

ChIMP Vignette

Courtney Schiebout, H. Robert Frost

Load Libraries

Libraries “CAMML” (Schiebout and Frost 2022) and “Seurat” (Satija et al. 2015) need to be loaded to carry out this vignette. Packages will also load additional libraries they depend on.

```
library(CAMML)
library(Seurat)
library(dplyr)
```

Data Processing

The following code outlines how the joint scRNA-seq/CITE-seq data from Lawlor, et al. (2021) (Lawlor et al. 2021), available on the 10X Genomics website, was processed for further analysis.

```
#load data
malt <- Read10X("raw_feature_bc_matrix/")

## 10X data contains more than one type and is being returned as a list containing matrices of each type

#isolate the RNA data and make it a Seurat Object
malt.data <- malt$`Gene Expression`
seurat <- CreateSeuratObject(counts = malt.data, min.cells=10,min.features=100)

#filter for mitochondrial genes
seurat[["percent.mt"]] <- PercentageFeatureSet(seurat, pattern = "^MT-")
seurat <- subset(seurat, subset = percent.mt < 10)

#normalize and scale the RNA data
seurat <- NormalizeData(seurat)
seurat <- FindVariableFeatures(seurat, selection.method = "vst", nfeatures = 2000)
seurat <- ScaleData(seurat)

## Centering and scaling data matrix

#cluster and visualize
seurat <- RunPCA(seurat)

## PC_1
## Positive: PCLAF, MKI67, RGS13, TYMS, MYBL2, CDK1, ZWINT, RRM2, UBE2C, AURKB
##          TK1, GRN, PKM, TOP2A, BIRC5, ACTB, CCNB2, PHGDH, DHFR, LMO2
```

```

##      NUF2, CST3, SPC25, CTSH, SERPINA9, ASPM, GTSE1, CDT1, SHCBP1, MAD2L1
## Negative: ANXA1, GPR171, GZMK, CCL5, CD8A, SPRY1, GZMA, GTSCR1, RTKN2, TRGC2
##      NKG7, CD40LG, KRT1, KLRD1, IFNG, CRTAM, CD8B, LINC02446, ITM2A, LYPD3
##      ITGA6, ID1, CDKN1C, KLRB1, TRDC, TRGC1, ALKAL2, KLRC2, LINC01871, ENC1
## PC_ 2
## Positive: PCLAF, MKI67, RRM2, ZWINT, CDK1, AURKB, UBE2C, TYMS, TOP2A, RGS13
##      BIRC5, TK1, SPC25, CDCA5, MYBL2, GTSE1, DHFR, NUF2, CDT1, CCNB2
##      CDCA7, MAD2L1, SERPINA9, RMI2, ASPM, CD79A, GINS2, CHEK1, ASF1B, SHCBP1
## Negative: CEBPD, CST3, TNFAIP2, LYZ, TYROBP, CSF2RA, NDRG2, LGALS2, NECTIN2, FCER1G
##      SERPINA1, RAB32, GOS2, ETS2, PLAUR, ALDH2, IFITM3, C15orf48, CXCL8, VEGFA
##      AC020656.1, CLEC7A, IL1B, CXCL2, PKP2, DST, AIF1, TIMP1, PLXDC2, CFP
## PC_ 3
## Positive: HLA-DRA, HLA-DQA1, HLA-DPA1, HLA-DQB1, HLA-DRB1, HLA-DPB1, CD74, IGHM, MS4A1, CD79A
##      HLA-DMA, IGKC, LY9, TCF4, BASP1, FCRL5, BCL2A1, TNFRSF13B, CTSZ, CD22
##      FTL, SWAP70, ID3, ITGAX, ARID3A, IGHA1, LDLRAD4, CTSH, SYNGR2, H3F3A
## Negative: ITM2A, IL32, ANXA1, MAF, GZMK, CTLA4, BATF, LDHB, TIGIT, HMGB2
##      ICA1, MT2A, KLRB1, CORO1B, H2AFZ, CCL5, LDHA, NCOA7, GZMA, S100A4
##      TNFRSF4, NKG7, GPR171, MAGEH1, PCLAF, GAPDH, CH25H, S100A10, PTPN13, ID2
## PC_ 4
## Positive: LYZ, TYROBP, LGALS2, FCER1G, CST3, AC020656.1, AIF1, IL1B, MS4A6A, CSF2RA
##      JAML, SERPINA1, CPVL, CFP, CD1E, AXL, GOS2, ITGAX, ID2, LST1
##      NLRP3, CLEC7A, EREG, C1QA, DUSP4, C15orf48, CLEC10A, CD4, PLAUR, C1QB
## Negative: CD79A, MS4A1, IGHM, S100A16, DSP, TM4SF1, RBP1, HLA-DRA, FOXC1, S100A14
##      CDC42EP1, EDN1, SOX9, NFIB, KRT7, ELF3, CD74, PITX1, ADRF, TRIM29
##      S100A2, KRT8, TJP1, CALD1, GABRP, PALMD, TACSTD2, RND3, MEIS2, MIA
## PC_ 5
## Positive: PCLAF, MKI67, RRM2, BIRC5, CDK1, AURKB, LYZ, ZWINT, UBE2C, TCL1A
##      TOP2A, TK1, TYMS, LGALS2, SPC25, CDCA5, CST3, GTSE1, NUF2, CCNB2
##      AIF1, ASPM, IGLC2, ESCO2, AC020656.1, DHFR, C1orf162, MAD2L1, IL1B, PBK
## Negative: DUSP4, ARID3A, TNFRSF13B, SLAMF7, CPEB4, IGHA1, TRPV3, NEAT1, PDGFA, CLNK
##      S100A4, LITAF, CD27, RAMP1, GUSB, RGS1, BCAR3, SPN, SQSTM1, UGCG
##      CYTOR, FCRL4, ACY3, CD63, YWHAH, VOPP1, RAB11FIP1, BSG, ACP5, CLECL1

```

```
seurat <- FindNeighbors(seurat, dims = 1:10)
```

```
## Computing nearest neighbor graph
```

```
##Computing SNN
```

```
seurat <- FindClusters(seurat, resolution = 0.5)
```

```

## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 6438
## Number of edges: 216819
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8886
## Number of communities: 11
## Elapsed time: 0 seconds

```

```
seurat <- RunUMAP(seurat, dims = 1:10)
```

```
## Warning: The default method for RunUMAP has changed from calling Python UMAP via reticulate to the R  
## To use Python UMAP via reticulate, set umap.method to 'umap-learn' and metric to 'correlation'  
## This message will be shown once per session
```

```
## 12:13:39 UMAP embedding parameters a = 0.9922 b = 1.112
```

```
## 12:13:39 Read 6438 rows and found 10 numeric columns
```

```
## 12:13:39 Using Annoy for neighbor search, n_neighbors = 30
```

```
## 12:13:39 Building Annoy index with metric = cosine, n_trees = 50
```

```
## 0%    10    20    30    40    50    60    70    80    90   100%
```

```
## [----|----|----|----|----|----|----|----|----|
```

```
## *****|
```

```
## 12:13:40 Writing NN index file to temp file /var/folders/wv/9lqlnj1571q8w6tn77wg10pr0000gp/T//RtmpOf
```

```
## 12:13:40 Searching Annoy index using 1 thread, search_k = 3000
```

```
## 12:13:41 Annoy recall = 100%
```

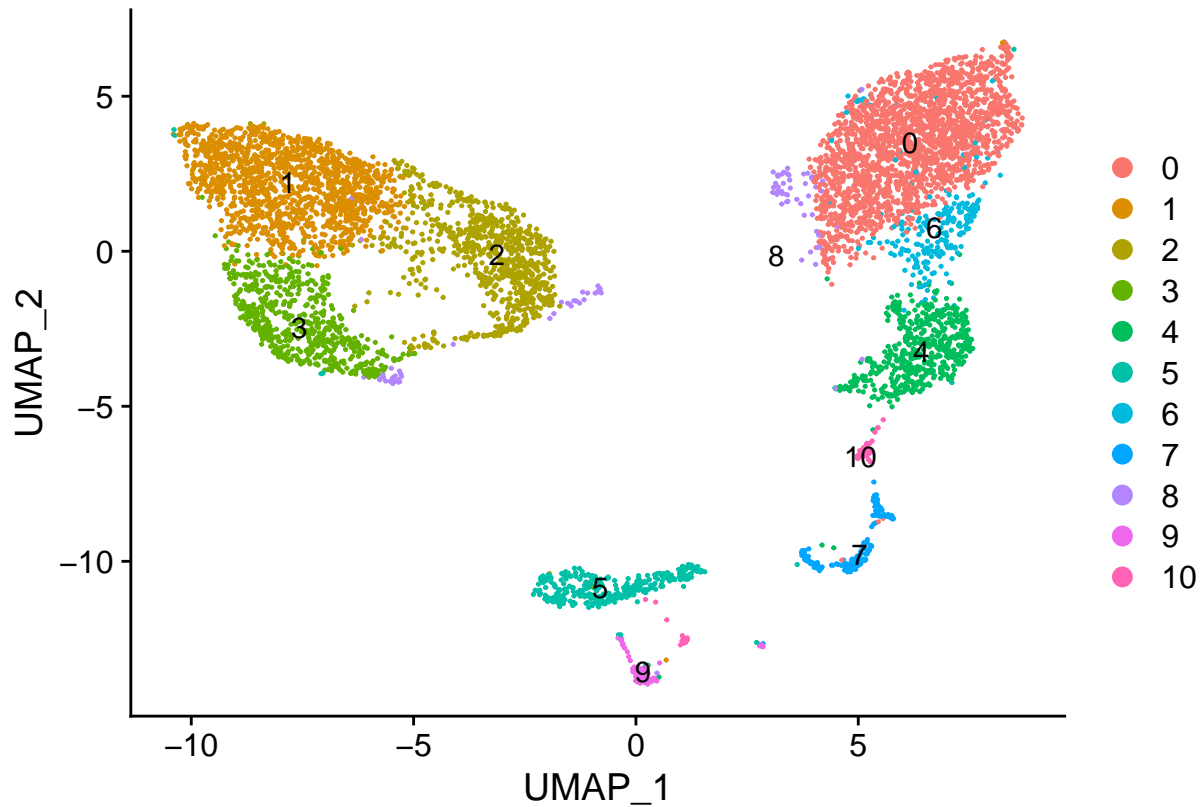
```
## 12:13:42 Commencing smooth kNN distance calibration using 1 thread
```

```
## 12:13:44 Initializing from normalized Laplacian + noise
```

```
## 12:13:44 Commencing optimization for 500 epochs, with 266746 positive edges
```

```
## 12:13:54 Optimization finished
```

```
UMAPPlot(seurat, label = T)
```



scRNA-seq and CITE-seq Integration

Following the RNA data processing, the CITE-seq data needs to be added back into the data as an additional assay in the Seurat Object (Satija et al. 2015; Stoeckius et al. 2017). Since we filtered the data, the CITE-seq data needs aligned with the remaining cells.

```
#read in data and CITE-seq
cb <- malt$`Antibody Capture`

#filter CITE-seq
adt_assay <- CreateAssayObject(counts =
                                cb[,colnames(cb) %in% colnames(seurat)])

## Warning: Feature names cannot have underscores ('_'), replacing with dashes
## ('-')

#add CITE-seq to SeuratObject
seurat[["ADT"]] <- adt_assay

#scale and normalize CITE-seq
seurat <- NormalizeData(seurat, assay = "ADT", normalization.method = "CLR")

## Normalizing across features
```

```
seurat <- ScaleData(seurat, assay = "ADT")
```

```
## Centering and scaling data matrix
```

Get Gene Sets and Run CAMML

In order to run CAMML and ChIMP, a gene set of cell types needs to be accessed. In the following code, “GetGeneSets” is used to load a pre-built gene set of 5 immune cell types. This can then be used to run CAMML. For this example HSCs will be removed.

```
#get gene sets
gene.set.df <- GetGeneSets(data = "immune.cells")

#filter out HSC
gene.set.df <- gene.set.df[-which(gene.set.df$cell.type == "HSC_CD34+"),]

gene.set.df
```

| ## | gene.symbol | cell.type | gene.weight | ensembl.id |
|-------|-------------|-----------|-------------|-----------------|
| ## 1 | ABCB4 | B_cell | 5.929227 | ENSG00000005471 |
| ## 3 | AIM2 | B_cell | 5.750254 | ENSG00000163568 |
| ## 5 | BANK1 | B_cell | 7.942142 | ENSG00000153064 |
| ## 8 | BLK | B_cell | 7.548835 | ENSG00000136573 |
| ## 9 | BTNL9 | B_cell | 5.071020 | ENSG00000165810 |
| ## 12 | CD19 | B_cell | 8.853964 | ENSG00000177455 |
| ## 13 | CD22 | B_cell | 6.364610 | ENSG00000012124 |
| ## 19 | CD79A | B_cell | 8.582138 | ENSG00000105369 |
| ## 24 | CPNE5 | B_cell | 6.580999 | ENSG00000124772 |
| ## 30 | E2F5 | B_cell | 7.159631 | ENSG00000133740 |
| ## 32 | FCRL1 | B_cell | 8.492002 | ENSG00000163534 |
| ## 33 | FCRL2 | B_cell | 6.669676 | ENSG00000132704 |
| ## 34 | FCRLA | B_cell | 8.898029 | ENSG00000132185 |
| ## 38 | HLA-DOB | B_cell | 7.261022 | ENSG00000241106 |
| ## 55 | LINC00926 | B_cell | 5.223783 | ENSG00000247982 |
| ## 60 | MS4A1 | B_cell | 7.927719 | ENSG00000156738 |
| ## 66 | P2RX5 | B_cell | 5.862599 | ENSG00000083454 |
| ## 69 | PKHD1L1 | B_cell | 6.031826 | ENSG00000205038 |
| ## 70 | PLEKHG1 | B_cell | 5.831675 | ENSG00000120278 |
| ## 71 | PNOC | B_cell | 7.489833 | ENSG00000168081 |
| ## 74 | RALGPS2 | B_cell | 5.228555 | ENSG00000116191 |
| ## 85 | SPIB | B_cell | 6.160680 | ENSG00000269404 |
| ## 86 | STAP1 | B_cell | 5.787053 | ENSG00000035720 |
| ## 92 | TLR10 | B_cell | 7.109604 | ENSG00000174123 |
| ## 2 | ADGRE2 | Monocyte | 7.729049 | ENSG00000127507 |
| ## 11 | C5AR1 | Monocyte | 8.813094 | ENSG00000197405 |
| ## 14 | CD300LF | Monocyte | 6.470491 | ENSG00000186074 |
| ## 18 | CD68 | Monocyte | 5.983464 | ENSG00000129226 |
| ## 22 | CDKN1C | Monocyte | 5.755567 | ENSG00000129757 |
| ## 25 | CPPED1 | Monocyte | 5.344299 | ENSG00000103381 |
| ## 27 | CSF1R | Monocyte | 6.984429 | ENSG00000182578 |
| ## 28 | CXCL16 | Monocyte | 6.330266 | ENSG00000161921 |

| | | | | |
|-------|----------|----------|----------|-----------------|
| ## 36 | HCK | Monocyte | 7.334923 | ENSG00000101336 |
| ## 37 | HK3 | Monocyte | 5.814265 | ENSG00000160883 |
| ## 51 | LILRA1 | Monocyte | 6.479104 | ENSG00000104974 |
| ## 52 | LILRA2 | Monocyte | 6.321271 | ENSG00000239998 |
| ## 53 | LILRA5 | Monocyte | 5.485005 | ENSG00000187116 |
| ## 54 | LILRB2 | Monocyte | 8.092027 | ENSG00000131042 |
| ## 57 | LST1 | Monocyte | 5.607879 | ENSG00000204482 |
| ## 61 | MS4A7 | Monocyte | 6.592873 | ENSG00000166927 |
| ## 62 | MSR1 | Monocyte | 6.644375 | ENSG00000038945 |
| ## 67 | PAPSS2 | Monocyte | 5.102467 | ENSG00000198682 |
| ## 68 | PILRA | Monocyte | 7.658938 | ENSG00000085514 |
| ## 77 | SERPINA1 | Monocyte | 6.364474 | ENSG00000197249 |
| ## 80 | SLC31A2 | Monocyte | 5.431567 | ENSG00000136867 |
| ## 81 | SLC7A7 | Monocyte | 5.888768 | ENSG00000155465 |
| ## 83 | SMPDL3A | Monocyte | 5.007270 | ENSG00000172594 |
| ## 84 | SPI1 | Monocyte | 5.337993 | ENSG00000066336 |
| ## 87 | TBC1D8 | Monocyte | 5.634734 | ENSG00000204634 |
| ## 88 | TBXAS1 | Monocyte | 5.145335 | ENSG00000059377 |
| ## 97 | VM01 | Monocyte | 7.072633 | ENSG00000182853 |
| ## 23 | CLIC3 | NK_cell | 7.119036 | ENSG00000169583 |
| ## 31 | FASLG | NK_cell | 5.610501 | ENSG00000117560 |
| ## 40 | IL18RAP | NK_cell | 5.080865 | ENSG00000115607 |
| ## 43 | KIR2DL4 | NK_cell | 5.716879 | ENSG00000189013 |
| ## 44 | KIR3DL1 | NK_cell | 6.315612 | ENSG00000167633 |
| ## 45 | KIR3DL2 | NK_cell | 5.439694 | ENSG00000240403 |
| ## 46 | KLRF1 | NK_cell | 5.481712 | ENSG00000150045 |
| ## 48 | KRT86 | NK_cell | 5.458481 | ENSG00000170442 |
| ## 73 | PRR5L | NK_cell | 5.241528 | ENSG00000135362 |
| ## 76 | S1PR5 | NK_cell | 5.062385 | ENSG00000180739 |
| ## 78 | SH2D1B | NK_cell | 7.356306 | ENSG00000198574 |
| ## 82 | SLFN13 | NK_cell | 5.210182 | ENSG00000154760 |
| ## 98 | XCL1 | NK_cell | 6.650246 | ENSG00000143184 |
| ## 99 | YES1 | NK_cell | 5.168530 | ENSG00000176105 |
| ## 6 | BCL11B | T_cells | 5.564075 | ENSG00000127152 |
| ## 15 | CD3D | T_cells | 6.559457 | ENSG00000167286 |
| ## 16 | CD3E | T_cells | 5.457156 | ENSG00000198851 |
| ## 17 | CD3G | T_cells | 7.904570 | ENSG00000160654 |
| ## 20 | CD8A | T_cells | 5.715826 | ENSG00000153563 |
| ## 21 | CD8B | T_cells | 7.828488 | ENSG00000172116 |
| ## 29 | DPP4 | T_cells | 5.142898 | ENSG00000197635 |
| ## 39 | ICOS | T_cells | 6.000820 | ENSG00000163600 |
| ## 41 | INPP4B | T_cells | 5.664718 | ENSG00000109452 |
| ## 42 | ITK | T_cells | 5.013501 | ENSG00000113263 |
| ## 47 | KLRG1 | T_cells | 5.244996 | ENSG00000139187 |
| ## 50 | LEF1 | T_cells | 5.500641 | ENSG00000138795 |
| ## 56 | LRRN3 | T_cells | 7.577637 | ENSG00000173114 |
| ## 58 | MAL | T_cells | 7.694320 | ENSG00000172005 |
| ## 63 | NELL2 | T_cells | 6.411428 | ENSG00000184613 |
| ## 65 | OXNAD1 | T_cells | 5.250065 | ENSG00000154814 |
| ## 75 | RNF157 | T_cells | 5.376098 | ENSG00000141576 |
| ## 79 | SIRPG | T_cells | 5.538280 | ENSG00000089012 |
| ## 89 | TC2N | T_cells | 5.002518 | ENSG00000165929 |
| ## 90 | TCF7 | T_cells | 5.961207 | ENSG00000081059 |
| ## 91 | THEMIS | T_cells | 6.889632 | ENSG00000172673 |

```
## 93      TRABD2A   T_cells    5.447930 ENSG00000186854
## 94      TRAC     T_cells    5.025658 ENSG00000277734
## 95      TRAT1    T_cells    7.189648 ENSG00000163519
## 96      UBASH3A   T_cells    5.305995 ENSG00000160185
```

```
#run CAMML
```

```
seurat <- CAMML(seurat, gene.set.df)
```

```
## Computing VAM distances for 4 gene sets, 6438 cells and 15518 genes.
```

```
## Min set size: 11, median size: 23.5
```

```
## Warning: Feature names cannot have underscores ('_'), replacing with dashes
## ('-')
```

```
## Warning: Feature names cannot have underscores ('_'), replacing with dashes
## ('-')
```

```
## Warning: Cannot add objects with duplicate keys (offending key: vamcdf_) setting
## key to original value 'camml_'
```

Integrate CITE-seq via ChIMP into CAMML with k-means Disretization

Following the running of CAMML, the cell type scores can be altered by the inclusion of CITE-seq data using ChIMP. In order to use this, a list of cell types and their corresponding CITE-seq markers needs to be built. This list, the Seurat Object, and a vector of booleans will then be fed into ChIMP. The vector serves to designate whether, in cases of multiple marker proteins in a cell type, any marker protein can be present to maintain the CAMML score or if ChIMP should require all marker proteins to be present to maintain the CAMML score.

In other words, if a user designates both CD4 and CD8 for T cells, anyMP=TRUE would require that just one of the two markers be present in a cell for the cell to have a nonzero cell type score. However, if anyMP=FALSE, both markers would have to be present in a cell for the cell type score to be nonzero.

In this example, we use anyMP=FALSE for monocytes to single out cells that are positive for both CD14 and CD16. We use anyMP=TRUE to select for T cells that are either CD4 or CD8 positive.

```
#compare ADT markers and cell types
```

```
rownames(seurat@assays$ADT)
```

```
## [1] "CD3-TotalSeqB"      "CD4-TotalSeqB"
## [3] "CD8a-TotalSeqB"     "CD14-TotalSeqB"
## [5] "CD15-TotalSeqB"     "CD16-TotalSeqB"
## [7] "CD56-TotalSeqB"     "CD19-TotalSeqB"
## [9] "CD25-TotalSeqB"     "CD45RA-TotalSeqB"
## [11] "CD45R0-TotalSeqB"   "PD-1-TotalSeqB"
## [13] "TIGIT-TotalSeqB"    "CD127-TotalSeqB"
## [15] "IgG2a-control-TotalSeqB" "IgG1-control-TotalSeqB"
## [17] "IgG2b-control-TotalSeqB"
```

```
rownames(seurat@assays$CAMML)
```

```
## [1] "B-cell" "Monocyte" "NK-cell" "T-cells"
```

```
#create CITE list
```

```
markers <- cbind(c(rownames(seurat),rownames(seurat)[2],rownames(seurat)[4]),  
  (rownames(seurat@assays$ADT)[c(8,4,7,2,6,3)]))  
citelist <- list()  
for (i in 1:length(rownames(seurat))) {  
  citelist[[i]] <- markers[which(markers[,1]==rownames(seurat)[i]),2]  
}  
names(citelist) <- rownames(seurat)  
citelist
```

```
## $`B-cell`  
## [1] "CD19-TotalSeqB"  
##  
## $Monocyte  
## [1] "CD14-TotalSeqB" "CD16-TotalSeqB"  
##  
## $`NK-cell`  
## [1] "CD56-TotalSeqB"  
##  
## $`T-cells`  
## [1] "CD4-TotalSeqB" "CD8a-TotalSeqB"
```

```
#run ChIMP
```

```
seuratk <- ChIMP(seurat, citelist, anyMP = c(T,F,T,T))
```

```
#visualize the cell type scores
```

```
seurat.markers = FindAllMarkers(seuratk, assay="ChIMP", only.pos = TRUE)
```

```
## Calculating cluster 0
```

```
## Calculating cluster 1
```

```
## Calculating cluster 2
```

```
## Calculating cluster 3
```

```
## Calculating cluster 4
```

```
## Calculating cluster 5
```

```
## Warning in FindMarkers.default(object = data.use, slot = data.slot, counts =  
## counts, : No features pass logfc.threshold threshold; returning empty data.frame
```

```
## Calculating cluster 6
```

```
## Calculating cluster 7
```



```
## Calculating cluster 8

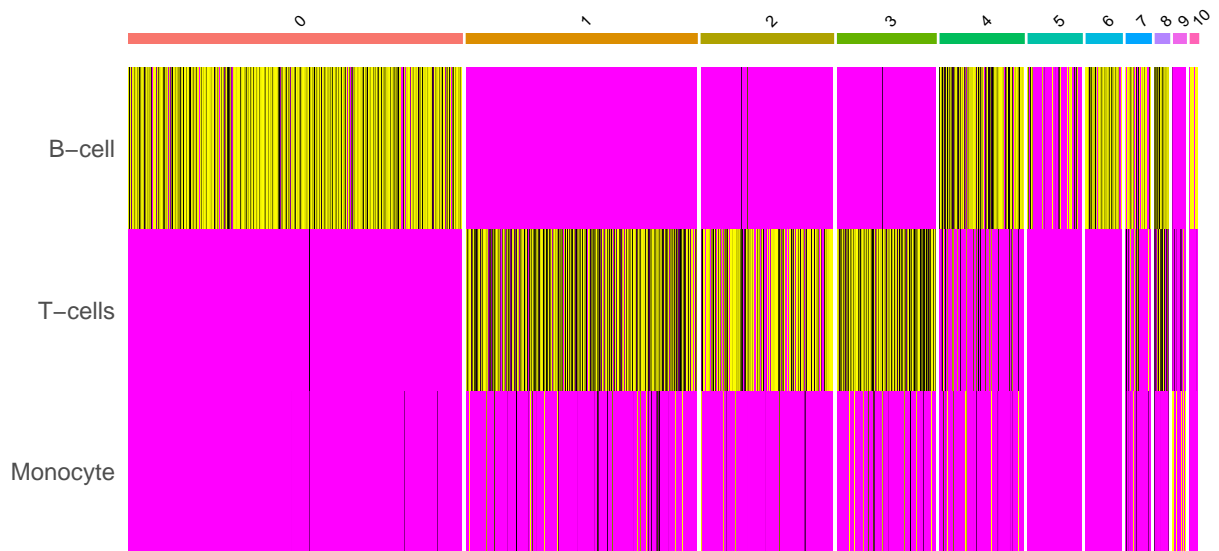
## Warning in FindMarkers.default(object = data.use, slot = data.slot, counts =
## counts, : No features pass logfc.threshold threshold; returning empty data.frame

## Calculating cluster 9

## Calculating cluster 10

## Warning in FindMarkers.default(object = data.use, slot = data.slot, counts =
## counts, : No features pass logfc.threshold threshold; returning empty data.frame

DefaultAssay(object = seuratk) = "ChIMP"
top.pathways <- seuratk.markers %>% group_by(cluster) %>% top_n(n = 3, wt = avg_log2FC)
DoHeatmap(seuratk, slot="data", features = top.pathways$gene,
          size=2, label=T, raster = F) + NoLegend()
```



Integrate CITE-seq via ChIMP into CAMML with Quantile Discretization

This example follows the same pipeline as the above example but uses a median discretization for CITE-seq instead of k-means for comparison.

```
#compare ADT markers and cell types
rownames(seurat@assays$ADT)
```

```
## [1] "CD3-TotalSeqB"      "CD4-TotalSeqB"
## [3] "CD8a-TotalSeqB"     "CD14-TotalSeqB"
## [5] "CD15-TotalSeqB"     "CD16-TotalSeqB"
## [7] "CD56-TotalSeqB"     "CD19-TotalSeqB"
## [9] "CD25-TotalSeqB"     "CD45RA-TotalSeqB"
## [11] "CD45R0-TotalSeqB"   "PD-1-TotalSeqB"
## [13] "TIGIT-TotalSeqB"    "CD127-TotalSeqB"
## [15] "IgG2a-control-TotalSeqB" "IgG1-control-TotalSeqB"
## [17] "IgG2b-control-TotalSeqB"
```

```
rownames(seurat@assays$CAMML)
```

```
## [1] "B-cell" "Monocyte" "NK-cell" "T-cells"
```

```
#create CITE list
markers <- cbind(c(rownames(seurat),rownames(seurat)[2],rownames(seurat)[4]),
  (rownames(seurat@assays$ADT)[c(8,4,7,2,6,3)]))
citelist <- list()
for (i in 1:length(rownames(seurat))) {
  citelist[[i]] <- markers[which(markers[,1]==rownames(seurat)[i]),2]
}
names(citelist) <- rownames(seurat)
citelist
```

```
## $`B-cell`
## [1] "CD19-TotalSeqB"
##
## $Monocyte
## [1] "CD14-TotalSeqB" "CD16-TotalSeqB"
##
## $`NK-cell`
## [1] "CD56-TotalSeqB"
##
## $`T-cells`
## [1] "CD4-TotalSeqB" "CD8a-TotalSeqB"
```

```
#run ChIMP
seuratq <- ChIMP(seurat, citelist, method = "q", anyMP = c(T,F,T,T))
```

```
#visualize the cell type scores
seurat.markers = FindAllMarkers(seuratq, assay="ChIMP", only.pos = TRUE)
```

```
## Calculating cluster 0
```

```
## Calculating cluster 1
```

```
## Calculating cluster 2
```

```

## Calculating cluster 3

## Calculating cluster 4

## Calculating cluster 5

## Warning in FindMarkers.default(object = data.use, slot = data.slot, counts =
## counts, : No features pass logfc.threshold threshold; returning empty data.frame

## Calculating cluster 6

## Calculating cluster 7

## Calculating cluster 8

## Warning in FindMarkers.default(object = data.use, slot = data.slot, counts =
## counts, : No features pass logfc.threshold threshold; returning empty data.frame

## Calculating cluster 9

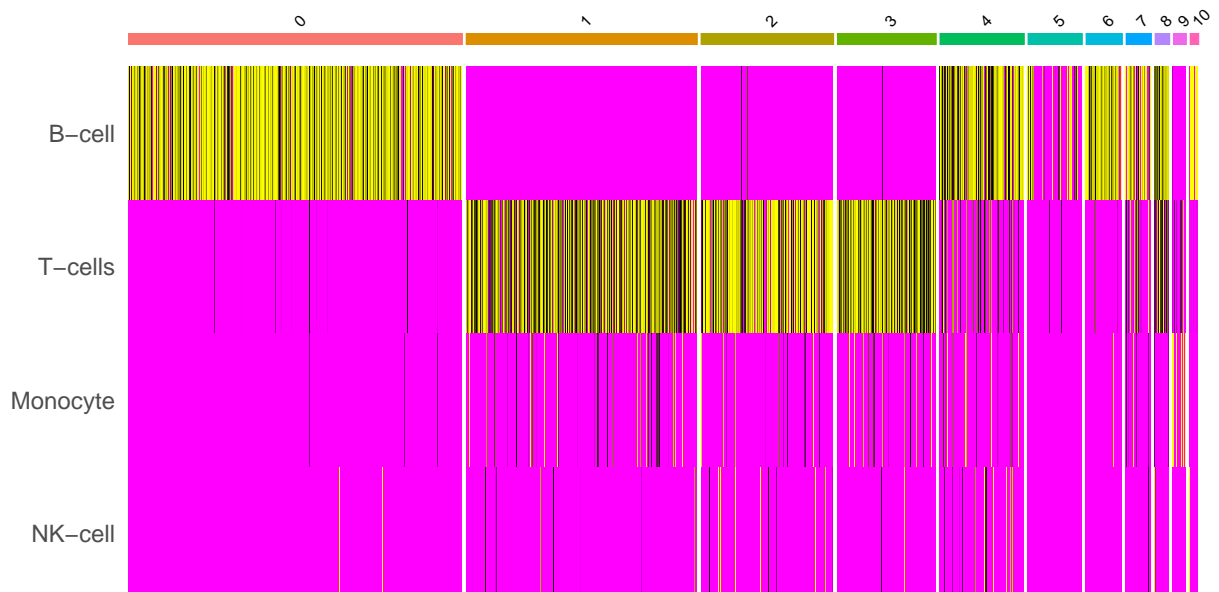
## Calculating cluster 10

```

```

DefaultAssay(object = seuratq) = "ChIMP"
top.pathways <- seurat.markers %>% group_by(cluster) %>% top_n(n = 3, wt = avg_log2FC)
DoHeatmap(seuratq, slot="data", features = top.pathways$gene,
          size=2, label=T, raster = F) + NoLegend()

```



References

- Lawlor, Nathan, Djamel Nehar-Belaid, Jessica D.S. Grassmann, Marlon Stoeckius, Peter Smibert, Michael L. Stitzel, Virginia Pascual, Jacques Banchereau, Adam Williams, and Duygu Ucar. 2021. “Single Cell Analysis of Blood Mononuclear Cells Stimulated Through Either LPS or Anti-CD3 and Anti-CD28.” *Frontiers in Immunology* 12 (March): 636720. <https://doi.org/10.3389/fimmu.2021.636720>.
- Satija, Rahul, Jeffrey A Farrell, David Gennert, Alexander F Schier, and Aviv Regev. 2015. “Spatial Reconstruction of Single-Cell Gene Expression Data.” *Nature Biotechnology* 33 (5): 495–502. <https://doi.org/10.1038/nbt.3192>.
- Schiebout, Courtney, and H. Robert Frost. 2022. “CAMML: Multi-Label Immune Cell-Typing and Stemness Analysis for Single-Cell RNA-Sequencing.” *Pacific Symposium on Biocomputing*.
- Stoeckius, Marlon, Christoph Hafemeister, William Stephenson, Brian Houck-Loomis, Pratip K Chattopadhyay, Harold Swerdlow, Rahul Satija, and Peter Smibert. 2017. “Simultaneous Epitope and Transcriptome Measurement in Single Cells.” *Nature Methods* 14 (9): 865–68. <https://doi.org/10.1038/nmeth.4380>.