Figure 3

EmptyNN - Figure 3

The following code reproduces the Figure 3 in our EmptyNN manuscript.

Load libraries

```
# Please download datasets and seurat objects before running this analysis (run download_datasets.sh in
library("Seurat")
```

```
## Warning: package 'Seurat' was built under R version 4.0.5
```

```
## Attaching SeuratObject
```

```
library("Matrix")
```

```
## Warning: package 'Matrix' was built under R version 4.0.5
```

```
library('pROC')
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library("ggplot2")
```

```
## Warning: package 'ggplot2' was built under R version 4.0.5
```

```
library("pheatmap")
load("./../data/BlueYellowColormaps_V1.RData")
# R version 3.6.3, Seurat_3.2.3, Matrix_1.3-2, ggplot_2_3.3.3, pheatmap_1.0.12
```

Load (1) raw data

(2) filtering results of four cell-calling algorithms: EmptyNN (neg.res), CellRanger 2.0 (ranger.keep), EmptyDrops (e.out, e.keep), CellBender (bender.keep).

```
raw_counts <- Read10X_h5("./../data/multiplexed_PBMC_raw.h5")
```

```
## Warning in sparseMatrix(i = indices[] + 1, p = indptr[], x = as.numeric(x =
## counts[]), : 'giveCsparse' has been deprecated; setting 'repr = "T"' for you
```

```
load("./../data/multiplexed_PBMC_results.RData")
```

Load demuxlets and probabilities from Demuxlet

```
### Demuxlet is run in a docker image, see detailes in https://github.com/statgen/demuxlet/tree/master/
### The corresponding bam and vcf files can be found in https://github.com/yelabucsf/demuxlet_paper_cod
### docker run --rm -v your/path/to/a/directory/:/data yimmieg/demuxlet --sam /data/C.merged.bam --vcf
demuxlet <- read.delim("./../data/multiplexed_PBMC_demuxlet.best",as.is=T)
demuxlet <- demuxlet[-1,]
demuxlet$identity <- sapply(demuxlet$BEST,function(x) {strsplit(x,"-")[[1]][[1]]})
rownames(demuxlet) <- demuxlet$BARCODE
```

Figure 3A

```
cell_contain <- colnames(raw_counts)[neg.res$nn.keep]
cell_free <- setdiff(colnames(raw_counts),cell_contain)
df1 <- demuxlet[intersect(cell_contain,rownames(demuxlet)),'PRB.SNG1',drop=F]
df2 <- demuxlet[intersect(cell_free,rownames(demuxlet)),'PRB.SNG1',drop=F]
df <- rbind(df2,df1)
df$predictions <- c(rep("cell-free",nrow(df2)),rep("cell-containing",nrow(df1)))
ggplot(df, aes(x=predictions, y=PRB.SNG1)) +
  geom_boxplot(outlier.colour="black", outlier.shape=16,
               outlier.size=1, notch=FALSE) +
  ggtitle("EmptyNN classification")+
  xlab("Singlet Probability")+theme_bw()
```

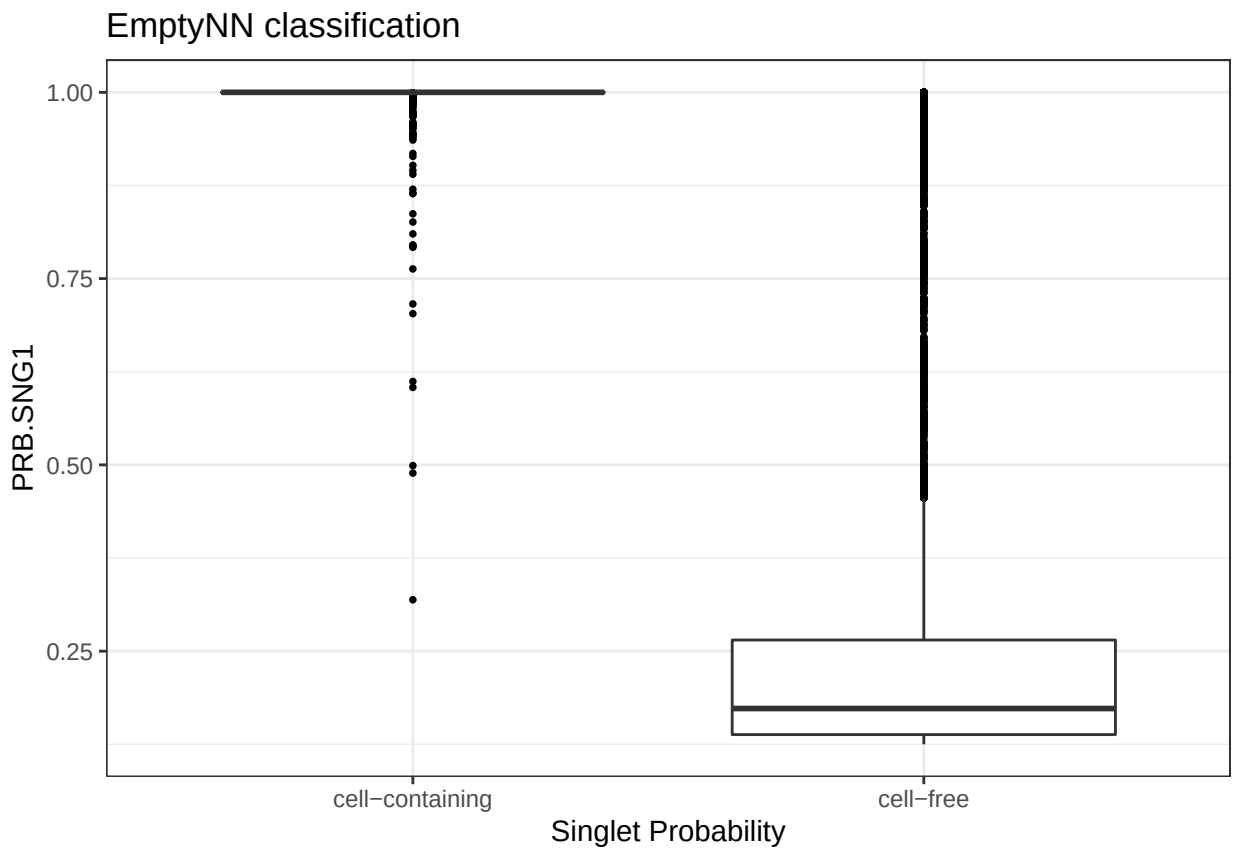


Figure 3B

```
label <- demuxlet[demuxlet$identity!="DBL",]
# EmptyNN
bcs <- intersect(rownames(neg.res$prediction),rownames(label))
rocobj1 <- roc(label[bcs,"identity"], neg.res$prediction[bcs,'mean.crossval'])
```

```
## Setting levels: control = AMB, case = SNG
```

```
## Setting direction: controls < cases
```

```
# Cell Ranger 2.0
names(ranger.keep) <- colnames(raw_counts)
bcs <- intersect(names(ranger.keep),rownames(label))
rocobj2 <- roc(label[bcs,"identity"], as.numeric(ranger.keep[bcs]))
```

```
## Setting levels: control = AMB, case = SNG
## Setting direction: controls < cases
```

```
# EmptyDrops
bcs <- intersect(rownames(e.out[!is.na(e.out$FDR),]),rownames(label))
```

```
## Loading required package: S4Vectors

## Loading required package: stats4

## Loading required package: BiocGenerics

## Warning: package 'BiocGenerics' was built under R version 4.0.5

## Loading required package: parallel

##
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB

## The following object is masked from 'package:pROC':
##
##     var

## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind, colnames,
##     dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##     union, unique, unsplit, which.max, which.min

##
## Attaching package: 'S4Vectors'

## The following object is masked from 'package:Matrix':
##
##     expand

## The following object is masked from 'package:base':
##
##     expand.grid
```

```
rocobj3 <- roc(label[bcs,'identity'], e.out[bcs,'FDR'])
```

```
## Setting levels: control = AMB, case = SNG

## Setting direction: controls > cases
```

```
# CellBender
names(bender.keep) <- colnames(raw_counts)
bcs <- intersect(names(bender.keep),rownames(label))
rocobj4 <- roc(label[bcs,"identity"], as.numeric(bender.keep[bcs]))
```

```
## Setting levels: control = AMB, case = SNG
```

```
## Setting direction: controls < cases
```

```
# CellRanger 3.0
bcs <- intersect(names(ranger3.keep),rownames(label))
rocobj5 <- roc(label[bcs,"identity"], as.numeric(ranger3.keep[bcs]))
```

```
## Setting levels: control = AMB, case = SNG
## Setting direction: controls < cases
```

```
ggroc(list("EmptyNN, 0.963"=rocobj1,"CellRanger 2.0, 0.8294"=rocobj2,
           "EmptyDrops, 0.8643"=rocobj3,"CellBender, 0.5802"=rocobj4,
           "CellRanger 3.0, 0.8303"=rocobj5),linetype='dashed',
      legacy.axes = TRUE) + labs(x = "FPR", y = "TPR")+
  geom_segment(aes(x = 0, xend = 1, y = 0, yend = 1), color="darkgrey", linetype="dashed")+theme_bw()
```

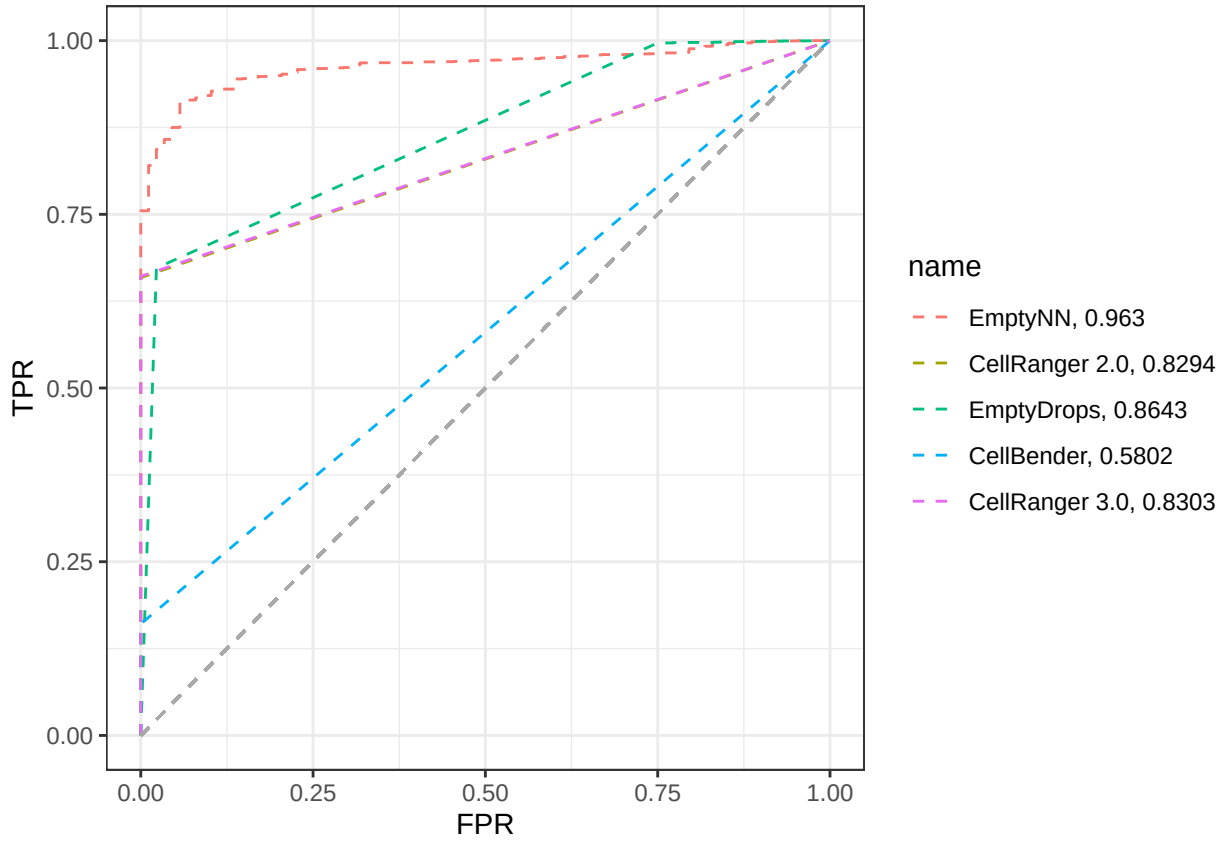


Figure 3C

```
keep <- neg.res$nn.keep | (e.keep & !is.na(e.keep)) | ranger.keep | bender.keep | ranger3.keep
retained <- CreateSeuratObject(counts = raw_counts[,keep])
```

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

```
retained <- retained[-grep("^MOUSE|^RPS|^RPL|^MT", rownames(retained)),]
retained <- NormalizeData(retained,verbose=FALSE)
retained <- FindVariableFeatures(retained,verbose=FALSE)
retained <- ScaleData(retained,features=VariableFeatures(retained),verbose=FALSE)
retained <- RunPCA(retained,features=VariableFeatures(retained),verbose=FALSE)
retained <- FindNeighbors(retained, dims = 1:10,verbose=FALSE)
retained <- FindClusters(retained, resolution = 0.3,verbose=FALSE)
retained <- RunTSNE(retained, dims = 1:10,verbose=FALSE)
retained$demuxlet_label <- demuxlet[colnames(retained),'identity']
DimPlot(retained, group.by="demuxlet_label")+
  ggtitle("Demuxlet classification")
```

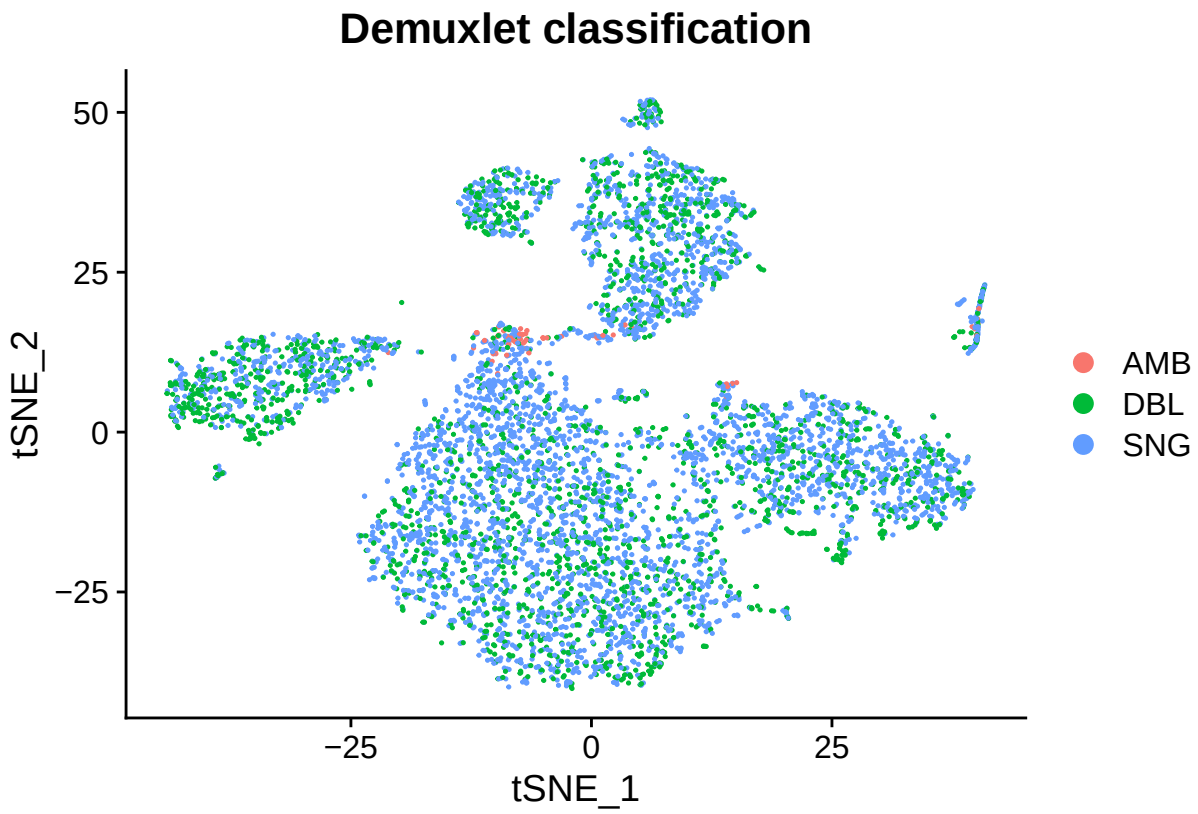


Figure 3D

```
retained$emptynn <- colnames(retained) %in% colnames(raw_counts)[neg.res$nn.keep]
retained$cellranger <- colnames(retained) %in% colnames(raw_counts)[ranger.keep]
retained$cellbender <- colnames(retained) %in% colnames(raw_counts)[bender.keep]
retained$emptydrops <- colnames(retained) %in% colnames(raw_counts)[e.keep & !is.na(e.keep)]
retained$cellranger3.0 <- colnames(retained) %in% colnames(raw_counts)[ranger3.keep]
DimPlot(retained, group.by="emptynn",cols=c('grey','steelblue3'))+
  ggtitle("EmptyNN classification")+NoLegend()
```

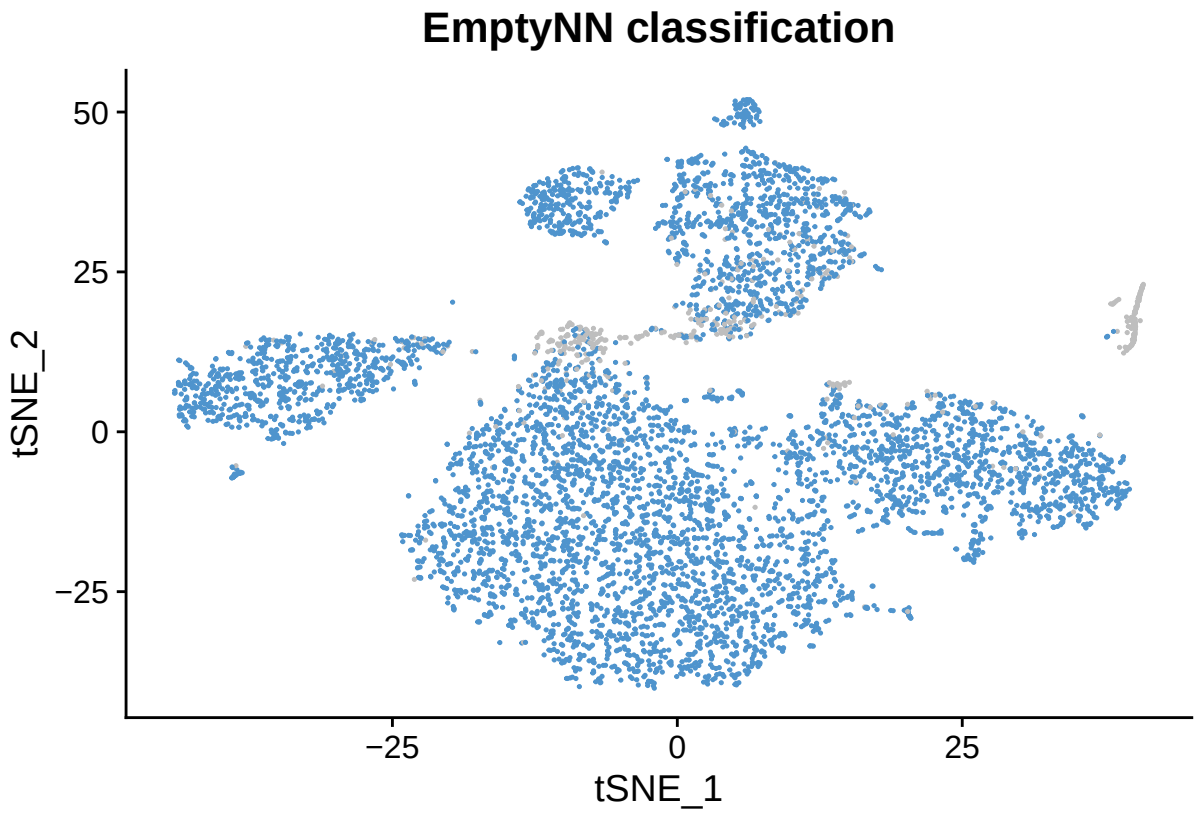


Figure 3E

```
new.cluster <- c("CD4","NK","CD14 Mono c1","CD14 Mono c2","Ambient","CD16 Mono",
                 "Platelets","DC","pDCs")
names(new.cluster) <- levels(retained)
retained <- RenameIdents(retained,new.cluster)
DimPlot(retained, label=T)+ggtitle("Celltype identities")+NoLegend()
```

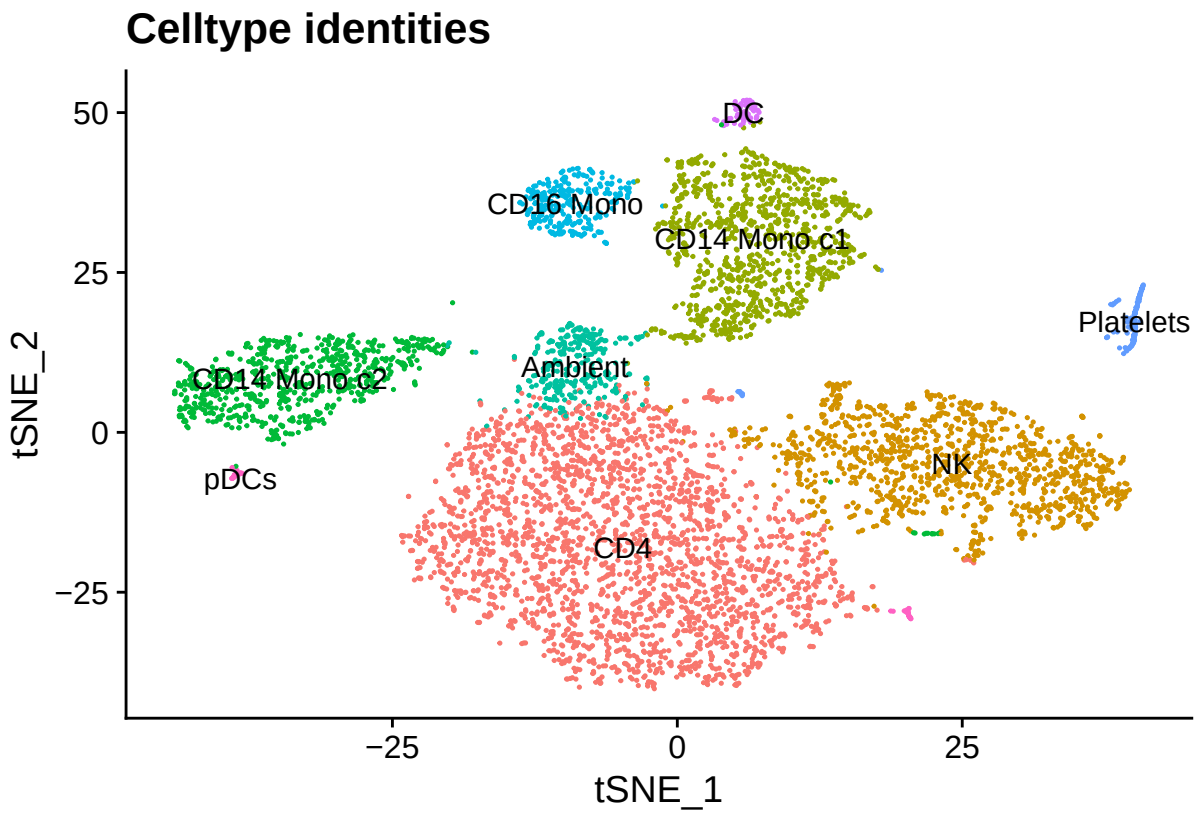
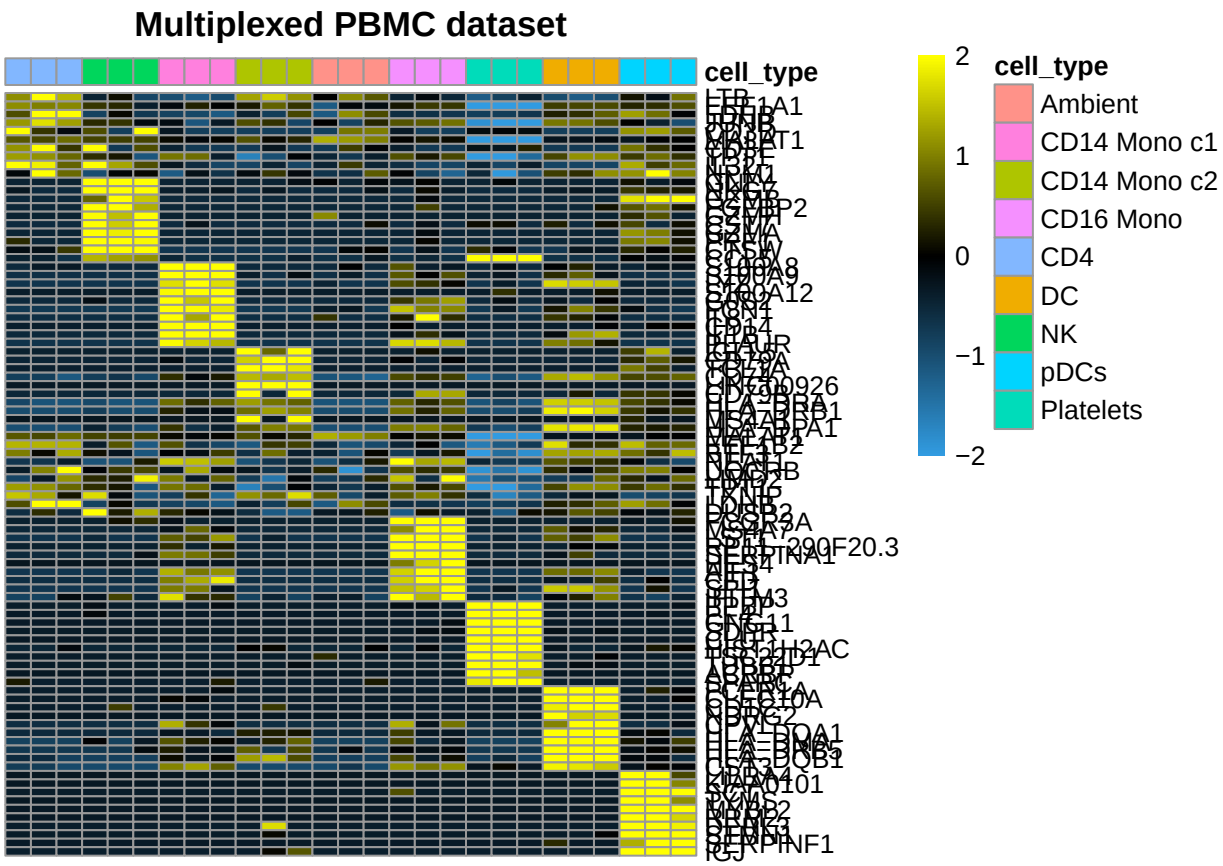


Figure 3F

```
des <- FindAllMarkers(retained, only.pos = TRUE, min.pct = 0.25,
                      logfc.threshold = 0.25,verbose=FALSE)
asplit_genes <- split(1:nrow(des), des$cluster)
genes <- unlist(lapply(asplit_genes, function(x) des[x[1:10], "gene"]))
genes <- genes[genes %in% rownames(retained@assays$RNA@data)]
# Average cells within each cluster
asplit_cells <- split(rownames(retained@meta.data), retained@active.ident)
means <- do.call(cbind, lapply(asplit_cells, function(x){
  s1 <- Matrix::rowMeans(retained@assays$RNA@data[genes, sample(unlist(x), 10)])
  s2 <- Matrix::rowMeans(retained@assays$RNA@data[genes, sample(unlist(x), 10)])
  s3 <- Matrix::rowMeans(retained@assays$RNA@data[genes, sample(unlist(x), 10)])
  cbind(s1, s2, s3)
}))
cell_type <- unlist(lapply(names(asplit_cells), function(x) rep(x, 3)))
# Create heatmap (sample 3 "replicates")
anno_col <- data.frame(cell_type)
rownames(anno_col) <- colnames(means) <- paste(colnames(means), cell_type)
pheatmap(means,cluster_rows = F, cluster_cols = F, scale = "row",
         breaks = seq(-2, 2, length = length(yellow2blue)+1),col=yellow2blue,
         annotation_col = anno_col,show_colnames = F,
         main='Multiplexed PBMC dataset')
```


Multiplexed PBMC dataset
cell_type
2
1
0
−1
−2
cell_type
Ambient
CD14 Mono c1
CD14 Mono c2
CD16 Mono
CD4
DC
NK
pDCs
Platelets