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

08 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::pander(utils::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.1.12.0)*, *car(v.3.0-10)*, *carData(v.3.0-4)*, *purrr(v.0.3.4)*, *pander(v.0.6.3)*, *kableExtra(v.1.3.1)*, *HH(v.3.1-42)*, *multcomp(v.1.4-14)*, *TH.data(v.1.0-10)*, *MASS(v.7.3-53)*, *survival(v.3.2-7)*, *mvtnorm(v.1.1-1)*, *latticeExtra(v.0.6-29)*, *diptest(v.0.75-7)*, *mixtools(v.1.2.0)*, *knitr(v.1.30)*, *reshape2(v.1.4.4)*, *tidyr(v.1.1.2)*, *dplyr(v.1.0.2)*, *flowFP(v.1.48.0)*, *flowViz(v.1.54.0)*, *lattice(v.0.20-41)*, *flowStats(v.4.2.0)*, *ggcyto(v.1.18.0)*, *ncdfFlow(v.2.36.0)*, *BH(v.1.72.0-3)*, *RcppArmadillo(v.0.10.1.0.0)*, *ggplot2(v.3.3.2)*, *flowUtils(v.1.54.0)*, *flowWorkspace(v.4.2.0)*, *flowCore(v.2.2.0)*, *Biobase(v.2.50.0)*, *BiocGenerics(v.0.36.0)* and *gridExtra(v.2.3)*

loaded via a namespace (and not attached): *readxl(v.1.3.1)*, *backports(v.1.2.0)*, *Hmisc(v.4.4-1)*, *igraph(v.1.2.6)*, *plyr(v.1.8.6)*, *ConsensusClusterPlus(v.1.54.0)*, *splines(v.4.0.2)*, *gmp(v.0.6-1)*, *fda(v.5.1.5.1)*, *digest(v.0.6.27)*, *htmltools(v.0.5.0)*, *fansi(v.0.4.1)*, *magrittr(v.1.5)*, *checkmate(v.2.0.0)*, *CytoML(v.2.2.1)*, *cluster(v.2.1.0)*, *ks(v.1.11.7)*, *openxlsx(v.4.2.3)*, *aws.signature(v.0.6.0)*, *fastcluster(v.1.1.25)*, *RcppParallel(v.5.0.2)*, *matrixStats(v.0.57.0)*, *sandwich(v.3.0-0)*, *cytolib(v.2.2.0)*, *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.0.19)*, *jsonlite(v.1.7.1)*, *crayon(v.1.3.4)*, *hexbin(v.1.28.1)*, *graph(v.1.68.0)*, *zoo(v.1.8-8)*, *glue(v.1.4.2)*, *gtable(v.0.3.0)*, *zlibbioc(v.1.36.0)*, *webshot(v.0.5.2)*, *kernlab(v.0.9-29)*, *IDPmisc(v.1.1.20)*, *Rgraphviz(v.2.34.0)*, *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.2.1.0)*, *foreign(v.0.8-80)*, *mclust(v.5.4.6)*, *FlowSOM(v.1.22.0)*, *Formula(v.1.2-4)*, *tsne(v.0.1-3)*, *stats4(v.4.0.2)*, *vcd(v.1.4-8)*, *htmlwidgets(v.1.5.2)*, *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)*, *farver(v.2.0.3)*, *nnet(v.7.3-14)*, *tidyselect(v.1.1.0)*, *labeling(v.0.4.2)*, *rlang(v.0.4.8)*, *later(v.1.1.0.1)*, *cellranger(v.1.1.0)*, *munsell(v.0.5.0)*, *tools(v.4.0.2)*, *cli(v.2.1.0)*, *generics(v.0.1.0)*, *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.3.1-150)*, *RBGL(v.1.66.0)*, *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.3.0.4)*, *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.1.13.2)*, *corpcor(v.1.6.9)*, *httpuv(v.1.5.4)*, *R6(v.2.5.0)*, *promises(v.1.1.1)*, *KernSmooth(v.2.23-18)*, *RProtoBufLib(v.2.2.0)*, *rio(v.0.5.16)*, *codetools(v.0.2-18)*, *assertthat(v.0.2.1)*, *withr(v.2.3.0)*, *metafor(v.2.4-0)*, *S4Vectors(v.0.28.0)*, *hms(v.0.5.3)*, *rpart(v.4.1-15)*, *rmarkdown(v.2.5)*, *segmented(v.1.3-0)*, *shiny(v.1.5.0)*, *base64enc(v.0.1-3)* and *tinytex(v.0.27)*

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
) {
  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
```

```r
# 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(
  file="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",
  "20"    = "1:20",
  "c2"    = "1:60",
  "60"    = "1:60",
  "c3"    = "1:100",
  "100"   = "1:100",
  "c4"    = "1:140",
  "140"   = "1:140",
  "c5"    = "1:180",
  "180"   = "1:180",
  "c6"    = "1:220",
  "220"   = "1:220",
  "r1"    = "786 n1",
  "r2"    = "786 n2",
  "r3"    = "786 n3",
  "786-0" = "786-0 (n=3)",
  "r4"    = "VHL n1",
  "r5"    = "VHL n2",
  "r6"    = "VHL n3",
  "VHL"   = "786-0/VHL (n=3)"
)
```

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/RtmpMzOVd5/file40691c39b492/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	2.2.0
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")
)
```

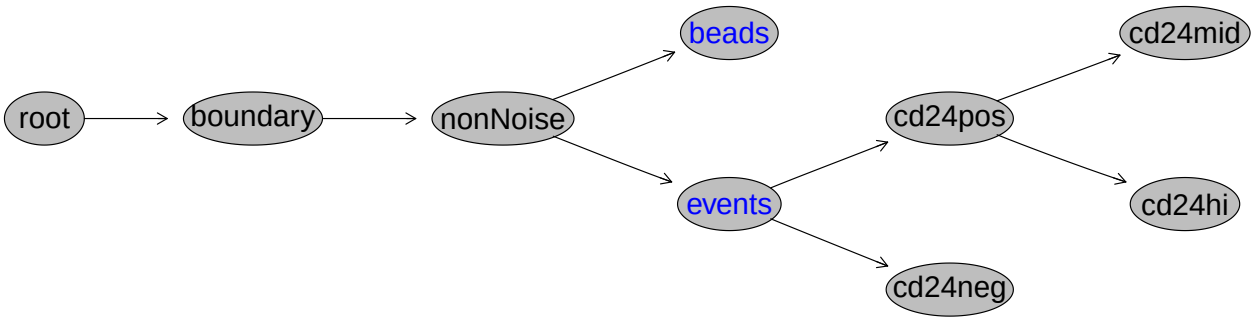
```
flowWorkspace::gs_pop_add(
  gs=gatingset,
  gate=flowWorkspace::booleanFilter(`beadsA & beadsB`),
  name="beads",
  parent="nonNoise"
)

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

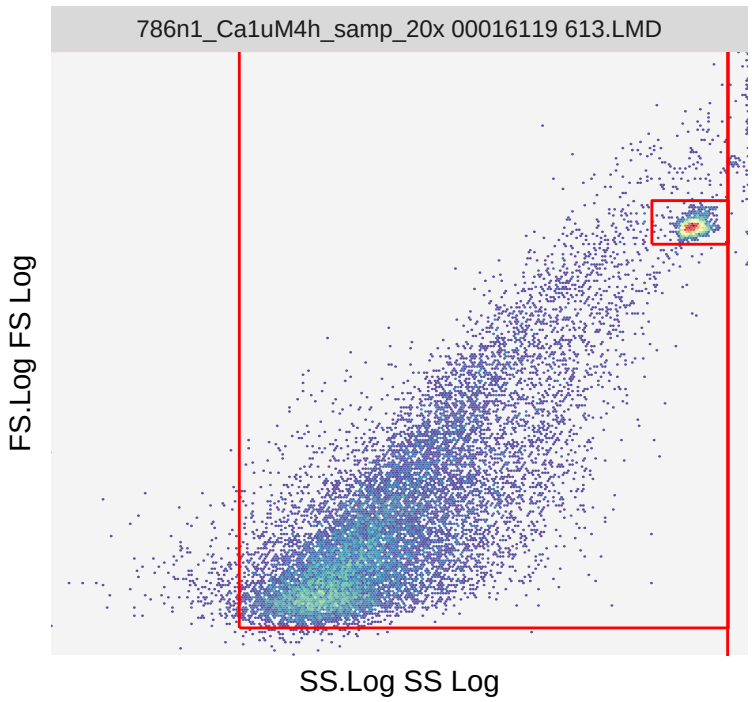
```
## Estimating landmarks for channel FS.Log ...Estimating landmarks for channel SS.Log ...
## Registering curves for parameter FS.Log ...
## Registering curves for parameter SS.Log ...
## Estimating landmarks for channel FS.Log ...Estimating landmarks for channel SS.Log ...
## Registering curves for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



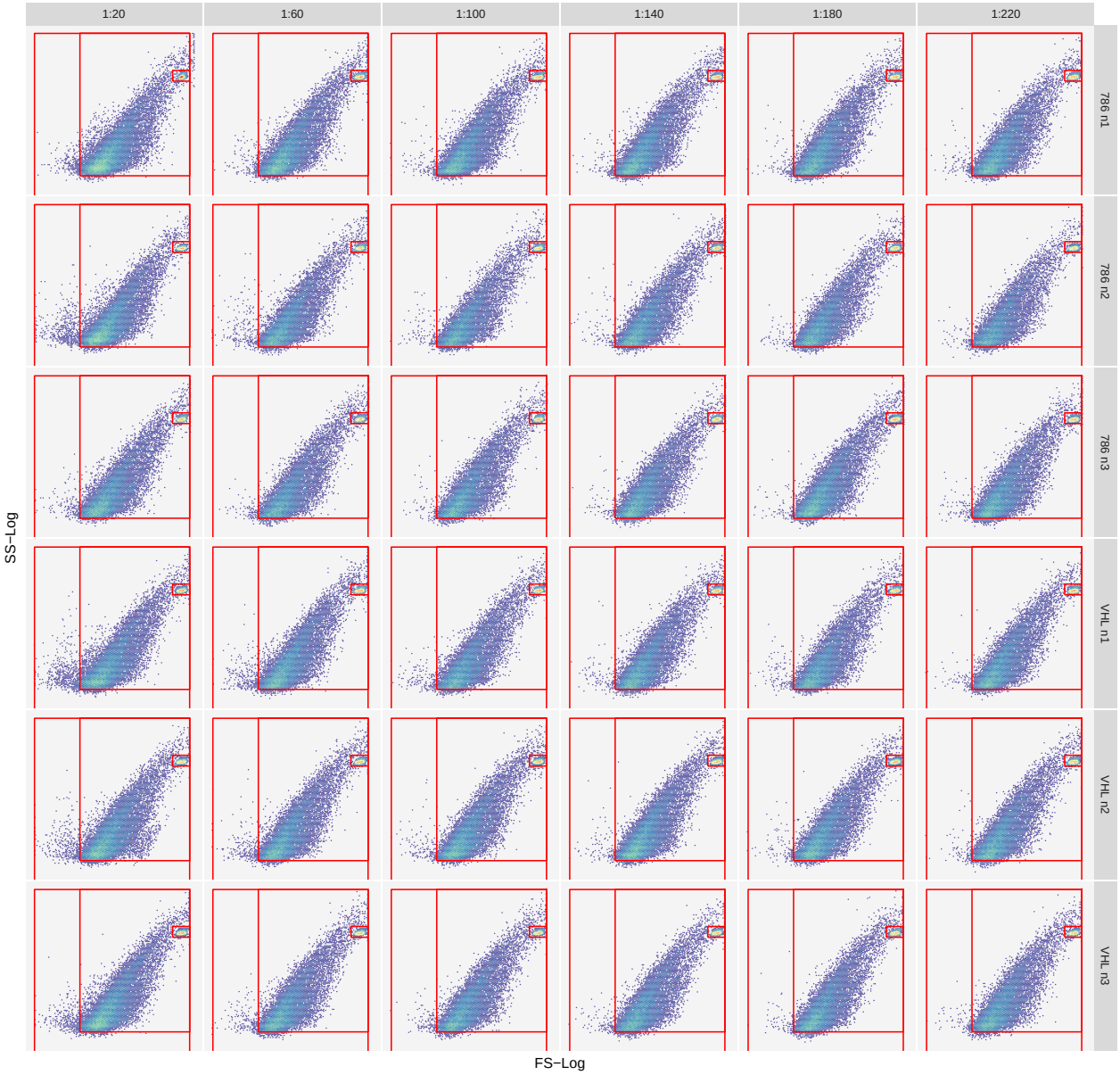
3.2 Whole experiment morphology plot

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
)
```

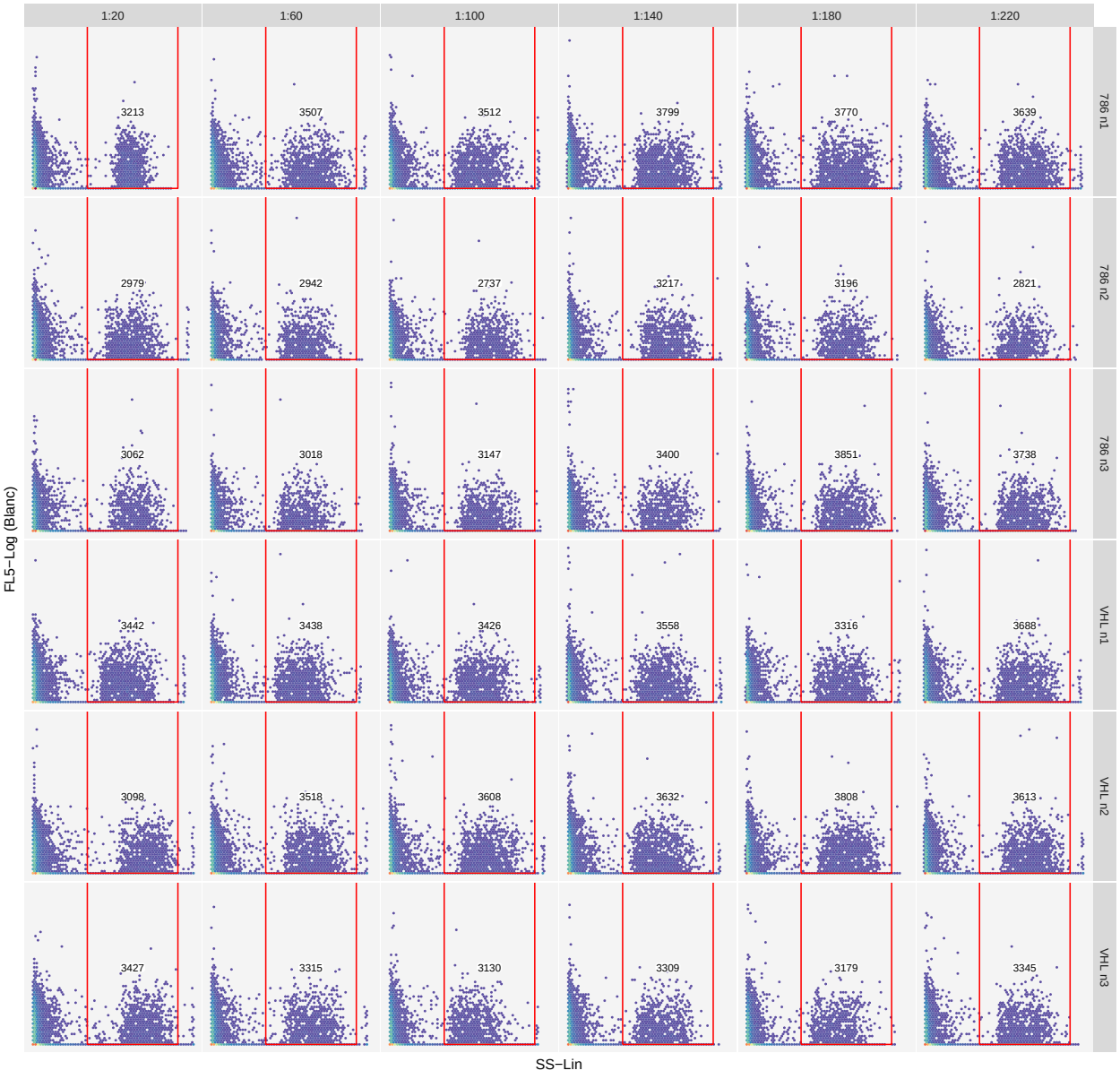
```
## cloning the gatingSet...
## Normalize beadsB
## Estimating landmarks for channel SS.Lin ...
## Registering curves for parameter SS.Lin ...
## normalizing sample 1
## normalizing sample 2
## normalizing sample 3
## normalizing sample 4
## normalizing sample 5
## normalizing sample 6
## normalizing 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



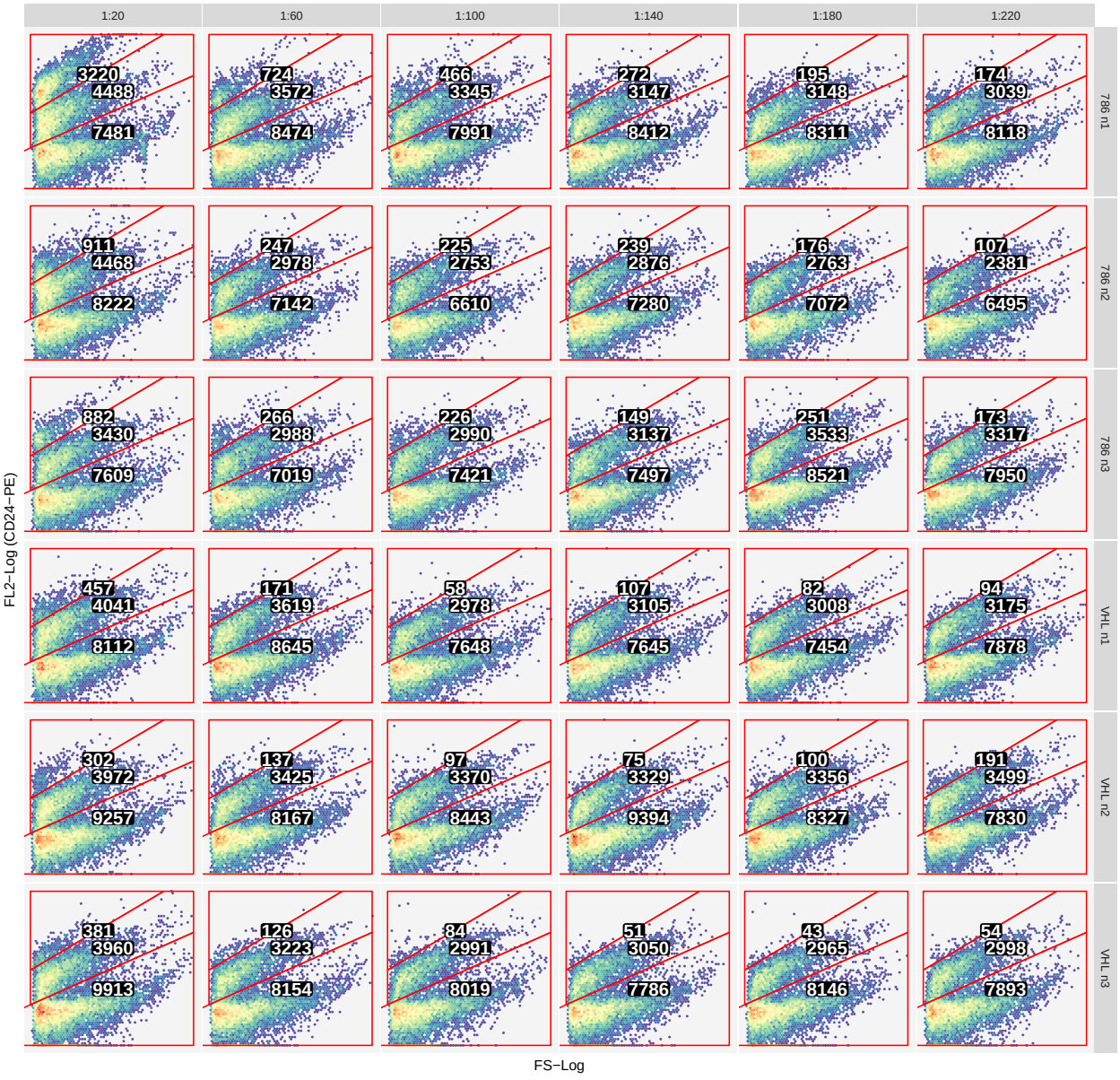
3.4 Whole experiment fluorescence with event counts

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.

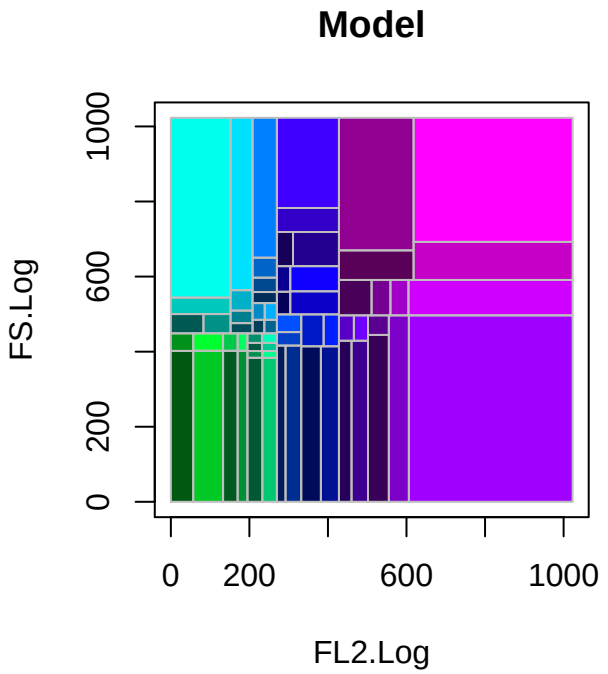
References:

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

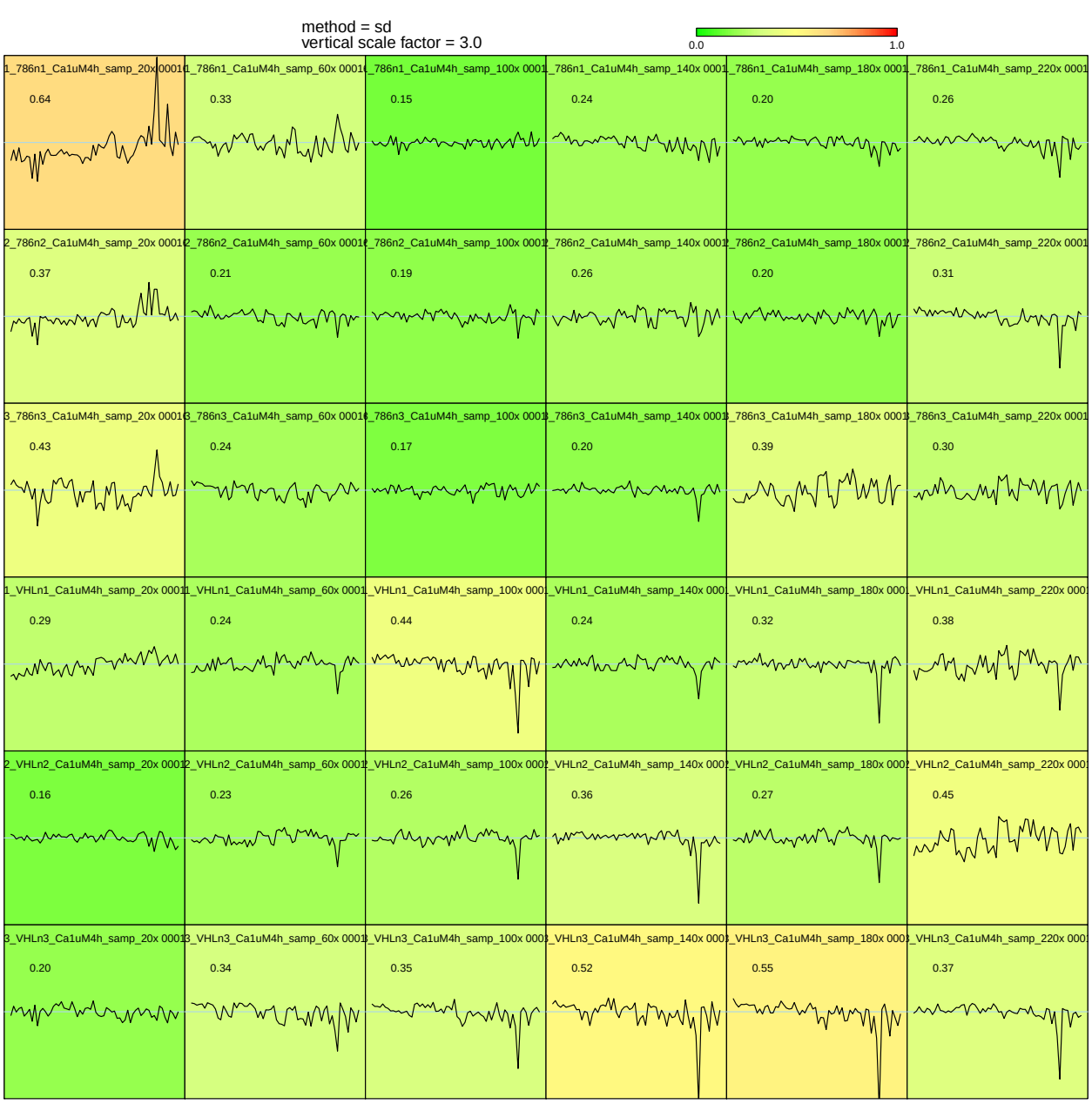
Technical documentation:

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

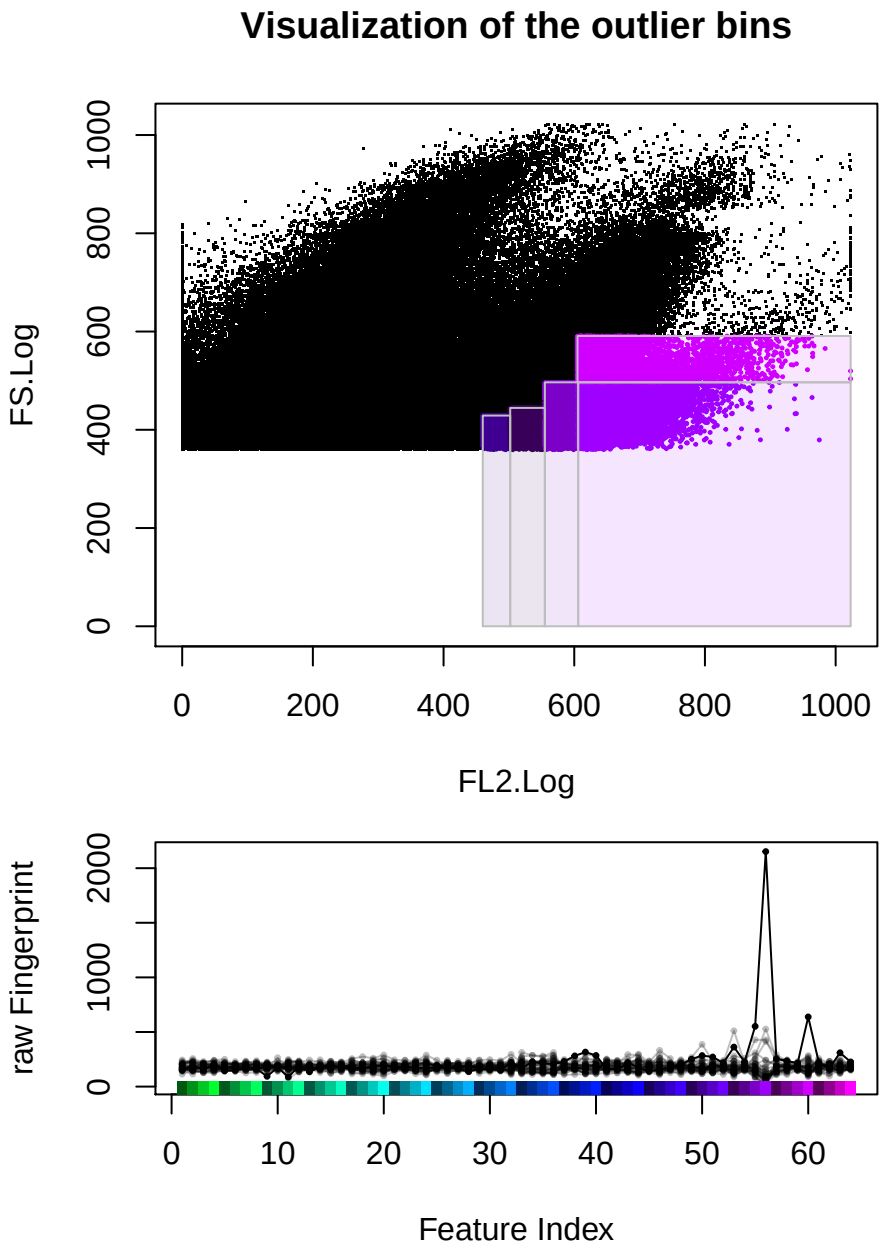
```
# Remove all irrelevant populations, such as beads and noise
fingerprintingData <- flowWorkspace::gs_pop_get_data(gatingset, "events")
fingerprintingData <- flowWorkspace::cytoset_to_flowSet(fingerprintingData)

# 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

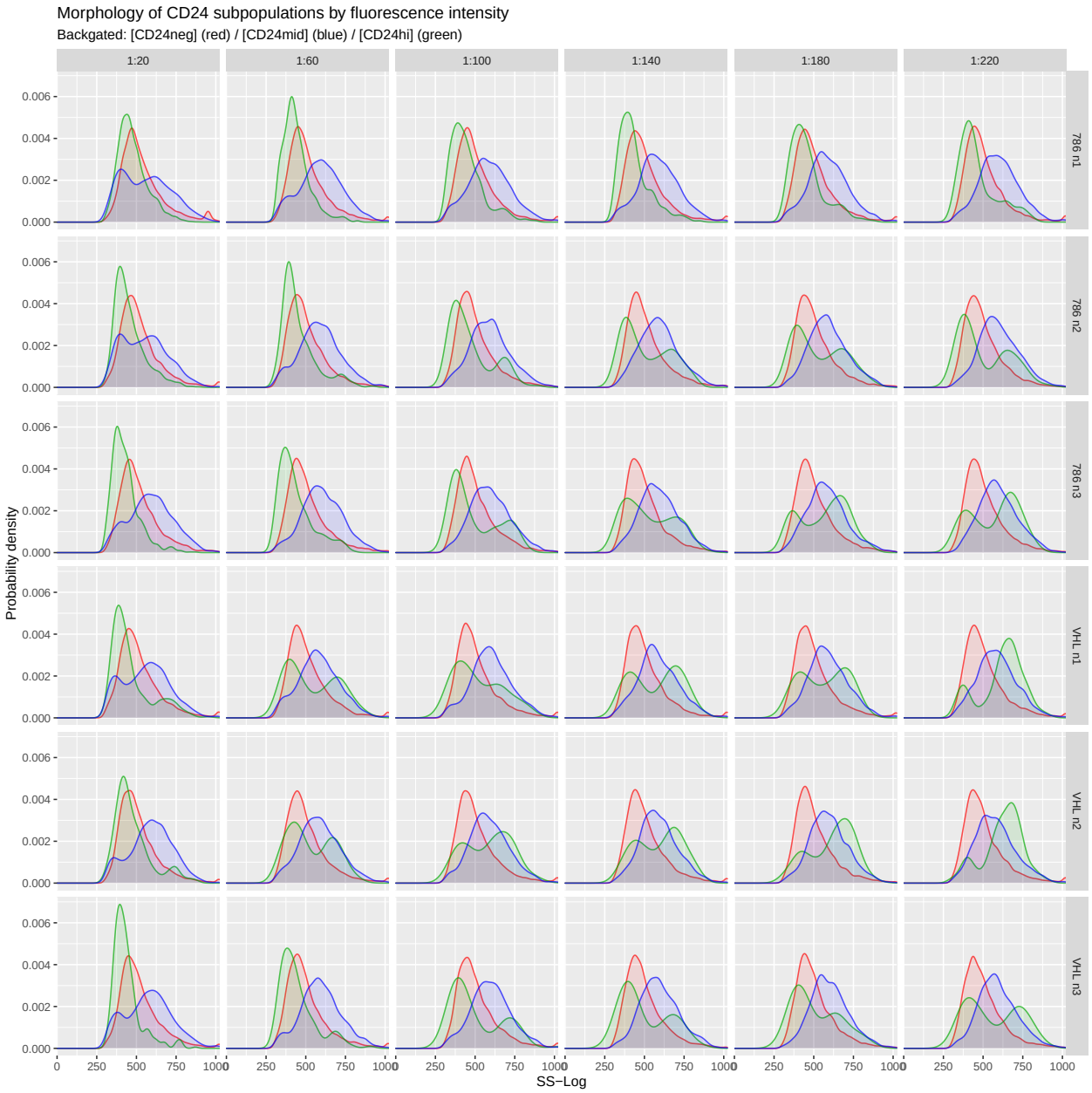
DECONVOLUTION TOOLS:

- 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:

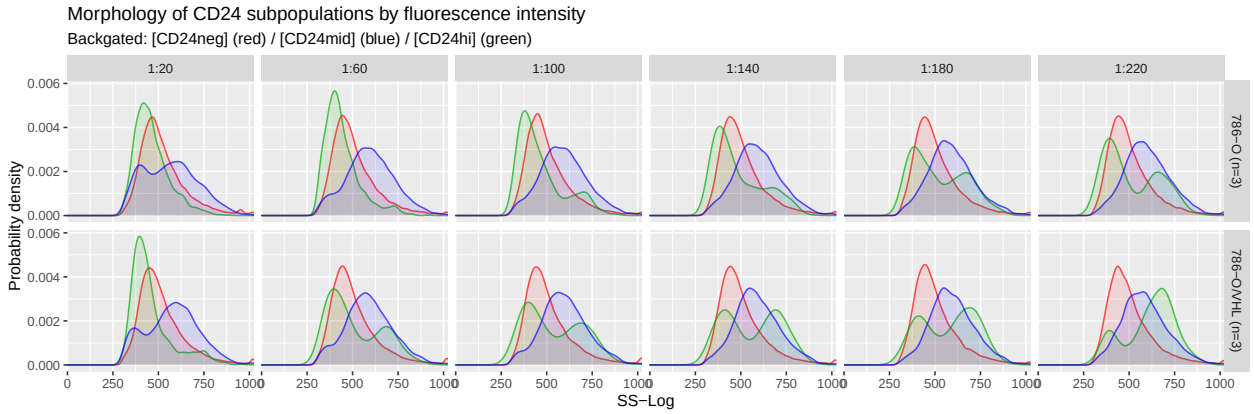
```
# Morphology characteristics of fluorescence-gated data can be visualized to
# gain quantitative insight into the magnitude of per-subpopulation swarm effect
ggplot2::ggplot(
  data=flowWorkspace::gs_pop_get_data(gatingset, "events"),
  mapping=ggplot2::aes(x=SS.Log)
) +
  ggplot2::labs(
    title="Morphology 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))
```


Morphology of CD24 subpopulations by fluorescence intensity
Backgated: [CD24neg] (red) / [CD24mid] (blue) / [CD24hi] (green)
1:20
1:60
1:100
1:140
1:180
1:220
786 n1
786 n2
786 n3
VHL n1
VHL n2
VHL n3
Probability density
0.006
0.004
0.002
0.000
0
250
500
750
1000
SS−Log

It might be easier to visualize this by combining data from all triplicates:

```
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(facetLabels))
```



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= 55
## number of iterations= 58
## number of iterations= 114
## number of iterations= 72
## number of iterations= 146
## number of iterations= 1879
## number of iterations= 58
## number of iterations= 54
## number of iterations= 2360
## number of iterations= 1529
## number of iterations= 1471
## number of iterations= 1607
## number of iterations= 55
## number of iterations= 211
## number of iterations= 1281
## number of iterations= 1080
## number of iterations= 1886
## number of iterations= 2908
## number of iterations= 52
## number of iterations= 66
## number of iterations= 56
## number of iterations= 1410
## number of iterations= 1088
## number of iterations= 1117
## number of iterations= 51
## number of iterations= 1710
## number of iterations= 1826
## number of iterations= 1742
## number of iterations= 4740
## number of iterations= 746
## number of iterations= 44
## number of iterations= 63
## number of iterations= 69
## number of iterations= 2414
## number of iterations= 4251
## number of iterations= 1361
```

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
)
```

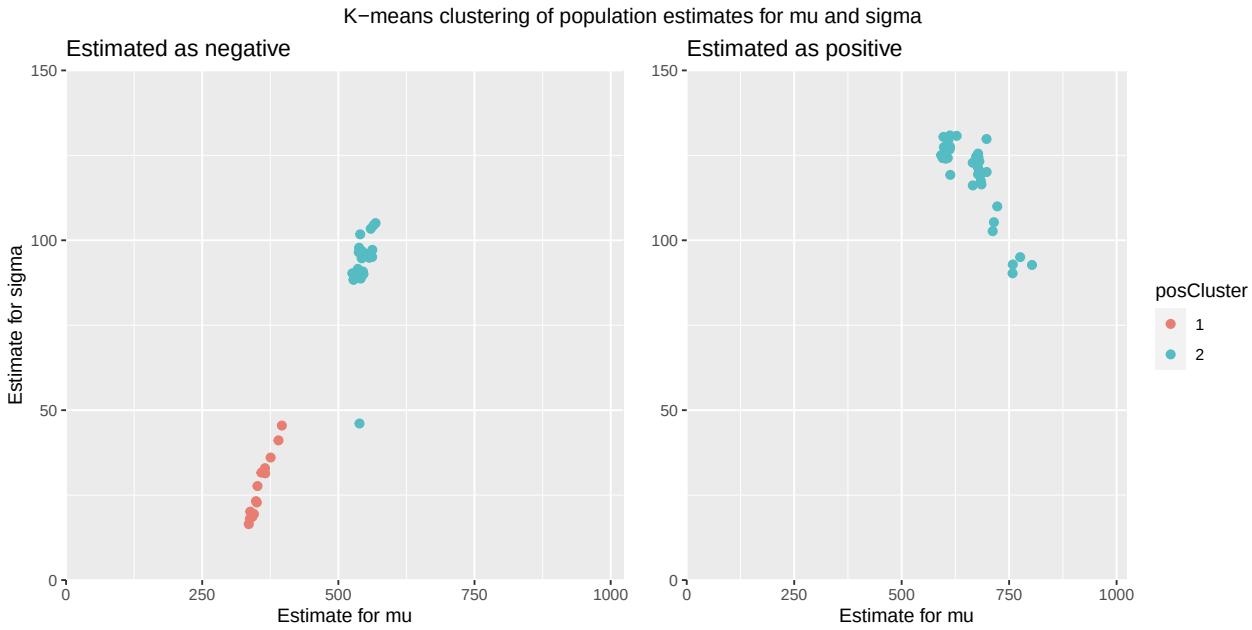
Table 3: Swarm deconvolution model: initial ML estimate on a per-sample basis

id	muNeg	sigmaNeg	pNeg	muPos	sigmaPos	pPos	hartigans	nNeg	nPos	welchTTest	facetRow	facetCol
1	396.2502	45.47152	0.2786945	627.9248	130.73895	0.7213055	0.0016110	836	2162	0.6186290	r1	c1
2	365.6965	31.41804	0.0695374	611.6603	126.74413	0.9304626	0.8431423	208	2790	0.7890223	r1	c2
3	342.1689	18.63245	0.0237496	597.3301	130.42876	0.9762504	0.9904091	71	2927	0.8046048	r1	c3
4	338.2744	20.14276	0.0195230	597.2075	125.27363	0.9804770	0.8238249	59	2939	0.8799453	r1	c4
5	335.5144	16.46352	0.0114226	595.4374	124.19694	0.9885774	0.9452544	34	2964	0.9053034	r1	c5
6	568.3445	105.04887	0.7503598	697.4892	129.81055	0.2496402	0.9157378	2250	748	0.1960943	r1	c6
7	390.0803	41.11502	0.2338329	600.0276	124.48319	0.7661671	0.0000530	701	2297	0.6370340	r2	c1
8	350.2706	22.82872	0.0349920	597.7028	125.29143	0.9650080	0.9639120	105	2893	0.8352026	r2	c2
9	564.5227	104.37868	0.8641998	775.7457	95.03789	0.1358002	0.8424461	2591	407	0.4256454	r2	c3
10	548.2789	95.97593	0.6961261	697.7414	120.09558	0.3038739	0.7689191	2087	911	0.2881748	r2	c4
11	556.6123	94.94855	0.8173548	758.1529	90.27652	0.1826452	0.9804892	2450	548	0.5129443	r2	c5
12	541.0980	88.72532	0.5796583	684.1912	117.44337	0.4203417	0.9791798	1738	1260	0.3312157	r2	c6
13	375.6986	36.05992	0.0924091	612.5806	127.51757	0.9075909	0.8959220	277	2721	0.7371015	r3	c1
14	538.8935	46.07318	0.0894823	612.4264	130.85571	0.9105177	0.8371142	268	2730	0.2086390	r3	c2
15	535.8466	91.58677	0.5656354	677.7137	119.41408	0.4343646	0.9258856	1696	1302	0.3190955	r3	c3
16	527.7733	88.34558	0.5245290	665.5281	116.14625	0.4754710	0.9131216	1573	1425	0.3376675	r3	c4
17	559.5072	103.38998	0.8999171	803.2024	92.73437	0.1000829	0.8848766	2698	300	0.4534181	r3	c5
18	546.1493	90.07899	0.6949456	711.8542	102.68044	0.3050544	0.9078441	2083	915	0.4235007	r3	c6
19	358.6582	31.61332	0.1609293	598.2759	127.41527	0.8390707	0.0018243	482	2516	0.7331244	r4	c1
20	348.7909	23.21436	0.0367610	591.9374	125.03769	0.9632390	0.8692413	110	2888	0.8230282	r4	c2
21	337.4828	18.00970	0.0184091	607.3286	124.23067	0.9815909	0.9922233	55	2943	0.9375205	r4	c3
22	538.1248	97.80426	0.6452778	678.6645	124.46502	0.3547222	0.9746102	1935	1063	0.2689527	r4	c4
23	542.9172	94.69995	0.6254988	673.4695	124.63933	0.3745012	0.9901248	1875	1123	0.2571261	r4	c5
24	537.4444	96.47953	0.6087724	665.4303	122.83264	0.3912276	0.6369635	1825	1173	0.2604128	r4	c6
25	351.5246	27.66860	0.0842575	613.2109	119.25196	0.9157425	0.6555798	253	2745	0.9492898	r5	c1
26	540.1524	101.75440	0.6448460	677.8515	125.51204	0.3551540	0.9656689	1933	1065	0.2563739	r5	c2
27	543.7558	96.76152	0.6440762	678.0264	121.22796	0.3559238	0.9897836	1931	1067	0.2697230	r5	c3
28	545.2359	90.79946	0.6670099	685.6661	116.44941	0.3329901	0.9171138	2000	998	0.3006023	r5	c4
29	557.0064	94.89099	0.7557928	714.6027	105.31809	0.2442072	0.9388683	2266	732	0.3490487	r5	c5
30	525.7306	90.28833	0.6552074	680.7265	123.23784	0.3447926	0.7701094	1964	1034	0.3057922	r5	c6
31	365.2591	32.92285	0.1306669	608.7117	129.57235	0.8693331	0.0496624	392	2606	0.7301999	r6	c1
32	341.3929	19.06819	0.0243098	602.4943	124.01819	0.9756902	0.9569248	73	2925	0.9080139	r6	c2
33	344.9372	19.43547	0.0136227	603.8282	124.17507	0.9863773	0.9763759	41	2957	0.8938668	r6	c3
34	562.3668	97.19095	0.8285733	758.5041	92.87095	0.1714267	0.7710601	2484	514	0.4614668	r6	c4
35	562.0382	95.08169	0.7643500	722.4071	109.97662	0.2356500	0.7564614	2292	706	0.3254166	r6	c5
36	543.5745	89.65852	0.6475730	684.1401	119.42593	0.3524270	0.9707170	1941	1057	0.2958504	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),
        .groups = "drop"
      )

  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),
        .groups = "drop"
      )

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

estimatedParameters <- estimateGlobalParameters(model=cd24midSwarmModel)

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

Table 4: Estimated parameters

	mu	sigma
neg	372.1273	35.25478
pos	608.0461	127.09155

```
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	372.1273	35.25478	0.2786945	608.0461	127.0915	0.7213055	0.0016110	836	2162	0.6186290	r1	c1	1	2
2	372.1273	35.25478	0.0695374	608.0461	127.0915	0.9304626	0.8431423	208	2790	0.7890223	r1	c2	1	2
3	372.1273	35.25478	0.0237496	608.0461	127.0915	0.9762504	0.9904091	71	2927	0.8046048	r1	c3	1	2
4	372.1273	35.25478	0.0195230	608.0461	127.0915	0.9804770	0.8238249	59	2939	0.8799453	r1	c4	1	2
5	372.1273	35.25478	0.0114226	608.0461	127.0915	0.9885774	0.9452544	34	2964	0.9053034	r1	c5	1	2
6	372.1273	35.25478	0.7503598	608.0461	127.0915	0.2496402	0.9157378	2250	748	0.1960943	r1	c6	2	2
7	372.1273	35.25478	0.2338329	608.0461	127.0915	0.7661671	0.0000530	701	2297	0.6370340	r2	c1	1	2
8	372.1273	35.25478	0.0349920	608.0461	127.0915	0.9650080	0.9639120	105	2893	0.8352026	r2	c2	1	2
9	372.1273	35.25478	0.8641998	608.0461	127.0915	0.1358002	0.8424461	2591	407	0.4256454	r2	c3	2	2
10	372.1273	35.25478	0.6961261	608.0461	127.0915	0.3038739	0.7689191	2087	911	0.2881748	r2	c4	2	2
11	372.1273	35.25478	0.8173548	608.0461	127.0915	0.1826452	0.9804892	2450	548	0.5129443	r2	c5	2	2
12	372.1273	35.25478	0.5796583	608.0461	127.0915	0.4203417	0.9791798	1738	1260	0.3312157	r2	c6	2	2
13	372.1273	35.25478	0.0924091	608.0461	127.0915	0.9075909	0.8959220	277	2721	0.7371015	r3	c1	1	2
14	372.1273	35.25478	0.0894823	608.0461	127.0915	0.9105177	0.8371142	268	2730	0.2086390	r3	c2	2	2
15	372.1273	35.25478	0.5656354	608.0461	127.0915	0.4343646	0.9258856	1696	1302	0.3190955	r3	c3	2	2
16	372.1273	35.25478	0.5245290	608.0461	127.0915	0.4754710	0.9131216	1573	1425	0.3376675	r3	c4	2	2
17	372.1273	35.25478	0.8999171	608.0461	127.0915	0.1000829	0.8848766	2698	300	0.4534181	r3	c5	2	2
18	372.1273	35.25478	0.6949456	608.0461	127.0915	0.3050544	0.9078441	2083	915	0.4235007	r3	c6	2	2
19	372.1273	35.25478	0.1609293	608.0461	127.0915	0.8390707	0.0018243	482	2516	0.7331244	r4	c1	1	2
20	372.1273	35.25478	0.0367610	608.0461	127.0915	0.9632390	0.8692413	110	2888	0.8230282	r4	c2	1	2
21	372.1273	35.25478	0.0184091	608.0461	127.0915	0.9815909	0.9922233	55	2943	0.9375205	r4	c3	1	2
22	372.1273	35.25478	0.6452778	608.0461	127.0915	0.3547222	0.9746102	1935	1063	0.2689527	r4	c4	2	2
23	372.1273	35.25478	0.6254988	608.0461	127.0915	0.3745012	0.9901248	1875	1123	0.2571261	r4	c5	2	2
24	372.1273	35.25478	0.6087724	608.0461	127.0915	0.3912276	0.6369635	1825	1173	0.2604128	r4	c6	2	2
25	372.1273	35.25478	0.0842575	608.0461	127.0915	0.9157425	0.6555798	253	2745	0.9492898	r5	c1	1	2
26	372.1273	35.25478	0.6448460	608.0461	127.0915	0.3551540	0.9656689	1933	1065	0.2563739	r5	c2	2	2
27	372.1273	35.25478	0.6440762	608.0461	127.0915	0.3559238	0.9897836	1931	1067	0.2697230	r5	c3	2	2
28	372.1273	35.25478	0.6670099	608.0461	127.0915	0.3329901	0.9171138	2000	998	0.3006023	r5	c4	2	2
29	372.1273	35.25478	0.7557928	608.0461	127.0915	0.2442072	0.9388683	2266	732	0.3490487	r5	c5	2	2
30	372.1273	35.25478	0.6552074	608.0461	127.0915	0.3447926	0.7701094	1964	1034	0.3057922	r5	c6	2	2
31	372.1273	35.25478	0.1306669	608.0461	127.0915	0.8693331	0.0496624	392	2606	0.7301999	r6	c1	1	2
32	372.1273	35.25478	0.0243098	608.0461	127.0915	0.9756902	0.9569248	73	2925	0.9080139	r6	c2	1	2
33	372.1273	35.25478	0.0136227	608.0461	127.0915	0.9863773	0.9763759	41	2957	0.8938668	r6	c3	1	2
34	372.1273	35.25478	0.8285733	608.0461	127.0915	0.1714267	0.7710601	2484	514	0.4614668	r6	c4	2	2
35	372.1273	35.25478	0.7643500	608.0461	127.0915	0.2356500	0.7564614	2292	706	0.3254166	r6	c5	2	2
36	372.1273	35.25478	0.6475730	608.0461	127.0915	0.3524270	0.9707170	1941	1057	0.2958504	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	372.1273	35.25478	0.2069882	608.0461	127.0915	0.7930118	0.0016110	621	2377	0.6186290	r1	c1	1	2	2998
2	372.1273	35.25478	0.0701490	608.0461	127.0915	0.9298510	0.8431423	210	2788	0.7890223	r1	c2	1	2	2998
3	372.1273	35.25478	0.0480505	608.0461	127.0915	0.9519495	0.9904091	144	2854	0.8046048	r1	c3	1	2	2998
4	372.1273	35.25478	0.0309369	608.0461	127.0915	0.9690631	0.8238249	93	2905	0.8799453	r1	c4	1	2	2998
5	372.1273	35.25478	0.0216558	608.0461	127.0915	0.9783442	0.9452544	65	2933	0.9053034	r1	c5	1	2	2998
6	372.1273	35.25478	0.0064874	608.0461	127.0915	0.9935126	0.9157378	19	2979	0.1960943	r1	c6	2	2	2998
7	372.1273	35.25478	0.2050072	608.0461	127.0915	0.7949928	0.0000530	615	2383	0.6370340	r2	c1	1	2	2998
8	372.1273	35.25478	0.0485236	608.0461	127.0915	0.9514764	0.9639120	145	2853	0.8352026	r2	c2	1	2	2998
9	372.1273	35.25478	0.0196519	608.0461	127.0915	0.9803481	0.8424461	59	2939	0.4256454	r2	c3	2	2	2998
10	372.1273	35.25478	0.0098448	608.0461	127.0915	0.9901552	0.7689191	30	2968	0.2881748	r2	c4	2	2	2998
11	372.1273	35.25478	0.0019706	608.0461	127.0915	0.9980294	0.9804892	6	2992	0.5129443	r2	c5	2	2	2998
12	372.1273	35.25478	0.0003010	608.0461	127.0915	0.9996990	0.9791798	1	2997	0.3312157	r2	c6	2	2	2998
13	372.1273	35.25478	0.0855947	608.0461	127.0915	0.9144053	0.8959220	257	2741	0.7371015	r3	c1	1	2	2998
14	372.1273	35.25478	0.0082531	608.0461	127.0915	0.9917469	0.8371142	25	2973	0.2086390	r3	c2	2	2	2998
15	372.1273	35.25478	0.0044363	608.0461	127.0915	0.9955637	0.9258856	13	2985	0.3190955	r3	c3	2	2	2998
16	372.1273	35.25478	0.0101301	608.0461	127.0915	0.9898699	0.9131216	30	2968	0.3376675	r3	c4	2	2	2998
17	372.1273	35.25478	0.0272748	608.0461	127.0915	0.9727252	0.8848766	82	2916	0.4534181	r3	c5	2	2	2998
18	372.1273	35.25478	0.0009867	608.0461	127.0915	0.9990133	0.9078441	3	2995	0.4235007	r3	c6	2	2	2998
19	372.1273	35.25478	0.1854264	608.0461	127.0915	0.8145736	0.0018243	556	2442	0.7331244	r4	c1	1	2	2998
20	372.1273	35.25478	0.0591501	608.0461	127.0915	0.9408499	0.8692413	177	2821	0.8230282	r4	c2	1	2	2998
21	372.1273	35.25478	0.0164812	608.0461	127.0915	0.9835188	0.9922233	49	2949	0.9375205	r4	c3	1	2	2998
22	372.1273	35.25478	0.0284215	608.0461	127.0915	0.9715785	0.9746102	85	2913	0.2689527	r4	c4	2	2	2998
23	372.1273	35.25478	0.0112362	608.0461	127.0915	0.9887638	0.9901248	34	2964	0.2571261	r4	c5	2	2	2998
24	372.1273	35.25478	0.0256813	608.0461	127.0915	0.9743187	0.6369635	77	2921	0.2604128	r4	c6	2	2	2998
25	372.1273	35.25478	0.0818582	608.0461	127.0915	0.9181418	0.6555798	245	2753	0.9492898	r5	c1	1	2	2998
26	372.1273	35.25478	0.0317052	608.0461	127.0915	0.9682948	0.9656689	95	2903	0.2563739	r5	c2	2	2	2998
27	372.1273	35.25478	0.0119778	608.0461	127.0915	0.9880222	0.9897836	36	2962	0.2697230	r5	c3	2	2	2998
28	372.1273	35.25478	0.0021962	608.0461	127.0915	0.9978038	0.9171138	7	2991	0.3006023	r5	c4	2	2	2998
29	372.1273	35.25478	0.0001912	608.0461	127.0915	0.9998088	0.9388683	1	2997	0.3490487	r5	c5	2	2	2998
30	372.1273	35.25478	0.0288335	608.0461	127.0915	0.9711665	0.7701094	86	2912	0.3057922	r5	c6	2	2	2998
31	372.1273	35.25478	0.1396307	608.0461	127.0915	0.8603693	0.0496624	419	2579	0.7301999	r6	c1	1	2	2998

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	372.1273	35.25478	0.0267785	608.0461	127.0915	0.9732215	0.9569248	80	2918	0.9080139	r6	c2	1	2	2998
33	372.1273	35.25478	0.0150539	608.0461	127.0915	0.9849461	0.9763759	45	2953	0.8938668	r6	c3	1	2	2998
34	372.1273	35.25478	0.0000874	608.0461	127.0915	0.9999126	0.7710601	0	2998	0.4614668	r6	c4	2	2	2998
35	372.1273	35.25478	0.0000013	608.0461	127.0915	0.9999987	0.7564614	0	2998	0.3254166	r6	c5	2	2	2998
36	372.1273	35.25478	0.0004375	608.0461	127.0915	0.9995625	0.9707170	1	2997	0.2958504	r6	c6	2	2	2998

4.2.2.4 Plotting the solutions

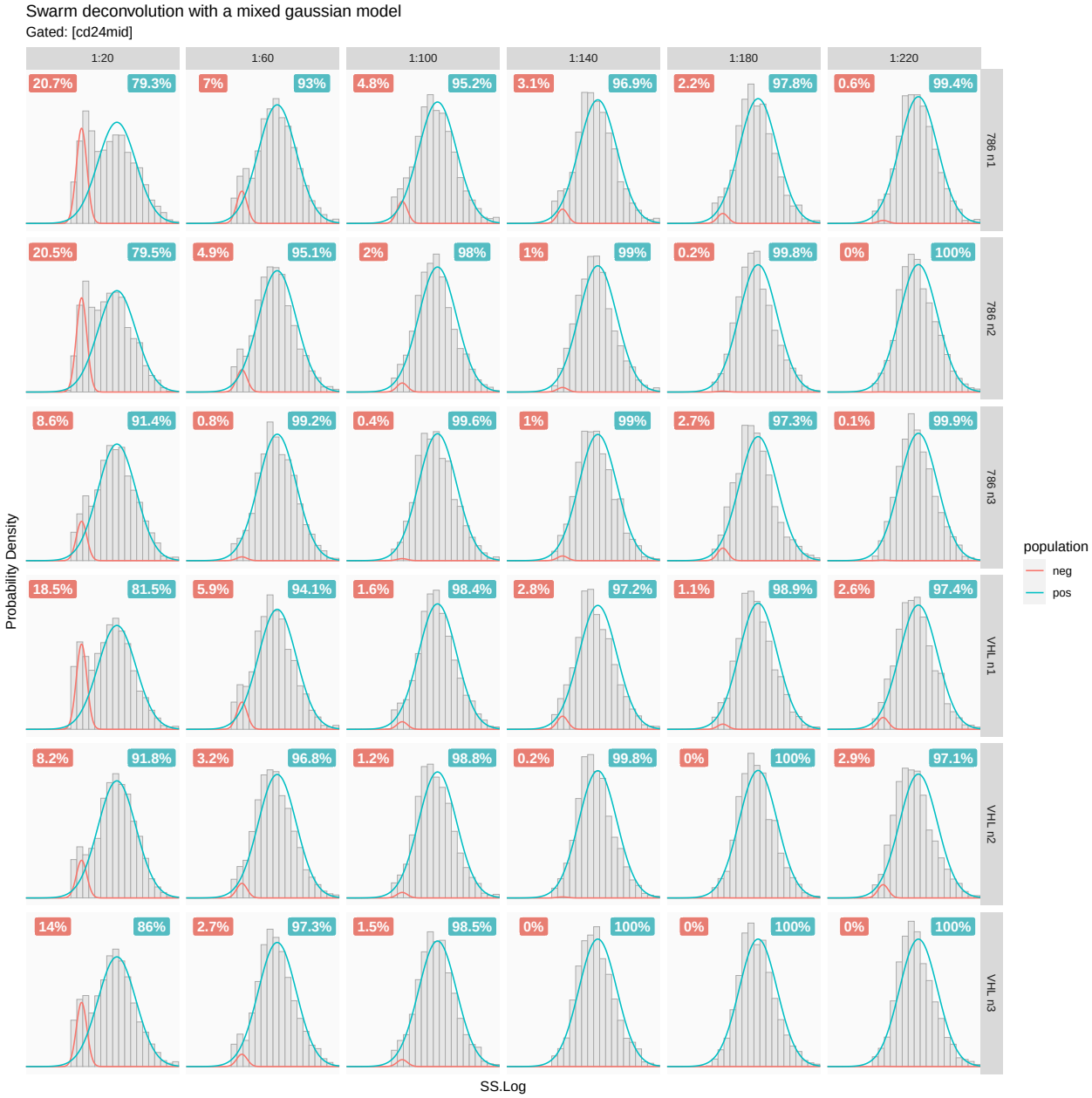
Now that we have obtained the optimal solution for our data set, we can create model distributions for use in plotting:

```
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).
```



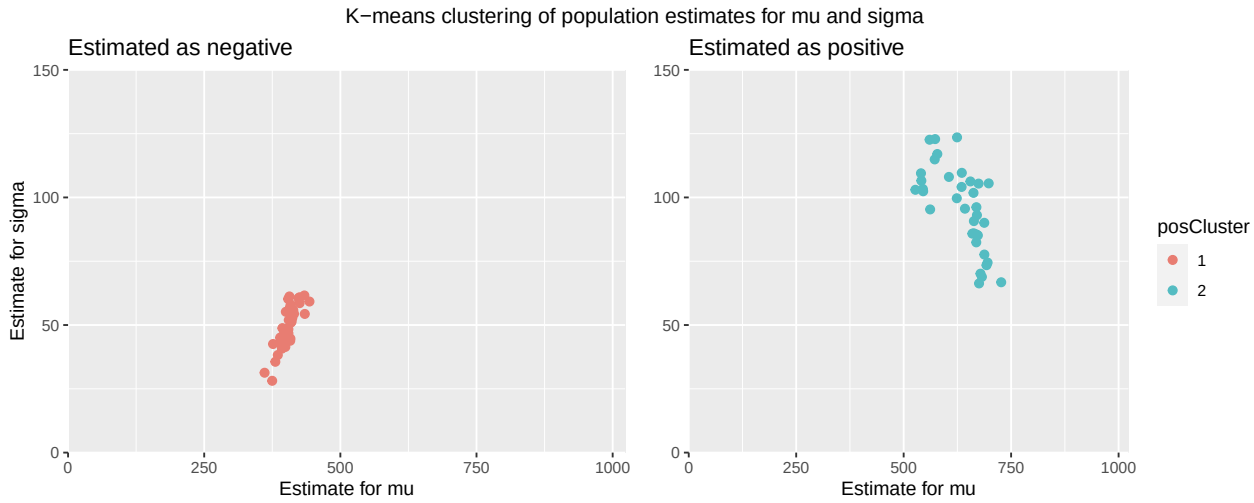
4.2.3 Putting it all together: CD24hi swarm deconvolution

```
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= 87
## number of iterations= 81
## number of iterations= 100
## number of iterations= 54
## number of iterations= 96
## number of iterations= 54
## number of iterations= 95
## number of iterations= 75
## number of iterations= 91
## number of iterations= 49
## number of iterations= 62
## number of iterations= 19
## number of iterations= 82
## number of iterations= 69
## number of iterations= 144
## number of iterations= 67
## number of iterations= 34
## number of iterations= 19
## number of iterations= 45
## number of iterations= 31
## number of iterations= 48
## number of iterations= 16
## number of iterations= 32
## number of iterations= 11
## number of iterations= 240
## number of iterations= 29
## number of iterations= 34
## number of iterations= 37
## number of iterations= 36
## number of iterations= 16
## number of iterations= 27
## number of iterations= 61
## number of iterations= 79
## number of iterations= 6
## number of iterations= 29
## number of iterations= 36
```

```
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]

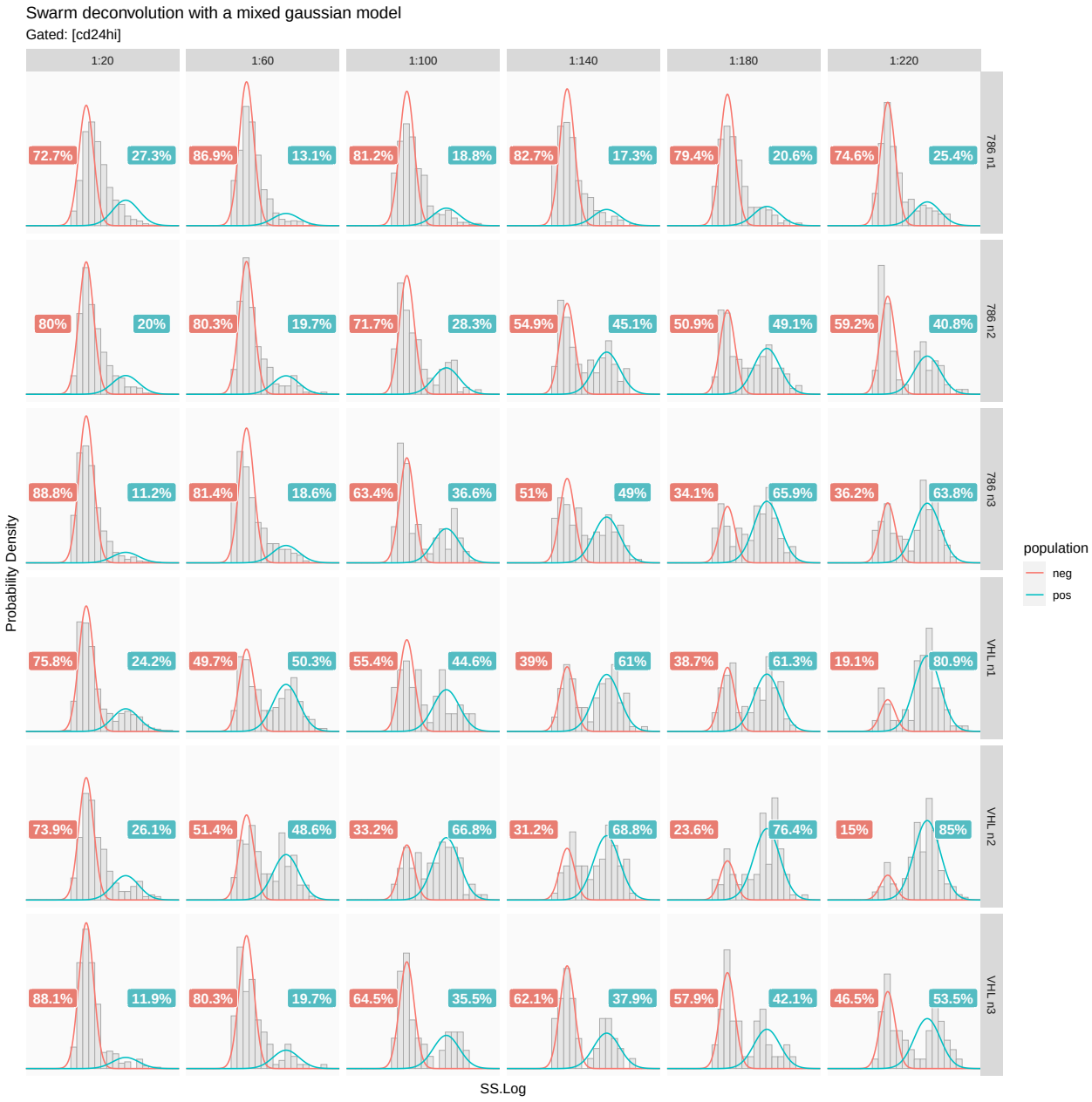
id	muNeg	sigmaNeg	pNeg	muPos	sigmaPos	pPos	hartigans	nNeg	nPos	welchTTest	facetRow	facetCol	negCluster	posCluster	nTot
1	403.0371	49.24017	0.7273027	666.712	87.27683	0.2726973	0.4342748	39	15	0.0577781	r1	c1	1	2	54
2	403.0371	49.24017	0.8693634	666.712	87.27683	0.1306366	0.9191252	47	7	0.0418458	r1	c2	1	2	54
3	403.0371	49.24017	0.8123510	666.712	87.27683	0.1876490	0.9924966	44	10	0.0441473	r1	c3	1	2	54
4	403.0371	49.24017	0.8267906	666.712	87.27683	0.1732094	0.9821544	45	9	0.0546780	r1	c4	1	2	54
5	403.0371	49.24017	0.7940713	666.712	87.27683	0.2059287	0.9923969	43	11	0.0585483	r1	c5	1	2	54
6	403.0371	49.24017	0.7457573	666.712	87.27683	0.2542427	0.9932503	40	14	0.0871630	r1	c6	1	2	54
7	403.0371	49.24017	0.8000086	666.712	87.27683	0.1999914	0.9105907	43	11	0.0551437	r2	c1	1	2	54
8	403.0371	49.24017	0.8026220	666.712	87.27683	0.1973780	0.9945522	43	11	0.0466224	r2	c2	1	2	54
9	403.0371	49.24017	0.7170207	666.712	87.27683	0.2829793	0.0899201	39	15	0.1323926	r2	c3	1	2	54
10	403.0371	49.24017	0.5486565	666.712	87.27683	0.4513435	0.1515512	30	24	0.1416740	r2	c4	1	2	54
11	403.0371	49.24017	0.5088559	666.712	87.27683	0.4911441	0.1195755	27	27	0.1218254	r2	c5	1	2	54
12	403.0371	49.24017	0.5918484	666.712	87.27683	0.4081516	0.0023661	32	22	0.1873880	r2	c6	1	2	54
13	403.0371	49.24017	0.8878478	666.712	87.27683	0.1121522	0.8378481	48	6	0.0433731	r3	c1	1	2	54
14	403.0371	49.24017	0.8144601	666.712	87.27683	0.1855399	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	49.24017	0.6344640	666.712	87.27683	0.3655360	0.0236009	34	20	0.1033086	r3	c3	1	2	54
16	403.0371	49.24017	0.5095356	666.712	87.27683	0.4904644	0.5332946	28	26	0.1414295	r3	c4	1	2	54
17	403.0371	49.24017	0.3407176	666.712	87.27683	0.6592824	0.0006891	18	36	0.1579794	r3	c5	1	2	54
18	403.0371	49.24017	0.3621814	666.712	87.27683	0.6378186	0.0051107	20	34	0.2355597	r3	c6	1	2	54
19	403.0371	49.24017	0.7580407	666.712	87.27683	0.2419593	0.9549641	41	13	0.0609301	r4	c1	1	2	54
20	403.0371	49.24017	0.4967608	666.712	87.27683	0.5032392	0.0243285	27	27	0.1689623	r4	c2	1	2	54
21	403.0371	49.24017	0.5537858	666.712	87.27683	0.4462142	0.6175997	30	24	0.1158097	r4	c3	1	2	54
22	403.0371	49.24017	0.3902395	666.712	87.27683	0.6097605	0.0080804	21	33	0.2393060	r4	c4	1	2	54
23	403.0371	49.24017	0.3867302	666.712	87.27683	0.6132698	0.0056401	21	33	0.1965286	r4	c5	1	2	54
24	403.0371	49.24017	0.1914880	666.712	87.27683	0.8085120	0.3538832	10	44	0.2603654	r4	c6	1	2	54
25	403.0371	49.24017	0.7394505	666.712	87.27683	0.2605495	0.9925139	40	14	0.0675004	r5	c1	1	2	54
26	403.0371	49.24017	0.5137010	666.712	87.27683	0.4862990	0.0061349	28	26	0.2209997	r5	c2	1	2	54
27	403.0371	49.24017	0.3321741	666.712	87.27683	0.6678259	0.2398218	18	36	0.1556186	r5	c3	1	2	54
28	403.0371	49.24017	0.3119163	666.712	87.27683	0.6880837	0.2860918	17	37	0.2059403	r5	c4	1	2	54
29	403.0371	49.24017	0.2362624	666.712	87.27683	0.7637376	0.2530825	13	41	0.1958969	r5	c5	1	2	54
30	403.0371	49.24017	0.1497176	666.712	87.27683	0.8502824	0.7357624	8	46	0.2186312	r5	c6	1	2	54
31	403.0371	49.24017	0.8810592	666.712	87.27683	0.1189408	0.9939827	48	6	0.0392643	r6	c1	1	2	54
32	403.0371	49.24017	0.8033035	666.712	87.27683	0.1966965	0.9546455	43	11	0.0487752	r6	c2	1	2	54
33	403.0371	49.24017	0.6448599	666.712	87.27683	0.3551401	0.0632619	35	19	0.1161000	r6	c3	1	2	54
34	403.0371	49.24017	0.6206471	666.712	87.27683	0.3793529	0.1258550	34	20	0.2565234	r6	c4	1	2	54
35	403.0371	49.24017	0.5794292	666.712	87.27683	0.4205708	0.3785653	31	23	0.1469772	r6	c5	1	2	54
36	403.0371	49.24017	0.4649794	666.712	87.27683	0.5350206	0.0095742	25	29	0.2754271	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(
```

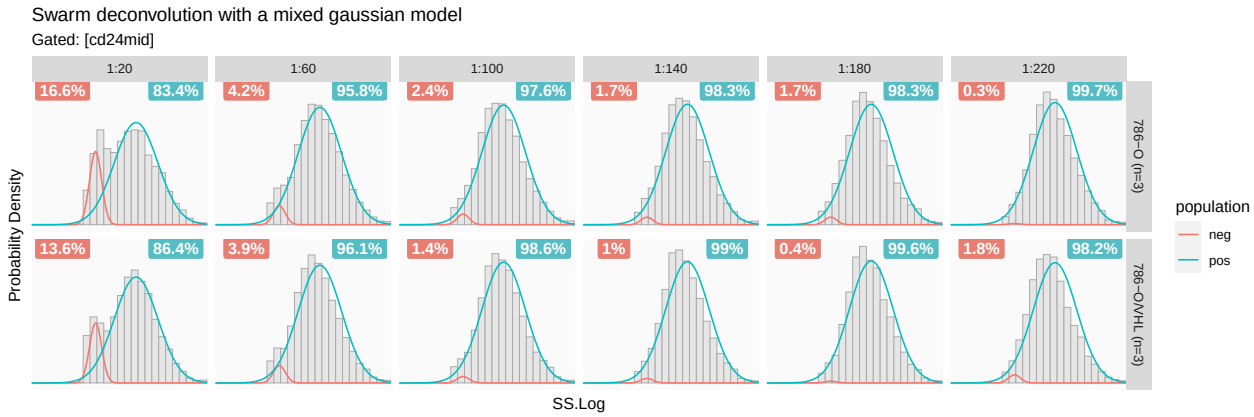
```
      x,
      mean=model[["muPos"]][i],
      sd=model[["sigmaPos"]][i]
    ) * model[["pPos"]][i]

    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),
      .groups  = "drop"
    )

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(facetLabels))
}

plotDeconvAggregateExperiment(
  gs=gatingset,
  model=cd24midSwarmModel,
  gate="cd24mid",
```

```
  axis="SS.Log"
)
```

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




```
plotDeconvAggregateExperiment(
  gs=gatingset,
  model=cd24hiSwarmModel,
  gate="cd24hi",
  axis="SS.Log"
)
```

```
## 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
sampleNames <- row.names(experiment)

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

# Change to camelCase
populationStatistics <- populationStatistics %>%
  dplyr::rename(
    "population"="Population",
    "parent"="Parent",
    "count"="Count",
    "parentCount"="ParentCount"
  )

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

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

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

  for (populationName in populations) {
    samplePopulationStatistic <- sampleStatistic %>%
      dplyr::filter(sampleStatistic[["population"]] == populationName)

    experiment[sampleName, populationName] <-
      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
)

# 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	2938	15217	7708	7481	4488	3220	0.576	0.793	0.273
data/786 n1/786n1_Ca1uM4h_samp_60x 00016122 616.LMD	3382	12773	4296	8474	3572	724	0.795	0.930	0.131
data/786 n1/786n1_Ca1uM4h_samp_100x 00016125 619.LMD	3421	11805	3811	7991	3345	466	0.858	0.952	0.188
data/786 n1/786n1_Ca1uM4h_samp_140x 00016128 622.LMD	3684	11832	3419	8412	3147	272	0.906	0.969	0.173
data/786 n1/786n1_Ca1uM4h_samp_180x 00016131 625.LMD	3666	11654	3343	8311	3148	195	0.933	0.978	0.206
data/786 n1/786n1_Ca1uM4h_samp_220x 00016134 628.LMD	3542	11332	3213	8118	3039	174	0.953	0.994	0.254
data/786 n2/786n2_Ca1uM4h_samp_20x 00016158 632.LMD	2886	13606	5379	8222	4468	911	0.694	0.795	0.200
data/786 n2/786n2_Ca1uM4h_samp_60x 00016161 635.LMD	2848	10367	3225	7142	2978	247	0.894	0.951	0.197
data/786 n2/786n2_Ca1uM4h_samp_100x 00016164 638.LMD	2646	9588	2978	6610	2753	225	0.928	0.980	0.283
data/786 n2/786n2_Ca1uM4h_samp_140x 00016167 641.LMD	3101	10395	3115	7280	2876	239	0.949	0.990	0.451
data/786 n2/786n2_Ca1uM4h_samp_180x 00016170 644.LMD	3112	10011	2939	7072	2763	176	0.968	0.998	0.491
data/786 n2/786n2_Ca1uM4h_samp_220x 00016173 647.LMD	2751	8983	2488	6495	2381	107	0.974	1.000	0.408
data/786 n3/786n3_Ca1uM4h_samp_20x 00016176 650.LMD	2938	11926	4312	7609	3430	882	0.750	0.914	0.112
data/786 n3/786n3_Ca1uM4h_samp_60x 00016179 653.LMD	2919	10275	3254	7019	2988	266	0.926	0.992	0.186
data/786 n3/786n3_Ca1uM4h_samp_100x 00016182 656.LMD	3051	10637	3216	7421	2990	226	0.951	0.996	0.366
data/786 n3/786n3_Ca1uM4h_samp_140x 00016185 659.LMD	3319	10783	3286	7497	3137	149	0.967	0.990	0.490
data/786 n3/786n3_Ca1uM4h_samp_180x 00016188 662.LMD	3723	12305	3784	8521	3533	251	0.952	0.973	0.659
data/786 n3/786n3_Ca1uM4h_samp_220x 00016191 665.LMD	3652	11440	3490	7950	3317	173	0.981	0.999	0.638
data/VHL n1/VHLn1_Ca1uM4h_samp_20x 00016196 670.LMD	3341	12611	4498	8112	4041	457	0.756	0.815	0.242
data/VHL n1/VHLn1_Ca1uM4h_samp_60x 00016199 673.LMD	3340	12436	3790	8645	3619	171	0.921	0.941	0.503
data/VHL n1/VHLn1_Ca1uM4h_samp_100x 00016202 676.LMD	3319	10684	3036	7648	2978	58	0.973	0.984	0.446
data/VHL n1/VHLn1_Ca1uM4h_samp_140x 00016205 679.LMD	3466	10858	3212	7645	3105	107	0.960	0.972	0.610
data/VHL n1/VHLn1_Ca1uM4h_samp_180x 00016208 682.LMD	3219	10544	3090	7454	3008	82	0.979	0.989	0.613
data/VHL n1/VHLn1_Ca1uM4h_samp_220x 00016211 685.LMD	3590	11147	3269	7878	3175	94	0.970	0.974	0.809
data/VHL n2/VHLn2_Ca1uM4h_samp_20x 00016215 689.LMD	2988	13532	4274	9257	3972	302	0.872	0.918	0.261
data/VHL n2/VHLn2_Ca1uM4h_samp_60x 00016218 692.LMD	3391	11729	3562	8167	3425	137	0.950	0.968	0.486
data/VHL n2/VHLn2_Ca1uM4h_samp_100x 00016221 695.LMD	3494	11910	3467	8443	3370	97	0.979	0.988	0.668
data/VHL n2/VHLn2_Ca1uM4h_samp_140x 00016224 698.LMD	3522	12799	3404	9394	3329	75	0.991	0.998	0.688
data/VHL n2/VHLn2_Ca1uM4h_samp_180x 00016227 701.LMD	3709	11784	3456	8327	3356	100	0.993	1.000	0.764
data/VHL n2/VHLn2_Ca1uM4h_samp_220x 00016230 704.LMD	3494	11520	3690	7830	3499	191	0.965	0.971	0.850
data/VHL n3/VHLn3_Ca1uM4h_samp_20x 00016234 708.LMD	3273	14254	4341	9913	3960	381	0.795	0.860	0.119
data/VHL n3/VHLn3_Ca1uM4h_samp_60x 00016237 711.LMD	3191	11503	3349	8154	3223	126	0.944	0.973	0.197
data/VHL n3/VHLn3_Ca1uM4h_samp_100x 00016240 714.LMD	3050	11094	3075	8019	2991	84	0.968	0.985	0.355
data/VHL n3/VHLn3_Ca1uM4h_samp_140x 00016243 717.LMD	3236	10887	3101	7786	3050	51	0.990	1.000	0.379
data/VHL n3/VHLn3_Ca1uM4h_samp_180x 00016246 720.LMD	3095	11154	3008	8146	2965	43	0.992	1.000	0.421
data/VHL n3/VHLn3_Ca1uM4h_samp_220x 00016249 723.LMD	3229	10945	3052	7893	2998	54	0.991	1.000	0.535

```
# 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"]], digits=0)
normalizedExperiment[["cd24pos"]] <-
  round(normalizedExperiment[["cd24pos"]], digits=0)
normalizedExperiment[["cd24neg"]] <-
  round(normalizedExperiment[["cd24neg"]], digits=0)
normalizedExperiment[["cd24mid"]] <-
  round(normalizedExperiment[["cd24mid"]], digits=0)
normalizedExperiment[["cd24hi"]] <-
  round(normalizedExperiment[["cd24hi"]], digits=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
```

```
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	5179374	2623553	2546290	1527570	1095984	0.576	0.793	0.273
data/786 n1/786n1_Ca1uM4h_samp_60x 00016122 616.LMD	40	11330278	3810763	7516854	3168539	642224	0.795	0.930	0.131
data/786 n1/786n1_Ca1uM4h_samp_100x 00016125 619.LMD	40	17253727	5570009	11679334	4888921	681087	0.858	0.952	0.188
data/786 n1/786n1_Ca1uM4h_samp_140x 00016128 622.LMD	40	22482085	6496471	15983713	5979642	516830	0.906	0.969	0.173
data/786 n1/786n1_Ca1uM4h_samp_180x 00016131 625.LMD	40	28610475	8207038	20403437	7728314	478723	0.933	0.978	0.206
data/786 n1/786n1_Ca1uM4h_samp_220x 00016134 628.LMD	40	35192547	9978261	25211180	9437888	540373	0.953	0.994	0.254
data/786 n2/786n2_Ca1uM4h_samp_20x 00016158 632.LMD	40	4714484	1863825	2848926	1548164	315662	0.694	0.795	0.200
data/786 n2/786n2_Ca1uM4h_samp_60x 00016161 635.LMD	40	10920295	3397121	7523174	3136938	260183	0.894	0.951	0.197
data/786 n2/786n2_Ca1uM4h_samp_100x 00016164 638.LMD	40	18117914	5627362	12490552	5202192	425170	0.928	0.980	0.283
data/786 n2/786n2_Ca1uM4h_samp_140x 00016167 641.LMD	40	23465011	7031603	16433409	6492099	539503	0.949	0.990	0.451
data/786 n2/786n2_Ca1uM4h_samp_180x 00016170 644.LMD	40	28952121	8499679	20452442	7990681	508997	0.968	0.998	0.491
data/786 n2/786n2_Ca1uM4h_samp_220x 00016173 647.LMD	40	35918939	9948382	25970556	9520538	427844	0.974	1.000	0.408
data/786 n3/786n3_Ca1uM4h_samp_20x 00016176 650.LMD	40	4059224	1467665	2589857	1167461	300204	0.750	0.914	0.112
data/786 n3/786n3_Ca1uM4h_samp_60x 00016179 653.LMD	40	10560123	3344296	7213772	3070915	273381	0.926	0.992	0.186
data/786 n3/786n3_Ca1uM4h_samp_100x 00016182 656.LMD	40	17431990	5270403	12161586	4900033	370370	0.951	0.996	0.366
data/786 n3/786n3_Ca1uM4h_samp_140x 00016185 659.LMD	40	22742091	6930401	15811690	6616149	314251	0.967	0.990	0.490
data/786 n3/786n3_Ca1uM4h_samp_180x 00016188 662.LMD	40	29746172	9147462	20598711	8540693	606769	0.952	0.973	0.659
data/786 n3/786n3_Ca1uM4h_samp_220x 00016191 665.LMD	40	34457831	10512048	23945783	9990964	521084	0.981	0.999	0.638
data/VHL n1/VHLn1_Ca1uM4h_samp_20x 00016196 670.LMD	40	3774618	1346304	2428016	1209518	136785	0.756	0.815	0.242
data/VHL n1/VHLn1_Ca1uM4h_samp_60x 00016199 673.LMD	40	11170060	3404192	7764970	3250599	153593	0.921	0.941	0.503
data/VHL n1/VHLn1_Ca1uM4h_samp_100x 00016202 676.LMD	40	16095209	4573667	11521543	4486291	87376	0.973	0.984	0.446
data/VHL n1/VHLn1_Ca1uM4h_samp_140x 00016205 679.LMD	40	21929025	6487017	15439988	6270917	216099	0.960	0.972	0.610
data/VHL n1/VHLn1_Ca1uM4h_samp_180x 00016208 682.LMD	40	29479963	8639329	20840634	8410065	229264	0.979	0.989	0.613
data/VHL n1/VHLn1_Ca1uM4h_samp_220x 00016211 685.LMD	40	34155153	10016435	24138719	9728412	288022	0.970	0.974	0.809
data/VHL n2/VHLn2_Ca1uM4h_samp_20x 00016215 689.LMD	40	4528782	1430388	3098059	1329317	101071	0.872	0.918	0.261
data/VHL n2/VHLn2_Ca1uM4h_samp_60x 00016218 692.LMD	40	10376585	3151283	7225302	3030080	121203	0.950	0.968	0.486
data/VHL n2/VHLn2_Ca1uM4h_samp_100x 00016221 695.LMD	40	17043503	4961362	12082141	4822553	138809	0.979	0.988	0.668
data/VHL n2/VHLn2_Ca1uM4h_samp_140x 00016224 698.LMD	40	25438103	6765474	18670642	6616411	149063	0.991	0.998	0.688
data/VHL n2/VHLn2_Ca1uM4h_samp_180x 00016227 701.LMD	40	28594230	8386088	20205716	8143435	242653	0.993	1.000	0.764
data/VHL n2/VHLn2_Ca1uM4h_samp_220x 00016230 704.LMD	40	36267888	11617058	24650830	11015741	601317	0.965	0.971	0.850
data/VHL n3/VHLn3_Ca1uM4h_samp_20x 00016234 708.LMD	40	4355026	1326306	3028720	1209899	116407	0.795	0.860	0.119
data/VHL n3/VHLn3_Ca1uM4h_samp_60x 00016237 711.LMD	40	10814478	3148543	7665935	3030085	118458	0.944	0.973	0.197
data/VHL n3/VHLn3_Ca1uM4h_samp_100x 00016240 714.LMD	40	18186885	5040984	13145902	4903279	137705	0.968	0.985	0.355
data/VHL n3/VHLn3_Ca1uM4h_samp_140x 00016243 717.LMD	40	23550371	6707973	16842398	6597651	110321	0.990	1.000	0.379
data/VHL n3/VHLn3_Ca1uM4h_samp_180x 00016246 720.LMD	40	32434895	8747011	23687884	8621971	125040	0.992	1.000	0.421
data/VHL n3/VHLn3_Ca1uM4h_samp_220x 00016249 723.LMD	40	37285537	10397027	26888510	10213069	183958	0.991	1.000	0.535

4.3.1.1 Dataset 1 - Raw gate counts

```
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	5179374	2623553	2546290	1527570	1095984	0.5065386	0.2949333	0.2116055
data/786 n1/786n1_Ca1uM4h_samp_60x 00016122 616.LMD	40	11330278	3810763	7516854	3168539	642224	0.3363345	0.2796524	0.0566821
data/786 n1/786n1_Ca1uM4h_samp_100x 00016125 619.LMD	40	17253727	5570009	11679334	4888921	681087	0.3228293	0.2833545	0.0394748
data/786 n1/786n1_Ca1uM4h_samp_140x 00016128 622.LMD	40	22482085	6496471	15983713	5979642	516830	0.2889621	0.2659736	0.0229885
data/786 n1/786n1_Ca1uM4h_samp_180x 00016131 625.LMD	40	28610475	8207038	20403437	7728314	478723	0.2868543	0.2701218	0.0167324
data/786 n1/786n1_Ca1uM4h_samp_220x 00016134 628.LMD	40	35192547	9978261	25211180	9437888	540373	0.2835334	0.2681786	0.0153548
data/786 n2/786n2_Ca1uM4h_samp_20x 00016158 632.LMD	40	4714484	1863825	2848926	1548164	315662	0.3953402	0.3283846	0.0669558
data/786 n2/786n2_Ca1uM4h_samp_60x 00016161 635.LMD	40	10920295	3397121	7523174	3136938	260183	0.3110833	0.2872576	0.0238256
data/786 n2/786n2_Ca1uM4h_samp_100x 00016164 638.LMD	40	18117914	5627362	12490552	5202192	425170	0.3105966	0.2871297	0.0234668
data/786 n2/786n2_Ca1uM4h_samp_140x 00016167 641.LMD	40	23465011	7031603	16433409	6492099	539503	0.2996633	0.2766715	0.0229918
data/786 n2/786n2_Ca1uM4h_samp_180x 00016170 644.LMD	40	28952121	8499679	20452442	7990681	508997	0.2935771	0.2759964	0.0175806
data/786 n2/786n2_Ca1uM4h_samp_220x 00016173 647.LMD	40	35918939	9948382	25970556	9520538	427844	0.2769676	0.2650562	0.0119114
data/786 n3/786n3_Ca1uM4h_samp_20x 00016176 650.LMD	40	4059224	1467665	2589857	1167461	300204	0.3615629	0.2876069	0.0739560
data/786 n3/786n3_Ca1uM4h_samp_60x 00016179 653.LMD	40	10560123	3344296	7213772	3070915	273381	0.3166910	0.2908030	0.0258881
data/786 n3/786n3_Ca1uM4h_samp_100x 00016182 656.LMD	40	17431990	5270403	12161586	4900033	370370	0.3023409	0.2810943	0.0212466
data/786 n3/786n3_Ca1uM4h_samp_140x 00016185 659.LMD	40	22742091	6930401	15811690	6616149	314251	0.3047390	0.2909209	0.0138180
data/786 n3/786n3_Ca1uM4h_samp_180x 00016188 662.LMD	40	29746172	9147462	20598711	8540693	606769	0.3075173	0.2871191	0.0203982
data/786 n3/786n3_Ca1uM4h_samp_220x 00016191 665.LMD	40	34457831	10512048	23945783	9990964	521084	0.3050699	0.2899476	0.0151224
data/VHL n1/VHLn1_Ca1uM4h_samp_20x 00016196 670.LMD	40	3774618	1346304	2428016	1209518	136785	0.3566729	0.3204345	0.0362381
data/VHL n1/VHLn1_Ca1uM4h_samp_60x 00016199 673.LMD	40	11170060	3404192	7764970	3250599	153593	0.3047604	0.2910100	0.0137504
data/VHL n1/VHLn1_Ca1uM4h_samp_100x 00016202 676.LMD	40	16095209	4573667	11521543	4486291	87376	0.2841633	0.2787346	0.0054287
data/VHL n1/VHLn1_Ca1uM4h_samp_140x 00016205 679.LMD	40	21929025	6487017	15439988	6270917	216099	0.2958188	0.2859642	0.0098545
data/VHL n1/VHLn1_Ca1uM4h_samp_180x 00016208 682.LMD	40	29479963	8639329	20840634	8410065	229264	0.2930577	0.2852807	0.0077769
data/VHL n1/VHLn1_Ca1uM4h_samp_220x 00016211 685.LMD	40	34155153	10016435	24138719	9728412	288022	0.2932628	0.2848300	0.0084328
data/VHL n2/VHLn2_Ca1uM4h_samp_20x 00016215 689.LMD	40	4528782	1430388	3098059	1329317	101071	0.3158439	0.2935264	0.0223175
data/VHL n2/VHLn2_Ca1uM4h_samp_60x 00016218 692.LMD	40	10376585	3151283	7225302	3030080	121203	0.3036917	0.2920113	0.0116804
data/VHL n2/VHLn2_Ca1uM4h_samp_100x 00016221 695.LMD	40	17043503	4961362	12082141	4822553	138809	0.2910999	0.2829555	0.0081444
data/VHL n2/VHLn2_Ca1uM4h_samp_140x 00016224 698.LMD	40	25438103	6765474	18670642	6616411	149063	0.2659583	0.2600984	0.0058598
data/VHL n2/VHLn2_Ca1uM4h_samp_180x 00016227 701.LMD	40	28594230	8386088	20205716	8143435	242653	0.2932790	0.2847929	0.0084861
data/VHL n2/VHLn2_Ca1uM4h_samp_220x 00016230 704.LMD	40	36267888	11617058	24650830	11015741	601317	0.3203125	0.3037326	0.0165799
data/VHL n3/VHLn3_Ca1uM4h_samp_20x 00016234 708.LMD	40	4355026	1326306	3028720	1209899	116407	0.3045461	0.2778167	0.0267293
data/VHL n3/VHLn3_Ca1uM4h_samp_60x 00016237 711.LMD	40	10814478	3148543	7665935	3030085	118458	0.2911415	0.2801878	0.0109536
data/VHL n3/VHLn3_Ca1uM4h_samp_100x 00016240 714.LMD	40	18186885	5040984	13145902	4903279	137705	0.2771769	0.2696052	0.0075717

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_140x 00016243 717.LMD	40	23550371	6707973	16842398	6597651	110321	0.2848351	0.2801506	0.0046845
data/VHL n3/VHLn3_Ca1uM4h_samp_180x 00016246 720.LMD	40	32434895	8747011	23687884	8621971	125040	0.2696790	0.2658239	0.0038551
data/VHL n3/VHLn3_Ca1uM4h_samp_220x 00016249 723.LMD	40	37285537	10397027	26888510	10213069	183958	0.2788488	0.2739150	0.0049338

4.3.1.2 Dataset 2 - Swarm-deconvoluted

```
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	5179374	1510567	2546290	1211363.0	299203.63	0.2916504	0.2338821	0.0577683
data/786 n1/786n1_Ca1uM4h_samp_60x 00016122 616.LMD	40	11330278	3030873	7516854	2946741.3	84131.34	0.2675021	0.2600767	0.0074254
data/786 n1/786n1_Ca1uM4h_samp_100x 00016125 619.LMD	40	17253727	4782297	11679334	4654252.8	128044.36	0.2771747	0.2697535	0.0074213
data/786 n1/786n1_Ca1uM4h_samp_140x 00016128 622.LMD	40	22482085	5883685	15983713	5794273.1	89411.59	0.2617055	0.2577285	0.0039770
data/786 n1/786n1_Ca1uM4h_samp_180x 00016131 625.LMD	40	28610475	7656908	20403437	7558291.1	98616.94	0.2676260	0.2641792	0.0034469
data/786 n1/786n1_Ca1uM4h_samp_220x 00016134 628.LMD	40	35192547	9518515	25211180	9381260.7	137254.74	0.2704696	0.2665695	0.0039001
data/786 n2/786n2_Ca1uM4h_samp_20x 00016158 632.LMD	40	4714484	1293923	2848926	1230790.4	63132.40	0.2744569	0.2610658	0.0133912
data/786 n2/786n2_Ca1uM4h_samp_60x 00016161 635.LMD	40	10920295	3034484	7523174	2983228.0	51256.05	0.2778757	0.2731820	0.0046937
data/786 n2/786n2_Ca1uM4h_samp_100x 00016164 638.LMD	40	18117914	5218471	12490552	5098148.2	120323.11	0.2880283	0.2813871	0.0066411
data/786 n2/786n2_Ca1uM4h_samp_140x 00016167 641.LMD	40	23465011	6670494	16433409	6427178.0	243315.85	0.2842741	0.2739048	0.0103693
data/786 n2/786n2_Ca1uM4h_samp_180x 00016170 644.LMD	40	28952121	8224617	20452442	7974699.6	249917.53	0.2840765	0.2754444	0.0086321
data/786 n2/786n2_Ca1uM4h_samp_220x 00016173 647.LMD	40	35918939	9695098	25970556	9520538.0	174560.35	0.2699161	0.2650562	0.0048598
data/786 n3/786n3_Ca1uM4h_samp_20x 00016176 650.LMD	40	4059224	1100682	2589857	1067059.4	33622.85	0.2711558	0.2628727	0.0082831
data/786 n3/786n3_Ca1uM4h_samp_60x 00016179 653.LMD	40	10560123	3097197	7213772	3046347.7	50848.87	0.2932917	0.2884765	0.0048152
data/786 n3/786n3_Ca1uM4h_samp_100x 00016182 656.LMD	40	17431990	5015988	12161586	4880432.9	135555.42	0.2877462	0.2799699	0.0077762
data/786 n3/786n3_Ca1uM4h_samp_140x 00016185 659.LMD	40	22742091	6703970	15811690	6549987.5	153982.99	0.2947825	0.2880117	0.0067708
data/786 n3/786n3_Ca1uM4h_samp_180x 00016188 662.LMD	40	29746172	8709955	20598711	8310094.3	399860.77	0.2928093	0.2793668	0.0134424
data/786 n3/786n3_Ca1uM4h_samp_220x 00016191 665.LMD	40	34457831	10313425	23945783	9980973.0	332451.59	0.2993057	0.2896576	0.0096481
data/VHL n1/VHLn1_Ca1uM4h_samp_20x 00016196 670.LMD	40	3774618	1018859	2428016	985757.2	33101.97	0.2699238	0.2611542	0.0087696
data/VHL n1/VHLn1_Ca1uM4h_samp_60x 00016199 673.LMD	40	11170060	3136071	7764970	3058813.7	77257.28	0.2807569	0.2738404	0.0069165
data/VHL n1/VHLn1_Ca1uM4h_samp_100x 00016202 676.LMD	40	16095209	4453480	11521543	4414510.3	38969.70	0.2766960	0.2742748	0.0024212
data/VHL n1/VHLn1_Ca1uM4h_samp_140x 00016205 679.LMD	40	21929025	6227152	15439988	6095331.3	131820.39	0.2839685	0.2779572	0.0060112
data/VHL n1/VHLn1_Ca1uM4h_samp_180x 00016208 682.LMD	40	29479963	8458093	20840634	8317554.3	140538.83	0.2869099	0.2821426	0.0047673
data/VHL n1/VHLn1_Ca1uM4h_samp_220x 00016211 685.LMD	40	34155153	9708483	24138719	9475473.3	233009.80	0.2842465	0.2774244	0.0068221
data/VHL n2/VHLn2_Ca1uM4h_samp_20x 00016215 689.LMD	40	4528782	1246693	3098059	1220313.0	26379.53	0.2752821	0.2694572	0.0058249
data/VHL n2/VHLn2_Ca1uM4h_samp_60x 00016218 692.LMD	40	10376585	2992022	7225302	2933117.4	58904.66	0.2883436	0.2826669	0.0056767
data/VHL n2/VHLn2_Ca1uM4h_samp_100x 00016221 695.LMD	40	17043503	4857407	12082141	4764682.4	92724.41	0.2850005	0.2795600	0.0054405
data/VHL n2/VHLn2_Ca1uM4h_samp_140x 00016224 698.LMD	40	25438103	6705734	18670642	6603178.2	102555.34	0.2636098	0.2595782	0.0040316
data/VHL n2/VHLn2_Ca1uM4h_samp_180x 00016227 701.LMD	40	28594230	8328822	20205716	8143435.0	185386.89	0.2912763	0.2847929	0.0064834
data/VHL n2/VHLn2_Ca1uM4h_samp_220x 00016230 704.LMD	40	36267888	11207404	24650830	10696284.5	511119.45	0.3090173	0.2949244	0.0140929
data/VHL n3/VHLn3_Ca1uM4h_samp_20x 00016234 708.LMD	40	4355026	1054366	3028720	1040513.1	13852.43	0.2421032	0.2389224	0.0031808
data/VHL n3/VHLn3_Ca1uM4h_samp_60x 00016237 711.LMD	40	10814478	2971609	7665935	2948272.7	23336.23	0.2747806	0.2726227	0.0021579
data/VHL n3/VHLn3_Ca1uM4h_samp_100x 00016240 714.LMD	40	18186885	4878615	13145902	4829729.8	48885.28	0.2682491	0.2655611	0.0026879
data/VHL n3/VHLn3_Ca1uM4h_samp_140x 00016243 717.LMD	40	23550371	6639463	16842398	6597651.0	41811.66	0.2819260	0.2801506	0.0017754
data/VHL n3/VHLn3_Ca1uM4h_samp_180x 00016246 720.LMD	40	32434895	8674613	23687884	8621971.0	52641.84	0.2674469	0.2658239	0.0016230
data/VHL n3/VHLn3_Ca1uM4h_samp_220x 00016249 723.LMD	40	37285537	10311487	26888510	10213069.0	98417.53	0.2765546	0.2739150	0.0026396

4.3.1.3 Dataset 3 - Swarm-deconvoluted and normalized

```
# 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
)
```

```
## $`cd24hiNoiseCeiling: `
## [1] 43
##
## $`cd24midNoiseCeiling: `
## [1] 2381
##
## $`cd24negNoiseCeiling: `
## [1] 6495
```

```
# 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	1021118	685516	335602	400948	284568	0.6713387	0.3926559	0.2786828
data/786 n1/786n1_Ca1uM4h_samp_60x 00016122 616.LMD	40	2636136	880666	1755470	834678	45988	0.3340746	0.3166293	0.0174452
data/786 n1/786n1_Ca1uM4h_samp_100x 00016125 619.LMD	40	3425969	1239474	2186495	1174277	65197	0.3617879	0.3427576	0.0190302
data/786 n1/786n1_Ca1uM4h_samp_140x 00016128 622.LMD	40	4920329	1277821	3642508	1270114	7707	0.2597023	0.2581360	0.0015664
data/786 n1/786n1_Ca1uM4h_samp_180x 00016131 625.LMD	40	6171221	1712956	4458265	1712956	0	0.2775717	0.2775717	0.0000000

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

	cmVolume	events	cd24pos	cd24neg	cd24mid	cd24hi	cd24posProp	cd24midProp	cd24hiProp
data/786 n1/786n1_Ca1uM4h_samp_220x 00016134 628.LMD	40	7030938	1990565	5040373	1986851	3714	0.2831151	0.2825869	0.0005282
data/786 n2/786n2_Ca1uM4h_samp_20x 00016158 632.LMD	40	1052412	454006	598406	405773	48233	0.4313957	0.3855648	0.0458309
data/786 n2/786n2_Ca1uM4h_samp_60x 00016161 635.LMD	40	1162644	481113	681531	475152	5961	0.4138094	0.4086823	0.0051271
data/786 n2/786n2_Ca1uM4h_samp_100x 00016164 638.LMD	40	855281	637972	217309	598904	39068	0.7459209	0.7002424	0.0456786
data/786 n2/786n2_Ca1uM4h_samp_140x 00016167 641.LMD	40	2970720	1198711	1772009	1052460	146251	0.4035086	0.3542778	0.0492308
data/786 n2/786n2_Ca1uM4h_samp_180x 00016170 644.LMD	40	2883036	1214334	1668702	1088774	125560	0.4211997	0.3776484	0.0435513
data/786 n2/786n2_Ca1uM4h_samp_220x 00016173 647.LMD	40	2623	2623	0	0	2623	1.0000000	0.0000000	1.0000000
data/786 n3/786n3_Ca1uM4h_samp_20x 00016176 650.LMD	40	654801	275631	379170	256644	18987	0.4209386	0.3919420	0.0289966
data/786 n3/786n3_Ca1uM4h_samp_60x 00016179 653.LMD	40	1144473	605932	538541	599276	6656	0.5294419	0.5236262	0.0058158
data/786 n3/786n3_Ca1uM4h_samp_100x 00016182 656.LMD	40	2561055	1043520	1517535	978433	65087	0.4074571	0.3820429	0.0254141
data/786 n3/786n3_Ca1uM4h_samp_140x 00016185 659.LMD	40	3704875	1591588	2113287	1528295	63293	0.4295929	0.4125092	0.0170837
data/786 n3/786n3_Ca1uM4h_samp_180x 00016188 662.LMD	40	7747827	2850164	4897663	2554252	295912	0.3678662	0.3296733	0.0381929
data/786 n3/786n3_Ca1uM4h_samp_220x 00016191 665.LMD	40	7394750	3012220	4382530	2809286	202934	0.4073458	0.3799028	0.0274430
data/VHL n1/VHLn1_Ca1uM4h_samp_20x 00016196 670.LMD	40	777315	293328	483987	273096	20232	0.3773605	0.3513325	0.0260281
data/VHL n1/VHLn1_Ca1uM4h_samp_60x 00016199 673.LMD	40	2889963	958825	1931138	920191	38634	0.3317776	0.3184093	0.0133683
data/VHL n1/VHLn1_Ca1uM4h_samp_100x 00016202 676.LMD	40	2564556	827587	1736969	827587	0	0.3227019	0.3227019	0.0000000
data/VHL n1/VHLn1_Ca1uM4h_samp_140x 00016205 679.LMD	40	3654158	1331596	2322562	1286619	44977	0.3644057	0.3520973	0.0123084
data/VHL n1/VHLn1_Ca1uM4h_samp_180x 00016208 682.LMD	40	4362100	1680833	2681267	1660518	20315	0.3853266	0.3806694	0.0046572
data/VHL n1/VHLn1_Ca1uM4h_samp_220x 00016211 685.LMD	40	6518789	2281185	4237604	2179930	101255	0.3499400	0.3344072	0.0155328
data/VHL n2/VHLn2_Ca1uM4h_samp_20x 00016215 689.LMD	40	1359812	435448	924364	423459	11989	0.3202266	0.3114100	0.0088167
data/VHL n2/VHLn2_Ca1uM4h_samp_60x 00016218 692.LMD	40	2326732	847522	1479210	826659	20863	0.3642542	0.3552876	0.0089667
data/VHL n2/VHLn2_Ca1uM4h_samp_100x 00016221 695.LMD	40	4176240	1388604	2787636	1357413	31191	0.3325010	0.3250323	0.0074687
data/VHL n2/VHLn2_Ca1uM4h_samp_140x 00016224 698.LMD	40	7649800	1888017	5761783	1870924	17093	0.2468061	0.2445716	0.0022344
data/VHL n2/VHLn2_Ca1uM4h_samp_180x 00016227 701.LMD	40	6892316	2446913	4445403	2365867	81046	0.3550204	0.3432615	0.0117589
data/VHL n2/VHLn2_Ca1uM4h_samp_220x 00016230 704.LMD	40	7778955	3576036	4202919	3200292	375744	0.4597065	0.4114038	0.0483026
data/VHL n3/VHLn3_Ca1uM4h_samp_20x 00016234 708.LMD	40	1358063	313761	1044302	313046	715	0.2310357	0.2305092	0.0005265
data/VHL n3/VHLn3_Ca1uM4h_samp_60x 00016237 711.LMD	40	2269488	709789	1559699	709789	0	0.3127529	0.3127529	0.0000000
data/VHL n3/VHLn3_Ca1uM4h_samp_100x 00016240 714.LMD	40	3424812	926451	2498361	926451	0	0.2705115	0.2705115	0.0000000
data/VHL n3/VHLn3_Ca1uM4h_samp_140x 00016243 717.LMD	40	4239802	1447157	2792645	1447157	0	0.3413266	0.3413266	0.0000000
data/VHL n3/VHLn3_Ca1uM4h_samp_180x 00016246 720.LMD	40	6499192	1698223	4800969	1698223	0	0.2612976	0.2612976	0.0000000
data/VHL n3/VHLn3_Ca1uM4h_samp_220x 00016249 723.LMD	40	6864354	2101889	4762465	2101889	0	0.3062035	0.3062035	0.0000000

4.3.2 Computing summary data

4.3.2.1 Dataset 1 - Raw gate counts

```
# 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(),
      .groups="drop"
    )
```

```
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	4651027	1985014	2661691	1414398	570616.7	0.4211473	0.5779771	0.3036416	0.1175058
786-O	60	3	10936899	3517393	7417933	3125464	391929.3	0.3213696	0.6784873	0.2859043	0.0354653
786-O	100	3	17601210	5489258	12110491	4997049	492209.0	0.3119223	0.6879930	0.2838595	0.0280627
786-O	140	3	22896396	6819492	16076271	6362630	456861.3	0.2977881	0.7021837	0.2778553	0.0199328
786-O	180	3	29102923	8618060	20484863	8086563	531496.3	0.2959829	0.7040171	0.2777458	0.0182371
786-O	220	3	35189772	10146230	25042506	9649797	496433.7	0.2885236	0.7114469	0.2743941	0.0141295
VHL	20	3	4219475	1367666	2851598	1249578	118087.7	0.3256876	0.6742614	0.2972592	0.0284283
VHL	60	3	10787041	3234673	7552069	3103588	131084.7	0.2998645	0.7001087	0.2877364	0.0121282
VHL	100	3	17108532	4858671	12249862	4737374	121296.7	0.2841467	0.7158534	0.2770984	0.0070483
VHL	140	3	23639166	6653488	16984343	6494993	158494.3	0.2822041	0.7177392	0.2754044	0.0067996
VHL	180	3	30169696	8590809	21578078	8391824	198985.7	0.2853386	0.7146331	0.2786325	0.0067060
VHL	220	3	35902859	10676840	25226020	10319074	357765.7	0.2974747	0.7025253	0.2874926	0.0099821

```
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	187588.2	195798.77	54535.65	71367.25	151682.367	0.0252850	0.0255546	0.0072463	0.0271893
786-O	60	3	128448.6	85144.94	58945.71	16604.39	72287.322	0.0044199	0.0044527	0.0018991	0.0061344
786-O	100	3	152089.7	63897.03	136005.19	59248.73	55284.048	0.0034361	0.0034810	0.0010164	0.0033151
786-O	140	3	169768.9	94761.23	107009.25	112475.66	41341.124	0.0026846	0.0026692	0.0041719	0.0017652
786-O	180	3	194224.3	160419.52	33864.67	138196.67	22307.471	0.0035132	0.0035132	0.0028775	0.0006397
786-O	220	3	243519.3	105719.81	340956.91	99445.19	20059.421	0.0049003	0.0048953	0.0045199	0.0006415
VHL	20	3	131644.0	18410.50	122822.58	23018.75	5972.073	0.0091407	0.0091512	0.0071842	0.0023714
VHL	60	3	132482.8	48938.15	95762.72	42438.42	6513.683	0.0025244	0.0025331	0.0021855	0.0004837
VHL	100	3	349117.7	83336.92	275021.88	73719.67	9793.782	0.0023205	0.0023205	0.0022746	0.0004772
VHL	140	3	585407.8	49002.31	539999.23	64760.69	17838.656	0.0050344	0.0050342	0.0045235	0.0009034
VHL	180	3	670362.7	61762.96	618173.05	79929.65	21462.597	0.0045207	0.0045290	0.0036984	0.0008314
VHL	220	3	532265.6	278731.25	487450.28	216726.09	72414.906	0.0070168	0.0070168	0.0050287	0.0019919

4.3.2.2 Dataset 2 - Swarm-deconvoluted

```
# 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(),
      .groups="drop"
    )
```

```
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	4651027	1301724	2661691	1169738	131986.29	0.2790877	0.5779771	0.2526069	0.0264808
786-O	60	3	10936899	3054184	7417933	2992106	62078.75	0.2795565	0.6784873	0.2739117	0.0056447
786-O	100	3	17601210	5005586	12110491	4877611	127974.30	0.2843164	0.6879930	0.2770368	0.0072795
786-O	140	3	22896396	6419383	16076271	6257146	162236.81	0.2802540	0.7021837	0.2732150	0.0070391
786-O	180	3	29102923	8197160	20484863	7947695	249465.08	0.2815039	0.7040171	0.2729968	0.0085071
786-O	220	3	35189772	9842346	25042506	9627591	214755.56	0.2798971	0.7114469	0.2737611	0.0061360
VHL	20	3	4219475	1106639	2851598	1082194	24444.64	0.2624363	0.6742614	0.2565112	0.0059251
VHL	60	3	10787041	3033234	7552069	2980068	53166.05	0.2812937	0.7001087	0.2763767	0.0049170
VHL	100	3	17108532	4729834	12249862	4669641	60193.13	0.2766485	0.7158534	0.2731320	0.0035165
VHL	140	3	23639166	6524116	16984343	6432054	92062.46	0.2765014	0.7177392	0.2725620	0.0039394
VHL	180	3	30169696	8487176	21578078	8360987	126189.19	0.2818777	0.7146331	0.2775865	0.0042912
VHL	220	3	35902859	10409125	25226020	10128276	280848.93	0.2899395	0.7025253	0.2820879	0.0078515

```
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	187588.2	68351.18	54535.65	29816.98	48521.395	0.0036680	0.0255546	0.0054138	0.0090719
786-O	60	3	128448.6	12431.11	58945.71	16797.71	6366.396	0.0043256	0.0044527	0.0047380	0.0005144
786-O	100	3	152089.7	72757.69	136005.19	73987.05	2538.799	0.0020622	0.0034810	0.0021158	0.0001936
786-O	140	3	169768.9	154743.41	107009.25	135178.56	25761.135	0.0056337	0.0026692	0.0050511	0.0010682
786-O	180	3	194224.3	175686.73	33864.67	125542.80	50207.475	0.0042624	0.0035132	0.0026280	0.0016663
786-O	220	3	243519.3	139136.86	340956.91	104620.44	34540.144	0.0056035	0.0048953	0.0045958	0.0010264
VHL	20	3	131644.0	40860.74	122822.58	40902.47	3256.515	0.0059372	0.0091512	0.0052627	0.0009319
VHL	60	3	132482.8	29880.79	95762.72	22871.83	9138.251	0.0022658	0.0025331	0.0018271	0.0008229
VHL	100	3	349117.7	79854.78	275021.88	74443.47	9535.274	0.0027919	0.0023205	0.0023564	0.0005572
VHL	140	3	585407.8	86434.81	539999.23	97207.68	15304.205	0.0037370	0.0050342	0.0037659	0.0007065
VHL	180	3	670362.7	58240.12	618173.05	80735.48	22508.631	0.0042289	0.0045290	0.0034242	0.0008216
VHL	220	3	532265.6	251405.12	487450.28	204935.61	70156.298	0.0056545	0.0070168	0.0037514	0.0019319

4.3.2.3 Dataset 3 - Swarm-deconvoluted and normalized

```
# 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(),
      .groups="drop"
    )
```

```
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	909443.7	471717.7	437726.0	354455.0	117262.67	0.5078910	0.4921090	0.3900542	0.1178368
786-O	60	3	1647751.0	655903.7	991847.3	636368.7	19535.00	0.4257753	0.5742247	0.4163126	0.0094627
786-O	100	3	2280768.3	973655.3	1307113.0	917204.7	56450.67	0.5050553	0.4949447	0.4750143	0.0300410
786-O	140	3	3865308.0	1356040.0	2509268.0	1283623.0	72417.00	0.3642679	0.6357321	0.3416410	0.0226270
786-O	180	3	5600694.7	1925818.0	3674876.7	1785327.3	140490.67	0.3555459	0.6444541	0.3282978	0.0272481
786-O	220	3	4809437.0	1668469.3	3140967.7	1598712.3	69757.00	0.5634870	0.4365130	0.2208299	0.3426571
VHL	20	3	1165063.3	347512.3	817551.0	336533.7	10978.67	0.3095409	0.6904591	0.2977505	0.0117904
VHL	60	3	2495394.3	838712.0	1656682.3	818879.7	19832.33	0.3362616	0.6637384	0.3288166	0.0074450
VHL	100	3	3388536.0	1047547.3	2340988.7	1037150.3	10397.00	0.3085715	0.6914285	0.3060819	0.0024896
VHL	140	3	5181253.3	1555590.0	3625663.3	1534900.0	20690.00	0.3175128	0.6824872	0.3126651	0.0048476
VHL	180	3	5917869.3	1941989.7	3975879.7	1908202.7	33787.00	0.3338815	0.6661185	0.3284095	0.0054720
VHL	220	3	7054032.7	2653036.7	4400996.0	2494037.0	158999.67	0.3719500	0.6280500	0.3506715	0.0212785

```
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	73693.81	68505.24	46949.25	28247.05	48542.236	0.0472155	0.0472155	0.0013014	0.0465169
786-O	60	3	285338.25	68136.73	221723.36	60870.20	7637.202	0.0327439	0.0327439	0.0345697	0.0023072
786-O	100	3	436021.11	102258.91	333771.62	97510.80	5017.977	0.0699473	0.0699473	0.0653466	0.0046379
786-O	140	3	328218.74	69263.38	332046.24	79401.66	23240.399	0.0304969	0.0304969	0.0259862	0.0081036
786-O	180	3	827356.77	279477.21	583744.64	245138.20	49506.646	0.0242008	0.0242008	0.0166842	0.0079164
786-O	220	3	1388931.86	510144.61	913324.07	481434.28	38445.318	0.1277002	0.1277002	0.0657791	0.1898116
VHL	20	3	111933.68	25612.25	98344.63	25961.53	3265.883	0.0245818	0.0245818	0.0205196	0.0043361
VHL	60	3	114301.04	41583.84	80356.19	35138.83	6445.870	0.0086806	0.0086806	0.0076995	0.0022709
VHL	100	3	268818.05	99823.88	180908.15	93908.73	6002.711	0.0111077	0.0111077	0.0102756	0.0014373
VHL	140	3	719261.71	97877.12	621601.87	100624.28	7567.742	0.0207705	0.0207705	0.0197387	0.0021857
VHL	180	3	453866.14	145787.62	378391.81	132265.68	14056.373	0.0215534	0.0215534	0.0203520	0.0019739
VHL	220	3	217047.89	268117.43	104507.12	204292.73	64804.598	0.0263610	0.0263610	0.0181511	0.0082195

4.3.3 Building sample graphs

4.3.3.1 Dataset 1 - Raw gate counts

```
# 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)
)
```

4.3.3.2 Dataset 2 - Swarm-deconvoluted

```
# 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"
)
```

4.3.3.3 Dataset 3 - Swarm-deconvoluted and normalized

```
# 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"
)
```

4.3.4 Building Summary graphs

4.3.4.1 Dataset 1 - Raw gate counts

```
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)
)
```

4.3.4.2 Dataset 2 - Swarm-deconvoluted

```
# 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"
)
```

4.3.4.3 Dataset 3 - Swarm-deconvoluted and normalized

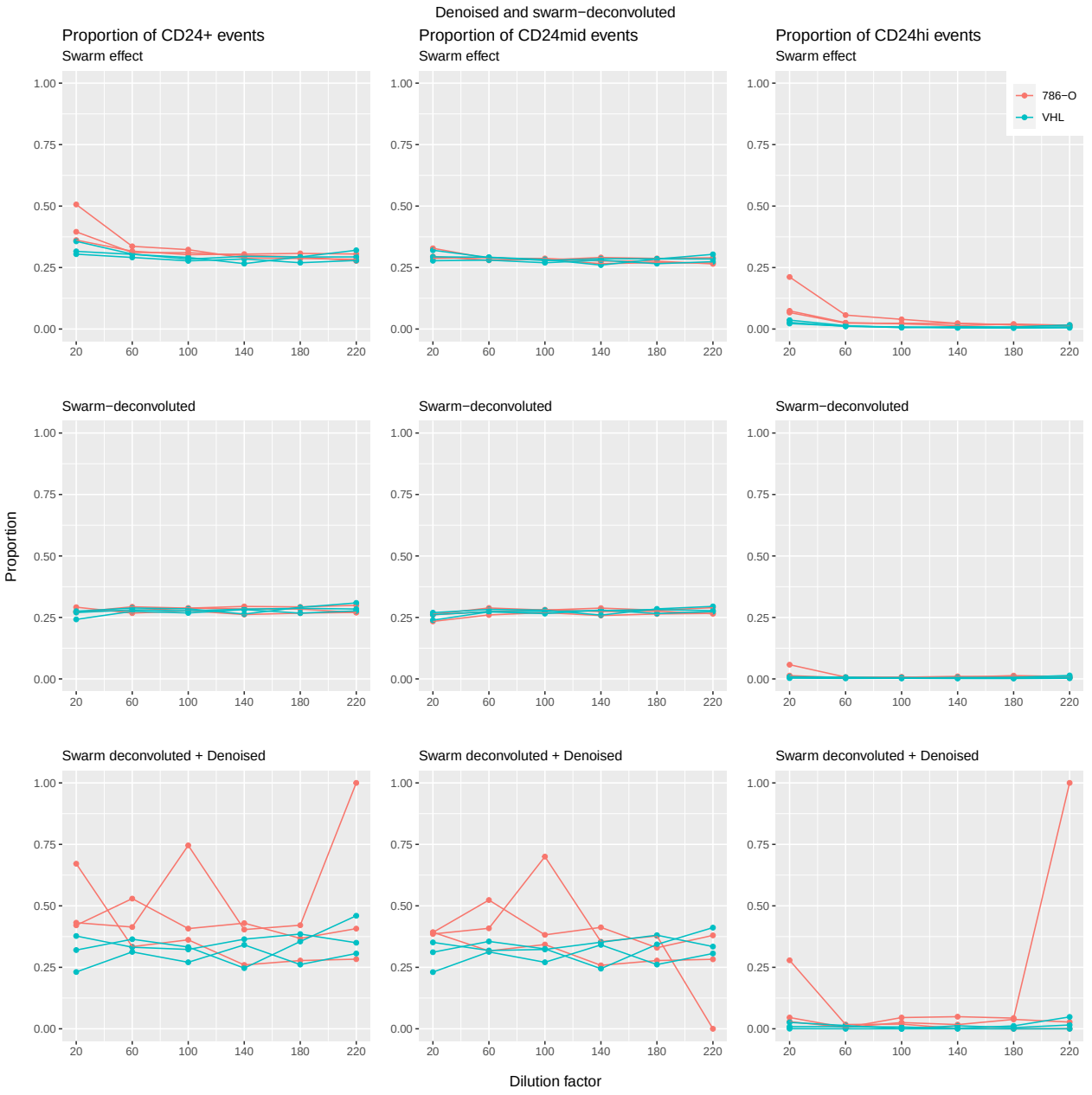
```r
# Let's generate plots to show the final result

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"
)
```

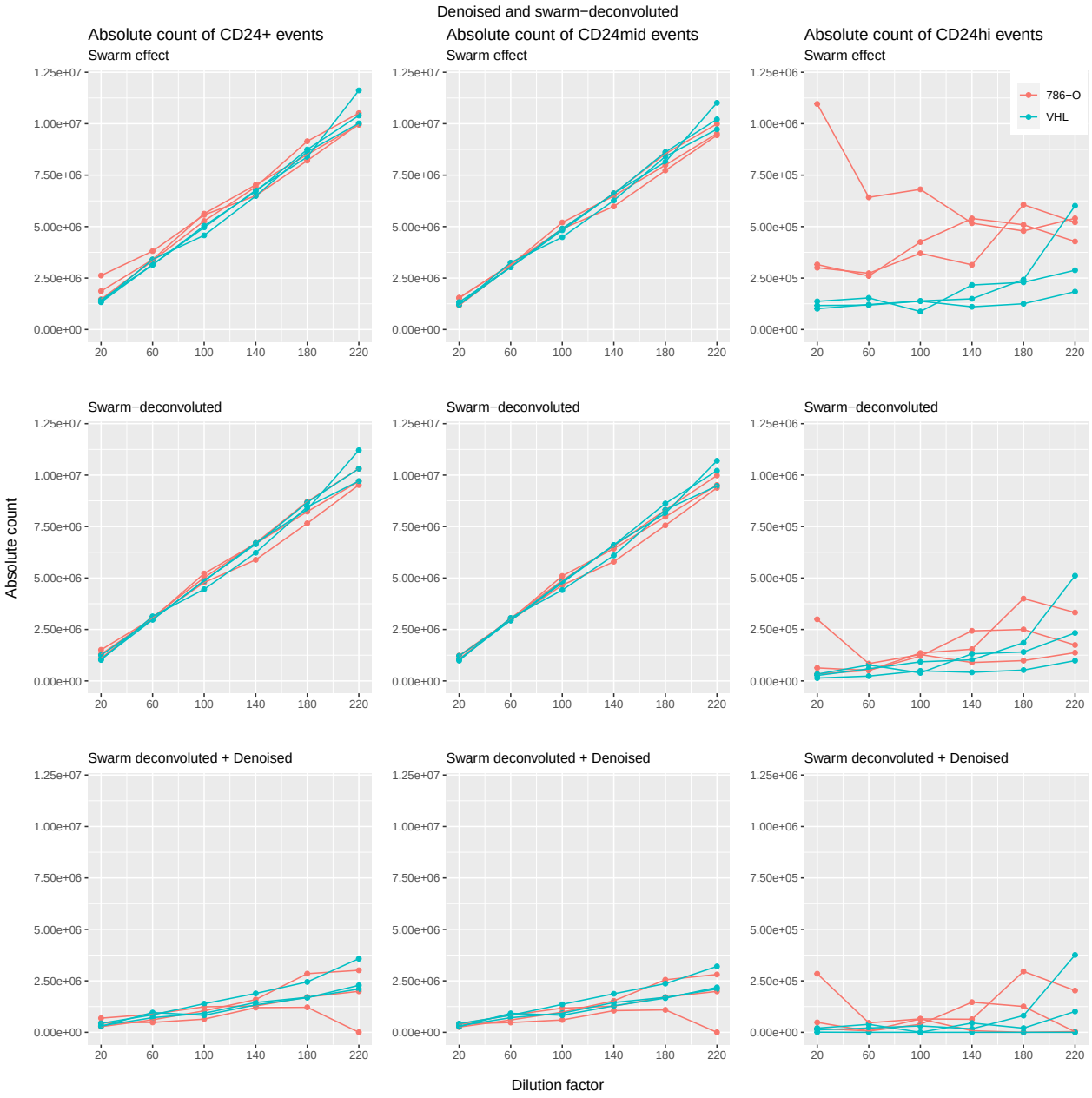
4.3.5 Visualizing the process

4.3.5.1 Sample graphs

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

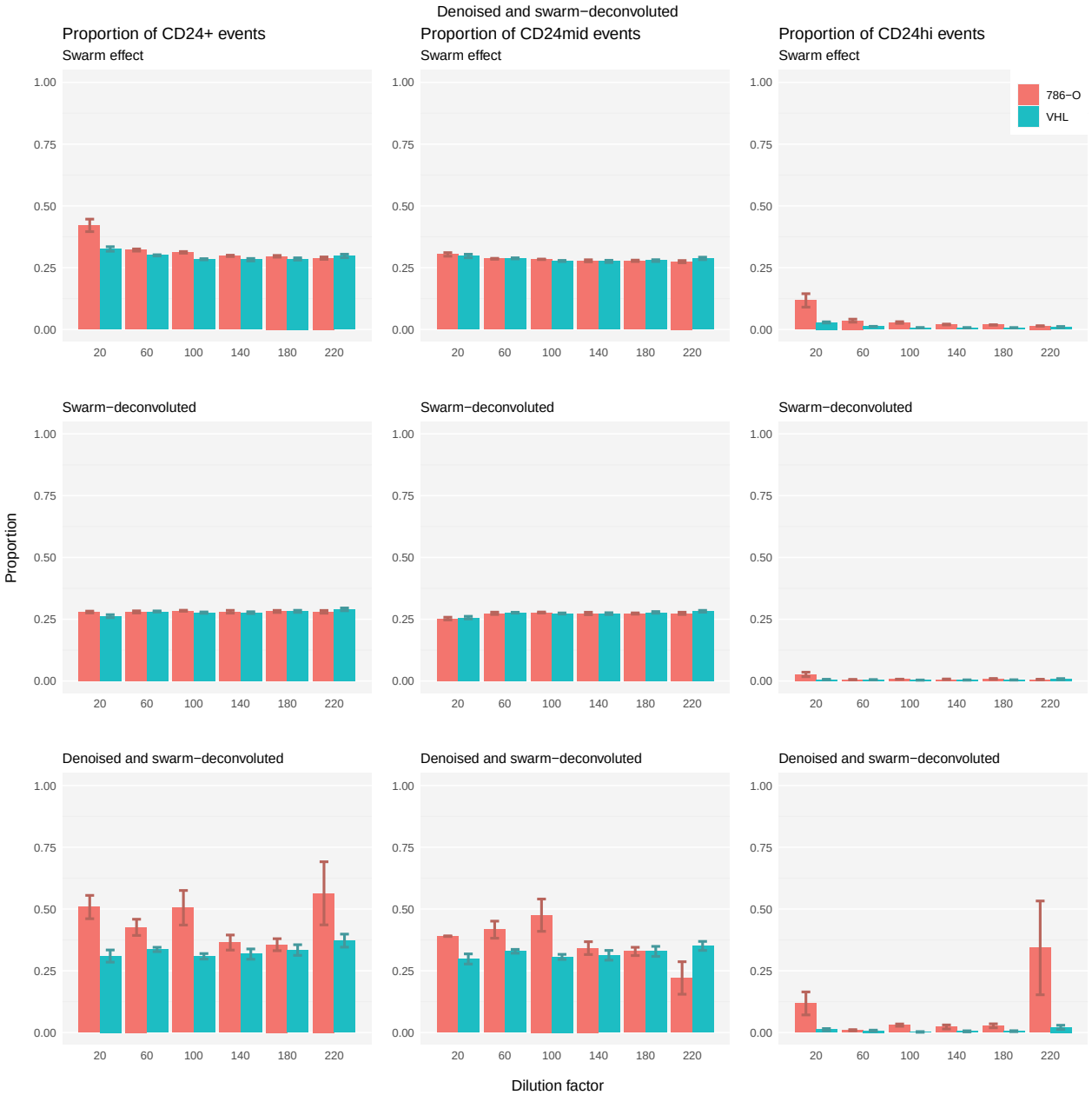


```
gridExtra::grid.arrange(
  plotA4, plotA5, plotA6,
  plotB4, plotB5, plotB6,
  plotC4, plotC5, plotC6,
  nrow=3,
  top="Denoised and swarm-deconvoluted",
  left="Absolute count",
  bottom="Dilution factor"
)
```



4.3.5.2 Summary graphs

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



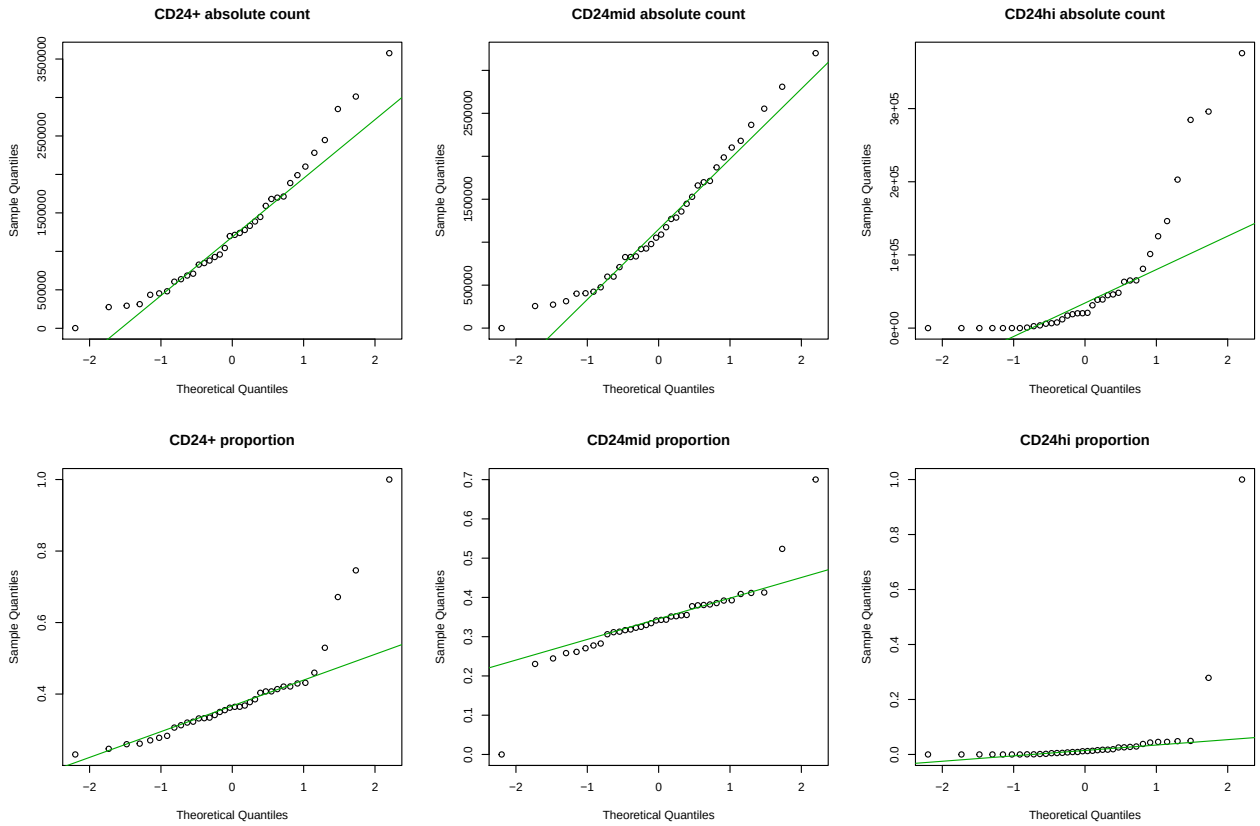
```
gridExtra::grid.arrange(
  plotD4, plotD5, plotD6,
  plotE4, plotE5, plotE6,
  plotF4, plotF5, plotF6,
  nrow=3,
  top="Denoised and swarm-deconvoluted",
  left="Absolute count",
  bottom="Dilution factor"
)
```

4.3.6 Significance testing

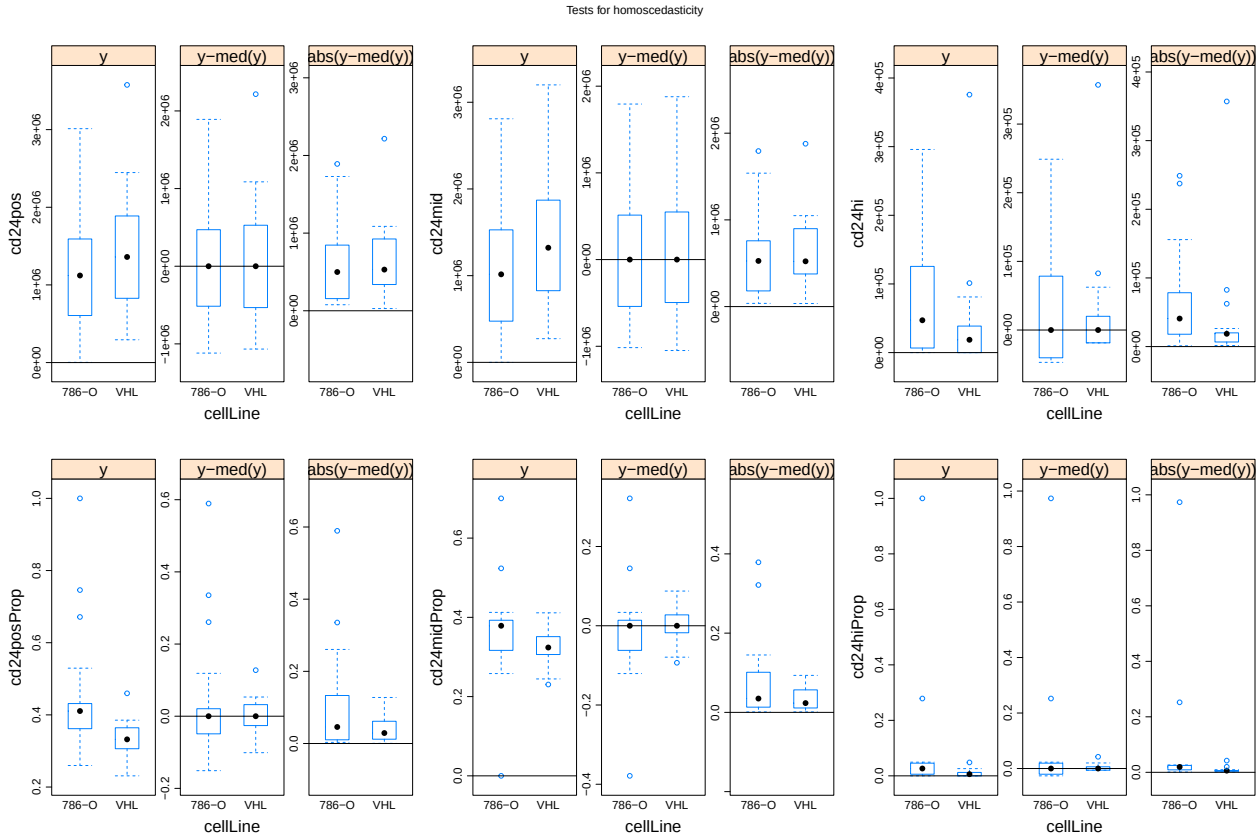
First we must verify that test assumptions are met.

```
# 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"]]
)
```

```r
# 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	0.0737589	0.7902400	0.8767720	0.7393307	0.6354020	0.3588705
cd24mid	0.1558384	0.7934795	0.9476231	0.7426260	0.0118552	0.3266931
cd24hi	0.0000002	0.3343487	0.7653241	0.1245900	0.3386673	0.0610075
cd24posProp	0.0000010	0.0723155	0.0000087	0.1613844	0.0028937	0.0029392

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

	Shapiro.Wilk	Brown.Forsythe	Bartlett	Fligner.Killeen	ANOVA	Kruskal.Wallis
cd24midProp	0.0000778	0.0928297	0.0000625	0.1447009	0.0663233	0.0462361
cd24hiProp	0.0000000	0.1847189	0.0000000	0.0057221	0.1421187	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	0.0737589	p < 0.10
cd24mid	Shapiro.Wilk	0.1558384	p < 0.20
cd24hi	Shapiro.Wilk	0.0000002	p < 0.01
cd24posProp	Shapiro.Wilk	0.0000010	p < 0.01
cd24midProp	Shapiro.Wilk	0.0000778	p < 0.01
cd24hiProp	Shapiro.Wilk	0.0000000	p < 0.01
cd24pos	Brown.Forsythe	0.7902400	p > 0.25
cd24mid	Brown.Forsythe	0.7934795	p > 0.25
cd24hi	Brown.Forsythe	0.3343487	p > 0.25
cd24posProp	Brown.Forsythe	0.0723155	p < 0.10
cd24midProp	Brown.Forsythe	0.0928297	p < 0.10
cd24hiProp	Brown.Forsythe	0.1847189	p < 0.20
cd24pos	Bartlett	0.8767720	p > 0.25
cd24mid	Bartlett	0.9476231	p > 0.25
cd24hi	Bartlett	0.7653241	p > 0.25
cd24posProp	Bartlett	0.0000087	p < 0.01
cd24midProp	Bartlett	0.0000625	p < 0.01
cd24hiProp	Bartlett	0.0000000	p < 0.01
cd24pos	Fligner.Killeen	0.7393307	p > 0.25
cd24mid	Fligner.Killeen	0.7426260	p > 0.25
cd24hi	Fligner.Killeen	0.1245900	p < 0.15
cd24posProp	Fligner.Killeen	0.1613844	p < 0.20
cd24midProp	Fligner.Killeen	0.1447009	p < 0.15
cd24hiProp	Fligner.Killeen	0.0057221	p < 0.01
cd24pos	ANOVA	0.6354020	p > 0.25
cd24mid	ANOVA	0.0118552	p < 0.05
cd24hi	ANOVA	0.3386673	p > 0.25
cd24posProp	ANOVA	0.0028937	p < 0.01
cd24midProp	ANOVA	0.0663233	p < 0.10
cd24hiProp	ANOVA	0.1421187	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	0.0462361	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))
```

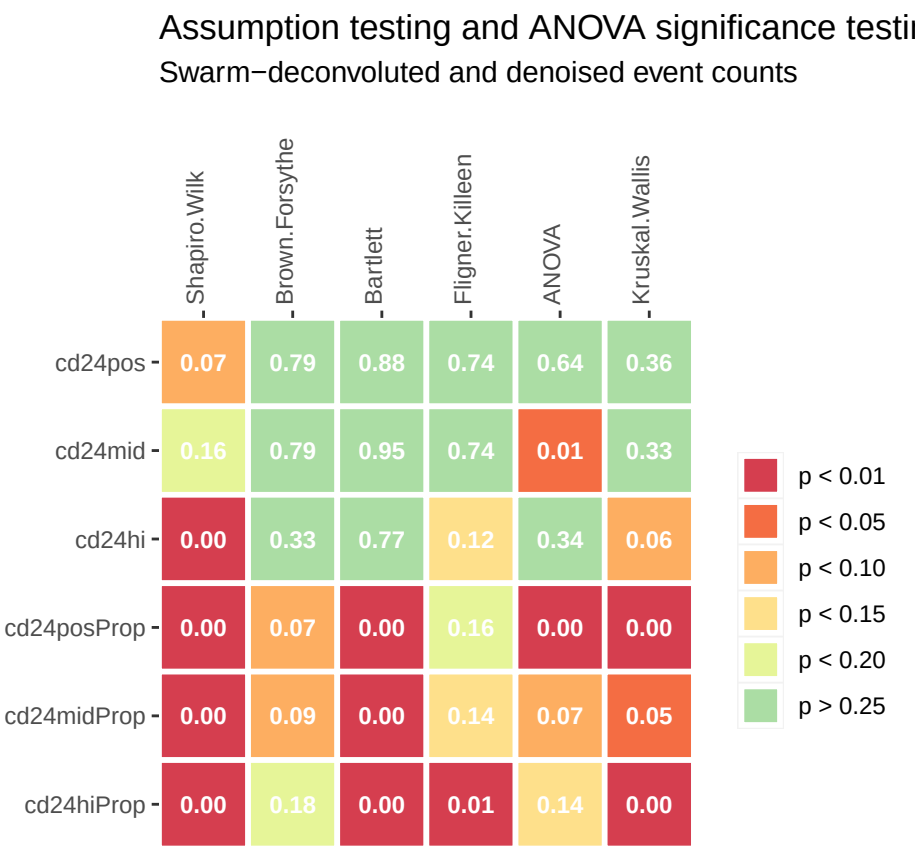
A Shapiro-Wilk test statistic > 0.05 indicates that the univariate normality condition is met. While normality is technically a test assumption, in practice the ANOVA is relatively robust to departures from normality.

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 0.01217  0.0118 0.91405           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    1.1806 0.28511           0.035
## dilution   1 34.648  34.648 3361.8541 0.00000          98.994
## Residuals 33  0.340   0.010                             0.972
```

```
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    1.1523 0.29111           0.035
## cd24midTp  1 16.5282 16.5282 1565.2728 0.00000          47.223
## dilution   1 18.1217 18.1217 1716.1825 0.00000          51.776
## Residuals 32  0.3379  0.0106                            0.965
```

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

decreaseInVarianceFromDilution
```

```
## [1] 0.4769724
```

```
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 17.046 17.0461  32.281 2.2329e-06          48.703
## Residuals 34 17.954  0.5281                             51.297
```

```
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 17.046 17.0461 32.9770 0.000002          48.703
## dilution   1  0.896  0.8960  1.7334 0.197050           2.560
```

```
## Residuals 33 17.058  0.5169                              48.737
```

```
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 17.0461 17.0461 33.8210 0.00000          48.703
## cd24hiTp   1  1.8256  1.8256  3.6222 0.06604           5.216
## dilution   1  0.0001  0.0001  0.0001 0.99081           0.000
## Residuals 32 16.1283  0.5040                          46.081
```

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

decreaseInVarianceFromDilution
```

```
## [1] 0.9999242
```

```
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  0.039 0.03852  0.0375 0.84768            0.11
## Residuals 34 34.961 1.02828                           99.89
```

```
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  0.039   0.039    2.3744 0.13288            0.11
## dilution   1 34.426  34.426 2121.9408 0.00000           98.36
## Residuals 33  0.535   0.016                              1.53
```

```
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  0.0385  0.0385    2.6164 0.11558            0.110
## cd24posTp  1 17.4329 17.4329 1184.0616 0.00000           49.808
## dilution   1 17.0575 17.0575 1158.5650 0.00000           48.736
## Residuals 32  0.4711  0.0147                              1.346
```

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

decreaseInVarianceFromDilution
```

```
## [1] 0.5045188
```

```
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)  0.0018  1    0.1689 0.68382           0.010
## cellLine     0.0131  1    1.2386 0.27404           0.071
## cd24midTp    0.0022  1    0.2088 0.65083           0.012
## dilution    18.1217  1 1716.1825 0.00000          98.079
## Residuals    0.3379 32                             1.829
```

```
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)  0.4395  1  0.8719 0.35741           1.324
## cellLine    15.6897  1 31.1299 0.00000          47.276
## cd24hiTp     0.9297  1  1.8446 0.18392           2.801
## dilution     0.0001  1  0.0001 0.99081           0.000
## Residuals   16.1283 32                         48.598
```

```
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)  0.0706  1    4.7940 0.03596           0.399
## cellLine     0.0047  1    0.3225 0.57404           0.027
## cd24posTp    0.0643  1    4.3642 0.04474           0.364
## dilution    17.0575  1 1158.5650 0.00000          96.543
## Residuals    0.4711 32                            2.667
```

5 References