- abcTemplate(
  data=dataset3,
  title="",
  subtitle="Swarm deconvoluted + Denoised",
  measurement="cd24midProp",
  scaleYLimits=c(0, 1),
  scaleYLabels=scales::number,
  legendPosition="none"
)

plotC3 <- abcTemplate(
  data=dataset3,
  title="",
  subtitle="Swarm deconvoluted + Denoised",
  measurement="cd24hiProp",
  scaleYLimits=c(0, 1),
  scaleYLabels=scales::number,
  legendPosition="none"
)

plotC4 <- abcTemplate(
  data=dataset3,
  title="",
  subtitle="Swarm deconvoluted + Denoised",
  measurement="CD24pos",
  scaleYLimits=c(0, 1.2e7),
  scaleYLabels=scales::scientific,
  legendPosition="none"
)
```

```
plotC5 <- abcTemplate(
  data=dataset3,
  title="",
  subtitle="Swarm deconvoluted + Denoised",
  measurement="CD24mid",
  scaleYLimits=c(0, 1.2e7),
  scaleYLabels=scales::scientific,
  legendPosition="none"
)

plotC6 <- abcTemplate(
  data=dataset3,
  title="",
  subtitle="Swarm deconvoluted + Denoised",
  measurement="CD24hi",
  scaleYLimits=c(0, 1.2e6),
  scaleYLabels=scales::scientific,
  legendPosition="none"
)
```

4.3.3.3 Dataset 3 - Swarm-deconvoluted and normalized

4.3.4 Building Summary graphs

```
defTemplate <- function(
  data,
  title,
  subtitle,
  measurement,
  sem,
  scaleYLimits,
  scaleYLabels,
  legendPosition
) {
  measurement <- rlang::sym(measurement)
  sem <- rlang::sym(sem)

  ggplot2::ggplot(
    data=data,
    mapping=ggplot2::aes(
      x=Dilution,
      y=!!measurement,
      fill=CellLine
    )
  ) +
    ggplot2::labs(
      title=title,
      subtitle=subtitle,
      x=ggplot2::element_blank(),
      y=ggplot2::element_blank()) +
    ggplot2::theme(
      panel.background=ggplot2::element_rect(fill="#F4F4F4"),
      panel.grid.major.x=ggplot2::element_blank(),
```

```
      panel.grid.minor.x=ggplot2::element_blank(),
      panel.grid.minor.y=ggplot2::element_line(colour="#EEEEEE"),
      axis.ticks=ggplot2::element_blank(),
      axis.title.x=ggplot2::element_text(vjust=-2),
      axis.title.y=ggplot2::element_text(
        angle=90,
        vjust=2),
      legend.position=legendPosition,
      legend.justification=c(1, 1),
      legend.title=ggplot2::element_blank()) +
    ggplot2::geom_bar(
      position=ggplot2::position_dodge(),
      stat="identity") +
    ggplot2::geom_errorbar(
      mapping=ggplot2::aes(
        ymin=!!measurement - !!sem,
        ymax=!!measurement + !!sem,
        color=CellLine),
      width=15,
      size=1,
      position=ggplot2::position_dodge(35),
      show.legend=FALSE) +
    ggplot2::scale_fill_manual(
      name="Cell line",
      values=c(
        "#F3756E",
        "#1DBDC3")) +
    ggplot2::scale_color_manual(
      values=c(
        "#B8645B",
        "#43989C")) +
    ggplot2::scale_x_continuous(
      limits=c(0, 240),
      breaks=c(20, 60, 100, 140, 180, 220),
      minor_breaks=c(40, 80, 120, 160, 200)) +
    ggplot2::scale_y_continuous(
      limits=scaleYLimits,
      labels=scaleYLabels)
}

# Show the swarm effect

plotD1 <- defTemplate(
  data=dataset1Summary,
  title="Proportion of CD24+ events",
  subtitle="Swarm effect",
  measurement="cd24posPropAvg",
  sem="cd24posPropSem",
  scaleYLimits=c(0, 1),
  scaleYLabels=scales::number,
  legendPosition="none"
)
```

```
plotD2 <- defTemplate(
  data=dataset1Summary,
  title="Proportion of CD24mid events",
  subtitle="Swarm effect",
  measurement="cd24midPropAvg",
  sem="cd24midPropSem",
  scaleYLimits=c(0, 1),
  scaleYLabels=scales::number,
  legendPosition="none"
)

plotD3 <- defTemplate(
  data=dataset1Summary,
  title="Proportion of CD24hi events",
  subtitle="Swarm effect",
  measurement="cd24hiPropAvg",
  sem="cd24hiPropSem",
  scaleYLimits=c(0, 1),
  scaleYLabels=scales::number,
  legendPosition=c(1, 1)
)

plotD4 <- defTemplate(
  data=dataset1Summary,
  title="Absolute CD24+ events count",
  subtitle="Swarm effect",
  measurement="cd24posAvg",
  sem="cd24posSem",
  scaleYLimits=c(0, 1.2e7),
  scaleYLabels=scales::scientific,
  legendPosition="none"
)

plotD5 <- defTemplate(
  data=dataset1Summary,
  title="Absolute CD24mid events count",
  subtitle="Swarm effect",
  measurement="cd24midAvg",
  sem="cd24midSem",
  scaleYLimits=c(0, 1.2e7),
  scaleYLabels=scales::scientific,
  legendPosition="none"
)

plotD6 <- defTemplate(
  data=dataset1Summary,
  title="Absolute CD24hi events count",
  subtitle="Swarm effect",
  measurement="cd24hiAvg",
  sem="cd24hiSem",
  scaleYLimits=c(0, 1.2e6),
  scaleYLabels=scales::scientific,
  legendPosition=c(1, 1)
```

```
)
```

4.3.4.1 Dataset 1 - Raw gate counts

```
# Swarm-deconvoluted data plots

plotE1 <- defTemplate(
  data=dataset2Summary,
  title="",
  subtitle="Swarm-deconvoluted",
  measurement="cd24posPropAvg",
  sem="cd24posPropSem",
  scaleYLimits=c(0, 1),
  scaleYLabels=scales::number,
  legendPosition="none"
)

plotE2 <- defTemplate(
  data=dataset2Summary,
  title="",
  subtitle="Swarm-deconvoluted",
  measurement="cd24midPropAvg",
  sem="cd24midPropSem",
  scaleYLimits=c(0, 1),
  scaleYLabels=scales::number,
  legendPosition="none"
)

plotE3 <- defTemplate(
  data=dataset2Summary,
  title="",
  subtitle="Swarm-deconvoluted",
  measurement="cd24hiPropAvg",
  sem="cd24hiPropSem",
  scaleYLimits=c(0, 1),
  scaleYLabels=scales::number,
  legendPosition="none"
)

plotE4 <- defTemplate(
  data=dataset2Summary,
  title="",
  subtitle="Swarm-deconvoluted",
  measurement="cd24posAvg",
  sem="cd24posSem",
  scaleYLimits=c(0, 1.2e7),
  scaleYLabels=scales::scientific,
  legendPosition="none"
)

plotE5 <- defTemplate(
  data=dataset2Summary,
  title="",
```

```
  subtitle="Swarm-deconvoluted",
  measurement="cd24midAvg",
  sem="cd24midSem",
  scaleYLimits=c(0, 1.2e7),
  scaleYLabels=scales::scientific,
  legendPosition="none"
)

plotE6 <- defTemplate(
  data=dataset2Summary,
  title="",
  subtitle="Swarm-deconvoluted",
  measurement="cd24hiAvg",
  sem="cd24hiSem",
  scaleYLimits=c(0, 1.2e6),
  scaleYLabels=scales::scientific,
  legendPosition="none"
)
```

4.3.4.2 Dataset 2 - Swarm-deconvoluted

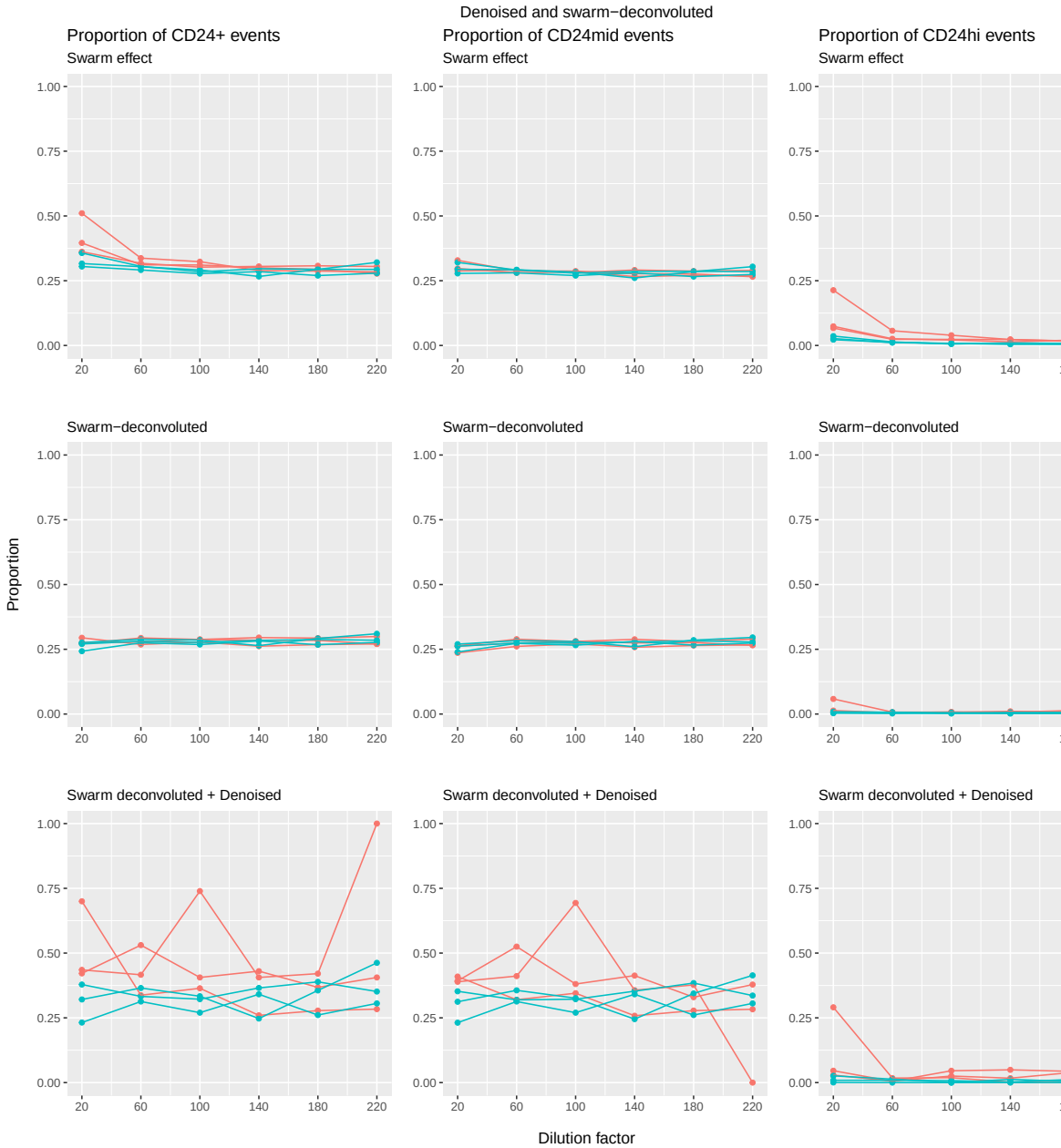
```
# Let's generate plots to show the final result

plotF1 <- defTemplate(
  data=dataset3Summary,
  title="",
  subtitle="Denoised and swarm-deconvoluted",
  measurement="cd24posPropAvg",
  sem="cd24posPropSem",
  scaleYLimits=c(0, 1),
  scaleYLabels=scales::number,
  legendPosition="none"
)

plotF2 <- defTemplate(
  data=dataset3Summary,
  title="",
  subtitle="Denoised and swarm-deconvoluted",
  measurement="cd24midPropAvg",
  sem="cd24midPropSem",
  scaleYLimits=c(0, 1),
  scaleYLabels=scales::number,
  legendPosition="none"
)

plotF3 <- defTemplate(
  data=dataset3Summary,
  title="",
  subtitle="Denoised and swarm-deconvoluted",
  measurement="cd24hiPropAvg",
  sem="cd24hiPropSem",
  scaleYLimits=c(0, 1),
  scaleYLabels=scales::number,
```

```
  legendPosition="none"
)

plotF4 <- defTemplate(
  data=dataset3Summary,
  title="",
  subtitle="Denoised and swarm-deconvoluted",
  measurement="cd24posAvg",
  sem="cd24posSem",
  scaleYLimits=c(0, 1.2e7),
  scaleYLabels=scales::scientific,
  legendPosition="none"
)

plotF5 <- defTemplate(
  data=dataset3Summary,
  title="",
  subtitle="Denoised and swarm-deconvoluted",
  measurement="cd24midAvg",
  sem="cd24midSem",
  scaleYLimits=c(0, 1.2e7),
  scaleYLabels=scales::scientific,
  legendPosition="none"
)

plotF6 <- defTemplate(
  data=dataset3Summary,
  title="",
  subtitle="Denoised and swarm-deconvoluted",
  measurement="cd24hiAvg",
  sem="cd24hiSem",
  scaleYLimits=c(0, 1.2e6),
  scaleYLabels=scales::scientific,
  legendPosition="none"
)
```

4.3.4.3 Dataset 3 - Swarm-deconvoluted and normalized

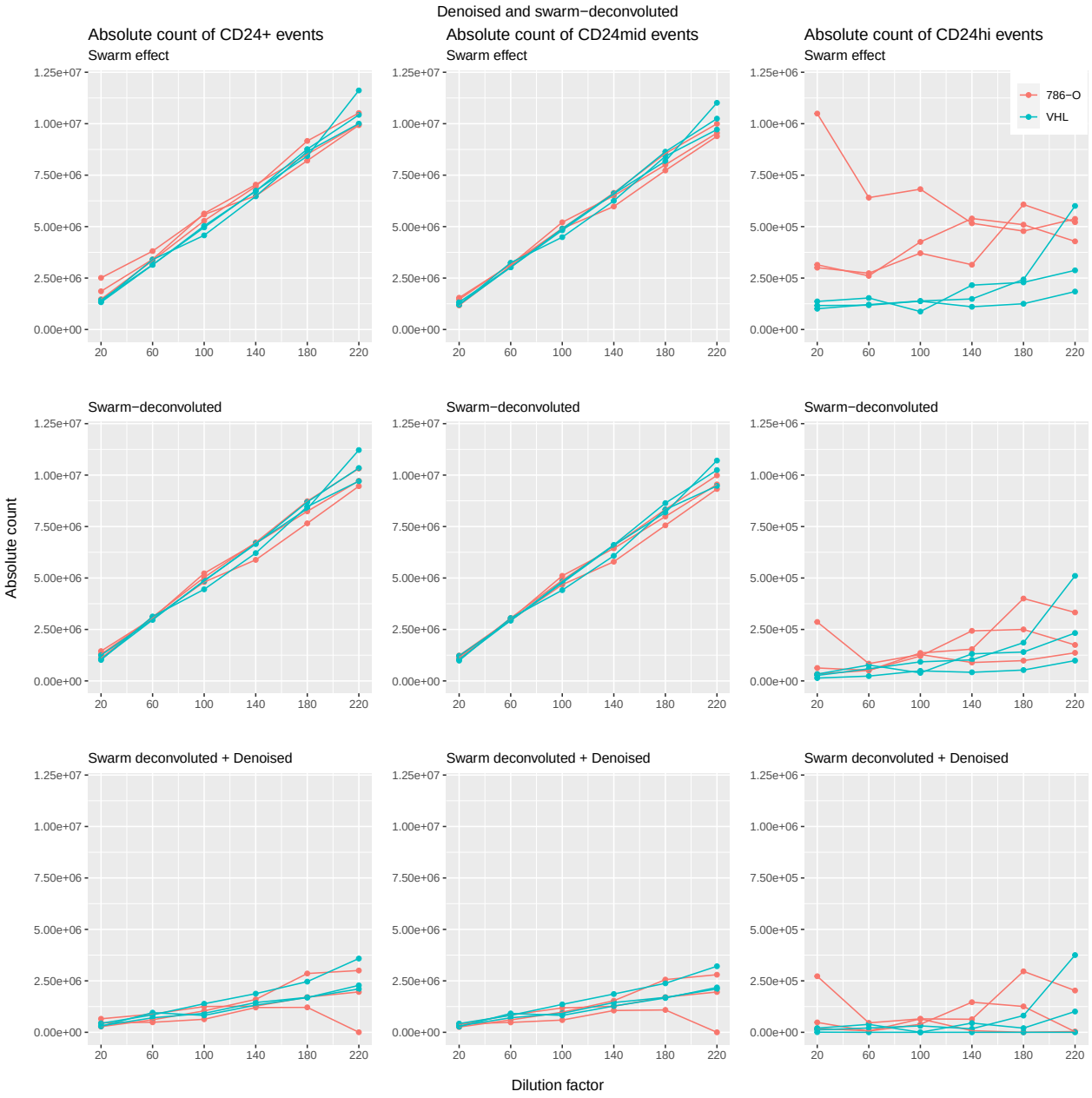
4.3.5 Visualizing the process

```
gridExtra::grid.arrange(
  plotA1, plotA2, plotA3,
  plotB1, plotB2, plotB3,
  plotC1, plotC2, plotC3,
  nrow=3,
  top="Denoised and swarm-deconvoluted",
  left="Proportion",
  bottom="Dilution factor"
)
```



4.3.5.1 Sample graphs

```
gridExtra::grid.arrange(
  plotA4, plotA5, plotA6,
  plotB4, plotB5, plotB6,
  plotC4, plotC5, plotC6,
  nrow=3,
  top="Denoised and swarm-deconvoluted",
  left="Absolute count",
  bottom="Dilution factor"
)
```



```
gridExtra::grid.arrange(
  plotD1, plotD2, plotD3,
  plotE1, plotE2, plotE3,
  plotF1, plotF2, plotF3,
  nrow=3,
  top="Denoised and swarm-deconvoluted",
  left="Proportion",
  bottom="Dilution factor"
)
```

4.3.5.2 Summary graphs

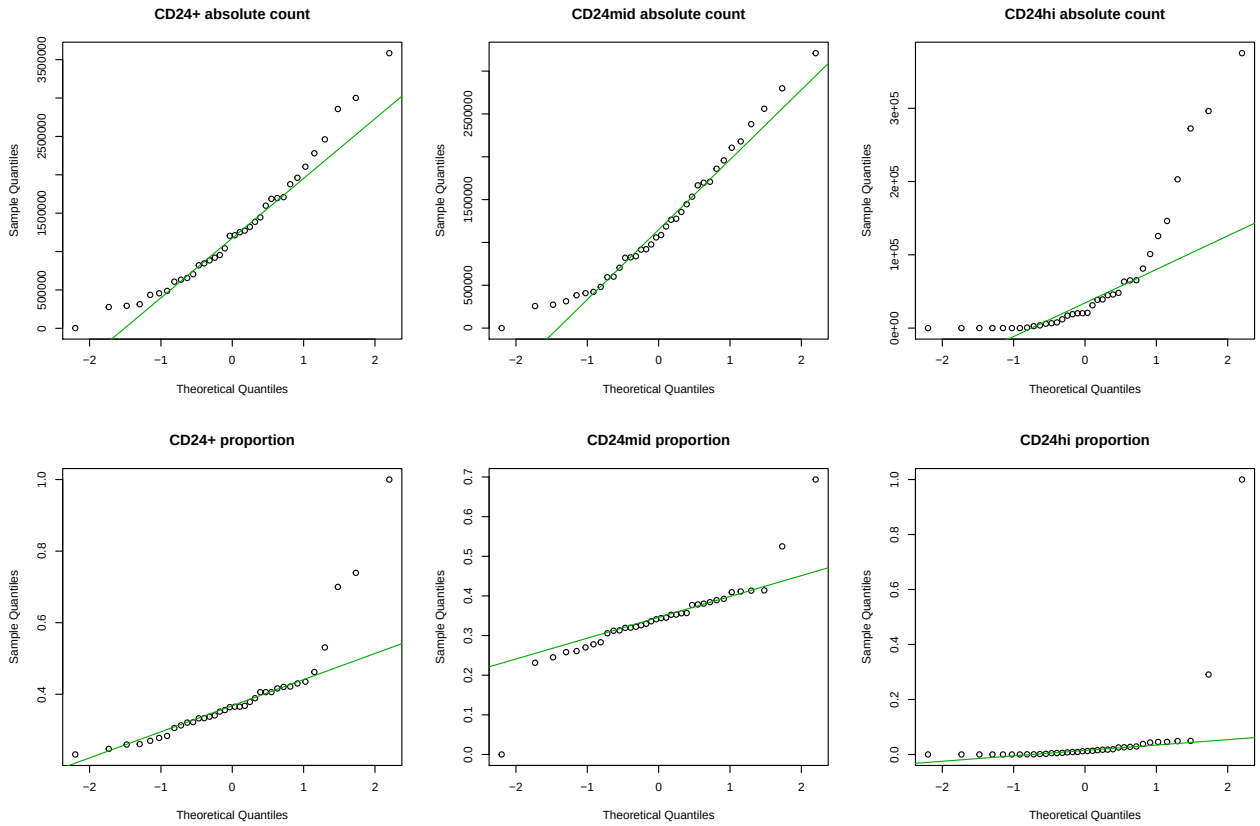
```
gridExtra::grid.arrange(
  plotD4, plotD5, plotD6,
  plotE4, plotE5, plotE6,
  plotF4, plotF5, plotF6,
  nrow=3,
  top="Denoised and swarm-deconvoluted",
  left="Absolute count",
  bottom="Dilution factor"
)
```

4.3.6 Significance testing

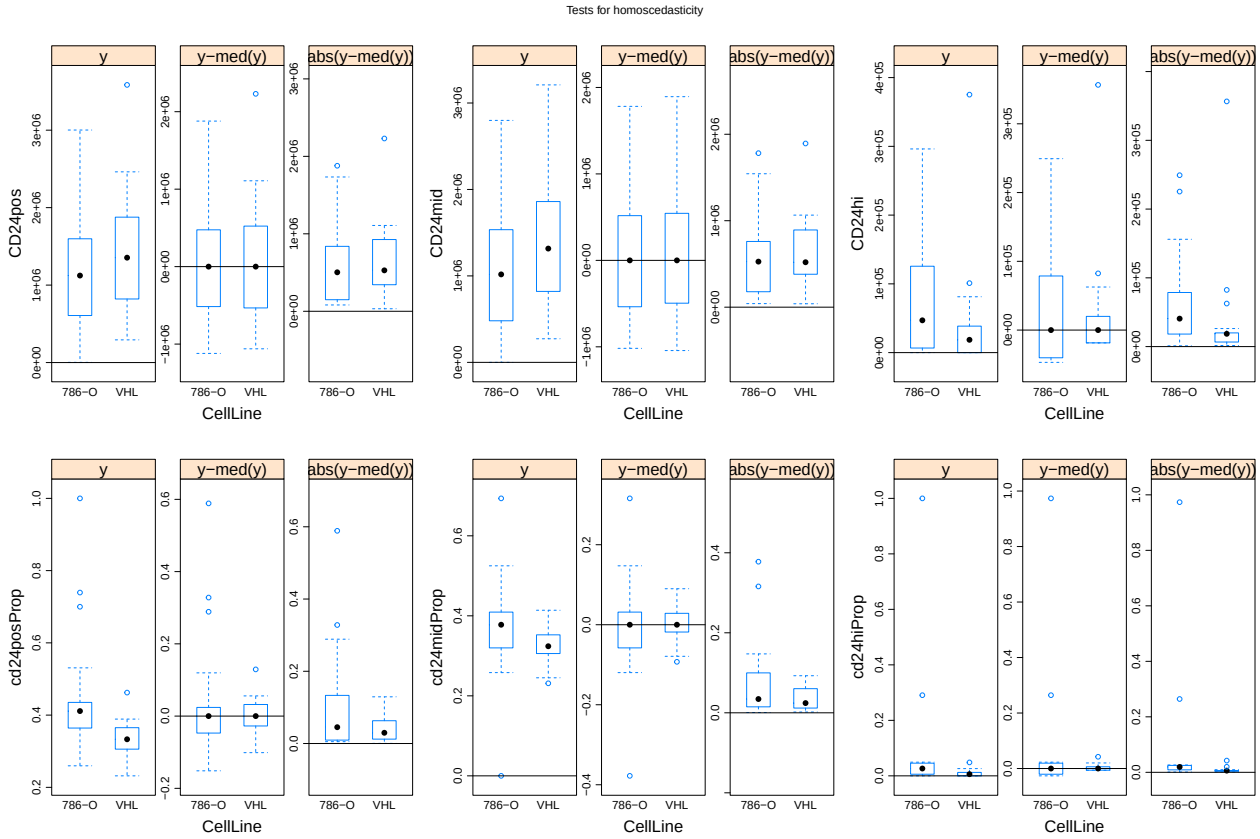
First we must verify that test assumptions are met.

```
# Testing for univariate normality
significanceDfShapiro <- c(
  stats::shapiro.test(dataset3$CD24pos)$p.value,
  stats::shapiro.test(dataset3$CD24mid)$p.value,
  stats::shapiro.test(dataset3$CD24hi)$p.value,
  stats::shapiro.test(dataset3$cd24posProp)$p.value,
  stats::shapiro.test(dataset3$cd24midProp)$p.value,
  stats::shapiro.test(dataset3$cd24hiProp)$p.value
)

# Testing for homoscedasticity
significanceDfHov <- c(
  HH::hov(CD24pos ~ CellLine, data=dataset3)$p.value,
  HH::hov(CD24mid ~ CellLine, data=dataset3)$p.value,
  HH::hov(CD24hi ~ CellLine, data=dataset3)$p.value,
  HH::hov(cd24posProp ~ CellLine, data=dataset3)$p.value,
  HH::hov(cd24midProp ~ CellLine, data=dataset3)$p.value,
  HH::hov(cd24hiProp ~ CellLine, data=dataset3)$p.value
)

significanceDfBartlett <- c(
  stats::bartlett.test(CD24pos ~ CellLine, data=dataset3)$p.value,
  stats::bartlett.test(CD24mid ~ CellLine, data=dataset3)$p.value,
  stats::bartlett.test(CD24hi ~ CellLine, data=dataset3)$p.value,
  stats::bartlett.test(cd24posProp ~ CellLine, data=dataset3)$p.value,
  stats::bartlett.test(cd24midProp ~ CellLine, data=dataset3)$p.value,
  stats::bartlett.test(cd24hiProp ~ CellLine, data=dataset3)$p.value
)

significanceDfFligner <- c(
  stats::fligner.test(CD24pos ~ CellLine, data=dataset3)$p.value,
  stats::fligner.test(CD24mid ~ CellLine, data=dataset3)$p.value,
  stats::fligner.test(CD24hi ~ CellLine, data=dataset3)$p.value,
  stats::fligner.test(cd24posProp ~ CellLine, data=dataset3)$p.value,
  stats::fligner.test(cd24midProp ~ CellLine, data=dataset3)$p.value,
  stats::fligner.test(cd24hiProp ~ CellLine, data=dataset3)$p.value
)
```

```
# Display multiplot in 2 rows and 3 columns
graphics::par(mfrow=c(2, 3))

# Base plots for univariate normality test
stats::qqnorm(dataset3$CD24pos, main="CD24+ absolute count")
stats::qqline(dataset3$CD24pos, col=colorPalette[["green"]])

stats::qqnorm(dataset3$CD24mid, main="CD24mid absolute count")
stats::qqline(dataset3$CD24mid, col=colorPalette[["green"]])

stats::qqnorm(dataset3$CD24hi, main="CD24hi absolute count")
stats::qqline(dataset3$CD24hi, col=colorPalette[["green"]])

stats::qqnorm(dataset3$cd24posProp, main="CD24+ proportion")
stats::qqline(dataset3$cd24posProp, col=colorPalette[["green"]])

stats::qqnorm(dataset3$cd24midProp, main="CD24mid proportion")
stats::qqline(dataset3$cd24midProp, col=colorPalette[["green"]])

stats::qqnorm(dataset3$cd24hiProp, main="CD24hi proportion")
stats::qqline(dataset3$cd24hiProp, col=colorPalette[["green"]])
```



```r
# Base plots for homoscedasticity test
cd24posHov     <- HH::hovPlot(CD24pos ~ CellLine, data=dataset3)
cd24midHov     <- HH::hovPlot(CD24mid ~ CellLine, data=dataset3)
cd24hiHov      <- HH::hovPlot(CD24hi ~ CellLine, data=dataset3)
cd24posPropHov <- HH::hovPlot(cd24posProp ~ CellLine, data=dataset3)
cd24midPropHov <- HH::hovPlot(cd24midProp ~ CellLine, data=dataset3)
cd24hiPropHov  <- HH::hovPlot(cd24hiProp ~ CellLine, data=dataset3)

gridExtra::grid.arrange(
  cd24posHov,     cd24midHov,     cd24hiHov,
  cd24posPropHov, cd24midPropHov, cd24hiPropHov,
  nrow=2,
  top="Tests for homoscedasticity"
)
```



The basic test we are interested in is whether there exists a difference in event counts between the two cell lines. In this case, event counts is the dependant variable, and the cell line is an independant variable. The basic notation for this is `(DependantVariable ~ IndependantVariable)`.

In our experimental model, the independant variable `CellLine` is a fixed variable: it can either be '786-O' or 'VHL', with no possibility for error. Conversely, the dilution factor is not a fixed variable but is rather a random variable which must be taken into account: while a sample may be prepared to be diluted in a 1:120 ratio, there is inherent error or variability in instrument precision while making these measurements. This can be taken into account for the ANOVA with the notation `+ Error(RandomVariable/DependantVariable)`.

```
# Perform an anova on each parameter

cd24posAov <- stats::aov(
  CD24pos ~ CellLine + Error(Dilution/CD24pos),
  data=dataset3
)

cd24posPropAov <- stats::aov(
  cd24posProp ~ CellLine + Error(Dilution/cd24posProp),
  data=dataset3
)

cd24midAov <- stats::aov(
  CD24mid ~ CellLine + Error(Dilution/CD24mid),
  data=dataset3
)

cd24midPropAov <- stats::aov(
  cd24midProp ~ CellLine + Error(Dilution/cd24midProp),
  data=dataset3
)

cd24hiAov <- stats::aov(
  CD24hi ~ CellLine + Error(Dilution/CD24hi),
  data=dataset3
)

cd24hiPropAov <- stats::aov(
  cd24hiProp ~ CellLine + Error(Dilution/cd24hiProp),
  data=dataset3
)

# Get the p-Value for each ANOVA performed
significanceDfAnova <- c(
  getANOVApValue(cd24posAov),
  getANOVApValue(cd24posPropAov),
  getANOVApValue(cd24midAov),
  getANOVApValue(cd24midPropAov),
  getANOVApValue(cd24hiAov),
  getANOVApValue(cd24hiPropAov)
)

# Also perform a non-parametric equivalent of an ANOVA and retrieve the p-Values
significanceDfKruskal <- c(
  stats::kruskal.test(CD24pos ~ CellLine, data=dataset3)$p.value,
```

```
  stats::kruskal.test(CD24mid ~ CellLine, data=dataset3)$p.value,
  stats::kruskal.test(CD24hi ~ CellLine, data=dataset3)$p.value,
  stats::kruskal.test(cd24posProp ~ CellLine, data=dataset3)$p.value,
  stats::kruskal.test(cd24midProp ~ CellLine, data=dataset3)$p.value,
  stats::kruskal.test(cd24hiProp ~ CellLine, data=dataset3)$p.value
)

# Create a big matrix with the p-Values for all the tests we did
significanceDf <- cbind(
  significanceDfShapiro,
  significanceDfHov,
  significanceDfBartlett,
  significanceDfFligner,
  significanceDfAnova,
  significanceDfKruskal
)

# Ensure rows and columns are named correctly
colnames(significanceDf) <- c(
  "Shapiro-Wilk",
  "Brown-Forsythe",
  "Bartlett",
  "Fligner-Killeen",
  "ANOVA",
  "Kruskal-Wallis"
)

significanceDfParameters <- c(
  "CD24pos",
  "CD24mid",
  "CD24hi",
  "cd24posProp",
  "cd24midProp",
  "cd24hiProp"
)

row.names(significanceDf) <- significanceDfParameters

# Convert the matrix into a dataframe
significanceDf <- data.frame(significanceDf)

# Let's see what we have
printDataFrame(
  dataframe=significanceDf,
  caption="Assumption testing and ANOVA significance testing summary",
  fontsize=7
)
```

Table 19: Assumption testing and ANOVA significance testing summary

	Shapiro.Wilk	Brown.Forsythe	Bartlett	Fligner.Killeen	ANOVA	Kruskal.Wallis
CD24pos	0.0684941	0.7781357	0.8589513	0.7239056	0.6415410	0.3588705
CD24mid	0.1537525	0.7763855	0.9260672	0.7426260	0.0128291	0.3266931
CD24hi	0.0000002	0.3408498	0.8096368	0.1245900	0.3497293	0.0610075
cd24posProp	0.0000011	0.0710915	0.0000095	0.1670916	0.0022994	0.0029392

Table 19: Assumption testing and ANOVA significance testing summary *(continued)*

	Shapiro.Wilk	Brown.Forsythe	Bartlett	Fligner.Killeen	ANOVA	Kruskal.Wallis
cd24midProp	0.0001278	0.0901714	0.0000809	0.1122854	0.0611137	0.0428807
cd24hiProp	0.0000000	0.1819636	0.0000000	0.0057219	0.1486423	0.0031465

```
# Transform this dataframe into a tidy format for plotting purposes
significanceDfMolten <- significanceDf
significanceDfMolten$parameter <- row.names(significanceDf)
significanceDfMolten <- reshape2::melt(
  data=significanceDfMolten,
  id.vars="parameter",
  value.name="p",
  variable.name="test"
)

# Break down p-Value into discrete intervals
significanceDfMolten$pDiscrete <- cut(
  significanceDfMolten$p,
  breaks=c(0.00, 0.01, 0.05, 0.10, 0.15, 0.20, 0.25, 1.00),
  labels=c(
    "p < 0.01",
    "p < 0.05",
    "p < 0.10",
    "p < 0.15",
    "p < 0.20",
    "p < 0.25",
    "p > 0.25"
  )
)

# Make sure the levels is plotted from top to bottom in order
significanceDfMolten$parameter <- factor(
  as.character(significanceDfMolten$parameter),
  levels=rev(significanceDfParameters)
)
```

```
# Let's see what we have
printDataFrame(
  dataframe=significanceDfMolten,
  caption=
    "Assumption testing and ANOVA significance testing summary (longform)",
  fontsize=7
)
```

Table 20: Assumption testing and ANOVA significance testing summary (longform)

parameter	test	p	pDiscrete
CD24pos	Shapiro.Wilk	0.0684941	p < 0.10
CD24mid	Shapiro.Wilk	0.1537525	p < 0.20
CD24hi	Shapiro.Wilk	0.0000002	p < 0.01
cd24posProp	Shapiro.Wilk	0.0000011	p < 0.01
cd24midProp	Shapiro.Wilk	0.0001278	p < 0.01
cd24hiProp	Shapiro.Wilk	0.0000000	p < 0.01
CD24pos	Brown.Forsythe	0.7781357	p > 0.25
CD24mid	Brown.Forsythe	0.7763855	p > 0.25
CD24hi	Brown.Forsythe	0.3408498	p > 0.25
cd24posProp	Brown.Forsythe	0.0710915	p < 0.10
cd24midProp	Brown.Forsythe	0.0901714	p < 0.10
cd24hiProp	Brown.Forsythe	0.1819636	p < 0.20
CD24pos	Bartlett	0.8589513	p > 0.25
CD24mid	Bartlett	0.9260672	p > 0.25
CD24hi	Bartlett	0.8096368	p > 0.25
cd24posProp	Bartlett	0.0000095	p < 0.01
cd24midProp	Bartlett	0.0000809	p < 0.01
cd24hiProp	Bartlett	0.0000000	p < 0.01
CD24pos	Fligner.Killeen	0.7239056	p > 0.25
CD24mid	Fligner.Killeen	0.7426260	p > 0.25
CD24hi	Fligner.Killeen	0.1245900	p < 0.15
cd24posProp	Fligner.Killeen	0.1670916	p < 0.20
cd24midProp	Fligner.Killeen	0.1122854	p < 0.15
cd24hiProp	Fligner.Killeen	0.0057219	p < 0.01
CD24pos	ANOVA	0.6415410	p > 0.25
CD24mid	ANOVA	0.0128291	p < 0.05
CD24hi	ANOVA	0.3497293	p > 0.25
cd24posProp	ANOVA	0.0022994	p < 0.01
cd24midProp	ANOVA	0.0611137	p < 0.10
cd24hiProp	ANOVA	0.1486423	p < 0.15
CD24pos	Kruskal.Wallis	0.3588705	p > 0.25
CD24mid	Kruskal.Wallis	0.3266931	p > 0.25
CD24hi	Kruskal.Wallis	0.0610075	p < 0.10
cd24posProp	Kruskal.Wallis	0.0029392	p < 0.01
cd24midProp	Kruskal.Wallis	0.0428807	p < 0.05
cd24hiProp	Kruskal.Wallis	0.0031465	p < 0.01

```
# Create a heat map of p-Values for each parameter
heatmap <- ggplot2::ggplot(
  data=significanceDfMolten,
  mapping=ggplot2::aes(
    x=test,
    y=parameter)
) +
  ggplot2::geom_tile(
    mapping=ggplot2::aes(fill=pDiscrete),
    colour="white",
    size=1) +
  ggplot2::geom_text(
    mapping=ggplot2::aes(label=sprintf("%.2f", round(p, digits=2))),
    colour="white",
    fontface="bold",
    size=3) +
  ggplot2::labs(
    title="Assumption testing and ANOVA significance testing summary",
    subtitle="Swarm-deconvoluted and denoised event counts",
    x="",
    y="") +
  ggplot2::scale_y_discrete(expand=c(0, 0)) +
  ggplot2::scale_x_discrete(
    expand=c(0, 0),
    position="top") +
  ggplot2::scale_fill_manual(
    values=c(
      "#d53e4f",
      "#f46d43",
      "#fdae61",
      "#fee08b",
      "#e6f598",
      "#abdda4",
      "#ddf1da"),
    na.value="grey90") +
  ggplot2::theme(
    plot.background=ggplot2::element_blank(),
    legend.title=ggplot2::element_blank(),
    axis.text.x=ggplot2::element_text(
      angle=90,
      hjust=0,
      vjust=0.5))
```

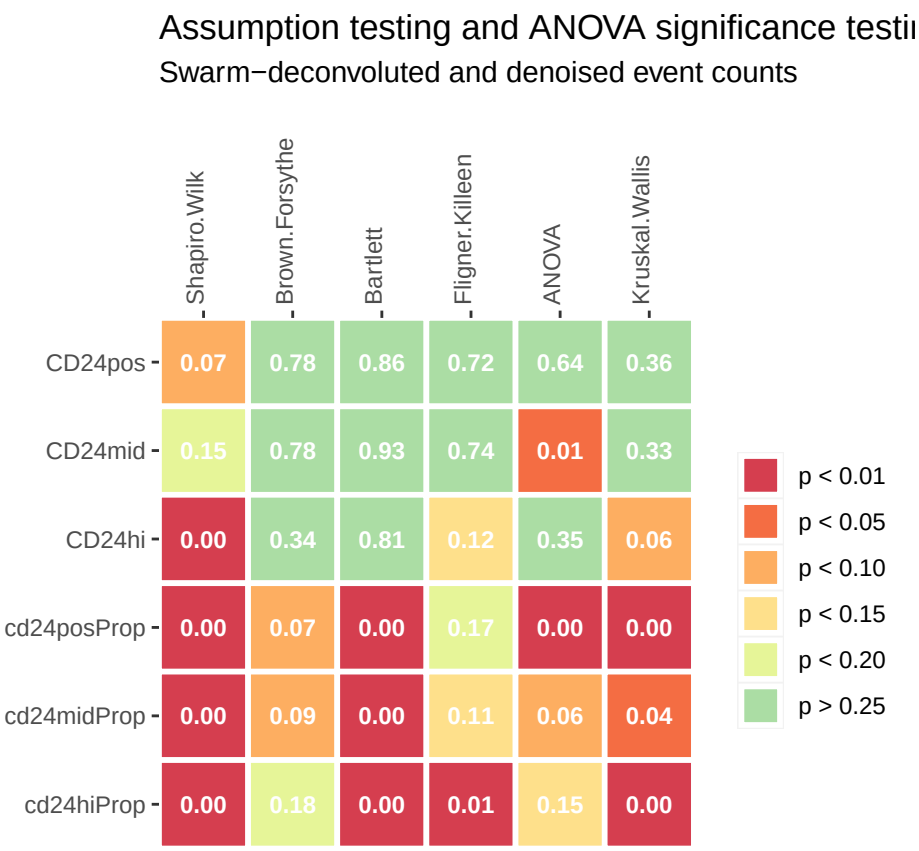
A Shapiro-Wilk test statistic > 0.05 indicates that the univariate normality condition is met. While normality is technically a test assumption, in practice the ANOVA is relatively robust to departures from normality.

A Brown-Forsythe test statistic > 0.05 indicates that the condition for homoscedasticity is met. This test is better suited to minor departures from normality.

A Bartlett test statistic > 0.05 indicates that the condition for homoscedasticity is met. This test is very sensitive to minor departures from normality.

A Fligner-Killeen test statistic > 0.05 indicates that the condition for homoscedasticity is met. This test is less sensitive to departures from normality or to the influence of outliers. Failing other tests but meeting the Fligner-Killeen test could justify the downstream use of nonparametric methods (Such as the Kruskal-Wallis rank sum test instead of the ANOVA).

```
# Visualization
heatmap
```



The results of significance testing for our data appear to indicate a statistically significant difference between 786-O and VHL cell-derived extracellular vesicles for the absolute count of CD24mid events, as well as for the overall proportion of CD24+ events and the overall proportion of CD24mid events.

```
datasetNormalizedForLm <- normalizedExperiment

datasetNormalizedForLm <- datasetNormalizedForLm %>%
  dplyr::mutate_at(
    c("Events", "CD24pos", "CD24mid", "CD24hi", "Dilution"),
    ~ (scale(.) %>% as.vector)
  )

lmFitBase <- stats::lm(
```

```
  CD24mid ~ CellLine,
  data=datasetNormalizedForLm
)

lmFitDilution <- stats::lm(
  CD24mid ~ CellLine + Dilution,
  data=datasetNormalizedForLm
)

lmFitSwarmDilution <- stats::lm(
  CD24mid ~ CellLine + cd24midTp + Dilution,
  data=datasetNormalizedForLm
)

afBase <- stats::anova(lmFitBase)
afBase$percentVariance <- (afBase$"Sum Sq" / sum(afBase$"Sum Sq")) * 100

afBase
```

```
## Analysis of Variance Table
##
## Response: CD24mid
##           Df Sum Sq Mean Sq F value  Pr(>F) percentVariance
## CellLine   1  0.012 0.01222  0.0119 0.91385           0.035
## Residuals 34 34.988 1.02905                          99.965
```

```
afDilution <- stats::anova(lmFitDilution)
afDilution$percentVariance <-
  (afDilution$"Sum Sq" / sum(afDilution$"Sum Sq")) * 100

afDilution
```

```
## Analysis of Variance Table
##
## Response: CD24mid
##           Df Sum Sq Mean Sq   F value  Pr(>F) percentVariance
## CellLine   1  0.012   0.012    1.1639 0.28848           0.035
## Dilution   1 34.641  34.641 3298.1863 0.00000          98.975
## Residuals 33  0.347   0.011                             0.990
```

```
afSwarmDilution <- stats::anova(lmFitSwarmDilution)
afSwarmDilution$percentVariance <-
  (afSwarmDilution$"Sum Sq" / sum(afSwarmDilution$"Sum Sq")) * 100

afSwarmDilution
```

```
## Analysis of Variance Table
##
## Response: CD24mid
##           Df  Sum Sq Mean Sq   F value  Pr(>F) percentVariance
## CellLine   1  0.0122  0.0122    1.1315 0.29541           0.035
## cd24midTp  1 16.6756 16.6756 1543.5222 0.00000          47.645
## Dilution   1 17.9664 17.9664 1663.0031 0.00000          51.333
## Residuals 32  0.3457  0.0108                            0.988
```

```
decreaseInVarianceFromDilution <- 1 - (
  afSwarmDilution["Dilution", "percentVariance"]
    / afDilution["Dilution", "percentVariance"])

decreaseInVarianceFromDilution
```

```
## [1] 0.4813558
```

```
datasetNormalizedForLm <- normalizedExperiment

datasetNormalizedForLm <- datasetNormalizedForLm %>%
  dplyr::mutate_at(
    c("Events", "CD24pos", "CD24mid", "CD24hi", "Dilution"),
    ~ (scale(.) %>% as.vector)
  )

lmFitBase <- stats::lm(
  CD24hi ~ CellLine,
  data=datasetNormalizedForLm
)

lmFitDilution <- stats::lm(
  CD24hi ~ CellLine + Dilution,
  data=datasetNormalizedForLm
)

lmFitSwarmDilution <- stats::lm(
  CD24hi ~ CellLine + cd24hiTp + Dilution,
  data=datasetNormalizedForLm
)

afBase <- stats::anova(lmFitBase)
afBase$percentVariance <- (afBase$"Sum Sq" / sum(afBase$"Sum Sq")) * 100

afBase
```

```
## Analysis of Variance Table
##
## Response: CD24hi
##           Df Sum Sq Mean Sq F value     Pr(>F) percentVariance
## CellLine   1 17.462 17.4621  33.853 1.4827e-06          49.892
## Residuals 34 17.538  0.5158                             50.108
```

```
afDilution <- stats::anova(lmFitDilution)
afDilution$percentVariance <-
  (afDilution$"Sum Sq" / sum(afDilution$"Sum Sq")) * 100

afDilution
```

```
## Analysis of Variance Table
##
## Response: CD24hi
##           Df  Sum Sq Mean Sq F value   Pr(>F) percentVariance
## CellLine   1 17.4621 17.4621 34.9170 0.000001          49.892
## Dilution   1  1.0346  1.0346  2.0687 0.159763           2.956
## Residuals 33 16.5034  0.5001                           47.152
```

```
afSwarmDilution <- stats::anova(lmFitSwarmDilution)
afSwarmDilution$percentVariance <-
  (afSwarmDilution$"Sum Sq" / sum(afSwarmDilution$"Sum Sq")) * 100

afSwarmDilution
```

```
## Analysis of Variance Table
##
## Response: CD24hi
##           Df  Sum Sq Mean Sq F value  Pr(>F) percentVariance
## CellLine   1 17.4621 17.4621 35.8117 0.00000          49.892
## cd24hiTp   1  1.9317  1.9317  3.9616 0.05515           5.519
## Dilution   1  0.0028  0.0028  0.0057 0.94022           0.008
## Residuals 32 15.6034  0.4876                          44.581
```

```
decreaseInVarianceFromDilution <- 1 - (
  afSwarmDilution["Dilution", "percentVariance"]
    / afDilution["Dilution", "percentVariance"])

decreaseInVarianceFromDilution
```

```
## [1] 0.9973076
```

```
datasetNormalizedForLm <- normalizedExperiment

datasetNormalizedForLm <- datasetNormalizedForLm %>%
  dplyr::mutate_at(
    c("Events", "CD24pos", "CD24mid", "CD24hi", "Dilution"),
    ~ (scale(.) %>% as.vector)
  )

lmFitBase <- stats::lm(
  CD24pos ~ CellLine,
  data=datasetNormalizedForLm
)

lmFitDilution <- stats::lm(
  CD24pos ~ CellLine + Dilution,
  data=datasetNormalizedForLm
)

lmFitSwarmDilution <- stats::lm(
  CD24pos ~ CellLine + cd24posTp + Dilution,
  data=datasetNormalizedForLm
)

afBase <- stats::anova(lmFitBase)
afBase$percentVariance <- (afBase$"Sum Sq" / sum(afBase$"Sum Sq")) * 100

afBase
```

```
## Analysis of Variance Table
##
## Response: CD24pos
##           Df Sum Sq Mean Sq F value  Pr(>F) percentVariance
## CellLine   1  0.037 0.03722  0.0362 0.85024           0.106
```

```
## Residuals 34 34.963 1.02832                              99.894
```

```
afDilution <- stats::anova(lmFitDilution)
afDilution$percentVariance <-
  (afDilution$"Sum Sq" / sum(afDilution$"Sum Sq")) * 100

afDilution
```

```
## Analysis of Variance Table
##
## Response: CD24pos
##           Df Sum Sq Mean Sq  F value  Pr(>F) percentVariance
## CellLine   1  0.037   0.037    2.357 0.13426           0.106
## Dilution   1 34.442  34.442 2180.831 0.00000          98.405
## Residuals 33  0.521   0.016                            1.489
```

```
afSwarmDilution <- stats::anova(lmFitSwarmDilution)
afSwarmDilution$percentVariance <-
  (afSwarmDilution$"Sum Sq" / sum(afSwarmDilution$"Sum Sq")) * 100

afSwarmDilution
```

```
## Analysis of Variance Table
##
## Response: CD24pos
##           Df  Sum Sq Mean Sq  F value  Pr(>F) percentVariance
## CellLine   1  0.0372  0.0372    2.534 0.12125           0.106
## cd24posTp  1 17.6166 17.6166 1199.268 0.00000          50.333
## Dilution   1 16.8761 16.8761 1148.862 0.00000          48.218
## Residuals 32  0.4701  0.0147                           1.343
```

```
decreaseInVarianceFromDilution <- 1 - (
  afSwarmDilution["Dilution", "percentVariance"]
    / afDilution["Dilution", "percentVariance"])

decreaseInVarianceFromDilution
```

```
## [1] 0.5100072
```

```
type3Anova <- car::Anova(
  stats::lm(
    CD24mid ~ CellLine + cd24midTp + Dilution,
    data=datasetNormalizedForLm,
  ),
  type=3
)

type3Anova$percentVariance <-
  (type3Anova$"Sum Sq" / sum(type3Anova$"Sum Sq")) * 100

type3Anova
```

```
## Anova Table (Type III tests)
##
## Response: CD24mid
##              Sum Sq Df   F value  Pr(>F) percentVariance
## (Intercept)  0.0006  1    0.0585 0.81048           0.003
## CellLine     0.0127  1    1.1795 0.28557           0.070
```

```
## cd24midTp    0.0009  1    0.0821 0.77636           0.005
## Dilution    17.9664  1 1663.0031 0.00000          98.036
## Residuals    0.3457 32                             1.886
```

```
type3Anova <- car::Anova(
  stats::lm(
    CD24hi ~ CellLine + cd24hiTp + Dilution,
    data=datasetNormalizedForLm,
  ),
  type=3
)

type3Anova$percentVariance <-
  (type3Anova$"Sum Sq" / sum(type3Anova$"Sum Sq")) * 100

type3Anova
```

```
## Anova Table (Type III tests)
##
## Response: CD24hi
##              Sum Sq Df F value  Pr(>F) percentVariance
## (Intercept)  0.4822  1  0.9888 0.32749           1.464
## CellLine    15.9506  1 32.7120 0.00000          48.425
## cd24hiTp     0.8999  1  1.8456 0.18380           2.732
## Dilution     0.0028  1  0.0057 0.94022           0.008
## Residuals   15.6034 32                          47.371
```

```
type3Anova <- car::Anova(
  stats::lm(
    CD24pos ~ CellLine + cd24posTp + Dilution,
    data=datasetNormalizedForLm,
  ),
  type=3
)

type3Anova$percentVariance <-
  (type3Anova$"Sum Sq" / sum(type3Anova$"Sum Sq")) * 100

type3Anova
```

```
## Anova Table (Type III tests)
##
## Response: CD24pos
##              Sum Sq Df   F value  Pr(>F) percentVariance
## (Intercept)  0.0567  1    3.8587 0.05822           0.325
## CellLine     0.0060  1    0.4079 0.52756           0.034
## cd24posTp    0.0511  1    3.4789 0.07135           0.293
## Dilution    16.8761  1 1148.8621 0.00000          96.656
## Residuals    0.4701 32                             2.692
```

5 References