EMCN_Manuscript_Adult_WT_Choroid

Ben Thomson

2024-01-11

Open the datasets and general seurat objects

```
  data.folder <- "./orig.data/"

  folderlist <- dir(data.folder)
  dataset.list <- vector(mode = "list")
  soups.list <- vector(mode = "list")



  #loop through all the files and convert them to Seurat objects for easier analysis
  for (folder in folderlist) {
     print(paste("Opening dataset:", folder))
     counts.filtered <- Read10X_h5(filename = paste(data.folder, folder, "filtered_feature_bc_matrix.h5
     counts.raw <- Read10X_h5(filename = paste(data.folder, folder, "raw_feature_bc_matrix.h5", sep = "
     dataset.list[folder] <- CreateSeuratObject(counts = counts.filtered, project = folder, min.cells =
     dataset.list[[folder]] <- RenameCells(dataset.list[[folder]], add.cell.id = folder)

     soups.list[[folder]] <- SoupChannel(counts.raw, counts.filtered, calcSoupProfile = F)
     soups.list[[folder]] <- estimateSoup(soups.list[[folder]])

  }
```

```
## [1] "Opening dataset: P56_WT1"

## Warning in `[<-`(`*tmp*`, folder, value = CreateSeuratObject(counts =
## counts.filtered, : implicit list embedding of S4 objects is deprecated

## [1] "Opening dataset: P56_WT2"

## Warning in `[<-`(`*tmp*`, folder, value = CreateSeuratObject(counts =
## counts.filtered, : implicit list embedding of S4 objects is deprecated
```

After opening the files, we perform SoupX. First initially cluster the seurat objects.

```
for (folder in folderlist) {

  dataset.list[[folder]] <- Seurat::NormalizeData(dataset.list[[folder]])
  dataset.list[[folder]] <- Seurat::ScaleData(dataset.list[[folder]])
  dataset.list[[folder]] <- Seurat::FindVariableFeatures(dataset.list[[folder]])
```

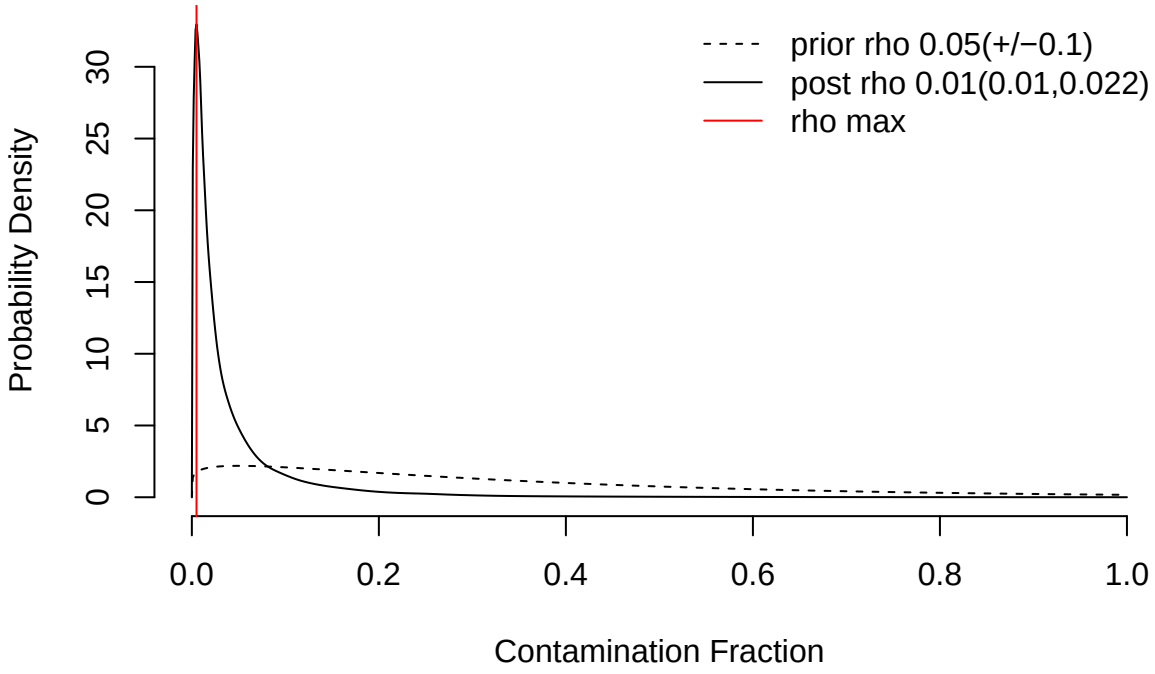
```
  dataset.list[[folder]] <- RunPCA(dataset.list[[folder]], npcs = 100)
  dataset.list[[folder]] <- FindNeighbors(dataset.list[[folder]], dims = 1:30)
  dataset.list[[folder]] <- FindClusters(dataset.list[[folder]], )
  dataset.list[[folder]] <- RunUMAP(dataset.list[[folder]], dims = 1:30)

  soups.list[[folder]] <- setClusters(soups.list[[folder]],setNames(dataset.list[[folder]]@meta.data$se
                                                                    rownames(dataset.list[[folder]]@me


  soups.list[[folder]] <- setDR(soups.list[[folder]], dataset.list[[folder]]@reductions$umap@cell.embed

  head(dataset.list[[folder]]@meta.data, 10)

  soups.list[[folder]] <- autoEstCont(soups.list[[folder]])

  head(soups.list[[folder]]$soupProfile[order(soups.list[[folder]]$soupProfile$est, decreasing = TRUE),

  dataset.list[[folder]] <- adjustCounts(soups.list[[folder]], roundToInt = TRUE)

  #regenerate seurat objects

  dataset.list[folder] <- CreateSeuratObject(counts = dataset.list[[folder]], project = folder, min.cel
  dataset.list[[folder]] <- RenameCells(dataset.list[[folder]], add.cell.id = folder)

  #add a col for % of genes which are mitochondrial
  dataset.list[[folder]][["percent.mt"]] <- PercentageFeatureSet(dataset.list[[folder]], pattern = "mt-


}
```

```
## Centering and scaling data matrix

## PC_ 1
## Positive:  Akap12, Emcn, Col4a1, Clec2d, Srgn, Hilpda, Ly6e, Fxyd5, Cxcl16, Slfn5
##     Dcn, Bgn, Icam1, Itm2a, Gsn, Cxcl1, S1pr1, Ptgs2, Col1a2, Tgm2
##     Col4a2, Lrrc32, Tpm2, Col3a1, Serpinf1, Nupr1, Cd24a, Cdkn1c, Myh9, Procr
## Negative:  Slc16a8, Lrat, Rlbp1, Rdh5, Rpe65, Rgr, Cltrn, Pon1, Car14, Ermn
##     Rrh, Cspg5, Krt18, Rd3l, Ttr, Col9a3, Rsph10b, Acsl6, Slc4a5, Dapl1
##     Slc6a13, Mt3, Krt8, Trpm3, Tmem72, Prxl2a, Car12, Tmem56, Pla2g5, Hist1h4i
## PC_ 2
## Positive:  Dcn, Col1a2, Bgn, Sparc, Serpinf1, Pcolce, Col1a1, Gsn, Lox, Ogn
##     Pdgfrl, Prelp, Igfbp6, S100a6, Cdkn1c, Itgbl1, Ecrg4, Col12a1, Ctsk, Ccn2
##     Thbs2, Cpxm1, Col3a1, Fndc1, Islr, Mfap5, Clec11a, Abi3bp, Fmod, Mfap4
## Negative:  Cd14, Spi1, Tyrobp, Slfn2, Srgn, Bcl2a1b, Fcer1g, Lilrb4a, Ctss, Lcp1
##     Plek, Cd83, Lilr4b, Bcl2a1d, Cd86, Laptm5, Cd74, Cd68, C1qb, Ptafr
##     Cd52, Nlrp3, C1qa, Lyz2, C1qc, Cd300c2, Lrrc25, Ncf4, Lyn, C5ar1
## PC_ 3
## Positive:  Tyrobp, Spi1, Ctss, C1qb, Bcl2a1b, C1qa, Lilrb4a, C1qc, Cd68, Lyz2
##     Fcer1g, Plek, Bcl2a1d, Cd83, Cd52, Cd86, Cd300c2, Laptm5, Ms4a7, Lilr4b
##     Lrrc25, Nlrp3, Ncf4, Ptafr, Ly86, Aif1, Lst1, C5ar1, Arl4c, Fyb
## Negative:  Flt1, Cdh5, Plvap, Egfl7, Rasip1, Kdr, Cldn5, Pecam1, Ctla2a, Ptprb
##     Podxl, Adgrl4, Mmrn2, Cyyr1, Adgrf5, Sema7a, Tie1, Robo4, Ecscr, Acer2
##     Shank3, Exoc3l2, Esam, Adgrg1, Icam2, Rapgef5, Grrp1, S1pr1, Sox18, Upp1
```

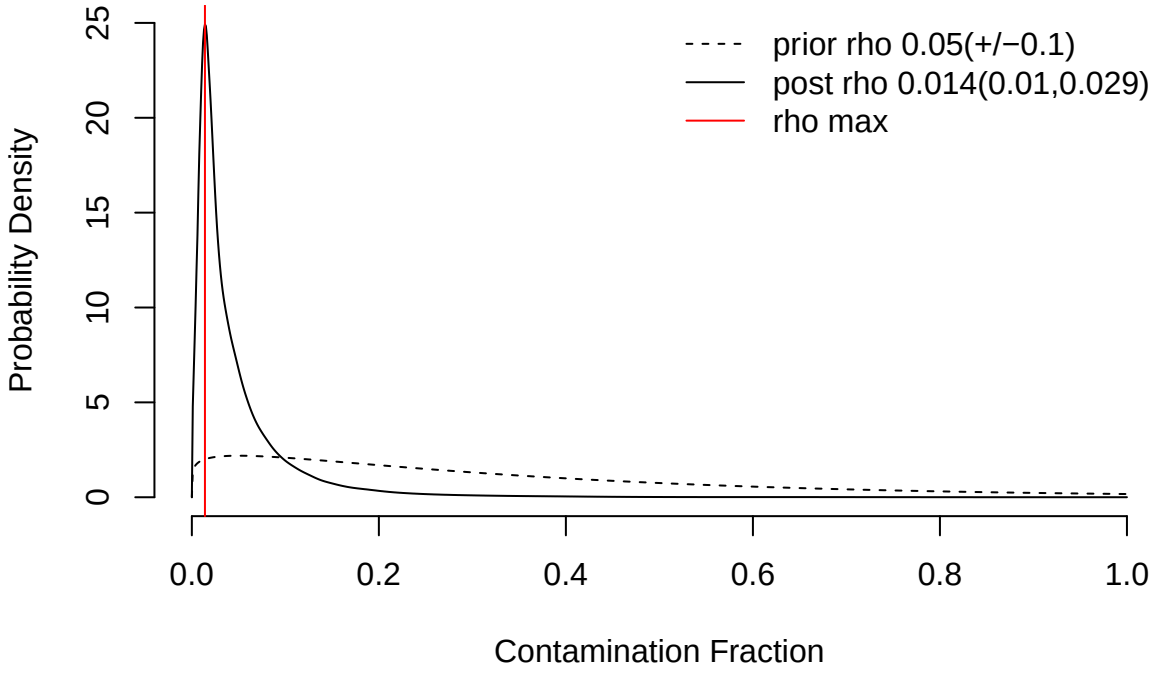
```
## PC_ 4
## Positive:  Ednrb, Pmel, Mlph, Mlana, Tyrp1, Syt4, Slc45a2, S100b, Gpnmb, Tyr
##     Syngr1, Gjb2, Scn8a, Slc24a5, Dct, Trpm1, Hpse, Gm3776, Rab38, Scrg1
##     Car6, Gsta1, Myo5a, Bace2, Nrcam, Cort, Gstp1, Tecpr1, Fmn1, Mgll
## Negative:  Pcolce, Clu, Dcn, Col1a2, Serpinf1, Bgn, Col8a2, Col8a1, Ogn, Col1a1
##     Cst3, Igfbp6, Islr, Cpxm2, Ecrg4, Lox, Thbs1, Itgbl1, Pdgfrl, Col12a1
##     Clec11a, Olfml3, Fmod, Igfbp2, Fndc1, Cdkn1c, Ccn2, Gas6, Cpxm1, Sostdc1
## PC_ 5
## Positive:  Fmod, Ecrg4, Col12a1, Igfbp6, Itgbl1, Chad, Clec11a, Tnmd, Sfrp2, Prelp
##     Col6a3, S100a10, Fn1, Comp, Serpinf1, Fndc1, Cilp2, Col8a2, Cpxm2, Chil1
##     Gas6, Igfbp2, Col16a1, Cyp2f2, Matn4, Col1a1, Omd, Olfml3, Svep1, Nell2
## Negative:  Kcnj8, Des, Rasd1, Gja4, Procr, Cfh, Rdh10, Gm13889, Rgs5, Tmem37
##     Fam107b, Epas1, Lrrc32, Rgs4, Emcn, Plac8, Ndufa4l2, Il6, Bdnf, Filip1l
##     Ifitm1, Akap12, Alkal2, Ccl11, Pcp4l1, Adamts9, Atp1b2, Myh11, Cd24a, Ifit3

## Computing nearest neighbor graph

## Computing SNN

## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 8993
## Number of edges: 312725
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.9067
## Number of communities: 26
## Elapsed time: 1 seconds

## 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

## 15:08:41 UMAP embedding parameters a = 0.9922 b = 1.112

## 15:08:41 Read 8993 rows and found 30 numeric columns

## 15:08:41 Using Annoy for neighbor search, n_neighbors = 30

## 15:08:41 Building Annoy index with metric = cosine, n_trees = 50

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

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

## **************************************************|
## 15:08:42 Writing NN index file to temp file C:\Users\brt221\AppData\Local\Temp\RtmpAvHLIL\file914c2f
## 15:08:42 Searching Annoy index using 1 thread, search_k = 3000
## 15:08:44 Annoy recall = 100%
## 15:08:45 Commencing smooth kNN distance calibration using 1 thread with target n_neighbors = 30
## 15:08:47 Initializing from normalized Laplacian + noise (using irlba)
```

```
## 15:08:49 Commencing optimization for 500 epochs, with 378492 positive edges
## 15:09:06 Optimization finished
## 2996 genes passed tf-idf cut-off and 443 soup quantile filter.  Taking the top 100.
## Using 1818 independent estimates of rho.
## Estimated global rho of 0.01

## Warning in sparseMatrix(i = out@i[w] + 1, j = out@j[w] + 1, x = out@x[w], :
## 'giveCsparse' is deprecated; setting repr="T" for you

## Expanding counts from 26 clusters to 8993 cells.

## Warning in `[<-`(`*tmp*`, folder, value = CreateSeuratObject(counts =
## dataset.list[[folder]], : implicit list embedding of S4 objects is deprecated

## Centering and scaling data matrix
## PC_ 1
## Positive:  Col4a1, Tm4sf1, Clec2d, Akap12, Emcn, Col1a2, Gsn, Slfn5, Dcn, Ly6e
##     Srgn, Col3a1, Cxcl16, Col4a2, Fxyd5, Ptgs2, Adamts4, Col1a1, Cxcl1, S100a6
##     Serpinf1, Hilpda, Ets1, Iigp1, Itm2a, Gbp2, Adamts9, Nupr1, Tpm2, Slfn2
## Negative:  Lrat, Slc16a8, Rlbp1, Ermn, Rdh5, Car14, Pon1, Rrh, Krt18, Cltrn
##     Rd3l, Cspg5, Rpe65, Rgr, Col9a3, Mt3, Tmem72, Slc4a5, Krt8, Acsl6
##     Trpm3, Dapl1, Rsph10b, Hist1h4i, Ttr, Slc6a13, Tshr, Prxl2a, Stra6, Itgb8
## PC_ 2
## Positive:  Spi1, Ctss, Tyrobp, Bcl2a1b, Fcer1g, Cd83, Lilrb4a, Plek, Laptm5, Lcp1
##     Cd14, Cd86, Slfn2, Csf1r, C1qc, Cybb, C1qa, Lilr4b, C1qb, Nlrp3
##     Cd52, Cd68, Bcl2a1d, Pik3ap1, Cd74, C3ar1, Mpeg1, Cd300c2, Lyz2, Ptafr
## Negative:  Sparc, Col1a2, Dcn, Col1a1, Serpinf1, Lox, Tm4sf1, Col3a1, Ccn2, Prelp
##     Gsn, Thbs1, Col12a1, Fndc1, Col4a1, Abi3bp, Cdkn1c, Itgbl1, Tpm2, Crispld2
##     Islr, Mfap5, S100a6, Lbp, Nbl1, Cpxm1, Col8a1, Clec11a, Ptn, Ecrg4
## PC_ 3
## Positive:  Cdh5, Flt1, Adgrl4, Plvap, Kdr, Pecam1, Rasip1, Podxl, Egfl7, Mmrn2
##     Ptprb, Ctla2a, Cldn5, Cyyr1, Adgrf5, Sema7a, Robo4, Rapgef5, Ecscr, Esam
##     Acer2, Adgrg1, Sox7, Exoc3l2, Tie1, Shank3, Icam2, Grrp1, Inhbb, Cd93
## Negative:  Serpinf1, Col1a2, Dcn, Gsn, Cryab, Cst3, Col1a1, S100a6, Prelp, Col12a1
##     Dkk3, Itgbl1, S100b, Lox, Ecrg4, Myo5a, Fmod, Fndc1, Igfbp6, Clec11a
##     Gsta4, Olfml3, Fxyd6, Abi3bp, Anxa1, Col6a3, Tnmd, Mfap5, Gstp1, Chsy3
## PC_ 4
## Positive:  Ednrb, Mlph, Pmel, Syt4, Mlana, Tyrp1, Slc45a2, Gpnmb, S100b, Tyr
##     Nrcam, Syngr1, Gjb2, Hpse, Car6, Dct, Rab38, Slc24a5, Scn8a, Cdk2
##     Fmn1, Trpm1, Tecpr1, Myo5a, Chsy3, Oca2, Rasgrp3, Cdh1, Plp1, Sytl2
## Negative:  Col1a2, Dcn, Serpinf1, Clu, Col8a1, Col8a2, Col1a1, Thbs1, Islr, Gas6
##     Lox, Igfbp6, Col12a1, Itgbl1, Ecrg4, Cpxm2, Fndc1, Cst3, Ccn2, Fmod
##     Clec11a, Slc6a6, Lbp, Olfml3, Chil1, Col6a3, Mfap5, Penk, Abi3bp, Svep1
## PC_ 5
## Positive:  Fmod, Ecrg4, Col12a1, Itgbl1, Igfbp6, Chad, Tnmd, Sfrp2, S100a10, Clec11a
##     Prss12, Fn1, Col6a3, Gas6, Svep1, Cd109, Col8a2, Comp, Chil1, Ltbp2
##     Prelp, Igfbp2, Omd, Penk, Fndc1, Serpinf1, Cyp2f2, Srpx2, Cilp2, Piezo2
## Negative:  Adamts4, Rasd1, Des, Kcnj8, Procr, Fam107b, Ndufa4l2, Gja4, Rgs5, Gm13889
##     Rdh10, Emcn, Akap12, Adamts9, Bdnf, Lrrc32, Rgs4, Serpini1, Il6, Plac8
##     Filip1l, Alkal2, Ace2, Atp1b2, Apold1, Myh11, Coch, Cd24a, Map3k20, Mrvi1
## Computing nearest neighbor graph
## Computing SNN
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 14107
## Number of edges: 495554
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.9133
## Number of communities: 27
## Elapsed time: 2 seconds

## 15:10:40 UMAP embedding parameters a = 0.9922 b = 1.112
## 15:10:40 Read 14107 rows and found 30 numeric columns
## 15:10:40 Using Annoy for neighbor search, n_neighbors = 30
## 15:10:40 Building Annoy index with metric = cosine, n_trees = 50
## 0%   10   20   30   40   50   60   70   80   90   100%
## [----|----|----|----|----|----|----|----|----|----|
## **************************************************|
## 15:10:42 Writing NN index file to temp file C:\Users\brt221\AppData\Local\Temp\RtmpAvHLIL\file914c50
## 15:10:42 Searching Annoy index using 1 thread, search_k = 3000
## 15:10:45 Annoy recall = 100%
## 15:10:46 Commencing smooth kNN distance calibration using 1 thread with target n_neighbors = 30
## 15:10:49 Initializing from normalized Laplacian + noise (using irlba)
## 15:10:51 Commencing optimization for 200 epochs, with 601454 positive edges
## 15:11:02 Optimization finished
## 3327 genes passed tf-idf cut-off and 383 soup quantile filter.  Taking the top 100.
## Using 1613 independent estimates of rho.
## Estimated global rho of 0.01
```



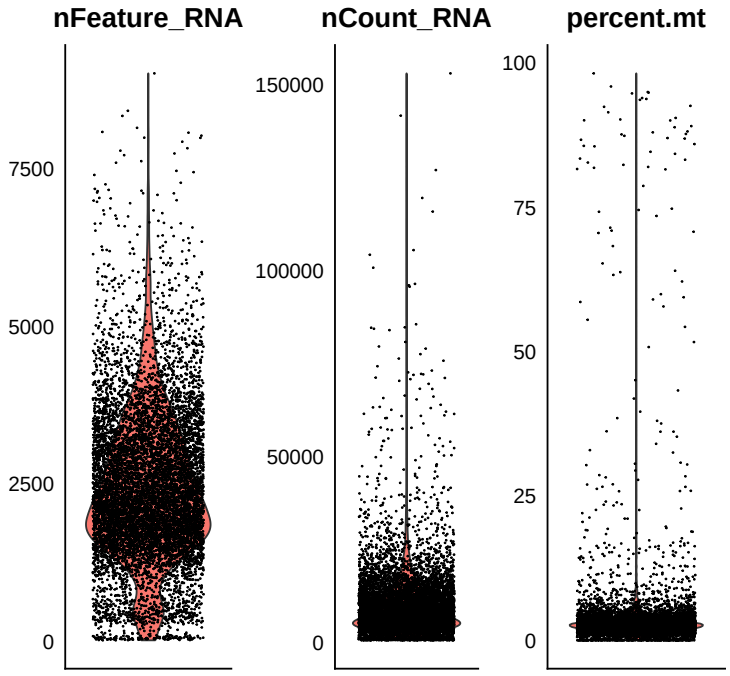

```
## Warning in sparseMatrix(i = out@i[w] + 1, j = out@j[w] + 1, x = out@x[w], :
## 'giveCsparse' is deprecated; setting repr="T" for you

## Expanding counts from 27 clusters to 14107 cells.

## Warning in `[<-`(`*tmp*`, folder, value = CreateSeuratObject(counts =
## dataset.list[[folder]], : implicit list embedding of S4 objects is deprecated
```



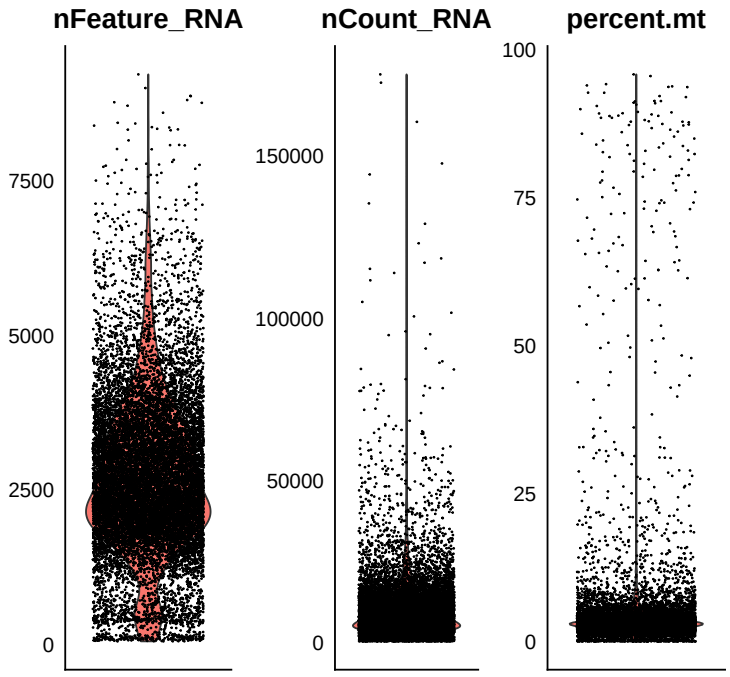
Next, we perform some basic QC to check the data. For initial analysis we removed all cells with <500 or >5000 unique features as well as cells with >10% mitochondrial RNA.

```
QC.Vln.list <- lapply(dataset.list, function(v) {VlnPlot(v, features = c("nFeature_RNA", "nCount_RNA",

QC.Vln.list <- lapply(QC.Vln.list, function(v) {lapply(v, function(i) { i <- i + theme(axis.text.x = el

dataset.list <- lapply(dataset.list, function(v) {subset(v, subset = nFeature_RNA > 500 & nFeature_RNA

plot_grid(plotlist = lapply(QC.Vln.list, function(x) {plot_grid(plotlist = x, ncol = 4)}), ncol = 1, la
```

P56_WT1



P56_WT2



After basic QC, doublet detection was performed using the scDblFinder pipeline. Germain P (2020). scDblFinder: scDblFinder. R package version 1.4.0, https://github.com/plger/scDblFinder. Doublet detection and removal using the standard perameters of 1% doublets per 1000 droplets sequenced.

```
scDbl.features <- 2000
scDbl.dims <- 12


#for each dataset, convert to SingleCellExperiment, run scDblFinder and return to Seurat object. To avo

dataset.list <- lapply(dataset.list, function(x) {
                    sce <- as.SingleCellExperiment(x)
                    sce <- scDblFinder(sce, verbose=F, nfeatures = scDbl.features, dims = scDbl.dims)
                    sce <- as.Seurat(sce)
                    x <-  AddMetaData(x, col.name = 'doubletScore', metadata = sce$scDblFinder.score )
                    x <-  AddMetaData(x, col.name = 'doubletClass', metadata = sce$scDblFinder.class )
                }
                )


calls <-  lapply(dataset.list, function(x) { x@meta.data %>% as.data.table } )

scDblFinder.calls <- rbindlist(calls)

kbl(scDblFinder.calls[, .N, by = c("orig.ident", "doubletClass")], caption = "scDblFinder predictions")
kable_minimal(full_width = FALSE, position = "left", latex_options = "hold_position")
```

Table 1: scDblFinder predictions

orig.ident	doubletClass	N
P56_WT1	singlet	7477
P56_WT1	doublet	714
P56_WT2	singlet	11395
P56_WT2	doublet	1205

Droplets identified as "singlets" were used for subsequent analysis and doublets were discarded.

```
  dataset.list <- lapply(dataset.list, function(x) {x <- subset(x, subset = doubletClass == "singlet")}
```

We then integrated the samples using RPCA as implemented the integration pipeline in Seurat. Only variable features present in all samples were selected for dataset integration.

```
  dataset.list <- lapply(dataset.list, function(v) {
    v <- FindVariableFeatures(object = v, selection.method = "vst", nfeatures = 8000, verbose = FALSE)
    v <- NormalizeData(v)
  })

  #shared features
  shared.var.features <- SelectIntegrationFeatures(object.list = dataset.list, nfeatures = 5000)

  integration.features <- shared.var.features
```

```
  dataset.list <- lapply(dataset.list, function(v) {
    v <- ScaleData(object = v, features = integration.features, verbose = FALSE)
    v <- RunPCA(object = v, features = integration.features, npcs = 35, verbose = FALSE)
  })

  #find anchors for integration
  anchors <- FindIntegrationAnchors(object.list = dataset.list,
                                    dims = 1:35, anchor.features = integration.features, reduction = "
```

```
## Scaling features for provided objects

## Computing within dataset neighborhoods

## Finding all pairwise anchors

## Projecting new data onto SVD
## Projecting new data onto SVD

## Finding neighborhoods

## Finding anchors

##  Found 14747 anchors
```

```
#intrgrate datasets
  integrated <- IntegrateData(anchorset = anchors, verbose = T)
```

```
## Merging dataset 1 into 2

## Extracting anchors for merged samples

## Finding integration vectors

## Finding integration vector weights

## Integrating data
```

```
#update the metadata based on original filenames

Idents(integrated) <- gsub(pattern=".+_", replacement="",
                           gsub(pattern = '[2-9]*', replacement = "", integrated$orig.ident))
  integrated[["genotype"]] <-Idents(integrated)



#remove unneeded variables

rm(anchors, counts.filtered, counts.raw, soups.list, dataset.list, calls, scDblFinder.calls)
gc()
```

```
##             used   (Mb) gc trigger   (Mb)   max used    (Mb)
## Ncells  10601624  566.2   23900298 1276.5   23900298  1276.5
## Vcells 201265658 1535.6 1244285316 9493.2 1537188829 11727.9
```

Following CCA integration, the dataset was scaled and principal components were calculated for plotting and clustering of the integrated dataset.

```
  full.dataset.dims <- 50

  DefaultAssay(object = integrated) <- "integrated"


  #integrated <- NormalizeData(integrated, assay = "integrated")
  integrated <- NormalizeData(integrated, assay = "RNA")
  integrated <- ScaleData(integrated, assay = "integrated")
```

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

```
  integrated <- ScaleData(integrated, assay = "RNA")
```

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

```
  #run the PCA using the same features used for integration
  integrated <- RunPCA(object = integrated, npcs = 100, verbose = T,
                       assay = "integrated")
```

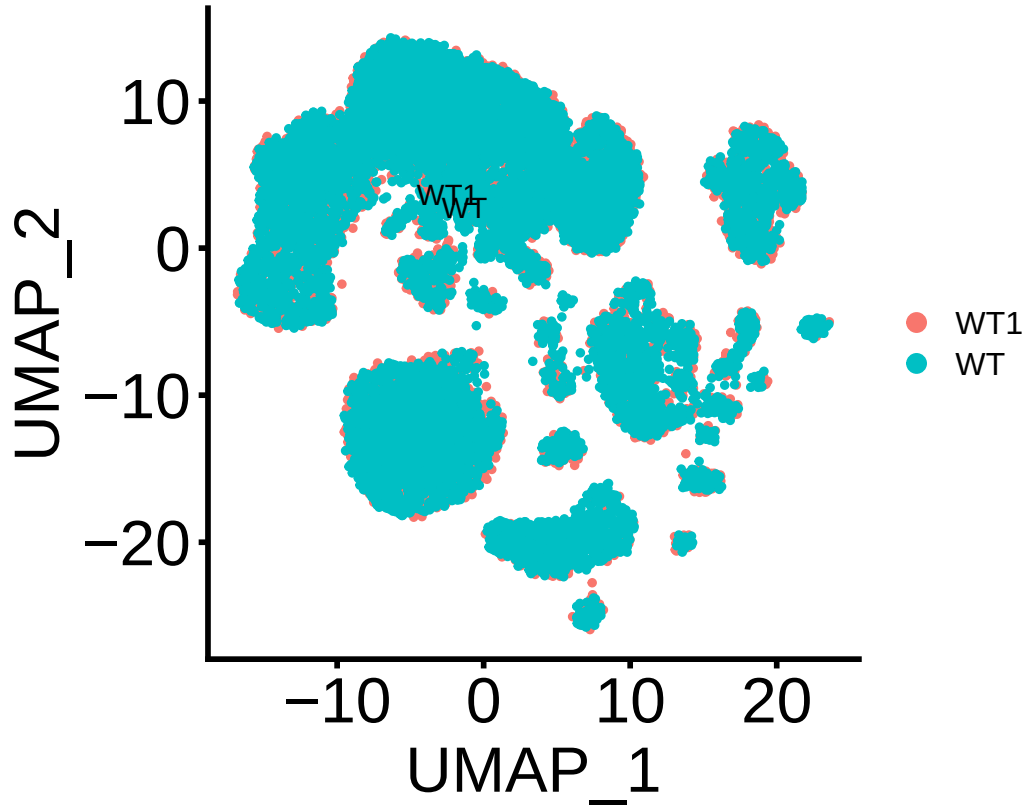
```
## PC_ 1
## Positive:  Ptgds, Dct, Chchd10, Slc24a5, Tyrp1, Kcnj13, Syngr1, S100a1, Mlana, Gpnmb
##     Mgll, Pmel, Tyr, Slc45a2, Met, Mlph, Oca2, Vegfb, Syt4, Trpm1
##     Ednrb, Myo5a, Fmn1, Tspan10, Rasgrp3, Gsta2, Tecpr1, Rab38, Tmod1, Nrcam
## Negative:  Igfbp7, Mgp, Igfbp5, Fstl1, Errfi1, Bgn, Igfbp4, Serpine2, Rarres2, Col1a2
##     Dcn, Emp1, Lum, Cebpd, Colec12, Serping1, Col1a1, Fgfr1, Col3a1, Col6a1
##     Ccdc80, Serpinf1, Plpp3, Mat2a, Cd200, Ugdh, Cdh11, Smoc1, Tm4sf1, Ltbp4
## PC_ 2
## Positive:  Tyrobp, Spi1, Ctss, Slfn2, Lcp1, Fcer1g, Laptm5, Cd83, Cd14, Plek
##     Bcl2a1b, Lilrb4a, Cd52, Cd86, Cybb, Cd74, Lilr4b, Nlrp3, C1qb, Lyz2
##     Arl4c, C1qa, Mpeg1, C1qc, Srgn, Pik3ap1, Lyn, Cd300c2, Bcl2a1d, Ptafr
## Negative:  Slc25a4, Bsg, Cryab, Tspan3, Rbp1, Gsta4, Ldhb, Aebp1, Ptgds, Sparc
##     Timp2, Gpx4, Kcnj13, Gstp1, Timp3, Dclk1, Chchd10, Fbln5, Mtch1, Met
##     Pebp1, Car14, Dct, Mgll, Dkk3, Hist1h2bc, Slc24a5, Enpp5, Smpd1, Clu
## PC_ 3
## Positive:  Sparcl1, Ednrb, Mlph, Pmel, Syt4, S100b, Phlda1, Nrcam, Dlc1, Gjb2
##     Car6, Hpse, Chsy3, Cdk2, Slc45a2, Pakap.1, Adamts1, Mlana, Scn8a, Fmn1
##     Tinagl1, Rab38, Mcoln3, Cdh1, Tecpr1, Plp1, Tyrp1, Gpnmb, Tyr, Scrg1
## Negative:  Slc16a8, Rlbp1, Rgr, Rpe65, Ermn, Krt18, Pon1, Cltrn, Ttr, Rd3l
##     Col9a3, Dapl1, Cspg5, Mt3, Krt8, Slc4a5, Slc6a13, Rrh, Tmem72, Trpm3
##     Rsph10b, Lrat, Acsl6, Itgb8, Rdh5, Tshr, Clu, Car12, Clic6, Slc24a1
## PC_ 4
## Positive:  Fmod, Col12a1, Itgbl1, Ecrg4, Aebp1, Serpinf1, Lgals1, Igfbp6, Timp2, Fbln5
##     Col6a3, Prelp, Fndc1, Tnmd, Clec11a, Rnase4, Chad, Col11a1, Sfrp2, S100a6
##     Svep1, Col16a1, Comp, Prss12, Col1a1, Dcn, Penk, Ahnak, Col1a2, Ltbp2
## Negative:  Epas1, Emcn, Esam, Adamts9, Apold1, Flt1, Arrdc3, Lrrc32, Adgrf5, Cdh5
```

```
##     Rasip1, Adgrl4, Cyyr1, Pecam1, Plvap, S1pr1, Ctla2a, Egfl7, Mmrn2, Kdr
##     Podxl, Cldn5, Tie1, Slc1a1, Ptprb, Cd93, Akap12, Slco1a4, Cfh, Stc1
## PC_ 5
## Positive:  Cdh5, Flt1, Ptprb, Adgrl4, Plvap, Rasip1, Kdr, Pecam1, Podxl, Egfl7
##     Mmrn2, Ctla2a, Cldn5, Sema7a, Ecscr, Cyyr1, Rapgef5, Robo4, Adgrf5, Plpp1
##     Adgrg1, Grrp1, Sox7, Icam2, Acer2, Exoc3l2, Shank3, Tek, Inhbb, Tnfsf10
## Negative:  Phlda1, Ndufs2, Ednra, Nr2f1, Lsamp, Tbx3, Rgs7bp, Metrnl, Medag, Alpl
##     Adamts4, Gucy1b1, Edn3, Des, Gpc3, Cfh, Kcnj8, Adamts1, Rasd1, Fam107b
##     Gucy1a1, Lamc3, Col23a1, Irs2, Notch3, Brinp1, Abcc9, S1pr3, Ngf, Pdgfrb
```

UMAP projection was then performed on the full dataset

```
DefaultAssay(object = integrated) <- "integrated"

integrated <- RunUMAP(object = integrated, dims = 1:50,
                      min.dist = 1.5, n.neighbors = 800, verbose = FALSE, assay = "integrated")


DimPlot(object = integrated, reduction = "umap", label = TRUE, pt.size = 1,
        label.size = 4) & coord_fixed() & theme(aspect.ratio=1) &
              theme(plot.title = element_text(size = 26, face = "italic")) &
              theme(axis.line = element_line(color="black", linewidth = 1), axis.ticks = element_line
              theme(axis.title = element_text(size = 24), axis.text = element_text(size = 24))
```

```
## Warning: The 'size' argument of 'element_line()' is deprecated as of ggplot2 3.4.0.
## i Please use the 'linewidth' argument instead.
```

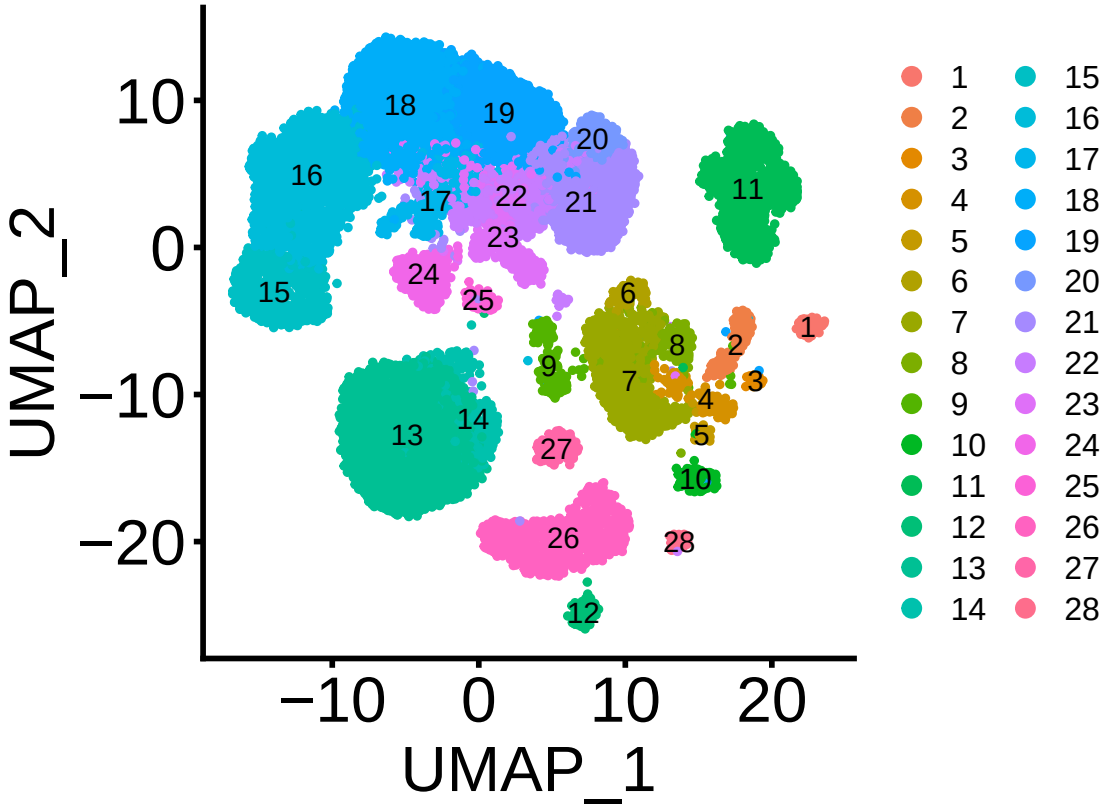


CCA-aligned data was then clustered at low resolution using a Louvain algorithm implemented in Seurat.

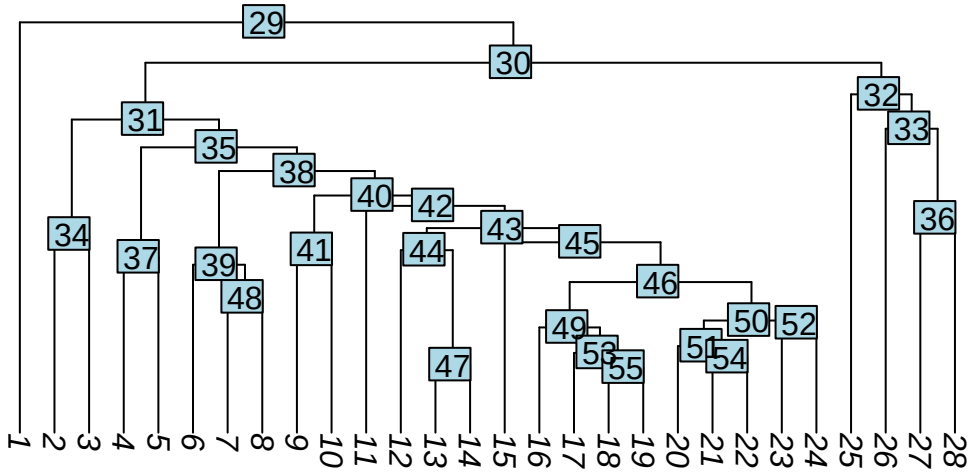
```
DefaultAssay(integrated) <- "integrated"
integrated <- FindNeighbors(object = integrated, dims = 1:50,
                            verbose = FALSE)

integrated <- FindClusters(object = integrated, resolution = 0.6, verbose = FALSE)

#Reorder cluster identities based on cluster tree
integrated <- BuildClusterTree(integrated, reorder.numeric = TRUE, reorder = TRUE,
                               verbose = T, assay = "integrated", dims = 1:50)
```

```
## Reordering identity classes and rebuilding tree
```

```
#Plot of cluster identities mapped on full dataset
DimPlot(object = integrated, reduction = "umap", label = TRUE, pt.size = 1,
        label.size = 4) & coord_fixed() & theme(aspect.ratio=1) &
              theme(plot.title = element_text(size = 26, face = "italic")) &
              theme(axis.line = element_line(color="black", linewidth = 1), axis.ticks = element_line
              theme(axis.title = element_text(size = 24), axis.text = element_text(size = 24))
```




```
PlotClusterTree(integrated)
```



Now we save the data for futher analysis in other scripts.

```
saveRDS(integrated, "./WT_8_WT_dataset_for_EMCN_MS_2-1-24.rds")
```

Current R session.

```
sessionInfo()
```

```
## R version 4.2.0 (2022-04-22 ucrt)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19045)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.utf8
## [2] LC_CTYPE=English_United States.utf8
## [3] LC_MONETARY=English_United States.utf8
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.utf8
##
## attached base packages:
## [1] stats4    stats     graphics  grDevices utils     datasets  methods
## [8] base
##
```

```
## other attached packages:
##  [1] tibble_3.1.8                reshape2_1.4.4
##  [3] purrr_1.0.1                 magrittr_2.0.3
##  [5] SingleCellExperiment_1.18.1 SummarizedExperiment_1.26.1
##  [7] Biobase_2.56.0              GenomicRanges_1.48.0
##  [9] GenomeInfoDb_1.32.4         IRanges_2.30.1
## [11] S4Vectors_0.34.0            BiocGenerics_0.42.0
## [13] MatrixGenerics_1.8.1        matrixStats_0.63.0
## [15] Matrix.utils_0.9.8          Matrix_1.5-3
## [17] SoupX_1.6.2                 kableExtra_1.3.4
## [19] scDblFinder_1.10.0          data.table_1.14.8
## [21] clustree_0.5.0              ggraph_2.1.0
## [23] gridExtra_2.3               ggplot2_3.4.1
## [25] cowplot_1.1.1               sctransform_0.3.5
## [27] gdata_2.18.0.1              dplyr_1.1.0
## [29] SeuratObject_4.1.3          Seurat_4.3.0
##
## loaded via a namespace (and not attached):
##   [1] utf8_1.2.3                spatstat.explore_3.0-6
##   [3] reticulate_1.28           tidyselect_1.2.0
##   [5] htmlwidgets_1.6.1         grid_4.2.0
##   [7] BiocParallel_1.30.4       Rtsne_0.16
##   [9] munsell_0.5.0             ScaledMatrix_1.4.1
##  [11] codetools_0.2-18          ica_1.0-3
##  [13] xgboost_1.7.3.1           statmod_1.5.0
##  [15] scran_1.24.1              future_1.31.0
##  [17] miniUI_0.1.1.1            withr_2.5.0
##  [19] spatstat.random_3.1-3     colorspace_2.1-0
##  [21] progressr_0.13.0          highr_0.10
##  [23] knitr_1.42                rstudioapi_0.14
##  [25] ROCR_1.0-11               tensor_1.5
##  [27] listenv_0.9.0             labeling_0.4.2
##  [29] GenomeInfoDbData_1.2.8    polyclip_1.10-4
##  [31] bit64_4.0.5               farver_2.1.1
##  [33] parallelly_1.34.0         vctrs_0.5.2
##  [35] generics_0.1.3            xfun_0.37
##  [37] R6_2.5.1                  ggbeeswarm_0.7.1
##  [39] graphlayouts_0.8.4        rsvd_1.0.5
##  [41] locfit_1.5-9.7            hdf5r_1.3.8
##  [43] bitops_1.0-7              spatstat.utils_3.0-1
##  [45] DelayedArray_0.22.0       promises_1.2.0.1
##  [47] BiocIO_1.6.0              scales_1.2.1
##  [49] beeswarm_0.4.0            gtable_0.3.1
##  [51] beachmat_2.12.0           globals_0.16.2
##  [53] goftest_1.2-3             tidygraph_1.2.3
##  [55] rlang_1.0.6               systemfonts_1.0.4
##  [57] splines_4.2.0             rtracklayer_1.56.1
##  [59] lazyeval_0.2.2            spatstat.geom_3.0-6
##  [61] yaml_2.3.7                abind_1.4-5
##  [63] httpuv_1.6.9              tools_4.2.0
##  [65] ellipsis_0.3.2            RColorBrewer_1.1-3
##  [67] ggridges_0.5.4            Rcpp_1.0.10
##  [69] plyr_1.8.8                sparseMatrixStats_1.8.0
##  [71] zlibbioc_1.42.0           RCurl_1.98-1.10
```

```
##  [73] deldir_1.0-6              pbapply_1.7-0
##  [75] viridis_0.6.2             zoo_1.8-11
##  [77] grr_0.9.5                 ggrepel_0.9.3
##  [79] cluster_2.1.4             scattermore_0.8
##  [81] lmtest_0.9-40             RANN_2.6.1
##  [83] fitdistrplus_1.1-8        patchwork_1.1.2
##  [85] mime_0.12                 evaluate_0.20
##  [87] xtable_1.8-4              XML_3.99-0.13
##  [89] compiler_4.2.0            scater_1.24.0
##  [91] KernSmooth_2.23-20        crayon_1.5.2
##  [93] htmltools_0.5.4           later_1.3.0
##  [95] tidyr_1.3.0               tweenr_2.0.2
##  [97] MASS_7.3-58.2             cli_3.6.0
##  [99] metapod_1.4.0             parallel_4.2.0
## [101] igraph_1.4.1              pkgconfig_2.0.3
## [103] GenomicAlignments_1.32.1  sp_1.6-0
## [105] plotly_4.10.1             scuttle_1.6.3
## [107] spatstat.sparse_3.0-0     xml2_1.3.3
## [109] svglite_2.1.1             vipor_0.4.5
## [111] dqrng_0.3.0               webshot_0.5.4
## [113] XVector_0.36.0            rvest_1.0.3
## [115] stringr_1.5.0             digest_0.6.31
## [117] RcppAnnoy_0.0.20          spatstat.data_3.0-0
## [119] Biostrings_2.64.1         rmarkdown_2.20
## [121] leiden_0.4.3              uwot_0.1.14
## [123] edgeR_3.38.4              DelayedMatrixStats_1.18.2
## [125] restfulr_0.0.15           shiny_1.7.4
## [127] Rsamtools_2.12.0          gtools_3.9.4
## [129] rjson_0.2.21              lifecycle_1.0.3
## [131] nlme_3.1-162              jsonlite_1.8.4
## [133] BiocNeighbors_1.14.0      viridisLite_0.4.1
## [135] limma_3.52.4              fansi_1.0.4
## [137] pillar_1.8.1              lattice_0.20-45
## [139] ggrastr_1.0.1             fastmap_1.1.1
## [141] httr_1.4.5                survival_3.5-3
## [143] glue_1.6.2                png_0.1-8
## [145] bit_4.0.5                 bluster_1.6.0
## [147] ggforce_0.4.1             stringi_1.7.12
## [149] BiocSingular_1.12.0       ape_5.7
## [151] irlba_2.3.5.1             future.apply_1.10.0
```