# JT AFC 05/02/2020 - cytofkit / cytofast - v2

S Granjeaud - A Meghraoui

05/02/2020

# Goals

Here is a notebook to perform

- a multi-parametric cytometry analysis using **cytofkitlab** (a fork of cytofkit)
- a standard differential analysis of percentages using the **cytofast** framework

The cytofkit analysis aims at clustering the cells and visualizing them in a two-dimensional map. This is similar to what you, experts of cytometry, are usually carrying out using standard or commercial software. Just imagine that the gating is automated by clustering approaches and that new axes resulting from dimension redcution techniques are presented on the classical bi-paramteric graph to help visualization. If you get those two points, you got the big picture of all staff here.

The cytofast analysis aims at finding differences between groups of samples using their percentages of cells in each clusters. The percentages are extracted from the FCS files exported at the end of the cytofkit analysis. Some new channels have been added to those FCS in order to report clustering and dimensions reduction results.

# Check installed packages

As a reminder, install the required packages.

```
# install devtools if not already done
if (!requireNamespace("devtools", quietly = TRUE)) install.packages("devtools")
# install Rphenograph if not already installed
if (!requireNamespace("Rphenograph", quietly = TRUE))
  devtools::install_github("i-cyto/Rphenograph")
# install cytofkitlab
devtools::install_github("i-cyto/cytofkitlab")
# install cytofast
if (!requireNamespace("cytofast", quietly = TRUE)) {
  BiocManager::install("cytofast")
}
```

You could check your installation by runnning the following chunk (ie set of commands).

```
err <- FALSE
for (pkg in c("cytofkitlab", "Rphenograph", "FlowSOM", "cytofast")) {
  if (!suppressWarnings(requireNamespace(pkg, quietly = TRUE))) {
    warning(pkg, " must be installed.")
    err <- TRUE
  }
}
if (!err) message("Your installation seems correct.")
```

```
## Your installation seems correct.
```

# cytofkit(lab) analysis (Via Graphical User Interface)

This part concerns the cytofkit analysis. This step is typically carried out using the graphical interface. If you want to repeat the analysis by changing parameters or automate the analysis for a few experiments, you will be interested in learning a few R commands at the end of this document.

# Computation first

Typical way to carry the analysis relies on R/Rstudio in which you enter the follwing commands.

```
library(cytofkitlab)
# Launch the Graphical User Interface for tuning the run
cytofkit_GUI()
# Note the path to the result file
```

# Exploration then

You could run the interactive exploration and analysis of the clustering and dimension reduction results using the Shiny application.

```
# Launch the Shiny interface to view and annotate the analysis
cytofkitShinyAPP()
```

If you know **exactly** the path to the results, you can run the Shiny application to explore a given RData file that holds the results.

```
# Launch the Shiny interface using a defined path
analysis_file = "c:/demo/200205-atelier/CLEAN_DATA_results/run_5k/run_5k.RData"
if (file.exists(analysis_file))
  cytofkitShinyAPP(analysis_file)
```

# Cytofast post-analysis

This part concerns the cytofast analysis. This step needs to be run without graphical interface. You will have to understand the steps of the analysis, to read the R code in order to change some parts in order to define the group comparison that are relevant to your analysis.

# Configuration 1: file location and clustering

## FCS location

Locate the directory (aka folder) where the FCS files of exported from cytofkit are. This directory ends typically with "_analyzedFCS".

Either using a GUI; select one of the exported FCS file.

```
# use a GUI to locate the FCS files that were exported from the cytofkit analysis or explo
ration
analysis_fcs_dir = dirname(chooseFiles_GUI())
```

Or by command lines; you need to know exactly where files are located.

```
# Main directory of all results
res_dir = "c:/demo/200205-atelier/CLEAN_DATA_results"
# Append project name and the analyzed FCS dir
analysis_fcs_dir = file.path(res_dir, "cytofkitlab_5k", "cytofkitlab_5k_analyzedFCS")
stopifnot(dir.exists(analysis_fcs_dir))  # check dir exists
```

## Clustering to use

The cytofkit exploration using the Shiny GUI may result in many clustering and manual grouping. In the following analysis you have to select one of them.

```
# Select the clustering to use
selected_clustering_method = "FlowSOM_clusterIDs"
#selected_clustering_method = "Rphenograph_clusterIDs"
```

# Import the FCS files exported by cytofkit into cytofast

```
# Load libraries
library(cytofast)
library(cytofkitlab)  # allow reading FCS for cytofast objects

# Load cytofkit FCS files
cfData = readCytofkitFCS(analysis_fcs_dir, clusterID = selected_clustering_method)
```

```
## Reading .FCS sample: 1
```

```
## Reading .FCS sample: 2
```

```
## Reading .FCS sample: 3
```

```
## Reading .FCS sample: 4
```

```
## Reading .FCS sample: 5
```

```
## Reading .FCS sample: 6
```

```
# cfData = readCytofkitFCS("demo/cytofkit_demo_analyzedFCS", clusterID = "Rphenograph_clus
terIDs")
table(cfData@expr$clusterID)
```

```
##
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
## 6567 1335   34  287  739  538  212  187  220  289  147   81 2655  236  183  239
##   17   18   19   20   21   22   23   24   25   26   27   28   29   30   31   32
##   60  632  442  122  119 2456  102  924  213  994  151  304  278  532  602 5168
##   33   34   35   36   37   38   39   40
##   56  528  810  535    9  381  254  379
```

Optionnally, do some channel cleaning and ordering

```
# Remove unneeded channels
keep <- colnames(cfData@expr)  # all
keep <- keep[!grepl("^(Time|Event_length|Cell_length)", keep, ignore.case = TRUE)]
keep <- keep[!grepl("^(Center|Offset|Width|Residual)", keep, ignore.case = TRUE)]
keep <- keep[!grepl("^(Cisplatin)", keep, ignore.case = TRUE)]
keep <- keep[!grepl("(_ADN1|_ADN2)", keep, ignore.case = TRUE)]
keep <- keep[!grepl("(_DNA1|_DNA2)", keep, ignore.case = TRUE)]
keep <- keep[!grepl("(_Bead)", keep, ignore.case = TRUE)]
keep <- keep[!grepl("^(NA\\.)", keep, ignore.case = TRUE)]  # flowCore bug on DNA channel
keep <- keep[!grepl("^X\\d+", keep) | grepl("_", keep)]  # unannotated X channel
cfData@expr <- cfData@expr[, keep]
keep
```

```
##  [1] "clusterID"  "sampleID"   "CCR4.149"   "CCR5.144"   "CCR7.159"
##  [6] "CD16.165"   "CD19.142"   "CD20.147"   "CD27.167"   "CD28.160"
## [11] "CD3.170"    "CD33.158"   "CD38.148"   "CD4.145"    "CD43.150"
## [16] "CD44.166"   "CD45.154"   "CD45RA.153" "CD45RO.164" "CD56.176"
## [21] "CD69.162"   "CD8.168"    "CXCR3.156"  "CXCR5.171"  "HLA.DR.163"
## [26] "ICOS.141"   "PD.1.174"   "PD.L1.175"  "TCRgd.152"  "TIM3.143"
```

```
# Rename channels
colnames(cfData@expr) <- gsub("(^X\\d+.+?_)", "", keep)  # remove metal tag
colnames(cfData@expr) <- gsub("(\\.\\d+.$)", "", colnames(cfData@expr))  # remove metal ta
g

# Manual ordering of channels
first <- c("CD20", "CD19", "CD3", "CD4", "CD8", "TCRgd", "CD56", "CD16", "CD45RA", "CD95",
"CD127")
setdiff(first, colnames(cfData@expr))  # not found in the FCS
```

```
## [1] "CD95"  "CD127"
```

```
first <- intersect(first, colnames(cfData@expr))

keep <- colnames(cfData@expr)  # all remaining
# Here are the channels in the final order
final_channels <- c(first, setdiff(keep[-(1:2)], first))
final_channels
```

```
##  [1] "CD20"   "CD19"   "CD3"    "CD4"    "CD8"    "TCRgd"  "CD56"   "CD16"
##  [9] "CD45RA" "CCR4"   "CCR5"   "CCR7"   "CD27"   "CD28"   "CD33"   "CD38"
## [17] "CD43"   "CD44"   "CD45"   "CD45RO" "CD69"   "CXCR3"  "CXCR5"  "HLA.DR"
## [25] "ICOS"   "PD.1"   "PD.L1"  "TIM3"
```

```
# select and reorder channels
cfData@expr <- cfData@expr[, c(keep[1:2], final_channels)]
```

# Configuration 2: Define meta data grouping FCS samples

To compute statistical tests, we need to define groups. This grouping comes from extra data called meta data. The user has two options to define this information. Then this meta-data is associated into the analysis workflow.

```
# Here are the un-annotated samples
cfData@samples
```

```
##          sampleID           FCSfilename
## clean_D1 clean_D1 cytofkit_clean_D1.fcs
## clean_D2 clean_D2 cytofkit_clean_D2.fcs
## clean_D3 clean_D3 cytofkit_clean_D3.fcs
## clean_P1 clean_P1 cytofkit_clean_P1.fcs
## clean_P2 clean_P2 cytofkit_clean_P2.fcs
## clean_P3 clean_P3 cytofkit_clean_P3.fcs
```

## Option 1: Create a template and fulfill it with Excel

The user has to fill the annotation using an Excel template. The template consists in a first column of sample identifiers and a second column called “status”.

```
meta <- data.frame(cfData@samples, status = "")
# The template is written on disk
if (!file.exists("meta.csv")) {
  write.csv(meta, "meta.csv")
} else {
  message("meta.csv already exists and will not be over-written.")
}
```

```
## meta.csv already exists and will not be over-written.
```

Now the user can annotate the template using Excel or so, and store it as CSV file keeping the original format (ie CSV).

```
# Import annotation
if (file.exists("meta.csv")) {
  meta <- read.csv("meta.csv", row.names = 1)
}
```

## Option 2: Create meta data using R commands

Either we fill meta data programmatically.

```
meta$status <- c(rep(c("D", "P"), each = 3))
# write a copy to disk
write.csv(meta, "meta.csv")
```

## Finally, associate annotations to data

Once grouping has been defined, we associate it with the samples.

```
# meta <- meta[match(row.names(cfData@samples), meta[,"sampleID"]),] # match sampleID
cfData@samples <- cbind.data.frame(cfData@samples, meta[, -1, drop = FALSE])
# remove duplicated columns
cfData@samples <- cfData@samples[,!duplicated(colnames(cfData@samples)), drop = FALSE]
# complete annotation
cfData@samples
```

```
##          sampleID           FCSfilename status
## clean_D1 clean_D1 cytofkit_clean_D1.fcs      D
## clean_D2 clean_D2 cytofkit_clean_D2.fcs      D
## clean_D3 clean_D3 cytofkit_clean_D3.fcs      D
## clean_P1 clean_P1 cytofkit_clean_P1.fcs      P
## clean_P2 clean_P2 cytofkit_clean_P2.fcs      P
## clean_P3 clean_P3 cytofkit_clean_P3.fcs      P
```

**NB1**: The annotation file could describes various sample grouping. Just use many columns. In the following we use the grouping defined in column “status”, and select the group called “D” as reference.

**NB2**: CSV file format is typically using “,” as spearator, but in Europe, this separator has been replaced by “;”. Instead of calling functions XXXX.*CSV*, use XXXX.*CSV_2_**

# Process MFI and counts

## Configuration 3: asinh cofactor

Adjust the cofactor for asinh transform. Typically it’s 5 for mass cytometry (the default). It should be set to 150 or 250 for flow cytometry.

```
asinh_cofactor <- 5
```

## Transform the intensity

Transform expression values, in the same way you display marker intensity

```
# Transform expression with asinh( x / cofactor )
cfData@expr[,-(1:2)] <- asinh(cfData@expr[,-(1:2)] / asinh_cofactor)
```

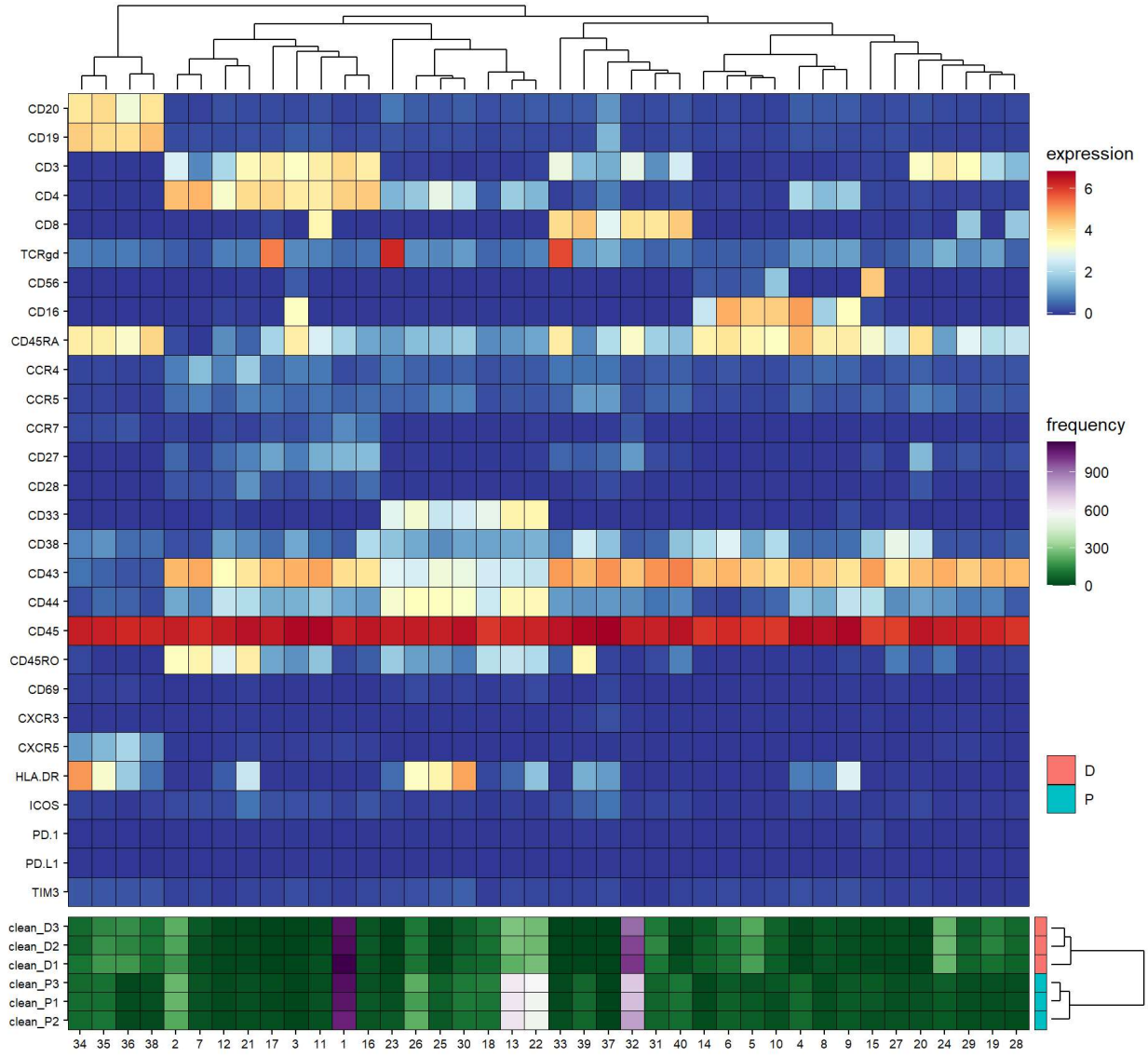
## Transform the cell counts

Here the overview of raw cell counts.

```
# Retrieve cell counts
cfData <- cellCounts(cfData)
head(cfData@counts)
```

```
##             1   2 3  4   5   6  7  8  9 10 11 12  13 14 15 16 17  18  19 20 21
## clean_D1 1142 178 8  9 200 107 29 37 24 59 29  9 235 69 19 36  0  86  99  8 11
## clean_D2 1106 208 1 11 183 113 30 42 25 64 21  9 273 66 20 47  1 100 139 11 12
## clean_D3 1087 231 6 19 219 133 46 42 27 71 13 12 283 50 21 39  2 107 125  5 10
## clean_P1 1081 247 7 86  48  66 35 22 50 26 33 18 622 17 47 44 13 110  24 34 28
## clean_P2 1053 221 7 84  55  68 34 23 48 27 29 14 633 21 28 40 23  98  26 34 26
## clean_P3 1098 250 5 78  34  51 38 21 46 42 22 19 609 13 48 33 21 131  29 30 32
##           22 23  24 25  26 27 28 29  30  31   32 33  34  35  36 37  38 39  40
## clean_D1 271  3 254 13 107 15 71 73  72 114 1008  1 109 182 180  0 101  6  26
## clean_D2 256  1 262 11 115 21 78 73  57 149  984  0  72 168 144  2 109  3  13
## clean_D3 264  3 249 14 115 22 81 61  63 119  918  1  86 150 161  3 108  5  29
## clean_P1 558 29  53 58 193 29 31 25 120  74  737 14  91  98  21  1  23 83 104
## clean_P2 542 45  48 47 234 34 21 24 100  74  810 26  86 111  19  2  14 78  93
## clean_P3 565 21  58 70 230 30 22 22 120  72  711 14  84 101  10  1  26 79 114
```

```
# View(cfData@counts)
cytoHeatmaps(cfData, group="status", legend=TRUE)
```



Here we see high and low counts. But what is of interest is the relative change of cell count within each cluster. We don't need to compare the absolute cell counts between clusters. So cell counts need to be transformed.

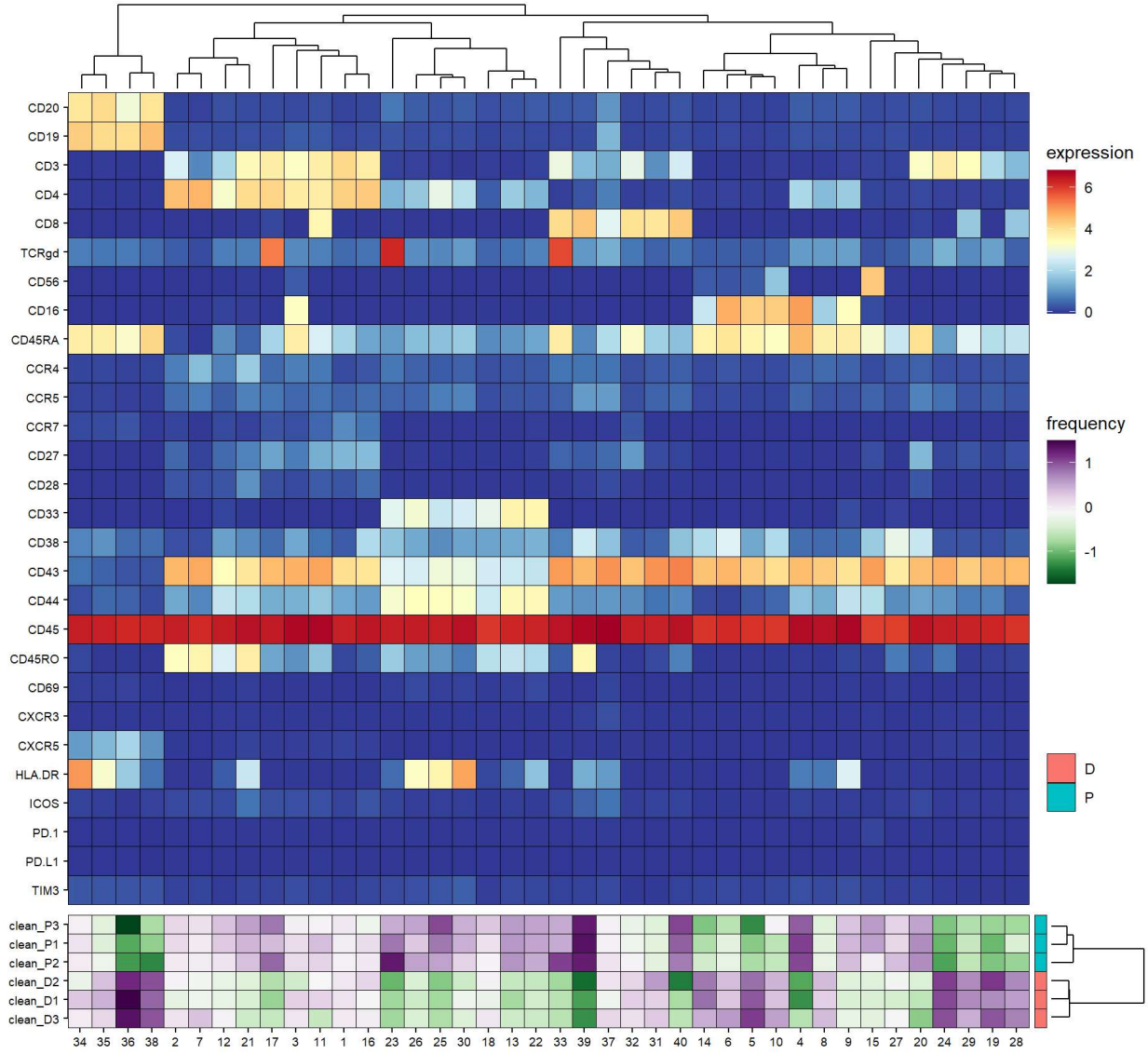
We transform counts with logarithm base 2 function and then centering within each cluster. The log2 transform means that a fold increase of 2 is +1, and a fold decrease of 2 is -1. Such a transformation compress the dynamic range and permits symmetric comparison among many of its properties. Then, because we are still not interested in the absolute values of the log2(cell counts), we center them to their average.

```
# Store raw counts
cellCountRaw <- cfData@counts

# Transform cell counts with log2( x + floor )
cfData@counts <- log2(cfData@counts+10)  # 10 cell floor

# Center to average
cfData@counts <- sweep(cfData@counts, 2, colMeans(cfData@counts), "-")
round(head(cfData@counts))
```

```
##          1 2 3  4  5  6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
## clean_D1 0 0 0 -1  1  0 0 0 0  0  0  0 -1  1  0  0 -1  0  1 -1  0  0 -1  1 -1
## clean_D2 0 0 0 -1  1  0 0 0 0  0  0  0 -1  1  0  0 -1  0  1  0  0 -1 -1  1 -1
## clean_D3 0 0 0 -1  1  1 0 0 0  1 -1  0 -1  0  0  0 -1  0  1 -1 -1 -1 -1  1 -1
## clean_P1 0 0 0  1 -1  0 0 0 0 -1  0  0  1 -1  1  0  0  0 -1  1  0  1  1 -1  1
## clean_P2 0 0 0  1 -1  0 0 0 0 -1  0  0  1  0  0  0  1  0 -1  1  0  0  1 -1  1
## clean_P3 0 0 0  1 -1 -1 0 0 0  0  0  0  1 -1  1  0  1  0 -1  1  1  1  0 -1  1
##          26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
## clean_D1 -1  0  1  1  0  0  0 -1  0  0  2  0  1 -1 -1
## clean_D2  0  0  1  1 -1  1  0 -1  0  0  1  0  1 -1 -1
## clean_D3  0  0  1  0  0  0  0 -1  0  0  1  0  1 -1 -1
## clean_P1  0  0  0 -1  0  0  0  0  0  0 -1  0 -1  1  1
## clean_P2  1  0 -1 -1  0  0  0  1  0  0 -1  0 -1  1  1
## clean_P3  1  0 -1 -1  0  0  0  0  0  0 -2  0 -1  1  1
```

# Heatmap centered to overall average

```
cytoHeatmaps(cfData, group="status", legend=TRUE)
```




Here we get a finer view of the data.

# Differential analysis and visualization

Differential analysis is usually carried between 2 conditions, one of them being the reference.

## Configuration 4: reference group

Let's take status D as reference.

```
# Define the reference group of the status column
reference_status <- "D"
```

## Centering to the reference

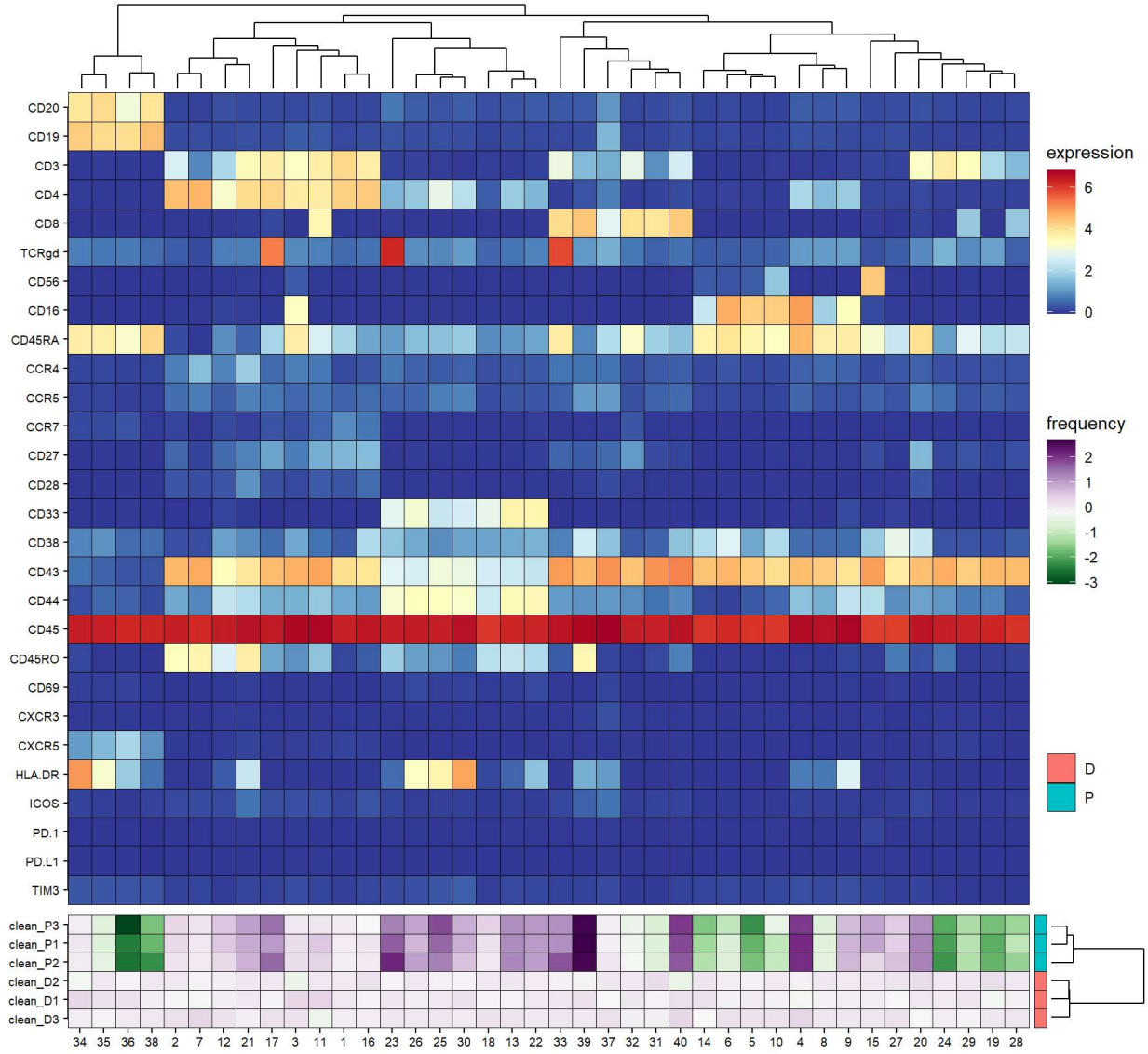
Here we consider that there is a reference group of samples. So this group is the reference for all variations that will be observed and analyzed. All the cell counts will be centered to its average (remember we are now working with in logarithm space, log(P) - log(Ref) = log (P / Ref)). The values of the samples of the reference group are also centered. This allows capturing the dispersion among the reference group, which is a good quality indicator.

```
# Center to the reference group
cfData@counts <- sweep(cfData@counts, 2,
                       colMeans(cfData@counts[cfData@samples$status == reference_statu
s,]), "-")
# View top 10 lines of the result
round(head(cfData@counts)*10)
```

```
##           1  2  3  4   5   6  7  8  9  10 11 12 13  14 15 16 17 18  19 20 21 22
## clean_D1  0 -2  3 -3   0  -1 -2 -1 -1  -1  4 -1 -2   2  0 -1 -1 -2  -3  0  0  0
## clean_D2  0  0 -4 -1  -1   0 -1  0  0   0  0 -1  1   1  0  2  0  0   2  2  1  0
## clean_D3  0  2  1  4   1   2  3  0  1   1 -4  1  1  -2  0  0  1  1   1 -2 -1  0
## clean_P1  0  3  2 21 -19  -7  0 -7  8 -10  5  5 12 -14  9  1 11  2 -19 13  9 11
## clean_P2 -1  1  2 21 -17  -7  0 -6  7 -10  4  3 12 -12  3  0 16  0 -19 13  8 10
## clean_P3  0  3  0 20 -23 -11  1 -7  7  -5  1  5 12 -16 10 -2 15  4 -17 12 10 11
##          23  24 25 26 27  28  29 30 31 32 33 34 35  36 37  38 39 40
## clean_D1  1   0  0 -1 -2  -1   1  2 -1  1  0  3  1   2 -2  -1  1  2
## clean_D2 -2   0 -1  0  1   0   1 -1  2  0 -1 -3  0  -2  0   0 -2 -5
## clean_D3  1   0  1  0  1   1  -2  0 -1 -1  0  0 -1   0  2   0  0  3
## clean_P1 17 -21 16  7  4 -11 -12  8 -7 -4 12  0 -7 -25 -1 -18 27 18
## clean_P2 22 -22 13 10  6 -15 -12  6 -7 -3 18  0 -5 -26  0 -23 26 17
## clean_P3 13 -20 18 10  5 -14 -13  8 -7 -4 12 -1 -7 -31 -1 -17 26 20
```

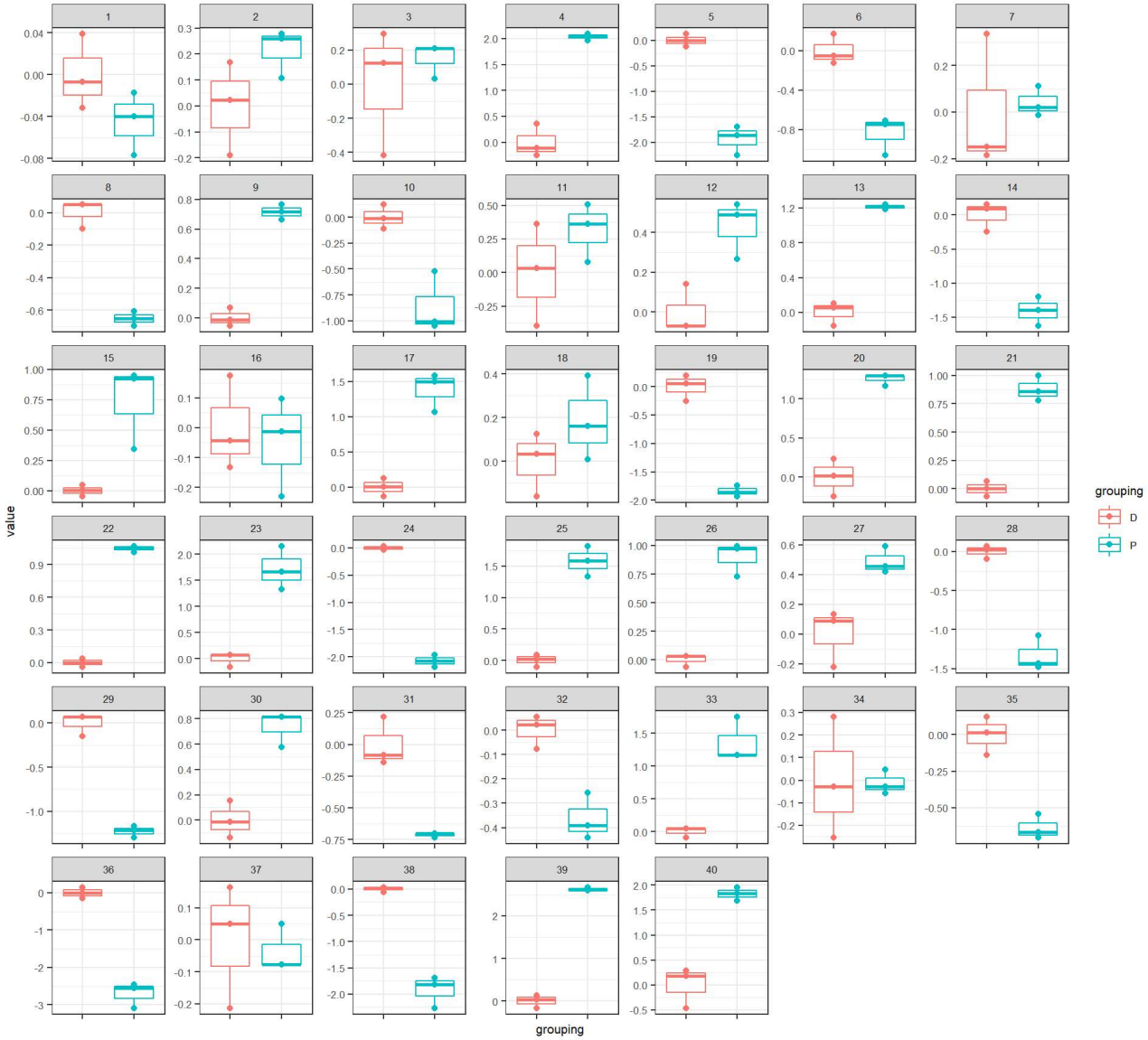
# Heatmap centered to the reference average

```
# Heatmap
cytoHeatmaps(cfData, group="status", legend=TRUE)
```




# Box plots

```
# Detailed view of counts aka percentages
cytoBoxplots(cfData, group = "status")
```




# Intensity distribution of functional markers

You can view many markers of you experiments

```
# List of markers

# Detailed of functional markers
msiPlot(cfData, markers = c("CD8", "CD4"), byGroup="status")
```

# Configuration 5: thresholds

Define your thresholds to extract the meaningful clusters of cells you want to focus on. Here we compromise between p-value and fold change.

```
# Thresholds for table and extract list of clusters
# p-value scale, although presented in logarithmic scale on Volcano
pvalue.cut <- 0.05
# fold change, either increase or decrease
# a value of 3 means a fold change of 3 ie 300% means that the value is multiplied or divi
ded by 3
# 3 means an increase of +200% or a deacrease of -66%
logFold.cut <- log2(3)  # log fold change
```

# Add statistical information

Carry out statistical computations and add fold changes information.

- diff is the difference between log2(counts), so it is the log2 fold change.
- fold is a multiplicative coefficient applied to the reference. It's the fold change, without the log2 part. When fold is negative to stand for 1/fold.

```
# Add t.test
cfData <- cytottest(cfData, group = "status", adjustMethod = "fdr")
# Construct the formula for adding fold change
coln <- colnames(cfData@results)
if (coln[2] == paste0("mean_", reference_status)) {
  diff_expr <- paste0(coln[3], "-", coln[2])
} else {
  diff_expr <- paste0(coln[2], "-", coln[3])
}
# Add some columns
cfData@results <- within(cfData@results, {
  diff <- eval(parse(text = diff_expr))
  fold <- sign(diff) * round(2^abs(diff), 2)
  unselected <- abs(diff) < logFold.cut | pvalue > pvalue.cut
  label <- clusters
  label[unselected] <- NA
  })
# overview of the statistical table
head(cfData@results)
```

```
##   clusters        mean_D      mean_P      pvalue   adjusted label unselected
## 1        1  3.469447e-18 -0.04476356 0.175361071 0.21255887  <NA>       TRUE
## 2        2 -1.387779e-17  0.21469296 0.164409488 0.20551186  <NA>       TRUE
## 3        3 -1.387779e-17  0.15081989 0.559345118 0.62149458  <NA>       TRUE
## 4        4  7.400583e-17  2.03358568 0.006313592 0.01262718     4      FALSE
## 5        5 -3.700969e-17 -1.93537531 0.002766062 0.00944703     5      FALSE
## 6        6  0.000000e+00 -0.83622783 0.005054980 0.01106549  <NA>       TRUE
##    fold        diff
## 1 -1.03 -0.04476356
## 2  1.16  0.21469296
## 3  1.11  0.15081989
## 4  4.09  2.03358568
## 5 -3.82 -1.93537531
## 6 -1.79 -0.83622783
```

# Volcano plot

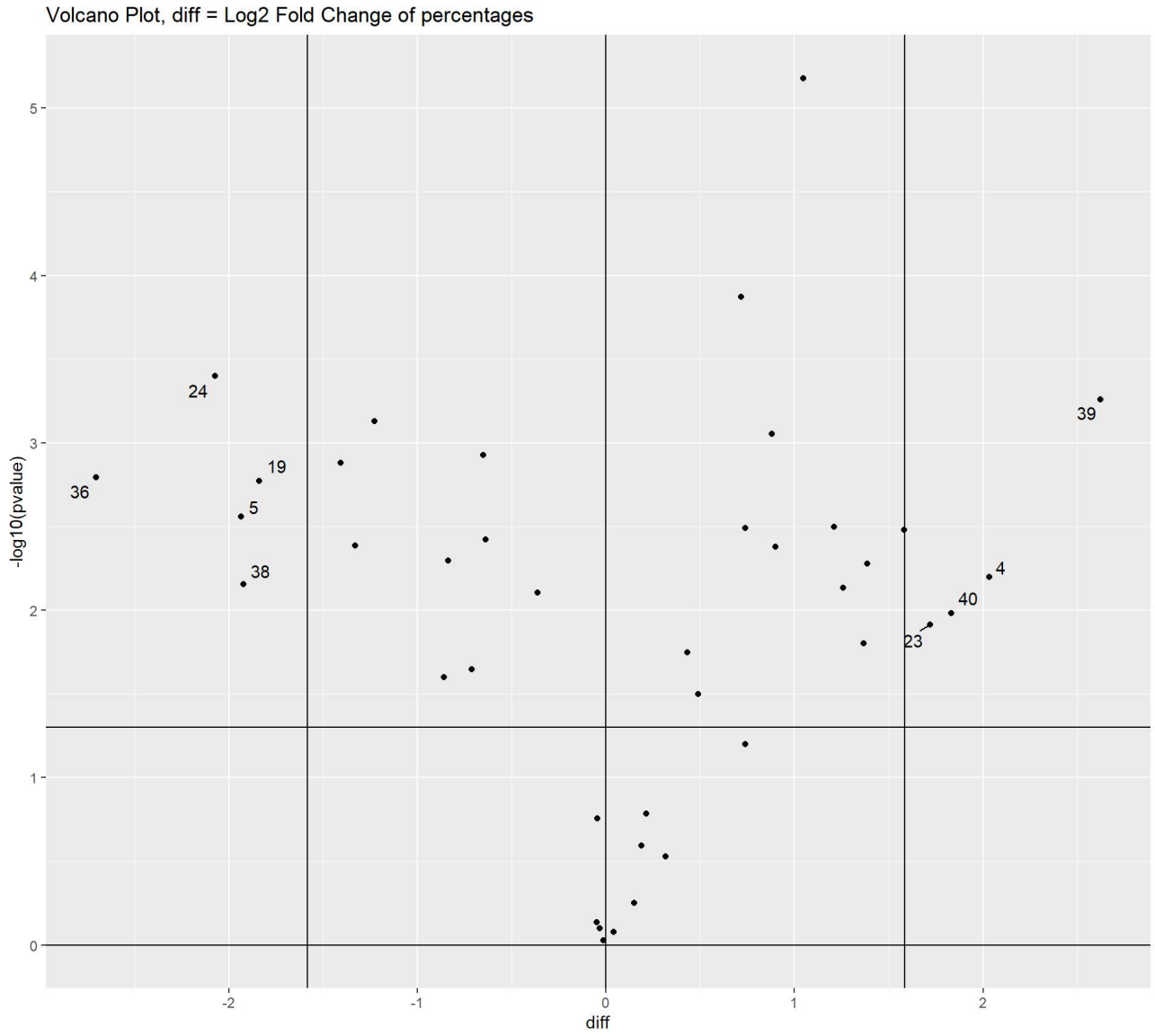
```
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:cytofast':
##
##     expr
```

```
library(ggrepel)
set.seed(42)

p <- ggplot(cfData@results, aes(diff, -log10(pvalue))) + geom_point() +
  geom_hline(yintercept = -log10(c(pvalue.cut, 1))) +
  geom_vline(xintercept = c(-logFold.cut, 0 , logFold.cut)) +
  ggtitle("Volcano Plot, diff = Log2 Fold Change of percentages")
p + geom_text_repel(aes(label = label))
```

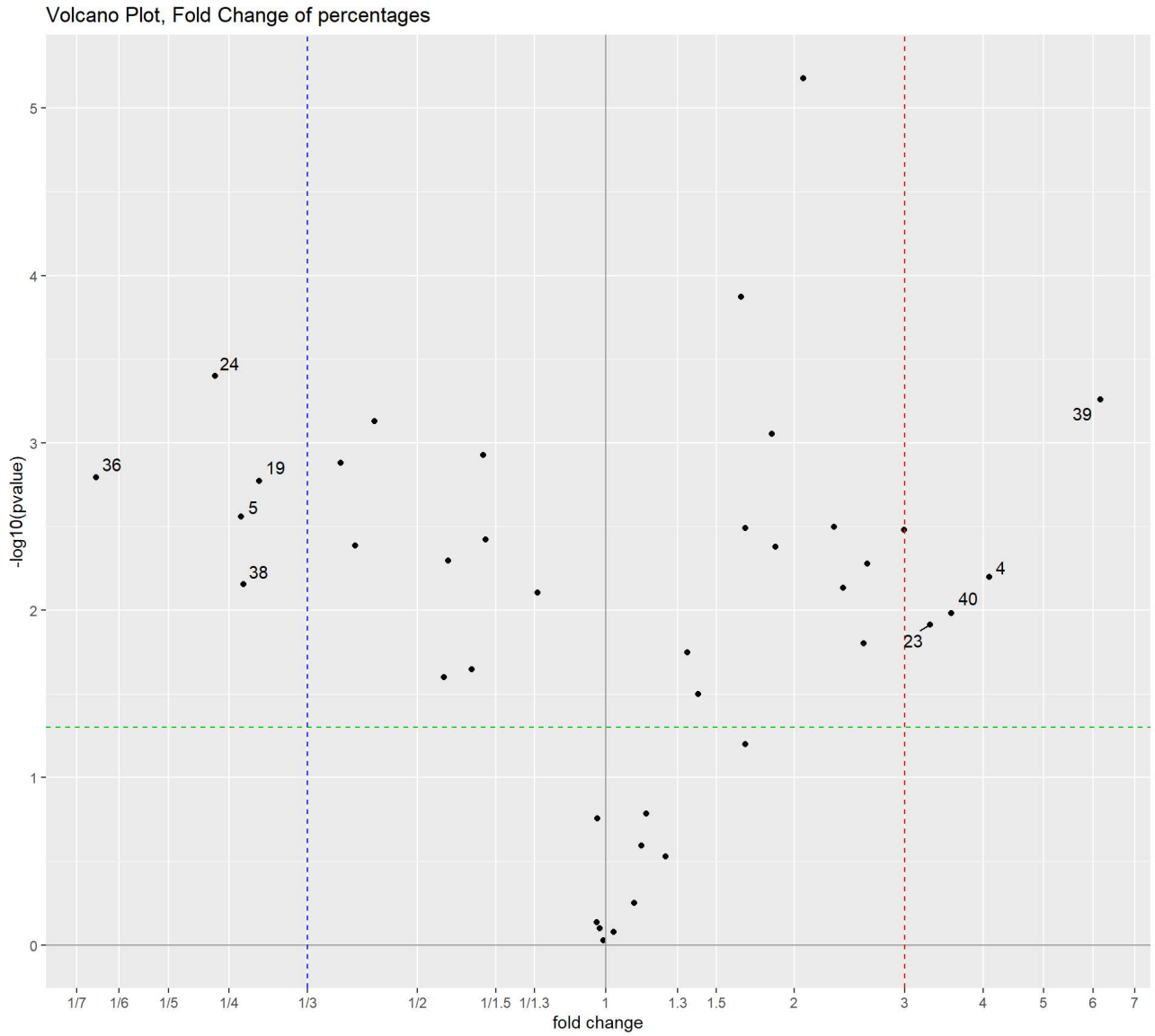
```
## Warning: Removed 31 rows containing missing values (geom_text_repel).
```




## Volcano plot in alternate scale

```
set.seed(10)
fcBreaks <- c(1.3, 1.5, 2:9)
p <- ggplot(cfData@results, aes(2^diff, -log10(pvalue))) + geom_point() +
  geom_hline(yintercept = -log10(c(pvalue.cut, 1)),
             lty = c(2, 1), col = c("green3", "grey60")) +
  geom_vline(xintercept = 2^c(-logFold.cut, 0 , logFold.cut),
             lty = c(2, 1, 2), col = c("blue", "grey60", "red")) +
  ggtitle("Volcano Plot, Fold Change of percentages") + labs(x = "fold change") +
  scale_x_continuous(trans = "log2",
                     breaks = c(1/rev(fcBreaks), 1, fcBreaks),
                     labels = c(sprintf("1/%g",rev(fcBreaks)), "1", sprintf("%g",fcBreak
s)),
                     minor_breaks = NULL)
p + geom_text_repel(aes(label = label))
```

```
## Warning: Removed 31 rows containing missing values (geom_text_repel).
```



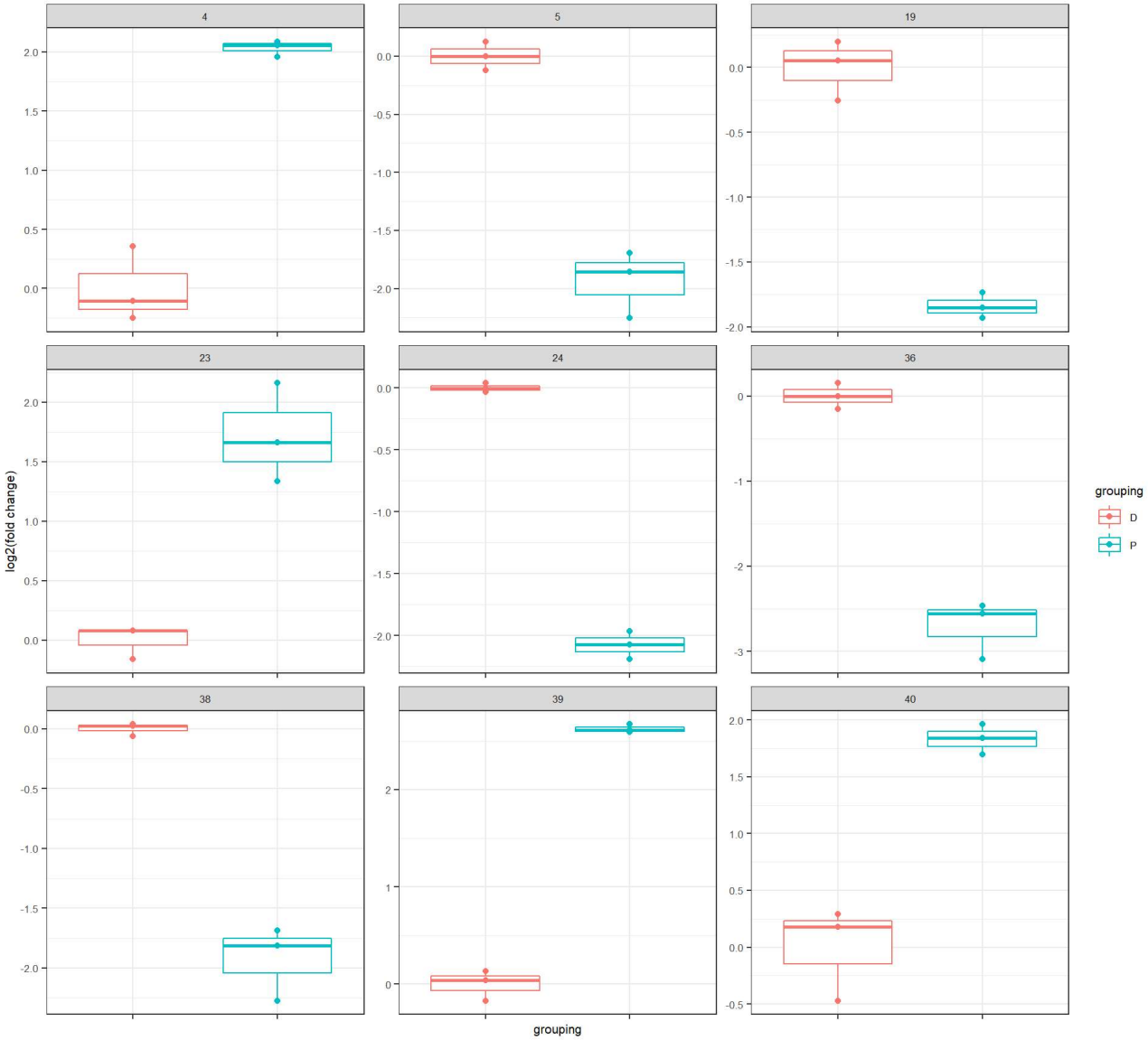

# Report interesting clusters

## Select clusters of interest

```
# remove unselected clusters
clusterSelection <- with(cfData@results, !unselected)
# duplicate and remove unselected clusters
cfDataSelection <- cfData
cfDataSelection@expr <- subset(cfDataSelection@expr,
                               clusterID %in% cfData@results$clusters[clusterSelection])
cfDataSelection@counts <- cfDataSelection@counts[,clusterSelection]
cfDataSelection@results <- cfDataSelection@results[clusterSelection,]
```

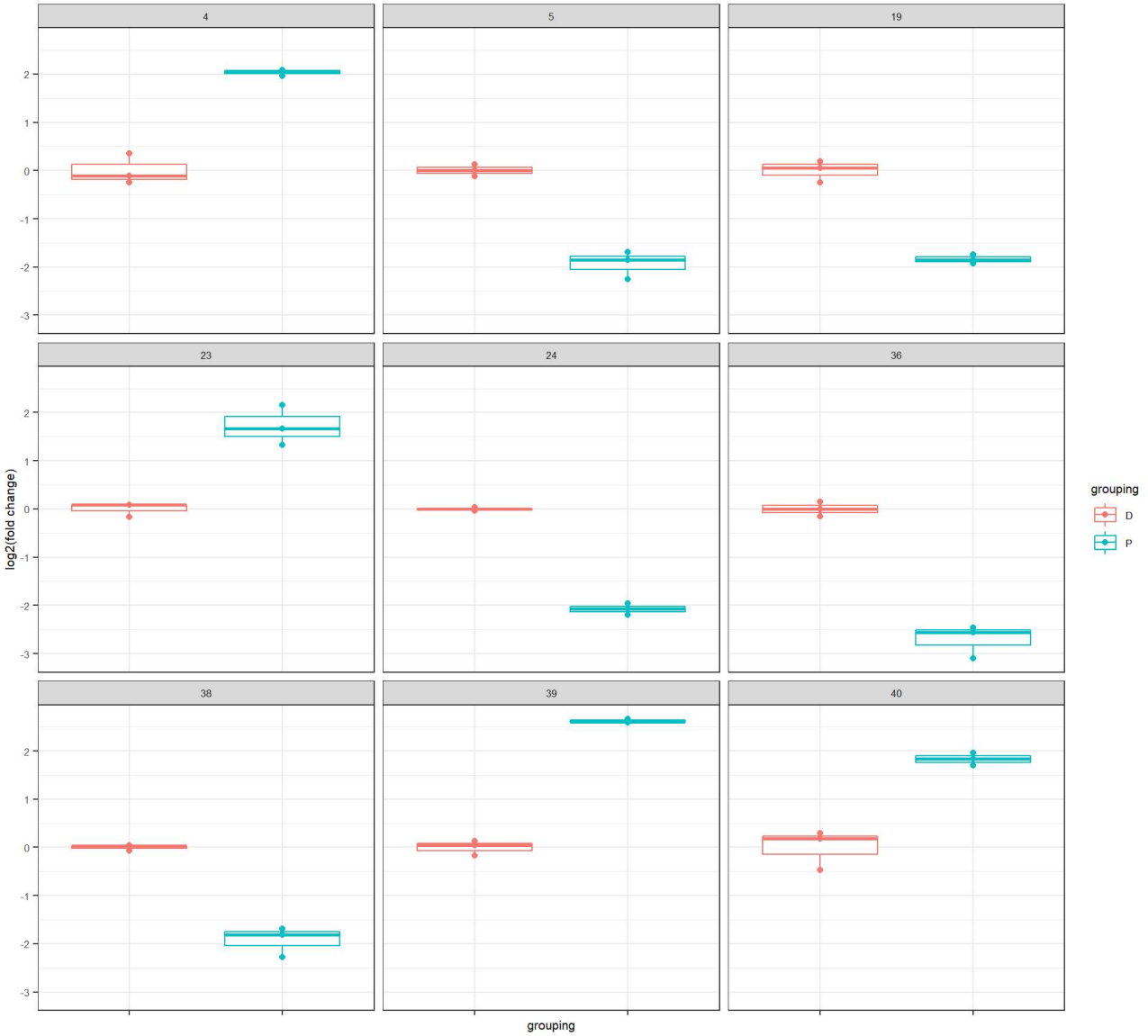
## Box plots

```
# Detailed view of log2 fold changes
cytoBoxplots(cfDataSelection, group = "status") + labs(y = "log2(fold change)")
```




Using the same scale.

```
# with the same scale across all boxplots
cytoBoxplots(cfDataSelection, group = "status") +
  labs(y = "log2(fold change)") +
  facet_wrap(~variable, scales = "fixed")
```




# Table of counts

Here is the report of the clusters that pass the thresholds.

```
library(knitr)
kable(
  cbind(cfData@results[clusterSelection,], t(
    cellCountRaw[,clusterSelection]))[,-c(1:3,6:7)],
  caption = "clusters that pass the thresholds"
)
```

clusters that pass the thresholds

| | pvalue | adjusted | fold | diff | clean_D1 | clean_D2 | clean_D3 | clean_P1 | clean_P2 | clean_P3 |
|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 0.0063136 | 0.0126272 | 4.09 | 2.033586 | 9 | 11 | 19 | 86 | 84 | 78 |
| 5 | 0.0027661 | 0.0094470 | -3.82 | -1.935375 | 200 | 183 | 219 | 48 | 55 | 34 |
| 19 | 0.0016880 | 0.0067520 | -3.58 | -1.840459 | 99 | 139 | 125 | 24 | 26 | 29 |

| | pvalue | adjusted | fold | diff | clean_D1 | clean_D2 | clean_D3 | clean_P1 | clean_P2 | clean_P3 |
|---|---|---|---|---|---|---|---|---|---|---|
| 23 | 0.0121924 | 0.0195078 | 3.29 | 1.720216 | 3 | 1 | 3 | 29 | 45 | 21 |
| 24 | 0.0003999 | 0.0053323 | -4.21 | -2.075314 | 254 | 262 | 249 | 53 | 48 | 58 |
| 36 | 0.0016124 | 0.0067520 | -6.53 | -2.706796 | 180 | 144 | 161 | 21 | 19 | 10 |
| 38 | 0.0070049 | 0.0133114 | -3.80 | -1.924198 | 101 | 109 | 108 | 23 | 14 | 26 |
| 39 | 0.0005512 | 0.0055117 | 6.16 | 2.622331 | 6 | 3 | 5 | 83 | 78 | 79 |
| 40 | 0.0103825 | 0.0173042 | 3.56 | 1.831566 | 26 | 13 | 29 | 104 | 93 | 114 |

# cytofkit(lab) analysis (Via Commands in R Console)

Learning R commands in the console (or writing a small script/notebook) is interesting when you need to tune the parameters of the analysis, or repeat the same analysis to different files. This requires to understand the file system and how to indicate the path to the files (FCS files or results).

# Understanding the file system organization

As you know, the files on your disk are organized in a **hierarchical structure** made of directories (also called folders). This organisation is very simple: a) a directory could contain directories and/or files, b) a file is an atomic piece that contain information but that could not contain any directory. The access to a file consists in specificying the path to it. The path could start at the root of the hierarchical structure (such as “/” for Linux and Mac, “C:/” for Windows). In this case, it’s called an **absolute path**. Of course, you could navigate through the hierarchical structure. Each time you enter a directory, the path is suffixed with a separator (typically “/”) and the name of the directory you entered in. This directory is called the **current directory**. From the current directory, it’s possible to specify paths in a relative way. This means that the relative root is the current directory. This allows specifying shorter paths to access to ressources in the neighborhood of the current directory. Paths become shorter because there is no need to repeat the absolute path from the root. This approach uses a **relative path** to specify directories and files. A directory in the current directory (ie a sub directory) is simply specified by its names “sub_dir”. The unique parent directory (or container) could be specified as “..”. A brother directory (ie that has the same parent as the current directory) is specified by “../brother_dir”. A file in it is specified as “../brother_dir/sample1.fcs”.

When you write a text document and store it, you only specify the absolute path to store it. When you process data files and generate results, you have to specify where the input data files are and where to store the results in the hierachical structure of the disk. Typically, the input files are already in a directory where you deposited them. For the results, you created a dedicated directory. So, you usually have to specify two directories. You could consider that one of those is your **working directory**. That operation transforms it as the relative root for accessing input files and storing results.

Although this might sound simple or complex, there is another layer of complexity. When the amount of data increases, both absolute and relative paths are needed, because both systems have advantages. But problems appear when you move part of the data structure on which the analysis depends. So, try to adopt a structure that will be sustainable in the long term.

Find below some commands to navigate through the file system.

```
# get the working directory.
getwd()
# set
setwd("C:/demo")
```

# cytofkit(lab) analysis (Via Commands in R Console)

## Project definition

The core analysis command of cytofkit requires the paths to the input files and the result directories. We organize the data and the results using those variable names. This is the sole configuration needed.

```
fcs_dir = "c:/demo/200205-atelier/CLEAN_DATA"
res_dir = "c:/demo/200205-atelier/CLEAN_DATA_results"
marker_file = "ck_markers_main.txt"  # this file is searched in the input directory
if (!dir.exists(res_dir)) dir.create(res_dir, recursive = TRUE)
```

It's usually difficult to navigate through the file system using R. An easier way is to use a graphical user interface for that task.

```
# the following tcltk command will help you to locate your files in the file system
# and select the markers; it writes the selected markers in a cytofkit_full.txt file
storeMarkers_GUI()
```

## Script version for tuning parameters

Here the main function of cytofkit. It exposes most of the parameters, but not all.

```
# the interactive exploration of results
analysis_file = file.path(res_dir, "run_5k", "run_5k.RData")
if (file.exists(analysis_file))
  cytofkitShinyAPP(analysis_file)
```

## Useful FCS files commands

If you want to get an insight of the FCS files, here are some commands to read FCS files and retrieve information. This is useful to get the count of events per file and their channel names.

```
library(flowCore)

# read the FCS as a flowset, ie a group of compatible FCS
fs = read.flowSet(path = fcs_dir, pattern = "*.fcs", transformation = FALSE, truncate_max_
range = FALSE)

# view the flowset
fs
```

```
## A flowSet with 6 experiments.
##
##   column names:
##   129 131 132 133 134 136 137 138 139 140 146 151 155 157 161 169 172 173 CCR4-149 CCR5
-144 CCR7-159 CD16-165 CD19-142 CD20-147 CD27-167 CD28-160 CD3-170 CD33-158 CD38-148 CD4-1
45 CD43-150 CD44-166 CD45-154 CD45RA-153 CD45RO-164 CD56-176 CD69-162 CD8-168 CXCR3-156 CX
CR5-171 Cell_length Cisplatin-195 HLA-DR-163 ICOS-141 NA-191 NA-193 PD-1-174 PD-L1-175 TCR
gd-152 TIM3-143 Time
```

```
# view annotations
pData(fs)
```

```
##                      name
## clean_D1.fcs clean_D1.fcs
## clean_D2.fcs clean_D2.fcs
## clean_D3.fcs clean_D3.fcs
## clean_P1.fcs clean_P1.fcs
## clean_P2.fcs clean_P2.fcs
## clean_P3.fcs clean_P3.fcs
```

```
# here, none

# view the cell counts
fsApply(fs, nrow)
```

```
##               [,1]
## clean_D1.fcs 15608
## clean_D2.fcs 16075
## clean_D3.fcs 15340
## clean_P1.fcs 15105
## clean_P2.fcs 14724
## clean_P3.fcs 15469
```

```
# view a specific FCS
ff = fs[[1]]  # extract 1st FCS as flowframe
# view some keywords
keyword(ff, "$CYT")  # cytometer
```

```
## $`$CYT`
## NULL
```

```
keyword(ff, "$GUID")
```

```
## $`$GUID`
## NULL
```

```
keyword(ff, "$FIL")  # file name
```

```
## $`$FIL`
## [1] "clean_D1.fcs"
```

```
keyword(ff, "$TOT")  # cell count
```

```
## $`$TOT`
## [1] "15608"
```

```
keyword(ff, "$COM")  # comment
```

```
## $`$COM`
## [1] "FCS file exported using Cytobank"
```

```
# view all keywords
kwd = keyword(ff)
length(kwd)  # there are too many
```

```
## [1] 378
```

```
# filter out some keywords
kwd_1 = kwd[!grepl("flowCore", names(kwd))]  # not flowcore
kwd_2 = kwd_1[-grep("^\\$P", names(kwd_1))]  # not standard parameters
#kwd_2  # display remaining keywords
unlist(kwd_2)
```

```
##                                     FCSversion
##                                            "3"
##                                          $FIL
##                                 "clean_D1.fcs"
##                                          $TOT
##                                        "15608"
##                                      $BYTEORD
##                                      "4,3,2,1"
##                                     $DATATYPE
##                                            "F"
##                                 FJ_FCS_VERSION
##                                            "3"
##                                $BEGINANALYSIS
##                                            "0"
##                                  $ENDANALYSIS
##                                            "0"
##                                   $BEGINSTEXT
##                                            "0"
##                                     $ENDSTEXT
##                                            "0"
##                                     $NEXTDATA
##                                            "0"
##                                         $MODE
##                                           "L"
##                                          $COM
##              "FCS file exported using Cytobank"
##                                            ID
##                                          "D1"
##                                    $BEGINDATA
##                                        "5014"
##                                     $ENDDATA
##                                    "3189045"
##                                    FILENAME
## "c:/demo/200205-atelier/CLEAN_DATA/clean_D1.fcs"
##                                          GUID
##                                 "clean_D1.fcs"
##                                  ORIGINALGUID
##                                 "clean_D1.fcs"
##                                   GUID.original
##                                 "clean_D1.fcs"
```