Check for updates

METHOD ARTICLE

REVISED CyTOF workflow: differential discovery in high-throughput high-dimensional cytometry datasets [version 2; referees: 2 approved]

Malgorzata Nowicka[1,2], Carsten Krieg[3], Lukas M. Weber[1,2], Felix J. Hartmann[3], Silvia Guglietta[4], Burkhard Becher[3], Mitchell P. Levesque[5], Mark D. Robinson[1,2]

[1]SIB Swiss Institute of Bioinformatics, University of Zurich, Zurich, 8057, Switzerland
[2]Institute for Molecular Life Sciences, University of Zurich, Zurich, 8057, Switzerland
[3]Institute of Experimental Immunology, University of Zurich, Zurich, 8057, Switzerland
[4]Department of Experimental Oncology, European Institute of Oncology, Via Adamello 16, Milan, I-20139, Italy
[5]Department of Dermatology, University Hospital Zurich, Zurich, CH-8091, Switzerland

v2 **First published:** 26 May 2017, **6**:748 (doi: 10.12688/f1000research.11622.1)
Latest published: 14 Nov 2017, **6**:748 (doi: 10.12688/f1000research.11622.2)

Abstract
High dimensional mass and flow cytometry (HDCyto) experiments have become a method of choice for high throughput interrogation and characterization of cell populations.Here, we present an R-based pipeline for differential analyses of HDCyto data, largely based on Bioconductor packages. We computationally define cell populations using FlowSOM clustering, and facilitate an optional but reproducible strategy for manual merging of algorithm-generated clusters. Our workflow offers different analysis paths, including association of cell type abundance with a phenotype or changes in signaling markers within specific subpopulations, or differential analyses of aggregated signals. Importantly, the differential analyses we show are based on regression frameworks where the HDCyto data is the response; thus, we are able to model arbitrary experimental designs, such as those with batch effects, paired designs and so on. In particular, we apply generalized linear mixed models to analyses of cell population abundance or cell-population-specific analyses of signaling markers, allowing overdispersion in cell count or aggregated signals across samples to be appropriately modeled. To support the formal statistical analyses, we encourage exploratory data analysis at every step, including quality control (e.g. multi-dimensional scaling plots), reporting of clustering results (dimensionality reduction, heatmaps with dendrograms) and differential analyses (e.g. plots of aggregated signals).

This article is included in the Bioconductor gateway.

Open Peer Review

Referee Status: ✓ ✓

	Invited Referees 1	2
REVISED **version 2** published 14 Nov 2017		
version 1 published 26 May 2017	✓ report	✓ report

1 **Aaron T. L. Lun**, University of Cambridge, UK
John C. Marioni, University of Cambridge, UK
Wellcome Genome Campus, UK
Wellcome Genome Campus, UK

2 **Greg Finak**, Fred Hutchinson Cancer Research Center, USA
Raphael Gottardo, Fred Hutchinson Cancer Research Center, USA

Discuss this article

Comments (0)

Corresponding author: Mark D. Robinson (mark.robinson@imls.uzh.ch)

Author roles: **Nowicka M**: Conceptualization, Formal Analysis, Investigation, Methodology, Software, Visualization, Writing – Original Draft Preparation, Writing – Review & Editing; **Krieg C**: Conceptualization, Investigation, Validation, Writing – Review & Editing; **Weber LM**: Conceptualization, Formal Analysis, Methodology, Software, Visualization, Writing – Review & Editing; **Hartmann FJ**: Conceptualization, Methodology, Writing – Review & Editing; **Guglietta S**: Conceptualization, Validation; **Becher B**: Investigation, Supervision; **Levesque MP**: Investigation, Resources, Supervision, Validation, Writing – Review & Editing; **Robinson MD**: Conceptualization, Formal Analysis, Investigation, Methodology, Project Administration, Software, Supervision, Visualization, Writing – Original Draft Preparation, Writing – Review & Editing

Competing interests: No competing interests were disclosed.

How to cite this article: Nowicka M, Krieg C, Weber LM *et al.* **CyTOF workflow: differential discovery in high-throughput high-dimensional cytometry datasets [version 2; referees: 2 approved]** *F1000Research* 2017, **6**:748 (doi: 10.12688/f1000research.11622.2)

Grant information: MN acknowledges the funding from a Swiss Institute of Bioinformatics (SIB) Fellowship.
The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

First published: 26 May 2017, **6**:748 (doi: 10.12688/f1000research.11622.1)

REVISED **Amendments from Version 1**

In version 2 of the workflow, we have added a section called "Obtaining higher resolution" where we describe strategies for deeper clustering analysis. We have fixed and added missing references, extended the discussion section. We have proposed additional visualizations of clusters and extended some of the figure captions.

To improve speed, the density of marker signals across clusters was changed to ridgeline plots (geom_density_ridges) instead of densities (geom_density). Furthermore, heatmaps of the identity marker clustering are now supplemented with a barplot highlighting the frequency of the cluster.

In total, two new figures were added: one to highlight the usage of the 'ComplexHeatmap' package (in the context of subpopulation-specific identity and functional marker signal) and another that corresponds with the new "Obtaining higher resolution" Section.

See referee reports

Introduction

Flow cytometry and the more recently introduced CyTOF (cytometry by time-of-flight mass spectrometry or mass cytometry) are high-throughput technologies that measure protein abundance on the surface or within cells. In flow cytometry, antibodies are labeled with fluorescent dyes and fluorescence intensity is measured using lasers and photodetectors. CyTOF utilizes antibodies tagged with metal isotopes from the lanthanide series, which have favorable chemistry and do not occur in biological systems; abundances per cell are recorded with a time-of-flight mass spectrometer. In either case, fluorescence intensities (flow cytometry) or ion counts (mass cytometry) are assumed to be proportional to the expression level of the antibody-targeted antigens of interest.

Due to the differences in acquisition, further distinct characteristics should be noted. Conventional fluorophore-based flow cytometry is non-destructive and can be used to sort cells for further analysis. However, because of the spectral overlap between fluorophores, *compensation* of the data needs to be performed (Roederer, 2001), which also limits the number of parameters that can be measured simultaneously. Thus, standard flow cytometry experiments measure 6–12 parameters with modern systems measuring up to 20 channels (Mahnke & Roederer, 2007), while new developments (e.g. BD FACSymphony) promise to increase this capacity towards 50. Moreover, flow cytometry offers the highest throughput with tens of thousands of cells measured per second at relatively low operating costs per sample.

By using rare metal isotopes in CyTOF, cell autofluorescence can be avoided and spectral overlap is drastically reduced. However, the sensitivity of mass spectrometry results in the measurement of metal impurities and oxide formations, which need to be carefully considered in antibody panel design (e.g. through antibody concentrations and coupling of antibodies to neighboring metals). Leipold *et al.* recently commented that *minimal spillover does not equal no spillover* (Leipold, 2015). Nonetheless, CyTOF offers a high dimension of parameters measured per cell, with current panels using ~40 parameters and the promise of up to 100. Throughput of CyTOF is lower, at the rate of hundreds of cells per second, and cells are destroyed during ionization.

The ability of flow cytometry and mass cytometry to analyze individual cells at high-throughput scales has resulted in a wide range of biological and medical applications. For example, immunophenotyping assays are used to detect and quantify cell populations of interest, to uncover new cell populations and compare abundance of cell populations between different conditions, for example between patient groups (van Unen *et al.*, 2016). Thus, it can be used as a biomarker discovery tool.

Various methodological approaches aim for biomarker discovery (Saeys *et al.*, 2016). A common strategy, which we will refer to through this workflow as the "classic" approach, is to first identify cell populations of interest by manual gating or automated clustering (Hartmann *et al.*, 2016; Pejoski *et al.*, 2016). Second, using statistical tests, one can determine which of the cell subpopulations or protein markers are associated with a phenotype (e.g. clinical outcome) of interest. Typically, cell subpopulation abundance expressed as cluster cell counts or median marker expression would be used in the statistical model to relate to the sample-level phenotype.

Importantly, there are many alternatives to what we propose below, and several new methods are emerging. For instance, *citrus* (Bruggner *et al.*, 2014) tackles the differential discovery problem by strong over-clustering of the cells, and by building a hierarchy of clusters from very specific to general ones. Using model selection and regularization techniques, clusters and markers that associate with the outcome are identified. A new machine learning approach, *CellCnn* (Arvaniti & Claassen, 2017), learns the representation of clusters that are associated with the considered phenotype by means of convolutional neural networks, which makes it particularly applicable to detecting discriminating rare cell populations. However, there are tradeoffs to consider. *citrus* performs feature selection but does not provide significance levels, such as p-values, for the strength of associations. Due to its computational requirements, *citrus* can not be run on entire mass cytometry datasets and one typically must analyze a subset of the data. The “filters” from *CellCnn* may identify one or more cell subsets that distinguish experimental groups, while these groups may not necessarily correspond to any of the canonical cell types, since they are learned with a data-driven approach.

A noticeable distinction between the machine-learning approaches and our classical regression approach is how the model is designed. *citrus* and *CellCnn* model the patient response as a function of the measured HDCyto values, whereas the classical approach models the HDCyto data itself as the response, thus putting the distributional assumptions on the experimental HDCyto data. This carries the distinct advantage that covariates (e.g. age, gender, batch) can be included, together with finding associations of the phenotype to the predictors of interest (e.g. cell type abundance). Specifically, neither *citrus* nor *CellCnn* are able to directly account for complex designs, such as paired experiments or presence of batches.

Within the classical approach, hybrid methods are certainly possible, where discovery of interesting cell populations is done with one algorithm, and quantifications or signal aggregations are modeled in standard regression frameworks. For instance, *CellCnn* provides p-values from a t-test or Mann-Whitney U-test conducted on the frequencies of previously detected cell populations. The models we propose below are flexible extensions of this strategy.

Step by step, this workflow presents differential discovery analyses assembled from a suite of tools and methods that, in our view, lead to a higher level of flexibility and robust, statistically-supported and interpretable results. Cell population identification is conducted by means of unsupervised clustering using the *FlowSOM* and *ConsensusClusterPlus* packages, which together were among the best performing clustering approaches for high-dimensional cytometry data (Weber & Robinson, 2016). Notably, *FlowSOM* scales easily to millions of cells and thus no subsetting of the data is required.

To be able to analyze arbitrary experimental designs (e.g. batch effects, paired experiments, etc.), we show how to conduct the differential analysis of cell population abundances using the generalized linear mixed models (GLMM) and of marker intensities using linear models (LM) and linear mixed models (LMM). Model fitting is performed with *lme4* and *stats* packages, and hypothesis testing with the *multcomp* package.

We use the *ggplot2* package as our graphical engine. Notably, we propose a suite of useful visual representations of HDCyto data characteristics, such as an MDS (multidimensional scaling) plot of aggregated signal for exploring sample similarities. The obtained cell populations are visualized using dimension reduction techniques (e.g. t-SNE via the *Rtsne* package) and heatmaps (via the *pheatmap* and *ComplexHeatmap* packages) to represent characteristics of the annotated cell populations and identified biomarkers.

The workflow is intentionally not fully automatic. First, we strongly advocate for exploratory data analysis to get an understanding of data characteristics before formal statistical modeling. Second, the workflow involves an optional step where the user can manually merge and annotate clusters (see Cluster merging and annotation section) but in a way that is easily reproducible. The CyTOF data used here (see Data description section) is already preprocessed; i.e. the normalization and de-barcoding, as well as removal of doublets, debris and dead cells, were already performed. To see how such an analysis could be performed, please see the Data preprocessing section.

Notably, this workflow is equally applicable to flow or mass cytometry datasets, for which the preprocessing steps have already been performed. In addition, the workflow is modular and can be adapted as new algorithms or new knowledge about how to best use existing tools comes to light. Alternative clustering algorithms such as the popular PhenoGraph algorithm (Levine *et al.*, 2015) (e.g. via the *Rphenograph* package), dimensionality reduction techniques, such as diffusion maps (Haghverdi *et al.*, 2015) via the *destiny* package (Angerer *et al.*, 2016), and SIMLR (Wang *et al.*, 2017) via the *SIMLR* package could be inserted to the workflow.

Data description

We use a subset of CyTOF data originating from Bodenmiller *et al.* (Bodenmiller *et al.*, 2012) that was also used in the *citrus* paper (Bruggner *et al.*, 2014). In the original study, peripheral blood mononuclear cells (PBMCs) in unstimulated and after 11 different stimulation conditions were measured for 8 healthy donors. For each sample, expression of 10 cell surface markers and 14 signaling markers was recorded. We perform our analysis on samples from the reference and one stimulated condition where cell were crosslinked for 30 minutes with B cell receptor/Fc receptor known as BCR/FcR-XL, resulting in 16 samples in total (8 patients, unstimulated and stimulated for each).

The original data is available from the Cytobank report. The subset used here can be downloaded from the Citrus Cytobank repository (files with `_BCR-XL.fcs` or `_Reference.fcs` endings) or from our web server (see Data import section).

In both the Bodenmiller *et al.* and *citrus* manuscripts, the 10 lineage markers were used to identify cell subpopulations. These were then investigated for differences between reference and stimulated cell subpopulations separately for each of the 14 functional markers. The same strategy is used in this workflow; 10 lineage markers are used for cell clustering and 14 functional markers are tested for differential expression between the reference and BCR/FcR-XL stimulation. Even though differential analysis of cell abundance was not in the scope of the Bodenmiller *et al.* experiment, we present it here to highlight the generality of the discovery.

Data preprocessing

Conventional flow cytometers and mass cytometers produce .fcs files that can be manually analyzed using programs such as FlowJo [TriStar] or Cytobank (Kotecha *et al.*, 2001), or using the R/Bioconductor packages, such as *flowWorkspace* (Finak & Jiang, 2011) and *openCyto* (Finak *et al.*, 2014). During this initial analysis step, dead cells are removed, compensation is checked and with simple two dimensional scatter plots (e.g. marker intensity versus time), marker expression patterns are checked. Often, modern experiments are barcoded in order to remove analytical biases due to individual sample variation or acquisition time. Preprocessing steps including normalization using bead standards (Finck *et al.*, 2013), de-barcoding (Zunder *et al.*, 2015) and compensation can be completed with the *CATALYST* package (Chevrier *et al.*, 2017), which also provides a Shiny app for the interactive analysis. Of course, preprocessing steps can occur using custom scripts within R or outside of R (e.g. Normalizer (Finck *et al.*, 2013)).

Data import

We recommend as standard practice to keep an independent record of all samples collected, with additional information about the experimental condition, including sample or patient identifiers, processing batch and so on. That is, we recommend having a trail of metadata for each experiment. In our example, the metadata file, PBMC8_metadata.xlsx, can be downloaded from the Robinson Lab server with the `download.file` function. For the workflow, the user should place it in the current working directory (`getwd()`). Here, we load it into R with the `read_excel` function from the *readxl* package and save it into a variable called `md`, but other file types and interfaces to read them in are also possible.

The data frame `md` contains the following columns:

- `file_name` with names of the .fcs files corresponding to the reference (suffix "Reference") and BCR/FcR-XL stimulation (suffix "BCR-XL") samples,
- `sample_id` with shorter unique names for each sample containing information about conditions and patient IDs,
- `condition` describes whether samples originate from the reference (`Ref`) or stimulated (`BCRXL`) condition,
- `patient_id` defines the IDs of patients.

The `sample_id` variable is used as row names in metadata and will be used all over the workflow to label the samples. It is important to carefully check whether variables are of the desired type (factor, numeric, character), since input methods may convert columns into different data types. For the statistical modeling, we want to make the condition variable a factor with the reference (`Ref`) samples being the reference level, where the order of factor levels can be defined with the `levels` parameter of the `factor` function. We also specify colors for the different conditions in a variable `color_conditions`.

```r
library(readxl)
url <- "http://imlspenticton.uzh.ch/robinson_lab/cytofWorkflow"
metadata_filename <- "PBMC8_metadata.xlsx"
download.file(paste0(url, "/", metadata_filename), destfile = metadata_filename,
  mode = "wb")
md <- read_excel(metadata_filename)

## Make sure condition variables are factors with the right levels
md$condition <- factor(md$condition, levels = c("Ref", "BCRXL"))
head(data.frame(md))
```

```
##                              file_name sample_id condition patient_id
## 1    PBMC8_30min_patient1_BCR-XL.fcs    BCRXL1     BCRXL   Patient1
## 2 PBMC8_30min_patient1_Reference.fcs      Ref1       Ref   Patient1
## 3    PBMC8_30min_patient2_BCR-XL.fcs    BCRXL2     BCRXL   Patient2
## 4 PBMC8_30min_patient2_Reference.fcs      Ref2       Ref   Patient2
## 5    PBMC8_30min_patient3_BCR-XL.fcs    BCRXL3     BCRXL   Patient3
## 6 PBMC8_30min_patient3_Reference.fcs      Ref3       Ref   Patient3
```

```
## Define colors for conditions
color_conditions <- c("#6A3D9A", "#FF7F00")
names(color_conditions) <- levels(md$condition)
```

The .fcs files listed in the metadata can be downloaded manually from the Citrus Cytobank repository or automatically from the Robinson Lab server where they are saved in a compressed archive file, PBMC8_fcs_files.zip.

```
fcs_filename <- "PBMC8_fcs_files.zip"
download.file(paste0(url, "/", fcs_filename), destfile = fcs_filename, mode = "wb")
unzip(fcs_filename)
```

To load the content of the .fcs files into R, we use the *flowCore* package (Hahne *et al.*, 2009). Using `read.flowSet`, we read in all files into a flowSet object, which is a general container for HDCyto data. Importantly, `read.flowSet` and the underlying `read.FCS` functions, by default, may transform the marker intensities and remove cells with extreme positive values. We keep these options off to be sure that we control the exact preprocessing steps.

```
library(flowCore)
fcs_raw <- read.flowSet(md$file_name, transformation = FALSE,
  truncate_max_range = FALSE)
fcs_raw
```

In our example, information about the panel is also available in a file called PBMC8_panel.xlsx, and can be downloaded from the Robinson Lab server and loaded into a variable called `panel`. It contains columns for `Isotope` and `Metal` that define the atomic mass number and the symbol of the chemical element conjugated to the antibody, respectively, and `Antigen`, which specifies the protein marker that was targeted; two additional columns specify whether a channel belongs to the lineage or surface type of marker.

The isotope, metal and antigen information that the instrument receives is also stored in the `flowFrame` (container for one sample) or `flowSet` (container for multiple samples) objects. You can type `fcs_raw[[1]]` to see the first `flowFrame`, which contains a table with columns `name` and `desc`. Their content can be accessed with functions `pData(parameters())`, which is identical for all the `flowFrame` objects in the `flowSet`. The variable `name` corresponds to the column names in the `flowSet` object, you can type in R `colnames(fcs_raw)`.

It should be checked that elements from `panel` can be matched to their corresponding entries in the `flowSet` object to make the analysis less prone to subsetting mistakes. Here, for example, the entries in `panel$Antigen` have their exact equivalents in the `desc` columns of the `flowFrame` objects. In the following analysis, we will often use marker IDs as column names in the tables containing expression values. As a cautionary note, during object conversion from one type to another (e.g. in the creation of data.frame from a matrix), some characters (e.g. dashes) in the dimension names are replaced with dots, which may cause problems in matching. To avoid this problem, we replace all the dashes with underscores. Also, we define two variables that indicate the lineage and functional markers.

```
panel_filename <- "PBMC8_panel.xlsx"
download.file(paste0(url, "/", panel_filename), destfile =  panel_filename, mode = "wb")
panel <- read_excel(panel_filename)
head(data.frame(panel))
```

```
##   Metal Isotope Antigen Lineage Functional
## 1    Cd 110:114     CD3       1          0
## 2    In     115    CD45       1          0
## 3    La     139     BC1       0          0
## 4    Pr     141     BC2       0          0
## 5    Nd     142   pNFkB       0          1
## 6    Nd     144    pp38       0          1

# Replace problematic characters
panel$Antigen <- gsub("-", "_", panel$Antigen)

panel_fcs <- pData(parameters(fcs_raw[[1]]))
head(panel_fcs)

##                 name        desc   range  minRange maxRange
## $P1             Time        Time 2377271   0.00000  2377270
## $P2      Cell_length Cell_length      66   0.00000       65
## $P3 CD3(110:114)Dd         CD3    1212 -13.66756     1211
## $P4   CD45(In115)Dd        CD45    2654   0.00000     2653
## $P5    BC1(La139)Dd         BC1   13357   0.00000    13356
## $P6    BC2(Pr141)Dd         BC2      39 -66.97583       38

# Replace problematic characters
panel_fcs$desc <- gsub("-", "_", panel_fcs$desc)

# Lineage markers
(lineage_markers <- panel$Antigen[panel$Lineage == 1])

##  [1] "CD3"    "CD45"   "CD4"    "CD20"    "CD33"    "CD123"    "CD14"
##  [8] "IgM"    "HLA_DR"  "CD7"

# Functional markers
(functional_markers <- panel$Antigen[panel$Functional == 1])

##  [1] "pNFkB"  "pp38"   "pStat5" "pAkt"   "pStat1" "pSHP2"  "pZap70"
##  [8] "pStat3" "pSlp76" "pBtk"   "pPlcg2" "pErk"   "pLat"   "pS6"

# Spot checks
all(lineage_markers %in% panel_fcs$desc)

## [1] TRUE

all(functional_markers %in% panel_fcs$desc)

## [1] TRUE
```

Data transformation

Usually, the raw marker intensities read by a cytometer have strongly skewed distributions with varying ranges of expression, thus making it difficult to distinguish between the negative and positive cell populations. It is common practice to transform CyTOF marker intensities using, for example, arcsinh (hyperbolic inverse sine) with cofactor 5 (Bendall *et al.*, 2011 Figure S2; Bruggner *et al.*, 2014) to make the distributions more symmetric and to map them to a comparable range of expression, which is important for clustering. A cofactor of 150 has been promoted for flow cytometry, but users are free to implement alternative transformations, some of which are available from the `transform` function of the *flowCore* package. In the following step, we include only those channels that correspond to the lineage and functional markers. We also rename the columns in the `flowSet` to the antigen names from `panel$desc`.

```
## arcsinh transformation and column subsetting
fcs <- fsApply(fcs_raw, function(x, cofactor=5){
  colnames(x) <- panel_fcs$desc
  expr <- exprs(x)
  expr <- asinh(expr[, c(lineage_markers, functional_markers)] / cofactor)
  exprs(x) <- expr
  x
})
fcs

## A flowSet with 16 experiments.
##
##   column names:
##   CD3 CD45 CD4 CD20 CD33 CD123 CD14 IgM HLA_DR CD7 pNFkB pp38 pStat5 pAkt
pStat1 pSHP2 pZap70 pStat3 pSlp76 pBtk pPlcg2 pErk pLat pS6
```

For some of the further analysis, it is more convenient for us to work using a matrix (called `expr`) that contains marker expression for cells from all samples. We create such a matrix with the `fsApply` function that extracts the expression matrices (function `exprs`) from each element of the `flowSet` object, and by default, concatenates them into one matrix.

```
## Extract expression
expr <- fsApply(fcs, exprs)
dim(expr)

## [1] 172791       24
```

As the ranges of marker intensities can vary substantially, we apply another transformation that scales expression of all markers to values between 0 and 1 using low (e.g. 1%) and high (e.g. 99%) percentiles as the boundary. This additional transformation of the arcsinh-transformed data can sometimes give better representation of relative differences in marker expression between annotated cell populations, however, it is only used here for visualization.

```
library(matrixStats)
rng <- colQuantiles(expr, probs = c(0.01, 0.99))
expr01 <- t((t(expr) - rng[, 1]) / (rng[, 2] - rng[, 1]))
expr01[expr01 < 0] <- 0
expr01[expr01 > 1] <- 1
```

Diagnostic plots

We propose some quick checks to verify whether the data we analyze globally represents what we expect; for example, whether samples that are replicates of one condition are more similar and are distinct from samples from another condition. Another important check is to verify that marker expression distributions do not have any abnormalities such as having different ranges or distinct distributions for a subset of the samples. This could highlight problems with the sample collection or HDCyto acquisition, or batch effects that were unexpected. Depending on the situation, one can then consider removing problematic markers or samples from further analysis; in the case of batch effects, a covariate column could be added to the metadata table and used below in the statistical analyses.

The step below generates a plot with per-sample marker expression distributions, colored by condition (see Figure 1). Here, we can already see distinguishing markers, such as pNFkB and CD20, between stimulated and unstimulated conditions.

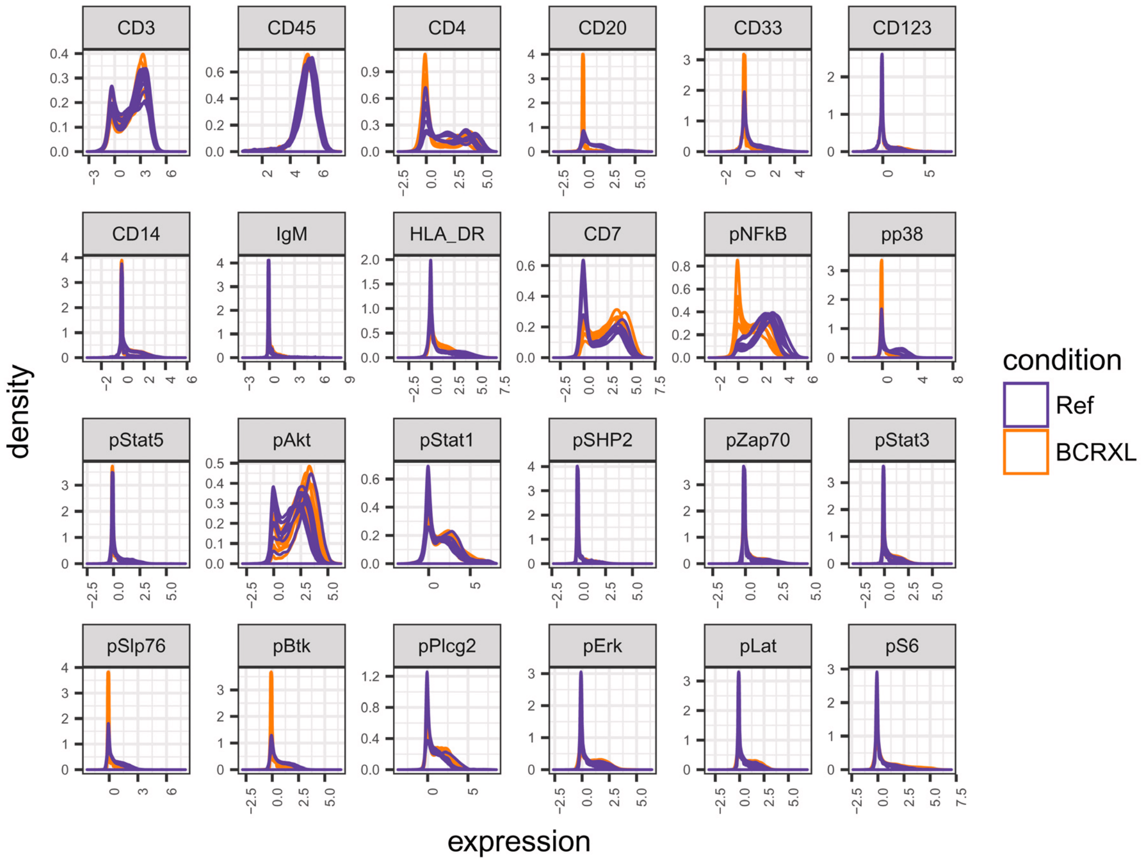


Figure 1. Per-sample smoothed densities of marker expression (arcsinh-transformed) of 10 lineage markers and 14 functional markers in the PBMC dataset. Two conditions: unstimulated (Ref) and stimulated with BCR/FcR-XL (BCRXL) for each of the 8 healthy donors are presented and colored by experimental condition.

```
## Generate sample IDs corresponding to each cell in the 'expr' matrix
sample_ids <- rep(md$sample_id, fsApply(fcs_raw, nrow))

library(ggplot2)
library(reshape2)

ggdf <- data.frame(sample_id = sample_ids, expr)
ggdf <- melt(ggdf, id.var = "sample_id",
  value.name = "expression", variable.name = "antigen")
mm <- match(ggdf$sample_id, md$sample_id)
ggdf$condition <- md$condition[mm]

ggplot(ggdf, aes(x = expression, color = condition,
  group = sample_id)) +
  geom_density() +
  facet_wrap(~ antigen, nrow = 4, scales = "free") +
  theme_bw() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1),
    strip.text = element_text(size = 7), axis.text = element_text(size = 5)) +
  scale_color_manual(values = color_conditions)
```

Another spot check is the number of cells per sample (see Figure 2). This plot can be used as a guide together with other readouts to identify samples where not enough cells were assayed.

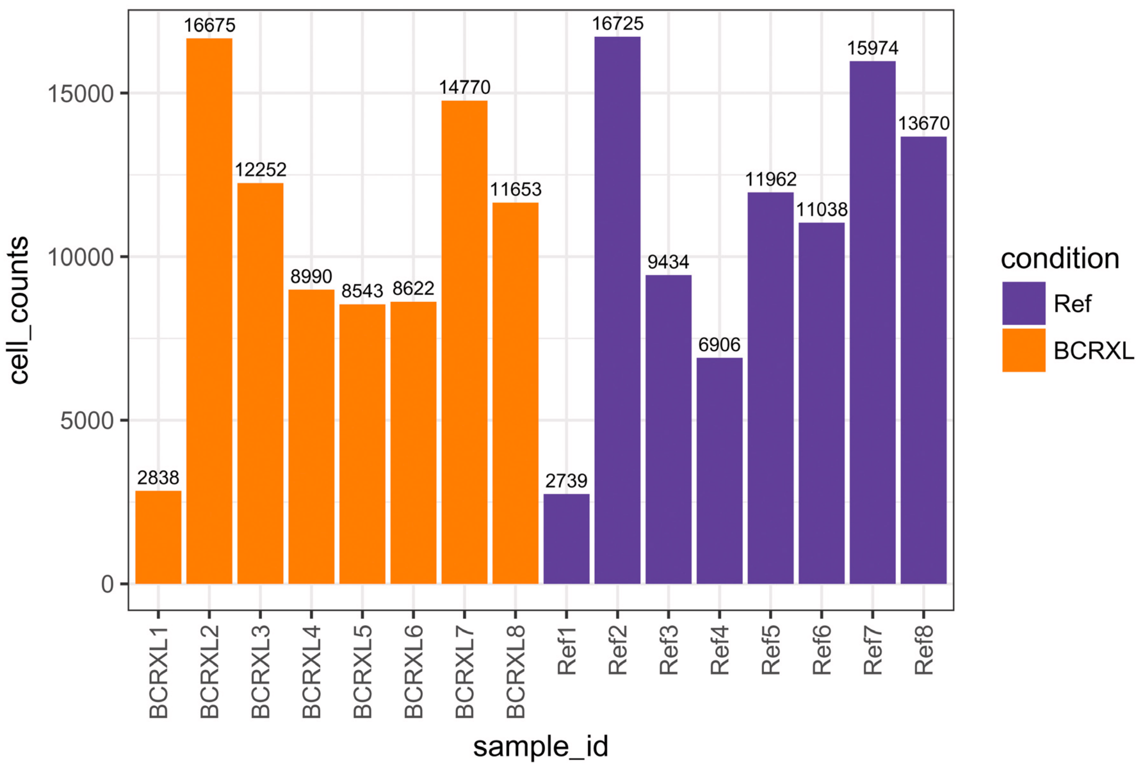


Figure 2. Barplot showing the number of cells measured for each sample in the PBMC dataset. Bars are colored by experimental condition: unstimulated (Ref) and stimulated with BCR/FcR-XL (BCRXL). Numbers in the names on the x-axis indicate patient IDs. Numbers on top of the bars indicate the cell counts.

```
cell_table <- table(sample_ids)

ggdf <- data.frame(sample_id = names(cell_table),
  cell_counts = as.numeric(cell_table))
mm <- match(ggdf$sample_id, md$sample_id)
ggdf$condition <- md$condition[mm]

ggplot(ggdf, aes(x = sample_id, y = cell_counts, fill = condition)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = cell_counts), hjust=0.5, vjust=-0.5, size = 2.5) +
  theme_bw() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1)) +
  scale_fill_manual(values = color_conditions, drop = FALSE) +
  scale_x_discrete(drop = FALSE)
```

MDS plot

In transcriptomics applications, one of the most utilized exploratory plots is the multi-dimensional scaling (MDS) plot or a principal component analysis (PCA) plot. Such plots show similarities between samples measured in an unsupervised way and give a sense of how much differential expression can be detected before conducting any formal tests. An MDS plot can be generated with the `plotMDS` function from the *limma* package. In transcriptomics, distances between samples are calculated based on the expression of the top varying genes. We propose a similar plot for HDCyto data using median marker expression over all cells to calculate dissimilarities between samples (other aggregations are also possible, and one could reduce the number of top varying markers to include in the calculation). Ideally, samples should cluster well within the same condition, although this depends on the magnitude of the difference between experimental conditions. With this diagnostic, one can identify the outlier samples and eliminate them if the circumstances warrant it. In our MDS plot on median marker expression values (see Figure 3), we can see that the first dimension (MDS1) separates the unstimulated and stimulated

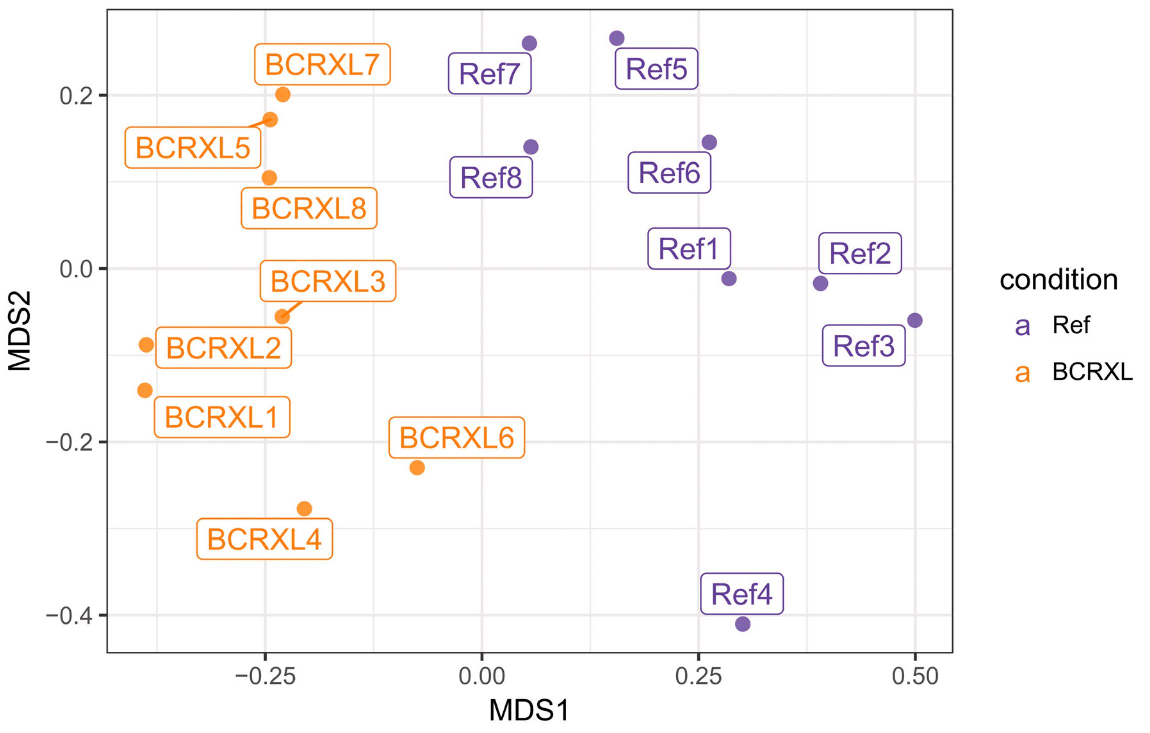


Figure 3. MDS plot for the unstimulated (Ref) and stimulated with BCR/FcR-XL (BCRXL) samples obtained for each of the 8 healthy donors in the PBMC dataset. Calculations are based on the median (arcsinh-transformed) marker expression of 10 lineage markers and 14 functional markers across all cells measured for each sample. Distances between samples in the plot approximate the typical change in medians. Numbers in the label names indicate patient IDs.

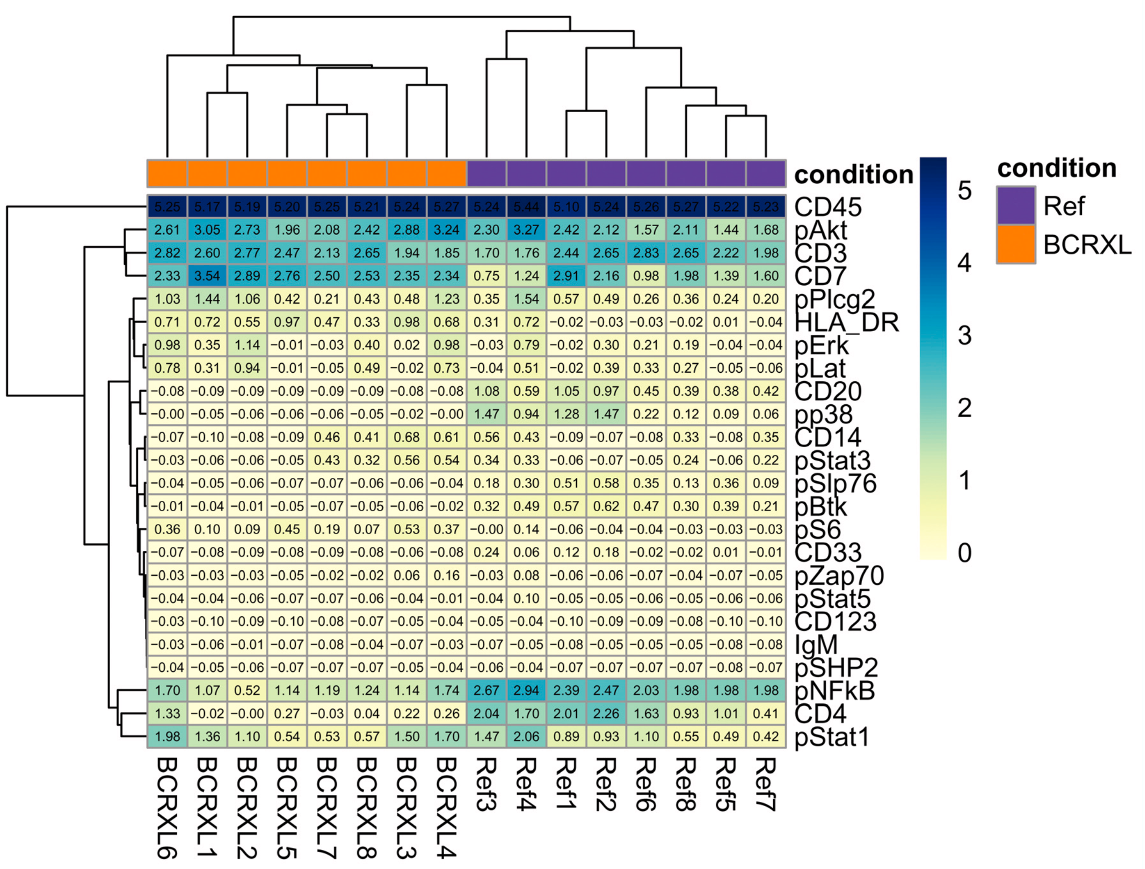
samples reasonably well. The second dimension (MDS2) represents, to some degree, differences between patients. Most of the samples that originate from the same patient are placed at a similar point along the y-axis, for example, samples from patients 7, 5, and 8 are at the top of the plot, samples from patient 4 are located at the bottom of the plot. This also indicates that the marker expression of individual patients is driving similarity and perhaps should be formally accounted for in the downstream statistical modeling.

```
library(dplyr)
# Get the median marker expression per sample

expr_median_sample_tbl <- data.frame(sample_id = sample_ids, expr) %>%
  group_by(sample_id) %>%  summarize_all(funs(median))

expr_median_sample <- t(expr_median_sample_tbl[, -1])
colnames(expr_median_sample) <- expr_median_sample_tbl$sample_id

library(limma)
mds <- plotMDS(expr_median_sample, plot = FALSE)

library(ggrepel)
ggdf <- data.frame(MDS1 = mds$x, MDS2 = mds$y,
  sample_id = colnames(expr_median_sample))
mm <- match(ggdf$sample_id, md$sample_id)
ggdf$condition <- md$condition[mm]
```

```
ggplot(ggdf, aes(x = MDS1, y = MDS2, color = condition)) +
  geom_point(size = 2, alpha = 0.8) +
  geom_label_repel(aes(label = sample_id)) +
  theme_bw() +
  scale_color_manual(values = color_conditions) +
  coord_fixed()
```

In contrast to genomic applications, the number of variables measured for each sample is much lower in HDCyto data. In the former, thousands of genes are surveyed, whereas in the latter, ~20–50 antigens are typically targeted. Similar to the MDS plot above, a heatmap of the same data also gives insight into the structure of the data. The heatmap shows median marker intensities with clustered columns (samples) and rows (markers). We have used hierarchical clustering with average linkage and euclidean distance, but also Ward's linkage could be used (Bruggner *et al.*, 2014), and in CyTOF applications, a cosine distance shows good performance (Bendall *et al.*, 2014). In this plot, we can see which markers drive the observed clustering of samples (see Figure 4).

As with the MDS plot, the dendrogram separates the reference and stimulated samples very well. Also, similar groupings of patients within experimental conditions are observed (patients 1-2, 5-7-8 and 3-4 are together in both conditions).



	BCRXL6	BCRXL1	BCRXL2	BCRXL5	BCRXL7	BCRXL8	BCRXL3	BCRXL4	Ref3	Ref4	Ref1	Ref2	Ref6	Ref8	Ref5	Ref7
CD45	5.25	5.17	5.19	5.20	5.25	5.21	5.24	5.27	5.24	5.44	5.10	5.24	5.26	5.27	5.22	5.23
pAkt	2.61	3.05	2.73	1.96	2.08	2.42	2.88	3.24	2.30	3.27	2.42	2.12	1.57	2.11	1.44	1.68
CD3	2.82	2.60	2.77	2.47	2.13	2.65	1.94	1.85	1.70	1.76	2.44	2.65	2.83	2.65	2.22	1.98
CD7	2.33	3.54	2.89	2.76	2.50	2.53	2.35	2.34	0.75	1.24	2.91	2.16	0.98	1.98	1.39	1.60
pPlcg2	1.03	1.44	1.06	0.42	0.21	0.43	0.48	1.23	0.35	1.54	0.57	0.49	0.26	0.36	0.24	0.20
HLA_DR	0.71	0.72	0.55	0.97	0.47	0.33	0.98	0.68	0.31	0.72	−0.02	−0.03	−0.03	−0.02	0.01	−0.04
pErk	0.98	0.35	1.14	−0.01	−0.03	0.40	0.02	0.98	−0.03	0.79	−0.02	0.30	0.21	0.19	−0.04	−0.04
pLat	0.78	0.31	0.94	−0.01	−0.05	0.49	−0.02	0.73	−0.04	0.51	−0.02	0.39	0.33	0.27	−0.05	−0.06
CD20	−0.08	−0.09	−0.09	−0.09	−0.09	−0.09	−0.08	−0.08	1.08	0.59	1.05	0.97	0.45	0.39	0.38	0.42
pp38	−0.00	−0.05	−0.06	−0.06	−0.06	−0.05	−0.02	−0.00	1.47	0.94	1.28	1.47	0.22	0.12	0.09	0.06
CD14	−0.07	−0.10	−0.08	−0.09	0.46	0.41	0.68	0.61	0.56	0.43	−0.09	−0.07	−0.08	0.33	−0.08	0.35
pStat3	−0.03	−0.06	−0.06	−0.05	0.43	0.32	0.56	0.54	0.34	0.33	−0.06	−0.07	−0.05	0.24	−0.06	0.22
pSlp76	−0.04	−0.05	−0.06	−0.07	−0.07	−0.06	−0.06	−0.04	0.18	0.30	0.51	0.58	0.35	0.13	0.36	0.09
pBtk	−0.01	−0.04	−0.01	−0.05	−0.07	−0.05	−0.06	−0.02	0.32	0.49	0.57	0.62	0.47	0.30	0.39	0.21
pS6	0.36	0.10	0.09	0.45	0.19	0.07	0.53	0.37	−0.00	0.14	−0.06	−0.04	−0.04	−0.03	−0.03	−0.03
CD33	−0.07	−0.08	−0.09	−0.08	−0.09	−0.08	−0.06	−0.08	0.24	0.06	0.12	0.18	−0.02	−0.02	0.01	−0.01
pZap70	−0.03	−0.03	−0.03	−0.05	−0.02	−0.02	0.06	0.16	−0.03	0.08	−0.06	−0.06	−0.07	−0.04	−0.07	−0.05
pStat5	−0.04	−0.04	−0.06	−0.07	−0.07	−0.06	−0.04	−0.01	−0.04	0.10	−0.05	−0.05	−0.06	−0.05	−0.06	−0.06
CD123	−0.03	−0.10	−0.09	−0.10	−0.08	−0.07	−0.05	−0.04	−0.05	−0.04	−0.10	−0.09	−0.09	−0.08	−0.10	−0.10
IgM	−0.03	−0.06	−0.01	−0.07	−0.08	−0.04	−0.07	−0.03	−0.07	−0.05	−0.08	−0.05	−0.05	−0.05	−0.08	−0.08
pSHP2	−0.04	−0.05	−0.06	−0.07	−0.07	−0.07	−0.05	−0.04	−0.06	−0.04	−0.07	−0.07	−0.07	−0.07	−0.08	−0.07
pNFkB	1.70	1.07	0.52	1.14	1.19	1.24	1.14	1.74	2.67	2.94	2.39	2.47	2.03	1.98	1.98	1.98
CD4	1.33	−0.02	−0.00	0.27	−0.03	0.04	0.22	0.26	2.04	1.70	2.01	2.26	1.63	0.93	1.01	0.41
pStat1	1.98	1.36	1.10	0.54	0.53	0.57	1.50	1.70	1.47	2.06	0.89	0.93	1.10	0.55	0.49	0.42

Figure 4. **Heatmap of the median (arcsinh-transformed) marker expression of 10 lineage markers and 14 functional markers across all cells measured for each sample in the PBMC dataset.** Color-coded with yellow for lower expression and blue for higher expression. The numbers in the heatmap represent the actual expression values. Dendrograms present clustering of samples (columns) and markers (rows) which is based on hierarchical clustering with Euclidean distance metric and average linkage. The two conditions: unstimulated (Ref) and stimulated with BCR/FcR-XL (BCRXL) for each of the 8 healthy donors are presented with a bar colored by experimental condition on top of the heatmap. Numbers in the column label names indicate patient IDs.

```
library(RColorBrewer)
library(pheatmap)

# Column annotation for the heatmap
mm <- match(colnames(expr_median_sample), md$sample_id)
annotation_col <- data.frame(condition = md$condition[mm],
  row.names = colnames(expr_median_sample))
annotation_colors <- list(condition = color_conditions)

# Colors for the heatmap
color <- colorRampPalette(brewer.pal(n = 9, name = "YlGnBu"))(100)

pheatmap(expr_median_sample, color = color, display_numbers = TRUE,
  number_color = "black", fontsize_number = 5, annotation_col = annotation_col,
  annotation_colors = annotation_colors, clustering_method = "average")
```

Marker ranking based on the non-redundancy score

In this step, we identify the ability of markers to explain the variance observed in each sample. In particular, we calculate the PCA-based non-redundancy score (NRS) (Levine *et al.*, 2015). Markers with higher score explain a larger portion of variability present in a given sample.

The average NRS can be used to select a subset of markers that are non-redundant in each sample but at the same time capture the overall diversity between samples. Such a subset of markers can be then used for cell population identification analysis (i.e. clustering). We note that there is no precise rule on how to choose the right cutoff for marker inclusion, but one of the approaches is to select a suitable number of the top-scoring markers. The number can be chosen by analyzing the plot with the NR scores (see Figure 5), where the markers are sorted by the decreasing average NRS. Based on the prior biological knowledge, one can refine the marker selection and drop out markers that are not likely to distinguish cell populations of interest, even if they have high scores, and add in markers with low scores but known to be important in discerning cell subgroups (Levine *et al.*, 2015). Thus, the NRS analysis serves more as a guide to marker selection and is not meant as a hardcoded rule.

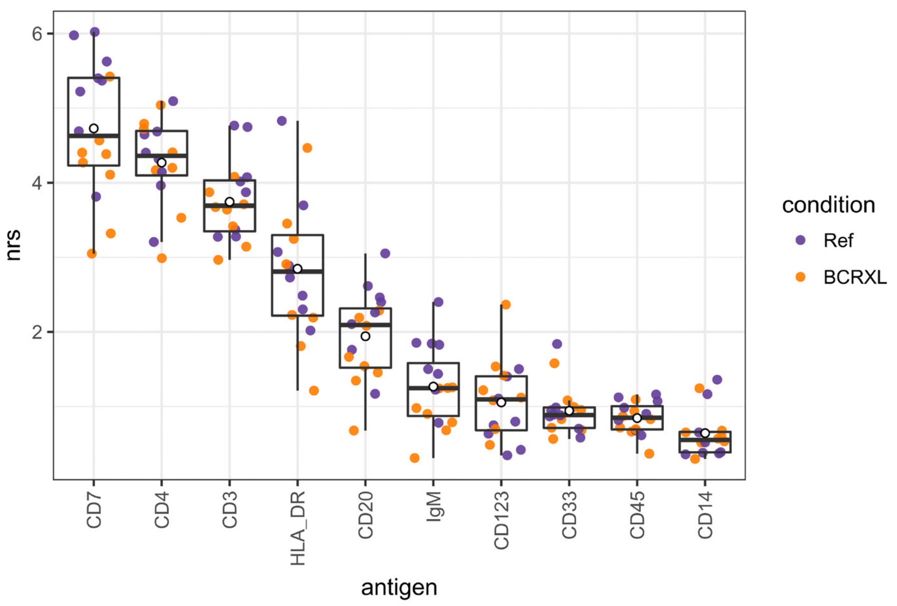


Figure 5. Non-redundancy scores for each of the 10 lineage markers and all samples in the PBMC dataset. The full points represent the per-sample NR scores (colored by experimental conditions), while empty black circles indicate the mean NR scores from all the samples. Markers on the x-axis are sorted according to the decreasing average NRS.

In the dataset considered here (Bodenmiller *et al.*, 2012; Bruggner *et al.*, 2014) we want to use all the 10 lineage markers, so there is no explicit need to restrict the set of cell surface markers, and the NRS serve as another quality control step. There may be other situations where this feature selection step would be of interest, for example, in panel design (Levine *et al.*, 2015).

```
## Define a function that calculates the NRS per sample
NRS <- function(x, ncomp = 3){
  pr <- prcomp(x, center = TRUE, scale. = FALSE)
  score <- rowSums(outer(rep(1, ncol(x)),
    pr$sdev[1:ncomp]^2) * abs(pr$rotation[,1:ncomp]))
  return(score)
}

## Calculate the score
nrs_sample <- fsApply(fcs[, lineage_markers], NRS, use.exprs = TRUE)
rownames(nrs_sample) <- md$sample_id
nrs <- colMeans(nrs_sample, na.rm = TRUE)

## Plot the NRS for ordered markers
lineage_markers_ord <- names(sort(nrs, decreasing = TRUE))
nrs_sample <- data.frame(nrs_sample)
nrs_sample$sample_id <- rownames(nrs_sample)

ggdf <- melt(nrs_sample, id.var = "sample_id",
  value.name = "nrs", variable.name = "antigen")

ggdf$antigen <- factor(ggdf$antigen, levels = lineage_markers_ord)
mm <- match(ggdf$sample_id, md$sample_id)
ggdf$condition <- md$condition[mm]

ggplot(ggdf, aes(x = antigen, y = nrs)) +
  geom_point(aes(color = condition), alpha = 0.9,
    position = position_jitter(width = 0.3, height = 0)) +
  geom_boxplot(outlier.color = NA, fill = NA) +
  stat_summary(fun.y = "mean", geom = "point", shape = 21, fill = "white") +
  theme_bw() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1)) +
  scale_color_manual(values = color_conditions)
```

Cell population identification with FlowSOM and ConsensusClusterPlus

Cell population identification typically has been carried out by manual gating, a method based on visual inspection of a series of two-dimensional scatterplots. At each step, a subset of cells, either positive or negative for the two visualized markers, is selected and further stratified in the subsequent iterations until populations of interest across a range of marker combinations are captured. However, manual gating has drawbacks, such as subjectivity, bias toward well-known cell types, and inefficiency when analyzing large datasets, which also contribute to a lack of reproducibility (Saeys *et al.*, 2016).

Considerable effort has been made to improve and automate cell population identification, such as unsupervised clustering (Aghaeepour *et al.*, 2013). However, not all methods scale well in terms of performance and speed from the lower dimensionality flow cytometry data to the higher dimensionality mass cytometry data (Weber & Robinson, 2016), since clustering in higher dimensions can suffer the “curse of dimensionality”.

Beside the mathematical and algorithmic challenges of clustering, cell population identification may be difficult due to the chemical and biological aspects of the cytometry experiment itself. Therefore, caution should be taken when designing panels aimed at detecting rare cell populations by assigning higher sensitivity metals to rare markers. The right choice of a marker panel used for clustering can also be important. It should include all markers that are relevant for cell type identification.

In this workflow, we conduct cell clustering with *FlowSOM* (Van Gassen *et al.*, 2015) and *ConsensusClusterPlus* (Wilkerson & Hayes, 2010), which appeared amongst the fastest and best performing clustering approaches in a recent study of HDCyto datasets (Weber & Robinson, 2016). This ensemble showed strong performance in detecting both high and low frequency cell populations and is one of the fastest methods to run, which enables its interactive usage. We use a slight modification of the original workflow presented in the *FlowSOM* vignette, which we find more flexible. In particular, we directly call the `ConsensusClusterPlus` function that is embedded in `metaClustering_consensus`. Thus, we are able to access all the functionality of the *ConsensusClusterPlus* package to identify the number of clusters.

The *FlowSOM* workflow consists of three main steps. First, a self-organizing map (SOM) is built using the `BuildSOM` function, where cells are assigned according to their similarities to 100 (by default) grid points (or, so-called codebook vectors or codes) of the SOM. The building of a minimal spanning tree, which is mainly used for graphical representation of the clusters, is skipped in this pipeline. And finally, *metaclustering* of the SOM codes, is performed directly with the `ConsensusClusterPlus` function. Additionally, we add an optional round of manual expert-based merging of the metaclusters and allow this to be done in a reproducible fashion.

FlowSOM output can be sensitive to random starts (Weber & Robinson, 2016). To make results reproducible, one must specify the seed for the random number generation in R using function `set.seed`. It is also advisable to rerun analyses with multiple random seeds, for two reasons. First, one can see how robust the detected clusters are, and second, when the goal is to find smaller cell populations, it may happen that, in some runs, random starting points do not represent rare cell populations, as the chance of selecting starting cells from them is low and they are merged into a larger cluster.

It is important to point out that we cluster all cells from all samples together. This strategy is beneficial, since we directly obtain cluster assignment for each cell, we label cell populations only once and the mapping of cell types between samples is automatically consistent. For a list of alternative approaches and their advantages and disadvantages, please refer to the Discussion section, where we consider: clustering per sample, clustering of data from different measurement batches and down-sampling in case of widely varying numbers of cells per sample.

In our analysis, cell populations are identified using only the 10 lineage markers as defined in the `BuildSOM` function with the `colsToUse` argument.

```
library(FlowSOM)
fsom <- ReadInput(fcs, transform = FALSE, scale = FALSE)
set.seed(1234)
som <- BuildSOM(fsom, colsToUse = lineage_markers)
## Get the cell clustering into 100 SOM codes
cell_clustering_som <- som$map$mapping[,1]
```

Automatic approaches for selecting the number of clusters in HDCyto data do not always succeed (Weber & Robinson, 2016). In general, we therefore recommend some level of over-clustering, and if desired, manual merging of clusters. Such a hierarchical approach is especially suited when the goal is to detect smaller cell populations.

The SPADE analysis performed by Bodenmiller *et al.* (Bodenmiller *et al.*, 2012) identified 6 main cell types (T-cells, monocytes, dendritic cells, B-cells, NK cells and surface- cells) that were further stratified into 14 more specific subpopulations (CD4+ T-cells, CD8+ T-cells, CD14+ HLA-DR high monocytes, CD14+ HLA-DR med monocytes, CD14+ HLA-DR low monocytes, CD14- HLA-DR high monocytes, CD14- HLA-DR med monocytes, CD14- HLA-DR low monocytes, dendritic cells, IgM+ B-cells, IgM- B-cells, NK cells, surface- CD14+ cells and surface- CD14- cells). In our analysis, we are interested in identifying the 6 main PBMC populations, including: CD4+ T-cells, CD8+ T-cells, monocytes, dendritic cells, NK cells and B-cells. Following the concept of over-clustering we perform the metaclustering of the (by default) 100 SOM codes into more than expected number of groups. For example, stratification into 20 groups should give enough resolution to detect these main clusters. We can explore the clustering in a wide variety of visualizations: t-SNE plots, heatmaps and a plot generated by `ConsensusClusterPlus` called “delta area”.

When the interest is in studying more specific subpopulations at higher detail, one can follow a strategy of reclustering as described in the Obtaining higher resolution section, where we propose to repeat the workflow (clustering and differential analyses) after gating out a selected cell type (e.g. one of the large populations).

We call `ConsensusClusterPlus` with maximum number of clusters `maxK = 20` and other clustering parameters set to the values as in the `metaClustering_consensus` function from the *FlowSOM* package. Again, to ensure that the analyses are reproducible, we define the random seed.

```
## Metaclustering into 20 clusters with ConsensusClusterPlus
library(ConsensusClusterPlus)

codes <- som$map$codes
plot_outdir <- "consensus_plots"
nmc <- 20

mc <- ConsensusClusterPlus(t(codes), maxK = nmc, reps = 100,
  pItem = 0.9, pFeature = 1, title = plot_outdir, plot = "png",
  clusterAlg = "hc", innerLinkage = "average", finalLinkage = "average",
  distance = "euclidean", seed = 1234)

## Get cluster ids for each cell
code_clustering1 <- mc[[nmc]]$consensusClass
cell_clustering1 <- code_clustering1[cell_clustering_som]
```

We can then investigate characteristics of identified clusters with heatmaps that illustrate median marker expression in each cluster (see Figure 6). As the range of marker expression can vary substantially from marker to marker, we use the 0–1 transformed data (`expr01`) for some visualizations. However, to stay consistent with *FlowSOM* and *ConsensusClusterPlus*, we use the (arcsinh-transformed) unscaled data (`expr`) to generate the dendrogram of the hierarchical structure of metaclusters.

Instead of using only medians, which do not give a full representation of cluster specifics, one can plot the entire marker expression distribution in each cluster (see Figure 7). Such a plot gives more detailed profile of each cluster,

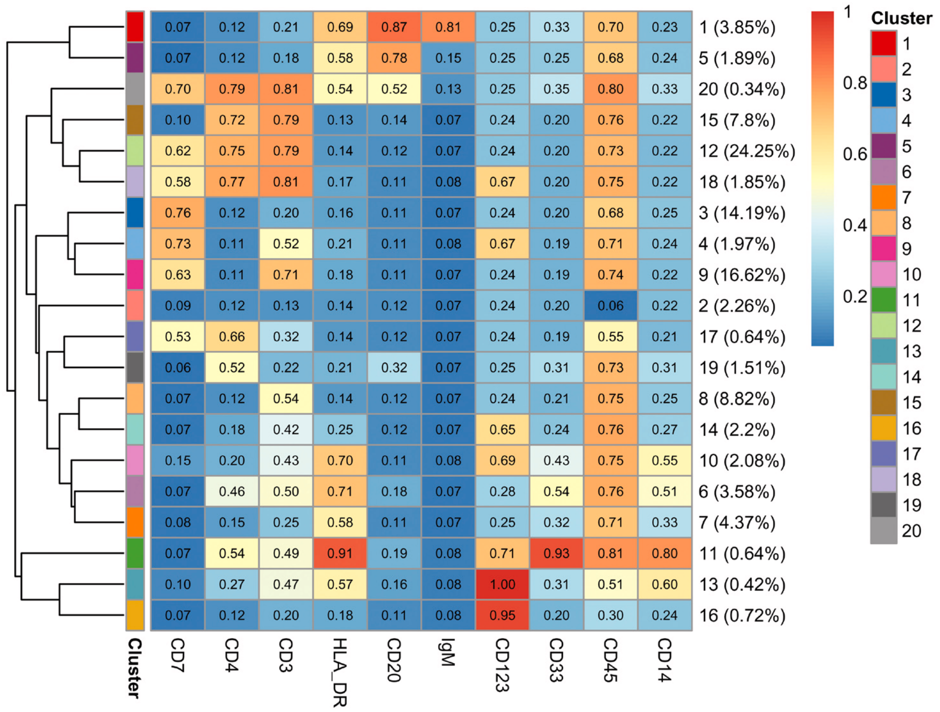


Figure 6. Heatmap of the median marker intensities of the 10 lineage markers across the 20 cell populations obtained with FlowSOM after the metaclustering step with ConsensusClusterPlus (PBMC data). The color in the heatmap represents the median of the arcsinh, 0-1 transformed marker expression calculated over cells from all the samples, varying from blue for lower expression to red for higher expression. The numbers indicate the actual expression values. The dendrogram on the left represents the hierarchical similarity between the 20 metaclusters (metric: Euclidean distance; linkage: average). Each cluster has a unique color assigned (bar on the left) which is identical in other visualizations of these 20 clusters (e.g. Figure t-SNE map in 10) facilitating the figure interpretation. Values in the brackets next to the cluster numbers indicate the relative size of clusters.

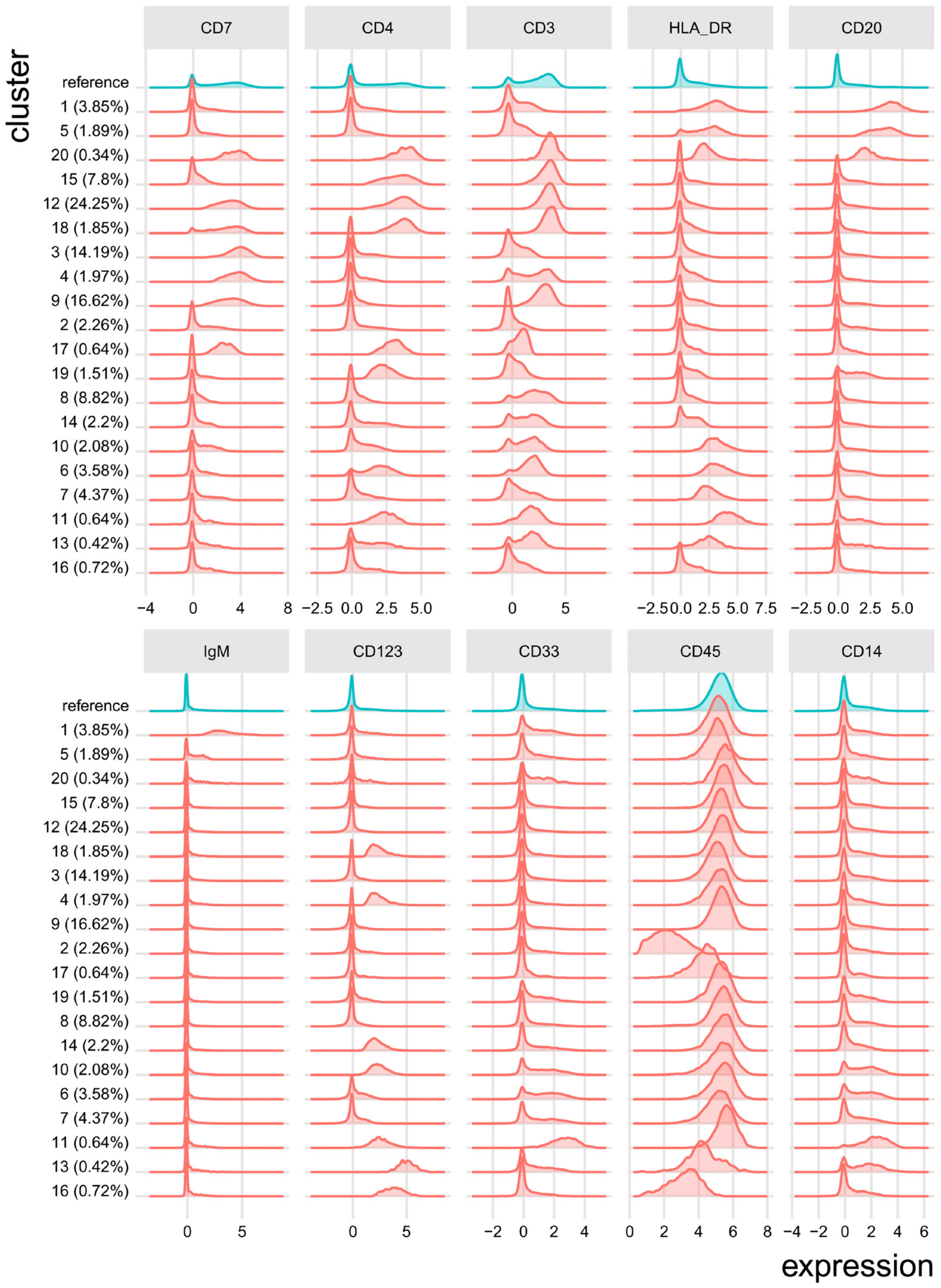


Figure 7. Distributions of marker intensities (arcsinh-transformed) of the 10 lineage markers in the 20 cell populations obtained with FlowSOM after the metaclustering step with ConsensusClusterPlus (PBMC data). Red densities represent marker expression for cells in a given cluster. Green densities are calculated over all the cells and serve as a reference.

but represents an increase in the amount of information to interpret. Heatmaps give the overall overview of clusters, are quicker and easier to interpret, and together with the dendrogram can be a good basis for further cluster merging (see Cluster merging and annotation section).

Since we will use the heatmap and density plots again later on in this workflow, in code chunks below, we create wrapper functions that generate these two types of plots.

```
# Define cluster colors (here there are 30 colors)
color_clusters <- c("#DC050C", "#FB8072", "#1965B0", "#7BAFDE", "#882E72",
  "#B17BA6", "#FF7F00", "#FDB462", "#E7298A", "#E78AC3",
  "#33A02C", "#B2DF8A", "#55A1B1", "#8DD3C7", "#A6761D",
  "#E6AB02", "#7570B3", "#BEAED4", "#666666", "#999999",
  "#aa8282", "#d4b7b7", "#8600bf", "#ba5ce3", "#808000",
  "#aeae5c", "#1e90ff", "#00bfff", "#56ff0d", "#ffff00")

plot_clustering_heatmap_wrapper <- function(expr, expr01,
  cell_clustering, color_clusters, cluster_merging = NULL){

  # Calculate the median expression
  expr_median <- data.frame(expr, cell_clustering = cell_clustering) %>%
    group_by(cell_clustering) %>% summarize_all(funs(median))
  expr01_median <- data.frame(expr01, cell_clustering = cell_clustering) %>%
    group_by(cell_clustering) %>% summarize_all(funs(median))

  # Calculate cluster frequencies
  clustering_table <- as.numeric(table(cell_clustering))
  clustering_prop <- round(clustering_table / sum(clustering_table) * 100, 2)

  # Sort the cell clusters with hierarchical clustering
  d <- dist(expr_median[, colnames(expr)], method = "euclidean")
  cluster_rows <- hclust(d, method = "average")

  expr_heat <- as.matrix(expr01_median[, colnames(expr01)])
  rownames(expr_heat) <- expr01_median$cell_clustering

  # Colors for the heatmap
  color_heat <- colorRampPalette(rev(brewer.pal(n = 9, name = "RdYlBu")))(100)
  legend_breaks = seq(from = 0, to = 1, by = 0.2)
  labels_row <- paste0(expr01_median$cell_clustering, " (", clustering_prop ,"%)")

  # Annotation for the original clusters
  annotation_row <- data.frame(Cluster = factor(expr01_median$cell_clustering))
  rownames(annotation_row) <- rownames(expr_heat)
  color_clusters1 <- color_clusters[1:nlevels(annotation_row$Cluster)]
  names(color_clusters1) <- levels(annotation_row$Cluster)
  annotation_colors <- list(Cluster = color_clusters1)
  # Annotation for the merged clusters
  if(!is.null(cluster_merging)){
    cluster_merging$new_cluster <- factor(cluster_merging$new_cluster)
    annotation_row$Cluster_merging <- cluster_merging$new_cluster
    color_clusters2 <- color_clusters[1:nlevels(cluster_merging$new_cluster)]
    names(color_clusters2) <- levels(cluster_merging$new_cluster)
    annotation_colors$Cluster_merging <- color_clusters2
  }

  pheatmap(expr_heat, color = color_heat, cluster_cols = FALSE,
    cluster_rows = cluster_rows, labels_row = labels_row,
    display_numbers = TRUE, number_color = "black",
    fontsize = 8, fontsize_number = 6,  legend_breaks = legend_breaks,
    annotation_row = annotation_row, annotation_colors = annotation_colors)

}

plot_clustering_heatmap_wrapper(expr = expr[, lineage_markers_ord],
  expr01 = expr01[, lineage_markers_ord],
  cell_clustering = cell_clustering1, color_clusters = color_clusters)

library(ggridges)

plot_clustering_distr_wrapper <- function(expr, cell_clustering){
  # Calculate the median expression
  cell_clustering <- factor(cell_clustering)
  expr_median <- data.frame(expr, cell_clustering = cell_clustering) %>%
  group_by(cell_clustering) %>% summarize_all(funs(median))
  # Sort the cell clusters with hierarchical clustering
  d <- dist(expr_median[, colnames(expr)], method = "euclidean")
  cluster_rows <- hclust(d, method = "average")
  # Calculate cluster frequencies
  freq_clust <- table(cell_clustering)
  freq_clust <- round(as.numeric(freq_clust)/sum(freq_clust)*100, 2)
  cell_clustering <- factor(cell_clustering,
    labels = paste0(levels(cell_clustering), " (", freq_clust, "%)"))
  ### Data organized per cluster
  ggd <- melt(data.frame(cluster = cell_clustering, expr),
    id.vars = "cluster", value.name = "expression",
    variable.name = "antigen")
  ggd$antigen <- factor(ggd$antigen, levels = colnames(expr))
  ggd$reference <- "no"
  ### The reference data
  ggd_bg <- ggd
  ggd_bg$cluster <- "reference"
  ggd_bg$reference <- "yes"

  ggd_plot <- rbind(ggd, ggd_bg)
  ggd_plot$cluster <- factor(ggd_plot$cluster,
    levels = c(levels(cell_clustering)[rev(cluster_rows$order)], "reference"))

  ggplot() +
    geom_density_ridges(data = ggd_plot, aes(x = expression, y = cluster,
      color = reference, fill = reference), alpha = 0.3) +
    facet_wrap( ~ antigen, scales = "free_x", nrow = 2) +
    theme_ridges() +
    theme(axis.text = element_text(size = 7),
      strip.text = element_text(size = 7), legend.position = "none")

}

plot_clustering_distr_wrapper(expr = expr[, lineage_markers_ord],
  cell_clustering = cell_clustering1)
```

In addition to investigating expression of the lineage markers, we can also have a look at expression of the functional markers. We propose a heatmap that depicts median expression of functional markers in each sample (see Figure 8) so the potential differential expression can be investigated already at this data exploration step before the formal testing is done. In order to plot all the heatmaps in one panel, we use the *ComplexHeatmap* package. Again, we created a wrapper function that does all the plotting and can be reused through the workflow.

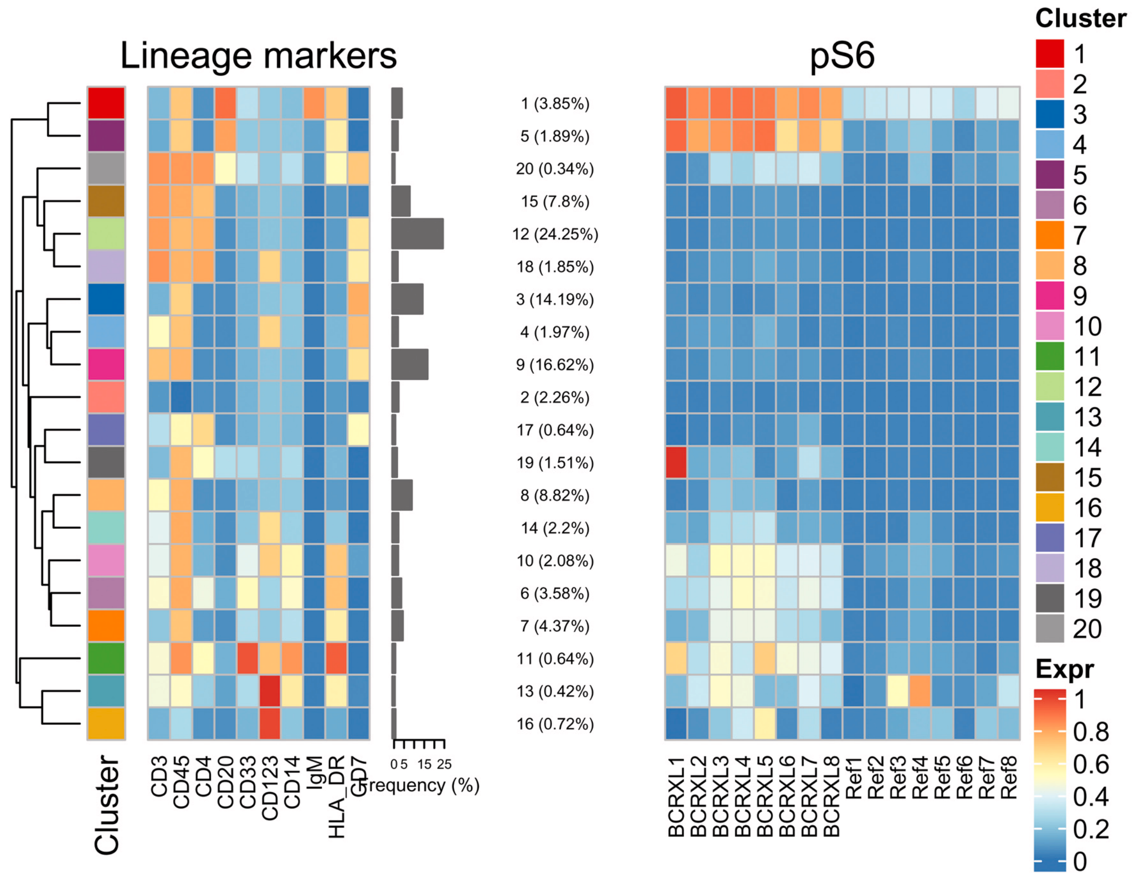


Figure 8. Heatmap of the median marker intensities of the 10 lineage markers and one signaling marker (pS6) across the 20 cell populations obtained with FlowSOM after the metaclustering step with ConsensusClusterPlus (PBMC data). This plot was generated using ComplexHeatmaps. The left panel presents a heatmap analogous to the one in Figure 6, additionally, it contains a barplot along the rows (clusters) which represents the relative sizes of clusters. Heatmap on the right represents the median of the arcsinh, 0-1 transformed marker expression for a signaling marker pS6 calculated over cells in each sample (columns) individually.

```
library(ComplexHeatmap)

plot_clustering_heatmap_wrapper2 <- function(expr, expr01,
  lineage_markers, functional_markers = NULL, sample_ids = NULL,
  cell_clustering, color_clusters, cluster_merging = NULL,
  plot_cluster_annotation = TRUE){

  # Calculate the median expression of lineage markers
  expr_median <- data.frame(expr[, lineage_markers],
    cell_clustering = cell_clustering) %>%
    group_by(cell_clustering) %>% summarize_all(funs(median))
  expr01_median <- data.frame(expr01[, lineage_markers],
    cell_clustering = cell_clustering) %>%
    group_by(cell_clustering) %>% summarize_all(funs(median))

  # Calculate cluster frequencies
  clustering_table <- as.numeric(table(cell_clustering))
  clustering_prop <- round(clustering_table / sum(clustering_table) * 100, 2)

  # Sort the cell clusters with hierarchical clustering
  d <- dist(expr_median[, lineage_markers], method = "euclidean")
  cluster_rows <- hclust(d, method = "average")

  expr_heat <- as.matrix(expr01_median[, lineage_markers])

  # Median expression of functional markers in each sample per cluster
  expr_median_sample_cluster_tbl <- data.frame(expr01[, functional_markers,
    drop = FALSE], sample_id = sample_ids, cluster = cell_clustering) %>%
    group_by(sample_id, cluster) %>% summarize_all(funs(median))

  # Colors for the heatmap
  color_heat <- colorRampPalette(rev(brewer.pal(n = 9, name = "RdYlBu")))(100)
  legend_breaks = seq(from = 0, to = 1, by = 0.2)
  labels_row <- paste0(expr01_median$cell_clustering, " (", clustering_prop , "%)")

 ### Annotation for the original clusters
  annotation_row1 <- data.frame(Cluster = factor(expr01_median$cell_clustering))
  color_clusters1 <- color_clusters[1:nlevels(annotation_row1$Cluster)]
  names(color_clusters1) <- levels(annotation_row1$Cluster)

 ### Annotation for the merged clusters
  if(!is.null(cluster_merging)){
    mm <- match(annotation_row1$Cluster, cluster_merging$original_cluster)
    annotation_row2 <- data.frame(Cluster_merging =
        factor(cluster_merging$new_cluster[mm]))
    color_clusters2 <- color_clusters[1:nlevels(annotation_row2$Cluster_merging)]
    names(color_clusters2) <- levels(annotation_row2$Cluster_merging)
  }

 ### Heatmap annotation for the original clusters
  ha1 <- Heatmap(annotation_row1, name = "Cluster",
    col = color_clusters1, cluster_columns = FALSE,
    cluster_rows = cluster_rows, row_dend_reorder = FALSE,
    show_row_names = FALSE, width = unit(0.5, "cm"),
    rect_gp = gpar(col = "grey"))
  ### Heatmap annotation for the merged clusters
  if(!is.null(cluster_merging)){
    ha2 <- Heatmap(annotation_row2, name = "Cluster \nmerging",
      col = color_clusters2, cluster_columns = FALSE,
      cluster_rows = cluster_rows, row_dend_reorder = FALSE,
      show_row_names = FALSE, width = unit(0.5, "cm"),
      rect_gp = gpar(col = "grey"))
  }
  ### Cluster names and sizes - text
  ha_text <- rowAnnotation(text = row_anno_text(labels_row,
    gp = gpar(fontsize = 6)), width = max_text_width(labels_row))
  ### Cluster sizes - barplot
  ha_bar <- rowAnnotation("Frequency (%)" = row_anno_barplot(
    x = clustering_prop, border = FALSE, axis = TRUE,
    axis_gp = gpar(fontsize = 5), gp = gpar(fill = "#696969", col = "#696969"),
    bar_width = 0.9), width = unit(0.7, "cm"), show_annotation_name = TRUE,
    annotation_name_rot = 0, annotation_name_offset = unit(5, "mm"),
    annotation_name_gp = gpar(fontsize = 7))
  ### Heatmap for the lineage markers
  ht1 <- Heatmap(expr_heat, name = "Expr", column_title = "Lineage markers",
    col = color_heat, cluster_columns = FALSE, cluster_rows = cluster_rows,
    row_dend_reorder = FALSE, heatmap_legend_param = list(at = legend_breaks,
      labels = legend_breaks, color_bar = "continuous"),
    show_row_names = FALSE, row_dend_width = unit(2, "cm"),
    rect_gp = gpar(col = "grey"), column_names_gp = gpar(fontsize = 8))

  if(plot_cluster_annotation){
    draw_out <- ha1
  }else{
    draw_out <- NULL
  }
  if(!is.null(cluster_merging)){
    draw_out <- draw_out + ha2 + ht1 + ha_bar + ha_text
  }else{
    draw_out <- draw_out + ht1 + ha_bar + ha_text
  }

  ### Heatmaps for the signaling markers
  if(!is.null(functional_markers)){
    for(i in 1:length(functional_markers)){
      ## Rearange so the rows represent clusters
      expr_heat_fun <- as.matrix(dcast(expr_median_sample_cluster_tbl[,
        c("sample_id", "cluster", functional_markers[i])],
        cluster ~ sample_id, value.var = functional_markers[i])[, -1])

      draw_out <- draw_out + Heatmap(expr_heat_fun,
        column_title = functional_markers[i], col = color_heat,
        cluster_columns = FALSE, cluster_rows = cluster_rows,
        row_dend_reorder = FALSE, show_heatmap_legend = FALSE,
        show_row_names = FALSE, rect_gp = gpar(col = "grey"),
        column_names_gp = gpar(fontsize = 8))
    }
  }
  draw(draw_out, row_dend_side = "left")
}
```

```
plot_clustering_heatmap_wrapper2(expr = expr, expr01 = expr01,
  lineage_markers = lineage_markers, functional_markers = "pS6",
  sample_ids = sample_ids, cell_clustering = cell_clustering1,
  color_clusters = color_clusters, cluster_merging = NULL)
```

Visual representation with t-SNE

One of the most popular plots for representing single cell data are t-SNE plots, where each cell is represented in a lower, usually two-dimensional, space computed using t-stochastic neighbor embedding (t-SNE) (Van Der Maaten & Hinton, 2008). More generally, dimensionality reduction techniques represent the similarity of points in 2 or 3 dimensions, such that similar objects in high dimensional space are also similar in lower dimensional space. Mathematically, there are a myriad of ways to define this similarity. For example, principal components analysis (PCA) uses linear combinations of the original features to find orthogonal dimensions that show the highest levels of variability; the top 2 or 3 principal components can then be visualized.

Nevertheless, there are few notes of caution when using t-SNE or any other dimensionality reduction technique. Since they are based on preserving similarities between cells, those that are similar in the original space will be close in the 2D/3D representation, but the opposite does not always hold. In our experience, t-SNE with default parameters for HDCyto data is often suitable; for more guidance on the specifics of t-SNE, see How to Use t-SNE Effectively (Wattenberg *et al.*, 2016). Due to the stochastic nature of t-SNE optimization, rerunning the method will result in different lower dimensional projections, thus it is advisable to run it a few times to identify the common trends and get a feeling about the variability of the results. As with other methods, to be sure that the analysis is reproducible, the user can define the random seed.

t-SNE is a method that requires significant computational time to process the data even for tens of thousands of cells. CyTOF datasets are usually much larger and thus to keep running times reasonable, one may use a subset of cells; for example, here we use 2000 cells from each sample. The t-SNE map below is colored according to the expression level of the CD4 marker, highlighting that the CD4+ T-cells are placed to the left side of the plot (see Figure 9). In this way, one can use a collection of markers to highlight where cell types of interest are located on the *map*.

Instead of t-SNE, one could also use other dimension reduction techniques, such as PCA, diffusion maps, SIMLR (Wang *et al.*, 2017) or isomaps, some of which are conveniently available via the `cytof_dimReduction` function from the *cytofkit* package (Chen *et al.*, 2016). To speed up the t-SNE analysis, one could use a multicore

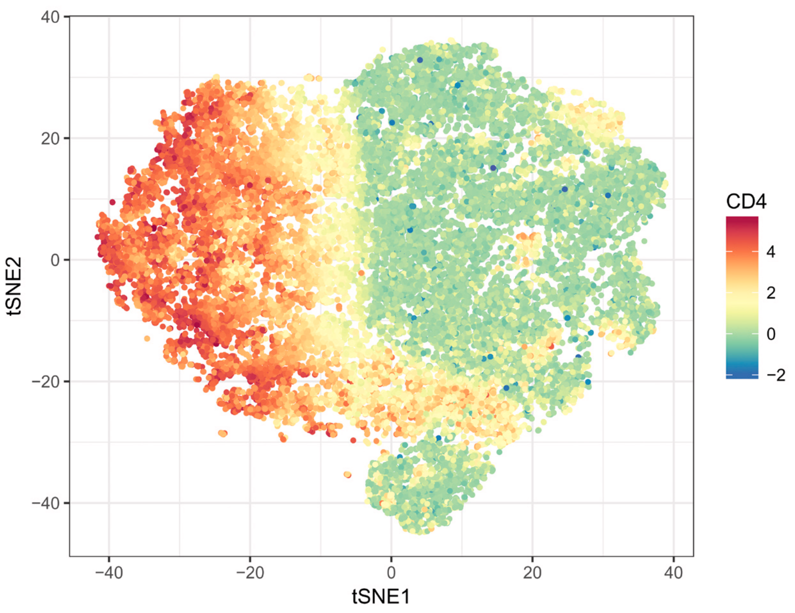


Figure 9. t-SNE plot based on the arcsinh-transformed expression of the 10 lineage markers in the cells from the PBMC dataset. t-SNE was run with no PCA step, perplexity equal to 30 and 1000 iterations. From each of the 16 samples, 2000 cells were randomly selected. Cells are colored according to the expression level of the CD4 marker.

version that is available via the *Rtsne.multicore* package. Alternative algorithms, such as largeVis (Tang *et al.*, 2016) (available via the *largeVis* package), can be used for dimensionality reduction of very large datasets without downsampling. Alternatively, the dimensionality reduction can be performed on the *codes* of the SOM, at a resolution (size of the SOM) specified by the user (see Figure 13).

```
## Find and skip duplicates
dups <- which(!duplicated(expr[, lineage_markers]))

## Data subsampling: create indices by sample
inds <- split(1:length(sample_ids), sample_ids)

## How many cells to downsample per-sample
tsne_ncells <- pmin(table(sample_ids), 2000)

## Get subsampled indices
set.seed(1234)
tsne_inds <- lapply(names(inds), function(i){
  s <- sample(inds[[i]], tsne_ncells[i], replace = FALSE)
  intersect(s, dups)
})

tsne_inds <- unlist(tsne_inds)

tsne_expr <- expr[tsne_inds, lineage_markers]

## Run t-SNE
library(Rtsne)

set.seed(1234)
tsne_out <- Rtsne(tsne_expr, check_duplicates = FALSE, pca = FALSE)

## Plot t-SNE colored by CD4 expression
dr <- data.frame(tSNE1 = tsne_out$Y[, 1], tSNE2 = tsne_out$Y[, 2],
  expr[tsne_inds, lineage_markers])

ggplot(dr,  aes(x = tSNE1, y = tSNE2, color = CD4)) +
  geom_point(size = 0.8) +
  theme_bw() +
  scale_color_gradientn("CD4",
    colours = colorRampPalette(rev(brewer.pal(n = 11, name = "Spectral")))(50))
```

We can color the cells by cluster. Ideally, cells of the same color should be close to each other (see Figure 10). When the plots are further stratified by sample (see Figure 11), we can verify whether similar cell populations are present in all replicates, which can help in identifying outlying samples. Optionally, stratification can be done by condition (see Figure 12). With such a spot-check plot, we can inspect whether differences in cell abundance are strong between conditions, and we can visualize and identify distinguishing clusters before applying formal statistical testing. A similar approach of data exploration was proposed in studies of treatment-specific differences of polyfunctional antigen-specific T-cells (Lin *et al.*, 2015).

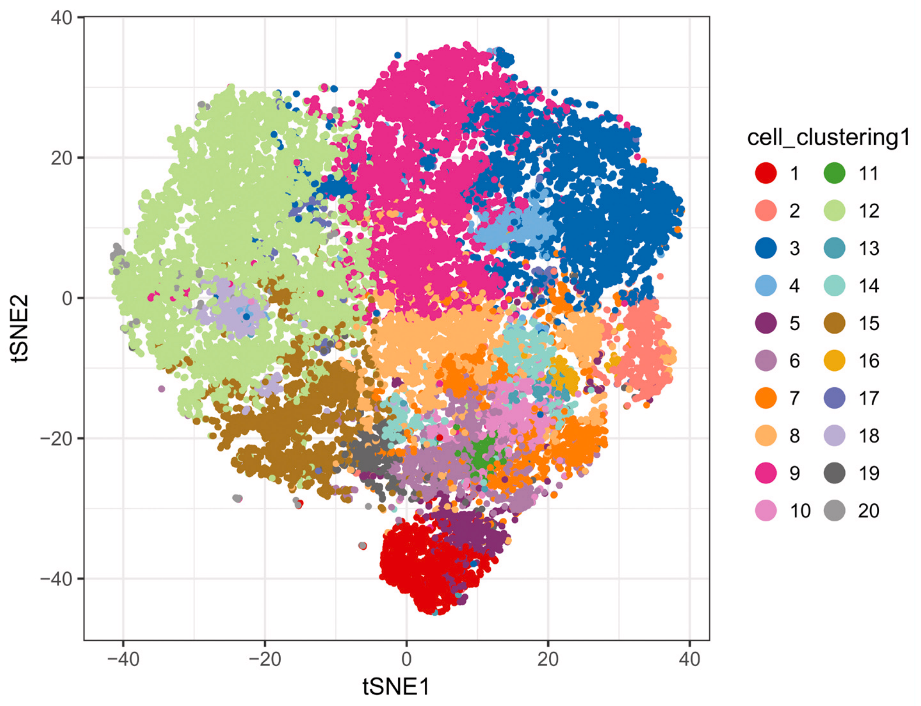


Figure 10. t-SNE plot based on the arcsinh-transformed expression of the 10 lineage markers in the cells from the PBMC dataset. From each of the 16 samples, 2000 cells were randomly selected. Cells are colored according to the 20 cell populations obtained with FlowSOM after the metaclustering step with ConsensusClusterPlus.

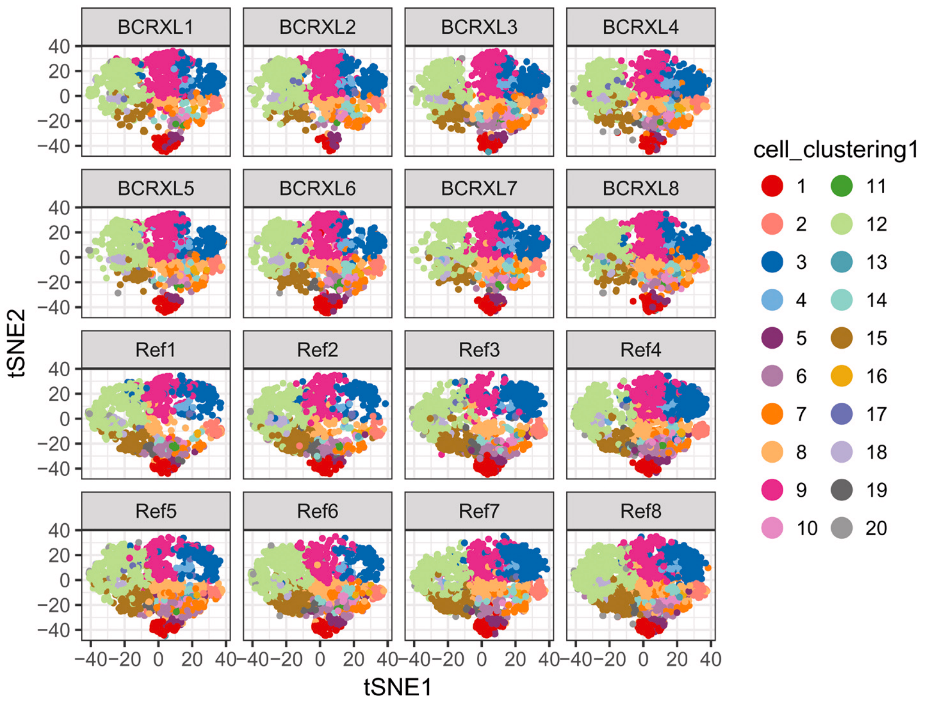


Figure 11. t-SNE plot as in the Figure 10, but stratified by sample.

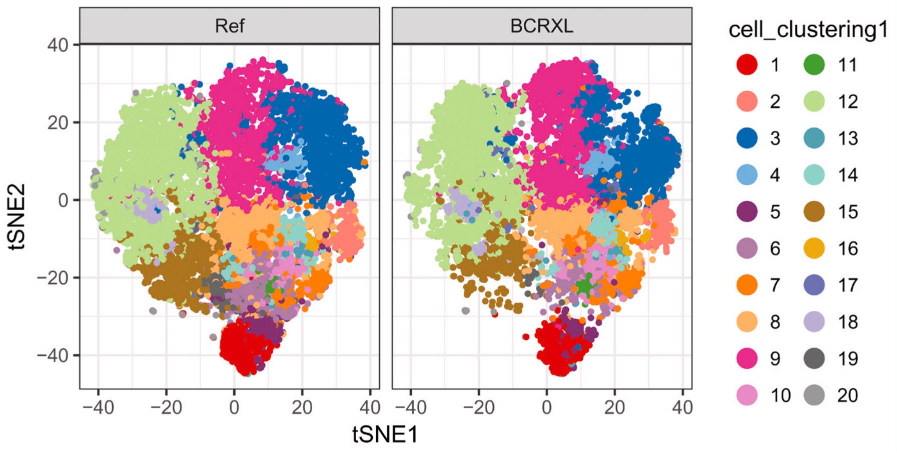


Figure 12. t-SNE plot as in the Figure 10, but stratified by condition.

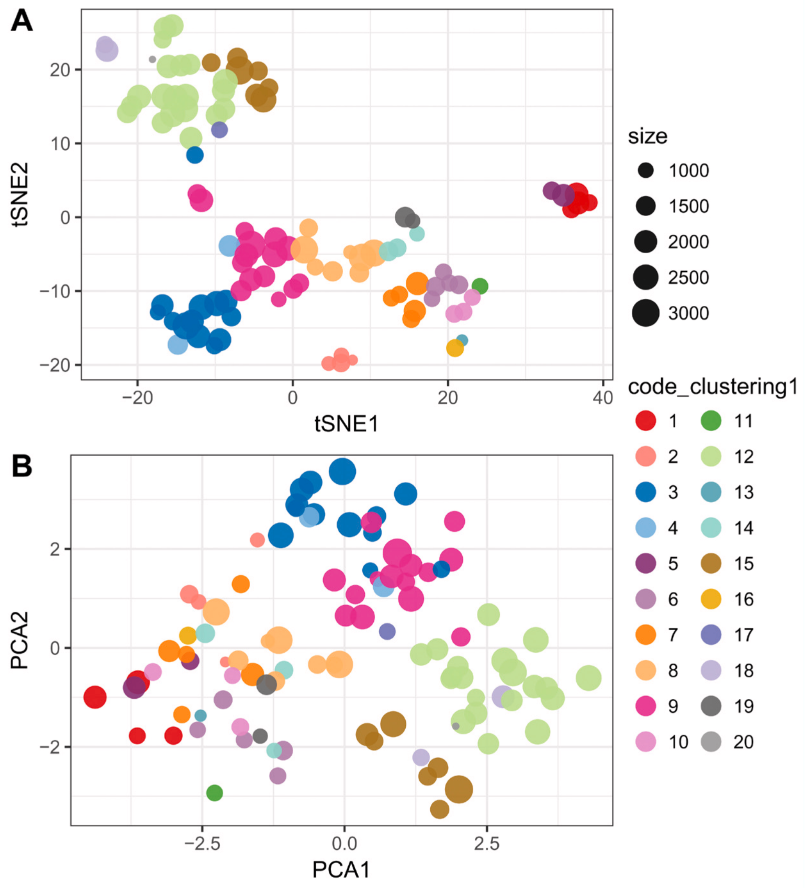


Figure 13. The 100 SOM codes in the PBMC dataset colored according to the metaclustering with ConsensusClusterPlus into 20 cell populations presented after the dimension reduction with (**A**) t-SNE and (**B**) PCA. The SOM codes represent characteristics of the 100 (by default) clusters generated in the first step of the FlowSOM pipeline. The size of the points corresponds to the number of cells that were assigned to a given code.

```
dr$sample_id <- sample_ids[tsne_inds]
mm <- match(dr$sample_id, md$sample_id)
dr$condition <- md$condition[mm]
dr$cell_clustering1 <- factor(cell_clustering1[tsne_inds], levels = 1:nmc)

## Plot t-SNE colored by clusters
ggp <- ggplot(dr,  aes(x = tSNE1, y = tSNE2, color = cell_clustering1)) +
  geom_point(size = 0.8) +
  theme_bw() +
  scale_color_manual(values = color_clusters) +
  guides(color = guide_legend(override.aes = list(size = 4), ncol = 2))
ggp

## Facet per sample
ggp + facet_wrap(~ sample_id)

## Facet per condition
ggp + facet_wrap(~ condition)
```

The SOM codes represent characteristics of the 100 (by default) clusters generated in the first step of the *FlowSOM* pipeline. Their visualization can also be helpful in understanding the cell population structure and determining the number of clusters. Ultimately, the metaclustering step uses the codes and not the original cells. We treat the codes as new representative cells and apply the t-SNE dimension reduction to visualize them in 2D (see Figure 13). The size of the points corresponds to the number of cells that were assigned to a given code. The points are colored according to the results of metaclustering. Since we have only 100 data points, the t-SNE analysis is fast.

As there are multiple ways to mathematically define similarity in high dimensional space, it is always good practice visualizing projections from other methods to see how consistent the observed patterns are. For instance, we also represent the *FlowSOM* codes via the first two principal components (see Figure 13).

```
## Get code sizes; sometimes not all the codes have mapped cells so they will have size 0
code_sizes <- table(factor(som$map$mapping[, 1], levels = 1:nrow(codes)))
code_sizes <- as.numeric(code_sizes)

## Run t-SNE on codes
set.seed(1234)
tsne_out <- Rtsne(codes, perplexity = 5, pca = FALSE)
## Run PCA on codes
pca_out <- prcomp(codes, center = TRUE, scale. = FALSE)

codes_dr <- data.frame(tSNE1 = tsne_out$Y[, 1], tSNE2 = tsne_out$Y[, 2],
  PCA1 = pca_out$x[, 1], PCA2 = pca_out$x[, 2])
codes_dr$code_clustering1 <- factor(code_clustering1)
codes_dr$size <- code_sizes

## Plot t-SNE on codes
gg_tsne_codes <- ggplot(codes_dr,  aes(x = tSNE1, y = tSNE2,
  color = code_clustering1, size = size)) +
  geom_point(alpha = 0.9) +
  theme_bw() +
  scale_color_manual(values = color_clusters) +
  guides(color = guide_legend(override.aes = list(size = 4), ncol = 2))

## Plot PCA on codes
gg_pca_codes <- ggplot(codes_dr,  aes(x = PCA1, y = PCA2,
  color = code_clustering1, size = size)) +
  geom_point(alpha = 0.9) +
  theme_bw() +
  scale_color_manual(values = color_clusters) +
  guides(color = guide_legend(override.aes = list(size = 4), ncol = 2)) +
  theme(legend.position = "right", legend.box = "vertical")

library(cowplot)

legend <- get_legend(gg_tsne_codes)
ggdraw() +
  draw_plot(gg_tsne_codes + theme(legend.position = "none"), 0, .5, .7, .5) +
  draw_plot(gg_pca_codes + theme(legend.position = "none"), 0, 0, .7, .5) +
  draw_plot(legend, .7, .0, .2, 1) +
  draw_plot_label(c("A", "B", ""), c(0, 0, .7), c(1, .5, 1), size = 15)
```

Using heatmaps, we can also visualize median marker expression in the 100 SOM codes as in Figure 14. Of note, the clustering presented with the dendrogram does not completely agree with the clustering depicted by the 20 colors because the first one is based on the hierarchical clustering with average linkage and Euclidean distance, while the second one results from the consensus clustering.

```
plot_clustering_heatmap_wrapper2(expr = expr, expr01 = expr01,
  lineage_markers = lineage_markers, functional_markers = "pS6",
  sample_ids = sample_ids, cell_clustering = cell_clustering_som,
  color_clusters = rep(color_clusters, length.out = 100),
  cluster_merging = data.frame(original_cluster = 1:100,
    new_cluster = code_clustering1), plot_cluster_annotation = FALSE)
```

Cluster merging and annotation

In our experience, manual merging of clusters leads to slightly different results compared to an algorithm with a specified number of clusters. In order to detect somewhat rare populations, some level of over-clustering is necessary so that the more subtle populations become separated from the main populations. In addition, merging can always follow an over-clustering step, but splitting of existing clusters is not generally feasible. In our setup, over-clustering is also useful when the interest is identifying the “natural” number of clusters present in the data. In addition to the t-SNE plots, one could investigate the delta area plot from the *ConsensusClusterPlus* package and the hierarchical clustering dendrogram of the over-clustered subpopulations, as shown in Figure 16 and Figure 18, respectively.

In our example, we expect around 6 specific cell types, and we have performed *FlowSOM* clustering into 20 groups as a reasonable over-estimate. After analyzing the heatmaps (Figure 6) and t-SNE plots (Figure 10), we can clearly see that stratification of the data into 20 clusters may be too strong. In the t-SNE map, many clusters are placed very close to each other, indicating that they could be merged together. The same can be deduced from the heatmaps, highlighting that marker expression patterns for some neighboring clusters are very similar. Cluster merging and annotating is somewhat manual, based partially on visual inspection of t-SNE plots and heatmaps and thus, benefits from expert knowledge of the cell types.

Manual cluster merging and annotating based on heatmaps. In our experience, the main reference for manual merging of clusters is the heatmap of marker characteristics across metaclusters (e.g. Figure 6), with dendrograms showing the hierarchy of similarities. Such plots aggregate information over many cells and thus show average marker expression for each cluster. Together with dimensionality reduction, these plots give good insight into the relationships between clusters and the marker levels within each cluster. Given expert knowledge of the cell types and markers, it is then left to the researcher to decide how exactly to merge clusters (e.g. with higher weight given to some markers).

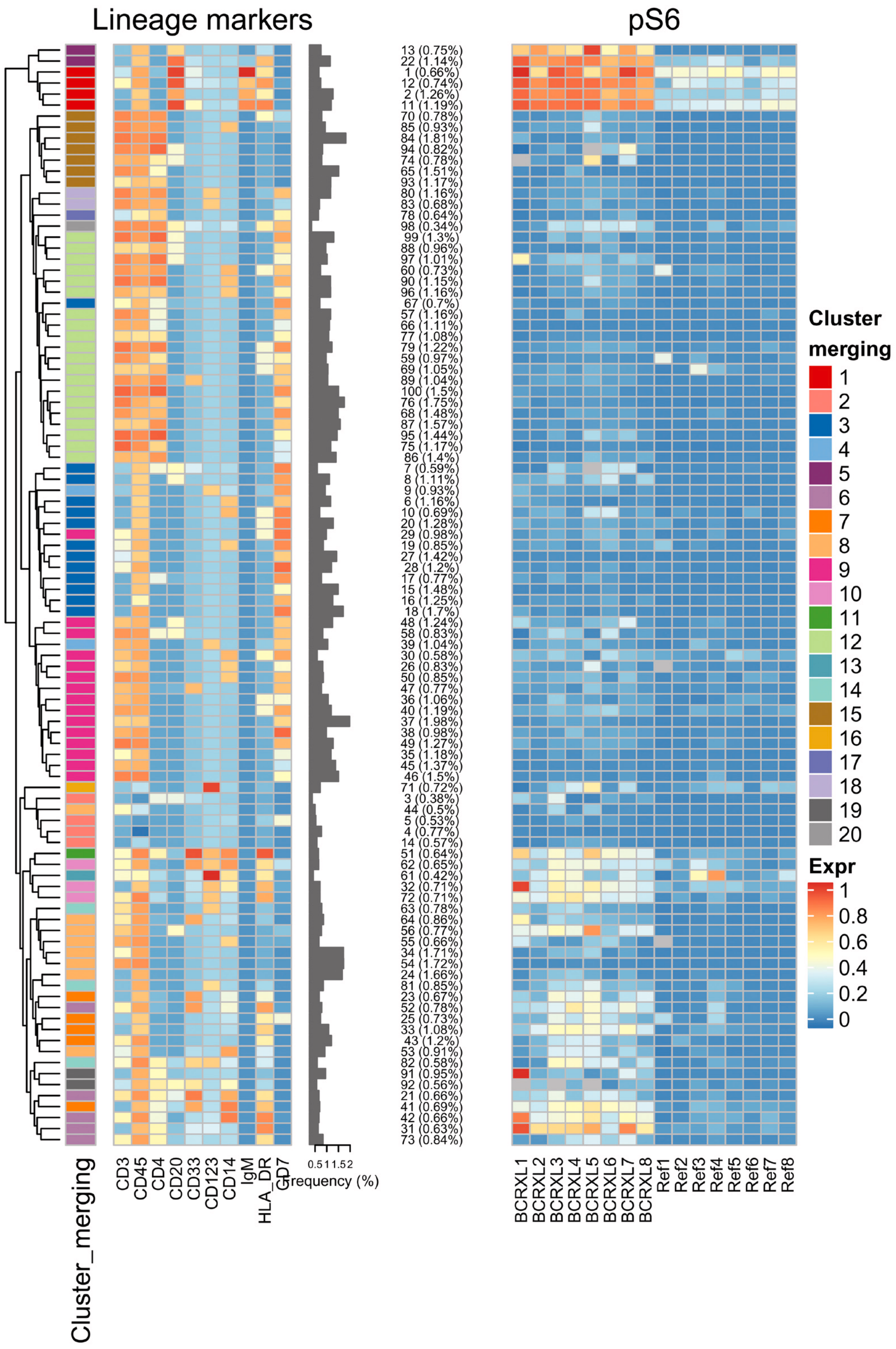


Figure 14. Heatmap of the median marker intensities of the 10 lineage markers (left panel) and one signaling marker pS6 (right panel) across the 100 SOM codes in the PBMC dataset. The color in the heatmap represents the median of the arcsinh, 0-1 transformed marker expression calculated over cells from all the samples, for the lineage markers, and over cells in each sample individually, for the signaling marker. The heat varies from blue for lower expression to red for higher expression. The dendrogram on the left represents the hierarchical similarity between the 100 codes (metric: Euclidean distance; linkage: average). The annotation bar on the left is colored according to the code metaclustering with ConsensusClusterPlus into 20 cell populations. The relative size of the codes is presented with the barplot along the rows and in the brackets next to the cluster numbers.

The dendrogram highlights the similarity between the metaclusters and can be used explicitly for the merging. However, there are reasons why we would not always follow the dendrogram exactly. In general, when it comes to clustering, blindly following the hierarchy of codes will lead to identification of populations of similar cells, but it does not necessarily mean that they are of biological interest. The distances between metaclusters are calculated across all the markers, and it may be that some markers carry higher weight for certain cell types. In addition, different linkage methods may lead to different hierarchy, especially when clusters are not fully distinct. Another aspect to consider in cluster merging is the cluster size, represented in the parentheses next to the cluster label in our plots. If the cluster size is very small, but the cluster seems relevant and distinct, one can keep it as separate. However, if it is small and different from the neighboring clustering only in a somewhat unimportant marker, it could be merged. And, if some of the metaclusters do not represent any specific cell types, they could be dropped out of the downstream analysis instead of being merged. However, in case an automated solution for cluster merging is truly needed, one could use the `cutree()` function applied to the dendrogram.

Based on the seed that was set, cluster merging of the 20 metaclusters is defined in the PBMC8_cluster_merging1.xlsx file on the Robinson Lab server with the IDs of the original clusters and new cluster names, and we save it as a `cluster_merging1` data frame. The expert has annotated 8 cell populations: CD8 T-cells, CD4 T-cells, B-cells IgM-, B-cells IgM+, NK cells, dendritic cells (DCs), monocytes and surface negative cells; monocytes could be further subdivided based on HLA-DR into high, medium and low subtypes.

```
cluster_merging1_filename <- "PBMC8_cluster_merging1.xlsx"
download.file(paste0(url, "/", cluster_merging1_filename),
  destfile = cluster_merging1_filename, mode = "wb")
cluster_merging1 <- read_excel(cluster_merging1_filename)
data.frame(cluster_merging1)
```

```
##      original_cluster  new_cluster
## 1                   1 B-cells IgM+
## 2                   2     surface-
## 3                   3     NK cells
## 4                   4  CD8 T-cells
## 5                   5 B-cells IgM-
## 6                   6    monocytes
## 7                   7    monocytes
## 8                   8  CD8 T-cells
## 9                   9  CD8 T-cells
## 10                 10    monocytes
## 11                 11    monocytes
## 12                 12  CD4 T-cells
## 13                 13           DC
## 14                 14  CD8 T-cells
## 15                 15  CD4 T-cells
## 16                 16           DC
## 17                 17  CD4 T-cells
## 18                 18  CD4 T-cells
## 19                 19  CD4 T-cells
## 20                 20  CD4 T-cells
```

```
## Convert to factor with merged clusters in desired order
levels_clusters_merged <- c("B-cells IgM+", "B-cells IgM-", "CD4 T-cells",
  "CD8 T-cells", "DC", "NK cells", "monocytes", "surface-")
cluster_merging1$new_cluster <- factor(cluster_merging1$new_cluster,
  levels = levels_clusters_merged)
## New clustering1m
mm <- match(cell_clustering1, cluster_merging1$original_cluster)
cell_clustering1m <- cluster_merging1$new_cluster[mm]
```

```
mm <- match(code_clustering1, cluster_merging1$original_cluster)
code_clustering1m <- cluster_merging1$new_cluster[mm]
```

We update the t-SNE plot with the new annotated cell populations, Figure 15.

```
dr$cell_clustering1m <- cell_clustering1m[tsne_inds]
ggplot(dr,  aes(x = tSNE1, y = tSNE2, color = cell_clustering1m)) +
  geom_point(size = 0.8) +
  theme_bw() +
  scale_color_manual(values = color_clusters) +
  guides(color = guide_legend(override.aes = list(size = 4)))
```

One of the useful representations of merging is a heatmap of median marker expression in each of the original clusters, which are labeled according to the proposed merging, Figure 16.

```
plot_clustering_heatmap_wrapper(expr = expr[, lineage_markers_ord],
  expr01 = expr01[, lineage_markers_ord], cell_clustering = cell_clustering1,
  color_clusters = color_clusters, cluster_merging = cluster_merging1)
```

To get a final summary of the annotated cell types, one can plot a heatmap of median marker expression, calculated based on the cells in each of the annotated populations, Figure 17.

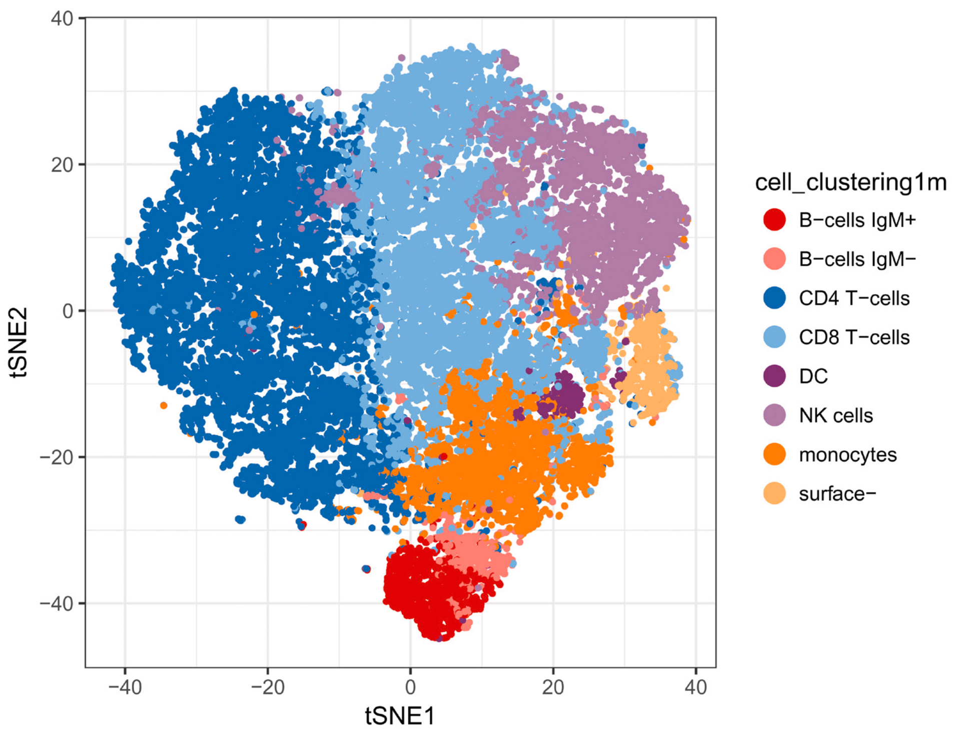


Figure 15. t-SNE plot for the PBMC dataset, where cells are colored according to the manual merging of the 20 cell populations, obtained with FlowSOM, into 8 PBMC populations. As in Figure 10, t-SNE analysis uses the arcsinh-transformed expression of the 10 lineage markers in 2000 randomly selected cells from each of the 16 samples.

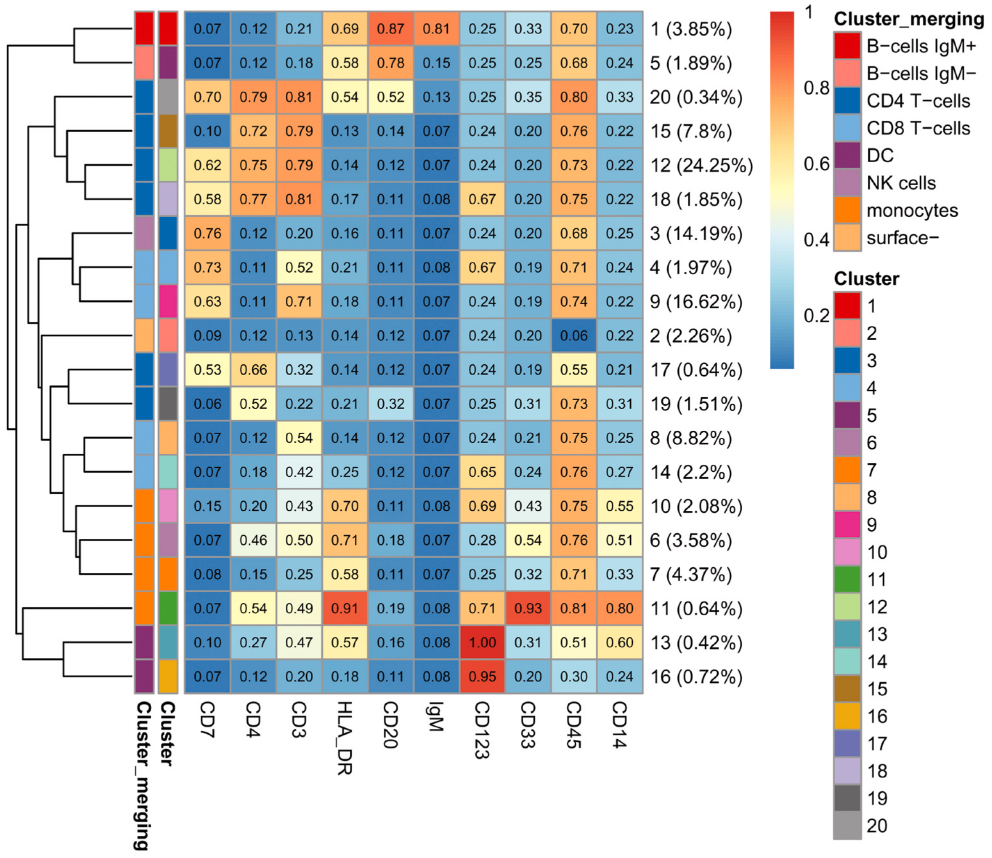


Figure 16. Heatmap as in Figure 6, where the additional color bar on the left indicates how the 20 metaclusters, obtained with FlowSOM, are merged into the 8 PBMC populations.

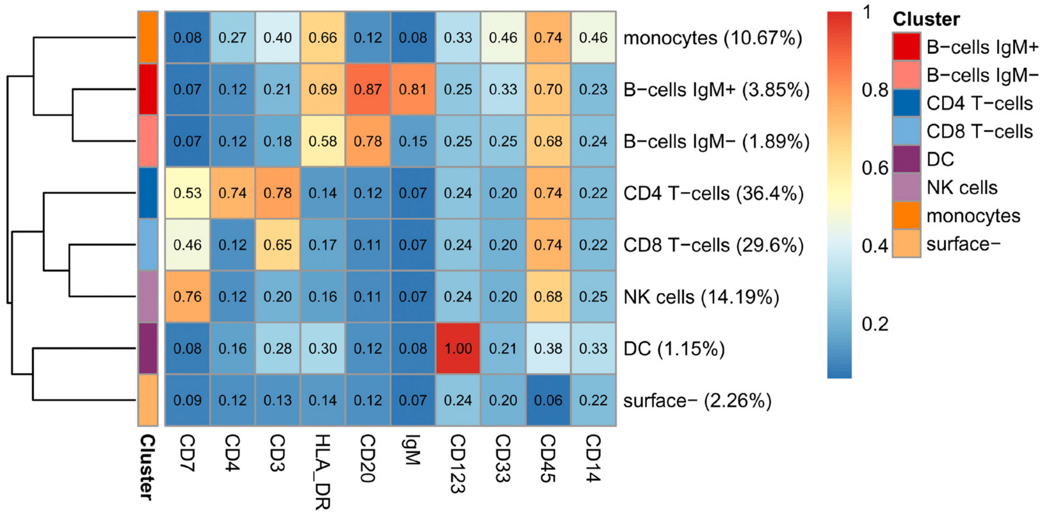


Figure 17. Heatmap of the median marker intensities of the 10 lineage markers in the 8 PBMC cell populations obtained by manual merging of the 20 metaclusters generated by FlowSOM. As in Figure 6, the heat represents the median of arcsinh and 0-1 transformed marker expression calculated over cells from all the samples. The dendrogram on the left represents the hierarchical similarity between the 8 populations calculated using Euclidean distance and average linkage. Values in the brackets indicate the relative size of each of the cell populations across all the samples.

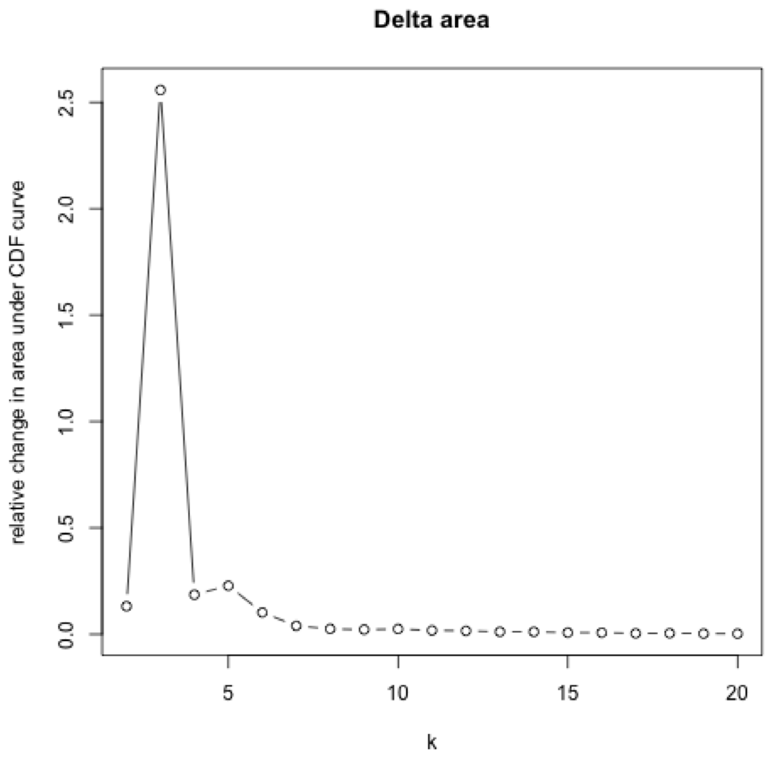


Figure 18. The delta area plot generated in the metaclustering step by the ConsensusClusterPlus function. The delta area score (y-axis) indicates the relative increase in cluster stability obtained when clustering the 100 SOM codes generated by FlowSOM into k groups (x-axis).

```
plot_clustering_heatmap_wrapper(expr = expr[, lineage_markers_ord],
  expr01 = expr01[, lineage_markers_ord], cell_clustering = cell_clustering1m,
  color_clusters = color_clusters)
```

Reducing the number of clusters in ConsensusClusterPlus. The *ConsensusClusterPlus* package provides visualizations that can help to understand the metaclustering process and the characteristics of the analyzed data. For example, the delta area plot (see Figure 18) highlights the amount of extra cluster stability gained when clustering into k groups as compared to k-1 groups (from k=2 to k=20). It can be expected that high stability of clusters can be reached when clustering into the number of groups that best fits the data. Thus, using the delta area plot could help finding the "natural" number of clusters present in the data, which would correspond to the value of k where there is no appreciable increase in stability. This strategy can be referred as the "elbow criterion". For more details about the meaning of this plot, the user can refer to the original description of the consensus clustering method (Monti *et al.*, 2003).

The elbow criterion is quite subjective since the "appreciable" increase is not defined exactly. For example, in the delta plot below, we could say that the last point before plateau is for k=6, or for k=5, or for k=3, depending on our perception of sufficient decrease of the delta area score. Moreover, the exact point where a plateau is reached may vary for runs with different random seeds, the drop may not always be so sharp and and the function is not guaranteed to be decreasing. It is advisable to investigate more of those plots and the resulting t-SNE and heatmaps before drawing any conclusions about the final number of "natural" clusters.

Manual merging of up to 20 clusters can be laborious. To simplify this task, one could reduce the strength of over-clustering and allow the metaclustering method to do a part of the merging, which then can be completed manually. Analyzing the delta plot from the right side, we can see how much we can reduce the strength of over-clustering while still obtaining stable clusters. In parallel, one should check the heatmaps to see whether the less stringent stratification is able to capture cell populations of interest.

As an example, we chose to reduce the strength of metaclustering to 12 groups.

```
## Get cluster ids for each cell
nmc2 <- 12
code_clustering2 <- mc[[nmc2]]$consensusClass
cell_clustering2 <- code_clustering2[som$map$mapping[, 1]]
```

In the t-SNE plot (see Figure 19), we can see that many small clusters obtained when stratifying data into 20 groups are now merged together, which should simplify the new cluster annotation.

```
dr$cell_clustering2 <- factor(cell_clustering2[tsne_inds], levels = 1:nmc2)
ggplot(dr,  aes(x = tSNE1, y = tSNE2, color = cell_clustering2)) +
  geom_point(size = 0.8) +
  theme_bw() +
  scale_color_manual(values = color_clusters) +
  guides(color = guide_legend(override.aes = list(size = 4), ncol = 2))

plot_clustering_heatmap_wrapper(expr = expr[, lineage_markers_ord],
  expr01 = expr01[, lineage_markers_ord], cell_clustering = cell_clustering2,
  color_clusters = color_clusters)
```

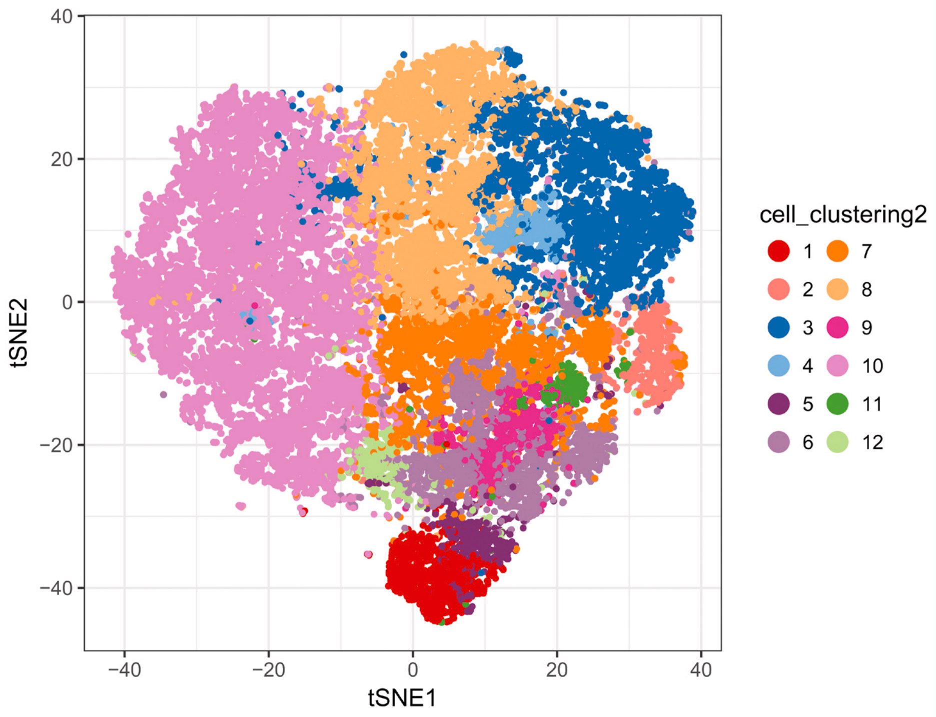


Figure 19. t-SNE plot for the PBMC dataset, where cells are colored according to the 12 cell populations obtained with FlowSOM after the metaclustering step with ConsensusClusterPlus. As in Figure 10, t-SNE analysis uses the arcsinh-transformed expression of the 10 lineage markers in 2000 randomly selected cells from each of the 16 samples.

Over-clustering into as few as 12 groups still allows us to identify the same 8 cell populations as when merging 20 clusters (see Figure 20–Figure 23), but it is simpler to define since fewer profiles need to be manually scanned. The expert-based merging is saved in the PBMC8_cluster_merging2.xlsx file on the Robinson Lab server.

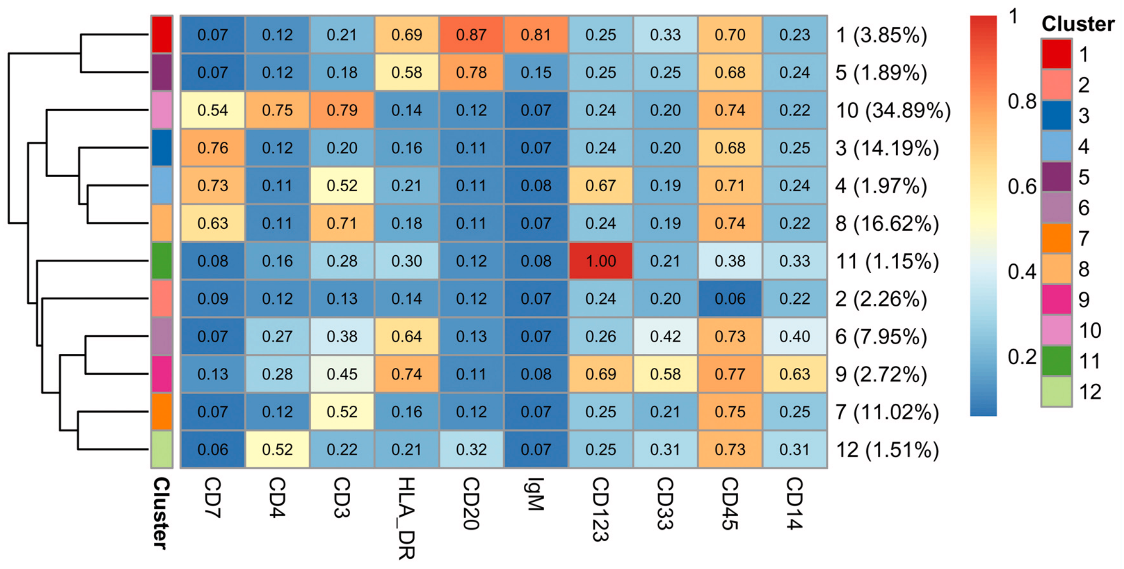


Figure 20. Heatmap of the median marker intensities of the 10 lineage markers in the 12 metaclusters obtained with FlowSOM after the metaclustering step with ConsensusClusterPlus (PBMC data). As in Figure 6, the heat represents the median of arcsinh and 0-1 transformed marker expression calculated over cells from all the samples. The dendrogram on the left represents the hierarchical similarity between the 12 clusters calculated using Euclidean distance and average linkage. Values in the brackets indicate the relative size of each cluster across all the samples.

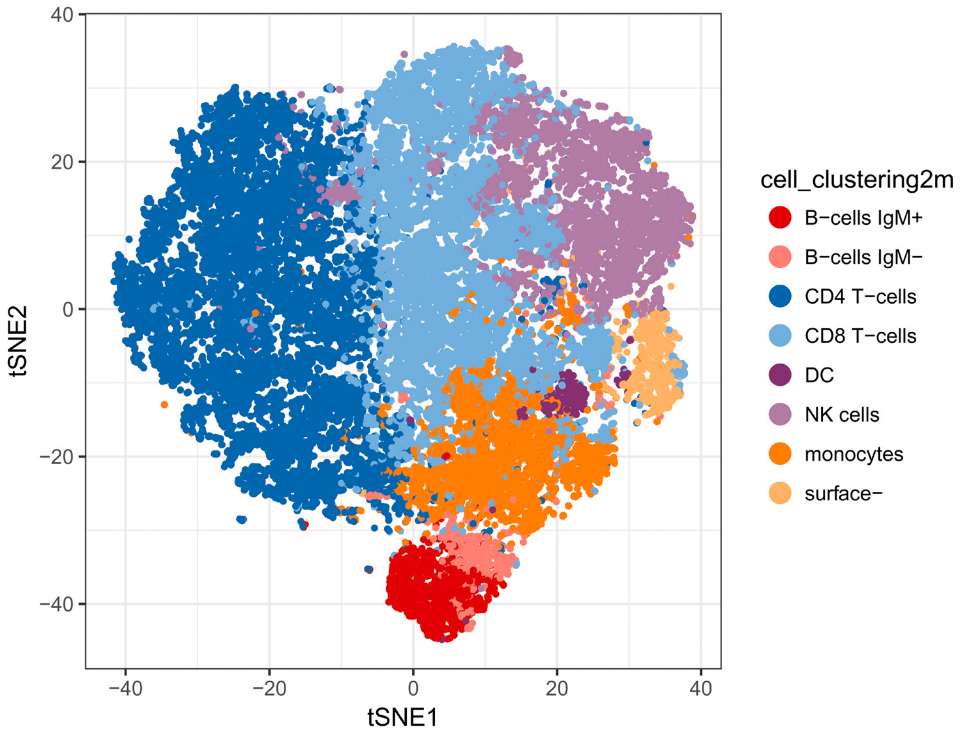


Figure 21. t-SNE plot for the PBMC dataset, where cells are colored according to the manual merging of the 12 metaclusters, obtained with FlowSOM, into 8 PBMC populations. As in Figure 10, t-SNE analysis uses the arcsinh-transformed expression of the 10 lineage markers in 2000 randomly selected cells from each of the 16 samples.

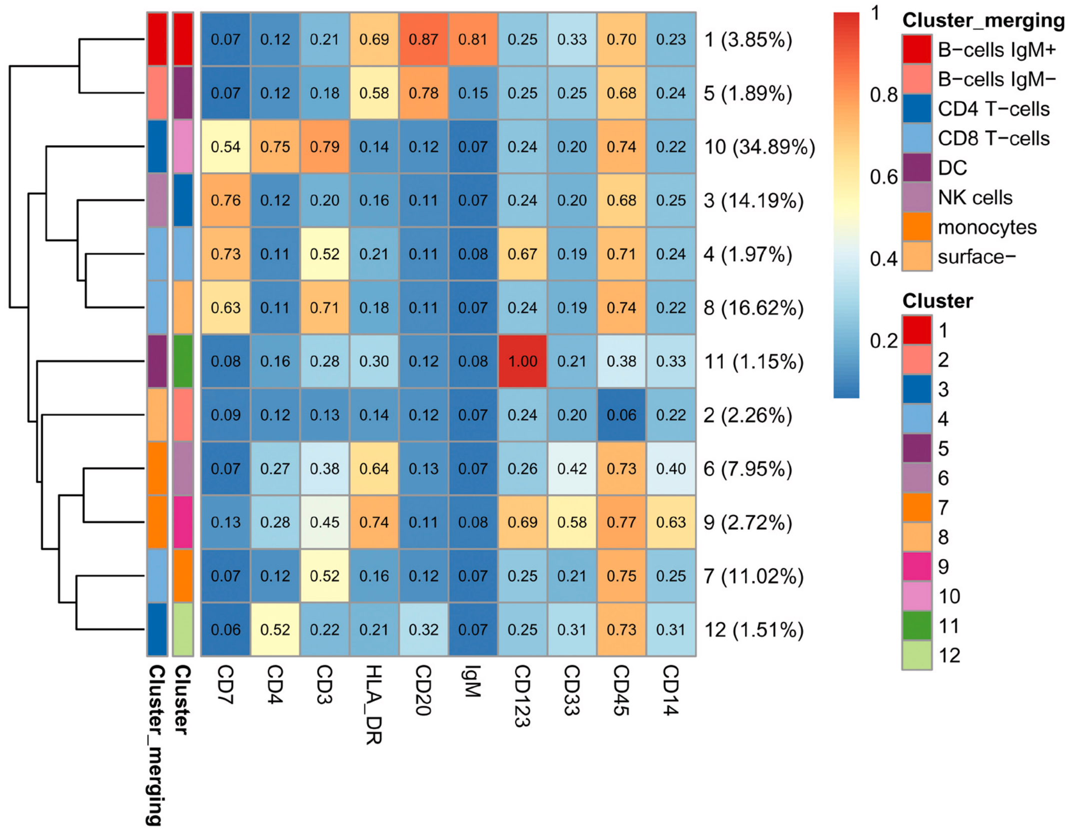


Figure 22. Heatmap as in Figure 20, where the additional color bar on the left indicates how the 12 metaclusters, obtained with FlowSOM, are merged into the 8 PBMC populations.

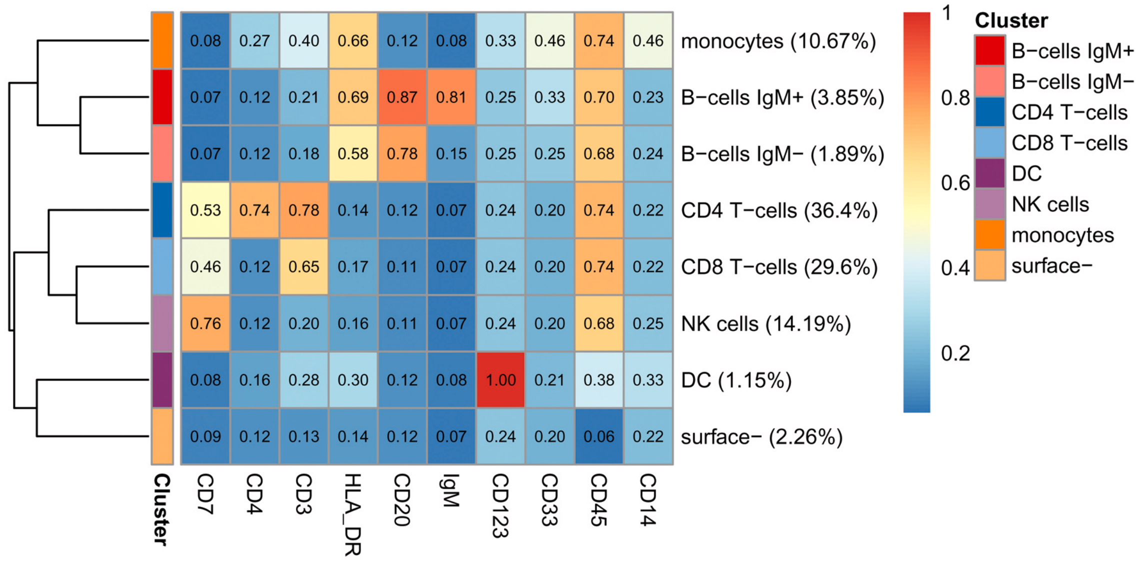


Figure 23. Heatmap of the median marker intensities of the 10 lineage markers in the 8 PBMC cell populations obtained by manual merging of the 12 metaclusters generated by FlowSOM. The heat represents the median of arcsinh and 0-1 transformed marker expression calculated over cells from all the samples. The dendrogram on the left represents the hierarchical similarity between the 8 populations calculated using Euclidean distance and average linkage. Values in the brackets indicate the relative size of each of the cell populations across all the samples.

```
cluster_merging2_filename <- "PBMC8_cluster_merging2.xlsx"
download.file(paste0(url, "/", cluster_merging2_filename),
  destfile = cluster_merging2_filename, mode = "wb")
cluster_merging2 <- read_excel(cluster_merging2_filename)
data.frame(cluster_merging2)

##    original_cluster  new_cluster
## 1                 1 B-cells IgM+
## 2                 2     surface-
## 3                 3     NK cells
## 4                 4  CD8 T-cells
## 5                 5 B-cells IgM-
## 6                 6    monocytes
## 7                 7  CD8 T-cells
## 8                 8  CD8 T-cells
## 9                 9    monocytes
## 10               10  CD4 T-cells
## 11               11           DC
## 12               12  CD4 T-cells

## Convert to factor with merged clusters in correct order
cluster_merging2$new_cluster <- factor(cluster_merging2$new_cluster,
  levels = levels_clusters_merged)
## New clustering2m
mm <- match(cell_clustering2, cluster_merging2$original_cluster)
cell_clustering2m <- cluster_merging2$new_cluster[mm]

dr$cell_clustering2m <- cell_clustering2m[tsne_inds]
gg_tsne_cl2m <- ggplot(dr, aes(x = tSNE1, y = tSNE2, color = cell_clustering2m)) +
  geom_point(size = 0.8) +
  theme_bw() +
  scale_color_manual(values = color_clusters) +
  guides(color = guide_legend(override.aes = list(size = 4)))
gg_tsne_cl2m

plot_clustering_heatmap_wrapper(expr = expr[, lineage_markers_ord],
  expr01 = expr01[, lineage_markers_ord], cell_clustering = cell_clustering2,
  color_clusters = color_clusters, cluster_merging = cluster_merging2)

plot_clustering_heatmap_wrapper(expr = expr[, lineage_markers_ord],
  expr01 = expr01[, lineage_markers_ord], cell_clustering = cell_clustering2m,
  color_clusters = color_clusters)
```

Comparison of automated and manual merging. The manual merging of 20 (or 12) clusters by an expert resulted in identification of 8 cell populations. To highlight the impact of manual merging versus algorithm-defined subpopulations, we compare to the results of an automated cluster merging that is set to stratify the data also into 8 clusters. We extract the result from the `ConsensusClusterPlus` output. Out of interest, we can see which clusters are split by tabulating the cell labels.

```
## Get cluster ids for each cell
nmc3 <- 8
code_clustering3 <- mc[[nmc3]]$consensusClass
cell_clustering3 <- code_clustering3[som$map$mapping[, 1]]
```

```
# tabular comparison of cell_clustering3 and cell_clustering2m
table(algorithm=cell_clustering3, manual=cell_clustering2m)
```

```
##          manual
## algorithm B-cells IgM+ B-cells IgM- CD4 T-cells CD8 T-cells    DC NK cells
##         1         6651            0           0           0     0        0
##         2            0            0           0           0     0        0
##         3            0            0           0       32112     0    24518
##         4            0         3265           0           0     0        0
##         5            0            0           0           0     0        0
##         6            0            0        2603       19038     0        0
##         7            0            0       60287           0     0        0
##         8            0            0           0           0  1980        0
##          manual
## algorithm monocytes surface-
##         1         0        0
##         2         0     3901
##         3         0        0
##         4         0        0
##         5     18436        0
##         6         0        0
##         7         0        0
##         8         0        0
```

In the t-SNE map (see Figure 24), we can see that part of the new cell populations (cluster 7, 1 and 4, 2, 5 and 8) overlap substantially with populations obtained by the means of manual merging (CD4 T-cells, B-cells, surface-, monocytes and DC). However, cells that belong to clusters 3 and 6 are subdivided in a different manner according to the manual merging. Cluster 3 consists of CD8 T-cells and NK cells, and the latter cannot be identified anymore based on the heatmap corresponding to clustering into 8 groups (see Figure 25).

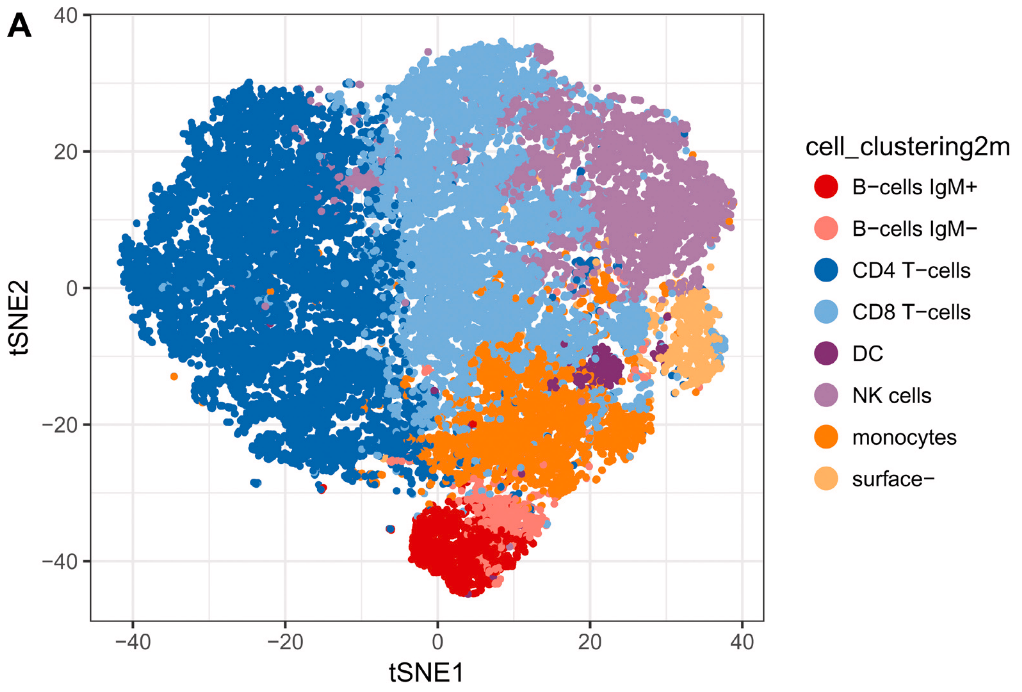
```
dr$cell_clustering3 <- factor(cell_clustering3[tsne_inds], levels = 1:nmc3)
gg_tsne_cl3 <- ggplot(dr, aes(x = tSNE1, y = tSNE2, color = cell_clustering3)) +
  geom_point(size = 0.8) +
  theme_bw() +
  scale_color_manual(values = color_clusters) +
  guides(color = guide_legend(override.aes = list(size = 4)))

plot_grid(gg_tsne_cl2m, gg_tsne_cl3, ncol = 1, labels = c("A", "B"))

plot_clustering_heatmap_wrapper(expr = expr[, lineage_markers_ord],
  expr01 = expr01[, lineage_markers_ord], cell_clustering = cell_clustering3,
  color_clusters = color_clusters)
```

The example above highlights the difference between automatic clustering and manual merging of algorithm-generated clusters in the search for biologically meaningful cell populations. Automated and manual merging may give different weight to marker importance and thus result in different populations being detected. However, in our view, the manual merging done here in a reproducible fashion results in a more biologically meaningful cell stratification.

Differential analysis

For the dataset used in this workflow (Bodenmiller *et al.*, 2012; Bruggner *et al.*, 2014), we perform three types of analyses that aim to identify subsets of PBMCs and signaling markers that respond to BCR/FcR-XL stimulation, by comparing stimulated samples to unstimulated samples. We first describe the differential abundance of the defined cell populations, followed by differential analysis of marker expression within each cluster. Finally, differential analysis of the overall aggregated marker expression could also be of interest.

The PBMC subset analyzed in this workflow originates from a paired experiment, where samples from 8 patients were treated with 12 different stimulation conditions for 30 minutes, together with unstimulated reference



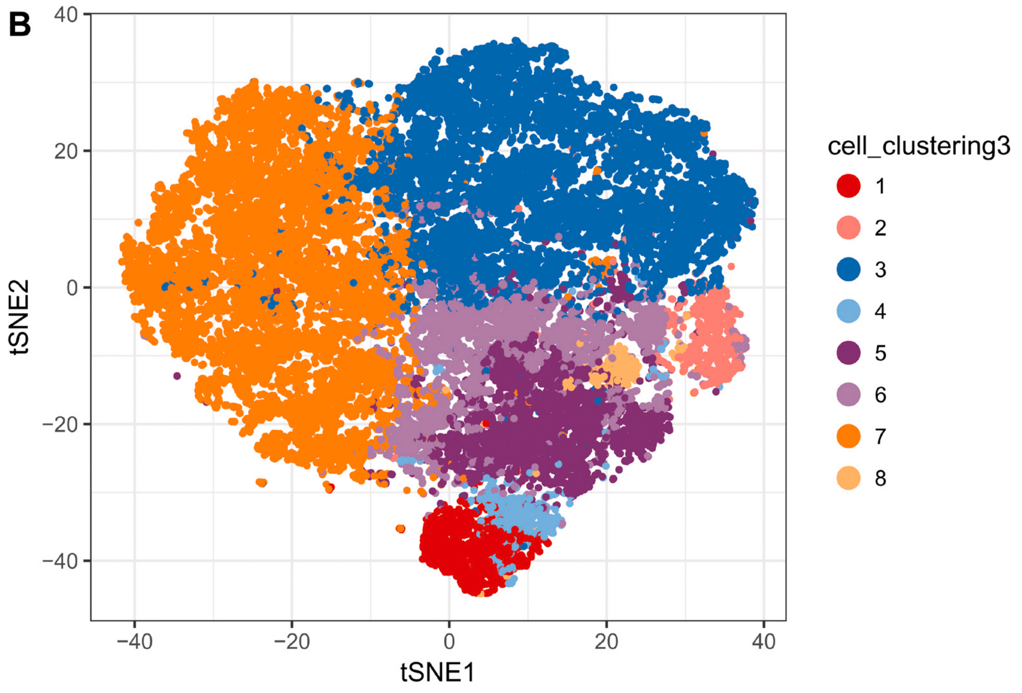
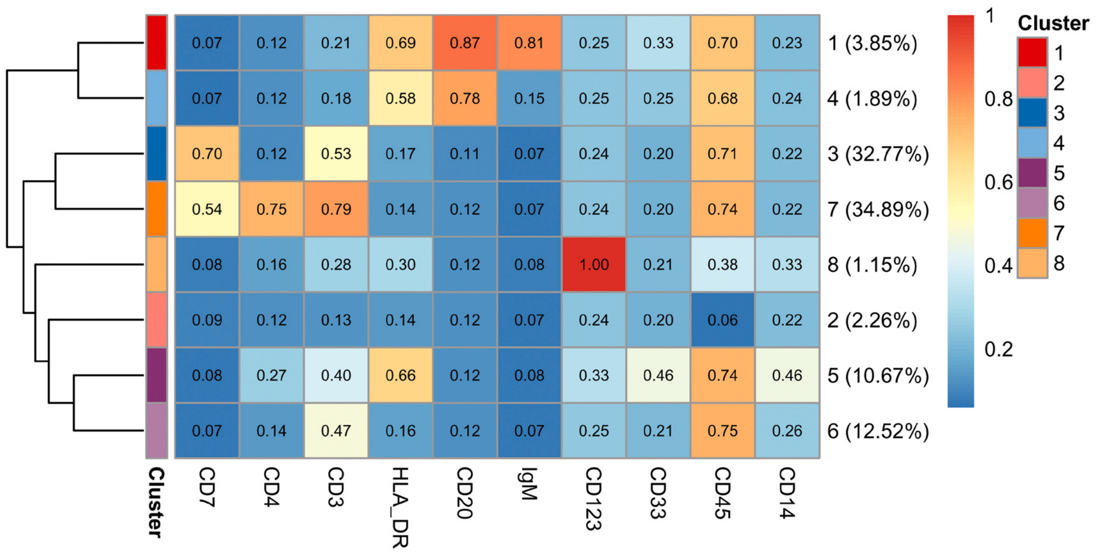


Figure 24. t-SNE plots with cells colored according to (**A**) the expert merging of 12 metaclusters obtained with FlowSOM into 8 PBMC populations; and (**B**) the 8 automatically detected with FlowSOM metaclusters.



Cluster	CD7	CD4	CD3	HLA_DR	CD20	IgM	CD123	CD33	CD45	CD14	
1	0.07	0.12	0.21	0.69	0.87	0.81	0.25	0.33	0.70	0.23	1 (3.85%)
4	0.07	0.12	0.18	0.58	0.78	0.15	0.25	0.25	0.68	0.24	4 (1.89%)
3	0.70	0.12	0.53	0.17	0.11	0.07	0.24	0.20	0.71	0.22	3 (32.77%)
7	0.54	0.75	0.79	0.14	0.12	0.07	0.24	0.20	0.74	0.22	7 (34.89%)
8	0.08	0.16	0.28	0.30	0.12	0.08	1.00	0.21	0.38	0.33	8 (1.15%)
2	0.09	0.12	0.13	0.14	0.12	0.07	0.24	0.20	0.06	0.22	2 (2.26%)
5	0.08	0.27	0.40	0.66	0.12	0.08	0.33	0.46	0.74	0.46	5 (10.67%)
6	0.07	0.14	0.47	0.16	0.12	0.07	0.25	0.21	0.75	0.26	6 (12.52%)

Figure 25. Heatmap of the median marker intensities of the 10 lineage markers in the 8 metaclusters obtained with FlowSOM after the metaclustering step with ConsensusClusterPlus (PBMC data). The heat represents the median of arcsinh and 0–1 transformed marker expression calculated over cells from all the samples. The dendrogram on the left represents the hierarchical similarity between the 8 clusters calculated using Euclidean distance and average linkage. Values in the brackets indicate the relative size of each cluster across all the samples.

samples (Bodenmiller *et al.*, 2012). This is a natural example where one would choose a mixed model to model the response (abundance or marker signal), and patients would be treated as a random effect. In this way, one can formally account for within-patient variability, observed to be quite strong in the MDS plot (see MDS plot section), and this should give a gain in power to detect differences between conditions.

We use the *stats* and *lme4* packages to fit the fixed and mixed models, respectively, and the *multcomp* package for hypothesis testing. In all differential analyses here, we want to test for differences between the reference (`Ref`) and BCR/FcR-XL treatment (`BCRXL`). The fixed model formula is straightforward: `~ condition`, where `condition` indicates the treatment group. The corresponding full model design matrix consists of the intercept and dummy variable indicating the treated samples. In the presence of batches, one can include them in the model by using a formula `~ condition + batch`, or if they affect the treatment, a formula with interactions `~ condition * batch`.

For testing, we use the general linear hypotheses function `glht`, which allows testing of arbitrary hypotheses using t-tests. The `linfct` parameter specifies the linear hypotheses to be tested. It should be a matrix where each row corresponds to one comparison (contrast), and the number of columns must be the same as in the design matrix. In our analysis, the contrast matrix indicates that the regression coefficient corresponding to `conditionBCRXL` is tested to be equal to zero; i.e. we test the null hypothesis that there is no effect of the BCR/FcR-XL treatment. The result of the test is a p-value, which indicates the probability of observing an as strong (or stronger) difference between the two conditions assuming the null hypothesis is true.

Testing is performed on each cluster and marker separately, resulting in 8 tests for differential abundance (one for each merged population), 14 tests for overall differential marker expression analysis and 8 × 14 tests for differential marker expression within-populations. Thus, to account for the multiple testing correction, we apply the Benjamini & Hochberg adjustment to each of them using an FDR cutoff of 5%.

```
library(lme4)
library(multcomp)
## Model formula without random effects
model.matrix( ~ condition, data = md)
```

```
##    (Intercept) conditionBCRXL
## 1            1              1
## 2            1              0
## 3            1              1
## 4            1              0
## 5            1              1
## 6            1              0
## 7            1              1
## 8            1              0
## 9            1              1
## 10           1              0
## 11           1              1
## 12           1              0
## 13           1              1
## 14           1              0
## 15           1              1
## 16           1              0
## attr(,"assign")
## [1] 0 1
## attr(,"contrasts")
## attr(,"contrasts")$condition
## [1] "contr.treatment"
```

```
## Create contrasts
contrast_names <- c("BCRXLvsRef")
k1 <- c(0, 1)
K <- matrix(k1, nrow = 1, byrow = TRUE, dimnames = list(contrast_names))
K
```

```
##              [,1] [,2]
## BCRXLvsRef      0    1
```

```
FDR_cutoff <- 0.05
```

Differential cell population abundance

Differential analysis of cell population abundance compares the proportions of cell types across experimental conditions and aims to highlight populations that are present at different ratios. First, we calculate two tables: one that contains cell counts for each sample and population and one with the corresponding proportions of cell types by sample. The proportions are used only for plotting, since the statistical modeling takes the cell counts by cluster and sample as input.

```
counts_table <- table(cell_clustering1m, sample_ids)
props_table <- t(t(counts_table) / colSums(counts_table)) * 100

counts <- as.data.frame.matrix(counts_table)
props <- as.data.frame.matrix(props_table)
```

For each sample, we plot its PBMC cell type composition represented with colored bars, where the size of a given stripe reflects the proportion of the corresponding cell type in a given sample (see Figure 26).

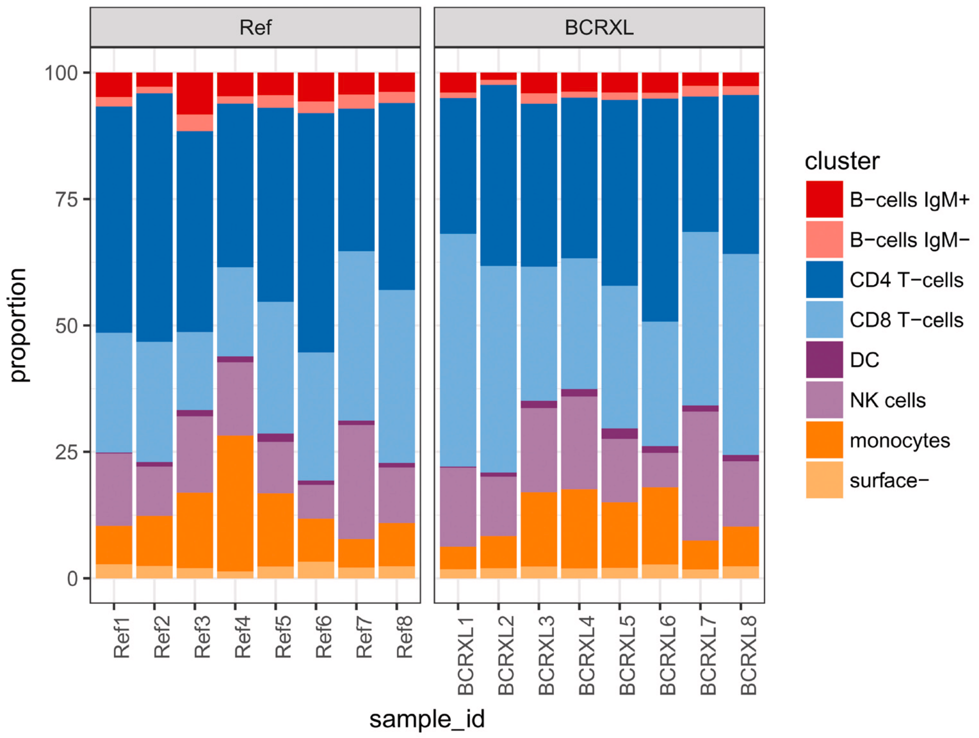


Figure 26. Relative abundance of the 8 PBMC populations in each sample (x-axis), in the PBMC dataset, represented with a barplot. The 8 cell populations are a result of manual merging of the 20 FlowSOM metaclusters.

```
ggdf <- melt(data.frame(cluster = rownames(props), props),
  id.vars = "cluster", value.name = "proportion", variable.name = "sample_id")
ggdf$cluster <- factor(ggdf$cluster, levels = levels_clusters_merged)
## Add condition info
mm <- match(ggdf$sample_id, md$sample_id)
ggdf$condition <- factor(md$condition[mm])

ggplot(ggdf, aes(x = sample_id, y = proportion, fill = cluster)) +
  geom_bar(stat = "identity") +
  facet_wrap(~ condition, scales = "free_x") +
  theme_bw() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  scale_fill_manual(values = color_clusters)
```

It may be quite hard to see the differences in cluster abundances in the plot above, especially for clusters with very low frequency. And, since boxplots cannot represent multimodal distributions, we show boxplots with jittered points of the sample-level cluster proportions overlaid (see Figure 27). The y-axes are scaled to the range of data plotted for each cluster, to better visualize the differences in frequency of lower abundance clusters. For this experiment, it may be interesting to additionally depict the patient information. We do this by plotting a different point shape for each patient. Indeed, we can see that often the direction of abundance changes between the two conditions are concordant among the patients.

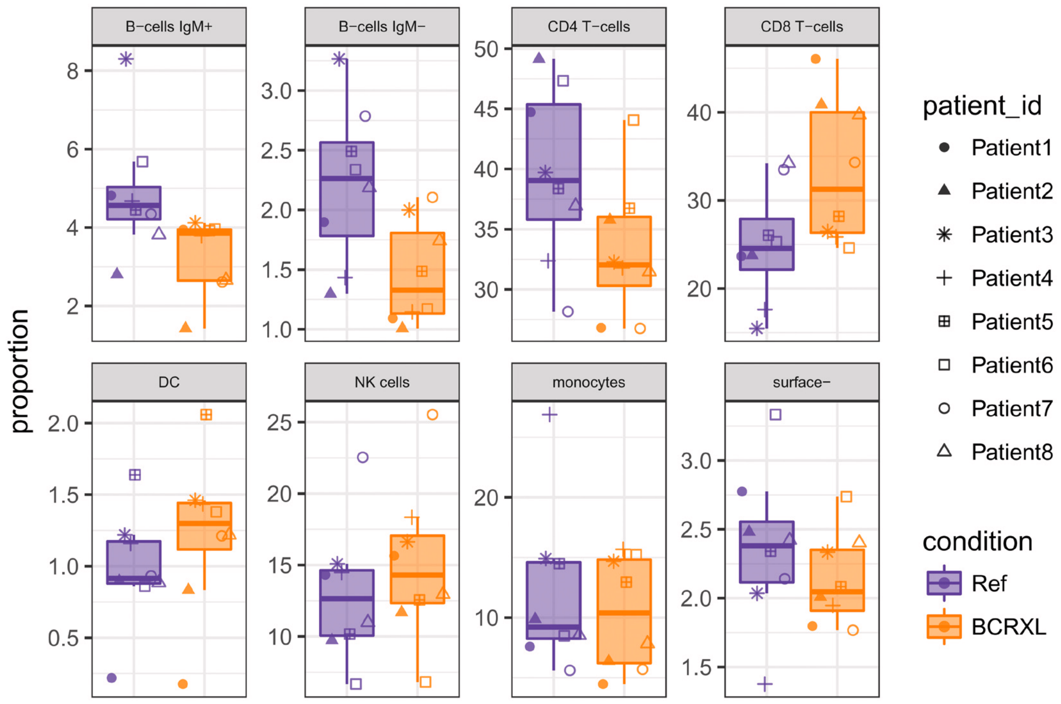


Figure 27. Relative abundance of the 8 PBMC populations in each sample, in the PBMC dataset, represented with boxplots. Values for the two conditions are indicated with different colors: violet for the unstimulated (Ref) and orange for the stimulated with BCR/FcR-XL (BCRXL) samples. Values for each patient are indicated with different shape. The 8 cell populations are a result of manual merging of the 20 FlowSOM metaclusters.

```
ggdf$patient_id <- factor(md$patient_id[mm])

ggplot(ggdf) +
  geom_boxplot(aes(x = condition, y = proportion, color = condition,
    fill = condition), position = position_dodge(), alpha = 0.5,
    outlier.color = NA) +
  geom_point(aes(x = condition, y = proportion, color = condition,
    shape = patient_id), alpha = 0.8, position = position_jitterdodge()) +
  facet_wrap(~ cluster, scales = "free", nrow = 2) +
  theme_bw() +
  theme(axis.text.x = element_blank(), axis.ticks.x = element_blank(),
    axis.title.x = element_blank(), strip.text = element_text(size = 6)) +
  scale_color_manual(values = color_conditions) +
  scale_fill_manual(values = color_conditions) +
  scale_shape_manual(values = c(16, 17, 8, 3, 12, 0, 1, 2))
```

As our goal is to compare proportions, one could take these values, transform them (e.g. logit) and use them as a dependent variable in a linear model. However, this approach does not take into account the uncertainty of proportion estimates, which is higher when ratios are calculated for samples with lower total cell counts. A distribution that naturally accounts for such uncertainty is the binomial distribution (i.e. logistic regression), which takes the cell counts as input (relative to the total for each sample). Nevertheless, as in the genomic data analysis, the pure logistic regression is not able to capture the overdispersion that is present in HDCyto data. A natural extension to model the extra variation is the generalized linear mixed model (GLMM), where the random effect is defined by the sample ID (Jia *et al.*, 2014; Zhao *et al.*, 2013). Additionally, in our example the patient pairing could be accounted in the model by incorporating a random intercept for each patient. Thus, we present two GLMMs. Both of them comprise a random effect defined by the sample ID to model the overdispersion in proportions. The second model includes a random effect defined by the patient ID to account for the experiment pairing.

In our model, the blocking variable is patient ID $i = 1, ..., n$, where $n = 8$. For each patient, there are n_i samples measured, and $j = 1, ..., n_i$ indicates the sample ID. Here, $n_i = 2$ for all i (one from reference and one from BCR/FcR-XL stimulated).

We assume that for a given cell population, the cell counts Y_{ij} follow a binomial distribution $Y_{ij} \sim Bin(m_{ij}, \pi_{ij})$, where m_{ij} is a total number of cells in a sample corresponding to patient i and condition j. The generalized linear mixed model with observation-level random effects ξ_{ij} is defined as follows:

$$E(Y_{ij} \mid \beta_0, \beta_1, \xi_{ij}) = logit^{-1}(\beta_0 + \beta_1 x_{ij} + \xi_{ij}),$$

where $\xi_{ij} \sim N(0, \sigma_\xi^2)$ and x_{ij} corresponds to the `conditionBCRXL` column in the design matrix and indicates whether a sample ij belongs to the reference ($x_{ij} = 0$) or treated condition ($x_{ij} = 1$). Since $E(Y_{ij} \mid \beta_0, \beta_1, \xi_{ij}) = \pi_{ij}$, the above formula can be written as follows:

$$logit(\pi_{ij}) = \beta_0 + \beta_1 x_{ij} + \xi_{ij}.$$

The generalized linear mixed model that furthermore accounts for the patient pairing incorporates additionally a random intercept for each patient i:

$$E(Y_{ij} \mid \beta_0, \beta_1, \gamma_i, \xi_{ij}) = logit^{-1}(\beta_0 + \beta_1 x_{ij} + \gamma_i + \xi_{ij}),$$

where $\gamma_i \sim N(0, \sigma_\gamma^2)$.

```
formula_glmer_binomial1 <- y/total ~ condition + (1|sample_id)
formula_glmer_binomial2 <- y/total ~ condition + (1|patient_id) + (1|sample_id)
```

The wrapper function below takes as input a data frame with cell counts (each row is a population, each column is a sample), the metadata table, and the formula, and performs the differential analysis specified with contrast `K` for each population separately, returning a table with non-adjusted and adjusted p-values.

```
differential_abundance_wrapper <- function(counts, md, formula, K){
  ## Fit the GLMM for each cluster separately
  ntot <- colSums(counts)
  fit_binomial <- lapply(1:nrow(counts), function(i){

    data_tmp <- data.frame(y = as.numeric(counts[i, md$sample_id]),
      total = ntot[md$sample_id], md)

    fit_tmp <- glmer(formula, weights = total, family = binomial,
      data = data_tmp)

    ## Fit contrasts one by one
    out <- apply(K, 1, function(k){
      contr_tmp <- glht(fit_tmp, linfct = matrix(k, 1))
      summ_tmp <- summary(contr_tmp)
      pval <- summ_tmp$test$pvalues
      return(pval)
    })
    return(out)
  })
  pvals <- do.call(rbind, fit_binomial)
  colnames(pvals) <- paste0("pval_", contrast_names)
  rownames(pvals) <- rownames(counts)
  ## Adjust the p-values
  adjp <- apply(pvals, 2, p.adjust, method = "BH")
  colnames(adjp) <- paste0("adjp_", contrast_names)
  return(list(pvals = pvals, adjp = adjp))
}
```

We fit both of the GLMMs specified above. We can see that accounting for the patient pairing increases the sensitivity to detect differentially abundant cell populations.

```
da_out1 <- differential_abundance_wrapper(counts, md = md,
  formula = formula_glmer_binomial1, K = K)
apply(da_out1$adjp < FDR_cutoff, 2, table)

##       adjp_BCRXLvsRef
## FALSE               5
## TRUE                3

da_out2 <- differential_abundance_wrapper(counts, md = md,
  formula = formula_glmer_binomial2, K = K)
apply(da_out2$adjp < FDR_cutoff, 2, table)

##       adjp_BCRXLvsRef
## FALSE               2
## TRUE                6
```

An output table containing the observed cell population proportions in each sample and p-values can be assembled (and optionally written to a file).

```
da_output2 <- data.frame(cluster = rownames(props), props,
  da_out2$pvals, da_out2$adjp, row.names = NULL)
print(head(da_output2), digits = 2)

##        cluster BCRXL1 BCRXL2 BCRXL3 BCRXL4 BCRXL5 BCRXL6 BCRXL7 BCRXL8
## 1 B-cells IgM+   3.95   1.43    4.1    3.8    3.9    4.0    2.6    2.7
## 2 B-cells IgM-   1.09   1.01    2.0    1.1    1.5    1.2    2.1    1.7
## 3  CD4 T-cells  26.81  35.78   32.3   31.8   36.7   44.1   26.8   31.5
## 4  CD8 T-cells  46.05  40.87   26.5   25.9   28.2   24.6   34.3   39.7
## 5           DC   0.18   0.83    1.5    1.4    2.1    1.4    1.2    1.2
## 6     NK cells  15.64  11.69   16.6   18.3   12.6    6.8   25.5   12.9
##    Ref1 Ref2 Ref3 Ref4 Ref5  Ref6  Ref7  Ref8 pval_BCRXLvsRef
## 1  4.82  2.8  8.3  4.7  4.4  5.68  4.34  3.82         3.5e-08
## 2  1.90  1.3  3.3  1.4  2.5  2.34  2.79  2.19         2.2e-11
## 3 44.72 49.1 39.7 32.4 38.4 47.33 28.16 36.94         1.9e-03
## 4 23.66 23.8 15.5 17.6 26.0 25.31 33.49 34.21         1.2e-03
## 5  0.22  0.9  1.2  1.2  1.6  0.86  0.93  0.89         7.1e-05
## 6 14.31  9.7 15.1 14.5 10.2  6.67 22.54 10.99         4.5e-13
##   adjp_BCRXLvsRef
## 1         9.2e-08
## 2         8.8e-11
## 3         2.5e-03
## 4         1.9e-03
## 5         1.4e-04
## 6         3.6e-12
```

We use a heatmap to report the differential cell populations (see Figure 28). Proportions are first scaled with the arcsine-square-root transformation (as an alternative to logit that does not return NAs when ratios are equal to

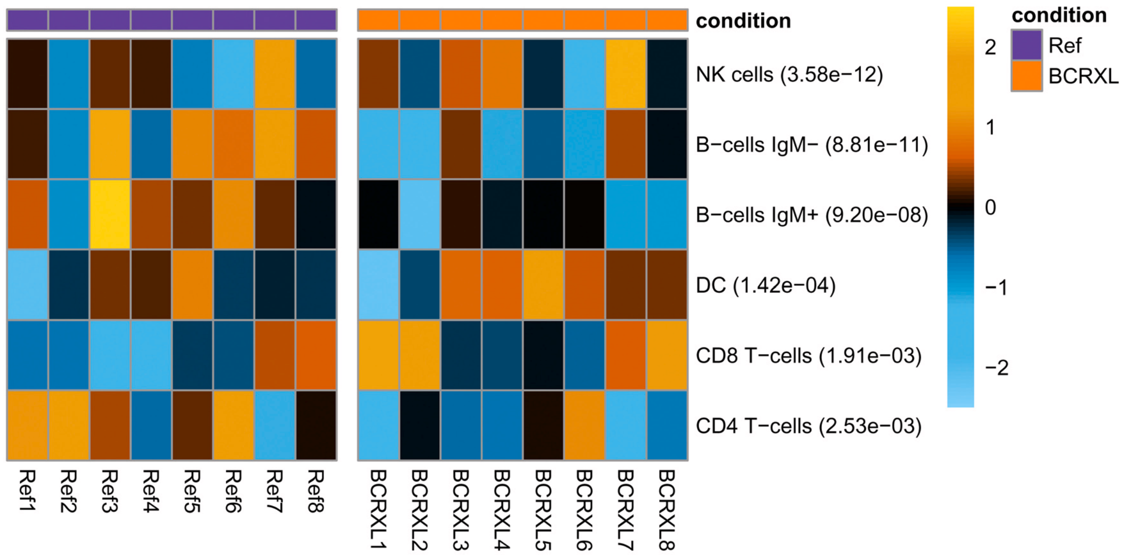


Figure 28. Normalized proportions of PBMC cell populations that are significantly differentially abundant between BCR/FcR-XL stimulated and unstimulated condition. The heat represents arcsine-square-root transformed cell frequencies that were subsequently normalized per cluster (rows) to mean of zero and standard deviation of one. The color of the heat varies from blue indicating relative under-representation to orange indicating relative over-representation. Bar at the top of the heatmap indicates the condition the samples (columns) belong to: violet for the unstimulated (Ref) and orange for the stimulated with BCR/FcR-XL (BCRXL) condition. Numbers in the brackets next to the cluster names indicate adjusted p-values. Shown are only the significant clusters for which adjusted p-values < 0.05. Clusters are sorted according to the significance so that a cluster on the top shows the most significant abundance changes between the two conditions.

zero or one). Then, Z-score normalization is applied to each population to better highlight the relative differences between compared conditions. We created two wrapper functions: `normalization_wrapper` performs the normalization of submitted expression `expr` to mean 0 and standard deviation 1, and `plot_differential_heatmap_wrapper` generates a heatmap of submitted expression `expr_norm`, where samples are grouped by `condition`, indicated with a color bar on top of the plot. Additionally, labels of clusters contain the adjusted p-values in parenthesis.

```
normalization_wrapper <- function(expr, th = 2.5){
  expr_norm <- apply(expr, 1, function(x){
    sdx <- sd(x, na.rm = TRUE)
    if(sdx == 0){
      x <- (x - mean(x, na.rm = TRUE))
    }else{
      x <- (x - mean(x, na.rm = TRUE)) / sdx
    }
    x[x > th] <- th
    x[x < -th] <- -th
    return(x)
  })
  expr_norm <- t(expr_norm)
}

plot_differential_heatmap_wrapper <- function(expr_norm, sign_adjp,
  condition, color_conditions, th = 2.5){
  ## Order samples by condition
  oo <- order(condition)
  condition <- condition[oo]
  expr_norm <- expr_norm[, oo, drop = FALSE]
  ## Create the row labels with adj p-values and other objects for pheatmap
  labels_row <- paste0(rownames(expr_norm), " (",
    sprintf( "%.02e", sign_adjp), ")")
  labels_col <- colnames(expr_norm)
  annotation_col <- data.frame(condition = factor(condition))
  rownames(annotation_col) <- colnames(expr_norm)
  annotation_colors <- list(condition = color_conditions)
  color <- colorRampPalette(c("#87CEFA", "#56B4E9", "#56B4E9", "#0072B2",
    "#000000", "#D55E00", "#E69F00", "#E69F00", "#FFD700"))(100)
  breaks = seq(from = -th, to = th, length.out = 101)
  legend_breaks = seq(from = -round(th), to = round(th), by = 1)
  gaps_col <- as.numeric(table(annotation_col$condition))

  pheatmap(expr_norm, color = color, breaks = breaks,
    legend_breaks = legend_breaks, cluster_cols = FALSE, cluster_rows = FALSE,
    labels_col = labels_col, labels_row = labels_row, gaps_col = gaps_col,
    annotation_col = annotation_col, annotation_colors = annotation_colors,
    fontsize = 8)
}

## Apply the arcsine-square-root transformation to the proportions
asin_table <- asin(sqrt((t(t(counts_table) / colSums(counts_table)))))
asin <- as.data.frame.matrix(asin_table)
## Get significant clusters and sort them by significance
sign_clusters <- names(which(sort(da_out2$adjp[, "adjp_BCRXLvsRef"]) < FDR_cutoff))
## Get the adjusted p-values for the significant clusters
sign_adjp <- da_out2$adjp[sign_clusters , "adjp_BCRXLvsRef", drop=FALSE]
## Normalize the transformed proportions to mean = 0 and sd = 1
asin_norm <- normalization_wrapper(asin[sign_clusters, ])

mm <- match(colnames(asin_norm), md$sample_id)
plot_differential_heatmap_wrapper(expr_norm = asin_norm, sign_adjp = sign_adjp,
  condition = md$condition[mm], color_conditions = color_conditions)
```

Differential analysis of marker expression stratified by cell population

For this part of the analysis, we calculate the median expression of the 14 signaling markers in each cell population (merged cluster) and sample. These will be used as the response variable Y_{ij} in the linear model (LM) or linear mixed model (LMM), for which we assume that the median marker expression follows a Gaussian distribution (on the already arcsinh-transformed scale). The linear model is formulated as follows:

$$Y_{ij} = \beta_0 + \beta_1 x_{ij} + \epsilon_{ij},$$

where $\epsilon_{ij} \sim N(0, \sigma^2)$, and the mixed model includes a random intercept for each patient:

$$Y_{ij} = \beta_0 + \beta_1 x_{ij} + \gamma_i + \epsilon_{ij},$$

where $\gamma_i \sim N(0, \sigma_\gamma^2)$. In the current experiment, we have an intercept (basal level) and a single covariate, x_{ij}, which is represented as a binary (stimulated/unstimulated) variable. For more complicated designs or batch effects, additional columns of a design matrix can be used.

One drawback of summarizing the protein marker intensity with a median over cells is that all the other characteristics of the distribution, such as bimodality, skewness and variance, are ignored. On the other hand, it results in a simple, easy to interpret approach, which in many cases will be able to detect interesting changes. Another issue that arises from using a summary statistic is the level of uncertainty, which increases as the number of cells used to calculate it decreases. In the statistical modeling, this problem could be partially handled by assigning observation weights (number of cells) to each cluster and sample (parameter `weights` in the `lm` and `lmer` functions). However, since each cluster is tested separately, these weights do not account for the differences in size between clusters.

There might be instances of small cell populations for which no cells are observed in some samples or where the number of cells is very low. For clusters absent from a sample (e.g. due to biological variance or insufficient sampling), NAs are introduced because no median expression can be calculated; in the case of few cells, the median may be quite variable. Thus, we apply a filter to remove samples that have fewer than 5 cells. We also remove cases where marker expression is equal to zero in all the samples, as this leads to an error during model fitting.

```
## Get median marker expression per sample and cluster
expr_median_sample_cluster_tbl <- data.frame(expr[, functional_markers],
  sample_id = sample_ids, cluster = cell_clustering1m) %>%
  group_by(sample_id, cluster) %>%
  summarize_all(funs(median))
## Melt
expr_median_sample_cluster_melt <- melt(expr_median_sample_cluster_tbl,
  id.vars = c("sample_id", "cluster"), value.name = "median_expression",
  variable.name = "antigen")
## Rearange so the rows represent clusters and markers
expr_median_sample_cluster <- dcast(expr_median_sample_cluster_melt,
  cluster + antigen ~ sample_id,  value.var = "median_expression")
rownames(expr_median_sample_cluster) <- paste0(expr_median_sample_cluster$cluster,
  "_", expr_median_sample_cluster$antigen)
## Eliminate clusters with low frequency
clusters_keep <- names(which((rowSums(counts < 5) == 0)))
keepLF <- expr_median_sample_cluster$cluster %in% clusters_keep
expr_median_sample_cluster <- expr_median_sample_cluster[keepLF, ]
## Eliminate cases with zero expression in all samples
keep0 <- rowSums(expr_median_sample_cluster[, md$sample_id]) > 0
expr_median_sample_cluster <- expr_median_sample_cluster[keep0, ]
```

It is helpful to plot the median expression of all the markers in each cluster for each sample colored by condition, to get a rough image of how strong the differences might be (see Figure 29). We do this by combining boxplots and jitter.

```
ggdf <- expr_median_sample_cluster_melt[expr_median_sample_cluster_melt$cluster
  %in% clusters_keep, ]
## Add info about samples
mm <- match(ggdf$sample_id, md$sample_id)
ggdf$condition <- factor(md$condition[mm])
ggdf$patient_id <- factor(md$patient_id[mm])
ggplot(ggdf) +
  geom_boxplot(aes(x = antigen, y = median_expression,
    color = condition, fill = condition),
    position = position_dodge(), alpha = 0.5, outlier.color = NA) +
  geom_point(aes(x = antigen, y = median_expression, color = condition,
    shape = patient_id), alpha = 0.8, position = position_jitterdodge(),
    size = 0.7) +
  facet_wrap(~ cluster, scales = "free_y", ncol=2) +
  theme_bw() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1)) +
  scale_color_manual(values = color_conditions) +
  scale_fill_manual(values = color_conditions) +
  scale_shape_manual(values = c(16, 17, 8, 3, 12, 0, 1, 2)) +
  guides(shape = guide_legend(override.aes = list(size = 2)))
```

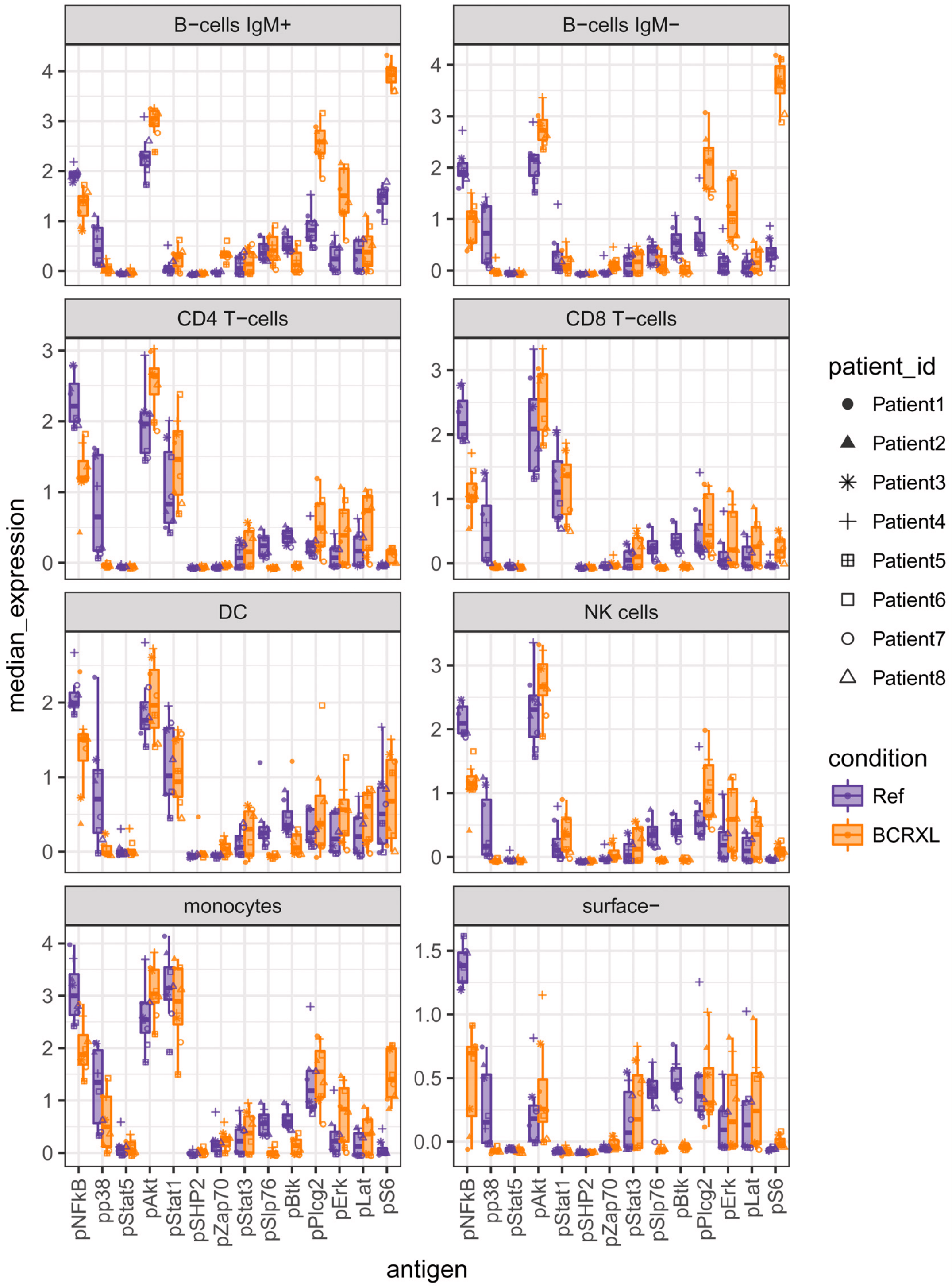


Figure 29. Median (arcsinh-transformed) expression of 14 signaling markers (x-axis) across the 8 identified PBMC cell populations (individual panels). Values for the two conditions are indicated with different colors: violet for the unstimulated (Ref) and orange for the stimulated with BCR/FcR-XL (BCRXL) samples. Values for each patient are indicated with different shape. The 8 cell populations are a result of manual merging of the 20 FlowSOM metaclusters.

We created a wrapper function `differential_expression_wrapper` that performs the differential analysis of marker expression. The user needs to specify a data frame `expr_median` with marker expression, where each column corresponds to a sample and each row to a cluster/marker combination. One can choose between fitting a regular linear model `model = "lm"` or a linear mixed model `model = "lmer"`. The `formula` parameter must be adjusted adequately to the model choice. The wrapper function returns the non-adjusted and adjusted p-values for each of the specified contrasts `K` for each cluster/marker combination.

```
differential_expression_wrapper <- function(expr_median, md, model = "lmer",
  formula, K){
  ## Fit LMM or LM for each marker separately
  fit_gaussian <- lapply(1:nrow(expr_median), function(i){
    data_tmp <- data.frame(y = as.numeric(expr_median[i, md$sample_id]), md)
    switch(model,
      lmer = {
        fit_tmp <- lmer(formula, data = data_tmp)
      },
      lm = {
        fit_tmp <- lm(formula, data = data_tmp)
      })
    ## Fit contrasts one by one
    out <- apply(K, 1, function(k){
      contr_tmp <- glht(fit_tmp, linfct = matrix(k, 1))
      summ_tmp <- summary(contr_tmp)
      pval <- summ_tmp$test$pvalues
      return(pval)
    })
    return(out)
  })
  pvals <- do.call(rbind, fit_gaussian)
  colnames(pvals) <- paste0("pval_", contrast_names)
  rownames(pvals) <- rownames(expr_median)
  ## Adjust the p-values
  adjp <- apply(pvals, 2, p.adjust, method = "BH")
  colnames(adjp) <- paste0("adjp_", contrast_names)
  return(list(pvals = pvals, adjp = adjp))
}
```

To present how accounting for the within patient variability with the mixed model increases sensitivity, we also fit a regular linear model. The linear mixed model has a random intercept for each patient.

```
formula_lm <- y ~ condition
formula_lmer <- y ~ condition + (1|patient_id)
```

By accounting for the patient effect, we detect almost twice as many cases of differential signaling compared to the regular linear model.

```
de_out1 <- differential_expression_wrapper(expr_median = expr_median_sample_cluster,
  md = md, model = "lm", formula = formula_lm, K = K)
apply(de_out1$adjp < FDR_cutoff, 2, table)

##       adjp_BCRXLvsRef
## FALSE              51
## TRUE               42

de_out2 <- differential_expression_wrapper(expr_median = expr_median_sample_cluster,
  md = md, model = "lmer", formula = formula_lmer, K = K)
apply(de_out2$adjp < FDR_cutoff, 2, table)
```

```
##       adjp_BCRXLvsRef
## FALSE              23
## TRUE               70
```

One can assemble together an output table with the information about median marker expression in each cluster and sample, and the obtained p-values.

```
de_output2 <- data.frame(expr_median_sample_cluster,
  de_out2$pvals, de_out2$adjp, row.names = NULL)
print(head(de_output2), digits = 2)
```

```
##        cluster antigen BCRXL1 BCRXL2 BCRXL3 BCRXL4 BCRXL5 BCRXL6 BCRXL7
## 1 B-cells IgM+   pNFkB  1.179  0.880  0.808   1.47  1.361  1.725  1.436
## 2 B-cells IgM+    pp38  0.109 -0.012  0.044   0.24 -0.046  0.083 -0.039
## 3 B-cells IgM+    pAkt  3.247  2.960  2.951   3.26  2.382  3.184  2.762
## 4 B-cells IgM+  pStat1  0.343  0.126  0.242   0.33 -0.010  0.616 -0.050
## 5 B-cells IgM+  pZap70  0.317  0.287  0.351   0.40  0.132  0.604  0.267
## 6 B-cells IgM+  pStat3 -0.047 -0.059  0.451   0.35 -0.058 -0.026  0.534
##    BCRXL8    Ref1   Ref2    Ref3    Ref4   Ref5   Ref6   Ref7   Ref8
## 1  1.5747  1.9639  1.869  1.7726  2.1833  1.861  1.953  1.915  1.979
## 2 -0.0055  0.8891  1.113  0.8534  0.6424  0.126  0.210  0.128  0.126
## 3  3.1439  2.3195  2.310  2.2688  3.0858  1.729  2.024  2.145  2.603
## 4  0.3795 -0.0058  0.064  0.0079  0.5151 -0.047  0.030 -0.034  0.191
## 5  0.3202 -0.0198 -0.033 -0.0336 -0.0056 -0.061 -0.060 -0.032 -0.017
## 6  0.3092 -0.0479 -0.082  0.2652  0.1567 -0.060 -0.066  0.275  0.381
##   pval_BCRXLvsRef adjp_BCRXLvsRef
## 1         6.1e-11         2.7e-10
## 2         7.5e-04         1.6e-03
## 3         2.6e-11         1.3e-10
## 4         6.2e-02         7.5e-02
## 5         1.6e-14         1.0e-13
## 6         5.6e-02         7.1e-02
```

To report the significant results, we use a heatmap (see Figure 30). Instead of plotting the absolute expression, we display the normalized expression, which better highlights the direction of marker changes. Additionally, we order the cluster-marker instances by their significance and group them by cell type (cluster).

```
## Keep the significant markers, sort them by significance and group by cluster
sign_clusters_markers <- names(which(de_out2$adjp[, "adjp_BCRXLvsRef"] < FDR_cutoff))
oo <- order(expr_median_sample_cluster[sign_clusters_markers, "cluster"],
  de_out2$adjp[sign_clusters_markers, "adjp_BCRXLvsRef"])
sign_clusters_markers <- sign_clusters_markers[oo]

## Get the significant adjusted p-values
sign_adjp <- de_out2$adjp[sign_clusters_markers , "adjp_BCRXLvsRef"]

## Normalize expression to mean = 0 and sd = 1
expr_s <- expr_median_sample_cluster[sign_clusters_markers,md$sample_id]
expr_median_sample_cluster_norm <- normalization_wrapper(expr_s)

mm <- match(colnames(expr_median_sample_cluster_norm), md$sample_id)
plot_differential_heatmap_wrapper(expr_norm = expr_median_sample_cluster_norm,
  sign_adjp = sign_adjp, condition = md$condition[mm],
  color_conditions = color_conditions)
```

DA of the overall marker expression

The analysis of *overall* expression is analogous to the previous section, except that median marker expression is aggregated from all the cells in a given sample, Figure 31.

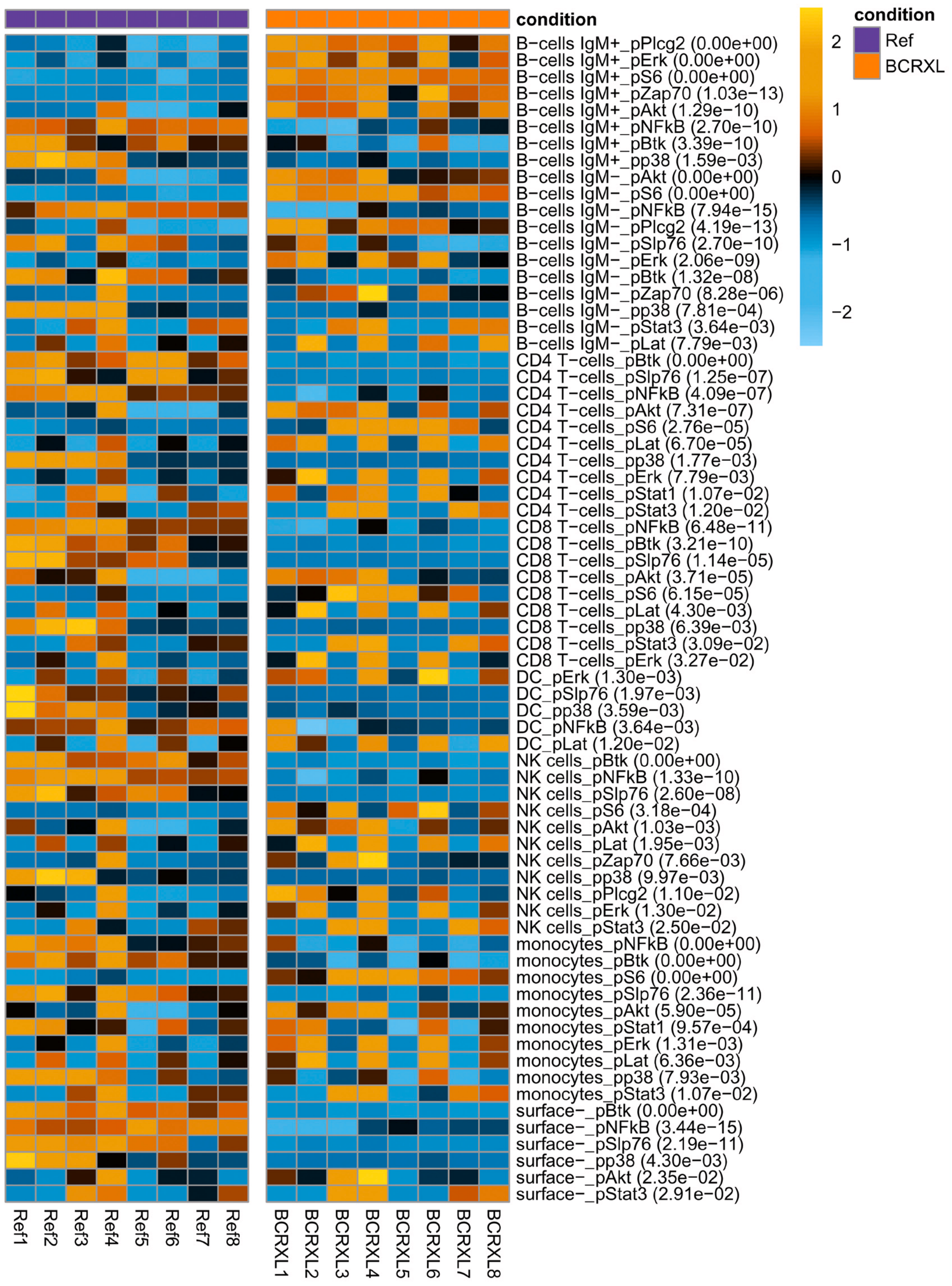


Figure 30. Normalized expression of signaling markers in the 8 PBMC populations that are significantly differentially expressed between BCR/FcR-XL stimulated and unstimulated condition. The heat represents median (arcsinh-transformed) marker expression that was subsequently normalized per cluster-marker (rows) to mean of zero and standard deviation of one. The color of the heat varies from blue representing relative under-expression to orange representing relative over-expression. Bar at the top of the heatmap indicates the condition the samples (columns) belong to: violet for the unstimulated (Ref) and orange for the stimulated with BCR/FcR-XL (BCRXL) condition. Numbers in the brackets next to the cluster-marker names indicate adjusted p-values and cluster-marker are sorted so that they block per cluster and within each block, markers on the top show the most significant changes between the two conditions. Shown are only the significant cluster-markers for which adjusted p-values < 0.05.

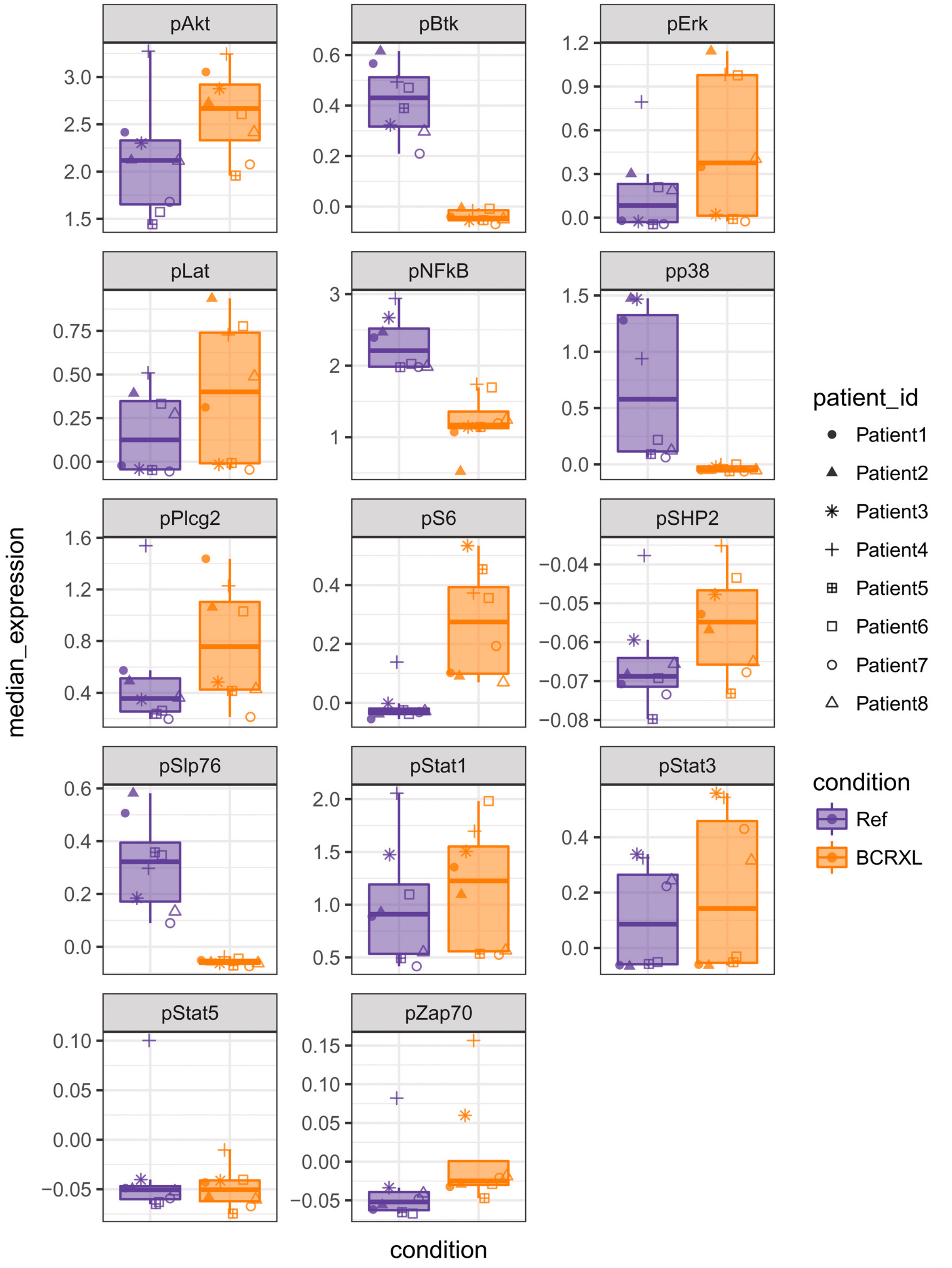


Figure 31. Median (arcsinh-transformed) expression of 14 signaling markers calculated from all the cells in a given sample in the PBMC dataset. Values for the two conditions are indicated with different colors: violet for the unstimulated (Ref) and orange for the stimulated with BCR/FcR-XL (BCRXL) samples. Values for each patient are indicated with different shape.

```
ggdf <- melt(data.frame(expr_median_sample[functional_markers, ],
  antigen = functional_markers), id.vars = "antigen",
  value.name = "median_expression", variable.name = "sample_id")
## Add condition info
mm <- match(ggdf$sample_id, md$sample_id)
ggdf$condition <- factor(md$condition[mm])
ggdf$patient_id <- factor(md$patient_id[mm])
ggplot(ggdf) +
  geom_boxplot(aes(x = condition, y = median_expression, color = condition,
    fill = condition),  position = position_dodge(), alpha = 0.5,
    outlier.color = NA) +
  geom_point(aes(x = condition, y = median_expression, color = condition,
    shape = patient_id), alpha = 0.8, position = position_jitterdodge()) +
  facet_wrap(~ antigen, scales = "free", nrow = 5) +
  theme_bw() +
  theme(axis.text.x = element_blank(), axis.ticks.x = element_blank()) +
  scale_color_manual(values = color_conditions) +
  scale_fill_manual(values = color_conditions) +
  scale_shape_manual(values = c(16, 17, 8, 3, 12, 0, 1, 2))
```

Similar to the analysis above, we identify more markers being differentially expressed with the LMM, which accounts for the within patient variability.

```
## Fit a linear model
de_out3 <- differential_expression_wrapper(expr_median =
    expr_median_sample[functional_markers, ],
  md = md, model = "lm", formula = formula_lm, K = K)
apply(de_out3$adjp < FDR_cutoff, 2, table)

##       adjp_BCRXLvsRef
## FALSE               9
## TRUE                5

## Fit a linear mixed model with patient ID as a random effect
de_out4 <- differential_expression_wrapper(expr_median =
    expr_median_sample[functional_markers, ],
  md = md, model = "lmer", formula = formula_lmer, K = K)
apply(de_out4$adjp < FDR_cutoff, 2, table)

##       adjp_BCRXLvsRef
## FALSE               3
## TRUE               11
```

As before, we create an output table with the median marker expression calculated in each sample and the p-values, and we plot a heatmap with the significant markers sorted by their statistical significance (Figure 32).

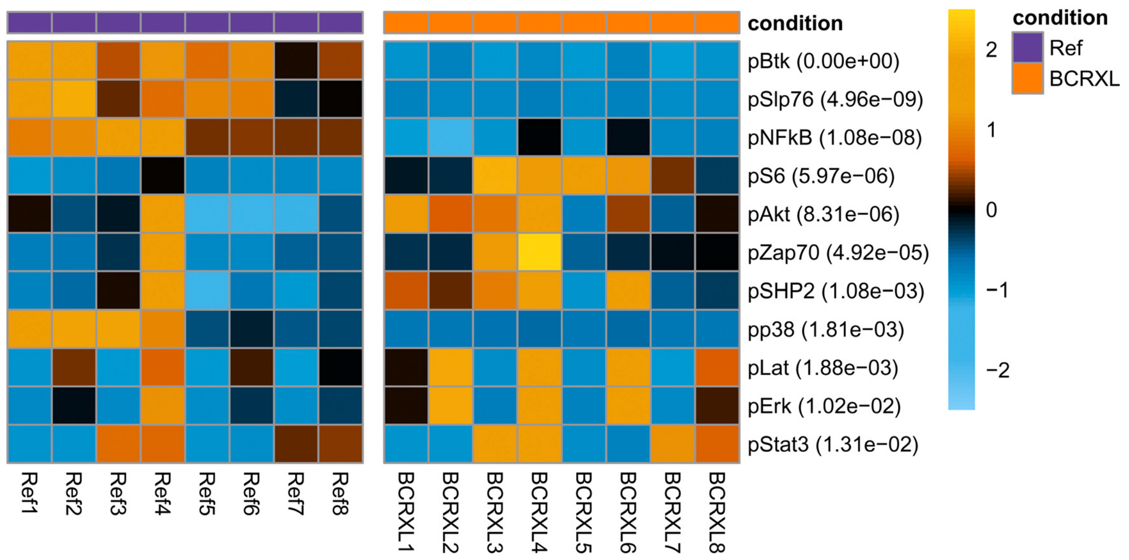


Figure 32. Normalized expression of signaling markers calculated over all the cells in the PBMC dataset that are significantly differentially expressed between BCR/FcR-XL stimulated and unstimulated condition. The heat represents median (arcsinh-transformed) marker expression that was subsequently normalized per marker (rows) to mean of zero and standard deviation of one. The color of the heat varies from blue representing relative under-expression to orange representing relative over-expression. Bar at the top of the heatmap indicates the condition the samples (columns) belong to: violet for the unstimulated (Ref) and orange for the stimulated with BCR/FcR-XL (BCRXL) condition. Numbers in the brackets next to the marker names indicate adjusted p-values and markers are sorted so that markers on the top exhibit the most significant changes between the two conditions. Shown are only the significant markers for which adjusted p-values < 0.05.

```
de_output4 <- data.frame(antigen = functional_markers,
  expr_median_sample[functional_markers, ], de_out4$pvals, de_out4$adjp)
print(head(de_output4), digits=2)
```

```
##        antigen BCRXL1 BCRXL2 BCRXL3  BCRXL4 BCRXL5  BCRXL6 BCRXL7 BCRXL8
## pNFkB    pNFkB  1.070  0.520  1.144  1.7397  1.143  1.6951  1.195  1.245
## pp38      pp38 -0.052 -0.057 -0.024 -0.0034 -0.061 -0.0006 -0.064 -0.053
## pStat5  pStat5 -0.043 -0.058 -0.041 -0.0103 -0.075 -0.0404 -0.067 -0.060
## pAkt      pAkt  3.053  2.727  2.876  3.2424  1.958  2.6068  2.075  2.416
## pStat1  pStat1  1.356  1.096  1.504  1.6960  0.535  1.9823  0.526  0.566
## pSHP2    pSHP2 -0.053 -0.057 -0.048 -0.0352 -0.073 -0.0435 -0.068 -0.065
##          Ref1   Ref2   Ref3   Ref4   Ref5   Ref6   Ref7   Ref8
## pNFkB   2.392  2.469  2.670  2.940  1.979  2.025  1.980  1.985
## pp38    1.280  1.474  1.467  0.939  0.091  0.218  0.062  0.122
## pStat5 -0.049 -0.049 -0.040  0.100 -0.065 -0.063 -0.059 -0.052
## pAkt    2.416  2.122  2.300  3.273  1.443  1.573  1.680  2.114
## pStat1  0.889  0.930  1.474  2.056  0.493  1.097  0.416  0.549
## pSHP2  -0.071 -0.068 -0.059 -0.038 -0.080 -0.069 -0.073 -0.066
##        pval_BCRXLvsRef adjp_BCRXLvsRef
## pNFkB          2.3e-09         1.1e-08
## pp38           1.0e-03         1.8e-03
## pStat5         3.0e-01         3.0e-01
## pAkt           3.0e-06         8.3e-06
## pStat1         1.9e-01         2.1e-01
## pSHP2          5.4e-04         1.1e-03
## Keep the significant markers and sort them by significance
sign_markers <- names(which(sort(de_out4$adjp[, "adjp_BCRXLvsRef"]) < FDR_cutoff))
## Get the adjusted p-values
sign_adjp <- de_out4$adjp[sign_markers , "adjp_BCRXLvsRef"]
## Normalize expression to mean = 0 and sd = 1
expr_median_sample_norm <- normalization_wrapper(expr_median_sample[sign_markers, ])

mm <- match(colnames(expr_median_sample_norm), md$sample_id)
plot_differential_heatmap_wrapper(expr_norm = expr_median_sample_norm,
  sign_adjp = sign_adjp, condition = md$condition[mm],
  color_conditions = color_conditions)
```

Obtaining higher resolution

In the proposed workflow, we concentrated on identification of the main cell types in PBMC. Our goal was to identify around 6 main cell types. Following the over-clustering strategy, we have chosen to performed the SOM clustering into 100 (the default) groups followed by the consensus clustering into 20 groups, from which we could annotate 8 cell types. These 8 cell types were then used in the differential analysis.

If the number of expected cell types is higher, the user can increase the size of the SOM grid in the `BuildSOM` function using the `xdim` and `ydim` arguments and increase the maximum number of consensus clusters in the `ConsensusClusterPlus` function with the `maxK` argument.

One could also use a strategy based on subsequent clustering of identified clusters, which we refer to as *reclustering*. In the starting step, one uses the presented workflow to identify the main cell types. In the following steps, the same clustering workflow is applied individually to the cell populations for which more resolution is desired. Restricting to one subpopulation at a time results in easier cluster annotation. The differential analysis can be applied to the final clusters in the same way as described in the workflow assuming tables with cell counts and median marker expression are available.

The differential analysis could be also conducted on the unmerged (20) consensus clusters and the manual annotation could be done at the end.

Discussion

In this workflow, we have presented a pipeline for diverse differential analyses of HDCyto datasets. First, we highlight quality control steps, where aggregate characteristics of the samples are visualized (e.g. an MDS plot), allowing for verification of the experimental design, detection of batch effects and outlying samples. Next, cell population identification was carried out via clustering, which forms the basis for subsequent differential analyses of cell population abundance, differential marker expression within a population or overall marker expression differences. The approaches to differential analyses proposed here are very general and thus able to model complex experimental designs via design matrices, such as factorial experiments, paired experiments or adjustment for batch effects. We have presented a range of visualizations that help in understanding the data and reporting the results of clustering and differential analyses. The wrapper functions presented in this workflow may need to be tailored to the needs of a different experiment.

Clustering is one of the most challenging steps in the workflow, and its accuracy is critical to the downstream differential analyses. Getting the right resolution of clusters is crucial, since there can be situations where a biologically meaningful cell population may be differentially enriched between conditions, but in an automatic clustering, was combined with another cell population that behaves differently. We have shown that some level of over-clustering is convenient for detecting meaningful cell populations, since automatic detection of the number of natural clusters is difficult (Weber & Robinson, 2016). However, there are tradeoffs between the resolution of clustering and the labor involved in aggregating them to biologically meaningful clusters. Overall, we take an interactive but flexible algorithm-guided approach together with subject-area experts to arrive at sensible cell populations. In particular, we rely on various visualizations, such as dendrograms, t-SNE maps or other dimension reduction techniques to guide us in the process. Alternative strategies could be combined with the statistical inference we present, such as over-clustering combined with data-driven aggregation to the optimal resolution.

While we have a good understanding of how computational algorithms recapitulate manual gating in high dimensions (Weber & Robinson, 2016), one of the open areas of research remains how to best cluster *across* samples.

The data analyzed here (Bodenmiller *et al.*, 2012; Bruggner *et al.*, 2014) was generated using sample barcoding; this strategy reduces inter-sample variability, since all samples are exposed to the same antibody cocktail and measured in a single acquisition (Zunder *et al.*, 2015). Thus, the range of marker expression for each channel should, in principle, be within a similar range across samples.

In our approach, we aggregated all cells together before clustering. Because of this aggregation, the clustering is blind to the sample labels, and thus in principle, does not bias the downstream statistical inferences. Moreover, we directly obtain consistent clustering between samples. However, some challenges may arise when there are substantial differences in numbers of cells in samples. There is a risk that larger samples may drive the final clustering results. A simple solution to this problem could be ensuring that each sample contributes an equal amount of cells into the clustering analysis. This could be done by sampling an equal number of cells from each sample. However, there are two main drawbacks of this strategy. First, a substantial amount of data (cells) may be removed from the analysis if there are samples with few cells, thus resulting in information loss. Second, during down-sampling, some of the smaller populations may become underrepresented or even skipped. An alternative would be to cluster within each sample and then aggregate a collection of metaclusters across samples (Pyne *et al.*, 2009). A recent approach, called PAC-MAN (Li *et al.*, 2017), uses a combination of high dimensional density estimation, hierarchical clustering and network inference and comparison to extract clusters across samples, with a possibility to handle batch effects.

Additional challenges may arise when combining data from different instrument acquisitions and additional preprocessing treatments may need to be applied. Despite adjustments through bead-based normalization (Finck *et al.*, 2013), the observed marker expression may be affected by the varying efficiency of antibody binding in each batch and by the ion detection sensitivity after machine calibration. Beyond normalization, other strategies have been proposed, such as equalizing the dynamic range between batches for each marker (e.g. normalization to the 0-1 range, z-scores, quantile normalization), the use of warping functions to eliminate non-linear distortions (see the *cydar* vignette), or learning marker distribution shifts between the batches based on a manually gated reference cell type and using it to correct marker expression for the whole dataset (Arvaniti & Claassen, 2017).

Alternatively, one could consider batch-wise clustering of samples. On the other hand, to be able to use those results, one still needs to match cell populations across batches. The matching could be done manually, or with the automated approaches developed for flow cytometry (Pyne *et al.*, 2009). However, a comprehensive evaluation of these approaches and their effect on downstream analyses is still missing, especially when batch effects are present. Overall, we expect that as a general rule, including batch parameters (or other covariates) in the linear modeling helps to mitigate the problem.

We presented a classical statistical approach where preprocessing of the HDCyto data leads to tables of summaries (e.g. cell counts) or aggregated measurements (e.g. cluster-specific signals) for each sample, which become the input to statistical model. Of course, there are a variety of alternative computational approaches available to the user. We have mentioned *citrus* and *CellCnn*, which are both machine-learning approaches that fit a reverse model to ours (i.e. phenotype of interest as the response variable).

Another set of methods (*MIMOSA* and *COMPASS*), based on Bayesian hierarchical framework, was proposed in the vaccine development field, where the antigen-specific T-cell response to stimulation for each subject is modeled using mixtures of beta-binomial or Dirichlet-multinomial distributions (Finak *et al.*, 2014; Lin *et al.*, 2015). These strategies bear similarity to the mixed models applied for differential abundance in this workflow while handling over-dispersion due to subject-to-subject variability.

Neither of these approaches are directly able to account for batch effects or complicated designs. However, they may have advantages in the search for rare distinguishing populations, which could be used together with our framework for formal statistical testing.

One of the main goals of this workflow was to highlight how a model-based approach is able to handle complex experimental designs. This becomes important in many experimental situations where covariates (e.g. age, gender, batch) may affect the observed HDCyto data. Thus, the classical regression framework allows also to flexibly test situations well beyond two-group differences. Of course, alternatives exist for two group comparisons, such as the nonparametric Mann-Whitney-Wilcoxon test (Hartmann *et al.*, 2016), which makes no assumptions about normality of the data, or the Student's t-test (Pejoski *et al.*, 2016) and its variations, such as the paired t-test.

We note that the LM, LMM and GLMM may perform poorly for extremely small samples. Solutions similar to those widely accepted in transcriptomics that share information over variance parameters (Love *et al.*, 2014; Ritchie *et al.*, 2015; Robinson & Smyth, 2007) could be leveraged. An example of such an approach is *cydar* (Lun *et al.*, 2017), which performs the differential abundance analysis (on hypersphere counts) using the generalized linear modeling capabilities of *edgeR* (McCarthy *et al.*, 2012).

In the differential marker expression analysis, we compare the median marker expression between samples, while in many cases this approach is sufficient to detect interesting changes, by summarizing marker expression over cells to a single value we ignore all the other characteristics of the expression distribution, such as bimodality, skewness and variance, which may be relevant in some studies. Thus, it may be interesting to extend our comparisons to the whole marker distributions, instead of just changes in the medians.

The approach presented in this workflow is not fully automated due to the cluster merging, annotating, and extensive exploratory data analysis steps. In general, our philosophy is that fully automated analyses are to be avoided, but rather a battery of diagnostic checks can be designed, as we have promoted here. Cluster annotation remains a manual step in many other approaches as well. Recently, a tool was proposed for consistent characterization of cell subsets using marker enrichment modeling (MEM) (Diggins *et al.*, 2017).

To keep the analysis of this workflow reproducible, one needs to define a random seed before running *FlowSOM* and t-SNE. This is especially important in the clustering step, where the order of clusters may change with different seeds, and the cluster merging needs to be matched to the seed used.

Software availability

All software packages used in this workflow are publicly available from the Comprehensive R Archive Network (https://cran.r-project.org) or the Bioconductor project (http://bioconductor.org). The specific version numbers of the packages used are shown below, along with the version of the R installation.

Version numbers of all the Bioconductor packages correspond to the release version 3.5 of the Bioconductor project.

Users can install all required packages and execute the workflow by following the instructions at https://www.bioconductor.org/help/workflows/cytofWorkflow.

```
sessionInfo()
```

```
## R version 3.4.1 (2017-06-30)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS Sierra 10.12.5
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/3.4_r3.5/Resources/lib/
libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.4_r3.5/Resources/lib/
libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] grid      stats     graphics   grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] bindrcpp_0.2                  cytofWorkflow_1.0.4
##  [3] multcomp_1.4-7                TH.data_1.0-8
##  [5] MASS_7.3-47                   survival_2.41-3
##  [7] mvtnorm_1.0-6                 lme4_1.1-14
##  [9] Matrix_1.2-10                 cowplot_0.8.0
## [11] Rtsne_0.13                    ConsensusClusterPlus_1.40.0
## [13] FlowSOM_1.8.0                  igraph_1.1.2
## [15] ComplexHeatmap_1.14.0          pheatmap_1.0.8
## [17] RColorBrewer_1.1-2             ggrepel_0.7.0
## [19] limma_3.32.10                  dplyr_0.7.4
## [21] reshape2_1.4.2                 ggridges_0.4.1
## [23] ggplot2_2.2.1                  matrixStats_0.52.2
## [25] flowCore_1.42.3                readxl_1.0.0
## [27] knitr_1.17                     BiocStyle_2.4.1
## [29] colorout_1.1-2
##
## loaded via a namespace (and not attached):
##  [1] nlme_3.1-131             tsne_0.1-3
##  [3] httr_1.3.1               rprojroot_1.2
##  [5] prabclus_2.2-6           tools_3.4.1
##  [7] backports_1.1.0          R6_2.2.2
##  [9] lazyeval_0.2.0           BiocGenerics_0.22.0
## [11] colorspace_1.3-2         trimcluster_0.1-2
## [13] nnet_7.3-12              GetoptLong_0.1.6
## [15] gridExtra_2.2.1          compiler_3.4.1
## [17] BiocWorkflowTools_1.2.0 graph_1.54.0
## [19] Biobase_2.36.2           sandwich_2.4-0
## [21] bookdown_0.4             diptest_0.75-7
## [23] scales_0.5.0             DEoptimR_1.0-8
## [25] robustbase_0.92-7        stringr_1.2.0
## [27] digest_0.6.12            minqa_1.2.4
## [29] rmarkdown_1.6            rrcov_1.4-3
## [31] pkgconfig_2.0.1          htmltools_0.3.6
## [33] highr_0.6                rlang_0.1.2
## [35] GlobalOptions_0.0.12     shape_1.4.3
## [37] bindr_0.1                zoo_1.8-0
## [39] mclust_5.3               dendextend_1.5.2
## [41] magrittr_1.5             modeltools_0.2-21
## [43] Rcpp_0.12.13             munsell_0.4.3
## [45] viridis_0.4.0            stringi_1.1.5
## [47] whisker_0.3-2            yaml_2.1.14
## [49] flexmix_2.3-14           plyr_1.8.4
## [51] parallel_3.4.1           lattice_0.20-35
## [53] splines_3.4.1            circlize_0.4.1
## [55] rjson_0.2.15             fpc_2.1-10
## [57] corpcor_1.6.9            codetools_0.2-15
## [59] stats4_3.4.1             XML_3.98-1.9
## [61] glue_1.1.1               evaluate_0.10.1
## [63] nloptr_1.0.4             cellranger_1.1.0
## [65] gtable_0.2.0             kernlab_0.9-25
## [67] assertthat_0.2.0         class_7.3-14
## [69] pcaPP_1.9-72             viridisLite_0.2.0
## [71] tibble_1.3.4             cluster_2.0.6
```

Author contributions

MN and MDR designed and ran analyses. MN drafted the manuscript with input from MDR. CK and SG performed experiments, analyzed data and gave feedback on clinical applications. CK performed the cluster merging. FJH and LMW contributed code and ideas for the analyses. MPL interpreted data and provided clinical perspective. BB gave feedback on the manuscript and the bioinformatics. All authors read and approved the final manuscript and have agreed to the content.

Competing interests
No competing interests were disclosed.

Grant information
MN acknowledges the funding from a Swiss Institute of Bioinformatics (SIB) Fellowship.

The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Acknowledgments
The authors wish to thank members of the Robinson, Bodenmiller and von Mering groups from the Institute of Molecular Life Sciences, University of Zurich for helpful discussions.

References

Aghaeepour N, Finak G, FlowCAP Consortium, *et al.*: **Critical assessment of automated flow cytometry data analysis techniques.** *Nat Methods.* Nature Publishing Group, a division of Macmillan Publishers Limited. 2013; **10**(3): 228–38.
PubMed Abstract | Publisher Full Text | Free Full Text

Angerer P, Haghverdi L, Büttner M, *et al.*: ***destiny*: diffusion maps for large-scale single-cell data in R.** *Bioinformatics.* 2016; **32**(8): 1241–3.
PubMed Abstract | Publisher Full Text

Arvaniti E, Claassen M: **Sensitive detection of rare disease-associated cell subsets via representation learning.** *Nat Commun.* 2017; **8**: 14825.
PubMed Abstract | Publisher Full Text | Free Full Text

Bendall SC, Davis KL, Amir el-AD, *et al.*: **Single-cell trajectory detection uncovers progression and regulatory coordination in human B cell development.** *Cell.* Elsevier. 2014; **157**(3): 714–25.
PubMed Abstract | Publisher Full Text | Free Full Text

Bendall SC, Simonds EF, Qiu P, *et al.*: **Single-cell mass cytometry of differential immune and drug responses across a human hematopoietic continuum.** *Science.* American Association for the Advancement of Science, 2011; **332**(6030): 687–96.
PubMed Abstract | Publisher Full Text | Free Full Text

Bodenmiller B, Zunder ER, Finck R, *et al.*: **Multiplexed mass cytometry profiling of cellular states perturbed by small-molecule regulators.** *Nat Biotechnol.* Nature Publishing Group, 2012; **30**(9): 858–67.
PubMed Abstract | Publisher Full Text | Free Full Text

Bruggner RV, Bodenmiller B, Dill DL, *et al.*: **Automated identification of stratifying signatures in cellular subpopulations.** *Proc Natl Acad Sci U S A.* 2014; **111**(26): E2770–7.
PubMed Abstract | Publisher Full Text | Free Full Text

Chen H, Lau MC, Wong MT, *et al.*: **Cytofkit: A Bioconductor Package for an Integrated Mass Cytometry Data Analysis Pipeline.** *PLoS Comput Biol.* Public Library of Science, 2016; **12**(9): e1005112.
PubMed Abstract | Publisher Full Text | Free Full Text

Chevrier S, Crowell H, Zanotelli VRT, *et al.*: **Channel crosstalk correction in suspension and imaging mass cytometry.** *bioRxiv.* Cold Spring Harbor Laboratory. 2017.
Publisher Full Text

Diggins KE, Greenplate AR, Leelatian N, *et al.*: **Characterizing cell subsets using marker enrichment modeling.** *Nat Methods.* 2017; **14**(3): 275–78.
PubMed Abstract | Publisher Full Text | Free Full Text

Finak G, Jiang M: **FlowWorkspace: Infrastructure for Representing and Interacting with the Gated Cytometry.** 2011.
Reference Source

Finak G, Frelinger J, Jiang W, *et al.*: **OpenCyto: an open source infrastructure for scalable, robust, reproducible, and automated, end-to-end flow cytometry data analysis.** *PLoS Comput Biol.* 2014; **10**(8): e1003806.
PubMed Abstract | Publisher Full Text | Free Full Text

Finak G, McDavid A, Chattopadhyay P, *et al.*: **Mixture models for single-cell assays with applications to vaccine studies.** *Biostatistics.* 2014; **15**(1): 87–101.
PubMed Abstract | Publisher Full Text | Free Full Text

Finck R, Simonds EF, Jager A, *et al.*: **Normalization of mass cytometry data with bead standards.** *Cytometry A.* 2013; **83A**(5): 483–94.
PubMed Abstract | Publisher Full Text | Free Full Text

Haghverdi L, Buettner F, Theis FJ: **Diffusion maps for high-dimensional single-cell analysis of differentiation data.** *Bioinformatics.* 2015; **31**(18): 2989–98.
PubMed Abstract | Publisher Full Text

Hahne F, LeMeur N, Brinkman RR, *et al.*: **flowCore: a Bioconductor package for high throughput flow cytometry.** *BMC Bioinformatics.* 2009; **10**(1): 106.
PubMed Abstract | Publisher Full Text | Free Full Text

Hartmann FJ, Bernard-Valnet R, Quériault C, *et al.*: **High-dimensional single-cell analysis reveals the immune signature of narcolepsy.** *J Exp Med.* Rockefeller University Press, 2016; **213**(12): 2621–33.
PubMed Abstract | Publisher Full Text | Free Full Text

Jia C, Hu Y, Liu Y, *et al.*: **Mapping Splicing Quantitative Trait Loci in RNA-Seq.** *Cancer Inform.* 2014; **13**(Suppl 4): 35–43.
PubMed Abstract | Publisher Full Text | Free Full Text

Kotecha N, Krutzik PO, Irish JM: **Web-based analysis and publication of flow cytometry experiments.** *Curr Protoc Cytom.* John Wiley & Sons, Inc. 2010; **Chapter 10**: Unit10.17.
PubMed Abstract | Publisher Full Text | Free Full Text

Leipold MD: **Another step on the path to mass cytometry standardization.** *Cytometry A.* 2015; **87**(5): 380–82.
PubMed Abstract | Publisher Full Text

Levine JH, Simonds EF, Bendall SC, *et al.*: **Data-Driven Phenotypic Dissection of AML Reveals Progenitor-like Cells that Correlate with Prognosis.** *Cell.* Elsevier, 2015; **162**(1): 184–97.
PubMed Abstract | Publisher Full Text | Free Full Text

Li YH, Li D, Samusik N, *et al.*: **Scalable Multi-Sample Single-Cell Data Analysis by Partition-Assisted Clustering and Multiple Alignments of Networks.** *bioRxiv.* Cold Spring Harbor Labs Journals. 2017.
Publisher Full Text

Lin L, Finak G, Ushey K, *et al.*: **COMPASS identifies T-cell subsets correlated with clinical outcomes.** *Nat Biotechnol.* Nature Publishing Group, a division of Macmillan Publishers Limited. 2015; **33**(6): 610–6.
PubMed Abstract | Publisher Full Text | Free Full Text

Lin L, Frelinger J, Jiang W, *et al.*: **Identification and visualization of multidimensional antigen-specific T-cell populations in polychromatic cytometry data.** *Cytometry A.* 2015; **87**(7): 675–82.
PubMed Abstract | Publisher Full Text | Free Full Text

Love MI, Huber W, Anders S: **Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2.** *Genome Biol.* BioMed Central. 2014; **15**(12): 550.
PubMed Abstract | Publisher Full Text | Free Full Text

Lun ATL, Richard AC, Marioni JC: **Testing for differential abundance in mass cytometry data.** *Nat Methods.* Nature Publishing Group, a division of Macmillan Publishers Limited. 2017; **14**(7): 707–9.
PubMed Abstract | Publisher Full Text

Mahnke YD, Roederer M: **Optimizing a multicolor immunophenotyping assay.** *Clin Lab Med.* 2007; **27**(3): 469–85, v.
PubMed Abstract | Publisher Full Text | Free Full Text

McCarthy DJ, Chen Y, Smyth GK: **Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation.** *Nucleic Acids Res.* 2012; **40**(10): 4288–97.
PubMed Abstract | Publisher Full Text | Free Full Text

Monti S, Tamayo P, Mesirov J, *et al.*: **Consensus Clustering: A Resampling-Based Method for Class Discovery and Visualization of Gene Expression Microarray Data.** *Mach Learn.* 2003; **52**(1–2): 91–118.
Publisher Full Text

Pejoski D, Tchitchek N, Rodriguez Pozo A, *et al.*: **Identification of Vaccine-Altered Circulating B Cell Phenotypes Using Mass Cytometry and a Two-Step Clustering Analysis.** *J Immunol.* American Association of Immunologists, 2016; **196**(11): 4814–31.
PubMed Abstract | Publisher Full Text

Pyne S, Hu X, Wang K, *et al.*: **Automated high-dimensional flow cytometric data analysis.** *Proc Natl Acad Sci U S A.* 2009; **106**(21): 8519–24.
PubMed Abstract | Publisher Full Text | Free Full Text

Ritchie ME, Phipson B, Wu D, *et al.*: ***limma* powers differential expression analyses for RNA-sequencing and microarray studies.** *Nucleic Acids Res.* Oxford University Press, 2015; **43**(7): e47.
PubMed Abstract | Publisher Full Text | Free Full Text

Robinson MD, Smyth GK: **Moderated statistical tests for assessing differences in tag abundance.** *Bioinformatics.* 2007; **23**(21): 2881–7.
PubMed Abstract | Publisher Full Text

Roederer M: **Spectral compensation for flow cytometry: visualization artifacts, limitations, and caveats.** *Cytometry.* John Wiley & Sons, Inc. 2001; **45**(3): 194–205.
PubMed Abstract | Publisher Full Text

Saeys Y, Gassen SV, Lambrecht BN: **Computational flow cytometry: helping to make sense of high-dimensional immunology data.** *Nat Rev Immunol.* Nature Publishing Group, a division of Macmillan Publishers Limited. 2016; **16**(7): 449–62.
PubMed Abstract | Publisher Full Text

Tang J, Liu J, Zhang M, *et al.*: **Visualizing Large-scale and High-dimensional Data.** *CoRR* abs/1602.00370; 2016.
Publisher Full Text

Van Gassen S, Callebaut B, Van Helden MJ, *et al.*: **FlowSOM: Using self-organizing maps for visualization and interpretation of cytometry data.** *Cytometry A.* 2015; **87**(7): 636–45.
PubMed Abstract | Publisher Full Text

van der Maaten L, Hinton G: **Visualizing high-dimensional data using t-sne.** *J Mach Learn Res.* 2008; 2579–2605.
Reference Source

van Unen V, Li N, Molendijk I, *et al.*: **Mass Cytometry of the Human Mucosal Immune System Identifies Tissue- and Disease-Associated Immune Subsets.** *Immunity.* 2016; **44**(5): 1227–39.
PubMed Abstract | Publisher Full Text

Wang B, Ramazzotti D, De Sano L, *et al.*: **SIMLR: A Tool for Large-Scale Single-Cell Analysis by Multi-Kernel Learning.** *bioRxiv.* Cold Spring Harbor Labs Journals. 2017.
Publisher Full Text

Wattenberg M, Viégas F, Johnson I: **How to Use t-SNE Effectively.** *Distill.* 2016.
Publisher Full Text

Weber LM, Robinson MD: **Comparison of clustering methods for high-dimensional single-cell flow and mass cytometry data.** *Cytometry A.* 2016; **89**(12): 1084–96.
PubMed Abstract | Publisher Full Text

Wilkerson MD, Hayes DN: **ConsensusClusterPlus: a class discovery tool with confidence assessments and item tracking.** *Bioinformatics.* 2010; **26**(12): 1572–3.
PubMed Abstract | Publisher Full Text | Free Full Text

Zhao K, Lu ZX, Park JW, *et al.*: **GLiMMPS: robust statistical model for regulatory variation of alternative splicing using RNA-seq data.** *Genome Biol.* BioMed Central Ltd, 2013; **14**(7): R74.
PubMed Abstract | Publisher Full Text | Free Full Text

Zunder ER, Finck R, Behbehani GK, *et al.*: **Palladium-based mass tag cell barcoding with a doublet-filtering scheme and single-cell deconvolution algorithm.** *Nat Protoc.* Nature Publishing Group. 2015; **10**(2): 316–33.
PubMed Abstract | Publisher Full Text | Free Full Text

Open Peer Review

Current Referee Status: ✓ ✓

Version 1

Referee Report 08 June 2017

doi:10.5256/f1000research.12553.r23051

✓ **Greg Finak** , **Raphael Gottardo**
Vaccine and Infectious Disease Division (VIDD), Fred Hutchinson Cancer Research Center, Seattle, WA, USA

Nowicka and colleagues present a detailed workflow for analyzing high dimensional cytometry data using open source tools within the Bioconductor framework.

The paper provides a clear path for analyzing high dimensional cytometry data, with biomarker discovery in mind, starting from raw data, through preprocessing, population discovery, annotation, and differential abundance analysis.

Two particular strengths of the proposed approach are i) the decision to use expert-guided merging of cell populations, and ii) the model-based differential abundance analysis of cell populations.

The proposed visualization and summaries of the data make i) straightforward to follow and justify, and adequate alternatives are provided and shown to perform equally well in instances where manual merging of many clusters would be cumbersome.

The modeling of cell population counts, rather than proportions, is an approach that we strongly support, and the use of logistic regression with mixed effects is a natural approach that is probably insufficiently appreciated by the community at large. That said, some of the methods proposed in the workflow have been in use in the vaccine development field for some time and should be appropriately cited. Specifically, in the section "Visual representation with tSNE", the authors promote coloring individual cells on a tSNE map by expression level, and later still, stratifying by condition (Fig. 11). We point the authors the article by Lin et al.[1], where a very similar approach, using bioconductor tools, is undertaken to identify and visualize polyfunctional Ag-specific T-cells.

The discussion of existing methodological approaches to identify cytometry biomarkers associated with outcome and the discussion of modeling cell counts in favor of modeling of proportions is important, but should also reference existing work in the vaccine development field. Our group has done substantial work in this area, developing count-based models for antigen-specific T-cell response to stimulation[ref-2,3], the latter of which identified a novel biomarker of infection risk in an HIV clinical trial.

While these methods do not account for covariates, they are relevant to the discussion since they utilize the Beta-binomial and Dirichlet-Multinomial distributions in a Bayesian formulation to handle over-dispersion due to subject-to-subject variability (an alternative to mixed effects modeling), and warrant mention here.

Some additional minor points: the citation of flowCore (p5) should reference the journal publication describing the software[4], since it is available, rather than the software vignette.

Finally, note that flowCore is not used for analysis (p4), which is this context we take to mean clustering or gating, but rather is an infrastructure package that will read, write and transform cytometry data, as well as defining gate objects.

The core infrastructure for actually performing data-driven gating in Bioconductor is implemented in packages like flowWorkspace (Finak G, Jiang M, Gottardo R. flowWorkspace: Infrastructure for representing and interacting with the gated cytometry. 2011.) and openCyto[5].

The citations above should be added and updated for completeness and clarity.

Other than the above, the article is scientifically sound and the conclusions are justified by the data.

References

1. Lin L, Frelinger J, Jiang W, Finak G, Seshadri C, Bart PA, Pantaleo G, McElrath J, DeRosa S, Gottardo R: Identification and visualization of multidimensional antigen-specific T-cell populations in polychromatic cytometry data.*Cytometry A*. 2015; **87** (7): 675-82 PubMed Abstract | Publisher Full Text
2. Finak G, McDavid A, Chattopadhyay P, Dominguez M, De Rosa S, Roederer M, Gottardo R: Mixture models for single-cell assays with applications to vaccine studies.*Biostatistics*. 2014; **15** (1): 87-101 PubMed Abstract | Publisher Full Text
3. Lin L, Finak G, Ushey K, Seshadri C, Hawn TR, Frahm N, Scriba TJ, Mahomed H, Hanekom W, Bart PA, Pantaleo G, Tomaras GD, Rerks-Ngarm S, Kaewkungwal J, Nitayaphan S, Pitisuttithum P, Michael NL, Kim JH, Robb ML, O'Connell RJ, Karasavvas N, Gilbert P, C De Rosa S, McElrath MJ, Gottardo R: COMPASS identifies T-cell subsets correlated with clinical outcomes.*Nat Biotechnol*. 2015; **33** (6): 610-6 PubMed Abstract | Publisher Full Text
4. Hahne F, LeMeur N, Brinkman RR, Ellis B, Haaland P, Sarkar D, Spidlen J, Strain E, Gentleman R: flowCore: a Bioconductor package for high throughput flow cytometry.*BMC Bioinformatics*. 2009; **10**: 106 PubMed Abstract | Publisher Full Text
5. Finak G, Frelinger J, Jiang W, Newell EW, Ramey J, Davis MM, Kalams SA, De Rosa SC, Gottardo R: OpenCyto: an open source infrastructure for scalable, robust, reproducible, and automated, end-to-end flow cytometry data analysis.*PLoS Comput Biol*. 2014; **10** (8): e1003806 PubMed Abstract | Publisher Full Text

Is the rationale for developing the new method (or application) clearly explained?
Yes

Is the description of the method technically sound?
Yes

Are sufficient details provided to allow replication of the method development and its use by others?
Yes

If any results are presented, are all the source data underlying the results available to ensure full reproducibility?
Yes

Are the conclusions about the method and its performance adequately supported by the findings presented in the article?
Yes

Competing Interests: No competing interests were disclosed.

We have read this submission. We believe that we have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Author Response 31 Oct 2017

Mark Robinson, University of Zurich, Switzerland

Thank you for taking the time to read and review our paper. Following the suggestion of the reviewer, we have incorporated the missing references, including the reference to Lin et al. [1] in the "Visual representation with tSNE" section, and references to MIMOSA [2] and COMPASS [3] methods in the "Discussion" section. We have also fixed references in the "Data preprocessing" section, including the reference to the flowCore package [4]. We have clarified that the packages that can be used for cell gating are flowWorkspace [5] and openCyto [6].

References

[1] Lin Lin, Jacob Frelinger, Wenxin Jiang, Greg Finak, Chetan Seshadri, Pierre-Alexandre Bart, Giuseppe Pantaleo, Julie McElrath, Steve DeRosa, and Raphael Gottardo. *Identification and visualization of multidimensional antigen-specific T-cell populations in polychromatic cytometry data*. **Cytometry Part A**, 87(7):675–682, 2015.
[2] Greg Finak, Andrew McDavid, Pratip Chattopadhyay, Maria Dominguez, Steve De Rosa, Mario Roederer, and Raphael Gottardo. *Mixture models for single-cell assays with applications to vaccine studies*. **Biostatistics**, 15(1):87–101, 2014.
[3] Lin Lin, Greg Finak, Kevin Ushey, Chetan Seshadri, Thomas R Hawn, Nicole Frahm, Thomas J Scriba, Hassan Mahomed, Willem Hanekom, Pierre-Alexandre Bart, Giuseppe Pantaleo, Georgia D Tomaras, Supachai Rerks-Ngarm, Jaranit Kaewkungwal, Sorachai Nitayaphan, Punnee Pitisuttithum, Nelson L Michael, Jerome H Kim, Merlin L Robb, Robert J O'Connell, Nicos Karasavvas, Peter Gilbert, Stephen C De Rosa, M Juliana McElrath, and Raphael Gottardo. *COMPASS identifies T-cell subsets correlated with clinical outcomes*. **Nat Biotech**, 33(6):610–616, jun 2015.
[4] Florian Hahne, Nolwenn LeMeur, Ryan R Brinkman, Byron Ellis, Perry Haaland, Deepayan Sarkar, Josef Spidlen, Errol Strain, and Robert Gentleman. *flowCore: a Bioconductor package for high throughput flow cytometry*. **BMC Bioinformatics**, 10(1):106, apr 2009.
[5] Greg Finak and Mike Jiang. *flowWorkspace: Infrastructure for representing and interacting with the gated cytometry*, 2011. R package version 3.24.4.
[6] Greg Finak, Jacob Frelinger, Wenxin Jiang, Evan W EW Newell, John Ramey, Mark MM Davis, SA Spyros a Kalams, SC Stephen C De Rosa, and Raphael Gottardo. OpenCyto: An *Open Source Infrastructure for Scalable, Robust, Reproducible, and Automated, End-to-End Flow Cytometry Data Analysis*. **PLoS Computational Biology**, 10(8):e1003806, 2014.

Competing Interests: No competing interests.

Referee Report 06 June 2017

doi:10.5256/f1000research.12553.r23055

✓ **Aaron T. L. Lun** [1], **John C. Marioni** [1,2,3]
[1] CRUK (Cancer Research UK) Cambridge Institute, University of Cambridge, Cambridge, UK
[2] EMBL-European Bioinformatics Institute (EMBL-EBI), Wellcome Genome Campus, Cambridge, UK
[3] Wellcome Trust Sanger Institute, Wellcome Genome Campus, Cambridge, UK

Nowicka et al. describe a comprehensive workflow for a multi-sample analysis of a mass cytometry data set. They provide methods and guidelines for data processing and quality control, clustering and interpretation/visualization of the clusters. They also describe the application of statistical methods for differential analyses within clusters. The article is clear and well-written, with some opportunities for improvement that we have listed below. Overall, this will be useful resource for people looking to use R/Bioconductor for cytometry data analysis.

MAJOR COMMENTS:

1. The principle of pooling samples prior to clustering is important to ensure that the clustering is blind to the sample labels, and thus does not bias the downstream statistical inferences. However, in data sets where the number of cells is highly variable across samples, larger samples may drive the final clustering result. The authors may consider downweighting cells from larger samples during the clustering procedure, to ensure that each sample contributes equally to the outcome.

2. In what scenarios is the NRS useful? In a mass cytometry experiment, the panel is explicitly designed to interrogate markers of interest, so outside of quality control it makes little sense to discard them in the analysis. Indeed, low variance contributions in PCA does not mean that the marker is not relevant, e.g., if it marks a small population.

3. If the expected number of cell types is not known in advance, how many metaclusters should be chosen? In very heterogeneous populations, it is easy to imagine that there may well be more than 20 distinct cell subpopulations.

4. In the discussion, the authors state that "Overall, we expect that as a general rule, including batch parameters (or other covariates) in the linear modeling largely mitigates the problem." This is true to some extent, but will not protect the clustering from batch effects. If the batch effect shifts the intensity distribution between batches, it is possible that a subpopulation in samples of one batch is clustered with the wrong subpopulation in samples of another batch. The counts or median intensities of the cluster are inherently compromised and cannot be fixed by blocking on the batch effect in the model. In other words; when testing for changes in abundance in mass cytometry, the cells are analogous to individual reads in a transcriptomics experiment, while the vector of intensities is analogous to the genic region in which reads are counted. If the batch effect is affecting the intensities, it is analogous to changes in the definition of the genes between batches.

5. The authors mention using observation weights to describe the uncertainty of the median intensities when testing for differential expression of markers. We note that we have also used this approach in cydar, and it seems to work well. For differential expression of markers within each

cluster, the differences in size between clusters are largely irrelevant - all else being equal, if clusters are small, the medians should be more variable, and this should be considered by the inference machinery when computing p-values.

MINOR COMMENTS:

- A mention should be made of the fact that fsApply combines the intensity matrices from all data sets; this was not obvious from the code.
- There are many ways to do hypothesis testing in GLMMs, with options ranging from Wald Z-tests, LRTs and parametric bootstrapping/MCMC. Some words on what glht actually does would be useful.
- Some of the figure captions could be explained in more detail. For example, the numbers and colouring of the entries of the heatmap in Figure 4 are presumably the median marker intensities, but this should be explicitly stated.
- Some minor typographical errors: " (Angerer et al., 2016))", "use the flowCore [package]."
- The PBMC data set is described as "12 different stimulation conditions", but presumably only one was actually used (BCR/FcR-XL). This could be clarified.

Is the rationale for developing the new method (or application) clearly explained?
Yes

Is the description of the method technically sound?
Yes

Are sufficient details provided to allow replication of the method development and its use by others?
Yes

If any results are presented, are all the source data underlying the results available to ensure full reproducibility?
Yes

Are the conclusions about the method and its performance adequately supported by the findings presented in the article?
Yes

Competing Interests: No competing interests were disclosed.

We have read this submission. We believe that we have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Author Response 31 Oct 2017

Mark Robinson, University of Zurich, Switzerland

Thank you for taking the time to read and review our paper.

MAJOR POINTS:

1. As far as we understand, downweighting of cells (e.g., from larger samples) is not currently possible with FlowSOM. We think that, in general, it may be difficult to incorporate down-weighting into existing clustering algorithms that are tailored for cytometry data, but indeed it is worth considering.
   One of the easy solutions to ensure that each sample contributes equally to the outcome could be down-sampling so that equal amount of cells from each sample is used in clustering. However, there are two main drawbacks of this strategy. First, a substantial amount of data (cells) may be removed from the analysis resulting in information loss. Second, during down-sampling, some of the rarer populations may become underrepresented or even skipped. Overall, it is also hard to know exactly what "drive the clustering" really means. We highlight these issues now in the "Discussion" section.
2. NRS can be used to define new panels as it was done in Levine et al. [1]. Indeed, when there is no need for redefining the panel, it can be used as a quality control step. We mention that now in the "Marker ranking based on the non-redundancy score" section.
   In Levine et al., the NRS score was used to identify a set of surface markers that is "sufficient" to detect the main clusters in the AML data. As the number of markers that can be measured is limited, they could use only 16 channels for surface markers, while the remaining 15 channels were used for signaling markers. Based on average NRS, they identified the 16 surface markers, among 42, that explained the highest amount of variance in their data. The final set was slightly redefined (two markers with high scores were excluded and two markers with low scores were included) based on the biological knowledge, which agrees with the reviewer' statement that low variance contributions in PCA do not mean that the marker is not relevant and vice versa. However, we think that NRS can still serve as a relevant guide in marker selection. The final set was then used in the panels in the following experiments.
3. In general, as rule of thumb, we would suggest setting the number of consensus clusters (parameter maxK in ConsensusClusterPlus) to at least twice the number of expected cell populations. As this number increases, it is necessary to also increase the size of the grid in the SOM step (parameters xdim and ydim in BuildSOM). It is not necessary to know the exact number of cell types but rather the upper boundary for this number and treat that as the expected number. We also propose a strategy of re-clustering, where first main cell types are identified and extracted and then reclustered in a secondary analysis. In the updated version of our workflow, we have also added a section called "Obtaining higher resolution" where we describe solutions using a higher amount of clusters for higher resolution.
4. Yes, we fully agree that including batch effects into the linear modeling does not fully protect the clustering from the batch effect. That is why it is important to account for batches already at the clustering step if possible, although this is itself still a rather open (methodological research) question. Current approaches rely on, for example, equalizing the dynamic range between batches for each marker (e.g. normalization to the 0-1 range, z-scores, quantile normalization), the use of warping functions to eliminate non-linear distortions (cydar [2]), or learning marker distribution shifts between the batches based on a manually gated reference cell type and using it to correct marker expression for the whole dataset (CellCnn [3]). A recent method called MASC [4] that, similarly to our workflow, employs mixed models for the differential abundance analysis deals with batches by identifying and excluding from the clustering analysis markers with high between-batch variability and poorly recorded

cells, such as cells with extreme expression values. One could also consider, batch-wise clustering and aggregation, but these strategies also require further study. However, the effectiveness of these approaches has not been sufficiently studied, yet. We still recommend including batch information in the differential analysis as it may further help to mitigate the problem.

5. Indeed, we agree that the size of clusters should be built into in the inference. This happens automatically in the differential abundance analysis, since we use (over-dispersed) logistic regression. In the differential marker expression analysis, where the medians are compared, one could account for the variability of medians calculated over clusters by assigning lower weights to clusters with lower cell counts. We have not done this in the current workflow, as we are still assessing the effect of it on the power and error control of the methods.

MINOR POINTS:

- We now mention that the fsApply function, by default, combines intensity matrices from all data sets.
- We have now added some text explaining that the glht function uses t-tests to test the hypothesis.
- We have now updated the figure captions, especially those that correspond to Figures 4, 6, 7, 8, 14, 28, 30, and 32.
- We have fixed the identified typos.
- We have updated the description of the PBMC dataset.

References

[1] Jacob H. Levine, Erin F. Simonds, Sean C. Bendall, Kara L. Davis, El-ad D. Amir, Michelle D. Tadmor, Oren Litvin, Harris G. Fienberg, Astraea Jager, Eli R. Zunder, Rachel Finck, Amanda L. Gedman, Ina Radtke, James R. Downing, Dana Pe’er, and Garry P. Nolan. *Data-Driven Phenotypic Dissection of AML Reveals Progenitor-like Cells that Correlate with Prognosis*. **Cell**, 162(1):184–97, jun 2015.
[2] Aaron T L Lun, Arianne C Richard, and John C Marioni. *Testing for differential abundance in mass cytometry data*. **Nat Meth**, 14(7):707–709, jul 2017.
[3] Eirini Arvaniti and Manfred Claassen. *Sensitive detection of rare disease-associated cell subsets via representation learning*. **Nature Communications**, 8:14825, apr 2017.
[4] Chamith Y Fonseka, Deepak A Rao, Nikola C Teslovich, Susan K Hannes, Kamil Slowikowski, Michael F Gurish, Laura T Donlin, Michael E Weinblatt, Elena M Massarotti, Jonathan S Coblyn, Simon M Helf- gott, Derrick J Todd, Vivian P Bykerk, Elizabeth W Karlson, Joerg Ermann, Yvonne C Lee, Michael B Brenner, and Soumya Raychaudhuri. *Reverse Association Of Single Cells To Rheumatoid Arthritis Accounting For Mixed Effects Identifies An Expanded CD27- HLA-DR+ Effector Memory CD4+ T Cell Population*. **bioRxiv**, 2017.

Competing Interests: No competing interests