Imaging mass cytometry analysis pipeline

Mick J.M. van Eijs et al.

2023-04-17

Contents

1. Setup

1.1. Loading packages and raw data

```
library(ggplot2)
library(RANN)
library(doParallel)
library(dplyr)
library(magrittr)
library(Rcpp)
library(sf)
library(stars)
library(raster)
library(devtools)
library(Rphenograph)
library(Rtsne)
library(stringi)
library(pheatmap)
library(caret)
library(PreProcess)
library(plyr)
library(popbio)
library(tidyverse)
library(MASS)
library(tiff)
library(raster)
library(matrixStats)
library(stats)
library(data.table)
library(cowplot)
library(RColorBrewer)
library(concaveman)
library(lemon)
library(ggpubr)
library(RVAideMemoire)
library(chisq.posthoc.test)
library(car)
library(nlme)
library(lme4)
library(lmerTest)
library(MuMIn)
library(viridis)
library(writexl)
library(readxl)
library(umap)
library(remotes)
library(neighbouRhood)
library(gplots)
library(reshape)
```

Load raw data file containing segmented cells and channel data per cell. This file contains the output from the MATISSE cell segmentation pipeline (Baars...Vercoulen, *BMC Biology* 2021;19:99.) and (Krijgsman...Vercoulen, *STAR Protocols* 2022;3:101034).

```
setwd("~")
setwd(".../Segmentation/")
load("Segmented_IMC_data.Rda")
```

```
raw_data <- FusedDF
```

1.2. Set parameter values

Below, parameters are set for further analysis.
- `bins` denotes the number of bins in all histograms;
- `upper` and `lower` denote respectively the upper and lower boundaries for linear scaling factors;
- `size` denotes the line size of thresholds in post-normalization histograms;
- The threshold for marker positivity is set at `times_sd` * standard deviation (SD) of the normal density function. SDs are estimated using the Full width at half maximum (FWHM) method ($FWHM \approx 2.36 * SD$) based on the left tail of the density function only, except for markers where the positive population is larger than the negative population. In the latter situation the right tail is used instead;
- `subset_markers` denotes markers for which a dummy variable is created to facilitate boolean cell type delineage;
- `subset_highly_expr` denotes markers where the positive population exceeds the negative population in size;
- Finally, `pts` represents all unique patients in the data set.

```
bins <- 100
upper <- 1.3
lower <- 0.7
size <- 0.4
times_sd <- 1.5
subset_markers <- c("CD56_Mean", "Ecad_Mean", "CD20_Mean",
    "CD45_Mean", "CD45RO_Mean", "CD45RA_Mean", "TCRd_Mean",
    "CD68_Mean", "CD14_Mean", "CD4_Mean", "CD8a_Mean",
    "CD3_Mean", "FoxP3_Mean", "FoxP3_Nucl_Mean", "CD11c_Mean")
subset_highly_expr <- c("CD45_Mean")
pts <- unique(raw_data$ImageNumber)
```

Below, the first five rows are shown for the first eight columns of `raw_data`:

ImageNumber	ROInr	Cell_Area	Nucl_Area	80ArAr_Mean	80ArAr_Median	80ArAr_Int	Ecad_Mean
ICI 1_C	1	80	48	2949.626	2931.490	117985.05	0.4495174
ICI 1_C	2	64	64	2922.958	2894.222	93534.66	0.1545192
ICI 1_C	3	108	68	2980.385	3003.128	160940.81	0.2732518
ICI 1_C	4	76	36	2952.897	2947.788	112210.09	0.4115321
ICI 1_C	5	156	94	2954.803	2957.262	230474.63	0.2417527

And the last three columns (129-131) of the first row, containing the centroid coordinates of the cytosol, geometrical data of the nuclear boundary and the nuclear centroid coordinates:

centroid_Cyto	geometry_Nucl	centroid_Nucl
174.3, 1524.5	176, 176, 175, 174, 174, 173, 172, 171, 170, 170, 169, 169, 170, 171, 171, 172, 173, 174, 175, 176, 177, 178, 178, 177, 177, 177, 176, 1522, 1521, 1521, 1521, 1522, 1522, 1522, 1522, 1522, 1523, 1523, 1524, 1524, 1524, 1525, 1525, 1525, 1525, 1525, 1525, 1525, 1525, 1524, 1524, 1523, 1522, 1522	173.750, 1523.333

Column names starting with a number are changed, because later on `ggplot2` will not be able to handle such column names.

```
names(raw_data)[5] <- paste("ArAr_Mean")
names(raw_data)[6] <- paste("ArAr_Median")
names(raw_data)[7] <- paste("ArAr_Int")

names(raw_data)[119] <- paste("Ir191_Mean")
names(raw_data)[120] <- paste("Ir191_Median")
names(raw_data)[121] <- paste("Ir191_Int")

names(raw_data)[122] <- paste("Ir193_Mean")
names(raw_data)[123] <- paste("Ir193_Median")
names(raw_data)[124] <- paste("Ir193_Int")

names(raw_data)[125] <- paste("Ir193_Nucl_Mean")
names(raw_data)[126] <- paste("Ir193_Nucl_Median")
names(raw_data)[127] <- paste("Ir193_Nucl_Int")
```

2. Data transformation and clean-up

2.1. Logarithmic transformation

Next, all columns containing channel data are natural-log transformed, while values that originally were 0 are kept as such by subtracting $log(0.01)$.

```
for (i in 5:127) {
    raw_data_log[, i] <- log(raw_data[, i] + 0.01) -
        log(0.01)
}
```

2.2. Deletion of manually annotated regions such as artefacts

Manually annotated regions are filtered in three steps.

Step 1: load CSV files with (x,y) coordinates for all manual annotations (55 in total across all subjects)

```
for (i in 1:55) {
    filename <- paste0("Anno", i)
    wd <- paste0("Manual_annotations/Anno", i, ".csv")
    assign(filename, read.csv(wd))
}
```

Step 2: invert y coordinates based on max y value per image

Manual annotations were created in Fiji (ImageJ), in which the upper left instead of the bottom left pixel equals y=0. Therefore, y-values must be inverted before they can be used with the `sf` package.

```
max_y1 <- 1531
# (...)
max_y23 <- 1291

Anno1$Y <- max_y1 - Anno1$Y
# (...)
Anno55$Y <- max_y23 - Anno55$Y
```

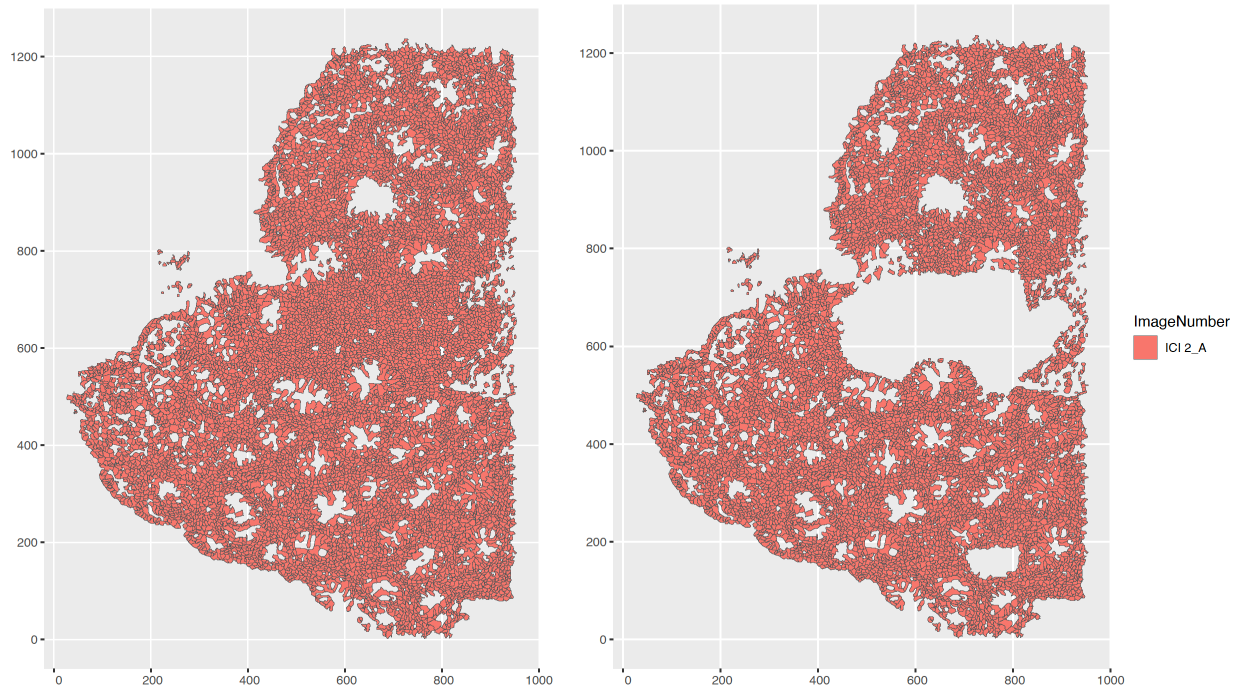
Step 3: Remove all single cells overlapping with manually annotated regions.

An example of the code used is shown for the first patient sample only.

```
Annot <- as.data.frame(Anno1, stringsAsFactors = T)
Annot_src <- st_as_sf(Annot, coords = c("X", "Y"))
Annot_res1 <- concaveman(Annot_src)

attach(raw_data_log)
data_subset1 <- raw_data_log[ImageNumber == "ICI 1_C",
    ]
geom_dataset1 <- st_as_sf(data_subset1)
intersect_pct1 <- st_intersection(geom_dataset1, st_make_valid(Annot_res1))
exclusion_list1 <- unique(intersect_pct1$ROInr)
data_subset1 <- data_subset1[!(data_subset1$ROInr %in%
    exclusion_list1), ]
data_subset1 <- st_as_sf(data_subset1)
pc1_2 <- ggplot() + geom_sf(data = data_subset1, aes(fill = ImageNumber),
    size = 0.1)
pdf("Check_subset1.pdf")
pc1_2
dev.off()
```

Below the segmented cells before (left) and after (right) removal of manually annotated regions is shown for the second patient sample.



The total number of single cells across all subjects was reduced from 215,293 to 184,975 after deletion of manual annotated regions. All individual patient sample images were then combined into one new data set.

```
newdata <- rbind(data_subset1, data_subset2, data_subset3,
    data_subset4, data_subset5, data_subset6, data_subset7,
    data_subset8, data_subset9, data_subset10, data_subset11,
    data_subset12, data_subset13, data_subset14, data_subset15,
    data_subset16, data_subset17, data_subset18, data_subset19,
    data_subset20, data_subset21, data_subset22, data_subset23)

print("Number of deleted cells over all samples is:")
nrow(raw_data_log) - nrow(newdata)
print("Number of cells in new data file is:")
nrow(newdata)
raw_data_log <- newdata
```

3. Data scaling

From here on, only mean channel expression per single cell is used in the analysis. These columns are stored in the variable `markers`:

```
markers <- names(raw_data_log)
markers <- grep("*Mean", markers, value = TRUE)
```

Graphical examples of channel signal across patient samples before and after scaling will be shown after the steps taken for scaling. First, for each channel, histograms are produced for each patient sample individually.

```
for (m in markers) {
    pre <- ggplot(raw_data_log, aes_string(x = m)) +
        geom_histogram(aes(color = "ImageNumber"),
            fill = "white", position = "identity",
            bins = bins) + facet_wrap(ImageNumber ~
        .)
    pdf(paste0("pre_", m, ".pdf"))
    plot(pre)
    dev.off()
}
```

Histograms per patient sample are then created for every marker, a density function is fit and the signal intensity value at the peak density (=mode) is retrieved.

```
pt_plot = list()  #list of histograms per patient (all cells) per marker
dens_peak = list()  #x value for peak density in histogram per patient (all cells) per marker
mean_marker = list()  #x value for mean peak density in histogram per marker (all cells)

for (m in markers) {
    for (i in pts) {
        pt_plot[[i]] <- ggplot(raw_data_log %>%
            filter(ImageNumber == i), aes_string(x = m)) +
            geom_histogram(aes(y = ..density..), fill = "white",
                position = "identity", bins = bins) +
            geom_density()
        d <- ggplot_build(pt_plot[[i]])$d
        dens_peak[[i]] <- d[[2]] %>%
            filter(y == max(y)) %>%
            .$x
    }

    # For each marker, compute mean x value of
    # peak density over all patient samples
    mean_marker[[m]] <- mean(unlist(dens_peak))
}
unlist(mean_marker)
```

Next, based on this signal intensity and the average of the signal intensities of all samples together, a patient- and channel-specific correction factor is calculated. Positive (negative) correction factors indicate that signal must be scaled up (down).

```
corr_factor = list()  #Create list with correction factors for each patient
corr_factors = data.frame()  #Create data frame to save patient-specific correction
# factors in for each channel

for (m in markers) {
    for (i in pts) {
        pt_plot[[i]] <- ggplot(raw_data_log %>%
            filter(ImageNumber == i), aes_string(x = m)) +
            geom_histogram(aes(y = ..density..), fill = "white",
                position = "identity", bins = bins) +
            geom_density()
        d <- ggplot_build(pt_plot[[i]])$d
```

```
        dens_peak[[i]] <- d[[2]] %>%
            filter(y == max(y)) %>%
            .$x
        corr_factor[[i]] <- mean_marker[[m]]/dens_peak[[i]]
        corr_factors[i, m] <- corr_factor[[i]]  #Here all patient-specific corr factors for
        # the mth marker are saved in the df
    }
}
```

The above results in a data frame with all correction factors. Below, the single cell expression values for each marker (or channel) m and each patient i are multiplied with the specific correction factor *(i,m)*.

```
for (m in markers) {
    for (i in pts) {
        if (corr_factors[i, m] > lower) {
            if (corr_factors[i, m] < upper) {
                raw_data_log$m[raw_data_log$ImageNumber ==
                  i] <- (raw_data_log$m[raw_data_log$ImageNumber ==
                  i]) * corr_factors[i, m]
                raw_data_log[[m]][ImageNumber == i] <- raw_data_log[[m]][ImageNumber ==
                  i] * corr_factors[i, m]
            }
        }
    }
}
```

Now, based on all *normalized and scaled* data, gating thresholds are defined for all markers (channels) to make a first selection of marker positivity. Intentionally, these thresholds were set at a too low (or too conservative) level, i.e., the group of cells with marker expression beyond the threshold will contain a large number of false-positives. The reason for this is that later on cell type annotation will be performed in two steps. In the first step, in which these "marker positivity thresholds" are used, candidate cell types are selected. Since this step only concerns candidates, a high number of false-positives will ensure that true-positives are as much as possible contained within the candidates. In the second step, the most likely cell type is allocated based on relative marker expression.

```
thresholds = list()
temp = data.frame()
minimums <- NULL
FWHM <- NULL
sd <- NULL

for (m in markers) {
    plot <- ggplot(raw_data_log, aes_string(x = m)) +
        geom_histogram(aes(y = ..density..), fill = "white",
            position = "identity", bins = bins) + geom_density()
    d <- ggplot_build(plot)$d
    dens_peak[[m]] <- d[[2]] %>%
        filter(y == max(y)) %>%
        .$x

    # Find minimums
    q <- density(raw_data_log[[m]])
    xmax <- q$x[q$y == max(q$y)]
```

```
    x1 <- q$x[q$x < xmax][which.min(abs(q$y[q$x < xmax] -
        max(q$y)/2))]
    FWHM[m] <- 2 * abs(xmax - x1)
    sd[m] <- FWHM[m]/2.36
    thresholds[m] <- as.numeric(dens_peak[m]) + as.numeric(sd[m] *
        times_sd)
}
```

For highly expressed markers where $population_{positive} > population_{negative}$, thresholds must be adjusted.

```
FWHM <- NULL
sd <- NULL

for (s in subset_highly_expr) {
    plot <- ggplot(raw_data_log, aes_string(x = s)) +
        geom_histogram(aes(y = ..density..), fill = "white",
            position = "identity", bins = bins) + geom_density()
    d <- ggplot_build(plot)$d
    dens_peak[[s]] <- d[[2]] %>%
        filter(y == max(y)) %>%
        .$x

    # Find minimums
    q <- density(raw_data_log[[s]])
    xmax <- q$x[q$y == max(q$y)]
    x2 <- q$x[q$x > xmax][which.min(abs(q$y[q$x > xmax] -
        max(q$y)/2))]
    FWHM[s] <- 2 * abs(xmax - x2)
    sd[s] <- FWHM[s]/2.36
    thresholds[s] <- as.numeric(dens_peak[s]) - as.numeric(sd[s] *
        times_sd)
}
```

Now, histograms after scaling can be created with *conservative* marker positivity thresholds.

```
for (m in markers) {
    post <- ggplot(raw_data_log, aes_string(x = m)) +
        geom_histogram(aes(color = "ImageNumber"),
            fill = "white", position = "identity",
            bins = bins) + geom_vline(xintercept = as.numeric(unlist(thresholds[m])),
        linetype = "longdash", color = "black", size = size) +
        facet_wrap(ImageNumber ~ .)
    pdf(paste0("post_", m, ".pdf"))
    plot(post)
    dev.off()
}
```

All post-scaling histograms have been visually checked. For all channel-sample combinations that have not been normalized correctly (because a given scaling factor was above `upper` or below `lower`), scaling factors are calculated manually. These correction factors are applied below, resulting in a *adjusted, normalized scaled* data set:

```
raw_data_log$Ecad_Mean[ImageNumber == "ICI 13_A"] <- raw_data_log$Ecad_Mean[ImageNumber ==
    "ICI 13_A"] * 1.6
raw_data_log$Ecad_Mean[ImageNumber == "ICI 18_A"] <- raw_data_log$Ecad_Mean[ImageNumber ==
    "ICI 18_A"] * 1.6
raw_data_log$Ecad_Mean[ImageNumber == "ICI 8_D"] <- raw_data_log$Ecad_Mean[ImageNumber ==
    "ICI 8_D"] * 1.3
raw_data_log$Ecad_Mean[ImageNumber == "ICI 5_C"] <- raw_data_log$Ecad_Mean[ImageNumber ==
    "ICI 5_C"] * 1.5
raw_data_log$Ecad_Mean[ImageNumber == "IBD 1_B"] <- raw_data_log$Ecad_Mean[ImageNumber ==
    "IBD 1_B"] * 1.33
raw_data_log$Ecad_Mean[ImageNumber == "ICI 2_A"] <- raw_data_log$Ecad_Mean[ImageNumber ==
    "ICI 2_A"] * 1.15
raw_data_log$CD68_Mean[ImageNumber == "IBD 1_B"] <- raw_data_log$CD68_Mean[ImageNumber ==
    "IBD 1_B"] * 1.85
raw_data_log$CD20_Mean[ImageNumber == "IBD 1_B"] <- raw_data_log$CD20_Mean[ImageNumber ==
    "IBD 1_B"] * 1.26
raw_data_log$CD45RA_Mean[ImageNumber == "IBD 1_B"] <- raw_data_log$CD45RA_Mean[ImageNumber ==
    "IBD 1_B"] * 1.5
raw_data_log$CD45RA_Mean[ImageNumber == "IBD 3_A"] <- raw_data_log$CD45RA_Mean[ImageNumber ==
    "IBD 3_A"] * 1.33
raw_data_log$CD45RA_Mean[ImageNumber == "ICI 12_B"] <- raw_data_log$CD45RA_Mean[ImageNumber ==
    "ICI 12_B"] * 1.5
raw_data_log$CD45RA_Mean[ImageNumber == "ICI 17_B"] <- raw_data_log$CD45RA_Mean[ImageNumber ==
    "ICI 17_B"] * 1.6
raw_data_log$PANHLA_Mean[ImageNumber == "IBD 1_B"] <- raw_data_log$PANHLA_Mean[ImageNumber ==
    "IBD 1_B"] * 2
raw_data_log$PANHLA_Mean[ImageNumber == "ICI 8_D"] <- raw_data_log$PANHLA_Mean[ImageNumber ==
    "ICI 8_D"] * 1.6
raw_data_log$PANHLA_Mean[ImageNumber == "ICI 5_C"] <- raw_data_log$PANHLA_Mean[ImageNumber ==
    "ICI 5_C"] * 1.6
raw_data_log$PANHLA_Mean[ImageNumber == "ICI 2_A"] <- raw_data_log$PANHLA_Mean[ImageNumber ==
    "ICI 2_A"] * 1.33
raw_data_log$PANHLA_Mean[ImageNumber == "ICI 17_B"] <- raw_data_log$PANHLA_Mean[ImageNumber ==
    "ICI 17_B"] * 1.6
raw_data_log$CTLA4_Mean[ImageNumber == "ICI 8_D"] <- raw_data_log$CTLA4_Mean[ImageNumber ==
    "ICI 8_D"] * 1.5
raw_data_log$RORgt_Mean[ImageNumber == "ICI 5_C"] <- raw_data_log$RORgt_Mean[ImageNumber ==
    "ICI 5_C"] * 1.3
raw_data_log$RORgt_Mean[ImageNumber == "ICI 8_D"] <- raw_data_log$RORgt_Mean[ImageNumber ==
    "ICI 8_D"] * 1.18
raw_data_log$RORgt_Mean[ImageNumber == "ICI 1_C"] <- raw_data_log$RORgt_Mean[ImageNumber ==
    "ICI 1_C"] * 1.16
raw_data_log$RORgt_Mean[ImageNumber == "IBD 3_A"] <- raw_data_log$RORgt_Mean[ImageNumber ==
    "IBD 3_A"] * 1.45
raw_data_log$RORgt_Nucl_Mean[ImageNumber == "ICI 5_C"] <- raw_data_log$RORgt_Nucl_Mean[ImageNumber ==
    "ICI 5_C"] * 1.45
raw_data_log$RORgt_Nucl_Mean[ImageNumber == "ICI 8_D"] <- raw_data_log$RORgt_Nucl_Mean[ImageNumber ==
    "ICI 8_D"] * 1.15
raw_data_log$RORgt_Nucl_Mean[ImageNumber == "ICI 1_C"] <- raw_data_log$RORgt_Nucl_Mean[ImageNumber ==
    "ICI 1_C"] * 1.13
raw_data_log$RORgt_Nucl_Mean[ImageNumber == "IBD 3_A"] <- raw_data_log$RORgt_Nucl_Mean[ImageNumber ==
    "IBD 3_A"] * 1.21
raw_data_log$CD45RO_Mean[ImageNumber == "ICI 14_B"] <- raw_data_log$CD45RO_Mean[ImageNumber ==
```

```
    "ICI 14_B"] * 1.33
raw_data_log$CD45RO_Mean[ImageNumber == "ICI 11_D"] <- raw_data_log$CD45RO_Mean[ImageNumber ==
    "ICI 11_D"] * 1.58
raw_data_log$CD45RO_Mean[ImageNumber == "ICI 12_B"] <- raw_data_log$CD45RO_Mean[ImageNumber ==
    "ICI 12_B"] * 1.33
raw_data_log$CD45RO_Mean[ImageNumber == "ICI 4_A"] <- raw_data_log$CD45RO_Mean[ImageNumber ==
    "ICI 4_A"] * 1.4
raw_data_log$CD45RO_Mean[ImageNumber == "ICI 8_D"] <- raw_data_log$CD45RO_Mean[ImageNumber ==
    "ICI 8_D"] * 1.33
raw_data_log$CD45RO_Mean[ImageNumber == "IBD 1_B"] <- raw_data_log$CD45RO_Mean[ImageNumber ==
    "IBD 1_B"] * 1.58
raw_data_log$TNFa_Mean[ImageNumber == "ICI 8_D"] <- raw_data_log$TNFa_Mean[ImageNumber ==
    "ICI 8_D"] * 1.5
raw_data_log$FoxP3_Mean[ImageNumber == "IBD 1_B"] <- raw_data_log$FoxP3_Mean[ImageNumber ==
    "IBD 1_B"] * 1.4
raw_data_log$FoxP3_Mean[ImageNumber == "ICI 5_C"] <- raw_data_log$FoxP3_Mean[ImageNumber ==
    "ICI 5_C"] * 1.4
raw_data_log$FoxP3_Nucl_Mean[ImageNumber == "IBD 1_B"] <- raw_data_log$FoxP3_Nucl_Mean[ImageNumber ==
    "IBD 1_B"] * 1.4
raw_data_log$FoxP3_Nucl_Mean[ImageNumber == "ICI 5_C"] <- raw_data_log$FoxP3_Nucl_Mean[ImageNumber ==
    "ICI 5_C"] * 1.4
raw_data_log$IL10_Mean[ImageNumber == "ICI 5_C"] <- raw_data_log$IL10_Mean[ImageNumber ==
    "ICI 5_C"] * 1.28
raw_data_log$IL10_Mean[ImageNumber == "ICI 8_D"] <- raw_data_log$IL10_Mean[ImageNumber ==
    "ICI 8_D"] * 1.13
raw_data_log$TIGIT_Mean[ImageNumber == "ICI 5_C"] <- raw_data_log$TIGIT_Mean[ImageNumber ==
    "ICI 5_C"] * 1.3
raw_data_log$TIGIT_Mean[ImageNumber == "ICI 8_D"] <- raw_data_log$TIGIT_Mean[ImageNumber ==
    "ICI 8_D"] * 1.13
raw_data_log$CD56_Mean[ImageNumber == "IBD 1_B"] <- raw_data_log$CD56_Mean[ImageNumber ==
    "IBD 1_B"] * 1.5
raw_data_log$CD56_Mean[ImageNumber == "ICI 1_C"] <- raw_data_log$CD56_Mean[ImageNumber ==
    "ICI 1_C"] * 0.8
raw_data_log$CD56_Mean[ImageNumber == "ICI 5_C"] <- raw_data_log$CD56_Mean[ImageNumber ==
    "ICI 5_C"] * 1.2
raw_data_log$IL17_Mean[ImageNumber == "ICI 5_C"] <- raw_data_log$IL17_Mean[ImageNumber ==
    "ICI 5_C"] * 1.4
raw_data_log$Ki67_Mean[ImageNumber == "IBD 1_B"] <- raw_data_log$Ki67_Mean[ImageNumber ==
    "IBD 1_B"] * 5
raw_data_log$Ki67_Mean[ImageNumber == "ICI 17_B"] <- raw_data_log$Ki67_Mean[ImageNumber ==
    "ICI 17_B"] * 1.5
raw_data_log$Ki67_Mean[ImageNumber == "IBD 8_D"] <- raw_data_log$Ki67_Mean[ImageNumber ==
    "IBD 8_D"] * 1.5
raw_data_log$Ki67_Mean[ImageNumber == "IBD 3_A"] <- raw_data_log$Ki67_Mean[ImageNumber ==
    "IBD 3_A"] * 1.5
raw_data_log$Ki67_Mean[ImageNumber == "ICI 13_A"] <- raw_data_log$Ki67_Mean[ImageNumber ==
    "ICI 13_A"] * 1.3
raw_data_log$Ki67_Mean[ImageNumber == "ICI 18_A"] <- raw_data_log$Ki67_Mean[ImageNumber ==
    "ICI 18_A"] * 1.3
raw_data_log$Ki67_Mean[ImageNumber == "ICI 12_B"] <- raw_data_log$Ki67_Mean[ImageNumber ==
    "ICI 12_B"] * 1.3
raw_data_log$Ki67_Mean[ImageNumber == "ICI 6_D"] <- raw_data_log$Ki67_Mean[ImageNumber ==
    "ICI 6_D"] * 1.6
```

```
raw_data_log$TCRd_Mean[ImageNumber == "ICI 5_C"] <- raw_data_log$TCRd_Mean[ImageNumber ==
    "ICI 5_C"] * 1.3
raw_data_log$TCRd_Mean[ImageNumber == "IBD 1_B"] <- raw_data_log$TCRd_Mean[ImageNumber ==
    "IBD 1_B"] * 1.3
raw_data_log$CD103_Mean[ImageNumber == "IBD 1_B"] <- raw_data_log$CD103_Mean[ImageNumber ==
    "IBD 1_B"] * 1.4
raw_data_log$CD3_Mean[ImageNumber == "IBD 1_B"] <- raw_data_log$CD3_Mean[ImageNumber ==
    "IBD 1_B"] * 1.5
raw_data_log$CD45_Mean[ImageNumber == "IBD 1_B"] <- raw_data_log$CD45_Mean[ImageNumber ==
    "IBD 1_B"] * 1.4
raw_data_log$GzmB_Mean[ImageNumber == "IBD 1_B"] <- raw_data_log$GzmB_Mean[ImageNumber ==
    "IBD 1_B"] * 1.5
raw_data_log$ICOS_Mean[ImageNumber == "IBD 1_B"] <- raw_data_log$ICOS_Mean[ImageNumber ==
    "IBD 1_B"] * 1.4
raw_data_log$IFNg_Mean[ImageNumber == "IBD 1_B"] <- raw_data_log$IFNg_Mean[ImageNumber ==
    "IBD 1_B"] * 1.25
raw_data_log$IL12Rb2_Mean[ImageNumber == "ICI 8_D"] <- raw_data_log$IL12Rb2_Mean[ImageNumber ==
    "ICI 8_D"] * 1.17
raw_data_log$pS6_Mean[ImageNumber == "IBD 1_B"] <- raw_data_log$pS6_Mean[ImageNumber ==
    "IBD 1_B"] * 1.36
raw_data_log$pS6_Mean[ImageNumber == "ICI 8_D"] <- raw_data_log$pS6_Mean[ImageNumber ==
    "ICI 8_D"] * 1.33
raw_data_log$pS6_Mean[ImageNumber == "ICI 1_C"] <- raw_data_log$pS6_Mean[ImageNumber ==
    "ICI 1_C"] * 1.28
raw_data_log$pS6_Mean[ImageNumber == "ICI 13_A"] <- raw_data_log$pS6_Mean[ImageNumber ==
    "ICI 13_A"] * 1.33
raw_data_log$pS6_Mean[ImageNumber == "ICI 18_A"] <- raw_data_log$pS6_Mean[ImageNumber ==
    "ICI 18_A"] * 1.36
```

After manually corrected scaling, histograms are again created to visually confirm correct scaling. By example, all patient-specific histograms for the E-cadherin channel are shown before and after scaling, and the final result after manual adjustments based on visual inspections.

```
for (m in markers) {
    post <- ggplot(raw_data_log, aes_string(x = m)) +
        geom_histogram(aes(color = "ImageNumber"),
            fill = "white", position = "identity",
            bins = bins) + geom_vline(xintercept = as.numeric(unlist(thresholds[m])),
        linetype = "longdash", color = "black", size = size) +
        facet_wrap(ImageNumber ~ .)
    pdf(paste0("post_manual_corr", m, ".pdf"))
    plot(post)
    dev.off()
}
```

Because multiple, partly random, technical and patient-related factors are assumed to contribute to noise, correction factors are expected to be normally distributed. This is checked with visualizations that have been included in the supplementary materials of our paper. The Excel file loaded below contains all correction factors that have *actually* been used (thus, including manually adjusted correction factors as explained earlier).

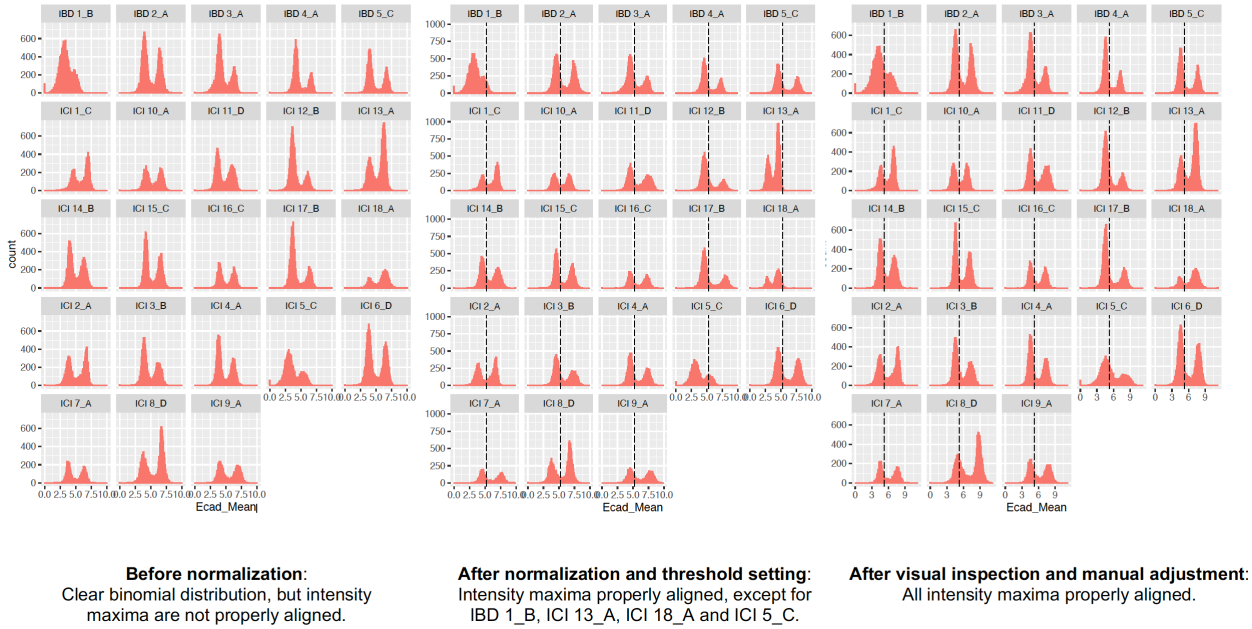


Figure 1: Example scaling steps for E-cadherin channel

```
manual_adj <- read_excel("Output/corr_factors_manual_adj.xlsx")
manual_adj <- as.data.frame(manual_adj)
rownames(manual_adj) <- manual_adj$patient
manual_adj_long <- melt(manual_adj)
manual_patient <- as.data.frame(t(manual_adj))
manual_patient_long <- melt(manual_patient)
plot_per_marker <- ggplot(manual_adj_long, aes(variable,
    value)) + geom_boxplot(outlier.shape = NA) + theme(axis.text.x = element_text(angle = 90,
    vjust = 0.5, hjust = 1))
plot_per_marker
hist_per_marker <- ggplot(manual_adj_long, aes(value,
    fill = variable)) + geom_histogram(bins = 40)
hist_per_marker

plot_per_patient <- ggplot(manual_patient_long, aes(variable,
    value)) + geom_boxplot(outlier.shape = NA) + theme(axis.text.x = element_text(angle = 90,
    vjust = 0.5, hjust = 1))
plot_per_patient
hist_per_patient <- ggplot(manual_patient_long, aes(value,
    fill = variable)) + geom_histogram(bins = 40)
hist_per_patient
```

4. Lineage determination

4.1. Lineage marker positivity

Dummy variables for marker positivity are created for a specified subset of markers with lineage-defining markers. For each cell, it is noted “positive” for a given marker if mean expression level exceeds the threshold.

Please recall that the group of cells labeled "positive" contains a relatively large fraction of false-positives.

```
for (i in 1:nrow(raw_data_log)) {
    for (n in subset_markers) {
        variable <- paste0("pos_", n)
        raw_data_log$variable <- NULL
        if (raw_data_log[[i, n]] > as.numeric(unlist(thresholds[n]))) {
            raw_data_log[i, variable] <- 1
        }
    }
}
raw_data_log[is.na(raw_data_log)] <- 0
```

Boolean expressions are created for the cell types of interest. For key markers (one key marker for each cell type) a variable is created in which the rank across all cells can be registered. Thus, the cell with highest mean expression of key marker x will be ranked first (highest) in `x_rank`, while the cell with lowest expression of x across all cells will be ranked last (lowest) in `x_rank`.

```
raw_data_log$CD4 <- NULL
raw_data_log$CD4_rank <- NULL
raw_data_log$CD8 <- NULL
raw_data_log$CD8_rank <- NULL
raw_data_log$B <- NULL
raw_data_log$CD20_rank <- NULL
raw_data_log$Treg <- NULL
raw_data_log$FoxP3_rank <- NULL
raw_data_log$gdT <- NULL
raw_data_log$gdT_rank <- NULL
raw_data_log$macrophage <- NULL
raw_data_log$macrophage_rank <- NULL
raw_data_log$DC <- NULL
raw_data_log$DC_rank <- NULL
raw_data_log$Epi <- NULL
raw_data_log$Ecad_rank <- NULL
raw_data_log$Other_immune <- NULL
raw_data_log$Other_immune_rank <- NULL
non_class_cell <- NULL  #miscellaneous group for non-classified cells

attach(raw_data_log)
# Create candidate cell types for all cells
raw_data_log$CD4 <- ifelse((pos_CD45_Mean == 1 | pos_CD45RA_Mean ==
    1 | pos_CD45RO_Mean == 1) & pos_CD3_Mean == 1 &
    pos_CD4_Mean == 1 & pos_FoxP3_Mean == 0, 1, 0)  #Excluding pos_FoxP3 is
# necessary to prevent fully overlapping Tregs
# with CD4+ T cells.
raw_data_log$CD8 <- ifelse((pos_CD45_Mean == 1 | pos_CD45RA_Mean ==
    1 | pos_CD45RO_Mean == 1) & pos_CD3_Mean == 1 &
    pos_CD8a_Mean == 1, 1, 0)
raw_data_log$B <- ifelse((pos_CD45_Mean == 1 | pos_CD45RA_Mean ==
    1 | pos_CD45RO_Mean == 1) & pos_CD3_Mean == 0 &
    pos_CD20_Mean == 1, 1, 0)
raw_data_log$Treg <- ifelse((pos_CD45_Mean == 1 | pos_CD45RA_Mean ==
    1 | pos_CD45RO_Mean == 1) & pos_CD3_Mean == 1 &
    pos_CD4_Mean == 1 & pos_FoxP3_Mean == 1, 1, 0)
```

```
raw_data_log$gdT <- ifelse((pos_CD45_Mean == 1 | pos_CD45RA_Mean ==
    1 | pos_CD45RO_Mean == 1) & pos_TCRd_Mean == 1,
    1, 0)
raw_data_log$Epi <- ifelse((pos_CD45_Mean == 0 & pos_CD45RA_Mean ==
    0 & pos_CD45RO_Mean == 0) & pos_Ecad_Mean == 1,
    1, 0)
raw_data_log$Other_immune <- ifelse((pos_CD45_Mean ==
    1 | pos_CD45RA_Mean == 1 | pos_CD45RO_Mean == 1) &
    pos_Ecad_Mean == 0 & pos_CD20_Mean == 0 & pos_CD3_Mean ==
    0 & pos_CD4_Mean == 0 & pos_CD8a_Mean == 0, 1,
    0)
raw_data_log$macrophage <- ifelse(pos_CD14_Mean ==
    1 & pos_CD68_Mean == 1 & pos_CD3_Mean == 0 & pos_CD20_Mean ==
    0, 1, 0)
raw_data_log$DC <- ifelse(pos_CD11c_Mean == 1 & pos_CD68_Mean ==
    0 & pos_CD3_Mean == 0 & pos_CD20_Mean == 0, 1,
    0)
raw_data_log$DP_im_epi <- ifelse(pos_CD45_Mean == 1 &
    pos_Ecad_Mean == 1, 1, 0)
raw_data_log$DN_im_epi <- ifelse(pos_CD45_Mean == 0 &
    pos_Ecad_Mean == 0, 1, 0)
raw_data_log$cell_type <- NULL

# Create ranks based on key lineage marker for
# all cells
raw_data_log$CD4_rank <- rank(raw_data_log$CD4_Mean)
raw_data_log$CD8_rank <- rank(raw_data_log$CD8a_Mean)
raw_data_log$B_rank <- rank(raw_data_log$CD20_Mean)
raw_data_log$Other_immune_rank <- max(c(rank(raw_data_log$CD45_Mean),
    rank(raw_data_log$CD45RO_Mean), rank(raw_data_log$CD45RA_Mean)))
raw_data_log$Treg_rank <- rank(raw_data_log$FoxP3_Mean)
raw_data_log$gdT_rank <- rank(raw_data_log$TCRd_Mean)
raw_data_log$Epi_rank <- rank(raw_data_log$Ecad_Mean)
raw_data_log$macrophage_rank <- rank(raw_data_log$CD68_Mean)
raw_data_log$DC_rank <- rank(raw_data_log$CD11c_Mean)
```

For all cells, marker-specific ranks are set to 0 (lowest by definition) if the mean expression level is below the positivity threshold (thus considered "negative"). Since thresholds are set at a "too low" level, the true-negative rate among cells with expression levels below this threshold is very high.

```
ranks <- c("CD4_rank", "CD8_rank", "B_rank", "Treg_rank",
    "gdT_rank", "Other_immune_rank", "Epi_rank", "macrophage_rank",
    "DC_rank")
names_used <- c("ROInr")
names_used <- c(names_used, names(raw_data_log)[names(raw_data_log) %in%
    ranks])

for (i in 1:nrow(raw_data_log)) {
    if (raw_data_log$CD4[[i]] == 0) {
        raw_data_log$CD4_rank[[i]] <- 0
    }
    if (raw_data_log$CD8[[i]] == 0) {
        raw_data_log$CD8_rank[[i]] <- 0
```

```
    }
    if (raw_data_log$B[[i]] == 0) {
        raw_data_log$B_rank[[i]] <- 0
    }
    if (raw_data_log$Treg[[i]] == 0) {
        raw_data_log$Treg_rank[[i]] <- 0
    }
    if (raw_data_log$gdT[[i]] == 0) {
        raw_data_log$gdT_rank[[i]] <- 0
    }
    if (raw_data_log$Epi[[i]] == 0) {
        raw_data_log$Epi_rank[[i]] <- 0
    }
    if (raw_data_log$Other_immune[[i]] == 0) {
        raw_data_log$Other_immune_rank[[i]] <- 0
    }
    if (raw_data_log$macrophage[[i]] == 0) {
        raw_data_log$macrophage_rank[[i]] <- 0
    }
    if (raw_data_log$DC[[i]] == 0) {
        raw_data_log$DC_rank[[i]] <- 0
    }
}
```

4.2. Lineage definition

As a last step, the key marker with highest rank (indicating the most likely cell type) is determined for all cells. Finally, the most likely cell type is determined. If none of the key markers is ranked highest (indicating that all key marker ranks are equal to 0, because none of the marker expression levels exceeded the threshold), the cell is labeled as "non_classified".

```
names_used2 <- c(names(raw_data_log)[names(raw_data_log) %in%
    ranks])
raw_data_log$Max_rank <- rowMaxs(as.matrix(st_drop_geometry(raw_data_log[,
    names_used2])))

raw_data_log$cell_type <- ifelse(rowSums(st_drop_geometry(raw_data_log[,
    names_used2])) == 0, "Non_classified", (raw_data_log$cell_type <- ifelse(Max_rank ==
    CD4_rank, "CD4_Tcell", (raw_data_log$cell_type <- ifelse(Max_rank ==
    CD8_rank, "CD8_Tcell", (raw_data_log$cell_type <- ifelse(Max_rank ==
    B_rank, "B_cell", (raw_data_log$cell_type <- ifelse(Max_rank ==
    gdT_rank, "gd_Tcell", (raw_data_log$cell_type <- ifelse(Max_rank ==
    Treg_rank, "Treg", (raw_data_log$cell_type <- ifelse(Max_rank ==
    Other_immune_rank, "Other_immune_cell", (raw_data_log$cell_type <- ifelse(Max_rank ==
    macrophage_rank, "Macrophage", (raw_data_log$cell_type <- ifelse(Max_rank ==
    DC_rank, "Dendritic_cell", (raw_data_log$cell_type <- ifelse(Max_rank ==
    Epi_rank, "Epithelial_cell", "Non_classified"))))))))))))))))))))
```

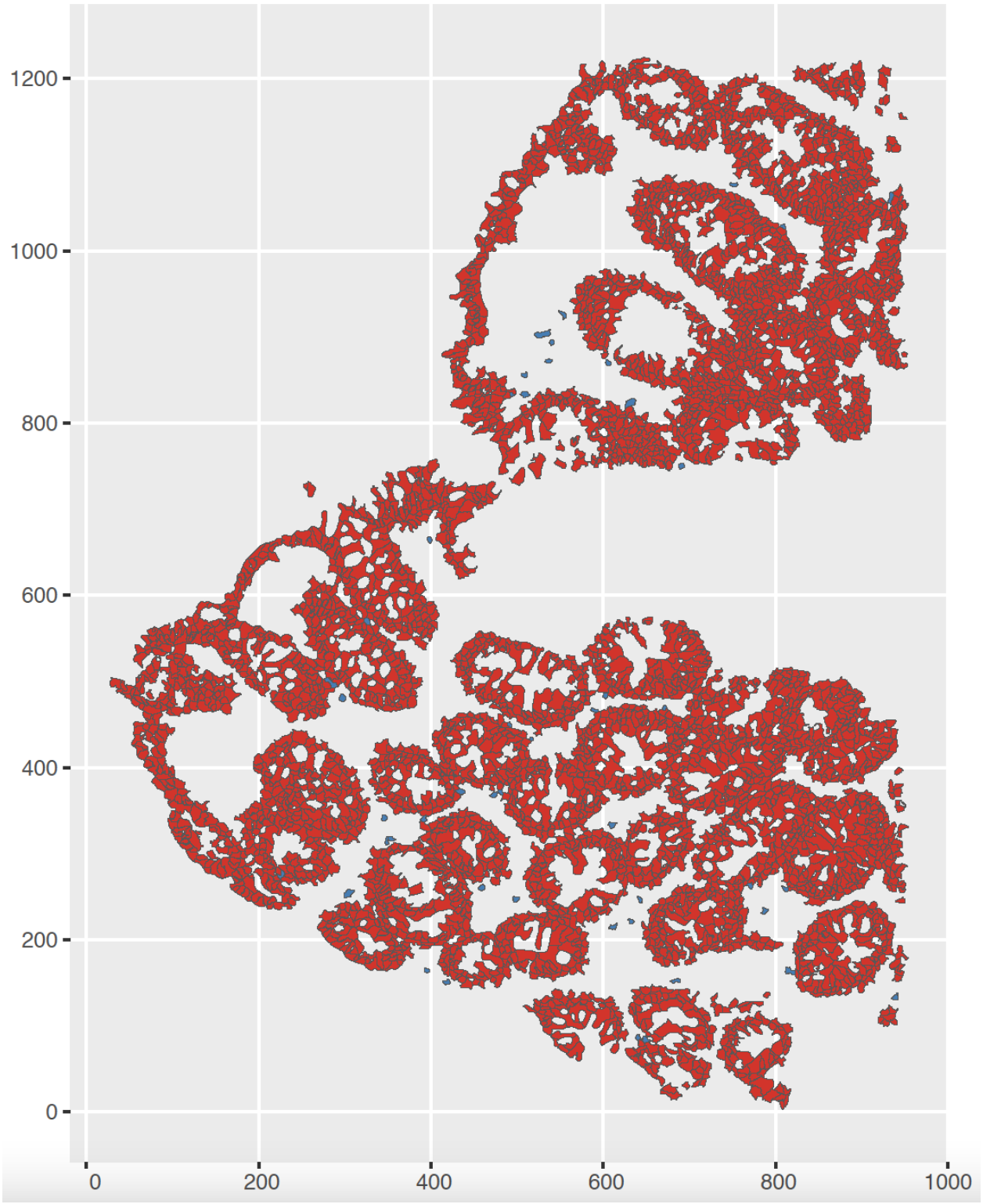
Lastly, a variable is created that indicates for every cell if it is in epithelial or lamina propria space. Masks covering epithelial space were created in Fiji (ImageJ) by applying Gaussian blur to pixel E-cadherin signal. Then, the (x,y) coordinates describing the edges of these masks were extracted from Fiji using a binary color filter.

```
epi_list <- NULL
for (i in seq_along(pts)) {
    filename <- paste0("Epi", i)
    wd <- paste0("Epi/Epi", i, ".csv")
    assign(filename, read.csv(wd))
    maxy <- get(paste0("max_y", i))
    Epi <- get(paste0("Epi", i))
    Epi$Y <- maxy - Epi$Y
    Epit <- as.data.frame(Epi, stringsAsFactors = T)
    Epit_src <- st_as_sf(Epit, coords = c("X", "Y"))

    epi_subset <- raw_data_log[ImageNumber == pts[i],
        ]
    geom_epi <- st_as_sf(epi_subset)
    intersect_epi <- st_intersection(geom_epi, st_make_valid(Epit_src))
    epi_list <- unique(intersect_epi$ROInr)
    name <- paste0("epi_list", i)
    assign(name, epi_list)
    print(paste0("Creating epi-list completed for: ",
        i, " from 23 samples."))
}

# Create column with epithelial status
# (1=epithelium, 0=LPL for all cells)
raw_data_log$Epithelium <- 0
raw_data_log <- as.data.table(raw_data_log)
attach(raw_data_log)
for (i in seq_along(pts)) {
    epi_list <- get(paste0("epi_list", i))
    image <- pts[i]
    raw_data_log[ImageNumber == image & ROInr %in%
        epi_list, Epithelium := 1]
    print(paste0("Assigning epi status completed for: ",
        i, " from 23 samples."))
}
table(raw_data_log$Epithelium)
```

As an example, below all cells within the epithelial space are displayed in red for patient 2. These cells will contain a large percentage of epithelial cells, but also other (immune) cell types as follows from Fig. 2D in our



paper.
The data set that resulted from this step will now proceed to single-cell data analysis.

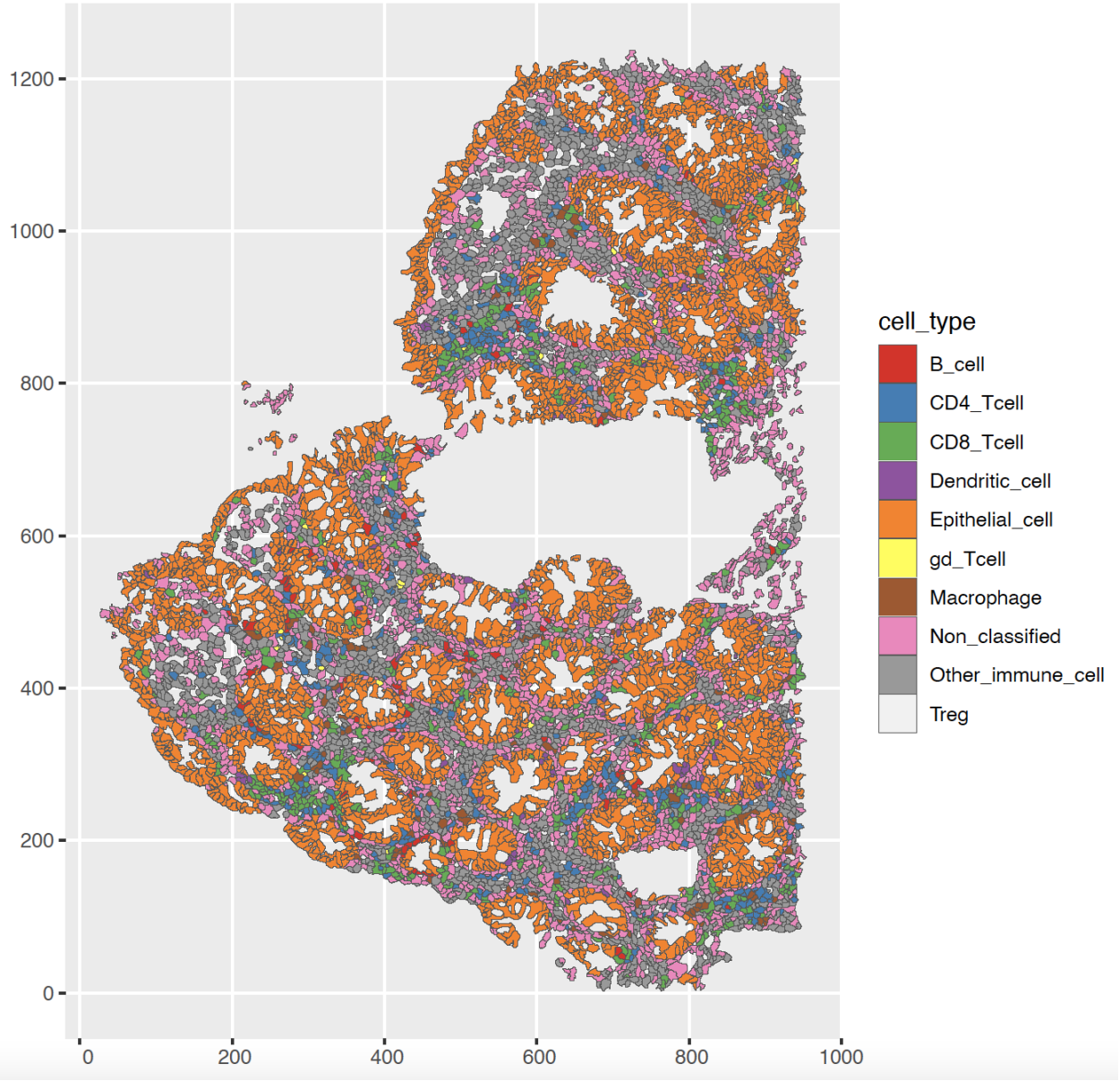
5. Single cell data analysis

5.1. Overall cell abundance and quality checks

Pseudo-color plots are created for all patient samples, displaying abundance and spatial distribution of cell types.

```
for (i in pts) {
    data_subset <- raw_data_log[ImageNumber == i, ]
    pc <- ggplot() + scale_fill_brewer(palette = "Set1") +
        geom_sf(data = data_subset, aes(fill = cell_type),
            size = 0.1)
    pdf(paste0("pseudocolor_", i, ".pdf"))
    plot(pc)
    dev.off()
}
```

As an example, below is the pseudo-color plot for patient 2.



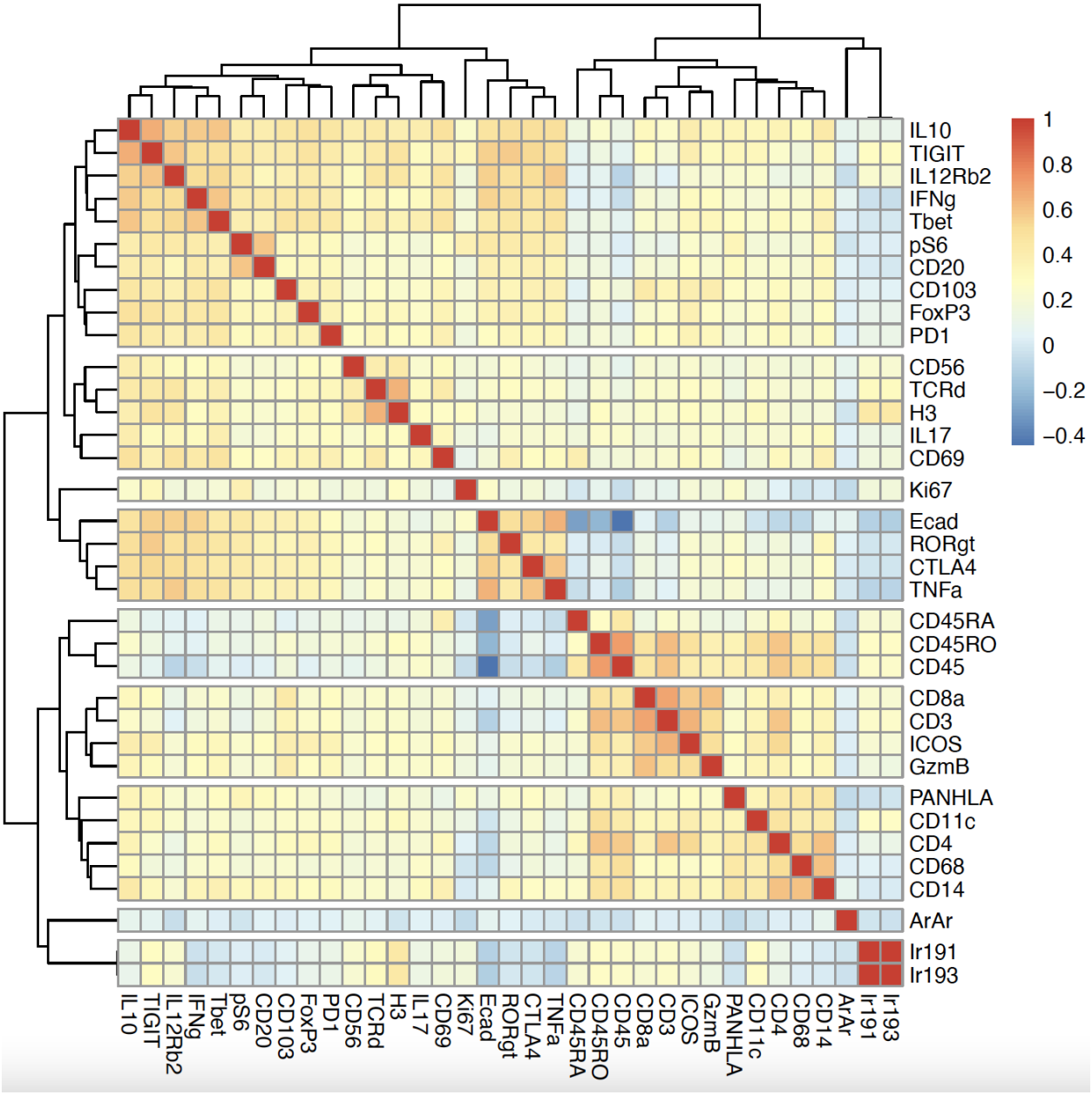
Next, we check cross-correlation of all markers with each other within the same cell. Biologically related markers such as granzyme B and CD8 are expected to be correlated within cells.

```
corr_check <- raw_data_log %>%
    select(c(ImageNumber, ROInr, Epithelium) | (contains("Mean") &
        !contains("Nucl")))
corr_check <- corr_check[, -(39:52)]
n_markers <- ncol(corr_check) - 3
corr_check$r_name <- paste0(corr_check$ImageNumber,
    "_", corr_check$ROInr)
corr_check$name <- gsub("\\s+", "", corr_check$r_name)
corr_check <- column_to_rownames(corr_check, var = "name")
corr_check_LPL <- corr_check %>%
    filter(Epithelium == 0)
corr_check_EPI <- corr_check %>%
    filter(Epithelium == 1)
```

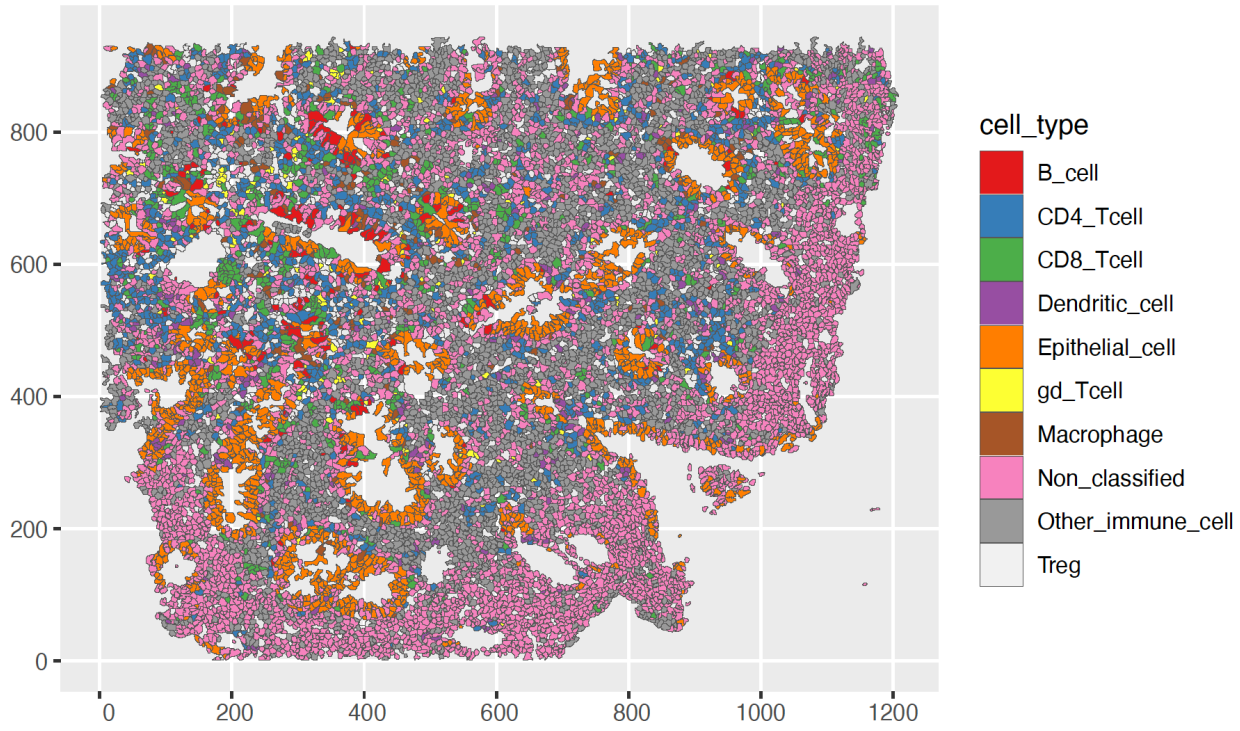
```
corr_check <- corr_check[, -c(1, 2, 3, 39)]
corr_check_LPL <- corr_check_LPL[, -c(1, 2, 3, 39)]
corr_check_EPI <- corr_check_EPI[, -c(1, 2, 3, 39)]

attach(corr_check)
corr_table <- data.frame(matrix(0, ncol = n_markers,
    nrow = n_markers))
colnames(corr_table) <- colnames(corr_check)
rownames(corr_table) <- colnames(corr_check)
for (i in 1:n_markers) {
    c1 <- colnames(corr_check[i])
    for (j in 1:n_markers) {
        c2 <- colnames(corr_check[j])
        corr_table[i, j] <- cor.test(get(c1), get(c2))[[4]]
    }
}
set.seed(42)
rownames(corr_table) <- sub("_.*", "", colnames(corr_table))
colnames(corr_table) <- sub("_.*", "", colnames(corr_table))
corr_heatmap <- pheatmap(corr_table, scale = "none",
    cluster_cols = T, cluster_rows = T, cutree_rows = 9)
corr_heatmap

save_pheatmap_pdf <- function(x, filename, width = 7,
    height = 7) {
    stopifnot(!missing(x))
    stopifnot(!missing(filename))
    pdf(filename, width = width, height = height)
    grid::grid.newpage()
    grid::grid.draw(x$gtable)
    dev.off()
}
save_pheatmap_pdf(corr_heatmap, "heatmap_correlation_channels.pdf")
```



All pseudo-color plots were visually inspected to confirm that spatial distribution and abundance of various immune cells were biologically likely. Based on this, we removed the sample from one patient with ulcerative colitis from the data set, because even after data clean-up, normalization and scaling the vast majority of cells could not be classified:



After removal of this sample, 172,930 single cells remained for analysis.

```
exp_data <- raw_data_log[!(raw_data_log$ImageNumber ==
    "IBD 1_B"), ]
attach(exp_data)
```

5.2. Merging single cell data with clinical metadata

Clinical metadata are loaded.

```
clin_data <- read.table("Manual_annotations/Data_PA_scoring.txt",
    sep = "\t", header = TRUE)
clin_data <- as.data.frame(clin_data)
merged_data <- merge(exp_data, clin_data, by = "ImageNumber")
```

5.3. UMAP dimensionality reduction

Clustering of single cells is displayed by UMAP visualizations and overlays are created for 1) cell type labels assigned in a supervised manner and 2) patient sample.

```
set.seed(42)
merged_data$Cell_ID <- paste0(merged_data$ImageNumber,
    merged_data$ROInr)
rownames(merged_data) <- merged_data$Cell_ID
sub_umap <- merged_data[, grepl("Mean", names(merged_data))]
sub_umap <- sub_umap[, !grepl("pos", names(sub_umap))]
sub_umap <- sub_umap[, !grepl("Nucl", names(sub_umap))]
subset_umap <- sub_umap[sample(1:nrow(sub_umap), 0.1 *
```

```
    nrow(sub_umap), replace = F), ]

umap_analysis <- umap(subset_umap)

subset_umap$ID <- 1
subset_umap <- subset_umap %>%
    mutate(ID = row_number())
subset_umap$Cell_ID <- rownames(subset_umap)
umap_analysis$layout <- as.data.frame(umap_analysis$layout)
colnames(umap_analysis$layout) <- c("UMAP1", "UMAP2")
umap_coord <- umap_analysis$layout
umap_coord$ID <- 1
umap_coord <- umap_coord %>%
    mutate(ID = row_number())
umap_merged <- merge(subset_tsne, umap_coord, by = c("ID"))
umap_coordinates <- umap_merged[, c(37:39)]
umap_meta_merged <- merge(merged_data, umap_coordinates,
    by = c("Cell_ID"), all = T)
umap_meta_merged <- umap_meta_merged %>%
    drop_na(UMAP1)

get_density <- function(x, y, ...) {
    dens <- MASS::kde2d(x, y, ...)
    ix <- findInterval(x, dens$x)
    iy <- findInterval(y, dens$y)
    ii <- cbind(ix, iy)
    return(dens$z[ii])
}

umap_meta_merged$density <- get_density(umap_meta_merged$UMAP1,
    umap_meta_merged$UMAP2, n = 100)

umap_density <- ggplot(data = umap_meta_merged, aes(UMAP1,
    UMAP2)) + geom_point(aes(color = density), size = 1) +
    scale_color_viridis() + theme(legend.position = "right",
    axis.title = element_blank(), axis.text = element_blank(),
    axis.ticks = element_blank()) + coord_fixed(ratio = 1)
pdf("umap_density.pdf")
plot(umap_density)
dev.off()

umap_cellType <- ggplot(data = umap_meta_merged, aes(UMAP1,
    UMAP2)) + geom_point(aes(color = cell_type), size = 1) +
    scale_color_manual(values = c("#E41A1C", "#4A72A6",
        "#48A462", "#7E6E85", "#D16948", "#FFB716",
        "#E1C62F", "#B75F49", "#EC83BA", "#999999",
        "#E0FFFF")) + theme(legend.position = "right",
    axis.title = element_blank(), axis.text = element_blank(),
    axis.ticks = element_blank()) + coord_fixed(ratio = 1)
pdf("umap_cell_type.pdf")
plot(umap_cellType)
dev.off()
```

```
umap_patient <- ggplot(data = umap_meta_merged, aes(UMAP1,
    UMAP2)) + geom_point(aes(color = ImageNumber),
    size = 1) + theme(legend.position = "right", axis.title = element_blank(),
    axis.text = element_blank(), axis.ticks = element_blank()) +
    coord_fixed(ratio = 1)
pdf("umap_patient.pdf")
plot(umap_patient)
dev.off()
```

5.4. Neighborhood analysis

We performed neighborhood analysis to investigate if specific cell types demonstrate interactions or avoidance and if this is related to the type of colitis. For this analysis, code is reused from Schapiro…Bodenmiller, *Nature Methods* 2017;14:873-876. Original code can be found through Bodenmiller group Github page. In our analysis, neighbors are defined as two cells that have direct pixel-pixel contact with at least 1 pixel on the cell boundary of both cells, formally expressed as $d(Cell_A, Cell_B) = 0$.

```
# Number of permutations
n_perm = 10000
# Number of cores used for multicore:
ncores = 2

# Recreate Cell ID column
merged_data$Cell_ID <- paste0(merged_data$ImageNumber,
    merged_data$ROInr)

merged_data_sf <- st_as_sf(merged_data)

neighbor_list <- data.frame(matrix(ncol = 2, nrow = 0))
colnames(neighbor_list) <- c("Cell_ID", "Cell_ID.1")
attach(merged_data_sf)
for (i in unique(ImageNumber)) {
    sub <- merged_data_sf %>%
        filter(ImageNumber == i)
    print(paste0("Now working on: ", i, " with total ",
        nrow(sub), " cells."))
    st_agr(sub) = "constant"
    n <- st_set_geometry(st_intersection(sub, sub),
        NULL)
    nn <- n %>%
        dplyr::select(Cell_ID, Cell_ID.1)
    neighbor_list <- rbind(neighbor_list, nn)
    n <- NULL
    nn <- NULL
    sub <- NULL
    print(paste0("Neighbor_list completed for ", i,
        ", total number of neighbours is now: ", nrow(neighbor_list),
        "."))
}

dat_cells_m <- merged_data %>%
    dplyr::select(ImageNumber, Cell_ID, cell_type)
colnames(dat_cells_m) <- c("ImageNumber", "ObjectNumber",
```

```
    "label")
dat_cells_m$group <- dat_cells_m$ImageNumber
dat_cells_m

dat_relation_m <- neighbor_list
dat_relation_m$Relationship <- "Neighbors"
dat_relation_m$FirstImageNumber <- gsub("\\d+$", "",
    dat_relation_m$Cell_ID.1)
dat_relation_m$FirstObjectName <- "cell"
dat_relation_m$SecondImageNumber <- dat_relation_m$FirstImage
dat_relation_m$SecondObjectName <- "cell"
dat_relation_m <- dat_relation_m[, c("Relationship",
    "FirstObjectName", "FirstImageNumber", "Cell_ID.1",
    "SecondObjectName", "SecondImageNumber", "Cell_ID")]
colnames(dat_relation_m) <- c("Relationship", "First Object Name",
    "First Image Number", "First Object Number", "Second Object Name",
    "Second Image Number", "Second Object Number")

dat_relation_m <- as.data.table(dat_relation_m)
save(dat_relation_m, file = "Neighbor_relationships_all_patients.rda")
dat_cells_m <- as.data.table(dat_cells_m)

d_m = prepare_tables(dat_cells_m, dat_relation_m)

# Calculate baseline statistics
dat_baseline_m = apply_labels(d_m[[1]], d_m[[2]]) %>%
    aggregate_histo()
# Calculate permutation statistics
set.seed(12312)
dat_perm_m = rbindlist(mclapply(1:n_perm, function(x) {
    dat_labels = shuffle_labels(d_m[[1]])
    apply_labels(dat_labels, d_m[[2]]) %>%
        aggregate_histo()
}, mc.cores = ncores), idcol = "run")

# Calculate P values
dat_m_p <- calc_p_vals(dat_baseline_m, dat_perm_m,
    n_perm = 10000, p_tresh = 0.01)
dat_m_p$p_adj <- p.adjust(dat_m_p$p, method = "BH")

# save results
save(dat_m_p, file = "Neighborhood_analysis_10000perm.rda")

# Create heatmap to visualize number of
# significant interactions for the labels
pmat_m = dcast(dat_m_p, "FirstLabel ~ SecondLabel",
    value.var = "sigval", fun.aggregate = sum, fill = 0,
    drop = F)
rname_m = pmat_m$FirstLabel
pmat_m = pmat_m %>%
    dplyr::select(-c("FirstLabel"))
pmat_m = as.data.frame(pmat_m)
pmat_m = as.matrix(pmat_m)
```

```
row.names(pmat_m) <- rname_m

cols = inferno(11)
cmap = colorRampPalette(cols)

hr <- hclust(dist(pmat_m), method = "ward.D")
heatmap.2(pmat_m, Colv = as.dendrogram(hr), Rowv = as.dendrogram(hr),
    trace = "none", col = cmap(75), density.info = "none")

# Merge dat_m_p dataframe with clinical metadata
meta_p <- merged_data %>%
    distinct(smaller = pmin(ImageNumber, treatment),
        larger = pmax(ImageNumber, treatment), .keep_all = T) %>%
    dplyr::select(-smaller, -larger) %>%
    dplyr::select(ImageNumber, treatment, grade)
colnames(dat_m_p)[1] <- "ImageNumber"
dat_m_p_meta <- merge(dat_m_p, meta_p, by = c("ImageNumber"))
dat_m_p_meta <- as.data.frame(dat_m_p_meta)

# Agglomerative clustering (only CD4, CD8, Treg,
# DC, Epithelial, B cell, gdTcell)
aggl_set <- dat_m_p_meta
aggl_set <- aggl_set %>%
    filter(FirstLabel == "CD4_Tcell" | FirstLabel ==
        "CD8_Tcell" | FirstLabel == "Treg" | FirstLabel ==
        "Dendritic_cell" | FirstLabel == "Epithelial_cell" |
        FirstLabel == "gd_Tcell" | FirstLabel == "B_cell")
aggl_set <- aggl_set %>%
    filter(SecondLabel == "CD4_Tcell" | SecondLabel ==
        "CD8_Tcell" | SecondLabel == "Treg" | SecondLabel ==
        "Dendritic_cell" | SecondLabel == "Epithelial_cell" |
        SecondLabel == "gd_Tcell" | SecondLabel ==
        "B_cell")
aggl_set$dirP_adj <- aggl_set$p_adj * aggl_set$sigval
aggl_set$interaction <- paste0(aggl_set$SecondLabel,
    "_surrounding_", aggl_set$FirstLabel)
aggl_set <- aggl_set[, c(1, 7, 9, 10, 11, 12, 13, 14)]
aggl_meta <- aggl_set %>%
    distinct(smaller = pmin(ImageNumber, treatment),
        larger = pmax(ImageNumber, treatment), .keep_all = T) %>%
    dplyr::select(-smaller, -larger) %>%
    dplyr::select(ImageNumber, treatment, grade)
rownames(aggl_meta) <- aggl_meta$ImageNumber
aggl_meta$treatment <- ifelse(aggl_meta$treatment ==
    1, "aCTLA4", ifelse(aggl_meta$treatment == 2, "aPD1",
    ifelse(aggl_meta$treatment == 3, "cICI", "UC")))
aggl_set_p <- aggl_set[, -c(2, 5, 6)]
aggl_set_p$log_dirP_adj <- ifelse(aggl_set_p$dirP_adj ==
    0, 0, -log10(aggl_set_p$p_adj) * aggl_set_p$sigval)
aggl_set_p <- aggl_set_p[, -c(2, 3, 4)]
aggl_set_wide <- spread(aggl_set_p, key = interaction,
    value = log_dirP_adj)
rownames(aggl_set_wide) <- aggl_set_wide$ImageNumber
```

```
aggl_set_wide <- aggl_set_wide[, -1]

annotation_row = data.frame(treatment = factor(aggl_meta$treatment))
rownames(annotation_row) = rownames(aggl_meta)
anno_colors <- list(treatment = c(aCTLA4 = "#440154FF",
    aPD1 = "#FDE725FF", cICI = "#006ee6", UC = "#CA0020"))

aggl_heatmap <- pheatmap(aggl_set_wide, annotation_colors = anno_colors,
    scale = "none", cluster_row = T, cluster_col = T,
    annotation_row = annotation_row, clustering_method = "ward.D",
    clustering_distance_cols = "manhattan", clustering_distance_rows = "manhattan",
    color = colorRampPalette(c("navy", "white", "red"))(30),
    na_col = "white", show_colnames = T, show_rownames = T,
    cutree_rows = 2, cutree_cols = 4)
dev.print(pdf, file = "Heatmap_selected_interactions_Manhattan_10000perm.pdf",
    width = 15, height = 10)
dev.off()
```

5.5. CD8+ T cell cytotoxicity

The extent of CD8+ T cell cytotoxicity is compared among all four colitis groups in terms of the fraction of CD8+ T cells with above median production of granzyme B.

```
# GzmB production by CD8 T cells
CD8_sub <- merged_data %>%
    filter(cell_type == "CD8_Tcell")
CD8_GzmB <- merged_data %>%
    filter(cell_type == "CD8_Tcell" & GzmB_Mean > median(CD8_sub$GzmB_Mean)) %>%
    group_by(ImageNumber, treatment, Epithelium) %>%
    tally()
CD8_total <- merged_data %>%
    filter(cell_type == "CD8_Tcell") %>%
    group_by(ImageNumber, treatment, Epithelium) %>%
    tally()
names(CD8_GzmB)[names(CD8_GzmB) == "n"] <- "n_CD8_GzmB"
names(CD8_total)[names(CD8_total) == "n"] <- "n_CD8_total"
CD8_sum <- merge(CD8_GzmB, CD8_total, by = c("ImageNumber",
    "treatment", "Epithelium"))
CD8_sum$GzmB_rate <- CD8_sum$n_CD8_GzmB/CD8_sum$n_CD8_total
CD8_GzmB_rate <- ggplot(CD8_sum, aes(x = factor(treatment),
    y = GzmB_rate, fill = factor(Epithelium))) + geom_boxplot(outlier.shape = NA) +
    geom_point(pch = 21, position = position_jitterdodge()) +
    scale_x_discrete(labels = c(expression(alpha *
        "CTLA4"), expression(alpha * "PD1"), "cICI",
        "UC")) + scale_fill_discrete(labels = c("Lamina propria",
    "Epithelium")) + theme(axis.title.x = element_blank(),
    legend.title = element_blank()) + ylab("Fraction of CD8 T cells with high granzyme B production") +
    geom_signif(annotation = "*", y_position = 0.94,
        xmin = 1, xmax = 3, tip_length = c(0, 0)) +
    geom_signif(annotation = "*", y_position = 0.88,
        xmin = 2, xmax = 3, tip_length = c(0, 0)) +
    geom_signif(annotation = "*", y_position = 0.91,
        xmin = 3, xmax = 4, tip_length = c(0, 0)) +
```

```
    ylim(0.125, 1)
pdf("CD8_GzmB_rate.pdf")
plot(CD8_GzmB_rate)
dev.off()

# For statistics, CD8_sum is recreated but
# without 'Epithelium' as a grouping variable
summary(aov(CD8_sum$GzmB_rate ~ factor(CD8_sum$treatment)))
TukeyHSD(aov(CD8_sum$GzmB_rate ~ factor(CD8_sum$treatment)))

# GzmB production by CD8 T cells visualized as
# density plots per colitis subtype
attach(CD8_sub)
summary(CD8_sub$GzmB_Mean)
density_graph <- ggplot(CD8_sub, aes(x = as.numeric(GzmB_Mean),
    y = as.factor(treatment))) + geom_density_ridges(aes(fill = as.factor(treatment)),
    scale = 2, alpha = 0.7, quantile_lines = TRUE,
    quantiles = 0.5) + scale_fill_manual(values = c("#00AFBB",
    "#E7B800", "#FC4E07", "#011F4B")) + scale_y_discrete(labels = c(expression(alpha *
    "CTLA4"), expression(alpha * "PD1"), "cICI", "UC")) +
    theme(legend.position = "none") + geom_vline(xintercept = summary(CD8_sub$GzmB_Mean)[[3]],
    linetype = "dashed") + labs(x = "CD8 T cell granzyme B expression level",
    y = "")
pdf("density_graph.pdf")
plot(density_graph)
dev.off()
```

Tissue-resident CD8+ T cells are classically characterized as CD69+CD103+ double positive cells. We identified T_{RM} CD8+ T cells as cells with above-median expression of CD69 and CD103 measured within CD8+ T cells only.

```
CD8_set <- merged_data %>%
    filter(cell_type == "CD8_Tcell")
merged_data$TRM <- ifelse(merged_data$CD69_Mean > median(CD8_set$CD69_Mean) &
    merged_data$CD103_Mean > median(CD8_set$CD103_Mean) &
    merged_data$cell_type == "CD8_Tcell", 1, 0)
```

5.5.1. Distance of lamina propria CD8+ T cells to nearest epithelium

For all CD8+ T cells in the lamina propria, the distance to the nearest epithelium (in micrometers) is calculated.

```
epi_list <- NULL

for (i in seq_along(pts)) {
    filename <- paste0("Epi", i)
    wd <- paste0("Epi/Epi", i, ".csv")
    assign(filename, read.csv(wd))
    maxy <- get(paste0("max_y", i))
    Epi <- get(paste0("Epi", i))
    Epi$Y <- maxy - Epi$Y
    Epit <- as.data.frame(Epi, stringsAsFactors = T)
```

```
    Epit_src <- st_as_sf(Epit, coords = c("X", "Y"))
    Epit_src <- st_make_valid(st_geometry(Epit_src))
    Epit_name <- paste0("Epit_src", i)
    assign(Epit_name, Epit_src)
}

for (i in seq_along(pts)) {
    LPL_sub <- merged_data %>%
        filter(ImageNumber == pts[i])
    LPL_subset <- LPL_sub %>%
        filter(cell_type == "CD8_Tcell" & Epithelium ==
            0)
    Epi_subset <- LPL_sub %>%
        filter(cell_type == "Epithelial_cell")
    mat <- matrix(nrow = nrow(LPL_subset), ncol = 3)
    LPL_centroids <- st_centroid(st_as_sf(LPL_subset))
    nrows <- nrow(LPL_subset)
    for (j in 1:nrows) {
        a <- LPL_centroids[j, ]
        b <- get(paste0("Epit_src", i))
        mat[j, 1] <- a$ImageNumber
        mat[j, 2] <- a$ROInr
        mat[j, 3] <- min(st_distance(a, b))
        print(paste0("Done: ", j, " of ", nrows, " for patient ",
            i, " of 23."))
    }
    mat <- as.data.frame(mat)
    names(mat) <- c("ImageNumber", "ROInr", "distance")
    result <- merge(mat, LPL_subset, by = c("ImageNumber",
        "ROInr"))
    distance_result <- paste0("result", i)
    assign(distance_result, result)
}
total_distance_result <- rbind(result1, result2, result3,
    result4, result5, result6, result7, result8, result9,
    result10, result11, result12, result13, result14,
    result15, result16, result17, result18, result19,
    result20, result21, result22, result23)
save(total_distance_result, file = "Epi_distance_CD8.rda")
result <- total_distance_result

# Add distances to subset of CD8 T cells.  Set
# epithelial distance of intra-epithelial T cells
# to 0.
result <- total_distance_result
result <- result[, c(1:3)]
CD8_test2 <- CD8_set
CD8_test2 <- merge(CD8_set, result, by = c("ImageNumber",
    "ROInr"), all = T)
names(CD8_test2)[188] <- "distance"
CD8_test2$distance[is.na(CD8_test2$distance)] <- 0
```

5.5.2. Other variables potentially associated with CD8+ T cell granzyme B production

Before a mixed-effects model is fitted in section 5.5.3., two more variables potentially associated with CD8+ granzyme B production are included in the data set: 1) activation status, defined as above-median expression of MHC-II (the antibody in our experiment was directed against all MHC-II subclasses HLA-DR/DP/DQ) and 2) proliferation status based on above/below median expression of Ki67.

```
CD8_test2$activated <- ifelse(CD8_test2$PANHLA_Mean >
    quantile(CD8_set$PANHLA_Mean, 0.5), 1, 0)
CD8_test2$proliferating <- ifelse(CD8_test2$Ki67_Mean >
    quantile(CD8_test2$Ki67_Mean, 0.5), 1, 0)
```

5.5.3. Mixed effects model explaining contributing factors of CD8+ T cell granzyme B production

```
mixed_model_GzmB <- lmer(as.numeric(GzmB_Mean)~as.factor(treatment)+as.factor(Epithelium)+
  as.numeric(distance)+as.factor(TRM)+as.factor(activated)+
  as.factor(proliferating)+(1|as.factor(ImageNumber)),data=CD8_test2)

#Results
r.squaredGLMM(mixed_model_GzmB)
summary(mixed_model_GzmB)
coef(summary(mixed_model_GzmB))

#Check assumptions
res <- residuals(mixed_model_GzmB)
qqnorm(res)
qqline(res)
par(ask=F)
linearity<-plot(resid(mixed_model_GzmB),#extract the residuals
                CD8_test2$GzmB_Mean) #specify original y variable
linearity
residuals <- residuals(mixed_model_GzmB)
prediction <- predict(mixed_model_GzmB)
plot(prediction,residuals)
abline(0,0)

#Create vertical bar plot displaying mixed-model coefficients
mm_outcome <- as.data.frame(coef(summary(mixed_model_GzmB))[,c(1,5)])
mm_outcome[,2] <- -log10(mm_outcome[,2])
mm_outcome[,3] <- rownames(mm_outcome)
mm_outcome[,3] <- c("Intercept","aPD-1 (aCTLA4 is ref)","cICI (aCTLA4 is ref)","UC (aCTLA4 is ref)","Ep
mm_outcome[6,1] <- 100*mm_outcome[6,1]
mm_outcome <- mm_outcome[-1,]
mm_outcome[,4] <- ifelse(mm_outcome[,2]>1.301,"*","")
colnames(mm_outcome) <- c("Coefficient","-log10 P value","Covariate","Sign")
mm_outcome <- mm_outcome[c(4,5,1,3,8,7,6,2),]
attach(mm_outcome)
bar_chart2 <- ggplot(mm_outcome,aes(reorder(Covariate,Coefficient),Coefficient))+
  geom_col(aes(fill=`-log10 P value`))+
  scale_fill_gradient2(low="blue",mid="grey",high="red")+
  coord_flip()+
```

```
  theme_classic()+
  ylab("Coefficient")+
  xlab("")+
  geom_text(aes(label=Sign),position=position_stack(vjust=0.5))
bar_chart2
```

6. Session info

```
sessionInfo()
```

```
## R version 4.2.0 (2022-04-22)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Big Sur/Monterey 10.16
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.2/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] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] formatR_1.14
##
## loaded via a namespace (and not attached):
##  [1] compiler_4.2.0  fastmap_1.1.1   cli_3.6.1       tools_4.2.0
##  [5] htmltools_0.5.5 rstudioapi_0.14 yaml_2.3.7      rmarkdown_2.21
##  [9] knitr_1.42      xfun_0.38       digest_0.6.31   rlang_1.1.0
## [13] evaluate_0.20
```