Figure 2

EmptyNN - Figure 2

The following code reproduces the Figure 2 in our EmptyNN manuscript.

Please download datasets and seurat objects before running this analysis (run download_datasets.sh in terminal)

Load libraries

Load (1) raw data

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

```
load("./../data/cell_hashing_raw.RData")
load("./../data/cell_hashing_results.RData")
```

Figure 2A

```
total_counts <- Matrix::rowSums(Stoeckius.counts)
aframe <- data.frame(total_counts, rank = rank(-total_counts),
                     emptynn = nn.res$nn.keep)
twohundred <- unique(aframe$rank[which(aframe$total_counts == 200)])
fifty <- unique(aframe$rank[which(aframe$total_counts == 50)])
recovered <- aframe$rank[which(aframe$emptynn &
                                     aframe$total_counts < 200
                               & aframe$total_counts > 50)]
ggplot(aframe, aes(rank, total_counts)) +
  scale_x_continuous(trans='log10') + scale_y_continuous(trans='log10') +
  geom_point() + theme_bw() +
  xlab("Sorted barcodes") + ylab("Total counts [log10]") +
  geom_vline(xintercept = twohundred, color = "dodgerblue", linetype = "dashed") +
  geom_vline(xintercept = fifty, color = "forestgreen", linetype = "dashed")
```

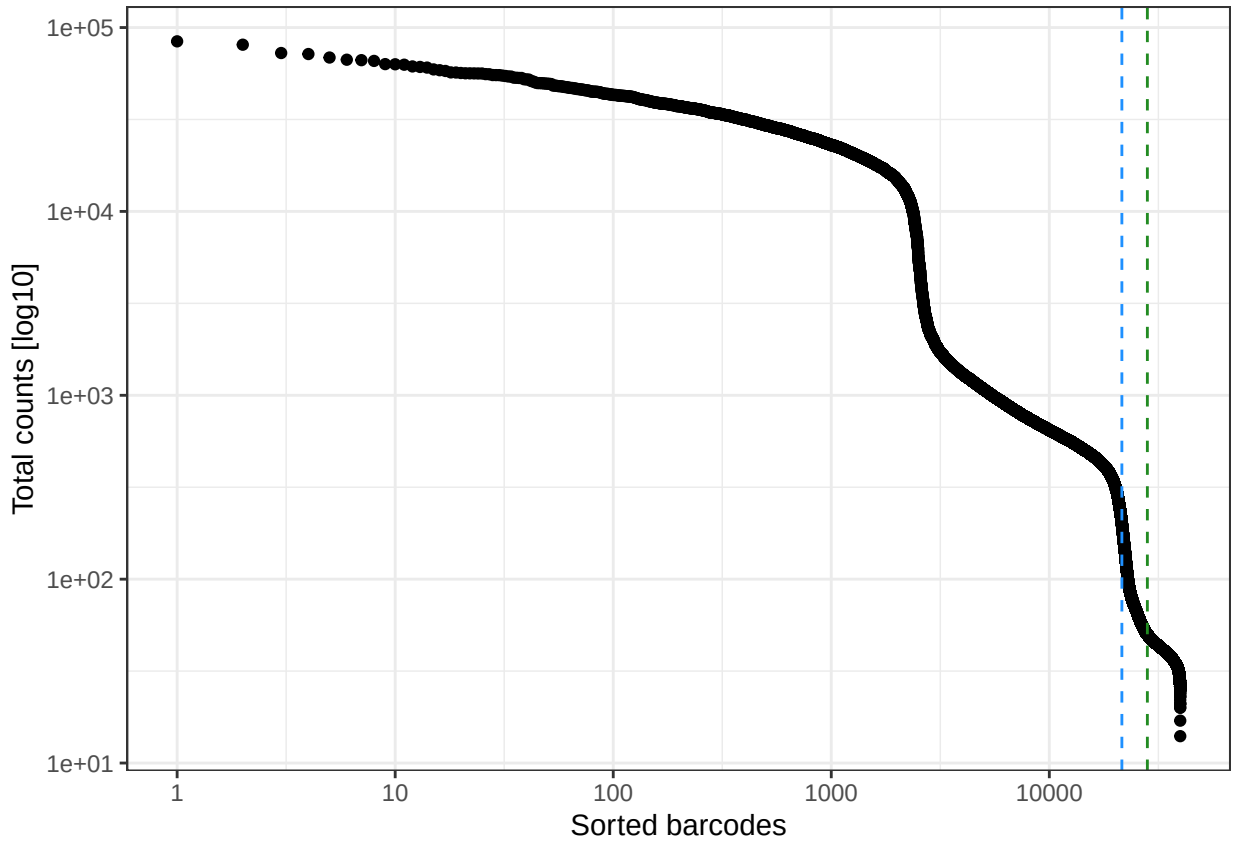


Figure 2B

```
ggplot(aframe[which(aframe$total_counts < 250 & aframe$total_counts > 50),], aes(rank,total_counts)) +
  xlab("Sorted barcodes") + ylab("Total counts") +
  geom_vline(xintercept = twohundred, color = "dodgerblue", linetype = "dashed") +
  geom_vline(xintercept = fifty, color = "forestgreen", linetype = "dashed") +
  geom_vline(xintercept = recovered, color = "darkred", alpha = 0.5) +
  geom_point() + theme_bw()
```

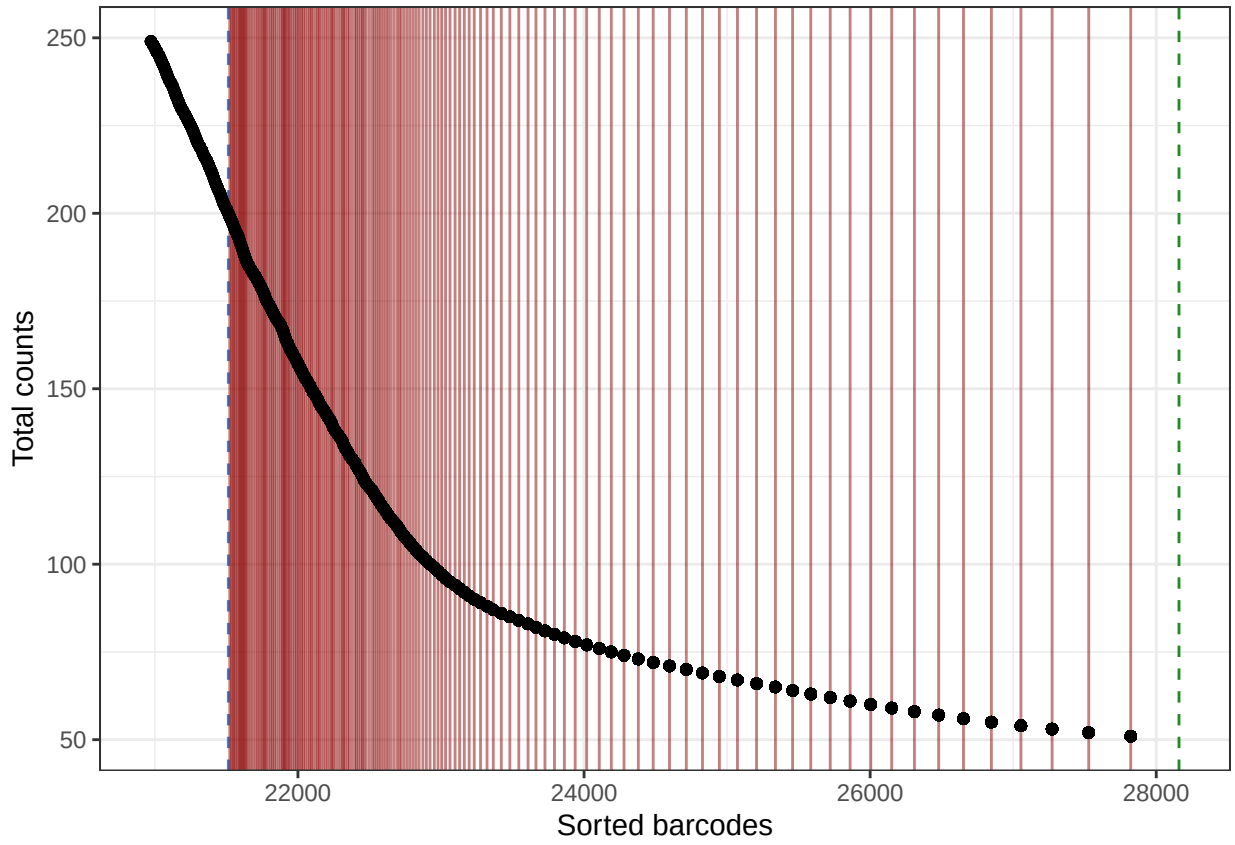


Figure 2C

```
original_paper <- readRDS("./../data/cell_hashing_original_paper.rds")
genes.use <- VariableFeatures(original_paper)
names(nn.res$nn.keep) <- rownames(Stoeckius.counts)
recover.bcs <- intersect(names(total_counts[total_counts<200]),
                         names(nn.res$nn.keep[nn.res$nn.keep]))
recover <- CreateSeuratObject(counts = t(Stoeckius.counts[recover.bcs,]))
```

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

```
recover <- NormalizeData(recover,verbose = FALSE)
recover <- ScaleData(recover, features = genes.use,verbose = FALSE)
sub.genes.use <- rownames(recover[["RNA"]]@scale.data)
recover <- RunPCA(recover,features=sub.genes.use,verbose = FALSE)
recover <- FindNeighbors(recover, dims = 1:10,verbose = FALSE)
recover <- FindClusters(recover, resolution = 0.2,verbose = FALSE)
recover <- RunTSNE(recover, dims = 1:10, check_duplicates = FALSE,verbose = FALSE)
new.cluster <- c("CD14 Mono","CD4","NK","B",'Platelet')
names(new.cluster) <- levels(recover)
recover <- RenameIdents(recover,new.cluster)
DimPlot(recover,label=T)+NoLegend()+labs(title="Cell type identities of recovered cells")
```

```
## Warning: Using ‘as.character()‘ on a quosure is deprecated as of rlang 0.3.0.
## Please use ‘as_label()‘ or ‘as_name()‘ instead.
## This warning is displayed once per session.
```

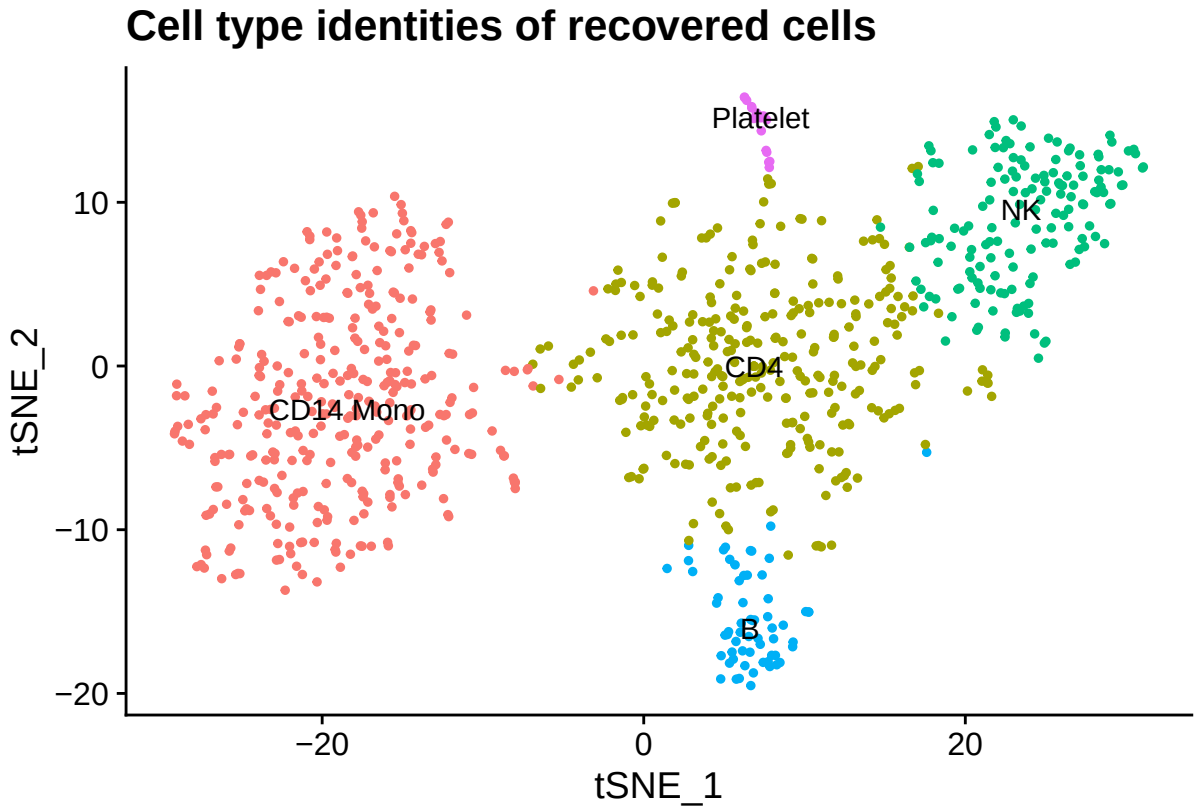


Figure 2D

```
des <- FindAllMarkers(recover, only.pos = TRUE, min.pct = 0.25,
                      logfc.threshold = 0.25,verbose=FALSE)
asplit_genes <- split(1:nrow(des), des$cluster)
genes <- unlist(lapply(asplit_genes, function(x) des[x[1:5], "gene"]))
genes <- genes[genes %in% rownames(recover@assays$RNA@data)]
# Average cells within each cluster
asplit_cells <- split(rownames(recover@meta.data), recover@active.ident)
means <- do.call(cbind, lapply(asplit_cells, function(x){
  s1 <- Matrix::rowMeans(recover@assays$RNA@data[genes, sample(unlist(x), 10)])
  s2 <- Matrix::rowMeans(recover@assays$RNA@data[genes, sample(unlist(x), 10)])
  s3 <- Matrix::rowMeans(recover@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",show_rownames = F,
         breaks = seq(-2, 2, length = length(yellow2blue) + 1), col = yellow2blue,
         annotation_col = anno_col,show_colnames = F)
```

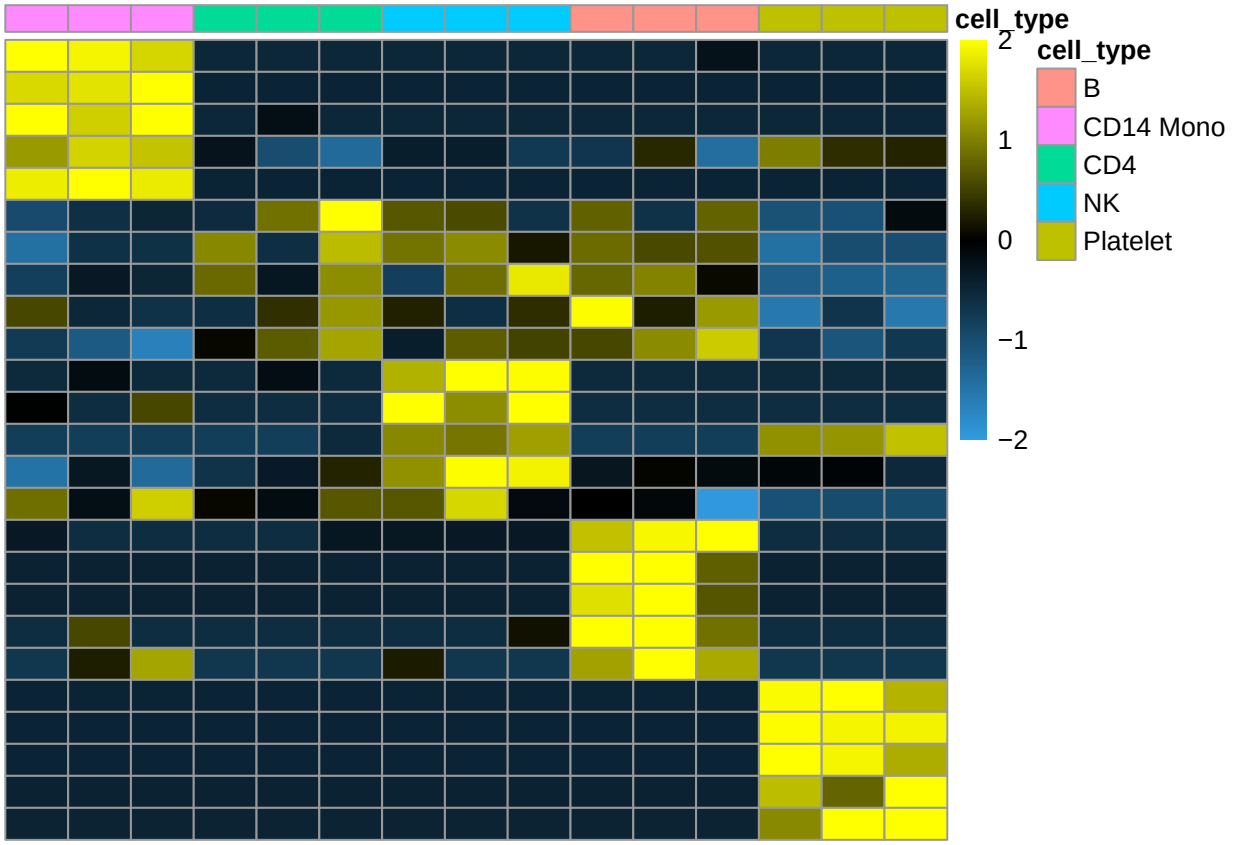


Figure 2E

```
pbmc <- readRDS("./../data/reference_pbmc_3k.rds")
pla <- subset(pbmc,idents="Platelet")
pla.recover <- subset(recover,idents="Platelet")
c1 <- as.matrix(pla.recover[['RNA']]@counts)
c2 <- as.matrix(pla[['RNA']]@counts)
features <- head(rownames(FindMarkers(recover, ident.1 = "Platelet",only.pos = TRUE)),300)
features <- intersect(intersect(rownames(c1),rownames(c2)),features)
expr <- cbind(c1[features,],c2[features,])
m1 <- rowMeans(c1[rownames(expr),])
m2 <- rowMeans(c2[rownames(expr),])
df <- data.frame("recovered"=m1,"bona_fide"=m2)
ggplot(df, aes(x=recovered, y=bona_fide)) +
        geom_point()+
        geom_smooth(method=lm)+
        xlab("recovered platelets")+ylab("bona fide platelets")
```

```
## `geom_smooth()` using formula 'y ~ x'
```

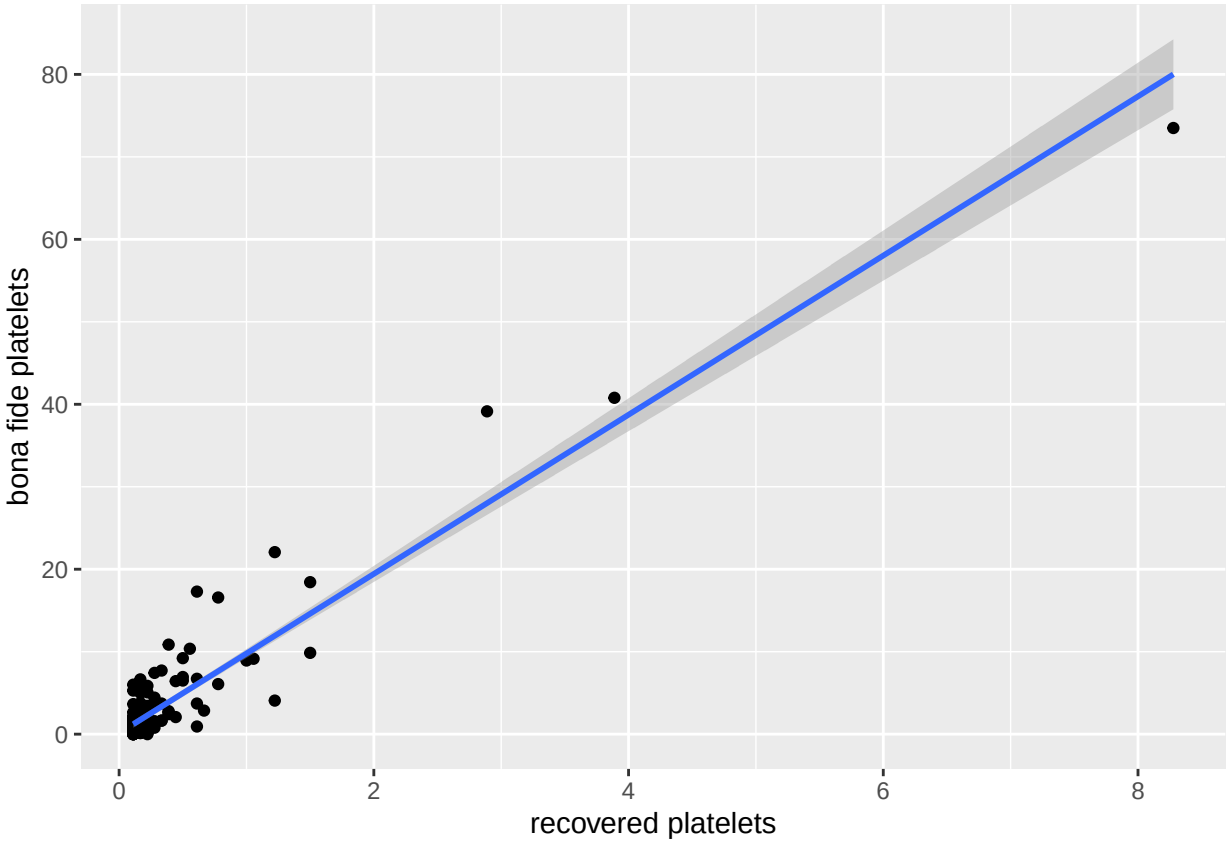


Figure S3A

```
retained <- readRDS("./../data/cell_hashing_retained.rds")
retained$label <- label[colnames(retained),1]
DimPlot(retained,reduction='tsne',group.by = 'label')+
  NoLegend()+labs(title="Cell hashing info")
```

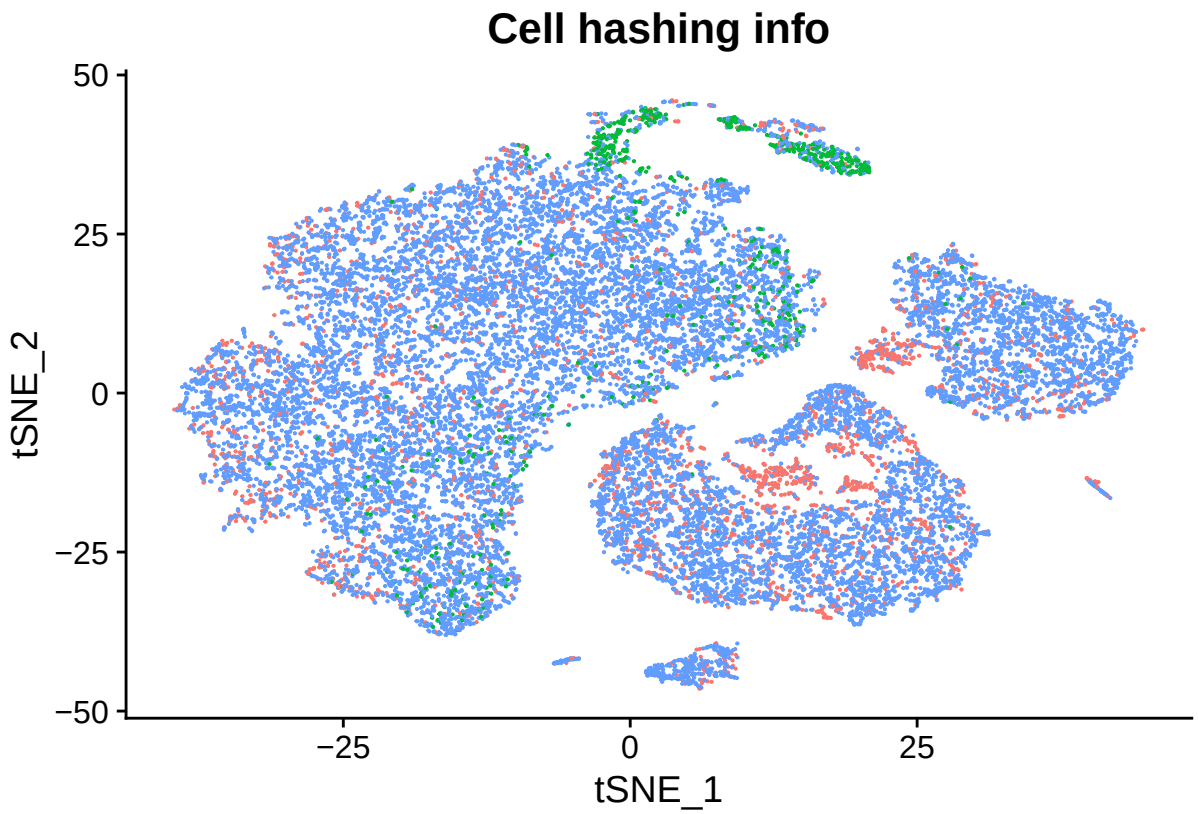


Figure S3B

```
DimPlot(retained,label=T,reduction='tsne')+NoLegend()+labs(title="Cell type identity")
```

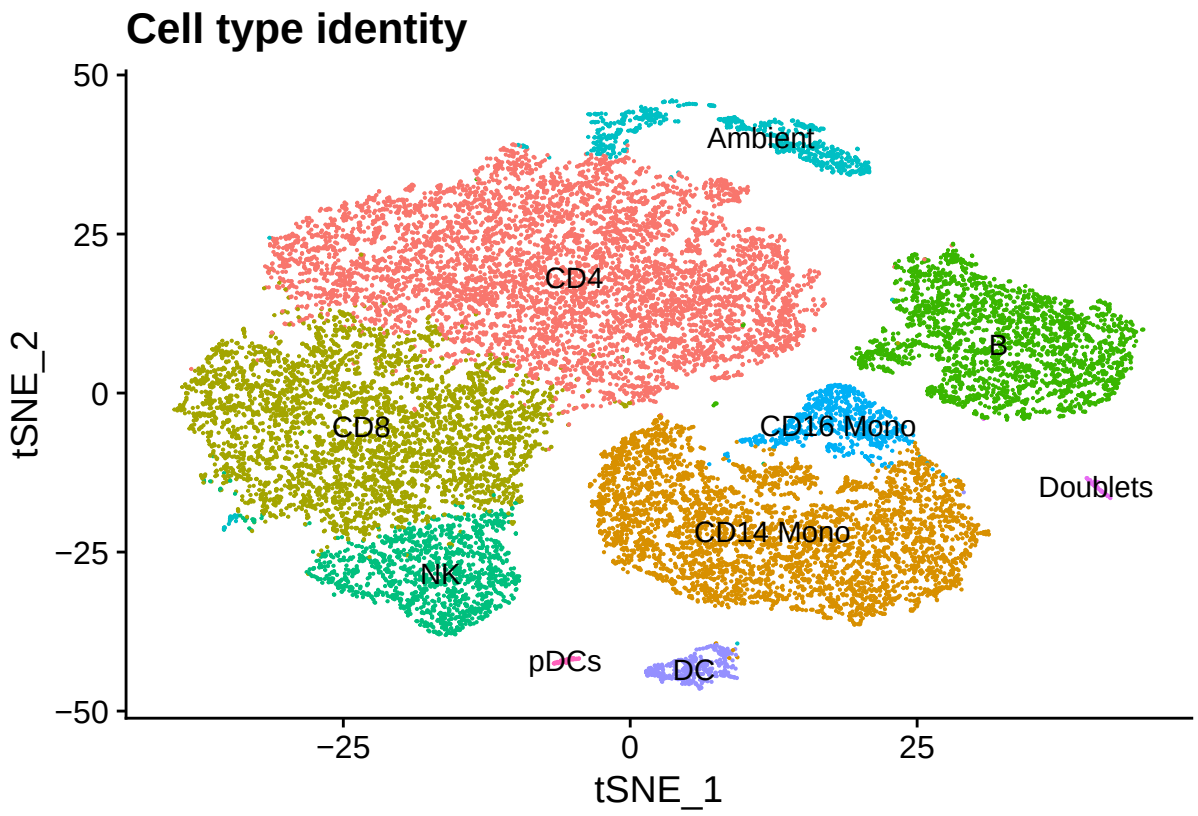
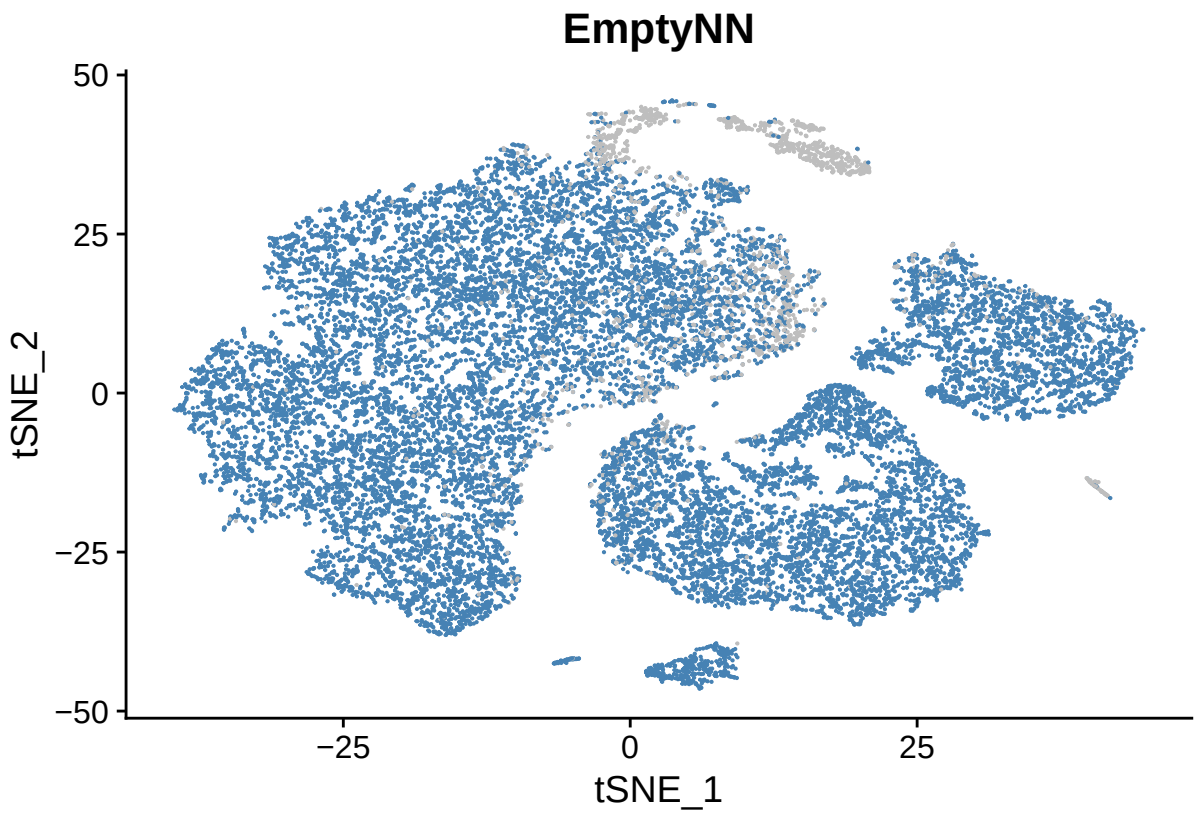
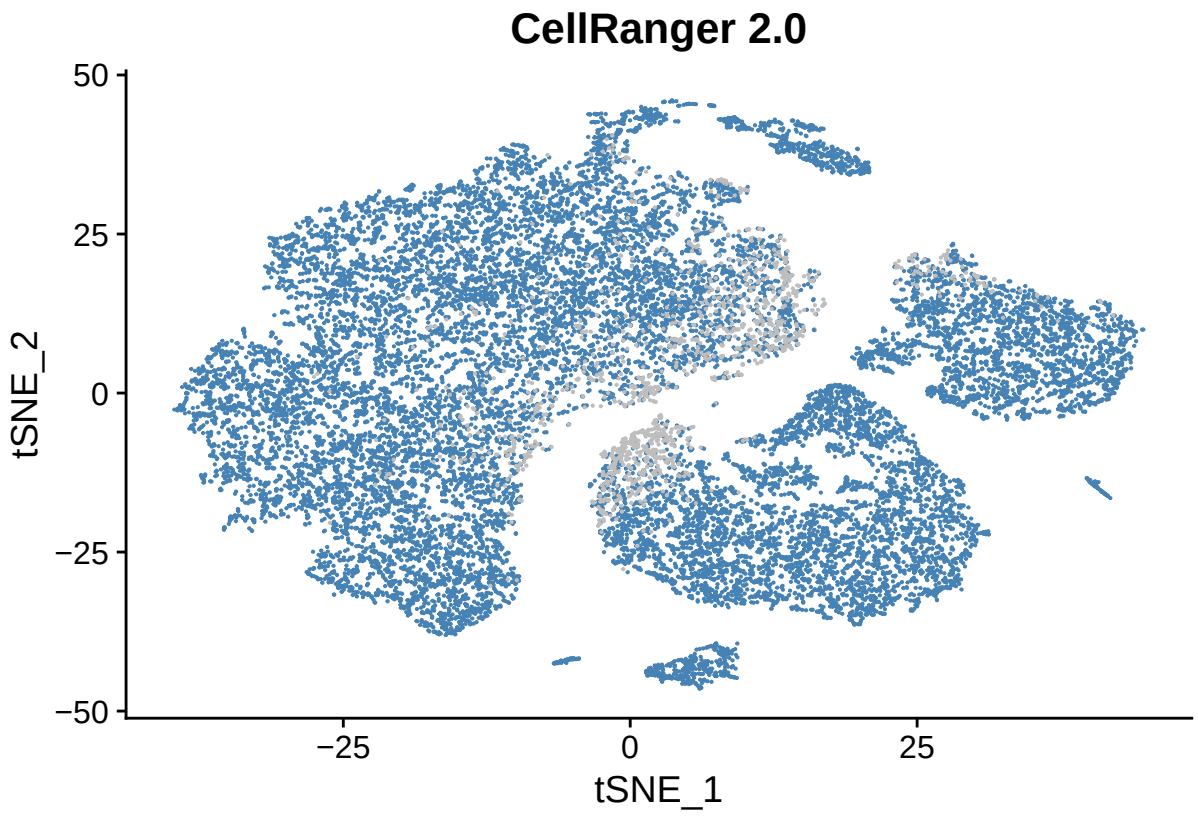


Figure S3C,D,E,F

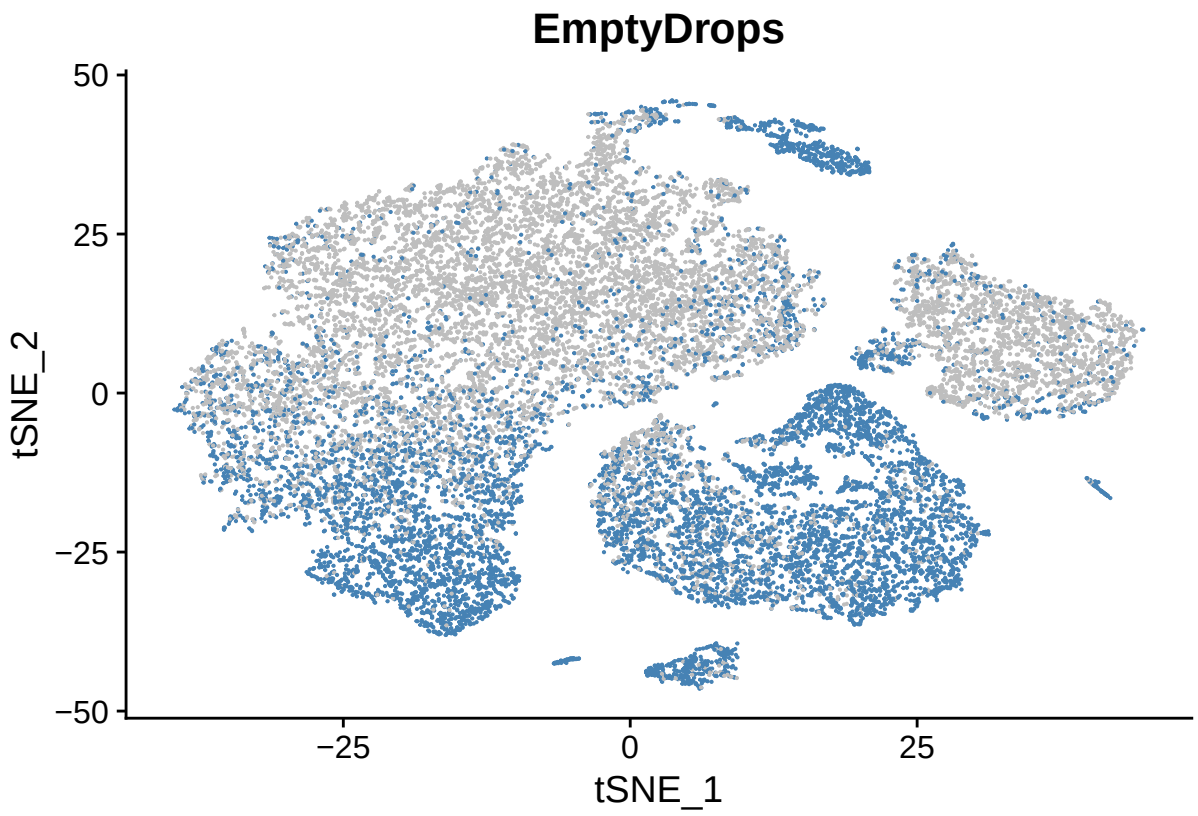
```
DimPlot(retained,reduction='tsne',group.by = 'emptynn',cols=c('grey','steelblue'))+
  NoLegend()+labs(title="EmptyNN")
```




```
DimPlot(retained,reduction='tsne',group.by = 'cellranger',cols=c('grey','steelblue'))+
  NoLegend()+labs(title="CellRanger 2.0")
```




```
DimPlot(retained,reduction='tsne',group.by = 'emptydrops',cols=c('grey','steelblue'))+
  NoLegend()+labs(title="EmptyDrops")
```




```
DimPlot(retained,reduction='tsne',group.by = 'cellbender',cols=c('grey','steelblue'))+
  NoLegend()+labs(title="CellBender")
```

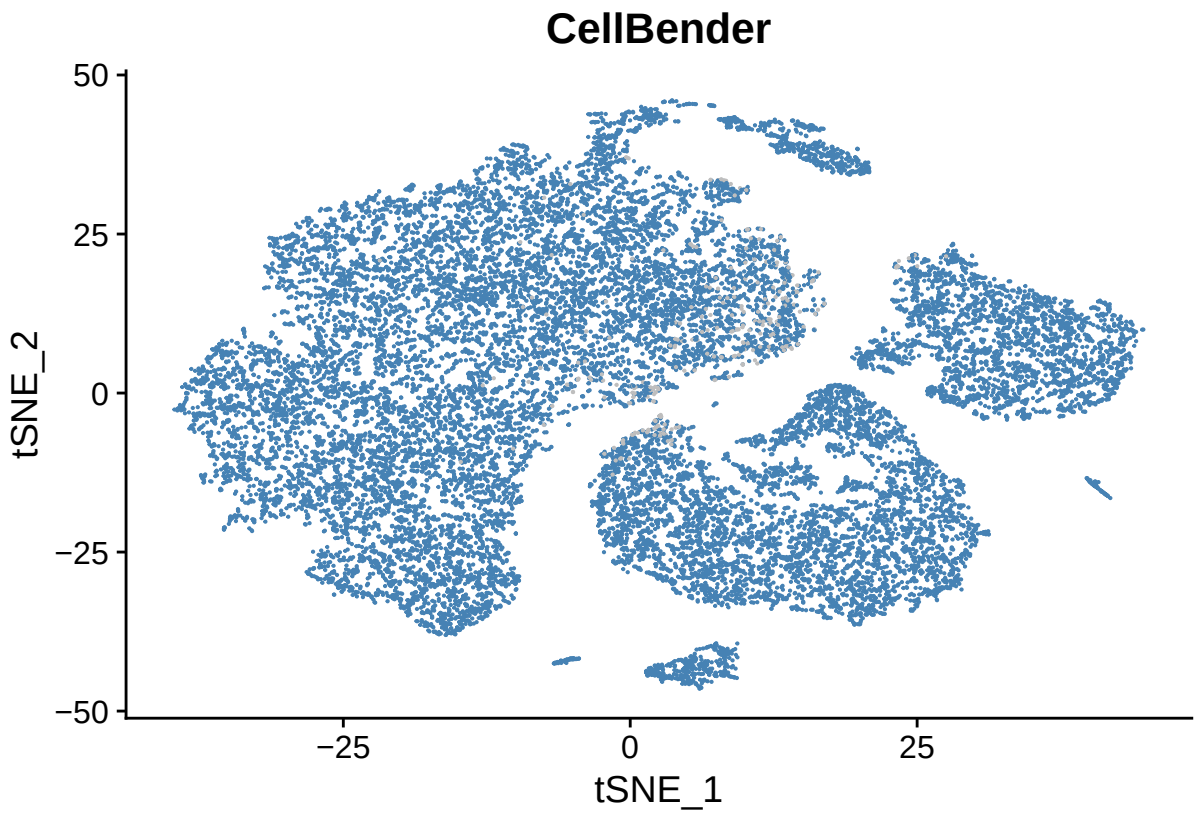


Figure 2F

```
sub <- retained[,sample(colnames(retained),10000)]
sub <- subset(sub,idents = c("Ambient","Doublets","pDCs"), invert=TRUE)
dist.mtx <- as.matrix(dist(sub@reductions$pca@cell.embeddings,
                           method = "euclidean",upper=TRUE))
sub$source <- 'recovered'
sub$source[rownames(sub@meta.data) %in% colnames(original_paper)] <- 'original'
sub$label <- Idents(sub)
# define function euclidean.calc()
euclidean.calc <- function(cluster){
  # recovered with original
  condition <- sub$label==cluster & sub$emptynn & sub$source == 'recovered'
  row <- rownames(sub@meta.data[condition,])
  column <- names(sub$source[sub$source=='original'])
  df <- dist.mtx[row,column]
  neg.m <- data.frame("dist"=colMeans(df))
  idx <- match(rownames(neg.m),names(sub$label))
  neg.m$label <- sub$label[idx]
  colnames(neg.m) <- c("dist",'label')
  # original with original
  condition <- sub$label==cluster & sub$source == 'original'
  column <- rownames(sub@meta.data[condition,])
```

```
  dist.b <- dist.mtx[column,column]
  orig.m <- data.frame('dist'=colMeans(dist.b),cluster)
  colnames(orig.m) <- c("dist",'label')
  # recover with recover
  condition <- sub$label==cluster & sub$emptynn & sub$source == 'recovered'
  column <- rownames(sub@meta.data[condition,])
  dist.b <- dist.mtx[column,column]
  recov.m <- data.frame('dist'=colMeans(dist.b),cluster)
  colnames(recov.m) <- c("dist",'label')
  tmp <- rbind(neg.m,orig.m,recov.m)
  tmp$group <- c(rep("recovered with original",nrow(neg.m)),
                 rep("original with original",nrow(orig.m)),
                 rep("recover with recover",nrow(recov.m)))
  return(tmp)
}
# calculate for NK cell type
tmp <- euclidean.calc("NK")
ggplot(tmp, aes(x=label, y=dist,fill=group)) +
  geom_boxplot(outlier.shape=NA)+
  xlab("original clusters")+ylab("Euclidean distance")
```

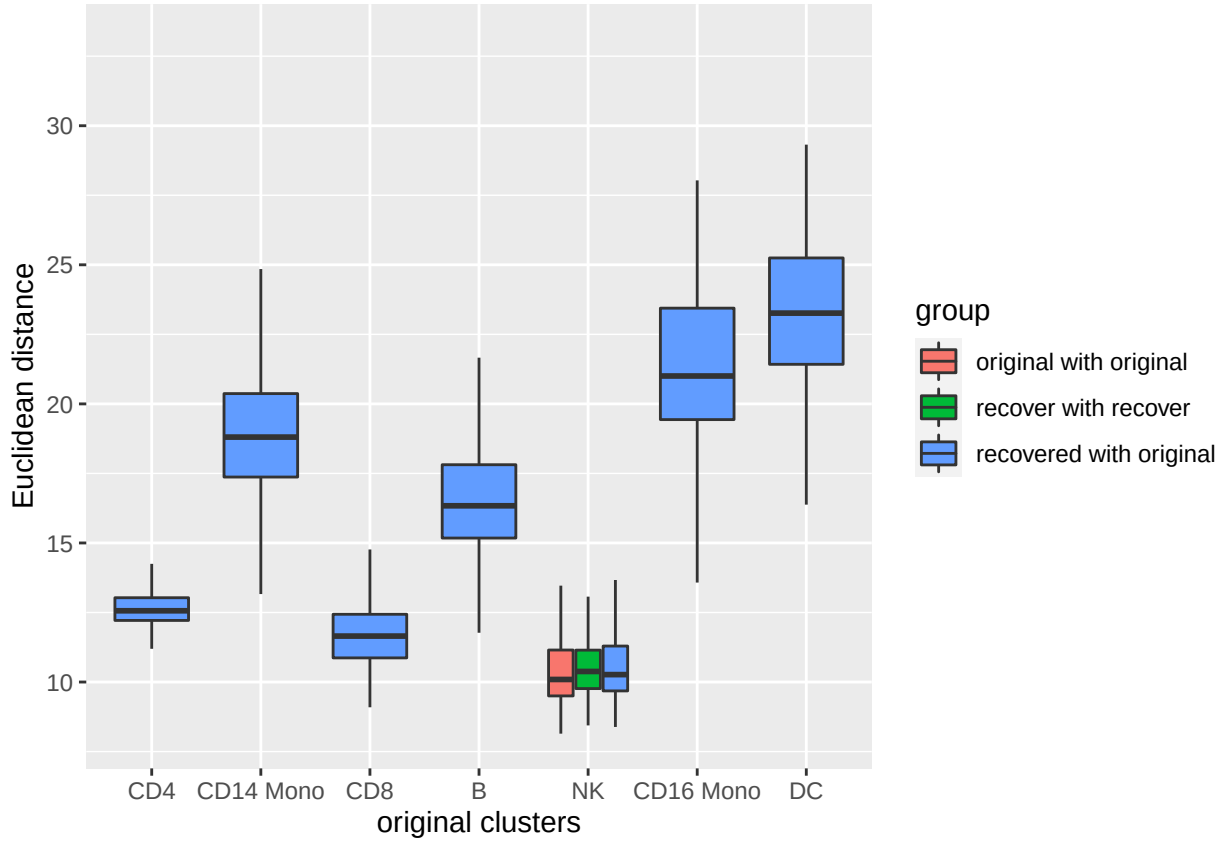


Figure 2G

```
tmp <- retained[,retained$emptynn==TRUE]
m1 <- FindMarkers(tmp, ident.1 = "CD4",ident.2 = "CD14 Mono",only.pos = TRUE,verbose=F)
tmp <- retained[,retained$cellranger==TRUE]
m2 <- FindMarkers(tmp, ident.1 = "CD4",ident.2 = "CD14 Mono",only.pos = TRUE,verbose=F)
tmp <- retained[,retained$emptydrops==TRUE]
m3 <- FindMarkers(tmp, ident.1 = "CD4",ident.2 = "CD14 Mono",only.pos = TRUE,verbose=F)
tmp <- retained[,retained$cellbender==TRUE]
m4 <- FindMarkers(tmp, ident.1 = "CD4",ident.2 = "CD14 Mono",only.pos = TRUE,verbose=F)
marker1 <- intersect(rownames(m1)[m1$p_val_adj<0.05],rownames(m2)[m2$p_val_adj<0.05])
marker2 <- intersect(rownames(m3)[m3$p_val_adj<0.05],rownames(m4)[m4$p_val_adj<0.05])
CD4.marker <- intersect(marker1,marker2)
df <- data.frame("logFC"=c(m1[CD4.marker,]$avg_logFC,
                           m2[CD4.marker,]$avg_logFC,
                           m3[CD4.marker,]$avg_logFC,
                           m4[CD4.marker,]$avg_logFC))
df$method <- c(rep("EmptyNN",length(CD4.marker)),rep("CellRanger 2.0",length(CD4.marker)),
               rep("EmptyDrops",length(CD4.marker)),rep("CellBender",length(CD4.marker)))
mu <- ddply(df, "method", summarise, grp.mean=mean(logFC))
ggplot(df, aes(x=logFC, color=method)) +
  geom_density()+xlab("Log2(foldchange)")+
  geom_vline(data=mu, aes(xintercept=grp.mean, color=method),linetype="dashed")
```

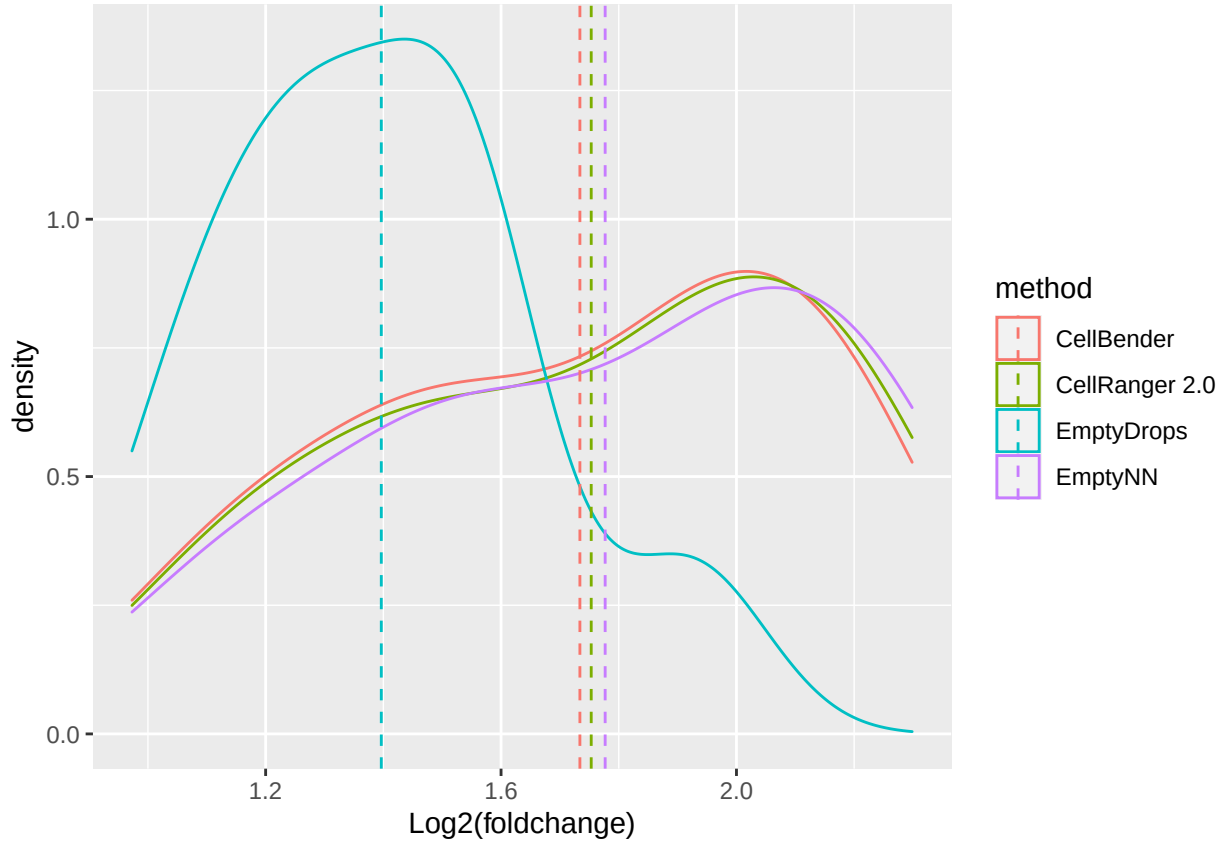


Figure 2H

```
label[,2] <- ifelse(label[,1]=="Negative","cell-free","cell-containing")
bcs <- rownames(nn.res$prediction)
rocobj1 <- roc(label[bcs,2], nn.res$prediction[bcs,'mean.crossval'])
```

```
## Setting levels: control = cell-containing, case = cell-free
```

```
## Setting direction: controls > cases
```

```
rocobj2 <- roc(label[names(ranger.keep),2], as.numeric(ranger.keep))
```

```
## Setting levels: control = cell-containing, case = cell-free
## Setting direction: controls > cases
```

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

```
## Loading required package: S4Vectors
```

```
## Loading required package: stats4
```

```
## Loading required package: BiocGenerics
```

```
## 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 object is masked from 'package:Matrix':
##
##     which
```

```
## 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, which.max, which.min

##
## Attaching package: 'S4Vectors'

## The following object is masked from 'package:plyr':
##
##     rename

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

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

```
rocobj3 <- roc(label[bcs,2], e.out[bcs,'FDR'])
```

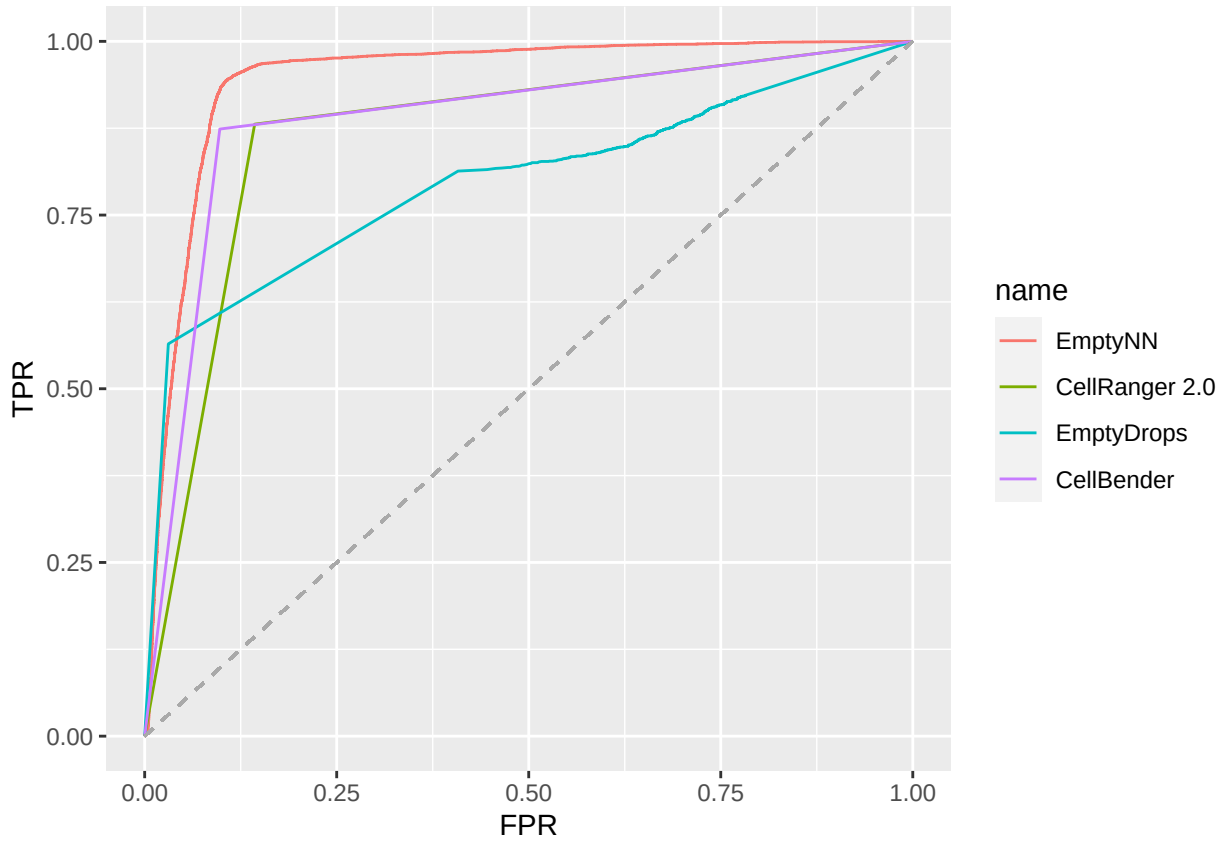
```
## Setting levels: control = cell-containing, case = cell-free

## Setting direction: controls > cases
```

```
rocobj4 <- roc(label[names(bender.keep),2], as.numeric(bender.keep))
```

```
## Setting levels: control = cell-containing, case = cell-free
## Setting direction: controls > cases
```

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


1.00
0.75
0.50
0.25
0.00
TPR
0.00
0.25
0.50
0.75
1.00
FPR
name
EmptyNN
CellRanger 2.0
EmptyDrops
CellBender