Angpt1 KO Limbus/Iridocorneal angle combined analysis for Nat Comms Rev 1

Ben Thomson

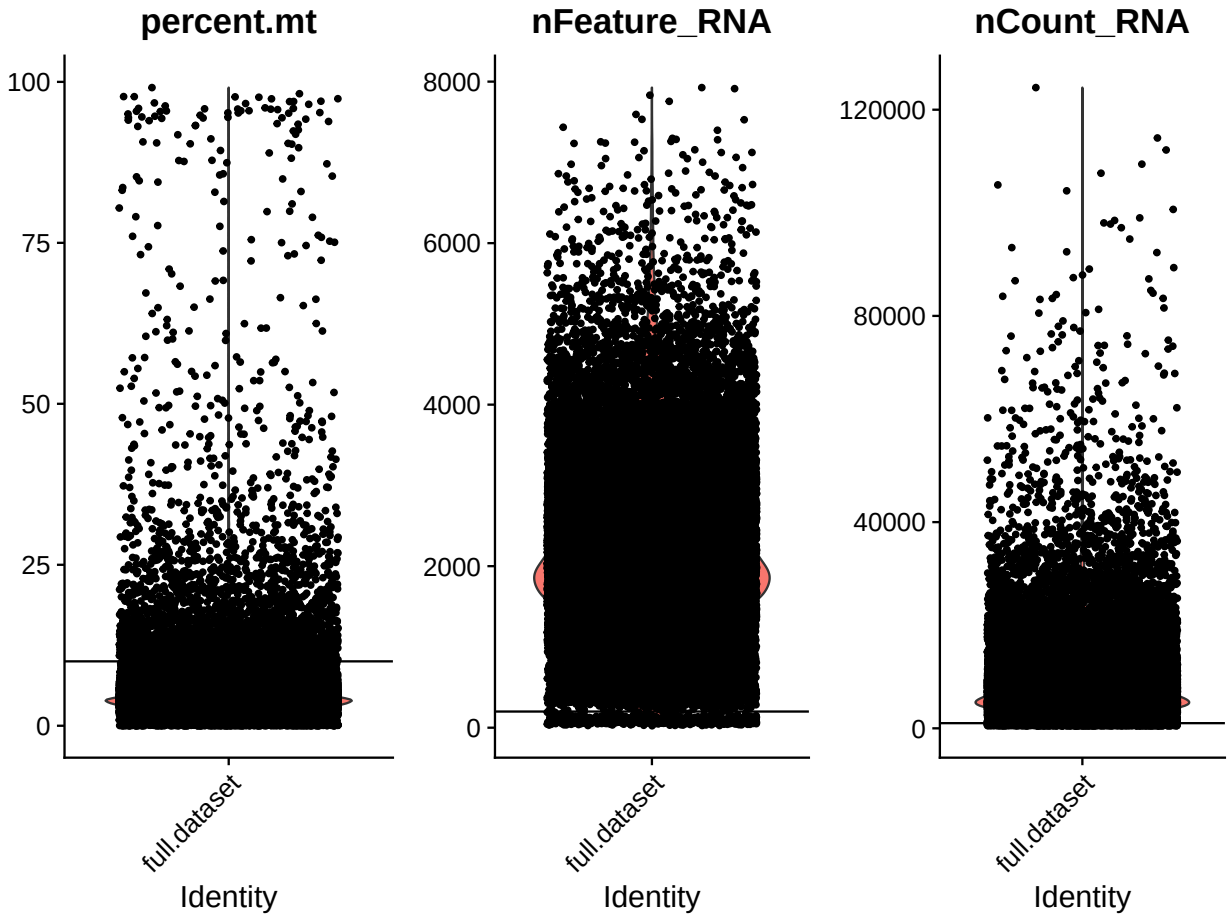
12/2/2020

Before beginning analysis, CellRanger output was imported into Seurat. Features present in at least 2 droplets were included in the dataset at this stage. A metadata column was added to each dataset indicating the percentage of reads in each cell which mapped to mitochondiral RNAs.

```
# load datasets and create seurat objects for analysis


wt185.counts <- Read10X(data.dir = "./raw_data/1_8-5/")
wt185 <- CreateSeuratObject(counts = wt185.counts, project = "wt185",
  min.cells = 2)

wt285.counts <- Read10X(data.dir = "./raw_data/2_8-5/")
wt285 <- CreateSeuratObject(counts = wt285.counts, project = "wt285",
  min.cells = 2)

ko1.counts <- Read10X(data.dir = "./raw_data/New_Sample_1.13.2020/")
ko1 <- CreateSeuratObject(counts = ko1.counts, project = "ko1",
  min.cells = 2)

# clean up...
rm(wt185.counts, wt285.counts, ko1.counts)

wt185 <- RenameCells(wt185, add.cell.id = "wt185")
wt285 <- RenameCells(wt285, add.cell.id = "wt285")
ko1 <- RenameCells(ko1, add.cell.id = "ko1")

# add a col for % of features which are mitochondrial
wt185[["percent.mt"]] <- PercentageFeatureSet(wt185, pattern = "^mt-",
  assay = "RNA")
wt285[["percent.mt"]] <- PercentageFeatureSet(wt285, pattern = "^mt-",
  assay = "RNA")
ko1[["percent.mt"]] <- PercentageFeatureSet(ko1, pattern = "^mt-",
  assay = "RNA")
```

After importation, datasets were screened to remove low-quality cells from the analysis. We removed all droplets with <200 features, <1000 unique molecular identifiers, or >10% of total reads mapping to mitochondrial RNAs.

```r
# set number of features, UMI and percentMT
min.features <- 200
min.UMI <- 1000
max.MT <- 10

# generate plots showing cell distribution with QC cutoffs
full.dataset <- merge(x = wt185, y = c(wt285, ko1))

Idents(full.dataset) <- "full.dataset"

p1 <- VlnPlot(full.dataset, features = "percent.mt", pt.size = 1) +
  NoLegend() + geom_hline(yintercept = max.MT)
p2 <- VlnPlot(full.dataset, features = "nFeature_RNA", pt.size = 1) +
  NoLegend() + geom_hline(yintercept = min.features)
p3 <- VlnPlot(full.dataset, features = "nCount_RNA", pt.size = 1) +
  NoLegend() + geom_hline(yintercept = min.UMI)


rm(full.dataset)
plot_grid(p1, p2, p3, ncol = 3)
```



Droplets falling outside of the QC thresholds were removed from analysis.

```
# store sample names and total number of cells prior to
# filtering
sample.names <- c("wt185", "wt285", "ko1")
cell.number.before <- c(length(wt185$orig.ident), length(wt285$orig.ident),
  length(ko1$orig.ident))

# remove cells which don't make the QC thresholds
wt185 <- subset(wt185, subset = nFeature_RNA > min.features &
  nCount_RNA > min.UMI & percent.mt < max.MT)
wt285 <- subset(wt285, subset = nFeature_RNA > min.features &
  nCount_RNA > min.UMI & percent.mt < max.MT)
ko1 <- subset(ko1, subset = nFeature_RNA > min.features & nCount_RNA >
  min.UMI & percent.mt < max.MT)

# cell number remaining after QC filtering
cell.number.after <- c(length(wt185$orig.ident), length(wt285$orig.ident),
  length(ko1$orig.ident))


# generate a data frame containing the results of QC
qc.frame <- data.frame(sample.names, cell.number.before, cell.number.after)
qc.frame$cells.filtered <- (qc.frame$cell.number.before - qc.frame$cell.number.after)

qc.frame %>% kbl(caption = "Results of initial QC filtering") %>%
  kable_minimal(full_width = FALSE, position = "left", latex_options = "hold_position")
```

Table 1: Results of initial QC filtering

sample.names	cell.number.before	cell.number.after	cells.filtered
wt185	11255	10008	1247
wt285	12625	11348	1277
ko1	8614	7254	1360

```
rm(min.features, min.UMI, max.MT, cell.number.after, cell.number.before,
  sample.names, qc.frame)
```

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.

First, to estimate the proportion of doublets in our dataset, we used sample ko1 (which contains equal numbers of male and female cells) to estimate doublet frequency by looking for the rate of droplets containing both uniquely male and female genes.

this calculation is performed as described in Bloom JD. Estimating the frequency of multiplets in single-cell RNA sequencing from cell-mixing experiments. PeerJ. 2018 Sep 3;6:e5578. doi: 10.7717/peerj.5578. PMID: 30202659; PMCID: PMC6126471.

```
# first we create a subset of ko1 which includes only cells
# where Xist or at least 1 of the Y chromosome genes are
# detected
```

```
doublet.subset <- subset(ko1, subset = Xist > 0 | Ddx3y > 0 |
  Eif2s3y > 0 | Gm29650 > 0 | Kdm5d > 0 | Uty > 0)

# find total number of droplets with male or female genes
# detected
total.cells <- length(doublet.subset$orig.ident)

# total number of Y chromosome-expressing droplets
male <- length(subset(doublet.subset, subset = Ddx3y > 0 | Eif2s3y >
  0 | Gm29650 > 0 | Kdm5d > 0 | Uty > 0)$orig.ident)

# total number of Xist-expressing droplets
female <- length(subset(doublet.subset, subset = Xist > 0)$orig.ident)

# total number of droplets containing both male and female
# genes
male.female <- length(subset(doublet.subset, subset = Xist >
  0 & (Ddx3y > 0 | Eif2s3y > 0 | Gm29650 > 0 | Kdm5d > 0 |
  Uty > 0))$orig.ident)


#' Multiplet frequency from cell-type mixing experiment.
#' function from Bloom, JD. 2018
#' @param n1 Number of droplets with at least one cell of type 1
#' @param n2 Number of droplets with at least one cell of type 2
#' @param n12 Number of droplets with cells of both types
#' @return The estimated multiplet frequency
multiplet_freq <- function(n1, n2, n12) {
  n <- n1 * n2/n12
  mu1 <- -log((n - n1)/n)
  mu2 <- -log((n - n2)/n)
  mu <- mu1 + mu2
  return(1 - mu * exp(-mu)/(1 - exp(-mu)))
}

ko1.multiplet_freq <- multiplet_freq(female, male, male.female)
multiplet.freq.per1k <- ko1.multiplet_freq/8.641

# clean up variables
rm(doublet.subset, total.cells, male, female, male.female)
```

Estimated multiplet frequency: ‘r ko1.multiplet_freq’ in a sample originally consisting of of 8614 sequenced droplets This frequency will be used as the baseline for doublet detection with scDblFinder. Consistant with our expectations, this value was very close to the 10x guideline of 1% per 1000 droplets sequenced, so we will allow scDblFinder to automatically determine doublet number based on 10x defaults.

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


# for each dataset, convert to SingleCellExperiment, run
# scDblFinder and return to Seurat object. To avoid
# introducing formatting issues to the Seurat object used for
```

```
# subsequent analysis, scDblFinder metadata are copied from
# this new object into the original dataset and the new
# object is then deleted.

sce <- as.SingleCellExperiment(wt185)
sce <- scDblFinder(sce, verbose = F, nfeatures = scDbl.features,
  dims = scDbl.dims)
sce <- as.Seurat(sce)


wt185 <- AddMetaData(wt185, col.name = "doubletScore", metadata = sce$scDblFinder.score)
wt185 <- AddMetaData(wt185, col.name = "doubletClass", metadata = sce$scDblFinder.class)

sce <- as.SingleCellExperiment(wt285)
sce <- scDblFinder(sce, verbose = F, nfeatures = scDbl.features,
  dims = scDbl.dims)
sce <- as.Seurat(sce)

wt285 <- AddMetaData(wt285, col.name = "doubletScore", metadata = sce$scDblFinder.score)
wt285 <- AddMetaData(wt285, col.name = "doubletClass", metadata = sce$scDblFinder.class)


sce <- as.SingleCellExperiment(ko1)
sce <- scDblFinder(sce, verbose = F, nfeatures = scDbl.features,
  dims = scDbl.dims)
sce <- as.Seurat(sce)

ko1 <- AddMetaData(ko1, col.name = "doubletScore", metadata = sce$scDblFinder.score)
ko1 <- AddMetaData(ko1, col.name = "doubletClass", metadata = sce$scDblFinder.class)


wt185.calls <- wt185@meta.data %>% as.data.table
wt285.calls <- wt285@meta.data %>% as.data.table
ko1.calls <- ko1@meta.data %>% as.data.table

scDblFinder.calls <- rbind(wt185.calls, wt285.calls, ko1.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 2: scDblFinder predictions

orig.ident	doubletClass	N
wt185	doublet	932
wt185	singlet	9076
wt285	doublet	1188
wt285	singlet	10160
ko1	doublet	482
ko1	singlet	6772

```
# cleanup
rm(sce, wt185.calls, wt285.calls, ko1.calls)
```

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

```
wt185 <- subset(wt185, subset = doubletClass == "singlet")
wt285 <- subset(wt285, subset = doubletClass == "singlet")
ko1 <- subset(ko1, subset = doubletClass == "singlet")
```

In preparation for dataset integration, each sample dataset was then normalized and a set of shared variable features was identified.

```
wt185 <- NormalizeData(object = wt185, verbose = FALSE)
wt185 <- FindVariableFeatures(object = wt185, selection.method = "vst",
  nfeatures = 5000, verbose = FALSE)

wt285 <- NormalizeData(object = wt285, verbose = FALSE)
wt285 <- FindVariableFeatures(object = wt285, selection.method = "vst",
  nfeatures = 5000, verbose = FALSE)

ko1 <- NormalizeData(object = ko1, verbose = FALSE)
ko1 <- FindVariableFeatures(object = ko1, selection.method = "vst",
  nfeatures = 5000, verbose = FALSE)

# shared features
shared.var.features <- Reduce(intersect, list(VariableFeatures(wt185),
  VariableFeatures(wt285), VariableFeatures(ko1)))

# remove the list of excluded features
integration.features <- setdiff(shared.var.features, bad.features)
```

The samples were then merged using the IntegrateData() pipeline in seurat to perform CCA using 35 dimensions.

```
# find anchors for integration
anchors <- FindIntegrationAnchors(object.list = c(wt185, wt285,
  ko1), dims = 1:35, anchor.features = integration.features)

# intrgrate datasets
integrated <- IntegrateData(anchorset = anchors, dims = 1:35,
  verbose = F)



# add genotype information to metadata in the 'genotype'
# column
integrated$genotype <- plyr::mapvalues(x = integrated$orig.ident,
  from = c("wt185", "wt285", "ko1"), to = c("WT", "WT", "KO"))

# cleanup
rm(anchors, wt185, wt285, ko1)
gc(verbose = F)
```

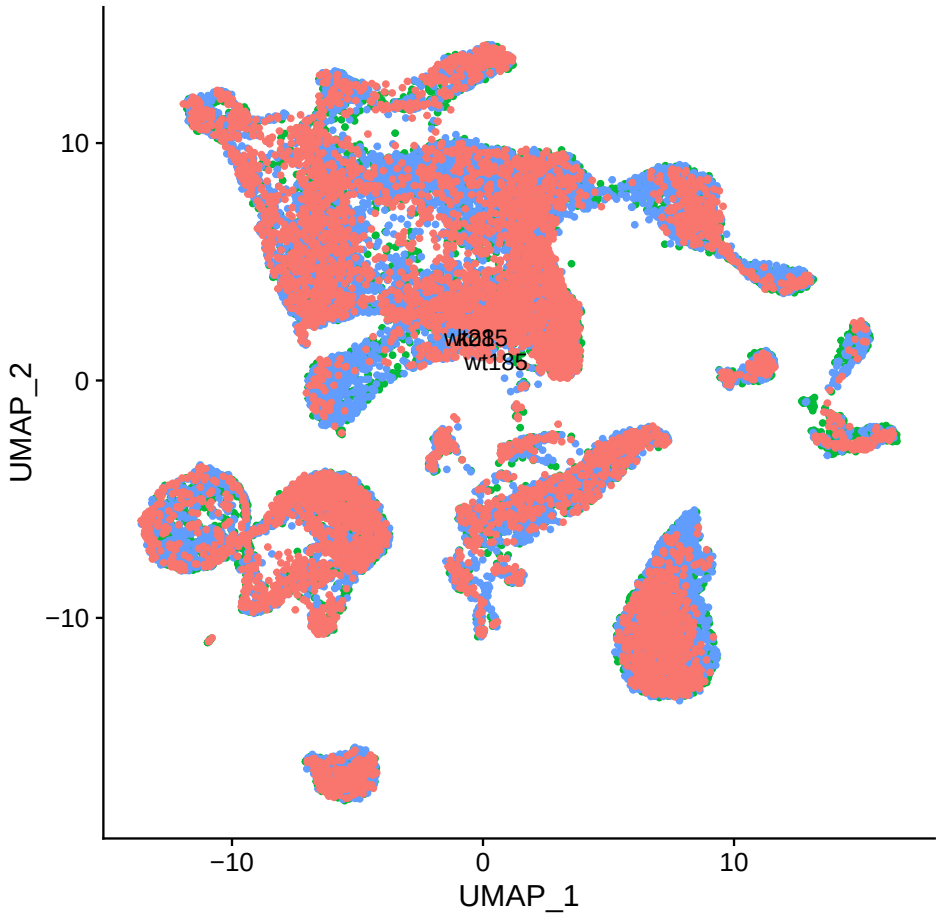
```
##             used   (Mb) gc trigger    (Mb)   max used    (Mb)
## Ncells   6552934  350.0   12183244   650.7   12183244   650.7
## Vcells 252686159 1927.9 1362720701 10396.8 1700868951 12976.7
```

Following CCA integration, the dataset was scaled and 20 pricipal components were calculated for plotting and clustering of the integrated dataset.

```
full.dataset.dims <- 20

DefaultAssay(object = integrated) <- "integrated"
integrated <- ScaleData(object = integrated, verbose = FALSE,
  assay = "integrated")
integrated <- ScaleData(object = integrated, verbose = FALSE,
  assay = "RNA")

# run the PCA using the same features used for integration
integrated <- RunPCA(object = integrated, npcs = full.dataset.dims,
  verbose = FALSE, features = integration.features, assay = "integrated")
```

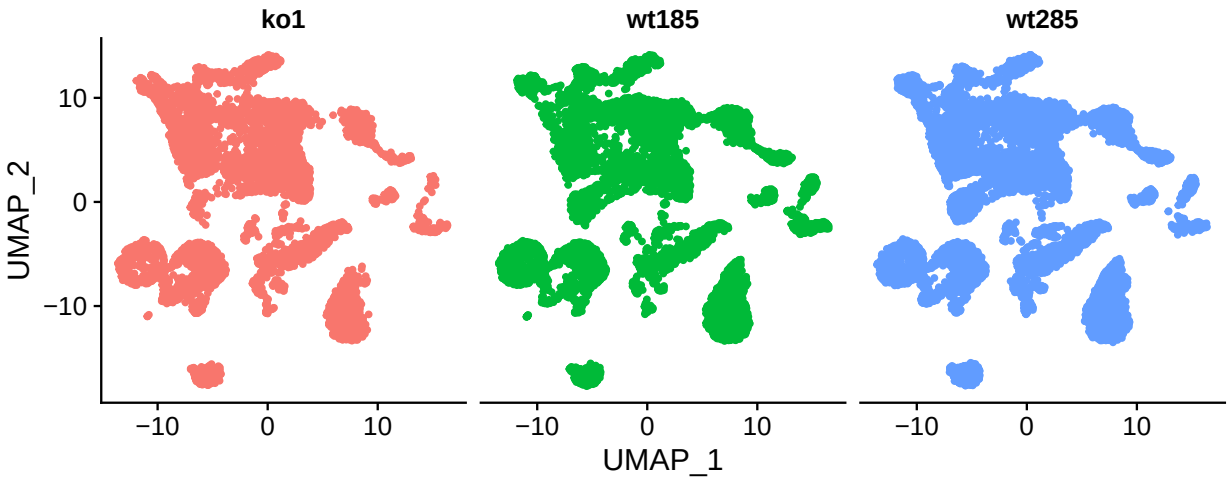
Following dimensionality esitmation, 20 dimensions will be used for UMAP projection and cluster identification of the integrated dataset.

```
integrated <- RunUMAP(object = integrated, dims = 1:full.dataset.dims,
  min.dist = 0.5, n.neighbors = 20, verbose = FALSE)


DimPlot(object = integrated, reduction = "umap", label = TRUE,
  pt.size = 1, label.size = 4) + NoLegend() + coord_fixed() +
  theme(aspect.ratio = 1)
```



Distribution of cells was similar in all samples.

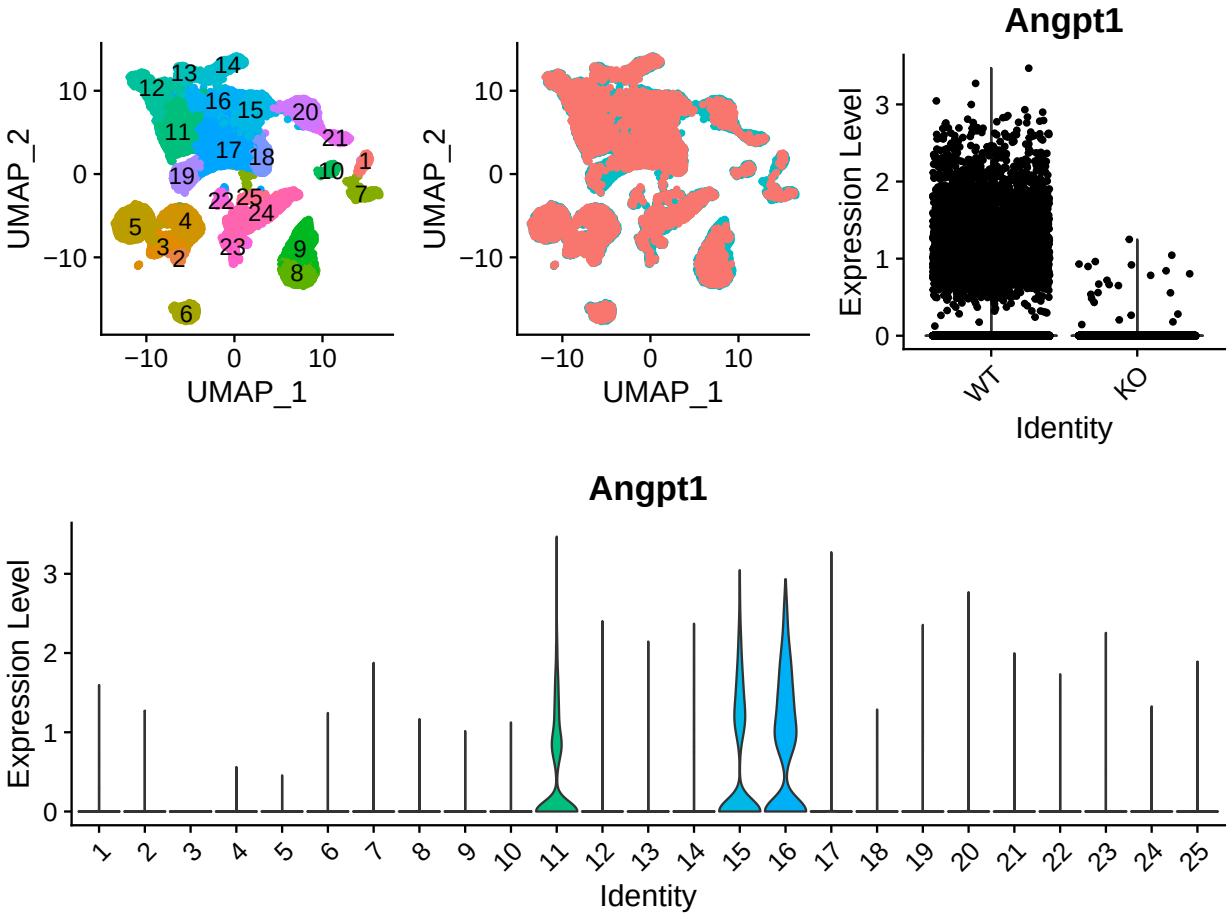
```
DimPlot(integrated, reduction = "umap", label = FALSE, pt.size = 1,
  split.by = "orig.ident") + NoLegend() + theme(aspect.ratio = 1)
```



CCA-aligned data was then clustered at low resolution using a Louvain algorithm implemented in Seurat at a resolution of 0.5 and a resolution of 20. This same data was used to generate figure 5.

```
integrated <- FindNeighbors(object = integrated, dims = 1:full.dataset.dims,
  verbose = FALSE, features = integration.features)
integrated <- FindClusters(object = integrated, resolution = 0.5,
  verbose = FALSE)

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


# Plot of cluster identities mapped on full dataset
p1 <- DimPlot(object = integrated, reduction = "umap", label = TRUE,
  pt.size = 1, label.size = 4, repel = FALSE) + NoLegend() +
  coord_fixed() + theme(aspect.ratio = 1)

# integrated$genotype <- factor(x = integrated$genotype,
# levels = c('WT', 'KO'))

# compare distribution of WT and Angpt1dNC cells
p2 <- DimPlot(object = integrated, reduction = "umap", label = FALSE,
  group.by = "genotype", pt.size = 1, label.size = 6) + coord_fixed() +
```

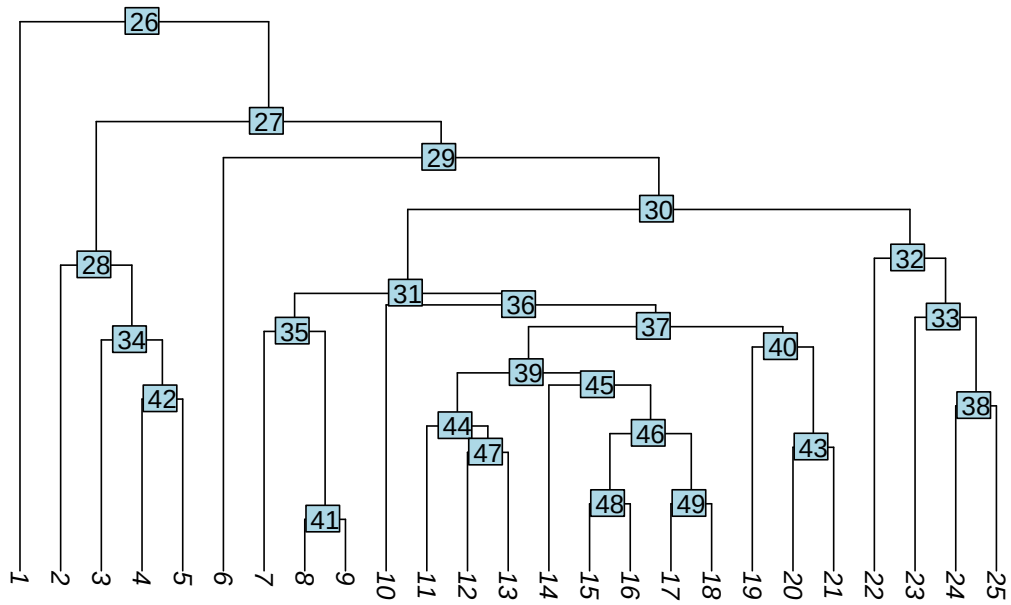
```
  NoLegend() + theme(aspect.ratio = 1)

integrated$genotype <- factor(x = integrated$genotype, levels = c("WT",
  "KO"))

# Show number of Angpt1 expressing cells in WT and Angpt1dNC
# samples.
p3 <- VlnPlot(integrated, group.by = "genotype", features = "Angpt1",
  assay = "RNA") + NoLegend()

# plot the figure, including a vln plot showing Angpt1
# expression by cluster in the WT dataset.
plot_grid(plot_grid(p1, p2, p3, nrow = 1), VlnPlot(subset(integrated,
  subset = genotype == "WT"), features = "Angpt1", pt.size = 0,
  assay = "RNA") + NoLegend(), nrow = 2)
```




```
# export dataset for figure generation
saveRDS(integrated, file = "./data/full_dataset_10-2-20.rds.gz",
  compress = T)
```

A dendrogram was then generated. A similar dendrogram was used in figure 5.

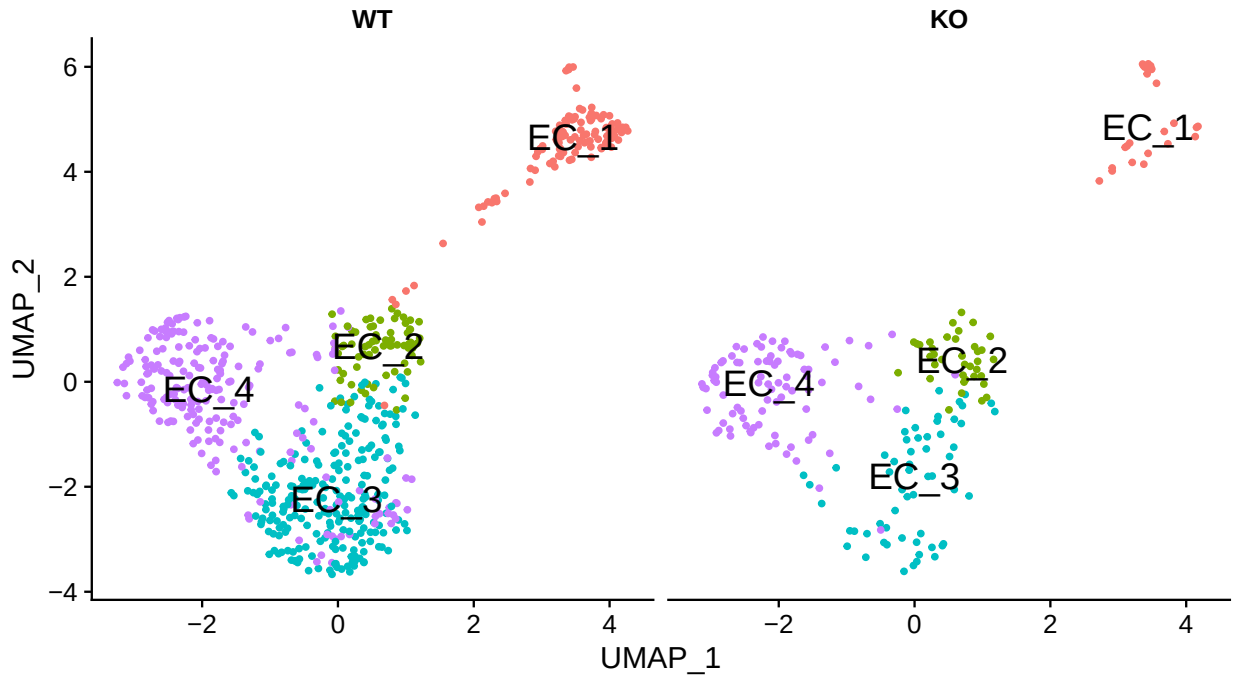
```
PlotClusterTree(object = integrated)
```



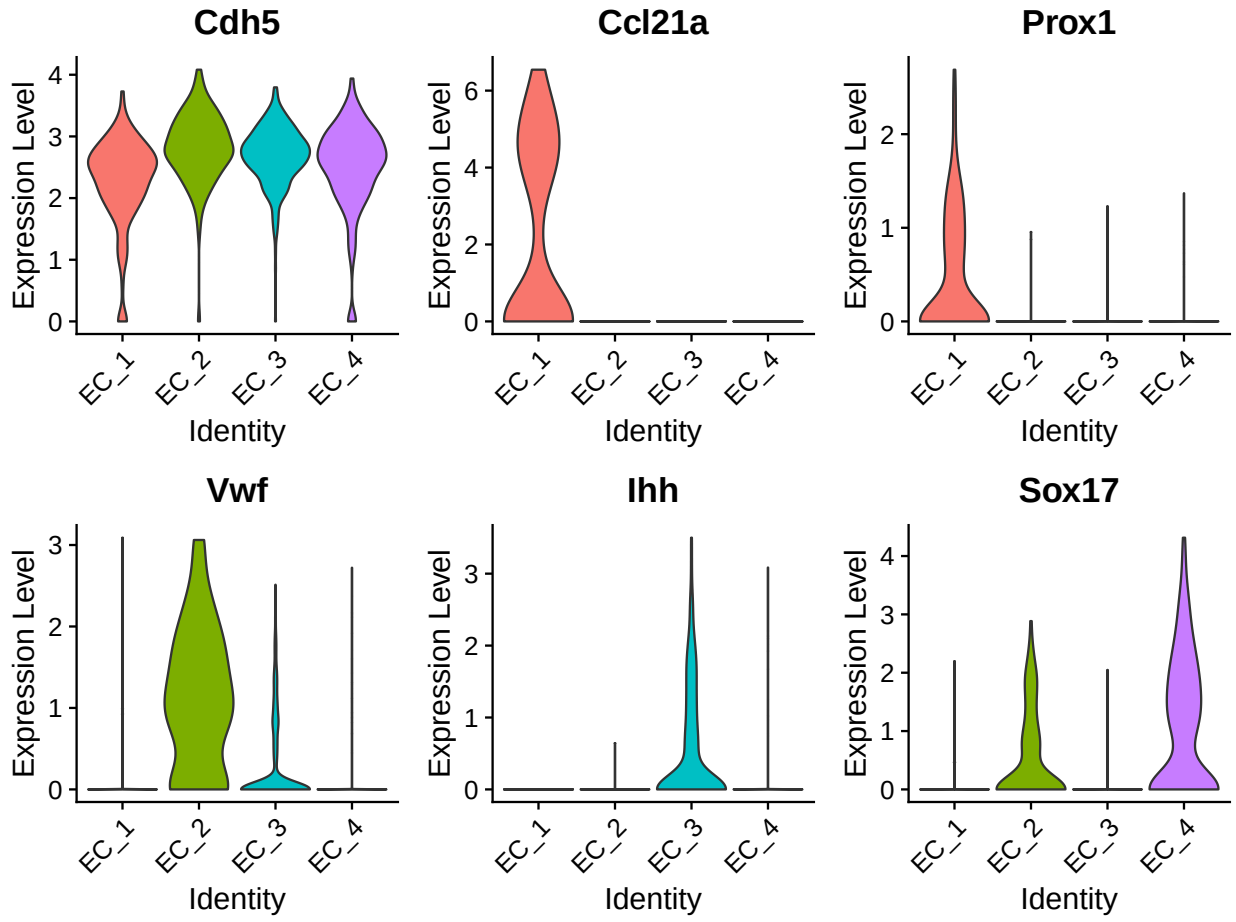
To identify SC endothelial cells, cluster 11 (representing Cdh5+ putative endothelial cells) was isolated and 100 new PCs were calculated. intrinsicDimension::maxLikGlobalDimEst() was then used to estimate the dimensionality of the isolated endothelial cell data. It was apparent that cluster EC_1, which contained cells expressing lymphatic markers Prox1 and Ccl21a, separated into two distinct clusters on UMAP. We therefore hypothesized that cluster EC_1 may have contained both Schlemm's canal and lymphatic endothelial cells.

```
# create the subsetted dataset and switch the default assay
# to Integrated
subset.endos <- subset(integrated, idents = c("6"))
DefaultAssay(object = subset.endos) <- "integrated"


endos.reclustered <- RunPCA(object = subset.endos, npcs = 100,
  features = integration.features)

# determine dimensionality of the EC dataset
ec.dataset.dims <- maxLikGlobalDimEst(Embeddings(endos.reclustered[["pca"]]),
  k = 10, unbiased = TRUE)
ec.dataset.dims <- ceiling(ec.dataset.dims[[1]])
```

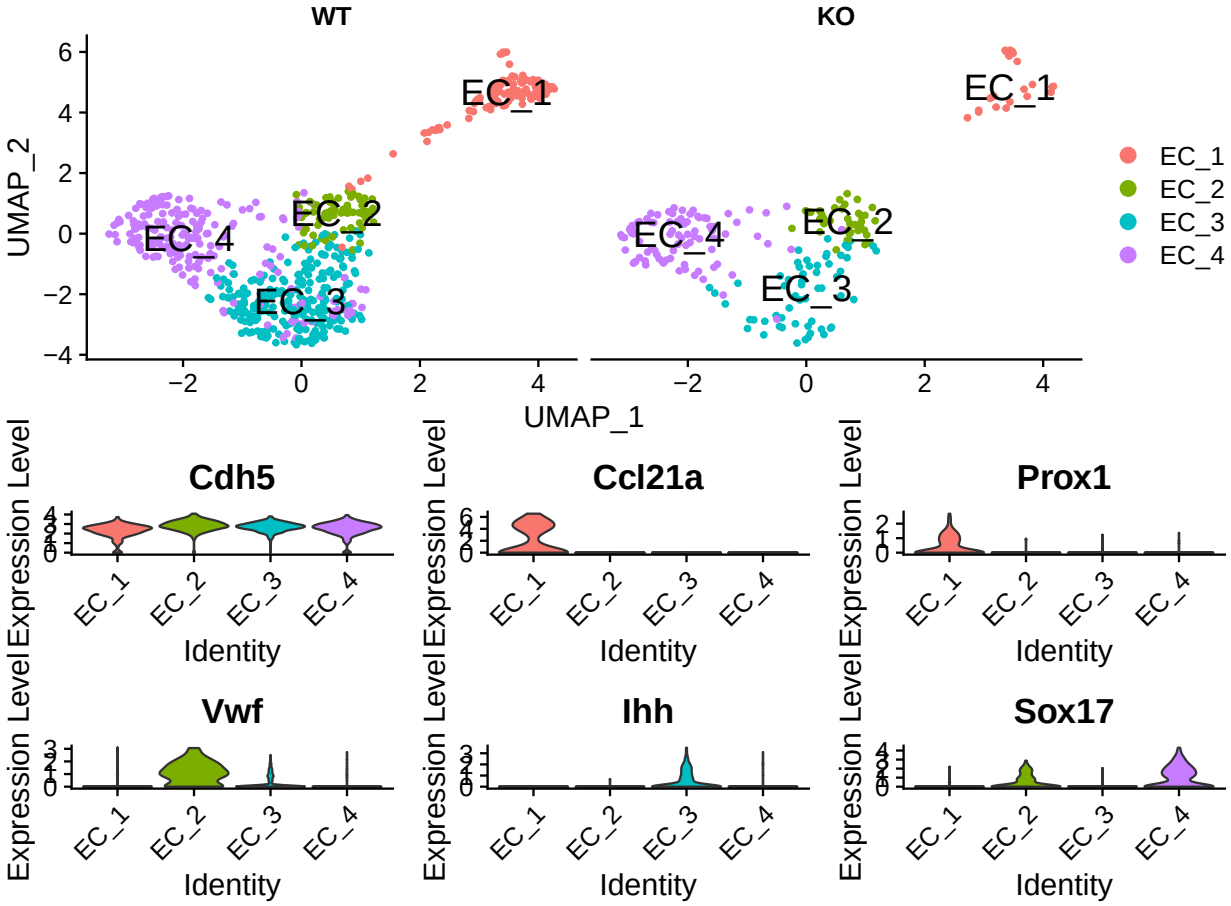
```r
# identify EC clusters
endos.reclustered <- FindNeighbors(object = endos.reclustered,
  dims = 1:ec.dataset.dims, verbose = FALSE, features = integration.features)
endos.reclustered <- FindClusters(object = endos.reclustered,
  resolution = 0.5, verbose = FALSE)

# build a cluster tree to reorder clusters
endos.reclustered <- BuildClusterTree(endos.reclustered, reorder.numeric = TRUE,
  reorder = TRUE, verbose = T, assay = "RNA", dims = 1:ec.dataset.dims)

# add a prefix to the cluster names to indicate their EC
# origin
endos.reclustered <- SetIdent(endos.reclustered, cells = Cells(endos.reclustered),
  value = as.factor(paste("EC_", endos.reclustered@active.ident,
    sep = "")))


endos.reclustered <- RunUMAP(object = endos.reclustered, dims = 1:ec.dataset.dims,
  min.dist = 0.1, n.neighbors = 100, verbose = FALSE)

ECplot1 <- DimPlot(object = endos.reclustered, reduction = "umap",
  label = TRUE, pt.size = 1, label.size = 6, split.by = "genotype")
ECplot1 + NoLegend() + coord_fixed() + theme(aspect.ratio = 1)
```



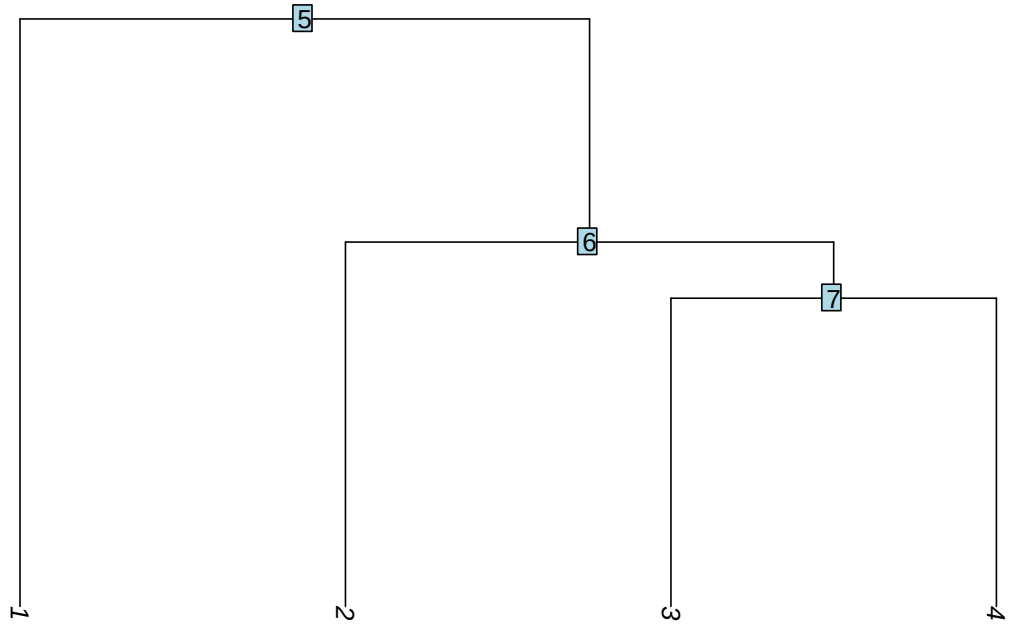
```
ECVln1 <- VlnPlot(endos.reclustered, pt.size = 0, features = c("Cdh5",
  "Ccl21a", "Prox1", "Vwf", "Ihh", "Sox17"), assay = "RNA")
ECVln1 + NoLegend()
```




```
plot_grid(ECplot1, ECVln1, nrow = 2, align = "hv")
```



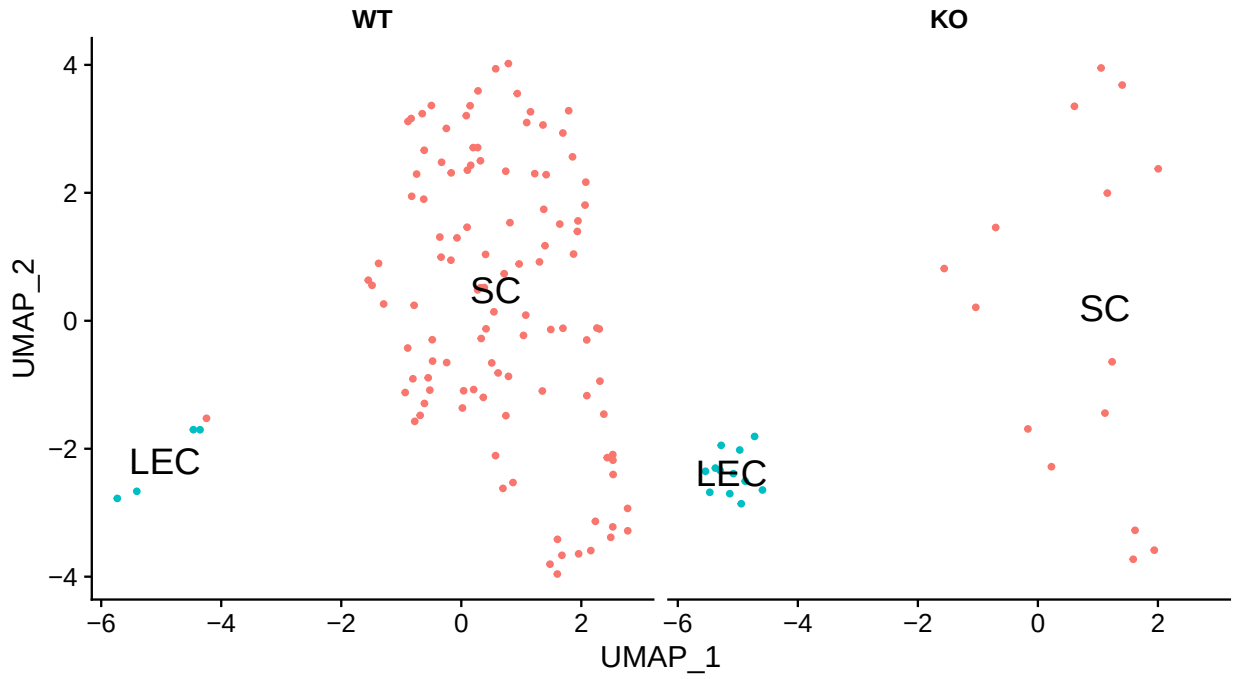

```
PlotClusterTree(endos.reclustered)
```



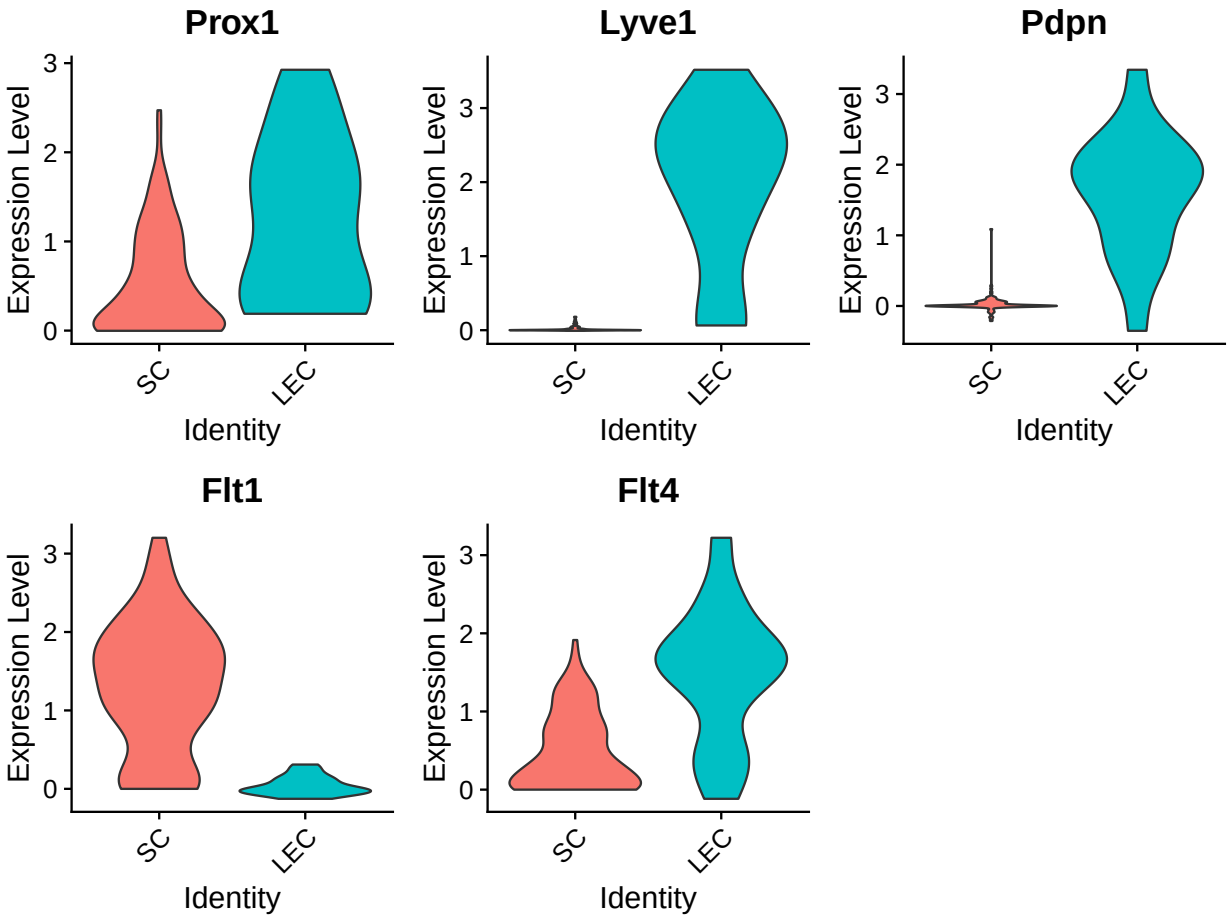
To investigate these two clusters in more detail, cluster ec_2 was isolated and re-clustered in a 3rd set of 15 PCs.

```
lec.reclustered <- subset(endos.reclustered, idents = "EC_1")

lec.reclustered <- RunPCA(lec.reclustered, npcs = 100)

lec.dataset.dims <- maxLikGlobalDimEst(Embeddings(lec.reclustered[["pca"]]),
  k = 10, unbiased = TRUE)
lec.dataset.dims <- ceiling(lec.dataset.dims[[1]])


lec.reclustered <- FindNeighbors(object = lec.reclustered, dims = 1:lec.dataset.dims,
  verbose = FALSE)

lec.reclustered <- FindClusters(object = lec.reclustered, resolution = 0.5,
  verbose = FALSE)

# rename clusters
lec.reclustered <- RenameIdents(lec.reclustered, `0` = "SC",
  `1` = "LEC")


lec.reclustered <- RunUMAP(object = lec.reclustered, dims = 1:lec.dataset.dims,
```

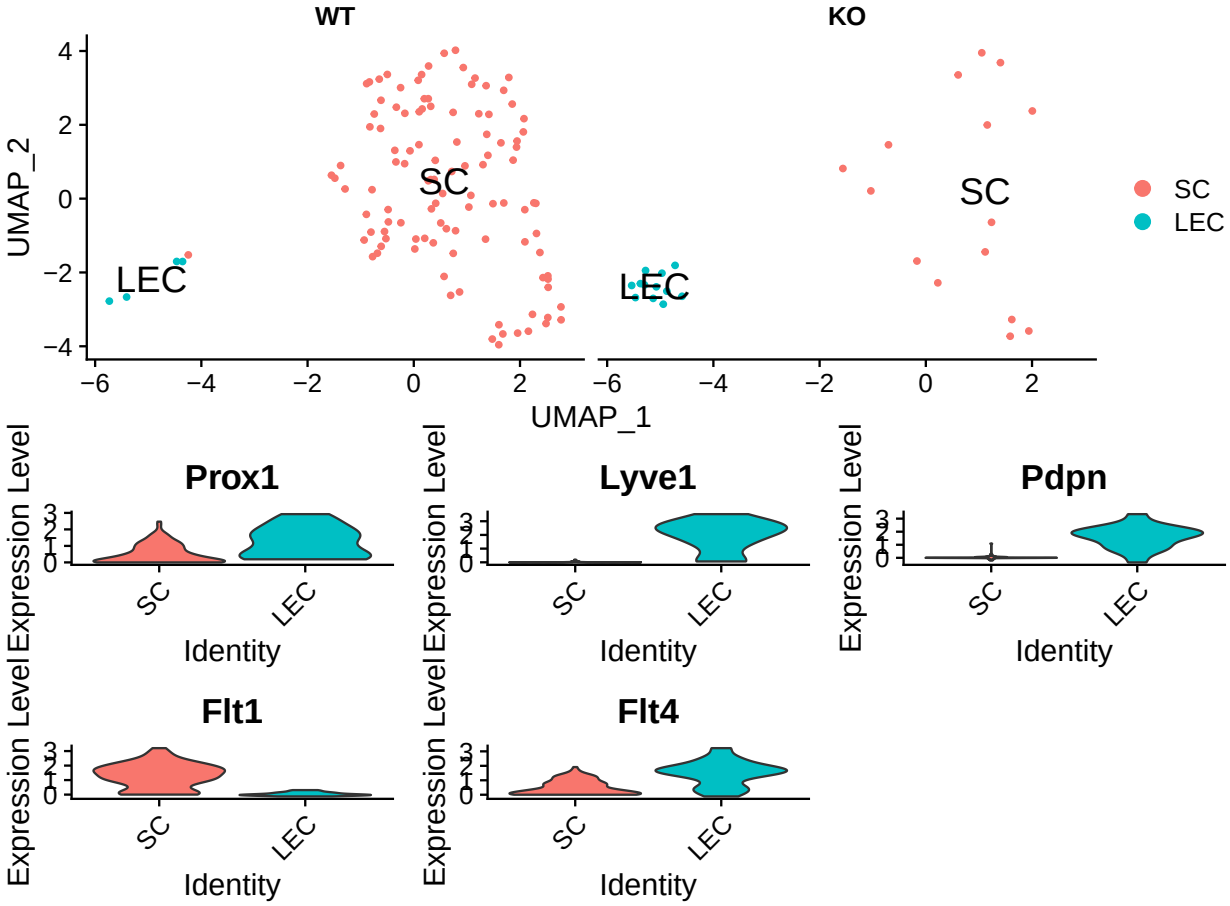
```
  min.dist = 0.2, n.neighbors = 5, verbose = FALSE)

LECplot1 <- DimPlot(object = lec.reclustered, reduction = "umap",
  label = TRUE, pt.size = 1, label.size = 6, split.by = "genotype")
LECplot1 + coord_fixed() + NoLegend() + theme(aspect.ratio = 1)
```




```
LECVln1 <- VlnPlot(lec.reclustered, features = c("Prox1", "Lyve1",
  "Pdpn", "Flt1", "Flt4"), pt.size = 0)
LECVln1 + NoLegend()
```

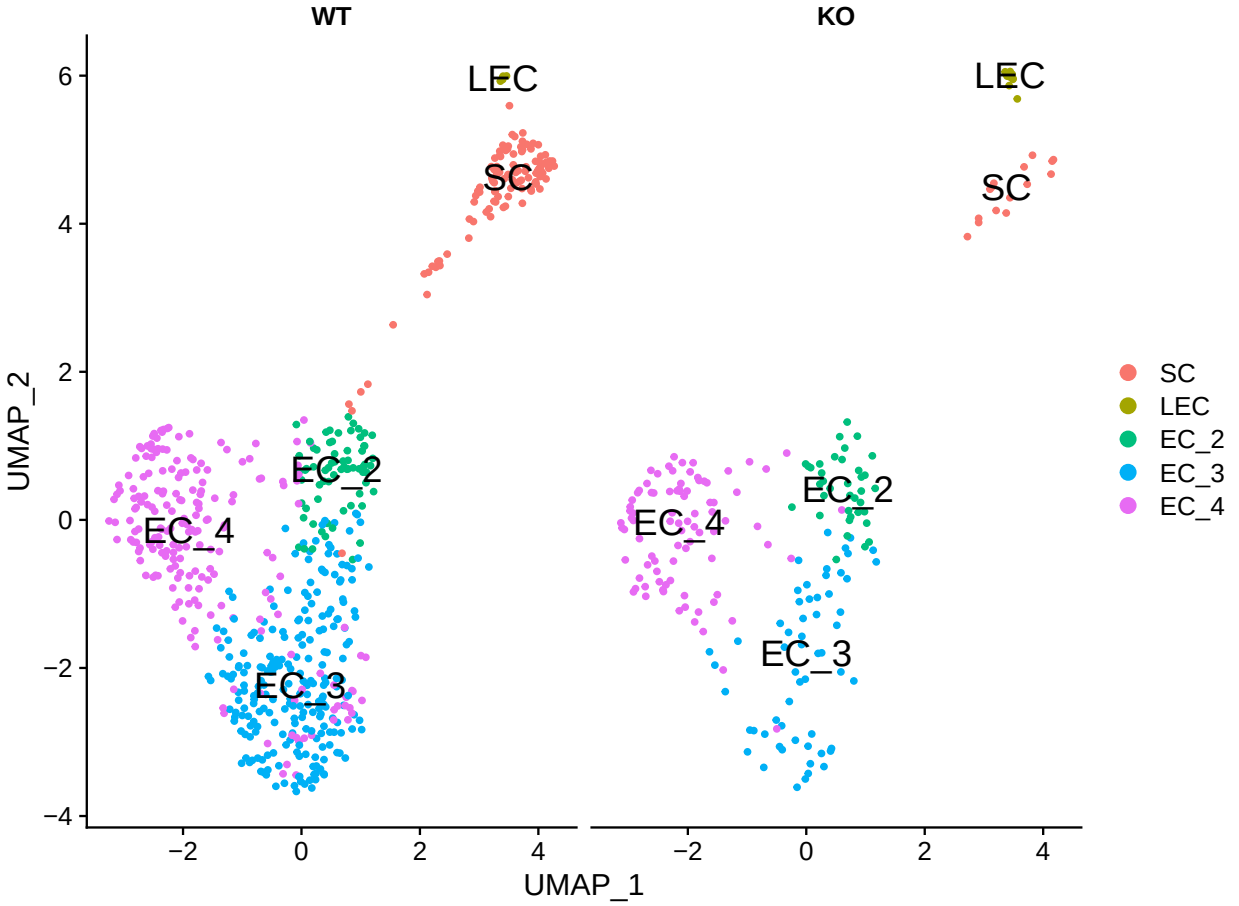



```
plot_grid(LECplot1, LECVln1, nrow = 2)
```



Group labels from this subset analysis were then updated into the full EC dataset for expression analysis and figure generation in a seperate notebook.

```
# copy cluster labels from the LEC subset analysis
endos.reclustered <- SetIdent(endos.reclustered, cells = Cells(lec.reclustered),
  value = lec.reclustered@active.ident)

# plot the updated endothelial dataset
DimPlot(object = endos.reclustered, reduction = "umap", label = TRUE,
  pt.size = 1, label.size = 6, split.by = "genotype")
```

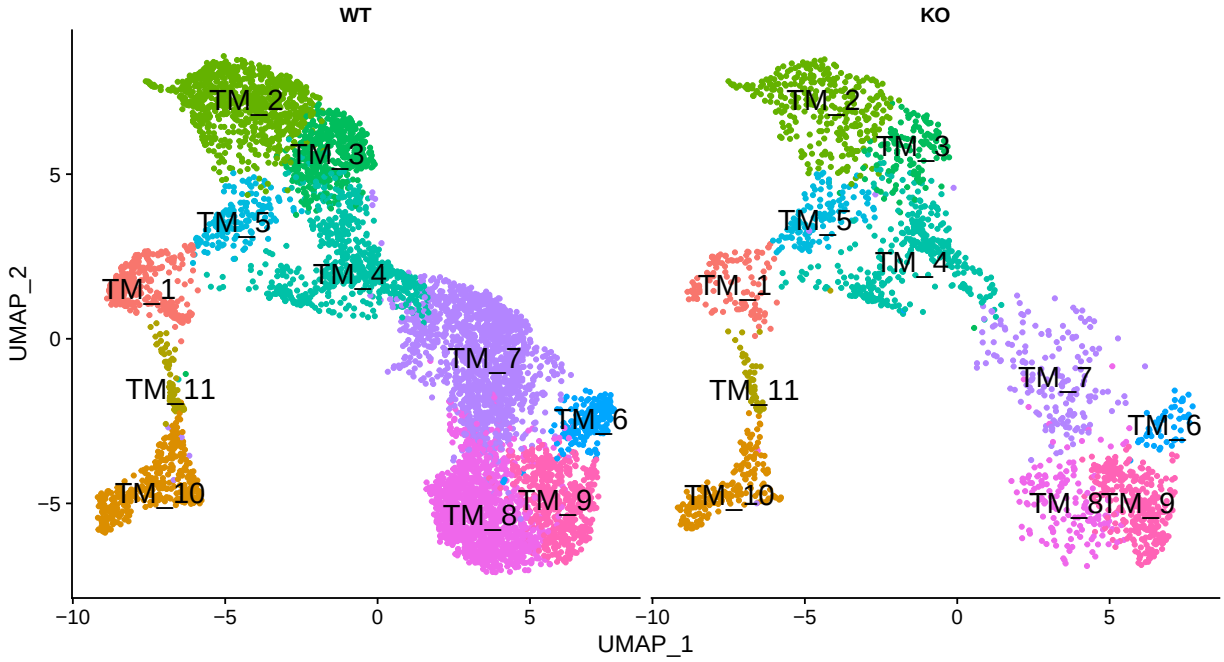



```
# export endothelial cell data for figure and dataset
# generation
saveRDS(endos.reclustered, file = "./data/6wk_endos_10-1-20.rds.gz",
  compress = T)
saveRDS(lec.reclustered, file = "./data/6wk_LECs_10-1-20.rds.gz",
  compress = T)
```

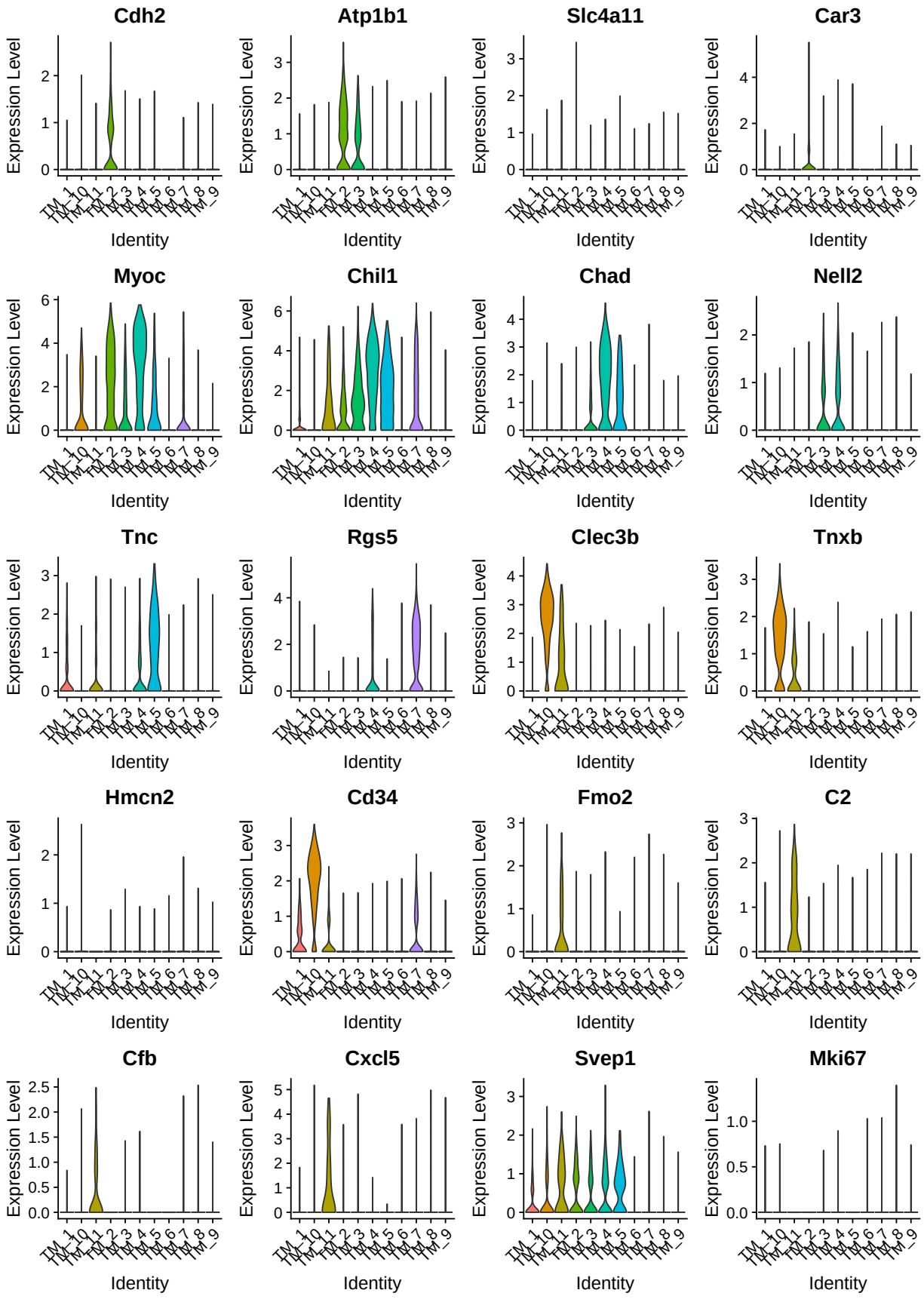
To identify TM cells, clusters 12,13,14 and 15 (containing cells which expressed TM markers Myoc and Chil1) were isolated and 100 new PCs were calculated. intrinsicDimension::maxLikGlobalDimEst() was then used to estimate the dimensionality of the subsetted data.

```
# create the subsetted dataset and switch the default assay
# to Integrated
subset.TM <- subset(integrated, idents = c(11, 12, 14, 15, 16))
DefaultAssay(object = subset.TM) <- "integrated"


TM.reclustered <- RunPCA(object = subset.TM, npcs = 100, features = integration.features)

# determine dimensionality of the TM dataset
TM.dataset.dims <- maxLikGlobalDimEst(Embeddings(TM.reclustered[["pca"]]),
  k = 10, unbiased = TRUE)
TM.dataset.dims <- ceiling(TM.dataset.dims[[1]])
```

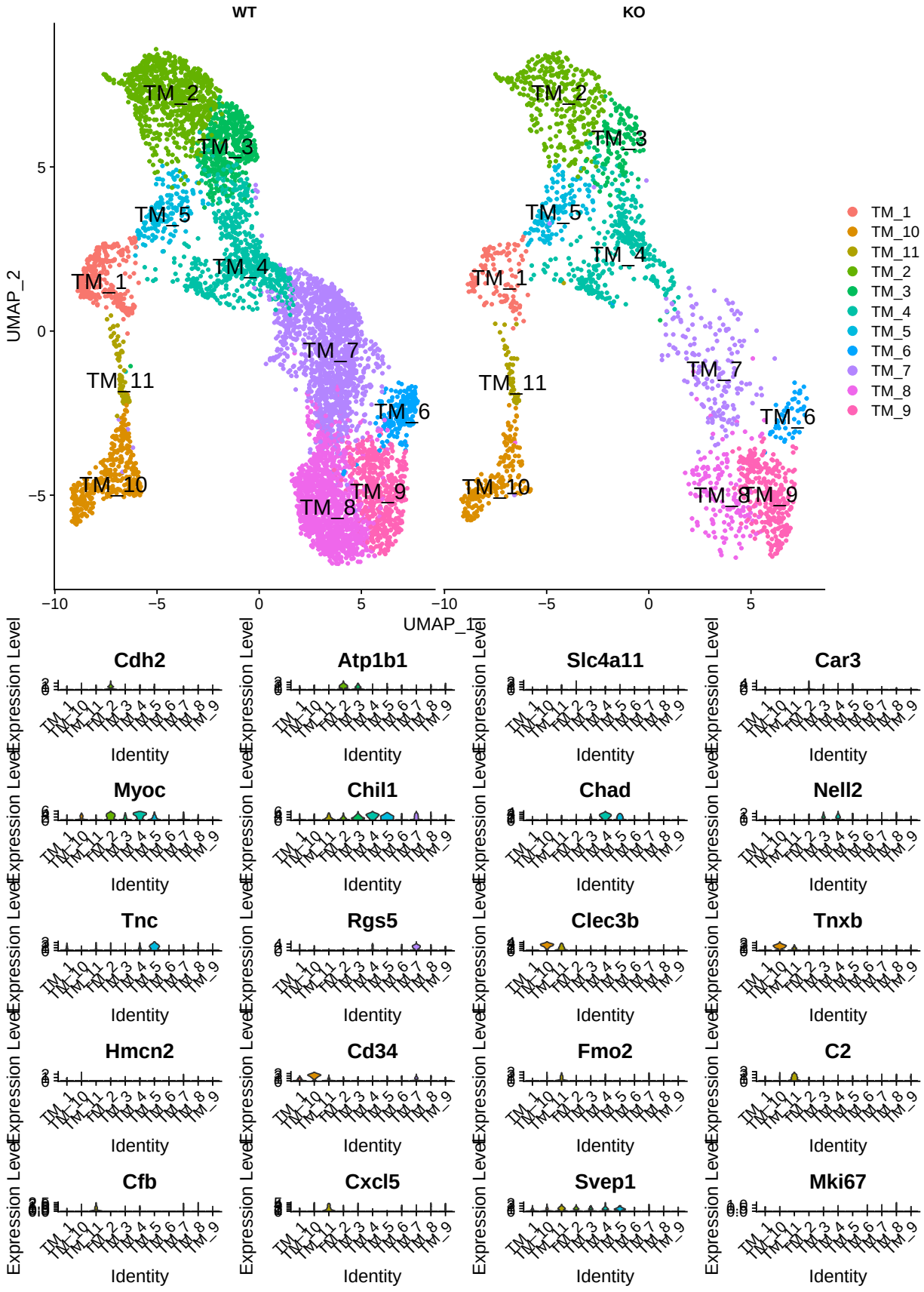
```r
# identify TM clusters
TM.reclustered <- FindNeighbors(object = TM.reclustered, dims = 1:TM.dataset.dims,
  verbose = FALSE, features = integration.features)
TM.reclustered <- FindClusters(object = TM.reclustered, resolution = 0.5,
  verbose = FALSE)

# build a cluster tree to reorder clusters
TM.reclustered <- BuildClusterTree(TM.reclustered, reorder.numeric = TRUE,
  reorder = TRUE, verbose = T, assay = "RNA", dims = 1:TM.dataset.dims)

# add a prefix to the cluster names to indicate their EC
# origin
TM.reclustered <- SetIdent(TM.reclustered, cells = Cells(TM.reclustered),
  value = as.factor(paste("TM_", TM.reclustered@active.ident,
    sep = "")))


TM.reclustered <- RunUMAP(object = TM.reclustered, dims = 1:TM.dataset.dims,
  min.dist = 0.4, n.neighbors = 80, verbose = FALSE)

TMplot1 <- DimPlot(object = TM.reclustered, reduction = "umap",
  label = TRUE, pt.size = 1, label.size = 6, split.by = "genotype")
TMplot1 + NoLegend() + coord_fixed() + theme(aspect.ratio = 1)
```


WT
KO
TM_2
TM_3
TM_5
TM_4
TM_1
TM_7
TM_11
TM_6
TM_10
TM_8
TM_9
UMAP_2
UMAP_1
5
0
−5
−10
−5
0
5

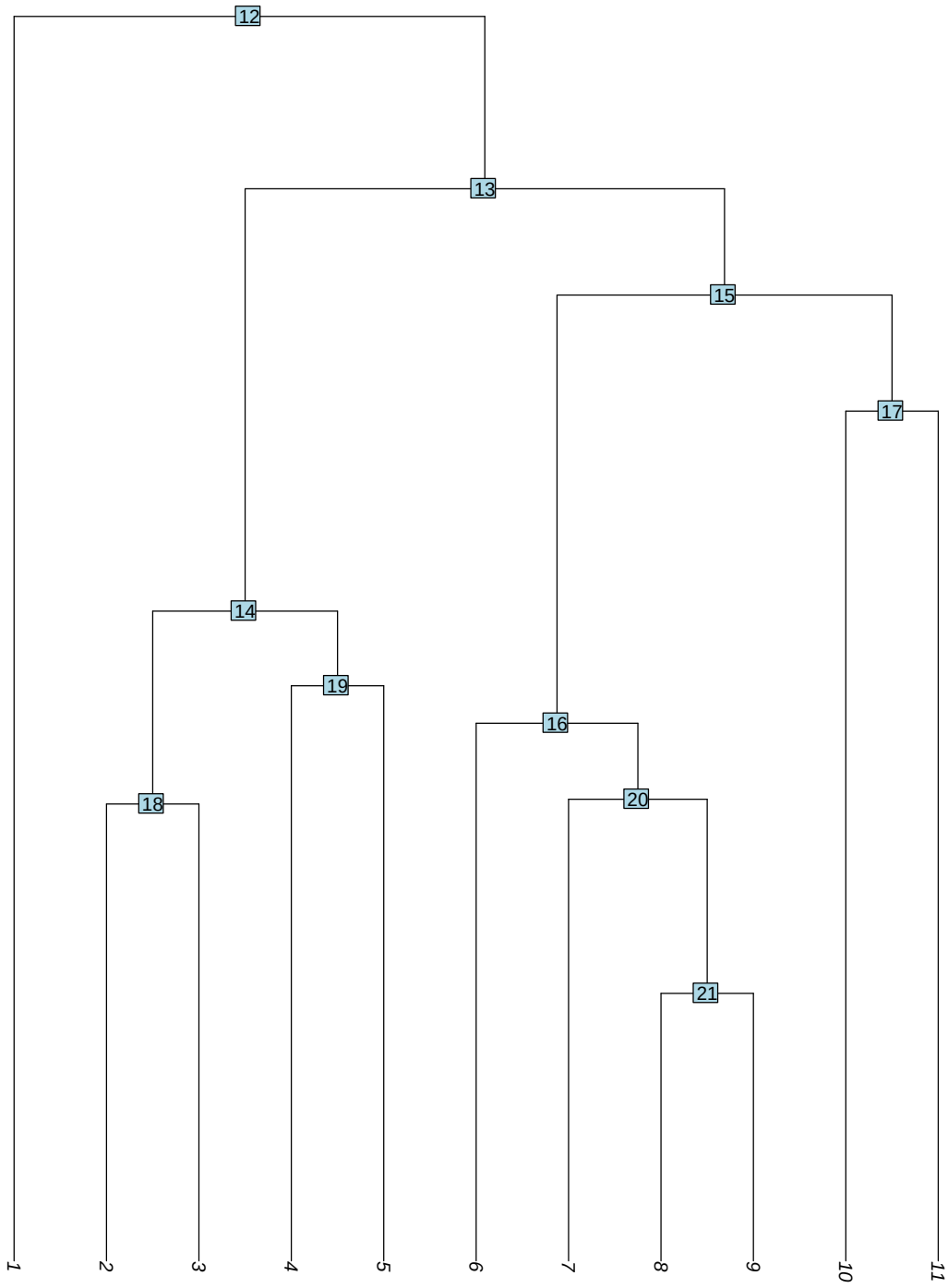
```
TMVln1 <- VlnPlot(TM.reclustered, pt.size = 0, features = c("Cdh2",
  "Atp1b1", "Slc4a11", "Car3", "Myoc", "Chil1", "Chad", "Nell2",
  "Tnc", "Rgs5", "Clec3b", "Tnxb", "Hmcn2", "Cd34", "Fmo2",
  "C2", "Cfb", "Cxcl5", "Svep1", "Mki67"), assay = "RNA")
TMVln1 + NoLegend()
```


Cdh2
Atp1b1
Slc4a11
Car3
Myoc
Chil1
Chad
Nell2
Tnc
Rgs5
Clec3b
Tnxb
Hmcn2
Cd34
Fmo2
C2
Cfb
Cxcl5
Svep1
Mki67
Expression Level
Identity
TM_10
TM_11
TM_1
TM_2
TM_3
TM_4
TM_5
TM_6
TM_7
TM_8
TM_9

```
saveRDS(TM.reclustered, file = "./data/6wk_TM_subset_10-7-20.rds.gz",
  compress = T)

plot_grid(TMplot1, TMVln1, nrow = 2, align = "hv")
```


WT
KO
UMAP_2
UMAP_1
TM_1
TM_10
TM_11
TM_2
TM_3
TM_4
TM_5
TM_6
TM_7
TM_8
TM_9
Expression Level
Identity
Cdh2
Atp1b1
Slc4a11
Car3
Myoc
Chil1
Chad
Nell2
Tnc
Rgs5
Clec3b
Tnxb
Hmcn2
Cd34
Fmo2
C2
Cfb
Cxcl5
Svep1
Mki67

```
PlotClusterTree(TM.reclustered)
```


12
13
15
17
14
19
16
18
20
21
1
2
3
4
5
6
7
8
9
10
11

```r
sessionInfo()
```

```
## R version 3.6.1 (2019-07-05)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 18363)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] intrinsicDimension_1.2.0 yaImpute_1.0-32          kableExtra_1.3.1
##  [4] future_1.14.0            scDblFinder_1.1.8        data.table_1.12.4
##  [7] clustree_0.4.1           ggraph_2.0.0             gridExtra_2.3
## [10] ggplot2_3.2.1            cowplot_1.0.0            sctransform_0.2.0
## [13] gdata_2.18.0             dplyr_0.8.3              Seurat_3.1.1
##
## loaded via a namespace (and not attached):
##   [1] sn_1.5-5                    plyr_1.8.4
##   [3] igraph_1.2.4.1              lazyeval_0.2.2
##   [5] splines_3.6.1               BiocParallel_1.20.1
##   [7] listenv_0.8.0               GenomeInfoDb_1.22.0
##   [9] scater_1.14.0               TH.data_1.0-10
##  [11] digest_0.6.21               htmltools_0.3.6
##  [13] viridis_0.5.1               magrittr_1.5
##  [15] cluster_2.1.0               ROCR_1.0-7
##  [17] limma_3.42.2                globals_0.12.5
##  [19] graphlayouts_0.5.0          RcppParallel_4.4.4
##  [21] matrixStats_0.55.0          R.utils_2.9.2
##  [23] sandwich_2.5-1              colorspace_1.4-1
##  [25] rvest_0.3.5                 ggrepel_0.8.1
##  [27] xfun_0.19                   crayon_1.3.4
##  [29] RCurl_1.98-1.1              jsonlite_1.6
##  [31] survival_3.1-8              zoo_1.8-6
##  [33] ape_5.3                     glue_1.3.1
##  [35] polyclip_1.10-0             gtable_0.3.0
##  [37] zlibbioc_1.32.0             XVector_0.26.0
##  [39] webshot_0.5.2               leiden_0.3.3
##  [41] DelayedArray_0.12.2         BiocSingular_1.2.2
##  [43] future.apply_1.4.0          SingleCellExperiment_1.8.0
##  [45] BiocGenerics_0.32.0         scales_1.0.0
##  [47] mvtnorm_1.0-11              edgeR_3.28.1
##  [49] bibtex_0.4.2                Rcpp_1.0.2
##  [51] metap_1.3                   plotrix_3.7-7
##  [53] viridisLite_0.3.0           dqrng_0.2.1
```

```
##  [55] reticulate_1.13             rsvd_1.0.2
##  [57] SDMTools_1.1-221.1          stats4_3.6.1
##  [59] tsne_0.1-3                  htmlwidgets_1.5.1
##  [61] httr_1.4.1                  gplots_3.0.1.2
##  [63] RColorBrewer_1.1-2          TFisher_0.2.0
##  [65] ica_1.0-2                   pkgconfig_2.0.3
##  [67] R.methodsS3_1.7.1           farver_2.0.1
##  [69] uwot_0.1.4                  locfit_1.5-9.1
##  [71] labeling_0.3                tidyselect_0.2.5
##  [73] rlang_0.4.4                 reshape2_1.4.3
##  [75] munsell_0.5.0               tools_3.6.1
##  [77] ggridges_0.5.2              evaluate_0.14
##  [79] stringr_1.4.0               yaml_2.2.1
##  [81] npsurv_0.4-0                knitr_1.28
##  [83] fitdistrplus_1.0-14         tidygraph_1.1.2
##  [85] caTools_1.17.1.2            purrr_0.3.3
##  [87] randomForest_4.6-14         RANN_2.6.1
##  [89] pbapply_1.4-2               nlme_3.1-140
##  [91] formatR_1.7                 scran_1.14.6
##  [93] R.oo_1.23.0                 xml2_1.2.2
##  [95] rstudioapi_0.11             compiler_3.6.1
##  [97] beeswarm_0.2.3              plotly_4.9.2
##  [99] png_0.1-7                   lsei_1.2-0
## [101] statmod_1.4.34              tibble_2.1.3
## [103] tweenr_1.0.1                stringi_1.4.3
## [105] RSpectra_0.15-0             lattice_0.20-38
## [107] Matrix_1.2-17               multtest_2.42.0
## [109] vctrs_0.2.2                 mutoss_0.1-12
## [111] pillar_1.4.3                lifecycle_0.1.0
## [113] Rdpack_0.11-1               lmtest_0.9-37
## [115] RcppAnnoy_0.0.13            BiocNeighbors_1.4.1
## [117] bitops_1.0-6                irlba_2.3.3
## [119] gbRd_0.4-11                 GenomicRanges_1.38.0
## [121] R6_2.4.1                    KernSmooth_2.23-15
## [123] vipor_0.4.5                 IRanges_2.20.2
## [125] codetools_0.2-16            MASS_7.3-51.4
## [127] gtools_3.8.1                assertthat_0.2.1
## [129] SummarizedExperiment_1.16.1 withr_2.1.2
## [131] mnormt_1.5-6                multcomp_1.4-12
## [133] S4Vectors_0.24.3            GenomeInfoDbData_1.2.2
## [135] parallel_3.6.1              grid_3.6.1
## [137] tidyr_1.0.0                 DelayedMatrixStats_1.8.0
## [139] rmarkdown_2.1               Rtsne_0.15
## [141] ggforce_0.3.1               numDeriv_2016.8-1.1
## [143] Biobase_2.46.0              ggbeeswarm_0.6.0
```