PAPER TITLE TO BE DEFINED (in common.yaml)

12 - cMAF- and Mafb-deficient IM

2022-01-13 14:14:00 +0100

Abstract

Lung interstitium macrophages (IMs) are non-alveolar resident tissue macrophages which contribute to the lung homeostasis. These cells were reported to be heterogeneous by our group and other teams, which contains two main distinct subpopulations: CD206+ IMs and CD206- IMs. However, the exact origin of IMs and the transcriptional programs that control IM differentiation remains unclear. In recent report, we analyzed the refilled IMs in the course of time after induced IM depletion with single-cell RNA sequencing (10X Genomics Chromium) and bulk RNA sequencing.

Contents

1 Description

We would like to know the functions of cMAF and MAFb transcription factors in IM differentiation or IM refilling in empty niche. We analyze the IM with cMAF-KO, MAFb-KO, cMAF/MAF-dKO and control phenotype using scRNAseq (10X Genomics). These mice are Ms4a3-Cre induced recombination KO mice, thus gene KO is dependent on the Ms4a3 expression. We know in 6-week old Ms4a3-dTomato mice, dTomato+ cells don't present in all the whole IM population (~80%).

The solution is to use the BM chimera with BM of Ms4a3-cMAF, Ms4a3-MAFb, Ms4a3-cMAF/MAFb and littermate control transferred into CD45.1-IM-DTR mice after thorax-protection irradiation. The monocytes with Ms4a3-induced KO will take over the lung after the IM niche is emptied by DT treatment. After 1 week, the CD45.2+CD45.1- (originated from transferred BM) IM and monocytes should be IM and monocytes with right gene KO.

In this experiment, we have 4 groups of chimera mice (host = CD45.1-IM-DTR): 1. Three chimera with littermate control BM (control group); 2. Three chimera with Ms4a3-cMAF BM (cMAF-KO group); 3. Three chimera with Ms4a3-MAFb BM (MAFb-KO group); 4. Three chimera with Ms4a3-cMAF/MAFb BM (cMAF/MAFb-dKO group).

Lung monocytes/macrophages will be sorted. For each sample, input 5K cells will be needed for scRNA-seq library construction to target recovered cell number of 3K cells. Sample from each time point should be stained with anti-MHCI-hashtag (Biolegend Hashtag) before pooled sequencing.

2 Demultiplexing by hashtag sequences

```
suppressMessages({library(Seurat)})
```

2.1 Read data

```
# Load in the UMI matrix (from CR)

umis <- Read10X("../../../counts/scRNAseq/Experiment-7-12-21-ScRNA_NGS21-U976/outs/filtered_feature_bc_matrix/")

# For generating a hashtag count matrix from FASTQ files, please refer to https://github.com/Hoohm/CITE-seq-Count.
# Load in the HTO count matrix
htos <- Read10X("../../../counts/HTO/read_count/", gene.column=1)
```

```
# change cell names to "-1" mode.
colnames(htos) <- paste0(colnames(htos), "-1")

# remove the unmapped as it's not a barcode. It's the last row.
htos <- htos[-nrow(htos), ]

# Select cell barcodes detected by both RNA and HTO
# In the example datasets we have already filtered the cells for you, but perform this step for clarity.
joint.bcs <- intersect(colnames(umis), colnames(htos))

# Subset RNA and HTO counts by joint cell barcodes
umis <- umis[, joint.bcs]
htos <- as.matrix(htos[, joint.bcs])
```

```
# Confirm that the HTO have the correct names
rownames(htos)
```

```
## [1] "HT5_Control-CTTTGTCTTTGTGAG" "HT6_cMAF_KO-TATGCTGCCACGGTA"
## [3] "HT7_MAFb_KO-GAGTCTGCCAGTATC" "HT8_dKO-TATAGAACGCCAGGC"
```

Setup Seurat object and add in the HTO data

```
# Setup Seurat object
hashtag <- CreateSeuratObject(counts = umis)

# Normalize RNA data with log normalization
hashtag <- NormalizeData(hashtag)
# Find and scale variable features
hashtag <- FindVariableFeatures(hashtag, selection.method = 'mean.var.plot
    ')
hashtag <- ScaleData(hashtag, features = VariableFeatures(hashtag))
```

2.2 Adding HTO data as an independent assay

You can read more about working with multi-modal data here

```
# Add HTO data as a new assay independent from RNA
hashtag[['HTO']] <- CreateAssayObject(counts = htos)
# Normalize HTO data, here we use centered log-ratio (CLR) transformation
hashtag <- NormalizeData(hashtag, assay = 'HTO', normalization.method = '
    CLR')
```

2.3 Demultiplex cells based on HTO enrichment

Here we use the Seurat function `HTODemux()` to assign single cells back to their sample origins.

```
# If you have a very large dataset we suggest using k_function = "clara".
    This is a k-medoid clustering function for large applications
# You can also play with additional parameters (see documentation for
    HTODemux()) to adjust the threshold for classification
# Here we are using the default settings
hashtag <- HTODemux(hashtag, assay = "HTO", positive.quantile = 0.94)
```

2.4 Visualize demultiplexing results

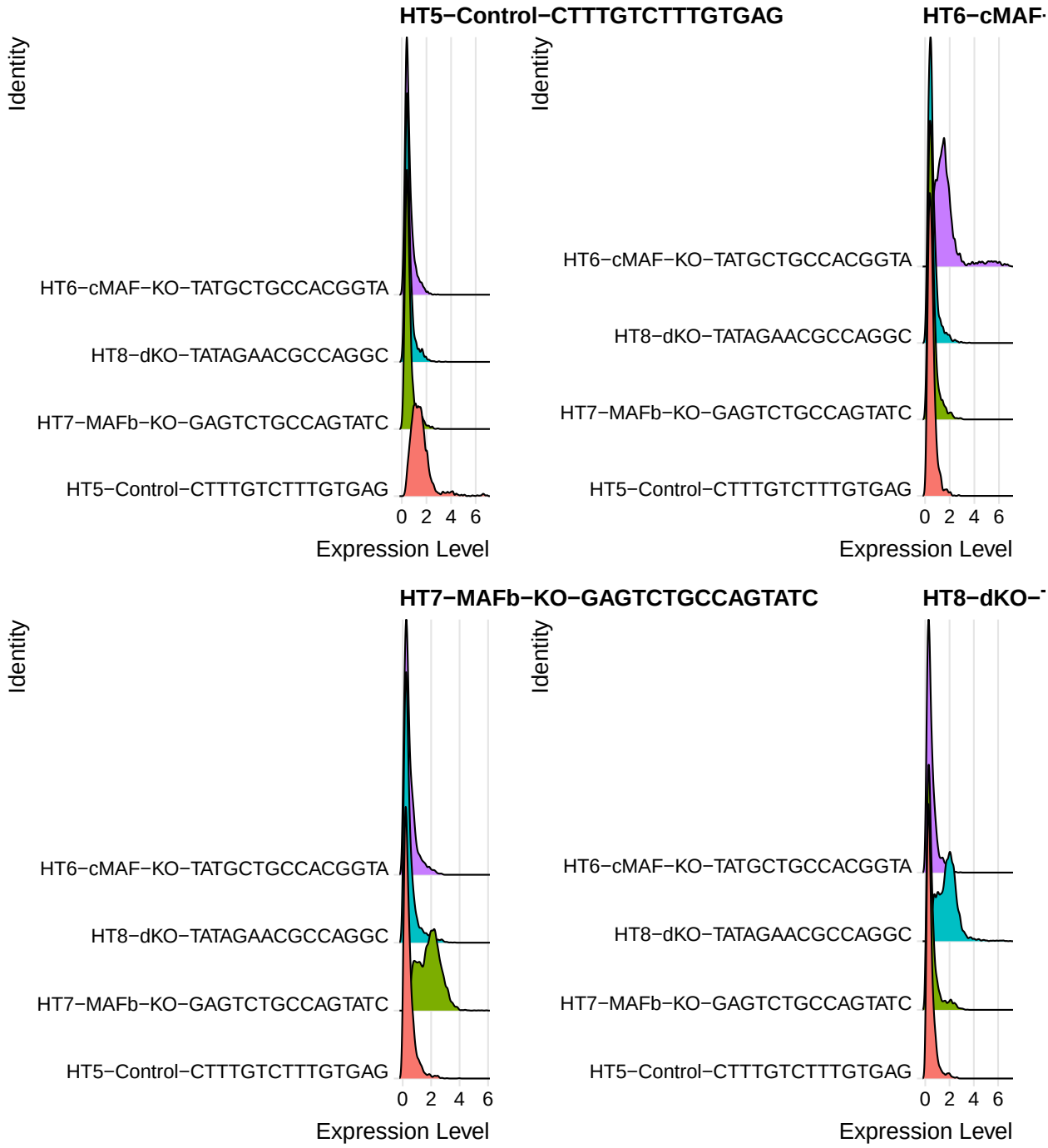
Output from running `HTODemux()` is saved in the object metadata. We can visualize how many cells are classified as singlets, doublets and negative/ambiguous cells.

```
# Global classification results
table(hashtag$HTO_classification.global)
```

```
##
##  Doublet Negative  Singlet
##     2094     2591     6653
```

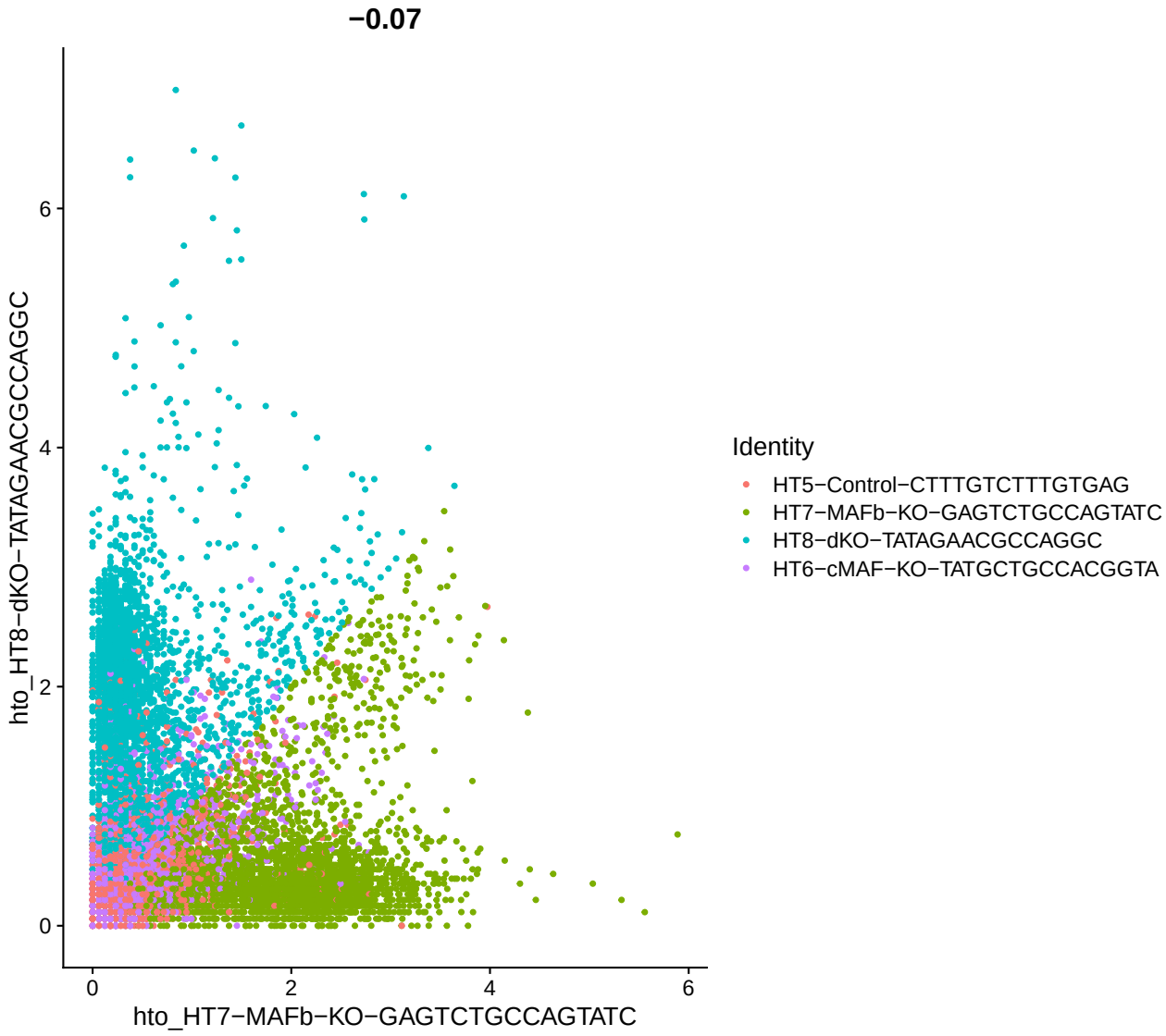
Visualize enrichment for selected HTOs with ridge plots

```
# Group cells based on the max HTO signal
Idents(hashtag) <- 'HTO_maxID'
RidgePlot(hashtag, assay = 'HTO', features = rownames(hashtag[['HTO']]),
    ncol = 2)
```

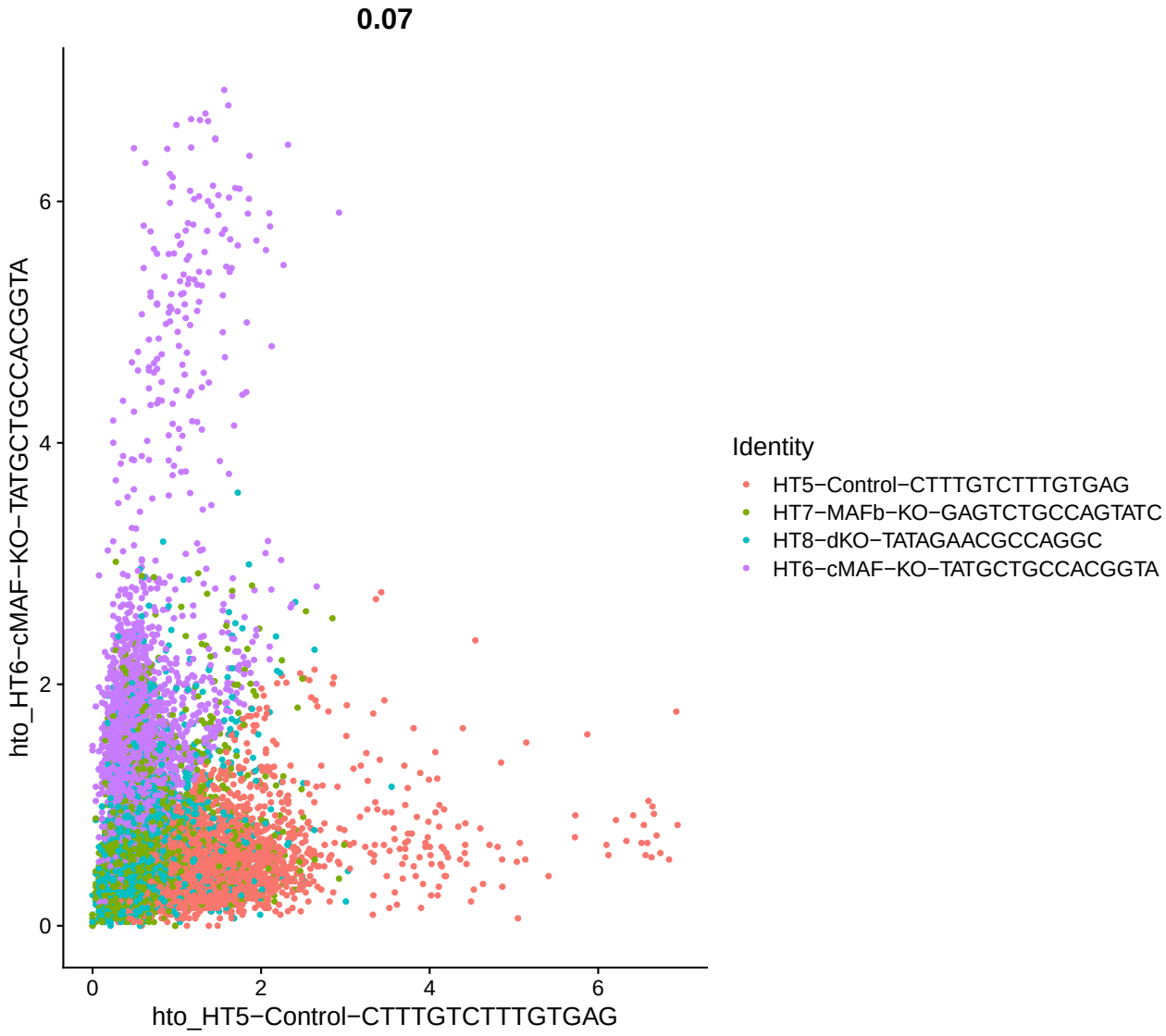


Visualize pairs of HTO signals to confirm mutual exclusivity in singlets

```
FeatureScatter(hashtag, feature1 = 'HT7-MAFb-KO-GAGTCTGCCAGTATC', feature2
    = 'HT8-dKO-TATAGAACGCCAGGC')
```

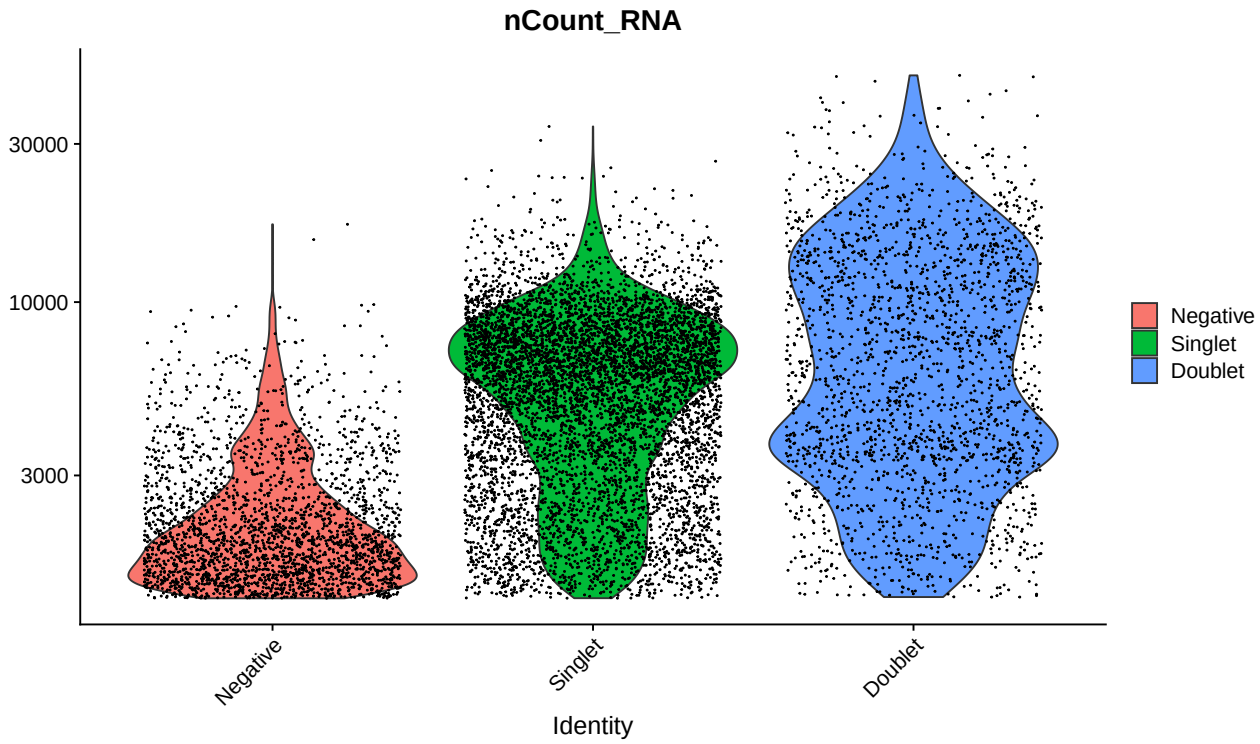



```
FeatureScatter(hashtag, feature1 = 'HT5-Control-CTTTGTCTTTGTGAG', feature2
    = 'HT6-cMAF-KO-TATGCTGCCACGGTA')
```



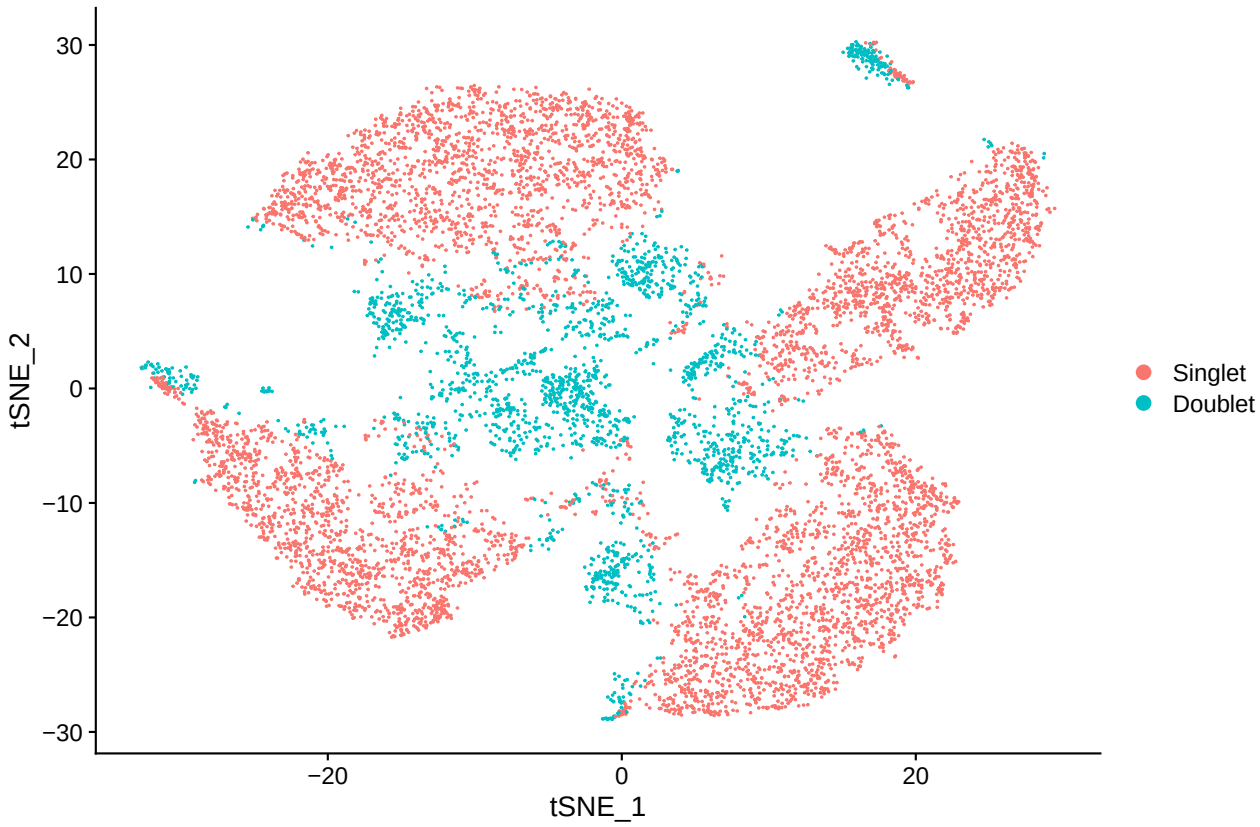
Compare number of UMIs for singlets, doublets and negative cells

```
Idents(hashtag) <- 'HTO_classification.global'
VlnPlot(hashtag, features = 'nCount_RNA', pt.size = 0.1, log = TRUE)
```



Generate a two dimensional tSNE embedding for HTOs.Here we are grouping cells by singlets and doublets for simplicity.

```
#First, we will remove negative cells from the object
hashtag.subset <- subset(hashtag, idents = 'Negative', invert = TRUE)

# Calculate a tSNE embedding of the HTO data
DefaultAssay(hashtag.subset) <- "HTO"
hashtag.subset <- ScaleData(hashtag.subset, features = rownames(hashtag.
    subset), verbose = FALSE)
hashtag.subset <- RunPCA(hashtag.subset, features = rownames(hashtag.
    subset), approx = FALSE)
hashtag.subset <- RunTSNE(hashtag.subset, dims = 1:8, perplexity = 100,
    check_duplicates = FALSE)

DimPlot(hashtag.subset)
```



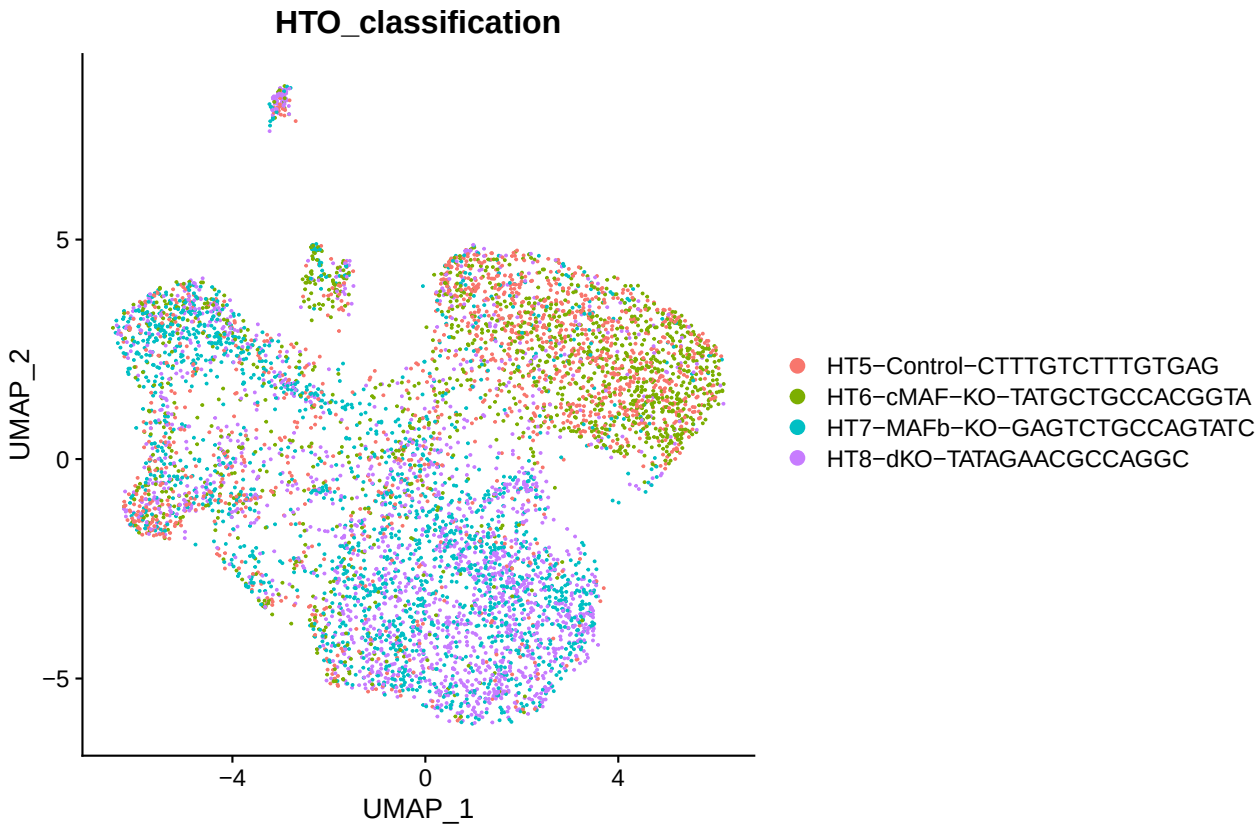
Create an HTO heatmap, based on Figure 1C in the Cell Hashing paper.

```
#To increase the efficiency of plotting, you can subsample cells using the
    num.cells argument
HTOHeatmap(hashtag, assay = 'HTO', ncells = 1000)
```

Cluster and visualize cells using the usual scRNA-seq workflow, and examine for the potential presence of batch effects.

```
# Extract the singlets
singlet <- subset(hashtag, idents = 'Singlet')

# Select the top 1000 most variable features
singlet <- FindVariableFeatures(singlet, selection.method = 'mean.var.plot
    ')

# Scaling RNA data, we only scale the variable features here for
    efficiency
singlet <- ScaleData(singlet, features = VariableFeatures(singlet))

# Run PCA
singlet <- RunPCA(singlet, features = VariableFeatures(singlet))
```

```
# We select the top 10 PCs for clustering and tSNE based on PCElbowPlot
singlet <- FindNeighbors(singlet, reduction = 'pca', dims = 1:10)
singlet <- FindClusters(singlet, resolution = 0.6, verbose = FALSE)
singlet <- RunTSNE(singlet, reduction = 'pca', dims = 1:10)
singlet <- RunUMAP(singlet, reduction = 'pca', dims = 1:10)

# Projecting singlet identities on TSNE visualization
DimPlot(singlet, group.by = "HTO_classification")
```



save to seurat object:

```
saveRDS(singlet, file = "./demultiplexed.seuratObject.rds")
```

3 Session information

R sesssion:

```
sessionInfo()
```

```
## R version 4.0.3 (2020-10-10)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 20.04.3 LTS
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3
## LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/liblapack.so.3
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_GB.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_GB.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_GB.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_GB.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] SeuratObject_4.0.4 Seurat_4.0.5
##
## loaded via a namespace (and not attached):
##   [1] nlme_3.1-153          spatstat.sparse_2.0-0 matrixStats_0.61.0
##   [4] RcppAnnoy_0.0.19      RColorBrewer_1.1-2    httr_1.4.2
##   [7] sctransform_0.3.2     tools_4.0.3           utf8_1.2.2
##  [10] R6_2.5.1              irlba_2.3.5           rpart_4.1-15
##  [13] KernSmooth_2.23-20    uwot_0.1.11           mgcv_1.8-33
##  [16] DBI_1.1.1             lazyeval_0.2.2        colorspace_2.0-2
##  [19] withr_2.4.3           tidyselect_1.1.1      gridExtra_2.3
##  [22] compiler_4.0.3        plotly_4.10.0         labeling_0.4.2
##  [25] scales_1.1.1          spatstat.data_2.1-0   lmtest_0.9-39
##  [28] ggridges_0.5.3        pbapply_1.5-0         goftest_1.2-3
##  [31] stringr_1.4.0         digest_0.6.29         spatstat.utils_2.2-0
##  [34] rmarkdown_2.11        pkgconfig_2.0.3       htmltools_0.5.2
##  [37] parallelly_1.29.0     highr_0.9             fastmap_1.1.0
##  [40] htmlwidgets_1.5.4     rlang_0.4.12          shiny_1.7.1
##  [43] farver_2.1.0          generics_0.1.1        zoo_1.8-9
##  [46] jsonlite_1.7.2        ica_1.0-2             dplyr_1.0.7
##  [49] magrittr_2.0.1        patchwork_1.1.1       Matrix_1.3-4
##  [52] Rcpp_1.0.7            munsell_0.5.0         fansi_0.5.0
##  [55] abind_1.4-5           reticulate_1.22       lifecycle_1.0.1
##  [58] stringi_1.7.6         yaml_2.2.1            MASS_7.3-53
##  [61] Rtsne_0.15            plyr_1.8.6            grid_4.0.3
##  [64] parallel_4.0.3        listenv_0.8.0         promises_1.2.0.1
##  [67] ggrepel_0.9.1         crayon_1.4.2          deldir_1.0-6
##  [70] miniUI_0.1.1.1        lattice_0.20-41       cowplot_1.1.1
##  [73] splines_4.0.3         tensor_1.5            knitr_1.36
##  [76] pillar_1.6.4          igraph_1.2.9          spatstat.geom_2.3-0
```

```
##  [79] future.apply_1.8.1    reshape2_1.4.4        codetools_0.2-18
##  [82] leiden_0.3.9          glue_1.5.1            evaluate_0.14
##  [85] data.table_1.14.2     png_0.1-7             vctrs_0.3.8
##  [88] httpuv_1.6.3          polyclip_1.10-0       gtable_0.3.0
##  [91] RANN_2.6.1            purrr_0.3.4           spatstat.core_2.3-2
##  [94] tidyr_1.1.4           scattermore_0.7       future_1.23.0
##  [97] assertthat_0.2.1      ggplot2_3.3.5         xfun_0.28
## [100] mime_0.12             xtable_1.8-4          RSpectra_0.16-0
## [103] later_1.3.0           survival_3.2-7        viridisLite_0.4.0
## [106] tibble_3.1.6          cluster_2.1.0         globals_0.14.0
## [109] fitdistrplus_1.1-6    ellipsis_0.3.2        ROCR_1.0-11
```

4 References